Open Digital Rights Language (ODRL)

Version: 0.8
Date: 2000-11-21
This Version: <http://odrl.net/ODRL-08.pdf>
Previous Versions: <http://odrl.net/ODRL-07.pdf>
Editor: Renato Iannella <renato@iprsystems.com>

0 Status

This document is an early draft and a work-in-progress and may be updated and/or replaced by other documents at any time.

The intention is to promote this draft document amongst multiple communities interested in the expression of Digital Rights Management statements and semantic interoperability across these communities.

ODRL will be standardised via an appropriate, open, and non-competitive organisation with an open process for the future maintenance of the standard. ***ODRL*** has no license requirements and is available in the spirit of "open source" software.

Comments are welcome to the editors from all interested parties.

Change Bars indicate modifications from Version 0.7

1 Overview

Digital Rights Management (DRM) involves the description, layering, analysis, valuation, trading and monitoring of the rights over an enterprise's assets; both in physical and digital form; and of tangible and intangible value. DRM covers the digital management of rights - be they rights in a physical manifestation of a work (eg a book), or be they rights in a digital manifestation of a work (eg an ebook). Current methods of managing, trading and protecting such assets are inefficient, proprietary, or else often require the information to be wrapped or embedded in a physical format [HIGGS].

A key feature of managing online rights will be the substantial increase in re-use of digital material on the Web as well as the increased efficiency for physical material. The pervasive Internet is changing the nature of distribution of digital media from a passive one way flow (from Publisher to the End User) to a much more interactive cycle where creations are re-used, combined and extended ad infinitum. At all stages, the Rights need to be managed and honoured with trusted services.

Current Rights management technologies include languages for describing the terms and conditions, tracking asset usages by enforcing controlled environments or encoded asset manifestations, and closed architectures for the overall management of rights.

The Open Digital Rights Language (***ODRL***) provides the semantics for DRM in open and trusted environments whilst being agnostic to mechanisms to achieve the secure architectures.

1.1 The Bigger Picture

It is envisaged that ***ODRL*** will "plug into" an open framework that enables peer-to-peer interoperability for DRM services. (See [ERICKSON] for an overview of this area). However, ***ODRL*** can also be used as an mechanism to express rights statements on its own and to plug into existing DRM architectures, for example, the Electronic Book Exchange [EBX] framework.

The editors consider that traditional DRM (even though it is still a new discipline) has taken a closed approach to solving problems. That is, the DRM has focused on the *content protection* issues more than the *rights management* issues. Hence, we see a movement towards "Open Digital Rights Management" (ODRM) with clear principles focused on interoperability across multiple sectors and support for fair-use doctrines.

The ODRM Framework consists of Technical, Business, Social, and Legal streams as shown in Figure 1.

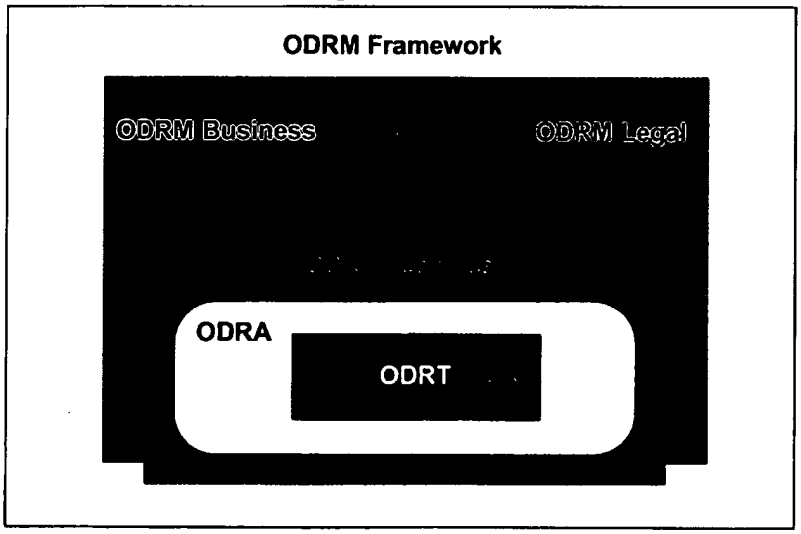


Figure 1. ODRM Framework

The ODRM Technical stream consists of an Architecture (ODRA), Trading Protocol (ODRT) and Protection (ODRP) mechanisms with ***ODRL*** clearly focused on solving a common and extendable way of expressing Rights assertions within this Architecture.

The ODRM Architecture exists in other forms that are specific to other communities needs, such as Privacy metadata. Hence, ODRA can be

achieved by abstracting and reusing such architectures to enable trusted metadata expressions about digital assets.

1.2 About this Specification

This document, along with its normative references, includes all the specification necessary for the implementation of interoperable ***ODRL*** applications.

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this specification are to be interpreted as described in [RFC2119] which defines the significance of each particular requirement.

Examples used in this document are for demonstration purposes only.

2 ODRL

ODRL complements existing analogue rights management standards by providing digital equivalents, and supports an expandible range of new services that can be afforded by the digital nature of the assets in the Web environment. In the physical environment, ***ODRL*** can be used to enable machine-based processing for Rights management.

ODRL is a standard vocabulary for the expression of terms and conditions over assets. ***ODRL*** covers a core set of semantics for these purposes including the rights holders and the expression of permissible usages for asset manifestations. Rights can be specified for a specific asset manifestation (format) or could be applied to a range of manifestations of the asset.

2.1 Scope

ODRL is focused on the semantics of expressing rights languages. ***ODRL*** can be used within trusted or untrusted systems for both digital and physical assets. However, ***ODRL*** does not determine the capabilities nor requirements of any trusted services (eg for content protection, digital/physical delivery, and payment negotiation) that utilises its language. Clearly, however, ***ODRL*** will benefit rights transactions over digital assets as these can be captured and managed as a single transaction. In the physical world, ***ODRL*** expressions would need an accompanying system with the distribution of the physical asset.

ODRL defines a core set of semantics. Additional semantics can be layered on top of ***ODRL*** for third-party value added services.

ODRL does not enforce or mandate any policies for DRM, but provides the mechanisms to express such policies. Communities or organisations, that establish such policies based on ***ODRL***, do so based on their specific business or public access requirements.

ODRL depends on the use of unique identification of assets. This is a very difficult problem to address and to have agreement across many sectors and is why identification mechanisms and policies of the assets

is outside the scope of ***ODRL***. Sector-specific versions of ***ODRL*** may address the need to infer information about the asset manifestation from its unique identifier.

ODRL model is based on an analysis and survey of sector specific requirements (models and semantics), and as such, aims to be compatible with a broad community base. ***ODRL*** aims to meet the common requirements for many sectors and has been influenced by the ongoing work and specifications/models of the following groups:

- <indecs> [INDECS]
- Electronic Book Exchange [EBX]
- IFLA
- DOI Foundation [DOI]
- ONIX International [ONIX]
- MPEG
- IMS
- Dublin Core Metadata Initiative [DCMI]
- Propagate Project [PROPAGATE]

ODRL proposes to be compatible with the above groups by defining an independent and extensible set of semantics. ***ODRL*** does not depend on any media types as it is aimed for cross-sector interoperability.

2.2 Foundation Model

ODRL is based on a simple, yet extensible, model for rights management which involves the clear separation of Parties, Assets, and Rights descriptions. This is shown in Figure 2.

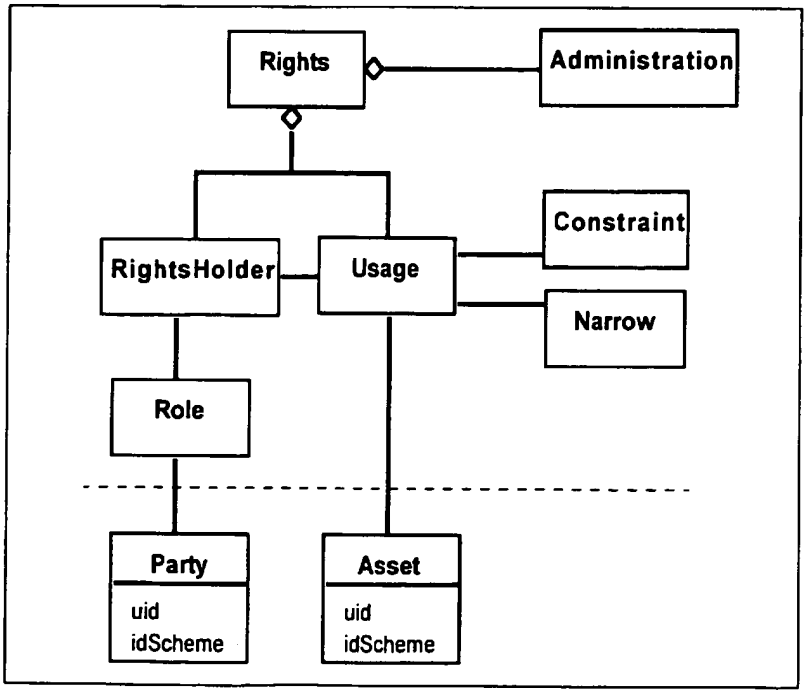


Figure 2. ODRL Foundation Model

The Rights entity consists of Usage, Constraint, Narrow, and RightsHolder which together enable the expression of digital rights over the identified Asset and their Rights Holders (parties). The Parties' Role with respect to their entitlements can also be expressed.

The description of the Party and Asset entities is outside the scope of ***ODRL***. What is in scope is that these entities must be referenced by using unique identification mechanisms (such as [URI], [DOI], [ISBN] etc).

The Asset entity (sometimes referred to as a Work, Content, Creation, or Intellectual Property), is viewed as a whole entity. If the Rights are assigned at the Asset's subpart level, then such parts would require to also be uniquely identifiable. However, ***ODRL*** can specify constraints on subparts of the asset.

The Rights entity also consists of an Administration entity that captures the responsible parties and valid dates of the Rights expression.

Complete and formal semantics for the ***ODRL*** Foundation Model properties and attributes are specified in Section 3.1 "Foundation Semantics" on page 12.

2.2.1 Example

The ***ODRL*** Foundation Model can be expressed using XML. A pseudo-example is shown below:

```
<rights>
   <asset>
      <uid idscheme="URI">http://byeme.com/myasset.pdf</uid>
   </asset>
   <usage>
      <usage-type>
         ...
         <constraint> ... </constraint>
      </usage-type>
      <usage-type>
         ...
         <constraint> ... </constraint>
      </usage-type>
      ...
   </usage>
   <narrow> ... </narrow>
   <rightsholder>
      <party>
         ...
         <role> ... </role>
      </party>
      ...
   </rightsholder>
   <admin>
      <party> ... </party>
      <datetime> .. </datetime>
   </admin>
</rights>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.3 Rights Usage Model

ODRL supports the expression of Rights Usages. This is the recognised set of allowable usage rights over the Asset. This is shown in Figure 3.

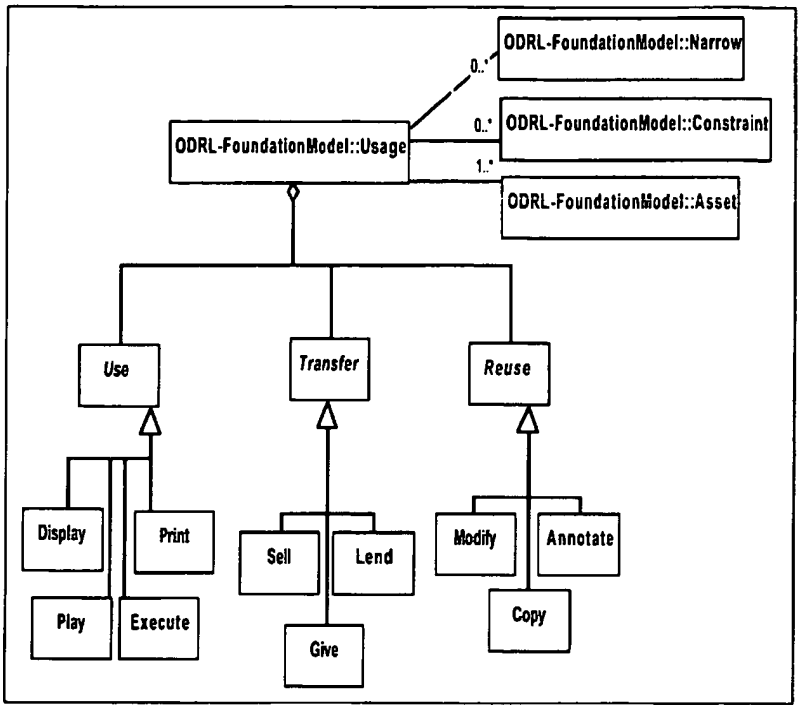


Figure 3. ODRL Usages Model

The Usage entity consists of an aggregation of three abstract entities:

- Use - indicates a set of usages in which the Asset can be consumed (realised with: Display, Print, Play, Execute).
- Transfer - indicates a set of usages in which the rights over the Asset can be transferred (realised with: Sell, Lend, Give).
- Reuse - indicates a set of usages in which the Asset (or portions of it) can be re-utilised (realised with: Modify, Copy, Annotate).

A Usage must be associated with one or more Assets. A Usage can be associated with zero or more Constraints. For any rights expression, all Usages are "and-ed" together including their constraints.

Important Note

A Usage Right that is not specified in any Rights Expressions is not granted. That is, no assumptions should be made in regard to Usage Rights if they are not explicitly mentioned.

Additionally, all Usages can be subject to an "Exclusivity" attribute that indicates if the constraint is exclusive or not.

Complete and formal semantics for the ***ODRL*** Usage Model properties and attributes are specified in Section 3.2 "Usage Semantics" on page 13.

2.3.1 Example

The ***ODRL*** Usage Model can be expressed using XML. A pseudo-example is shown below in which the identified asset has display, print (with constraints), and annotate rights.

```
<usage>
   <asset>
      <uid idscheme="URI">http://byeme.com/myasset.pdf</uid>
   </asset>
   <display/>
   <print>
      <constraint> ... </constraint>
   </print>
   <annotate/>
   ...
</usage>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.4 Rights Constraint Model

ODRL supports the expression of Rights Constraints. This is the recognised set of restrictions on the usage rights over the Asset. This is shown in Figure 4.

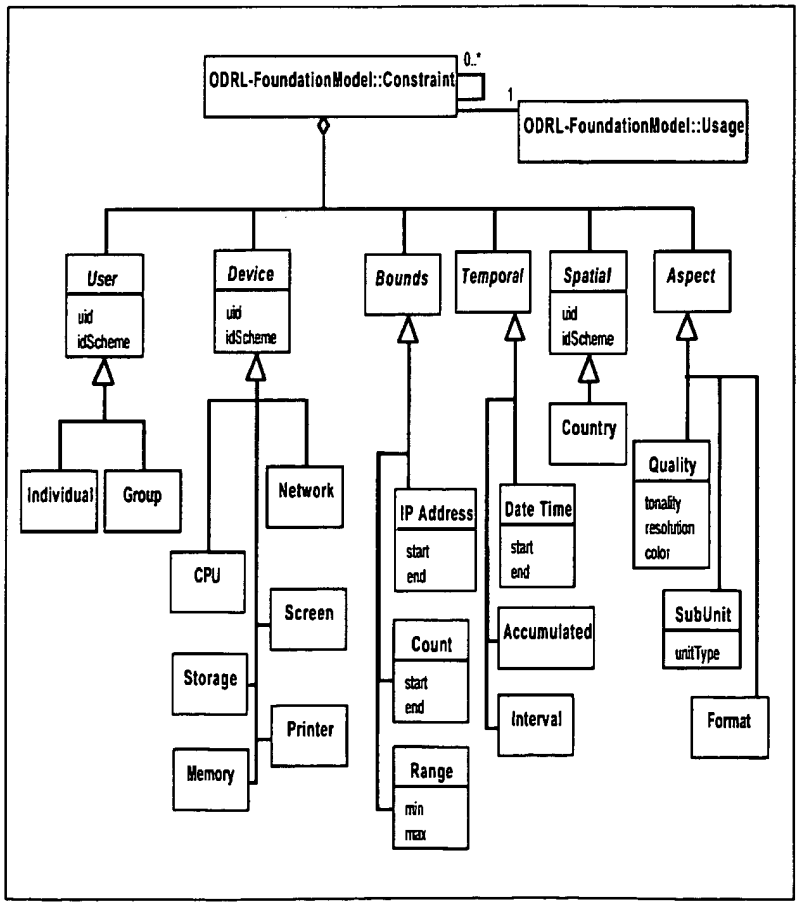


Figure 4. ODRL Constraints Model

The Constraint entity consists of an aggregation of six abstract entities:

- User - indicates a set of constraints which limits usage to identified user(s) (realised with: Individual, Group).
- Device - indicates a set of constraints which limits usage to physical devices (realised with: Network, CPU, Screen, Storage, Printer, Memory).
- Bounds - indicates a set of constraints which limits usage to a fixed number or extent (realised with: Count, Range, IP Address).
- Temporal - indicates a set of constraints which limits usage to temporal boundaries (realised with: Date Time, Accumulated, Interval).
- Spatial - indicates a set of constraints which limits usage to spatial boundaries (realised with: Country).
- Aspect - indicates a set of constraints which limits usage to distinct features of the asset (realised with: Quality, SubUnit, Format).

Additionally, all Constraints can be subject to an "Exclusivity" attribute that indicates if the constraint is exclusive or not.

A Constraint is associated with one Usage. If a Constraint appears at the same level as a number of Usages, then the Constraint applies to all of the Usages. Constraints can also have zero or more other Constraints. For Usages with multiple constraints, all contraints must be "and-ed" together and no conflicts should arise. An error must be generated if the latter is true.

Important Note

Any Constraint that is expressed but can not be performed by the consuming system, must not be granted. That is, if a system does not understand how to guarantee that a specified constraint be honoured it must not grant the Usage right at all.

Complete and formal semantics for the ***ODRL*** Constraint Model properties and attributes are specified in Section 3.3 "Constraint Semantics" on page 16.

2.4.1 Example

The ***ODRL*** Constraint Model can be expressed using XML. A pseudo-example is shown below in which the display usage right is constrained to a particular network within an identified IP address range.

```
<display>
  <constraint>
    <network>
      <constraint>
        <ipaddress start="111.222.333.1" end ="111.222.333.255" />
      <constraint>
    </network>
  </constraint>
</display>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.5 Rights Narrow Model

ODRL supports the expression of Narrowing of Rights. This is the ability to specify if the current Rights can be modified (narrowed or removed) when re-issuing the Rights expression. This is shown in Figure 5.

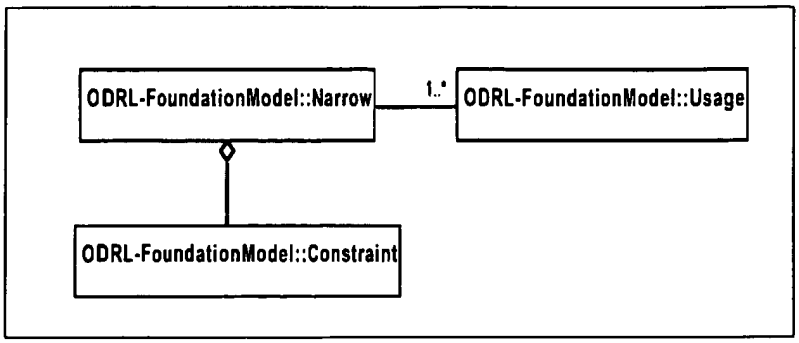


Figure 5. ODRL Narrow Model

The Narrow entity is an aggregation of one other existing entity:

- Constraint - indicates any constraints that the Narrow rights must conform to.

Complete and formal semantics for the ***ODRL*** Narrow Model properties and attributes are specified in Section 3.4 "Narrow Semantics" on page 20.

2.5.1 Example

The ***ODRL*** Narrow Model can be expressed using XML. A pseudo-example is shown below in which sell and lend transfer rights exist for the identified asset and narrow rights are applicable and are also constrained to a particular country.

```
<rights>
   <asset>
      <uid idscheme="URI">http://byeme.com/myasset.pdf</uid>
   </asset>
   <usage>
      <sell/>
      <lend/>
      <narrow>
         <constraint>
            <country>
               <uid idscheme="ISO3166"> AU </uid>
            </country>
         </constraint>
      </narrow>
   </usage>
</rights>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.6 RightsHolder Model

ODRL supports the identification of Rights Holders. This is the recognised Party, their (optional) Role, and any set of rewarding

mechanisms for the usage of the Asset for the Rights Holder. This is shown in Figure 6.

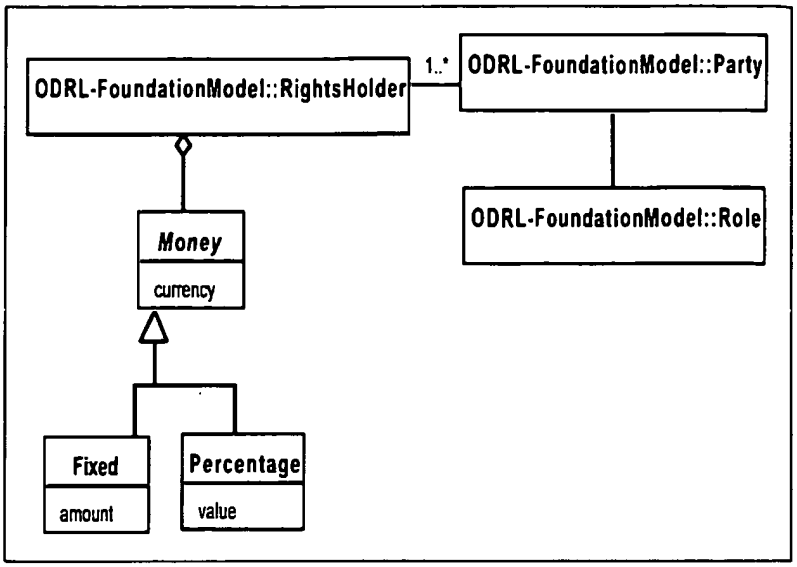


Figure 6. ODRL Rights Holder Model

The RightsHolder entity is an aggregation of one abstract and one existing entity:

- Money - indicates a set of financial rewards associated with the usage of an Asset (realised with: Fixed, Percentage).
- Party - indicates the Rights Holder and the role they play.

One or more Parties must be identified with the RightsHolder expression. The Role of the Party may also be indicated.

Complete and formal semantics for the ***ODRL*** RightsHolder Model properties and attributes are specified in Section 3.5 "RightsHolder Semantics" on page 21.

2.6.1 Example

The ***ODRL*** RightsHolder Model can be expressed using XML. A pseudo-example is shown below in which two identified Rights Holders (parties) share the financial rewards with 90% to the Author and 10% to the Publisher.

```
<rightsholder>
   <party>
      <uid idscheme="X500">c=FOO;o=Federal Library;ou=Registry;
                                    cn=Maria Brown</uid>
      <role>Author</role>
      <percentage value="90" currency="AUD"/>
   </party>
   <party>
      <uid idscheme="X500"> c=FOO;o=Federal Library;ou=Registry;
                                    cn=Bye Me Inc</uid>
      <role>Publisher</role>
      <percentage value="10" currency="AUD"/>
   </party>
</rightsholder>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.7 Rights Administration Model

ODRL supports the Administrative information about the Rights expression. This is shown in Figure 7.

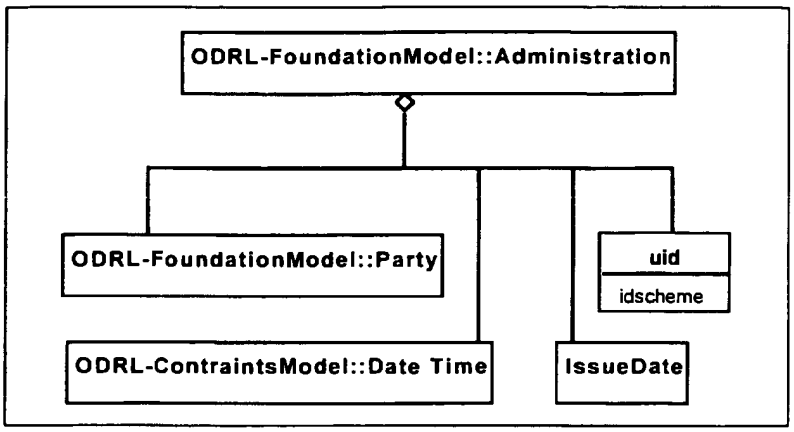


Figure 7. ODRL Administration Model

The Administration entity is an aggregation of three other existing entities and one new entity:

- Party - indicates who is responsible for maintenance of this Rights expression and the (optional) role they play.
- Date Time - indicates the valid date range for the Rights expression.
- Issue Date - indicates the date/time that the Rights expression was issued.
- UID - a unique identification number for the Rights expression

Complete and formal semantics for the ***ODRL*** Administration Model properties and attributes are specified in Section 3.6 "Administration Semantics" on page 21.

2.7.1 Example

The ***ODRL*** Administration Model can be expressed using XML. A pseudo-example is shown below in which the Rights expression is managed by the identified party (the Rights Cataloguer) and is valid for a two year period.

```
<rights>
   <admin>
      <party>
         <uid idscheme="X500"> c=FOO;o=Federal Library;ou=Registry;
                                    cn=Maria Brown</uid>
         <role>Rights Cataloguer</role>
      </party>
      <issuedate> 2000-12-31 </issuedate>
      <datetime start="2001-01-01" end="2001-12-31"/>
      <uid idscheme="URI"> http://byeme.com/mybook-rights.xml</uid>
   </admin>
   ...
</rights>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

3 Semantics

This section details the semantics of all the properties and attributes used in the ***ODRL*** Models.

3.1 Foundation Semantics

Rights

Identifier	rights
Definition	The digital expression of intellectual property rights over an asset
Cardinality	mandatory
Content (entities)	usage rightsholder administration asset narrow

Usage Rights

Identifier	usage
Definition	A defined set of actions or operations allowed over an asset
Cardinality	mandatory
Content (entities)	use transfer reuse

Rights Holder

Identifier	rightsholder
Definition	Any party that holds any form of Rights over the asset
Cardinality	optional
Content (entities)	money party role

Asset

Identifier	asset
Definition	Any object (digital or physical) of value which rights can be assigned
Comment	Must be uniquely identifiable
Cardinality	mandatory
Content (entity)	uid - unique identifier

Party

Identifier	party
Definition	An identifiable person or organisation to which rights may be assigned over assets
Comment	Must be uniquely identifiable
Cardinality	optional
Content (entities)	uid - unique identifier role - role played by the party

UID

Identifier	uid
Definition	The unique identification number/code for the entity
Comment	The uid may be applied to assets, parties, constraints and admin entities.
Cardinality	optional
Content (attribute)	idScheme - encoding scheme used for the unique identifier value

Role

Identifier	role
Definition	The role played by the Party
Comment	The role values may be selected from existing vocabulary schemes. For example: • marc - MARC Code List for Relators [MARC] • onix - ONIX International Contributor Role Code List [ONIX]
Cardinality	optional
Content (attribute)	idScheme - identifies the vocabulary scheme used for the role value

3.2 Usage Semantics

Use

Identifier	use
Definition	A set of Usage rights pertaining to the end use of an asset
Comment	This entity is abstract and used to group common Rights Usages.
Cardinality	optional
Content (entities)	display print play execute

Use: Display

Identifier	display
Definition	The act of rendering the asset onto a screen or visual device
Cardinality	optional
Content (entities)	constraint

Use: Print

Identifier	print
Definition	The act of rendering the asset onto paper or hard copy form
Cardinality	optional
Content (entities)	constraint

Use: Play

Identifier	play
Definition	The act of rendering the asset into audio/video form
Cardinality	optional
Content (entities)	constraint

Use: Execute

Identifier	execute
Definition	The act of rendering the asset into machine-readable form
Cardinality	optional
Content (entities)	constraint

Transfer

Identifier	transfer
Definition	A set of Usage rights pertaining to the transfer of ownership of an asset
Comment	This entity is abstract and used to group common Rights Usages
Cardinality	optional
Content (entities)	sell lend give

Transfer: Sell

Identifier	sell
Definition	The act of allowing the asset to be sold for exchange of value
Cardinality	optional
Content (entities)	constraint

Transfer: Lend

Identifier	lend
Definition	The act of allowing the asset to be available for temporary use then returned
Comment	Time-based constraints are required
Cardinality	optional
Content (entities)	constraint (mandatory)

Transfer: Give

Identifier	give
Definition	The act of allowing the asset to be given away (without exchange of value)
Cardinality	optional
Content (entities)	constraint

Reuse

Identifier	reuse
Definition	A set of Usage rights pertaining to the re-utilisation of an asset
Comment	This entity is abstract and used to group common Rights Usages.
Cardinality	optional
Content (entities)	modify copy annotate

Reuse: Modify

Identifier	modify
Definition	The act of changing parts of the asset creating a new asset
Cardinality	optional
Content (entities)	constraint

Reuse: Copy

Identifier	copy
Definition	The act of extracting parts (or all) of the asset for reuse into another asset
Cardinality	optional
Content (entities)	constraint

Reuse: Annotate

Identifier	annotate
Definition	The act of adding notations/commentaries to the asset creating a new asset
Cardinality	optional
Content (entities)	constraint

3.3 Constraint Semantics

Constraint

Identifier	constraint
Definition	A restriction that applies to the Usage of an asset
Cardinality	optional
Content (entities)	user device bounds temporal spatial aspect

User

Identifier	user
Definition	Any human or organisation
Comment	This entity is abstract and used to group common Constraints
Cardinality	optional
Content (entities)	individual group

User: Individual

Identifier	individual
Definition	An identifiable party acting as an individual
Cardinality	optional
Content (entity)	uid - unique identifier

User: Group

Identifier	group
Definition	A number of identifiable party acting as a collection of individuals
Cardinality	optional
Content (entity)	uid - unique identifier

Device

Identifier	device
Definition	Any electronic or digital equipment
Comment	This entity is abstract and used to group common Constraints
Cardinality	optional
Content (entities)	network cpu screen storage printer memory

Device: Network

Identifier	network
Definition	An identifiable data network
Comment	If below attributes are not sufficient, then IP Address Range can also be used to limit the network.
Cardinality	optional
Content (entity)	uid - unique identifier

Device: CPU

Identifier	cpu
Definition	An identifiable system with a central processing unit (CPU)
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Screen

Identifier	screen
Definition	An identifiable display output screen device
Comment	For example, a screen reader or braille device
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Storage

Identifier	storage
Definition	An identifiable storage media device
Comment	For example, a hard disk or removable cartridge
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Printer

Identifier	printer
Definition	An identifiable hard copy printer
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Memory

Identifier	memory
Definition	An identifiable memory device
Comment	For example, the clipboard
Cardinality	optional
Content (entity)	uid - unique identifier

Bounds

Identifier	bounds
Definition	The numeric limits within which any entity can function
Comment	This entity is abstract and used to group common Constraints.
Cardinality	optional
Content (entities)	Count Range IP Address

Bounds: Count

Identifier	count
Definition	A numeric count indicating the number of times the corresponding entity may be exercised
Comment	For example, the Print usage may be constraint with a count of 1 to 10 meaning that the asset can be printed once or up to 10 times. If there is no "start" or "end" value, then the count is open-ended. Integer, Floats must be supported. Note, "start" must always be less than or equal to "end" and one must always be present.
Cardinality	optional
Content (attributes)	start - the beginning of the count (inclusive) end - the end of the count (inclusive)

Bounds: Range

Identifier	range
Definition	A numeric range indicating the min/max values of the corresponding entity that the constraint applies to
Comment	For example, this is used to specify that only pages numbered 1 to 10 may be printed (using the subunit entity). If there is no "min" or "max" value, then the range is open-ended. Integer, Floats and negative numbers must be supported. Note, "min" must always be less than or equal to "max" and one must always be present.
Cardinality	optional
Content (attributes)	min - the beginning of the range (inclusive) max - the end of the range (inclusive)

Bounds: IP Address

Identifier	ipaddress
Definition	A network IP address range
Comment	There must be "start" and "end" values specified. The IP address format must be supported (Eg xxx.xxx.xxx.xxx).
Cardinality	optional
Content (attributes)	start - the beginning of the range (inclusive) end - the end of the range (inclusive)

Temporal

Identifier	temporal
Definition	The time limits within which any entity can function
Comment	This entity is abstract and used to group common Constraints. [ISO8601] Date format must be supported for all values.
Cardinality	optional
Content (entities)	Date Time Accumulated Interval

Temporal: Date Time

Identifier	datetime
Definition	A date/time-based range
Comment	If there is no "start" and/or "end" value, then the range is open-ended.
Cardinality	optional
Content (attributes)	start - the beginning of the range (inclusive) end - the end of the range (inclusive)

Temporal: Accumulated

Identifier	accumulated
Definition	The maximum amount of metered usage time
Cardinality	optional
Content	data value

Temporal: Interval

Identifier	interval
Definition	Recurring period of time in which rights can be exercised
Cardinality	optional
Content	data value

Spatial

Identifier	spatial
Definition	Any geographical range or extent
Comment	This entity is abstract and used to group common Constraints.
Cardinality	optional
Content (entities)	Country

Spatial: Country

Identifier	country
Definition	Specification of a Country code
Comment	Recommended best practice is to use the codes specified by the [ISO3166] Scheme.
Cardinality	optional
Content (entity)	uid - unique identifier

Aspect

Identifier	aspect
Definition	Any distinct feature of the Asset
Comment	This entity is abstract and used to group common Constraints
Cardinality	optional
Content (entities)	Quality SubUnit Format

Aspect: Quality

Identifier	quality
Definition	Specification of quality aspects of the asset
Cardinality	optional
Content (attributes)	tonality - the bit-depth resolution - the pixel size color - the number of colors

Aspect: SubUnit

Identifier	subunit
Definition	Specification of any sub-part of the asset
Comment	The values for the unittype attribute should be from a well known vocabulary and the source clearly identified.
Cardinality	optional
Content	unittype (attribute) constraint (entity)

Aspect: Format

Identifier	format
Definition	Specification of format(s) of the asset
Comment	The values are taken from the Internet Media Type [IMT] list.
Cardinality	optional
Content	data value

3.4 Narrow Semantics

Narrow

Identifier	narrow
Definition	Specifies modification of down-stream Rights
Cardinality	optional
Content (entities)	constraint

3.5 RightsHolder Semantics

Money

Identifier	money
Definition	Rewards in the form of financial payments
Comment	This entity is abstract and used to group common Reward types for Rights Holders
Cardinality	optional
Content (entities)	Fixed Percentage

Money: Fixed

Identifier	fixed
Definition	A fixed monetary value
Comment	The total of the Fixed values for a single asset must not exceed the Retail Price.
Cardinality	optional
Content (attributes)	amount - the value of the payment (an positive integer to two decimal places) currency - the currency for the amount (use [ISO4217] codes)

Money: Percentage

Identifier	percentage
Definition	A proportion of the value of the asset
Comment	The total of the Percentage values for a single asset must not exceed 100%.
Cardinality	optional
Content (attributes)	value - a number from 0 to 100 inclusive currency - the currency for the amount (use [ISO4217] codes)

3.6 Administration Semantics

Administration

Identifier	admin
Definition	Administrative information about the Rights expression
Cardinality	optional
Content (entities)	party datetime uid issuedate

Administration:
Issue Date

Identifier	issuedate
Definition	The date the Rights expression was issued/released
Comment	[ISO8601] Date format must be supported for all values.
Cardinality	optional
Content	data value

4 Syntax

ODRL can be expressed in [XML] (see [DTD] in Appendix A and [XML SCHEMA] in Appendix B for formal definitions). However, it is also conceivable that ***ODRL*** could be expressed in other syntaxes.

ODRL is XML Namespace aware as its primary target is use with other content description and management systems. The ***ODRL*** XML Namespace URI for *this version* is:

http://odrl.net/0.8/

The final Version 1.0 ***ODRL*** XML Namespace URI will be:

http://odrl.net/1.0/

NOTE: These URIs should be considered *experimental* until the ***ODRL*** specification is formalised by an appropriate body and the new URI is assigned.

ODRL uses XML XLink [XLINK] to refer from XML fragments to other fragments. This is used to express the relationship between the core ***ODRL*** entities such as Asset, Reward, and Usage. Such elements can be identified with the standard ID attribute then referred to via XLink's href attribute. Note, only the "xlink:href" attribute is required to be recognised to support ***ODRL*** expressions.

It is important to recognise that as the ***ODRL*** expressions become more complicated, the need to partition and express linkages (using XLink) becomes paramount in order to have manageable and reusable rights expressions. The linking mechanism allows for quite complex expressions to be generated whilst preserving the interpretability of the overall rights language.

All elements can also have optional Name and Remark elements for human-readable documentation. If the human language needs to be specified for any elements, then the use of the "xml:lang" attribute is recommended.

The XML syntax will be explained via a serious of Use Cases covering different content sectors (ebooks, image, audio, video).

4.1 Ebook Use Case #1

Corky Rossi (an author) and Addison Rossi (an illustrator) publish their ebook via "EBooksRUS Publishers". They wish to allow consumers to purchase the ebook which is restricted to a single CPU only and they are allowed to print a maximum of 2 copies. They will

also allow the first 5 pages (SubUnits) of the ebook to be viewed online for free.

The revenue split is $AUD 10.00 to the Author, $AUD 2.00 to the Illustrator and $AUD 8.00 to the Publisher.

Massimo DiAngelo from "EBooksRUS Publishers" is responsible for maintaining the Rights metadata which has a policy of one year validity on all its metadata.

The XML encoding of this in ***ODRL*** would be:

```
<?xml version="1.0"?>
<rights   xmlns="http://odrl.net/0.8/"
          xmlns:xlink="http://www.w3.org/1999/xlink">

   <admin>
      <party>
         <uid idscheme="DOI">doi://10.9999/EP/mdiangelo-001</uid>
         <role>Rights Manager</role>
      </party>
      <datetime start="2000-07-01" end="2001-06-30"/>
   </admin>

   <asset ID="001">
      <uid idscheme="DOI">doi://10.9999/EB/rossi-0001</uid>
      <name> How to Wash Cats </name>
   </asset>

   <usage ID="002">
      <asset xlink:href="#001"/>
      <rightsholder xlink:href="#003"/>
      <display>
         <remark> Constrain to a particular CPU only </remark>
         <constraint>
            <cpu/>
         </constraint>
      </display>
      <print>
         <remark> Can only Print 2 Copies </remark>
         <constraint>
            <count start="0" end="2"/>
         </constraint>
      </print>
   </usage>

   <rightsholder ID="003">
      <party>
         <uid idscheme="DOI">doi://10.9999/EP/crossi-001</uid>
         <role>Author</role>
         <fixed amount="10.00" currency="AUD"/>
      </party>
      <party>
         <uid idscheme="DOI">doi://10.9999/EP/arossi-001</uid>
         <role>Illustrator</role>
```

```
            <fixed amount="2.00" currency="AUD"/>
        </party>
        <party>
            <uid idscheme="DOI">doi://10.9999/EP/ebooksrus-01</uid>
            <role>Publisher</role>
            <fixed amout="8.00" currency="AUD"/>
        </party>
    </rightsholder>

    <usage ID="004">
        <asset xlink:href="#001"/>
        <remark> Allow the first 5 pages to be viewable </remark>
        <display>
            <constraint>
                <subunit unittype="page">
                    <constraint>
                        <range start="1" end ="5"/>
                    </constraint>
                </subunit>
            </constraint>
        </display>
    </usage>

</rights>
```

4.2 Ebook Use Case #2

ByeMe.Com is a distributor of ebooks. The ***ODRL*** expression below indicates that they have Sell rights for the identified ebook assets. The next Usage right is constrained to a particular individual (Mary Smith). Mary can also only print the HTML format of the asset for one to a maximum of 100 times. Mary is also limited to a maximum accumulated time of 10 hours of Display rights every 4 days.

```
<?xml version="1.0"?>
<rights    xmlns="http://odrl.net/0.8/"
           xmlns:xlink="http://www.w3.org/1999/xlink">

    <asset ID="001">
        <uid idscheme="URI">http://byeme.com/mybook.pdf</uid>
        <uid idscheme="URI">http://byeme.com/mybook.html</uid>
    </asset>

    <rightsholder ID="002">
        <party>
            <uid idscheme="X500">c=ZZ;o=Bye Me;cn=R Owner</uid>
            <role idscheme="marc"> dst </role>
        </party>
    </rightsholder>

    <usage>
        <remark> This usage associates the Distributor with the Sell rights
                 of the assets </remark>
        <asset xlink:href="#001"/>
        <rightsholder xlink:href="#002"/>
```

```
        <sell/>
    </usage>

    <usage ID="003>
        <asset xlink:href="#001"/>
        <display>
            <constraint>
                <individual>
                    <uid idscheme="X500" >c=ZZ;o=People Directory;
                                        cn=Mary Smith</uid>
                </individual>
                <accumulated> P10H </accumulated>
                <interval> P4D </interval>
            </constraint>
        </display>
        <print>
            <constraint>
                <format>text/html</format>
                <count start="1" end="100"/>
            </constraint>
        </print>
    </usage>

</rights>
```

4.3 Ebook Use Case #3

The ebook for an "Electronic Book Exchange" voucher is entitled "XML: A Manager's Guide" by "Kevin Dick". The rights owner is Addison-Wesley.

The Distributor of this book is a company called "XYZ". They have rights to Sell up to 5000 copies of the book. The have "Narrow" rights for Sell.

The licensed end user for this book is "John Doe". He has rights to view the book for 30 days before the end of 2004. He can print up to 5 copies on a "trusted printer" before the end of the year 2004. He can print up to 5 pages between page 1 and 100 every week - up to a total of 100 pages - on a "conventional printer" - before the end of 2004. He can also extract 5000 bytes every week up to a total of 1,000,000 bytes onto the Clipboard before the end of 2004. He has a right to Give the book away after one year of the usage starting.

```
<?xml version="1.0"?>
<rights    xmlns="http://odrl.net/0.8/"
           xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:ebx="http://ebxwg.org/voucher/1.0/">

    <admin>
        <remark> Info about the Voucher </remark>
        <datetime start="2000-06-07" end="2001-06-07"/>
    </admin>

    <asset ID="Ebook-0001">
        <remark> The product ID info </remark>
```

```
    <uid idscheme="ISBN"> 0201433354 </uid>
    <name> XML: A Managers Guide </name>
</asset>

<rightsholder ID="RH-PUB-1">
    <remark> The Rights Holder (publisher) info </remark>
    <party>
        <uid idscheme="URL"> http://www.awl.com/ </uid>
        <name> Addison-Wesley </name>
        <role> Publisher </role>
    </party>
</rightsholder>

<usage ID="USE-DIST-1">
    <remark> Usage Rights for the Distributor </remark>
    <asset xlink:href="#Ebook-0001"/>

    <rightsholder>
        <remark> The Rights Holder (distributor) info </remark>
        <party>
            <uid idscheme="CG-ID"> ABDC-1234 </uid>
            <name> XYZ </name>
            <role> Distributor </role>
        </party>
    </rightsholder>

    <sell>
        <constraint> <count start="0" end ="5000"/> </constraint>
        <narrow/>
    </sell>
    <remark> Distributor also has Narrow rights over the End User
            rights </remark>
    <narrow xlink:href="EU-00001"/>
</usage>

<usage ID="EU-00001">
    <remark> Usage Rights for a typical End User</remark>
    <asset xlink:href="#Ebook-0001"/>

    <constraint>
        <remark> All usages are Licensed to Mr Doe </remark>
        <datetime start="1999-10-13"/>
        <individual>
            <uid idscheme="Lic-ID"> 92840-AA9-39849-00 </uid>
            <name> John Doe</name>
        </individual>
    </constraint>

    <display>
        <remark> View the work for 30 day period until 2004 </remark>
        <constraint>
            <accumulated> P30D </accumulated>
            <datetime end="2004-12-31"/>
```

```
    </constraint>
</display>

<print>
    <remark> Print the work up to 5 times on a trusted printer
                until 2004 </remark>
    <constraint>
        <count start="0" end ="5"/>
        <printer>
            <uid idscheme="ABC"> MyTrustedPrinterID </uid>
        </printer>
        <datetime end="2004-12-31"/>
    </constraint>
</print>

<print>
    <remark> Print up to 5 pages in any week period - between the
                pages 1 and 100 - up to a total of 100 pages - on a
                conventional printer - until 2004 </remark>
    <constraint>
        <subunit unittype="ebx:page">
            <constraint> <count end = "100"/> </constraint>
        </subunit>
        <subunit unittype="ebx:page">
            <constraint>
                <count end = "5"/>
                <range min= "1" max = "100"/>
                <interval> P7D </interval>
            </constraint>
        </subunit>
        <printer>
            <uid idscheme = "ABC"> AnyPrinterID </uid>
        </printer>
        <datetime end="2004-12-31"/>
    </constraint>
</print>

<copy>
    <remark> Extract 5000 Bytes onto the Clipboard every
        week - up to a total of 1,000,000 Bytes - until 2004 </remark>
    <constraint>
        <memory/>
        <subunit unittype="ebx:byte">
            <constraint>
                <count end ="5000"/>
                <interval> P7D </interval>
            </constraint>
        </subunit>
        <subunit unittype="ebx:byte">
            <constraint>
                <count end ="1,000,000"/>
            </constraint>
        </subunit>
```

```
            <datetime end="2004-12-31"/>
        </constraint>
    </copy>

    <remark> All the ebook to be given away (after one year) </remark>
    <give>
        <constraint>
            <datetime start="2000-10-13"/>
        </constraint>
    </give>

</usage>

</rights>
```

4.4 Image Use Case	To do...
4.5 Video Use Case	To do...
4.6 Audio Use Case	To do...

5 References

Technical Standards:

- [DCMI] Dublin Core Metadata Initiative
  <http://purl.org/DC/>
- [DOI] Digital Object Identifier
  <http://www.doi.org/>
- [DTD] Document Type Definition
- [EBX] Electronic Book Exchange
  <http://www.ebxwg.org/>
- [IFLA] Functional Requirements for Bibliographic Records
  <http://www.ifla.org/VII/s13/frbr/frbr.htm>
- [IMS] Instructional Management Systems
  <http://www.imsproject.org/>
- [IMT] Internet Media Types
  <http://www.isi.edu/in-notes/iana/assignments/media-types/media-types>
- [INDECS] Interoperability of Data in Ecommerce Systems
  <http://www.indecs.org/>
- [ISBN] International Standard Book Number
- [ISO3166] Country Names and Code Elements
  <http://www.din.de/gremien/nas/nabd/iso3166ma/codlstp1/>
- [ISO4217] Currency Names
  <http://www.xe.net/gen/iso4217.htm>

- [ISO8601] ISO (International Organization for Standardization). Representations of dates and times
  <http://www.iso.ch/markete/8601.pdf>
- [MARC] MARC Code List for Relators
  <http://www.loc.gov/marc/relators/re0002r1.html>
- [MPEG] Moving Picture Experts Group (WG 4,7,21)
  <http://www.cselt.it/leonardo/mpeg/>
- [ONIX] ONIC International V1.1
  <http://www.editeur.org/onix.html>
- [PROPAGATE] Propagate Project
  <http://www.propagate.net/>
- [RFC2119] Key words for use in RFCs to Indicate Requirement Levels
  <http://www.ietf.org/rfc/rfc2119.txt>
- [URI] Uniform Resource Identifiers (URI): Generic Syntax
  <http://www.ietf.org/rfc/rfc2396.txt>
- [VCARD] vCard MIME Directory Profile
  <http://www.ietf.org/rfc/rfc2426.txt>
- [XLINK] XML Linking Language (XLink) Version 1.0
  <http://www.w3.org/TR/xlink/>
- [XML] Extensible Markup Language 1.0
  <http://www.w3.org/TR/REC-xml>
- [XML NAMESPACE] Namespaces in XML
  <http://www.w3.org/TR/REC-xml-names/>
- [XML SCHEMA] XMl Schemas Part 1: Structures
  <http://www.w3.org/TR/xmlschema-1/>

Position Papers:

- [ERICKSON] Toward an Open Rights Management Interoperability Framework, John S Erickson.
  <http://www.oasis-open.org/cover/ericksonRT19990624.pdf>
- [HIGGS] The Nature of Knowledge and Rights Management Systems, Peter Higgs.
  <http://www.iprsystems.com/html/rights_management.html>

6 Acknowledgements

The editors wish to acknowledge contributions and feedback to this document from:

- Peter Higgs, IPR Systems
- David Parrott, Reuters
- Robert Bolick, McGraw-Hill

Appendix: A ODRL DTD (Normative)

```
<!ELEMENT rights (admin? | asset+ | usage+ | rightsholder* | name? | remark? |
narrow* )>
<!ATTLIST   rights      xmlns:xlink   CDATA   #REQUIRED
                        xmlns         CDATA   #REQUIRED >
```

```
<!ELEMENT name (#PCDATA )>
<!ELEMENT remark (#PCDATA )>

<!ELEMENT admin (name? | remark? | party* | datetime? | issuedate? | uid?)>

<!ELEMENT party (name? | remark? | uid+ | role? | percentage? | fixed? )>

<!ELEMENT uid (#PCDATA )>
<!ATTLIST uid idscheme CDATA #REQUIRED >

<!ELEMENT role (#PCDATA )>
<!ATTLIST role idscheme CDATA #IMPLIED >

<!ELEMENT issuedate (#PCDATA )>

<!ELEMENT asset (uid+ | name? | remark? )>
<!ATTLIST asset     xlink:href CDATA    #IMPLIED
                    ID         CDATA    #IMPLIED >

<!ELEMENT usage (asset* | display* | rightsholder* | print* | play* | execute* | sell*
| lend* | give* | modify* | annotate* | copy* | constraint* | name? | remark? )>
<!ATTLIST usage     xlink:href CDATA    #IMPLIED
                    ID         CDATA    #IMPLIED >


<!ELEMENT print (name? | remark? | constraint* )>

<!ELEMENT display (name? | remark? | constraint* )>

<!ELEMENT play (name? | remark? | constraint* )>

<!ELEMENT execute (name? | remark? | constraint* )>

<!ELEMENT sell (name? | remark? | constraint* )>

<!ELEMENT lend (name? | remark? | constraint* )>

<!ELEMENT give (name? | remark? | constraint* )>

<!ELEMENT modify (name? | remark? | constraint* )>

<!ELEMENT annotate (name? | remark? | constraint* )>

<!ELEMENT copy (name? | remark? | constraint* )>

<!ELEMENT constraint (accumulated* | interval* | datetime* | country* | quality* |
count* | range* | ipaddress* | subunit* | individual* | group* | format* | cpu* |
network* | screen* | storage* | memory* | printer* | name? | remark? )>

<!ELEMENT individual (uid+ | name? | remark? | constraint* )>

<!ELEMENT group (uid+ | name? | remark? | constraint* )>

<!ELEMENT cpu (uid+ | name? | remark? | constraint* )>

<!ELEMENT network (uid+ | name? | remark? | constraint* )>

<!ELEMENT screen (uid+ | name? | remark? | constraint* )>

<!ELEMENT storage (uid+ | name? | remark? | constraint* )>

<!ELEMENT memory (uid+ | name? | remark? | constraint* )>
```

```
<!ELEMENT printer (uid+ | name? | remark? | constraint* )>

<!ELEMENT count EMPTY>
<!ATTLIST count end    CDATA   #IMPLIED
                start  CDATA   #IMPLIED >

<!ELEMENT range EMPTY>
<!ATTLIST range min    CDATA   #IMPLIED
                max    CDATA   #IMPLIED >

<!ELEMENT ipaddress EMPTY>
<!ATTLIST ipaddress    end    CDATA   #REQUIRED
                       start  CDATA   #REQUIRED >

<!ELEMENT datetime EMPTY>
<!ATTLIST datetime end    CDATA   #IMPLIED
                   start  CDATA   #IMPLIED >

<!ELEMENT accumulated (#PCDATA )>

<!ELEMENT interval (#PCDATA )>

<!ELEMENT country (uid+ | name? | remark? | constraint* )>

<!ELEMENT quality EMPTY>
<!ATTLIST quality   resolution CDATA   #IMPLIED
                    color      CDATA   #IMPLIED
                    tonality   CDATA   #IMPLIED >

<!ELEMENT subunit (name? | remark? | constraint* )>
<!ATTLIST subunit unittype CDATA #REQUIRED >

<!ELEMENT format (#PCDATA )>

<!ELEMENT rightsholder (party+ | fixed* | percentage* | name? | remark? )>
<!ATTLIST rightsholder ID          CDATA   #IMPLIED
                       xlink:href  CDATA   #IMPLIED >

<!ELEMENT fixed (name? | remark? | party+ )>
<!ATTLISTfixed currency  CDATA   #REQUIRED
               amount    CDATA   #REQUIRED >

<!ELEMENT percentage (name? | remark? | party+ )>
<!ATTLIST percentage currency  CDATA   #REQUIRED
                     value     CDATA   #REQUIRED >

<!ELEMENT narrow EMPTY>
<!ATTLIST narrow xlink:href CDATA #IMPLIED >
```

Appendix: B ODRL XML Schema (Non-Normative)

NOTE: The XML Schema will become Normative when the XML Schema becomes a W3C Recommendation. Version 0.9 of ODRL will contain the XML Schema based on its current "Candidate Recommendation" status.

A12

wais-concepts
Wide Area Information Server Concepts

Brewster Kahle
Thinking Machines

11/3/89
Version 4, Draft

Wide Area Information Servers answer questions over a network feeding information into personal workstations or other servers. As personal workstations become sophisticated computers, much of the role of finding, selecting, and presenting can be done locally to tailor to the users interests and preferences. This paper describes how current technology can be used to open a market of information services that will allow user's workstation to act as librarian and information collection agent from a large number of sources. These ideas form the foundation of a joint project between Apple Computer, Thinking Machines, and Dow Jones. This document is intended for those that are interested in the theoretical concepts and implications of a broad-based information system.

The paper is broken up in three parts corresponding to the three components of the system: the user workstation, the servers, and the protocol that connects them. Whereas a workstation can act as a server, and a server can request information from other servers, it is useful to break up the functionality into client and server roles. A final section in the appendix outlines related systems.

Ideas for this have come from Charlie Bedard, Franklin Davis, Tom Erlickson, Carl Feynman, Danny Hillis, the Seeker group, Jim Salem, Gitta Salomon, Dave Smith, Steve Smith, Craig Stanfill, and others. I am acting as scribe. Comments are welcome (brewster@think.com).

Table of Contents

I. Introduction

Distributing knowledge was first done with human memory and oral tradition, later by manuscript, and then by paper books. While paper distribution is still efficient distribution mechanism for some information, electronic transmission makes sense for other. This project attempts to install an electronic "backbone" for distribution of information. Some information is already distributed electronically whether it is printed before it is consumed or not. This project attempts to make electronic networks the distribution technique for more types of information by exploiting new technology and standardizing on an information interchange protocol.

The problems that are being addressed in the design of this system include human interface issues, merging of information of many sources, finding applicable sources of information, and setting up a framework for the rapid proliferation of information servers. Accessing private, group, and public information with one user model implemented on personal workstations is attempted to allow users access to many sources without learning specialized commands. A system for finding information in the sea of possible sources without asking every question of every source can be accomplished by searching descriptions of sources and selecting the sources by hand.

An open protocol for connecting user interfaces on workstations and server computers is critical to the expansion of the available information servers. The success of this system lies in a "critical mass" of users and servers. This protocol, then, could be used on any electronic network from digital networks to phone lines.

For the information owners to make their data available over a server, they must be easy to start, inexpensive to operate, and profitable. One possible approach would be to provide software at a low price that will help those with information holdings to put their data on an electronic network. The power of the current personal workstations is enough to enable sophisticate information servicing capabilities. Charging for services can be done in a number of ways that do not entail setting up large billing operations. In this way, it is easy to set up, operate, and charge for information services.

The key ideas that the WAIS system are that information services should be easily and freely distributed, that the power of the current workstations can provide sophisticated tools as servers and consumers, and that electronic networks should be exploited to distribute information.

II. The Workstation's Role in WAIS

The personal workstation has grown to be a sophisticated computer that can store hundreds of books worth of information, multiprocess, and communicate over a variety of networks. The advanced capabilities of the workstation are used to find appropriate information for the user by contacting, probing, and negotiating with information servers. The explosion of available information may change the way we use computers since the usual approaches to information on workstations may not grow to make the new information environment understandable. The proposed mechanism involves finding information with one mechanism called "Content Navigation" whether the data is local or remote, available immediately or over time. This section details what a workstation might do to collect and present information from a variety of sources.

A. Accessing Documents with Content Navigation

Currently, the common way to find a document (or file) is the "Finder" on

the Macintosh or most other machines. This tree structure requires the user to remember where s/he has put each file. This approach works when a user is familiar with the file organization. It is also computationally efficient. To aid those that have forgotten the exact location many systems have some way to locate files anywhere in the structure based on the filename ("Find File" on the the Mac, and "find" on Unix machines). The number of potential files increases as the disk space become less expensive and networks let users access remote files. At some point, when the number of files becomes large, this organization can become unwieldy because of the amount the user has to remember. Another technique that is currently popular is to augment documents with static HyperText links 1,2. These links help users move through 500 Megabyte CD-ROMs of data without being overwhelmed. HyperText systems allows the author to provide "paths" through the document. The HyperCard system, from Apple, also has a simple content searching mechanism that helps navigate without those links. HyperText links give the author another tool to guide the user and augment the capabilities of the file system.

A different technique that would allow access to a large collection of documents based on document content and similarity can be called "Content Navigation." With this tool, documents are retrieved by starting with a question in English. A single line, or headline, would describe possible documents that are appropriate. These documents can be viewed, or used to further direct the search by asking for "more documents like that one". Each document on the disk (or some other source) is then scored on how well it answers the question and the top scoring documents are listed for the user. Since full natural language processing is currently impossible, each document type, be it and newspaper article or a spread sheet, must have some simple measure to determine how relevant it is to the question asked. For text documents a useful and powerful measure is to count the number of words in common between the question and the text. This well known technique of Information Retrieval1 can be augmented with different weighting schemes for different words or constructions. Other types of information might be retrieved with specific question formats.

Thus, documents can be found by asking the "navigator" for documents that contain a set of words. Those documents that share the most words with the question will come back at the top of the list (have the best "score"). In this system the "answer" to a question is not a single document, rather it is an ordered list of candidate documents.

Content navigation is not new; NeXT and Lotus have implemented systems for personal computers,2 many text database systems on mini-computers, and the DowQuest system using a super-computer. In general, there is no standardization yet on how these systems should be queried and used.

B. Dynamic Folders Find Information for the User

Content navigation takes a question and returns an ordered list of possibly relevant documents. The question can be further refined by giving feedback as to how relevant the documents were. The results of a question can be seen as cousin to the file folder in that it contains a list of documents. In reality, the answers to a questions might not be a "copy" of a document, but a "reference" or pointer to a document. These question and answer sessions can be saved just like a file folder can be saved. Saving a session also frees the machine to find answers when the user in not looking. This capability becomes important when some of the questions take time to answer because the data might be far away or difficult to answer. This section discusses one way to think of a saved question: a Dynamic Folder.

"Dynamic Folders" are a cross between a database query and a Macintosh folder that can give us great power in defining questions and probing databases. Text database queries respond with a list of pointers to "hit articles", in the form of titles or headlines, that might interest the user. At that point, the entire article can then be retrieved, if desired. A Dynamic Folder, similarly, has a question that is used to retrieve

headlines. Further a Dynamic Folder can be saved and viewed later. Since a folder is a also structure that holds documents so that they can be viewed later, a Dynamic Folder is a folder that has a question associated with it.. In that way a dynamic view acts like a database query in collecting pointers to interesting documents and like a folder in that it can be closed and opened at different times. A Dynamic Folder's question or "charter" acts as instructions to an active agent as to what what should be put in the folder. This charter gives the folder a mission to keep itself full of appropriate pointers to files or documents. This charter might be as simple as "all files on my personal disk that have a .c suffix", or all mail received in the last day. In some circumstances, it is important for a Dynamic Folder to contain pointers to a part of a file rather than to an entire file. Treating parts of files as first class documents is important in systems that group many independent documents in one file, such often done with e-mail or news articles. In this way, "documents" and "files" are slightly different.

A Dynamic Folder's contents will change when the charter has changed, at fixed intervals, or when external events happen. The user interface should indicate how current the folder is if it does not always appear up to date. Ideally, when a user changes the charter of a Dynamic Folder, the contents would reflect this instantly. This is possible for local searches and some remote searches. Sometimes, however, changes in the available documents can not be reflected immediately. This is the case when indexing the contents of new files can take a while and is done in the background. Some folders should be updated periodically to reflect new documents in remote databases. For example, a folder that uses the New York Times should be rechecked every day for new articles. Other updates to folders could be done based on events happening such as a new document being stored on the local disk. This could cause all appropriate folders to see if that file is appropriate to add to the contents.

C. Using Information Servers

Information servers sit on a network and answer questions. A server, whether local or remote, has some database that can be queried and retrieved from. These servers can be easily accessed by a workstation over a network with a standard protocol (see the Protocol section) using the Content Navigation tool to state queries and the Dynamic Folders to hold and coordinate the responses. In this way, a user's sources of information can be seamlessly expanded past the contents of the workstation without an extra conceptual burden on the user. Part of the "charter" of a Dynamic Folder, then, is the servers that it should use. This combination of tools extends the reach of the user while maintaining a consistent view of information. The capabilities of the servers will be discussed more in the server section, but it is important to see at this point that the workstation can be negotiating with a large number of local and remote servers.

D. Other User Interface Possibilities

The "Dynamic Folder" is just one way to portray the results of a question. Other visual and aural possibilities have been suggested including draw from newspapers, books, library shelves, and sound recordings. This section touches on these possibilities.

Presenting information in newspaper format has been tried at the MIT Media Lab (NewsPeek). This approach shows not only a one-line headline, but also the writer, date, place, and first few paragraphs of the article. This format expresses importance by the size of the headline typeface, the organization of the articles on the page, and the amount of text include on the first page. Advertisements also have a place in such a presentation.

Using a book or a loose-leaf binder metaphor has been explored by the Hearst group at Apple. In this model, the inside flap of the book is used to describe the charter of the book. A table of contents is the headlines

that can be retrieved. Further, the book can have sections to it separated by tabs. An index fits naturally into this model. The Dynamic Folder is a version of this idea.

Borrowing from e-mail programs, listing the possibilities in order of importance has been the technique used by Thinking Machines and NeXT for displaying candidates. Selecting an article brought the text to another window. This interface style allows the user to mark "good" documents to further refine the question. This approach is closely related to the Babyl, Rmail, and Zmail mail handler programs(ref?).

Showing the source of documents geographically was suggested by Tom Erickson of Apple. In this approach, a world map can be used to show areas of interest. This might be a good way to initiate browsing if geographical relevance is an important factor to the user. The number of articles concerning or originating from an area can be displayed conveniently.

Presenting documents like books on a shelf is a familiar metaphor to librarians. Information about the age of the book, how frequently it has been used, its size, if it is a picture book or monograph or pamphlet, when it was published (by the age of the font) are easily gathered with this presentation. Grabbing a book and looking at it, or looking on the shelves close by are natural reactions in this metaphor. I do not know of any attempts to display information in this way.

Generating a recording of a person reading the top articles can be useful for commuters. With simple skip forward and back capabilities, this might be an effective way to deliver a custom newspaper to someone driving a car. This ideally would be done with a CD player, but a cassette could be used.

The Dynamic Folder is just one possible presentation idea. This area will be an interesting area for research and prototypes.

E. Advantages of Remote and Local Filtering

When a user subscribes to a remote server, the user can get a complete copy of the database unfiltered, or can instruct the server to filter the documents remotely. Printed newspapers are delivered whole whether all of it is relevant or not. With electronic distribution, one can imagine a user asking for all sports articles but not the business articles. A query is a form of filter that works at the server. A broad query will retrieve a large number of documents that can be further filtered on the personal workstation. The system and protocols can handle filtering at either or both ends.

Local filtering can done by the content navigation on the local disk after the documents have been retrieved. The quality of this filtering will depend on the quality of the content navigator on the local workstation. The filtering might be able to use knowledge about the user that is impractical to deliver to a server. Local filtering gives the user the most flexibility, but it could entail too much communication or too much disk space. How much filtering will be done on the local workstation has tradeoffs that must be made on a server-by-server basis. If the filtering is done locally, then the workstation might have a subscription to a server that periodically retrieves the newest articles.

Remote filtering can reduce the communications bandwidth as well as possibly offer better filtering. A server can have better filtering capabilities because it can be database specific as opposed to the workstation's navigator that must be quite general. Remote filtering, just like an interactive query, in initiated by using a question.

As communications, storage, and local computation costs change relative to each other, different filtering structures might make sense.

F. Local Caching of Documents

Documents that have been retrieved from a server are stored locally on the personal workstation in a cache. A cache is a computer architecture term meaning fast, short term storage that helps speed up access by remembering commonly used entries. In this context, a cache would store documents that the user has seen or might want to see so that access to those documents would be faster and easier. A fundamental property of computer caches is that the use of the cache only makes access faster rather than changing any functionality. In certain circumstances, it might be useful to relax this constraint, but this will be seen below. Most interactive queries will only use the cache and local files because the cache will be up-to-date on its information subscriptions. The cache is very important to make queries interactive even though data may have come from remote servers.

The document cache would be stored locally but is shared between all Dynamic Folders. In this way, an article retrieved for one reason could be used in another folder without requiring two copies. A central repository would have to be managed carefully to keep the most relevant articles but not to overload the storage. A quota might be allocated to the cache, and a cache manager would make decisions about what should stay and what should go. Sometimes the user should be consulted, and other times it can be done automatically. The cache manager should keep header information on how each document in the cache such as: (1) what server the document came from, (2) how big it is, (3) if it was looked at by the user, (4) when it was retrieved, (5) what folders point to it, (6) if the user asked to keep it permanently, (7) what the user thought about it , (8) how hard is it to retrieve it again, (9) how to retrieve it again, if at all. If a document has been deleted from the cache, but it is still being referenced by a Dynamic Folder, the header information should be preserved enough to be able to retrieve the document again. In this way, deleting a document is not a catastrophe.

Since a cache can hold many of the articles seen by a user, the cache is useful in answering retrieving documents based on "I read an article once about..." (In a study of libraries users of scientific journals, about 60% of the articles read were found by browsing, and about 30% were from remembering that they saw it before and they wanted to know more). Supporting this type of question is important for a WAIS interface. The cache can help here by storing all the documents that the user has read. If the cache can not store all of them then it can be instructed as to what type of documents it should keep on hand.

G. Local Scoring of Competing Servers

Since a Dynamic Folder can get its data from many servers, it must merge this data and present it in a meaningful way to the user. While servers that rate other servers can help determine which server's answers should be valued (see the ***ratings section), these servers only rate the server as a whole and not the individual documents. Furthermore, the article could be very good, just not appropriate to the question. One way to order the responses presented to the user could be based on a "score" that is assigned to each response by the server. Each server might, for instance, judge the appropriateness of its response to the question on a scale of 1-10. These lists from multiple sources could be merged in that order (weighted by the ratings of the servers) and presented to the user. Unfortunately, since a server would want its data to be used, it has every incentive to rate all articles with at 10. Thus, determining how much to trust the server's scores will improve the selection of documents presented to the user.

One possible solution to this problem is to have local scores for servers to augment what the server says. Therefore, if a server always says "this answer is worth 10" and the user never finds it useful, then the personal workstation can lower the trustworthiness of that server's estimation of

itself. Saying 10 all the time is the equivalent to crying wolf; if it does it too often, then users will stop listening. In such a scenario, then, all responses from that server could be degraded by 30% before it is used to merge in with the other database's responses. On the other hand, other databases may underrate themselves and should be boosted. This local scoring can be used to indicate a user's satisfaction with a database and could be used by others to help in rating it. Further, this local score could be used to determine if the server is worth subscribing to or keeping its articles in the cache.

H. Budgeting the User's Time and Money

Since the users workstation will be spending the users money to contact some servers, a system of accounting and budgeting must be installed so that users get the most value for their money. The trade-offs of time and money can be tricky to try to represent, so a simple system should be attempted first. The underlying premise is that the computer knows how much it cost to use different services. This can be easy if a service charges for connect time. If a service is reached with a long distance phone call, however this rate could be difficult. (Maybe a server should be set up that knows how much the phone companies charge for different calls.) Further, if a server charges based on the question, there must be a way for the protocol for limiting the amount spent.

Some queries are going to be very important to happen quickly or they are of no use. Working this into the interface can be tricky.

Ideas towards automatic budgeting are still quite primitive. They involve global limits per month, or limits per Dynamic Folder, etc. Should the workstation enforce the limits? Who can override the limits? We need ideas on this one.

III. The Server's Role in WAIS

Servers sit on networks and answer questions. Successful servers will have some expertise or service that others find useful whether it is primary information, information about other servers, or a service. A file server, a printer, and a human travel agent can all be viewed as forms of servers. This section describes how servers might be used in a Wide Area Information Servers system.

A. Probing Information Servers

Finding documents (or more generally, information) on one's personal disk is important, but finding relevant information on remote systems would extend the usefulness of personal computers. Currently, most remote database accesses are not integrated with the workstation model using a "glass terminal" interface which does not use the power of the workstation. Some servers look like extensions of the file system and do integrate naturally (such as Sun NFS and AppleShare) but do not provide ways documents based on content. One of the major goals of the WAIS project is to integrate wide area requests in a natural way with local area requests. This section will describe how different information servers could be integrated into this model.

Using the Dynamic Folder, the user creates lasting questions that can collect answers over time from a variety of sources. The charter of a Dynamic Folder includes what sources should be used, which might include the local disk, local special purpose information servers (such as dictionaries etc), AppleShare file servers, and remote databases or WAIS (see the Examples of Information Servers section).

A wide area information server is a computer which provides information on a particular theme to other computers. Servers sit on a network, such as

the phone system, the Internet, or X.25, accept connections from other servers or users in order to answer questions in a standard format.

Each information server can be queried at the time the charter is updated, or it can be periodically polled for new information. Newspaper servers, for instance, should be polled to find new articles, while dictionary servers should only be queried once because repeatedly asking the same question is pointless. Thus, the user's workstation keeps information about each server.

While a map, a spread sheet, an airline ticket, or music might be the appropriate reply to a specific query, the initial question is stated in English. A charter (or question) about "Beethoven's choral works" might result in an article from the encyclopedia server, a schedule of concerts from the newspaper server, and recordings from a music server. Depending on the networks used, some responses might be impractical to retrieve, but the architecture allows for any type of information exchange.

A Dynamic Folder can also be used as an information server to other workstations. This simple form of server can enable others to share information easily. This capability should be put into the user interface to encourage people to exchange information. A Dynamic Folder could be "exported" or made available to those that know about it, or "advertised" by adding it to a directory of services. If it is entered into a directory (which is just another information server) then an English description of the folder should be included.

An information server is probed by putting it in the sources section of the folder's charter. These servers can be varied in size, content, and location. Using content navigation and Dynamic Folders we have an metaphor for accessing many types of information servers.

B. Examples of Information Servers

Information servers, in the broadest sense, answer questions on a particular subject on some network. Electronic networks have been used for years to distribute information in this way. Some of the servers that are available on local area networks have been:

File serving
Printers
Compute servers (such as supercomputers)
FAX
Mail services and archives
Bboard services
Modem pools
Shared databases
Text searching and automatic indexing
CD-ROM servers
Conferencing
Dictionary lookup
User's locations (finger)
Scanners/OCR
35mm Slide output

Wide area networks open up other possibilities for other services. Some services will be offered because they are expensive to offer on a local basis, because it requires some special expertise or machinery, or because it is used infrequently on a local basis. Examples of wide area services that could be offered: Current newspapers and periodicals Movie and TV schedules with reviews Bulletin boards and chat lines Archive searching through public databases Hobby specific information (ie sports scores or newletters) Mail order shopping services Banking services Talk services, bboard, and party line styles Directory information (both online sources and Yellow Pages) Scientific papers Government databases, such as patents, congressional record, and laws.

Library catalogs (eg. OCLC)
Weather predictions and maps
Usenet and Arpanet articles
Maps with driving directions included
Software distribution
Remote conferencing
Voice mail
Music and video archives
Pizza ordering

What services will be popular or commercially successful can only be guessed.

C. Navigating through the "Directory of Services"

The Directory of Servers is an information server maintains a database of available servers and how they are contacted. Like the white pages of the phone system the directory should be easy and cheap to use and include everyone. Equally important, this directory is easy to add to. Thus, people with something interesting to offer are encouraged to add their service to the directory.

A directory entry, however, should give enough information to understand what the service is and how to connect to it. This entry is similar to a yellow-pages entry in the phone book since the goal is to advertise the service. A directory entry includes: (1) Description of server in English, (2) the parent server if it is a subsidiary of a larger server, (3) related servers, (4) public encryption key, and (5) contact information including networks and contact points, (6) cost information. A local workstation would keep extra information such as: (1) locally determined "score" reflecting usefulness (2) subscription information (if any), (3) user comments, and (4) time of last contact.

This information would be used to help determine when and if the server should be contacted, and how the responses should be handled.

Navigating in the sea of servers to find new servers can be done using the content navigation technique. In this way a question on classical music would retrieve documents as well as directory entries. This could be done by storing the directory entries on the local disk (in the cache) and accessing it just like local documents based on the appropriateness of the description. Thus retrieving the document would show all the directory information. In that way, a user that is unaware of a certain server would be presented with a description of that server with a listing of its hits for the current question so that s/he could effectively evaluate its potential value of the server. If the server is added to the list of servers for that viewer, then it would be queried in the future. Maintaining an up-to-date list of services in the cache naturally falls out of content navigation and Dynamic Folders model because a directory of services viewer would have the charter to keep itself up-to-date on directory changes, and can be probed using content navigation. The directory of services viewer would list the remote directory server or servers in the sources slot. That way, the directory is kept locally and is fast to access.

Cost and availability information can help guide the workstation to alert its user to new choices of databases. If a new server appears in the directory that is cheaper than the current server, then it could be suggested as an alternative server. This can be complicated to do well, but the benefits of not having the user cull through new directory listings can warrant work in this direction. As Stewart Brand said, "One of the problems with a market based system is that you are always shopping!" Hopefully, the workstation can do some of the mindless part of comparing servers.

Directories are classically owned and serviced by the communications companies. In this role, the communications company is an unbiased party

that profits from the use of the system as a whole. Further, communications companies generally take on a teaching role to get users familiar with the system and aid those with problems. This has been true with AT&T with the telephone, the different phone companies with the 900 numbers, and the Network Information Center for the Arpanet. Whether the communications companies take over this role or not, the directory must be supported by some organization or organizations that profit from the use of the system.

D. Servers that Rate other Servers

With a large number of servers, it would be nice to know which ones are sponsored by crooks, and which ones are gems. The directory of information servers necessarily accepts all applications for inclusion, just as the white pages do. Unlike the white pages, however, is a description (or advertisement) of the server is included which can be misleading with the result that users are charged for contacting fraudulent servers. Some protection can be offered by independent servers that rate or grade other servers. These servers can serve somewhat the same roles as Consumer Reports, Better Business Bureau, and movie reviewers. This section describes what rating services might do within the WAIS system.

Just as people use movie reviewers to help them select what movies to see, rating services can help in the selection of quality servers. Servers that provide "grades" or reviews of other servers will become useful as the number of servers grow. These ratings can come in many forms such as a numeric grade, formatted reviews that can be used with filters, or a free form discussion. Thresholds can be used by different users to ensure that a server is proven before it is used. This threshold might best be used in conjunction with the cost so that even worthless, but free databases might be tried.

These rating services can come from professional servers or from friends. A user does not have to subscribe to just one rating service, since a combination might be more useful. Combining information from multiple ratings is an interesting topic for exploration. Creating the ratings server with personal ratings could also be automated somewhat since, each user's workstation keeps track of how frequently a server has been found useful. This information, or any other, can be exported so that other people can select servers that are commonly used.

Numeric ratings of servers can be merged into the user interface by helping order the documents suggested to the user. Therefore, for some user, articles from the Wall Street Journal might get better scores than a similar article in the People's Enquirer. This information could also be displayed by the color of the headline, for instance, so that unrated services would not be overly penalized.

Just as movie goers start to trust a reviewer that has agrees with them on past movies, users will trust rating services that they agree with. Selecting a rating service based on this criteria can have some interesting effects. The rating services that a user has agreed with the most will single themselves out automatically. Users with similar tastes would then find each other. With such an arrangement, one could be lead to find other servers just because other users have liked it whether it is logically related to the common servers or not. This is an automated form of the "if you like this book, then you will like this other book" system. Further, if two users like many of the same things, then they might want to meet.

A generation of server speculators can also arise. Since servers are paid based on people using them, a ratings server will want people to use them often. If agreeing with user's past evaluations is criteria for using a ratings service, then predicting what people will like will be a lucrative business. If a server turns out to be right, then it will be used more. This type of speculation is closely related to the stock market advisers that have become notable of late. A difference would be that this form of speculation is trying to predict what will be interesting to people.

E. The Role of Editors

One of the conclusions from the NewsPeek personal newspaper project at MIT (I hear) was that editors still had a place in the electronic age by reviewing and selecting certain articles as important. Unlike the rating services, an editor grades specific articles as whether they are important. These grades are similar in many ways to the rating services and might be able to be merged.

A Dynamic Folder might have a charter like: "any article from the front page of the New York Times" which is a command to use what the editor suggests the top articles are. Like the rating services, this can be independent of the sources of the articles and combine the information from multiple sources.

A form of editor server would be if users kept track of their favorite articles and put them in a Dynamic Folder and exported it for others. This way, many favorite servers might emerge and articles could be selected based on friend's suggestions.

Automatically figuring out what the user thought of a document is tricky. Clues as to what the user thought of it are: (1) how many folders point to it, (2) if the user read it, how much of it, and for how long, (3) has the user ever taken any information from it to be used in other documents, (4) has the user ever referenced it.

This type of information could greatly improve users ability to deal with the flood of available information. Furthermore, throwing away all the thoughts a user has about a document is denying others of that mental effort.

F. Markets and Hierarchies: Using Silicon Valley

Currently there are several online information providers and many online information "brokers". Brokers provide the connections between the workstations and the information providers (such as PC-link and Compuserve). Sometimes these brokers have services of their own such as electronic mail and bulletin board services. These brokers try provide a complete information environment by providing access to servers. This structure forces a new information server to be connected to many brokers to have their product used since many users only use a few brokers.. The airline reservation program Eaasy Sabre, for example, is available on 20 of these broker networks. The approach of WAIS is to have an open system of interconnection between users and servers where the brokers can act as a server, but is not an all encompassing information environment. With an open system we have a "market" of information servers rather than a controlled environment or a "hierarchy"1 . Such a structure could open up the field to many more servers and more sophisticated front-ends.

A market based approach would only standardize on the interchange formats leaving different companies free to store and service queries in any way deemed efficient. The user interfaces, similarly, are free to evolve to fit users needs. Since the protocol is not "terminal oriented" (as most systems are today), it frees the computers on either side to be sophisticated in serving the user.

Rapid evolution of a technology can happen in a market system if the structure is designed well. As long as the protocols are flexible enough to start with, and a procedure for changing the protocol is established, then the components will evolve independently by companies seeking to gain a competitive edge.

Silicon valley is an example of a market based system that led to rapid evolution of hardware in the 1970's and software in the 1980's. As the needs of the customers became understood and defined, larger companies that

had good marketing and service reputations could make the profitable components without the help of the plethora of small companies. Information servers is an innately niche-based market given the diverse information needs of the population. Furthermore, the industry is more like a service industry than a manufacturing one because of the continual need for updates and new information. For these reasons, the silicon valley structure can help in the rapid evolution of this market.

The key is to have enough users to make the servers profitable. Since, small companies can not wait long before investment turns to profit, achieving early income is important to get the system started. A "critical mass" of users might form if the first interfaces were inexpensive or free, and a few useful servers were available.

G. How Server Companies Can Make Money

If the WAIS system is to take off, then server companies must be able to make money. Companies that offer servers can make money by billing users directly, using credit cards, or by using 900 numbers to have the phone system bill the users. Direct billing is difficult to set up and can be expensive to operate, but large providers might want to do this. Credit card billing has been a popular one for information providers. This enables any network to connect the user to the server and then the user is charged for use of the server. Typically, the first transaction with a server is a negotiation of how payment will occur and the allocation of a password for future transactions. This could be automated in the WAIS system so that the workstation could know how much the costs will be and keep a total of everything spent. A risk with the credit card system is that a credit card number in the hands of a crook can enable him to make fraudulent charges. With the potentially large number of WAIS systems, this might prove dangerous. Ratings services might be able to help weed out the fraudulent information providers (if any).

Another approach is to use a phone company service over 900 numbers. When a company is assigned one of these numbers, callers are charged per minute of phone conversation and these charges appear on the phone bill every month. Typically the phone company gets 50% of the revenue from this and the charges range from $.10 to $2 per minute (PacBell gets $.25 for the first minute and $.20 thereafter). This approach eliminates the need to have a negotiation of credit card information and limits some of the risks of disclosing a credit card number. On the other hand, the charge for billing is high. Another limitation is that one must use the phone system to connect with the server.

In any case, there is very low overhead in starting a server and earning money. All one needs is a phone, a computer, and some desirable information. This is crucial to the success of the system.

All methods of billing are likely to be used and should be supported by the WAIS interfaces.

IV. The Protocol's Role in WAIS

"... they have all one language; and this is only the beginning of what they will do; and nothing that they propose to do will now be impossible for them"

Genesis 11:6

To connect a workstation to a server requires a communication network and a language to talk. The communications network can be anything that allows computers to communicate such as modems, Internet, or digital phone networks. A protocol is the language used to relate questions and receive answers between the workstations and servers. This section describes some of the issues involved in this protocol.

A. Open Protocols Promotes Wider Acceptance

It is important to the success of this system to have an open protocol that allows users to connect with servers. Several models for how to create an open standard have been tried, such as: have a company own it and license it (Adobe, for instance), have a university develop it (X Windows, for instance), have a standards organization bless it (Common Lisp, for instance), and simply make the specification available and declare is open (IBM PC, for instance). Each approach has advantages and disadvantages. The key point is that certain attributes be adhered to.

1. The companies that are developing the protocol must be open to using existing standards, and not feeling that new protocols should be protected.

2. A system for enhancements to the standard should be set up. Standards committees are often used for this.

3. The standard should be able to transmit data in a variety of formats. There are many emerging multi-media standards. A good standard will be able to transmit these information standards.

4. The query part of the protocol should be able to accept different formats of queries. Queries might, eventually, have multimedia expressions. These should be free to evolve with periodic standardization.

5. The query must have some method to transmit cost restrictions and time-outs. It should also be able to handle query forwarding while avoiding circularities.

An idea for a query language is to use English that is restricted by the constructs that are understood by the servers. As systems become more complicated, they can handle more English constructs. In this way, future server systems can get more information from a query and produce more appropriate responses, simpler systems might use the words in the query without parsing the structure of the query. This approach would allow the servers to change, while the not changing the human interface and the protocols. The English language approach has been very successful for untrained users of the Dow Jones DowQuest system.

The overall success of this system largely depends on how well these protocols work and how they are made available. There is a standard that could solve part of the problem: NISO Z39.50-1988. This standard can help with connecting to servers, delivering queries, and getting responses back. It does not specify the query language or the format of the retrieved records. Other standards may be able to aid other communications needs.

B. Hardware Independence

Since this system depends on an open protocol rather than a particular implementation, the workstation, servers, and communications systems can all be made up of various hardware technologies that would evolve in time. This independence fosters an appropriate use of all hardware pieces, and a freedom to compete to produce the best components.

Each personal workstation platform has attributes that are appropriate to exploit differently. These can be used to make tailored user interfaces. Further, a competition for the best caching and selection criteria should emerge which will hopefully settle into a good general standard. As personal workstations start to handle audio and video, these can be retrieved with the WAIS system if the bandwidth is available.

Nintendo, for instance, makes a home computer that connects to the television that is installed about 25% of all American homes. They are providing information services to 150,000 Japanese households using this technology. This might be an attractive front-end to a WAIS system.

The server computers will range from personal workstations to

supercomputers. Most databases are under 1 gigabyte so they can be stored and processed with a personal workstation unless there are a very large number of users. Supercomputers will be used in applications where there is a large amount of data or there are a very large number of users. Supercomputers can offer superior query handling by doing extensive work on each query.

The communications systems used should be any that are locally available. The bandwidth requirements for text can be satisfied with current phone systems using modems. As advances in bandwidth and connectivity emerge, such as X.25, ISDN, and InterNet; then the range of offerings from the information providers should go up.

Since no component is centralized, this system is free to be established anywhere in the world. Other more centralized systems, such as Minitel, have had difficulty in expanding outside of France. This system should encourage independent regions to set up a compatible system because of the availability of software for servers and workstations.

C. Protecting the User's Privacy

"Electrical information devices for universal, tyrannical womb-to-tomb surveillance are causing a very serious dilemma between our claim to privacy and the community's need to know."
Marshall McLuhan, Media is the Message

To encourage users to trust their personal machines with their data and interests, we must be sure to protect people's sense of privacy. As machines start to learn more about their users and start to contact other machines on their user's behalf, the dangers to privacy are significant. There are technical as well as legal issues involved. This section will cover the technical issues in protecting privacy (any good ref for the legal side?).

There is no easy way to protect a personal workstation if an intruder can get at the keyboard. Since the workstation acts on behalf of the user the potential damage that could be done by a crook at the controls would be worse than is currently possible. Since users will be leaving their computer on all the time so that it can contact servers and be used by other servers, we lose the security of the computer being off at night. One way around this might be to able to turn off input from the user while leaving the computer on to contact servers over the network. If a user knows that she is never around at night or on weekends, then this profile might help lead the system to not trust off hour use and require a password. The assumption so far in personal computers is that the machine stays in a secure physical environment and all protection must be directed to network connections. This is not a safe long term solution, and should be thought through carefully.

Other risks are involved when dealing with networks. There are problems with intruders, spies, and forgers. An intruder will try to read, modify, or destroy data that the user did not intend to leave accessible. Spies will watch the traffic from a user to determine the servers contacted and the content of the messages. A forger will copy password information to act like a different user.

Network intruders can be prevented from reading unwanted data by the user only exporting certain Dynamic Folders to become servers for the outside world. A question is whether we want "group" access as well as "world" access as in the Unix file system or some other layered approach. A Dynamic Folder only contains pointers to information. If the information is on the local disk, should that be accessible by a remote machine? Should those files be protected from being read? If the information came from a remote database, should the requester be required to get it from the source even if a copy is on site? What are the copyright issues here? Spies can watch communications networks and collect passwords and credit

card data if this information is sent in clear text (not encrypted) as well as read the data. A public key system makes sense in this application because the directory information can include a key. Public key systems are those that everyone can lock a message (encrypt) for a recipient, but only the recipient can read it. Presumably the public key system would be used in establishing a connection and a special key for the conversation would be established. Current public key systems are too compute intensive to be used for large volumes of data. A conversation key could be used with DES or some other encryption system that is easier to compute (usrEZ software has a product that runs at 30k characters/second on a MacII). Adoption of such a system early in the WAIS development would ensure that this type of protection is assumed in modern information systems.

Forgers can be foiled with a system of authentication. Authentication is important when the charges are high or when the system is used for ordering goods. One solution is to use a public key signature system that is easy to implement using the public key system (ref the Public Key papers). A signature is passed so that only the sender could have created it.

V. Conclusion: Why WAIS will Change the World

Historically, when the distribution of information became easier or less expensive, and explosive growth in learning occurred. Wide area information servers are a new way to distribute information. Since anyone with a personal computer, a phone, and some information can be a server, people are free to create and distribute their work in ways that paper distribution made impractical. The current electronic databases, in general, do not have a standard for interchange. Just as the railroads were owned and controlled by relatively few people current database brokers control access and hence the production of data. The highway system was not owned by anyone and the incremental cost to start a new business was very low. Small businesses flourished partly because of this. WAIS systems, similarly, have very low initial costs and low distribution costs which can pave the way to many servers in a short time.

Since the WAIS system is founded on computer to computer communications, new servers that just learn from other servers and produce useful information or analysis can become profitable. Such a server could be thought of as "smart" and the better servers will learn from other servers and from its own mistakes. Thus a distributed "smart" intelligence can be formed.

BBoard systems have not produced any astounding works of literature, I suggest, because it is difficult to reference older works. If older works were easy to find and reference, then people would be more inclined to make better entries. Better entries would get more references and be used more. No BBoard systems, that I know of, make this easy. Since editors, content searching, and archiving are all fundamental parts of the WAIS architecture, we stand a better chance of high quality works being produced.

A large server, or sage, has a role in this distributed system because it can infer correspondences between many pieces of information. Further, large servers will have many users that it can learn from. Users will teach a server what is important just by using the server. Thus a large server will be the place that great new ideas will be created based on lots of existing information. This new form of intelligence, that is formed out of many participating people and machines, is an exciting prospect.

VI. Related Documents

Blip Culture Hypermedia, Harry Chesley, Apple.

Catalyzing a Market of Wide Area Information Servers, Brewster Kahle.

Wide Area Information Server Demonstration, Brewster Kahle and Charlie Bedard.

Electronic Markets and Electronic Hierarchies, Thomas Malone CACM June 1987.

Introduction to Modern Information Retrieval, Gerald Salton, Cornell. McGraw Hill.

Parallel Free-text search on the Connection Machine, Stanfill and Kahle CACM Dec 1986.

VII. Appendix: Comparisons to Existing Systems

There are always precedents to any system, this one included. Some are academic and some are commercial; some are computer oriented and some are human services; some are special purpose and some are generally useful.

A. Compuserve;(of Columbus Ohio, 1-800-848-8199) is a phone based service with about 1000 services with 500,000 PC subscribers. It includes BBoards, hobby services, home shopping, email, multiuser online games, etc. Interestingly, they have contracted with the government to accept Export License Application transactions and other user interface functions. They have "Personal Newspaper" products and deliver data from many publishers. They own a lot of the underlying communication system, but are afraid of ATT and Baby Bells. They are building sophisticated user interfaces for the PCs and MACs.

Compuserve is owned by H&R Block and charges by the minute. They handle their own billing. They have recently bought most of their competitors (The Source, Access, Software House of Cambridge, and Collier-Jackson of Tampa Florida) and are making a fortune. They turned a profit in 4th quarter fiscal 1985 and by the end of fiscal 1986 it recorded a profit of $1.7 million on $100 million revenues and 300,000 users.

Compuserve is the closest model and can be easily accessed with the WAIS system. On the other hand, WAIS helps you find the database you are interested in, does not use a terminal interface (you use your PC with all of its speed), and WAIS offers subscriptions to services where your PC will keep itself informed automatically. Most importantly, WAIS is not "owned" by anyone and is free to grow independently from a centralized company.

(For more technical information I have a book of their services, Thinking Machines has an account, and I have a series of articles describing their business activities.) B. Minitel; in France is an outgrowth of the phone company. As an alternative to phone books, users were offered terminals for their homes. Many people took the terminal. By all reports it has been a very popular system. A 1986 news report said: "The directory for Minitel services is now the size of a phone directory for a small city, evidence that Minitel is a success." George Nahon, managing directory of Intelmatique: "Then need to create a market of users emerged as a prerequisite for a service." One reports speculated that France has put about $500 million into the system by 1986.

Their interface is a terminal type interface and the servers are both human and machine. [Europe is the most exciting continent for information services. It seems that they take this very seriously, while the US

government has yet to take the bold steps of investment and standardization.] C. NetLib; is a free Unix utility for distributing files through the email. Anyone that has access to the servers via electronic mail can make inquiries and file requests. This system currently has about 100 (a guess) collections world-wide and is growing. In 1987, about 10,000 requests per month were serviced. The bulk of the offerings are software programs rather than raw data. Since no charges are made for queries or requests this system is used by academics and researchers. ATT and Argonne labs are supporting this work.

The automatic reply system (remote-machine-to-local-machine rather than remote-machine-to-local-human interface) in NetLib is similar to the WAIS system. WAIS, however, is not centered solely around EMail as a transport layer; it uses the phone system as well for interactive use. Also, WAIS would help find databases that are relevant and handle the queries and requests through a more "user friendly" interface. (For more on NetLib see Distribution of Mathematical Software via Electronic Mail in Communications of the ACM May 1987) D. Switzerland system; Still assessing this system.

E. Lotus and NeXT text system Both Lotus and NeXT have text searching systems that are similar to Thinking Machine's Dow Jones system, but are based on local data (LAN based). Since disks hold close to 1 gigabyte these days, and the entire CM at Dow Jones holds 1 gigabyte, we are close in scope but not performance. On the other hand, a PC will serve its 20 users adequately and the new daily information can be effectively distributed from Dow Jones and other places. Lotus seems to be getting into the information distribution business and is writing software to process that data locally.

These companies see themselves as critically involved in this area. I believe cooperating with them is in our best interest.

F. Information Brokers Many companies act as brokers to other information providers. Often these services will offer electronic mail and bulletin boards. These private systems rarely communicate with each other. The systems that I know of are listed below. If anyone has any information on these or other companies, please tell me.

AppleLink(Personal Edition)	1-800-227-6364	getting info
Delphi	1-800-544-4005	getting info
Dialcom, Inc.	1-800-435-7342	
GE Information Services	1-800-433-3683	getting info

This company services the fortune 500 companies with network and processing services using Honeywell and IBM mainframes. They lease lines from ATT and provide an environment for their customers including network services and value added filtering and massaging of data.

GEnie	1-800-638-9636	getting info
IBM Information Network	1-800-IBM-2468 ext 100	
INet 2000/TravelNet	1-800-267-8480	bad number
Inet	1-800-322-INET	
NWI	1-800-624-5916	

Quantum Computer Services since 1985, privately held, "multimillion dollars" official commodore info service. Has been supported by commodore.

PC-link	1-800-458-8532	IBM PC product
Q-Link	1-800-392-8200	Commodore product
America online		Mac product

Snet	1-800-272-SNET Dept AA
The Source	1-800-336-3366
StarText	1-817-390-7905
Travel+Plus	1-800-544-4005
US videotel	1-713-323-3000
Western Union EasyLink	1-800-779-1111 Dept 31

Minitel Services	1-914-694-6266
Omnet/SCIENCEnet	1-617-265-9230

Other systems that I would like to find out more about: Holland system, Prodigy, Knight Ridder, Audio Tex, Airline Reservations system, Hospital Ordering System, Verity, Personal Newspaper (Media lab), Information Lens (Media Lab), SuperText.

G. Hypertext Hypertext and WAIS share many attributes for accessing textual information. In some sense, WAIS is an attempt at a large-scale hypertext system by allowing links to be deduced at run-time and across many databases stored in many places. Since servers provide pointers to documents, a pointer to a document can be put in a document and retrieved at a later time. Thus document pointers can be thought of as a crude form of hypertext link. This form of deducing hypertext links through content navigation might lead to interesting paths that are tailored to a particular user. Automatic systems will never replace the value of having users suggesting links. Suggested links can be added directly to the documents (as in most hypertext systems) or then can be made available in a distributed manner through the favorites databases. In this way, users that found certain articles to be similar or usefully viewed together can put them in a folder and export it as a database. One might ask, "Does anyone have these documents grouped in a server, and if so, what other documents are in that server?" These databases could then be used by others as evidence that they belong together. By combining many people's groupings, one can navigate through large number of documents in potentially interesting ways in a hypertext style.

1 Nelson, Ted. Literary Machines.

2 HyperCard by Apple (ref?)

1 Salton, Gerald. Introduction to Modern Information Retrieval, McGraw Hill. 1989.

2 NeXT calls theirs the Digital Librarian, and Lotus calls theirs Megellan (sp?).

1 Malone, Thomas, et al. Electronic Markets Electronic Hierarchies, CACM June 1987 Volume 30, number 6.

AZ

Deposit, Registration and Recordation in an Electronic Copyright Mangement System

by Robert E. Kahn

ABSTRACT

This document proposes the development of a testbed for deposit, registration and recordation of copyright material in a computer network environment. The testbed will involve the Library of Congress and provide for electronic deposit of information in any of several standard formats, automated submission of claims to copyright, notification of registration and support for on-line clearance of rights in an interactive network. "Digital signatures" and "privacy enhanced mail" will be used for registration and transfer of exclusive rights and other copyright related documents. Electronic mail will be used for licensing of non-exclusive rights with or without recordation. Verification and authentication of deposits can be carried out within the testbed using the original digital signatures. A system of distributed redundant "Repositories" is assumed to hold user deposits of electronic information. The testbed provides an experimental platform for concept development and evaluation, a working prototype for system implementation and a basis for subsequent deployment, if desired.

INTRODUCTION AND BACKGROUND

Deposit, registration and recordation of copyright material and its associated claims to rights have generally been handled manually. Over the past two decades, the economics of information technology has enabled an electronic foundation for such material and claims. The key elements of this foundation are the personal computers, workstations, computer networks and peripheral devices such as scanners, printers and digital storage systems which have now become sufficiently powerful and cost effective to be put into widespread use. It is now essential that the underlying systems used to manage copyright be conformed to be compatible with the promise of this new computer networking environment. This paper addresses several essential steps that should now be taken to facilitate that process.

In the current manual system, claims to copyright are registered with the Copyright Office, Library of Congress. Deposits are accepted and stored in physical form including tapes and diskettes as well as paper and other substances. Notification of registration is also made in physical form. In addition, documents transferring copyright ownership and other documents pertaining to copyright may be submitted to the Copyright Office for recordation. While an on-line record of recent registrations and recordations may be accessed at the Copyright Office, there is only limited external dissemination of this information in electronic form for access at remote sites.

This approach requires considerable physical storage at the Library of Congress for deposited materials which can only increase over time. Materials stored in physical form will slowly degrade unless deposited in digital media in which case the contents may be reproduced subsequently without loss of information but at some cost for duplication. Even if it is available digitally, much, if not most, of this material will not generally be accessible on-line from any source. Rights to use the information in a computer network environment cannot usually be acquired easily or quickly, even if the identity of the rightsholder is accurately known. Fortunately, these limitations can also be overcome with the use of information technology and only minor modification to the current manual system.

COMPONENTS OF THE PROPOSED SYSTEM

This document proposes building a testbed to develop and evaluate key elements of an electronic copyright management system. These elements include:

a. Automated copyright registration and recordation

b. Automated transactional framework for on-line clearance of rights

c. Privacy enhanced mail and digital signatures to facilitate on-line transactions

d. Methodology for deposit, registration, recordation and clearance

Current registration and recordation activities of the Library of Congress would be maintained and enhanced in the proposed testbed. It provides for repositories and recordation systems both within and without the Library of Congress, which would serve as agents for authors and other copyright owners which seek to register works with the library. In addition, the testbed provides for automated rights clearance, outside of but linked to the library, which would accelerate permissions and royalty transfers between users and rightsholders.

Electronic Copyright Management Testbed

A testbed is proposed to develop and evaluate these concepts and to obtain experience in the implementation and operation of an experimental system (see Figure 1). The proposed testbed consists of a Registration and Recording System (RRS), a Digital Library System (DLS) and a Rights Management System (RMS). The RRS will be operated by the Library of Congress and will permit automated registration of claims to copyright and recordation of transfer of ownership and other copyright related documents. The RRS would also provide evidence of "chain of title." The DLS will be a distributed system involving authors, publishers, database providers, users, and numerous organizations both public and private. It will be a repository of network accessible digital information and contain a powerful network based method of deposit, search and retrieval. The RMS will be an interactive distributed system that grants certain rights on-line and permits the selective use of copyright material on the network.

Information may be stored in the DLS, located within the DLS and retrieved from the DLS using any of several mechanisms such as file transfer, electronic mail or agents such as Knowbot programs. Material may be imported into the DLS from other independent systems, from paper and other sources or exported from the DLS to other independent systems, to paper or to other materials such as CD-ROM, DAT, and microcassettes. The electronic copyright management system described in this document would be directly linked to the DLS.

The testbed would contain a digital storage system connected to an applications gateway (which is, in turn, connected to multiple communication systems including the Internet) to which documents would be submitted. The storage system would constitute an experimental repository for information. The applications gateway would be designed to support multiple access methods including direct login. The RRS and RMS would be servers connected to the Internet. Initially, they would be on a common machine, but they could later be easily separated. After development, the RRS would be relocated to the Library of Congress or its designated agent prior to being placed in operation. After initial implementation, the repository and the RMS would be replicable at other sites.

Electronic Bibliographic Records

An electronic bibliographic record (EBR) is created by the user for each digital document submission and supplied with the document for registration. The EBR is also suitable for use in cataloging and retrieval. The EBR may be supplied to other systems without the actual document but with a pointer to it. The EBR must contain a unique name for the document per author. If a name is provided that has already been used by the same author, it will be rejected with an explanation. An acknowledgment of deposit will be returned to the user along with a unique numerical identifier and a retrieval pointer to the document, and, in the event of a claim to copyright, a certificate of registration from the RRS.

Claims Registration

When the EBR indicates a claim to copyright, the RRS will be supplied a copy of the EBR by the repository along with a digital signature (to be described shortly) that can be used to verify the accuracy of a deposit at a later time. The actual work would remain in the repository. The digital signature consists of a few hundred bytes of data and is approximately the size of the EBR. It should allow the authenticity of the retrieved document to be formally established at any time for legal and other purposes.

Repositories

The RRS need not be collocated with a repository. It is expected that an operational RRS would be operated by the Library of Congress. The repositories would be operated by the Library of Congress as well as other organizations or individuals. Deposits in certain

qualified repositories will constitute deposit for public record purposes. The Library of Congress will maintain its own repository of selected deposits.

Although a set of distributed repositories is envisioned for a widely deployed system, the proposed testbed will only have a single repository for experimentation. The repositories would be established in such a way as to insure the survival of the deposited information with perhaps different degrees of confidence (much like the treasury, banks and brokerage houses, for example). Certain information would probably not be deposited for purposes of registration and might be stored at the users local site or in a commercial repository. Highly valued information could be stored in rated repositories (5-star down to 1-star) with varying degrees of backup and corresponding costs. The most critical information, as determined by Copyright Office regulations, might be stored at the Library of Congress or the National Archives as a safeguard. The structure of such a system of repositories should be developed as part of the project.

The advantages of a distributed repository system are:

1. Large amounts of physical storage is not required to be made available at the Library of Congress.
2. Access to the original documentation is guaranteed by the DLS to the confidence level selected by the user's choice of repository (again like the banks).
3. Repositories serve as interfaces to the users, thus offloading and insulating any central servers and systems such as the RRS from potentially large user loadings and specialized customer service requests.
4. Access to the RRS in transaction mode is available only to authorized repositories and RMSs that are qualified to use the RRS in that mode. An individual author, a collective licensing organization, a government or corporate entity or others may run an RMS. Authors and other copyright owners, as well as users may also connect directly to the RRS through a separate interactive user interface.

The Computer Network Environment

There are three specific actions of concern in a network environment. One is the movement of information already contained in a computer network environment thereby greatly facilitating the creation of multiple copies in multiple machines in fractions of a second. The second is the importation of external information, such as print material or isolated CD-ROM based material, which must first be scanned or read into the system before it can be used. The third is export of internal network based information to paper using digital printers or facsimile machines or copied to separable media such as tape or DAT for external transport to others. Some of these actions, such as local use on paper in very small quantities, may or may not be covered by fair use provisions. However, non fair use actions would require approval of rightsholders.

In addition to the above three actions, there is a fourth action that is facilitated by the computer network environment. Information in digital form has the property of being easily manipulated on a computer to produce derivative works. Such derivative works can also

be easily moved about in a computer network environment and be subject to further manipulation by other parties. The technology makes it possible for parallel and concurrent manipulation of such information to result in an exponential proliferation of such derivative works.

Rights Management System

The four actions described above form a basis for a rights management system. In general, there will be many such systems operated by rightsholders or their agents for required permissions on either an exclusive or non-exclusive basis for a given type of action. To locate an RMS, a user requires the existence of a domain server that knows about the network names and addresses of all RMS servers. Transactions involving rights may be handled by direct exchange on-line between the user system and the corresponding RMS. Typically, this exchange would occur rapidly on-line, and we refer to this as the interactive clearance of rights. Privacy enhanced electronic mail would be available for exclusive licenses and other transfers of rights. Non-exclusive licenses might be handled by regular electronic mail.

Transfer of copyright ownership would usually involve recordation in the RRS and could conceivably be handled automatically by the RMS on behalf of the rightsholder and the user to facilitate matters. The confirmation from the RRS would also be passed back to the rightsholder and user directly or via the RMS using privacy enhanced mail. Various enabling mechanisms in the normal screen-based computer interface could be developed and invoked by a user to achieve rapid clearance. If included in the user interface, this capability would have the effect of creating an instant electronic marketplace for such information.

Recordation is defined to mean the official keeping of records of transfers of copyright ownership and other documents pertaining to copyright by the Copyright Office, Library of Congress. For legal purposes, proof of official registration of claims and recordations will be provided by the Copyright Office (via the RRS). Other registrations (at repositories) and non-exclusive licenses (via RMSs) will be certified by privacy enhanced mail. It will be up to the parties to such registrations and recordations to maintain electronic records of their transactions. These could also be stored within the DLS.

Identification Systems

The electronic copyright management system actually requires several types of domain servers. First, documents can be easily retrieved via the DLS if the citation is accurately known or through one or more search and browsing processes otherwise. However, the mapping of a bibliographic pointer (to the designated repository) into its network name and address may require a separate server. Second, the above mentioned domain server for RMSs is needed. Third, the date and time that transactions have been requested and taken may need to be formally validated. An electronic notary and time server would provide such a capability as part of the privacy enhanced mail system.

Retrieval, Appearance and Submission of Documents

public part of a pair of keys could use it to prepare a message which would remain confidential until the person knowing the private key used it to decrypt the message. The public keys could be listed in public directories without any special protection since knowing them did not help anyone decrypt messages encrypted using the public key. This feature makes it far simpler to manage key distribution since the public part need not be protected.

Three researchers at MIT, Rivest, Shamir and Adelman developed a pair of functions meeting the requirements specified by Diffie and Hellman. These functions are now known as the RSA algorithms (from the last names of the inventors).

Digital Signatures

Since either key of a public key cryptography pair can be used to perform the initial encryption, an interesting effect can be achieved by using the secret key of the pair to encrypt messages to be sent. Anyone with access to the public key can decrypt the message and on doing so successfully, knows that the message must have been sent by the person holding the corresponding secret key. The use of the secret key acts like a "signature" since the decryption only works with the matching public key.

Buyers could send digitally signed messages to sellers and the sellers could verify the identity of the sender by looking up the public key of the sender in a public directory and using it to verify the source of the message by successfully decrypting it.

Privacy- Enhanced Mail (PEM)

Public key cryptography can be combined with electronic mail to provide a flexible way to send confidential messages or digitally signed messages or both. In actual practice, a combination of public key, conventional secret key and another special function called cryptographic hashing is used to implement the features of privacy- enhanced mail. The public key algorithms require a substantial amount of computing power compared to conventional secret key algorithms. The older secret key algorithms, such as the Data Encryption Standard (DES) developed by the National Institutes of Standards and Technology (NIST), are much more efficient. Consequently, confidential messages are typically encrypted using a conventional secret key which, itself, is sent, encrypted in the public key of the recipient. Thus, only the recipient can decrypt the conventional secret key and, eventually, decrypt the message.

To send digitally- signed messages, each message is run through a "hashing" algorithm which produces a compressed residue which is then encrypted in the private key of the sender. The message itself is left in plain text form. The recipient can apply the same hashing algorithm and compare the compressed residue against the one that was sent (after decrypting it with the sender's public key).

One of the basic problems with this application of public key cryptography is knowing whether the public key found in the directory for a given correspondent is really that correspondent's key or a bogus one inserted by a malicious person. The way this is dealt with in the Privacy- Enhanced Mail system is to create certificates containing the name of

the owner of the public key and the public key itself, all of which are digitally signed by a well- known issuing authority. The public key of the issuing authority is widely publicized so it is possible to determine whether a given certificate is valid. The actual system is more complex because it has a hierarchy of certificate issuers, but the principles remain the same.

Notarization

Using digital signatures, it is possible to establish an on- line notarization service which accepts messages, time- stamps them and digitally signs them, then returns them in that form. If the person desiring notarization digitally signs the message at the time it is sent to the notarizing service, then it will be possible, later, to establish that the person requesting the notarizing had the document/message in question at the time it was notarized. One can imagine that the originator of a message might have it notarized for the record and the recipient might independently do so. By this means, for instance, evidence of a contract's existence in the hands of each party at particular times might be established.

VERIFICATION, AUTHENTICATION AND CERTIFICATION

The verification process uses stored digital signatures to ascertain whether a given copy is identical to the version which was originally deposited. If any portion of the copy differs from the original, the verification process will fail. Authentication or formal certification of deposits may be provided to a requesting party in traditional ways or via electronic mail. Privacy enhanced mail would be used to certify the authenticity of a deposit, as well as to certify registration and recordation records, for legal purposes.

The deployment of an electronic deposit, registration and recordation capability for use in a computer network environment would greatly facilitate and accelerate the move to a network base for information creation and dissemination. The system would be compatible with the current manual system and would support the ability of the Library of Congress to provide automated registration and recordation services. It would provide a foundation for straightforward and easy expansion and evolution and provide a direct linkage for the Library of Congress to the DLS. It would provide a prime working example for all other kinds of activities where claims registration and rights management come into play. Verification and authentication of copies of deposits may be performed electronically using digital signatures. Formal certification of deposits, as well as registration and recordation records, using privacy enhanced mail may be provided for legal purposes. A testbed which demonstrates the relevant concepts and ideas can be implemented within a two to three year period with initial limited use within a year.

Robert E. Kahn, Ph.D.
President
Corporation for National Research Initiatives
Suite 100
1895 Preston White Drive
Reston, VA 22091-5434

#413

THE DIGITAL LIBRARY PROJECT
VOLUME 1: The World of Knowbots
(DRAFT)

AN OPEN ARCHITECTURE FOR A DIGITAL LIBRARY SYSTEM

AND

A PLAN FOR ITS DEVELOPMENT

Robert E. Kahn and Vinton G. Cerf
Corporation for National Research Initiatives
March 1988

Table of Contents

Summary

This volume describes an open architecture for an important new kind of national information infrastructure which we call the Digital Library System (DLS). The architectural framework includes the DLS functional components, the methodology by which the participating systems communicate with each other, and active, mobile software components, called Knowbots, which perform services for the users. Subsequent volumes will address detailed technical aspects of the architecture such as the design of Knowbots and the protocols required to bind the DLS components together. This research was carried out by the Corporation for National Research Initiatives to specify the overall structure and function of a DLS and to provide a basis for subsequent creation of an experimental system to evaluate the concept with real users.

The term "library" conjures a variety of different images. For some, a library is a dim and dusty place filled with out-of-date texts of limited historical interest. For others, it is a rich collection of archival quality information which may include video and audio tapes, disks, printed books, magazines, periodicals, reports and newspapers. As used in this report, a library is intended to be an extension of this latter concept to include material of current and possibly only transient interest. Seen from this new perspective, the digital library is a seamless blend of the conventional archive of current or historically important information and knowledge, along with ephemeral material such as drafts, notes, memoranda and files of ongoing activity.

In its broadest sense, a DLS is made up of many Digital Libraries sharing common standards and methodologies. It involves many geographically distributed users and organizations, each of which has a digital library which contains information of both local and/or widespread interest.

Each user in the DLS manages his information with a Personal Library System (PLS) uniquely tailored to his needs. A PLS has the ability to act as a stand-alone system for its user, but under normal conditions it will be connected into a rich network of public, personal, commercial, organizational, specialized and national digital libraries.

The DLS provides each user with the capability to use other cooperating digital libraries and provides the necessary search, retrieval and accounting capabilities to support ready access to local and network-based information. The various digital libraries and the associated access to network based capabilities are integral parts of the Digital Library System. Convenient access to local and remote information, without regard for its location, is an essential goal of the system design.

The initial application of the DLS will be retrieval of specific documents for which a user may be able to supply only an imprecise description. For this purpose, we assume each retrieval request has a known target document which the user cannot describe or locate with precision, but can recognize when retrieved. Natural language and visual aids are used to assist the user in this process. However, the possible uses of a DLS are extensive and several innovative applications, discussed in the document, will be explored during the course of the project.

The potential utility for the Digital Library System technology is extremely high if agreement can be reached on appropriate standards and the relevant parties participate on a national scale. The efforts of multiple organizations such as computing equipment suppliers, publishers and other information providers and communications companies are needed to achieve the goals of the project. If successful, the results of this research offer the distinct possibility of enhancing productivity and should stimulate others to develop a vast array of new information products and services. The societal implications of success are very significant.

This document presents one practical path to the creation of such a capability. The benefits of this work depend on the outcome of the scientific research proposed herein. A plan is outlined for the development of an experimental digital library system which depends upon the active involvement of both the research community and the suppliers of equipment and services. The Corporation for National Research Initiatives looks forward to playing a leadership role in exploring the feasibility of this concept.

1. Introduction

This volume describes an open architecture for the development of a Digital Library System. The many users of such a system, even those with only limited or even no knowledge of information technology, can benefit enormously from quick and easy access to the information it contains. Its initial users will be drawn from the research community. However, the system is designed to accommodate a broad class of users (researchers and all others) in productive use of the digital library.

The Digital Library System design allows individual organizations to include their own material in the Digital Library System or to take advantage of network based information and services offered by others. It includes data that may be internal to a given organization and that which crosses organizational boundaries. This document presents a plan to develop such a system on an experimental basis with the cooperation of the research community. Finally, it addresses the application of a Digital Library System to meet a wide variety of user needs.

The productivity gains from having access to a Digital Library System are easily as large as those derived from internal combustion engines and electric motors in the early part of this century. Just as a car on an interstate highway is vastly more effective than one on a rutted dirt road, computer-based information "vehicles" can be made dramatically more effective given the proper operating environment. Computer and communications technology has made it possible for old fashioned, slow retrieval methods to be replaced by virtually instantaneous electronic retrieval. Each user of this technology can anticipate enormous potential benefit, but we lack the natural infrastructure to support this capability on a widespread basis today. This absence of infrastructure represents both a barrier and an opportunity of dramatic proportions.

1.1 A Perspective

Let us assume it is now 2003 and a little over a decade and a half has passed since the work which led to the development of a new national Information Infrastructure was started. The reality has surpassed early technological visions in unexpected ways although some of the more esoteric research ideas are still the subject of active investigation. To understand the profound nature of the revolution which has resulted from the establishment of this new infrastructure, we must reach back into history to trace the roots of human information processing.

In the early history of man, we had only an oral tradition with which to maintain our increasing fund of knowledge and recollection of history. Fathers handed down to sons the oral tradition of the tribe. The capture and rendition of this knowledge was a time-consuming process requiring frequent repetition to avoid its loss. With the invention of writing, we were able to speed up the process of information capture and simplify its reproduction. The price we paid to obtain maximum benefits for this improvement was a need to teach a larger community to read and write.

With the invention of paper and, especially with Gutenberg's invention of the printing press, the time and cost required to reproduce information was reduced dramatically. The invention of

the typewriter brought personal printing within reach of a mass market, but the modification of printed documents to reflect changes and new ideas was still a laborious process which often required the re-typing and/or reprinting of the entire document.

Then dry-process reproduction methods were discovered and subsequently fashioned into copier products. This brought rapid reproduction of printed material to a mass market at an affordable cost. Once again, the time from the creation of a document to having multiple copies available for distribution dropped dramatically.

Computers brought yet another increment in flexibility and speed to the task of recording and sharing human knowledge. In the mid-to-late 1960's, time-sharing applications and networking, along with CRT displays, made it far less costly to prepare and alter documents before committing them to paper or other permanent media. Computer-supported text editing grew even more accessible and affordable with the emergence of microprocessors in word processors and then personal computers. By the early 1980's, most documents were prepared using word processing software running on personal computers.

An adjunct development, the computer-controlled laser printer, brought an additional level of convenience and flexibility to the recording of human knowledge. At very little cost, it was possible to produce fully formatted, multi-font documents with the same degree of revisability as one had with earlier, single-font systems. Products were developed which permitted the integration of imagery and graphics in digital form along with the textual components of documents. These products were called, collectively, "desktop publishing systems."

In the late 1960's and throughout the 1970's, the sharing of mainframe resources and information through networking was a popular method for distributing the cost of gathering and maintaining special information bases. Among the many such information services developed during that period, the bibliographic retrieval systems were among the most popular. Services such as the National Library of Medicine's MEDLINE, Lockheed's DIALOG and the Bibliographic Retrieval Service (BRS) became important reference tools for a variety of users. In the legal community, Mead Data's Lexis and Nexis databases became important resources supporting the preparation and evaluation of legal cases. These tools provide full text in addition to citations to their researchers.

The practitioners of Library Science, like many others dealing with increasing amounts of information, turned to computer-based methods for assistance. The Library of Congress, the Research Libraries Group at Stanford, the National Library of Medicine and the Online Computer Library Center, Inc., joined together in a project to exchange information between their databases. This was called the Linked Systems Project.

Other information services, focused around the concept of remotely-accessible, on-line databases, emerged in this period. These included the Dow Jones/News Retrieval Service, the Data Resources, Inc., economic databases, the Thomas Register of companies and products, and hundreds of special databases reachable through Compuserve and The Source. These services, together, probably did not generate more than $300M/year by 1985; but there was a growing

interest in access to information in this computer-processible form. Most of the information service providers were desired technology which would make more uniform the varied environment in which they had to work.

Networking, along with time-sharing and personal computing combined to form the technical support base for another important technology: electronic messaging. This technology emerged in the computer science research community in the 1970's and in the public domain in the 1980's. Electronic messaging further reduced the potential time delay for propagation of documents by allowing them to be sent electronically to the appropriate recipients. By the early 1990's, standards had been established and business relationships forged which permitted the interoperation of public and private electronic messaging services. The development and deployment of Integrated Services Digital Networking facilities, which emerged slowly in the marketplace, reached an average penetration of 30-35% by 1994. The bulk of this penetration was in the commercial sector where over 70% of businesses had some form of electronic messaging service installed, while residential use had reached only about 15% of the market by that time. By the year 2000, this class of telecommunications had reached about 95% of the business market and about 35% of the residential market. Much of the early usage in the business market was attributable to "electronic data interchange" or "EDI" applications. In these applications, business documents (purchase orders, confirmation, shipping manifests and the like) were exchanged electronically along with electronic funds transfers.

The availability of a prototype Digital Library System (DLS) in 1992, with its innovative approach to intellectual property tracking, opened up a new publishing medium for information providers. In addition to the traditional book, magazine and newspaper markets and existing interactive database markets, the new Digital Library publications allowed the user to selectively view, reorganize and even update the contents for his or her personal use. Certain literary objects even had the ability to automatically update themselves with fresh information or to provide references to recent arrivals in the DLS.

Certain fundamental effects of the Digital Library System were the consequence of four distinct but strongly interacting developments during the 1990's. First, the conventions adopted for the representation of documentary material in the Digital Library were widely implemented by all vendors of combined document processing, database and spread sheet software, by vendors of electronic desktop publishing systems and by public and private libraries around the country. The existence of such common conventions made it feasible for virtually any personal computer or workstation to access and use information produced by any other similarly equipped workstation.

Second, the continuing trend in reduced cost, increased power and memory in portable computers finally reached the point where real-time speech recognition applications became cost-effective. This meant that the transcription of voice to text became affordable to the business community in 1997 and to individual users by 2001. This same computing capacity, along with the development of high resolution, flat screen, touch sensitive displays, provided a basis for the recognition and transcription of hand written or pre-printed material as well.

Direct interaction with a tablet/display and/or processing of scanned material became an affordable alternative to manual (i.e., keyboard) transcription.

The ready capture of imagery through high resolution scanning and telemetry added a third leg to the convenient creation, capture and processing of compound documents. Real-time manipulation and storage of material was also achieved.

Fourth, the incorporation of digitized sound, recorded or synthesized voice and high definition video sequences into documents stored in the Digital Library made it possible to combine most traditional forms of information publication into a common digital format. Conversion into and out of the digital forms and into the more traditional media provided bridges to older existing technologies. The structure and elementary content of printed material were determined during the scanning process.

The ability to register, store, catalog, search, retrieve and manipulate digital information in the library, combined with the variety of affordable media conversion capabilities available by the early 21st Century have led to a revolution in our social, economic and intellectual frameworks. Aided by computer-based Knowbots, easily reproduced and distributed computing cycles augmented human brainpower in the collection, use and creation of information in virtually every aspect of our lives.

Spurred, in part, by the focus of scientific attention on biology and biochemistry during the 1980's, and by the application of computer-intensive processing to non-invasive medical evaluations, the technology of the Digital Library System was applied to the capture and storage of high resolution magnetic resonance imagery (MRI), sonograms, X-ray and other similar diagnostic information. Increasingly detailed genetic analysis capabilities in combination with atomic level biochemical simulations have made it possible to carry out patient-specific bio- and chemo-therapy unthinkable in the past.

Massive amounts of detailed patient history, including the various kinds of digitized imagery, were stored in Digital Libraries around the country. This information provided a basis for epidemiological studies, simulation of experimental therapies, analysis of the population for various health trends, tissue matching and statistical analyses for predictive or retrospective purposes. Coupled with the increasing use of computers for the fabrication of prosthetics, the conduct of surgery and the evaluation of drugs for therapeutic effects, Digital Library Systems are now playing a central role in health care in the 21st century.

Virtually all economic and social transactions are now recorded in Digital Libraries: property exchanges and documentation of ownership, the creation and dissolution of businesses and other legal entities, regulations, the judgments of courts and the acts of legislatures, births, deaths, marriages and divorces, the filing of intellectual property claims and the publication of intellectual works of all kinds are registered within the framework. Entertainment and advertising, product information and actual products, if representable in digital form, are lodged in and made available through these systems. Blueprints and designs for buildings, and other kinds of physical components are required to be deposited in Digital Libraries.

The exploration of information accumulated in Digital Libraries is now an essential part of our educational and research infrastructure. Computer-based tools for search and retrieval of information (including documents) are readily available to students at all levels. The results of manned and unmanned space exploration are indelibly recorded and made accessible as part of the system. Similar aggregations of information are accumulated daily from national and international high energy physics research activities. Economic information, generated and captured in the natural course of daily transactions, is sorted, analyzed and mined by tireless Knowbots making their endless journeys through information space. Malthusian concerns about data overpopulation are easily solved by a combination of advances in high density storage systems and techniques which allow data to die a natural death.

The users of these systems draw upon natural language and visual capabilities embedded within them to find the information they need and to put it into a form suitable for further use. This information-rich, computer-aided environment has significantly changed our ability to organize into groups to achieve specific objectives. Our business organizations have taken on a much more fluid and "horizontal" character, now that the assembling and sharing of information has been made a natural side-effect of everyday interaction. New information-based products are introduced daily and are often discovered and used by other programs that serve our needs, without the need for our personal intervention at all.

Digital Libraries have now become such a pervasive part of everyday living that it's hard to remember what life was like without them. Like other infrastructure, one never really thinks about how it works, how it evolved or, how it is maintained, any more than one thinks about water, electricity, telephones and highways when they are readily available.

1.2 Technology and Infrastructure

Infrastructure plays a key role in the economic vitality of every nation. Viewed from an evolutionary perspective, infrastructure develops in response to the creation of new technologies. For example, the invention of the steam engine and its application in locomotives led to the development of railroads which, in the U.S., were instrumental in opening up the American West. Similarly, the harnessing of electricity and the invention of the light bulb preceded and motivated the development of a national power generation and distribution system. The invention of the automobile and the capability for its mass production ultimately led to the national interstate highway system which drove the evolution of suburban America. The telephone and the underlying communications technology led to a national telecommunications infrastructure.

Few inventions lead to the creation of infrastructure, but every so often, technology appears which drives this kind of development. Nearly every application which emerges at the heart of an infrastructure has an aspect of geographic dispersion and connectivity (e.g., telephone, television, roads, railroads, power generation). However, some technologies can form a kind of infrastructure without connectivity. Videocassette recorders are a prime example. Their

penetration into the residential market is the basis for the cassette rental business which could not exist otherwise.

An important characteristic of infrastructure is simplicity of use. As with electricity, the user's view of the telephone, television, and automobile is essentially simple, although each of the underlying systems is quite complex.

Simple standards governing the use and application of an infrastructure also contribute to its utility in the social and economic structure. For instance, while power generation itself can be complex, ordinary 60 cycle, 120 VAC service is easily described and used to support an unending array of devices.

Computer technology, especially the personal computer or workstation, has all the characteristics consistent with infrastructure. PCs and workstations are widely dispersed in the geographic sense. Like many new technologies, their initial applications displace older methods of achieving similar objectives (word processing versus typewriters, for example) just as cars were thought of initially as simply horseless carriages. Once these displacement applications achieve sufficient penetration, though, it is possible to introduce quite new applications which have no previous counterparts. Moreover, the relatively recent development and spread of packet communication and internetting technology adds an important ingredient, namely connectivity.

These observations suggest that the ingredients are present for the creation of a new information infrastructure based on the wide and increasing penetration of computing and communication technology into the American social and economic fabric. Although personal computers and workstations still suffer from user interface complexity, techniques have emerged (e.g., icons, mice, windows, natural language) which have the potential to simplify the use of computers considerably.

Infrastructure does not happen by accident. It is planned (either well or poorly) and deliberately created often with the direct involvement of the government. It is also preceded by a great deal of experimentation and research. The development of an information infrastructure will be no different in this regard. Unlike the industrial revolution which focused on the augmentation of human manual skills and abilities, computers offer the opportunity to enhance human cognitive capability and capacity. Over the last 40 years, the evolution of digital electronics, communication networks and computer-based applications has amply demonstrated the fertile potential of this technologies.

What is different at this juncture is that computers and digital storage technology are now readily accessible at reasonable cost to be applied to personal information tasks. At the home or office, and even on travel, the availability of computation is becoming pervasive. In the near future, shirt-pocket-size CD-ROMs (perhaps even writeable versions) will be commonplace. Use of networks to access remote databases will also continue to grow.

It is now likely that a substantial portion of the written information we encounter in the U.S. was, at one time, in computer manipulable form. Much of it has never been in that form, but the rate of production of information is so high that the more recent material significantly dominates that which has been produced in the past. Of course, the bulk of this information arrives as "marks on paper" in part because our information distribution methods are still dominated by the low cost and convenience of the printed medium.

Nowhere is the effect of this enormous influx of printed information more painfully recognized than in the research world where rapid access to relevant current and historically important results may make the critical difference between impasse and breakthrough. Finding relevant material, and even learning of its existence, is often a massive challenge. This problem is not unique to the research domain. It plagues virtually every information-dependent human endeavor.

Even if much of this information exists, however fleetingly, in computer processible form, it may not be saved or made accessible in that form. It is usually impossible for others to obtain access to it, even if they know about its existence. The need exists to establish an electronic information infrastructure to capture and allow this information to be made available (under the control of its proprietor). Computer and communication technology can be applied to augment our ability to search for, correlate, analyze and synthesize the available information. We describe such a strategy in this document. Our initial results will make it possible to find and access copies of relevant documentation rapidly and in digital form, which will be a major improvement over current practice. Moreover, it will demonstrate an important example of information infrastructure which can provide the seed for a quantum leap in the way we handle all forms of information.

As information becomes increasingly accessible and fungible (in the sense that once in digital form, it can be readily processed by computer), the entire framework for compensating the creators of intellectual property may have to shift. At present, the basis for intellectual property protection in the U.S. is Patent and Copyright law. The large scale aggregations of information found on CD-ROMs and the selective access to information found in on-line databases may require substantial re-thinking of the ways in which the creators and owners of such information are compensated for its use.

There are many issues at stake in this area, not the least of which relate to the ease with which information can be replicated once in digital form and the rapidity with which large quantities of information can be processed (accessed, transferred, analyzed, integrated, etc.). Concepts of value and pricing and royalty for use of information could require considerable revision if the cost of such use is to remain within reason. One does not now pay an author a royalty each time a book is read. However, a royalty may be earned each time a song is played in public, though not in private. If a thousand books are combined on a single CD-ROM and the acquirer of the CD-ROM only intends to read one of them, what sort of royalty arrangement is appropriate to compensate the copyright owners? How would compensation be extended for cases in which electronic copies are provided to users? In fact, the concept of copying or

duplicating a work may no longer be the essential factor in calculating royalties since far more complex actions may now be taken on digital information.

These questions are not trivial in nature nor have many workable solutions been proposed thus far. It is critical that the interplay of various user and provider interests in information be considered and reflected in the design of the new information infrastructure.

1.3 The Digital Library Project

The digital library project is a broadly based effort to achieve coherent development of our national information resources. The existence of an open architecture for Digital Library Systems will provide the necessary structure for developing rapid access to existing information resources and for creating new information resources; some will be public, some commercial, some organizational and some personal. These will all be pieces of a larger composite library system if they adhere to the open architecture. Just as the highway system required judicious choices within each region and coordination at the boundaries, so will the Digital Library System. It can and should evolve to provide a seamless structure of access to information to encompass, in as far as practicable, the needs of all members of society.

By making it easier to use existing information resources, more people will utilize them naturally and hence the size of the user base will grow. The approach outlined here is to allow the user to stipulate simply what he or she wants to have happen and to let the system take the necessary actions. For example, to retrieve and print a specific document, the user would simply cite it by name. The library system would provide the necessary means for locating the information, retrieving it, and subsequently billing the user (the user could identify that he wants to know the cost before printing).

An overall architecture is needed to guide our use of such information in the future. The Digital Library System represents one practical path to the development of a coherent information base for the management and retrieval of data. The embodiment of this architecture and its assorted functions, protocols and standards in tangible experimental system will be a major contribution to the information infrastructure of the nation.

With the development of Digital Library Systems, enormous opportunities can be foreseen for creating and selling new products and services and for stimulating very significant increases in the demand for existing products such as workstations and print servers. One potential new product is a Personal Library System (PLS) which can manage all of a user's information needs. Personal and organizational data systems are logical extensions of today's myriad software packages and numerous services based on them can easily be envisioned. As with word processing and spread sheets, the use of a PLS within the business community has the potential to streamline operations and improve productivity. For the research community, the ability to achieve quick and easy dissemination of results through electronic publishing will allow source knowledge to be propagated rapidly. For educational use, convenient and rapid access to reference material will quicken the educational pace and stimulate individual initiatives in teaching and learning.

To obtain the potential benefits of an information infrastructure, it is essential to promote the digitization of information and to insure that it becomes computer-accessible. Just as the widespread proliferation of the video cassette recorder has formed a technology base supporting an entirely new alternative to broadcasting, cablecasting and motion pictures, the provision of easy and affordable access to computer processible information leads to interesting new notions such as 1) "digital-back" publications as counterparts to hardcover and paperback books, 2) multi-media documents, whose elements may range over a substantial portion of computer-based publications, and 3) semi-automated retrieval services which can scan very large quantities of published and unpublished material for relevance to research and analysis.

Satisfying all the demands for access to on-line digital information is an overwhelming task for any one organization to undertake. Some of this information will be provided by existing suppliers and some will be created in computer-based form for the first time by new suppliers to the Digital Library. A significant portion of the material in the user's personal library will be created by the user or collected from a variety of informal sources such as personal electronic mail, clippings and intra-organizational memoranda.

For the library to be a repository for personal and organizational information, it must have the participation of many different individuals and groups within each organization. By collectively engaging the creative energies of the many individuals and organizations in the country, a critical mass effort can be realized on a national scale. This broadly based participatory theme is an important aspect of the evolution of the Digital Library System.

The introduction of an information infrastructure is strongly affected by the environment from which it must emerge. There already exists an array of mass media types (newspapers, television, magazines, books) and some fragmentary electronic facilities such as electronic mail, and computer-based teleconferencing services, on-line financial, bibliographic, technical and business databases. Alternate technologies for mass publication of digital information are beginning to proliferate. For example, Compact Disk Read Only Memories (CD-ROMs) appear to be very attractive for many applications. These include the storage of large quantities of geographic, topographic and medical imagery (e.g. Defense Mapping Agency databases, NASA LANDSAT imagery, medical magnetic resonance imagery, etc.) and for large amounts of text and imagery as might be found in an encyclopedia, Patent and Trademark files, design documents (architectural, aircraft, ships, integrated circuits, automobiles, etc.) or other reference volumes.

The development of an advanced information infrastructure must take into account a variety of existing and likely future interests and capabilities if it is to succeed. Publishers and authors must have reasonable incentives to make use of the new infrastructure. Existing libraries and their users must be able to make use of new technologies. Likewise, the educational system must be able to acquire and apply the products and services arising from a new information infrastructure if it is to serve their needs.

As viewed in this volume, the new electronic information infrastructure has a heavy computer-based aspect to it. Moreover, because the information is likely to be kept in digital form, the

telecommunications industry (including telephone, television, local and wide-area networks, cable, fiber and satellite elements) will have an important role to play in support of access, retrieval and dissemination of digital information. For example, the planned development of Integrated Services Digital Networks and the longer term Broadband-ISDN could have a profound impact on the evolution of information infrastructure by providing easily used, variable rate, switched digital communication facilities. However, the role of the carriers could change in unforseen ways due to uncertainty in the regulatory arena.

1.4 Spectrum of the Digital Library System

A large amount of information is already available in computer-based form but is not easily accessible; therefore, relatively little use is made of it. Unless one already knows how to access such information, it may not be obvious even how to get started. Exploring databases for new information is at best a highly speculative process that is often expensive and unproductive. To the providers of database services, and the suppliers of user equipment, this situation translates directly into unrealized potential. Moreover, the vast majority of information that a user may ultimately wish to retrieve surely exceeds the currently available supply by a considerable amount. Without a system for convenient and widespread access to such information by unsophisticated as well as experienced users, it may never be economical to provide it. Until it is provided, however, widespread use may be stifled. Here we see a classic chicken-egg dilemma and hence progress on both fronts moves at a glacial pace.

The spectrum of possibilities for use of a Digital Library System system ranges from the tangible to the intangible, from the very specific to the vague and from the visual to the invisible. We depict one such range of possibilities by the series of six overlapping circles in Figure 1.

At the right-hand side of the spectrum, we denote fixed format documents intended to be read by people. These are generally assumed to be prepared for publication and have definite presentation formats. These documents are stared and retrieved in their presentation form. They are guaranteed to be reproduced as they were originally created, subject only to scale and resolution limitations of the print server.

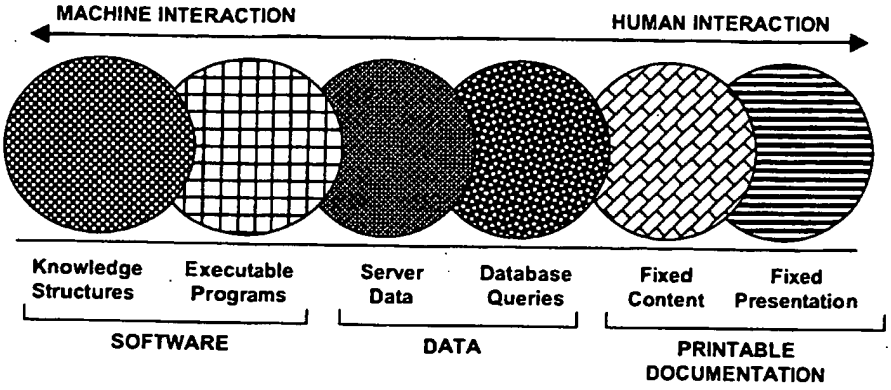


Figure 1 The Spectrum Of Library Contents

Fixed Content, flexible format documents, shown just to the left of the fixed presentation documents in Figure 1, require the user or his system to specify how to present the information, assuming its content remains unchanged. For this class of documentation, the user might wish the text to be single spaced, double spaced, margins adjusted, page boundaries adjusted, fonts changed and so forth. This is in marked contrast with the fixed format documents, where no substantial visual changes of any kind are permitted.

In the middle of the figure are shown database queries and data of the kind collected from sensors. The system treats sensor data along with database entries as if the were new types of objects in the library; this treatment requires understanding the semantics of objects in the library for the purpose of analysis and question/answering. When prestored answers are available without the need for searching documents, retrieval requests can be satisfied more quickly. Obviously, it will not be possible to anticipate all such questions in advance.

To the extreme left in Figure 1 are the two most speculative aspects of the spectrum of the Digital Library content. Although many attempts have been made to achieve reusable software, the infrastructure to reach this goal is still largely unexplored. Further, the preparation and reuse of knowledge structures in the development of intelligent systems is also virgin territory. This latter subject will be the focus of the second volume in this series.

The initial version of the Digital Library System will be tailored for the domain of printable documents (the two right hand circles in Figure 1). However, the underlying technology will be designed to allow evolution to cover the remaining portions of the spectrum. Ultimately, we see the library system encompassing the entire range of possibilities shown.

Even with this initial restriction on content, the span of possibilities for inclusion in the library is enormous. In the implementation plan (see Section 4), we discuss how the library system will be developed aid how the supply of documentation can begin and expand.

Most users subscribe to a given information service to retrieve highly selective pieces of information. Rarely do they learn to use the full complement of capabilities available on that or

any other system. Almost all existing on-line informational services support users that connect via simple alphanumeric terminals or PCs in terminal emulation mode. Most users are able to do little more than print a received text string or view it on a screen. The power of personal computers is rarely used to exploit further processing of received on-line information. With the exception of spread sheet programs that accept certain financial data obtained electronically, and mail systems that allow for forwarding, little or no user processing of received information typically occurs.

The underlying technology of the Digital Library System allows a user to access any available document within the entire Digital Library System. Using the PLS, he can modify a document in any way he chooses, incorporate it in another document, print it, search it or supply it as input to another program for further processing or display. Parts of the document can be extracted and manipulated.

Unlimited access to specific documents raises fundamental issues of intellectual property protection. A technological approach to this problem is outlined briefly in Section 3.1 of this report.

We plan to explore how to support vague and imprecise retrieval requests for specific printable documents while insuring that other well-defined requests are effectively handled as well. Requests for a specific manual, report, or equipment specification might be precise enough for the system to retrieve straightforwardly. The same might not be true if there was any uncertainty in the request. For example, if the author or title of a report were unknown, and only a general description of its subject was available, an intermediate process would be required to resolve the query further, to ask more questions of the user, or to produce a list of possible documents for selection.

1.5 A Guide to the System

A schematic description of the Digital Library System is shown in Figure 2. Its components are Personal Library Systems for the users, Organizational Library Systems for serving groups of individuals or activities, new as well as existing Databases stored locally and across the country, Database Servers to handle remote requests, and a variety of system functions to coordinate and manage the entry and retrieval of data. The system components are assumed to be linked by means of one or more interconnected computer networks.

Local requests for information, if not satisfiable by the local Personal Library, are dispatched to other, larger or possibly more specialized sources of information available through the network. A single inquiry may spawn tens to thousands of exchanges among various parts of the full Digital Library System. This could easily happen if the system must first query several databases before responding to a particular inquiry.

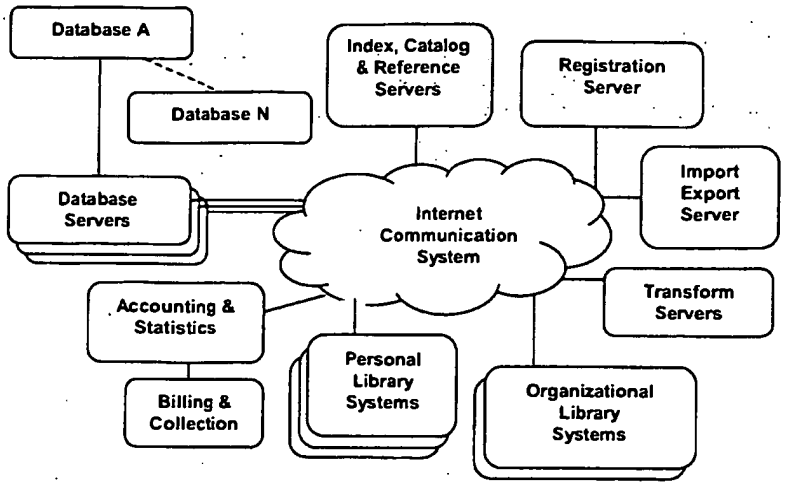


Figure 2 Structure of the Digital Library System

These exchanges are, for the most part, mediated by Knowbots, which are active intelligent programs capable of exchanging messages with each other and moving from one system to another in carrying out the wishes of the user. They may carry intermediate results, search plans and criteria, output format and organization requirements and other information relevant to the satisfaction of a user's query.

A Knowbot is typically constructed on behalf of a user at his Personal Library System and dispatched to a Database Server which interfaces the network to a particular database or set of databases. To accommodate existing database systems which are not capable of direct interaction with Knowbots, these servers can assist Knowbots in translating their information requests into terms which are compatible with the existing database's access methods. In the future, we expect to witness the development of databases systems with built-in mechanisms for housing and catering to resident or transient Knowbots. It is possible, and even likely, that more than one Knowbot may be dispatched either directly from a Personal Library System or indirectly as a result of actions taken at a particular Database Server. These various Knowbots may rendezvous at a common server or all return to the originating workstation for assembly of the results.

Two important components of the DLS shown in Figure 2, are the Import/Export Servers and the Representation Transformation Servers. The former components are responsible for accepting new documents into the Digital Library System and for dispatching documents out of the system. The latter components convert document from one internal representation to another. Depending on the nature of the output required, the obtained results may be passed through a Representation Transformation Server for conversion before being delivered. The results may be destined for either an originating PLS, a target PLS (or other workstation) designated in the original query or to an Import/Export Server if the destination is outside the particular Digital Library System "universe". For example, if the results are to be produced on

CD-ROM delivered physically to the user, this process will involve passage of the results out of the Export Server.

When Knowbots and originating workstations or other intermediate information servers need assistance in finding information, they invoke Indexing, Cataloging and Referencing Servers by causing one or more Knowbots or messages from them to be dispatched there. The Indexing, Cataloging and Referencing Servers collectively contain information about the content and organization of the Digital Library System and help to identify which Database Servers should be accessed to respond to particular types of queries.

The Indexing, Cataloging and Referencing Servers are, in turn, kept up-to-date by the Registration Server which accepts new information into the Digital Library System. The Registration Server makes use of Indexing, Cataloging and Referencing Server(s) to determine where to store new information. The Registration Server also updates the Accounting Server so that providers of information can be identified and compensation for the use of information in the Library can be properly accounted for. Users of the Digital Library are also registered with the Registration Server and information about them passed to the Accounting Server, so that access to information and billing for its use can be supported.

Records of accesses and results are collected, by means of additional Knowbots, and reported to an accounting and statistics collection system for subsequent rating or analysis. The results of accounting collection are passed to a billing and collection system for further action.

1.6 Applying the Digital Library System

The Digital Library System will only be as effective as the various uses to which it is put. A few of these will be developed during the DLS project; the remainder will occur over time through the determination of motivated individuals and organizations. By way of comparison, it is noted that most of the applications for electricity came well after its introduction. However, a few key needs drove its early development such as its use for urban lighting. Applications using electrical motors came later on.

We can only begin to speculate on the many uses of a Digital Library System. However, several needs seem clearer than others; four of them are outlined in Section 3. One need is to examine and prioritize the contents of various publications which have been identified in advance and are known to be relevant to a given worker's field. A second need is to support computer-based design activities in which access to prior designs and their context or rationale is essential. A third need is to support research activities which involved searching for documents which contain relevant information and extracting critical portions for further and possibly detailed analysis. A fourth need is to link images and text for diagnosis. Many additional needs will undoubtedly occur to others.

2. The Architecture of a Digital Library System

Before describing specific features of the Digital Library System, it will be helpful to review some of the fundamental assumptions which strongly affect its design. Perhaps the most dominant of these assumptions are that the system is distributed, heterarchical, hierarchical, networked and strongly display-oriented. In addition, it must have an ability to interact with other autonomous Digital Library Systems that do not adhere to its internal standards and procedures.

The rationale behind the first assumption (distribution), in part, is that existing digital information sources are not physically collocated and that, as a practical matter, the Digital Library System design has to accommodate many geographically distributed components. The distributed system design does not rule out the centralization or at least concentration of resources where this meets pragmatic needs for minimizing operating costs, aggregating communications facilities, and so on. The important point is that the design forces neither centralization nor pure decentralization but accommodates both styles.

We assume that users will access the services of the Digital Library from powerful, geographically distributed and often locally networked workstations. This assumption places networking at the center of the distributed architecture. Even if all the data content of the Digital Library were centralized, its users cannot be.

Distinctions between entirely different (autonomous) library systems leads to at least one level of hierarchical structure in the architecture. Components which can interact among themselves using an internal set of conventions are distinguishable from the set of components which use an external conventions. Distributed, decentralized but hierarchically structured computer services seem to be a natural consequence of the organization of the present and foreseeable marketplace for the use of systems like the Digital Library. Computer services which cross the jurisdictional boundaries between organizations, or even between divisions or departments of one organization, require management structures for access control and accounting. Services which span multiple organizations typically exhibit two or more levels of hierarchical structure stemming from the necessity to draw boundaries around component operating and management responsibilities.

Another rationale behind the hierarchical structure of the system is to constrain the scope of the data management problems so that system growth does not lead to exponential amounts of database updating and consistency checking activity. Similar motivations often impose structure on otherwise unstructured telecommunication networks, for example.

The importance of scaling in all dimensions cannot be over-emphasized. The architecture must scale in sizes and numbers of databases, numbers of users, numbers of components, bandwidth of underlying data communication, varieties of archived content and variation in presentation media and access methods.

By deliberately treating parts of the Digital Library as distinct, networked components, it becomes possible to simplify implementations and to identify explicit protocol, management and control interfaces required to carry out the functions of the system. Such structuring also has the benefit of accommodating potential competition among multiple sources for the provision of products, services and functions, which in the long run, improves user choices and enhances the opportunities for growth of the Digital Library System.

The assumption that users will access and use the services of the Digital Library through powerful, display-oriented workstations is rooted partly in the observation that personal computing and graphics-based workstation technologies are rapidly converging. As costs drop, personal computer users tend to buy increasing capability at the same cost, rather than spending less to obtain previously available capabilities. Economics aside, another reason for assuming the use of high power workstations is the need to support multi-font text, graphics, imagery and, possibly, other modalities (sound and video for example), if the full range of potential Digital Library services be supported.

Such reasoning does not rule out catering to "disadvantaged workstations," but these are treated explicitly with the realization that there is a potential loss of fidelity, functionality, or quality of service when accessing Digital Library services through these less capable devices.

The heterarchical assumption is motivated by the likelihood that more than one such Digital Library System will emerge as the national and global information infrastructure evolves. In the past, architectural designs for distributed systems often have been based on the assumption that there is only a single, monolithic, integrated architecture. Such assumptions usually lead to serious limitations on interactions between autonomous distributed systems and thus inhibit any ability for them to coordinate, cooperate and interoperate. Examples of such lack of vision may be found in many of the private and public electronic mail systems which make no provision for addressing messages outside the domain of the specific mail system in question. The Digital Library design specifically contemplates the existence of multiple instances of autonomously operating Digital Library Systems requiring compatible external interfaces. Each Personal Library System will also comprise multiple internal components which need to interact closely.

The second assumption motivating the heterarchical design stems from a belief that useful, self-contained, workstation-based, personal digital libraries are needed which can interoperate seamlessly with other internal or external library components of an organizational, regional, state or national character. The system design supports crosslinks among components at various levels in the structure and, in fact, makes heavy use of such linkages to achieve efficient interactions.

2.1 Overview of Major Library System Components

In the sections that follow, we will examine each of the major Library System components in turn, describing their functionality and relationships with other components. Figure 2 illustrates a top-level view of the Digital Library System. The rationale for the order in which these

components are described is based on following a document [or, more generally, an object as it makes its way into the Digital Library and is then accessed and used.

The principal components of the system are:

i) Import/Export Server
ii) Registration Server
iii) Indexing, Cataloging and Reference Servers
iv) Database Servers
v) Accounting and Statistics Servers
vi) Billing System
vii) Representation Transformation Servers
viii) Personal Library System

In addition to these eight basic components, there are two fundamental concepts which are intrinsic to the interaction of these various subsystems. These concepts are Knowbots and Shared Icon Geography which are discussed in more detail in Section 3. The initial information in the Digital Library system is assumed to be material which was originally intended to be printed (including multi-font text, graphics, bitmapped imagery) or otherwise displayed in static form. In addition to books, reports and periodicals, the system can include other material such as electronic mail, VLSI designs and organization charts. However, the underlying concepts will be easily extendable to allow more ambitious kinds of information such as holographs and digital films. The initial formulation of the system is organized around printable information to give the project focus and a concrete development target.

2.2 Import/Export Servers

An Import/Export Server acts as a primary interface between the Digital Library System and the outside world. Contributions to and acquisitions for the Digital Library are presented through an Import Server. The method of interaction with an Import Server forms one of the most important interfaces in the system. An Import Server will be capable of accepting contributions to the Library in many forms. Contributions and submissions might arrive as part of an electronic mail message, as a CD-ROM, as a magnetic tape, as a PC diskette or even as a facsimile scan. The common denominator is that the information has been converted to some definable digital form. One of the most important steps in the Digital Library design will be the determination of how many and which arrival formats will be acceptable. Conversion from analog to digital form, while an important consideration, is outside the scope of the library project.

The arriving objects (e.g., documents) must come with additional information if they are to be successfully entered into the DLS. Among other things, the Digital Library needs to know the origin of the object (bona fides); the owner of it (especially if any intellectual property rights are to be accounted for); terms and conditions for use, reproduction and access (including access control lists on an individual or organizational basis, for instance); descriptive information

which might aid in retrieval; relation to existing information in the Library (e.g., part of a periodical series, book series, revision, etc.); and format definition.

Information which is not in a form which can be directly accepted at an Import Server will have to be prepared by services outside the Digital Library (an opportunity for any number of public agencies or private businesses). Similarly, Import Servers for particular classes of information might be implemented and operated or sold competitively.

An Import Server extracts the information relevant to registration from the arriving submission, packages it for processing by a Registration Server, and then forms and launches a Knowbot to deliver it there. At this point, the simple model is to send all of the information, including the actual submission, along with the registration Knowbot. This could prove impractical for significant contributions such as books. An alternative is for the registration Knowbot to carry only the information needed by a Registration Server and to carry references to the storage facilities at an Import Server for use when the information is to be transferred and incorporated into a database or catalogued by an Indexing, Cataloging and Reference Server.

An Import/Export Server also provides a basic mechanism for the equivalent of interlibrary exchange services. It should be possible for several, otherwise distinct, Digital Library Systems to exchange information, queries, responses and library contents. Analogous to conventional inter-library loans, this capability is essential if the Digital Library System technology is to be independently proliferated to support a variety of products and services. Every effort must be made to assure that the architecture is free of the assumption that a single system is unique in the information universe. This does not rule out the need to tightly integrate some Digital Library components into a particular coherent system, but emphasizes the need to tolerate and accommodate diversity.

It is not yet clear whether the inter-library exchange facility can be implemented merely as an electronic message exchange or whether the interaction should also permit more immediate and direct forms of Knowbot exchange. The latter may require too much context sharing or accounting/billing and authentication mechanism to be implemented for essentially distinct Digital Library Systems. Additional research will be required on these matters. For the present, it is assumed that an electronic message exchange convention will be the basis for interactions among distinct Digital Library Systems.

All such systems, if they are to interact at all, must share a common name and address space to support message exchange. This could be provided by relying on international electronic messaging conventions which include provision of such a common name and address space for electronic mailboxes.

In addition to its import functions, an Import/Export server has the responsibility for exporting information (objects) from the local library environment to other environments. The latter may be other libraries or other presentation media (paper, CD-ROM, facsimile, etc.). An object may be exported either as the result of an action taken by a user (or a Knowbot acting on behalf of a user) or as a consequence of a request for service imported from another library system.

Although the inter-library exchange mechanism is assumed to be based on electronic mail, other less general but perhaps more efficient choices are possible. Other media conversions (e.g., to print) may have to be handled in idiosyncratic ways.

2.3 Registration Servers

Registration Server(s) are responsible for 1) receiving messages from or hosting arriving Knowbots carrying new information (or references to new information) to be added to the Digital Library, and 2) registering new users, sources of information (databases) or other components newly added to the system.

One of the most important tasks of a Registration Server is to associate a unique identifier with any new object in the system. Ideally, it should never be necessary to re-use any identifier; thus the identifiers need to be allowed to increase in length. If identifiers are to be assigned by more than one Registration Server, methods must be invoked to assure uniqueness (e.g., by prefixing the object identifiers with Registration Server identifiers).

A Registration Server reports the existence of a new object to the relevant Digital Library component. If the object is a new user, this is reported to the Accounting System and to the Indexing, Cataloging and Reference Server(s) so that queries regarding that particular user can be properly answered. New information to be added to the Library is likewise reported to the Accounting system in the event that charges are to be associated with its access and use. A Registration Server may also supply a description of the charging algorithm to be used for this information. This might be as simple as a reference to a standard algorithm or as complex as a program for computing use charges for the particular item.

If it is readily apparent which database server(s) should house the arriving object, a Registration Server will so inform the Indexing, Cataloging and Referencing Server(s) and direct the Registration Knowbot to ferry the data to the appropriate Database Server. Alternatively, if the information did not come along with the registration Knowbot, a Registration Server can form a new Knowbot to pick up the information from the Import Server and deliver it to the appropriate Database Servers.

Registration Servers interact directly with Indexing, Cataloging and Referencing Servers by providing them with an instance of the object being registered. An Indexing, Cataloging and Referencing Server determines which database can house the object (there may be more than one) and reports this information to the Registration Server. Other items, in addition to documents, which require registration in the reference database include, inter alia, all intra-library servers, users and other known Digital Library Systems.

2.4 Indexing, Cataloging and Referencing Servers

The principal function of the Indexing, Cataloging and Referencing Server(s) in the Digital Library System is to provide global cataloging and indexing services for the retrieval of Library content. The system is organized to support multiple, cooperating servers. It is also planned to

accommodate alternative, specialized Indexing, Cataloging and Referencing Servers within this architecture to take advantage of new ideas and implementations without requiring the removal or replacement of existing services.

An important design issue will be the control of potentially open-ended interactions between Registration Servers and multiple Indexing, Cataloging and Referencing Servers to avoid network congestion and deal with the resulting multiple copy database update problem. Criteria for selecting among alternative Indexing, Cataloging and Referencing Servers must be worked out, if several deal with the same or inter-related information. It is easier to deal with the case that knowledge about the content of the Digital Library is partitioned non-redundantly among several multiple servers. For instance, one server might specialize in cataloging and indexing electronic mail messages, another in books and a third in journals or other periodicals. Alternatively, if redundancy is to be supported, it might be based on multiple, complete, copies of identical indexing and cataloging information, rather than overlapping or partitioned components. Maintaining a consistent set of registration database copies is an interesting challenge in its own right.

Indexing, Cataloging and Referencing Servers are also used to locate services and users as well as information in the Digital Library. This function has an analog in the electronic mail domain in which name servers make it possible to find mailboxes associated with users. Search criteria for the name servers may be as simple as first and last personal names or complex conditional expressions, involving job title and/or function, company name, special interests (if known), locale and other identifying characteristics.

There are two distinct questions which can be answered by the Indexing, Cataloging and Referencing Server when it is dealing with Library content:

- "Here is the data, where should it be stored?"
- "Here is the kind of data I want, where is it?"

These two functions are, in fact, very similar and require the same, base level input information. Thus, any tools developed for one function can potentially carry over to the other.

Each Indexing, Cataloging and Referencing Server is capable of carrying out a repertoire of functions which can be invoked by Knowbots arriving at the Server. Knowbots arriving at a Indexing, Cataloging and Referencing Server will usually be performing one of several specific tasks:

- Cataloging/indexing of a new Library acquisition.
- Searching for a cataloged or indexed item.
- Collecting statistics about the content or usage of the Library.

When a new item is registered, a Knowbot is dispatched to a Indexing, Cataloging and Referencing Server for guidance in cataloging and indexing. The arriving Knowbot carries with it any key word or other cataloging and index terms that may have been assigned on

publication (e.g., by the Library of Congress, the journal publisher, the author, etc.). It may also carry the actual item content so as to support cataloging and indexing algorithms which operate on the full "text" of the new item. Of course, the Knowbot also carries information such as the source (author), copyright owner (if any), International Standard Book Number (or other identification of this type), publisher, date, place (and time?) of publication. Both published and unpublished works could be included.

The indexing or cataloging information may vary depending on the nature of the new item. For example, arriving electronic mail would typically be indexed by origin, To: and CC: recipients, date and time of origin, unique message identifier, originating mail system, subject matter, and depending on the Indexing, Cataloging and Referencing Server, by key words or user-provided search terms.

2.5 Database Servers

The design of the Digital Library System is intended to accommodate existing databases and database services and to provide a framework for new databases organized around the concept of Knowbotic information storage and retrieval. Database Servers bridge the gap between already existing, database services and the Digital Library System by providing support for resident and arriving Knowbots and exchange of inter-Knowbot messages. The principal tasks of the Database Servers are:

- To accept and store new information, and
- To house arriving Knowbots bearing queries

Some Database Servers may only provide the second of these functions as is likely to be the case if the actual database is managed and updated essentially outside the Digital Library System context. For database systems which are designed to operate within the Knowbotic paradigm of the Digital Library System, the functions of the Database Server may actually be combined with the database system itself. It is possible, of course, that these functions might still be supported by a separate Database Server for efficiency reasons.

Another motivation for including the Database Server in the architecture is to utilize new parallel processing technologies to speed the search and retrieval Process for both new and existing database systems. Full text databases could be searched in their entirety at very high speed. Coupled with the Knowbot concept, such special purpose servers could revolutionize the utility of existing databases. To achieve this goal, it would probably be necessary to collocate the Database Server and the database system it serves so as to provide an economical but very high speed interconnection between the two. For existing databases, such a specialized Database Server would absorb the entire database so as to permit ultra-high speed and novel searching algorithms to be applied independent of its pre-existing computational base.

Such an intimate link between the Database Server and the database will doubtless require both technical and business arrangements, particularly in cases where the database is considered to be proprietary. Where such an arrangement proves infeasible, the alternative is to configure the

Database Server so that it looks to the database as an ordinary user but provides all the of required framework for interfacing to the Knowbots of the Digital Library System.

2.6 Accounting and Statistics Servers

The function of the Accounting and Statistics Server is to collect and store data relating to the use of the Digital Library System and to send the accounting portion of it to the Billing Server. Information collected by the Accounting and Statistics Server includes not only retrieval data appropriate for billing purposes, but also statistics needed to guide operational decisions. Examples include information needed to identify capacity problems; profiles of information use (e.g., to identify the need to replicate data to reduce delay or increase transaction processing throughput); and inter-Knowbot message traffic (e.g., to determine when it would be more efficient for a Knowbot to be resident and exchange messages as opposed to moving Knowbots between a given pair of sites).

It is important to note that more than one Accounting and Statistics Server can be incorporated in the Digital Library both for redundancy and for load sharing. This means that any element of the Digital Library that produces data of interest to the Accounting and Statistics Server(s) must be configured to know to which server the data should be sent. To increase system integrity, the Accounting and Statistics Servers should be configured to accept data only from the appropriate sources and to raise alarms when data arrives from an unexpected source. Obviously, if redundancy is to be used to deal with various potential system failures, more than one Accounting and Statistics Server needs to be configured to accept data from a given source and these sources need to be configured to report to more than one Accounting and Statistics Server. This is a sensitive design area because the cost of sorting through multiple copies of accounting data collected at multiple sites is potentially very high.

The principal sources of accounting data are the Database Servers since they have direct access to querying Knowbots and their inter-Knowbot message traffic. This information is conveyed on a periodic basis (or based on the quantity of data accumulated) to the Accounting and Statistics Server. Other important sources of accounting data are the Import/Export Servers which process inter-library requests. In principle, the accounting for such queries should originate at the appropriate Database Server, but for inter-library reconciliation, the Import/Export Servers also capture traffic exchange information and pass this to the Accounting and Statistics Server.

The Registration Server is another source of accounting information since the registration of a new Library object or a new user often has accounting and pricing implications. In effect, most of the Servers in a Digital Library can be sources of accounting or statistics data, depending on the charging policy adopted by the operator of the system. An important area for agreement between two Library Systems will be their inter-library pricing and reconciliation practices.

2.7 Billing System

The Billing System generates invoices for use of the Digital Library System based on information it gets from the Accounting and Statistics Servers. The Billing System also needs to capture information about newly registered objects and users and may do this either through records sent to the Accounting and Statistics Servers by the Registration Servers or by direct exchange with the Registration Servers.

The details still need to be worked out, but it is possible that accounting data can be collected and delivered as objects like any others in the Digital Library. The mechanics of billing users and collecting revenues for service are still to be determined. By the time such a system becomes operational, direct electronic funds transfers may be the preferred collection strategy, but for the sake of backward compatibility, the system should also be capable of interfacing with a conventional lockbox service. This also implies that invoices may need to be sent either electronically (e.g., via Electronic Messaging Services) or on paper (via the postal service).

2.8 Representation Transformation Servers

The design of the Digital Library is posited on the assumption that only a few internal standard representations for library objects will be required. There will be a vast degree of heterogeneity in the actual sources of information to be placed in the Library and an equally heterogeneous collection of recipients with preferences as to the format of retrieved objects.

To avoid the need to build into the Database Servers the ability to accept or generate the entire panoply of possible object representations, the Digital Library employs Representation Transformation Servers which can accept a standard library object and convert it into any of several output representations for delivery to a user. Similarly, objects arriving at the Import/Export Server which are not in a standard library form may be converted at an appropriate Transformation Server.

It is anticipated that Transformation Services will be a lively area for competition among vendors of Digital Library products and services. Any number of such servers might operate within the context of a given Digital Library. Alternatively, the developers of such software might configure it to run in the context of a Personal Library System (see below) which would interact externally using standard object representations but could manage conversions internally using software acquired for this purpose or by means of exchanges with a Representation Transformation Server. Although the standard library representations have yet to be selected, a variety of potential representations into which or out of which it must be possible to transform already exist and will be used wherever possible.

The Association of American Publishers have adopted a version of the Standard Generalized Markup Language (SGML) as their preferred representation for the exchange of compound documents. Compound documents incorporate multi-font text and graphics in addition to raster or other bit-oriented images. If it is the case that most books and periodicals published in the U.S. will have an SGML form at some point in the process of preparation for publication, it seems reasonable that the Digital Library support this form as one of its internal standards.

In the international community, particularly in the International Standards Organization (ISO), a representation known as Office Document Architecture (ODA) is solidifying as an international standard. The National Science Foundation EXPRES project has adopted a version of ODA as its preferred representation for compound documents. This choice is compatible with the X.400 electronic messaging format recommendation of the Consultative Committee on International Telephony and Telegraphy (CCITT). Indeed, X.400 can accommodate the transport of either SGML or ODA encoded objects.

A third representation of considerable and growing popularity in the U.S. is PostScript, developed by Adobe Systems, which comprises an executable language capable of very detailed descriptions of document presentation, page layout, imagery and fonts.

A fourth representation of potential interest derives from the American National Standards Institute (ANSI) X.12C committee which is working on standards for Electronic Data Interchange (EDI), focusing particularly on business documents such as purchase requests, purchase orders, bills of lading, invoices and the like. A related set of standards have been prepared by the ANSI X.9 committee for electronic funds transfer. This representation might be important to the Personal Library System if it is applied to tracking of personal or organizational financial transactions. Electronic funds transfer mechanisms might also be invoked within and between Digital Library Systems for the purpose of achieving royalty or other compensatory payments for access to and use of the content of these systems.

A fifth representation of increasing importance is facsimile (especially Group III and Group IV). A large number of documents are now received in that form and printed on thermal paper (or plain paper), but it is not far-fetched to capture this information in digital form for storage in the Digital Library. In the long-term, one can hope for better character recognition capability so that facsimile scanned documents can be reconverted to ASCII or some multi-font encoding.

There are, in addition, a large number of different word processing formats such as those used by Wordperfect, Wordstar and Microsoft Word, to name just three. There are also numerous proprietary document representations developed by industry. The Digital Library would rely on the Representation Transformation Server to deal with these various proprietary document encodings, translating them as needed into one of the several Digital Library standards.

2.9 Personal Library System

The Personal Library System (PLS) should satisfy two distinct needs in the architecture of the Digital Library System. The first is to provide a basis for a completely stand alone instance of a library system which can operate independently from the collection of other Digital Library Systems or even components of a given DLS. The second is to interact with the other distributed components of the DLS. Both of these requirements are treated in this section.
Figure 3 illustrates an abstract view of the internal structure of a Personal Library System. The horizontal layering shown is essentially notional. There is no attempt to portray with any precision, the vertical relationships among components.

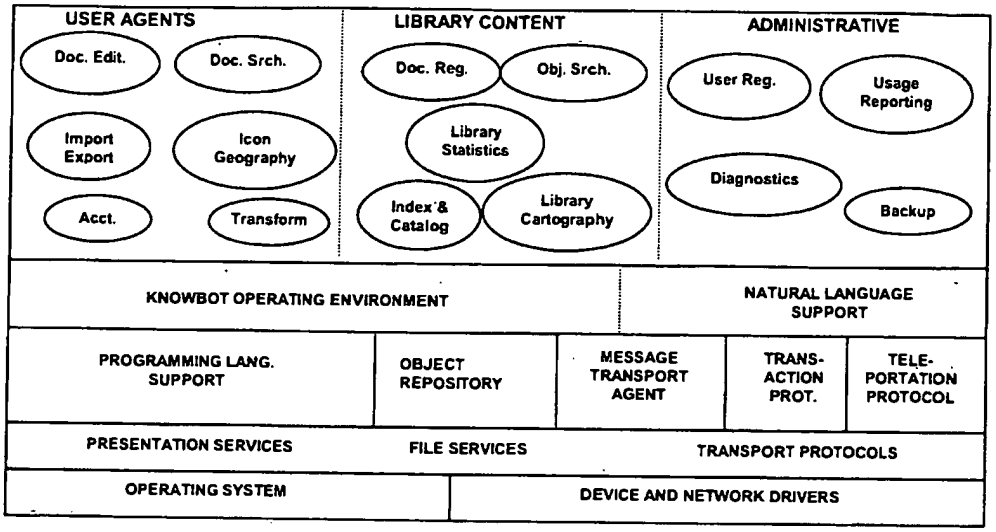


Figure 3 Personal Library System Structure

At the lowest level in the figure are the operating system and associated device and/or network drivers. It is not necessary for a PLS to be networked but it is increasingly common to find workstations interlinked on local area nets or at least capable of accessing dial-up telecommunication facilities. Although for convenience and simplicity they are not shown, included in the family of device drivers is support for common user interfacing devices such as keyboards, displays, mice and printers. These devices may eventually include audio input and output facilities and special high-resolution color displays to meet the "presentation" requirements of the contents of the library system.

The operating system will have to be capable of supporting multiple process execution. Many examples of such systems exist but the design of the Personal Library System does not impose a requirement to use one particular operating system. Whichever operating systems are selected, it is essential that they have low overhead support for interprocess communication and large scale file storage.

Transport protocols are essential when the PLS must operate as part of a larger collection of library systems within the DLS. In combination with the appropriate device drivers, the Transport Protocols enable the PLS to establish a presence in the rich networking environment and provide an avenue for access to external library services. Examples of the kinds of protocols which might be used include DOD TCP/IP/UDP, ISO TP/IP or other packet-oriented, multi-vendor protocols.

At least three application-related protocols are needed in the PLS if it is to occupy a useful place in a common networking environment. To handle electronic mail services, the PLS should support some kind of electronic message transport agent (MTA). This might be the

DOD SMTP (Simple Mail Transfer Protocol), the Multimedia Messaging Protocol (MMP) or the CCITT X.400 (Mail Handling System) protocol. To support the exchange of Knowbots between the PLS and DLS components, a "teleportation protocol" is needed. Finally, to support remote (or even local) message interaction between Knowbots, a "transaction protocol" is required. The inter-Knowbot messaging accomplished by means of the transaction protocol is distinct from the electronic mail interaction achieved using the MTA.

There is a two-fold need for the electronic mail capability in the Personal Library System. First, the PLS should be capable of assisting users in the searching, management and manipulation of their electronic mail. The PLS organization should attempt to accommodate this "under one roof;" however it is entirely reasonable for MTA functions to be provided by an electronic mail server external to the PLS but which the PLS can access to obtain copies of electronic mail intended for the PLS user. The second reason to have access to electronic mail is to provide an indirect, non-Knowbotic interface to external Digital Library Services. Distinct Digital Library Systems may not be able to share a common Knowbot Operating Environment but may want or need to exchange information. Electronic messaging technology offers one means for achieving this objective.

The Object Repository is a facility for storing the contents of the Personal Library System. The Object Repository is supported by the services of the Filing System (File Services in Fig. 3) which can be fairly conventional, but has its own organizational structure, access control mechanisms, indexing, storage and retrieval primitives. In the current design, all information stored in the Personal Library (and, in general, in the Digital Library System) is object oriented. By this it is meant that the objects have "callable interfaces." Rather than knowing the details of internal representation of an object, it is enough to be able to call on the object to supply various pieces of information (e.g., provide a bit map representation for part of a document, provide information about the content of the object such as key words, provide information about the source of an object and so on). The representation returned from such calls does have to be standardized, to permit Knowbots to manipulate arbitrary objects and their contents. The motivation for this point of view is similar to the motive for the development of object oriented languages: simplified and standardized interactions with objects while allowing substantial variation in internal representations. For information in pre-existing databases, Database Servers are used to mediate and provide arriving Knowbots with an object view of the information. The concept of Knowbots is explored in more detail in Section 3.

At the present state of design, it appears that both Knowbots and the objects they deal with can be represented using object-oriented languages. Inter-Knowbot and Knowbot-object messaging is mediated through the Transaction Protocol. Although it is still not determined, the programming language support illustrated in Fig. 3 may turn out to be identical for Knowbots and objects.

There are a number of potential object-oriented languages which might serve for the representation of objects in the Digital Library System or for the representation of Knowbots. In the most general case, even Knowbots ought to be storable in the Digital Library as objects. Examples of existing languages include Smalltalk, Common Lisp, Common LOOPS and C + +.

The selection of a methodology for building Knowbots and even the determination whether an object-oriented language is essential are two of the highest priority research questions for the Digital Library Project to resolve.

A related representation concept called "hypermedia" or "hypertext" (a term coined by Ted Nelson) also needs to be taken into account. Originating with the early work of Engelbart on the On-Line System (NLS), the notion of threading text together in multiple ways with a variety of indexing and marking mechanisms has gained currency in the late 1980's. The notion has been picked up and expanded upon by others (e.g., Xerox with its Notecards experiment and by Apple with its Hypercard product for the Macintosh).

Ultimately, the Digital Library must implement methods for the creation, maintenance and extension of a rich collection of information registered in the system. Out of this will come facilities for easy browsing and association of related information. Whether and how notions such as hypermedia are reflected in the Knowbotic paradigm of the Digital Library is one of the intriguing research areas which will be exposed by the effort to construct and use an experimental system.

The Presentation Services subsystem concerns itself with the management of user interaction facilities and includes such functions as window management; icon, graphics and multi-font text rendering; linking of displayed constructs with screen coordinates to aid mouse utilization; and sound synthesis or capture. Most of these capabilities already exist and are assumed to be available for use in the Digital Library System.

Together, the Object Repository, Programming Language Support and Protocols subsystems provide the primary support for the Knowbot Operating Environment (KNOE) which is described in Section 3. The KNOE is a collection of software which mediates the creation, cloning, destruction, scheduling and migration of Knowbots. It provides an interface to the various underlying support services, including inter-Knowbot messaging, Knowbot teleportation and access to the Object Repository. Associated with the KNOE is a Natural Language Support subsystem which is built into the environment to make more efficient the processing of natural language by Knowbots. Natural language processing requirements arise from at least two sources: the content of objects in the Library and interactions with users.

Above the level of the KNOE and its associated natural language facilities, the PLS houses a variety of Knowbots whose functions can be roughly classified into three categories: user agents, library content and administrative. The Knowbots are illustrated at the top of Fig. 3, enclosed in ellipses. The ones shown are not intended to be exhaustive but rather to suggest the kinds of functions which would be present in a stand-alone Personal Library System. Many, if not all, of these functions would also be needed for a PLS to operate in the even richer environment of multiple Digital Library Systems (or even one Digital Library System or even just another Personal Library System).

The document editing Knowbot interfaces with a user, making use of the variety of interaction support mechanisms discussed earlier. This Knowbot is capable of creating, interacting with

and altering objects in the Object Repository, presenting them or otherwise rendering ("playing" in the case of audio output) their contents. The actual implementation of a compound document editor might involve a number of Knowbots, each with specific expertise in the manipulation of different classes of information.

The document searching Knowbot has knowledge of the contents of the PLS and is capable of interacting with the user to determine what information is desired. In the context of the larger Digital Library, the document searching Knowbot must have access to knowledge about the nature and whereabouts of non-local information. Such information, contained in the Object Repository, might range from precise identification of the location of a document to information only of other Knowbots to contact to assist with the search. A consequence of program or user interaction with the document searching Knowbot may be the creation of one or more new Knowbots which can assist in carrying out the search.

One of the more interesting concepts in the user interfacing part of the Digital Library is the notion of "shared icon geography." The idea is to extend the use of icon and window style interactions to linked three dimensional models of information space which can be shared across multiple PLSs. Distributed Library contents can be visually represented and can be organized in a familiar, physical, geographic or topographic fashion. Users might travel from place to place in this space, selecting objects for examination or organizing them in a new virtual space. Object representations might be linked or stored in various places in a fictitious information space. Search Knowbots, aided by Knowbots capable of producing three dimensional renderings, could organize information in accordance with user requests. Thus, the information landscape need not be uniform or constant for all users or even for the same user.

Accounting, import/export and transformation Knowbots would provide local services to the PLS similar to those contemplated in earlier description of principal Digital Library components with similar names. The accounting Knowbot, for instance, would keep track of the usage of or reference to personal library contents and, through the usage reporting Knowbot in the administrative category (see Fig. 3), identify usage for statistical or royalty reporting purposes.

The user agent Knowbots deal largely with users or on behalf of users and interact with Knowbots in the library content and administrative categories. The library content Knowbots assist in the registration of new objects (e.g., documents), deal with searching the local object repository and capture statistics about the use or content of the library. The indexing and cataloging Knowbots are responsible for assisting in the search for or the installation of new objects in the library. As objects are added to the local library, the library cartography Knowbot keeps track of their presence. If the PLS is used to interact with other components of the Digital Library System, the cartography Knowbot captures data about the location and nature of these components and their content. Thus, the cartographic Knowbot can learn where to find objects or to find information about certain topics.

The diagnostics and backup Knowbots are tools for initiating special functional checks for proper system operation or for assuring that information stored in a Personal Library System can be reliably and redundantly archived.

In its Personal Library System mode, the user registration Knowbot is concerned with validating a local user for purposes of access control and possibly for accounting, especially in the case of access to information with associated usage fees. In the more general environment, the user registration Knowbot may be needed to validate incoming requests for information or to decide whether to host an arriving Knowbot. The Personal Library System is thus a microcosm of the larger scale Digital Library System.

3. Knowbots and Their Application

3.1 Overview

A Knowbot is an active program capable of operating in its native software environment. Knowbots are present in each of the various components of a Digital Library System. They can be cloned, replicated, created, destroyed, can be resident at a given host system or can move from one host machine to another. Knowbots communicate with each other by means of messages.

Knowbots act as the primary medium of communication and interaction between various major components of the Digital Library System. They may even transport other Knowbots. Generally, a Knowbot may be viewed as a user Knowbot or as a system Knowbot depending on whether it directly serves an individual user or not.

A user Knowbot will accept retrieval instructions from a user and determine how best to meet the stated requirements, perhaps by interacting with other Knowbots and functional elements of the Digital Library System. Knowbots then proceed to acquire the desired information by accessing the appropriate parts of the library system. In carrying out this task, they may rely on intelligent indexing services provided by other Knowbots or perform actual text searching where needed.

One set of system Knowbots specifically attend to locally available library information. They take requests from user Knowbots and actually retrieve the documents from storage (or conversely store them away). Another set of system Knowbots attend to background and administrative tasks such as diagnostics, backup and accounting.

A class of trusted Knowbots called *couriers* have the special responsibility to look after selected objects on behalf of their authors or other owners of rights in the objects. A courier may be entrusted with responsibility for an entire database or a specific document or only a portion of it. Public domain documents which may be freely transmitted, used and copied will not generally require courier services. However, we view this as a special case of a courier which is passive. For purposes of this discussion, we assume that all documents are entrusted to couriers and never appear in the library system without an accompanying courier to protect the owner's rights. The combination of a courier and its accompanying entity (e.g., paragraph, document, database) is controlled object in the system.

When a controlled object is provided to a user, all access to its contained entity is handled via its courier. If the owner of the entity originally wished to charge on a per use basis, the courier will be instructed to report such usage when it actually occurs, or to seek permission for use immediately beforehand and to deny access if it cannot be granted. Should a user wish to extract a portion of the controlled object, say for inclusion in another document, a new courier and controlled object would be created to convey the information and to represent the owner's potential interest in the user's new work.

Certain Knowbots have a permanent status within each user's system and are known as resident Knowbots. Another class of Knowbots may be spawned dynamically for the purpose of carrying out a specific task and are deleted with the task is done. These are known as transient Knowbots.

Both resident and transient Knowbots have equal status within the Digital Library System while they exist. Should a resident Knowbot need to carry out a function at another site, it will cause a transient Knowbot to be cloned for that purpose. Transient Knowbots can also be used for system updating and for populating new user systems. In this case, they might be used as templates for creating permanent resident Knowbots at the destination and then deleted.

Although the details of Knowbot construction and operation are not fully determined, the structure of a Knowbot will be refined as we explore the design of the Digital Library System. Initially, however, we envision it to behave somewhat like a cross between a Smalltalk-like object and an expert system. Thus, we expect to use many of the attributes of object-oriented programming and rule-based systems initially. As experience with this type of active programming style develops, we would expect the Knowbot concept to evolve in both structure and capability.

3.2 The Knowbot Operating Environment

Knowbots are created, destroyed and otherwise managed by a Knowbot Operating Environment called a KNOE. The KNOE provides the context in which Knowbots function within a Digital Library System. It manages the system resources needed to support them and supports inter-Knowbot communication.

A cross section of the DLS is illustrated in Figure 4. It depicts the KNOE as an annular ring and the Knowbots as circles or spheres on its periphery. Each PLS is shown as a sector or wedge containing a portion of the KNOE and some Knowbots. The principal components of the DLS are also wedges in the figure.

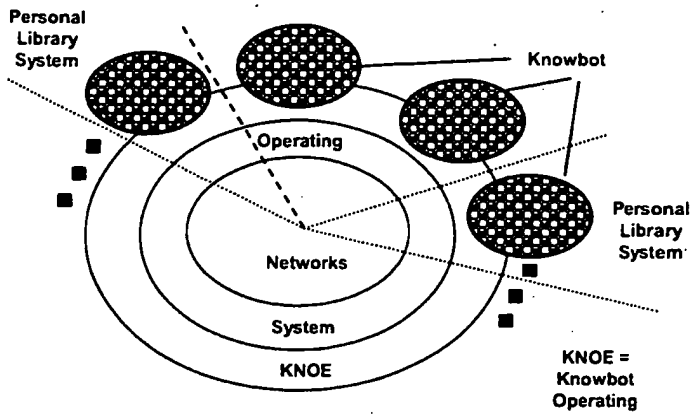


Figure 4 A Cross Section of the DLS

Each principal component of the Digital Library System contributes to and participates in the common Knowbot Operating Environment. Each local KNOE will know about all Knowbots in its local system and selected Knowbots elsewhere in the common KNOE.

Interactions between Knowbots are mediated by the KNOE. It assists in transporting messages between Knowbots in a given personal system and between systems. The KNOE will validate and authenticate messages when necessary. In a given local KNOE, any underlying message passing capabilities of the underlying operating system will be used by the KNOE in providing its layer of support.

Ideally, the KNOE could itself be created out of resident system Knowbots so that only a single architectural style is needed. However, the pragmatics of implementing the system may dictate that portions be programmed more conventionally. This aspect will be examined carefully during the early phases of the program. When detailed design and implementation choices must be made, whichever strategy (or combination) appears most desirable will be selected.

The resulting system will be designed for easy portability to other hardware and software bases. The ease of portability will depend on the extent to which the KNOE can be transported. If most of the KNOE is composed of Knowbots, then only a bootstrap version of the KNOE may be required. This is the minimum requirement on the underlying hardware and operating system. If the entire KNOE is conventionally programmed, the demands made on the underlying hardware and software may be larger as well.

3.3 Knowbots as Agents

Knowbots may themselves be nested or defined recursively by drawing on the capabilities of other Knowbots, including themselves. For example, there might be a Knowbot created to handle compound documents. This, in turn might invoke separate Knowbots for handling text, images, graphics and even electronic mail (which might itself contain a form of compound

document). It will be a design choice as to which Knowbots are visible to the user and which are hidden, in effect.

The top level of Knowbot in the system is called an *agent.* Initially, three agents are defined in the system. These are the user agent, the content agent and the administrative agent. Each agent consists of a set of resident Knowbots and each may request of the KNOE to generate transient Knowbots to assist with its work. At least initially, new agent types must be created outside the system.

The user agent consists of Knowbots for compound document generation and editing, document search and retrieval, document analysis, import/export, organizational structuring, accounting and authorization and interfacing with the user. Users deal directly with the user agent and each of its Knowbots, in turn, deals with the other agents.

The library content agent consists of Knowbots that handle document registration, indexing and cataloging, object storage and retrieval, storage management and icon geography, accounting and statistics. In addition, it contains a Knowbot to interface with other agents. The content agent is a system agent responsible for dealing directly with the object library. It receives input requests primarily from the user agent, but users do not communicate directly with it.

The administrative agent is concerned with tasks such as user registration, operations, diagnostics, backup and other similar functions such as financial analysis, billing and collection. Its main function is to support the other agents.

A typical user request for service might proceed as follows. The interface Knowbot would first determine the user's general intent and then attempt to capture what it believes is a valid user request. Let us assume the user wishes to retrieve a particular document but can only describe it generally in natural language.

The interface Knowbot would verify that the request was valid (this is largely research but simple tests can be used initially) and pass it along to the search and retrieval Knowbot to formulate a plan for satisfying the request. It might then invoke a strategy Knowbot, or a domain Knowbot to further refine its plan and then spawn one or more transient retrieval Knowbots. Each of these Knowbots might interact with other Knowbots to carry out its task.

If the requested document is not likely to be located in the user's personal database, but rather elsewhere in the system, the retrieval Knowbot is dispatched over the internet to other parts of the DLS. If the document is local, the Knowbot interacts with the storage and retrieval Knowbot in the database agent to hand off its specific task. Upon retrieval from the database, the document is supplied to the retrieval Knowbot representing the user agent after which it is ultimately made available to the user.

3.4 The User Interface

Knowbots have the primary responsibility for crafting the user's view of the library. The user conveys what he wants to see and how he would like the information presented. One or more Knowbots may then collaborate in creating the view.

The interface to the Digital Library System is essentially visual although we do not rule out other modalities such as sound. Knowbots as well as documents and controlled objects are depicted as icons and both may move dynamically in certain cases. Each Knowbot is represented by an iconic, three-dimensional symbol and its name, both of which may vary depending on the context.

The use of visual as well as logical recursion is intrinsic to the user interface. A Knowbot may be visible or invisible at the interface depending on its level of abstraction. For example, a simple search Knowbot which consists of a strategy Knowbot, an execution Knowbot and a domain Knowbot may be represented to the user as a single virtual Knowbot that does searching or as some combination of these three.

Knowbots collaborate to depict distributed objects in the DLS. Messages are used to convey the necessary information from one Knowbot to another. Multiple users will also be able to jointly participate in a joint retrieval exercise and maintain consistent views no matter which user initiates or takes an action.

The object repository may reside at multiple locations, yet the user's view should enable a single coherent logical representation of the objects independent of their location. Two users collaborating in the library system should be able to share a combination of their views as a single coherent and integrated view of the system. We refer to this aspect of the system as shared icon geography.

One of the more important concepts in the Digital Library Systems is the idea of being able to share object representations, including the details of iconic presentation, with other parts of the system. For example, if a user has a Personal Library System which contains a number of objects, it should be possible to copy the iconic representation of these objects to another Personal Library so that two users can explore the same object space together. The resulting "shared icon geography," which includes both the details of iconic presentation and the cartographic relationship among objects, would permit groups of users to work concurrently in a common information environment, coordinating the joint manipulation, examination and use of portions of the Digital Library's information space.

A particularly important issue is how to present retrieved information to the user when 1) the amount of it is inherently large or, equivalently, 2) when there are more than a few objects to be presented. This is fundamentally a research issue. We plan to seek a solution compatible with the use of shared icon geography. In addition, a simple way must be created to specify parts of the object space to browse. Electronic messages are a particularly good set of objects on which to start.

A specific request to "find the message I received 6-8 months ago about the design of the next generation workstation" may be too imprecise. Even if the system were told it was one or two pages in length with unknown sender, the Knowbot may still have to read each and every message to find the right one, if indeed it still exists (or ever did). If a few hundred messages should happen to fit the bill, the system might be unable to resolve the issue without first presenting choices to the user. The key question here is how to present this information most effectively.

As in the real world, the concept of "place" and "object" have meaning in the information space of a Digital Library. The iconic representations of places and objects in the Library are essentially multi-dimensional although they would be portrayed on a screen as two-dimensional projections of three-dimensional entities. Users interacting with iconic representations of objects should be able to use familiar, real-world paradigms to manipulate the objects and maneuver in the places that populate the Digital Library.

Objects should be able to convey the notions of containment, emptiness and fullness. It should be possible to move objects, open and close them, enter them, move about inside of them, move other objects into them and so on. It should be possible to copy all or part of an object, assuming the user has the appropriate access rights. It should also be possible to designate portions of objects to be copied and transported elsewhere. This may be achieved by some combination of highlighting and annotating the appropriate portions of the object. Two very simple examples are 1) selecting bibliographic information from a given text to be incorporated automatically in a personal data base and 2) collecting information typically found in address books such as name, address and telephone number.

The sorting and searching activities of Knowbot agents, working on behalf of one or more cooperating users, may result in the construction of three dimensional views of iconic objects found in the information space. Some of these representational ideas were originally explored by N. Negroponte of MIT in the Spatial Database Management System. Others are motivated by powerful notions of the visualization of active processes and the value of emulating common sense real-world behavior in the artificial information environment of the Digital Library System.

The realization of these ideas will require the application of leading technology in high resolution, color workstations, as well as research in powerful three-dimensional static and dynamic rendering methods, and techniques used in cinematography and television production to help "viewers" maintain context in complex visual scenes. To be effective, the user interface to the Digital Library will have to draw upon internal models of information space and options for navigating through it, techniques for animation, models of purposeful behavior and notions of goals and tasks to be accomplished. In short, the strengths of nearly every aspect of computer science, video-graphics, simulation and artificial intelligence must be marshalled to achieve the goals of the project.

3.5 Other Applications of the Digital Library System

Four possible applications of the DLS are described below. These are referred to as the Filter-Presenter, the Design Database Manager, the Researcher-Analyst and the Diagnostic Imager. Each of these uses would be implemented as an agent in the system.

The Filter-Presenter aids a user who is normally burdened by too much arriving information in the ordinary mailstream (e.g., magazines, newspapers, journals and electronic mail). If the material can be scanned electronically by the Knowbots, the user can be presented with only those aspects of the documentation he wishes to see. Of course, the user must first supply the Agent with sufficient guidance to carry out its task (including how he wants to see the results presented). Many research questions abound.

The filter may be too strong and therefore important items may be missed. Conversely, it may be too weak and the user will still be overloaded with irrelevant information. Irrelevant information may also be produced by a strong filter and relevant information missed by a weak filter, but both cases are much less likely.

The Design Database Manager couples a design program to an underlying database of relevant support information. It can also augment multiple design programs working collaboratively on a common design. In the case of VLSI design, for example, the elements in the database might be chip designs that could be used as supplements in a larger design. This could represent work underway by a team of designers. Or it could include standard designs such as simple microprocessors which may have limited use otherwise and could be used as pieces in a larger state-of-the-art chip. The advantage of this approach is that a new microprocessor design is not needed and users can be expected to be experienced in the use of existing designs for which software is already available. Alternately, the database manager could know about blueprints and bow to use them and assist a user in retrieving and interpreting them.

The Researcher-Analyst assists a person who would normally search through large collections of documentation seeking specific types of information about a particular topic. It might identify hundreds of possibly relevant items that the user would have no time to explore. This agent could search all of them and develop information for the user depending on the nature of his research. For example, if the researcher was concerned about the history of infrastructure, he might ask the system to locate as many documented examples of early uses of electricity as possible. This task might normally take weeks or months to accomplish manually but the agent using the library might accomplish it in minutes or less.

The Diagnostic Imager assists a person to find ways of binding textual or quantitative information with imagery. A reference to or selection of a given portion of an image or chart will automatically select the related textual information or vice-versa. This must be done in a well defined semantic context and not merely a geometric one.

Medical information needed to assist in patient diagnosis and evaluation covers an extreme range of modalities and levels of abstraction. From patient interviews to blood sample analyses

to X-rays, CAT scans and electrocardiograms, the Diagnostician is confronted with a rich and often perplexing array of information from which must be distilled an evaluation. In the context of the Digital Library System, the diagnostic task calls for access to a broad range of information which may range from specific information about the side effects of various drugs and chemicals to treatment protocols to indices of comparative medical imagery. The agent interacting with the user and accessing and manipulating the digital library content will rely on the use of Knowbots to transform symptomatic terms or analytic descriptions into appropriate keys for selecting useful imagery or to aid in searching for relevant treatments.

Finally, it should be noted that the structure of the DLS as a Knowbotic system makes it well suited for problems and applications that involve process control. A collection of Knowbots can be spawned to carry out a process control task and the library system architecture can be used to monitor its execution as if one were retrieving information from a more conventional library. The importance of this concept is noted, but it is not elaborated on further here.

3.6. Systems of Digital Library Systems

If an integral Digital Library System were to be constructed and placed in operation, experience predicts that evolution will result in other autonomous Digital Library Systems being generated in the future. The specifics of each system will surely differ from those of the others, and thus will arise the need for communication between these different Digital Library Systems. We call this inter-DLS communication.

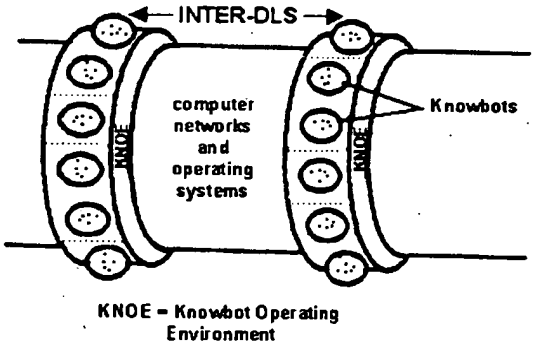


Figure 5 The KNOE Bracelet Model

We plan to define a language for inter-DLS communication which enables autonomous independent Digital Library Systems to interoperate. Most likely, this language will rely heavily on natural language as if it were supporting a normal user of either system. In this case, however, the user would be another DLS. The exact form of this language will be developed early in the project and will enable certain documents and requests to pass between systems. Inter-DLS communication is illustrated in Figure 5, where each DLS is shown as a separate bracelet on an internet-based, distributed substrate.

While a given DLS may be quite powerful in its own capabilities, the user should expect a more limited set of capabilities when multiple DLSs are involved. This may be dictated by administrative or legal restrictions dealing with enforcing copyrights as much as by technical limitations. For example, we assume Knowbots cannot be passed between autonomous DLSs and that certain documents may not be permitted to leave one DLS for another in digital form.

4. Implementation Plan

The architecture described in the previous section consists of eight significant functional component types interacting according to a set of protocols and a methodology which is observed by all the components. A three phase incremental development plan is presented here to achieve the library system program objectives. We first describe the elements which will be addressed in phase one which lasts an estimated 24 months, followed in succession by the second and third phases which last an estimated 24 and 12 months respectively. The activities in these various phases are described below.

4.1 Phase One

Four major tasks are presently envisioned for phase one. These are 1) the Knowbot Operating Environment, 2) the user interface, 3) populating the initial data bank for testing, and 4) Natural Language Text Searching. In addition, activities will be pursued in parallel to refine and further develop the overall system architecture, to explore one or more applications for the DLS and to consider matters relating to reasonable compensation for digital access and use of intellectual property.

4.1.1 The Knowbot Operating Environment (KNOE)

The description of the KNOE, the role of Knowbots, and their activities in the KNOE was presented in Section 3. This task will define the KNOE in detail, will develop a prototype system and implement three simple but functional Knowbots. Two separate subprojects are envisioned here to allow for competing views of the KNOE during this preliminary phase of the effort and to generate a total of six functional Knowbots.

The first of these two efforts, called KNOE-I, will concern itself with the issues of document creation and entry into the Digital Library System (DLS). It will focus on the user agent side of the library system and will implement Knowbots for the Document Editor, the Importer/Exporter and the Transformation Server. Using this system, documents prepared according to one standard may be manipulated by users with access to other standards or merged with documents prepared in other standards. The combined document will appear to the user as if it were a single properly merged document independent of whether the representation form of the document is changed or not. In general, the Transformation Server will convert the internal representation of a document without affecting the representation of the original copy. It will also insure that suitable notice is taken of the derivation of the work.

The Importer/Exporter will utilize a conventional database to demonstrate the issues involved with bringing new material into the Digital Library System or allowing it to migrate outside. It will also facilitate the incorporation of electronic mail or separate objects in the library system even if they were originally transmitted or received without any awareness of the Digital Library System.

The second KNOE effort, called KNOE-II, will concern itself with retrieval of documents in the library system and issues of management of the database itself It will focus on the content of the library. In particular, it will implement Knowbots for search and retrieval, indexing and cataloging and a minimal registration function. At the time of initial deposit, each document will be cataloged and indexed for future retrieval.

The search and retrieval aspect will focus primarily on interactions with the database and with the user to a lesser extent. In both of these efforts, experiments with cooperating Knowbots and multiple KNOE's will be undertaken. Knowbots will move from one operating environment to another and will cause new Knowbots to appear at the other site by message passing. These activities will be under the overall supervision and control of the KNOE. Simple document handling scenarios will be executed.

4.1.2. The User Interface

This effort will implement a prototype visual user interface consisting of a shared icon geographic view of the contents of the library system and its components. Two components of the user interface will be the icon geography system which is responsible for interacting with the user and the library cartographic system which maps the relevant contents of the library and interacts with the object repository. These two systems interact with each other directly.

Initially, the functionality of these systems will be explored and demonstrated. Ultimately, each will be represented as separate Knowbots within the system, once the concept of Knowbots has been demonstrated.

4.1.3 Populating the Database

This effort will address those key aspects of the database having to do with personal, organizational (inter-organizational) and public information. Initially, the focus will be on populating an experimental database with objects that are publicly available or of organizational interest to be used for testing. After the Database Server is developed in phase two, network connections will be made available to existing databases.

We envision this effort will involve use of representation standards already in existence (or developed in the program, if necessary) and concentrate on collection and creation of an initial database. The equipment used will include scanners, optical character readers, and facsimile devices.

Public Documents - Our initial focus here will be to collect in digital form relevant standard documents from organizations such as ANSI, CCITT, ISO, IEEE, NBS or governmental agency pronouncements.

Organizational - Here we plan to focus on selected equipment manuals that are nominally available to customers or for internal use. This might cover hardware, software, and procedures for installation, use and repair. Eventually it might include brochures, pictures and

specification sheets as well. Another organizational focus will be to collect and represent vita's of graduate researchers in the nation's colleges and universities. This will be carried out in cooperation with the university librarians who will each have responsibility for accurately representing their own school.

Personal - Finally, to make this effort interesting to the research community and to motivate some of their work, we plan to include selected research reports issued by the various universities which are not easily available otherwise. In addition, we shall include selected scientific and business publications based on their relevance and the willingness of the publishers to cooperate. Candidates are AAAI, IEEE, Scientific American, and ACM on the scientific side, and Business Week, Harvard Business Review, Fortune and Forbes on the business side.

4.1.4 Natural Language Text Search

This effort will focus on demonstration of the use of natural language for search and retrieval. Initially, electronic mail will be chosen as a candidate to demonstrate retrieval based on imprecise English requests. When an object base with actual documentation is ready in the library, the domain will be expanded to include it as well.

This task will entail dealing with ungrammatical writing (but will not necessitate building a theory of such writing) and with understanding quite a bit about messages in general. Nonetheless, the domain appears to be relatively bounded.

The Natural Language System will be structured so that it draws upon existing state of the art technology but modifies it so that the system may be handed critical information it needs to do its job. Ultimately, this information will be provided directly by the Knowbots. In general, the Natural Language System will know about everyday English words. The system will receive a lexicon of relevant specialized words and their meanings, along with any additional helpful information. The system shall be structured such that new grammars and basic vocabularies may eventually be supplied to handle other languages.

In parallel with the above activities we expect several organizations to begin work on the development of a Personal Library System which provides user access to the library. These efforts may be simply to interact with the external development efforts, provide one or more individuals to work on them or begin an internal effort.

Throughout phase one we shall need to maintain, refine and update the architectural specification of the library system. At critical junctures, preliminary protocol specification documents will be produced for the critical interfaces between the principal components that were shown in Figure 2, for representation standards as well as for normal system capabilities such as Knowbots. This will be a living document which shall serve as the "sheet music" for the entire effort.

During this phase we will explore several possible applications of the Digital Library System along the lines indicated in Section 3.3.6. Initial designs for one or more of these efforts will begin. We shall also explore concepts for handling the reasonable compensation of intellectual property owners, since fragments of their as well as entire works may be involved.

4.2 Phase Two

In the second phase, we plan to begin development of a Personal Library System based on the research efforts in the first phase. In addition, we expect several industrial organizations to actively participate in the process so that a commercial source of the technology will ultimately be available.

The initial system will be based on a powerful workstation with local disk, scanner, printer and high resolution display (plus mouse and of course keyboard). Eventually, it will be graphics capable and be equipped with facsimile, low-cost personal high density storage such as a CD-ROM, plus an acoustic subsystem (including speech) and video.

Research will continue on several fronts. First, the Knowbot research will be expanded to include exchanges between Knowbots and making Knowbots work in collaboration with the object base. This effort will include Knowbots with domain expertise and the ability to do simple domain related problem solving.

The registration Knowbot will be fully developed and outfitted to handle users and services as well as documents. In addition, it will be expanded to be knowledgeable about an organizational level as well as a personal level of objects. It will also have a mechanism for internal collection of accounting information.

Research will also be undertaken to develop a Knowbot which understands organizational structure and documents associated with it. The structure of an organizational library system will be developed and simple exchanges between organizational and personal Knowbots explored. The Personal Library System technology will provide the basis for the organizational system as well; only the contents of the two systems will differ.

The task of database population will continue as a larger and richer set of documents is added to the object base. We expect this task to focus primarily on the more sophisticated elements of the existing documents such as equations, graphics and images. However, additional documentation will be added as appropriate for the research to be conducted.

The natural language capability for text search will be improved and a user front end will be incorporated which relies on a common body of natural language software. Experimentation with sample retrieval requests will take place interactively.

A Multi-Processor Database Server will be developed to access remote Database Systems. Initial experiments will be conducted with one or more cooperative suppliers of information. Candidates are the National Library of Medicine and Dow Jones. The Database Server will

incorporate much of the Personal Library System software but be outfitted to operate several orders of magnitude faster than the personal workstation and support multiple users. The server will draw directly upon the results of the Personal Library System research and apply it to multi-processors.

Finally, an effort will be undertaken to focus on some of the practical and administrative considerations such as tools for billing and collection, diagnostics, back up, etc. Also, those aspects which allow new capabilities to be added on the system to be reconfigured over time will be included here.

Sometime during Phase Two, we expect to obtain a working prototype of the initial Digital Library System and to begin making it available to selected members in the research community for experimentation and as an object of research itself. In addition, the development of one or more of the applications will be undertaken.

4.3 Phase Three

The components of the Digital Library System (DLS) will be fully integrated and a quasi-operational DLS will be created during this phase for research purposes. The existence of the system will serve to expand the Digital Library System to more users and a larger set of documentation. We expect to gain operational experience and to incorporate several additional public and private systems such as NTIS, Westlaw, Lexis/Nexis, Compuserve or Dialog.

An organizational prototype will be created working with one or more groups and experiments with inter-organization exchanges explored. Examples of inter-organizational exchange requirements will be developed and simple interactions carried out. These might include access to manuals, student vita, electronic mail, or open memoranda. Acoustic Input and Output including speech and other audible sounds will be incorporated into the user interface and further work on the shared icon geography carried out for dealing with complex representations. One or more applications will be substantially completed.

Finally, we assume that other concepts for a Digital Library System will emerge and compatibility with them will be required. Hence, we will investigate the requirements for inter-Digital Library System exchanges and will begin experimenting with such in the context of two autonomous (but homogeneous) Digital Library Systems.

4.4 Follow-on Plans

At this stage, an experimental Digital Library System will be functioning with a small but interesting class of documents, an initial application, a nominal class of users and an expandable architecture. If the system performs effectively, as we expect it will, it is now a candidate to turn into a genuine piece of infrastructure for the entire research community. Support will be sought to expand the system and make it more widely available. A number of possible vehicles exist to do this and we expect to create several promising alternatives for evaluation. Assuming the financial basis for developing such a system can be made available,

NRI is prepared to assist in building it. If not, the technology base will be available for the sponsors to pursue the concept by independent sales of equipment for the individual user.

Network Working Group
Request for Comments: 1510

J. Kohl
Digital Equipment Corporation
C. Neuman
ISI
September 1993

The Kerberos Network Authentication Service (V5)

Status of this Memo

This RFC specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document gives an overview and specification of Version 5 of the protocol for the Kerberos network authentication system. Version 4, described elsewhere [1,2], is presently in production use at MIT's Project Athena, and at other Internet sites.

Overview

Project Athena, Athena, Athena MUSE, Discuss, Hesiod, Kerberos, Moira, and Zephyr are trademarks of the Massachusetts Institute of Technology (MIT). No commercial use of these trademarks may be made without prior written permission of MIT.

This RFC describes the concepts and model upon which the Kerberos network authentication system is based. It also specifies Version 5 of the Kerberos protocol.

The motivations, goals, assumptions, and rationale behind most design decisions are treated cursorily; for Version 4 they are fully described in the Kerberos portion of the Athena Technical Plan [1]. The protocols are under review, and are not being submitted for consideration as an Internet standard at this time. Comments are encouraged. Requests for addition to an electronic mailing list for discussion of Kerberos, kerberos@MIT.EDU, may be addressed to kerberos-request@MIT.EDU. This mailing list is gatewayed onto the Usenet as the group comp.protocols.kerberos. Requests for further information, including documents and code availability, may be sent to info-kerberos@MIT.EDU.

Background

The Kerberos model is based in part on Needham and Schroeder's trusted third-party authentication protocol [3] and on modifications suggested by Denning and Sacco [4]. The original design and implementation of Kerberos Versions 1 through 4 was the work of two former Project Athena staff members, Steve Miller of Digital Equipment Corporation and Clifford Neuman (now at the Information Sciences Institute of the University of Southern California), along with Jerome Saltzer, Technical Director of Project Athena, and Jeffrey Schiller, MIT Campus Network Manager. Many other members of Project Athena have also contributed to the work on Kerberos. Version 4 is publicly available, and has seen wide use across the Internet.

Version 5 (described in this document) has evolved from Version 4 based on new requirements and desires for features not available in Version 4. Details on the differences between Kerberos Versions 4 and 5 can be found in [5].

Table of Contents

1. Introduction

Kerberos provides a means of verifying the identities of principals, (e.g., a workstation user or a network server) on an open (unprotected) network. This is accomplished without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts on the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will. (Note, however, that many applications use Kerberos' functions only upon the initiation of a stream-based network connection, and assume the absence of any "hijackers" who might subvert such a connection. Such use implicitly trusts the host addresses involved.) Kerberos performs authentication under these conditions as a trusted third-party authentication service by using conventional cryptography, i.e., shared secret key. (shared secret key - Secret and private are often used interchangeably in the literature. In our usage, it takes two (or more) to share a secret, thus a shared DES key is a secret key. Something is only private when no one but its owner knows it. Thus, in public key cryptosystems, one has a public and a private key.)

The authentication process proceeds as follows: A client sends a request to the authentication server (AS) requesting "credentials" for a given server. The AS responds with these credentials, encrypted in the client's key. The credentials consist of 1) a "ticket" for the server and 2) a temporary encryption key (often called a "session key"). The client transmits the ticket (which contains the client's identity and a copy of the session key, all encrypted in the server's key) to the server. The session key (now shared by the client and server) is used to authenticate the client, and may optionally be used to authenticate the server. It may also be used to encrypt further communication between the two parties or to exchange a separate sub-session key to be used to encrypt further communication.

The implementation consists of one or more authentication servers running on physically secure hosts. The authentication servers maintain a database of principals (i.e., users and servers) and their secret keys. Code libraries provide encryption and implement the Kerberos protocol. In order to add authentication to its

transactions, a typical network application adds one or two calls to the Kerberos library, which results in the transmission of the necessary messages to achieve authentication.

The Kerberos protocol consists of several sub-protocols (or exchanges). There are two methods by which a client can ask a Kerberos server for credentials. In the first approach, the client sends a cleartext request for a ticket for the desired server to the AS. The reply is sent encrypted in the client's secret key. Usually this request is for a ticket-granting ticket (TGT) which can later be used with the ticket-granting server (TGS). In the second method, the client sends a request to the TGS. The client sends the TGT to the TGS in the same manner as if it were contacting any other application server which requires Kerberos credentials. The reply is encrypted in the session key from the TGT.

Once obtained, credentials may be used to verify the identity of the principals in a transaction, to ensure the integrity of messages exchanged between them, or to preserve privacy of the messages. The application is free to choose whatever protection may be necessary.

To verify the identities of the principals in a transaction, the client transmits the ticket to the server. Since the ticket is sent "in the clear" (parts of it are encrypted, but this encryption doesn't thwart replay) and might be intercepted and reused by an attacker, additional information is sent to prove that the message was originated by the principal to whom the ticket was issued. This information (called the authenticator) is encrypted in the session key, and includes a timestamp. The timestamp proves that the message was recently generated and is not a replay. Encrypting the authenticator in the session key proves that it was generated by a party possessing the session key. Since no one except the requesting principal and the server know the session key (it is never sent over the network in the clear) this guarantees the identity of the client.

The integrity of the messages exchanged between principals can also be guaranteed using the session key (passed in the ticket and contained in the credentials). This approach provides detection of both replay attacks and message stream modification attacks. It is accomplished by generating and transmitting a collision-proof checksum (elsewhere called a hash or digest function) of the client's message, keyed with the session key. Privacy and integrity of the messages exchanged between principals can be secured by encrypting the data to be passed using the session key passed in the ticket, and contained in the credentials.

The authentication exchanges mentioned above require read-only access to the Kerberos database. Sometimes, however, the entries in the

database must be modified, such as when adding new principals or changing a principal's key. This is done using a protocol between a client and a third Kerberos server, the Kerberos Administration Server (KADM). The administration protocol is not described in this document. There is also a protocol for maintaining multiple copies of the Kerberos database, but this can be considered an implementation detail and may vary to support different database technologies.

1.1. Cross-Realm Operation

The Kerberos protocol is designed to operate across organizational boundaries. A client in one organization can be authenticated to a server in another. Each organization wishing to run a Kerberos server establishes its own "realm". The name of the realm in which a client is registered is part of the client's name, and can be used by the end-service to decide whether to honor a request.

By establishing "inter-realm" keys, the administrators of two realms can allow a client authenticated in the local realm to use its authentication remotely (Of course, with appropriate permission the client could arrange registration of a separately-named principal in a remote realm, and engage in normal exchanges with that realm's services. However, for even small numbers of clients this becomes cumbersome, and more automatic methods as described here are necessary). The exchange of inter-realm keys (a separate key may be used for each direction) registers the ticket-granting service of each realm as a principal in the other realm. A client is then able to obtain a ticket-granting ticket for the remote realm's ticket-granting service from its local realm. When that ticket-granting ticket is used, the remote ticket-granting service uses the inter-realm key (which usually differs from its own normal TGS key) to decrypt the ticket-granting ticket, and is thus certain that it was issued by the client's own TGS. Tickets issued by the remote ticket-granting service will indicate to the end-service that the client was authenticated from another realm.

A realm is said to communicate with another realm if the two realms share an inter-realm key, or if the local realm shares an inter-realm key with an intermediate realm that communicates with the remote realm. An authentication path is the sequence of intermediate realms that are transited in communicating from one realm to another.

Realms are typically organized hierarchically. Each realm shares a key with its parent and a different key with each child. If an inter-realm key is not directly shared by two realms, the hierarchical organization allows an authentication path to be easily constructed. If a hierarchical organization is not used, it may be necessary to consult some database in order to construct an

authentication path between realms.

Although realms are typically hierarchical, intermediate realms may be bypassed to achieve cross-realm authentication through alternate authentication paths (these might be established to make communication between two realms more efficient). It is important for the end-service to know which realms were transited when deciding how much faith to place in the authentication process. To facilitate this decision, a field in each ticket contains the names of the realms that were involved in authenticating the client.

1.2. Environmental assumptions

Kerberos imposes a few assumptions on the environment in which it can properly function:

+ "Denial of service" attacks are not solved with Kerberos. There are places in these protocols where an intruder intruder can prevent an application from participating in the proper authentication steps. Detection and solution of such attacks (some of which can appear to be not-uncommon "normal" failure modes for the system) is usually best left to the human administrators and users.

+ Principals must keep their secret keys secret. If an intruder somehow steals a principal's key, it will be able to masquerade as that principal or impersonate any server to the legitimate principal.

+ "Password guessing" attacks are not solved by Kerberos. If a user chooses a poor password, it is possible for an attacker to successfully mount an offline dictionary attack by repeatedly attempting to decrypt, with successive entries from a dictionary, messages obtained which are encrypted under a key derived from the user's password.

+ Each host on the network must have a clock which is "loosely synchronized" to the time of the other hosts; this synchronization is used to reduce the bookkeeping needs of application servers when they do replay detection. The degree of "looseness" can be configured on a per-server basis. If the clocks are synchronized over the network, the clock synchronization protocol must itself be secured from network attackers.

+ Principal identifiers are not recycled on a short-term basis. A typical mode of access control will use access control lists (ACLs) to grant permissions to particular principals. If a

stale ACL entry remains for a deleted principal and the principal identifier is reused, the new principal will inherit rights specified in the stale ACL entry. By not re-using principal identifiers, the danger of inadvertent access is removed.

1.3. Glossary of terms

Below is a list of terms used throughout this document.

Authentication	Verifying the claimed identity of a principal.
Authentication header	A record containing a Ticket and an Authenticator to be presented to a server as part of the authentication process.
Authentication path	A sequence of intermediate realms transited in the authentication process when communicating from one realm to another.
Authenticator	A record containing information that can be shown to have been recently generated using the session key known only by the client and server.
Authorization	The process of determining whether a client may use a service, which objects the client is allowed to access, and the type of access allowed for each.
Capability	A token that grants the bearer permission to access an object or service. In Kerberos, this might be a ticket whose use is restricted by the contents of the authorization data field, but which lists no network addresses, together with the session key necessary to use the ticket.

Ciphertext	The output of an encryption function. Encryption transforms plaintext into ciphertext.
Client	A process that makes use of a network service on behalf of a user. Note that in some cases a Server may itself be a client of some other server (e.g., a print server may be a client of a file server).
Credentials	A ticket plus the secret session key necessary to successfully use that ticket in an authentication exchange.
KDC	Key Distribution Center, a network service that supplies tickets and temporary session keys; or an instance of that service or the host on which it runs. The KDC services both initial ticket and ticket-granting ticket requests. The initial ticket portion is sometimes referred to as the Authentication Server (or service). The ticket-granting ticket portion is sometimes referred to as the ticket-granting server (or service).
Kerberos	Aside from the 3-headed dog guarding Hades, the name given to Project Athena's authentication service, the protocol used by that service, or the code used to implement the authentication service.
Plaintext	The input to an encryption function or the output of a decryption function. Decryption transforms ciphertext into plaintext.
Principal	A uniquely named client or server instance that participates in a network communication.

Principal identifier The name used to uniquely identify each different principal.

Seal — To encipher a record containing several fields in such a way that the fields cannot be individually replaced without either knowledge of the encryption key or leaving evidence of tampering.

Secret key — An encryption key shared by a principal and the KDC, distributed outside the bounds of the system, with a long lifetime. In the case of a human user's principal, the secret key is derived from a password.

Server — A particular Principal which provides a resource to network clients.

Service — A resource provided to network clients; often provided by more than one server (for example, remote file service).

Session key — A temporary encryption key used between two principals, with a lifetime limited to the duration of a single login "session".

Sub-session key — A temporary encryption key used between two principals, selected and exchanged by the principals using the session key, and with a lifetime limited to the duration of a single association.

Ticket — A record that helps a client authenticate itself to a server; it contains the client's identity, a session key, a timestamp, and other information, all sealed using the server's secret key. It only serves to authenticate a client when presented along with a fresh Authenticator.

2. Ticket flag uses and requests

Each Kerberos ticket contains a set of flags which are used to indicate various attributes of that ticket. Most flags may be requested by a client when the ticket is obtained; some are automatically turned on and off by a Kerberos server as required. The following sections explain what the various flags mean, and gives examples of reasons to use such a flag.

2.1. Initial and pre-authenticated tickets

The INITIAL flag indicates that a ticket was issued using the AS protocol and not issued based on a ticket-granting ticket. Application servers that want to require the knowledge of a client's secret key (e.g., a passwordchanging program) can insist that this flag be set in any tickets they accept, and thus be assured that the client's key was recently presented to the application client.

The PRE-AUTHENT and HW-AUTHENT flags provide addition information about the initial authentication, regardless of whether the current ticket was issued directly (in which case INITIAL will also be set) or issued on the basis of a ticket-granting ticket (in which case the INITIAL flag is clear, but the PRE-AUTHENT and HW-AUTHENT flags are carried forward from the ticket-granting ticket).

2.2. Invalid tickets

The INVALID flag indicates that a ticket is invalid. Application servers must reject tickets which have this flag set. A postdated ticket will usually be issued in this form. Invalid tickets must be validated by the KDC before use, by presenting them to the KDC in a TGS request with the VALIDATE option specified. The KDC will only validate tickets after their starttime has passed. The validation is required so that postdated tickets which have been stolen before their starttime can be rendered permanently invalid (through a hot-list mechanism).

2.3. Renewable tickets

Applications may desire to hold tickets which can be valid for long periods of time. However, this can expose their credentials to potential theft for equally long periods, and those stolen credentials would be valid until the expiration time of the ticket(s). Simply using shortlived tickets and obtaining new ones periodically would require the client to have long-term access to its secret key, an even greater risk. Renewable tickets can be used to mitigate the consequences of theft. Renewable tickets have two "expiration times": the first is when the current instance of the

ticket expires, and the second is the latest permissible value for an individual expiration time. An application client must periodically (i.e., before it expires) present a renewable ticket to the KDC, with the RENEW option set in the KDC request. The KDC will issue a new ticket with a new session key and a later expiration time. All other fields of the ticket are left unmodified by the renewal process. When the latest permissible expiration time arrives, the ticket expires permanently. At each renewal, the KDC may consult a hot-list to determine if the ticket had been reported stolen since its last renewal; it will refuse to renew such stolen tickets, and thus the usable lifetime of stolen tickets is reduced.

The RENEWABLE flag in a ticket is normally only interpreted by the ticket-granting service (discussed below in section 3.3). It can usually be ignored by application servers. However, some particularly careful application servers may wish to disallow renewable tickets.

If a renewable ticket is not renewed by its expiration time, the KDC will not renew the ticket. The RENEWABLE flag is reset by default, but a client may request it be set by setting the RENEWABLE option in the KRB_AS_REQ message. If it is set, then the renew-till field in the ticket contains the time after which the ticket may not be renewed.

2.4. Postdated tickets

Applications may occasionally need to obtain tickets for use much later, e.g., a batch submission system would need tickets to be valid at the time the batch job is serviced. However, it is dangerous to hold valid tickets in a batch queue, since they will be on-line longer and more prone to theft. Postdated tickets provide a way to obtain these tickets from the KDC at job submission time, but to leave them "dormant" until they are activated and validated by a further request of the KDC. If a ticket theft were reported in the interim, the KDC would refuse to validate the ticket, and the thief would be foiled.

The MAY-POSTDATE flag in a ticket is normally only interpreted by the ticket-granting service. It can be ignored by application servers. This flag must be set in a ticket-granting ticket in order to issue a postdated ticket based on the presented ticket. It is reset by default; it may be requested by a client by setting the ALLOW-POSTDATE option in the KRB_AS_REQ message. This flag does not allow a client to obtain a postdated ticket-granting ticket; postdated ticket-granting tickets can only by obtained by requesting the postdating in the KRB_AS_REQ message. The life (endtime-starttime) of a postdated ticket will be the remaining life of the ticket-

granting ticket at the time of the request, unless the RENEWABLE option is also set, in which case it can be the full life (endtime-starttime) of the ticket-granting ticket. The KDC may limit how far in the future a ticket may be postdated.

The POSTDATED flag indicates that a ticket has been postdated. The application server can check the authtime field in the ticket to see when the original authentication occurred. Some services may choose to reject postdated tickets, or they may only accept them within a certain period after the original authentication. When the KDC issues a POSTDATED ticket, it will also be marked as INVALID, so that the application client must present the ticket to the KDC to be validated before use.

2.5. Proxiable and proxy tickets

At times it may be necessary for a principal to allow a service to perform an operation on its behalf. The service must be able to take on the identity of the client, but only for a particular purpose. A principal can allow a service to take on the principal's identity for a particular purpose by granting it a proxy.

The PROXIABLE flag in a ticket is normally only interpreted by the ticket-granting service. It can be ignored by application servers. When set, this flag tells the ticket-granting server that it is OK to issue a new ticket (but not a ticket-granting ticket) with a different network address based on this ticket. This flag is set by default.

This flag allows a client to pass a proxy to a server to perform a remote request on its behalf, e.g., a print service client can give the print server a proxy to access the client's files on a particular file server in order to satisfy a print request.

In order to complicate the use of stolen credentials, Kerberos tickets are usually valid from only those network addresses specifically included in the ticket (It is permissible to request or issue tickets with no network addresses specified, but we do not recommend it). For this reason, a client wishing to grant a proxy must request a new ticket valid for the network address of the service to be granted the proxy.

The PROXY flag is set in a ticket by the TGS when it issues a proxy ticket. Application servers may check this flag and require additional authentication from the agent presenting the proxy in order to provide an audit trail.

2.6. Forwardable tickets

Authentication forwarding is an instance of the proxy case where the service is granted complete use of the client's identity. An example where it might be used is when a user logs in to a remote system and wants authentication to work from that system as if the login were local.

The FORWARDABLE flag in a ticket is normally only interpreted by the ticket-granting service. It can be ignored by application servers. The FORWARDABLE flag has an interpretation similar to that of the PROXIABLE flag, except ticket-granting tickets may also be issued with different network addresses. This flag is reset by default, but users may request that it be set by setting the FORWARDABLE option in the AS request when they request their initial ticket-granting ticket.

This flag allows for authentication forwarding without requiring the user to enter a password again. If the flag is not set, then authentication forwarding is not permitted, but the same end result can still be achieved if the user engages in the AS exchange with the requested network addresses and supplies a password.

The FORWARDED flag is set by the TGS when a client presents a ticket with the FORWARDABLE flag set and requests it be set by specifying the FORWARDED KDC option and supplying a set of addresses for the new ticket. It is also set in all tickets issued based on tickets with the FORWARDED flag set. Application servers may wish to process FORWARDED tickets differently than non-FORWARDED tickets.

2.7. Other KDC options

There are two additional options which may be set in a client's request of the KDC. The RENEWABLE-OK option indicates that the client will accept a renewable ticket if a ticket with the requested life cannot otherwise be provided. If a ticket with the requested life cannot be provided, then the KDC may issue a renewable ticket with a renew-till equal to the the requested endtime. The value of the renew-till field may still be adjusted by site-determined limits or limits imposed by the individual principal or server.

The ENC-TKT-IN-SKEY option is honored only by the ticket-granting service. It indicates that the to-be-issued ticket for the end server is to be encrypted in the session key from the additional ticket-granting ticket provided with the request. See section 3.3.3 for specific details.

3. Message Exchanges

The following sections describe the interactions between network clients and servers and the messages involved in those exchanges.

3.1. The Authentication Service Exchange

Summary

Message direction	Message type	Section
1. Client to Kerberos	KRB_AS_REQ	5.4.1
2. Kerberos to client	KRB_AS_REP or KRB_ERROR	5.4.2 5.9.1

The Authentication Service (AS) Exchange between the client and the Kerberos Authentication Server is usually initiated by a client when it wishes to obtain authentication credentials for a given server but currently holds no credentials. The client's secret key is used for encryption and decryption. This exchange is typically used at the initiation of a login session, to obtain credentials for a Ticket-Granting Server, which will subsequently be used to obtain credentials for other servers (see section 3.3) without requiring further use of the client's secret key. This exchange is also used to request credentials for services which must not be mediated through the Ticket-Granting Service, but rather require a principal's secret key, such as the password-changing service. (The password-changing request must not be honored unless the requester can provide the old password (the user's current secret key). Otherwise, it would be possible for someone to walk up to an unattended session and change another user's password.) This exchange does not by itself provide any assurance of the the identity of the user. (To authenticate a user logging on to a local system, the credentials obtained in the AS exchange may first be used in a TGS exchange to obtain credentials for a local server. Those credentials must then be verified by the local server through successful completion of the Client/Server exchange.)

The exchange consists of two messages: KRB_AS_REQ from the client to Kerberos, and KRB_AS_REP or KRB_ERROR in reply. The formats for these messages are described in sections 5.4.1, 5.4.2, and 5.9.1.

In the request, the client sends (in cleartext) its own identity and the identity of the server for which it is requesting credentials. The response, KRB_AS_REP, contains a ticket for the client to present to the server, and a session key that will be shared by the client and the server. The session key and additional information are encrypted in the client's secret key. The KRB_AS_REP message contains information which can be used to detect replays, and to

associate it with the message to which it replies. Various errors can occur; these are indicated by an error response (KRB_ERROR) instead of the KRB_AS_REP response. The error message is not encrypted. The KRB_ERROR message also contains information which can be used to associate it with the message to which it replies. The lack of encryption in the KRB_ERROR message precludes the ability to detect replays or fabrications of such messages.

In the normal case the authentication server does not know whether the client is actually the principal named in the request. It simply sends a reply without knowing or caring whether they are the same. This is acceptable because nobody but the principal whose identity was given in the request will be able to use the reply. Its critical information is encrypted in that principal's key. The initial request supports an optional field that can be used to pass additional information that might be needed for the initial exchange. This field may be used for preauthentication if desired, but the mechanism is not currently specified.

3.1.1. Generation of KRB_AS_REQ message

The client may specify a number of options in the initial request. Among these options are whether preauthentication is to be performed; whether the requested ticket is to be renewable, proxiable, or forwardable; whether it should be postdated or allow postdating of derivative tickets; and whether a renewable ticket will be accepted in lieu of a non-renewable ticket if the requested ticket expiration date cannot be satisfied by a nonrenewable ticket (due to configuration constraints; see section 4). See section A.1 for pseudocode.

The client prepares the KRB_AS_REQ message and sends it to the KDC.

3.1.2. Receipt of KRB_AS_REQ message

If all goes well, processing the KRB_AS_REQ message will result in the creation of a ticket for the client to present to the server. The format for the ticket is described in section 5.3.1. The contents of the ticket are determined as follows.

3.1.3. Generation of KRB_AS_REP message

The authentication server looks up the client and server principals named in the KRB_AS_REQ in its database, extracting their respective keys. If required, the server pre-authenticates the request, and if the pre-authentication check fails, an error message with the code KDC_ERR_PREAUTH_FAILED is returned. If the server cannot accommodate the requested encryption type, an error message with code

KDC_ERR_ETYPE_NOSUPP is returned. Otherwise it generates a "random" session key ("Random" means that, among other things, it should be impossible to guess the next session key based on knowledge of past session keys. This can only be achieved in a pseudo-random number generator if it is based on cryptographic principles. It would be more desirable to use a truly random number generator, such as one based on measurements of random physical phenomena.).

If the requested start time is absent or indicates a time in the past, then the start time of the ticket is set to the authentication server's current time. If it indicates a time in the future, but the POSTDATED option has not been specified, then the error KDC_ERR_CANNOT_POSTDATE is returned. Otherwise the requested start time is checked against the policy of the local realm (the administrator might decide to prohibit certain types or ranges of postdated tickets), and if acceptable, the ticket's start time is set as requested and the INVALID flag is set in the new ticket. The postdated ticket must be validated before use by presenting it to the KDC after the start time has been reached.

The expiration time of the ticket will be set to the minimum of the following:

+The expiration time (endtime) requested in the KRB_AS_REQ message.

+The ticket's start time plus the maximum allowable lifetime associated with the client principal (the authentication server's database includes a maximum ticket lifetime field in each principal's record; see section 4).

+The ticket's start time plus the maximum allowable lifetime associated with the server principal.

+The ticket's start time plus the maximum lifetime set by the policy of the local realm.

If the requested expiration time minus the start time (as determined above) is less than a site-determined minimum lifetime, an error message with code KDC_ERR_NEVER_VALID is returned. If the requested expiration time for the ticket exceeds what was determined as above, and if the "RENEWABLE-OK" option was requested, then the "RENEWABLE" flag is set in the new ticket, and the renew-till value is set as if the "RENEWABLE" option were requested (the field and option names are described fully in section 5.4.1). If the RENEWABLE option has been requested or if the RENEWABLE-OK option has been set and a renewable ticket is to be issued, then the renew-till field is set to the minimum of:

+Its requested value.

+The start time of the ticket plus the minimum of the two maximum renewable lifetimes associated with the principals' database entries.

+The start time of the ticket plus the maximum renewable lifetime set by the policy of the local realm.

The flags field of the new ticket will have the following options set if they have been requested and if the policy of the local realm allows: FORWARDABLE, MAY-POSTDATE, POSTDATED, PROXIABLE, RENEWABLE. If the new ticket is postdated (the start time is in the future), its INVALID flag will also be set.

If all of the above succeed, the server formats a KRB_AS_REP message (see section 5.4.2), copying the addresses in the request into the caddr of the response, placing any required pre-authentication data into the padata of the response, and encrypts the ciphertext part in the client's key using the requested encryption method, and sends it to the client. See section A.2 for pseudocode.

3.1.4. Generation of KRB_ERROR message

Several errors can occur, and the Authentication Server responds by returning an error message, KRB_ERROR, to the client, with the error-code and e-text fields set to appropriate values. The error message contents and details are described in Section 5.9.1.

3.1.5. Receipt of KRB_AS_REP message

If the reply message type is KRB_AS_REP, then the client verifies that the cname and crealm fields in the cleartext portion of the reply match what it requested. If any padata fields are present, they may be used to derive the proper secret key to decrypt the message. The client decrypts the encrypted part of the response using its secret key, verifies that the nonce in the encrypted part matches the nonce it supplied in its request (to detect replays). It also verifies that the sname and srealm in the response match those in the request, and that the host address field is also correct. It then stores the ticket, session key, start and expiration times, and other information for later use. The key-expiration field from the encrypted part of the response may be checked to notify the user of impending key expiration (the client program could then suggest remedial action, such as a password change). See section A.3 for pseudocode.

Proper decryption of the KRB_AS_REP message is not sufficient to

verify the identity of the user; the user and an attacker could cooperate to generate a KRB_AS_REP format message which decrypts properly but is not from the proper KDC. If the host wishes to verify the identity of the user, it must require the user to present application credentials which can be verified using a securely-stored secret key. If those credentials can be verified, then the identity of the user can be assured.

3.1.6. Receipt of KRB_ERROR message

If the reply message type is KRB_ERROR, then the client interprets it as an error and performs whatever application-specific tasks are necessary to recover.

3.2. The Client/Server Authentication Exchange

Summary

Message direction	Message type	Section
Client to Application server	KRB_AP_REQ	5.5.1
[optional] Application server to client	KRB_AP_REP or KRB_ERROR	5.5.2 5.9.1

The client/server authentication (CS) exchange is used by network applications to authenticate the client to the server and vice versa. The client must have already acquired credentials for the server using the AS or TGS exchange.

3.2.1. The KRB_AP_REQ message

The KRB_AP_REQ contains authentication information which should be part of the first message in an authenticated transaction. It contains a ticket, an authenticator, and some additional bookkeeping information (see section 5.5.1 for the exact format). The ticket by itself is insufficient to authenticate a client, since tickets are passed across the network in cleartext(Tickets contain both an encrypted and unencrypted portion, so cleartext here refers to the entire unit, which can be copied from one message and replayed in another without any cryptographic skill.), so the authenticator is used to prevent invalid replay of tickets by proving to the server that the client knows the session key of the ticket and thus is entitled to use it. The KRB_AP_REQ message is referred to elsewhere as the "authentication header."

3.2.2. Generation of a KRB_AP_REQ message

When a client wishes to initiate authentication to a server, it obtains (either through a credentials cache, the AS exchange, or the

TGS exchange) a ticket and session key for the desired service. The client may re-use any tickets it holds until they expire. The client then constructs a new Authenticator from the the system time, its name, and optionally an application specific checksum, an initial sequence number to be used in KRB_SAFE or KRB_PRIV messages, and/or a session subkey to be used in negotiations for a session key unique to this particular session. Authenticators may not be re-used and will be rejected if replayed to a server (Note that this can make applications based on unreliable transports difficult to code correctly, if the transport might deliver duplicated messages. In such cases, a new authenticator must be generated for each retry.). If a sequence number is to be included, it should be randomly chosen so that even after many messages have been exchanged it is not likely to collide with other sequence numbers in use.

The client may indicate a requirement of mutual authentication or the use of a session-key based ticket by setting the appropriate flag(s) in the ap-options field of the message.

The Authenticator is encrypted in the session key and combined with the ticket to form the KRB_AP_REQ message which is then sent to the end server along with any additional application-specific information. See section A.9 for pseudocode.

3.2.3. Receipt of KRB_AP_REQ message

Authentication is based on the server's current time of day (clocks must be loosely synchronized), the authenticator, and the ticket. Several errors are possible. If an error occurs, the server is expected to reply to the client with a KRB_ERROR message. This message may be encapsulated in the application protocol if its "raw" form is not acceptable to the protocol. The format of error messages is described in section 5.9.1.

The algorithm for verifying authentication information is as follows. If the message type is not KRB_AP_REQ, the server returns the KRB_AP_ERR_MSG_TYPE error. If the key version indicated by the Ticket in the KRB_AP_REQ is not one the server can use (e.g., it indicates an old key, and the server no longer possesses a copy of the old key), the KRB_AP_ERR_BADKEYVER error is returned. If the USE-SESSION-KEY flag is set in the ap-options field, it indicates to the server that the ticket is encrypted in the session key from the server's ticket-granting ticket rather than its secret key (This is used for user-to-user authentication as described in [6]). Since it is possible for the server to be registered in multiple realms, with different keys in each, the srealm field in the unencrypted portion of the ticket in the KRB_AP_REQ is used to specify which secret key the server should use to decrypt that ticket. The KRB_AP_ERR_NOKEY

error code is returned if the server doesn't have the proper key to decipher the ticket.

The ticket is decrypted using the version of the server's key specified by the ticket. If the decryption routines detect a modification of the ticket (each encryption system must provide safeguards to detect modified ciphertext; see section 6), the KRB_AP_ERR_BAD_INTEGRITY error is returned (chances are good that different keys were used to encrypt and decrypt).

The authenticator is decrypted using the session key extracted from the decrypted ticket. If decryption shows it to have been modified, the KRB_AP_ERR_BAD_INTEGRITY error is returned. The name and realm of the client from the ticket are compared against the same fields in the authenticator. If they don't match, the KRB_AP_ERR_BADMATCH error is returned (they might not match, for example, if the wrong session key was used to encrypt the authenticator). The addresses in the ticket (if any) are then searched for an address matching the operating-system reported address of the client. If no match is found or the server insists on ticket addresses but none are present in the ticket, the KRB_AP_ERR_BADADDR error is returned.

If the local (server) time and the client time in the authenticator differ by more than the allowable clock skew (e.g., 5 minutes), the KRB_AP_ERR_SKEW error is returned. If the server name, along with the client name, time and microsecond fields from the Authenticator match any recently-seen such tuples, the KRB_AP_ERR_REPEAT error is returned (Note that the rejection here is restricted to authenticators from the same principal to the same server. Other client principals communicating with the same server principal should not be have their authenticators rejected if the time and microsecond fields happen to match some other client's authenticator.). The server must remember any authenticator presented within the allowable clock skew, so that a replay attempt is guaranteed to fail. If a server loses track of any authenticator presented within the allowable clock skew, it must reject all requests until the clock skew interval has passed. This assures that any lost or re-played authenticators will fall outside the allowable clock skew and can no longer be successfully replayed (If this is not done, an attacker could conceivably record the ticket and authenticator sent over the network to a server, then disable the client's host, pose as the disabled host, and replay the ticket and authenticator to subvert the authentication.). If a sequence number is provided in the authenticator, the server saves it for later use in processing KRB_SAFE and/or KRB_PRIV messages. If a subkey is present, the server either saves it for later use or uses it to help generate its own choice for a subkey to be returned in a KRB_AP_REP message.

The server computes the age of the ticket: local (server) time minus the start time inside the Ticket. If the start time is later than the current time by more than the allowable clock skew or if the INVALID flag is set in the ticket, the KRB_AP_ERR_TKT_NYV error is returned. Otherwise, if the current time is later than end time by more than the allowable clock skew, the KRB_AP_ERR_TKT_EXPIRED error is returned.

If all these checks succeed without an error, the server is assured that the client possesses the credentials of the principal named in the ticket and thus, the client has been authenticated to the server. See section A.10 for pseudocode.

3.2.4. Generation of a KRB_AP_REP message

Typically, a client's request will include both the authentication information and its initial request in the same message, and the server need not explicitly reply to the KRB_AP_REQ. However, if mutual authentication (not only authenticating the client to the server, but also the server to the client) is being performed, the KRB_AP_REQ message will have MUTUAL-REQUIRED set in its ap-options field, and a KRB_AP_REP message is required in response. As with the error message, this message may be encapsulated in the application protocol if its "raw" form is not acceptable to the application's protocol. The timestamp and microsecond field used in the reply must be the client's timestamp and microsecond field (as provided in the authenticator). [Note: In the Kerberos version 4 protocol, the timestamp in the reply was the client's timestamp plus one. This is not necessary in version 5 because version 5 messages are formatted in such a way that it is not possible to create the reply by judicious message surgery (even in encrypted form) without knowledge of the appropriate encryption keys.] If a sequence number is to be included, it should be randomly chosen as described above for the authenticator. A subkey may be included if the server desires to negotiate a different subkey. The KRB_AP_REP message is encrypted in the session key extracted from the ticket. See section A.11 for pseudocode.

3.2.5. Receipt of KRB_AP_REP message

If a KRB_AP_REP message is returned, the client uses the session key from the credentials obtained for the server (Note that for encrypting the KRB_AP_REP message, the sub-session key is not used, even if present in the Authenticator.) to decrypt the message, and verifies that the timestamp and microsecond fields match those in the Authenticator it sent to the server. If they match, then the client is assured that the server is genuine. The sequence number and subkey (if present) are retained for later use. See section A.12 for

pseudocode.

3.2.6. Using the encryption key

After the KRB_AP_REQ/KRB_AP_REP exchange has occurred, the client and server share an encryption key which can be used by the application. The "true session key" to be used for KRB_PRIV, KRB_SAFE, or other application-specific uses may be chosen by the application based on the subkeys in the KRB_AP_REP message and the authenticator (Implementations of the protocol may wish to provide routines to choose subkeys based on session keys and random numbers and to orchestrate a negotiated key to be returned in the KRB_AP_REP message.). In some cases, the use of this session key will be implicit in the protocol; in others the method of use must be chosen from a several alternatives. We leave the protocol negotiations of how to use the key (e.g., selecting an encryption or checksum type) to the application programmer; the Kerberos protocol does not constrain the implementation options.

With both the one-way and mutual authentication exchanges, the peers should take care not to send sensitive information to each other without proper assurances. In particular, applications that require privacy or integrity should use the KRB_AP_REP or KRB_ERROR responses from the server to client to assure both client and server of their peer's identity. If an application protocol requires privacy of its messages, it can use the KRB_PRIV message (section 3.5). The KRB_SAFE message (section 3.4) can be used to assure integrity.

3.3. The Ticket-Granting Service (TGS) Exchange

Summary

Message direction	Message type	Section
1. Client to Kerberos	KRB_TGS_REQ	5.4.1
2. Kerberos to client	KRB_TGS_REP or KRB_ERROR	5.4.2 5.9.1

The TGS exchange between a client and the Kerberos Ticket-Granting Server is initiated by a client when it wishes to obtain authentication credentials for a given server (which might be registered in a remote realm), when it wishes to renew or validate an existing ticket, or when it wishes to obtain a proxy ticket. In the first case, the client must already have acquired a ticket for the Ticket-Granting Service using the AS exchange (the ticket-granting ticket is usually obtained when a client initially authenticates to the system, such as when a user logs in). The message format for the TGS exchange is almost identical to that for the AS exchange. The primary difference is that encryption and decryption in the TGS

exchange does not take place under the client's key. Instead, the session key from the ticket-granting ticket or renewable ticket, or sub-session key from an Authenticator is used. As is the case for all application servers, expired tickets are not accepted by the TGS, so once a renewable or ticket-granting ticket expires, the client must use a separate exchange to obtain valid tickets.

The TGS exchange consists of two messages: A request (KRB_TGS_REQ) from the client to the Kerberos Ticket-Granting Server, and a reply (KRB_TGS_REP or KRB_ERROR). The KRB_TGS_REQ message includes information authenticating the client plus a request for credentials. The authentication information consists of the authentication header (KRB_AP_REQ) which includes the client's previously obtained ticket-granting, renewable, or invalid ticket. In the ticket-granting ticket and proxy cases, the request may include one or more of: a list of network addresses, a collection of typed authorization data to be sealed in the ticket for authorization use by the application server, or additional tickets (the use of which are described later). The TGS reply (KRB_TGS_REP) contains the requested credentials, encrypted in the session key from the ticket-granting ticket or renewable ticket, or if present, in the subsession key from the Authenticator (part of the authentication header). The KRB_ERROR message contains an error code and text explaining what went wrong. The KRB_ERROR message is not encrypted. The KRB_TGS_REP message contains information which can be used to detect replays, and to associate it with the message to which it replies. The KRB_ERROR message also contains information which can be used to associate it with the message to which it replies, but the lack of encryption in the KRB_ERROR message precludes the ability to detect replays or fabrications of such messages.

3.3.1. Generation of KRB_TGS_REQ message

Before sending a request to the ticket-granting service, the client must determine in which realm the application server is registered [Note: This can be accomplished in several ways. It might be known beforehand (since the realm is part of the principal identifier), or it might be stored in a nameserver. Presently, however, this information is obtained from a configuration file. If the realm to be used is obtained from a nameserver, there is a danger of being spoofed if the nameservice providing the realm name is not authenticated. This might result in the use of a realm which has been compromised, and would result in an attacker's ability to compromise the authentication of the application server to the client.]. If the client does not already possess a ticket-granting ticket for the appropriate realm, then one must be obtained. This is first attempted by requesting a ticket-granting ticket for the destination realm from the local Kerberos server (using the

KRB_TGS_REQ message recursively). The Kerberos server may return a TGT for the desired realm in which case one can proceed. Alternatively, the Kerberos server may return a TGT for a realm which is "closer" to the desired realm (further along the standard hierarchical path), in which case this step must be repeated with a Kerberos server in the realm specified in the returned TGT. If neither are returned, then the request must be retried with a Kerberos server for a realm higher in the hierarchy. This request will itself require a ticket-granting ticket for the higher realm which must be obtained by recursively applying these directions.

Once the client obtains a ticket-granting ticket for the appropriate realm, it determines which Kerberos servers serve that realm, and contacts one. The list might be obtained through a configuration file or network service; as long as the secret keys exchanged by realms are kept secret, only denial of service results from a false Kerberos server.

As in the AS exchange, the client may specify a number of options in the KRB_TGS_REQ message. The client prepares the KRB_TGS_REQ message, providing an authentication header as an element of the padata field, and including the same fields as used in the KRB_AS_REQ message along with several optional fields: the enc-authorization-data field for application server use and additional tickets required by some options.

In preparing the authentication header, the client can select a sub-session key under which the response from the Kerberos server will be encrypted (If the client selects a sub-session key, care must be taken to ensure the randomness of the selected subsession key. One approach would be to generate a random number and XOR it with the session key from the ticket-granting ticket.). If the sub-session key is not specified, the session key from the ticket-granting ticket will be used. If the enc-authorization-data is present, it must be encrypted in the sub-session key, if present, from the authenticator portion of the authentication header, or if not present in the session key from the ticket-granting ticket.

Once prepared, the message is sent to a Kerberos server for the destination realm. See section A.5 for pseudocode.

3.3.2. Receipt of KRB_TGS_REQ message

The KRB_TGS_REQ message is processed in a manner similar to the KRB_AS_REQ message, but there are many additional checks to be performed. First, the Kerberos server must determine which server the accompanying ticket is for and it must select the appropriate key to decrypt it. For a normal KRB_TGS_REQ message, it will be for the

ticket granting service, and the TGS's key will be used. If the TGT was issued by another realm, then the appropriate inter-realm key must be used. If the accompanying ticket is not a ticket granting ticket for the current realm, but is for an application server in the current realm, the RENEW, VALIDATE, or PROXY options are specified in the request, and the server for which a ticket is requested is the server named in the accompanying ticket, then the KDC will decrypt the ticket in the authentication header using the key of the server for which it was issued. If no ticket can be found in the padata field, the KDC_ERR_PADATA_TYPE_NOSUPP error is returned.

Once the accompanying ticket has been decrypted, the user-supplied checksum in the Authenticator must be verified against the contents of the request, and the message rejected if the checksums do not match (with an error code of KRB_AP_ERR_MODIFIED) or if the checksum is not keyed or not collision-proof (with an error code of KRB_AP_ERR_INAPP_CKSUM). If the checksum type is not supported, the KDC_ERR_SUMTYPE_NOSUPP error is returned. If the authorization-data are present, they are decrypted using the sub-session key from the Authenticator.

If any of the decryptions indicate failed integrity checks, the KRB_AP_ERR_BAD_INTEGRITY error is returned.

3.3.3. Generation of KRB_TGS_REP message

The KRB_TGS_REP message shares its format with the KRB_AS_REP (KRB_KDC_REP), but with its type field set to KRB_TGS_REP. The detailed specification is in section 5.4.2.

The response will include a ticket for the requested server. The Kerberos database is queried to retrieve the record for the requested server (including the key with which the ticket will be encrypted). If the request is for a ticket granting ticket for a remote realm, and if no key is shared with the requested realm, then the Kerberos server will select the realm "closest" to the requested realm with which it does share a key, and use that realm instead. This is the only case where the response from the KDC will be for a different server than that requested by the client.

By default, the address field, the client's name and realm, the list of transited realms, the time of initial authentication, the expiration time, and the authorization data of the newly-issued ticket will be copied from the ticket-granting ticket (TGT) or renewable ticket. If the transited field needs to be updated, but the transited type is not supported, the KDC_ERR_TRTYPE_NOSUPP error is returned.

If the request specifies an endtime, then the endtime of the new ticket is set to the minimum of (a) that request, (b) the endtime from the TGT, and (c) the starttime of the TGT plus the minimum of the maximum life for the application server and the maximum life for the local realm (the maximum life for the requesting principal was already applied when the TGT was issued). If the new ticket is to be a renewal, then the endtime above is replaced by the minimum of (a) the value of the renew_till field of the ticket and (b) the starttime for the new ticket plus the life (endtimestarttime) of the old ticket.

If the FORWARDED option has been requested, then the resulting ticket will contain the addresses specified by the client. This option will only be honored if the FORWARDABLE flag is set in the TGT. The PROXY option is similar; the resulting ticket will contain the addresses specified by the client. It will be honored only if the PROXIABLE flag in the TGT is set. The PROXY option will not be honored on requests for additional ticket-granting tickets.

If the requested start time is absent or indicates a time in the past, then the start time of the ticket is set to the authentication server's current time. If it indicates a time in the future, but the POSTDATED option has not been specified or the MAY-POSTDATE flag is not set in the TGT, then the error KDC_ERR_CANNOT_POSTDATE is returned. Otherwise, if the ticket-granting ticket has the MAYPOSTDATE flag set, then the resulting ticket will be postdated and the requested starttime is checked against the policy of the local realm. If acceptable, the ticket's start time is set as requested, and the INVALID flag is set. The postdated ticket must be validated before use by presenting it to the KDC after the starttime has been reached. However, in no case may the starttime, endtime, or renew-till time of a newly-issued postdated ticket extend beyond the renew-till time of the ticket-granting ticket.

If the ENC-TKT-IN-SKEY option has been specified and an additional ticket has been included in the request, the KDC will decrypt the additional ticket using the key for the server to which the additional ticket was issued and verify that it is a ticket-granting ticket. If the name of the requested server is missing from the request, the name of the client in the additional ticket will be used. Otherwise the name of the requested server will be compared to the name of the client in the additional ticket and if different, the request will be rejected. If the request succeeds, the session key from the additional ticket will be used to encrypt the new ticket that is issued instead of using the key of the server for which the new ticket will be used (This allows easy implementation of user-to-user authentication [6], which uses ticket-granting ticket session keys in lieu of secret server keys in situations where such secret

keys could be easily compromised.).

If the name of the server in the ticket that is presented to the KDC as part of the authentication header is not that of the ticket-granting server itself, and the server is registered in the realm of the KDC, If the RENEW option is requested, then the KDC will verify that the RENEWABLE flag is set in the ticket and that the renew_till time is still in the future. If the VALIDATE option is rqeuested, the KDC will check that the starttime has passed and the INVALID flag is set. If the PROXY option is requested, then the KDC will check that the PROXIABLE flag is set in the ticket. If the tests succeed, the KDC will issue the appropriate new ticket.

Whenever a request is made to the ticket-granting server, the presented ticket(s) is(are) checked against a hot-list of tickets which have been canceled. This hot-list might be implemented by storing a range of issue dates for "suspect tickets"; if a presented ticket had an authtime in that range, it would be rejected. In this way, a stolen ticket-granting ticket or renewable ticket cannot be used to gain additional tickets (renewals or otherwise) once the theft has been reported. Any normal ticket obtained before it was reported stolen will still be valid (because they require no interaction with the KDC), but only until their normal expiration time.

The ciphertext part of the response in the KRB_TGS_REP message is encrypted in the sub-session key from the Authenticator, if present, or the session key key from the ticket-granting ticket. It is not encrypted using the client's secret key. Furthermore, the client's key's expiration date and the key version number fields are left out since these values are stored along with the client's database record, and that record is not needed to satisfy a request based on a ticket-granting ticket. See section A.6 for pseudocode.

3.3.3.1. Encoding the transited field

If the identity of the server in the TGT that is presented to the KDC as part of the authentication header is that of the ticket-granting service, but the TGT was issued from another realm, the KDC will look up the inter-realm key shared with that realm and use that key to decrypt the ticket. If the ticket is valid, then the KDC will honor the request, subject to the constraints outlined above in the section describing the AS exchange. The realm part of the client's identity will be taken from the ticket-granting ticket. The name of the realm that issued the ticket-granting ticket will be added to the transited field of the ticket to be issued. This is accomplished by reading the transited field from the ticket-granting ticket (which is treated as an unordered set of realm names), adding the new realm to the set,

then constructing and writing out its encoded (shorthand) form (this may involve a rearrangement of the existing encoding).

Note that the ticket-granting service does not add the name of its own realm. Instead, its responsibility is to add the name of the previous realm. This prevents a malicious Kerberos server from intentionally leaving out its own name (it could, however, omit other realms' names).

The names of neither the local realm nor the principal's realm are to be included in the transited field. They appear elsewhere in the ticket and both are known to have taken part in authenticating the principal. Since the endpoints are not included, both local and single-hop inter-realm authentication result in a transited field that is empty.

Because the name of each realm transited is added to this field, it might potentially be very long. To decrease the length of this field, its contents are encoded. The initially supported encoding is optimized for the normal case of inter-realm communication: a hierarchical arrangement of realms using either domain or X.500 style realm names. This encoding (called DOMAIN-X500-COMPRESS) is now described.

Realm names in the transited field are separated by a ",". The ",", "\", trailing "."s, and leading spaces (" ") are special characters, and if they are part of a realm name, they must be quoted in the transited field by preceding them with a "\".

A realm name ending with a "." is interpreted as being prepended to the previous realm. For example, we can encode traversal of EDU, MIT.EDU, ATHENA.MIT.EDU, WASHINGTON.EDU, and CS.WASHINGTON.EDU as:

"EDU,MIT.,ATHENA.,WASHINGTON.EDU,CS.".

Note that if ATHENA.MIT.EDU, or CS.WASHINGTON.EDU were endpoints, that they would not be included in this field, and we would have:

"EDU,MIT.,WASHINGTON.EDU"

A realm name beginning with a "/" is interpreted as being appended to the previous realm (For the purpose of appending, the realm preceding the first listed realm is considered to be the null realm ("")). If it is to stand by itself, then it should be preceded by a space (" "). For example, we can encode traversal of /COM/HP/APOLLO, /COM/HP, /COM, and /COM/DEC as:

"/COM,/HP,/APOLLO, /COM/DEC".

Like the example above, if /COM/HP/APOLLO and /COM/DEC are endpoints, they they would not be included in this field, and we would have:

"/COM,/HP"

A null subfield preceding or following a "," indicates that all realms between the previous realm and the next realm have been traversed (For the purpose of interpreting null subfields, the client's realm is considered to precede those in the transited field, and the server's realm is considered to follow them.). Thus, "," means that all realms along the path between the client and the server have been traversed. ",EDU, /COM," means that that all realms from the client's realm up to EDU (in a domain style hierarchy) have been traversed, and that everything from /COM down to the server's realm in an X.500 style has also been traversed. This could occur if the EDU realm in one hierarchy shares an inter-realm key directly with the /COM realm in another hierarchy.

3.3.4. Receipt of KRB_TGS_REP message

When the KRB_TGS_REP is received by the client, it is processed in the same manner as the KRB_AS_REP processing described above. The primary difference is that the ciphertext part of the response must be decrypted using the session key from the ticket-granting ticket rather than the client's secret key. See section A.7 for pseudocode.

3.4. The KRB_SAFE Exchange

The KRB_SAFE message may be used by clients requiring the ability to detect modifications of messages they exchange. It achieves this by including a keyed collisionproof checksum of the user data and some control information. The checksum is keyed with an encryption key (usually the last key negotiated via subkeys, or the session key if no negotiation has occured).

3.4.1. Generation of a KRB_SAFE message

When an application wishes to send a KRB_SAFE message, it collects its data and the appropriate control information and computes a checksum over them. The checksum algorithm should be some sort of keyed one-way hash function (such as the RSA-MD5-DES checksum algorithm specified in section 6.4.5, or the DES MAC), generated using the sub-session key if present, or the session key. Different algorithms may be selected by changing the checksum type in the message. Unkeyed or non-collision-proof checksums are not suitable for this use.

The control information for the KRB_SAFE message includes both a

timestamp and a sequence number. The designer of an application using the KRB_SAFE message must choose at least one of the two mechanisms. This choice should be based on the needs of the application protocol.

Sequence numbers are useful when all messages sent will be received by one's peer. Connection state is presently required to maintain the session key, so maintaining the next sequence number should not present an additional problem.

If the application protocol is expected to tolerate lost messages without them being resent, the use of the timestamp is the appropriate replay detection mechanism. Using timestamps is also the appropriate mechanism for multi-cast protocols where all of one's peers share a common sub-session key, but some messages will be sent to a subset of one's peers.

After computing the checksum, the client then transmits the information and checksum to the recipient in the message format specified in section 5.6.1.

3.4.2. Receipt of KRB_SAFE message

When an application receives a KRB_SAFE message, it verifies it as follows. If any error occurs, an error code is reported for use by the application.

The message is first checked by verifying that the protocol version and type fields match the current version and KRB_SAFE, respectively. A mismatch generates a KRB_AP_ERR_BADVERSION or KRB_AP_ERR_MSG_TYPE error. The application verifies that the checksum used is a collisionproof keyed checksum, and if it is not, a KRB_AP_ERR_INAPP_CKSUM error is generated. The recipient verifies that the operating system's report of the sender's address matches the sender's address in the message, and (if a recipient address is specified or the recipient requires an address) that one of the recipient's addresses appears as the recipient's address in the message. A failed match for either case generates a KRB_AP_ERR_BADADDR error. Then the timestamp and usec and/or the sequence number fields are checked. If timestamp and usec are expected and not present, or they are present but not current, the KRB_AP_ERR_SKEW error is generated. If the server name, along with the client name, time and microsecond fields from the Authenticator match any recently-seen such tuples, the KRB_AP_ERR_REPEAT error is generated. If an incorrect sequence number is included, or a sequence number is expected but not present, the KRB_AP_ERR_BADORDER error is generated. If neither a timestamp and usec or a sequence number is present, a KRB_AP_ERR_MODIFIED error is generated.

Finally, the checksum is computed over the data and control information, and if it doesn't match the received checksum, a KRB_AP_ERR_MODIFIED error is generated.

If all the checks succeed, the application is assured that the message was generated by its peer and was not modified in transit.

3.5. The KRB_PRIV Exchange

The KRB_PRIV message may be used by clients requiring confidentiality and the ability to detect modifications of exchanged messages. It achieves this by encrypting the messages and adding control information.

3.5.1. Generation of a KRB_PRIV message

When an application wishes to send a KRB_PRIV message, it collects its data and the appropriate control information (specified in section 5.7.1) and encrypts them under an encryption key (usually the last key negotiated via subkeys, or the session key if no negotiation has occured). As part of the control information, the client must choose to use either a timestamp or a sequence number (or both); see the discussion in section 3.4.1 for guidelines on which to use. After the user data and control information are encrypted, the client transmits the ciphertext and some "envelope" information to the recipient.

3.5.2. Receipt of KRB_PRIV message

When an application receives a KRB_PRIV message, it verifies it as follows. If any error occurs, an error code is reported for use by the application.

The message is first checked by verifying that the protocol version and type fields match the current version and KRB_PRIV, respectively. A mismatch generates a KRB_AP_ERR_BADVERSION or KRB_AP_ERR_MSG_TYPE error. The application then decrypts the ciphertext and processes the resultant plaintext. If decryption shows the data to have been modified, a KRB_AP_ERR_BAD_INTEGRITY error is generated. The recipient verifies that the operating system's report of the sender's address matches the sender's address in the message, and (if a recipient address is specified or the recipient requires an address) that one of the recipient's addresses appears as the recipient's address in the message. A failed match for either case generates a KRB_AP_ERR_BADADDR error. Then the timestamp and usec and/or the sequence number fields are checked. If timestamp and usec are expected and not present, or they are present but not current, the KRB_AP_ERR_SKEW error is generated. If the server name, along with

the client name, time and microsecond fields from the Authenticator match any recently-seen such tuples, the KRB_AP_ERR_REPEAT error is generated. If an incorrect sequence number is included, or a sequence number is expected but not present, the KRB_AP_ERR_BADORDER error is generated. If neither a timestamp and usec or a sequence number is present, a KRB_AP_ERR_MODIFIED error is generated.

If all the checks succeed, the application can assume the message was generated by its peer, and was securely transmitted (without intruders able to see the unencrypted contents).

3.6. The KRB_CRED Exchange

The KRB_CRED message may be used by clients requiring the ability to send Kerberos credentials from one host to another. It achieves this by sending the tickets together with encrypted data containing the session keys and other information associated with the tickets.

3.6.1. Generation of a KRB_CRED message

When an application wishes to send a KRB_CRED message it first (using the KRB_TGS exchange) obtains credentials to be sent to the remote host. It then constructs a KRB_CRED message using the ticket or tickets so obtained, placing the session key needed to use each ticket in the key field of the corresponding KrbCredInfo sequence of the encrypted part of the the KRB_CRED message.

Other information associated with each ticket and obtained during the KRB_TGS exchange is also placed in the corresponding KrbCredInfo sequence in the encrypted part of the KRB_CRED message. The current time and, if specifically required by the application the nonce, s-address, and raddress fields, are placed in the encrypted part of the KRB_CRED message which is then encrypted under an encryption key previosuly exchanged in the KRB_AP exchange (usually the last key negotiated via subkeys, or the session key if no negotiation has occured).

3.6.2. Receipt of KRB_CRED message

When an application receives a KRB_CRED message, it verifies it. If any error occurs, an error code is reported for use by the application. The message is verified by checking that the protocol version and type fields match the current version and KRB_CRED, respectively. A mismatch generates a KRB_AP_ERR_BADVERSION or KRB_AP_ERR_MSG_TYPE error. The application then decrypts the ciphertext and processes the resultant plaintext. If decryption shows the data to have been modified, a KRB_AP_ERR_BAD_INTEGRITY error is generated.

If present or required, the recipient verifies that the operating system's report of the sender's address matches the sender's address in the message, and that one of the recipient's addresses appears as the recipient's address in the message. A failed match for either case generates a KRB_AP_ERR_BADADDR error. The timestamp and usec fields (and the nonce field if required) are checked next. If the timestamp and usec are not present, or they are present but not current, the KRB_AP_ERR_SKEW error is generated.

If all the checks succeed, the application stores each of the new tickets in its ticket cache together with the session key and other information in the corresponding KrbCredInfo sequence from the encrypted part of the KRB_CRED message.

4. The Kerberos Database

The Kerberos server must have access to a database containing the principal identifiers and secret keys of principals to be authenticated (The implementation of the Kerberos server need not combine the database and the server on the same machine; it is feasible to store the principal database in, say, a network name service, as long as the entries stored therein are protected from disclosure to and modification by unauthorized parties. However, we recommend against such strategies, as they can make system management and threat analysis quite complex.).

4.1. Database contents

A database entry should contain at least the following fields:

Field	Value
name	Principal's identifier
key	Principal's secret key
p_kvno	Principal's key version
max_life	Maximum lifetime for Tickets
max_renewable_life	Maximum total lifetime for renewable Tickets

The name field is an encoding of the principal's identifier. The key field contains an encryption key. This key is the principal's secret key. (The key can be encrypted before storage under a Kerberos "master key" to protect it in case the database is compromised but the master key is not. In that case, an extra field must be added to indicate the master key version used, see below.) The p_kvno field is the key version number of the principal's secret key. The max_life field contains the maximum allowable lifetime (endtime - starttime) for any Ticket issued for this principal. The max_renewable_life

field contains the maximum allowable total lifetime for any renewable Ticket issued for this principal. (See section 3.1 for a description of how these lifetimes are used in determining the lifetime of a given Ticket.)

A server may provide KDC service to several realms, as long as the database representation provides a mechanism to distinguish between principal records with identifiers which differ only in the realm name.

When an application server's key changes, if the change is routine (i.e., not the result of disclosure of the old key), the old key should be retained by the server until all tickets that had been issued using that key have expired. Because of this, it is possible for several keys to be active for a single principal. Ciphertext encrypted in a principal's key is always tagged with the version of the key that was used for encryption, to help the recipient find the proper key for decryption.

When more than one key is active for a particular principal, the principal will have more than one record in the Kerberos database. The keys and key version numbers will differ between the records (the rest of the fields may or may not be the same). Whenever Kerberos issues a ticket, or responds to a request for initial authentication, the most recent key (known by the Kerberos server) will be used for encryption. This is the key with the highest key version number.

4.2. Additional fields

Project Athena's KDC implementation uses additional fields in its database:

Field	Value
K_kvno	Kerberos' key version
expiration	Expiration date for entry
attributes	Bit field of attributes
mod_date	Timestamp of last modification
mod_name	Modifying principal's identifier

The K_kvno field indicates the key version of the Kerberos master key under which the principal's secret key is encrypted.

After an entry's expiration date has passed, the KDC will return an error to any client attempting to gain tickets as or for the principal. (A database may want to maintain two expiration dates: one for the principal, and one for the principal's current key. This allows password aging to work independently of the principal's

expiration date. However, due to the limited space in the responses, the KDC must combine the key expiration and principal expiration date into a single value called "key_exp", which is used as a hint to the user to take administrative action.)

The attributes field is a bitfield used to govern the operations involving the principal. This field might be useful in conjunction with user registration procedures, for site-specific policy implementations (Project Athena currently uses it for their user registration process controlled by the system-wide database service, Moira [7]), or to identify the "string to key" conversion algorithm used for a principal's key. (See the discussion of the padata field in section 5.4.2 for details on why this can be useful.) Other bits are used to indicate that certain ticket options should not be allowed in tickets encrypted under a principal's key (one bit each): Disallow issuing postdated tickets, disallow issuing forwardable tickets, disallow issuing tickets based on TGT authentication, disallow issuing renewable tickets, disallow issuing proxiable tickets, and disallow issuing tickets for which the principal is the server.

The mod_date field contains the time of last modification of the entry, and the mod_name field contains the name of the principal which last modified the entry.

4.3. Frequently Changing Fields

Some KDC implementations may wish to maintain the last time that a request was made by a particular principal. Information that might be maintained includes the time of the last request, the time of the last request for a ticket-granting ticket, the time of the last use of a ticket-granting ticket, or other times. This information can then be returned to the user in the last-req field (see section 5.2).

Other frequently changing information that can be maintained is the latest expiration time for any tickets that have been issued using each key. This field would be used to indicate how long old keys must remain valid to allow the continued use of outstanding tickets.

4.4. Site Constants

The KDC implementation should have the following configurable constants or options, to allow an administrator to make and enforce policy decisions:

+ The minimum supported lifetime (used to determine whether the KDC_ERR_NEVER_VALID error should be returned). This constant should reflect reasonable expectations of round-trip time to the

KDC, encryption/decryption time, and processing time by the client and target server, and it should allow for a minimum "useful" lifetime.

+ The maximum allowable total (renewable) lifetime of a ticket (renew_till - starttime).

+ The maximum allowable lifetime of a ticket (endtime - starttime).

+ Whether to allow the issue of tickets with empty address fields (including the ability to specify that such tickets may only be issued if the request specifies some authorization_data).

+ Whether proxiable, forwardable, renewable or post-datable tickets are to be issued.

5. Message Specifications

The following sections describe the exact contents and encoding of protocol messages and objects. The ASN.1 base definitions are presented in the first subsection. The remaining subsections specify the protocol objects (tickets and authenticators) and messages. Specification of encryption and checksum techniques, and the fields related to them, appear in section 6.

5.1. ASN.1 Distinguished Encoding Representation

All uses of ASN.1 in Kerberos shall use the Distinguished Encoding Representation of the data elements as described in the X.509 specification, section 8.7 [8].

5.2. ASN.1 Base Definitions

The following ASN.1 base definitions are used in the rest of this section. Note that since the underscore character (_) is not permitted in ASN.1 names, the hyphen (-) is used in its place for the purposes of ASN.1 names.

```
Realm ::=           GeneralString
PrincipalName ::=   SEQUENCE {
                    name-type[0]      INTEGER,
                    name-string[1]    SEQUENCE OF GeneralString
}
```

Kerberos realms are encoded as GeneralStrings. Realms shall not contain a character with the code 0 (the ASCII NUL). Most realms will usually consist of several components separated by periods (.), in the style of Internet Domain Names, or separated by slashes (/) in

the style of X.500 names. Acceptable forms for realm names are specified in section 7. A PrincipalName is a typed sequence of components consisting of the following sub-fields:

name-type This field specifies the type of name that follows. Pre-defined values for this field are specified in section 7.2. The name-type should be treated as a hint. Ignoring the name type, no two names can be the same (i.e., at least one of the components, or the realm, must be different). This constraint may be eliminated in the future.

name-string This field encodes a sequence of components that form a name, each component encoded as a General String. Taken together, a PrincipalName and a Realm form a principal identifier. Most PrincipalNames will have only a few components (typically one or two).

```
KerberosTime ::=   GeneralizedTime
                   -- Specifying UTC time zone (Z)
```

The timestamps used in Kerberos are encoded as GeneralizedTimes. An encoding shall specify the UTC time zone (Z) and shall not include any fractional portions of the seconds. It further shall not include any separators. Example: The only valid format for UTC time 6 minutes, 27 seconds after 9 pm on 6 November 1985 is 19851106210627Z.

```
HostAddress ::=     SEQUENCE  {
                    addr-type[0]            INTEGER,
                    address[1]              OCTET STRING
}

HostAddresses ::=   SEQUENCE OF SEQUENCE {
                    addr-type[0]            INTEGER,
                    address[1]              OCTET STRING
}
```

The host adddress encodings consists of two fields:

addr-type This field specifies the type of address that follows. Pre-defined values for this field are specified in section 8.1.

address This field encodes a single address of type addr-type.

The two forms differ slightly. HostAddress contains exactly one

address; HostAddresses contains a sequence of possibly many addresses.

```
AuthorizationData ::=   SEQUENCE OF SEQUENCE {
                        ad-type[0]              INTEGER,
                        ad-data[1]              OCTET STRING
}
```

ad-data This field contains authorization data to be interpreted according to the value of the corresponding ad-type field.

ad-type This field specifies the format for the ad-data subfield. All negative values are reserved for local use. Non-negative values are reserved for registered use.

```
APOptions ::=   BIT STRING {
                reserved(0),
                use-session-key(1),
                mutual-required(2)
}

TicketFlags ::=   BIT STRING {
                  reserved(0),
                  forwardable(1),
                  forwarded(2),
                  proxiable(3),
                  proxy(4),
                  may-postdate(5),
                  postdated(6),
                  invalid(7),
                  renewable(8),
                  initial(9),
                  pre-authent(10),
                  hw-authent(11)
}

KDCOptions ::=   BIT STRING {
                 reserved(0),
                 forwardable(1),
                 forwarded(2),
                 proxiable(3),
                 proxy(4),
                 allow-postdate(5),
                 postdated(6),
```

```
                                unused7(7),
                                renewable(8),
                                unused9(9),
                                unused10(10),
                                unused11(11),
                                renewable-ok(27),
                                enc-tkt-in-skey(28),
                                renew(30),
                                validate(31)
                        }

        LastReq ::=     SEQUENCE OF SEQUENCE {
                        lr-type[0]              INTEGER,
                        lr-value[1]             KerberosTime
        }
```

lr-type This field indicates how the following lr-value field is to be interpreted. Negative values indicate that the information pertains only to the responding server. Non-negative values pertain to all servers for the realm.

If the lr-type field is zero (0), then no information is conveyed by the lr-value subfield. If the absolute value of the lr-type field is one (1), then the lr-value subfield is the time of last initial request for a TGT. If it is two (2), then the lr-value subfield is the time of last initial request. If it is three (3), then the lr-value subfield is the time of issue for the newest ticket-granting ticket used. If it is four (4), then the lr-value subfield is the time of the last renewal. If it is five (5), then the lr-value subfield is the time of last request (of any type).

lr-value This field contains the time of the last request. The time must be interpreted according to the contents of the accompanying lr-type subfield.

See section 6 for the definitions of Checksum, ChecksumType, EncryptedData, EncryptionKey, EncryptionType, and KeyType.

5.3. Tickets and Authenticators

This section describes the format and encryption parameters for tickets and authenticators. When a ticket or authenticator is included in a protocol message it is treated as an opaque object.

5.3.1. Tickets

A ticket is a record that helps a client authenticate to a service. A Ticket contains the following information:

```
Ticket ::=                  [APPLICATION 1] SEQUENCE {
                            tkt-vno[0]                  INTEGER,
                            realm[1]                    Realm,
                            sname[2]                    PrincipalName,
                            enc-part[3]                 EncryptedData
}
-- Encrypted part of ticket
EncTicketPart ::=     [APPLICATION 3] SEQUENCE {
                      flags[0]              TicketFlags,
                      key[1]                EncryptionKey,
                      crealm[2]             Realm,
                      cname[3]              PrincipalName,
                      transited[4]          TransitedEncoding,
                      authtime[5]           KerberosTime,
                      starttime[6]          KerberosTime OPTIONAL,
                      endtime[7]            KerberosTime,
                      renew-till[8]         KerberosTime OPTIONAL,
                      caddr[9]              HostAddresses OPTIONAL,
                      authorization-data[10]   AuthorizationData OPTIONAL
}
-- encoded Transited field
TransitedEncoding ::=       SEQUENCE {
                            tr-type[0]  INTEGER, -- must be registered
                            contents[1]          OCTET STRING
}
```

The encoding of EncTicketPart is encrypted in the key shared by Kerberos and the end server (the server's secret key). See section 6 for the format of the ciphertext.

tkt-vno This field specifies the version number for the ticket format. This document describes version number 5.

realm This field specifies the realm that issued a ticket. It also serves to identify the realm part of the server's principal identifier. Since a Kerberos server can only issue tickets for servers within its realm, the two will

always be identical.

sname This field specifies the name part of the server's identity.

enc-part This field holds the encrypted encoding of the EncTicketPart sequence.

flags This field indicates which of various options were used or requested when the ticket was issued. It is a bit-field, where the selected options are indicated by the bit being set (1), and the unselected options and reserved fields being reset (0). Bit 0 is the most significant bit. The encoding of the bits is specified in section 5.2. The flags are described in more detail above in section 2. The meanings of the flags are:

Bit(s)	Name	Description
0	RESERVED	Reserved for future expansion of this field.
1	FORWARDABLE	The FORWARDABLE flag is normally only interpreted by the TGS, and can be ignored by end servers. When set, this flag tells the ticket-granting server that it is OK to issue a new ticket- granting ticket with a different network address based on the presented ticket.
2	FORWARDED	When set, this flag indicates that the ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket.
3	PROXIABLE	The PROXIABLE flag is normally only interpreted by the TGS, and can be ignored by end servers. The PROXIABLE flag has an interpretation identical to that of the FORWARDABLE flag, except that the PROXIABLE flag tells the ticket-granting server that only non- ticket-granting tickets may be issued with different network addresses.

4	PROXY	When set, this flag indicates that a ticket is a proxy.
5	MAY-POSTDATE	The MAY-POSTDATE flag is normally only interpreted by the TGS, and can be ignored by end servers. This flag tells the ticket-granting server that a post- dated ticket may be issued based on this ticket-granting ticket.
6	POSTDATED	This flag indicates that this ticket has been postdated. The end-service can check the authtime field to see when the original authentication occurred.
7	INVALID	This flag indicates that a ticket is invalid, and it must be validated by the KDC before use. Application servers must reject tickets which have this flag set.
8	RENEWABLE	The RENEWABLE flag is normally only interpreted by the TGS, and can usually be ignored by end servers (some particularly careful servers may wish to disallow renewable tickets). A renewable ticket can be used to obtain a replacement ticket that expires at a later date.
9	INITIAL	This flag indicates that this ticket was issued using the AS protocol, and not issued based on a ticket-granting ticket.
10	PRE-AUTHENT	This flag indicates that during initial authentication, the client was authenticated by the KDC before a ticket was issued. The strength of the preauthentication method is not indicated, but is acceptable to the KDC.
11	HW-AUTHENT	This flag indicates that the protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named

client. The hardware authentication method is selected by the KDC and the strength of the method is not indicated.

12-31 RESERVED Reserved for future use.

key This field exists in the ticket and the KDC response and is used to pass the session key from Kerberos to the application server and the client. The field's encoding is described in section 6.2.

crealm This field contains the name of the realm in which the client is registered and in which initial authentication took place.

cname This field contains the name part of the client's principal identifier.

transited This field lists the names of the Kerberos realms that took part in authenticating the user to whom this ticket was issued. It does not specify the order in which the realms were transited. See section 3.3.3.1 for details on how this field encodes the traversed realms.

authtime This field indicates the time of initial authentication for the named principal. It is the time of issue for the original ticket on which this ticket is based. It is included in the ticket to provide additional information to the end service, and to provide the necessary information for implementation of a 'hot list' service at the KDC. An end service that is particularly paranoid could refuse to accept tickets for which the initial authentication occurred "too far" in the past.

This field is also returned as part of the response from the KDC. When returned as part of the response to initial authentication (KRB_AS_REP), this is the current time on the Kerberos server (It is NOT recommended that this time value be used to adjust the workstation's clock since the workstation cannot reliably determine that such a KRB_AS_REP actually came from the proper KDC in a timely manner.).

starttime This field in the ticket specifies the time after which the ticket is valid. Together with endtime, this field specifies the life of the ticket. If it is absent from the ticket, its value should be treated as that of the

authtime field.

endtime This field contains the time after which the ticket will not be honored (its expiration time). Note that individual services may place their own limits on the life of a ticket and may reject tickets which have not yet expired. As such, this is really an upper bound on the expiration time for the ticket.

renew-till This field is only present in tickets that have the RENEWABLE flag set in the flags field. It indicates the maximum endtime that may be included in a renewal. It can be thought of as the absolute expiration time for the ticket, including all renewals.

caddr This field in a ticket contains zero (if omitted) or more (if present) host addresses. These are the addresses from which the ticket can be used. If there are no addresses, the ticket can be used from any location. The decision by the KDC to issue or by the end server to accept zero-address tickets is a policy decision and is left to the Kerberos and end-service administrators; they may refuse to issue or accept such tickets. The suggested and default policy, however, is that such tickets will only be issued or accepted when additional information that can be used to restrict the use of the ticket is included in the authorization_data field. Such a ticket is a capability.

Network addresses are included in the ticket to make it harder for an attacker to use stolen credentials. Because the session key is not sent over the network in cleartext, credentials can't be stolen simply by listening to the network; an attacker has to gain access to the session key (perhaps through operating system security breaches or a careless user's unattended session) to make use of stolen tickets.

It is important to note that the network address from which a connection is received cannot be reliably determined. Even if it could be, an attacker who has compromised the client's workstation could use the credentials from there. Including the network addresses only makes it more difficult, not impossible, for an attacker to walk off with stolen credentials and then use them from a "safe" location.

authorization-data The authorization-data field is used to pass authorization data from the principal on whose behalf a ticket was issued to the application service. If no authorization data is included, this field will be left out. The data in this field are specific to the end service. It is expected that the field will contain the names of service specific objects, and the rights to those objects. The format for this field is described in section 5.2. Although Kerberos is not concerned with the format of the contents of the subfields, it does carry type information (ad-type).

By using the authorization_data field, a principal is able to issue a proxy that is valid for a specific purpose. For example, a client wishing to print a file can obtain a file server proxy to be passed to the print server. By specifying the name of the file in the authorization_data field, the file server knows that the print server can only use the client's rights when accessing the particular file to be printed.

It is interesting to note that if one specifies the authorization-data field of a proxy and leaves the host addresses blank, the resulting ticket and session key can be treated as a capability. See [9] for some suggested uses of this field.

The authorization-data field is optional and does not have to be included in a ticket.

5.3.2. Authenticators

An authenticator is a record sent with a ticket to a server to certify the client's knowledge of the encryption key in the ticket, to help the server detect replays, and to help choose a "true session key" to use with the particular session. The encoding is encrypted in the ticket's session key shared by the client and the server:

```
-- Unencrypted authenticator
Authenticator ::=    [APPLICATION 2] SEQUENCE    {
              authenticator-vno[0]          INTEGER,
              crealm[1]                     Realm,
              cname[2]                      PrincipalName,
              cksum[3]                      Checksum OPTIONAL,
              cusec[4]                      INTEGER,
              ctime[5]                      KerberosTime,
              subkey[6]                     EncryptionKey OPTIONAL,
              seq-number[7]                 INTEGER OPTIONAL,
```

```
        authorization-data[8]        AuthorizationData OPTIONAL
              }
```

authenticator-vno This field specifies the version number for the format of the authenticator. This document specifies version 5.

crealm and cname These fields are the same as those described for the ticket in section 5.3.1.

cksum This field contains a checksum of the the application data that accompanies the KRB_AP_REQ.

cusec This field contains the microsecond part of the client's timestamp. Its value (before encryption) ranges from 0 to 999999. It often appears along with ctime. The two fields are used together to specify a reasonably accurate timestamp.

ctime This field contains the current time on the client's host.

subkey This field contains the client's choice for an encryption key which is to be used to protect this specific application session. Unless an application specifies otherwise, if this field is left out the session key from the ticket will be used.

seq-number This optional field includes the initial sequence number to be used by the KRB_PRIV or KRB_SAFE messages when sequence numbers are used to detect replays (It may also be used by application specific messages). When included in the authenticator this field specifies the initial sequence number for messages from the client to the server. When included in the AP-REP message, the initial sequence number is that for messages from the server to the client. When used in KRB_PRIV or KRB_SAFE messages, it is incremented by one after each message is sent.

For sequence numbers to adequately support the detection of replays they should be non-repeating, even across connection boundaries. The initial sequence number should be random and uniformly distributed across the full space of possible sequence numbers, so that it cannot be guessed by an attacker and so that it and the successive sequence numbers do not repeat other sequences.

authorization-data This field is the same as described for the ticket in section 5.3.1. It is optional and will only appear when additional restrictions are to be placed on the use of a ticket, beyond those carried in the ticket itself.

5.4. Specifications for the AS and TGS exchanges

This section specifies the format of the messages used in exchange between the client and the Kerberos server. The format of possible error messages appears in section 5.9.1.

5.4.1. KRB_KDC_REQ definition

The KRB_KDC_REQ message has no type of its own. Instead, its type is one of KRB_AS_REQ or KRB_TGS_REQ depending on whether the request is for an initial ticket or an additional ticket. In either case, the message is sent from the client to the Authentication Server to request credentials for a service.

The message fields are:

```
AS-REQ ::=          [APPLICATION 10] KDC-REQ
TGS-REQ ::=         [APPLICATION 12] KDC-REQ

KDC-REQ ::=         SEQUENCE {
           pvno[1]                  INTEGER,
           msg-type[2]              INTEGER,
           padata[3]                SEQUENCE OF PA-DATA OPTIONAL,
           req-body[4]              KDC-REQ-BODY
}

PA-DATA ::=         SEQUENCE {
           padata-type[1]           INTEGER,
           padata-value[2]          OCTET STRING,
                        -- might be encoded AP-REQ
}

KDC-REQ-BODY ::=    SEQUENCE {
            kdc-options[0]          KDCOptions,
            cname[1]                PrincipalName OPTIONAL,
                        -- Used only in AS-REQ
            realm[2]                Realm, -- Server's realm
                        -- Also client's in AS-REQ
            sname[3]                PrincipalName OPTIONAL,
            from[4]                 KerberosTime OPTIONAL,
            till[5]                 KerberosTime,
            rtime[6]                KerberosTime OPTIONAL,
            nonce[7]                INTEGER,
```

```
        etype[8]              SEQUENCE OF INTEGER, -- EncryptionType,
                        -- in preference order
        addresses[9]          HostAddresses OPTIONAL,
        enc-authorization-data[10]   EncryptedData OPTIONAL,
                        -- Encrypted AuthorizationData encoding
        additional-tickets[11]       SEQUENCE OF Ticket OPTIONAL
}
```

The fields in this message are:

pvno This field is included in each message, and specifies the protocol version number. This document specifies protocol version 5.

msg-type This field indicates the type of a protocol message. It will almost always be the same as the application identifier associated with a message. It is included to make the identifier more readily accessible to the application. For the KDC-REQ message, this type will be KRB_AS_REQ or KRB_TGS_REQ.

padata The padata (pre-authentication data) field contains a of authentication information which may be needed before credentials can be issued or decrypted. In the case of requests for additional tickets (KRB_TGS_REQ), this field will include an element with padata-type of PA-TGS-REQ and data of an authentication header (ticket-granting ticket and authenticator). The checksum in the authenticator (which must be collisionproof) is to be computed over the KDC-REQ-BODY encoding. In most requests for initial authentication (KRB_AS_REQ) and most replies (KDC-REP), the padata field will be left out.

This field may also contain information needed by certain extensions to the Kerberos protocol. For example, it might be used to initially verify the identity of a client before any response is returned. This is accomplished with a padata field with padata-type equal to PA-ENC-TIMESTAMP and padata-value defined as follows:

```
padata-type     ::= PA-ENC-TIMESTAMP
padata-value    ::= EncryptedData -- PA-ENC-TS-ENC

PA-ENC-TS-ENC   ::= SEQUENCE {
        patimestamp[0]              KerberosTime, -- client's time
        pausec[1]                   INTEGER OPTIONAL
}
```

with patimestamp containing the client's time and pausec containing the microseconds which may be omitted if a client will not generate more than one request per second. The ciphertext (padata-value) consists of the PA-ENC-TS-ENC sequence, encrypted using the client's secret key.

The padata field can also contain information needed to help the KDC or the client select the key needed for generating or decrypting the response. This form of the padata is useful for supporting the use of certain "smartcards" with Kerberos. The details of such extensions are beyond the scope of this specification. See [10] for additional uses of this field.

padata-type The padata-type element of the padata field indicates the way that the padata-value element is to be interpreted. Negative values of padata-type are reserved for unregistered use; non-negative values are used for a registered interpretation of the element type.

req-body This field is a placeholder delimiting the extent of the remaining fields. If a checksum is to be calculated over the request, it is calculated over an encoding of the KDC-REQ-BODY sequence which is enclosed within the req-body field.

kdc-options This field appears in the KRB_AS_REQ and KRB_TGS_REQ requests to the KDC and indicates the flags that the client wants set on the tickets as well as other information that is to modify the behavior of the KDC. Where appropriate, the name of an option may be the same as the flag that is set by that option. Although in most case, the bit in the options field will be the same as that in the flags field, this is not guaranteed, so it is not acceptable to simply copy the options field to the flags field. There are various checks that must be made before honoring an option anyway.

The kdc_options field is a bit-field, where the selected options are indicated by the bit being set (1), and the unselected options and reserved fields being reset (0). The encoding of the bits is specified in section 5.2. The options are described in more detail above in section 2. The meanings of the options are:

Bit(s)	Name	Description
0	RESERVED	Reserved for future expansion of this field.
1	FORWARDABLE	The FORWARDABLE option indicates that the ticket to be issued is to have its forwardable flag set. It may only be set on the initial request, or in a subsequent request if the ticket-granting ticket on which it is based is also forwardable.
2	FORWARDED	The FORWARDED option is only specified in a request to the ticket-granting server and will only be honored if the ticket-granting ticket in the request has its FORWARDABLE bit set. This option indicates that this is a request for forwarding. The address(es) of the host from which the resulting ticket is to be valid are included in the addresses field of the request.
3	PROXIABLE	The PROXIABLE option indicates that the ticket to be issued is to have its proxiable flag set. It may only be set on the initial request, or in a subsequent request if the ticket-granting ticket on which it is based is also proxiable.
4	PROXY	The PROXY option indicates that this is a request for a proxy. This option will only be honored if the ticket-granting ticket in the request has its PROXIABLE bit set. The address(es) of the host from which the resulting ticket is to be valid are included in the addresses field of the request.
5	ALLOW-POSTDATE	The ALLOW-POSTDATE option indicates that the ticket to be issued is to have its MAY-POSTDATE flag set. It may only be set on the initial request, or in a subsequent request if

the ticket-granting ticket on which it is based also has its MAY-POSTDATE flag set.

6	POSTDATED	The POSTDATED option indicates that this is a request for a postdated ticket. This option will only be honored if the ticket-granting ticket on which it is based has its MAY-POSTDATE flag set. The resulting ticket will also have its INVALID flag set, and that flag may be reset by a subsequent request to the KDC after the starttime in the ticket has been reached.
7	UNUSED	This option is presently unused.
8	RENEWABLE	The RENEWABLE option indicates that the ticket to be issued is to have its RENEWABLE flag set. It may only be set on the initial request, or when the ticket-granting ticket on which the request is based is also renewable. If this option is requested, then the rtime field in the request contains the desired absolute expiration time for the ticket.
9-26	RESERVED	Reserved for future use.
27	RENEWABLE-OK	The RENEWABLE-OK option indicates that a renewable ticket will be acceptable if a ticket with the requested life cannot otherwise be provided. If a ticket with the requested life cannot be provided, then a renewable ticket may be issued with a renew-till equal to the the requested endtime. The value of the renew-till field may still be limited by local limits, or limits selected by the individual principal or server.
28	ENC-TKT-IN-SKEY	This option is used only by the ticket-granting service. The ENC-TKT-IN-SKEY option indicates that the ticket for the end server is to be

encrypted in the session key from the additional ticket-granting ticket provided.

29	RESERVED	Reserved for future use.
30	RENEW	This option is used only by the ticket-granting service. The RENEW option indicates that the present request is for a renewal. The ticket provided is encrypted in the secret key for the server on which it is valid. This option will only be honored if the ticket to be renewed has its RENEWABLE flag set and if the time in its renew till field has not passed. The ticket to be renewed is passed in the padata field as part of the authentication header.
31	VALIDATE	This option is used only by the ticket-granting service. The VALIDATE option indicates that the request is to validate a postdated ticket. It will only be honored if the ticket presented is postdated, presently has its INVALID flag set, and would be otherwise usable at this time. A ticket cannot be validated before its starttime. The ticket presented for validation is encrypted in the key of the server for which it is valid and is passed in the padata field as part of the authentication header.

cname and sname These fields are the same as those described for the ticket in section 5.3.1. sname may only be absent when the ENC-TKT-IN-SKEY option is specified. If absent, the name of the server is taken from the name of the client in the ticket passed as additional-tickets.

enc-authorization-data The enc-authorization-data, if present (and it can only be present in the TGS_REQ form), is an encoding of the desired authorization-data encrypted under the sub-session key if present in the Authenticator, or alternatively from the session key in the ticket-granting ticket, both from the padata field in the KRB_AP_REQ.

realm This field specifies the realm part of the server's principal identifier. In the AS exchange, this is also the realm part of the client's principal identifier.

from This field is included in the KRB_AS_REQ and KRB_TGS_REQ ticket requests when the requested ticket is to be postdated. It specifies the desired start time for the requested ticket.

till This field contains the expiration date requested by the client in a ticket request.

rtime This field is the requested renew-till time sent from a client to the KDC in a ticket request. It is optional.

nonce This field is part of the KDC request and response. It it intended to hold a random number generated by the client. If the same number is included in the encrypted response from the KDC, it provides evidence that the response is fresh and has not been replayed by an attacker. Nonces must never be re-used. Ideally, it should be gen erated randomly, but if the correct time is known, it may suffice (Note, however, that if the time is used as the nonce, one must make sure that the workstation time is monotonically increasing. If the time is ever reset backwards, there is a small, but finite, probability that a nonce will be reused.).

etype This field specifies the desired encryption algorithm to be used in the response.

addresses This field is included in the initial request for tickets, and optionally included in requests for additional tickets from the ticket-granting server. It specifies the addresses from which the requested ticket is to be valid. Normally it includes the addresses for the client's host. If a proxy is requested, this field will contain other addresses. The contents of this field are usually copied by the KDC into the caddr field of the resulting ticket.

additional-tickets Additional tickets may be optionally included in a request to the ticket-granting server. If the ENC-TKT-IN-SKEY option has been specified, then the session key from the additional ticket will be used in place of the server's key to encrypt the new ticket. If more than one option which requires additional tickets has been specified, then the additional tickets are used in the order specified by the ordering of the options bits (see kdc-options, above).

The application code will be either ten (10) or twelve (12) depending on whether the request is for an initial ticket (AS-REQ) or for an additional ticket (TGS-REQ).

The optional fields (addresses, authorization-data and additional-tickets) are only included if necessary to perform the operation specified in the kdc-options field.

It should be noted that in KRB_TGS_REQ, the protocol version number appears twice and two different message types appear: the KRB_TGS_REQ message contains these fields as does the authentication header (KRB_AP_REQ) that is passed in the padata field.

5.4.2. KRB_KDC_REP definition

The KRB_KDC_REP message format is used for the reply from the KDC for either an initial (AS) request or a subsequent (TGS) request. There is no message type for KRB_KDC_REP. Instead, the type will be either KRB_AS_REP or KRB_TGS_REP. The key used to encrypt the ciphertext part of the reply depends on the message type. For KRB_AS_REP, the ciphertext is encrypted in the client's secret key, and the client's key version number is included in the key version number for the encrypted data. For KRB_TGS_REP, the ciphertext is encrypted in the sub-session key from the Authenticator, or if absent, the session key from the ticket-granting ticket used in the request. In that case, no version number will be present in the EncryptedData sequence.

The KRB_KDC_REP message contains the following fields:

```
AS-REP ::=     [APPLICATION 11] KDC-REP
TGS-REP ::=    [APPLICATION 13] KDC-REP

KDC-REP ::=    SEQUENCE {
               pvno[0]                    INTEGER,
               msg-type[1]                INTEGER,
               padata[2]                  SEQUENCE OF PA-DATA OPTIONAL,
               crealm[3]                  Realm,
               cname[4]                   PrincipalName,
               ticket[5]                  Ticket,
               enc-part[6]                EncryptedData
}

EncASRepPart ::=     [APPLICATION 25[25]] EncKDCRepPart
EncTGSRepPart ::=    [APPLICATION 26] EncKDCRepPart

EncKDCRepPart ::=   SEQUENCE {
            key[0]                        EncryptionKey,
            last-req[1]                   LastReq,
```

```
        nonce[2]                INTEGER,
        key-expiration[3]       KerberosTime OPTIONAL,
        flags[4]                TicketFlags,
        authtime[5]             KerberosTime,
        starttime[6]            KerberosTime OPTIONAL,
        endtime[7]              KerberosTime,
        renew-till[8]           KerberosTime OPTIONAL,
        srealm[9]               Realm,
        sname[10]               PrincipalName,
        caddr[11]               HostAddresses OPTIONAL
}
```

NOTE: In EncASRepPart, the application code in the encrypted part of a message provides an additional check that the message was decrypted properly.

pvno and msg-type These fields are described above in section 5.4.1. msg-type is either KRB_AS_REP or KRB_TGS_REP.

padata This field is described in detail in section 5.4.1. One possible use for this field is to encode an alternate "mix-in" string to be used with a string-to-key algorithm (such as is described in section 6.3.2). This ability is useful to ease transitions if a realm name needs to change (e.g., when a company is acquired); in such a case all existing password-derived entries in the KDC database would be flagged as needing a special mix-in string until the next password change.

crealm, cname, srealm and sname These fields are the same as those described for the ticket in section 5.3.1.

ticket The newly-issued ticket, from section 5.3.1.

enc-part This field is a place holder for the ciphertext and related information that forms the encrypted part of a message. The description of the encrypted part of the message follows each appearance of this field. The encrypted part is encoded as described in section 6.1.

key This field is the same as described for the ticket in section 5.3.1.

last-req This field is returned by the KDC and specifies the time(s) of the last request by a principal. Depending on what information is available, this might be the last time that a request for a ticket-granting ticket was made, or the last time that a request based on a ticket-granting ticket

was successful. It also might cover all servers for a realm, or just the particular server. Some implementations may display this information to the user to aid in discovering unauthorized use of one's identity. It is similar in spirit to the last login time displayed when logging into timesharing systems.

nonce This field is described above in section 5.4.1.

key-expiration The key-expiration field is part of the response from the KDC and specifies the time that the client's secret key is due to expire. The expiration might be the result of password aging or an account expiration. This field will usually be left out of the TGS reply since the response to the TGS request is encrypted in a session key and no client information need be retrieved from the KDC database. It is up to the application client (usually the login program) to take appropriate action (such as notifying the user) if the expira tion time is imminent.

flags, authtime, starttime, endtime, renew-till and caddr These fields are duplicates of those found in the encrypted portion of the attached ticket (see section 5.3.1), provided so the client may verify they match the intended request and to assist in proper ticket caching. If the message is of type KRB_TGS_REP, the caddr field will only be filled in if the request was for a proxy or forwarded ticket, or if the user is substituting a subset of the addresses from the ticket granting ticket. If the client-requested addresses are not present or not used, then the addresses contained in the ticket will be the same as those included in the ticket-granting ticket.

5.5. Client/Server (CS) message specifications

This section specifies the format of the messages used for the authentication of the client to the application server.

5.5.1. KRB_AP_REQ definition

The KRB_AP_REQ message contains the Kerberos protocol version number, the message type KRB_AP_REQ, an options field to indicate any options in use, and the ticket and authenticator themselves. The KRB_AP_REQ message is often referred to as the "authentication header".

```
AP-REQ ::=      [APPLICATION 14] SEQUENCE {
                pvno[0]                         INTEGER,
                msg-type[1]                     INTEGER,
```

```
                ap-options[2]           APOptions,
                ticket[3]               Ticket,
                authenticator[4]        EncryptedData
}

APOptions ::=   BIT STRING {
                reserved(0),
                use-session-key(1),
                mutual-required(2)
}
```

pvno and msg-type These fields are described above in section 5.4.1. msg-type is KRB_AP_REQ.

ap-options This field appears in the application request (KRB_AP_REQ) and affects the way the request is processed. It is a bit-field, where the selected options are indicated by the bit being set (1), and the unselected options and reserved fields being reset (0). The encoding of the bits is specified in section 5.2. The meanings of the options are:

Bit(s)	Name	Description
0	RESERVED	Reserved for future expansion of this field.
1	USE-SESSION-KEY	The USE-SESSION-KEY option indicates that the ticket the client is presenting to a server is encrypted in the session key from the server's ticket-granting ticket. When this option is not specified, the ticket is encrypted in the server's secret key.
2	MUTUAL-REQUIRED	The MUTUAL-REQUIRED option tells the server that the client requires mutual authentication, and that it must respond with a KRB_AP_REP message.
3-31	RESERVED	Reserved for future use.

ticket This field is a ticket authenticating the client to the server.

authenticator This contains the authenticator, which includes the client's choice of a subkey. Its encoding is described in section 5.3.2.

5.5.2. KRB_AP_REP definition

The KRB_AP_REP message contains the Kerberos protocol version number, the message type, and an encrypted timestamp. The message is sent in in response to an application request (KRB_AP_REQ) where the mutual authentication option has been selected in the ap-options field.

```
AP-REP ::=          [APPLICATION 15] SEQUENCE {
            pvno[0]                  INTEGER,
            msg-type[1]              INTEGER,
            enc-part[2]              EncryptedData
}

EncAPRepPart ::=    [APPLICATION 27]     SEQUENCE {
            ctime[0]                 KerberosTime,
            cusec[1]                 INTEGER,
            subkey[2]                EncryptionKey OPTIONAL,
            seq-number[3]            INTEGER OPTIONAL
}
```

NOTE: in EncAPRepPart, the application code in the encrypted part of a message provides an additional check that the message was decrypted properly.

The encoded EncAPRepPart is encrypted in the shared session key of the ticket. The optional subkey field can be used in an application-arranged negotiation to choose a per association session key.

pvno and msg-type These fields are described above in section 5.4.1. msg-type is KRB_AP_REP.

enc-part This field is described above in section 5.4.2.

ctime This field contains the current time on the client's host.

cusec This field contains the microsecond part of the client's timestamp.

subkey This field contains an encryption key which is to be used to protect this specific application session. See section 3.2.6 for specifics on how this field is used to negotiate a key. Unless an application specifies otherwise, if this field is left out, the sub-session key from the authenticator, or if also left out, the session key from the ticket will be used.

5.5.3. Error message reply

If an error occurs while processing the application request, the KRB_ERROR message will be sent in response. See section 5.9.1 for the format of the error message. The cname and crealm fields may be left out if the server cannot determine their appropriate values from the corresponding KRB_AP_REQ message. If the authenticator was decipherable, the ctime and cusec fields will contain the values from it.

5.6. KRB_SAFE message specification

This section specifies the format of a message that can be used by either side (client or server) of an application to send a tamper-proof message to its peer. It presumes that a session key has previously been exchanged (for example, by using the KRB_AP_REQ/KRB_AP_REP messages).

5.6.1. KRB_SAFE definition

The KRB_SAFE message contains user data along with a collision-proof checksum keyed with the session key. The message fields are:

```
KRB-SAFE ::=        [APPLICATION 20] SEQUENCE {
            pvno[0]                 INTEGER,
            msg-type[1]             INTEGER,
            safe-body[2]            KRB-SAFE-BODY,
            cksum[3]                Checksum
}

KRB-SAFE-BODY ::=   SEQUENCE {
            user-data[0]            OCTET STRING,
            timestamp[1]            KerberosTime OPTIONAL,
            usec[2]                 INTEGER OPTIONAL,
            seq-number[3]           INTEGER OPTIONAL,
            s-address[4]            HostAddress,
            r-address[5]            HostAddress OPTIONAL
}
```

pvno and msg-type These fields are described above in section 5.4.1. msg-type is KRB_SAFE.

safe-body This field is a placeholder for the body of the KRB-SAFE message. It is to be encoded separately and then have the checksum computed over it, for use in the cksum field.

cksum This field contains the checksum of the application data. Checksum details are described in section 6.4. The

checksum is computed over the encoding of the KRB-SAFE-BODY sequence.

user-data This field is part of the KRB_SAFE and KRB_PRIV messages and contain the application specific data that is being passed from the sender to the recipient.

timestamp This field is part of the KRB_SAFE and KRB_PRIV messages. Its contents are the current time as known by the sender of the message. By checking the timestamp, the recipient of the message is able to make sure that it was recently generated, and is not a replay.

usec This field is part of the KRB_SAFE and KRB_PRIV headers. It contains the microsecond part of the timestamp.

seq-number This field is described above in section 5.3.2.

s-address This field specifies the address in use by the sender of the message.

r-address This field specifies the address in use by the recipient of the message. It may be omitted for some uses (such as broadcast protocols), but the recipient may arbitrarily reject such messages. This field along with s-address can be used to help detect messages which have been incorrectly or maliciously delivered to the wrong recipient.

5.7. KRB_PRIV message specification

This section specifies the format of a message that can be used by either side (client or server) of an application to securely and privately send a message to its peer. It presumes that a session key has previously been exchanged (for example, by using the KRB_AP_REQ/KRB_AP_REP messages).

5.7.1. KRB_PRIV definition

The KRB_PRIV message contains user data encrypted in the Session Key. The message fields are:

```
KRB-PRIV ::=        [APPLICATION 21] SEQUENCE {
            pvno[0]                  INTEGER,
            msg-type[1]              INTEGER,
            enc-part[3]              EncryptedData
}
```

```
EncKrbPrivPart ::=   [APPLICATION 28] SEQUENCE {
             user-data[0]             OCTET STRING,
             timestamp[1]             KerberosTime OPTIONAL,
             usec[2]                  INTEGER OPTIONAL,
             seq-number[3]            INTEGER OPTIONAL,
             s-address[4]             HostAddress, -- sender's addr
             r-address[5]             HostAddress OPTIONAL
                                           -- recip's addr
}
```

NOTE: In EncKrbPrivPart, the application code in the encrypted part of a message provides an additional check that the message was decrypted properly.

pvno and msg-type These fields are described above in section 5.4.1. msg-type is KRB_PRIV.

enc-part This field holds an encoding of the EncKrbPrivPart sequence encrypted under the session key (If supported by the encryption method in use, an initialization vector may be passed to the encryption procedure, in order to achieve proper cipher chaining. The initialization vector might come from the last block of the ciphertext from the previous KRB_PRIV message, but it is the application's choice whether or not to use such an initialization vector. If left out, the default initialization vector for the encryption algorithm will be used.). This encrypted encoding is used for the enc-part field of the KRB-PRIV message. See section 6 for the format of the ciphertext.

user-data, timestamp, usec, s-address and r-address These fields are described above in section 5.6.1.

seq-number This field is described above in section 5.3.2.

5.8. KRB_CRED message specification

This section specifies the format of a message that can be used to send Kerberos credentials from one principal to another. It is presented here to encourage a common mechanism to be used by applications when forwarding tickets or providing proxies to subordinate servers. It presumes that a session key has already been exchanged perhaps by using the KRB_AP_REQ/KRB_AP_REP messages.

5.8.1. KRB_CRED definition

The KRB_CRED message contains a sequence of tickets to be sent and information needed to use the tickets, including the session key from

each. The information needed to use the tickets is encryped under an encryption key previously exchanged. The message fields are:

```
KRB-CRED        ::= [APPLICATION 22]   SEQUENCE {
                pvno[0]                INTEGER,
                msg-type[1]            INTEGER, -- KRB_CRED
                tickets[2]             SEQUENCE OF Ticket,
                enc-part[3]            EncryptedData
}

EncKrbCredPart  ::= [APPLICATION 29]   SEQUENCE {
                ticket-info[0]         SEQUENCE OF KrbCredInfo,
                nonce[1]               INTEGER OPTIONAL,
                timestamp[2]           KerberosTime OPTIONAL,
                usec[3]                INTEGER OPTIONAL,
                s-address[4]           HostAddress OPTIONAL,
                r-address[5]           HostAddress OPTIONAL
}

KrbCredInfo     ::=                    SEQUENCE {
                key[0]                 EncryptionKey,
                prealm[1]              Realm OPTIONAL,
                pname[2]               PrincipalName OPTIONAL,
                flags[3]               TicketFlags OPTIONAL,
                authtime[4]            KerberosTime OPTIONAL,
                starttime[5]           KerberosTime OPTIONAL,
                endtime[6]             KerberosTime OPTIONAL
                renew-till[7]          KerberosTime OPTIONAL,
                srealm[8]              Realm OPTIONAL,
                sname[9]               PrincipalName OPTIONAL,
                caddr[10]              HostAddresses OPTIONAL
}
```

pvno and msg-type These fields are described above in section 5.4.1. msg-type is KRB_CRED.

tickets

These are the tickets obtained from the KDC specifically for use by the intended recipient. Successive tickets are paired with the corresponding KrbCredInfo sequence from the enc-part of the KRB-CRED message.

enc-part This field holds an encoding of the EncKrbCredPart sequence encrypted under the session key shared between the sender and the intended recipient. This encrypted encoding is used for the enc-part field of the KRB-CRED message. See section 6 for the format of the ciphertext.

nonce If practical, an application may require the inclusion of a nonce generated by the recipient of the message. If the same value is included as the nonce in the message, it provides evidence that the message is fresh and has not been replayed by an attacker. A nonce must never be re-used; it should be generated randomly by the recipient of the message and provided to the sender of the mes sage in an application specific manner.

timestamp and usec These fields specify the time that the KRB-CRED message was generated. The time is used to provide assurance that the message is fresh.

s-address and r-address These fields are described above in section 5.6.1. They are used optionally to provide additional assurance of the integrity of the KRB-CRED message.

key This field exists in the corresponding ticket passed by the KRB-CRED message and is used to pass the session key from the sender to the intended recipient. The field's encoding is described in section 6.2.

The following fields are optional. If present, they can be associated with the credentials in the remote ticket file. If left out, then it is assumed that the recipient of the credentials already knows their value.

prealm and pname The name and realm of the delegated principal identity.

flags, authtime, starttime, endtime, renew-till, srealm, sname, and caddr These fields contain the values of the corresponding fields from the ticket found in the ticket field. Descriptions of the fields are identical to the descriptions in the KDC-REP message.

5.9. Error message specification

This section specifies the format for the KRB_ERROR message. The fields included in the message are intended to return as much information as possible about an error. It is not expected that all the information required by the fields will be available for all types of errors. If the appropriate information is not available when the message is composed, the corresponding field will be left out of the message.

Note that since the KRB_ERROR message is not protected by any encryption, it is quite possible for an intruder to synthesize or

modify such a message. In particular, this means that the client should not use any fields in this message for security-critical purposes, such as setting a system clock or generating a fresh authenticator. The message can be useful, however, for advising a user on the reason for some failure.

5.9.1. KRB_ERROR definition

The KRB_ERROR message consists of the following fields:

```
KRB-ERROR ::=   [APPLICATION 30] SEQUENCE {
                pvno[0]                 INTEGER,
                msg-type[1]             INTEGER,
                ctime[2]                KerberosTime OPTIONAL,
                cusec[3]                INTEGER OPTIONAL,
                stime[4]                KerberosTime,
                susec[5]                INTEGER,
                error-code[6]           INTEGER,
                crealm[7]               Realm OPTIONAL,
                cname[8]                PrincipalName OPTIONAL,
                realm[9]                Realm, -- Correct realm
                sname[10]               PrincipalName, -- Correct name
                e-text[11]              GeneralString OPTIONAL,
                e-data[12]              OCTET STRING OPTIONAL
}
```

pvno and msg-type These fields are described above in section 5.4.1. msg-type is KRB_ERROR.

ctime This field is described above in section 5.4.1.

cusec This field is described above in section 5.5.2.

stime This field contains the current time on the server. It is of type KerberosTime.

susec This field contains the microsecond part of the server's timestamp. Its value ranges from 0 to 999. It appears along with stime. The two fields are used in conjunction to specify a reasonably accurate timestamp.

error-code This field contains the error code returned by Kerberos or the server when a request fails. To interpret the value of this field see the list of error codes in section 8. Implementations are encouraged to provide for national language support in the display of error messages.

crealm, cname, srealm and sname These fields are described above in

section 5.3.1.

e-text This field contains additional text to help explain the error code associated with the failed request (for example, it might include a principal name which was unknown).

e-data This field contains additional data about the error for use by the application to help it recover from or handle the error. If the errorcode is KDC_ERR_PREAUTH_REQUIRED, then the e-data field will contain an encoding of a sequence of padata fields, each corresponding to an acceptable pre-authentication method and optionally containing data for the method:

```
METHOD-DATA ::=    SEQUENCE of PA-DATA
```

If the error-code is KRB_AP_ERR_METHOD, then the e-data field will contain an encoding of the following sequence:

```
METHOD-DATA ::=    SEQUENCE {
                   method-type[0]    INTEGER,
                   method-data[1]    OCTET STRING OPTIONAL
    }
```

method-type will indicate the required alternate method; method-data will contain any required additional information.

6. Encryption and Checksum Specifications

The Kerberos protocols described in this document are designed to use stream encryption ciphers, which can be simulated using commonly available block encryption ciphers, such as the Data Encryption Standard [11], in conjunction with block chaining and checksum methods [12]. Encryption is used to prove the identities of the network entities participating in message exchanges. The Key Distribution Center for each realm is trusted by all principals registered in that realm to store a secret key in confidence. Proof of knowledge of this secret key is used to verify the authenticity of a principal.

The KDC uses the principal's secret key (in the AS exchange) or a shared session key (in the TGS exchange) to encrypt responses to ticket requests; the ability to obtain the secret key or session key implies the knowledge of the appropriate keys and the identity of the KDC. The ability of a principal to decrypt the KDC response and present a Ticket and a properly formed Authenticator (generated with the session key from the KDC response) to a service verifies the identity of the principal; likewise the ability of the service to

extract the session key from the Ticket and prove its knowledge thereof in a response verifies the identity of the service.

The Kerberos protocols generally assume that the encryption used is secure from cryptanalysis; however, in some cases, the order of fields in the encrypted portions of messages are arranged to minimize the effects of poorly chosen keys. It is still important to choose good keys. If keys are derived from user-typed passwords, those passwords need to be well chosen to make brute force attacks more difficult. Poorly chosen keys still make easy targets for intruders.

The following sections specify the encryption and checksum mechanisms currently defined for Kerberos. The encodings, chaining, and padding requirements for each are described. For encryption methods, it is often desirable to place random information (often referred to as a confounder) at the start of the message. The requirements for a confounder are specified with each encryption mechanism.

Some encryption systems use a block-chaining method to improve the the security characteristics of the ciphertext. However, these chaining methods often don't provide an integrity check upon decryption. Such systems (such as DES in CBC mode) must be augmented with a checksum of the plaintext which can be verified at decryption and used to detect any tampering or damage. Such checksums should be good at detecting burst errors in the input. If any damage is detected, the decryption routine is expected to return an error indicating the failure of an integrity check. Each encryption type is expected to provide and verify an appropriate checksum. The specification of each encryption method sets out its checksum requirements.

Finally, where a key is to be derived from a user's password, an algorithm for converting the password to a key of the appropriate type is included. It is desirable for the string to key function to be one-way, and for the mapping to be different in different realms. This is important because users who are registered in more than one realm will often use the same password in each, and it is desirable that an attacker compromising the Kerberos server in one realm not obtain or derive the user's key in another.

For a discussion of the integrity characteristics of the candidate encryption and checksum methods considered for Kerberos, the the reader is referred to [13].

6.1. Encryption Specifications

The following ASN.1 definition describes all encrypted messages. The enc-part field which appears in the unencrypted part of messages in

section 5 is a sequence consisting of an encryption type, an optional key version number, and the ciphertext.

```
EncryptedData ::=   SEQUENCE {
                    etype[0]      INTEGER, -- EncryptionType
                    kvno[1]       INTEGER OPTIONAL,
                    cipher[2]     OCTET STRING -- ciphertext
}
```

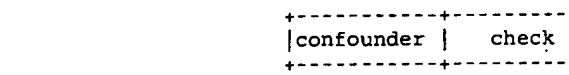
etype This field identifies which encryption algorithm was used to encipher the cipher. Detailed specifications for selected encryption types appear later in this section.

kvno This field contains the version number of the key under which data is encrypted. It is only present in messages encrypted under long lasting keys, such as principals' secret keys.

cipher This field contains the enciphered text, encoded as an OCTET STRING.

The cipher field is generated by applying the specified encryption algorithm to data composed of the message and algorithm-specific inputs. Encryption mechanisms defined for use with Kerberos must take sufficient measures to guarantee the integrity of the plaintext, and we recommend they also take measures to protect against precomputed dictionary attacks. If the encryption algorithm is not itself capable of doing so, the protections can often be enhanced by adding a checksum and a confounder.

The suggested format for the data to be encrypted includes a confounder, a checksum, the encoded plaintext, and any necessary padding. The msg-seq field contains the part of the protocol message described in section 5 which is to be encrypted. The confounder, checksum, and padding are all untagged and untyped, and their length is exactly sufficient to hold the appropriate item. The type and length is implicit and specified by the particular encryption type being used (etype). The format for the data to be encrypted is described in the following diagram:



The format cannot be described in ASN.1, but for those who prefer an ASN.1-like notation:

```
CipherText ::=   ENCRYPTED       SEQUENCE {
        confounder[0]   UNTAGGED OCTET STRING(conf_length)     OPTIONAL,
        check[1]        UNTAGGED OCTET STRING(checksum_length) OPTIONAL,
        msg-seq[2]      MsgSequence,
        pad             UNTAGGED OCTET STRING(pad_length) OPTIONAL
}
```

In the above specification, UNTAGGED OCTET STRING(length) is the notation for an octet string with its tag and length removed. It is not a valid ASN.1 type. The tag bits and length must be removed from the confounder since the purpose of the confounder is so that the message starts with random data, but the tag and its length are fixed. For other fields, the length and tag would be redundant if they were included because they are specified by the encryption type.

One generates a random confounder of the appropriate length, placing it in confounder; zeroes out check; calculates the appropriate checksum over confounder, check, and msg-seq, placing the result in check; adds the necessary padding; then encrypts using the specified encryption type and the appropriate key.

Unless otherwise specified, a definition of an encryption algorithm that specifies a checksum, a length for the confounder field, or an octet boundary for padding uses this ciphertext format (The ordering of the fields in the CipherText is important. Additionally, messages encoded in this format must include a length as part of the msg-seq field. This allows the recipient to verify that the message has not been truncated. Without a length, an attacker could use a chosen plaintext attack to generate a message which could be truncated, while leaving the checksum intact. Note that if the msg-seq is an encoding of an ASN.1 SEQUENCE or OCTET STRING, then the length is part of that encoding.). Those fields which are not specified will be omitted.

In the interest of allowing all implementations using a particular encryption type to communicate with all others using that type, the specification of an encryption type defines any checksum that is needed as part of the encryption process. If an alternative checksum is to be used, a new encryption type must be defined.

Some cryptosystems require additional information beyond the key and the data to be encrypted. For example, DES, when used in cipher-block-chaining mode, requires an initialization vector. If required, the description for each encryption type must specify the source of such additional information.

6.2. Encryption Keys

The sequence below shows the encoding of an encryption key:

```
EncryptionKey ::=   SEQUENCE {
                    keytype[0]     INTEGER,
                    keyvalue[1]    OCTET STRING
}
```

keytype This field specifies the type of encryption key that follows in the keyvalue field. It will almost always correspond to the encryption algorithm used to generate the EncryptedData, though more than one algorithm may use the same type of key (the mapping is many to one). This might happen, for example, if the encryption algorithm uses an alternate checksum algorithm for an integrity check, or a different chaining mechanism.

keyvalue This field contains the key itself, encoded as an octet string.

All negative values for the encryption key type are reserved for local use. All non-negative values are reserved for officially assigned type fields and interpretations.

6.3. Encryption Systems

6.3.1. The NULL Encryption System (null)

If no encryption is in use, the encryption system is said to be the NULL encryption system. In the NULL encryption system there is no checksum, confounder or padding. The ciphertext is simply the plaintext. The NULL Key is used by the null encryption system and is zero octets in length, with keytype zero (0).

6.3.2. DES in CBC mode with a CRC-32 checksum (des-cbc-crc)

The des-cbc-crc encryption mode encrypts information under the Data Encryption Standard [11] using the cipher block chaining mode [12]. A CRC-32 checksum (described in ISO 3309 [14]) is applied to the confounder and message sequence (msg-seq) and placed in the cksum field. DES blocks are 8 bytes. As a result, the data to be encrypted (the concatenation of confounder, checksum, and message) must be padded to an 8 byte boundary before encryption. The details of the encryption of this data are identical to those for the des-cbc-md5 encryption mode.

Note that, since the CRC-32 checksum is not collisionproof, an

attacker could use a probabilistic chosenplaintext attack to generate a valid message even if a confounder is used [13]. The use of collision-proof checksums is recommended for environments where such attacks represent a significant threat. The use of the CRC-32 as the checksum for ticket or authenticator is no longer mandated as an interoperability requirement for Kerberos Version 5 Specification 1 (See section 9.1 for specific details).

6.3.3. DES in CBC mode with an MD4 checksum (des-cbc-md4)

The des-cbc-md4 encryption mode encrypts information under the Data Encryption Standard [11] using the cipher block chaining mode [12]. An MD4 checksum (described in [15]) is applied to the confounder and message sequence (msg-seq) and placed in the cksum field. DES blocks are 8 bytes. As a result, the data to be encrypted (the concatenation of confounder, checksum, and message) must be padded to an 8 byte boundary before encryption. The details of the encryption of this data are identical to those for the descbc-md5 encryption mode.

6.3.4. DES in CBC mode with an MD5 checksum (des-cbc-md5)

The des-cbc-md5 encryption mode encrypts information under the Data Encryption Standard [11] using the cipher block chaining mode [12]. An MD5 checksum (described in [16]) is applied to the confounder and message sequence (msg-seq) and placed in the cksum field. DES blocks are 8 bytes. As a result, the data to be encrypted (the concatenation of confounder, checksum, and message) must be padded to an 8 byte boundary before encryption.

Plaintext and DES ciphtertext are encoded as 8-octet blocks which are concatenated to make the 64-bit inputs for the DES algorithms. The first octet supplies the 8 most significant bits (with the octet's MSbit used as the DES input block's MSbit, etc.), the second octet the next 8 bits, ..., and the eighth octet supplies the 8 least significant bits.

Encryption under DES using cipher block chaining requires an additional input in the form of an initialization vector. Unless otherwise specified, zero should be used as the initialization vector. Kerberos' use of DES requires an 8-octet confounder.

The DES specifications identify some "weak" and "semiweak" keys; those keys shall not be used for encrypting messages for use in Kerberos. Additionally, because of the way that keys are derived for the encryption of checksums, keys shall not be used that yield "weak" or "semi-weak" keys when eXclusive-ORed with the constant F0F0F0F0F0F0F0F0.

A DES key is 8 octets of data, with keytype one (1). This consists of 56 bits of key, and 8 parity bits (one per octet). The key is encoded as a series of 8 octets written in MSB-first order. The bits within the key are also encoded in MSB order. For example, if the encryption key is:
(B1,B2,...,B7,P1,B8,...,B14,P2,B15,...,B49,P7,B50,...,B56,P8) where B1,B2,...,B56 are the key bits in MSB order, and P1,P2,...,P8 are the parity bits, the first octet of the key would be B1,B2,...,B7,P1 (with B1 as the MSbit). [See the FIPS 81 introduction for reference.]

To generate a DES key from a text string (password), the text string normally must have the realm and each component of the principal's name appended(In some cases, it may be necessary to use a different "mix-in" string for compatibility reasons; see the discussion of padata in section 5.4.2.), then padded with ASCII nulls to an 8 byte boundary. This string is then fan-folded and eXclusive-ORed with itself to form an 8 byte DES key. The parity is corrected on the key, and it is used to generate a DES CBC checksum on the initial string (with the realm and name appended). Next, parity is corrected on the CBC checksum. If the result matches a "weak" or "semiweak" key as described in the DES specification, it is eXclusive-ORed with the constant 00000000000000F0. Finally, the result is returned as the key. Pseudocode follows:

```
string_to_key(string,realm,name) {
     odd = 1;
     s = string + realm;
     for(each component in name) {
          s = s + component;
     }
     tempkey = NULL;
     pad(s); /* with nulls to 8 byte boundary */
     for(8byteblock in s) {
          if(odd == 0)  {
              odd = 1;
              reverse(8byteblock)
          }
          else odd = 0;
          tempkey = tempkey XOR 8byteblock;
     }
     fixparity(tempkey);
     key = DES-CBC-check(s,tempkey);
     fixparity(key);
     if(is_weak_key_key(key))
          key = key XOR 0xF0;
     return(key);
}
```

6.4. Checksums

The following is the ASN.1 definition used for a checksum:

```
Checksum ::=   SEQUENCE {
                   cksumtype[0]   INTEGER,
                   checksum[1]    OCTET STRING
               }
```

cksumtype This field indicates the algorithm used to generate the accompanying checksum.

checksum This field contains the checksum itself, encoded as an octet string.

Detailed specification of selected checksum types appear later in this section. Negative values for the checksum type are reserved for local use. All non-negative values are reserved for officially assigned type fields and interpretations.

Checksums used by Kerberos can be classified by two properties: whether they are collision-proof, and whether they are keyed. It is infeasible to find two plaintexts which generate the same checksum value for a collision-proof checksum. A key is required to perturb or initialize the algorithm in a keyed checksum. To prevent message-stream modification by an active attacker, unkeyed checksums should only be used when the checksum and message will be subsequently encrypted (e.g., the checksums defined as part of the encryption algorithms covered earlier in this section). Collision-proof checksums can be made tamper-proof as well if the checksum value is encrypted before inclusion in a message. In such cases, the composition of the checksum and the encryption algorithm must be considered a separate checksum algorithm (e.g., RSA-MD5 encrypted using DES is a new checksum algorithm of type RSA-MD5-DES). For most keyed checksums, as well as for the encrypted forms of collisionproof checksums, Kerberos prepends a confounder before the checksum is calculated.

6.4.1. The CRC-32 Checksum (crc32)

The CRC-32 checksum calculates a checksum based on a cyclic redundancy check as described in ISO 3309 [14]. The resulting checksum is four (4) octets in length. The CRC-32 is neither keyed nor collision-proof. The use of this checksum is not recommended. An attacker using a probabilistic chosen-plaintext attack as described in [13] might be able to generate an alternative message that satisfies the checksum. The use of collision-proof checksums is recommended for environments where such attacks represent a

significant threat.

6.4.2. The RSA MD4 Checksum (rsa-md4)

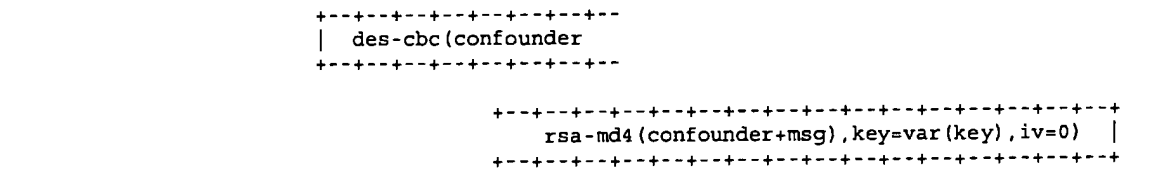
The RSA-MD4 checksum calculates a checksum using the RSA MD4 algorithm [15]. The algorithm takes as input an input message of arbitrary length and produces as output a 128-bit (16 octet) checksum. RSA-MD4 is believed to be collision-proof.

6.4.3. RSA MD4 Cryptographic Checksum Using DES (rsa-md4des)

The RSA-MD4-DES checksum calculates a keyed collisionproof checksum by prepending an 8 octet confounder before the text, applying the RSA MD4 checksum algorithm, and encrypting the confounder and the checksum using DES in cipher-block-chaining (CBC) mode using a variant of the key, where the variant is computed by eXclusive-ORing the key with the constant F0F0F0F0F0F0F0F0 (A variant of the key is used to limit the use of a key to a particular function, separating the functions of generating a checksum from other encryption performed using the session key. The constant F0F0F0F0F0F0F0F0 was chosen because it maintains key parity. The properties of DES precluded the use of the complement. The same constant is used for similar purpose in the Message Integrity Check in the Privacy Enhanced Mail standard.). The initialization vector should be zero. The resulting checksum is 24 octets long (8 octets of which are redundant). This checksum is tamper-proof and believed to be collision-proof.

The DES specifications identify some "weak keys"; those keys shall not be used for generating RSA-MD4 checksums for use in Kerberos.

The format for the checksum is described in the following diagram:



The format cannot be described in ASN.1, but for those who prefer an ASN.1-like notation:

```
rsa-md4-des-checksum ::=   ENCRYPTED        UNTAGGED SEQUENCE {
                           confounder[0]    UNTAGGED OCTET STRING(8),
                           check[1]         UNTAGGED OCTET STRING(16)
}
```

6.4.4. The RSA MD5 Checksum (rsa-md5)

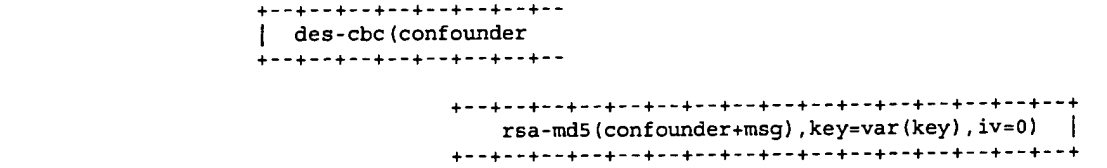
The RSA-MD5 checksum calculates a checksum using the RSA MD5 algorithm [16]. The algorithm takes as input an input message of arbitrary length and produces as output a 128-bit (16 octet) checksum. RSA-MD5 is believed to be collision-proof.

6.4.5. RSA MD5 Cryptographic Checksum Using DES (rsa-md5des)

The RSA-MD5-DES checksum calculates a keyed collisionproof checksum by prepending an 8 octet confounder before the text, applying the RSA MD5 checksum algorithm, and encrypting the confounder and the checksum using DES in cipher-block-chaining (CBC) mode using a variant of the key, where the variant is computed by eXclusive-ORing the key with the constant F0F0F0F0F0F0F0F0. The initialization vector should be zero. The resulting checksum is 24 octets long (8 octets of which are redundant). This checksum is tamper-proof and believed to be collision-proof.

The DES specifications identify some "weak keys"; those keys shall not be used for encrypting RSA-MD5 checksums for use in Kerberos.

The format for the checksum is described in the following diagram:



The format cannot be described in ASN.1, but for those who prefer an ASN.1-like notation:

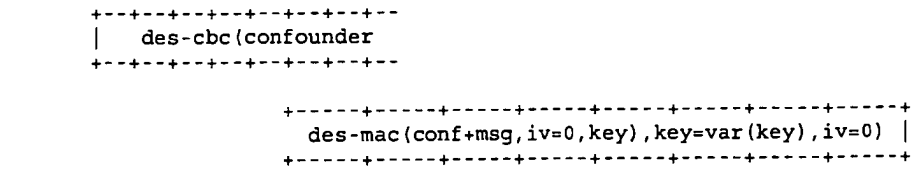
```
rsa-md5-des-checksum ::=   ENCRYPTED       UNTAGGED SEQUENCE {
                           confounder[0]   UNTAGGED OCTET STRING(8),
                           check[1]        UNTAGGED OCTET STRING(16)
}
```

6.4.6. DES cipher-block chained checksum (des-mac)

The DES-MAC checksum is computed by prepending an 8 octet confounder to the plaintext, performing a DES CBC-mode encryption on the result using the key and an initialization vector of zero, taking the last block of the ciphertext, prepending the same confounder and encrypting the pair using DES in cipher-block-chaining (CBC) mode using a a variant of the key, where the variant is computed by

eXclusive-ORing the key with the constant F0F0F0F0F0F0F0F0. The initialization vector should be zero. The resulting checksum is 128 bits (16 octets) long, 64 bits of which are redundant. This checksum is tamper-proof and collision-proof.

The format for the checksum is described in the following diagram:



The format cannot be described in ASN.1, but for those who prefer an ASN.1-like notation:

```
des-mac-checksum ::=    ENCRYPTED       UNTAGGED SEQUENCE {
                        confounder[0]   UNTAGGED OCTET STRING(8),
                        check[1]        UNTAGGED OCTET STRING(8)
}
```

The DES specifications identify some "weak" and "semiweak" keys; those keys shall not be used for generating DES-MAC checksums for use in Kerberos, nor shall a key be used whose veriant is "weak" or "semi-weak".

6.4.7. RSA MD4 Cryptographic Checksum Using DES alternative (rsa-md4-des-k)

The RSA-MD4-DES-K checksum calculates a keyed collision-proof checksum by applying the RSA MD4 checksum algorithm and encrypting the results using DES in cipherblock-chaining (CBC) mode using a DES key as both key and initialization vector. The resulting checksum is 16 octets long. This checksum is tamper-proof and believed to be collision-proof. Note that this checksum type is the old method for encoding the RSA-MD4-DES checksum and it is no longer recommended.

6.4.8. DES cipher-block chained checksum alternative (desmac-k)

The DES-MAC-K checksum is computed by performing a DES CBC-mode encryption of the plaintext, and using the last block of the ciphertext as the checksum value. It is keyed with an encryption key and an initialization vector; any uses which do not specify an additional initialization vector will use the key as both key and initialization vector. The resulting checksum is 64 bits (8 octets) long. This checksum is tamper-proof and collision-proof. Note that

this checksum type is the old method for encoding the DESMAC checksum and it is no longer recommended.

The DES specifications identify some "weak keys"; those keys shall not be used for generating DES-MAC checksums for use in Kerberos.

7. Naming Constraints

7.1. Realm Names

Although realm names are encoded as GeneralStrings and although a realm can technically select any name it chooses, interoperability across realm boundaries requires agreement on how realm names are to be assigned, and what information they imply.

To enforce these conventions, each realm must conform to the conventions itself, and it must require that any realms with which inter-realm keys are shared also conform to the conventions and require the same from its neighbors.

There are presently four styles of realm names: domain, X500, other, and reserved. Examples of each style follow:

```
     domain:   host.subdomain.domain (example)
       X500:   C=US/O=OSF (example)
      other:   NAMETYPE:rest/of.name=without-restrictions (example)
   reserved:   reserved, but will not conflict with above
```

Domain names must look like domain names: they consist of components separated by periods (.) and they contain neither colons (:) nor slashes (/).

X.500 names contain an equal (=) and cannot contain a colon (:) before the equal. The realm names for X.500 names will be string representations of the names with components separated by slashes. Leading and trailing slashes will not be included.

Names that fall into the other category must begin with a prefix that contains no equal (=) or period (.) and the prefix must be followed by a colon (:) and the rest of the name. All prefixes must be assigned before they may be used. Presently none are assigned.

The reserved category includes strings which do not fall into the first three categories. All names in this category are reserved. It is unlikely that names will be assigned to this category unless there is a very strong argument for not using the "other" category.

These rules guarantee that there will be no conflicts between the

various name styles. The following additional constraints apply to the assignment of realm names in the domain and X.500 categories: the name of a realm for the domain or X.500 formats must either be used by the organization owning (to whom it was assigned) an Internet domain name or X.500 name, or in the case that no such names are registered, authority to use a realm name may be derived from the authority of the parent realm. For example, if there is no domain name for E40.MIT.EDU, then the administrator of the MIT.EDU realm can authorize the creation of a realm with that name.

This is acceptable because the organization to which the parent is assigned is presumably the organization authorized to assign names to its children in the X.500 and domain name systems as well. If the parent assigns a realm name without also registering it in the domain name or X.500 hierarchy, it is the parent's responsibility to make sure that there will not in the future exists a name identical to the realm name of the child unless it is assigned to the same entity as the realm name.

7.2. Principal Names

As was the case for realm names, conventions are needed to ensure that all agree on what information is implied by a principal name. The name-type field that is part of the principal name indicates the kind of information implied by the name. The name-type should be treated as a hint. Ignoring the name type, no two names can be the same (i.e., at least one of the components, or the realm, must be different). This constraint may be eliminated in the future. The following name types are defined:

name-type	value	meaning
NT-UNKNOWN	0	Name type not known
NT-PRINCIPAL	1	Just the name of the principal as in DCE, or for users
NT-SRV-INST	2	Service and other unique instance (krbtgt)
NT-SRV-HST	3	Service with host name as instance (telnet, rcommands)
NT-SRV-XHST	4	Service with host as remaining components
NT-UID	5	Unique ID

When a name implies no information other than its uniqueness at a particular time the name type PRINCIPAL should be used. The principal name type should be used for users, and it might also be used for a unique server. If the name is a unique machine generated ID that is guaranteed never to be reassigned then the name type of UID should be used (note that it is generally a bad idea to reassign names of any type since stale entries might remain in access control lists).

If the first component of a name identifies a service and the remaining components identify an instance of the service in a server specified manner, then the name type of SRV-INST should be used. An example of this name type is the Kerberos ticket-granting ticket which has a first component of krbtgt and a second component identifying the realm for which the ticket is valid.

If instance is a single component following the service name and the instance identifies the host on which the server is running, then the name type SRV-HST should be used. This type is typically used for Internet services such as telnet and the Berkeley R commands. If the separate components of the host name appear as successive components following the name of the service, then the name type SRVXHST should be used. This type might be used to identify servers on hosts with X.500 names where the slash (/) might otherwise be ambiguous.

A name type of UNKNOWN should be used when the form of the name is not known. When comparing names, a name of type UNKNOWN will match principals authenticated with names of any type. A principal authenticated with a name of type UNKNOWN, however, will only match other names of type UNKNOWN.

Names of any type with an initial component of "krbtgt" are reserved for the Kerberos ticket granting service. See section 8.2.3 for the form of such names.

7.2.1. Name of server principals

The principal identifier for a server on a host will generally be composed of two parts: (1) the realm of the KDC with which the server is registered, and (2) a two-component name of type NT-SRV-HST if the host name is an Internet domain name or a multi-component name of type NT-SRV-XHST if the name of the host is of a form such as X.500 that allows slash (/) separators. The first component of the two- or multi-component name will identify the service and the latter components will identify the host. Where the name of the host is not case sensitive (for example, with Internet domain names) the name of the host must be lower case. For services such as telnet and the Berkeley R commands which run with system privileges, the first component will be the string "host" instead of a service specific identifier.

8. Constants and other defined values

8.1. Host address types

All negative values for the host address type are reserved for local use. All non-negative values are reserved for officially assigned

type fields and interpretations.

The values of the types for the following addresses are chosen to match the defined address family constants in the Berkeley Standard Distributions of Unix. They can be found in <sys/socket.h> with symbolic names AF_xxx (where xxx is an abbreviation of the address family name).

Internet addresses

Internet addresses are 32-bit (4-octet) quantities, encoded in MSB order. The type of internet addresses is two (2).

CHAOSnet addresses

CHAOSnet addresses are 16-bit (2-octet) quantities, encoded in MSB order. The type of CHAOSnet addresses is five (5).

ISO addresses

ISO addresses are variable-length. The type of ISO addresses is seven (7).

Xerox Network Services (XNS) addresses

XNS addresses are 48-bit (6-octet) quantities, encoded in MSB order. The type of XNS addresses is six (6).

AppleTalk Datagram Delivery Protocol (DDP) addresses

AppleTalk DDP addresses consist of an 8-bit node number and a 16-bit network number. The first octet of the address is the node number; the remaining two octets encode the network number in MSB order. The type of AppleTalk DDP addresses is sixteen (16).

DECnet Phase IV addresses

DECnet Phase IV addresses are 16-bit addresses, encoded in LSB order. The type of DECnet Phase IV addresses is twelve (12).

8.2. KDC messages

8.2.1. IP transport

When contacting a Kerberos server (KDC) for a KRB_KDC_REQ request using IP transport, the client shall send a UDP datagram containing only an encoding of the request to port 88 (decimal) at the KDC's IP

address; the KDC will respond with a reply datagram containing only an encoding of the reply message (either a KRB_ERROR or a KRB_KDC_REP) to the sending port at the sender's IP address.

8.2.2. OSI transport

During authentication of an OSI client to and OSI server, the mutual authentication of an OSI server to an OSI client, the transfer of credentials from an OSI client to an OSI server, or during exchange of private or integrity checked messages, Kerberos protocol messages may be treated as opaque objects and the type of the authentication mechanism will be:

```
OBJECT IDENTIFIER ::= {iso (1), org(3), dod(5),internet(1),
                       security(5), kerberosv5(2)}
```

Depending on the situation, the opaque object will be an authentication header (KRB_AP_REQ), an authentication reply (KRB_AP_REP), a safe message (KRB_SAFE), a private message (KRB_PRIV), or a credentials message (KRB_CRED). The opaque data contains an application code as specified in the ASN.1 description for each message. The application code may be used by Kerberos to determine the message type.

8.2.3. Name of the TGS

The principal identifier of the ticket-granting service shall be composed of three parts: (1) the realm of the KDC issuing the TGS ticket (2) a two-part name of type NT-SRVINST, with the first part "krbtgt" and the second part the name of the realm which will accept the ticket-granting ticket. For example, a ticket-granting ticket issued by the ATHENA.MIT.EDU realm to be used to get tickets from the ATHENA.MIT.EDU KDC has a principal identifier of "ATHENA.MIT.EDU" (realm), ("krbtgt", "ATHENA.MIT.EDU") (name). A ticket-granting ticket issued by the ATHENA.MIT.EDU realm to be used to get tickets from the MIT.EDU realm has a principal identifier of "ATHENA.MIT.EDU" (realm), ("krbtgt", "MIT.EDU") (name).

8.3. Protocol constants and associated values

The following tables list constants used in the protocol and defines their meanings.

Encryption type	etype value	block size	minimum pad size	confounder size
NULL	0	1	0	0
des-cbc-crc	1	8	4	8
des-cbc-md4	2	8	0	8
des-cbc-md5	3	8	0	8

Checksum type	sumtype value	checksum size
CRC32	1	4
rsa-md4	2	16
rsa-md4-des	3	24
des-mac	4	16
des-mac-k	5	8
rsa-md4-des-k	6	16
rsa-md5	7	16
rsa-md5-des	8	24

padata type	padata-type value
PA-TGS-REQ	1
PA-ENC-TIMESTAMP	2
PA-PW-SALT	3

authorization data type	ad-type value
reserved values	0-63
OSF-DCE	64
SESAME	65

alternate authentication type	method-type value
reserved values	0-63
ATT-CHALLENGE-RESPONSE	64

transited encoding type	tr-type value
DOMAIN-X500-COMPRESS	1
reserved values	all others

Label	Value	Meaning or MIT code
pvno	5	current Kerberos protocol version number
message types		
KRB_AS_REQ	10	Request for initial authentication
KRB_AS_REP	11	Response to KRB_AS_REQ request
KRB_TGS_REQ	12	Request for authentication based on TGT
KRB_TGS_REP	13	Response to KRB_TGS_REQ request
KRB_AP_REQ	14	application request to server
KRB_AP_REP	15	Response to KRB_AP_REQ_MUTUAL
KRB_SAFE	20	Safe (checksummed) application message
KRB_PRIV	21	Private (encrypted) application message
KRB_CRED	22	Private (encrypted) message to forward credentials
KRB_ERROR	30	Error response
name types		
KRB_NT_UNKNOWN	0	Name type not known
KRB_NT_PRINCIPAL	1	Just the name of the principal as in DCE, or for users
KRB_NT_SRV_INST	2	Service and other unique instance (krbtgt)
KRB_NT_SRV_HST	3	Service with host name as instance (telnet, rcommands)
KRB_NT_SRV_XHST	4	Service with host as remaining components
KRB_NT_UID	5	Unique ID
error codes		
KDC_ERR_NONE	0	No error
KDC_ERR_NAME_EXP	1	Client's entry in database has expired
KDC_ERR_SERVICE_EXP	2	Server's entry in database has expired
KDC_ERR_BAD_PVNO	3	Requested protocol version number not supported
KDC_ERR_C_OLD_MAST_KVNO	4	Client's key encrypted in old master key
KDC_ERR_S_OLD_MAST_KVNO	5	Server's key encrypted in old master key
KDC_ERR_C_PRINCIPAL_UNKNOWN	6	Client not found in Kerberos database
KDC_ERR_S_PRINCIPAL_UNKNOWN	7	Server not found in Kerberos database
KDC_ERR_PRINCIPAL_NOT_UNIQUE	8	Multiple principal entries in database

KDC_ERR_NULL_KEY	9	The client or server has a null key
KDC_ERR_CANNOT_POSTDATE	10	Ticket not eligible for postdating
KDC_ERR_NEVER_VALID	11	Requested start time is later than end time
KDC_ERR_POLICY	12	KDC policy rejects request
KDC_ERR_BADOPTION	13	KDC cannot accommodate requested option
KDC_ERR_ETYPE_NOSUPP	14	KDC has no support for encryption type
KDC_ERR_SUMTYPE_NOSUPP	15	KDC has no support for checksum type
KDC_ERR_PADATA_TYPE_NOSUPP	16	KDC has no support for padata type
KDC_ERR_TRTYPE_NOSUPP	17	KDC has no support for transited type
KDC_ERR_CLIENT_REVOKED	18	Clients credentials have been revoked
KDC_ERR_SERVICE_REVOKED	19	Credentials for server have been revoked
KDC_ERR_TGT_REVOKED	20	TGT has been revoked
KDC_ERR_CLIENT_NOTYET	21	Client not yet valid - try again later
KDC_ERR_SERVICE_NOTYET	22	Server not yet valid - try again later
KDC_ERR_KEY_EXPIRED	23	Password has expired - change password to reset
KDC_ERR_PREAUTH_FAILED	24	Pre-authentication information was invalid
KDC_ERR_PREAUTH_REQUIRED	25	Additional pre-authentication required*
KRB_AP_ERR_BAD_INTEGRITY	31	Integrity check on decrypted field failed
KRB_AP_ERR_TKT_EXPIRED	32	Ticket expired
KRB_AP_ERR_TKT_NYV	33	Ticket not yet valid
KRB_AP_ERR_REPEAT	34	Request is a replay
KRB_AP_ERR_NOT_US	35	The ticket isn't for us
KRB_AP_ERR_BADMATCH	36	Ticket and authenticator don't match
KRB_AP_ERR_SKEW	37	Clock skew too great
KRB_AP_ERR_BADADDR	38	Incorrect net address
KRB_AP_ERR_BADVERSION	39	Protocol version mismatch
KRB_AP_ERR_MSG_TYPE	40	Invalid msg type
KRB_AP_ERR_MODIFIED	41	Message stream modified
KRB_AP_ERR_BADORDER	42	Message out of order
KRB_AP_ERR_BADKEYVER	44	Specified version of key is not available
KRB_AP_ERR_NOKEY	45	Service key not available
KRB_AP_ERR_MUT_FAIL	46	Mutual authentication failed
KRB_AP_ERR_BADDIRECTION	47	Incorrect message direction
KRB_AP_ERR_METHOD	48	Alternative authentication method required*
KRB_AP_ERR_BADSEQ	49	Incorrect sequence number in message
KRB_AP_ERR_INAPP_CKSUM	50	Inappropriate type of checksum in

		message
KRB_ERR_GENERIC	60	Generic error (description in e-text)
KRB_ERR_FIELD_TOOLONG	61	Field is too long for this implementation

*This error carries additional information in the e-data field. The contents of the e-data field for this message is described in section 5.9.1.

9. Interoperability requirements

Version 5 of the Kerberos protocol supports a myriad of options. Among these are multiple encryption and checksum types, alternative encoding schemes for the transited field, optional mechanisms for pre-authentication, the handling of tickets with no addresses, options for mutual authentication, user to user authentication, support for proxies, forwarding, postdating, and renewing tickets, the format of realm names, and the handling of authorization data.

In order to ensure the interoperability of realms, it is necessary to define a minimal configuration which must be supported by all implementations. This minimal configuration is subject to change as technology does. For example, if at some later date it is discovered that one of the required encryption or checksum algorithms is not secure, it will be replaced.

9.1. Specification 1

This section defines the first specification of these options. Implementations which are configured in this way can be said to support Kerberos Version 5 Specification 1 (5.1).

Encryption and checksum methods

The following encryption and checksum mechanisms must be supported. Implementations may support other mechanisms as well, but the additional mechanisms may only be used when communicating with principals known to also support them: Encryption: DES-CBC-MD5 Checksums: CRC-32, DES-MAC, DES-MAC-K, and DES-MD5

Realm Names

All implementations must understand hierarchical realms in both the Internet Domain and the X.500 style. When a ticket granting ticket for an unknown realm is requested, the KDC must be able to determine the names of the intermediate realms between the KDCs realm and the requested realm.

Transited field encoding

DOMAIN-X500-COMPRESS (described in section 3.3.3.1) must be supported. Alternative encodings may be supported, but they may be used only when that encoding is supported by ALL intermediate realms.

Pre-authentication methods

The TGS-REQ method must be supported. The TGS-REQ method is not used on the initial request. The PA-ENC-TIMESTAMP method must be supported by clients but whether it is enabled by default may be determined on a realm by realm basis. If not used in the initial request and the error KDC_ERR_PREAUTH_REQUIRED is returned specifying PA-ENCTIMESTAMP as an acceptable method, the client should retry the initial request using the PA-ENC-TIMESTAMP preauthentication method. Servers need not support the PAENC-TIMESTAMP method, but if not supported the server should ignore the presence of PA-ENC-TIMESTAMP pre-authentication in a request.

Mutual authentication

Mutual authentication (via the KRB_AP_REP message) must be supported.

Ticket addresses and flags

All KDC's must pass on tickets that carry no addresses (i.e., if a TGT contains no addresses, the KDC will return derivative tickets), but each realm may set its own policy for issuing such tickets, and each application server will set its own policy with respect to accepting them. By default, servers should not accept them.

Proxies and forwarded tickets must be supported. Individual realms and application servers can set their own policy on when such tickets will be accepted.

All implementations must recognize renewable and postdated tickets, but need not actually implement them. If these options are not supported, the starttime and endtime in the ticket shall specify a ticket's entire useful life. When a postdated ticket is decoded by a server, all implementations shall make the presence of the postdated flag visible to the calling server.

User-to-user authentication

Support for user to user authentication (via the ENC-TKTIN-SKEY KDC option) must be provided by implementations, but individual realms may decide as a matter of policy to reject such requests on a per-principal or realm-wide basis.

Authorization data

Implementations must pass all authorization data subfields from ticket-granting tickets to any derivative tickets unless directed to suppress a subfield as part of the definition of that registered subfield type (it is never incorrect to pass on a subfield, and no registered subfield types presently specify suppression at the KDC).

Implementations must make the contents of any authorization data subfields available to the server when a ticket is used. Implementations are not required to allow clients to specify the contents of the authorization data fields.

9.2. Recommended KDC values

Following is a list of recommended values for a KDC implementation, based on the list of suggested configuration constants (see section 4.4).

minimum lifetime	5 minutes
maximum renewable lifetime	1 week
maximum ticket lifetime	1 day
empty addresses	only when suitable restrictions appear in authorization data
proxiable, etc.	Allowed.

10. Acknowledgments

Early versions of this document, describing version 4 of the protocol, were written by Jennifer Steiner (formerly at Project Athena); these drafts provided an excellent starting point for this current version 5 specification. Many people in the Internet community have contributed ideas and suggested protocol changes for version 5. Notable contributions came from Ted Anderson, Steve Bellovin and Michael Merritt [17], Daniel Bernstein, Mike Burrows, Donald Davis, Ravi Ganesan, Morrie Gasser, Virgil Gligor, Bill Griffeth, Mark Lillibridge, Mark Lomas, Steve Lunt, Piers McMahon, Joe Pato, William Sommerfeld, Stuart Stubblebine, Ralph Swick, Ted T'so, and Stanley Zanarotti. Many others commented and helped shape this specification into its current form.

11. References

[1] Miller, S., Neuman, C., Schiller, J., and J. Saltzer, "Section E.2.1: Kerberos Authentication and Authorization System", M.I.T. Project Athena, Cambridge, Massachusetts, December 21, 1987.

[2] Steiner, J., Neuman, C., and J. Schiller, "Kerberos: An Authentication Service for Open Network Systems", pp. 191-202 in Usenix Conference Proceedings, Dallas, Texas, February, 1988.

[3] Needham, R., and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM, Vol. 21 (12), pp. 993-999, December 1978.

[4] Denning, D., and G. Sacco, "Time stamps in Key Distribution Protocols", Communications of the ACM, Vol. 24 (8), pp. 533-536, August 1981.

[5] Kohl, J., Neuman, C., and T. Ts'o, "The Evolution of the Kerberos Authentication Service", in an IEEE Computer Society Text soon to be published, June 1992.

[6] Davis, D., and R. Swick, "Workstation Services and Kerberos Authentication at Project Athena", Technical Memorandum TM-424, MIT Laboratory for Computer Science, February 1990.

[7] Levine, P., Gretzinger, M, Diaz, J., Sommerfeld, W., and K. Raeburn, "Section E.1: Service Management System, M.I.T. Project Athena, Cambridge, Mas sachusetts (1987).

[8] CCITT, Recommendation X.509: The Directory Authentication Framework, December 1988.

[9] Neuman, C., "Proxy-Based Authorization and Accounting for Distributed Systems," in Proceedings of the 13th International Conference on Distributed Computing Systems", Pittsburgh, PA, May 1993.

[10] Pato, J., "Using Pre-Authentication to Avoid Password Guessing Attacks", Open Software Foundation DCE Request for Comments 26, December 1992.

[11] National Bureau of Standards, U.S. Department of Commerce, "Data Encryption Standard", Federal Information Processing Standards Publication 46, Washington, DC (1977).

[12] National Bureau of Standards, U.S. Department of Commerce, "DES Modes of Operation", Federal Information Processing Standards Publication 81, Springfield, VA, December 1980.

[13] Stubblebine S., and V. Gligor, "On Message Integrity in Cryptographic Protocols", in Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, California, May 1992.

[14] International Organization for Standardization, "ISO Information Processing Systems - Data Communication High-Level Data Link Control Procedure - Frame Structure", IS 3309, October 1984, 3rd Edition.

[15] Rivest, R., "The MD4 Message Digest Algorithm", RFC 1320, MIT Laboratory for Computer Science, April 1992.

[16] Rivest, R., "The MD5 Message Digest Algorithm", RFC 1321, MIT Laboratory for Computer Science, April 1992.

[17] Bellovin S., and M. Merritt, "Limitations of the Kerberos Authentication System", Computer Communications Review, Vol. 20(5), pp. 119-132, October 1990.

12. Security Considerations

Security issues are discussed throughout this memo.

13. Authors' Addresses

John Kohl
Digital Equipment Corporation
110 Spit Brook Road, M/S ZKO3-3/U14
Nashua, NH 03062

Phone: 603-881-2481
EMail: jtkohl@zk3.dec.com

B. Clifford Neuman
USC/Information Sciences Institute
4676 Admiralty Way #1001
Marina del Rey, CA 90292-6695

Phone: 310-822-1511
EMail: bcn@isi.edu

A. Pseudo-code for protocol processing

This appendix provides pseudo-code describing how the messages are to be constructed and interpreted by clients and servers.

A.1. KRB_AS_REQ generation

```
request.pvno := protocol version; /* pvno = 5 */
request.msg-type := message type; /* type = KRB_AS_REQ */

if(pa_enc_timestamp_required) then
        request.padata.padata-type = PA-ENC-TIMESTAMP;
        get system_time;
        padata-body.patimestamp,pausec = system_time;
        encrypt padata-body into request.padata.padata-value
                using client.key; /* derived from password */
endif

body.kdc-options := users's preferences;
body.cname := user's name;
body.realm := user's realm;
body.sname := service's name; /* usually "krbtgt",
                                 "localrealm" */
if (body.kdc-options.POSTDATED is set) then
        body.from := requested starting time;
else
        omit body.from;
endif
body.till := requested end time;
if (body.kdc-options.RENEWABLE is set) then
        body.rtime := requested final renewal time;
endif
body.nonce := random_nonce();
body.etype := requested etypes;
if (user supplied addresses) then
        body.addresses := user's addresses;
else
        omit body.addresses;
endif
omit body.enc-authorization-data;
request.req-body := body;

kerberos := lookup(name of local kerberos server (or servers));
send(packet,kerberos);

wait(for response);
if (timed_out) then
        retry or use alternate server;
endif
```

A.2. KRB_AS_REQ verification and KRB_AS_REP generation

```
decode message into req;

client := lookup(req.cname,req.realm);
server := lookup(req.sname,req.realm);
get system_time;
kdc_time := system_time.seconds;

if (!client) then
        /* no client in Database */
        error_out(KDC_ERR_C_PRINCIPAL_UNKNOWN);
endif
if (!server) then
        /* no server in Database */
        error_out(KDC_ERR_S_PRINCIPAL_UNKNOWN);
endif

if(client.pa_enc_timestamp_required and
   pa_enc_timestamp not present) then
        error_out(KDC_ERR_PREAUTH_REQUIRED(PA_ENC_TIMESTAMP));
endif

if(pa_enc_timestamp present) then
        decrypt req.padata-value into decrypted_enc_timestamp
                using client.key;
                using auth_hdr.authenticator.subkey;
        if (decrypt_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        if(decrypted_enc_timestamp is not within allowable
                skew) then error_out(KDC_ERR_PREAUTH_FAILED);
        endif
        if(decrypted_enc_timestamp and usec is replay)
                error_out(KDC_ERR_PREAUTH_FAILED);
        endif
        add decrypted_enc_timestamp and usec to replay cache;
endif

use_etype := first supported etype in req.etypes;

if (no support for req.etypes) then
        error_out(KDC_ERR_ETYPE_NOSUPP);
endif

new_tkt.vno := ticket version; /* = 5 */
new_tkt.sname := req.sname;
new_tkt.srealm := req.srealm;
reset all flags in new_tkt.flags;
```

```
        /* It should be noted that local policy may affect the  */
        /* processing of any of these flags.  For example, some */
        /* realms may refuse to issue renewable tickets         */

        if (req.kdc-options.FORWARDABLE is set) then
                set new_tkt.flags.FORWARDABLE;
        endif
        if (req.kdc-options.PROXIABLE is set) then
                set new_tkt.flags.PROXIABLE;
        endif
        if (req.kdc-options.ALLOW-POSTDATE is set) then
                set new_tkt.flags.ALLOW-POSTDATE;
        endif
        if ((req.kdc-options.RENEW is set) or
            (req.kdc-options.VALIDATE is set) or
            (req.kdc-options.PROXY is set) or
            (req.kdc-options.FORWARDED is set) or
            (req.kdc-options.ENC-TKT-IN-SKEY is set)) then
                error_out(KDC_ERR_BADOPTION);
        endif

        new_tkt.session := random_session_key();
        new_tkt.cname := req.cname;
        new_tkt.crealm := req.crealm;
        new_tkt.transited := empty_transited_field();

        new_tkt.authtime := kdc_time;

        if (req.kdc-options.POSTDATED is set) then
           if (against_postdate_policy(req.from)) then
                error_out(KDC_ERR_POLICY);
           endif
           set new_tkt.flags.INVALID;
           new_tkt.starttime := req.from;
        else
           omit new_tkt.starttime; /* treated as authtime when
                                      omitted */
        endif
        if (req.till = 0) then
                till := infinity;
        else
                till := req.till;
        endif

        new_tkt.endtime := min(till,
                               new_tkt.starttime+client.max_life,
                               new_tkt.starttime+server.max_life,
                               new_tkt.starttime+max_life_for_realm);
```

```
if ((req.kdc-options.RENEWABLE-OK is set) and
    (new_tkt.endtime < req.till)) then
        /* we set the RENEWABLE option for later processing */
        set req.kdc-options.RENEWABLE;
        req.rtime := req.till;
endif

if (req.rtime = 0) then
        rtime := infinity;
else
        rtime := req.rtime;
endif

if (req.kdc-options.RENEWABLE is set) then
        set new_tkt.flags.RENEWABLE;
        new_tkt.renew-till := min(rtime,
        new_tkt.starttime+client.max_rlife,
        new_tkt.starttime+server.max_rlife,
        new_tkt.starttime+max_rlife_for_realm);
else
        omit new_tkt.renew-till; /* only present if RENEWABLE */
endif

if (req.addresses) then
        new_tkt.caddr := req.addresses;
else
        omit new_tkt.caddr;
endif

new_tkt.authorization_data := empty_authorization_data();

encode to-be-encrypted part of ticket into OCTET STRING;
new_tkt.enc-part := encrypt OCTET STRING
    using etype_for_key(server.key), server.key, server.p_kvno;


/* Start processing the response */

resp.pvno := 5;
resp.msg-type := KRB_AS_REP;
resp.cname := req.cname;
resp.crealm := req.realm;
resp.ticket := new_tkt;

resp.key := new_tkt.session;
resp.last-req := fetch_last_request_info(client);
resp.nonce := req.nonce;
resp.key-expiration := client.expiration;
```

```
        resp.flags := new_tkt.flags;

        resp.authtime := new_tkt.authtime;
        resp.starttime := new_tkt.starttime;
        resp.endtime := new_tkt.endtime;

        if (new_tkt.flags.RENEWABLE) then
                resp.renew-till := new_tkt.renew-till;
        endif

        resp.realm := new_tkt.realm;
        resp.sname := new_tkt.sname;

        resp.caddr := new_tkt.caddr;

        encode body of reply into OCTET STRING;

        resp.enc-part := encrypt OCTET STRING
                         using use_etype, client.key, client.p_kvno;
        send(resp);
```

A.3. KRB_AS_REP verification

```
        decode response into resp;

        if (resp.msg-type = KRB_ERROR) then
                if(error = KDC_ERR_PREAUTH_REQUIRED(PA_ENC_TIMESTAMP))
                        then set pa_enc_timestamp_required;
                        goto KRB_AS_REQ;
                endif
                process_error(resp);
                return;
        endif

        /* On error, discard the response, and zero the session key */
        /* from the response immediately */

        key = get_decryption_key(resp.enc-part.kvno, resp.enc-part.etype,
                                 resp.padata);
        unencrypted part of resp := decode of decrypt of resp.enc-part
                                 using resp.enc-part.etype and key;
        zero(key);

        if (common_as_rep_tgs_rep_checks fail) then
                destroy resp.key;
                return error;
        endif

        if near(resp.princ_exp) then
```

```
            print(warning message);
        endif
        save_for_later(ticket,session,client,server,times,flags);
```

A.4. KRB_AS_REP and KRB_TGS_REP common checks

```
        if (decryption_error() or
            (req.cname != resp.cname) or
            (req.realm != resp.crealm) or
            (req.sname != resp.sname) or
            (req.realm != resp.realm) or
            (req.nonce != resp.nonce) or
            (req.addresses != resp.caddr)) then
                destroy resp.key;
                return KRB_AP_ERR_MODIFIED;
        endif

        /* make sure no flags are set that shouldn't be, and that  */
        /* all that should be are set                              */
        if (!check_flags_for_compatability(req.kdc-options,resp.flags))
                then destroy resp.key;
                return KRB_AP_ERR_MODIFIED;
        endif

        if ((req.from = 0) and
            (resp.starttime is not within allowable skew)) then
                destroy resp.key;
                return KRB_AP_ERR_SKEW;
        endif
        if ((req.from != 0) and (req.from != resp.starttime)) then
                destroy resp.key;
                return KRB_AP_ERR_MODIFIED;
        endif
        if ((req.till != 0) and (resp.endtime > req.till)) then
                destroy resp.key;
                return KRB_AP_ERR_MODIFIED;
        endif

        if ((req.kdc-options.RENEWABLE is set) and
            (req.rtime != 0) and (resp.renew-till > req.rtime)) then
                destroy resp.key;
                return KRB_AP_ERR_MODIFIED;
        endif
        if ((req.kdc-options.RENEWABLE-OK is set) and
            (resp.flags.RENEWABLE) and
            (req.till != 0) and
            (resp.renew-till > req.till)) then
                destroy resp.key;
                return KRB_AP_ERR_MODIFIED;
```

```
        endif
```

A.5. KRB_TGS_REQ generation

```
        /* Note that make_application_request might have to     */
        /* recursivly call this routine to get the appropriate  */
        /* ticket-granting ticket                               */

        request.pvno := protocol version; /* pvno = 5 */
        request.msg-type := message type; /* type = KRB_TGS_REQ */

        body.kdc-options := users's preferences;
        /* If the TGT is not for the realm of the end-server  */
        /* then the sname will be for a TGT for the end-realm */
        /* and the realm of the requested ticket (body.realm) */
        /* will be that of the TGS to which the TGT we are    */
        /* sending applies                                    */
        body.sname := service's name;
        body.realm := service's realm;

        if (body.kdc-options.POSTDATED is set) then
                body.from := requested starting time;
        else
                omit body.from;
        endif
        body.till := requested end time;
        if (body.kdc-options.RENEWABLE is set) then
                body.rtime := requested final renewal time;
        endif
        body.nonce := random_nonce();
        body.etype := requested etypes;
        if (user supplied addresses) then
                body.addresses := user's addresses;
        else
                omit body.addresses;
        endif

        body.enc-authorization-data := user-supplied data;
        if (body.kdc-options.ENC-TKT-IN-SKEY) then
                body.additional-tickets_ticket := second TGT;
        endif

        request.req-body := body;
        check := generate_checksum (req.body,checksumtype);

        request.padata[0].padata-type := PA-TGS-REQ;
        request.padata[0].padata-value := create a KRB_AP_REQ using
                                          the TGT and checksum
```

```
        /* add in any other padata as required/supplied */

        kerberos := lookup(name of local kerberose server (or servers));
        send(packet,kerberos);

        wait(for response);
        if (timed_out) then
                retry or use alternate server;
        endif
```

A.6. KRB_TGS_REQ verification and KRB_TGS_REP generation

```
        /* note that reading the application request requires first
        determining the server for which a ticket was issued, and
        choosing the correct key for decryption.  The name of the
        server appears in the plaintext part of the ticket. */

        if (no KRB_AP_REQ in req.padata) then
                error_out(KDC_ERR_PADATA_TYPE_NOSUPP);
        endif
        verify KRB_AP_REQ in req.padata;

        /* Note that the realm in which the Kerberos server is
        operating is determined by the instance from the
        ticket-granting ticket.  The realm in the ticket-granting
        ticket is the realm under which the ticket granting ticket was
        issued.  It is possible for a single Kerberos server to
        support more than one realm. */

        auth_hdr := KRB_AP_REQ;
        tgt := auth_hdr.ticket;

        if (tgt.sname is not a TGT for local realm and is not
                req.sname) then error_out(KRB_AP_ERR_NOT_US);

        realm := realm_tgt_is_for(tgt);

        decode remainder of request;

        if (auth_hdr.authenticator.cksum is missing) then
                error_out(KRB_AP_ERR_INAPP_CKSUM);
        endif
        if (auth_hdr.authenticator.cksum type is not supported) then
                error_out(KDC_ERR_SUMTYPE_NOSUPP);
        endif
        if (auth_hdr.authenticator.cksum is not both collision-proof
            and keyed)  then
                error_out(KRB_AP_ERR_INAPP_CKSUM);
        endif
```

```
set computed_checksum := checksum(req);
if (computed_checksum != auth_hdr.authenticatory.cksum) then
        error_out(KRB_AP_ERR_MODIFIED);
endif

server := lookup(req.sname,realm);

if (!server) then
        if (is_foreign_tgt_name(server)) then
                server := best_intermediate_tgs(server);
        else
                /* no server in Database */
                error_out(KDC_ERR_S_PRINCIPAL_UNKNOWN);
        endif
endif

session := generate_random_session_key();


use_etype := first supported etype in req.etypes;

if (no support for req.etypes) then
        error_out(KDC_ERR_ETYPE_NOSUPP);
endif

new_tkt.vno := ticket version; /* = 5 */
new_tkt.sname := req.sname;
new_tkt.srealm := realm;
reset all flags in new_tkt.flags;

/* It should be noted that local policy may affect the  */
/* processing of any of these flags.  For example, some */
/* realms may refuse to issue renewable tickets         */

new_tkt.caddr := tgt.caddr;
resp.caddr := NULL; /* We only include this if they change */
if (req.kdc-options.FORWARDABLE is set) then
        if (tgt.flags.FORWARDABLE is reset) then
                error_out(KDC_ERR_BADOPTION);
        endif
        set new_tkt.flags.FORWARDABLE;
endif
if (req.kdc-options.FORWARDED is set) then
        if (tgt.flags.FORWARDABLE is reset) then
                error_out(KDC_ERR_BADOPTION);
        endif
        set new_tkt.flags.FORWARDED;
        new_tkt.caddr := req.addresses;
```

```
                resp.caddr := req.addresses;
        endif
        if (tgt.flags.FORWARDED is set) then
                set new_tkt.flags.FORWARDED;
        endif

        if (req.kdc-options.PROXIABLE is set) then
                if (tgt.flags.PROXIABLE is reset)
                        error_out(KDC_ERR_BADOPTION);
                endif
                set new_tkt.flags.PROXIABLE;
        endif
        if (req.kdc-options.PROXY is set) then
                if (tgt.flags.PROXIABLE is reset) then
                        error_out(KDC_ERR_BADOPTION);
                endif
                set new_tkt.flags.PROXY;
                new_tkt.caddr := req.addresses;
                resp.caddr := req.addresses;
        endif

        if (req.kdc-options.POSTDATE is set) then
                if (tgt.flags.POSTDATE is reset)
                        error_out(KDC_ERR_BADOPTION);
                endif
                set new_tkt.flags.POSTDATE;
        endif
        if (req.kdc-options.POSTDATED is set) then
                if (tgt.flags.POSTDATE is reset) then
                        error_out(KDC_ERR_BADOPTION);
                endif
                set new_tkt.flags.POSTDATED;
                set new_tkt.flags.INVALID;
                if (against_postdate_policy(req.from)) then
                        error_out(KDC_ERR_POLICY);
                endif
                new_tkt.starttime := req.from;
        endif


        if (req.kdc-options.VALIDATE is set) then
                if (tgt.flags.INVALID is reset) then
                        error_out(KDC_ERR_POLICY);
                endif
                if (tgt.starttime > kdc_time) then
                        error_out(KRB_AP_ERR_NYV);
                endif
                if (check_hot_list(tgt)) then
```

```
                error_out(KRB_AP_ERR_REPEAT);
        endif
        tkt := tgt;
        reset new_tkt.flags.INVALID;
endif

if (req.kdc-options.(any flag except ENC-TKT-IN-SKEY, RENEW,
                     and those already processed) is set) then
        error_out(KDC_ERR_BADOPTION);
endif

new_tkt.authtime := tgt.authtime;

if (req.kdc-options.RENEW is set) then
  /* Note that if the endtime has already passed, the ticket */
  /* would have been rejected in the initial authentication  */
  /* stage, so there is no need to check again here          */
        if (tgt.flags.RENEWABLE is reset) then
                error_out(KDC_ERR_BADOPTION);
        endif
        if (tgt.renew-till >= kdc_time) then
                error_out(KRB_AP_ERR_TKT_EXPIRED);
        endif
        tkt := tgt;
        new_tkt.starttime := kdc_time;
        old_life := tgt.endttime - tgt.starttime;
        new_tkt.endtime := min(tgt.renew-till,
                               new_tkt.starttime + old_life);
else
        new_tkt.starttime := kdc_time;
        if (req.till = 0) then
                till := infinity;
        else
                till := req.till;
        endif
        new_tkt.endtime := min(till,
                           new_tkt.starttime+client.max_life,
                           new_tkt.starttime+server.max_life,
                           new_tkt.starttime+max_life_for_realm,
                           tgt.endtime);

        if ((req.kdc-options.RENEWABLE-OK is set) and
            (new_tkt.endtime < req.till) and
            (tgt.flags.RENEWABLE is set) then
                /* we set the RENEWABLE option for later  */
                /* processing                             */
                set req.kdc-options.RENEWABLE;
                req.rtime := min(req.till, tgt.renew-till);
```

```
        endif
endif

if (req.rtime = 0) then
        rtime := infinity;
else
        rtime := req.rtime;
endif

if ((req.kdc-options.RENEWABLE is set) and
    (tgt.flags.RENEWABLE is set)) then
        set new_tkt.flags.RENEWABLE;
        new_tkt.renew-till := min(rtime,
        new_tkt.starttime+client.max_rlife,
        new_tkt.starttime+server.max_rlife,
        new_tkt.starttime+max_rlife_for_realm,
        tgt.renew-till);
else
        new_tkt.renew-till := OMIT;
                    /* leave the renew-till field out */
endif
if (req.enc-authorization-data is present) then
        decrypt req.enc-authorization-data
                into    decrypted_authorization_data
                using auth_hdr.authenticator.subkey;
        if (decrypt_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        endif
endif
new_tkt.authorization_data :=
req.auth_hdr.ticket.authorization_data +
                        decrypted_authorization_data;

new_tkt.key := session;
new_tkt.crealm := tgt.crealm;
new_tkt.cname := req.auth_hdr.ticket.cname;

if (realm_tgt_is_for(tgt) := tgt.realm) then
        /* tgt issued by local realm */
        new_tkt.transited := tgt.transited;
else
        /* was issued for this realm by some other realm */
        if (tgt.transited.tr-type not supported) then
                error_out(KDC_ERR_TRTYPE_NOSUPP);
        endif
        new_tkt.transited
           := compress_transited(tgt.transited + tgt.realm)
endif
```

```
        encode encrypted part of new_tkt into OCTET STRING;
        if (req.kdc-options.ENC-TKT-IN-SKEY is set) then
                if (server not specified) then
                        server = req.second_ticket.client;
                endif
                if ((req.second_ticket is not a TGT) or
                    (req.second_ticket.client != server)) then
                        error_out(KDC_ERR_POLICY);
                endif

                new_tkt.enc-part := encrypt OCTET STRING using
                        using etype_for_key(second-ticket.key),
                                                    second-ticket.key;
        else
                new_tkt.enc-part := encrypt OCTET STRING
                        using etype_for_key(server.key), server.key,
                                                    server.p_kvno;
        endif

        resp.pvno := 5;
        resp.msg-type := KRB_TGS_REP;
        resp.crealm := tgt.crealm;
        resp.cname := tgt.cname;
        resp.ticket := new_tkt;

        resp.key := session;
        resp.nonce := req.nonce;
        resp.last-req := fetch_last_request_info(client);
        resp.flags := new_tkt.flags;

        resp.authtime := new_tkt.authtime;
        resp.starttime := new_tkt.starttime;
        resp.endtime := new_tkt.endtime;

        omit resp.key-expiration;

        resp.sname := new_tkt.sname;
        resp.realm := new_tkt.realm;

        if (new_tkt.flags.RENEWABLE) then
                resp.renew-till := new_tkt.renew-till;
        endif


        encode body of reply into OCTET STRING;

        if (req.padata.authenticator.subkey)
                resp.enc-part := encrypt OCTET STRING using use_etype,
```

```
                        req.padata.authenticator.subkey;
        else resp.enc-part := encrypt OCTET STRING
                              using use_etype, tgt.key;

        send(resp);
```

A.7. KRB_TGS_REP verification

```
        decode response into resp;

        if (resp.msg-type = KRB_ERROR) then
                process_error(resp);
                return;
        endif

        /* On error, discard the response, and zero the session key from
        the response immediately */

        if (req.padata.authenticator.subkey)
                unencrypted part of resp :=
                        decode of decrypt of resp.enc-part
                        using resp.enc-part.etype and subkey;
        else unencrypted part of resp :=
                        decode of decrypt of resp.enc-part
                        using resp.enc-part.etype and tgt's session key;
        if (common_as_rep_tgs_rep_checks fail) then
                destroy resp.key;
                return error;
        endif

        check authorization_data as necessary;
        save_for_later(ticket,session,client,server,times,flags);
```

A.8. Authenticator generation

```
        body.authenticator-vno := authenticator vno; /* = 5 */
        body.cname, body.crealm := client name;
        if (supplying checksum) then
                body.cksum := checksum;
        endif
        get system_time;
        body.ctime, body.cusec := system_time;
        if (selecting sub-session key) then
                select sub-session key;
                body.subkey := sub-session key;
        endif
        if (using sequence numbers) then
                select initial sequence number;
                body.seq-number := initial sequence;
        endif
```

A.9. KRB_AP_REQ generation

```
obtain ticket and session_key from cache;

packet.pvno := protocol version; /* 5 */
packet.msg-type := message type; /* KRB_AP_REQ */

if (desired(MUTUAL_AUTHENTICATION)) then
        set packet.ap-options.MUTUAL-REQUIRED;
else
        reset packet.ap-options.MUTUAL-REQUIRED;
endif
if (using session key for ticket) then
        set packet.ap-options.USE-SESSION-KEY;
else
        reset packet.ap-options.USE-SESSION-KEY;
endif
packet.ticket := ticket; /* ticket */
generate authenticator;
encode authenticator into OCTET STRING;
encrypt OCTET STRING into packet.authenticator
                    using session_key;
```

A.10. KRB_AP_REQ verification

```
receive packet;
if (packet.pvno != 5) then
        either process using other protocol spec
        or error_out(KRB_AP_ERR_BADVERSION);
endif
if (packet.msg-type != KRB_AP_REQ) then
        error_out(KRB_AP_ERR_MSG_TYPE);
endif
if (packet.ticket.tkt_vno != 5) then
        either process using other protocol spec
        or error_out(KRB_AP_ERR_BADVERSION);
endif
if (packet.ap_options.USE-SESSION-KEY is set) then
        retrieve session key from ticket-granting ticket for
         packet.ticket.{sname,srealm,enc-part.etype};
else
   retrieve service key for
   packet.ticket.{sname,srealm,enc-part.etype,enc-part.skvno};
endif
if (no_key_available) then
        if (cannot_find_specified_skvno) then
                error_out(KRB_AP_ERR_BADKEYVER);
        else
                error_out(KRB_AP_ERR_NOKEY);
        endif
```

```
        endif
        decrypt packet.ticket.enc-part into decr_ticket
                                        using retrieved key;
        if (decryption_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        endif
        decrypt packet.authenticator into decr_authenticator
                using decr_ticket.key;
        if (decryption_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        endif
        if (decr_authenticator.{cname,crealm} !=
            decr_ticket.{cname,crealm}) then
                error_out(KRB_AP_ERR_BADMATCH);
        endif
        if (decr_ticket.caddr is present) then
                if (sender_address(packet) is not in decr_ticket.caddr)
                        then error_out(KRB_AP_ERR_BADADDR);
                endif
        elseif (application requires addresses) then
                error_out(KRB_AP_ERR_BADADDR);
        endif
        if (not in_clock_skew(decr_authenticator.ctime,
                              decr_authenticator.cusec)) then
                error_out(KRB_AP_ERR_SKEW);
        endif
        if (repeated(decr_authenticator.{ctime,cusec,cname,crealm}))
                then error_out(KRB_AP_ERR_REPEAT);
        endif
        save_identifier(decr_authenticator.{ctime,cusec,cname,crealm});
        get system_time;
        if ((decr_ticket.starttime-system_time > CLOCK_SKEW) or
            (decr_ticket.flags.INVALID is set)) then
                /* it hasn't yet become valid */
                error_out(KRB_AP_ERR_TKT_NYV);
        endif
        if (system_time-decr_ticket.endtime > CLOCK_SKEW) then
                error_out(KRB_AP_ERR_TKT_EXPIRED);
        endif
        /* caller must check decr_ticket.flags for any pertinent */
        /* details */
        return(OK, decr_ticket, packet.ap_options.MUTUAL-REQUIRED);
```

A.11. KRB_AP_REP generation

```
        packet.pvno := protocol version; /* 5 */
        packet.msg-type := message type; /* KRB_AP_REP */
        body.ctime := packet.ctime;
        body.cusec := packet.cusec;
```

```
        if (selecting sub-session key) then
                select sub-session key;
                body.subkey := sub-session key;
        endif
        if (using sequence numbers) then
                select initial sequence number;
                body.seq-number := initial sequence;
        endif

        encode body into OCTET STRING;

        select encryption type;
        encrypt OCTET STRING into packet.enc-part;
```

A.12. KRB_AP_REP verification

```
        receive packet;
        if (packet.pvno != 5) then
                either process using other protocol spec
                or error_out(KRB_AP_ERR_BADVERSION);
        endif
        if (packet.msg-type != KRB_AP_REP) then
                error_out(KRB_AP_ERR_MSG_TYPE);
        endif
        cleartext := decrypt(packet.enc-part)
                     using ticket's session key;
        if (decryption_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        endif
        if (cleartext.ctime != authenticator.ctime) then
                error_out(KRB_AP_ERR_MUT_FAIL);
        endif
        if (cleartext.cusec != authenticator.cusec) then
                error_out(KRB_AP_ERR_MUT_FAIL);
        endif
        if (cleartext.subkey is present) then
                save cleartext.subkey for future use;
        endif
        if (cleartext.seq-number is present) then
                save cleartext.seq-number for future verifications;
        endif
        return(AUTHENTICATION_SUCCEEDED);
```

A.13. KRB_SAFE generation

```
        collect user data in buffer;

        /* assemble packet: */
        packet.pvno := protocol version; /* 5 */
        packet.msg-type := message type; /* KRB_SAFE */
```

```
        body.user-data := buffer; /* DATA */
        if (using timestamp) then
                get system_time;
                body.timestamp, body.usec := system_time;
        endif
        if (using sequence numbers) then
                body.seq-number := sequence number;
        endif
        body.s-address := sender host addresses;
        if (only one recipient) then
                body.r-address := recipient host address;
        endif
        checksum.cksumtype := checksum type;
        compute checksum over body;
        checksum.checksum := checksum value; /* checksum.checksum */
        packet.cksum := checksum;
        packet.safe-body := body;
```

A.14. KRB_SAFE verification

```
        receive packet;
        if (packet.pvno != 5) then
                either process using other protocol spec
                or error_out(KRB_AP_ERR_BADVERSION);
        endif
        if (packet.msg-type != KRB_SAFE) then
                error_out(KRB_AP_ERR_MSG_TYPE);
        endif
        if (packet.checksum.cksumtype is not both collision-proof
                                              and keyed) then
                error_out(KRB_AP_ERR_INAPP_CKSUM);
        endif
        if (safe_priv_common_checks_ok(packet)) then
                set computed_checksum := checksum(packet.body);
                if (computed_checksum != packet.checksum) then
                        error_out(KRB_AP_ERR_MODIFIED);
                endif
                return (packet, PACKET_IS_GENUINE);
        else
                return common_checks_error;
        endif
```

A.15. KRB_SAFE and KRB_PRIV common checks

```
        if (packet.s-address != O/S_sender(packet)) then
            /* O/S report of sender not who claims to have sent it */
            error_out(KRB_AP_ERR_BADADDR);
        endif
        if ((packet.r-address is present) and
            (packet.r-address != local_host_address)) then
```

```
                /* was not sent to proper place */
                error_out(KRB_AP_ERR_BADADDR);
        endif
        if (((packet.timestamp is present) and
             (not in_clock_skew(packet.timestamp,packet.usec))) or
            (packet.timestamp is not present and timestamp expected))
                then error_out(KRB_AP_ERR_SKEW);
        endif
        if (repeated(packet.timestamp,packet.usec,packet.s-address))
                then error_out(KRB_AP_ERR_REPEAT);
        endif
        if (((packet.seq-number is present) and
             ((not in_sequence(packet.seq-number)))) or
            (packet.seq-number is not present and sequence expected))
                then error_out(KRB_AP_ERR_BADORDER);
        endif
        if (packet.timestamp not present and
            packet.seq-number not present) then
                error_out(KRB_AP_ERR_MODIFIED);
        endif

        save_identifier(packet.{timestamp,usec,s-address},
                        sender_principal(packet));

        return PACKET_IS_OK;
```

A.16. KRB_PRIV generation

```
        collect user data in buffer;

        /* assemble packet: */
        packet.pvno := protocol version; /* 5 */
        packet.msg-type := message type; /* KRB_PRIV */

        packet.enc-part.etype := encryption type;

        body.user-data := buffer;
        if (using timestamp) then
                get system_time;
                body.timestamp, body.usec := system_time;
        endif
        if (using sequence numbers) then
                body.seq-number := sequence number;
        endif
        body.s-address := sender host addresses;
        if (only one recipient) then
                body.r-address := recipient host address;
        endif
```

```
        encode body into OCTET STRING;

        select encryption type;
        encrypt OCTET STRING into packet.enc-part.cipher;
```

A.17. KRB_PRIV verification

```
        receive packet;
        if (packet.pvno != 5) then
                either process using other protocol spec
                or error_out(KRB_AP_ERR_BADVERSION);
        endif
        if (packet.msg-type != KRB_PRIV) then
                error_out(KRB_AP_ERR_MSG_TYPE);
        endif

        cleartext := decrypt(packet.enc-part) using negotiated key;
        if (decryption_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        endif

        if (safe_priv_common_checks_ok(cleartext)) then
            return(cleartext.DATA, PACKET_IS_GENUINE_AND_UNMODIFIED);
        else
                return common_checks_error;
        endif
```

A.18. KRB_CRED generation

```
        invoke KRB_TGS; /* obtain tickets to be provided to peer */

        /* assemble packet: */
        packet.pvno := protocol version; /* 5 */
        packet.msg-type := message type; /* KRB_CRED */

        for (tickets[n] in tickets to be forwarded) do
                packet.tickets[n] = tickets[n].ticket;
        done

        packet.enc-part.etype := encryption type;

        for (ticket[n] in tickets to be forwarded) do
                body.ticket-info[n].key = tickets[n].session;
                body.ticket-info[n].prealm = tickets[n].crealm;
                body.ticket-info[n].pname = tickets[n].cname;
                body.ticket-info[n].flags = tickets[n].flags;
                body.ticket-info[n].authtime = tickets[n].authtime;
                body.ticket-info[n].starttime = tickets[n].starttime;
                body.ticket-info[n].endtime = tickets[n].endtime;
                body.ticket-info[n].renew-till = tickets[n].renew-till;
```

```
                body.ticket-info[n].srealm = tickets[n].srealm;
                body.ticket-info[n].sname = tickets[n].sname;
                body.ticket-info[n].caddr = tickets[n].caddr;
        done

        get system_time;
        body.timestamp, body.usec := system_time;

        if (using nonce) then
                body.nonce := nonce;
        endif

        if (using s-address) then
                body.s-address := sender host addresses;
        endif
        if (limited recipients) then
                body.r-address := recipient host address;
        endif

        encode body into OCTET STRING;

        select encryption type;
        encrypt OCTET STRING into packet.enc-part.cipher
        using negotiated encryption key;
```

A.19. KRB_CRED verification

```
        receive packet;
        if (packet.pvno != 5) then
                either process using other protocol spec
                or error_out(KRB_AP_ERR_BADVERSION);
        endif
        if (packet.msg-type != KRB_CRED) then
                error_out(KRB_AP_ERR_MSG_TYPE);
        endif

        cleartext := decrypt(packet.enc-part) using negotiated key;
        if (decryption_error()) then
                error_out(KRB_AP_ERR_BAD_INTEGRITY);
        endif
        if ((packet.r-address is present or required) and
           (packet.s-address != O/S_sender(packet)) then
            /* O/S report of sender not who claims to have sent it */
            error_out(KRB_AP_ERR_BADADDR);
        endif
        if ((packet.r-address is present) and
            (packet.r-address != local_host_address)) then
                /* was not sent to proper place */
                error_out(KRB_AP_ERR_BADADDR);
```

```
        endif
        if (not in_clock_skew(packet.timestamp,packet.usec)) then
                error_out(KRB_AP_ERR_SKEW);
        endif
        if (repeated(packet.timestamp,packet.usec,packet.s-address))
                then error_out(KRB_AP_ERR_REPEAT);
        endif
        if (packet.nonce is required or present) and
           (packet.nonce != expected-nonce) then
                error_out(KRB_AP_ERR_MODIFIED);
        endif

        for (ticket[n] in tickets that were forwarded) do
                save_for_later(ticket[n],key[n],principal[n],
                               server[n],times[n],flags[n]);
        return
```

A.20. KRB_ERROR generation

```
        /* assemble packet: */
        packet.pvno := protocol version; /* 5 */
        packet.msg-type := message type; /* KRB_ERROR */

        get system_time;
        packet.stime, packet.susec := system_time;
        packet.realm, packet.sname := server name;

        if (client time available) then
                packet.ctime, packet.cusec := client_time;
        endif
        packet.error-code := error code;
        if (client name available) then
                packet.cname, packet.crealm := client name;
        endif
        if (error text available) then
                packet.e-text := error text;
        endif
        if (error data available) then
                packet.e-data := error data;
        endif
```

01

CDMA Systems Engineering Handbook

Jhong Sam Lee
Leonard E. Miller

Artech House
Boston • London

Library of Congress Cataloging-in-Publication Data
Lee, Jhong S.
CDMA systems engineering handbook /Jhong S. Lee, Leonard E. Miller.
p. cm. — (Artech House mobile communications library)
Includes bibliographical references and index.
ISBN 0-89006-990-5 (alk. paper)
1. Code division multiple access. 2. Mobile communication systems.
I. Miller, Leonard E. II. Title. III. Series.
TK5103.45.L44 1998
621.382—dc21 98-33846
CIP

British Library Cataloguing in Publication Data
Lee, Jhong S.
CDMA systems engineering handbook.— (Artech House mobile communications library)
1. Code division multiple access—Handbooks, manuals, etc.
I. Title II. Miller, Leonard E.
621.3'84'56

ISBN 0-89006-990-5

Cover design by Lynda Fishbourne

International Standard Book Number: 0-89006-990-5
Library of Congress Catalog Card Number: 98-33846

10 9 8 7 6 5 4 3

To our wives, Helen Lee and Fran Miller

register, with a period of $2^{42} - 1 \approx 4.4 \times 10^{12}$ chips (lasting over 41 days at 1.2288 Mcps), that implements the recursion that is specified by the characteristic polynomial

$$f(x) = 1 + x^7 + x^9 + x^{11} + x^{15} + x^{16} + x^{17} + x^{20} + x^{21} + x^{23} + x^{24} + x^{25} + x^{26} + x^{32} + x^{35} + x^{36} + x^{37} + x^{39} + x^{40} + x^{41} + x^{42} \quad (4.4)$$

A phase offset of the original long PN code sequence that is unique to the particular paging channel and base station is obtained by combining the outputs of the shift register stages selected by a 42-bit mask, as illustrated in Figure 4.16. For paging channels, the mask is derived from the paging channel number and and the base station ID in the form of its pilot PN code offset; that is, the paging channel mask (beginning with the most significant bit) is given by

$$1100011001101000 0xxx 000000000000 yyyyyyyyyy \quad (4.5)$$

where xxx is the paging channel number and yyyyyyyyy is the base station pilot PN sequence offset index.

The decimated PN sequence is some shift of the original sequence, running at 1/64 times the rate. Because the sampled PN code has the same rate as the encoded data, the data waveform is not spread in bandwidth by combining with this particular PN code. The factors affecting the selection of the long-code masks used on the forward and reverse links are discussed in Chapter 6, as is the effect of decimating a PN sequence.

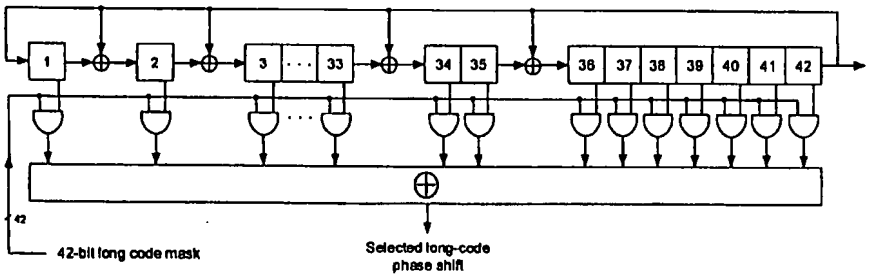


Figure 4.16 42-Stage long-code PN generator with user-distinct mask.

Each paging channel symbol data stream is spread by its designated Walsh sequence, corresponding to one of the first through seventh rows of the Hadamard matrix H_1–H_7, then spread in quadrature by the short PN codes. Paging channel modulation parameters are summarized in Table 4.2.

A paging message directed to a particular mobile unit could be broadcast periodically on one of the paging channels, requiring mobile n to scan the active paging channels for a possible message addressed to mobile n. Or, to relieve the mobile of the burden of scanning the different paging channels, the paging message could be broadcast periodically on all paging channels; however, this procedure is wasteful of paging channel capacity. In IS-95, each active paging channel has a number of periodically recurring message slots (e.g., 2,048 slots) available for transmitting pages and other base-to-mobile messages. When a message is queued up for a particular mobile, using a *hash function*, the base station pseudorandomly selects one of the paging channels and pseudorandomly selects one of the message slots in that paging channel for transmission to the particular mobile. The mobile knows exactly which paging channel and message slot to monitor for possible messages because the pseudorandom selection is based on its own identification number and known system parameters. The purpose of the hash function is to distribute the message traffic evenly among the paging channels and message slots. The IS-95 hash functions are discussed in Appendix 4B.

4.2.3.4 Traffic Channels

There are nominally up to 55 forward *traffic channels* that carry the digital voice or data to the mobile user. It is possible to use one or more of the seven Walsh sequences normally assigned to paging channels for traffic channels, for a total of up to 62 traffic channels, if the system becomes heavily loaded.

Table 4.2 Paging channel modulation parameters

Parameter	Value		Units
Data rate	9600	4800	bps
PN chip rate	1.2288	1.2288	Mcps
Code rate	1/2	1/2	bits/code symbol
Code repetition	1	2	mod symbols/code symbols
Modulation symbol rate	19,200	19,200	symbols/sec (sps)
PN chips/mod symbol	64	64	
PN chips/bit	128	256	

It is usually the case, however, that mutual interference on the reverse link limits the number of simultaneous calls to fewer than 55 calls, so the forward link's capacity is more than adequate for the traffic that can be supported by the system. Walsh sequences designated for use on the traffic channels are H_8–H_{31} and H_{33}–H_{63}. A block diagram for the forward traffic channel modulation is given in Figure 4.17. As shown in the diagram, voice data for the mth user is encoded on a frame-by-frame basis using a variable-rate voice coder, which generates data at 8.6, 4.0, 2.0, or 0.8 kbps depending on voice activity, corresponding respectively to 172, 80, 40, or 16 bits per 20-ms frame. A cyclic redundancy check (CRC) error-detecting code calculation is made at the two highest rates, adding 12 bits per frame for the highest rate and 8 bits per frame at the second highest rate. At the mobile receiver, which voice

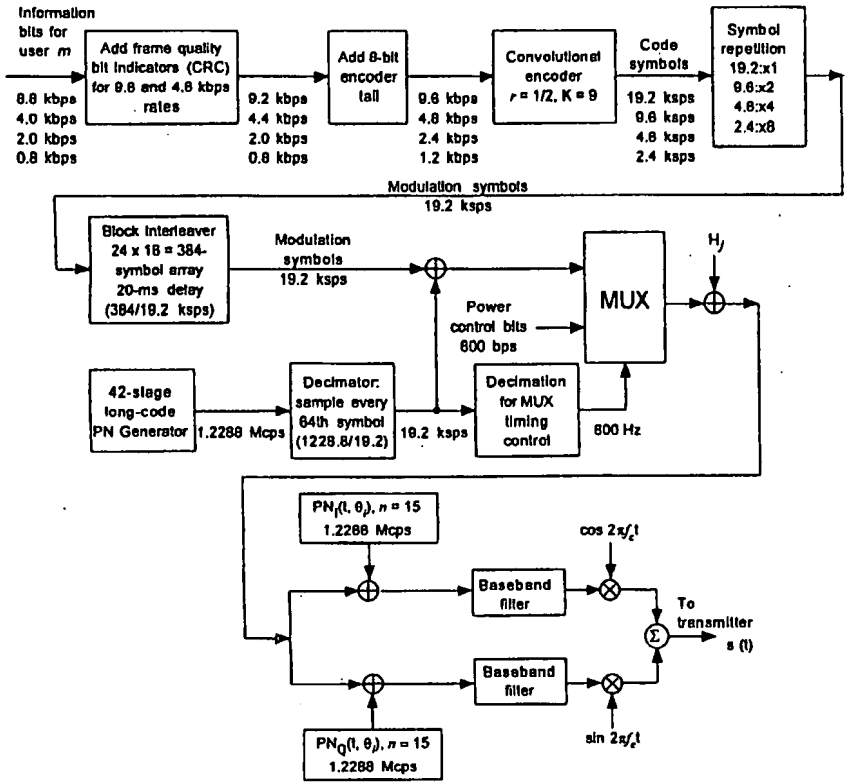


Figure 4.17 Traffic channel modulation.

data rate is being received is determined in part from performing similar CRC calculations, which also provide frame error reception statistics for forward link power control purposes. The theory and use of CRC codes are discussed in Chapter 5.

In anticipation of convolutional coding on a block basis (code symbols in one frame not affecting those in adjacent frames), a convolutional encoder "tail" of 8 bits is added to each block of data to yield blocks of 192, 96, 48, or 24 bits per frame, corresponding to the data rates of 9.6, 4.8, 2.4, and 1.2 kbps going into the encoder, which are defined as full, one-half, one-quarter, and one-eighth rates, respectively. Convolutional encoding is performed using a rate 1/2, constraint length 9 code, resulting in coded symbol rates of 19.2, 9.6, 4.8, and 2.4 ksps.

Coded symbols are repeated as necessary to give a constant number of coded symbols per frame, giving a constant symbol data rate of 19.2 ksps (i.e., 19.2 ksps × 1, 9.6 ksps × 2, 4.8 ksps × 4, 2.4 ksps × 8). The 19.2 ksps × 20 ms = 384 symbols within the same 20-ms frame are interleaved to combat burst errors due to fading, using the same interleaving scheme as on the paging channels.

Each traffic channel's encoded voice or data symbols are scrambled to provide voice privacy by a different phase offset of the long PN code, decimated to yield a code rate of 19.2 kcps. Note that different mobile users are distinguished on the forward link by the orthogonal Walsh sequence associated with the particular traffic channel, not by the user-specific long-code phase offset.

The scrambled data are punctured (overwritten) at an average rate of 800 bps by symbols that are used to control the power of the mobile station; the details of this "power control subchannel" are discussed in Section 4.4.

One of 64 possible Walsh-Hadamard periodic sequences is modulo-2 added to the data stream at 1.2288 Mcps, thus increasing the rate by a factor of 64 chips/modulation symbol (scrambled code symbol). Each symbol for a given traffic channel is represented by the same assigned 64-chip Walsh sequence for a data symbol value of 0 and the sequence's complement for a data symbol value of 1. Walsh-Hadamard sequences of order 64 have the property that all 64 of the sequences are mutually orthogonal. A unique sequence is assigned to each traffic channel so that upon reception at their respective mobile stations, the traffic channels can be distinguished (demultiplexed) based on the orthogonality of the assigned sequences. This orthogonally spread data stream is passed to the quadrature modulator for PN spreading and RF transmission.

$$\begin{array}{cccccccc|cccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} & r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \end{array}$$

$$\langle (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8), (r_{16}, r_{15}, r_{14}, r_{13}, r_{12}, r_{11}, r_{10}, r_9) \rangle$$
$$= \langle (0, 0, 0, 1, 1, 0, 1, 0), (1, 0, 1, 0, 0, 1, 0, 1) \rangle = -6$$

Thus, $x_4 = 1$ and the decoded data sequence is now given as $X = (1, 1, 0, 1) = X_{13}$, which is the correct sequence.

The generalized decoding rule based on the fast Walsh transform can be stated as follows: In a set of Walsh sequences of order $N = 2^K$, correlate every other 2^{j-1}-tuple of symbols with the reverse order of the following 2^{j-1}-tuple. Take the sum of the correlation measures. If the sum is positive, the jth ($j = 1, 2, \ldots, K$) symbol of the index sequence is $x_j = 0$; if the sum is negative, then $x_j = 1$; and if the sum is zero, the index symbol is arbitrarily chosen; that is:

$$\sum_{i=0}^{N/2^j-1} \langle (r_{2^j \cdot i+1}, r_{2^j \cdot i+2}, \ldots, r_{2^j \cdot i+2^{j-1}}), (r_{2^j \cdot i+2^j}, r_{2^j \cdot i+2^{j-1}}, \ldots, r_{2^j \cdot i+2^{j-1}+1}) \rangle \begin{cases} > 0 \Rightarrow x_j = 0 \\ = 0 \Rightarrow \text{pick } x_j = 0 \text{ or } 1 \\ < 0 \Rightarrow x_j = 1 \end{cases} \quad (5.42)$$

In summary, we have considered two methods for decoding the Walsh sequences: correlation decoding and fast Walsh transform decoding. Another possible scheme that can be employed is *matched filter decoding*.

5.6 IS-95 Data Frames

In both cellular [1] and PCS [16] CDMA systems, the forward and reverse link signals are transmitted over the channel in frames or packets. The frame stuctures vary, depending upon the channel category and data rate, such as synchronization channel, paging channel, access channel, and traffic channel.

As an example, consider the forward traffic channel frame structures shown in Figure 5.9. The 9,600-bps, 4,800-bps, 2,400-bps, and 1,200-bps data rates for the traffic channel specify different frame structures. These data rates are the input data rates for the convolutional encoder, whose output

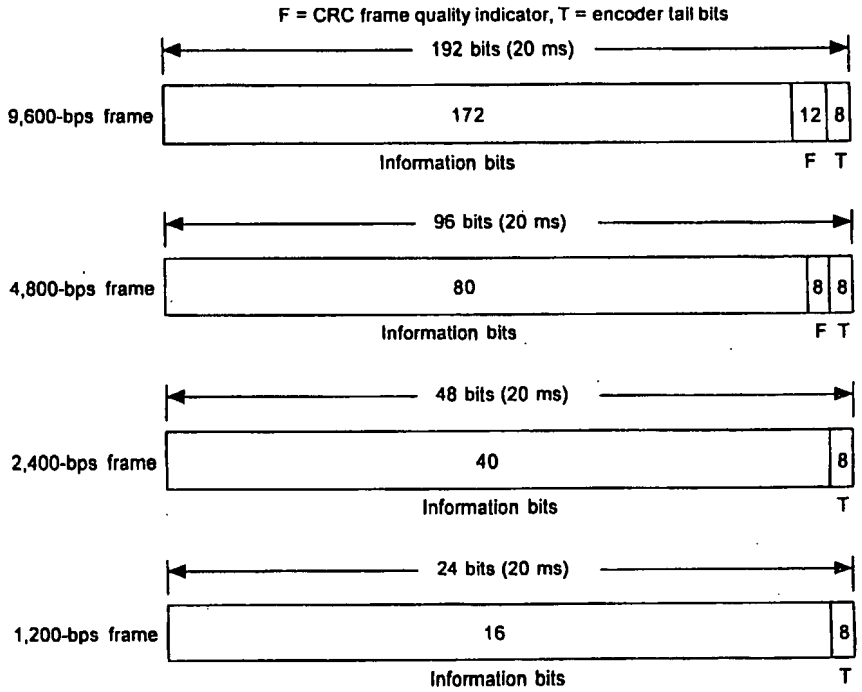


Figure 5.9 Forward traffic channel frame structures (from [1]).

is fed to the 20-ms delay block interleaver after symbol repetition to make the effective symbol rate 19.2 ksps for all four data rates, as shown previously in Figure 4.17.

For the 9,600-bps transmission rate, a total of 192 bits can be transmitted in a 20-ms frame duration. These 192 bits are composed of 172 "information" bits, followed by 12 *frame quality indicator bits* and 8 *encoder tail bits*. It is also observed from Figure 5.9 that the forward traffic frame for the 4,800-bps transmission rate consists of 96 bits that are composed of 80 information bits, 8 frame quality indicator bits, and 8 encoder tail bits. The forward traffic channel frames for the 2,400-bps and 1,200-bps transmission rates contain only information bits (40 and 16 bits, respectively) and 8 encoder tail bits each, without frame quality indicator bits. The frame quality indicator bits are "parity check" bits used in the system's error-detection

scheme, which employs CRC codes [17]. We clarify and explain in detail all these new terms later in this section.

When information bits contained within a block of bits designated as a "frame" or "packet" are transmitted, as depicted in Figure 5.9, it is necessary to determine whether the frame is received in error at the receiving end. To determine the status of "error" or "no error," a scheme is employed wherein frame quality indicator bits are used in an automatic error detection coding technique using cyclic codes. In the following section, we develop the fundamental theory of cyclic codes and proceed to the level of understanding completely the design and operation of the frame quality indicator calculations performed in the cellular CDMA system [1] as well as PCS CDMA systems [16]. In a way, the frames shown in Figure 5.9 can be looked at as codewords in the error detection coding scheme, and thus we begin with some basic concepts of block codes to accomplish our objectives.

5.7 Linear Block Codes

Assume that we have a sequence of binary bits coming out of an information source. We wish to implement a block coding scheme [18–19] for either detecting or correcting transmission errors. In any case, we must first "encode" the information sequence. The block-encoding process involves the following procedure: (1) Collect k successive information bits as a message block; (2) feed the k bits of the message block into the encoder and obtain the coded sequence of n digits, where $n > k$, as diagrammed in Figure 5.10. The result of this procedure is an (n, k) linear code with rate k/n.

Because the message block consists of k information bits, 2^k distinct message blocks are possible, and the encoder output generates 2^k possible

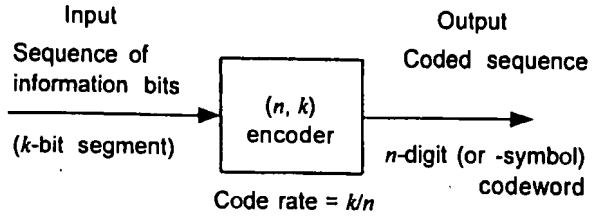


Figure 5.10 Concept of the encoding process.

coded sequences of length n digits, called *codewords*. We say that the block code consists of a set of 2^k codewords. The word "code" connotes an ensemble or a set, whereas the codewords are the elements in the set. Note that the n-digit codeword is an n-tuple, and thus it is a *code vector* in the vector space V_n of all n-tuples. The encoder is a machine (apparatus) or a mathematical rule that transforms the k information bits into an n-digit "coded" sequence.

Now, obviously the block code consisting of 2^k n-tuple codewords is a particular set of n-tuple sequences (vectors) chosen from the set of 2^n possible binary n-tuple vectors. This chosen set, based on a particular encoding rule, is the linear block code, and it is defined as a group:

Definition: A linear block code is a set of 2^k n-tuple vectors that form a subspace of the n-dimensional vector space V_n of all n-tuples.

From the definition of a linear code, the chosen code, being a subspace, must include the all-zero n-tuple, and the dimension of the subspace is k, as we defined these concepts previously in Section 5.3.4. Figure 5.11 depicts the concept of an (n, k) code in the vector space V_n of all n-tuples. The question is, how do we select 2^k n-tuple codewords from the set of all 2^n n-tuple vectors? This is the problem of encoding or designing a coding scheme.

Recall the Walsh sequences that we have defined in the first part of this chapter. The Walsh sequences of order 64 consist of $64 = 2^6$ 64-tuple vectors, and they are the codewords chosen in effect from all the 2^{64} possible 64-tuple sequences. It is in this sense that the Walsh sequences are codewords, and they form a subspace of the 64-dimensional subspace of all 64-tuples—the dimension of the subspace is only six, the number of digits in the index sequence. In fact, this is the orthogonal code that is used in the reverse link in CDMA systems [1, 16]. In Section 5.3.5, we treated the subject of generating Walsh functions using basis vectors. This is exactly the method we use in a block-encoding scheme.

The transformation of the message into a codeword is done in the encoder. The encoder's function for an (n, k) code can be fulfilled by specifying the *generator matrix* G of the code (see eq. (5.26)), which consists of k *linearly independent* n-tuple row vectors. Then all codewords can be generated by a *linear combination* of these row vectors. The coding therefore is to execute the operation of linear combining of the row vectors of the

03

Protection of Data and Delegated Keys in Digital Distribution

Masahiro Mambo[1], Eiji Okamoto[1] and Kouichi Sakurai[2]

[1] School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai Tatsunokuchi Nomi Ishikawa, 923-12 Japan
Email: {mambo,okamoto}@jaist.ac.jp
[2] Dept. of Computer and Communication Engineering, Kyushu University
Hakozaki Higashi-ku Fukuoka, 812-81 Japan Email: sakurai@csce.kyushu-u.ac.jp

Abstract. A cryptography is quite effective in protecting digital information from unauthorized access. But if a receiver of information is determined after the encryption of the information, e.g. a posted encrypted news is withdrawn by an arbitrary user in open networks, we need an additional mechanism for converting the encrypted information into a form accessible only to an admissible user. Even though such a transformation is done by the consecutive execution of decryption of a ciphertext and re-encryption of a recovered plaintext, an intermediary plaintext may be stolen during the re-encryption. In this paper we examine secure digital distribution systems, information storage system and information provider system, in which encrypted information is directly transformed into a ciphertext of an admissible user. We show that the technique of a proxy cryptosystem is useful for establishing these distribution systems. Proposed protocols can be constructed base on the ElGamal cryptosystem or the RSA cryptosystem. Meanwhile, a blind decryption protocol provides privacy protection with respect to the selection of a ciphertext to be decrypted. In terms of digital distribution it also provides a secure information delivery. An information provider system using a blind decryption protocol possesses a problem such that a decrypting person computes exponentiation for a message freely selected by a requesting person. For such an oracle problem, a solution is known with use of a transformable signature. In this paper we show another measure prohibiting the abuse of the blind decryption protocol.

1 Introduction

One of the greatest advantages of an open network, e.g. Internet, is that people can obtain a large amount of information all over the world. In particular, information created at a local place can be easily read by people living at very far distance. In such a network digitized information is distributed widely and freely. During distribution the digital information passes through several sites, and it is sometimes locally stored in some of these sites. Since the open network is a decentralized insecure computer network, the digital information transmitted over the network and locally stored information should be protected from external threats like eavesdropping and illegal access. In addition to the external

threat, we should be careful about internal threats. For example, information stored in a storage of an organization may be copied by a malicious employee, and leaked out of the organization. It is a very important subject to achieve security of digital information in a distribution system over the open network.

Of course, cryptography is an effective means for protecting digital information from unauthorized access. As long as a cryptosystem used is not broken, the information is not read by anyone other than that knows a secret. Encrypted information can be securely transmitted, and put in any storage with keeping its secrecy. However, the direct use of cryptography is not good enough because a recipient of ciphertext is often not known in advance in the information distribution to the public. An information provider may collect interesting information, and try to sell it. Collected valuable information is encrypted, but since its recipient is not known at the time of encryption, the encrypted information has to be converted when its user is determined. The information provider may decrypt its ciphertext and successively re-encrypt a recovered plaintext for the user. Then there is a threat such that a malicious employee and an external entity try to obtain the intermediary plaintext. Naturally speaking, a creator of digital information wants to hide his information until it arrives at a legitimate end user. Therefore, a secure information distribution mechanism which is appropriate for the open network should be studied.

In this paper we mainly deal with two types of distribution systems. One is an information storage system and the other is an information provider system.

Information storage system: It costs to keep material products in a storage. Likewise, digital information needs disk space, and one needs to spend a certain amount of money for a storage facility. Hence, it is imaginable that information storage business emerges. A small company which lacks of funds for a storage equipment deposits its own data in a storage keeper, and withdraws the data from time to time over a network. In order to prepare for a disaster, e.g. earthquake or fire, even a large company may use this type of service as a backup of huge amount of data. If the amount of data a company needs to keep drastically fluctuates, the company can receive large benefit from the storage service by renting an adequate amount of disk space in each period. The demand for storage business will definitely increase if the communication cost becomes lower than the storage cost. In such a business, the owner of the digital information wants to hide deposited information from the keeper. Moreover, if a company retrieves information on demand of a user as in the travel agency and in the weather forecast company, it needs to show the same information to many users. That means it has to convert the information into a form accessible to admissible users.

The information storage system is expected to have a great effect not only as a business between companies but also inside a company. A company has a storage section, and files of different sections are stored in the section. It is useful both as a back up and as normal storage. Since data is gathered in one section, messages should be protected from steal by dishonest employees.

Information provider system: One of important problems in the open network is how to find useful information from large amount of data in the network. Without doubt, we can show our information to the public more easily than before the open network has been built. But it is still not an easy matter to find necessary information from the public. Therefore, one can conduct a business by collecting and providing information over the network. Superdistribution [Mori90], which is a concept on a software distribution system such as a software company can charge a user for each use of a software, also has to overcome a problem of providing information attractive enough to users. Although an electronic news system or a share-ware program system offers us an opportunity to find useful information, it does not ensure the security of posted information. Hence, we study a secure information provider system by taking an electronic news system, a newspaper system and a software distribution system as examples.

We show that the proxy cryptosystem [MO97] is quite effective in constructing secure distribution systems described above. Concrete distribution protocols can be constructed based on either the RSA cryptosystem [RSA84] or the ElGamal cryptosystem [ElG85].

In the meantime, a blind decryption system [SY96, MSO96] provides privacy protection with respect to the selection of a ciphertext to be decrypted. In terms of digital distribution it also provides a secure information delivery. But a blind decryption based on ElGamal cryptosystem has a problem such that a decrypting person computes exponentiation for any message selected by a requesting person. This type of protocol abuse is generally called an oracle problem. A person participating in a cryptographic protocol is exploited by an enemy, and plays like an oracle to the enemy. The oracle problem in the ElGamal blind decryption is solved in [MSO96] by utilizing the transformability of the ElGamal signature. In this paper we show a new solution to this problem.

After the introduction, related work is shown in Sect.2. Then two types of distribution systems are described in Sect.3. Protocol 1 in Sect.3.2 is an information storage system, and Protocol 2, Protocol 3 and Protocol 4 in Sect.3.3 are information provider systems. Protocol 4 uses a blind decryption protocol with a protective mechanism against the abuse by users. Finally, conclusion is given in Sect.4.

2 Related work

Authorization in open networks has been paid a great attentions for years, and there are much work related to it, e.g. [NS78, VAB91, Neu93]. In their schemes a user or a client computer proves that he or it is a really authorized person or computer. Once a check of user verification is passed, he can receive a service from a server. We can consider our systems are one kind of authorization protocol. Even so, our protocols are more devoted to the authorization of the power to decrypt ciphertext and delegated keys than that to prove an identity of a user.

Information provider systems are shown in [TIY95]. One of their systems, a temporary-type system, offers a user a way to use information in on-line basis, e.g. video on demand. In another system, a permanent-type system, encrypted information is delivered in advance to a user via MD, CD-ROM and so on. In both systems a session key is exchanged before information is decrypted. In our proposed systems the session key exchange is not carried out in each transmission. Instead, encrypted information is converted into a form a receiver can read.

In [OT94] an on-line shopping system protecting privacy is proposed, where three sections, customer section, intermediary section and commodity section, are established in a catalog sales company. Who buys which goods is kept unknown as long as not more than two sections collude. In our distribution systems, we do not need to consider several sections in the same company. The privacy is protected in our schemes in a sense such that a keeper cannot know the content of retrieved information in the information storage system and such that an intermediary company does not know the content of transferred information in one of the information provider systems.

Proxy decryption(Proxy decoding): The proxy decryption is a process in a proxy cryptosystem. The proxy cryptosystem [MO97] is a method by which an original decryptor D can allow a designated proxy decryptor Pxy to decrypt a ciphertext of the original decryptor. A proxy ρ is preliminarily given from the original decryptor to the proxy decryptor through a secure channel, and when the original decryptor wants to delegate the decrypting operation to the proxy decryptor, the original decryptor transforms its ciphertext $C^{(D)}$ into a ciphertext $C^{(Pxy)}$ for the proxy decryptor. Using ρ the proxy decryptor extracts a plaintext m from $C^{(Pxy)}$. This decryption by the proxy decryptor is called proxy decryption. Even the proxy decryptor given ρ cannot compute a secret s_D of the original decryptor. A process offering the same functionality as the proxy cryptosystem can be performed by first decrypting a ciphertext $C^{(D)}$ and then re-encrypting an obtained plaintext under the proxy decryptor's public key. This obvious re-encryption method is not efficient enough, and more efficient methods are shown in [MO97] based on the ElGamal cryptosystem or the RSA cryptosystem.

Blind decryption(Blind decoding): The blind decryption is a process by which a user U possessing a ciphertext $C^{(C)}$ of other user or other organization, say a company C, makes C decrypt $C^{(C)}$ without telling C which ciphertext U tries to decrypt. At the same time, C can keep hiding its secret value s_C from U. In contrast to the blind signature [Cha85] where a digital signature is created for a document unknown to a signer, the decrypting operation is executed for a document unknown to a decryptor. The blind decryption is useful for protecting privacy in software distribution in the following way. A software product is encrypted, and the key is also encrypted by C's public key. Then a set of pairs of an encrypted program and a key is delivered to a user. The user selects a favorite program and recovers it without showing which program he wants to obtain by conducting the blind decryption protocol with C. The blind decryption tech-

nique has already been used in the fair public-key cryptosystem [Mic92] based on the RSA cryptosystem and in [SY96] based on the ElGamal cryptosystem. The scheme shown in [SY96] can be estimated as a refinement of the Massey-Omura cryptosystem described in [Kob87] in the ElGamal cryptosystem. The Massey-Omura cryptosystem is also called Shamir's three-pass message transmission scheme. The scheme in [SY96] ensures secure message transmission to a receiver, either.

3 Proposed Digital Distribution Systems

3.1 General framework

In proposed systems digital information is encrypted by a secret-key cryptosystem and a ciphertext $\mathcal{C}$ is created. We can select any favorite secret-key cryptosystem. A receiver of digital information is informed what kind of secret-key cryptosystem is used, either in advance or in each transmission. The key used in the secret-key cryptosystem is randomly generated, and it is processed by the proxy cryptosystem or the blind decryption system, which is based on a public-key cryptosystem. In order to simplify the description, we do not particularly mention authentication methods of messages or users. At the same time, we do not specify a payment protocol, a receipt protocol and a simultaneous bit exchange protocol throughout the paper. Since these protocols are important by their own, we discuss them in another occasion.

3.2 Information storage system

In this subsection two different frameworks of information storage system are described. In the first framework a company C possessing data uses information storage service conducted by a keeper K. K can be an independent company or a section of C. In response to a request from a user U, C retrieves requested information from K, and sends it back to U after converting it in a form such that only U can read. U is given a proxy ρ for decryption in advance. The following protocol is constructed under such a framework. This protocol is based on the ElGamal cryptosystem. Denote by v_C and s_C a public key and a secret key of C, respectively. In the ElGamal cryptosystem $v_C = g^{s_C} \bmod p$, where p is a prime, g is a generator of Z_p^* and $s_C \in_R Z_{p-1}^*$. If we prepare a generator of Z_q^* s.t. $q|p-1$, we can construct the similar protocol using q in place of $p-1$ as in [Schn91].

Protocol 1

Step 0. (Preliminary)

Step 0-1. (Storage) C creates $\mathcal{C}$ by encrypting its data under a randomly generated key m, and encrypts m under v_C. Then $((g^r \bmod p, m v_C^r \bmod p), \mathcal{C})$, where $r \in_R Z_{p-1}$, together with a ciphertext number is sent to K. The ciphertext number is used for specifying a corresponding ciphertext among C and K.

Step 0-2. (Proxy delivery) C selects a random number $u \in_R Z^*_{p-1}$, and computes $\rho = us_C \bmod p-1$. ρ is given to U through a secure channel. This step needs to be done only once, for example when U registers himself for C's service. C computes $u^{-1} \bmod p-1$ and keeps the result in a database with the name of the corresponding registered user.

Step 1. (User's request) U asks C to send data he wants to look at. The choice is made after checking a list of stored information.

Step 2. (Retrieval request) C sends K a ciphertext number corresponding to the data he should return to U.

Step 3. (Return from keeper) After receiving the ciphertext number, K returns $((x_1, x_2), C)$, where $x_1 = g^r \bmod p$ and $x_2 = mv_C^r \bmod p$. If C has requested to hide the correspondence of a ciphertext in Step 0-1 and that in this step, K returns $((x_1, x_2), C) = ((g^{r+k_K} \bmod p, mv_C^{r+k_K} \bmod p), C)$ after computing $g^r g^{k_K} \bmod p$ and $(mv_C^r)v_C^{k_K} \bmod p$ for $k_K \in_R Z_{p-1}\backslash\{0\}$.

Step 4. (Transformation and return from company) C looks for $u^{-1} \bmod p-1$ of U in the database. C transforms the received (x_1, x_2) into $(x_1^{(u^{-1} \bmod p-1)} \bmod p, x_2)$ and sends $(y_1, y_2, C) = ((x_1^{(u^{-1} \bmod p-1)} \bmod p, x_2), C)$ to U. If C wants to hide correspondence of a ciphertext in Step 3 and that in this step, C sends $(y_1, y_2) = (((x_1 g^{k_C})^{(u^{-1} \bmod p-1)} \bmod p, x_2 v_C^{k_C} \bmod p), C)$ to U using $k_C \in_R Z_{p-1}\backslash\{0\}$.

Step 5. (Decryption by user) After receiving $((y_1, y_2), C)$, U obtains the key m by computing $y_2/y_1^{\rho} \bmod p \equiv m$. Using m, U decrypts C.

If the database in Step 0-2 becomes very large, C should also put the data stored in its database into K's storage in an encrypted form. In this case, U and C need extra time for retrieving data.

Discussion: Since information K stores is a ciphertext of C, K knows nothing on the plaintext. K does not know who retrieves the information, either. If C wants to check whether K has returned correct information, C should decrypt (x_1, x_2) returned from K in Step 3. As long as a secret of C is not stolen by a malicious employee, information delivered to a user is not read even in the intermediary state because a ciphertext keeps an encrypted form during the distribution.

Instead of returning (y_1, y_2, C) directly to U, C can send this triplet to U via K. In this case K takes full responsibility for the information delivery. K has to do retransmission when U claims message is not delivered.

The proxy delivery in Step 0-2, the ciphertext transformation in Step 4 and the proxy decryption in Step 5 are based on the steps of the proxy cryptosystem. Protocol 1 can also be constructed based on the RSA cryptosystem. Please refer to [MO97] for the proxy cryptosystem based on the RSA cryptosystem.

In the above framework users receiving information do not belong to the company. In contrast, every transactions are conducted inside a company C in the following framework. C organizes a storage service in its keeping section K, and employees of C are users of the information storage service. In this framework

a user U of the storage service encrypts his information by his own public key v_U. Encrypted information is transmitted to K, and stored in it. Upon request from U, K returns encrypted information to U. Inside a company employees may be required to share some of their information with keeping security against external and internal threat. Essentially, we can use the method in Protocol 1 for the secure data transmission. Suppose a user U_1 with a public key v_{U1} wants to share with another user U_2 information stored in K. Step 4 and Step 5 of Protocol 1 are performed by U_1 and U_2, respectively. That is, U_1 with a secret key s_{U1} transforms $(x_1, x_2(= mv_{U1}^r \bmod p))$ into $(x_1^{(u^{-1} \bmod p-1)} \bmod p, x_2)$ and sends $(y_1, y_2, C) = ((x_1^{(u^{-1} \bmod p-1)} \bmod p, x_2), C)$ to U_2 possessing $\rho = us_{U1} \bmod p-1$. U_2 computes $y_2/y_1^{\rho} \bmod p \equiv m$.

U_2 can share her information with U_1 without securely delivering another proxy to U_1. U_2 can use the proxy ρ given by U_1. U_2 calculates $(x_1^{\rho s_{U2}} \bmod p, x_2(= mv_{U2}^r \bmod p))$ and sends $(y_1, y_2, C) = ((x_1^{us_{U1}s_{U2}} \bmod p, x_2), C)$ to U_1. U_1 can obtain m by computing $y_2/y_1^{(u^{-1}s_{U1}^{-1} \bmod p-1)} \bmod p \equiv y_2/y_1^{(\rho^{-1} \bmod p-1)} \bmod p \equiv m$. If such a data delivery frequently occurs, U_1 should keep $\rho^{-1} \bmod p-1$ in his own database.

3.3 Information provider system

In open networks, there should be sites collecting information, then people can relatively easily find necessary information without browsing around the network. We discuss three information provider systems. As the first system, we examine how to bring encryption into a news system mentioned in the introduction. In this system each user can get information from a site he belongs to. In other two systems a company C plays the role of a site collecting information. In the latter two systems a person who releases information to the site of C either allows C to get access to the information after a conversion process, or directly executes a decryption protocol with a user U without allowing C to read the information.

A proposed news system employs the method shown in Protocol 1. There are sites $S_1, S_2, \ldots$ of the news system. A user U_1 posts a news to a site, say S_1, to which she connects. The posted news is distributed to all of relevant sites, and retrieved by a user U_2 from a site, say S_3, to which he connects. Let $s_{Si} \in_R Z_{p-1}^*$ and $v_{Si}(= g^{s_i} \bmod p)$ be secret and public keys of S_i, respectively.

Protocol 2

Step 0. (Preliminary)

Step 0-1. (Proxy delivery between sites) Each site shares a proxy with each sites it connects to. S_i prepares a proxy $\rho_{SiSj}(= u_{Sj}s_{Si} \bmod p-1)$ as described in Protocol 1 and gives it to S_j through a secure channel. S_i securely stores u_{Sj} and $\rho_{SiSj}^{-1} \bmod p-1$, and S_j securely stores ρ_{SiSj} in a database with the name of the corresponding site. Suppose there are connections between S_1 and S_2 and between S_2 and S_3, then S_1 possesses u_2 and $\rho_{S1S2}^{-1} \bmod p-1$, S_2 possesses ρ_{S1S2}, u_3 and $\rho_{S2S3}^{-1} \bmod p-1$, and S_3 possesses ρ_{S2S3} in their databases.

Step 0-2. (Registration of users) When a user U_k wants to participate in the news system, she obtains a proxy from one of sites, say S_t. S_t prepares $\rho_{StUk}(= u_{Uk}s_{St} \bmod p-1)$ as described in Protocol 1 and gives it to U_k through a secure channel. S_t and U_t securely store u_{Uk} and ρ_{StUk} in a database with the name of the corresponding site or user, respectively. Suppose U_1 and U_2 connects with S_1 and S_3, respectively. Then U_1 and U_2 store ρ_{S1U1} and ρ_{S3U2} in the database, respectively. S_1 and S_3 store u_{U1} and u_{U2} in the database, respectively.

Step 1. (Posting) $U_1 \longrightarrow S_1$: U_1 creates C by encrypting a part or whole of her article under a randomly generated key m, and encrypts m under a public key v_{S1} of S_1 to which she connects. Then $((x_1, x_2), C) = ((g^r \bmod p, mv_{S1}^r \bmod p), C)$, where $r \in_R Z_{p-1}$, is delivered to S_1.

Step 2. (Distribution) S_1 distributes the posted information by converting it to a ciphertext for its neighbor sites. Other sites also execute conversion successively.

Step 2-1. $S_1 \longrightarrow S_2$: S_1 computes $y_1 = x_1^{(u_{S_1}^{-1} \bmod p-1)} \bmod p$, and sends $((y_1, y_2(= x_2)), C)$ to S_2.

Step 2-2. $S_2 \longrightarrow S_3$: S_2 computes $\alpha_1 = y_1^{(\rho_{S1S2}\rho_{S2S3}^{-1} \bmod p-1)} \bmod p \equiv g^{(ru_{S_2}^{-1}u_{S_2}s_{S_1}\rho_{S2S3}^{-1} \bmod p-1)} \bmod p \equiv g^{(rs_{S1}\rho_{S2S3}^{-1} \bmod p-1)} \bmod p$, and sends $((\alpha_1, \alpha_2(= y_2)), C)$ to S_3.

Step 3. (Retrieval) $S_3 \longrightarrow U_2$: When a user wants to read encrypted posted information, he sends a request to a site he connects to. Upon request from U_2, S_3 converts a ciphertext $((\alpha_1, \alpha_2, C)$ into $((\beta_1(= \alpha_1^{(\rho_{S2S3}u_{U_2}^{-1} \bmod p-1)}), \beta_2(= \alpha_2)), C)$. Created $((\beta_1, \beta_2), C)$ is delivered to U_2.

Step 4. (Decryption by user) U_2 obtains the key m by computing $\beta_2/\beta_1^{\rho_{S3U2}} \bmod p \equiv m$. Using m, U_2 can decrypt C.

In order not to expose the plaintext, exponents in Step 2-2 and Step 3 must be computed at first. In Step 0-1, S_1 and S_2 have created proxies. In place of these sites it is possible its partners S_2 and S_3 create proxies. Basically, sites having many neighbor sites should create a proxy and give the same proxy to its neighbor sites. Then these sites do not need to perform conversion many times for the same ciphertext.

In the following protocol a free reporter R writes articles and asks a newspaper company C to buy them. These articles are basically encrypted, and only their headline can be read. C buys the article from R when there is at least one request for purchase, or there are enough amount of requests for purchase. Once C buys the article, C can sell it to subscribers who have requested to read them. Denote by (v_i, s_i) a pair of public and secret keys of $i \in \{C, R, U\}$. $v_i = g^{s_i} \bmod p$, where $s_i \in_R Z_{p-1}^*$.

Protocol 3

Step 0. (Preliminary)

Step 0-1. (Collection) R creates a program and partially encrypts it by a secret-key cryptosystem under a randomly generated key m. A computed ciphertext is C. R also encrypts the key m using v_C. $(x_1, x_2) = (g^r \bmod p, mv_C^{s_R r} \bmod p)$, where $r \in_R Z_{p-1}$. Then $((x_1, x_2), C)$ is sent to C.

Step 0-2. (Proxy delivery) C selects a random number $u \in_R Z_{p-1}$ and gives u to U as a proxy $\rho(= u)$ through a secure channel. This step needs to be done only once, for example when U becomes a subscriber of C's newspaper. C computes $u - s_C \bmod p - 1$ and keeps the result in a database with the name of the corresponding registered user.

Step 0-3. (Partial information retrieval) U sees a headline of news of C and selects a program he wants to read.

Step 1. (User's request) U requests an article he wants to read.

Step 2. (Request for blind decryption) C sends x_1 of the requested article to R.

Step 3. (Return from programmer) After receiving x_1, R computes $y_1 = x_1^{s_R} \bmod p (\equiv g^{s_R r} \bmod p)$, and returns y_1 to C.

Step 4. (Transformation) C receives y_1. $(\alpha_1, \alpha_2) = (y_1, x_2) = (g^{s_R r} \bmod p, mv_C^{s_R r} \bmod p)$ is a ciphertext of C. C can decrypt it if C wants. If C needs to hide the correspondence of the blind decryption and the article bought from R, R gives $((g^{s_R(r+k_C)} \bmod p, mv_C^{s_R(r+k_C)} \bmod p), C)$ to U after computing $\alpha_1 v_R^{k_C} \bmod p$ and $\alpha_2 v_R^{s_C k_C} \bmod p$ for $k_C \in_R Z_{p-1} \backslash \{0\}$.

C calculates $\beta_2 = \alpha_2 \alpha_1^{(u-s_C)} \bmod p \equiv mv_R^{ru} \bmod p$, and $((\beta_1, \beta_2), C) = ((\alpha_1, \beta_2), C) = ((v_R)^r \bmod p, mv_R^{ru} \bmod p), C)$ is delivered to U.

Step 5. (Decryption by user) U obtains the key m by computing $\beta_2/\beta_1^{\rho} \bmod p \equiv m$. Using m, U can decrypt C.

Discussion: In this protocol C can compute m. But it is difficult for a third party to extract it from communicated messages. R does not fail to get money for his article because C cannot decrypt articles without executing Step 2 and Step 3. These steps can incorporate the blind decryption technique if C wants to hide its choice of article. Meanwhile, as easily observed, information provider system can be executed together with the information storage system.

In the next protocol we exemplify the information provider system by a program distribution. Distributed programs are created by programmers P, and encrypted. A user U registered to C chooses a program from a program list. When U decides to buy a program, U executes a blind decryption protocol [SY96, MSO96] with P. Then he has permission for the use of the program. In this protocol C charges P for the use of its web site, and does not participate in the blind decryption protocol.

As studied in [MSO96] the blind decryption based on ElGamal cryptosystem has a problem that a decrypting person P computes $\sigma^{s_P} \bmod p$ for any message σ selected by a requesting person. Protocol 3 also has such an oracle problem, where a decrypting person is R. This problem is solved in [MSO96] by utilizing the transformability of the ElGamal signature. In this paper we show another solution to the oracle problem. In the following protocol P with a public and secret key pair $(v_P (= g^{s_P} \bmod p), s_P \in Z^*_{p-1})$ prepares a random number t_P such that $t_P \in_R Z^*_{p-1}$ and $t_P \neq s_P$. Then P computes $T_P = g^{t_P} \bmod p$. T_P and t_P are public and secret keys of P additional to v_P and s_P. $h(\cdot)$ is a cryptographically secure hash function.

Protocol 4

Step 0. (Preliminary)

Step 0-1. (Collection) P creates a program and partially encrypts it by a secret-key cryptosystem under a randomly generated key m. A computed ciphertext is C. P also encrypts the key m using v_P and T_P. $(x_1, x_2, x_3) = (g^r \bmod p, m v_P^r \bmod p, m T_P^r \bmod p)$, where $r \in_R Z_{p-1}$. P computes the following signature s. After selecting $w \in_R Z_{p-1}$, $h(g^w \bmod p, (v_P/T_P)^w \bmod p) = e$ is generated. s is determined by $s = w - er \bmod p - 1$. $((x_1, x_2, x_3, x_4, x_5), C)$ is sent to C, where $x_4 = s$ and $x_5 = e$.

Step 0-2. (Retrieval of encrypted information) U watches a list of programs exhibited by C and selects a program he needs. U downloads $((x_1, x_2, x_3, x_4, x_5), C)$ from C. He checks the verification equation $x_5 = h(g^{x_4} x_1^{x_5} \bmod p, (v_P/T_P)^{x_4} (x_2/x_3)^{x_5} \bmod p)$. If it is not satisfied, he requests a valid triplet to C. If satisfied, he proceeds to Step 1.

Step 1. (Request for blind decryption) U chooses $a \in_R Z^*_{p-1}$, and computes $(y_1, y_6) = (x_1^{(a^{-1} \bmod p-1)} \bmod p, (x_2/x_3)^{(a^{-1} \bmod p-1)} \bmod p)$
$\equiv (g^{(ra^{-1} \bmod p-1)} \bmod p, g^{(ra^{-1}(s_P - t_P) \bmod p-1)} \bmod p)$. (y_1, y_6) is sent to P. a is kept secret by U.

Step 2. (Return from programmer) P first checks whether a congruence $y_1^{(s_P - t_P)} \equiv y_6 \bmod p$ satisfies or not. If the check fails, P does not respond. Otherwise, P computes $z_1 = y_1^{s_P} \bmod p$, and returns z_1 to C.

Step 3. (Decryption by user) After receiving z_1, U computes $\alpha_1 = z_1^a \bmod p$. Then he has $(\alpha_1, \alpha_2) = (\alpha_1, x_2)(\equiv (v_P^r \bmod p, m v_P^r \bmod p))$. U obtains the key m by calculating $\alpha_2 \alpha_1^{-1} \bmod p \equiv m$. Using m, U decrypts C.

Discussion: Protocol 4 provides a secure message delivery to a receiver. The receiver can be determined after encryption. As stated in Sect.2, the protocol described above is closely related to the Shamir's three-pass message transmission scheme.

Because of the blind decryption protocol, U can hide which program he is going to buy. Nonetheless, P does not fail to get money for his software product by charging each execution of the blind decryption protocol.

As in information storage systems, communicated messages are kept in an encrypted form. Thus it is similarly difficult for C and a third party to extract a message from the communicated messages.

Meanwhile, it is considered to be difficult to create a pair (y_1, y_6) satisfying the verification congruence in Step 2 for a selected y_1 without knowing the secrets s_P and t_P. Therefore, P is unlikely to compute exponentiation for a message selected by U. In Protocol 4 two public keys v_P and T_P are used for detecting the abuse of the blind decryption protocol. A similar technique is used in the modified ElGamal cryptosystem shown in [Dam91].

P may try to relate a requested pair (y_1, y_6) with ciphertexts he has placed on the site of C by choosing his secret t_P as an output of a function f taking a message m as input. By checking $y_1^{(s_P - f(m))} \equiv y_6 \bmod p$ for a message m, he can find whether the requested pair corresponds with the message m. In this sense, privacy is slightly violated. However, if t_P has only one preimage, he cannot checks more than two messages because T_P compatible with P's secret t_P is made public. In case t_P has many preimages m's, he can check the correspondence to many messages. But he cannot know the precise correspondence among many messages. From this observation we believe the proposed method offers an acceptable level of privacy.

We can apply the countermeasure in Protocol 4 to Protocol 3. In this case, Protocol 3 are modified as follows. In Step 0-1, $(x_1, x_2, x_3, x_4, x_5) = (g^r \bmod p, mv_C^{s_R r} \bmod p, mv_C^{tr} \bmod p, w - er \bmod p - 1, e)$ is delivered to C. C verifies it by checking $x_5 = h(g^{x_4} x_1^{x_5} \bmod p, (v_R/T_R)^{x_C x_4} (x_2/x_3)^{x_5} \bmod p)$. In Step 2, C sends $(\widetilde{x_1}, \widetilde{x_2}) = (x_1^{v_C(a^{-1} \bmod p-1)} \bmod p, (x_2/x_3)^{(a^{-1} \bmod p-1)} \bmod p)$ to R. Verification of $(\widetilde{x_1}, \widetilde{x_2})$ is executed in Step 3 as in Step 2 of Protocol 4. Only when verification is passed, R proceeds to the next step.

4 Conclusion

In this paper we have studied several information distribution systems, information storage system and information provider system, which keep the secrecy of transmitted data and delegated keys in open networks. These distribution systems are based on either the proxy cryptosystem, the blind decryption system or both of them. The presented information provider systems can be combined with the information storage system explained in Protocol 1. Because digital information preserves an encrypted form in an intermediary organization in the proxy cryptosystem and communicated messages are independent of a ciphertext to be decrypted in the blind decryption system, digital information is not read until it arrives at an end user as long as secret keys are not compromised.

We have also shown a new method to prevent the abuse of the blind decryption protocol. By this new method users cannot obtain an exponentiated value for a message they have selected. Privacy of users participating in this protocol is preserved at an acceptable level.

References

[Dam91] I. Damgård: "Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks," Lecture Notes in Computer Science 576, Advances in Cryptology -Crypto '91, Springer-Verlag, pp.445-456 (1992).

[Cha85] D. Chaum: "Security without Identification: Transaction System to make Big Brother Obsolete," Communications of the ACM, Vol.28, No.10, pp.1030-1044 (Oct. 1985).

[ElG85] T. ElGamal: "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithm," IEEE Trans. on Information Theory, Vol.IT-31, No.4, pp.469-472 (Jul. 1985).

[Kob87] N. Koblitz: A Course in Number Theory and Cryptography, GTM 114, Springer-Verlag (1987).

[MO97] M. Mambo and E. Okamoto: "Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts," IEICE Transactions on Fundamentals, Vol.E80-A, No.1, pp.54-63 (Jan. 1997).

[MSO96] M. Mambo, K. Sakurai and E. Okamoto: "How to Utilize the Transformability of Digital Signatures for Solving the Oracle Problem," Lecture Notes in Computer Science 1163, Advances in Cryptology -Asiacrypt '96, Springer-Verlag, pp.322-333 (1996).

[Mic92] S. Micali: "Fair Public-Key Cryptosystems," Lecture Notes in Computer Science 740, Advances in Cryptology -Crypto '92, Springer-Verlag, pp.113-138 (1993).

[Mori90] R. Mori: "Superdistribution: The Concept and the Architecture," The Proc. of The 1990 Symposium on Cryptography and Information Security, SCIS90-6A (Jan. 1990).

[NS78] R. M. Needham and M. D. Schroeder: "Using Encryption for Authentication in Large Networks of Computers," Communications of the ACM, Vol.21, No.12, pp.993-999 (Dec. 1978).

[Neu93] B. C. Neuman: "Proxy-Based Authorization and Accounting for Distributed Systems," Proc. of the 13th International Conference on Distributed Computing Systems, pp.283-291 (May 1993).

[OT94] M. Ohmori and M. Tatebayashi: "An On-line Shopping System Protecting User's Privacy," IEICE Technical Report Vol.94, IT94-66, ISEC94-26, pp.25-32 (1995). [in Japanese]

[RSA84] R. L. Rivest, A. Shamir and L. Adleman: "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Communications of the ACM, Vol.21, No.2, pp.120-126 (1978).

[Schn91] C. P. Schnorr: "Efficient Signature Generation by Smart Cards," Journal of Cryptology, Vol.4, No.3, pp.161-174 (1991).

[TIY95] Y. Takashima, S. Ishii and K. Yamanaka: "An Intellectual Property Protection System Using a PCMCIA Card," Proc. of The 1995 Symposium on Cryptography and Information Security, SCIS95-B5.5 (Jan. 1995). [in Japanese]

[VAB91] V. Varadharajan, P. Allen and S. Black: "An Analysis of the Proxy Problem in Distributed Systems," Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pp.255-275 (May 1991).

[SY96] K. Sakurai and Y. Yamane: "Blind Decoding, Blind Undeniable Signatures, and their Applications to Privacy Protection," Lecture Notes in Computer Science 1174, Information Hiding, Springer-Verlag, pp.257-264 (1996).

IEICE TRANS. FUNDAMENTALS, VOL. E80-A, NO. 1 JANUARY 1997

PAPER Special Section on Cryptography and Information Security

Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts

Masahiro MAMBO[†] and Eiji OKAMOTO[†], Members

SUMMARY In this paper a new type of public-key cryptosystem, proxy cryptosystem, is studied. The proxy cryptosystem allows an original decryptor to transform its ciphertext to a ciphertext for a designated decryptor, proxy decryptor. Once the ciphertext transformation is executed, the proxy decryptor can compute a plaintext in place of the original decryptor. Such a cryptosystem is very useful when an entity has to deal with large amount of decrypting operation. The entity can actually speed-up the decrypting operation by authorizing multiple proxy decryptors. Concrete proxy cryptosystems are constructed for the ElGamal cryptosystem and the RSA cryptosystem. A straightforward construction of the proxy cryptosystem is given as follows. The original decryptor decrypts its ciphertext and re-encrypts an obtained plaintext under a designated proxy decryptor's public key. Then the designated proxy decryptor can read the plaintext. Our constructions are more efficient than such consecutive execution of decryption and re-encryption. Especially, the computational work done by the original decryptor is reduced in the proxy cryptosystems.

key words: proxy cryptosystem, proxy, proxy decryptor, ciphertext transformation

1. Introduction

Digitized information in computer networks is copied very easily. If information is encrypted, one cannot obtain the content from the copied message. Additionally, if an access control mechanism is employed with the use of user authentication protocol, an inadmissible entity cannot even make an access to digital information. Cryptography related techniques, typically cryptography itself in the former application, provides us an effective way to limit users who makes an access to digital information. One of cryptographies, public-key cryptography, which has been intensively studied after the advent of [4], is useful in an open network for transmitting information only to a specified person. In the public-key cryptography a key v is made public while a compatible key s is kept secret by its owner. This is why it is suitable for the open network. It is hard in a computational complexity sense to determine a secret s even when an attacker knows a public value v. In the similar context, if there exits a pair (s, ρ), where it is hard to compute s even with the knowledge of ρ, and a possessor of ρ can decrypt a ciphertext which is originally encrypted under v, then such a pair can be used for authorizing a user for getting an access to the encrypted digital information. Such delegation is required in the following occasions.

Suppose an organization, e.g. research company or government, plans to conduct a survey, and one section is assigned to this job. From the privacy reason, questionnaires returned from people are encrypted under a public key v of the organization, or if the questionnaire is very long, it should be encrypted by a secret-key cryptography and a key used in the secret-key cryptography is encrypted by a public-key cryptography. A president of the organization wants members of a section to decrypt the questionnaires and to analyze the survey. But he has no intention to show a secret s of the organization to them. Because it breaches the security assumption of the public-key cryptography. Moreover, the president should limit the access only to the members of the section in order to keep the user's privacy. The president may decrypt the ciphertexts and simply transmit the decrypted questionnaires to the members. Then the privacy could be violated. So, the president should encrypt the decrypted questionnaires under a public key of the members before the transmission. However, an attacker may try to see decrypted intermediate plaintexts. Such a threat is not totally overcome in this re-encryption approach. On top of that, the processes in this approach is a bit cumbersome, and a more direct and efficient method to allow a new access should be studied.

Other example is file access. Suppose a user has a file in a publicly reachable directory or in a directory which is private but possibly illegally accessed by others. In order to keep the secrecy of the file she encrypts it under her public key, or as mentioned above, both the secret-key cryptography and the public-key cryptography are used to encrypt the file. Then the access to the file is limited to her. In some occasion, the user wants to temporarily permit other user to access to the file. She first transforms the encrypted file to the form such that only a new user can read it, and after some time, the original owner revokes the access permission. Like in the above example, such a transformation is achieved by the combination of decryption and re-encryption, but we should avoid such a transformation from the following reason. A file handling system or an editor program often offers us an automatic logging system. Under such

Manuscript received March 25, 1996.
Manuscript revised July 15, 1996.

[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-12 Japan.

circumstances, even if we remove a decrypted plaintext, we may forget to remove its backup file. That indicates we are not totally sure whether an attacker obtains no information at all as long as we recover a plaintext from a ciphertext in a transit point. Furthermore, it is preferable that the transformation be as simple as possible.

In this paper a cryptosystem called *proxy cryptosystem*, which is well suited to the situations above, is studied. The proxy cryptosystem allows an original decryptor to delegate its decrypting operation to a designated decryptor, *proxy decryptor*. Concrete proxy cryptosystems are presented for the ElGamal cryptosystem [5] and the RSA cryptosystem [14]. In the proposed scheme the computational work is less than that in consecutive execution of decryption and re-encryption.

This paper is organized in the following way. After this introduction, related work is explained in Sect. 2. Then following the explanation on conditions of proxy cryptosystems in Sect. 3, three concrete proxy cryptosystems are proposed in Sect. 4. In Sect. 5 the efficiency of the proposed scheme is discussed, and two kinds of revocation methods are described. In addition to that, how to deal with multiple proxy decryptors is studied. Finally conclusions are given in Sect. 6.

2. Related Work

A privacy homomorphism introduced in [13] as cited in [1] is an encryption function such that the operation of its outputs for several unknown input plaintexts results in indirect operation of these plaintexts. With the use of this encryption function, a function of plaintexts is evaluated without the knowledge of the plaintexts and a decryption function corresponding to the encryption function, and an encrypted value of the output of the function is obtained. The privacy homomorphism is useful for securely evaluating a function while hiding the plaintexts. In our situation the function evaluation is not performed, and an authorized person can obtain the plaintext as the original user can.

By a directly transformed link encryption proposed in [9], one connected to a node can securely send a message to a person connected to other node through a computer network equipping a secure data-link layer. The computer network is composed of intermediate nodes and terminals to which a user has access. The user at a terminal encrypts a message under a key of a node to which the terminal connects. The ciphertext is directly transformed in each node into other form of ciphertext encrypted under a key of the next node. A trusted center generates all original keys of nodes and terminals, and computes one key for each node from both the original key of the node and that of the next node. Each node is given only this derived key, and the original key of the node is kept secret by the center. Without transforming the encrypted data into a plaintext, direct transformation is performed in this link encryption by utilizing the derived key. Diminishing the opportunity to expose the plaintext in intermediate nodes makes the data transmission very secure. Nevertheless, this method does not deal with a situation like ours, where a user diverts his ciphertext to the other user. Moreover, all communications between users require nodes and terminals performing transformation. Since keys for the transformation used in intermediate nodes are unknown to users, a receiver of a ciphertext cannot compute a diverted ciphertext a communication partner can directly read. Instead, the receiver may compute a ciphertext which would be transferred through the network, and whose plaintext would be extracted at a communication partner side. To make this possible, the receiver of a ciphertext has to either decrypt the ciphertext and re-encrypt an obtained plaintext for the terminal, or compute a secret for ciphertext transformation from the user to the terminal and behave like a node in the network with the use of the computed secret. The latter measure is closely related to our scheme, but it has not been clearly discussed in [9]. Additionally, although the idea of avoiding to recover a plaintext in transit and the transformation performed in a node are quite similar to ours, a node cannot read a message in [9], while an original decryptor can do so in our method.

Verifiable implicit asking, e.g. in [8], or server-aided secret computation, e.g. in [17], has something to do with our topic. It has a very interesting and practical framework where a relatively powerless device executes with the assistance of powerful auxiliary device(s) a polynomial time computation which exceeds the power of the device. For example, a smart card communicates with a computational center, and it executes a large amount of computation with the aid of it. In this type of computation the powerless device converts its own secret into other values by using random numbers. The powerful device receives the converted values and performs computation for assistance. After receiving the results of the computation done by the powerful device, the powerless device does the final computation and obtains a final result. The assistant powerful device cannot compute the final result by itself because it does not know the random numbers used in conversion. If it could, it would derive the secret of the powerless device. As pointed out in [16] the powerful device sometimes finds the final result after the protocol execution. For example, a digital signature computed by server-aided computation will be known to the powerful device. Even in this case, it is not the powerful device but the powerless one that computes the final result. In the proxy cryptosystem the final result is computed by the proxy decryptor or by both the original and proxy decryptors. Hence methods for verifiable implicit asking are not appropriate enough for our situation.

Proxy signatures proposed in [7] is a tool to delegate signing operation to a designated person, proxy signer. A proxy signature for partial delegation is spe-

cially important since this type of proxy signature is more efficient than signing twice, once by an original signer and once by a proxy signer. In this signature a proxy signer cannot compute a secret of an original signer from a given proxy. Similarly a secret of an original decryptor is not computed from a given proxy in the proxy cryptosystem. The signing operation is delegated in the proxy signature while the decrypting operation is delegated in the proxy cryptosystem. The proxy cryptosystem can be combined with the proxy signature. Email is an example of such an application as described in [7].

3. Conditions of Proxy Cryptosystem

In proxy cryptosystem an original decryptor asks a proxy decryptor to carry out decryption. A ciphertext is created either by the original decryptor or by an encryptor other than the original decryptor. In the latter case, the encryptor sends the created ciphertext to the original decryptor.

In the proxy signatures[7] an original signer can determine the identity of the proxy signer who has created a given proxy signature. Even when the proxy signature is created by a person who is passed the proxy from an authorized proxy signer, the original signer considers that it has been created by the originally authorized proxy signer. Proxy signatures leave an evidence of signing operation. In this sense the proxy signer bears full responsibility on signatures created from his proxy. Unlike the proxy signatures, an original decryptor has no way to detect an illegal access to a plaintext. A decryptor to which the original decryptor has never released permission but is given a proxy can decrypt a ciphertext without endangering itself. Great care should be taken for selecting faithful proxy decryptors.

A normal cryptosystem is called a proxy cryptosystem if the following conditions are satisfied. Let $C^{(U)}$ be a ciphertext for a user U.

Conditions of proxy cryptosystem:

(i) **(Transformation)** Given a ciphertext $C^{(U)}$ for an original decryptor D, only the original decryptor or only both the original decryptor and a creator of $C^{(D)}$ can transform $C^{(U)}$ into a ciphertext $C^{(P)}$ for a proxy decryptor P.

(ii) **(Authorization)** Given a ciphertext $C^{(P)}$ of a plaintext m, m is computed either from a proxy ρ, or from information computed from ρ in polynomial time. Without this information, m cannot be polynomially extracted from $C^{(P)}$.

Although no detecting method for illegal release of a proxy has been found until now, the original decryptor can have from the condition (i) control over the access to the plaintext by the proxy decryptor or possibly by others who are given ρ or information derived from ρ. The condition (ii) ensures an original decryptor that a decryptor authorized with a proxy can decrypt a transformed ciphertext.

As mentioned in the introduction, $C^{(D)}$ can be transformed into $C^{(P)}$ by first decrypting a ciphertext and then re-encrypting an obtained plaintext under the proxy decryptor's public key. This re-encryption method satisfies the conditions of the proxy cryptosystem. Obviously the re-encryption method is not efficient, and a more efficient method should be constructed.

4. Proposed Proxy Cryptosystems

Two proxy cryptosystems for the ElGamal cryptosystem and one for the RSA cryptosystem are shown in this section. In Sects. 4.1 and 4.2, two proxy cryptosystems are constructed for the ElGamal cryptosystem. It is not impossible to construct many different forms of proxy cryptosystems for one normal cryptosystem.

4.1 Proxy Cryptosystem for ElGamal Cryptosystem

Let v and s be a public key and a private key of an original decryptor, respectively, and $v \equiv g^s \bmod p$, where $s \in_R Z_{p-1}\backslash\{0\}$. T is an additional public key of the original decryptor. $t \in_R Z_{p-1}\backslash\{0\}$ is a secret key of the original decryptor, satisfying $T \equiv g^t \bmod p$. p is a prime number whose length is taken greater than 512 bits. g is a generator for Z_p^*.

[Protocol 1]

Step 1. (Proxy generation) An original decryptor computes $\rho = s - tT \bmod p-1$, where t is randomly generated from $Z_{p-1}\backslash\{0\}$.

Step 2. (Proxy delivery) The original decryptor gives ρ to a proxy decryptor in a secure way.

Step 3. (Proxy verification) The proxy decryptor checks a congruence such that

$$v \equiv g^{\rho} T^{T} \bmod p. \tag{1}$$

If (ρ, T) passes this congruence, the proxy decryptor accepts it as a valid proxy. Otherwise, it rejects it and requests the original decryptor a valid one, or it stops this protocol.

Step 4. (Encryption) A document m is encrypted into (x, y), where $r \in_R Z_{p-1}\backslash\{0\}$, $x = g^r \bmod p$ and $y = mv^r \bmod p$.

Step 5. (Ciphertext transformation) The original decryptor transforms the (x, y) into (w, x, y), where $w = x^t (\equiv T^r) \bmod p$.

Step 6. (Decryption by proxy decryptor) The proxy decryptor computes

$$\begin{aligned} y/(x^{\rho}w^{T}) &\equiv (mv^{r})/(g^{r})^{\rho}(T^{r})^{T} \bmod p \\ &\equiv m(v/(g^{\rho}T^{T}))^{r} \bmod p \\ &\qquad \text{(from the congruence (1))} \\ &\equiv m \bmod p. \end{aligned}$$

In the application for questionnaire described in the introduction the amount of computational work of the president is reduced by the following approach.

Step 1'. (Proxy generation, delivery and verification) The original proxy decryptor, the president of a research company, generates ρ as described above, and gives the same ρ to multiple proxy decryptors, members of a section assigned to the survey, in a secure way. Each proxy decryptor checks the validity of given ρ.

Step 2'. (Encryption) A document m is encrypted into (w, x, y), where $r \in_R Z_{p-1}\backslash\{0\}$, $w = T^r \bmod p$, $x = g^r \bmod p$ and $y = mv^r \bmod p$. (w, x, y) is sent to the research company.

Step 3'. (Forwarding) Each ciphertext (w, x, y) sent to the research company is transferred without any modification to one decryptor in a group of proxy decryptors.

Step 4'. (Decryption by proxy decryptors) Based on the computation in step 6 of Protocol 1, transferred (w, x, y)'s are decrypted in parallel by multiple proxy decryptors.

Due to parallel decryption, this type of decryption is faster than decryption by a single person. Moreover, the president is exempt from performing the ciphertext transformation.

The above method does not allow the president to prohibit members of the survey section, who is not designated as a proxy decryptor, to decrypt a ciphertext. Admissible proxy decryptors in a whole group of proxy decryptors can be restricted by allowing proxy decryptor to conduct a ciphertext transformation. Further discussion is given in Sect. 5.

When a three-component ciphertext is received, the original decryptor can choose one out of three cases. Either the original decryptor decrypts the ciphertext by itself, the proxy decryptor decrypts it in place of the original decryptor or both the original and proxy decryptors decrypt it. In the first and third cases where the original decryptor decrypts the ciphertext, the security of the ElGamal cryptosystem is increased by the following procedure. The original decryptor first checks $w \equiv x^t \bmod p$, and if the check is passed, it calculates further $y/x^u \bmod p$. Otherwise, it outputs nothing. This is the exactly the procedure of the cryptosystem secure against indifferently chosen ciphertext attacks proposed in [2]. In the similar context, our approach is applicable to the cryptosystem secure against adaptively chosen ciphertext attacks described in [20]. Appendix A gives the algorithm of the original cryptosystem [20] and its modification into the proxy cryptosystem.

In stead of making T public, both the original and proxy decryptors can treat it as a secret value among them. In this case, a sender of an encrypted email cannot compute a ciphertext for the proxy decryptor alone any more, and the proxy decryptor requires assistance by the original decryptor.

Security considerations: Similar to the proxy signature scheme for partial delegation [7], the security of Protocol 1 resides in the difficulty of computing ρ satisfying a congruence $g^{\rho}T^{T} \equiv v \bmod p$, given T and v. Modified ElGamal signature schemes for a signature σ, a random number K and a public key v described in [19] and [15] are based on a congruence $g^{\sigma} \equiv vK^{\{K m \bmod q\}} \bmod p$ with q satisfying $q|p-1$ and $g^{\sigma} \equiv v^{m}K^{K} \bmod p$, respectively. The congruence for Protocol 1 can be reduced to the congruence of these modified ElGamal signature schemes for a constant message, 1. To authors' knowledge no crucial attack against these modified ElGamal signature schemes has been reported up to now.

As described in Sect. 5 different t's can be assigned to multiple decryptor. In such a case two proxy decryptors may try to find s by the following method.

Step 1. Two proxy decryptors bring their proxies, ρ_1 ($\equiv s - t_1T_1 \bmod p-1$) and ρ_2 ($\equiv s - t_2T_2 \bmod p-1$).

Step 2. i_1 and i_2 are chosen from $Z_{p-1}\backslash\{0\}$ until i_1 and i_2 are found satisfying $\rho_1 + i_1T_1 \bmod p-1 \equiv \rho_2 + i_2T_2 \bmod p-1$.

Step 3. A congruence $g^{\rho_1 + i_1T_1} \equiv v \bmod p$ is checked for i_1 obtained in the former step. If the check succeeds, true s is $\rho_1 + i_1T_1 \bmod p-1$. Otherwise, go to step 2.

Such a birthday attack is not enough effective in general because the step 2 is passed after about $1.17\sqrt{p-1}$ choices are made. If T_1 and/or T_2 have large common divisors with $p-1$, $\{(g^{T_1})^{i_1} \bmod p | i_1 \in Z_{p-1}\backslash\{0\}\}$ and/or $\{(g^{T_2})^{i_2} \bmod p | i_2 \in Z_{p-1}\backslash\{0\}\}$ become a small set, and the above attack may be feasible. In order to avoid such an attack, one should select T which does not have large common divisors with $p-1$, or T which satisfies $\gcd(T, p-1) = 1$. An alternative and better way is that p is selected such that $(p-1)/2$ is also a prime, or g is selected as $h^{\frac{p-1}{q}} \bmod p$ for a large q satisfying $q|p-1$ and a primitive root h of Z_p^*.

When multiple decryptors are involved, a proxy decryptor may attempt to compute other proxy, i.e. to compute ρ_2 from (ρ_1, T_1, T_2, v, p). This is the problem

of signature forgery pointed out above, and no serious attack is known.

(On transformation) Under the assumption on the difficulty of Diffie-Hellman problem, DH problem, it is hard to compute $w = T^r \bmod p$ from (x, y) without the knowledge of t, which is the secret of the original decryptor.

Even though the proxy decryptor has the proxy ρ, it is hard for the proxy decryptor to compute w. This is because both s and t are unknown values in a congruence $\rho \equiv s - tT \bmod p - 1$. Computing s of this congruence, in other words t, means breaking the modified ElGamal signature schemes [15], [19].

(On authorization) The proxy decryptor can extract m by following step 6 of Protocol 1. On the other hand, a third party observes only (g, T, p, w, x, y). (T, w) is additional to the ElGamal cryptosystem, and (T, x, w) is simply the values processed in the Diffie-Hellman key agreement between a user possessing x and the other possessing T. t of T is chosen independently of y and the message m so that (T, w) does not releases information on m.

4.2 Another Proxy Cryptosystem for ElGamal Cryptosystem

In this section another proxy cryptosystem applied for the ElGamal cryptosystem is shown. In the following protocol, s of v is randomly selected from Z^*_{p-1}.

[Protocol 2]

Step 1. (Proxy generation) An original decryptor computes $\rho = sd \bmod p - 1$, where d is randomly generated from Z^*_{p-1}.

Step 2. (Proxy delivery) The original decryptor gives ρ to a proxy decryptor in a secure way.

Step 3. (Encryption) A document m is encrypted into (x, y), where $r \in Z_{p-1}\backslash\{0\}$, $x = g^r \bmod p$ and $y = mv^r \bmod p$.

Step 4. (Ciphertext transformation) The original decryptor transforms the (x, y) into (w, y) or (w, x, y) where $w = x^e \bmod p$ and $ed \equiv 1 \bmod p - 1$.

Step 5. (Decryption by proxy decryptor) The proxy decryptor computes

$$\begin{aligned} y/w^\rho &\equiv (mv^r)/g^{sder} \bmod p \\ &\equiv m(v/g^s)^r \bmod p \\ &\equiv m \bmod p. \end{aligned}$$

Proxy verification process is not included in Protocol 2. If the original decryptor reveals $E = g^e \bmod p$ in step 2, the proxy decryptor can confirm that (ρ, E) satisfies $v \equiv E^\rho \bmod p$. But such a pair can be computed without the knowledge of s, see Lemma 1. Additionally E is not used for decryption in step 5. Thus the verification of proxy is not required in this protocol.

If an improper e is used in the ciphertext transformation, the proxy decryptor cannot recover a proper plaintext. The same trouble occurs in Protocol 1. The original decryptor should behave properly in ciphertext transformation.

Security considerations:

Lemma 1: The intractability of the problem of computing s given $(g^s \bmod p, g^r \bmod p, g^{er} \bmod p, \rho, p, g)$, where $r \in_R Z_{p-1}\backslash\{0\}$, $s, e, d \in_R Z^*_{p-1}$, $\rho \equiv sd \bmod p-1$ and $ed \equiv 1 \bmod p - 1$, is equivalent to that of the problem of computing s given $(g^s \bmod p, p, g)$, where $s \in_R Z^*_{p-1}$.

Proof: It is trivial that if the discrete logarithm problem is solved, the former problem is solved. The opposite reduction is proved as follows.

First select $\rho \in_R Z^*_{p-1}$. Then compute $E = (g^s)^{\rho^{-1}} \bmod p$, using the given $g^s \bmod p$ and the selected ρ. Select $r \in_R Z_{p-1}\backslash\{0\}$, and feed $(g^s \bmod p, g^r \bmod p, E^r \bmod p, \rho)$ in an oracle for solving the former problem. s is returned from the oracle and the difficulty of solving two problems is proven to be equivalent.

Lemma 2: The intractability of the problem of computing w $(\equiv g^{er} \bmod p)$ given $(g^s \bmod p, g^r \bmod p, \rho, p, g)$, where $r \in_R Z_{p-1}\backslash\{0\}$, $s, e, d \in_R Z^*_{p-1}$, $\rho \equiv sd \bmod p - 1$ and $ed \equiv 1 \bmod p - 1$, is equivalent to that of the problem of computing w given $(E \ (\equiv g^e \bmod p), g^r \bmod p, p, g)$, where $r \in_R Z_{p-1}\backslash\{0\}$ and $e \in_R Z^*_{p-1}$.

Proof: If the latter Diffie-Hellman problem [4] is solved, the former problem is solved by first computing $E = (g^s)^{\rho^{-1}} \bmod p$. Then feed $(E, g^r \bmod p, p, g)$ in a DH oracle.

If the former problem is solved, the DH problem is solved by first generating a random number $\rho \in_R Z^*_{p-1}$. Then compute $g^{e\rho} \bmod p$, and feed $(g^{e\rho} \bmod p, g^r \bmod p, \rho, p, g)$ in an oracle for the former problem.

Therefore, the difficulty of these two problems are equivalent.

(On transformation) Since d is randomly selected, e is considered to be a random number. An attacker does not know ρ. Hence, it is hard for an attacker to transform x into w.

The DH problem is regarded as hard, and it is known [3], [10] that under a certain condition it is equivalent to the discrete logarithm problem. Therefore, from the Lemma 2 even the proxy decryptor given ρ cannot compute w for ciphertext transformation.

(On authorization) The proxy decryptor can extract m by following step 5 of Protocol 2. On the other hand, a third party observes only (g, p, w, x, y). w is additional to the ElGamal cryptosystem. It is hard for

the third party to compute a correct d of the proxy decryptor corresponding to w without the knowledge of e, and the third party cannot extract m.

Unlike in Protocol 1, the proxy decryptor can compute a pair $(\rho', \bar{e})$ satisfying $\rho' \equiv \rho^{\bar{d}} \bmod p - 1$ and $\bar{e}\bar{d} \equiv p - 1$. With these values, (w, y) is transformed into $(w^{\bar{e}}, y)$, and m is extracted with the knowledge of ρ' by the same procedure in step 5 of Protocol 2. If the legally authorized decryptor gives other decryptor only ρ' and keeps $\bar{e}$ in secret, the second proxy decryptor has to wait until the legal proxy decryptor transforms the ciphertext (w, y) into $(w^{\bar{e}}, y)$. In this way the authorized proxy decryptor can further delegate its decrypting operation to other decryptors. Such a chain of delegation, which does not occur in Protocol 1, can be useful for an implementation in a hierarchical organization.

As in an ordinary cryptosystem, a proxy cryptosystem has a problem of illegal access. A proxy decryptor may give its proxy to an unauthorized decryptor. In the proxy signature schemes a created proxy signature is different for each proxy signer, and an original signer can identify a corresponding proxy signer from a created proxy signature. This indicates even if a proxy signer gives his proxy to other users, a signature created by the delegated user is identified as a signature of the original proxy signer. In contrast, an original decryptor is unable to detect the illegal access in the proxy cryptosystem. Let us consider two types of illegal accesses. One is an access by an person who is given ρ by the proxy decryptor but not by the original decryptor. The other is an access by a person who possesses a new proxy ρ' described above. The latter undetected illegal access is unique in the proxy cryptosystem. The original decryptor cannot distinguish or even detect both decryptions. So, a person attempting illegal access do not care which proxy, a proxy ρ or a newly generated ρ', he receives. He cannot be identified with the use of any one of them. Moreover, as easily observed, one can compute the original proxy ρ from a pair $(\rho', \bar{e})$, and vice versa. This means the possession of $(\rho', \bar{e})$ is equivalent to the possession of ρ.

Meanwhile, an illegally authorized decryptor does not have a stronger privilege than an original decryptor. The illegally authorized decryptor cannot recover the ciphertext without the ciphertext transformation from the original ciphertext (x, y) to the ciphertext (w, y) for the proxy decryptor.

4.3 Proxy Cryptosystem for RSA Cryptosystem

The following is a proxy cryptosystem for the RSA cryptosystem. An original decryptor selects two primes p and q, and computes $n = pq$. (e_s, n) and (s, p, q) are a public key and a secret key of the original decryptor, respectively, satisfying $\gcd(s, \lambda(n)) = 1$, $e_s s \equiv 1 \bmod \lambda(n)$. $\lambda(\cdot)$ is the Carmichael function.

[Protocol 3]

Step 1. (Proxy generation) An original decryptor computes $\rho = sd \bmod \lambda(n)$, where d is randomly generated from $Z^*_{\lambda(n)}$.

Step 2. (Proxy delivery) The original decryptor gives ρ to a proxy decryptor in a secure way.

Step 3. (Encryption) A document m is encrypted into x by $x = m^{e_s} \bmod n$.

Step 4. (Ciphertext transformation) The original decryptor transforms the x into w by $w = x^e \bmod n$, where $ed \equiv 1 \bmod \lambda(n)$.

Step 5. (Decryption by proxy decryptor) The proxy decryptor computes

$$\begin{aligned} w^\rho &\equiv (m^{e_s e})^{\{sd \bmod \lambda(n)\}} \bmod n \\ &\equiv m^{\{ede_s s \bmod \lambda(n)\}} \bmod n \\ &\equiv m \bmod n. \end{aligned}$$

As in Protocol 2, the proxy verification process is not included in this protocol.

Security considerations:

Lemma 3: If a random number $f \in_R Z_n \backslash \{0\}$ relatively prime to $\lambda(n)$ can be generated within a polynomial number of trials of n, the intractability of the problem of computing m given $(m^{e_s} \bmod n,\ m^{e_s e} \bmod n, e_s, n)$, where e_s, e and n meet the condition of the RSA cryptosystem, is equivalent to that of the problem of computing m given $(m^{e_s} \bmod n, e_s, n)$.

Proof: If the latter RSA problem is solved, the former is solved immediately.

If the former problem is solved, the RSA problem is solved by first generating f from $Z_n \backslash \{0\}$ at random. Then $(m^{e_s})^f \bmod n$ is computed, and feed $(m^{e_s} \bmod n,\ m^{e_s f} \bmod n, e_s, n)$ into an oracle for the former problem. If the oracle does not output anything or output an incorrect answer, repeat the above process by generating another f. This procedure is repeated until the oracle outputs a correct m.

In general, the number of integers relatively prime to an integer N is $6N/\pi^2$ in average. This means f should be selected roughly $\lceil \pi^2/6 \rceil \cong \lceil 1.64 \rceil = 2$ times in average. Usually, the RSA primes, p and q, need to satisfy several security conditions, e.g. $p - 1$ has a large prime factor. Assume that $p - 1 = 2\tilde{p}$ and $q - 1 = 2\tilde{q}$ for primes $\tilde{p}$ and $\tilde{q}$ s.t. $|\tilde{p}| \cong |\tilde{q}|$. Then $\frac{|Z^*_{\lambda(n)}|}{|Z_n \backslash \{0\}|} = \frac{(\tilde{p}-1)(\tilde{q}-1)}{(2\tilde{p}+1)(2\tilde{q}+1)-1} \cong \frac{1}{4}$. In both cases the number of trials for generating f is within polynomial of n. Lemma 3 indicates that two problems have the same degree of difficulty under a certain condition.

(On transformation) A third party and a proxy decryptor do not know e for transformation. Even given a group of $(m^{e_s} \bmod n, m^{e_s e} \bmod n)$, the third party

with no knowledge on e cannot execute the ciphertext transformation. For the third party this problem can be regarded as a problem to solve the RSA cryptosystem without knowing both a secret value e and a public value d. In order to illegally transform a ciphertext, the proxy decryptor faces a problem to compute $(m^{e_s})^e \bmod n$ for a given $m^{e_s} \bmod n$ from (e_s, n, p) and a set of $(m_i^{e_s} \bmod n, m_i^{e_s e} \bmod n)$ for a certain amount of i. The degree of the difficulty of this problem is not totally clear, but this problem has not been solved until now.

(On authorization) As explained in Lemma 3, a third party has to solve the RSA problem in Protocol 3. So, as long as the RSA problem is not violated, the proposed scheme is considered to be secure.

5. Efficiency, Proxy Revocation and Authorization of Multiple Decryptors

In this section properties of the proposed proxy cryptosystems are examined.

Efficiency: The amount of computational work needed in the ordinary and the proxy version of ElGamal cryptosystem is shown in Tables 1, 2 and 3. Since the operation in Protocol 3 is similar to that in Protocol 2 except the inverse operations, the evaluation of computational work of Protocol 3 is omitted. The amount of computational work of the proposed schemes is roughly estimated by the number of $|p|$-bit exponentiations modulo p. Inv($|p|$) denotes the computational work of taking inverse modulo $|p|$-bit integer.

Table 1 shows the amount of computational work required in a consecutive execution of decryption and re-encryption.

Table 2 and Table 3 show the amount of computational work required in Protocols 1 and 2. In these tables, total (case 1) shows values in a case such that an original decryptor does not decrypt the ciphertext, and total (case 2) shows values in a case such that an original decryptor decrypts the ciphertext. The computational work of proxy generation and proxy verification is shown below Table 2 and Table 3. Since the generation and verification of proxies are performed only when a proxy is given to a proxy decryptor at the first time, this computational work per one ciphertext transformation becomes negligible as the number of decrypted messages increases.

In Protocol 1 the total amount of computational work is about $31/6+\text{Inv}(|p|)$ $(= 3+(13/6+2\text{Inv}(|p|)))$ in the case 1 and about $5+2\text{Inv}(|p|)$ $(= 3+(2+2\text{Inv}(|p|)))$ in the case 2. In both cases Protocol 1 requires less amount of total computational work than the consecutive execution of decryption and re-encryption does, i.e. $6+2\text{Inv}(|p|)$ $(= 4+(2+2\text{Inv}(|p|)))$.

In Protocol 2 the total amount of computational work is about $4+\text{Inv}(|p|)$ $(= 3+(1+\text{Inv}(|p|)))$ in the case 1 and about $5+2\text{Inv}(|p|)$ $(= 3+(2-2\text{Inv}(|p|)))$ in the case 2. Protocol 2 also requires less amount of total computational work than the consecutive execution of decryption and re-encryption does.

A proxy decryptor needs to execute in Protocol 2 the same amount of computation as in the re-encryption method, and the proxy decryptor needs to perform a slightly more computation, i.e. 1/6, in Protocol 1 than in the re-encryption method. Even so, the original decryptor's process costs 1 $|p|$-bit exponentiation modulo p, whereas 2 $|p|$-bit exponentiations modulo p are required in consecutive processing of decryption and re-encryption. Such a property is well suited to a situation such that an original decryptor receives many ciphertexts and delegates their decryptions to multiple decryptors, e.g. survey. Concerning the computational work, the proxy signature and the proxy cryptosystem have the following relation. The computational work in verification operation is mainly reduced in the proxy signature and the computational work in ciphertext transformation is mainly reduced in the proxy cryptosystem.

Table 1 Computational work of the decryption and re-encryption method based on ElGamal cryptosystem.

	Encryption	Decryption
Total	4	$2+2\text{Inv}(\|p\|)$
Sender	2	
Original decryptor	2	$1+\text{Inv}(\|p\|)$
Proxy decryptor		$1+\text{Inv}(\|p\|)$

Table 2 Computational work of the proxy cryptosystem for ElGamal cryptosystem (Protocol 1).

	Encryption (Transformation)	Decryption
Total (Case 1)*	3	$7/6+\text{Inv}(\|p\|)$
Total (Case 2)†	3	$13/6+2\text{Inv}(\|p\|)$
Sender	2	
Original decryptor	1	$1+\text{Inv}(\|p\|)$
Proxy decryptor		$7/6+\text{Inv}(\|p\|)$

Proxy generation: Almost 0, Proxy verification: 2

Table 3 Computational work of the proxy cryptosystem for ElGamal cryptosystem (Protocol 2).

	Encryption (Transformation)	Decryption
Total (Case 1)*	3	$1+\text{Inv}(\|p\|)$
Total (Case 2)†	3	$2+2\text{Inv}(\|p\|)$
Sender	2	
Original decryptor	1	$1+\text{Inv}(\|p\|)$
Proxy decryptor		$1+\text{Inv}(\|p\|)$

Proxy generation: Almost 0, Proxy verification: No method

* Total (case 1): Original decryptor does not decrypt the ciphertext.

† Total (case 2): Original decryptor decrypts the ciphertext.

Signing operation in the proxy signature and decrypting operation in the proxy cryptosystem require almost the same amount of computational work as an original signer and an original decryptor require, respectively, so that many signatures are created and many ciphertexts are decrypted by delegating signing power and decrypting power to multiple proxy signers and multiple proxy decryptors.

Revocation: There are two ways to prohibit proxy decryptor's access to the ciphertext after the ciphertext transformation is executed:

- To revoke the proxy of the proxy decryptor, which leads to prohibition of any further access to ciphertexts of the original decryptor.
- To disqualify the proxy decryptor for the access to a specified ciphertext.

The first method is achieved by changing the public key of the original decryptor, and accordingly update all proxies of proxy decryptors whom the original decryptor considers as qualified. There is a very simple method [7] to update proxies through an insecure channel. Thereby the owner of the proxy decryptor does not need to visit the original decryptor again, in other words no secure channel is needed. This protocol is shown in Appendix B.

The second method is useful only when the proxy decryptor makes an access to the message for the first time, or when the proxy decryptor has not stored the decrypted message at the earlier access and tries to get an access to the ciphertext again. In this revocation, the original decryptor selects a transformed ciphertext (w, x, y), and computes $x' = x \cdot g^{\bar{r}} \bmod p$ $(\equiv g^{r'} \bmod p)$ and $y' = y \cdot v^{\bar{r}} \bmod p$ $(\equiv mv^{r'} \bmod p)$, where $\bar{r}$ is a randomly selected number and $r' \equiv r + \bar{r} \bmod p - 1$. Since r of w $(\equiv T^r \bmod p)$ and r' are independent, the proxy decryptor cannot compute m from the updated ciphertext (w, x', y'). Once the form of a ciphertext is changed, an original decryptor is sure that the encrypted message is not read by a further access.

Authorization of multiple decryptors: In some cases, e.g. in file access, an original decryptor wants to restrict an access group in a group of proxy decryptors.

The original decryptor may authorize multiple proxy decryptors with different proxies for this purpose. In Protocol 1, the original decryptor gives each proxy decryptor D_i a proxy $\rho_i = s - t_i I_i \bmod p - 1$ with a distinct t_i. A transformed ciphertext for a group G of proxy decryptors is $(\{w_j | D_j \in G\}, x, y)$, where $w_j = x^{t_j} \equiv T_j^r \bmod p$. The proxy decryptor D_j with ρ_j treats (w_j, x, y) as its ciphertext, and executes decryption. This method is not enough efficient in terms of the message length and the amount of computational work since both of them are proportional to $O(\xi)$, where ξ is the number of proxy decryptors.

The original decryptor can authorize multiple proxy decryptors by combining a proxy cryptosystem with a selective broadcasting method. In this case, the multi-dimensional method for a secure broadcast communication proposed in [6] is quite useful for reducing the message length and the amount of computational work from $O(\xi)$ of the above method to $O(m \sqrt[m]{\xi})$ of m-dimensional method. In the m-dimensional method a user $U_{i_1 i_2 \cdots i_m}$ is assigned to a point $(i_1, i_2, \cdots, i_m)$ in an m-dimensional space, and public values $g^{t_{i_1}} \bmod p$, $g^{t_{i_2}} \bmod p$, $\cdots$, $g^{t_{i_m}} \bmod p$ are assigned to the point $(i_1, i_2, \cdots, i_m)$.

Step 1. (Proxy generation, delivery and verification) $\rho_{i_1 i_2 \cdots i_m}$ is computed by the original decryptor, and given to $D_{i_1 i_2 \cdots i_m}$. $\rho_{i_1 i_2 \cdots i_m}$ satisfies $\rho_{i_1 i_2 \cdots i_m} = s - t_{i_1 i_2 \cdots i_m} T_{i_1 i_2 \cdots i_m} \bmod p - 1$, where $t_{i_1 i_2 \cdots i_m} = t_{i_1} + t_{i_2} + \cdots + t_{i_m} \bmod p - 1$ and $T_{i_1 i_2 \cdots i_m} = g^{t_{i_1} t_{i_2} \cdots t_{i_m}} \bmod p$. Each $D_{i_1 i_2 \cdots i_m}$ checks the validity of given $\rho_{i_1 i_2 \cdots i_m}$.

Step 2. (Encryption) A ciphertext (x, y) is computed as in step 4 of Protocol 1.

Step 3. (Ciphertext transformation) In order to transfer (x, y) to a proxy decryptor $D_{i_1 i_2 \cdots i_m}$, the original decryptor computes $w_{i_j} = x^{t_{i_j}} \equiv T_{i_j}^r \bmod p$ for all i_j corresponding to $D_{i_1 i_2 \cdots i_m}$, where $T_{i_j} = g^{t_{i_j}} \bmod p$. A ciphertext to a group G of proxy decryptors is $(\{w_{i_j} | w_{i_j} = x^{t_{i_j}} \bmod p$ for all i_j corresponding to $D_{i_1 i_2 \cdots i_m} \in G\}, x, y)$.

Step 4. (Decryption by proxy decryptors) $D_{i_1 i_2 \cdots i_m}$ obtains m by computing step 6 of Protocol 1 by replacing ρ and T with $\rho_{i_1 i_2 \cdots i_m}$ and $T_{i_1 i_2 \cdots i_m}$, respectively.

As described in Sect. 4.1, a creator of a ciphertext can take charge of the operation for the ciphertext transformation as follows.

Step 2'. (Encryption) In addition to (x, y), all public values $T_{i_j} = g^{t_{i_j}} \bmod p$ corresponding to a group of users are selected, and these T_{i_j} are raised to rth power modulo p. A ciphertext to a group G of proxy decryptors is $(\{w_{i_j} | w_{i_j} = T_{i_j}^r \bmod p$ for all i_j corresponding to $D_{i_1 i_2 \cdots i_m} \in G\}, x, y)$.

Step 3'. (Forwarding) The original decryptor selects $\{w_{i_j}\}$ for each of $D_{i_1 i_2 \cdots i_m}$, and forwards selected $\{w_{i_j}\}$ together with (x, y) to $D_{i_1 i_2 \cdots i_m}$.

Please refer to [6] for more detail description of the m-dimensional method.

6. Conclusions

In this paper proxy cryptosystems have been proposed. By the proxy cryptosystem an original decryptor can securely transfer its ciphertext to a proxy decryptor, and

it can permit the proxy decryptor to recover the message instead of the original decryptor. Many different constructions for the proxy cryptosystem are possible for one normal cryptosystem. In this paper two proxy cryptosystems for the ElGamal cryptosystem and one proxy cryptosystem for the RSA cryptosystem have been described. In these proxy cryptosystems the transformation of ciphertexts is more efficient than consecutive processing of decryption and re-encryption, and decrypting operation requires almost the same amount of computational work as the ordinary decrypting operation. Therefore, an entity receiving a lot of encrypted messages can efficiently conduct decryption of these ciphertexts by giving the decrypting power to multiple proxy decryptors.

Acknowledgments

Authors would like to thank Dr. Renè Peralta for discussion on problems related to the discrete logarithm problem. Authors would also like to thank anonymous referees for their comments.

References

[1] E.F. Brickell and Y. Yacobi, "On privacy homomorphism," Lecture Notes in Computer Science 304, Advances in Cryptology — Eurocrypt'87, pp.117–125, Spring-Verlag, 1988.
[2] I. Damgård, "Towards practical public key systems secure against chosen ciphertext attacks," Lecture Notes in Computer Science 576, Advances in Cryptology - Crypto'91, pp.445–456, Spring-Verlag, 1992.
[3] B. den Boer, "Diffie-Hellman is as strong as discrete log for certain primes," Lecture Notes in Computer Science 403, Advances in Cryptology — Crypto'88, pp.530–539, Spring-Verlag, 1990.
[4] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–645, Nov. 1976.
[5] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithm," IEEE Trans. Inf. Theory, vol.IT-31, no.4, pp.469–472, July 1985.
[6] M. Mambo, A. Nishikawa, E. Okamoto, and S. Tsujii, "A secure broadcast communication method with short messages," IEICE Trans. Fundamentals, vol.E77-A, no.8, pp.1319–1327, Aug. 1994.
[7] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the power to sign messages," IEICE Trans. Fundamentals, vol.E79-A, no.9, Sept. 1996. Or "Proxy signatures for delegating signing operations," Proc. of 3rd ACM Conference on Computer and Communications Security, pp.48–57, 1996.
[8] T. Matsumoto, K. Kato, and H. Imai, "Speeding up secret computation with insecure auxiliary devices," Lecture Notes in Computer Science 403, Advances in Cryptology — Crypto'88, pp.497–506, Spring-Verlag, 1990.
[9] T. Matsumoto, T. Okada, and H. Imai, "Directly transformed link encryption," IECE Trans., vol.J65-D, no.11, pp.1443–1450, Nov. 1982.
[10] U.M. Maurer, "Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms," Lecture Notes in Computer Science 839, Advances in Cryptology — Crypto'94, pp.271–281, Spring-Verlag, 1994.
[11] T. Okamoto, "Provably secure and practical identification schemes and corresponding signature schemes," Lecture Notes in Computer Science 740, Advances in Cryptology — Crypto'92, pp.31–53, Spring-Verlag, 1993.
[12] R. Rivest, "The MD5 message-digest algorithm," Request for Comments 1321, April 1992.
[13] R. Rivest, L. Adleman, and M. Dertouzos, "On databanks and privacy homomorphisms," in Foundations of Secure Computation, eds. R.A. Demillo et al., pp.168–177, Academic Press, 1978.
[14] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," Commun. of the ACM, vol.21, pp.120–126, 1978.
[15] A. Shimbo, "Multisignature schemes based on the ElGamal scheme," Proc. 1994 Symposium on Cryptography and Information Security, SCIS94-2C, Jan. 1994.
[16] A. Shimbo and S. Kawamura, "A factorization attack against some server-aided computation protocols for the RSA secret computation," Proc. 1990 Symposium on Cryptography and Information Security, SCIS90-3B, Jan. 1990.
[17] A. Shimbo and S. Kawamura, "Performance analysis of server-aided secret computation protocols," IEICE Trans., vol.E73, no.7, pp.1073–1080, July 1990.
[18] M. Tompa and H. Woll, "Random self-reducibility and zero-knowledge interactive proofs of possession of information," Proc. of Symp. on Foundation of Computer Science, pp.472–482, 1987.
[19] S.M. Yen and C.S. Laih, "New digital signature scheme based on discrete logarithm," Electron. Letters, vol.29, no.12, pp.1120–1121, 1993.
[20] Y. Zheng and J. Seberry, "Practical approaches to attaining security against adaptively chosen ciphertext attacks," Lecture Notes in Computer Science 740, Advances in Cryptology — Crypto'92, pp.292–304, Spring-Verlag, 1993.

Appendix A

In [20] a public-key cryptosystem secure against adaptively chosen ciphertext attacks is presented. The protocol is as follows.

Let v and s be a public key and a private key of a decryptor, respectively, and $G(\tau)_{|1\cdots P(\eta)|}$ be a cryptographically strong pseudorandom string generator based on the difficulty of computing discrete logarithms in finite fields that outputs a $P(\eta)$-bit message, where $P(\eta)$ is an arbitrary polynomial with $P(\eta) > \eta$. $h(\cdot)$ is a one way hash function.

Algorithms by Zheng and Sheberry:

[Encryption]

Step 1. $k_1 \in_R [1, p-1]$, $k_1 \in_R [1, p-1]$, and $gcd(k_2, p-1) = 1$.

Step 2. $r = v^{k_1+k_2} \bmod p$.

Step 3. $z = G(\tau)_{|1\cdots P(\eta)|}$.

Step 4. $c_1 = g^{k_1} \bmod p$.

Step 5. $c_2 = g^{k_2} \bmod p$.

Step 6. $c_3 = (h(m) - k_1\tau)/k_2 \bmod p - 1$.

Step 7. $c_4 = z \oplus m$

Step 8. Output (c_1, c_2, c_3, c_4) as a ciphertext.

[Decryption]

Step 1. $\tilde{r} = (c_1 c_2)^s \bmod p$.

Step 2. $\tilde{z} = G(\tilde{r})_{[1 \cdots P(\eta)]}$.

Step 3. $\tilde{m} = \tilde{z} \oplus c_4$.

Step 4. Output $\tilde{m}$ if $g^{h(\tilde{m})} \equiv c_1^{\tilde{r}} c_2^{r_3} \bmod p$. Otherwise, output $\emptyset$.

Modification to proxy cryptosystem: By the following modification a proxy cryptosystem secure against adaptively chosen ciphertext attack is constructed. This cryptosystem is based on the proposed Protocol 1. The same modification can be done based on Protocol 2.

Step 4 in decryption algorithm of an original decryptor: If a message is output, the original decryptor computes $w_1 = c_1^t \equiv g^{k_1 t} \bmod p$ and $w_2 = c_2^t \equiv g^{k_2 t} \bmod p$. The transformed ciphertext is composed of $(w_1, w_2, c_1, c_2, c_3, c_4)$.

Step 1 in decryption algorithm of a proxy decryptor: $\tilde{r}$ should be computed by $\tilde{r} = (c_1 c_2)^{\rho} (w_1 w_2)^{\tau} \equiv (t^s)^{k_1 + k_2} \bmod p$.

Appendix B

An original decryptor and a legal decryptor share a proxy, so that they can update the proxy even through an insecure channel where an eavesdropping is possibly conducted. By the following protocol a proxy of Protocol 2 is updated in on-line basis. A proxy of Protocol 1 and Protocol 3 is similarly updated [7], either.

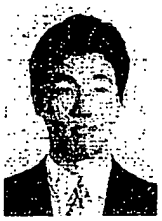
[On-line proxy updating protocol]

Step 1. (New public-key creation) An original decryptor selects its new secret $s' \in_R Z_{p-1}^*$ and computes its new public key v' by $v' = g^{s'} \bmod p$. Then the original decryptor announces the revocation to all of related proxy decryptors.

Step 2. (Identification) After the announcement, a proxy decryptor requests the update of its proxy. To this end, the proxy decryptor proves its identity by some identification protocol.

Step 3. (New proxy creation and implicit delivery) If the original decryptor is convinced of the identity of the proxy decryptor, it looks for its old secret proxy variable d in its secret proxy variable list. It calculates its old proxy $\rho = ds \bmod p - 1$ and her new proxy $\rho' = d's' \bmod p - 1$, where $d' \in_R Z_{p-1}^*$ is selected randomly. Then it can compute $\bar{\rho} = \rho' - \rho \bmod p - 1$. $\bar{\rho}$ is returned to her.

Step 4. (New proxy construction) Using the received information the proxy decryptor calculates its new proxy ρ' simply by $\rho' = \rho + \bar{\rho} \bmod p - 1$.



Masahiro Mambo received a B.Eng. degree from Kanazawa University, Japan, in 1988 and M.S.Eng. and Dr.Eng. degrees in electronic engineering from Tokyo Institute of Technology, Japan, in 1990 and 1993, respectively. He is currently a research associate in the school of information science at JAIST (Japan Advanced Institute of Science and Technology), and is involved in research on information security.

Eiji Okamoto received B.S., M.S. and Ph.D. degrees in electrical engineering from Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. He worked for NEC Corporation from 1978 to 1991 and studied communication theory and information security. He has been a professor of Information Science at JAIST (Japan Advanced Institute of Science and Technology) since 1991. He was a visiting professor of Mathematics, Texas A & M University, USA, 1993–1994. He is a senior member of the IEEE Communications Society and Information Theory Group. He is also a member of the Information Processing Society of Japan, and International Association of Cryptographic Research. He received the best Paper Award of the IEICE in 1990 and the best Author Award of the IPSJ in 1993.

Summary and Analysis of A13

An Overview of the Technical Goals and Concepts in "*An Open Architecture for a Digital Library System and a Plan for its Development*" by Robert E. Kahn and Vinton G. Cerf (1988)

by Mark Stefik

Research Fellow
Palo Alto Research Center
Palo Alto, California 94304

Prepared for ContentGuard
May 30, 2007

Contents

Executive Summary

In 1988 Robert Kahn and Vinton Cerf wrote a draft paper (A13) proposing an architecture and plans for a digital library system. In broad terms, they proposed an infrastructure intended to link libraries together and to enable people to share information and documents.

In setting out their proposal, they made important design choices. A crucial choice at the onset was to base their design on the metaphor of a traditional library. In contrast, digital rights management systems such as those invented in the early 1990s and deployed today were intended to support digital commerce. These DRM designs were based on the metaphor of the electronic marketplace[1]. The difference in choice of design metaphor led to different goals for what the systems were expected to do, leading in many crucial ways to essentially opposite design choices.

Traditional libraries in the United States are not commercial organizations. Traditional libraries are not concerned with regulating the particular uses to which information is put. Nor are they concerned with meeting the needs of commerce, such as pricing regimes, licensing arrangements, sales territories, or special deals for students or members of various groups. The most important legal basis for libraries is Copyright Law. In contrast, the legal basis for business agreements between parties is Contract Law.

This difference in orientation between A13 and later DRM systems is profound. In the main, A13 does not focus on the needs of commerce, where parties are often competitive and sometimes adversarial. For example, A13 does not consider any need for visible agreements reflecting the interests of parties in commercial transactions. It does not recognize a need for a threat analysis and arrangements for robust security in system architecture.

Table 1 summarizes some of the main differences in design choices between A13 and many later DRM approaches.

---

[1] For example, see the book *Internet Dreams: Archetypes, Myths, and Metaphors* by Mark Stefik (with a foreword by Vinton Cerf (one of the authors of A13), published in 1996 by The MIT Press.

Design Issue	Design Choice in A13	Design Choice in DRM systems
Main design goal.	Promote sharing of information. A13 was intended to provide friendlier services akin to traditional libraries.	Promote a viable system to enable new business models for the commercial distribution of digital content.
Mechanism for overseeing rights.	Software-only agents ("Knowbots") that are attached to documents and travel from system to system as mobile code. Potentially, each different set of requirements is met by a different program.	Digital rights may be expressed in a declarative rights language for digital contracts. These contracts are intended for use both in user interfaces and by standard and certified trusted systems that enforce usage obligations. Communication among trusted systems is often protected by encryption. The trusted systems get the parameters of usage from the digital contracts. The same trusted systems potentially work for all documents. Foundations for trust in these systems may include physical (hardware) security, communication security (e.g. encryption), and behavioral security (certified programs).
Organizational basis of trust.	Content consumers must trust the producers of content, who commission the writing of different Knowbots for each use. Content producers must trust the consumers of content, who configure the operating environments on their machines for Knowbots. There is no consideration of the practical verifiability of system correctness or security.	Producers and consumers of content agree on terms and conditions and express them in digital contracts. They do not depend on each other's code. Rather, they rely on qualified (and disinterested) third parties to develop and certify trusted systems that enforce the digital contracts.
Purpose of computer oversight.	Knowbots are intended to represent the interests of content owners. Knowbots are largely concerned with accounting for the amount of usage for each document.	Digital contracts are designed to represent *all* essential elements in the agreements between all of the parties to the transactions. Digital contracts address the terms and conditions of use – including allowed operations, fees, timing, special licenses,

		distribution, and so on.
Accommodation of existing systems.	A13 recognized that the digital library would be distributed, heterarchical, networked, and display oriented. It must have an ability to interact with Digital Library Systems that do not adhere to its internal standards and procedures.	Most DRM systems invented in the 1990s and deployed today were a departure from existing networked file systems in their foundations of trust. In order to prevent the compromise of commerce or private content, they were designed specifically to only operate with other trusted systems that could meet their standards.

The rest of this paper discusses the technical goals and choices in A13 in more detail, and shows how the design choices took opposite paths from the DRM systems that were invented in the 1990s.

1 Introduction

In 1988 Robert Kahn and Vinton Cerf wrote a visionary white paper, "An Open Architecture for a Digital Library System and A Plan for its Development" (A13). It presents a draft research and development plan for a public information infrastructure that would enable digital library services. Kahn and Cerf were not newcomers to the creation of public infrastructure. They have received prestigious awards[2] for their earlier leadership and central roles in the creation and early development of the Arpanet – which famously became the Internet. Their recognized technical contributions to the Internet were for the protocols for packet-switching – the TCP/IP protocols. These transport protocols are called "low-level" because they govern how computers robustly send bits to each other in a digital network. With their backgrounds in computer science and electrical engineering, and networking specifically, Kahn and Cerf were well prepared to define these protocols. From their positions in the Information Processing Techniques Office of ARPA, they were deeply connected to the major centers of computer science research at the time, and well positioned to guide the creation of the net.

[2] Among other awards, Robert Kahn and Vinton Cerf were the winners of the Turing Award in 2004 for their pioneering work on networking. In 2005 they were awarded the Presidential Medal of Freedom for this work. They were inducted into the National Inventors Hall of Fame in 2006.

In A13, Kahn and Cerf considered a new challenge. They chose the metaphor of the "library" to set goals and expectations for a proposed higher-level digital information infrastructure.

> The term "library" conjures a variety of different images. For some, a library is a dim and dusty place filled with out-of-date texts of limited historical interest. For others, it is a rich collection of archival quality information which may include video and audio tapes, disks, printed books, magazines, periodicals, reports and newspapers. As used in this report, a library is intended to be an extension of this latter concept to include material of current and possibly only transient interest. Seen from this new perspective, the digital library is a seamless blend of the conventional archive of current or historically important information and knowledge, along with ephemeral material such as drafts, notes, memoranda and files of ongoing activity. *[A13, Summary, page 3]*

The theme of Kahn and Cerf's earlier, seminal contribution to the Internet was to "link computers together," creating an infrastructure for sharing data. In A13, they develop the related theme to "link libraries together" in an infrastructure that would enable people to share information and documents. In their view, users would participate with their personal computers acting as personal digital libraries.

> In its broadest sense, a DLS is made up of many Digital Libraries sharing common standards and methodologies. It involves many geographically distributed users and organizations, each of which has a digital library which contains information of both local and/or widespread interest. *[A13, page 3]*

Ultimately, the success and scope of A13 were limited by the library metaphor itself, which has proven unsuitable for fully characterizing commerce and information sharing in human organizations. This problem is that the normal activities in information exchange among competitive and sometimes adversarial human organizations are outside the scope of what normally happens in libraries. When A13 left behind the relatively simple world of computers that exchange bits, it entered the complex world of human organizations where activities involve ownership, competition, collaboration, and variations in law and agreement. This broader world of human commerce has many complexities and requirements that are not significant in libraries. Nor are these requirements comprehended or addressed in the goals and technologies described in A13.

- ***A13 Goals.*** In the U.S., the most influential libraries are public libraries and university libraries. The needs and experiences of these cooperative, non-

commercial institutions have proven to be somewhat misleading when used to set goals for a large-scale, commercially-oriented information infrastructure. For example, the library-oriented vision lacks a realistic consideration of information security and regulation of how information is used, which play central roles in the activities of online commercial systems.

- ***A13 Technology***. For their leadership of the digital library project, Kahn and Cerf had to reach well beyond the EE/CS research that they knew best. For example, the technical projections on which elements of their approach depended (such as progress in natural language understanding by 2003) have proven to be overly optimistic at least in time-scale. Furthermore, some of the main technical ideas in their proposal have so far proven unworkable, and have been superseded by very different approaches. For example, A13 suggests transmitting mobile code ("Knowbots") on behalf of a user to run at distant sites in order to search through their information files. The technology approaches that have come to dominate searching services in the World Wide Web are a study in contrast to A13's proposals. For example, Web search services such as Google employ web-crawlers and indexing engines with massive resources[3] for computation and storage[4]. Restated, the technical approaches that have proven most practical for massive search transmit content rather than transmitting programs ("Knowbots").

The issues around goals and technologies in A13 are elaborated in the following sections.

2. Setting Goals for Digital Libraries

In the United States, most major libraries are either public libraries or are affiliated with higher educational institutions. Although there are also corporate research libraries and some collections in private ownership, most of these libraries are connected to academic libraries, especially for purposes of obtaining access to collections through inter-library loans. Kahn and Cerf also mention databases – for information such as scientific data, public records, law records and medical data. Such databases are mostly outside of the

[3] Computation facilities known as "server farms" are employed by search companies to index the web. Server farms can cover several acres and have upwards of tens of thousands of computers.
[4] Although web sites can have their own search services, their information is part of what is sometimes called the "dark web" if the information is not open for search by the main search engines.

library system and have not been freely accessible to the public. In their proposed "digital library project," however, Kahn and Cerf sweep all of these sources and kinds of information into the familiar library pattern.

The mainstream activities of public and academic libraries establish key expectations. For example, libraries are not organized as commercial enterprises and their services are not for hire. Public libraries serve a public good, and academic libraries prioritize the particular needs of their academic communities. Libraries do not charge for information. Monetary transactions in libraries are mainly about record keeping and cost recovery for inter-library loans. Libraries do not advertise or suggest that people should buy certain sources of information. Librarians often provide facilities for making free copies of materials and are at least generally aware of copyright laws and the principle of fair use, whereby library clients may copy certain materials for their own scholarly purposes. Most libraries don't house secret information such as private commercial data or data for which there are privacy concerns. Their goal is to make all of their information holdings available to all of their clients rather than making selected information differentially available to different people.

The mission of traditional libraries is to make information available to the public. In the main, libraries are not concerned with regulating who has access to information or what people can do with information once they have it. These library-centered assumptions about information and its use are reflected in the design goals and assumptions of the digital library proposal described in A13. Specifically, they affect how A13 sets expectations and goals about regulated uses, about commerce and business models, and about security.

2.1 Setting Goals for Regulated Access

In the world of competitive and adversarial commercial activities, organizations create trust boundaries that regulate access to information. For example, the accounting department in an enterprise keeps its files of information within a trust boundary where access is limited to people with particular authorizations. The ability to read or update accounting records is limited to those who have authorization to operate inside the trust boundary. A company often has multiple trust boundaries for different parts of its

operations. For example, human resource records are used by different people than engineering plans for future products. For another example, law firms representing multiple clients must keep the client information separate. Two competing companies do not generally have access to each others' customer, payroll, sales, or strategic planning information.

A diagram of the information infrastructure for a commercial environment would have elements supporting information use and elements supporting security. For example, it would include firewalls and other security systems to detect intrusions and to keep malicious software out. It would have cryptographic services that support codes to protect information from prying eyes and, as part of a communication subsystem, to robustly identify communicating parties and to protect the integrity of content. It would include authentication services to identify and authorize particular parties to access information.

Figure 1 reproduces a diagram from A13 showing the structure of the proposed Digital Library System. The figure shows that personal and organizational digital libraries are linked together on the Internet together with various databases. Boxes in the figure indicate services for registering documents, importing documents into the system, indexing and cataloging. There are also accounting and billing services – suitable for handling library fees. In the figure, multiple personal library systems co-exist with multiple organizational library systems without trust boundaries and few provisions for security[5].

[5] For example, page 23 of A13 describes a registration server as responsible for "registering new users, sources of information (databases) or other components newly added to the system." In the further description about this, the main concern is about assigning unique identifiers as an aid to indexing and cataloging. No where are issues of verifying the authenticity of users or content discussed. Like an open library, the users are expected to behave in a kind of honor system. In contrast, more is at stake in an adversarial commercial environment. Digital systems supporting commerce and information access in such an environment would need robust and automatic means for checking the credentials of users and content before authorizing either access or changes to information.

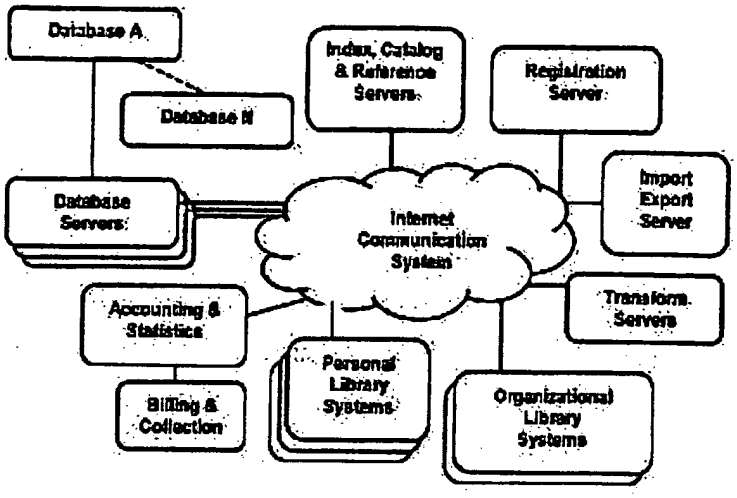


Figure 1. Reproduction of Figure 2 from A13 – "Structure of the Digital Library System"

In understanding the scope and limitations of A13, what is interesting is what is missing. None of the normal commercial considerations involving security, authorization and trust are indicated in the diagram or elsewhere in the text of A13. There is no system "threat analysis" to characterize security requirements. Such provisions are outside the scope of libraries and they are outside the scope of the proposed digital library design[6]. Overall, A13 pays scant attention to commerce or security.

2.2 Setting Goals for Commercial Use

The early government-supported computer networks in the United States allowed no commerce. In the 1970s, if someone sent an email that could be construed as a "commercial message," it would often set off a flurry of discussion about inappropriate use of the network. This "non-commercial" sensibility – coupled with the context of libraries – is part of the background context for A13.

Among the commercial businesses that use information, publishing stands out for its dependence on the regulation of uses of content in order to sustain its business. In publishing, authors and people in other creative roles create new content. Publishers and

[6] On page 26 there is the following sentence: "To increase system integrity, the Accounting and Statistics Servers should be configured to accept data only from the appropriate sources and to raise alarms when data arrives from an unexpected source." In this way the design focuses on accounting reliability, in contrast to security as it relates to regulated use of content.

distributors publish, distribute, and sell content. Consumers purchase and consume content for their information or entertainment purposes – such as playing music, watching a movie, or reading a book. People carry out different activities around information according to their roles as creators, owners, distributors and consumers. For example, consumers do not typically write in or modify books that they buy, make copies, and then distribute them in competition with the original authors or publishers.

Various arrangements are routinely employed to promote commerce in the sale and distribution of content. For example, content may be offered at a temporary discount to encourage sales. Tiered pricing is used to maximize profits by enabling different pricing in different markets. For example, price may depend on the time of sale, the location of the sale, or the affiliations of the purchaser. Special discounts may be offered for students, for members of a particular organization, for senior citizens, or for handicapped persons. For another example, businesses distinguish between the making of additional copies of content and the serial re-use of a single copy. Rental services circulate content without increasing the number of copies. Volume discounts are offered – reflecting differences in the costs of sales when large numbers of copies are sold at once. These sometimes socially-aware concepts arise in commercial publishing business practices. Successful use of these approaches in a digital publishing regime requires that information systems have adequate means for regulating the distribution and use of content.

In contrast, such an elaboration of requirements for success of commercial arrangements in a digital content regime is outside the scope of A13. A13 reflects the idea that a library is a place where clients get information. A13 has a model of owner's interests which is largely based on some accounting of levels of library usage *without* regulation of how information is used.

To their credit, Kahn and Cerf recognize that when information is digital, there are natural concerns about intellectual property protection. After all, compared to the substantial effort involved in reproducing (say) books as bound volumes, it is often very easy to copy (unprotected) digital information regardless of the amounts. Kahn and Cerf enumerate some of the problems and also acknowledge that solutions were not known at the time of their writing.

> At present, the basis for intellectual property protection in the U.S. is Patent and Copyright law. The large scale aggregations of information found on CD-ROMs and the selective access to information found in on-line databases may require substantial re-thinking of the ways in which the creators and owners of such information are compensated for its use. There are many issues at stake in this area, not the least of which relate to the ease with which information can be replicated once in digital form and the rapidity with which large quantities of information can be processed (accessed, transferred, analyzed, integrated, etc.). Concepts of value and pricing and royalty for use of information could require considerable revision if the cost of such use is to remain within reason. One does not now pay an author a royalty each time a book is read. However, a royalty may be earned each time a song is played in public, though not in private. If a thousand books are combined on a single CD-ROM and the acquirer of the CD-ROM only intends to read one of them, what sort of royalty arrangement is appropriate to compensate the copyright owners? How would compensation be extended for cases in which electronic copies are provided to users? In fact, the concept of copying or duplicating a work may no longer be the essential factor in calculating royalties since far more complex actions may now be taken on digital information.
>
> These questions are not trivial in nature nor have many workable solutions been proposed thus far. *[A13, pages 11-12]*

In this way, Kahn and Cerf encounter difficulties that are inherent in the library metaphor. Libraries do not themselves make copies of books (for example) in order to save on purchasing costs from publishers. They sometimes provide copiers so that clients can make their own limited copies of information for personal use. Such activities are governed by the provisions of U.S. copyright law. In the preceding quotation, Kahn and Cerf acknowledge that the kinds of activities likely to take place in a digital information infrastructure go beyond the routine activities in a library. When they suggest that "the concept of copying or duplicating a work may no longer be the essential factor in calculating royalties" they are in effect acknowledging that the economic agreements that form the basis of the publishing business break down when people make many (unauthorized and unrecorded) copies. In a digital network, provisions of Copyright Law apply, but enforcement can be intractable.

Kahn and Cerf recognized that solutions to this problem are needed. In terms of the legal basis for regulating use of content, they did not anticipate the possibility of using

Contract Law where Copyright Law falls short. Not recognizing the relevance of Contract Law, they also did not recognize the value of having a machine-understandable declarative language in which "contracts" can be expressed – so that their terms could be explicitly presented both to people and to computer systems. They did not anticipate that secure computer systems could play a practical role in the enforcement of the sort of declarative "digital contracts" that could robustly support the range of human agreements and activities typical in commerce.

In the 1990's, other people developed concepts and technologies that addressed these problems[7]. These approaches led to today's digital rights management (DRM) systems. In contrast to the library metaphor, DRM approaches recognize that there are different and distinguished uses or operations on content, such as making copies, modifying content, printing it, distributing it, selling it, and so on. DRM systems provide means for regulating these different uses in order to sustain an economy in which content is produced and consumed. Conditions are associated with the operations. These conditions must be satisfied before the operations can be performed. For example, a person might need to pay a fee, be a certain age, live in a certain jurisdiction, or belong to a particular group, and so on.

It is not surprising that these distinguished operations and conditions are outside the scope of the library metaphor. Libraries do not create or sell content. They are concerned mainly with making information available to the public. Once a client "gets" information from a library – such as checking out a book—the library is not involved with what the client does with it. Libraries generally do not regulate how information is used or who has access[8].

---

[7] See for example, Stefik, Mark. "Letting Loose the Light: Igniting Commerce in Electronic Publication." In *Internet Dreams*, The MIT Press, Cambridge, Ma., 1996, pages 219-253. The foreword of this book was written by Vint Cerf, one of the authors of A13. The book proposes four metaphors for understanding the origins and directions of the Internet: digital libraries, electronic mail, electronic markets, and digital worlds. In contrasting digital libraries to electronic markets, it highlights the different capabilities that are suggested by the different metaphors. Specifically, this book elaborates how the metaphor of the electronic market usefully covers activities that are not associated with libraries.

[8] The more recent controversies about regulating access by children to certain materials on the Internet, and the reporting of which people have had access to certain information, has been controversial in part because of the conflict with the over all mission of libraries to provide the public with access to information with a minimum of barriers.

Discussion of these issues in A13 adheres to the expectations set by libraries. When Kahn and Cerf imagine what people will do with the information in digital libraries, they say people will "register, store, catalog, search, retrieve, and manipulate digital information in the library" (A13, page 8). These are the same kinds of activities that people could already do in traditional libraries. Notably absent from this list are commercial activities like selling or distributing information, or re-publishing information in revised forms.

In summary, A13 does not investigate the commercial situations where different people have different roles and sanctioned activities around information. A13 characterizes operations on information in library terms – relating to catalogs and retrieval and so on. It does not analyze requirements of a publishing business or requirements among competitive enterprises around regulated use of information.

3. Technology Choices

A13 proposes a high-level draft architecture for a digital library infrastructure. Analogous to the network architecture of the Internet, Kahn and Cerf proposed a network of computers where the nodes of the network are digital libraries. The architecture also included specialized servers for indexing, cataloging, registration, and other specialized functions.

Figure 2 reproduces a diagram from A13 illustrating its high-level system elements for a personal library system (PLS)—a node in the network of libraries. The diagram shows the structure of the PLS in terms of "layers" with an operating system and its device drivers in the bottom layer and application elements in the top layer. The second layer from the bottom contains file services, presentation (display) services, and network transport services. The top layer of the diagram divides elements into ones that serve the user (user interface), ones that access library content, and administrative functions.

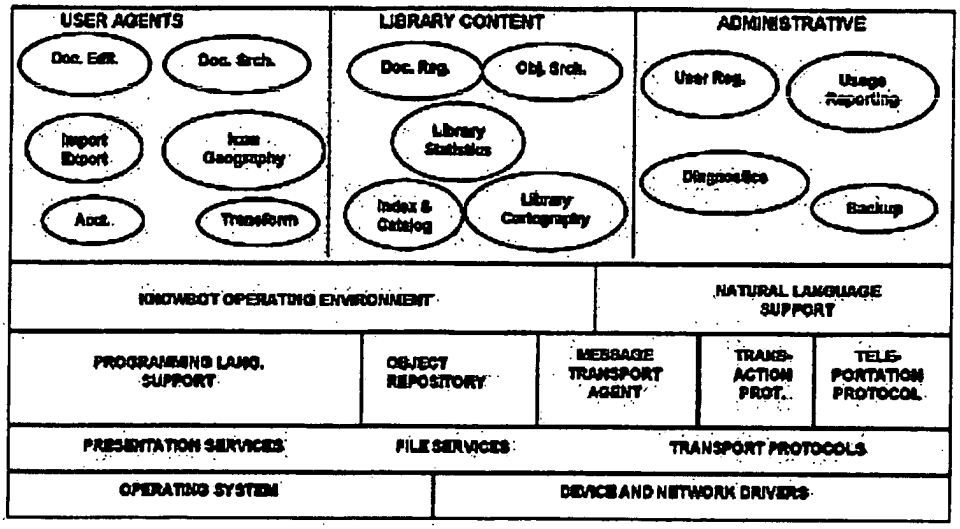


Figure 2. Reproduction of Figure 3 from A13 – "Personal Library System Structure"

The fourth layer of the architecture is of particular interest and contains the most novel aspects of the design. It contains a Knowbot operating environment (KNOE) and natural language support. Knowbots are central in Kahn and Cerf's design for many of the capabilities of the digital library infrastructure. A13 describes Knowbots as follows:

> A Knowbot is an active program capable of operating in its native software environment. Knowbots are present in each of the various components of a Digital Library System. They can be cloned, replicated, created, destroyed, can be resident at a given host system or can move from one host machine to another. Knowbots communicate with each other by means of messages.
>
> Knowbots act as the primary medium of communication and interaction between various major components of the Digital Library System. They may even transport other Knowbots. Generally, a Knowbot may be viewed as a user Knowbot or as a system Knowbot depending on whether it directly serves an individual user or not. [A13, page 34]

Knowbots are more than an incidental part of the proposal for a digital library infrastructure. Unlike the other system architectural concepts, A13 devotes a full section for describing Knowbots (Section 3: Knowbots and Their Application). As Kahn and Cerf noted:

> The selection of a methodology for building Knowbots and even the determination whether an object-oriented language is essential are two

> of the highest priority research questions for the Digital Library Project to resolve. [A13, page 31].

Many of the issues with the architecture in A13 arise from fundamental difficulties with Knowbots, especially security issues[9]. The use of Knowbots as mobile code in the system architecture was an aggressive design choice. Although concepts of object-oriented programming had been developed in the Computer Science community for over a decade[10], the research focused on systems for single computers[11]. Object approaches for distributed and mobile code applications were less explored. Nor had object-oriented applications been deeply explored involving mobile objects or their security considerations across multiple organizations.

The following sections examine Knowbot-related problems with the design in A13 involving trust among multiple parties, security vulnerabilities, and practical problems in using opaque ("black box") mobile code to represent agreements between producers and consumers of content.

3.1 Trust among Multiple Parties

Knowbots are employed in the proposed digital library architecture to carry out many functions. As described in A13, Knowbots can be asked to retrieve or file documents. They also watch over information objects on behalf of owners.

> One set of system Knowbots specifically attend to locally available library information. They take requests from user Knowbots and actually retrieve the documents from storage (or conversely store them away). Another set of system Knowbots attend to background and administrative tasks such as diagnostics, backup and accounting. [A13, page 34]

---

[9] Since the focus of A13 is on libraries rather than commercial organizations, it does not focus on security issues. Nor does A13 explore security requirements for Knowbots.

[10] On page 30, A13 cites Smalltalk, Common Lisp, Common LOOPS and C++ as examples of existing object languages. Smalltalk and Common LOOPS were both largely developed at PARC where I work. I was one of the contributors to the LOOPS and Common LOOPS specifications at the time and was an active researcher in the object-oriented programming community.

[11] For example, see Stefik, M. Bobrow, D.G. Object-oriented programming: Themes and Variations. *AI Magazine* 6:4, pp. 40-62, Winter 1986. (Reprinted in Peterson, G.E. (ed), *Object-Oriented Computing*, Volume 1: Concepts, IEEE Computer Society Press, pp. 182-204, 1987. Also reprinted in Richer, M.H. (ed.) *AI Tools and Techniques*, pp. 3-45, Ablex Publishing Corporation, Norwood, New Jersey.)

In carrying out these functions, Knowbots would have to travel through the network of libraries. When we consider this architecture in the context of commercial activities, a problem of trust arises. Suppose that Company A has a computer system. A document and its Knowbot arrives from Company B. How does Company A know that the Knowbot can be trusted?

A13 only partially addresses the issue of trust. In the context of courier Knowbots, it says the following:

> A class of trusted Knowbots called *couriers* have the special responsibility to look after selected objects on behalf of their authors or other owners of rights in the objects. [A13, page 34]

In the example, the courier is expected to look after the interests of Company B. The problem is that the Knowbot consists of mobile code that is being enabled to run on the computers of Company A. What assurances does Company A have about what the Knowbot will do? Will it correctly enforce the agreements that Company A has with Company B about use of the document? When the Knowbot reports back to Company B, will it also send back any additional information that compromises the business interests of Company A? Will it confine its activities to the document from Company B, or will it read or modify any other sensitive documents in the library of Company A? Even if the intentions of Company B are completely legitimate, what if there is a bug in the mobile code that inadvertently leads to commercial losses for Company A? After all, the coders of the Knowbot have presumably not been able to test the Knowbot on the computers of Company A. Allegorically, these scenarios are akin to "turning a fox loose in the hen house."

Such concerns about mobile code are not unrealistic. The most familiar examples of mobile code today are computer viruses. Analogous to Knowbots, viruses travel from system to system "carrying out the wishes of their creators" – which are generally disruptive. Most businesses invest heavily in virus detection and firewalls in order to prevent unwanted infections that can compromise their computers and disrupt their

businesses. Even today nearly two decades since A13 was written, mobile code has had very limited use beyond support for user interfaces in web pages[12].

Nor are all of the risks associated with Company A. Suppose that the KNOE (Knowbot Operating Environment) in the computers of Company A has been altered. It could potentially separate a Knowbot from its document, and provide its own altered-Knowbot which acts differently. It could under-report the usage statistics in order to reduce fees. It could alter the Knowbot from Company B and send it on to Company C in a way that intercepts a "funding stream" to the coffers of company A rather than Company B. Allegorically, these scenarios are akin to "sending a lamb to a den of wolves."

A fundamental problem in A13's conception of Knowbots is that they are designed to serve the interests of *one* party. However, commercial transactions *inherently* involve multiple parties with differing interests, such as a producer and a consumer. A13 does not discuss any requirement for robust, verifiable and accountable trust across multiple parties. Furthermore, since programs and programming environments are so complex, the concepts of Knowbots and KNOEs does not appear to be a workable approach for achieving such security[13].

DRM systems like those invented in the 1990s address multi-party security issues in a way more appropriate for commercial use: DRM systems like those invented in the 1990s use host computers configured as trusted systems, rather than as untrusted hosts. Each trusted system is designed and verified by a third party to guarantee to two contracting parties that the trusted system can be relied upon to act as an impartial agent to enforce the terms of any digital contract that has been agreed to by those parties. In this way, DRM approaches address multi-party security issues in a way more appropriate for commercial use.

In summary, the Knowbot architecture of A13 does not recognize that trust in a commercial setting requires a basis of trust across multiple parties. The Knowbot-based

[12] Even the use of mobile code in the limited context of "JavaScript" in web pages introduces security issues.

[13] Even if the "source code" of Knowbots and KNOEs were made available to both parties, since there would be so much variation in Knowbots and environments it would not be practical to prove their correctness. It would also be difficult to prove (for example) that Knowbots were not transported to different environments than the ones that were tested.

architecture contains no provisions for this. As suggested by scenarios above, it appears that the Knowbot approach is not a sound basis for building such systems.

3.2 Communication and Code Security

Libraries generally do not often have active adversaries and their every day operations do not give much attention on security in communication. For example, when a library loans a client a book, there is typically no concern about whether the book has been altered to contain false information. In contrast, commercial organizations have private and critical communication. It is normal security practice to use encryption, redundant check sums, and other techniques in commercial communication systems to assure secrecy and integrity. For example, regular web users are familiar with providing passwords and with the "encrypted page" messages that they get in their web browsers when critical information is requested. Such concepts, however, are outside the activities of traditional libraries. Furthermore, they are not considered or even mentioned in A13.

The lack of attention to cryptography, certification and other means of securing communication has widespread implications for the system design and its applications. For example, consider again the transmission of mobile Knowbots. Knowbots are proposed for many functions in the digital libraries – used for retrieval and search, to billing, and even as couriers. When a Knowbot is being transmitted from one system to another, it is represented by bits in packets over the communications network. Today, several years of experience with computer viruses has made designers of computer systems aware of a myriad of possible "attacks" on communications systems[14]. By various means, communications can be intercepted, faked, copied, altered, and blocked. Lacking such understanding, A13 does not recognize that its Knowbots are vulnerable to being intercepted, faked, copied, altered, or blocked.

To restate the design issue, A13 proposes Knowbots as the security mechanism to act on behalf of owners. However, Knowbots are software-only entities. They are expected to travel over networks where they are vulnerable to modification, and to run on many

[14] For example, see Chapter 3, "The Digital Wallet and the Copyright Box: The Coming Arms Race in Trusted Systems" in Stefik, Mark. *The Internet Edge: Social, Technical, and Legal Challenges for a Networked World.* Cambridge, Ma. The MIT Press, 1999. Robert Kahn and Vinton Cerf, the authors of A13, both provided endorsements of the book to the publisher. These appear on the back cover of this book.

different computer systems where they potentially could be modified. This leaves the foundations of security in A13 – the infrastructure that is supposed to look after the interests of content owners – profoundly vulnerable and insecure. In contrast, modern security designs have design features that address security in communication (communication integrity), authentication (behavioral integrity), and hardware (physical integrity).

A13 specifically proposes to leverage the Internet but to integrate the digital library infrastructure with existing technologies.

> Before describing specific features of the Digital Library System, it will be helpful to review some of the fundamental assumptions which strongly affect its design. Perhaps the most dominant of these assumptions are that the system is distributed, heterarchical, hierarchical, networked and strongly display-oriented. In addition, it must have an ability to interact with other autonomous Digital Library Systems that do not adhere to its internal standards and procedures. [A13, page 19]

From a computer security point-of-view, a chain is as strong as its weakest link. Every subsystem that does not meet security standards creates a vulnerability. This is the opposite of the trusted system approach of DRM systems of the 1990s, which require that every system in a transaction be certified as trustworthy.

In summary, the approach taught in A13 is vulnerable to communications attacks which would need to be addressed in a context of commercial use.

3.3 Representing Agreements

When people engage in financial transactions, they often formalize their agreements with contracts. For example, contracts and warranties are common when people rent an apartment, buy a home, take out a loan, or buy an expensive appliance. There are many variations in contracts – in terms of what provisions are included. At the same time, there are often a few key issues, such as the loan amount, fees, and the interest rate and term of a loan. Contracts explain the terms and conditions of the agreements, and lay out the rights of the parties. The substance of an agreement is not hidden. Rather it is written openly in the contracts. Contracts are intended to express the terms and conditions so that any of the parties to a transaction can examine them and understand them.

Contracts do not play a highly visible role in traditional libraries. For example, there are very few variations in the terms and conditions of book loans. There may be some books that cannot be checked out, and there may be some books that have to be returned more quickly than others. However, contracts are not a major focus for a traditional library. When Kahn and Cerf chose the library metaphor for their proposed digital library infrastructure in A13, they did not mention contracts as a design element.

A13 proposes Knowbots to take the responsibility to look after the interests of users and others in the digital library. Absent any other means, Knowbots are A13's only vehicle for representing and enforcing agreements. When there are multiple, different agreements between parties, multiple, different Knowbots would be required.

The problem with this approach is that Knowbots are mobile computer code and computer code is notoriously non-transparent and hard to understand for most people. Most of the code in a program is generally about its internal bookkeeping and managing relations with other objects in its operating environment. To understand what code does, a programmer must understand not only the computer language, but also the operating environment in which the code is operating.

Using Knowbots to represent a myriad of business agreements is not practical. The important and salient elements of human agreements would be buried and essentially obfuscated by putting them into a program. The opaqueness would make it very difficult to understand what agreement is represented by a Knowbot, or even to tell the difference between a genuine Knowbot and a rogue Knowbot.

The DRM systems that were invented in the early 1990s addressed the problem of representing agreements through the use of a declarative digital rights language[15]. These languages were designed to express the kinds of operations, terms and conditions that are salient for practical contracts about the use of digital content. For example, there were specific operations for copying, loaning, printing and other common things. Terms and conditions could express a range of requirements for payments, times of use, and so on.

---

[15] For example, see "Letting Loose the Light" in Stefik's *Internet Dreams* book for an example of a digital rights language. See "The Bit and the Pendulum" in Stefik's *The Internet Edge* book for a discussion of digital contracts.

Digital contracts could be expressed in a grammar. From there they could be presented to people in clear user interfaces. They could also be interpreted and enforced by computers.

In summary, the Knowbot approach is not practical for supporting the negotiation or understanding of agreements. The subsequently-developed rights language approach side-steps the main difficulties of the Knowbots approach by making it unnecessary for people to try to discern the meaning of an agreement from the programming code of a Knowbot.

4. Concluding Remarks

A13 was a visionary proposal for a digital library system. The design was guided by the metaphor of a traditional library as a system to enable people to share information. Like traditional libraries – A13's focus is on exchange of information. True to the purpose of libraries, A13 is not much concerned with an infrastructure for commerce in digital content, which would require much more attention to mechanisms for commerce and to security requirements for transactions among potentially competitive parties. A13 made very different (and often opposite) design choices from many of the DRM systems that were invented in the 1990's and the systems that are deployed today.

Appendix A. Qualifications

Education and Research

My university education was at Stanford University, both undergraduate and graduate. I received my Bachelors degree in Mathematics in 1970 and my doctorate in Computer Science in 1980. I am a Fellow in the American Association for the Advancement of Science (AAAS) and also in the American Association for Artificial Intelligence (AAAI).

I work at the Palo Alto Research Center (PARC), where I am a research fellow. Since I started at PARC in 1980 I have taken several tours of duty in research management, leading three technical areas and one of PARC's laboratories for several years. I occasionally teach courses and give lectures at Stanford University and U.C. Berkeley. I have been an external thesis advisor and dissertation committee member for Ph.D. students at Stanford, U.C. Berkeley, and the University of Maryland.

I have published five technical books including *The Internet Edge: Social, Technical, and Legal Challenges for a Networked World* (MIT Press, 1999), *Internet Dreams: Archetypes, Myths, and Metaphors* (MIT Press, 1996), and *Introduction to Knowledge Systems* (Morgan Kaufmann Press, 1995). I have published over forty technical papers.

Some of my technical work is mentioned in the *Open Architecture for a Digital Library System* paper (A13), which cites Common LOOPS among current object languages. I was one of the main creators of the Loops system and a contributor to the Common LOOPS specification. PARC was a center for much of the research that inspired Kahn and Cerf during this period – including the technologies for distributed computing, the SmallTalk language, and the NoteCards hypermedia system which are mentioned in the paper.

As a computer scientist, I am somewhat of a generalist and have switched my area of focus every few years. A unifying goal in my work has been to enhance the creation and sharing of knowledge. My dissertation work on an expert system for experiment planning included a frame-based knowledge representation system. A version of this was later commercialized by Intellicorp. My research on collaboration in electronic meeting rooms ("Colab") included creating an infrastructure for distributed objects. The Colab research

led to collaboration with Bob Kahn and others in the creating of the "National Collaboratory" projects in the U.S.

My current research on "sensemaking systems" is about technology to help people facing information overload to master and understand large amounts of information in carrying out their work. Our sensemaking projects at PARC are multi-disciplinary, involving computer scientists and cognitive psychologists as well as specialists in natural language technology, user interfaces, and distributed systems. The current directions in this involve what we call "augmented social cognition," which is the technology that aggregates and combines the sensemaking contributions of large groups of people.

Relationship to Robert Kahn and Vinton Cerf

Bob Kahn was a fairly frequent visitor to the Heuristic Programming Project at Stanford University in the mid 1970s when I was a graduate student there. I got to know him since he worked closely with Edward Feigenbaum, who was one of my faculty advisors. In the early 1980's after I had graduated, I was a participant together with about a dozen others in some weekend workshops led by Bob Kahn at Stanford when the early ideas for a "digital library project" were being developed. Some of the other participants at the workshop were professors from Stanford (Edward Feigenbaum, Joshua Lederberg, John McCarthy) and MIT (Marvin Minsky). In this way I was able to contribute in a small way to the early stages of the Digital Library Project, even before the Open Architecture paper was written.

Bob and I have worked together on various projects over the years. I typically see him two or three times a year. When I first developed the concepts of digital rights management in the early 1990s, Bob signed a non-disclosure agreement with the Palo Alto Research Center so that we could discuss the ideas in some depth and also plan participation in some coordinated activities, such as the "digital object identifier" project.

Vinton Cerf was a professor at Stanford in the Computer Science Department when I started there as a graduate student. Initially, my research plan was to do a dissertation in the Systems area and he was my graduate advisor. Within a year or so, however, he decided to leave Stanford to work on developing the Internet (then ARPANET) at DARPA. I switched research areas in Computer Science to artificial intelligence,

focusing on expert systems. We have kept in occasional contact over the years. When I published the book *Internet Dreams* with MIT Press in 1995, he graciously wrote the foreword to the book. This book contained my first publication (beyond patents) of the ideas for digital rights management in the chapter titled "Letting Loose the Light: Igniting Commerce in Electronic Publication." The book also includes an excerpt from Kahn and Cerf's paper *The World of Knowbots*.

In 1999 I wrote a second Internet book – *The Internet Edge* – that provided a deeper analysis of the trends in networks, and discussed social, technical, and legal challenges. Both Robert Kahn and Vinton Cerf provided endorsements of the book to the publisher that appeared on the jacket of the book.

Mark Stefik 30 May 2007

Atomic Proxy Cryptography

Matt Blaze

AT&T Shannon Laboratory

This talk introduces *atomic proxy cryptography*, in which an atomic proxy function, in conjunction with a public proxy key, converts ciphertext (messages in a public key encryption scheme or signatures in a digital signature scheme) for one key (k_1) into ciphertext for another (k_2). Proxy keys, once generated, may be made public and proxy functions applied in untrusted environments. Various kinds of proxy functions might exist; {\em symmetric} atomic proxy functions assume that the holder of k_2 unconditionally trusts the holder of k_1, while {\em asymmetric} proxy functions do not. It is not clear whether proxy functions exist for previous public-key cryptosystems. Several new public-key cryptosystems with symmetric proxy functions are described: an encryption scheme, which is at least as secure as Diffie-Hellman, an identification scheme, which is at least as secure as the discrete log, and a signature scheme derived from the identification scheme via a hash function.

Full paper available.

This is joint work with Martin Strauss.

Matt Blaze, Atomic Proxy Cryptography

Gates 498, 10/20/98, 4:15 PM

ElGamal encryption

From Wikipedia, the free encyclopedia

The ElGamal algorithm is an asymmetric key encryption algorithm for public key cryptography which is based on Diffie-Hellman key agreement. It was described by Taher Elgamal in 1984. The ElGamal algorithm is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm is a variant of the ElGamal signature scheme, which should not be confused with the ElGamal algorithm.

ElGamal can be defined over any cyclic group G. Its security depends upon the difficulty of a certain problem in G related to computing discrete logarithms (see below).

Contents

The algorithm

ElGamal consists of three components: the key generator, the encryption algorithm, and the decryption algorithm.

The key generator works as follows:

- Alice generates an efficient description of a cyclic group G of order q with generator g. See below for specific examples of how this can be done.
- Alice chooses a random x from $\{0, \ldots, q-1\}$.
- Alice computes $h = g^x$.
- Alice publishes h, along with the description of G,q,g, as her public key. Alice retains x as her secret key.

The encryption algorithm works as follows: to encrypt a message m to Alice under her public key (G,q,g,h),

- Bob converts m into an element of G.
- Bob chooses a random y from $\{0, \ldots, q-1\}$, then calculates $c_1 = g^y$ and $c_2 = m \cdot h^y$.
- Bob sends the ciphertext (c_1,c_2) to Alice.

The decryption algorithm works as follows: to decrypt a ciphertext (c_1,c_2) with her secret key x,

- Alice computes $\frac{c_2}{c_1^x}$ as the plaintext message.

The decryption algorithm produces the intended message, since

$$\frac{c_2}{c_1^x} = \frac{m \cdot h^y}{g^{xy}} = \frac{m \cdot g^{xy}}{g^{xy}} = m$$

If the space of possible messages is larger than the size of G, then the message can be split into several pieces and each piece can be encrypted independently. Typically, however, a short key to a symmetric-key cipher is first encrypted under ElGamal, and the (much longer) intended message is encrypted more efficiently using the symmetric-key cipher — this is termed *hybrid encryption*.

Security

ElGamal is a simple example of a semantically secure asymmetric key encryption algorithm (under reasonable assumptions). It is probabilistic, meaning that a single plaintext can be encrypted to many possible ciphertexts, with the consequence that a general ElGamal encryption produces a 2:1 expansion in size from plaintext to ciphertext.

ElGamal's security rests, in part, on the difficulty of solving the discrete logarithm problem in G. Specifically, if the discrete logarithm problem could be solved efficiently, then ElGamal would be broken. However, the security of ElGamal actually relies on the so-called Decisional Diffie-Hellman (DDH) assumption. This assumption is often stronger than the discrete log assumption, but is still believed to be true for many classes of groups.

Generating the group G

As described above, ElGamal can be defined over any cyclic group G, and is secure if a certain computational assumption (the "DDH Assumption") about that group is true. Unfortunately, the straightforward use of $G = Z_p$ for a prime p is **insecure**, because the DDH Assumption is false in this group. In contrast, computing discrete logs is believed to be hard in Z_p, but this is not enough for the security of ElGamal.

The two most popular types of groups used in ElGamal are *subgroups* of Z_p and groups defined over certain elliptic curves. Here is one popular way of choosing an appropriate subgroup of Z_p which is believed to be secure:

- Choose a random large prime p such that $p - 1 = kq$ for some small integer k and large prime q. This can be done, for example with $k = 2$, by first choosing a random large prime q and checking if $p = 2q + 1$ is prime.
- Choose a random element $g \in Z_p$ such that $g \neq 1$ and $g^q = 1 \bmod p$, i.e. such that g is of order q.
- The group G is the subgroup of Z_p generated by g, i.e. the set of kth residues mod p.

When encrypting, care must be taken to properly encode the message m as an element of G, and not, say, as just an arbitrary element of Z_p.

Efficiency

Encryption under ElGamal requires two exponentiations; however, these exponentiations are independent of the message and can be computed ahead of time if need be. The ciphertext is twice as long as the plaintext, which is a disadvantage as compared to some other algorithms. Decryption only requires one exponentiation (instead of division, exponentiate c_1 to $q - x$). Unlike in the RSA and Rabin systems, ElGamal decryption *cannot* be sped up via the Chinese remainder theorem.

Miscellaneous

ElGamal is malleable in an extreme way: for example, given an encryption (c_1,c_2) of some (possibly unknown) message m, one can easily construct an encryption $(c_1, 2 \cdot c_2)$ of the message $2m$. Therefore ElGamal is not secure under chosen ciphertext attack. On the other hand, the Cramer-Shoup system (which is based on ElGamal) *is* secure under chosen ciphertext attack.

See also

- ElGamal Signature scheme

References

- Taher ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, v. IT-31, n. 4, 1985, pp469–472 *or* CRYPTO 84, pp10–18, Springer-Verlag.
- Handbook of Applied Cryptography (http://www.cacr.math.uwaterloo.ca/hac/), contains a detailed description of ElGamal Algorithm in Chapter 8 (http://www.cacr.math.uwaterloo.ca/hac/about/chap8.pdf) (PDF file).

Public-key cryptography

Algorithms: Cramer-Shoup | DH | DSA | ECDH | ECDSA | EKE | **ElGamal** | GMR | IES | Lamport | MQV | NTRUEncrypt | NTRUSign | Paillier | Rabin | Rabin-Williams | RSA | Schnorr | SPEKE | SRP | XTR

Theory: Discrete logarithm | Elliptic curve cryptography | RSA problem

Standardization: ANS X9F1 | CRYPTREC | IEEE P1363 | NESSIE | NSA Suite B **Misc:** Digital signature | Fingerprint | PKI | Web of trust | Key size

Cryptography

History of cryptography | Cryptanalysis | Cryptography portal | Topics in cryptography

Symmetric-key algorithm | Block cipher | Stream cipher | Public-key cryptography | Cryptographic hash function | Message authentication code | Random numbers

Category: Asymmetric-key cryptosystems

O12 Evidence with regard to computer network architecture, satellite communications and digital television distribution

EXTENDED ABSTRACT

A Secure Distributed Capability Based System

Howard L. Johnson
Information Intelligence Sciences, Inc.
University of Denver, New College

John F. Koegel, Rhonda M. Koegel
Department of Mathematics and Computer Science
University of Denver
Denver, Colorado 80210

A novel design for a secure distributed system is described and evaluated. A capability based computer architecture is combined with cryptographic network security techniques to protect global objects and preserve access rights across system boundaries. The resulting architecture is evaluated against several criteria, including the DoD Computer System Evaluation Criteria. The strengths and weaknesses of the approach are presented.

key words: computer security; distributed system security; capability architecture; network encryption

1. Introduction

A distributed system connects various computing entities in several locations so resources can be shared by users. Distributed computing offers the advantage of flexibility so that each facility can be locally controlled and configured for a specific application. It also offers incremental growth so that additional features can be easily added, usually at a lower cost than upgrading a central host. The connection of distributed systems facilitates information sharing. The physical network can be implemented by point-to-point or multi-point links, LAN's or WAN's.

In a single centralized computing facility, system security is achieved through physical, operational, and system controls. System controls include operating system functions such as login passwords, file system protection, and memory management. In a distributed environment, these controls can still be effective for securing each specific system. However, additional problems arise because of the interconnection of systems and the information flows between systems.

There are two areas of concern in securing a distributed system. The first, that of securing the network facilities, has received greater attention in the literature. This need stems from the fact that physical facilities in most prevalent use today as communication media (land lines, microwave links, and satellite channels) offer little protection for themselves [1]. To secure these facilities, some type of cryptography is employed. The user who wishes to obtain an off-the-shelf solution to the problem can use a conventional substitution-permutation algorithm, such as the NBS's DES [2] or a public key algorithm such as RSA [3]. Although there is active research in both breaking and strengthening these techniques, for many applications currently available methods will suffice.

Even with encryption, a network is still vulnerable to certain types of threats against the communications protocol being employed [4]. Conventional link-level protocols only allow the data field to be encrypted, while control and address fields are transmitted unencrypted. This leaves a network open to such attacks as message modification and message replay.

The second area of concern, that has received relatively little attention in the literature, is the control of information protection across system boundaries. Within a given computer facility, the operating system can be used to enforce uniform and constant protection of information. However, once the information is removed from the computer, these controls no longer apply. Protection of information can only be maintained in a local environment. It would be preferable if access rights could be enforced across system boundaries. This would produce a secure distributed system and protect proprietary software and data.

Consider the case of a remote database user who has purchased read access to certain information in the database. If the user accesses the

1985 ACM 0-89791-170-9/85/1000-0392 $00.75

database with a personal computer, it is a straightforward step for the user to read the database and store the information in the PC. Once the user has a local copy, then he/she is free to distribute this data to any other party, regardless of whether that party has purchased access to the database. Thus, the access protection of a single system is easily violated by availability of distributed computing.

The database owner could protect his/her investment by requiring the user to purchase a proprietary interface program to access and manipulate the data. Not only does this restrict the user and provide an economic deterrent to the sale of information, it also makes this protection dependent on the copy protection of the interface program.

Another example is if the host is used as a central distribution point for software, possibly for a CAI application. Once a module is removed from the host, it is very difficult to limit the production of duplicates. Encryption of the key elements of a program has been proposed as a solution [5]. However, not only does this place additional burden on the applications programmer, but also requires a design that may not be met by many programs.

Most secure network strategies deal only with encryption of data as it is transmitted across network facilities, and not at all with the management of protection across system boundaries. However, there are numerous instances of distributed information system security and proprietary software protection not solved by network encryption. The authors believe that an integrated solution involving both capability-based computers and network protection using encryption and a secure protocol can provide distributed system security.

After discussion of network security and capability architectures in a distributed environment, we present an integrated design of a secure distributed capability based system. The resulting architecture is evaluated against several criteria, including the DoD Computer System Evaluation Criteria.

2. Network Security

2.1 General

This paper pertains to hardware (and a few hardware/software) data system security protection mechanisms, but design must be accomplished in the context of existing or proposed physical security, personnel security, operations security, emanations security, and communications security. The implementation of each guides implementation of the network data system security. A key criteria is minimized degradation in throughput and response.

A brief summary of network security follows. The term "association" is used to refer to a (potentially bidirectional) end-to-end data path through the network. The reader is directed to Voydock and Kent [4] and to Davies and Price [6] for a more complete treatment of these topics.

2.2 Threat

From [4], passive attacks to network security are intended to bring about the unauthorized release of information or authorized release of information sufficient to perform a traffic analysis. Passive attacks usually cannot be detected but can be prevented.

Active attacks include unauthorized modification of information, unauthorized resource use denial, and attempts to initiate spurious associations. Active attacks cannot be prevented, but can usually be detected. In a network environment we are equally concerned with threats internal to the system as those outside.

2.3 Protection Principles

2.3.1 Encryption Techniques

Rushby and Randell [7] observed that separation is one of the key elements in enforcing a secure system, and that four separation methods exist: physical, temporal, cryptographical and logical. In a communication system, physical separation is the most desirable, but unless the system is completely contained in a secure building environment or in a specially constructed tunnel vault, the distances involved leave too much line unprotected.

Some transmission media are more secure than others, such as fiber optics, directional satellite links, and exotic military communications systems, but each has a reasonable vulnerability to capture or disruption of data flow. Within a secure environment, either logical or cryptographical means can be employed to protect data authenticity. Methods analogous to periods processing can make use of transmission links for different levels of control at different isolated periods of protection, performing necessary cleansing of storage registers or buffers, if any exist.

Data encryption is the primary means by which communicated data are protected. It directly prevents passive attacks by preventing an intruder from seeing data in the clear. Data patterns can be masked by using a unique key for each association, employing cipher block chaining which causes each encrypted value to be a complex function of previously encrypted data, and appropriately selecting the proper initialization vector for chaining.

There are three ways to incorporate cryptography into a communications system: link, node and end-to-end encryption. In link encryption, cryptographic devices bracket a communication line between two nodes. Node encryption uses a protected security module to absolutely protect data at the node. In end-to-end encryption, data are deciphered only at their final destination, requiring several keys at each origin and destination.

There are several tradeoff variables in choosing between link, node, and end-to-end encryption:

- the number of encryption units required (and therefore the potential response degradation)
- the number of keys required by each node originating and receiving data
- the complexity required in specifying a routing path independent of the specification of data, or alternately the overhead in interim decryption attempts.

The number of security devices are fewer in end-to-end encryption, but number of keys required is greater. Addressing information must be developed independent of the data, or interim decryption attempts must be made. Both create a difficult design problem. Link and node encryption are normally transparent to the user, but so is end-to-end encryption if initiated by system services. The message and its header can both be encrypted with node encryption; however, with link encryption and end-to-end encryption normally both message and header are encrypted. The exception is a technique whereby each node attempts to decrypt the message and passes it if unsuccessful or if the successfully decrypted message indicates another addressee. If not all nodes have encryption facilities or if encryption of only selected messages is desired due to overhead, an additional mechanism is required to enable and disable the encryption function.

Voydock and Kent [4] observed that a communication network can also be viewed as providing a medium for establishing associations between protocol entities. An association oriented approach constitutes a refinement to end-to-end measures. It not only protects the path, but reduces the probability of undetected cross talk, whether induced by hardware or software.

2.3.2 Detection Techniques

If the communications header is in clear form, transmitting bogus messages helps prevent traffic analysis. The protocol layer selection determines the precision with which traffic analysis can be done. If encryption is performed in the presentation layer, an intruder could determine which presentation, session, and transport entities were involved. Performing encryption in the transport, network, or link layers limits the intruder to observing patterns at the network address levels. Contradistinctively, the higher the layer, the more of the path protected.

To prevent message stream modification, there are measures that ensure message integrity. Measures that ensure message authenticity rely on the integrity measures. Measures that ensure message ordering rely on both of the previous measures. Countermeasures involve use of unique keys, sequence numbers, and error detection codes.

Denial of service attacks often can be detected by message stream modification countermeasures. If the attacks begin when an association is quiescent, a request response mechanism must be employed.

For spurious association attacks, hierarchic or public key systems can defeat attempts to establish an association under a false identity. Timestamp, checksums, and/or random challenge-response mechanisms detect playing back of a previously legitimate association-initiation.

A covert channel allows a process to transfer information in a manner that violates the systems security policy. A covert timing channel is a covert channel in which one process signals information to another by modulating its own use of system resource (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process. Covert channels with low bandwidths represent a lower threat than those with high bandwidths. In any complex system there are a number of relatively low-bandwidth covert channels whose existence is deeply ingrained in the system design. Faced with the large potential cost of reducing the bandwidths of such covert channels, it is felt that those with a maximum bandwidth of less than one bit per second are acceptable in most applications environments [8]. The channel bandwidth can be reduced by introducing noise, or complicated traffic patterns, making it difficult to detect and extract deliberate modulation.

These measures provide security only in a probabilistic sense, providing a high probability that the intruder cannot subvert the encryption algorithm and that active attacks will be detected. The goal is to make it more difficult for the intruder to break the system than to create the information through other means.

2.4 Protection Mechanisms

2.4.1 Reference Monitors

A reference monitor [9] must be tamperproof, must always be invoked, and must be small enough to be subject to analysis and tests, the completeness of which can be assured. The reference monitor is the most popular type of authentication mechanism. Interaction is generally only with the message header, whereas cryptographic compatibility serves to authenticate an entire message. Further, data can remain encrypted for continued protection while in buffers, storage, and internal communications. The reference monitor allows such things as separate encryption of the message without the header and requires neither the time and cost spent in encryption nor the cost of a key management and distribution system.

2.4.2 Authentication and Secrecy

Cryptography can not only be used for security, but can also be employed for authenticity. Solutions using encryption are equally applicable to local area networks as they are to large long-haul communications networks. Different applications lead to different solutions, as do design tradeoffs based on changing technologies (e.g., fiber optics), speed, cost, and level of protec-

tion. The following are key topics associated with cryptography.

Secrecy and Authentication - Secrecy exists when it is computationally infeasible to determine the deciphering transformation. Authenticity exists when it is computationally infeasible to determine the enciphering transformation. The latter establishes the validity of a claimed identity (e.g., of the sender in a digital signature or user verification application).

Substitution-Permutation Ciphers (e.g., the DES) - Information theory has allowed theoretical data protection to any degree desired, based on the length of the key and repeated application of the algorithm steps; even when the algorithm is known to the perpetrator. This class of cipher has been implemented into a very fast chip. As cryptanalysis capability increases, the dimensionality of the implementation can be increased, with a corresponding loss in efficiency, (unless microcircuit technology makes up the difference). A DES block cipher breaks the message into blocks and enciphers each with the same key. A stream cipher breaks the message into characters or bits and enciphers them with successive elements of a key stream (which might be the prior encrypted text as in the cipher block chaining mode and the cipher feedback mode of the DES).

Public Key Ciphers (e.g., the RSA scheme) - These methods of protection provide both secrecy and authenticity. Several public key ciphers have fallen prey to cryptanalysts, but the RSA cipher stands a good chance of surviving these attacks based on the mathematical history of factorization of large numbers (although a surprisingly large number was factored on the Cray at Sandia Laboratories recently). Keys are large and computation still relatively complex. Technologies such as gallium arsenide and parallel bit stream implementations should solve immediate speed problems, however, as cryptanalysis comes closer, the size of the prime numbers must be increased.

One-way Ciphers - These virtually unknown, but simply implemented ciphers are important to design because once data are encrypted they cannot be simply decrypted, even by the originator. They are useful in applications, for example, where authentication of passwords can be accomplished by comparing pairs of encrypted data values. Certain simple functions such as comparison can be accomplished in encryption space.

2.4.3 Key Management Design

The responsibility for key management depends on the security policy and the choice of implementation. Unless keys are given at least the same level of protection as the data, they will be the weak link. Once the penetrator has gained access to the key (generally a very small piece of data) he has gained access to all data. Techniques of generating, transmitting and protecting keys include host keys, hierarchical key protection, partitioning of keys for different protection levels, and diverse means by which the key management system interfaces with the rest of the system [7].

In the normal implementation of public key systems, the public key is published with no protection whatsoever. The private key is originated and held by only one person. Certain implementations require distribution of the private key under a protected key distribution scheme, especially where the private key is used within the processor as a means of both secrecy and authentication of source or another system variable.

A third party or a host can provide the authentication necessary for key distribution. There are several established approaches for the implementation of a distributed session key system, appropriate to network communications. The public key system has the property that two parties can establish a secret key for use in a unique session between them, obviating involvement of a third party. The strategy can be repeated often for a greater degree of protection. Prolonged use of a single key makes a system more vulnerable to cryptanalysis. The degree of added vulnerability depends on the cryptographic technique used, which in turn is related to the nature of data transmission, intercommunication requirements, and security inherent to the communications system.

2.5 Network Protocol Considerations

In late 1970's, the International Standards Organization adopted a network architecture known as the reference model for open system interconnection, ISO/OSI. Layers 1 to 3 are concerned with data transmission/routing and deal respectively with physical, data link, and network concerns. Layer 4 provides end-to-end control of data transport. Layers 5 and 7 are the session, presentation, and application layers. Some of the possible approaches to implementing security under ISO/OSI are as follows:

Layer	Protocol	Security
7	Applications Services	User identification, encryption of stored data, key distribution.
6	Presentation Formats	User controlled use of encryption for secrecy and identification including a user request for encryption.
5	User Session Control	Establishing secrecy and authentication during the conduct of a session between system users (people and programs). The most desirable encryption point in high level protocols [4].

4	Transport Flow Control	
3	Network Routing	
2	Data Link Control	Security control entirely in the communications systems such as link encryption where the data are protected between adjacent network nodes and are decrypted and re-encrypted at each node. Security control entirely in the network communications use of node encryption schemes where data are not in the clear at an intermediate node, but are rather decrypted and re-encrypted by a special security module.
1	Physical Connection	

Design should be such that acceptance of data or requests into the memory associated with a node should be based on the assurance that the transaction is legitimate and does not violate the security policy. An example of protocol layer 2 (data link) encryption is provided in [10], in which source and destination subnets and trusted interface units are designated in the packet formats for the carrier sense multiple access with collision detection (CSMA/CD) protocol. The protocol also specifies the data security level.

Popek and Kline [11] identified the important issues to be addressed in defining secure protocols:

- establishing initial cleartext/ciphertext/cleartext channel from sender to receiver
- passing cleartext addresses without providing a leakage path
- determining error recovery and resynchronization mechanisms to be employed
- performing flow control
- closing channels
- interaction of the encryption protocols with the rest of the protocols
- dependence on software in implementation.

3. Capability Architectures

3.1 Description

A capability-based computer uses an architecture in which objects are addressed by means of a two-component entity called a capability. One component of the capability is a unique object identification number which is translated by the hardware into an actual machine address. The other component of the capability can be viewed as an access rights field which identifies to the hardware the operations that the owner of the capability may perform on the object.

Capability architectures have been promoted for a number of reasons including their hardware support for object-based programming [12] and system security [13]. A capability-based computer offers greater generality than does a conventional computer architecture. This generality includes hardware support for object identification and management which allows the user to approach the machine interface at a higher level of abstraction. By encapsulating objects and defining unique object identification numbers, the system can provide a more secure hardware base on which to place the operating system.

To maintain system security and integrity, it is typical for a capability-based computer to use hardware tagging of capabilities stored in memory [14,15]. When a user attempts to use a capability to reference an object, the hardware tag indicates that use of the capability is a legal one. The capability itself will be further compared with the operation that the user is attempting to ensure its validity. Since the tag controlled by hardware, the user is not able to arbitrarily modify the tag bits associated with a memory address. If the user attempts to modify a capability, the hardware will reset the associated tag bits.

Another feature of capability architectures is that the machine interface is usually implemented at a higher level than that of a conventional architecture. This higher level includes functions that relate to object addressing and object management. By placing greater functionality in the firmware, the goal is to improve the performance of the architecture while ensuring that the object related operations can not be interrupted and possibly altered by another process. Thus, the security of a capability-based computer follows the precept that hardware is inherently more secure than software.

3.2 Design Issues

There are a number of issues to be faced by the designer of a capability machine. These include:

- generating and maintaining unique object id's for a large number of objects
- managing objects, including object deletion and the dangling pointer problem

- controlling the copying of capabilities for object sharing
- defining object categories
- speeding-up object address translation
- permitting called programs to have more access rights than their callers for operating system functions
- providing object encapsulation to promote object protection.

The resolution of these issues can take various forms. Levy [16] surveys many of these in his book on capability based systems.

3.3 Goals

For the purposes of a discussion of capability system goals, we assume that the network facilities for the distributed system have already been secured using encryption and secure high-level protocols as described in the previous section. By employing capabilities for defining and protecting objects in a distributed environment, the following goals can be achieved:

- Objects can be transferred across system boundaries while preserving access rights across these boundaries. This is accomplished by forcing any object transfer between systems to be accompanied by the transfer of the capability needed to access the object. Without this capability, the object can not be accessed.

  The process performing the copy operation must possess the original capability on the source computer to effect the copy operation. The capability which results on the destination computer must uniquely identify the copied object and must have access rights equal to or less than those of the original capability. The network interfaces for each host are responsible for checking the validity of the operation. The network interface at the destination must generate a unique object id (possibly using already existing firmware for object creation) and must translate the source capability accordingly. At the same time it must preserve or decrease the access rights of the translated capability.

- Capabilities for objects can be transferred across system boundaries. This allows capabilities to be used to reference remote objects. This requires that the capability contain a field which identifies the network node containing the object. Alternatively, the capability could reference a local "network reference object" which would contain the information needed by the operating system and network interface to address the remote object.

- Objects can be referenced across system boundaries using either user-local or user-remote capabilities for these objects. This is analogous to a distributed file system, but is generalized to all the object categories defined in a given architecture.

  A user-local capability is one which is contained in the user's capability list in the local host from which the object reference is being made. Similarly, a user-remote capability is one that is contained in the user's capability list on the remote host that contains the object being referenced. Capabilities used to access objects created remotely are derived from the capability generated by the system where the object was created.

In describing these goals, it is assumed that object identification and addressing are defined locally. When a capability is transferred between systems, a new object id will be created by the destination host automatically. This object id will have meaning only in the context of this host. This will preclude the need for designing a universal object identification scheme that would be impractical both in terms of the size of the id needed and the overhead to coordinate the use of id's. It is also assumed that a capability can be safely and accurately transmitted between systems. The network interface for the capability-based computer controls the encryption and protocols needed to effect secure communications.

To support the preceding goals, a number of issues need to be addressed. First, in keeping with the fine granularity of capability access rights, it would be beneficial to define additional access rights that deal with network operations. These might include the capability to copy an object or the capability itself across the network interface. Access rights for remote operations on capabilities or objects might also be defined this way. Controlling the copying of a capability across a network interface has the same implication as controlling it between users on a single system.

Second, in some systems, an object can be given its own capability list for accessing whatever objects are needed in its operations. When the object is copied from one system to another, is this capability list also preserved? Although it may be desirable to define a network copy operation for capability lists, it does not seem advisable to automatically copy this list and translate it when the object itself is copied. This should be a separate operation, if done at all.

In translating a capability copied from one system to another, there are a number of conditions to be observed. First, the translated capability should never be greater than the original capability. This would violate the basic security principles of capability-based architectures. Second, the process receiving the copied capability should not be able to increase its access over any other objects by means of the copy operation. The situation where the copying of a capability gives the owner greater privilege must be avoided. Finally, if the two computers do not define their objects in the same fashion

(heterogeneous distributed capability system case), the host receiving the capability must translate it to an equivalent or lower object and access rights pair, or else reject the operation.

4. A Secure Distributed Capability System

4.1 Integrated Design

In this paper we deal with distributed systems of user terminals, processing hosts, storage elements, and other resources. The processors and terminals may be heterogeneous or of a compatible family. Our goal is to consider a design based on a combination of cryptography and a capability based control to provide network security.

There is a strong desire in a distributed system for the system to be transparent to the user. Rushby and Randell [7] established that network transparency is most easily achieved if all system components have a common interface. The "recursive structuring" principle for the design of distributed systems states: each component of a distributed system should be functionally equivalent to the entire system of which it is a part. This does not preclude heterogeneous sub-elements, since each system interface must contain provisions for exception conditions to be returned when a requested operation cannot be carried out. The value of the recursive structuring of a system is that, by definition, it is indefinitely extensible.

To use the capability approach in a distributed environment, additional capability categories are needed. These include definitions that protect the network interface and that validate specific network operations:

- network interface to a specific node can be used
- network parameters can be modified, examined, or tested
- capability can be copied across network
- object can be copied across network
- object can be used remotely
- object can be deleted remotely if user has delete capability
- capability can be translated (needed by network interface)
- network object (for referencing remote objects) can be created, managed, or deleted
- audit trail enable.

The network interface design should follow the standard seven-layer ISO OSI model. It will be subject to the same protection that the operating system is given on a capability machine, plus additional protection provided by whatever capabilities are required to use the interface. The various network protocol layers should be designed to promote detection of active network attacks. Data encryption can be built into the user session layer.

All network operations which require capability checking for validation are passed by the network interface to the operating system and/or firmware. Outgoing network transactions are checked in the normal way by comparing the attempted operation with the capability list of the agent process. Incoming transactions that involve the copying of a capability from a remote system will also involve the translation of the object identification within the capability and the object encapsulation to a valid object identification for the destination host. This translation will also be a firmware function that most closely resembles object creation.

4.2 Multilevel Considerations

If a distributed capability system were used in a multilevel security environment, both network security mechanisms and the capability architecture would need to be enhanced to recognize and protect objects of different classification levels.

Here we review some of the characteristics of a multilevel secure system and then discuss its relation to the one proposed. Users are assigned levels, some resources are assigned maximum levels and one must keep track of the high watermark (highest level received since cleansing) of the device. Objects have levels indicated by labels. A process keeps track of the high watermark of objects used in a current period. Users can specify the level of an object created and a process can specify the level of the objects it creates (which must dominate, i.e., be greater than or equal to, the current high watermark). There are several other details that pertain to specific implementations that will not be dealt with here, such as the principals that control the flow of data based on dominance rules.

The protection domain extends across the network, encompassing its nodes. Capabilities are used to determine transmission of objects across nodes, the same as they are within a node. The transmission is not allowed if the process does not possess the capability (e.g., the high watermark is greater than the security level of the destination). At the receiving node the processes cannot have access to the object without the appropriate capability.

Encryption for authenticity, key passing, and secrecy protection is within the encapsulated portion of the capability protocol, implemented in firmware. Also, detection techniques such as those discussed earlier -- unique transmission key, sequence numbers, error detection, request response, and time stampa -- are implemented and initiated at that level.

Encryption is at the user session protocol (layer 5), so that there is end-to-end encryption between geographically separate parts of the protection domain. The capability system would communicate the necessary protocol information to the

transport and other lower layers, providing the necessary protocol parameters.

Modifications to the capability hardware would consist of additional types of capabilities and additional bits to the object identification field of the capability. When a user account is created on a system, the profile of that user would be given capabilities to read, write, create and delete objects of specific classification levels. The capability to perform an operation at one classification level would allow the same operation to be performed at a lower level, provided that an indirect data leakage did not result. The user could also be given the capability to create objects, which could also be given the capability to read, write, create and delete sub-objects of different levels, all of which must be dominated by the user's own capabilities.

When an object is created, it would be created at a given classification level. This level could be economically encoded in the object identification field (2 bits provides 4 levels), which would also be encapsulated with the object itself. Thus, when any data transfer operation is performed on a given object, the object's classification level is used to insure that a legal data flow is occurring.

Additional capabilities would be needed to permit the changing of an object's classification level. Both the classification checks and capability tests would be performed by firmware. The rules governing legal and illegal data movements between levels would also be stored in firmware.

5. Evaluation

Just as the user community is slow to accept some of the most obviously beneficial computing improvements, it is felt that part of the task in portraying an unfamiliar way of thinking is to show consistency with present approaches. Rushby and Randell [7] have described a distributed computing system composed of small trustworthy security mechanisms linked together to provide multilevel security in such a way that the entire system appears as single system to its users. A prototype has been successfully demonstrated. Key to this system are separate security processors, operating in parallel with the general purpose processors, and a software subsystem "the Newcastle Connection," that links multiple UNIX systems, and does not require applications programs or operating system to be changed.

The Department of Defense Trusted Computer System Evaluation Criteria [8] will serve as a standard for the accreditation of commercial systems, at least in the near term, thus it was considered important to compare this system against those criteria. We have also considered Saltzer and Schroeder's [17] principles of design.

5.1 Definitions [8]

"Trusted Computing Base - All protection mechanisms within a computer system (including hardware, firmware, and software) the combination of which is responsible for enforcing the security policy." The cryptographic capabilities network can be considered a trusted computer base, but has an unusually large scope in that it encompasses a network.

"Domain - The set of objects that a subject has the ability to access." An object is defined here as a passive entity that contains or receives information, for which access potentially implies access to the information it contains. The capabilities system considers domain in the same context, however, it further specifies and controls resources and enforces the extent and type of access.

"Dominate - Security level S1 is said to dominate security level S2 if the hierarchical classification of S1 is greater than or equal to that of S2 and the non-hierarchical categories of S1 include those of S2 as a subset." A dominant capability can be enforced categorizing object id's into the appropriate classifications. Another approach would be to define a capabilities base at each independent level. In either case, the capabilities system can further restrict usage to what is required by a task.

"Reference Monitor Concept - An access control concept that refers to an abstract machine that mediates all accesses to objects by subjects." The hardware, firmware, and software elements of a Trusted Computing Base that implement the reference monitor concept are referred to as the security kernel. The capabilities based system employs and enforces a reference monitor type of control, independent of special hardware (although special hardware may be required to enhance performance).

"Star Property - A Bell-LaPadula security model [18] rule allowing a subject write access to an object only if the security level of an subject is dominated by the security level of the object." This rule can be enforced in a capabilities based system, but the implementation must place capabilities in control of the system and not the user.

5.1.2 Requirements [8]

"Discretionary access control - The trusted computer base (TCB) shall define and control access between named users and named objects. The enforcement mechanism shall allow users to specify and control sharing of those objects." Capability access control involves restricting access to objects or resources based on the possession of a ticket that unconditionally authorizes the possessor (user or process) access to the named object with specific rights, where objects include both resources and data. The list is actually inverted from the normal access control list, but contains at least the same information. It can be used by the operating system to emulate the discre-

tionary access model. If the system places the user "in charge", he can establish his own policy with respect to the capabilities possessed by him. In most DoD implementations, however, only a special user (the security officer) can pass capabilities to a user that has not previously possessed them at that level.

"Object Reuse - When a storage object is initially assigned, allocated, or reallocated to a subject from the TCB's pool of unused storage objects, the TCB shall assure that the object contains no data for which the subject is not authorized." This requires cleansing of the resource upon reallocation.

"Labels - Sensitivity labels associated with each ADP system resource that is directly or indirectly accessible by subjects external to the TCB shall be maintained by the TCB and shall be used as the basis for mandatory access control decisions." The assignment of capabilities can be based on the sensitivity of resources. The sensitivity labels can be built directly into the encapsulation scheme as a standard part of the object control. The resources are assigned virtually with the security manager having ownership of the assignment table with the right of revocation and reassignment.

"Label Integrity - Sensitivity labels shall accurately represent security levels of the specific subjects or objects with which they are associated. When exported by the TCB, sensitivity labels shall accurately and unambiguously represent the internal labels and shall be associated with the information being exported." As stated before, the sensitivity labels can be inherent to the definition of the capabilities and become part of the encapsulation scheme. The capability system enforces the authorization for exportation.

"Exportation of Label Information - The TCB shall designate each communications channel and I/O device as either single-level or multilevel, with changes done manually and any changes auditable. When the TCB exports an object to an I/O device, the sensitivity label associated with that object shall also be exported and, in the case of multilevel devices, shall reside on the same physical medium as the exported information and shall be in the same form (i.e., machine readable or human readable form). When the TCB exports or imports an object over a multilevel communication channel, the protocol used on that channel shall provide for the unambiguous pairing between the sensitivity labels and the associated information that is sent or received." This functionality can be incorporated in the capability system. The capability system enforces the transfer request, whereas a conventional system may not.

"Device Labels - The TCB shall support the assignment of minimum and maximum security levels to all attached physical devices to enforce the constraints imposed by the physical environments in which the devices are located." This is indirectly accomplished by the assignment of capabilities. This corresponds better with non data processing information control.

"Mandatory Access Control - The TCB shall enforce a mandatory access control policy over all resources (i.e., subjects, storage objects, and I/O devices) that are directly or indirectly accessible by subjects external to the TCB." External subjects become internally controlled by the capabilities list when they are given the capability of access, otherwise they possess none.

"Identification and Authentication - The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall maintain authentication data that includes information for verifying the identity of individual users as well as maximum security levels to all attached physical devices." The identification must be part of the issuing of capabilities. The association with devices is more restrictive than simple security levels.

"Trusted Path - The TCB shall support a trusted communications path between itself and users for use when a positive TCB-to-user connection is required. Communications via this trusted path shall be activated exclusively by the user or the TCB and shall be logically isolated and unmistakably distinguishable from other paths." Since user consoles are resources, and because of the cryptographic requirements of this system, this requirement is rigidly enforced.

"Audit - The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of access to the object it protects." The audit trail will be a capability assigned solely to the security control function.

5.2 Principles of Design

Saltzer and Schroeder [17] identified several design principles for protection mechanisms. Following is an evaluation of this approach against those criteria:

Least privilege - The capability system enforces this principle to a greater extent than existing implementations.

Economy of mechanism - This architecture supports security control to a far greater degree than general architectures and therefore should be verifiable. In general, hardware is simpler to verify than software or software/hardware mechanisms.

Complete mediation - This requirement is a basic design principle.

Open Design - The design is completely open and does not depend on any secret parts.

Separation of privilege - Satisfaction of this requirement is moot, although the implementation depends on the technique for allocation of capabilities and identification when logging on the system. The implementation of labels and a consistency check against user identification should satisfy this requirement.

Least common mechanism - The mechanism is protected and each user has a separate virtual capability. The concept of distributed control in physically distributed elements tends to support this principle, but certainly not to its ultimate intent.

Psychological acceptability - The mechanism cannot be bypassed and is transparent to the user.

5.3 Advantages and Disadvantages

A capability approach to distributed system security offers strong object protection in both local and distributed contexts. This strength derives from firmware support of access rights at the machine addressing level. In addition, the design offers greater granularity of access rights than is found in a conventional operating system.

A distributed capability system is not without its complications. One potential problem is the vulnerability of capabilities as they are transmitted across the network. This is analogous to the problem of password transmission across a network in a conventional system. Both can be solved by encryption.

Another possible problem is the translation of capabilities in an environment of heterogeneous capability machines. Because object categories may vary from machine to machine, the difficulty is in preserving the meaning of the capability when it is translated. From a security standpoint, security is not compromised if the original capability dominates the translated capability.

A more difficult situation is the linking of a conventional computer to a network of capability systems. Since conventional operating systems do not support the same granularity of protection, meaningful sharing and strong security will probably not be compatible goals. The conventional computer will be the Achille's heel of the distributed capability network if remote object references are uncontrolled.

A final issue is the translation of the capability list for an object that is being copied from one system to another. For efficiency reasons, we have considered it advantageous for the copy operation to copy only the object and the capability for its use, and ignore the capability lists belonging to the object and any of its creations.

6. Summary

The meshing of capability characteristics and a cryptographically supported network is natural. Cryptography will support network communications and detection functions using public key systems or trusted interface modules to provide satisfaction of security protection from the outside world, as well as authentication functions. The capability based resource control provides a simpler environment than that dealt with by a discretionary kernelized system. There is a natural checking mechanism for determination of system misuse and simpler recovery in the event of a malicious internal attack. The system can be changed as the security policy changes without hardware/software modification.

A capability approach can provide a distributed system where data originators or some central authority determine the data, program, and sharing policy. The distributed capability system described here solves the problem of preserving access rights across system boundaries, since an object can not be referenced or copied across the network interface without processing the capability for a specific operation. In comparison to a conventional operating system, a capability based design offers greater protection and more granularity.

With proper implementation, the system also appears to be capable of supporting the DoD trusted system requirements under the unique DoD security policy implementation. Further, a properly architected capability machine and network interface could provide a secure multilevel distributed system. The DoD security requirements could be met by a design including the following provisions:

- Star property should be enforced by the system through assignment of high water mark levels to capabilities, objects, and resources.
- Sensitivity labels need to be integrated into the capabilities protection mechanism, and then be supported accordingly.
- User identification and authentication must be part of the capability issue and usage mechanism
- End-to-end encryption needs to be integrated and network protocol interfaces need to be developed

References:

1. Grayson, W.C., "Vulnerabilities of Data Telecommunications," in Advances in Computer Security Management, V2, ed. by M.M. Wofsey, John Wiley, 1983, 161-172.

2. "Data Encryption Standard," FIPS PUB 46, National Bureau of Standards, Washington D.C., January 1977.

3. Rivest, R.L., A. Shamir, and L. Adelman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications ACM, V21(2), 120-126, February 1978.

4. Voydock, V.L., and S.T. Kent, "Security Mechanisms in High-Level Network Protocols," ACM Computing Surveys, Vol. 15, No.2, June 1983.

5. DeMillo, R., R. Lipton, and L. McNeil, "Proprietary Software Protection," in Foundations of Secure Communication, ed. by R. A. DeMillo, et al, Orlando, FL: Academic Press, 1978, 115-132.

6. Davies, D.W., and W.L. Price, Security for Computer Networks, John Wiley and Sons, 1984

7. Rushby, J.M., and B. Randell, "A Distributed Secure System," Computing Laboratory, University of Newcastle upon Tyne, December 1982.

8. Department of Defense Trusted Computer System Evaluation Criteria, CSC-STD-01-83, 15 August 1983

9. Anderson, J.P., Computer Security Technology Planning Study, ESD-TR-73-51, Vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, Bedford, Mass., October 1972.

10. Gasser, M., and D.P. Sidhu, "A Multilevel Secure Local Area Network," Symposium on Security and Privacy, 1982

11. Popek, G.J., and C.S. Kline, "Encryption Protocols, Public Key Algorithms, and Digital Signatures in Computer Networks," in Foundations of Secure Communication, ed. by R. A. DeMillo, et al, Orlando, FL: Academic Press, 1978, 133-154.

12. Organick, I. E., A Programmer's View of the Intel 432 System, New York: McGraw-Hill. 1983

13. Routh, Capt. R. L., "A Proposal for an Architectural Approach Which Apparently Solves All Known Software-Based Internal Computer Security Problems," ACM Operating Systems Review, (Sept 1984), 31-39.

14. Lorriore, L., "Capability Based Tagged Architectures," IEEE Transaction on Computer, vol C-33, no. 9. (Sept 1984), 786-803

15. Houdek, M. E., F,G, Soltis, and R. L. Hoffman, "IBM System/38 Support for Capability-Based Addressing," Proc. 8th Annual Symposium on Computer Architecture, Minneapolis, MN, 1981, 341-348

16. Levy, H.M., Capability-Based Computer Systems, Digital Press, 1984.

17. Saltzer, J.H., and M.D. Schroeder, "The Protection of Information in Computer Systems," Proceedings IEEE, Vol. 63(9) pp 1278-1308, September 1975.

18. Bell, D.E., and L.S. LaPadula, Secure Computer Systems: Unified Exposition and Multics Interpretation, MTR-2997 Rev. 1, Mitre Corporation, Bedford, Mass., March 1976.

19. Rauch-Hindon, W., "Distributed Databases," Systems and Software, September 1983.

What is the ElGamal Cryptosystem?

The ElGamal system is a public-key cryptosystem based on the discrete logarithm problem. It consists of both encryption and signature algorithms. The encryption algorithm is similar in nature to the Diffie-Hellman key agreement protocol (see Question 24).

The system parameters consist of a prime p and an integer g, whose powers modulo p generate a large number of elements, as in Diffie-Hellman. Alice has a private key a and a public key y, where $y = g^a \pmod p$. Suppose Bob wishes to send a message m to Alice. Bob first generates a random number k less than p. He then computes

$y_1 = g^k \pmod p$ and $y_2 = m \text{ xor } y^k$,

where xor denotes the bit-wise exclusive-or. Bob sends (y_1, y_2) to Alice. Upon receiving the ciphertext, Alice computes

$m = (y_1^a \bmod p) \text{ xor } y_2$.

The ElGamal signature algorithm is similar to the encryption algorithm in that the public key and private key have the same form; however, encryption is not the same as signature verification, nor is decryption the same as signature creation as in RSA (see Question 8). DSA (see Question 26) is based in part on the ElGamal signature algorithm.

Analysis based on the best available algorithms for both factoring and discrete logarithms shows that RSA and ElGamal have similar security for equivalent key lengths. The main disadvantage of ElGamal is the need for randomness, and its slower speed (especially for signing). Another potential disadvantage of the ElGamal system is that message expansion by a factor of two takes place during encryption. However, such message expansion is negligible if the cryptosystem is used only for exchange of secret keys.

| Question 30 |

015. Examples of known word processing methods and systems such as MS Word 2000.

User's Guide

Microsoft.

WORD

The World's Most Popular Word Processor

Document No. WB51157-1093
Printed in Ireland :09

For example, suppose you installed Word in the WINWORD directory of a file server—where X designates the file server—and distribute a silent script that uses the MYSCRIPT.STF table file to a user named Paul Tanner. The command line to run the script would be:

```
x:\winword\setup.exe /t myscript.stf /n "Paul Tanner" /q
```

Distributing a Script with Microsoft Mail

If you use Microsoft Mail to distribute a script, create a new message and then choose Insert Object from the Edit menu. In the Object Type box, select Package, and then choose the OK button. From the Edit menu in Object Packager, choose Command Line. Type the full path to SETUP.EXE in the WINWORD directory of the file server or the shared directory. (If your network supports UNC pathnames, use that syntax. If not, users will need to make the network connection themselves by using the same drive letter you specified before running Setup.) Type **setup** and the switches and arguments as needed, and then choose the OK button.

To attach the Word Setup icon to the command line, choose the Insert Icon button in Object Packager. Choose the Browse button to locate SETUP.EXE in the WINWORD directory of the network file server, and then choose the OK button. Choose Update from the File menu to add the icon to the Mail message, and then choose Exit from the File menu to close Object Packager. The icon is now ready to distribute. Anyone who receives the message can double-click the icon to run Setup from the network and install Word by using the script you specified with the /t switch.

Network Considerations for Workstation Users

There are two ways to run Word in a network environment:

- You can run Word entirely off the network, without installing it on your own computer.
- You can install Word on your own computer.

Installing Word on a Workstation

If your computer is connected to a network file server or a shared directory, your network administrator may have installed a copy of Word on the network that you can then install on your workstation. The administrator may also have created a process you can use to install Word automatically. Check with your administrator to determine the best way for you to install Word.

The procedure for installing Word on a workstation is discussed in Chapter 1, "Installing and Starting Word," in *Microsoft Word Quick Results*. Once you have installed Word, read the following section for important information about using Word in a network environment. You may also need special network software to manage and synchronize shared files on the file server. For more information, check with your network administrator.

Sharing and Protecting Documents on Networks

Using Word on a network is essentially the same as using Word on a stand-alone computer. On a network, however, you can use the network file server to store documents and exchange them with other users, so you may want to protect some documents from unauthorized access.

Things to Remember About Shared Documents

- If your workgroup uses a standard set of templates to ensure consistency, do not use other templates when you're working on a shared document.
- Be careful when you assign custom shortcut keys, especially when you redefine built-in shortcut keys.
- In order for everyone who works on a shared document to display and print it the same way, the fonts used in the document must be available on the other computers and printers in your workgroup.

  If you use TrueType fonts in shared documents, however, the fonts can be embedded in documents so that others who do not have those fonts installed can still see and print them. For more information, choose Options from the Tools menu, select the Save tab, and then choose the Help button.
- If you assign a file-protection password, you should write it down. Without the password, no one can open the document. Also bear in mind that some kinds of passwords—such as those that prevent any changes except annotations and form field edits—do not prevent other users from setting a file-protection password.

For more information about sharing and protecting documents, see Chapter 2, "Opening, Saving, and Protecting Documents."

Ensuring Compatibility

If you frequently use Word with documents created in other applications, you set options to compensate for limitations in the conversion process. These options don't change the documents, but they can make them easier to work with in Word. To view these options, choose Options from the Tools menu, and then select the Compatibility tab. For more information, see Chapter 26, "Converting File Formats."

Protecting a Form from Changes

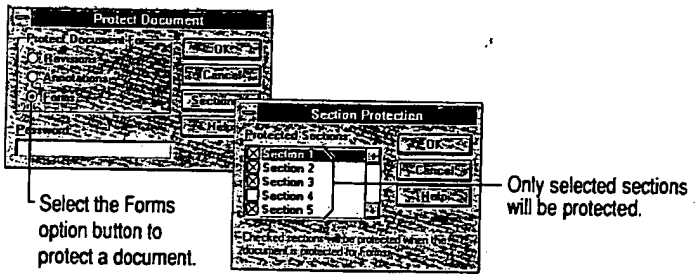
To activate form fields and to ensure that users don't accidentally change a form as they fill it in on line, you must lock the template for the form by choosing the Protect Document command from the Tools menu. When a document is protected, form fields are available for fill-in, and users can type only in form fields or unprotected sections.

Each time a user creates a form based on the protected template, a new, untitled document is created with the same protection as the template.

When you protect a document, Word changes it in the following ways:

- Form fields are activated.
- Field results are displayed instead of field codes.
- The insertion point can move only to form fields and unprotected sections.
- The entire document cannot be selected.
- Table column width is fixed.
- Some commands, such as Find, Replace, and Go To, are usable only in form fields and in sections from which protection has been removed.
- Some menu commands are unavailable.
- Entry macros, exit macros, and form field Help (described in the following sections) are activated.

You can specify a password when you protect a form. Only users who know the password can remove the protection and change the form.



▶ **To protect a form**

1. From the Tools menu, choose Protect Document.
2. Under Protect Document For, select the Forms option button.

   To assign a password to the form, type a password in the Password box. A password can contain up to 15 characters and can include letters, numbers, symbols, and spaces. As you type the password, Word displays an asterisk (*) for each character you type. If you assign a password, you must use the same password to remove protection from the document. Note that passwords are case sensitive. Each time you type the password you must use the same combination of uppercase and lowercase letters.
3. Choose the OK button.

   If you assigned a password, retype the password in the Confirm Password dialog box, and then choose the OK button.

When the active document is protected, the Protect Document command changes to Unprotect Document.


Protect Form button

Tip When you are designing a form, you can quickly turn protection on or off by clicking the Protect Form button on the Forms toolbar or by choosing Protect Document or Unprotect Document from the Tools menu.

▶ **To prevent a section from being protected**

1. From the Tools menu, choose Protect Document.
2. Select the Forms option button, and then choose the Sections button.

   If the Sections button is dimmed. there is only one section.
3. Under Protected Sections, select the check boxes to the left of the sections that you want to protect. Clear the check boxes next to the sections you want to leave unprotected, and then choose the OK button.
4. Choose the OK button to protect the document.

Note Some commands (such as Form Field Options) are unavailable in unprotected sections of a protected document. For full access to all word commands, remove protection from the document.

Protecting Documents from Changes

Word provides several ways to restrict changes to documents. You can assign a password to prevent other users from opening a document or to keep others from saving changes to the document. You can also request or require that other users on a network open a document as read-only.

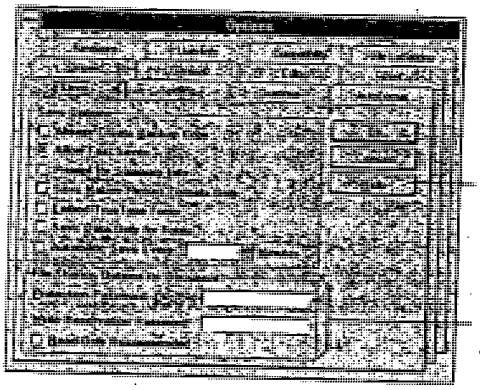
You can also assign a password so that other users can annotate a document and mark revisions. You or someone else who knows the password must open the document normally and review the changes before they become permanent. For more information, see Chapter 25, "Annotating, Revising, and Routing Documents."

If you use form fields to create a form, you can assign a password so that other users can fill in those parts of the form but cannot change anything else in the document. For more information, see Chapter 14, "Forms."

Warning If you assign a password for any of these types of protection, it's a good idea to write it down. Without the password, you cannot open the document.

Setting Passwords and Selecting Save Options

To assign a password to a document and set options that control whether changes can be saved, choose the Options button in the Save As dialog box or choose Options from the Tools menu, and then select the Save tab.



Protection Password To prevent other users from opening a document, type a password in the Protection Password box. Only users who know the password can open the document. Passwords are case-sensitive.

Write Reservation Password To prevent other users from saving changes to a document, type a password in the Write Reservation Password box, and then choose the OK button. Word will prompt you to type the password again to confirm it. Word then requires you to type the password to open the document normally. If you do not know the password, you can still open the document as read-only by choosing the Read Only button in the Password dialog box that appears when you open the document.

Read-Only Recommended To recommend, but not require, that other users open a document as read-only, select the Read-Only Recommended check box. When another user opens a document that's protected by this option, Word indicates that the document should be opened as read-only unless changes need to be saved. The user can then open the document normally or as a read-only document.

▶ **To protect a document with a password**

1. Open the document you want to protect with a password.
2. From the File menu, choose Save As.

   If you have not yet named the document, type a name in the File Name box.
3. Choose the Options button.
4. In the Protection Password box or the Write Reservation Password box, type a password, and then choose the OK button.

   A password can contain up to 15 characters and can include letters, numbers, symbols, and spaces. As you type the password, Word displays an asterisk (*) for each character you type. Note that passwords are case-sensitive.
5. When Word prompts you to confirm the password, retype it and then choose the OK button.
6. To save the document, choose the OK button.

   Make sure that you write down the document password, exactly as you typed it. You will need to type it the next time you open the document.

Tip If you want to allow other users to add only comments to a document, you can protect it by using the Protect Document command on the Tools menu. Other users can then open the document, but they can only make comments by using annotations.

▶ **To change or delete a password**

1. Open the document whose password you want to change or delete.
2. From the File menu, choose Save As.
3. Choose the Options button.
4. In the Protection Password box or the Write Reservation Password box, select the row of asterisks that represents the existing password, and then do one of the following:
   - To change the password, type the new password.
   - To delete the password, press DELETE.
5. Choose the OK button.

   If you changed the password, Word asks you to retype the new password.
6. To save the document with the new password, choose the OK button.

Other Ways of Protecting Documents

Word offers other methods of protecting your documents.

For information about	See
Opening documents as read-only	"To open an existing document," earlier in this chapter
Preventing any changes to documents except for filling in form fields	Chapter 14, "Forms"
Preventing any changes to documents except for annotations and marked revisions	Chapter 25, "Annotating, Revising, and Routing Documents"

Note Protecting a form or locking a document for annotations or revisions does not keep another user from saving that document with a password or from setting other save options. If you want to protect a document from all types of changes, save it with a password by using one of the methods described in this chapter.

Some operating systems and networks also provide ways to protect documents. To find out if your system has these features, check with your network administrator or see the documentation for your operating system or network.

Note You can customize the marks Word uses to show document differences. For more information, see "Customizing Revision Marks," earlier in this chapter.

▶ **To compare two versions of a document**

1. Open the edited version of the document.
2. From the Tools menu, choose Revisions.
3. Choose the Compare Versions button.
4. In the Original File Name box, type or select the name of the original document, and then choose the OK button.

   Word displays the edited document marking inserted, deleted, and revised text with revision marks. The options for displaying revision marks are set on the Revisions tab in the Options dialog box (Tools menu).
5. To accept or reject the revisions, choose Revisions from the Tools menu. For more information, see "Incorporating Revisions," earlier in this chapter.

Protecting a Document for Annotations and Revisions

For more information on ways to protect a document, see Chapter 21, "Opening, Saving, and Protecting Documents."

To allow reviewers to comment on but not make changes to a document, you can protect it for annotations. To allow reviewers to change a document and keep a record of all changes, you can protect it for revisions.

For maximum protection, you should also use a password when you protect a document for annotations or revisions. Otherwise, anyone can remove protection from the document by choosing Unprotect Document from the Tools menu.

▶ **To protect a document for annotations or revision marks**

1. Open the document you want to protect.
2. From the Tools menu, choose Protect Document.
3. Do one of the following:
   - To allow reviewers to insert annotations but not change the contents of the document, select the Annotations option button.
   - To track revisions, select the Revisions option button. The reviewers cannot turn off revision marking, and revisions cannot be accepted or rejected.
4. To ensure that a document is protected against untracked changes, type a password. This prohibits anyone who does not know the password from unprotecting the document.
5. Choose the OK button.

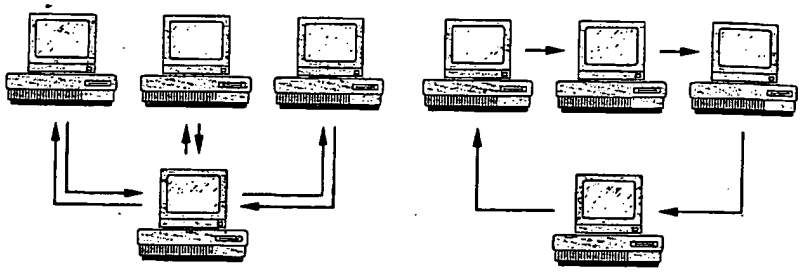
▶ **To unprotect a document for annotations or revision marks**

- From the Tools menu, choose Unprotect Document.

  If the author has protected the document with a password, you must know the password to unprotect the document.

Routing a Document Online

You can use Word and Microsoft Mail or a compatible mail program to route documents online. For example, you might want others to review an important memo before sending it out, or you might want several people to complete an online questionnaire or form.

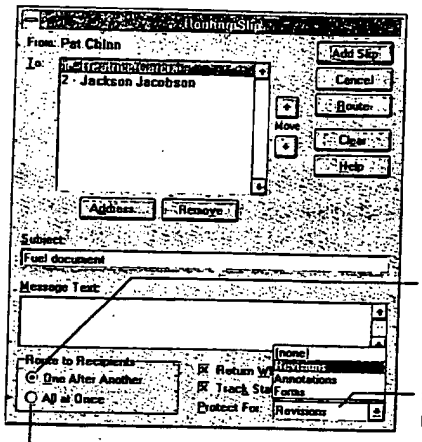


You can route a document to all reviewers at once ...

or you can route it to one reviewer after another.

You can route online copies in two ways. You can send a separate copy to all reviewers at the same time, or you can send a single copy that goes to each person on the list in turn, allowing each reviewer to see the comments of all previous reviewers.

Reviewers return their annotated or revised copies to you or the next person on the distribution list by choosing the Send command from the File menu. When all the copies have been returned, you can merge the annotations and revisions into the original document to simplify review of the comments. For more information on merging comments, see "Merging Annotations and Revisions," later in this chapter.



To route a document to others

1. Open the document you want to route.
2. From the File menu, choose Add Routing Slip.
3. Choose the Address button. Select the names of the people to whom you want to route the document, choose the Add button, and then choose the OK button.

   If you want to route the document to one recipient after another, use the Move up and down arrows to put the names in the correct routing order.
4. In the Subject and Message Text boxes, type the subject and any message or instructions you want to send with the document. Each recipient will receive the same subject and message.

   Word automatically appends instructions to your message telling recipients to choose the Send command when they are finished.
5. Under Route To Recipients, do one of the following:
   - To route one copy of a document to one recipient after another, select the One After Another option button.
   - To route multiple copies of a document to all recipients at the same time, select the All At Once option button.

6. Select any other options you want, and then choose the Route button.

   If you want to continue to edit the document before you route it, choose the Add Slip button, and continue to edit the document. When you are ready to send the document, choose Send from the File menu. Word displays a message asking you to confirm that you want to route the document.

   The document is sent to the distribution list as an attached Word file. The recipients can add annotations or revisions to the document and then return the copy to you by choosing the Send command on the File menu.

   If the document is being routed to one recipient after another, the Send command automatically routes it to the next person on the list before it returns to you. You will receive all the recipients' comments in one document after it has been routed to the last person on the list.

   If you send the document to all recipients at the same time, you will receive multiple copies of the document. You can then merge all changes into one document. For more information, see the following section.

Merging Annotations and Revisions

If you have given individual copies of a document to multiple reviewers, you can combine their annotations and revisions into the original document. When you merge annotations and revisions, any annotations and revisions already in the original document are preserved as additional comments are merged. Word assigns a different color to each reviewer. If there are more than eight reviewers, Word uses the colors again, so some reviewers may share the same color.

Note Annotations and revisions cannot be merged back to the original document unless they are marked. To ensure that revisions to a document are marked, you should protect the document for revisions or annotations before making the revisions. For information on protecting documents, see "Protecting Documents for Annotations and Revisions," earlier in this chapter.

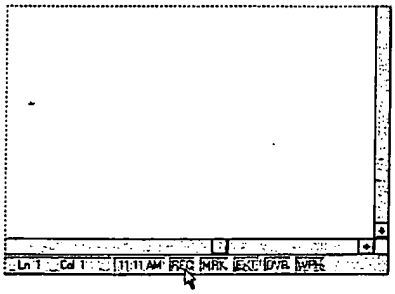
▶ **To merge revision marks and annotations**

1. Open the document that has revisions you want to merge into the original document.
2. From the Tools menu, choose Revisions.
3. Choose the Merge Revisions button.

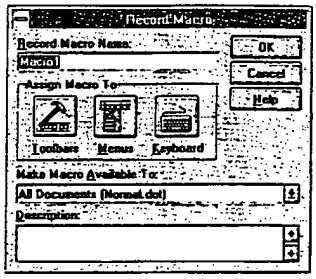
QUICK START

Recording a Macro

You double-click "REC" on the status bar to display the Record Macro dialog box, where you can type a name for your macro or accept the name Word proposes. When you choose the OK button to close the dialog box and begin recording your macro, Word displays the Macro Record toolbar. Word records each command you choose and action you take until you click either the Pause button to temporarily suspend recording or the Stop button to finish your macro. You can also double-click "REC" on the status bar to stop recording a macro.



Double-click "REC" on the status bar to display the Record Macro dialog box.



Accept the name Word proposes or type another name.

To indicate that the recorder is on, Word attaches a recorder graphic to the mouse pointer.

Use the Stop and Pause buttons on the Macro Record toolbar to stop or pause recording.

Assigning a Macro to a Toolbar, a Menu, or Shortcut Keys

Assigning a macro to a toolbar, a menu, or shortcut keys is a good way to make the macro more accessible. You can do this by choosing the appropriate button in the Record Macro dialog box.

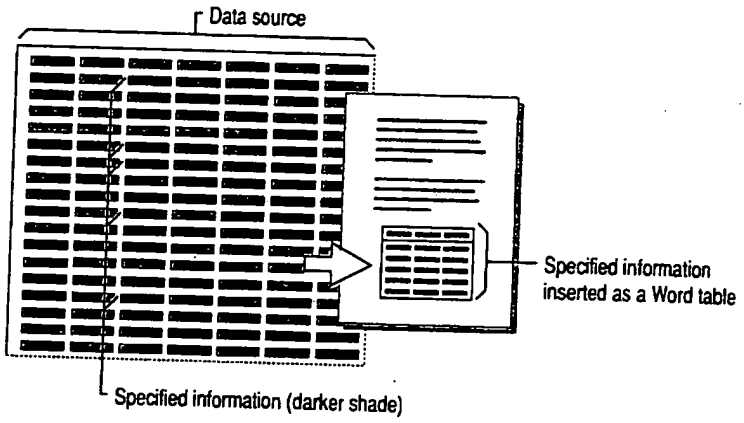
Choose one of these buttons to assign the macro you are recording to a toolbar, a menu, or shortcut keys.

Running a Macro

Once you've assigned a macro to a toolbar, a menu, or shortcut keys, running the macro is as simple as clicking the appropriate toolbar button, choosing the appropriate menu command, or pressing the corresponding key combination. You

Inserting Tables of Information from a Database

Sometimes you may want to include in a Word document information from an existing database, a Microsoft Excel worksheet, or another source of data. By using the Database command on the Insert menu, you can specify the information you want and automatically insert it as a table in a Word document. You can screen, or "filter," the information according to criteria you select. You can also instruct Word to update the information in the Word document if the source file has changed.



Word can retrieve information from the following types of files:

- Files from the following applications that are installed on your system:

  Microsoft Access®
  Microsoft Excel

- Files from single-tier, file-based database applications for which you have an open database connectivity (ODBC) driver installed in the System subdirectory of your Windows directory. ODBC drivers for the following applications are supplied with Word:

  Microsoft Access
  Paradox®
  Microsoft FoxPro® (or other Xbase database application such as dBASE®)

For a list of file converters provided with Word, see Chapter 26, "Converting File Formats."

- Files for which you have a file converter installed. In addition to converters for ASCII text files, Word provides file converters for many applications, including:

Microsoft Word for Windows	WordPerfect 5.*x* for MS-DOS and Windows
Microsoft Word for the Macintosh versions 3.*x*,[1] 4.*x*, and 5.*x*	Microsoft Excel 2.x,[2] 3.0, 4.0,[1] and 5.0[3]
Microsoft Word for MS-DOS 3.0–6.0	Lotus 1-2-3 2.*x*[2] and 3.*x*[1]

[1] Converts only from this format.

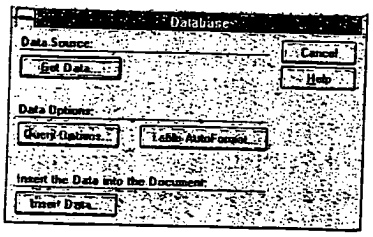
[2] Converter works only with Windows version.

[3] Converter works only with Macintosh version.

You can also insert information from another Word document. For example, you might have set up a membership directory for use as a mail merge data source. Instead of copying and pasting information from various data records, use the Database command to insert just the information you request.

Inserting the Data

When you choose the Database command from the Insert menu, Word displays the Database dialog box. Now you can locate the data source, select the information you want, and format the table in which the information is displayed.



Once you select the data source, the other buttons in the dialog box become available.

By default, Word inserts all of the information from the selected data source. In most cases, however, you'll want to use only some of the available information. For example, from a large personnel file, you might want to list only the names, departments, and hiring dates of all employees who have worked for your company 10 years or longer.

If information in the data source changes frequently and you want to keep your document up to date, you can insert the information as a Word *field*. The field is simply a "placeholder" that represents the table in your document. For more information, see "Keeping the Table Information Up to Date," later in this chapter.

▶ **To insert information from a data source as a table**

1. Position the insertion point where you want the new table of information to be included.
2. From the Insert menu, choose Database.
3. Choose the Get Data button.
4. In the Open Data Source dialog box, type or select the filename of the data source you want to open, and then choose the OK button.

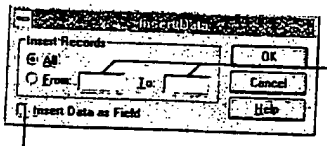
   If the data source is not listed, select the appropriate drive and directory or folder. Then select the appropriate option in the List Files Of Type box.

   If you open a Microsoft Excel worksheet, you can insert the entire worksheet or a range of cells. If you open a Microsoft Access database, you can insert records from a table or a selection of records defined by a query. For more information, see the documentation for the application you are using.
5. To insert specific information from the data source or list the information in a particular order, choose the Query Options button. Do one or more of the following, and then choose the OK button.
   - On the Filter Records tab, specify criteria to select the data records to insert.
   - On the Sort Records tab, select the data fields by which you want to sort the information.
   - On the Select Fields tab, remove any fields you don't want from the Selected Fields list. The order of the fields in the list determines the order in which the fields are inserted left to right.

     If you don't want to insert the field names from the header row with the data records, clear the Include Field Names check box.
6. To format the table, choose the Table AutoFormat button.
7. Choose the Insert Data button.

   In the Insert Data dialog box, you can specify the range of records you want to insert. The range refers only to the records that were selected by the query. If you want to be able to update the information in the table automatically, select the Insert Data As Field check box. Then choose the OK button.

Note If you insert more than 31 data fields, Word inserts tab characters to separate the columns of information.



To specify which of the selected records are inserted, type starting and ending record numbers in the From and To boxes.

Select this check box if you want to keep the table information up to date.

Modifying the table format If you don't select the Insert Data As Field check box, Word inserts the information as an ordinary table. You can resize the table columns and otherwise modify the table by using the commands on the Table menu. If you insert the information as a field, however, you must choose the Database command again to reinsert the table and update the table format by choosing the Table AutoFormat button. Otherwise, the table formatting you've applied is removed the next time you update the DATABASE field. Formatting you've applied to text in the table is also removed. For more information, see "Keeping the Table Information Up to Date," later in this chapter.

Modifying the information in the table You may want to modify the information in the table later. For example, you might want to include another column of information or select a different set of records from the data source. To do this, click in the table and then choose the Database command again to select the information you want in the table. If you insert the information as a field and then edit or format the text in the displayed table, your changes will be deleted the next time you update the DATABASE field. For more information, see "Keeping the Table Information Up to Date," later in this chapter.

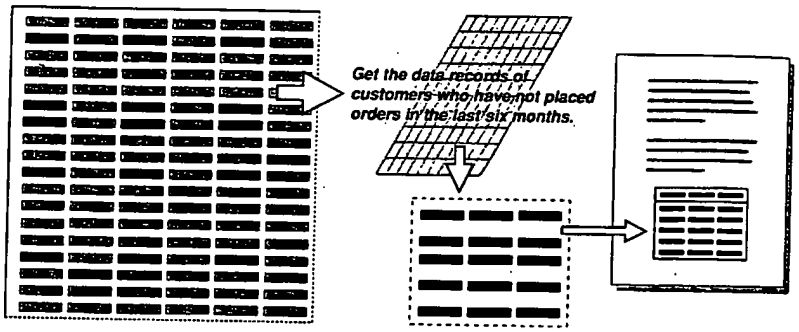
If Word can't recognize the field and record delimiters If Word can't recognize the characters used to separate data fields and data records in text-delimited files (files in which data is separated by commas, tab characters, or other characters), Word asks you to select the separating characters (delimiters). Word recognizes one data field delimiter and one data record delimiter. If a combination of two or more characters is used as a delimiter, then the remaining characters are treated as text in the data fields.

Selecting the Data

To get only the information you want from a data source, you create a *query*. A query is simply a set of instructions, or rules, that describes the information you want from the data source. You can think of the following statement as a query:

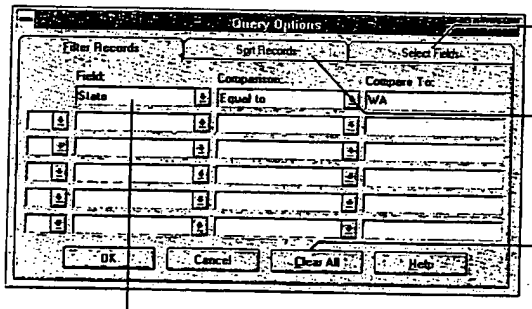
"Give me the names, addresses, and account numbers of all customers who have not placed orders in the last six months."

The first part of the statement identifies the categories of information you want—names, addresses, and account numbers. The second part of the statement indicates that you want information only for certain customers—those who have not ordered anything in the last six months.



A query tells Word which information to select from a data source.

You create queries by selecting options in the Query Options dialog box. You select data fields to specify the categories of information you want. The order in which you select the data fields determines the order of the columns of information in the table, from left to right. To get the information only from certain data records, you specify one or more rules for selecting the records. To list the rows of information in a particular order, you can sort the data records.



Select this tab to specify the categories of information in the table.

Select this tab to specify the order information is listed in the table.

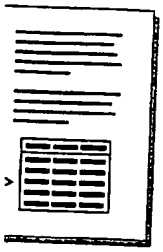
Choose this button to delete the current rules.

Word selects all data records with "WA" in the data field "State."

eate a *query*. A
nformation you
nent as a query:

omers who have

ition you want—
tatement
those who have



g box. You
. The order in
is of
only from
records. To
ata records.

to specify the
nformation in

to specify the
on is listed in

tton to delete
s.

Specifying the Record-Selection Rules

On the Filter Records tab, you specify the rules that Word uses to retrieve the information you want, based on the contents of selected data fields. When specifying a rule, you can select any data field in the data source—even a data field you don't want to include in the table.

A record-selection rule is made up of three parts:

- A field name corresponding to a data field in the selected data source
- A comparison phrase, such as "Equal To" or "Is Not Blank"
- Text or numbers you want the contents of the data field to be compared with

If you compare text When comparing a data field that contains text, Word compares the sequence of characters based on the ANSI sorting order. The text "apple" is considered "less than" the word "berry" because, alphabetically, "apple" precedes "berry." Whether the text is uppercase or lowercase isn't significant.

For example, to retrieve data records for members of your organization whose last names begin with "A" through "L," you specify the following rule:

LastName Is Less Than M

Any name beginning with "A" through "L" is considered less than "M," so only the data records that contain those names are selected. (The last name must be contained in a separate data field, or else it must precede the first name in the field—for example, "Bendal, Maria".)

If you compare numbers mixed with text If numbers are mixed with letters, hyphens, plus or minus signs, or other nonnumeric characters, Word compares the numbers as though they were a sequence of text characters. For example, a five-digit U.S. ZIP Code is compared as a number, whereas a nine-digit "ZIP+4" code such as 99999-9099 is compared as text, as are non-U.S. postal codes that contain letters.

Comparing sequences of mixed numbers and nonnumeric characters—code numbers, for example—can have different results if some items contain more sequential numerals than others. For example, the following items are sorted in this order:

0002xy, 002, 011y, 1, 1x, 1yz, 22x, 2x

The following items, however, are sorted in this order:

0001, 0001x, 0001yz, 0002, 0002x, 0002xy, 0011y, 0022x

Specifying Multiple Rules

You can specify as many as six selection rules. Using multiple rules allows you to narrow the range of data records that are selected. When you select multiple rules, you must specify AND or OR to connect each additional rule to the preceding rule, as in the following examples.

Example 1	Example 2
State (Is)-Equal To Oregon	State (Is) Equal To Oregon
AND City (Is) Equal To Portland	OR State (Is) Equal To California

The rules connected by AND select only data records that contain both "Oregon" in the State field and "Portland" in the City field. The rules connected by OR select all data records that contain either "Oregon" or "California" in the State field—a potentially larger number of records. The key difference between AND and OR is as follows:

- When you use AND to connect rules, Word selects only those records that satisfy both (or all) rules. Each rule connected by AND *eliminates* more of the records in the data source.
- When you use OR to connect rules, Word selects any record that satisfies at least one of the connected rules. Each rule connected by OR *selects more* of the records in the data source.

AND has precedence You can use AND and OR separately or in combination. In sets of rules that contain both AND and OR, rules connected by AND have precedence over rules connected by OR. This means that the set of rules connected by AND is used to select records before the set of rules connected by OR. How you connect the rules—by using AND or by using OR—affects which data records are selected.

Suppose you want to select data records of all clients who live in either Portland or Salem, Oregon. In the Query Options dialog box, you would specify the following rules to determine the contents of the data fields "City" and "State":

State (Is) Equal To Oregon
AND City (Is) Equal To Portland
OR State (Is) Equal To Oregon
AND City (Is) Equal To Salem

Using the first set of rules connected by AND, Word compares the data records to identify the clients who live in Portland, Oregon. Next, Word compares the data records with the next set of rules connected by AND. Word then selects only data records of clients in Oregon who live in either Portland or Salem.

Notice that the following set of rules does *not* produce the same result:

State (Is) Equal To Oregon
AND City (Is) Equal To Portland
OR City (Is) Equal To Salem

Because AND takes precedence, the first set of rules connected by AND selects records of clients who live in Portland, Oregon. However, the rule connected by OR also selects records for clients in any city named Salem—including Salem, Massachusetts, for instance

Comparing a range of values You can also use AND to compare a selected field with a range of values rather than a single value. For example, given the following rules, Word selects all data records that have a value of 98001 through 98500 in the PostalCode field.

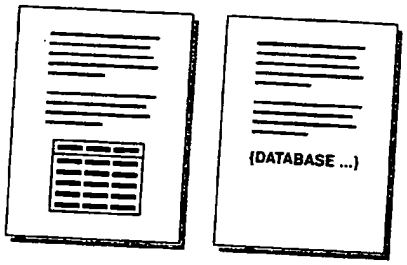
PostalCode (Is) Greater Than Or Equal (To) 98001
AND PostalCode (Is) Less Than Or Equal (To) 98500

Keeping the Table Information Up to Date

If you select the Insert Data As Field check box when you insert the information, Word does not insert an actual table; instead, it inserts a DATABASE field to represent the table.

With field codes hidden, the information is displayed as a table. With field codes displayed, the information is displayed as a DATABASE field. The field contains all information needed to locate and open the selected data source, carry out the query, and insert the information in your document.

To display or hide the field codes, press ALT+F9 (Windows) or OPTION+F9 (Macintosh).



Document with field codes hidden ...

... and with field codes displayed.

Updating information in the table To bring in the latest information from the data source, you update the DATABASE field. To update the field, click anywhere in the table, and then press F9. Word then repeats the query instructions you specified in the Query Options dialog box and inserts the latest information from the data source.

For more information about Word fields, see Chapter 32, "Inserting Information with Fields." For specific information about the DATABASE field, click in the field in your document and then press F1 (Windows) or the HELP key (Macintosh).

Preserving the table format The DATABASE field also describes the table format you selected in the Table AutoFormat dialog box. If you manually resize the table columns, change the text formatting, apply borders, or change other aspects of the table format, the added formatting is lost the next time you update the DATABASE field.

To retain changes to the table format when you update the information, first click anywhere in the table. Then choose Database from the Insert menu, open the data source, and select the data. Choose the Table AutoFormat button to select a new table format, and then reinsert the table.

Inserting data from a Microsoft Access database You can insert information from any table or query you saved in a Microsoft Access database. If you select a saved query and are using Microsoft Access version 1.1 or later, you can specify how Word uses the query instructions. If you clear the Link To Query check box in the Microsoft Access dialog box (Queries tab), Word repeats the original query instructions each time the DATABASE field is updated. If you select the Link To Query check box, Word carries out the latest query instructions saved in the selected query each time the DATABASE field is updated. For more information, see "Linking to a Microsoft Access Query" in Chapter 30, "Mail Merge: Advanced Techniques."

Troubleshooting

The type of object I want to embed does not appear in the Object dialog box.
Object types are listed in the Object dialog box only if the original application was properly installed by using its setup (installation) program. Try reinstalling the source application, making sure that you use its setup program. For example, if you want to embed a Microsoft Excel worksheet and Microsoft Excel is not listed, reinstall Microsoft Excel by using the installation program in the original disk set. You should run the newly installed source application independently at least once before you try to embed an object from that application. On the Macintosh, rebuild the desktop.

Row and column numbers appear in links from Microsoft Excel.

If you copy cells from a Microsoft Excel worksheet and link them to a Word document as a picture or bitmap, the row and column numbers are included in the Word document even if you have not selected them in Excel. If you don't want these numbers in your Word document, do the following before you copy the object in Microsoft Excel:

1. From the Options menu in Microsoft Excel, choose Display.
2. Under Cells, clear the Row & Column Headings check box.
3. Choose the OK button.

I embedded an object and then copied and pasted it, but the copies are not being updated.

Embedding creates a single, independent object. When you copy and paste that object, you create a new, independent object. The copies are not connected, so changes in one are not reflected in the other. If you want multiple copies of an object that reflect changes made to the original object, you should create a link to the source file. First, create a separate source file. Then, in the Word document, create as many links to the source file as you want. Any changes you make to the source file will be reflected in each linked copy of the object.

Word displays a message that it cannot find the application needed to edit an embedded object.

The application may have been either deleted or moved to a different location. If the application is located on a network drive, the drive may no longer be connected. Reinstall the application, or reconnect the network drive.

Word cannot update a particular link.

There are several possible reasons why Word might not be able to update a link. Most commonly, it is because the source document has been moved, renamed, or deleted. If the source document has been deleted, you won't be able to update the link. If the source document has been moved or renamed, follow this procedure:

1. In the Word document, choose Links from the Edit menu.
2. Select the link you want to update, and then choose the Change Source button.
3. Locate the source file, and then choose the OK button.
4. In the Links dialog box, choose the Update Now button to update the link.

I resized an embedded object, but after I edited and updated the object, it resumed its original size.

Display the field codes by choosing Options from the Tools menu, selecting the View tab, and then selecting the Field Codes check box. The \s switch in the field code retains the original scaling and cropping information. Delete the \s from the field code.

For information about editing fields, see Chapter 32, "Inserting Information with Fields."

010. Examples of evidence with regard to network monitoring and control systems

provided using a single 64
rvices can be provided using
low bit rate speech vocoder

sult of video coding, can be
rks. Predicting the overall
h is discussed in Chapter 17.
are discussed in Chapters 18

sidered to be composed of 550
l is equiprobable among 64
aster scanning at a 25 Hz frame
ne minimum bandwidth required
on reception. [4.44 MHz]

he picture frames described in
l, as defined in Figure 5.12, with

y be considered to be composed
ith straightforward 24-bit colour
e. Calculate the time required in

te for a television signal for
ines per frame = M, horizont
each pixel comprises one
tem which employs 819 lines
ne period and 100 levels.

Part Four

Networks

Part Four is devoted to communication networks which now exist on all scales from geographically small LANs to the global ISDN.

It starts with a discussion, in Chapter 17, of queuing theory which may be used to predict the delay suffered by digital information packets as they propagate through a data network.

Chapter 18 describes the topologies and protocols employed by networks to ensure the reliable, accurate and timely, delivery of information packets between network terminals. Rings, buses and their associated medium access protocols are discussed, and international standards such as ISO OSI, X.25 and FDDI are described. The optical transmission medium, which now forms an integral part of many communications networks, is also examined.

Part Four ends, in Chapter 19, by examining public networks. The current plesiochronous digital hierarchy (PDH) is reviewed before introducing a more detailed discussion of the new synchronous digital heirarchy (SDH) which will gradually come to replace it. The Chapter concludes with a brief discussion of PSTN/PDN data access techniques including the ISDN standard, ATM, and the probable future development of the local loop.

CHAPTER 17

Queuing theory for packet networks

17.1 Introduction

In packet switching, secure bundles of information are assembled, addressed and transmitted through a network without the need for dedicated end-to-end connection paths to be established. The packets are individually transported and delivered by the network to the required destination. The network additionally ensures that packets are output in the correct order at the receiver.

Packet switched networks have existed for many years. Early examples were Euronet which links nine EC countries and Switzerland, running the Direct Information Access Network-Europe (DIANE) service. In 1981, British Telecom opened its first national public network, known as the Packet Switched Service (PSS). This system is controlled by a network management centre, based on duplicated minicomputers. PSS uses the X.25 protocol (see Section 18.6.1). The most well known network today is the Internet which supports the information retrieval service known as the World Wide Web (WWW).

The Joint Academic Network (JANET), Figure 18.2, and its successor SuperJANET are funded by the UK research councils. JANET has a node at every university and runs on leased lines. It provides a service for the communication of data, such as computer files and electronic mail, between sites and onward via the Internet (see section 18.7.6). All these networks introduce queuing, with consequent delays and possible loss of traffic data.

Queuing theory [Nussbaumer] can be used to model these or other networks where customers or data packets arrive, wait their turn for handling or service, are subsequently serviced, and are then transmitted through the network. (Supermarket checkouts, ticket booths, and doctors' waiting rooms are all commonly encountered examples of queuing systems.) Queuing theory was developed originally to model analogue teletraffic but is now widely applied to digital packet traffic [Tanenbaum]. A queuing system can be characterised by the following five attributes:

- The interarrival-time probability density function.
- The service-time probability density function.
- The number of servers, or server processes.
- The queuing discipline.
- The amount of buffer, or waiting, space in the queues.

The interarrival-time probability density function (pdf) describes the interval between consecutive arrivals. After a sufficiently long sampling time, the arrival times can be grouped to obtain the pdf which characterises the arrival process.

Each customer requires a certain amount of the server's time which varies from customer to customer. To analyse a queuing system, the service-time pdf, like the interarrival-time pdf, must be known.

The number of servers speaks for itself. Many banks, for example, have one queue for all customers. Whenever a teller is free, the customer at the front of the queue goes directly to that teller. Such a system is a multiserver queuing system. In other banks, each teller has his, or her, own private queue. This corresponds to a collection of independent single-server queues.

The queuing discipline describes the order in which customers are taken from the queue. Supermarkets use first come, first served. Hospital emergency rooms often use sickest attended to first. In friendly office environments, shortest job first often prevails at the photocopy machine. Not all queuing systems have an infinite amount of buffer space. When too many customers are queued up in a finite number of available slots, some customers can get lost or rejected.

This chapter concentrates predominantly on infinite-buffer, single-server systems using a first come first served queuing discipline. The Kendall notation A/B/m [Kleinrock] is widely used in the queuing literature for these systems. A represents the interarrival-time pdf, B the service-time pdf, and m the number of servers employed. The probability densities A and B are usually chosen from the following set:

M – Markov (implying an exponential pdf);
D – deterministic (all customers have the same constant value implying an impulsive pdf);
G – general (i.e. some arbitrary pdf);
E_k – Erlang distributed.

The state of the art ranges from the M/M/1 system, about which everything is known, to the G/G/m system, for which no exact analytical solution is yet available. We will concentrate on the M/M/1 model. Figure 17.1 shows the queue for such a single server process. We now need to develop a mathematical analysis for queuing systems which can be used to show what limits or restricts the practical performance of these systems. This is achieved by first examining arrivals only, before progressing to model the combined arrival and service processes within the queuing buffer memory.

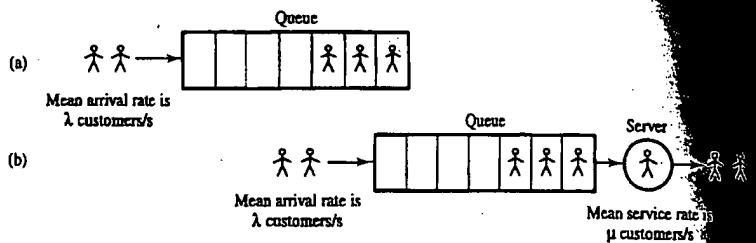


Figure 17.1 *Single server queue model and outcomes for a counting process: (a) arrivals only; (b) arrivals and departures.*

17.2 The arrival process

We make the following assumptions:

- The arrival process is memoryless in the sense that any arrival is statistically independent of all other arrivals;
- The arrival process is statistically stationary. (This implies that the probability of an arrival occuring in any small time interval depends only on the interval's width and not its location in time.)

Queuing systems in which the only transitions are to adjacent states are known as birth–death systems, which is a mathematician's terminology for a counting process with both arrivals and departures. We are considering, at present, *only* arrivals.

We have to model the evolution of the system from one state to another [Gelenbe and Pujolle] where the *state* is synonymous with the number of customers waiting for service. The approach via the Markov probability chain [Chung] is inappropriate, since the probability of transition between any two states at any given point in time, t, is zero. While we cannot characterise the *probability* of transition, we *can* characterise the *rate* of transitions between two states. Suppose that for two particular states the rate of transitions between them is a constant λ. What we mean by this is that in a time δt we can expect an average of $\lambda\ \delta t$ transitions. If δt is very small then $\lambda\ \delta t$ is a number much smaller than unity, and the probability of more than one transition in time δt is vanishingly small. Under these conditions, we can think of $\lambda\ \delta t$ as the probability of one transition (P_1) in time δt, and $(1 - \lambda\ \delta t)$ as the probability of no transition (P_0) in this time.

This leads us to a transition diagram and associated set of differential equations. The transition diagram, Figure 17.2, associates a node with each state. Within node j we denote the probability of being in that state at time t as $P_j(t)$.

17.2.1 pdf for j arrivals in t seconds

Assume arrivals (A) are governed by a randomly distributed pure birth process as shown in Figure 17.3 where the arrival rate does not depend on the state of the system. The

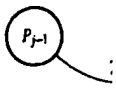


Figure 17.2 *Markov model*

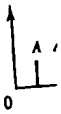


Figure 17.3 *Example of r*

only arrivals then the pr
and P_{j-1}:

$$P_j(t + \delta t)$$

hence:

$$\frac{P_j(t + \delta t)}{\delta t}$$

Now let $\delta t \to 0$:

$$\frac{dP_j(t)}{dt} =$$

For $j = 0$, $P_{j-1} = 0$, a

$$\frac{dP_0(t)}{dt} =$$

The solution to equat

$$P_0(t) =$$

which represents the
starting with a pro
exponential time co
derivation of the P
exponential interarr
shown that:

$$P_j(t)$$

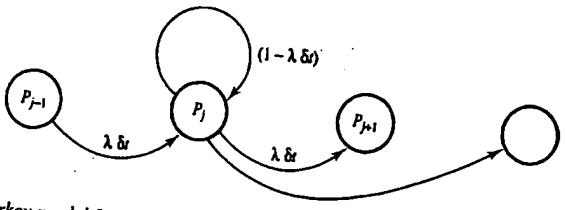


Figure 17.2 *Markov model for queue states corresponding to $j-1$, j, $j+1$ packet arrivals.*

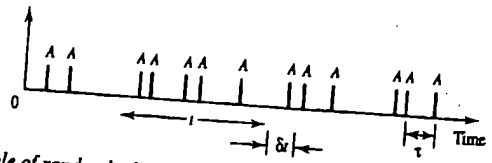


Figure 17.3 *Example of randomly distributed arrivals (A) with interarrival time τ.*

only arrivals then the probability of being in state j after δt seconds is dependent on P_j and P_{j-1}:

$$P_j(t+\delta t) = P_j(t)\,(1-\lambda\,\delta t) + P_{j-1}(t)\,\lambda\,\delta t \tag{17.1}$$

hence:

$$\frac{P_j(t+\delta t) - P_j(t)}{\delta t} = \lambda(P_{j-1}(t) - P_j(t))$$

Now let $\delta t \to 0$:

$$\frac{dP_j(t)}{dt} = \lambda(P_{j-1}(t) - P_j(t)) \tag{17.2}$$

For $j = 0$, $P_{j-1} = 0$, as we cannot have less than zero arrivals. Therefore:

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) \tag{17.3}$$

The solution to equation (17.3) is:

$$P_0(t) = e^{-\lambda t} \tag{17.4}$$

which represents the probability of no arrivals in time t. This is plotted in Figure 17.4, starting with a probability of 1 at $t = 0$, decaying asymptotically to zero with an exponential time constant of λ. Thus at $t = 1/\lambda$, $P_0(1/\lambda) = 0.37$. This is the start of the derivation of the Poisson distribution for the number of arrivals j in t seconds for an exponential interarrival time distribution, with a mean arrival rate of λ. It can further be shown that:

$$P_j(t) = \frac{(\lambda t)^j}{j!}\,e^{-\lambda t} \tag{17.5(a)}$$

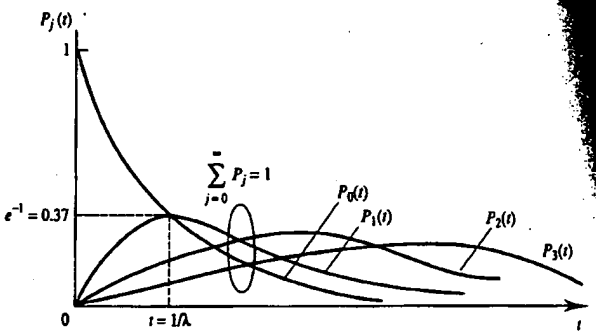


Figure 17.4 *Probability of j arrivals plotted against time for j = 0, 1, 2, 3.*

i.e.:

$$P_1(t) = \lambda t\, e^{-\lambda t} \qquad (17.5(b))$$

$$P_2(t) = \tfrac{1}{2}(\lambda t)^2\, e^{-\lambda t} \quad etc \qquad (17.5(c))$$

Figure 17.4 shows that the probability of only one arrival, $P_1(t)$, peaks at the time interval $t = 1/\lambda$. If Figure 17.4 is plotted with the horizontal axis as offered traffic, λt, then the peak value of $P_1(t)$ occurs at an offered traffic value of unity. For longer time intervals, it becomes increasingly likely that there will be more than one arrival. The probabilities of there being two or three arrival events, $P_2(t)$ and $P_3(t)$, peak at later time intervals and also have progressively smaller probability peaks, so that, for any specified time interval, all the probabilities sum to unity as required, i.e.:

$$\sum_{j=0}^{\infty} P_j(t) = 1 \qquad (17.6)$$

17.2.2 CD and pdf for the time between arrivals

If the time between successive arrivals is τ, as in Figure 17.3, the probability that τ is less than, or equal to, some value of time t, $P(\tau \le t)$, is given by:

$$P(\tau \le t) = 1 - P(\tau > t) \qquad (17.7(a))$$

But $P(\tau > t)$ is the probability of no arrivals in time t. Thus, for the Poisson process:

$$P(\tau > t) = P_0(t) = e^{-\lambda t} \qquad (17.7(b))$$

hence:

$$P(\tau \le t) = 1 - e^{-\lambda t} \qquad (17.7(c))$$

Equation (17.7(c)) is the cumulative distribution (CD) function, equations (3.10) and

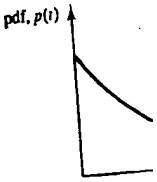


Figure 17.5 *(a) pdf; and (b*

(3.13(b)), for the time
interarrival probability d
obtained by differentiatin

$$P(t \le \tau < t$$

where the mean or avera
is $1/\lambda^2$ (see section 3.2.

17.2.3 Other arrival

These can be specified

- Unpunctual – whi
  random variable;
- Discrete-time arriv
- Non-stationary – v
- Correlated – when

17.3 The servic

The service or trans
which defines the ca
given traffic.

17.3.1 Service ti

As for arrival times
times or alternative
times, the latter aga

$$P(t) =$$

for a service rate

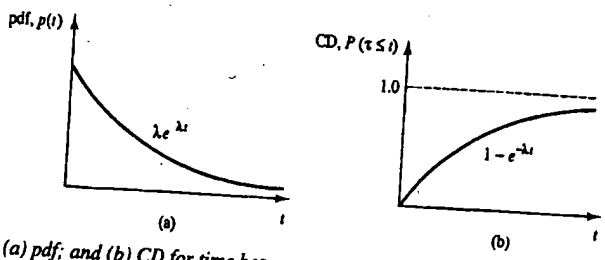


Figure 17.5 *(a) pdf; and (b) CD for time between successive arrivals.*

(3.13(b)), for the time between arrivals, Figure 17.5. This implies an exponential interarrival probability density function (pdf). As shown in section 3.2.4, the pdf is obtained by differentiating the CD, i.e. equation (17.7(c)):

$$P(t \le \tau < t + dt) = p(t) = \frac{dP(\tau \le t)}{dt} = \lambda e^{-\lambda t} \qquad (17.7(d))$$

where the mean or average value of t is $1/\lambda$ and the variance (or second central moment) is $1/\lambda^2$ (see section 3.2.5).

17.2.3 Other arrival patterns

These can be specified to be:

- Unpunctual – which occur at $t = a + E_1, 2a + E_2, \cdots, ma + E_m$, where E_m is a random variable;
- Discrete-time arrivals – these can only occur at a discrete set of allowed instants;
- Non-stationary – where the probabilities vary with time;
- Correlated – where the arrival rate may be affected by the state of the system.

17.3 The service process

The service or transmission mechanism is described by the service time distribution, which defines the capacity, or number of servers, which must be deployed to handle the given traffic.

17.3.1 Service time distributions

As for arrival times there are two extremes. We can have constant (deterministic) service times or alternatively we can have 'completely random' (stastically independent) service times, the latter again leading to an exponential pdf given by:

$$P(t) = \mu e^{-\mu t} \qquad (17.8)$$

for a service rate of μ customers/s, Figure 17.1. As before, the mean value, or average

service time, is $1/\mu$ and the variance is $1/\mu^2$. The statistical independence of succ... service times (resulting in an exponential distribution of service time pdf) means ... service time, like arrival interval, has been modelled as a Poisson process. It should ... noted, however, that:

- service times may be discrete (word or packet multiples);
- service time may be non-stationary.

The queuing discipline determines how customers are selected from the queue and allocated to servers.

17.3.2 Single server queues

These typically use one of the following queuing disciplines:

- First-in-first-out (FIFO) – the simple queue;
- Last-in-first-out (LIFO or last come first served, LCFS);
- First-in-random-out (FIRO);
- Priority queuing.

17.3.3 Multiserver queues

Here service is allocated according to rules such as:

- Rotation – customers assigned in strict rotation to each queue;
- Random selection – customers themselves decide which queue to join;
- Single queue – customer at the head of queue goes to the next available server.

17.4 The simple single server queue

It is instructive to find the distribution of queue lengths and waiting times in an M/M/1 system, where total waiting time equals queuing time plus service time.

17.4.1 Simple queue analysis

We define the *state* of the system as the number of customers waiting (i.e. state m implies m customers waiting or a queue of length m), and denote the probability of being in a state m at time t as $P_m(t)$. Assume, when the system is in state m, customers arrive randomly at an average rate λ_m, and are randomly serviced at rate μ_m (i.e. the average service time is $1/\mu_m$).

What is the probability of such a system, Figure 17.6, being in state m at time $t + \delta t$? This is obtained by extending equation (17.1) to include departures as well as arrivals:

$$\begin{aligned} P_m(t+\delta t) &= P_{m-1}(t)\lambda_{m-1}\delta t + P_{m+1}(t)\mu_{m+1}\delta t + P_m(t)(1-\mu_m\delta t)(1-\lambda_m\delta t) \\ &= P_{m-1}(t)\lambda_{m-1}\delta t + P_{m+1}(t)\mu_{m+1}\delta t + P_m(t)[1-(\mu_m+\lambda_m)\delta t] \end{aligned} \quad (17.9)$$

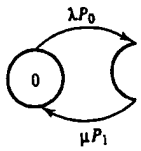


Figure 17.6 *State transitio*

(for small δt). Therefore

$$\frac{P_m(t+\delta t)-}{\delta t}$$

In the limit as $\delta t \to 0$:

$$\frac{dP_m(t)}{dt}$$

(for $m \geq 0$ where $P_{-1}($ and Pujolle] for arrival with time for state m i $m-1$ and $m+1$, are minus the rate at whic probability of state m. The process starts in s

$$P_m(t_0) =$$

and assuming a statio

$$0 = \lambda_m$$

where:

$$P_{-1} =$$

$$\lambda_{-1} =$$

$$\mu_0 =$$

and:

$$P_0 + P$$

A typical queue r staircase occur wi staircase imply s displacement meas

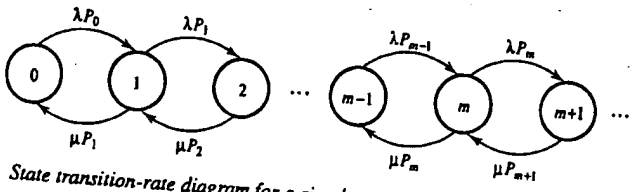


Figure 17.6 *State transition-rate diagram for a simple queue.*

(for small δt). Therefore:

$$\frac{P_m(t+\delta t) - P_m(t)}{\delta t} = \lambda_{m-1}P_{m-1}(t) + \mu_{m+1}P_{m+1}(t) - (\mu_m + \lambda_m)P_m(t) \quad (17.10)$$

In the limit as $\delta t \to 0$:

$$\frac{dP_m(t)}{dt} = \lambda_{m-1}P_{m-1}(t) + \mu_{m+1}P_{m+1}(t) - (\mu_m + \lambda_m)\,P_m(t) \quad (17.11)$$

(for $m \geq 0$ where $P_{-1}(t) = 0$). This is the full Chapman-Kolmogorov equation [Gelenbe and Pujolle] for arrivals and departures. It states that the rate of increase of probability with time for state m is equal to the rate at which transitions into that state, from states $m-1$ and $m+1$, are occurring (multiplied by the current probability of those states) minus the rate at which transitions out of state m are occurring (multiplied by the current probability of state m). To solve equation (17.11) we must specify the initial conditions. The process starts in state zero, as there are no arrivals before time t_0, i.e.:

$$P_m(t_0) = \begin{cases} 1, & m = 0 \\ 0, & m > 0 \end{cases}$$

and assuming a stationary solution so that $dP_m(t)/dt = 0$ then:

$$0 = \lambda_{m-1}P_{m-1} + \mu_{m+1}P_{m+1} - (\mu_m + \lambda_m)\,P_m \quad (17.12(a))$$

where:

$$P_{-1} = P_{-2} = \cdots = 0 \quad (17.12(b))$$

$$\lambda_{-1} = \lambda_{-2} = \cdots = 0 \quad (17.12(c))$$

$$\mu_0 = \mu_{-1} = \cdots = 0 \quad (17.12(d))$$

$$P_0 + P_1 + P_2 + \cdots = 1 \quad (17.13)$$

A typical queue result is shown in Figure 17.7. Here the vertical steps in the solid staircase occur with new customers arriving while the vertical steps in the dashed staircase imply service has been completed for these customers. The horizontal displacement measures the queue plus service time for each customer or unique arrival.

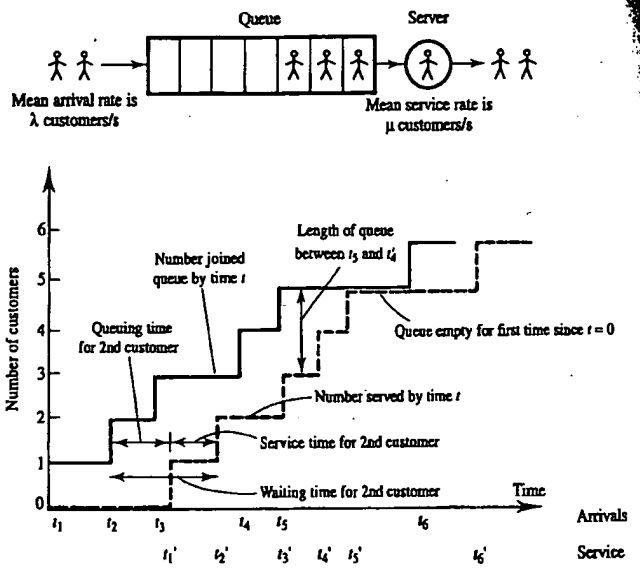


Figure 17.7 *Typical queue performance showing customer (packet) arrivals and departures (after service).*

17.4.2 Queue parameters

The total delay in a time t, $\gamma(t)$, is the sum of the waiting times. If the total number of customers who have arrived in a time t is denoted by $\alpha(t) = \lambda t$ then:

$$\text{average delay,} \qquad T = \frac{\gamma(t)}{\alpha(t)} \tag{17.14(a)}$$

$$\text{average queue length,} \qquad N = \frac{\gamma(t)}{t} \tag{17.14(b)}$$

$$\text{average arrival rate,} \qquad \lambda = \frac{\alpha(t)}{t} \tag{17.14(c)}$$

Now we can rewrite the average queue length as $N = \gamma(t)/t = (\gamma(t)/\alpha(t)) \times (\alpha(t)/t)$, where N is given by the sum, over all m, of the product of queue length, m, and its probability, P_m, to obtain Little's result:

$$N = \sum_{m=0}^{\infty} mP_m = T\lambda \tag{17.15}$$

The performance of a queuing system is controlled by its utilisation factor, defined as:

$$\rho = \frac{\text{demand for service}}{\text{maximum rate of supply}} \tag{17.16}$$

...here the demand for service ...easure of the traffic intensity ...neer of teletraffic theory. (...qual to $\lambda\tau$ where λ is the a ...uration. A circuit carrying or The service rate is often c ...ength. For example a 2 Mbi ...as a service rate, $\mu = (2 \times$ ...aximum capacity or rate of . In this case:

$$\rho = \lambda/\mu$$

...and $\rho < 1$ is required to pre

17.4.3 Classical queue

Assume a Poisson arrival ... arrival and service rates ar ... λ and μ_m simplifies to μ. customers are served on F

From equation (17.5) ... detailed balancing, the in ... same rate. Thus starting

$$P_1 = \frac{\lambda}{\mu} P_0$$

$$P_2 = \frac{\lambda^2}{\mu^2}$$

$$P_m = \left(\frac{\lambda}{\mu}$$

But $\sum_{m=0}^{\infty} P_m = 1$, as ... $\sum_{m=0}^{\infty} P_m = \sum_{m=0}^{\infty} \rho^m P_0$ ... $\sum_{m=0}^{\infty} \rho^m = 1/(1-\rho)$.

$$P_0 = 1$$

Figure 17.8 (curve ... empty queue, P_0, de

$$P_m =$$

implying a geometr ... is $1 - P_0 = \rho$. $\rho <$ ... 17.9 shows a typic

where the demand for service = arrival rate × mean service time = λ/μ. This is also a measure of the traffic intensity in erlangs [Dunlop and Smith], named after the Danish pioneer of teletraffic theory. (In telephony systems the total applied traffic in erlangs is equal to $\lambda\tau$ where λ is the arrival rate for call connections and τ is the average call duration. A circuit carrying one call continuously then carries one erlang of traffic.)

The service rate is often controlled directly by the output transmission rate and packet length. For example a 2 Mbit/s link (Chapter 19) with 500 byte packets and 8-bit bytes, has a service rate, $\mu = (2 \times 10^6)/(500 \times 8) = 500$ packet/s. For single server queues, maximum capacity or rate of supply is 1 s of service/s, i.e. the maximum rate of supply = 1. In this case:

$$\rho = \lambda/\mu \tag{17.17}$$

and $\rho < 1$ is required to prevent server overload.

17.4.3 Classical queue with single server

Assume a Poisson arrival process and exponentially distributed service times so that the arrival and service rates are independent of the state of the system. Thus λ_m simplifies to λ and μ_m simplifies to μ. Further assume that infinite queuing space is available and that customers are served on FIFO basis.

From equation (17.5) and Figure 17.6, by applying the conditions of equilibrium or detailed balancing, the input and output transitions to and from a state must occur at the same rate. Thus starting with state zero, $\lambda P_0 = \mu P_1$, these balances can be written as:

$$P_1 = \frac{\lambda}{\mu} P_0$$

$$P_2 = \frac{\lambda^2}{\mu^2} P_0$$

$$P_m = \left(\frac{\lambda}{\mu}\right)^m P_0 = \rho^m P_0 \tag{17.18}$$

But $\sum_{m=0}^{\infty} P_m = 1$, as shown previously (Figure 17.4 and equation (17.6)). Now $\sum_{m=0}^{\infty} P_m = \sum_{m=0}^{\infty} \rho^m P_0 = P_0 \sum_{m=0}^{\infty} \rho^m = 1$ and the sum of the geometric series $\sum_{m=0}^{\infty} \rho^m = 1/(1-\rho)$. Therefore:

$$P_0 = 1 - \rho = 1 - \frac{\lambda}{\mu} \tag{17.19}$$

Figure 17.8 (curve (a)) shows, as ρ increases from 0 to 1, how the probability of an empty queue, P_0, decreases. Also by applying equation (17.18):

$$P_m = \rho^m P_0 = \rho^m (1 - \rho) \tag{17.20}$$

implying a geometric distribution for P_m. The probability of the single server being busy is $1 - P_0 = \rho$. $\rho < 1$ ensures that the server has more capacity than is required. Figure 17.9 shows a typical queue length pdf, for $\rho = 0.5$.

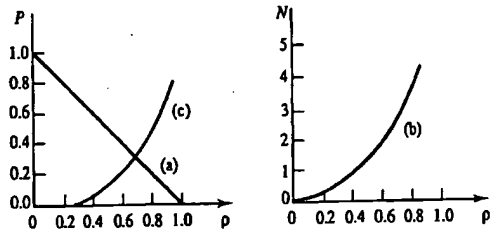


Figure 17.8 *Average queue size or length: (a) probability of an empty queue; (b) mean queue length, N, in packets; (c) probability that queue length exceeds 4.*

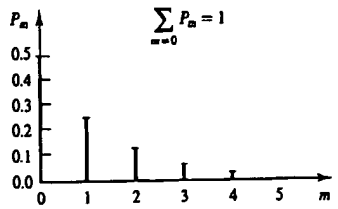


Figure 17.9 *Queue length pdf for ρ = ½.*

EXAMPLE 17.1

For a single server queuing system with Poisson distributed arrivals of average rate 1 message/s and Poisson distributed service of capacity 3 messages/s calculate the probability of receiving no messages in a 5 s period. Also find the probabilities of queue lengths of 0, 1, 2, 3. If the queue length is limited to 4 what percentage of messages will be lost?

From equation (17.5(a)):

$$P_0(t) = e^{-\lambda t}$$

and $\lambda = 1$ and $t = 5$ for a 5 s period. Thus the probability of no arrivals in a 5 s period is given by:

$$P_0(5) = e^{-5} = 0.00674$$

Now from equation (17.18):

$$P_m = \left(\frac{\lambda}{\mu}\right)^m P_0$$

and from equation (17.19):

$$P_0 = 1 - \frac{\lambda}{\mu} = 1 - \frac{1}{3} = \frac{2}{3}$$

Thus the various queue lengt

$$P_1 = \frac{1}{3} \times \frac{2}{3} =$$

$$P_2 = \left(\frac{1}{3}\right)^2 \times \frac{2}{3}$$

$$P_3 = \left(\frac{1}{3}\right)^3 \times \frac{2}{3}$$

This differs slightly from
subsequent magnitudes fall
of exceeding this restricted

$$P(m > 4) =$$

$$=$$

17.4.4 Queue lengt

The average queue len
as the sum of the geon
by:

$$N = \sum_{m=}^{\infty}$$

Figure 17.8 (curve (b
for $\rho > ½$, N increas
queue length becom
probability of exceed

$$P(m > 1$$

This is shown in F
about 0.8 there is

Thus the various queue lengths can be calculated as:

$$P_1 = \frac{1}{3} \times \frac{2}{3} = \frac{2}{9}$$

$$P_2 = \left(\frac{1}{3}\right)^2 \times \frac{2}{3} = \frac{2}{27}$$

$$P_3 = \left(\frac{1}{3}\right)^3 \times \frac{2}{3} = \frac{2}{81}$$

This differs slightly from Figure 17.9 in that the P_0 value is larger and, in consequence, the subsequent magnitudes fall off more rapidly. Finally for a queue length limited to 4 the probability of exceeding this restricted length is given by:

$$P(m > 4) = 1 - P(m \leq 4) = 1 - (P_0 + P_1 + P_2 + P_3 + P_4)$$

$$= 1 - \left(\frac{2}{3} + \frac{2}{9} + \frac{2}{27} + \frac{2}{81} + \frac{2}{243}\right) = 0.0041 = 0.41\%$$

17.4.4 Queue length and waiting times

The average queue length, equation (17.15), $N = \Sigma_{m=0}^{\infty} m P_m = (1-\rho)\Sigma_{m=0}^{\infty} m\rho^m$. Further, as the sum of the geometric series $\Sigma_{m=0}^{\infty} m\rho^m = \rho/(1-\rho)^2$ the mean queue length is given by:

$$N = \sum_{m=k+1}^{\infty} P_m = \frac{\rho}{1-\rho} \tag{17.21}$$

Figure 17.8 (curve (b)) shows how N increases with increasing ρ. At $\rho = ½$, $N = 1$ and for $\rho > ½$, N increases above unity. As traffic intensity increases and ρ approaches 1 the queue length becomes infinite. If the queue is restricted to some finite value k then the probability of exceeding this value is given by:

$$P(m > k) = \sum_{m=k+1}^{\infty} P_m$$

$$= (1-\rho) \sum_{m=k+1}^{\infty} \rho^m$$

$$= (1-\rho)\left[\sum_{m=0}^{\infty} \rho^m - \sum_{m=0}^{k} \rho^m\right]$$

$$= (1-\rho)\left[\frac{1}{1-\rho} - \frac{1-\rho^{k+1}}{1-\rho}\right]$$

$$= \rho^{k+1} \tag{17.22}$$

This is shown in Figure 17.8 (curve (c)) for various values of ρ. Clearly when ρ exceeds about 0.8 there is a problem. To find average delay or waiting time, T, we use Little's

result, equation (17.15) and equation (17.21):

$$T = \frac{\text{average queue length}}{\text{average arrival rate}} = \frac{N}{\lambda} = \frac{\rho}{\lambda(1-\rho)} \qquad (17.23(a))$$

or, using equation (17.17):

$$T = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu-\lambda} \qquad (17.23(b))$$

Average delay (normalised by μ) is plotted in Figure 17.10, against ρ in the range $0 \leq \rho < 1$, and it is this key result which forms the basis of network delay analysis. When constrained by a finite queue length of k, then $\Sigma_{m=0}^{k} P_m = 1 = P_0 \Sigma_{m=0}^{k} \rho^m$. For this case $P_0 = (1-\rho)/(1-\rho^{k+1})$.

For packets transmitted over a link, at a bit rate of R_b bit/s with packet size K bits, the mean packet delay is, from equation (17.23(b)):

$$T = \frac{1}{R_b/K - \lambda} \qquad (17.24)$$

where R_b/K represents the packet transmission or service rate, μ, in packet/s and λ is the packet arrival rate.

EXAMPLE 17.2

Consider a switch at which packets arrive according to a Poisson distribution. The mean arrival rate is 3 packet/s. The service time is exponentially distributed with a mean value of 100 ms. Assume the packet comprises 70 8-bit bytes and the output transmission rate is 5.6 kbit/s. How long does a packet have to wait in the queue?

The mean service rate is $\mu = 5600/(8 \times 70) = 10$ packet/s. From equation (17.23(b)):

$$T = 1/(\mu(1-\rho)) = 1/(\mu-\lambda)$$

and we find that the mean packet delay is $T = 0.143$ s for queuing plus service. Since the mean service time is 100 ms, the mean queuing time is therefore 143 – 100 = 43 ms.

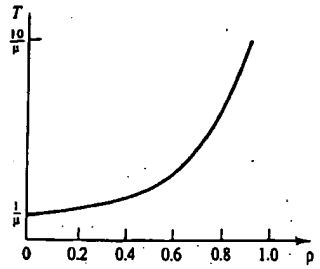


Figure 17.10 *Average time delay (T) for queue against ρ.*

Calculations, such a
links. The mean overa
times their delay divide
analysis is also used fo
Chapter 18 in order to f

17.5 Packet spe

Early packet networks
- store-and-forward,
(with a channel cap
- satellite networks,
kbit/s);
- radio networks, e.
kbit/s, for local dat

The most general
system (Figure 17.11),
packets are switched
each packet. The inter
a host computer, or a
full range of control an

17.5.1 The compon

Analogue speech mu
waveform processing,
as LPC, section 9.7. T
speech must be compr
(1.7 to 7.4) than delta
kbit/s.

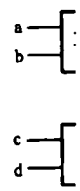


Figure 17.11 *Conventi*
example,

Calculations, such as those shown above, allow us to model the delays on packet data links. The mean overall network delay is given by the sum of the transmitted packets times their delay divided by the total number of transmitted packets. (This type of queue analysis is also used for determining the performance of the Banyan switch networks in Chapter 18 in order to find their blocking performance.)

17.5 Packet speech transmission

Early packet networks for communication (see Chapter 18) were:

- store-and-forward, hard-wired, long-haul networks, e.g. the American ARPANET (with a channel capacity of 50 kbit/s), and the British SuperJANET (Chapter 18);
- satellite networks, e.g. the American Atlantic SATNET (with a channel capacity of 64 kbit/s);
- radio networks, e.g. the American PRNET (with a channel capacity of 100 to 400 kbit/s, for local data distribution).

The most general distinction is that, unlike the circuit switched TDMA telephone system (Figure 17.11), in a packet switched network the individual data (D) and voice (V) packets are switched (Figure 17.12) in accordance with the header information within each packet. The interface between the user and a packet network can be a data terminal, a host computer, or a packet voice terminal which comprises a telephone handset with a full range of control and signalling capabilities.

17.5.1 The components of packet speech

Analogue speech must first be converted into a digital sequence by a coder using waveform processing, e.g. delta modulation, Figure 5.28, or other coding techniques such as LPC, section 9.7. The vocoder is favoured when there is limited channel capacity and speech must be compressed to lower data rates than PCM. LPC produces fewer packet/s (1.7 to 7.4) than delta modulation which generates 9.4 to 38.5 packet/s at a bit rate of 16 kbit/s.

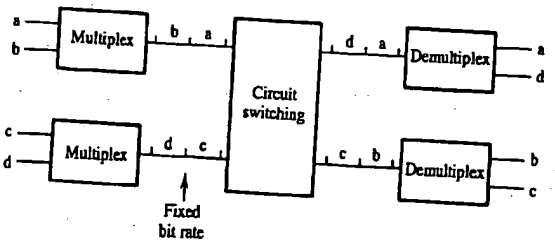


Figure 17.11 *Conventional circuit switched network (e.g. TDMA digital telephony transmission example, as described in section 6.5).*

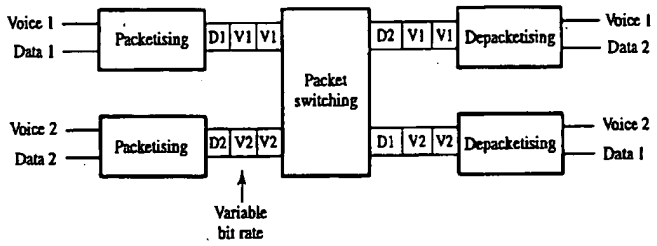


Figure 17.12 *Packet switched network with variable rate traffic where voice rate exceeds the input data rate.*

The digital bit stream is next partitioned into segments and some control (header) information added to each segment to form a packet. The control information consists of a time stamp and a sequence number, Figure 17.13, to assist reconstruction at the receiver. To reduce the bandwidth required for speech transmission, silences between bursts of speech are not packetised or transmitted. The time stamp enables the receiver to generate the appropriate duration of silence before processing the subsequent speech samples. Time stamps also allow reordering of packets which are received out of order. Since the time stamp cannot differentiate silence from packet loss, a sequence number is included to allow detection of lost packets.

It is important that a continuous stream of bits is provided at the receiver in order to produce smooth speech. This is achieved by delaying the arriving packets at the receiver queueing buffer, Figure 17.14. The size of this buffer must be sufficiently long to avoid packet loss due to overflow. Normally the delay will be chosen so that, statistically, a large proportion (e.g. 95%) of packets will be expected to arrive in the time allocated to the buffer delay, which is usually in the range 100 to 170 ms. The delay must not be so long that it dominates the performance of the speech transmission system, however.

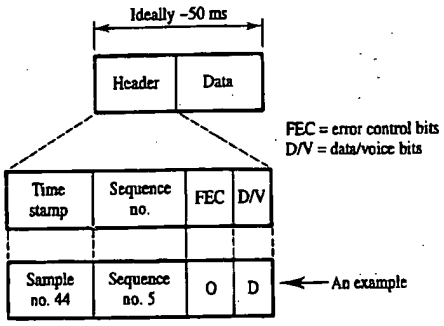


Figure 17.13 *Details of the header information carried within a data packet.*

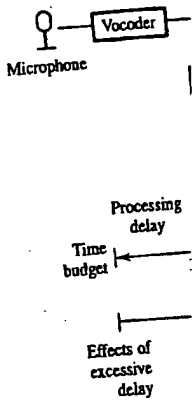


Figure 17.14 *Packet speeci*

EXAMPLE 17.3

An X.25 packet switch has
each packet is 960 bytes.
queue, is to be less than 1
(ii) the average length of t
the input is converted into
19.7.2 for an explanation c

In the X.25 switch the ou
service time is equal to pa
(i) Now if $T \leq 15$ ms, fr

$$T = \frac{1}{\mu(1 -$$

Maximum gross inpu

(ii) Average length of qu

$$N = \rho/(1 -$$

(iii) ATM switch
Each packet is 960 l
The cell input rate i
Now the output rate

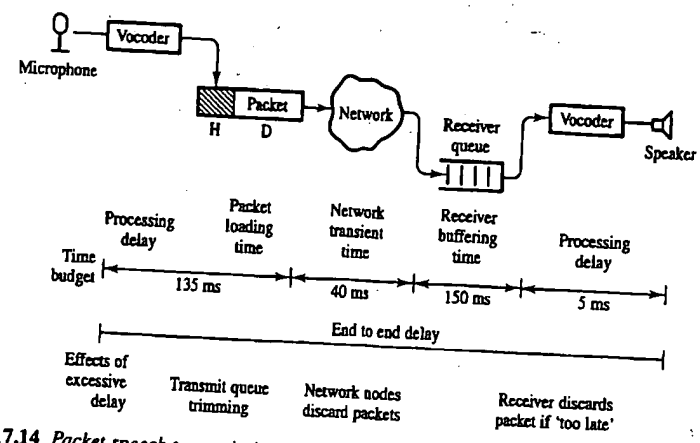


Figure 17.14 *Packet speech transmission system with inherent transmission delays.*

EXAMPLE 17.3

An X.25 packet switch has a single outgoing transmission link at 2 Mbit/s. The average length of each packet is 960 bytes. If the average packet delay through the switch, assuming an M/M/1 queue, is to be less than 15 ms, determine: (i) the maximum gross input packet rate to the switch; (ii) the average length of the queue; (iii) the utilisation factor through the switch if each packet in the input is converted into ATM cells having 48 bytes of data and a 5 byte overhead (see section 19.7.2 for an explanation of ATM).

In the X.25 switch the outgoing link is 2 Mbit/s. An average packet is 960 bytes or 7680 bits. If service time is equal to packet length, then: $\mu = 2 \times 10^6/7680 = 260.4$ packet/s

(i) Now if $T \leq 15$ ms, from equation (17.23(b)):

$$T = \frac{1}{\mu(1-\rho)} = \frac{1}{260.4(1-\rho)} \leq 15 \times 10^{-3} \quad \text{and} \quad \rho_{\min} = 0.744$$

Maximum gross input rate is $\lambda = \rho\mu = 0.744 \times 260.4 = 193.7$ packet/s.

(ii) Average length of queue from equation (17.21) is:

$$N = \rho/(1-\rho) = 2.91 \text{ packets}$$

(iii) ATM switch

Each packet is 960 bytes, and corresponds to 960/48 = 20 ATM cells.

The cell input rate is therefore $20 \times 193.7 = 3{,}874$ cell/s.

Now the output rate is $2 \times 10^6/(48+5) \times 8 = 4{,}717$ cell/s. Therefore $\rho = 3874/4717 = 0.821$.

17.5.2 Speech service performance

Three key parameters that describe the performance of a packet speech service are end-to-end delay, throughput and reliability. Delay is the time between speech being presented to the system, to the packet carrying that speech being played at the loudspeaker. Experimental tests show that few people notice any quality degradation if delay is kept below 0.3 s while a delay above 1.5 s is intolerable. Throughput is limited by the processing capability of the nodes. Reliability is defined as the proportion of packets that arrive at the destination in time to be used to reconstruct the speech.

Appropriate choice of packet size and rate can minimise delay and allow high throughput. In particular we must control the number of bits in the message header. Since the header is constant for every packet, regardless of size, to maintain high channel utilisation the number of speech bits per packet should be maximised. Large packets are also more desirable from the point of view of network throughput. However, to minimise the effects of lost packets and delay at the transmitter, packets should be short and, ideally, a packet should contain no more than 50 ms of speech. The trade-off is particularly difficult for narrowband speech, e.g. using LPC, because 50 ms of 2.4 kbit/s speech comprises only 120 data bits. Typical packet size for speech transmission across the Internet is about 300 bits comprising 100 to 170 ms speech segments. Channel loading information should be provided to the voice terminal so that packet rate and size can be varied according to the network load. In cases where the network is lightly loaded, it is capable of supporting a higher packet rate, hence smaller packets with less delay are used while, if the network loading is high, packets are made larger and packet rate is reduced.

17.6 Summary

Packets are groups of data bits to which have been added (as headers and/or trailers) addressing and other control information to facilitate routing through a digital data network. Queuing theory can be used to model packet behaviour at the switching nodes of a network and, in particular, can be used to predict average packet delay, average queue length and probability of packet loss at a node, given the packet interarrival-time pdf, the packet service-time pdf, the number of servers, the queuing discipline and the queuing storage space. For real-time applications, such as packet speech, resources can be saved by not transmitting empty or 'silent' packets. This necessitates packets being time stamped, however, for the receiver to regenerate the appropriate speech gaps.

In current Ethernet based networks data rates are in the range 10 to 100 Mbit/s and propagation delays (typically less than 5 μs) are small compared with the time required to transmit a 1 kbit packet. With the trend towards Gbit/s optical fibre links, operating over long distances, propagation delay becomes much longer than the packet duration, which will significantly alter the analysis of these systems.

17.7 Problems

17.1. The number of mes
network may be assumed
messages per minute, calcul
(a) Probability of receiving
(b) Probability of receiving
(c) Probability of receiving

17.2. In Problem 17.1: (
messages of greater than 2(
range 20 to 30 s inclusive?

17.3. Assuming a classical
empty queue, (b) a queue
for the service is 0.6. [0.4,

17.4. If the queue length i
the service? [0.36%]

17.5. A packet data netw
data. It is realised with a
London and Southend. Bo
the Northampton and Sou
50 packets/s to Northamp
mean size of 2 kbit? [1.2

17.7 Problems

17.1. The number of messages arriving at a particular node in a message switched computer network may be assumed to be Poisson distributed. Given that the average arrival rate is 5 messages per minute, calculate the following:
(a) Probability of receiving no messages in an interval of 2 minutes. [4.5×10^{-5}]
(b) Probability of receiving just 1 message in the next 30 s. [0.2]
(c) Probability of receiving 10 messages in any 30 s period. [2.16×10^{-4}]

17.2. In Problem 17.1: (a) What is the probability of having a gap between two successive messages of greater than 20 s? (b) What is the probability of having gaps between messages in the range 20 to 30 s inclusive? [0.189, 0.106]

17.3. Assuming a classical queue with a single server, determine the probabilities of having: (a) an empty queue, (b) a queue of 4 or more 'customers'. You should assume that the utilisation factor for the service is 0.6. [0.4, 0.13]

17.4. If the queue length in Problem 17.3 is limited to 10, what percentage of customers are lost to the service? [0.36%]

17.5. A packet data network links London, Northampton and Southend for credit card transaction data. It is realised with a 2-way link between London and Northampton and, separately, between London and Southend. Both links operate with primary rate access at 2 Mbit/s in each direction. If the Northampton and Southend nodes send 200 packets/s to each other and the London node sends 50 packets/s to Northampton what is the mean packet delay on the network when the packets have a mean size of 2 kbit? [1.275 ms]

CHAPTER 18

Network topology and protocols

18.1 Introduction

In 1969, the first major packet-switched communications network, the ARPANET, began operation. The network was originally conceived by the Advanced Research Projects Agency (ARPA) of the US Department of Defense for the interconnection of dissimilar computers, each with a specialised capability. Today systems range from small networks interconnecting microcomputers, hard-disks and laser-printers in a single room (e.g. Appletalk), through terminals and computers within a single building or campus (e.g. Ethernet), to large geographically distributed networks spanning the globe, e.g. the Internet. They are often classified as local, metropolitan or wide area networks (LANs, MANs or WANs). Figure 18.1 shows the relationships between LANs, MANs, WANs, the 'plain old telephone system' (POTS) and other more recent types of network. The major features of LANs, MANs and WANs are summarised in Table 18.1, after [Smythe 1991].

UK examples of WANs are the BT packet switched service (PSS) and the new joint academic network (SuperJANET). The original JANET interconnected all UK university

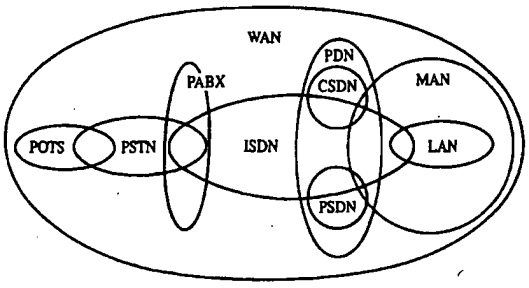


Figure 18.1 *Relationships between network architectures.*

ANs with a 5000 km l
ndividual university I
progressively increased
Manchester, Edinburgh,
and 64 kbit/s access rat
rate to 34 and 140 M
SuperJANET Phase B
standard SDH interfac
equivalent US system is

Table

Issue
Geographic
No. of node
R_b
P_b
Delays
Routing
Linkages

The general distin
network end-to-end
ratio is close to unity
unity. In the loosely
Broadly speaking

- Circuit-switched:
of communicatin
communications
of the PSTN. C
the circuit must
towards packet s
in section 19.7.2
- Message-switch
stored and forw
pairs are made
and are broken
path need there
- Packet-switche
are then route
forwarded at e
packets at the
circuit packet
network, and

LANs with a 5000 km backbone. With 100 Mbit/s transmission rates available on the individual university LANs the JANET backbone transmission rate has been progressively increased from 9.6 kbit/s to 2 Mbit/s. The main centres (London, Manchester, Edinburgh, etc.) were interconnected at 2 Mbit/s while other sites had 9.6 and 64 kbit/s access rates. The SuperJANET improvement upgraded the backbone bit rate to 34 and 140 Mbit/s using ATM (see section 19.7.2). Figure 18.2 shows the SuperJANET Phase B network, in which access rates are multiples of the 51.8 Mbit/s standard SDH interfaces (see section 19.4), within the PSTN core network. The equivalent US system is NSFnet which operates at 45 Mbit/s.

Table 18.1 *Comparison of LAN/MAN/WAN characteristics.*

Issue	*WANs*	*MANs*	*LANs*
Geographic size	1000s km	1 – 100 km	0 – 5 km
No. of nodes	10,000s	1 – 500	1 – 200
R_b	0.1 – 100 kbit/s	1 – 100 Mbit/s	1 – 100s Mbit/s
P_b	$10^{-3} - 10^{-6}$	$<10^{-9}$	$< 10^{-9}$
Delays	>0.5 s	100 – 100s ms	1 – 100 ms
Routing	sophisticated	simple	none
Linkages	gateways	bridges	bridges

The general distinction between a LAN, MAN and WAN depends on the ratio of the network end-to-end propagation delay to the average packet duration. For a MAN this ratio is close to unity while the more closely coupled LAN has a ratio much smaller than unity. In the loosely coupled WAN, the ratio is much larger than unity [Smythe 1991].

Broadly speaking, networks can be divided into three main categories:

- Circuit-switched: in which a continuous physical link is established between the pair of communicating data terminal equipments (DTEs) for the entire duration of the communications session. Circuit switched networks most commonly utilise portions of the PSTN. Circuit switching is inefficient for variable bit rate transmission since the circuit must always support the highest data rate expected – hence the move towards packet switching and asynchronous transfer mode, ATM. (ATM is discussed in section 19.7.2.)
- Message-switched: in which the complete message (of any reasonable length) is stored and forwarded at each data network node. Physical connections between node pairs are made only for the duration of the message transfer between those node pairs and are broken as soon as the message transfer is complete. No complete physical path need therefore exist between communicating DTEs at any time.
- Packet-switched: in which each message is divided into many standard packets which are then routed individually through the network. Each packet is stored and forwarded at each network node. Messages are reassembled from their constituent packets at the receiving DTE. Two distinct varieties of packet switching exist: virtual circuit packet switching in which all packets follow the same route through the network, and datagram packet switching in which different packets (within the same

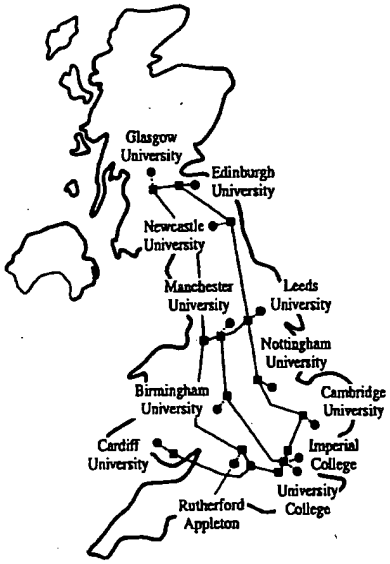


Figure 18.2 *SuperJANET wide area network (WAN).*

message) are routed entirely independently. Virtual circuit systems ensure that packets are received in their correct chronological order whilst datagram systems must include packet sequencing information for correct message reassembly.

Hybrids of, and variations on, the above switching philosophies are also sometimes used.

One of the major activities which accelerated the development of packet-switched technology to its present state was the development of the layered communications architecture concept. The proliferation of various architectures, creating possible barriers between different manufacturers' systems, led the International Standards Organisation (ISO) to launch the reference model for Open Systems Interconnection (OSI). This standardised the systems interfaces.

18.2 Network topologies and examples

Any network [Hoiki] must fundamentally be based on some interconnection topology, to link its constituent terminals. The main network topologies are reviewed here.

18.2.1 Point-to-point

his is undoubtedly the si
ransitory and exist only f
nay exist permanently, a
sed when a limited numt

18.2.2 Multidrop

When a large number of
are required to be conne
All transmissions from
node A can receive data
one time over the netwo
enforced by employing
permitting only the addr

Multidrop connectio
a single branched circui
this topology is the con
several locations. Mult
replacement of the PS
optical network (PON).

18.2.3 Star

Centralised switched-s
the PSTN and, for this
the star configuration
point links, to a centr
limitations:

Figure 18.3 *Multidr*

18.2.1 Point-to-point

This is undoubtedly the simplest wired network type, and is extensively used. It may be transitory and exist only for the duration of the call, as on the circuit-switched PSTN, or may exist permanently, as on a private (leased) line. This configuration is commonly used when a limited number of physically distinct routes are required.

18.2.2 Multidrop

When a large number of locations, which can be partitioned into geographical clusters, are required to be connected the multidrop configuration is often employed, Figure 18.3. All transmissions from node A can be received by nodes B, C and D. However, only node A can receive data from B, C and D, and only one of the latter may transmit at any one time over the network, as there is only one data transmission path. This constraint is enforced by employing a polling protocol at A which addresses B, C and D in turn, permitting only the addressed node to reply.

Multidrop connection provides a way of reducing transmission link costs by utilising a single branched circuit to connect A to B, C and D. The principal current application of this topology is the connection of host computers to terminals – or terminal clusters – at several locations. Multidrop connection is under consideration, however, for optical fibre replacement of the PSTN local loop copper connection (section 19.7.3) with a passive optical network (PON).

18.2.3 Star

Centralised switched-star network configurations have now existed for over a century in the PSTN and, for this reason, represent perhaps the best understood class of network. In the star configuration, the devices comprising the network are connected by point-to-point links, to a central node or computer, Figure 18.4. The star network has two major limitations:

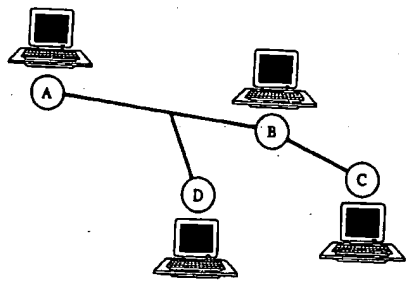


Figure 18.3 *Multidrop configuration.*

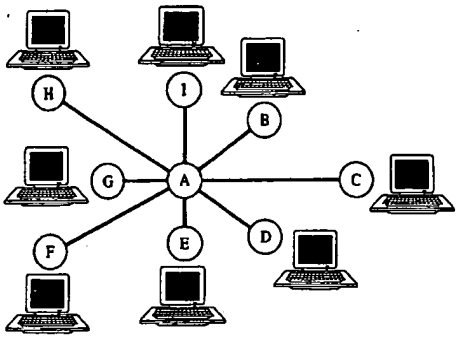


Figure 18.4 *Basic star network.*

- The remote devices are unable to communicate directly and must do so via the central node, which is required to switch these transmissions as well as carrying out its primary processing function.
- Such a network is very vulnerable to failure, either of the central node, causing a complete suspension of operation, or of a transmission link. Therefore, reliability and/or redundancy are particularly important considerations for this topology.

Despite these limitations, the star configuration is important as it has been used extensively for telephone exchange connections and in fibre optic systems. (Until recently it has been difficult to realise cheap, passive, optical couplers for implementation of an optical fibre ring or bus system, hence the attraction of the star network.) This topology is also used for many VSAT networks, section 14.3.9.

18.2.4 Ring

This network consists of a number of devices connected together to form a ring, Figure 18.5. Ring networks employ broadcast transmission, in that messages are passed around the ring from device to device. Each device receives each message, regenerates it, and retransmits it to its neighbour. The message is only retained, however, by the device to which it was addressed. Two variations of the ring network exist. These are:

- Unidirectional: in which messages are passed between the nodes in one direction only. The host, A, controls communication using a mechanism known as 'list polling'. The failure of a single data link will then halt all transmissions.
- Bidirectional: in which the ring is capable of supporting transmission in both directions. In the event of a single data link failing, the host, A, can then maintain contact with the two sectors of the network.

That each network node is involved in the transmission of all data on the network is a potential weakness. The ring topology is simple both in concept and implementation, however, and is popular for fibre optic LANs in which regenerative repeaters are required

Figure 18.5 *Ring networ*

at each node. Access i
follows.

A token bit pattern
data using a technique
'captures' the token b
'possessing' the token
in the ring acting as a
its header which is rec
data and signifies rec
trailer. It then retran
with the response bits
around the ring.

Cambridge ring

The Cambridge ring
data packets – slots
full/empty indicator
transmit a node occ
placing its message
before the sending
variation is to allow

The CR82 Cam
ring using twisted
monitor station che

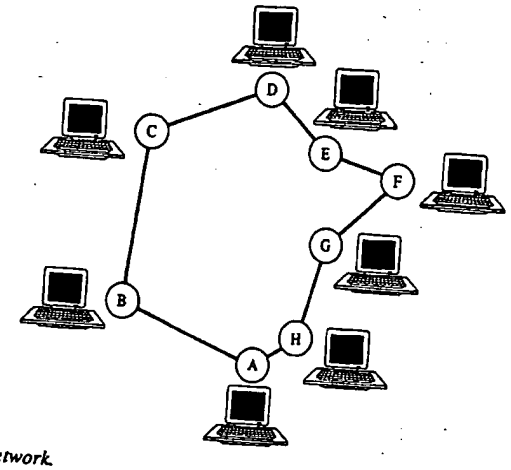


Figure 18.5 *Ring network.*

at each node. Access is via slots or tokens. A simple token ring operates essentially as follows.

A token bit pattern (e.g. 11111111), which is prevented from appearing in genuine data using a technique called bit stuffing, circulates while the ring is idle. A terminal 'captures' the token by removing it, or altering it (e.g. to 11111110). The terminal 'possessing' the token can then transmit one or more packets around the ring, each node in the ring acting as a repeater. Each transmitted packet contains a destination address in its header which is recognised by the destination terminal. The destination node reads the data and signifies receipt by setting response (or acknowledgement) bits in the packet trailer. It then retransmits the packet. When the sending terminal receives the packet with the response bits correctly set it resets the token to the idle pattern and recirculates it around the ring.

Cambridge ring

The Cambridge ring employs the empty slot principle. A constant number of fixed-length data packets – slots – circulate continuously around the ring in one direction only, a full/empty indicator within the slot header being used to signal the state of the slot. To transmit a node occupies the first empty slot by setting the full/empty flag to full and placing its message in the slot. The data packet then completes one revolution of the ring before the sending node 'empties' the slot and resets the indicator to empty. (A minor variation is to allow the receiver to empty the slot.)

The CR82 Cambridge ring, Figure 18.6, is a baseband implementation of a slotted ring using twisted wire pair cable, a bit rate of 10 Mbit/s and, typically, 10 slots. A monitor station checks which stations are active, and fills dummy slots to confirm that the

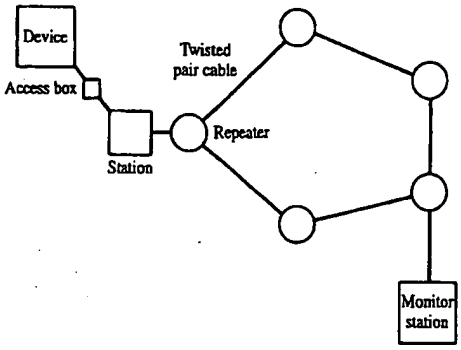


Figure 18.6 *Cambridge backbone ring example.*

ring is unbroken. With optical fibre implementations the data rate can be increased to 600 Mbit/s and beyond.

18.2.5 Mesh

While the star configuration is best suited for host computer/slave terminal connections on a one-to-many basis, mesh networks, Figure 18.7, are primarily used in a many-to-many situation, such as typically exists in WANs. Fully interconnected mesh networks, for more than a small number of nodes, are generally expensive as they require a total of $\frac{1}{2}n(n-1)$ links for n nodes. They are very resilient to failure, however, since alternative routes are available if a link fails. Where link lengths are long or data volumes low, a public packet-switched service may offer a significant cost advantage over a private mesh network. Unlike the ring or star topologies, adding a node to an existing (n node) mesh

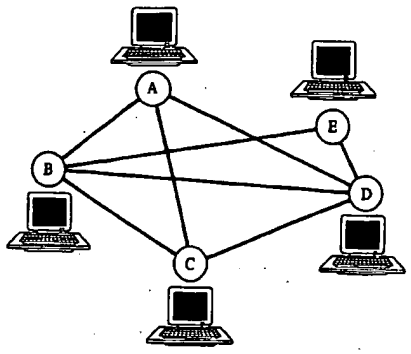


Figure 18.7 *Mesh network.*

...rk necessitates a fur

18.2.6 Bus

Bus networks employ a t
to which devices are atta
device are transmitted (b
devices accept only thc
required to retransmit th
associated with the ring
The bus configuratio
network will usually cc
advantage is that bus ne
often expanded to form
buildings. (Another br
traditionally used by sa
is discussed in section
bus used to connect dis

Ethernet bus

Ethernet [Hoiki] is ar
implementation a sim
employed. Of particu
as matched loads to
rates are 1 Mbit/s and
shows, for a 10 Mbit
bus for different level
10.8.1) for error detec

Te

Figure 18.8 *Bus net*

network necessitates a further $n - 1$ new connections.

18.2.6 Bus

Bus networks employ a broadcast philosophy. The bus is formed from a length of cable to which devices are attached by cable interfaces, or taps, Figure 18.8. Messages from a device are transmitted (bidirectionally) to all devices on the bus simultaneously; however, devices accept only those messages addressed to themselves. Since devices are not required to retransmit the messages, there is none of the delay (latency) or complexity associated with the ring topology.

The bus configuration is extremely tolerant of terminal failures, since operation of the network will usually continue if one of the active-component devices fails. A further advantage is that bus networks are easily reconfigured and extended. The bus topology is often expanded to form a tree structure, especially appropriate for use in multistorey buildings. (Another broadcast technique, similar in concept to a physical bus, is that traditionally used by satellite linked networks.) Access to the bus transmission medium is discussed in section 18.5. The small computer system interface (SCSI) is a dedicated bus used to connect discs and tape drives directly to the processor of a computer system.

Ethernet bus

Ethernet [Hoiki] is an example of a proprietary bus network, Figure 18.9(a). In this implementation a simple passive medium and transparent high impedance taps are employed. Of particular importance are the (coaxial cable) line terminators, which serve as matched loads to ensure reflections do not corrupt data transmission. Typical data rates are 1 Mbit/s and 10 Mbit/s, with a maximum bus length of 500 m. Figure 18.9(b) shows, for a 10 Mbit/s system, the expected number of attempts required to access the bus for different levels of network load. Ethernet uses 32-bit polynomial codes (section 10.8.1) for error detection.

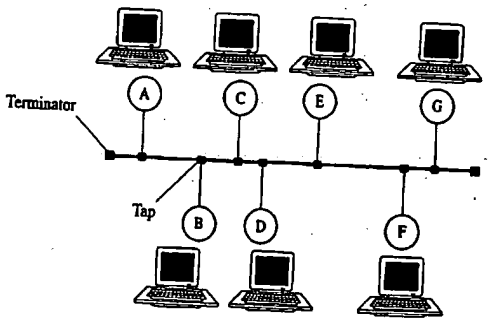


Figure 18.8 *Bus network.*

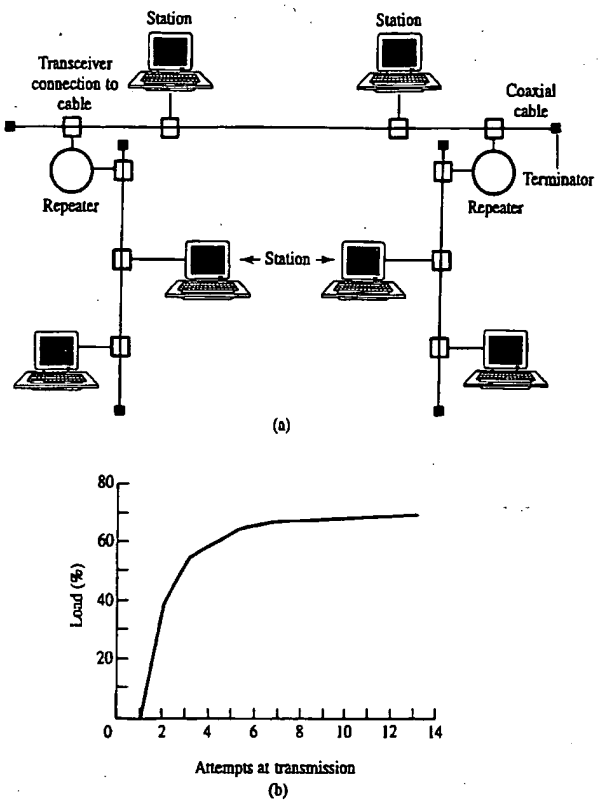


Figure 18.9 *Ethernet: (a) example configuration; (b) typical system response showing how the number of attempts at transmission varies with network load.*

18.2.7 Transmission media

Networks can utilise wire, coaxial cable, fibre optic or wireless links. Table 18.2 compares the different cabled and wireless transmission media. Metallic cable is preferred for many systems as simple, passive, tapped junctions cannot easily be realised for optical fibres. In all systems propagation loss, distortion, delay and noise are potential impairment mechanisms. Many systems use coaxial cable operating with baseband pulse rates up to 10 Mbit/s over 500 m paths. At higher rates 'broadband' data is modulated onto an RF carrier of, typically, 100 MHz, or fibre optic transmission (section 19.5) is used. Recently short runs of simple twisted wire pair have also become attractive for broadband transmission, because optical fibres are 5 to 10 times more expensive than twisted pair cable installations. In the near future broadband wireless LAN technology

sing carrier frequencies o

Table 18.2 *Typical cost*

Transmission medium	*Tw*
Range, m	1
Data rate, kbit/s	0
Cost/node, US dollars	

18.3 Network cov

The first computer WAN
With the rise in the numb
responsible for post, te
networks. These WANs
held in a node store b
another node nearer the
the end-to-end delay is c

- Public networks: Fc
international EURO
accessing informatic
- Private networks: F
required. Private t
such as SWIFT emp
- Value added netw
facilities combined
network. The user
include TRANSPA

Local area networ
computer systems and
building or university
the features of a WAN

- Wide bandwidth, c
- Low (1 to 10 μs) d
- Low probabilities
- Simple protocols
- Low cost and easy
- High degree of co
- Geographically b

Metropolitan net
LANs and WANs. A
access to a WAN ar

using carrier frequencies of 5 GHz and 17 GHz will become important.

Table 18.2 *Typical cost per node, data rate and ranges for different transmission media.*

Transmission medium	*Twisted pair*	*Coaxial*	*Fibre optic*	*Radio*	*Infra-red*
Range, m	1 – 1,000	10 – 10,000	10 – 10,000	50 – 10,000	0.5 – 30
Data rate, kbit/s	0.3 – 2,000	300 – 10,000	1 – 100,000	1 – 10	0.05 – 20
Cost/node, US dollars	10 – 30	30 – 50	75 – 200	50 – 100	20 – 75

18.3 Network coverage and access

The first computer WAN, ARPANET, spanned the US and was later extended to Europe. With the rise in the number of potential digital communications users, many of the bodies responsible for post, telegraph and telephone (PTT) services have now built such networks. These WANs generally use store and forward systems in which messages are held in a node store before being switched, via an appropriate transmission link, to another node nearer the message's ultimate destination. With 10 to 50 kbit/s data rates the end-to-end delay is of the order of seconds. WANs can be classified as follows:

- Public networks: For example, the British Telecom Gold electronic mail service, the international EURONET (a packet switched service (PSS), which exists primarily for accessing information databases) and the Internet.
- Private networks: For many users, public data networks do not provide the services required. Private telecommunications networks leased on a semi-permanent basis, such as SWIFT employed by the banking fraternity, are one alternative.
- Value added networks: A value added network (VAN) uses conventional PTT facilities combined with specialised message-processing services to add value to the network. The user is offered flexibility and the economies of shared usage. Examples include TRANSPAC and the specialised banking EFTPOS networks.

Local area networks (LANs) are specifically designed for the interconnection of computer systems and peripherals within a geographically small site, such as a single building or university campus, and are generally privately owned. A LAN has many of the features of a WAN, but it also has its own, distinct, characteristics, i.e.:

- Wide bandwidth, of the order of tens of Mbit/s.
- Low (1 to 10 μs) delay due to resource sharing, and absence of buffering.
- Low probabilities of bit error, typically 10^{-9} to 10^{-11}.
- Simple protocols (compared with those necessary for the longer ranges of WANs).
- Low cost and easy installation.
- High degree of connectability and compatibility of physical connections.
- Geographically bounded, with a maximum range of approximately 5 km.

Metropolitan networks or MANs which may be up to 50 km in diameter lie between LANs and WANs. An access method (or protocol) defines the set of procedures for LAN access to a WAN and vice versa. Since LAN transmission rates are much higher than

those of interconnecting WANs one LAN network node must normally be dedicated to the WAN interface. With high speed MAN interconnection of LANs it is possible to transfer large files electronically, rather than physically using, for example, discs, tapes or CD-ROMs. Two of the most common network interconnection techniques utilise bridges and gateways.

18.3.1 Bridges and gateways

A bridge is a device that interconnects two networks of the same type (using the same protocol). The bridge utilises a store and forward feature to receive, regenerate and retransmit packets while filtering the addresses between connected segments. A gateway on the other hand [Smythe 1995] connects networks using different protocols, typically LANs and WANs. They can therefore provide transparent access to resources on other, remote, networks. It is a similar device to the bridge but also performs the necessary protocol conversion. The JANET network of Figure 18.2 has transparent gateway connections to other X.25 international networks, which link it to the Internet, section 18.7.6.

18.3.2 Network switches

With the increasing power of VLSI technology, a large switch array can now be implemented on a single chip. National Semiconductor developed a 16 × 16 switching matrix in 2 μm CMOS gate array technology in the late 1980s. Figure 18.10 shows an 8 × 8 *Banyan* switch, consisting of twelve 2 × 2 switching elements. There is only one route through the switch from each input to each output. At each stage in this switch the upper or lower output is chosen depending on whether a specific digit in the route control overhead is 1 or 0. This is one of the simplest switch arrays that can be constructed and illustrates the self-routing capability of a packet navigating a network composed of such switches. When routing and sorting is performed locally at each switch in a network

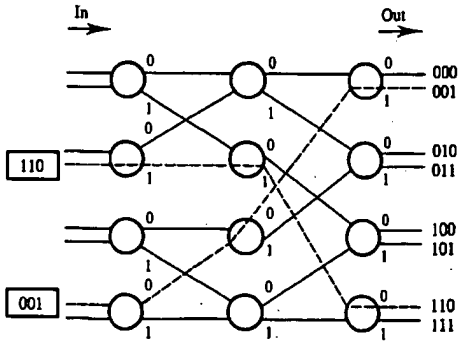


Figure 18.10 *8 × 8 Banyan packet switching network.*

here is no need for recor
herefore greater than th
how two example paths
network.
A problem inherent
locking (i.e. packets a
element). To reduce pa
the inputs or storage buf
To remove packet c
way that their paths ne
is known as Batcher so
descending order acco
crossbar switch which
The Starlight netwc
is shown in Figure 18.
1990s operated at 21(
problem of output bloc
but this can be overc
switching networks,
recirculating the dupli
need for buffered swit
The network is pac
same time. Banyan
packet's path through
presence of other pack
of up to 150 Mbit/s,
associated with a part

Pac
rando
por

Figure 18.11 *Starlig*

there is no need for recourse to a centrally located processing centre and the throughput is therefore greater than that of a circuit switched network. Dashed lines in Figure 18.10 show two example paths that packets with specific route headers would take through the network.

A problem inherent in all packet switched networks is that of packet collision or blocking (i.e. packets arriving simultaneously on the two inputs of a single switching element). To reduce packet collisions, the network can be operated at higher speed than the inputs or storage buffers can be introduced at the switch sites.

To remove packet collisions completely, the input packets must be sorted in such a way that their paths never cross. The technique normally used to perform this operation is known as Batcher sorting, in which the incoming packets are arranged in ascending or descending order according to route header. (Another non-blocking technique is the crossbar switch which was used in the 1970s for PSTN design.)

The Starlight network switch, which uses both Batcher sorting and Banyan switching, is shown in Figure 18.11. A CMOS 32 × 32 Batcher–Banyan switch chip in the early 1990s operated at 210 Mbit/s. Batcher–Banynan switching is not able to resolve the problem of output blocking, when multiple packets are destined for the same output port, but this can be overcome by the insertion of a trap network between the sorting and switching networks, Figure 18.11. The loss of trapped packets is overcome by recirculating the duplicate address packets into the next sorting cycle to eliminate the need for buffered switch elements.

The network is packet synchronous if it requires all packets to enter the network at the same time. Banyan networks can be operated packet asynchronously because each packet's path through the self-routing network is, at least to first order, unaffected by the presence of other packets. Packet switches such as these, which can operate at input rates of up to 150 Mbit/s, have been realised. (Note that the Banyan name is now often associated with a particular commercial system.)

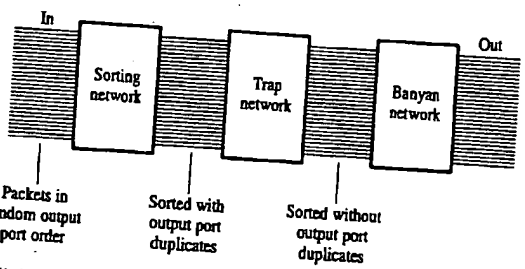


Figure 18.11 *Starlight packet sorting–switching network.*

18.4 Reference model for terminal interfacing

Most network architectures are organised as a series of layers. Previously the number, name and function of each layer differed from network to network. In all cases, however, the purpose of each layer was, and is, to offer certain services to higher layers, while shielding the details of exactly how those services are implemented in the lower layers.

Layer *n* at one node holds a conversation with layer *n* at another node. The rules and conventions used in this conversation are collectively known as the layer *n* protocol. The main functions of such a protocol are:

- Link initiation and termination.
- Synchronisation of data unit boundaries.
- Link control, with reference to polling, contention restriction and/or resolution, time-out, deadlock and restart.
- Error detection and correction.

The messages or blocks of data passed between entities in adjacent layers (i.e. across interfaces) are known as data units. With the exception of layer 1 no data is directly transferred from layer *n* at one node to layer *n* at another. Data and control information is passed by each layer to the layer immediately below, until the lowest layer is reached. It is only here that there is physical communication between nodes. The interfaces between layers must be clearly defined so that:

- The amount of data exchanged between adjacent layers is minimised.
- Replacing the implementation of a layer (and possibly its subordinate layers) with an alternative (which provides exactly the same set of services to its upstairs neighbour) is easily achieved.

18.4.1 The ISO model

This is a set of layers and protocols used to model network architecture, Figure 18.12. The overall purpose of the International Standards Organisation (ISO) model is to define standard procedures for the interconnection of network systems, i.e. to achieve open systems interconnection (OSI). ISO OSI processing is normally performed in software but, with the continuous rise in data rates, hardware protocol processors are, increasingly, being deployed. Several major principles were observed in the design of the ISO OSI model. These principles are that:

- A new layer should be created whenever a different level of abstraction is required.
- Each layer should perform a well defined service related to existing protocol standards.
- The layer boundaries should minimise information flow across layer interfaces.
- The number of layers should be sufficient so that distinct functions are not combined in the same layer, but remain small enough to give a compact architecture.

This has resulted in agreement to use seven layers as the OSI standard. The layers of the model are presented from the viewpoint of connection-mode transmission, starting from the interface to the physical medium. A key feature of the ISO OSI model is that it

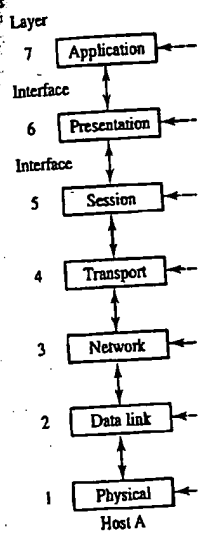


Figure 18.12 *ISO OS*

achieves standardisa
A summary of func
directly interfaces
mechanical aspects,
specification of vol
procedural aspects f

The task of th
transform it into a l
by breaking the in
processing acknow
Figure 18.13. Her
acknowledgement
from the remote t
retransmitted, as sh

The data link l
the protocols for a
10) are widely use
bit-serial data. T
packages, such as
facilities. The firs

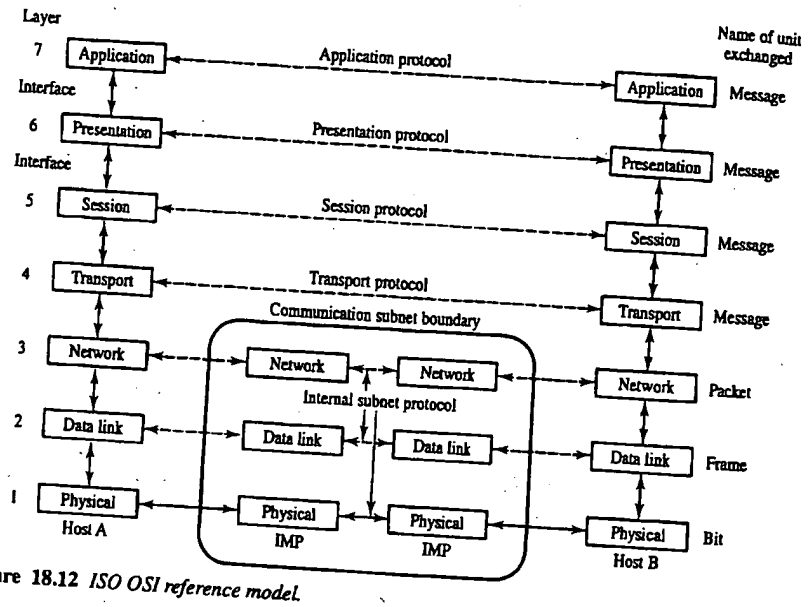


Figure 18.12 *ISO OSI reference model.*

achieves standardisation for data communications combined with error free transmission. A summary of function distribution is provided in Figure 18.12. The physical layer (1) directly interfaces with the transmission medium. This layer therefore includes: mechanical aspects, e.g. specification of cables and connectors; electrical aspects, e.g. specification of voltage levels, current levels and modulation techniques; functions and procedural aspects for the interface to the physical circuit connection.

The task of the data link layer (2) is to take the raw transmission facility and transform it into a link that appears free of transmission errors. It accomplishes this task by breaking the input data into data frames, transmitting the frames sequentially, and processing acknowledgement frames returned by the receiver. This is illustrated in Figure 18.13. Here once the packet is transmitted the transmitter stops and waits for an acknowledgement before the next packet is sent. If the acknowledgement does not arrive from the remote terminal within a prescribed time interval then the original packet is retransmitted, as shown in Figure 18.13 for packet 2.

The data link layer must both create and recognise frame boundaries. It thus defines the protocols for access to the network. Cyclic redundancy or polynomial codes (Chapter 10) are widely used in the data link layer to achieve an error detection capability on the bit-serial data. These processing functions are implemented in hardware. Software packages, such as Kermit, are located here to provide terminal emulation and file transfer facilities. The first two layers together are sometimes called the hardware layer.

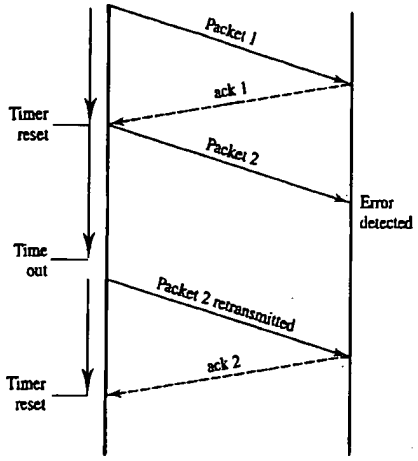


Figure 18.13 *Protocol for acknowledging (ack) successful packet receipt.*

The purpose of the network layer (3) is to provide an end-to-end communications circuit. It has responsibility for tasks such as routing, switching, and interconnection, including the use of multiple transmission resources, to provide a virtual circuit. The bottom three layers, collectively, implement the network function – and can be envisaged as a 'transparent pipe' to the physical medium, Figure 18.12. Data routed between networks, or from node to node within a network, require these functions alone.

The next layer (4) provides a transport service, suitable for the particular terminal equipment, that is independent of the network and the type of service. This includes multiplexing independent message streams over a single connection and segmenting data into appropriately sized units for efficient handling by the lower layers. Modulo $2^n - 1$ checksums (Chapter 10) are frequently computed within the software communications protocols of this layer, and compared with the received value, to determine whether data corruption has occurred. The transport layer can implement up to five different levels of error correction, but in LANs little error correction is required.

The functions of the session layer (5) are to negotiate, initiate, maintain and terminate sessions between application processes. This could for example be a transaction at a bank cash card machine. The layer operates in either half or full duplex, section 15.1.1.

While the session layer (5) selects the type of service, the network layer (3) chooses appropriate facilities, and the data link layer (2) formats the messages. The presentation layer (6) ensures that information is delivered in a form that the receiving system can interpret, understand and use. For example it defines the standard format for date and time information. The services provided by this layer include classification, compression and encryption of the data using codes and ciphers and also conversion, where necessary, of text between different types, e.g. to and from ASCII. The overall aim of layer 6 is to

make communication
The application l
Hence it provides res
emulation, distributed
electronic mail [Wilk
applications, layer 7
when network capaci
These ISO layers
systems and hence t
OSI equivalence. A
come to dominate e
order of magnitude l

18.4.2 Connectio

The ISO OSI model
modes. In the for
connection, which h
This mechanism is t
is employed in the
operate at up to 64 k
In connectionle
to a service, witho
self-contained data
appropriate where
in electronic mail o

18.4.3 Physical

For many years the
which defines the
and reverse) chan
PCs, printers, etc.,
can accommodate
distances (up to
RS423A (10 kbit
effects of noise pi

18.4.4 Synchrc

A key requireme
synchronisation,
synchronisation
network's termi

make communication machine independent.

The application layer (7) defines the network applications that support file serving. Hence it provides resource management for file transfer, virtual file and virtual terminal emulation, distributed processing and other functions. Conceptually this is where X.400 electronic mail [Wilkinson] and other network utility software resides. In electronic mail applications, layer 7 contains the memories which store the messages for forwarding when network capacity becomes available.

These ISO layers are equivalent to similar functions in other layered communications systems and hence there is commonality between OSI and other standards which have OSI equivalence. As one moves down the layers overhead bits (added by each layer) come to dominate each data packet making the effective data rate at layer 7 at least an order of magnitude less than the actual bit rate at the data link layer.

18.4.2 Connectionless transmission

The ISO OSI model can accommodate both connection and connectionless transmission modes. In the former an association between two or more peer entities, termed a connection, which has a clearly defined lifetime is established for data transfer purposes. This mechanism is the logical equivalent to a (circuit-switched) PSTN telephone call, and is employed in the virtual circuit strategy of X.25 packet-switched networks which operate at up to 64 kbit/s.

In connectionless-mode transmission data transfer can be achieved by a single access to a service, without the requirement for establishment and release phases. Here each self-contained data unit can be routed independently. This mode is particularly appropriate where interaction between pairs of entities is intermittent and infrequent, as in electronic mail or when bursty, variable bit rate, traffic occurs.

18.4.3 Physical layer protocol

For many years the most common, physical layer, connection standard has been RS232C, which defines the signal functions and their electrical characteristics in the two (forward and reverse) channel directions. RS232C is widely used for interconnecting terminals, PCs, printers, etc., and uses a minimum of three wires for the two channel connection. It can accommodate data rates of a little over 1 kbit/s on paths of up to 20 m. For longer distances (up to 1 km) and higher data rates, balanced transmission systems such as RS423A (10 kbit/s) and RS422A (1 Mbit/s) have been developed which reduce the effects of noise pickup on cables.

18.4.4 Synchronisation and line coding

A key requirement of the protocol processor is that it must be able to handle clock synchronisation, line encoding/decoding and frame synchronisation. A special synchronisation problem in LANs and MANs is that of the clock oscillators in the network's terminal equipment and nodes. For synchronised clock operation all

transmissions are synchronised with a common master station clock. For plesiochronous operation (section 19.3) each individual node receives data with a clock signal locked to the clock of the preceding node (or terminal). It then transmits data with its own local clock signal. Due to clock oscillator tolerance, this implies that the number of transmitted bits can differ from the number of information bits, requiring justification.

Line coding usually introduces redundancy into the data stream (see Chapter 6). This redundancy can be utilised for reliable clock recovery and for the coding of special control symbols. Long streams of consecutive identical bits are not desirable if this means that timing information cannot be regenerated easily from the incoming signal. Line codes can be divided into three classes, namely scrambled codes, bit insertion codes, and block codes. The function of a scrambler, in this context, is to generate transitions in the line-encoded data stream when the original data stream contains long sequences of identical bits, as in HDB3, section 6.4.5.

Examples of bit insertion codes are the 5B6B and 8B1C line codes. In the 5B6B-PMSI (periodic mark space insertion) scheme, the encoder alternately inserts a mark (one) and a space (zero) for every five information bits. In the 8B1C (8 binary with 1 complement insertion) scheme, every eight bits are preceded by one extra bit, which is the complement of the first of the eight bits. Block codes allow unique words to be transmitted for synchronisation purposes.

18.5 Medium access control

High-speed channel access schemes are divided into four main classes: random access, demand assignment, fixed assignment, and adaptive assignment protocols [Skov]. For each class, the protocols are further subdivided according to network topology.

18.5.1 Random access and demand assignment protocols

Random access is typified by the ALOHA packet switched systems which simply transmit whenever a message is ready to send and then waits to see if this transmission collides with other data on the system. This is very inefficient giving a maximum channel utilisation (or normalised throughput) of 18% [Kleinrock]. Slotted ALOHA, in which data is constrained to time slots, avoids partial packet overlap and therefore achieves a maximum utilisation of double this.

The best known access scheme for bus systems, because of its low installation cost, is carrier sense multiple access with collision detection (CSMA/CD). This scheme allows nodes to contend for use of the network and, in this sense, is similar to ALOHA. In CSMA/CD, however, any node ready to transmit first listens, sensing the carrier, to check whether another node is transmitting an information frame. If another transmission is already in progress, the node defers operation until the end of the current transmission. Once the network is free, the node transmits its addressed message. However, due to the non-zero propagation delay, two or more nodes can attempt to transmit simultaneously causing contention, which results in a collision. Given this situation, the transmission

ttempt is aborted and the
ossible step-lock. This is
ealise a channel utilisatio
ntroduce unpredictable net
With demand assignme
serving, or allocating acces
order. A representative t
multimode optical fibre sy
FDDI networks can emp
repeaters. The active o
regenerate data pulses w
18.7.4) has been develope
rate for image transmissi
circuit switching for voic

EXAMPLE 18.1 – Token p
A token passing ring opera
4-bit token. If the cable
access?

The total ring latency is 1
latency plus the token proc

A variation on the
propagate round the r
already filled with dat
which station empties
optic 600 Mbit/s Cam
Unidirectional bu
Access to the bus typi
access is employed,
begins its transmissi
from any upstream
reservation access.

18.5.2 Fixed assi

In fixed assignme
participating station
always occupies a p
protocols are TDM
section 15.5. Mos

attempt is aborted and the node waits a random period before retransmission to avoid possible step-lock. This is a more sophisticated access scheme than ALOHA and can realise a channel utilisation of 75%. At this utilisation efficiency, however, collisions introduce unpredictable network delay or latency.

With demand assignment access protocols, message transmissions are governed by serving, or allocating access to, the attached stations in a predetermined (typically cyclic) order. A representative token ring protocol (see section 18.2.4) for packet switched multimode optical fibre systems is the 125 Mbit/s fibre distributed data interface (FDDI). FDDI networks can employ ring lengths of 100 km with 2 km spacings between repeaters. The active optical repeaters receive light from the incoming fibre and regenerate data pulses which are coupled into the outgoing fibre. FDDI (see section 18.7.4) has been developed primarily as a dual broadband backbone ring with a high data rate for image transmission applications. Current developments in FDDI II are aimed at circuit switching for voice and video multimedia transmission.

EXAMPLE 18.1 – Token passing

A token passing ring operates at 2 Mbit/s with 10 stations, each with a latency of 2 bits, and uses a 4-bit token. If the cable delay round the ring is 10 μs what is the maximum waiting time for access?

The total ring latency is $10 + (10 \times 2)/2 = 20$ μs. Maximum waiting time is given by the total ring latency plus the token processing time which is $20 + 4/2 = 22$ μs.

A variation on the conventional token ring is to partition time into slots and let these propagate round the ring. Access to a slot is possible, as it passes, provided it is not already filled with data. Such *slotted rings* can be divided into two groups depending on which station empties a full slot, the source station or the destination station. The fibre optic 600 Mbit/s Cambridge fast ring, section 18.2.4, employs source release.

Unidirectional buses integrate traffic with different priorities by means of rounds. Access to the bus typically follows one of three access schemes. When attempt-and-defer access is employed, a station wishing to transmit waits until the media is idle. It then begins its transmission and continues to listen to the media. If it detects a transmission from any upstream station, it aborts its transmission. The other schemes use polled or reservation access.

18.5.2 Fixed assignment protocols

In fixed assignment protocols the total network resource is divided among the participating stations in time, frequency, or code domain, in a fixed way. Thus, a station always occupies a part of the channel capacity, whether or not it has data to transmit. The protocols are TDMA, Figure 5.2, FDMA, Figure 5.12, and contention free CDMA, see section 15.5. Most of the FDMA and CDMA experimental ring and star networks use

optical signalling. CDMA permits uncoordinated accessing but may not offer the capacity of FDMA and TDMA, unless power control is adopted to avoid the near-far problem (see section 15.5.6).

18.5.3 Adaptive assignment protocols

At low loads, a device can usually transmit sucessfully as soon as it senses that the channel is idle. As the load rises, however, packet collisions may occur, thus destroying data. In such cases, adaptive protocols may switch to a more restricted, conflict-free mode of operation such as token passing, attempt-and-defer, or TDMA access. An example of a high-speed network based on adaptive assignment access is the US 100 Mbit/s fibre optic HYPERchannel-100 network. This is based on a bus topology, and employs an access protocol that starts in CSMA/CD mode and switches to TDMA mode when collisions become too frequent.

18.6 International standards for data transmissions

18.6.1 X.25

This is the PSS standard which effectively replaces the telephone network, using the V-series recommendations (see section 11.6), with a digital system having superior error performance and fast switching. The X.25 recommendation defines the interface between terminal equipments and the public data network for packet-switched communication. A set of associated standards, X.3, X.28, X.29, have been developed to enable simple terminals to access an X.25 network. X.25 is actually a layered network access (interface) protocol that exhibits many of the properties of network architectures. The functionality of the X.25 specification corresponds entirely to the lower three layers in the ISO OSI model, Figure 18.12. In X.25, error checking is conducted at each node in the network. In the new frame relay systems this is only performed at the terminal stations.

X.25 PSS is now available within the UK as a core network for data communications, electronic mail, etc. One use of the network is for credit card verification and connections are available via telephone lines or radio access at 8 kbit/s [Davie and Smith]. Radio coverage in 1991 extended to 75% of the UK population for this data service.

18.6.2 IEEE 802

The IEEE 802.n specifications also map to the bottom three layers of the ISO OSI reference model. The IEEE split the data link layer into sub-layers: logical link control and medium-access control, which are used for bridging between networks. The 802.3 to 802.6 standards describe physical connections and define how access to the physical medium is coordinated for each LAN type. They therefore correspond to layer 1, and a sub-layer of layer 2, in the ISO OSI model.

- 802.2: Defines a lo
- 802.3: Defines a C
on coaxial, and twis
- 802.4: Defines a to
- 802.5: Defines a to
- 802.6: Is a broadb
- 802.7: Specifies a

These IEEE standa
eventually be replaced

18.7 Network e

18.7.1 Manufactur

The manufacturers a
multidrop bus LAN
modulation. By oper
MHz, simultaneous (
signalling (Chapter
manufacturing envir
interconnection of n
Mbit/s with a maxin
other similar networ
digitising, encoding

18.7.2 Admiral

The Alvey Admiral
1980s, consists of b
interconnected by a
configured as a star
central switch locat
switch providing c
time slots within t
required. User sit
commands to the s
first practical comb
ISDN interconnect

18.7.3 Military L

The US MIL-STI
applies to militar

- 802.2: Defines a logical link control layer.
- 802.3: Defines a CSMA/CD protocol, which is the basis of Ethernet, as implemented on coaxial, and twisted pair, cables.
- 802.4: Defines a token-passing bus protocol.
- 802.5: Defines a token-ring protocol (FDDI).
- 802.6: Is a broadband MAN standard (45 Mbit/s) for voice, data and video.
- 802.7: Specifies a broadband FDMA LAN (with 400 MHz of bandwidth).

These IEEE standards are now internationally accepted as ISO equivalent. 802.4 may eventually be replaced by FDDI-II.

18.7 Network examples

18.7.1 Manufacturers application protocol (MAP)

The manufacturers application protocol is implemented on a 10 Mbit/s broadband multidrop bus LAN using an 802.4 token passing bus channel employing RF carrier modulation. By operating on two separate carriers at frequencies between 59.74 and 264 MHz, simultaneous (FDM) data transmission and reception is accomplished using QAM signalling (Chapter 11). MAP is designed to interconnect plant (such as robots) in a manufacturing environment. Another commercial development is the fieldbus for interconnection of measuring instruments [Jordan]. In one network the bit rate is 1 Mbit/s with a maximum access time of 5 ms for a 32-node system. There are various other similar network designs, many of which share the IEEE 488 interface standard for digitising, encoding and transmitting signal samples from remote locations.

18.7.2 Admiral

The Alvey Admiral multimedia prototype network (Figure 18.14), developed in the mid 1980s, consists of baseband CSMA/CD Ethernet LANs at each of the five project sites, interconnected by a high speed network. This early example of a high speed network is configured as a star, with 2 Mbit/s primary rate access circuits from each user site to a central switch located at the star hub. At the centre of the network is a non-blocking switch providing configurable interconnection of any number of consecutive 64 kbit/s time slots within the 2 Mbit/s bearer to provide point-to-point links from site to site as required. User sites are able to alter the configuration of the network by sending commands to the switch controller, via the public X.25 network. This network was the first practical combination of fast local Cambridge rings and Ethernets with primary rate ISDN interconnections, as described in the following chapter.

18.7.3 Military LAN systems

The US MIL-STD-1553B and its UK equivalent DEF STAN 00-18 or STANAG 3838 applies to military ship, submarine and aircraft LANs. These 1970s systems used

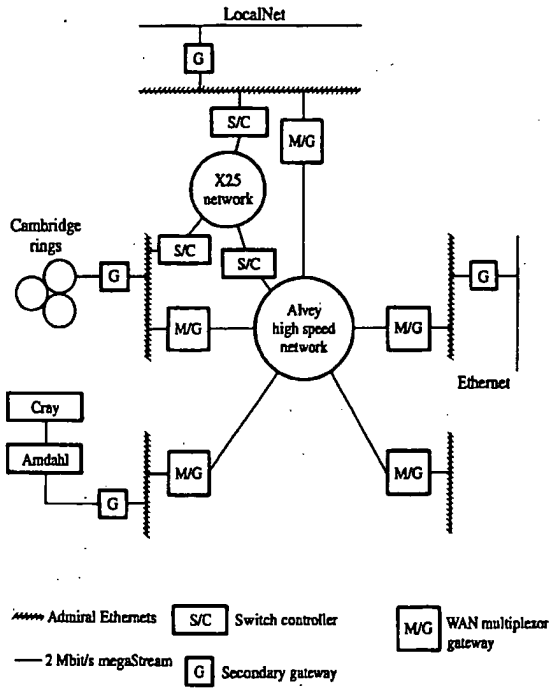


Figure 18.14 *The Admiral high-speed network.*

screened twisted pair cable to achieve 1 Mbit/s transmission rates with up to 31 remote terminals. Further US work has achieved high speed (20 to 40 Mbit/s) fibre optic networks while, in the UK, DEF STAN 00-19 is a 300 m multi-drop bus topology on a screened twin coaxial cable which signals at 3 Mbit/s. The bus for the European fighter aircraft, STANAG 3910, is a dual redundant 20 Mbit/s fibre optic implementation. It defines low speed 1 Mbit/s dual redundant wired channels plus the high speed 20 Mbit/s fibre optic channels. These specialised systems are not generally fully ISO compatible.

In contrast to civilian LANs, military ones require priority accessing schemes with short access delays for threat messages and must also be secure and free from transmission errors. These constraints inevitably result in specialised networks being developed for military applications.

18.7.4 Fibre distributed data interface (FDDI)

FDDI is a backbone broadband ring to which other, slower speed, tree networks and peripherals can be connected, Figure 18.15. (FDDI was conceived as a fast or broadband service to handle multimedia data transfer for applications such as desktop conferencing.)

Figure 18.15 *Networ*

The overall structure plug-in multimode fi techniques described code. Link P_b is 2. length of 200 km, Packet latency for Currently there is i cables, but this int structure of Figure employing ring net

18.7.5 Wireless

There is much cu Ethernet wired co proximity to each for 50 to 300 m employed to comb hundreds of kbit/s spectrum) module permits the multi for by equalisatio GHz carrier frequ

EXAMPLE 18.2
A wireless networ computers dispers obstructions in the space inverse squa

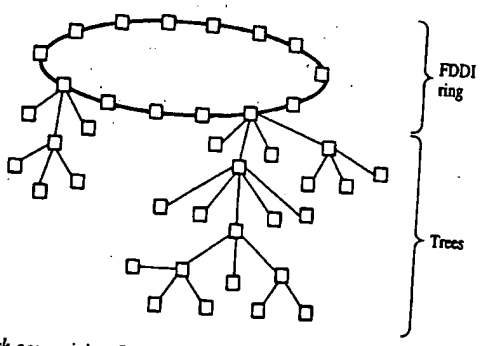


Figure 18.15 *Network comprising FDDI high speed ring with tree connections.*

The overall structure then looks like a ring-of-trees. FDDI operates at 100 Mbit/s as a plug-in multimode fibre ring (section 19.5) with 4B5B coding (which is a variation on the techniques described in section 6.4.7) implying a 125 MHz clock rate for this bit insertion code. Link P_b is 2.5×10^{-10} and packet error probability is 10^{-9}. On the maximum ring length of 200 km, ring transit time is 2.7 ms, corresponding to 60 full length packets. Packet latency for access to a network at 45 Mbit/s load is typically 50 to 200 μs. Currently there is interest in copper distributed data interface (CDDI) using twisted pair cables, but this interest is more intense in North America than Europe. The FDDI structure of Figure 18.15 is not dissimilar to the future SDH inner core proposals employing ring networks (see section 19.4.1).

18.7.5 Wireless networks

There is much current interest in developing wireless systems to obviate the need for Ethernet wired connections between terminals and printers which are located in close proximity to each other. The US NCR WaveLAN is such a system which typically caters for 50 to 300 m link connections. In the indoor environment, if equalisation is not employed to combat multipath responses, then the data rate is typically limited to several hundreds of kbit/s. In WaveLAN systems 2 Mbit/s data is spread by a wideband (spread spectrum) modulator into an 11 MHz bandwidth channel. This wideband transmission permits the multipath signals to be separately resolved and their presence compensated for by equalisation in the receiver. Another system, HYPERLAN, operates using a 5.2 GHz carrier frequency with a 20 Mbit/s BPSK transmission rate [Halls].

EXAMPLE 18.2

A wireless network is to be designed for interfacing several portable, battery powered, laptop computers dispersed around an office environment using the HYPERLAN standard. Due to obstructions in the office path loss follows an inverse cube law with distance (rather than the free space inverse square law of Chapter 12). The portable computer power consumption is 15 W and

the wireless network should not *significantly* degrade the battery operating life. Estimate, using the link budget analysis of Chapter 12, the maximum operating range you might expect to achieve. Assume that the receiver noise figure is 10 dB, the (BPSK) signal requires a receiver CNR of 15 dB, a fade margin of 6 dB is necessary, the total implementation loss is 2 dB and the transmitter efficiency is 40%.

If the transmitter uses 10 to 15% of the battery power then the power available for the radio modem is 1.5 to 2.2 W. At 40% efficiency this gives approximately 0.75 W or 29 dBm of transmitted power.

The thermal noise floor in a 25 MHz bandwidth, from equation (12.8), is −100 dBm. The received carrier level must be larger than this by 10 dB (noise figure), 2 dB (implementation loss), 15 dB (receiver CNR requirement) plus 6 dB (fade margin). Thus, adding a total of 33 dB, the required received carrier level is $C = -67$ dBm.

The maximum tolerable path loss is thus +29 +67 = 96 dB $= 4.0 \times 10^9$. FSPL is defined in Chapter 12. Using an R^{-3} law, equation (12.71) becomes:

$$\text{FSPL} = \left(\frac{4\pi}{\lambda}\right)^2 R^3$$

For isotropic antennas $G_T = G_R = 0$ dB ≡ 1. Thus for an operating frequency f and a free space propagation velocity c, the transmission loss, P_T/C, equals the FSPL, i.e.:

$$\frac{P_T}{C} = \text{FSPL} = \left(\frac{4\pi f}{c}\right)^2 R^3$$

Substituting numerical values and solving to find R:

$$4.0 \times 10^9 = \left(\frac{4\pi \times 5.2 \times 10^9}{300 \times 10^6}\right)^2 R^3 = \frac{4.74}{10^{-4}} R^3$$

$$R^3 = \frac{4.00}{4.74} 10^5 \text{ and } R = 44 \text{ m}$$

This range is typical for an office environment.

18.7.6 Internet

The Internet started life over 25 years ago as the ARPANET, section 18.1, which grew, initially, to link many university and industry laboratories in the US defence research community. It then became linked to Europe and gradually spread beyond the defence community, particularly as people realised that electronic mail could assist groups in many different organisations and countries to communicate effectively. The Internet is a communications medium, and a common set of protocols, which have been unified to form a coherent network.

Two separate developments ensured its rapid and widespread application. Firstly, it was demonstrated how a word in one document, located in one computer connected to the Internet, could be linked electronically to a previously unrelated document (a photograph, for example) stored on another computer, often in a different country. Many millions of

these *hypertext* links
range of different kir
which uses a globa
collection of inform
information, such a
'Hubble' would reac
colour images origi
analysing the data i
the cometary impact

The Internet is a
statistics about it ar
more than 2 millio
around the world.
individual owns or
rapid evolution anc
individuals and org
traffic on the Intern

At its current r
Internet by 2003.
commercial use of
and maintenance o
sector. In 1994 th
through which com
and services. The
the Internet. The
provide services fc

18.8 Summa

LANs, MANs a
individual buildir
networks may be
packet switching
former requiring
the latter allowir
network operatio
message switchin

Network top
systems. Transm
Microwave and
network terminal

Bridges are u
LANs). Gatewa

these *hypertext* links have now been put in place to better organise, and access, a wide range of different kinds of information. This has resulted in the world wide web (WWW) which uses a global web of linkages between an already huge, but ever growing, collection of information items and databases. Secondly, individuals started sharing information, such as pictures, on the Internet. For example, selecting the key word 'Hubble' would readily access not just data about the space telescope, but also the latest colour images originating directly from those computers controling the telescope and analysing the data it generates. Thus, many individuals first saw the Hubble images of the cometary impact on Jupiter via the Internet.

The Internet is a network of networks, hence the name – *inter-net*work network, and statistics about it are legion. In early 1995 it comprised around 50,000 networks, linking more than 2 million computers and perhaps 10 to 15 million users in many countries around the world. (It is not accurately known what the numbers are, because no single individual owns or controls the Internet.) It is self-regulating, capable of a high degree of rapid evolution and growth, and has been operated up to now by loose federations of individuals and organisations. What is really startling is the 20% per month growth of traffic on the Internet.

At its current rate of expansion, everyone on the planet would be connected to the Internet by 2003. A factor which is contributing to the growth of traffic is that commercial use of the Internet is now beginning to become important as the operation and maintenance of the underlying backbone networks in the US moves into the private sector. In 1994 there were over 21,000 commercial 'domains' registered on the Internet through which companies offer facilities to browse electronic catalogues and order goods and services. These commercial developments will change the fundamental character of the Internet. The Internet will evolve as this new technology matures and is used to provide services for home shopping, video entertainment, electronic banking, etc.

18.8 Summary

LANs, MANs and WANs refer to communications networks typically spanning individual buildings, individual towns, and individual countries, respectively. Such networks may be circuit switched, message switched or packet switched. Two forms of packet switching can be distinguished, i.e. virtual circuit and datagram switching – the former requiring all packets in a given message to traverse the same network route and the latter allowing packets to traverse independent routes. Historically the trend in network operation is from circuit switching (typified by the traditional PSTN), through message switching and virtual circuit switching, to datagram switching.

Network topologies include point-to-point, multidrop, star, ring, mesh and bus systems. Transmission media include twisted wire pairs, coaxial cable and optical fibres. Microwave and infrared links are also used to provide wireless connections between network terminals and nodes.

Bridges are used to interconnect networks using the same protocols (e.g. two similar LANs). Gateways are devices used to interconnect networks using different protocols

(e.g. a LAN and a WAN). Switches with self-routing properties can be employed at network nodes to improve switching speed over switches which require centralised control. Packet collisions at switch inputs can be avoided using an input sorting network and collisions at switch outputs avoided using a trap network.

The ISO OSI model for network protocol architectures has seven layers. The lowest three layers (physical, data link and network) specify the operation of the communications sub-net. The upper four layers are concerned with terminal equipment/network compatibility, initiation and control of a communication session, data formatting for correct presentation and the particular application being run by the user.

Medium access control (MAC) is one function of the data link layer in the OSI model. MAC protocols govern the allocation of physical medium resources (time, bandwith, orthogonal codes) between network terminal equipments. These resources can be allocated on a fixed, demand or random assignment basis. Fixed assignment protocols are efficient if data terminal equipments (DTEs) have a heavy, and relatively constant, traffic load. Random access protocols are more efficient if traffic loads are highly variable and uncorrelated between terminals. Random access protocols suffer, however, from potential packet collisions if two or more DTEs attempt to access the medium simultaneously. Such contention can be resolved using CSMA/CD protocols. Fixed assignment systems may be adaptive in that the proportions of medium resource allocated to different terminals may track long term variations in terminal traffic loads. Systems may also be adaptive in the sense that an essentially fixed assignment protocol may be substituted for a (normally) random access protocol if collision detection occurs too frequently. Demand assignment systems allow terminals to commandeer medium resources as and when they are required. They may operate using centralised control, in which terminals are polled to ascertain their need for resources, or distributed control, in which token passing is employed. Bus systems typically use CSMA/CD protocols whilst ring systems typically use token passing.

X.25 is the ITU-T standard defining the interface between DTEs and data communication equipment (DCEs) for packet switched networks. The DCE is the interface with a node of the data network and may be located at the same site as the network node or be remote from it (as in the case of a modem used to connect a computer to the packet switched public data network via an analogue local loop of the PSTN). Other X-series standards specify the protocols for connecting low speed character mode terminals to packet networks using packet assemblers and disassemblers (PADs).

FDDI is a 100 Mbit/s, optical fibre, backbone ring network which supports lower speed tree networks at its nodes. Its maximum circumference is 200 km. Wireless networks, using radio transmission between nodes, are susceptible to frequency selective fading due to multipath propagation. Spread spectrum techniques can be used to mitigate the resulting distortion, allowing transmission at data rates many times that which the channel would ordinarily support.

The Internet is, at present, the ultimate WAN in that it gives, potentially, global coverage. In addition to the conventional data communication uses of a WAN, hypertext connections between different documents held on computers connected to the Internet result in the unparalleled information resource called the World Wide Web.

CHAPTER 19

Public i
integra
networ

19.1 Introduc

This chapter exam
applied within the
services digital n
plesiochronous, ar
digital hierarchy (
payload capacity d

As much of th
capabilities of f
amplification, are
chapter concludes
in the local loop
basic, rate ISDN a

19.2 The tele

The internationall
multiplexing (Fig
access level at 1
individual 64 kbi
USA the first lev
stream.) Multipl
for the duration o
voice communica
as they require
terrestial telepho
satellite circuits t

CHAPTER 19

Public networks and the integrated services digital network (ISDN)

19.1 Introduction

This chapter examines how many of the techniques discussed earlier in the book are applied within the international public communications network. It covers the integrated services digital network (ISDN) [Griffiths], and includes discussion of the older plesiochronous, and newer synchronous, multiplexing techniques. The synchronous digital hierarchy (SDH) [Omidyar and Aldridge] is described, its frame structure and payload capacity defined, and its advantages outlined.

As much of this network now employs wideband fibre optic links, the principles and capabilities of fibre transmission, and optical pulse generation, reception and amplification, are summarised and a typical optical link budget presented. Finally this chapter concludes with a brief account of accessing schemes and on-going developments in the local loop network for digitised speech and data connections using primary, and basic, rate ISDN access.

19.2 The telephone network

The internationally agreed European ITU-T standard for PCM, TDM digital telephony multiplexing (Figure 5.27), is shown in Figure 19.1. Although this shows the basic access level at 144 kbit/s, the multiplexing hierarchy provides for the combining of 32 individual 64 kbit/s channels (Chapter 6) into a composite 2.048 Mbit/s signal. (In the USA the first level in the multiplex combines only 24 channels into a 1.544 Mbit/s data stream.) Multiplexing allocates a complete communications channel to each active user for the duration of his call or connection. In principle, as the channel utilisation factor for voice communications is low, we could concentrate the traffic by switching between users as they require transmission capacity. This resource saving strategy is not used in terrestial telephony but digital speech interpolation (DSI) is employed on international satellite circuits to achieve a significant increase in capacity, (see section 14.3.7).

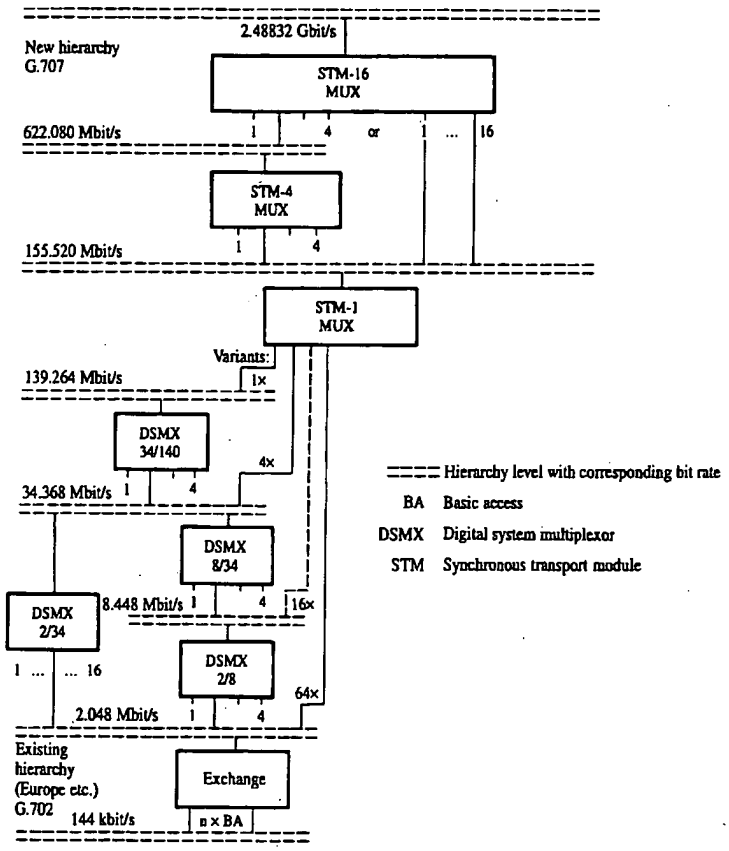


Figure 19.1 *ISDN access to European PCM TDM hierarchy with SDH at the upper levels.*

The ITU-T provides for higher levels of multiplexing, above 2.048 Mbit/s, combining four signals, in the digital system multiplexers (DSMX) 2/8 and 8/34, Figure 19.1, to form the signal at the higher multiplexing level. At each level the bit rate increases by slightly more than a factor of 4 since extra bits are added to provide for frame alignment and to facilitate satisfactory demultiplexing (see section 19.3). The upper levels, beyond 140 Mbit/s, form the synchronous digital hierarchy (SDH), and are only in limited use at present. These will be described later.

The extent of the current digital UK telephone network is shown in Figure 19.2. This is divided into an outer core of main processor exchanges and an inner core of about 55 trunk exchanges (digital main switching centres) which are fully interconnected on 140 Mbit/s, or higher bit rate, transmission links. These are now mainly optical links but also

include coaxial, and
fibre links, describe
on microwave radi
access in Figure 19.
Each subscriber
building, Figure 19.

Figure 19.2 *ISDI*
199

include coaxial, and terrestial microwave relay, paths. On 140 Mbit/s, 1.3 μm, optical fibre links, described later in section 19.5, the repeater spacing is typically 20 km while on microwave radio links, Chapter 14, the spacing is closer to 50 km. International access in Figure 19.2 is provided via satellite links or fibre-optic undersea cables.

Each subscriber telephone has two copper wires that go directly to the local exchange building, Figure 19.3. (The distance is typically 1 to 10 km, being smaller in cities than

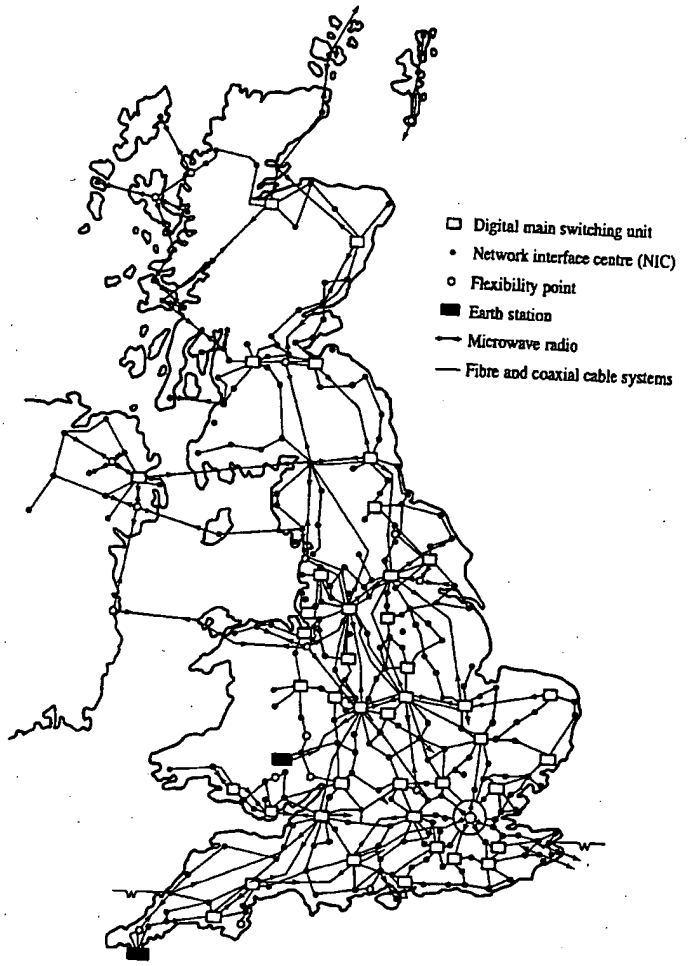


Figure 19.2 *ISDN locations and UK digital trunk telecommunications network (source: Leakey, 1991, reproduced with the permission of British Telecommunications plc.).*

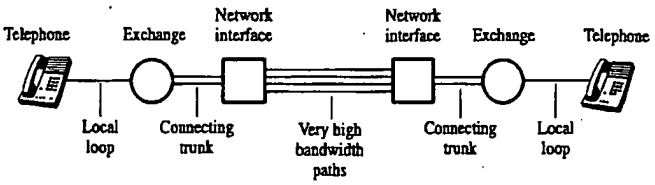


Figure 19.3 *Trunk telecommunications system used for establishing a connection.*

in rural areas where one exchange normally serves 40 km^2 or 5,000 to 50,000 customers.) The two-wire access connection between each subscriber's telephone and the exchange is known as the *local loop*. Each exchange has a number of outgoing lines to one or more nearby switching centres which provide the network interface. Figure 19.3 shows the typical connection for a long distance (trunk) call involving both inner and outer core connections. In this system there are four wires to the microphone and speaker in the handset, two wires in the local loop and four wires again in the national trunk network. (A four wire circuit implies separate transmit and receive channels as in cellular radio systems, Chapter 15.) In the UK there are approximately 6300 local exchange buildings and the local access network comprises 36 million metallic pair cables.

Traditional control signalling in circuit-switched telephone networks for establishing or initiating call connections may either use a separate communications channel or operate on an in-channel basis. With in-channel signalling, the same channel is used to carry control signals as is used to carry message traffic. Such signalling begins at the originating subscriber and follows the same path as the call itself. This has the merit that no additional transmission facilities are needed for signalling. Two forms of in-channel signalling are in use, in-band and out-of-band. In-band signalling uses not only the same physical path as the call it serves, it also uses the same frequency band as the voice signals that are carried. Out-of-band signalling takes advantage of the fact that voice signals do not use the full 4 kHz of available bandwidth. A separate narrow signalling band, within the 4 kHz, is used to send control signals. A drawback of in-channel signalling schemes is the relatively long delay from the time that a subscriber dials a number to the connection being made.

This problem is addressed by common channel signalling in which control signals are carried by an independent signalling network. Since the control signal bandwidth is small, one separate control signal path can carry the signals for a number of subscriber channels. The common channel uses a signalling protocol, and requires the network architecture to support that protocol, which is more complex than the in-channel signalling case. The control signals are messages that are passed between switches and between a switch and the network management centre.

Over the past decade several different general purpose signalling systems have been developed by ITU and other standards organisations. The most important of these, and the one of major relevance to the ISDN, is the set of procedures known as signalling system No. 7, which is structured in accordance with the ISO OSI model of Figure 18.12. The overall objective is to provide an internationally standardised, general purpose,

common channel si
- is optimised for
stored program
- is designed to
control, remote
- provides a reli
sequence witho
- is suitable for u

19.3 Plesioch

The 2.048 Mbit/s
Figures 19.1 and
Figure 19.5. Time
slot 16 is used for
channels are used
one 8-bit voice sig

The system fo
multiplexers (DSI
implies separate f
the name plesioch
than the incoming
allows for small e
requires some ext
speed oscillator.
sufficient bits ar
required because

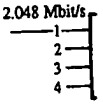


Figure 19.4 *DSI*

common channel signalling system which:

- is optimised for use in digital telecommunication networks in conjunction with digital stored program control exchanges utilising 64 kbit/s digital signals;
- is designed to meet present and future information transfer requirements for call control, remote network management, and maintenance;
- provides a reliable means for the transfer of information packets in the correct sequence without loss or duplication;
- is suitable for use on point-to-point terrestrial and satellite links.

19.3 Plesiochronous multiplex

The 2.048 Mbit/s multiplex level (also called the PCM primary multiplex group) in Figures 19.1 and 19.4 comprises frames with 32 8-bit time slots within each frame, Figure 19.5. Time slot zero is reserved for frame alignment and service bits, and time slot 16 is used for multiframe alignment, service bits and signalling. The remaining 30 channels are used for information carrying, or payload capacity, each channel containing one 8-bit voice signal sample.

The system for assembling the TDM telephony data stream assumes that the digital multiplexers (DSMX) in Figure 19.4 are located at physically separate sites which implies separate free running oscillators at each stage in the multiplex hierarchy, hence the name plesiochronous. These oscillators must therefore run at speeds slightly higher than the incoming data, Figure 19.4, to permit local variations to be accommodated. This allows for small errors in the exact data rates in each of the input paths or tributaries but requires some extra bits to be added (i.e. stuffed or justified) to take account of the higher speed oscillator. Elastic stores, Figure 19.6, are used in a typical multiplexer to ensure sufficient bits are always available for transmission or reception. These stores are required because the plesiochronous digital hierarchy (PDH) works by interleaving bytes

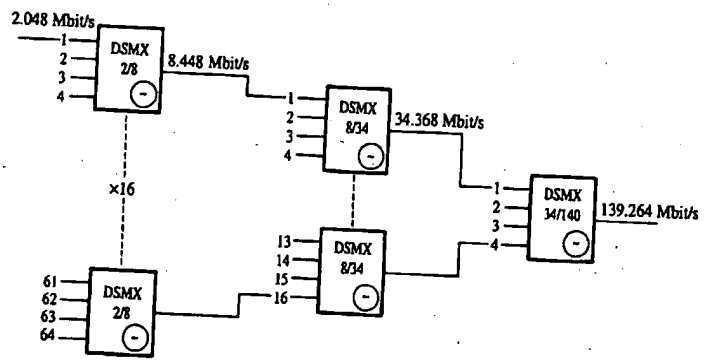


Figure 19.4 *DSMX interconnection to form a plesiochronous multiplex hierarchy.*

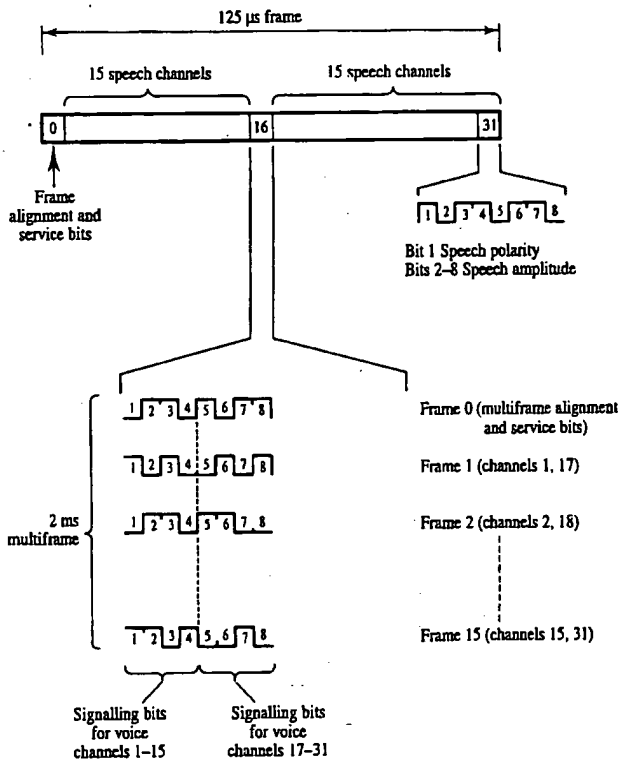


Figure 19.5 *PCM primary multiplex group frame structure.*

or words from each 64 kbit/s tributary, rather than bit interleaving, to form the 2 Mbit/s multiplex. Thus, at the 8 Mbit/s multiplexing level, and above, where bit interleaving is employed, bits must be accumulated for high speed readout.

Figure 19.7 shows more detail of the multiplexer hardware. The code translators in Figure 19.6 convert binary data from, and to, HDB3 (see section 6.4.5). Figure 19.8 shows details of the plesiochronous frame structure at 8 Mbit/s. (Note the bit interleaving, shown explicitly for the justification bits, used at multiplexing levels above 2 Mbit/s.) The ITU-T G series of recommendations (G.702) defines the complete plesiochronous multiplex hierarchy. The frame alignment signal, which is a unique word recognised in the receiver, ensures that the appropriate input tributary is connected to the correct output port. The unique word also permits receiver recovery from loss of synchronisation, if it occurs. The plesiochronous multiplex system was developed at a time when transmission costs were low and switching costs were high. With recent advances in VLSI this premise is now no longer valid, hence the movement to new standards.

Figure 19.6 2/8

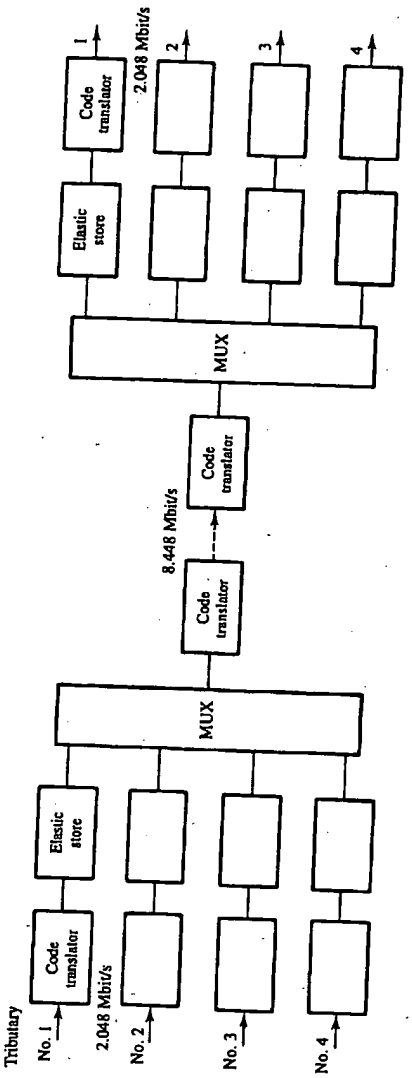


Figure 19.6 *2/8 Multiplexer block diagram.*

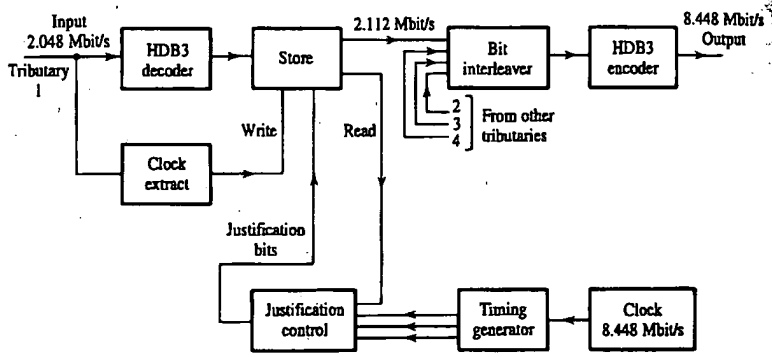


Figure 19.7 *2/8 multiplexer timing details.*

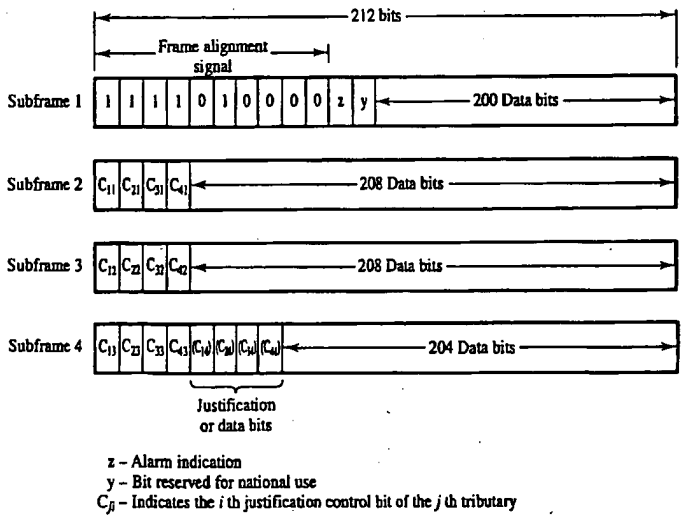


Figure 19.8 *Plesiochronous frame structure.*

A disadvantage of the plesiochronous multiplex is that it was designed for point-to-point transmission applications in which the entire multiplex would be decoded at each end. This is a complicated process since it requires full demultiplexing at each level to recover the bit interleaved data and remove the justification bits. Thus a single 2 Mbit/s channel, for example, cannot be extracted from, or added to, a higher level multiplex signal without demultiplexing down, and then remultiplexing up, through the entire PDH, Figure 19.9. The plesiochronous multiplex does *not* support definitive or clear identification of the various individual channels which are being carried.

Figure 19.9 *i*

Many hi
plesiochrono
limited, how
It does not p
respond to t
since netwo
availability c
plesiochronc
maintenance
of a standar
different pa
whose netw
their networ
The PDI
wideband o
SONET an
following s
provide the
to connect
away from
single 64-c
an input to

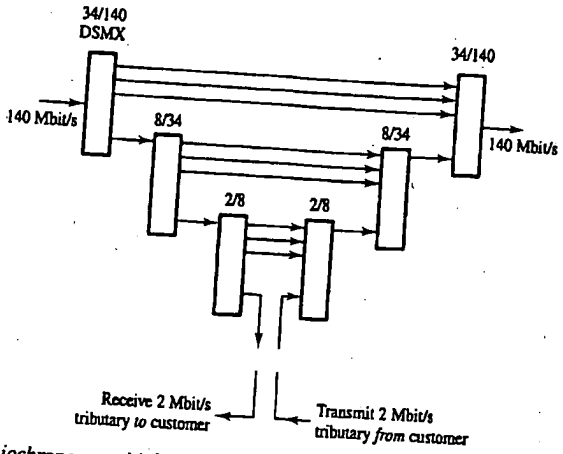


Figure 19.9 *Plesiochronous multiplex add-drop scheme for inserting, and removing, a 2 Mbit/s tributary to, and from, a 140 Mbit/s stream.*

Many high capacity transmission networks are provided by a hierarchy of digital plesiochronous signals. The plesiochronous approach to signal multiplexing is severely limited, however, in its ability to meet the foreseeable requirements of network operators. It does not provide, cost-effectively, the flexible network architecture which is required to respond to the demands of today's evolving telecommunications market. Furthermore, since network management and maintenance strategies were based, historically, on the availability of a manual distribution frame, there was no need to add extra capacity to the plesiochronous frame structure in order to support network operations, administration, maintenance and provisioning (OAM&P) activities. Other PDH drawbacks are the lack of a standard above 140 Mbit/s and the use of different plesiochronous hierarchies in different parts of the world. This leads to problems of interworking between countries whose networks are based on 1.544 Mbit/s (e.g. Japan, North America) and those basing their networks on 2.048 Mbit/s (e.g. Europe, Australia).

The PDH limitations described above, and the desire to move from metallic cable to wideband optical fibres, have been important motivations in the development of the new SONET and synchronous digital hierarchy (SDH) systems which are described in the following sections. These new systems offer improved flexibility and can more readily provide the 2 Mbit/s leased lines which, for example, cellular telephone operators require to connect their base station transmitters to switching centres. (Just prior to the move away from PDH towards SDH systems British Telecom (BT) did develop, in the 1980s, a single 64-channel 2 to 140 Mbit/s multiplexer. This has been included in Figure 19.1 as an input to the new 155.52 Mbit/s standard multiplexer rate.)

EXAMPLE 19.1
Find the number of standard PCM voice signals which can be carried by a PDH level 4 multiplex and estimate the maximum channel utilisation efficiency at this multiplexing level.

PDH Level 1 (primary multiplex group) carries 30 voice signals. Each subsequent level combines four tributaries from previous levels. At level n the number of potential voice signals is therefore given by:

$$\chi = 30 \times 4^{n-1}$$

At level four:

$$\chi = 30 \times 4^{4-1} = 1920 \text{ voice signals}$$

Nominal bit rate at PDH level 4 is 140 Mbit/s. Therefore channel utilisation efficiency is:

$$\eta_{ch} = \frac{\chi}{R_b/R_v} = \frac{1920}{(140 \times 10^6)/(64 \times 10^3)} = 88\%$$

19.4 SONET and SDH

The concept of the digital SONET (synchronous optical network) was initially introduced in the USA in 1986 to establish wideband transmission standards so that international operators could interface using standard frame formats and signalling protocols. The concept also included network flexibility and intelligence, and overhead channels to carry control and performance information between network elements (line systems, multiplexers) and control centres.

In 1988 these concepts were adopted by ITU and ETSI (European Telecommunications Standards Institute) and renamed synchronous digital hierarchy (SDH) with the aim of agreeing worldwide standards for transmission covering optical interfaces, control aspects, equipment, signalling, etc. [Miki and Siller]. Bit transport rates start as low as 52 Mbit/s and go up through 155 Mbit/s with a hierarchy to 622 Mbit/s, 2488 Mbit/s and beyond. The ITU-T G.707/8/9 standards have now reached a mature stage allowing manufacturers to produce common hardware.

19.4.1 Advantages and flexibility

The key advantages of SDH are as follows:

- it is cheaper to add and drop signals to meet customer requirements.
- more bandwidth is available for network management.
- equipment is smaller and cheaper.
- worldwide standards allow a larger manufacturers' marketplace.

- it is easier to int
- it is cheaper t
  between transm

Network flexibi
centre in order to:

- improve capac
  transported in t
- improve availa
  protection sche
- reduce mainten
- provide easier
  upgraded.

Flexibility can
systems or betwee
gradually replace
capacity within th
ability to extract o
high order signal, a

19.4.2 Synchro

The synchronous s
a frame structure s
to a framing (or m
used here is one i
rows and columns
the masterframe st

The signal bit
video signal, start
byte in row 1, an
Then the bits in th
of row 2, and so o

This transmis
duration of each f
the frame is 8 × 8
to one PCM voi
structure may be
The fixed sequen
streams within th

- it is easier to introduce new services.
- it is cheaper to achieve remote digital access to services and cross-connections between transmission systems.

Network flexibility implies the ability to rapidly reconfigure networks from a control centre in order to:

- improve capacity utilisation by maximising the number of 2 Mbit/s channels transported in the higher order system;
- improve availability of digital paths by centrally allocating spare capacity and protection schemes to meet service requirements;
- reduce maintenance costs by diverting traffic away from failed network elements;
- provide easier growth with temporary diversion of traffic around areas being upgraded.

Flexibility can be achieved using automatic cross-connect switches between SDH systems or between SDH and plesiochronous systems. Automatic cross-connects will gradually replace existing manual cross-connects and allow remote reconfiguration of capacity within the network at 2 Mbit/s and above. Add-drop multiplexing refers to the ability to extract or insert individual channels without the need to demultiplex the entire high order signal, as required in the plesiochronous system.

19.4.2 Synchronous signal structure

The synchronous signal comprises a set of 8-bit bytes which are carefully interleaved into a frame structure such that the identity of each byte is preserved and known with respect to a framing (or marker) word. The description of the synchronous signal frame structure used here is one in which the bytes of the signal are represented by boxes appearing in rows and columns on a 2-dimensional map, Figure 19.10 [Hawker]. (This is not unlike the masterframe structure of Figure 14.43.)

The signal bits are transmitted in a raster scanned sequence, similar to the lines on a video signal, starting with those in the top left hand byte, followed by those in the 2nd byte in row 1, and so on, until the bits in the Mth (last) byte in row 1 are transmitted. Then the bits in the 1st byte of row 2 are transmitted, followed by the bits in the 2nd byte of row 2, and so on, until the bits in the Mth byte of the Nth (last) row are transmitted.

This transmission sequence repeats, the repetition rate being 8000 frame/s. The duration of each frame is, therefore, 125 μs and the bit rate associated with each byte in the frame is 8×8 kbit/s = 64 kbit/s. Each byte in the frame is thus equivalent in capacity to one PCM voice channel. One or more 8-bit bytes within the synchronous signal structure may be allocated to provide channel capacity for a lower rate (tributary) signal. The fixed sequence frame alignment word allows the positions of all individual data streams within the frame to be identified.

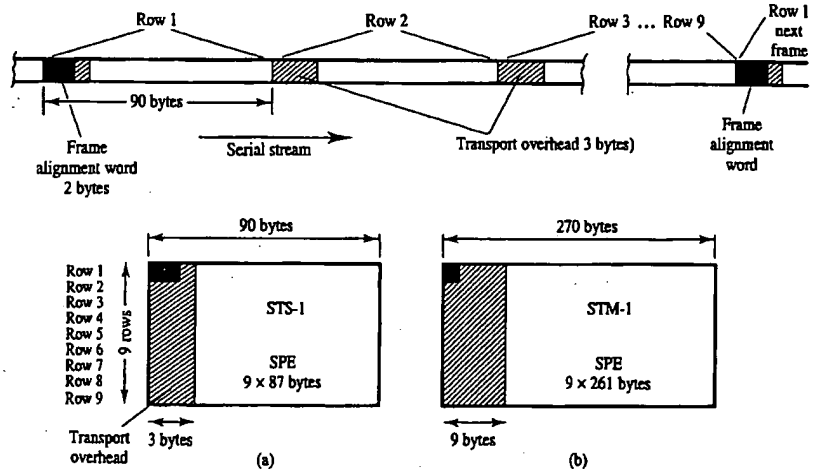


Figure 19.10 *Frame structures showing serial bit stream and the equivalent two dimensional data maps: (a) basic SONET STS-1 frame; and (b) SDH STM-1 frame.*

19.4.3 Frame structure

The lowest level SONET signal is called the synchronous transport signal level 1 (STS-1). It consists of 90 columns and 9 rows of 8-bit bytes giving a total 810 bytes (6480 bits), Figure 19.10(a). With a frame duration of 125 μs, the STS-1 bit rate is 51.84 Mbit/s. The basic SDH frame corresponds to three SONET STS-1 frames and is called a level 1 synchronous transport module (STM-1), Figure 19.10(b). The STM-1 bit rate is therefore 155.52 Mbit/s. Each STS-1 frame can be seen to comprise two parts – an overhead part and a service payload part.

Transport overhead: The first three columns of the STS-1 frame, a total of 27 bytes, Figure 19.10(a), are allocated to overheads that provide operations and maintenance (O&M) facilities. The three bytes in rows 1, 2 and 3 comprise the section overhead while the remaining three bytes in rows 4 to 9 (18 bytes) comprise the line overhead. The section and line terminology is defined in Figure 19.11 for a communications link. (Line terminating equipment might be represented, for example, by an add-drop multiplexer.)

Synchronous payload envelope (SPE): The remaining 87 columns of the STS-1 frame, a total of 783 bytes, provide a channel capacity of 50.112 Mbit/s which supports the transport of the service payload, or traffic data, and also the path overhead, Figure 19.12.

Path overhead: The path overhead supports and maintains the transportation of the SPE between the locations where the SPE is assembled and disassembled. It comprises a total of 9 bytes, Figure 19.12, and is allocated the first column (one byte wide) within the STS-1 SPE.

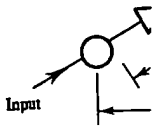


Figure 19.11 *Path, l*

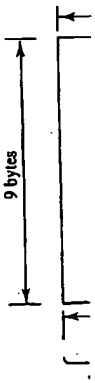


Figure 19.12 *STS-*

Payload capa
columns of 9 byte
with a frame repet
The SONET/S
cross-connect fun

- a modular stru
- extensive over
- byte interleavi
- byte stuffing t

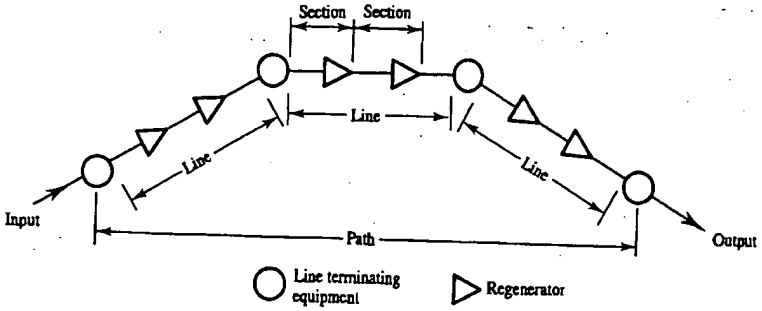


Figure 19.11 *Path, line and section details in a communications link.*

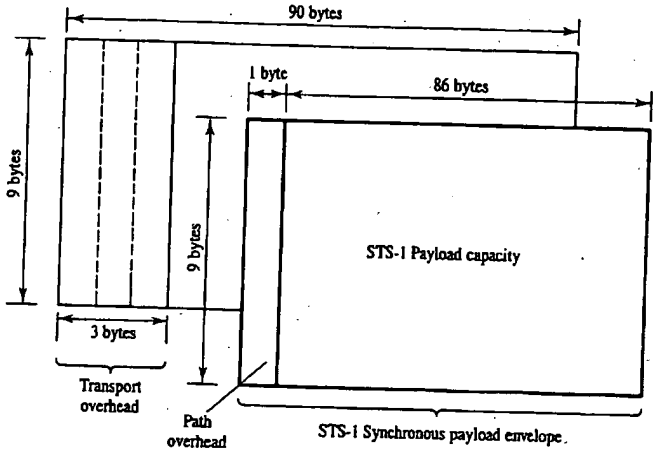


Figure 19.12 *STS-1 synchronous payload envelope (SPE).*

Payload capacity: The STS-1 information capacity comprises the remaining 86 columns of 9 bytes, i.e. a total of 774 bytes providing a 49.536 Mbit/s channel capacity with a frame repetition rate of 8 kHz.

The SONET/SDH frame structure features greatly simplified 'drop and insert' and cross-connect functions by utilising:

- a modular structure made up from SONET/SDH tributaries;
- extensive overheads for monitoring and control;
- byte interleaving with direct visibility of 64 kbit/s channels;
- byte stuffing to improve robustness against loss of synchronisation;

- standard mappings for all common data rates into the SONET/SDH frame format.

The STS-1 SPE represents the unshaded part of Figure 19.10(a). Additional transport capacity is obtained by effectively concatenating SPEs, Figure 19.10(b). Alternatively a reduction in transport capacity may be obtained by partitioning the SPE into smaller segments, called virtual tributaries or VTs (see section 19.4.5).

19.4.4 Payload pointer

To facilitate efficient multiplexing and cross-connection of signals in the synchronous network, the SPE is allowed to 'float' within the payload capacity provided by the STS-1 frames, Figure 19.13. This means that the STS-1 SPE may begin anywhere in the STS-1 frame and is unlikely to be wholly contained in one frame. More likely than not, the STS-1 SPE will begin in one frame and end in the next.

The STS-1 payload pointer, contained in the transport overhead, indicates the location of the first byte of the STS-1 SPE. Byte 1 of the STS-1 SPE is also the first byte of the SPE path overhead. It permits non-synchronous data to be accommodated within the SDH structure, without resorting to justification or bit stuffing.

For synchronous transport, payload pointers provide a means of allowing an SPE to be transferred between network nodes operating plesiochronously. Since the SPE floats freely within the transport frame, with the payload pointer value indicating the location of the first active byte of the SPE, the problem in justified plesiochronous multiplex where the traffic is mixed with stuffing bits is overcome.

Payload pointer processing does, however, introduce a new signal impairment known as 'tributary jitter'. This appears on a received tributary signal after recovery from a synchronous payload envelope which has been subjected to payload pointer movements from frame-to-frame. Excessive tributary jitter will influence the operation of the downstream plesiochronous network equipment processing the tributary signal.

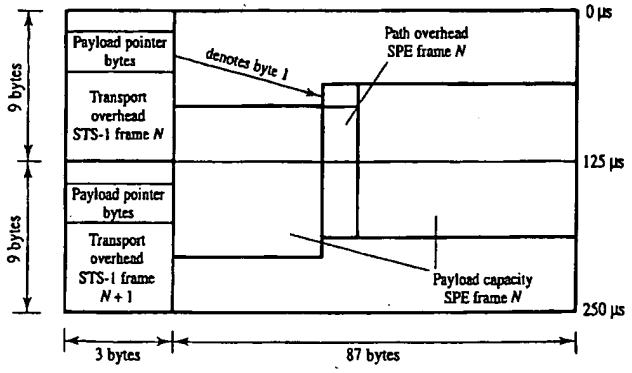


Figure 19.13 *Payload pointer details for locating the start of the SPE.*

19.4.5 Payload ca

The virtual tributary
been designed to su
than that provided b
Figure 19.14. These

- VT1.5, consistir
  8 kHz, provides
  1.544 Mbit/s DS
- VT2, consisting
  capacity of 2.30
- VT3, consisting
  capacity of 3.45

These and othe
demand to handle d
plesiochronous mul
be easily written to.

A 155.52 Mbit/
SPEs into one STM
interleaving $N \times 1$
synchronous transp
format allows simu
(e.g. 45/140 Mbit/s

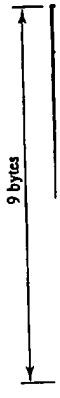


Figure 19.14 *Virtu*

19.4.5 Payload capacity partitioning

The virtual tributary (VT) structure (or tributary unit structure in SDH terminology) has been designed to support the transport and switching of payload capacity which is less than that provided by the full STS-1 SPE. There are three sizes of VTs in common use, Figure 19.14. These are:

- VT1.5, consisting of 27 bytes, structured as 3 columns of 9, which, at a frame rate of 8 kHz, provides a transport capacity of 1.728 Mbit/s and will accommodate a US 1.544 Mbit/s DS1 signal.
- VT2, consisting of 36 bytes, structured as 4 columns of 9, which provides a transport capacity of 2.304 Mbit/s and will accommodate a European 2.048 Mbit/s signal.
- VT3, consisting of 54 bytes, structured as 6 columns of 9, to achieve a transport capacity of 3.456 Mbit/s which will accommodate a US DS1C signal.

These and other VTs allow the SDH structure to be electronically reconfigured on demand to handle different customer requirements. It circumvents the fixed nature of the plesiochronous multiplex and, as the VTs are distributed over the entire payload, they can be easily written to, and read from, the system without requiring large data buffer stores.

A 155.52 Mbit/s SDH transmission capability is obtained by combining three STS-1 SPEs into one STM-1 SPE, Figure 19.10(b). Higher order systems are formed by byte interleaving $N \times 155$ Mbit/s channels (e.g. 16×155.5 Mbit/s = 2488 Mbit/s) to form the synchronous transport module level N (e.g. STM-16) rate, Figure 19.1. The SDH frame format allows simultaneous transport of both narrowband (e.g. 2 Mbit/s) and broadband (e.g. 45/140 Mbit/s) services within the 155 Mbit/s capacity system.

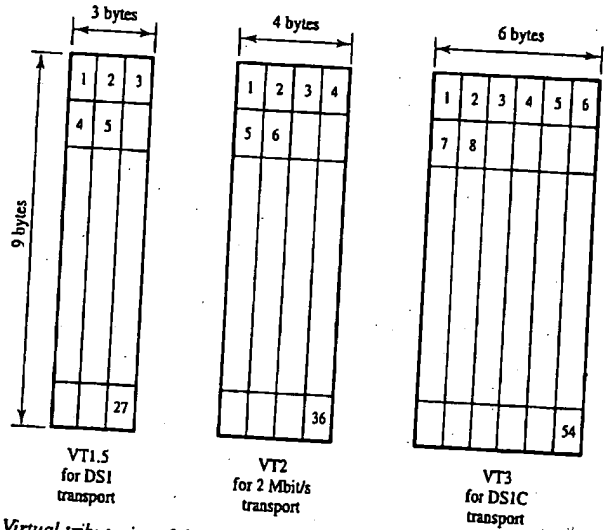


Figure 19.14 *Virtual tributaries of the STS-1 frame.*

EXAMPLE 19.2

Estimate the number of voice channels which can be accommodated by an SDH STM-4 signal assuming that the STM-4 is filled with ITU primary multiplex group signals. Also estimate the channel utilisation efficiency.

Each STS-1 payload envelope has 86 columns (not including the path overhead column). Each primary multiplex group occupies 4 columns. Each STS-1 payload envelope can therefore transport 86/4 = 21 (whole) primary multiplexes.

Each STM-1 payload envelope corresponds to 3 STS-1 payload envelopes and therefore carries $3 \times 21 = 63$ primary multiplexes. STM-4 carries 4 STM-1 signals and therefore carries $4 \times 63 = 252$ primary multiplexes. Each primary multiplex carries 30 voice channels, Figure 19.5. The STM-4 signal can, therefore, carry $30 \times 252 = 7560$ voice channels.

Channel utilisation efficiency, η_{ch}, is thus given by:

$$\eta_{ch} = \frac{7560}{R_b/R_v} = \frac{7560}{(622.08 \times 10^6)/(64 \times 10^3)}$$

$$= \frac{7560}{9720} = 78\%$$

19.5 Fibre optic transmission links

Optical fibres comprise a core, cladding and protective cover and are much lighter than metallic cables [Gowar]. This advantage, coupled with the rapid reduction in propagation loss to its current value of 0.2 dB/km or less, Figure 1.6(e), and the enormous potential bandwidth available, make optical fibre now the only serious contender for the majority of long-haul trunk transmission links. The potential capacity of optical fibres is such that all the radar, navigation and communication signals in the microwave and millimetre wave region, which now exist as free space signals, could be accommodated within 1% of the potential operational bandwidth of a single fibre. Current commercial systems can accommodate 31,000 simultaneous telephone calls in a single fibre and 1M call capacity has been achieved in the laboratory.

19.5.1 Fibre types

In the fibre a circular core of refractive index n_1 is surrounded by a cladding layer of refractive index n_2, where $n_2 < n_1$, Figure 19.15. This results in optical energy being guided along the core with minimal losses.

The size of the core and the nature of the refractive index change from n_1 to n_2 determine the three basic types of optical fibre [Gowar], namely:

- multimode step-index;
- multimode graded-index;

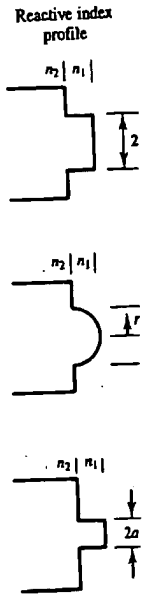


Figure 19.15 *Three*

- monomode ste

The refractive schematic ray dia are shown in Figu the refractive inde originally, a 20% In the more rec typically 0.5%. T had limited bandv of modern multin cladding, pulse br between the diffe 10 Mbit/s. Grade representing diffe propagation velo delay for all the approximately t monomode fibres dominant pulse

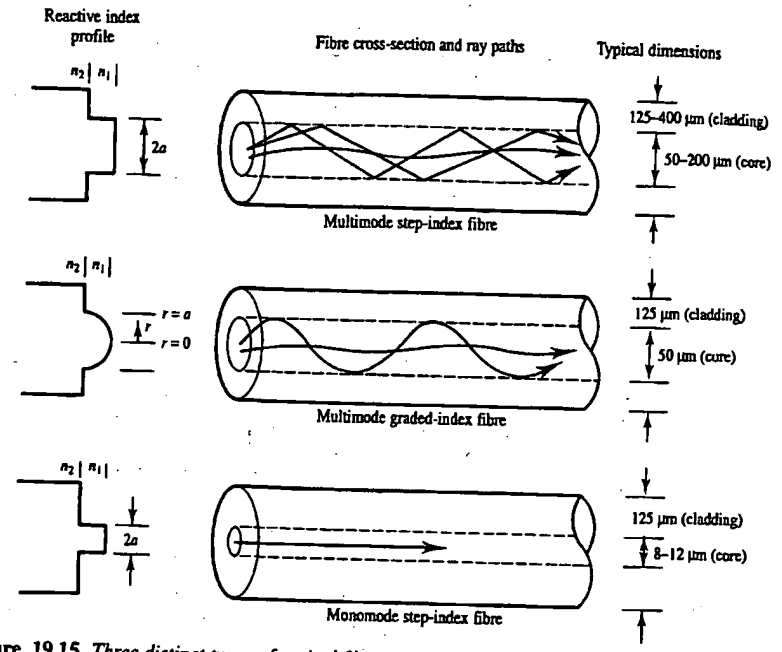


Figure 19.15 *Three distinct types of optical fibre.*

- monomode step-index.

The refractive index profiles, typical core and cladding layer diameters and a schematic ray diagram representing the distinct optical modes in these three fibre types are shown in Figure 19.15. The optical wave propagates down the fibre via reflections at the refractive index boundary or refraction in the core. In multimode fibres there was, originally, a 20% difference between the refractive indices of the core and the cladding. In the more recently developed monomode fibres, the difference is much smaller, typically 0.5%. The early multimode step index fibres were cheap to fabricate but they had limited bandwidth and a limited section length between repeaters. In a 1 km length of modern multimode fibre with a 1% difference in refractive index between core and cladding, pulse broadening or dispersion, caused by the difference in propagation velocity between the different electromagnetic modes, limits the maximum data rate to, typically, 10 Mbit/s. Graded index fibres suffer less from mode dispersion because the ray paths representing differing modes encounter material with differing refractive index. Since propagation velocity is higher in material with lower refractive index, the propagation delay for all the modes can, with careful design of the graded-n profile, be made approximately the same. For obvious reasons *modal* dispersion is absent from monomode fibres altogether and in these fibre types *material* dispersion is normally the dominant pulse spreading mechanism. Material dispersion occurs because refractive

index is, generally, a function of wavelength, and different frequency components therefore propagate with different velocities. (Material dispersion is exacerbated due to the fact that practical sources often emit light with a narrow, but not monochromatic, spectrum.) The rate of change of propagation velocity with frequency (dv/df), and therefore dispersion, in silica fibres changes sign at around 1.3 μm, Figure 19.16, resulting in zero material dispersion at this wavelength. (Fortuitously, this wavelength also corresponds to a local minimum in optical attenuation, see Figure 1.6(e).)

If both modal and material dispersion are zero, or very small, *waveguide* dispersion, which is generally the weakest of the dispersion mechanisms, may become significant. Waveguide dispersion arises because the velocity of a waveguide mode depends on the normalised dimensions (d/λ) of the waveguide supporting it. Since the different frequency components in the transmitted pulse have different wavelengths these components will travel at different velocities even though they exist as the same electromagnetic mode. Because both material and waveguide dispersion relate to changes in propagation velocity with wavelength they are sometimes referred to collectively as chromatic dispersion. The various fibre types are further defined in ITU-T recommendation G.652.

19.5.2 Fibre transmission systems

There have been three generations of optical fibre systems operating at 0.85 μm, 1.3 μm and 1.5 μm wavelengths to progressively exploit lower optical attenuation, Figure 1.6(e), and permit longer distances to be achieved between repeaters. Monomode fibres, with core diameters in the range 8 to 12 μm, have been designed at the two longer wavelengths for second and third generation systems. Since material dispersion is zero at around 1.3

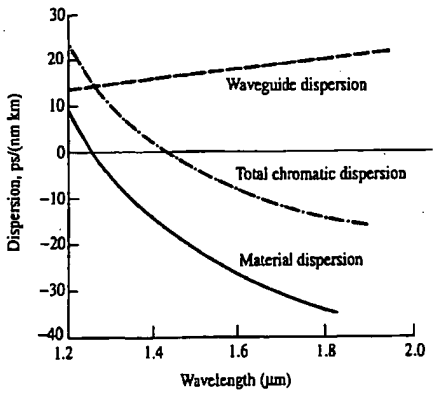


Figure 19.16 *Variation of material dispersion and waveguide dispersion, giving zero total dispersion near λ = 1.5 μm (source, Flood and Cochrane, 1989, reproduced with the permission of Peter Peregrinus).*

μm, where the op
wavelength chosen

Third generatic
Mbit/s to exploit t
resulting increased
Thus, the choice a
wants to maximise

Figure 19.16 sl
chosen correctly, h
at about 1.5 μm re
band.

The impact of
illustrates the evol
coaxial cable used
failures (MTBF)
plesiochronous mu
still needed a repe
breakthrough cam
1.5 μm, there is n
loss is typically 1
section 12.4.2.) E
80% of the UK t
transmission capa
capacity doubles e

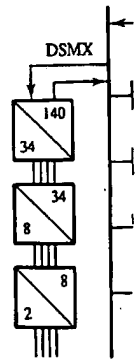


Figure 19.17 *Evo*
199

μm, where the optical attenuation in silica is also a local minimum, this was the wavelength chosen for second generation systems, which typically operate at 280 Mbit/s.

Third generation systems operate at wavelengths around 1.5 μm and bit rates of 622 Mbit/s to exploit the lowest optical attenuation value of 0.15 dB/km, and tolerate the resulting increased chromatic dispersion which in practice may be 15 to 20 ps $nm^{-1}km^{-1}$. Thus, the choice at present between 1.3 and 1.5 μm wavelength depends on whether one wants to maximise link repeater spacing or signalling bandwidth.

Figure 19.16 shows that if core dimensions, and core cladding refractive indices, are chosen correctly, however, material dispersion can be cancelled by waveguide dispersion at about 1.5 μm resulting in very low total chromatic dispersion in this lowest attenuation band.

The impact of fibre developments is clearly seen in Figures 1.7 and 19.17. The latter illustrates the evolution of transmission technology for a 100 km wideband link. The coaxial cable used in the 1970s with its associated 50 repeaters had a mean time between failures (MTBF) of 0.4 years, which was much lower than the 2 year MTBF of the plesiochronous multiplex equipment at each terminal station. Multi-mode fibre (MMF) still needed a repeater every 2 km but was a more reliable transmission medium. The real breakthrough came with single mode fibre (SMF) and, with the low optical attenuation at 1.5 μm, there is now no need for any repeaters on a 100 km link, in which the fibre path loss is typically 10 to 28 dB. (This is *very* much lower than the microwave systems of section 12.4.2.) By 1991 there were 1.5 million km of installed optical fibre carrying 80% of the UK telephone traffic. This UK investment represented 20% of the world transmission capability installed in optical fibre at that time. Optical fibre transmission capacity doubles each year with an exponentially reducing cost.

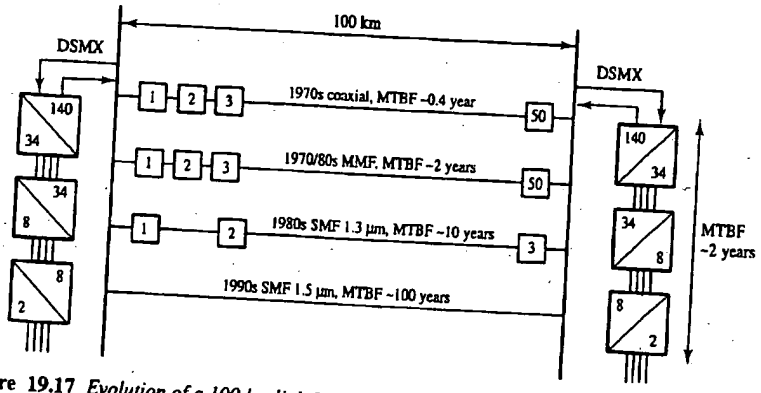


Figure 19.17 *Evolution of a 100 km link from coaxial to optical transmission (source: Cochrane, 1990, reproduced with the permission of British Telecommunications plc.).*

19.5.3 Optical sources

Two devices are commonly used to generate light for fibre optic communications systems: light-emitting diodes (LEDs) and injection laser diodes (ILDs). The edge emitting LED is a PN junction diode made from a semiconductor material such as aluminium–gallium–arsenide (AlGaAs) or gallium–arsenide–phosphide (GaAsP). The wavelength of light emitted is typically 0.94 μm and output power is approximately 3 mW at 100 mA of forward diode current. The primary disadvantage of this type of LED is the non-directionality of its light emission which makes it a poor choice as a light source for fibre optic systems. The planar hetero-junction LED generates a more brilliant light spot which is easier to couple into the fibre. It can also be switched at higher speeds to accommodate wider signal bandwidth.

The injection laser diode (ILD) is similar to the LED but, above the threshold current, an ILD oscillates and lasing occurs. The construction of the ILD is similar to that of an LED, except that the ends are highly polished. The mirror-like ends trap photons in the active region which, as they are reflected back and forth, stimulate free electrons to recombine with holes at a higher-than-normal energy level to achieve the lasing process. ILDs are particularly effective because the optical radiation is easy to couple into the fibre. Also the ILD is powerful, typically giving 10 dB more output power than the equivalent LED, thus permitting operation over longer distances. Finally, ILDs generate close to monochromatic light, which is especially desirable for single mode fibres operating at high bit rates.

19.5.4 Optical detectors

There are two devices that are commonly used to detect light energy in fibre optic systems: PIN (positive–intrinsic–negative) diodes and APDs (avalanche photodiodes). In the PIN diode, the most common device, light falls on the intrinsic material and photons are absorbed by electrons, generating electron-hole pairs which are swept out of the device by the applied electric field. The APD is a positive–intrinsic–positive–negative structure, which operates just below its avalanche breakdown voltage to achieve an internal gain. Consequently, APDs are more sensitive than PIN diodes, each photon typically producing 100 electrons, their outputs therefore requiring less additional amplification.

19.5.5 Optical amplifiers

In many optical systems it is necessary to amplify the light signal to compensate for fibre losses. Light can be detected, converted to an electrical signal and then amplified conventionally before remodulating the semiconductor source for the next stage of the communications link. Optical amplifiers, based on semiconductor or fibre elements employing both linear and non-linear devices, are much more attractive and reliable; they permit a range of optical signals (at different wavelengths) to be amplified simultaneously and are especially significant for sub-marine cable systems.

Basic travellir
the range 10 to
operation and are
with distributed f
common with all
results in an optic
cascaded TWSLA

Recent resear
or undoped fibre
mechanism or th
erbium doped fib
fibre pumped w
coupling of fibre-
30 to 50 nm optic
15 dB. The key
achievable and t
exceeds the 300
Two interesting
wavelength via
effects. They ca
and can operate e

Injecting a h
Introducing a lo
energy, results i
power, coupled
Figure 19.18.

Brillouin sca
realise high gain

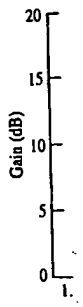


Figure 19.18 *Co*
19

Basic travelling wave semiconductor laser amplifier (TWSLA) gains are typically in the range 10 to 15 dB, Figure 19.18. These Fabry-Perot lasers are multimode in operation and are used in medium distance systems. Single mode operation is possible with distributed feedback (DFB) lasers for longer distance, high bit rate, systems. In common with all optical amplifiers, the TWSLA generates spontaneous emissions which results in an optical (noise) output in the absence of an input signal. For a system with cascaded TWSLAs these noise terms can accumulate.

Recent research has resulted in fibre amplifiers consisting of 10 m to 50 km of doped or undoped fibre. These amplifiers use either a linear, rare earth (erbium), doping mechanism or the non-linear Raman/Brillouin mechanism [Cochrane *et al.* 1990]. The erbium doped fibre amplifier (EDFA) uses a relatively short section (1 to 100 m) of silica fibre pumped with optical, rather than electrical, energy. Because of the efficient coupling of fibre-to-fibre splices, high gains (20 dB) are achievable, Figure 19.18, over a 30 to 50 nm optical bandwidth. Practical amplifier designs generally have gains of 10 to 15 dB. The key attraction of this amplifier is the excellent end-to-end link SNR which is achievable and the enormous 4 to 7 THz (Hz $\times 10^{12}$) of optical bandwidth. (This far exceeds the 300 GHz of the entire radio, microwave and millimeter-wave spectrum.) Two interesting features of these amplifiers are the precise definition of the operating wavelength via the erbium doping and their relative immunity to signal dependent effects. They can therefore be engineered to maintain wide bandwidths when cascaded, and can operate equally well with OOK, FSK or PSK signals.

Injecting a high power laser beam into an optical fibre results in Raman scattering. Introducing a lower intensity signal-bearing beam into the same fibre with the pump energy, results in its amplification with gains of approximately 15 dB per W of pump power, coupled with bandwidths that are slightly smaller than the erbium amplifiers, Figure 19.18.

Brillouin scattering is a very efficient non-linear amplification mechanism that can realise high gains with modest optical pump powers (approximately 1 mW). However,

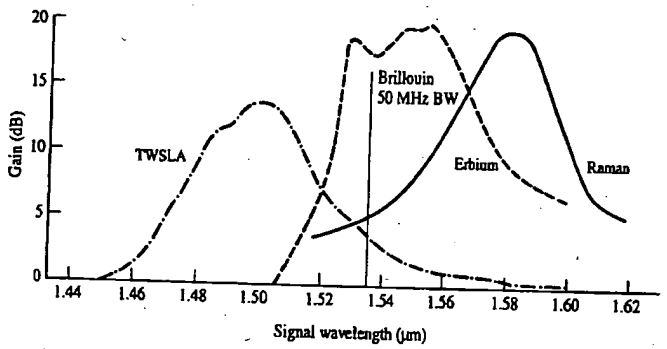


Figure 19.18 *Comparison of the gain of four distinct optical amplifier types (source: Cochrane, 1990, reproduced with the permission of British Telecommunications plc.).*

the bandwidth of only 50 MHz, Figure 19.18, is very limited in pure silica which makes such devices more applicable as narrowband tunable filters. This limited bandwidth fundamentally restricts the brillouin amplifier to relatively low bit rate communication systems.

A comprehensive comparison of the features of optical amplifiers is premature. However, what is clear is that long-haul optical transmission systems (10,000 km) are now feasible, with fibre amplifier based repeaters, and bit rates of 2 to 10 Gbit/s. At the present time TWSLA and erbium amplifiers generally require similar electrical pump power but TWSLAs achieve less gain, Figure 19.18, due to coupling losses. Erbium fibre amplifiers have the lowest noise and WDM channel crosstalk performance of all the amplifier types reported but high splice reflectivities can cause these amplifiers to enter the lasing condition. With the exception of brillouin, these amplifiers can be expected to be used across a broad range of system applications including transmitter power amplifiers, receiver preamplifiers, in-line repeater amplifiers and switches. The key advantage of EDFAs is that the lack of conversion to, and from, electrical signals for amplification gives rise to the 'dark fibre' – a highly reliable data super highway operating at tens of Gbit/s.

19.5.6 Optical repeater and link budgets

The electro-optic repeater is similar to the metallic line regenerator of Figures 6.15 and 6.30. For monomode fibre systems the light emitter is a laser diode and the detector uses an APD. The symbol timing recovery circuit uses zero crossing detection of the equalised received signal followed by pulse regeneration and filtering to generate the necessary sampling signals. Due to the high data rates, the filters often use surface acoustic wave devices [Matthews] exploiting their high frequency operation combined with acceptable Q value. The received SNR is given by:

$$\frac{S}{N} = \frac{I_p^2}{2q_e B(I_p + I_D) + I_n^2} \tag{19.1}$$

where I_p is the photodetector current, I_D is the leakage current, q_e is the charge on an electron, B is bandwidth and I_n is the RMS thermal noise current given by:

$$I_n^2 = \frac{4kTB}{R_L} \tag{19.2}$$

For received power levels of –30 dBm, $I_n^2 >> 2q_e B(I_p + I_D)$ giving, typically, a 15 to 20 dB SNR and hence an OOK BER in the range 10^{-7} to $< 10^{-10}$ [Alexander].

There are many current developments concerned with realising monolithic integrated electro-optic receivers. Integrated receivers can operate with sensitivities of –20 to –30 dBm and a BER of 10^{-10} at 155, 625 and 2488 Mbit/s, with a 20 dB optical overload capability. They are optimised for low crosstalk with other multiplex channels.

The power budget for a typical link in a fibre transmission system might have a transmitted power of 3 dBm, a 60 km path loss of 28 dB, a 1 dB path dispersion allowance and 4 dB system margin to give a –30 dBm received signal level which is

consistent with
with a similar
interfaces on suc
Long haul ex
using up to 12
10^{-8} have been
which the indivi

EXAMPLE 19.3
A monomode, 1.3

Optical output of
Connector loss at
Fibre specific atte
Average fibre spli
Fibre lengths, d
Connector loss at
Design margin (in
Required optical

Find the maximu

Let estimated lo
distributed over t

Total

Allo

Therefore:

8.0

$D' =$

The assumption

ΔL_S

This excess loss
extended by:

consistent with a low cost 155 Mbit/s transmission rate. The higher rate of 2.5 Gbit/s, with a similar receiver sensitivity of −30 dBm, would necessitate superior optical interfaces on such a 60 km link.

Long haul experiments and trials have achieved bit rates from 140 Mbit/s to 10 Gbit/s using up to 12 cascaded amplifiers spanning approximately 1000 km. BERs of 10^{-4} to 10^{-8} have been measured on a 500 km, five amplifier system, operating at 565 Mbit/s in which the individual amplifier gains were 7 to 12 dB [Cochrane *et al.* 1993].

EXAMPLE 19.3

A monomode, 1.3 μm, optical fibre communications system has the following specification:

Optical output of transmitter, P_T	0.0 dBm
Connector loss at transmitter, L_T	2.0 dB
Fibre specific attenuation, γ	0.6 dB/km
Average fibre splice (joint) loss, L_S	0.2 dB
Fibre lengths, d	2.0 km
Connector loss at receiver, L_R	1.0 dB
Design margin (including dispersion allowance), M	5.0 dB
Required optical carrier power at receiver, C	−30 dBm

Find the maximum loss-limited link length which can be operated without repeaters.

Let estimated loss-limited link length be D' km and assume, initially, that the splice loss is distributed over the entire fibre length.

$$\text{Total loss} = L_T + (D' \times \gamma) + \left(\frac{D'}{d} \times L_S\right) + L_R + M$$

$$= 2.0 + 0.6D' + \left(\frac{0.2}{2.0} \times D'\right) + 1.0 + 5.0$$

$$= 8.0 + 0.7D' \text{ dB}$$

$$\text{Allowed loss} = P_T - C = 0.0 - (-30) = 30.0 \text{ dB}$$

Therefore:

$$8.0 + 0.7D' = 30$$

$$D' = \frac{30.0 - 8.0}{0.7} = 31.4 \text{ km}$$

The assumption of distributed splice loss means that this loss has been over estimated by:

$$\Delta L_S = L_s [\, D' - int\,(D'/d)d \,] / 2$$

$$= 0.2 [\, 31.4 - int\,(31.4/2)2 \,] / 2$$

$$= 0.14 \text{ dB}$$

This excess loss can be reallocated to fibre specific attenuation allowing the link length to be extended by:

$$\Delta D = \Delta L_s/\gamma = 0.14/0.6 = 0.2 \text{ km}$$

The maximum link length, D, therefore becomes:

$$D = D' + \Delta D = 31.4 + 0.2 = 31.6 \text{ km}$$

19.5.7 Optical FDM

With the theoretical 50 THz of available bandwidth in an optical fibre transmission system, and with the modest linewidth of modern optical sources, it is now possible to implement optical FDM and transmit multiple optical carriers along a single fibre. The optical carriers might typically be spaced by 1 nm wavelengths. With the aid of optical filters these signals can be separated in the receiver to realise wavelength division multiplex (WDM) communications [Oliphant *et al.*]. It is envisaged that eventually up to 100 separate channels could be accommodated using this technique but the insertion loss of the multiplexers and crosstalk between channels still needs to be assessed. It has been demonstrated that 10 such combined signals, each modulated at 10 Gbit/s, can be transmitted through a practical fibre, and amplified using a single fibre amplifier without having to demultiplex the signals. WDM promises to increase by a hundredfold the information carrying capacity of fibre based systems when the necessary components for modulators and demodulators are fully developed.

Soliton transmission uses pulses that retain their shape for path lengths of thousands of kilometres due to the reciprocal effects of chromatic dispersion and a refractive index which is a function of intensity. Such systems have been constructed for 1,000 km paths with bit rates of 10 to 50 Gbit/s. In the laboratory, 10^6 km recirculating links have been demonstrated, corresponding to many circulations of the earth before the received SNR is unacceptable [Cochrane *et al.* 1993].

19.6 Network advantages of SDH systems

In the current plesiochronous hierarchy, within the transport signal at 140 Mbit/s, we may want to route component signals, for example 2 Mbit/s streams or tributaries, through the network. This requires us to demultiplex the transport signals layer-by-layer through the hierarchy, switch the tributary signals, and then remultiplex them into the next transport signal, Figure 19.9.

In the SDH, individual component signals do not have to be demultiplexed to their original bit rate; instead they are incorporated into a signal called a 'container' which can be handled in a convenient way throughout the network, Figure 19.14. Direct access to these component signals is thus possible, Figure 19.19. The result is a considerable reduction in multiplexer hardware in SDH systems, combined with improved operational flexibility.

With the introduction of SDH the opportunity can now be taken to replace network layers and topologies with those better suited to long haul resilient networks. With the

Figure 19.19 *Ad mu*

availability of, reconsider the st paths between th line systems, by (input line, outp provide ring acc PDN and is fur Figure 19.20 rep independent clo described in C information to t intended destina

Outer-core exchanges, in cc central site. Th much more reli has less interfac topologies, but

19.7 Data a

19.7.1 ISDN c

ISDN digital ac kbit/s and prima 19.1. The cus communication channel.

Basic rate channels at 6 International st Communicatio signalling or da

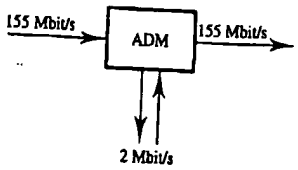


Figure 19.19 *Add-drop multiplexer (ADM) for simplified channel dropping which permits multiplexing into ring networks. (Compare with Figure 19.9.)*

availability of, very high speed, flexible SDH links it now becomes economic to reconsider the structure of the PSTN and replace simple (multiple) two-way transmission paths between the major centres, in which terminal multiplexers are two port or tributary-line systems, by high speed optical rings, as illustrated in Figure 19.20. The three port (input line, output line and tributary) add-drop multiplexers (ADMs), Figure 19.19, provide ring access and egress. This structure (Figure 19.20) will form the heart of the PDN and is fundamentally more reliable and less costly than the previous solution. Figure 19.20 represents an enhanced version of Figure 18.14. When the rings incorporate independent clockwise and anti-clockwise transmission circuits, as in the FDDI example described in Chapter 18, they offer immense flexibility and redundancy allowing information to be transmitted, via the ADM, in either direction around the ring to its intended destination.

Outer-core topologies will be mainly rings of SDH multiplexers linking local exchanges, in contrast to a plesiochronous multiplex where all traffic is routed through a central site. The SDH ring structure with its clockwise and anti-clockwise routing is much more reliable than centre-site routing. Furthermore, the SDH ring based network has less interface, and other, equipment. Access regions will remain, principally, star topologies, but will probably be implemented using optical technology, Figure 19.21.

19.7 Data access

19.7.1 ISDN data access

ISDN digital access was opened in the UK in 1985 and provided basic rate access at 144 kbit/s and primary rate access at 2 Mbit/s to the multiplex hierarchical structure of Figure 19.1. The customer interface for primary rate access provides for up to 30 PCM communications channels (e.g. a PABX) under the control of a common signalling channel.

Basic rate access provides the customer with two independent communications channels at 64 kbit/s together with a common signalling channel at 16 kbit/s. International standards for ISDN access (I.420) have now been agreed within ITU. Communications or bearer channels (B-channels) operate at 64 kbit/s, whilst the signalling or data channel (D-channel) operates at 16 kbit/s giving the basic rate total of

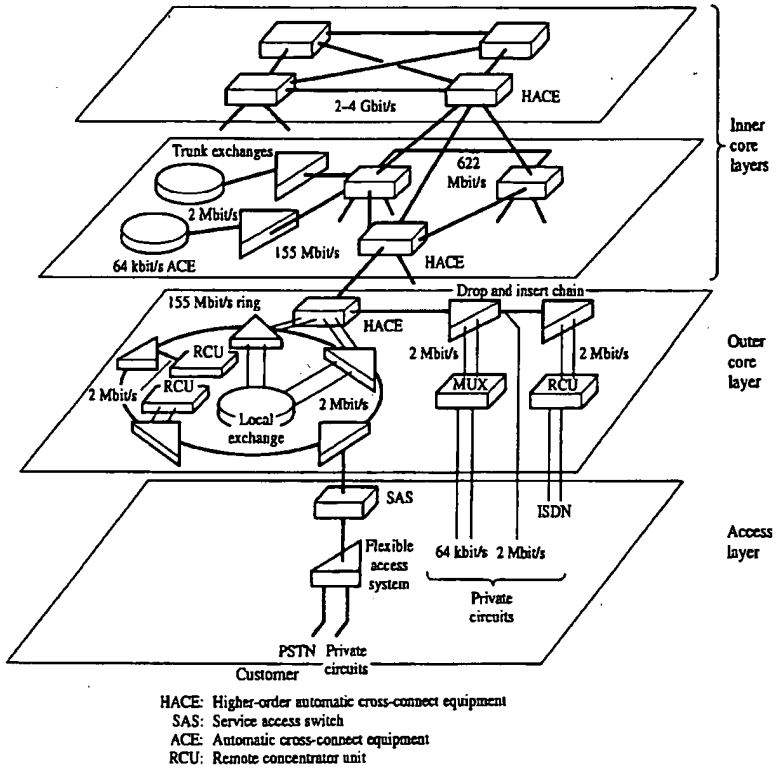


Figure 19.20 *Probable future SDH transmission network hierarchy (source: Leakey, 1991, reproduced with the permission of British Telecommunications plc.).*

144 kbit/s. The basic rate voice/data terminal transmits, full duplex, over a two-wire link with a reach of up to 2 km using standard telephone local loop copper cables. The transceiver integrated circuits employ a 256 kbaud, modified DPSK, burst modulation technique, Chapter 11, to minimise RFI/EMI and crosstalk. The D channel is used for signalling to establish (initiate) and disestablish (terminate) calls via standard protocols. During a call there is no signalling information and hence the D channel is available for packet switched data transmission.

Data access at 64 kbit/s is used in low bit rate image coders for videophone applications, Chapter 16. For high quality two way confravision services with full TV (512 × 512 pixel) resolution reduced bit rate coders have been designed to use primary access at 2 Mbit/s. Access at 2 Mbit/s is also required to implement wide area networks, Chapter 18, or to carry cellular telephone traffic between cell sites, Chapter 15.

TDM/WDM fibre trunk

Figure 19.21 *Pc*

The layer 1 the physical c (network termi are basically ec equipment (TE

TE2

Figure 19.22 *E*

Inner core layers

Outer core layer

Access layer

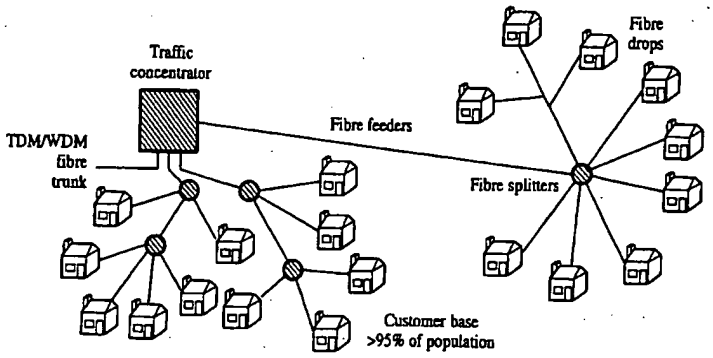


Figure 19.21 *Passive optical network for future local loop implementation.*

The layer 1 specification based on ITU-T I series recommendations [Fogarty] defines the physical characteristics of the user-network interface, Figure 19.22. The NT1 (network termination 1) terminates the transmission system and includes functions which are basically equivalent to layer 1 of the OSI architecture, Figure 18.12. The terminal equipment (TE) includes the ITU-T NT2 (network termination 2) function which

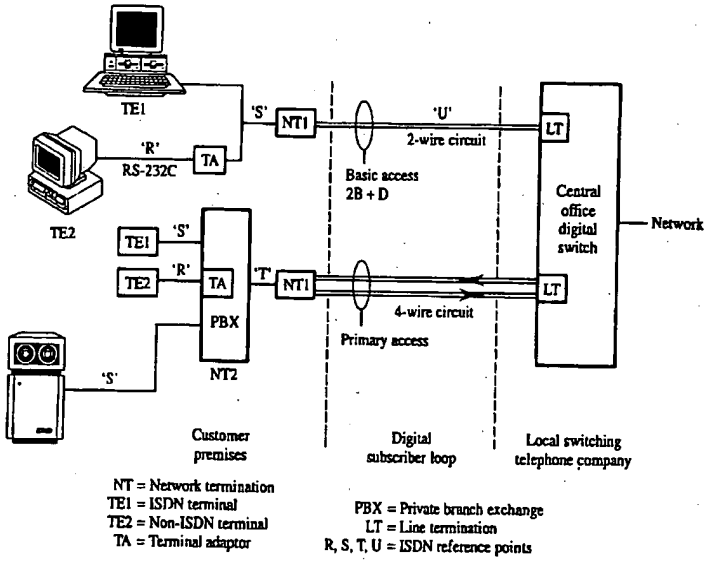


Figure 19.22 *Basic rate (144 kbit/s) and primary rate (2 Mbit/s) access to the ISDN.*

terminates ISO layers 1 to 3 of the interface. Figure 19.22 illustrates how digitised speech and data have access to the ISDN and includes the R, S, T and U ISDN reference points, which can be interconnected by standard interface integrated circuits.

Examples of TE1 equipments are ISDN telephone or fax machines which use the S interface. A TE2 might be a V.24 (RS232) data terminal or computer which requires the terminal adaptor (TA) to interface with the ISDN. The TA performs the processing to establish and disestablish calls over the ISDN and handles the higher level OSI protocol processing. For computer connection the TA is usually incorporated in the PC.

19.7.2 STM and ATM

Synchronous transfer mode (STM) and asynchronous transfer mode (ATM) both refer to techniques which deal with the allocation of usable bandwidth to user services. STM as used here is not to be confused with the synchronous transport module defined in section 19.4.3. In a digital voice network, STM allocates information blocks, at regular intervals (bytes/125 μs). Each STM channel is identified by the position of its time slots within the frame, Figure 19.5, which could easily be extended to a 30 channel, 2 Mbit/s system. STM works best when the network is handling a single service, such as the continuous bit-rate (bandwidth) requirements of voice, or a limited heterogeneous mix of services at fixed channel rates. Now, however, a dynamically changing mix of services requires a much broader range of bandwidths, and a switching capability adequate for both continuous traffic (such as voice and video conferencing) and non-continuous traffic (such as high-speed data and coded video traffic) in which bandwidth may change with time depending on the information rate.

While STM can provide data transfer services, the network operator must supply, and charge for, facilities with the full bandwidth needed by each service for 100% of the time – even if users require the peak bandwidth for only a small fraction of the time. The network therefore operates at low efficiency and the cost to users is prohibitive. This is not an attractive option for variable bit rate (VBR) traffic, such as coded video transmission, in which the data rate is dependent on how fast the image is changing.

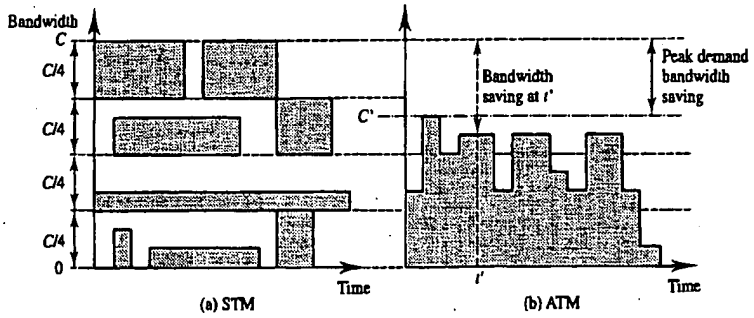


Figure 19.23 *Example of (a) STM and (b) ATM with mix of fixed and variable bit rate traffic.*

Figure 16.19 s
much smaller t
transmission ra

The structu
sharing both ba
fixed, predeterr
Figure 19.23(a)
service require
system, Chapte
traffic. When
users, there is
overhead is sm
representing a
audio, and 12
generally, a co
traffic applicati
for data applic
10.8.1, is imple

M

to achieve a
comparison to
(149.76 Mbit/s
hardware VLSI
network nodes.

One of the
LANs and PCs
ATM layer all
switched netw
efficiency, end
carried over lo
the power of *A*
much enhanced

The term a
connection may
actual demand.
transmission te
services in a gr
ISDN, for pac
Chapter 17, all
switches descr
seamlessly into
directly into the

Figure 16.19 shows an example of such traffic for which the average data rate is very much smaller than the peak rate. When many VBR sources are averaged then the peak transmission rate requirement comes much closer to the average rate.

The structure of the ATM protocol improves on the limited flexibility of STM by sharing both bandwidth and time [de Prycker]. Instead of breaking down bandwidth into fixed, predetermined, channels to carry information, as shown schematically for STM in Figure 19.23(a), ATM transfers fixed-size blocks of information, called cells, whenever a service requires transmission bandwidth, in a manner analogous to a packet switched system, Chapter 17. Figure 19.23(b) illustrates the (statistical) multiplexing of the ATM traffic. When the transmission bandwidth is dynamically allocated to variable bit rate users, there is a consequent peak bandwidth saving ($C' < C$), provided that the packet overhead is small. ATM uses a 5 byte header combined with a 48 byte data packet representing a compromise between the optimum data packet lengths of 16 bytes, for audio, and 128 bytes for video traffic. (The 53 bit ATM cell also represents, more generally, a compromise between short cell lengths required for real time, low delay, traffic applications such as packet speech, and long cell lengths which are more efficient for data applications due to their reduced proportion of overhead bits.) CRC, section 10.8.1, is implemented on the header information only using the polynomial:

$$M(x) = x^8 + x^2 + x^1 + 1 \tag{19.3}$$

to achieve a single error correction, but multiple error detection, capability. In comparison to other packet data networks (e.g. packet speech) ATM achieves high speed (149.76 Mbit/s corresponding to the payload bit rate of the SDH STM-1) by using hardware VLSI chips, rather than software, for the protocol processing and switching at network nodes.

One of the first services to use an ATM network for efficient data transport between LANs and PCs was a high-speed switched data service which carried VBR traffic. The ATM layer allows much higher data transfer rates than is possible with X.25 packet-switched networks (Chapter 18). As a result of ATM's remarkable flexibility and efficiency, end users can enjoy very-high-bandwidth services. These services can be carried over long distances by the PSTN with, potentially, very attractive tariffs. When the power of ATM is joined with the bandwidth and transmission quality of ISDN, a much enhanced service is achieved.

The term asynchronous in ATM refers to the fact that cells allocated to the same connection may exhibit an irregular occurrence pattern, as cells are filled according to the actual demand. This is an unfortunate term as it implies that ATM is an asynchronous transmission technique which is not the case. ATM facilitates the introduction of new services in a gradual and flexible manner, and is now the preferred access technique to the ISDN, for packet speech, video and other variable bit rate traffic. Queuing theory, Chapter 17, allows the analysis of ATM cell throughput rates and losses. The routing switches described in Chapter 18 are used for the ATM interfaces. ATM can fit seamlessly into the SDH frames of Figures 19.12 and 19.13 by accommodating the cells directly into the SDH payload envelope.

19.7.3 The local loop

Half of the investment of a telephone company is in the connections between subscriber handsets and their local exchange. Furthermore, this part of the network generates the least revenue since local calls are often cheap or, as in the USA, free. The length of these connections is 2 km on average and they seldom exceed 7 km. In rural areas the expense of installing copper connections now favours radio access for the local loop implementation. ISDN access demands 144 kbit/s duplex operation over 4 to 5 km which requires sophisticated signal processing if copper pair cables are used.

Assuming that the local loop is to be used for speech telephony and low speed data connections only, its replacement by fibre systems will be very gradual. If, however, we were to combine this requirement with cable TV and many other broadband services such as videophone, Chapter 16, then there would be an immediate demand for wideband local loop connections.

There will thus be a progressive move from copper based conductors to a fibre based passive optical network (PON) for the local connection. This will not be based on the current structure of one dedicated wire pair, or fibre, per household because of the large fibre bandwidth, section 19.5. (Furthermore, with 750 million telephones worldwide it would take more than 300 years for manufacturers to produce all the required cable at current production rates!) The future local network is therefore likely to comprise wideband fibre feeders with splitters and subsequent single fibre drops to each household, Figure 19.21. (One problem with the PON configuration is that there is no longer a copper connection to carry the power required for standby telephone operation.)

19.8 Summary

Multiplexing of PCM–TDM telephone traffic has been traditionally provided using the plesiochronous digital hierarchy. The PDH frame rate is 8000 frame/s and its multiplexing levels, bit rates and constituent signals are as follows:

PDH-1	2.048 Mbit/s	30+2, byte interleaved, 64 kbit/s voice channels – the PCM primary multiplex group
PDH-2	8.448 Mbit/s	4 bit-interleaved PDH-1 signals
PDH-3	34.368 Mbit/s	4 bit-interleaved PDH-2 signals
PDH-4	139.264 Mbit/s	4 bit-interleaved PDH-3 signals

Bit rates increase by a little more than a factor of four at each successive PDH level to allow for small differences in multiplexer clock speeds. Empty slots in a multiplexer output are filled with justification bits as necessary. A serious disadvantage of PDH multiplexing is the multiplex mountain which must be scaled each time a lower level signal is added to, or dropped from, a higher order signal. This is necessary because bit interleaving combined with the presence of justification bits means that complete demultiplexing is required in order to identify the bytes belonging to a given set of voice

channels.

The synch
equivalent, SO
that low level
lower level mu
higher order m
cross-connecti
8000 frame/s
(STMs). The
synchronous tr
rates, and cons

SON
SD
SD
SD

SONET si
The capacity o
the operation c

SONET a
transmission s
Laser diodes
performance,
dispersion and
contain PIN d
more sensitive
powers are –3
detector, a con
TWSLAs) ma
and consist of
signal propaga

Fibres cur
1.3 and 1.5 μ
refractive ind
dimensions a
Multimode gr
refractive inde
in larger avai
dispersion wh
(material disp
electromagnet
contributions
which also co

channels.

The synchronous digital hierarchy (SDH) and its originating North American equivalent, SONET, will eventually replace the PDH. The principal advantage of SDH is that low level signals remain visible in the multiplexing frame structure. This allows lower level multiplexes (down to individual voice channels) to be added or dropped from higher order multiplex signals without demultiplexing the entire frame. This simplifies cross-connection of traffic from one signal multiplex to another. The SDH frame rate is 8000 frame/s and each frame contains one or more synchronous transport modules (STMs). The SONET frame rate is also 8000 frame/s and each contains one or more synchronous transport signals (STSs). The standard SONET/STM payload capacities, bit rates, and constituent signals are as follows:

SONET	STS-1	9 × 90 bytes	51.84 Mbit/s	
SDH	STM-1	9 × 270 bytes	155.52 Mbit/s	3 STS-1s
SDH	STM-4	9 × 1080 bytes	622.08 Mbit/s	4 STM-1s
SDH	STM-16	9 × 4320 bytes	2.488 Gbit/s	4 STM-4s

SONET signals with capacities based on other multiples of STS-1 are also possible. The capacity of an STM-1 is such that it can carry one PDH-4 signal which will facilitate the operation of PSTNs during the period in which both multiplexing schemes are in use.

SONET and SDH have been designed, primarily, to operate with optical fibre transmission systems. Optical sources may be coherent [Hooijmans] or incoherent. Laser diodes have narrow spectra and are therefore usually the choice for high performance, high bit rate links. LEDs are less spectrally pure, leading to greater dispersion and smaller useful bandwidth, but are cheaper. Optical detectors typically contain PIN diodes or avalanche photodiodes (APDs). APDs are more expensive but more sensitive. Typical optical transmit powers are 0 dBm and typical optical receiver powers are −30 dBm. An optical fibre repeater may be implemented using an optical detector, a conventional electronic repeater and an optical source. Optical amplifiers (e.g. TWSLAs) may also be used. The most recent types of optical amplifier are distributed and consist of doped fibre sections which are optically pumped and lase as the optical signal propagates through them.

Fibres currently operate, in decreasing order of attenuation, at wavelengths of 0.85, 1.3 and 1.5 μm. They can be divided into three types depending on the profile of their refractive index variations. Multimode step-index fibres have relatively large core dimensions and suffer from modal dispersion which limits their useful bandwidth. Multimode graded-index fibres also have large core dimensions but use their variation in refractive index to offset the difference in propagation velocity between modes resulting in larger available bandwidth. Monomode (step-index) fibres suffer only chromatic dispersion which arises partly from the frequency dependence of refractive index (material dispersion) and partly from the frequency dependence of a propagating electromagnetic mode velocity in a waveguide of fixed dimensions. These two contributions can be made to cancel at an operating wavelength around 1.5 μm, however, which also corresponds to a minimum in optical attenuation of about 0.15 dB/km. This

wavelength is therefore an excellent choice for long, low dispersion, high bit rate, links. Repeaterless links, hundreds of kilometres long, operating at Gbit/s data rates are now possible. Wavelength division multiplexing (WDM) promises to increase the communications capacity of a single fibre still further.

SDH, combined with optical fibre transmission, has allowed a re-evaluation of the PDN network topology. Future access is likely to be via a passive optical network (PON) at ISDN basic (144 kbit/s) or primary (2 Mbit/s) rates. 155 Mbit/s rings will connect to a network of higher-order automatic cross-connect equipment (HACE) which will themselves be interconnected at bit rates of 155 and 622 Mbit/s and higher. The highest layer of cross-connect equipment will be fully interconnected with 2.4 Gbit/s links.

Asynchronous transfer mode (ATM) will provide efficient use of time and bandwidth resources in the access layers of the network when variable bit rate services are provided. ATM frames consist of 53 byte cells, 48 of which carry traffic and 5 of which carry overhead. At the higher network layers ATM cells will be carried within the payload envelopes of SDH frames.

19.9 Problems

19.1. How does a plesiochronous multiplex function?

19.2. Explain the notion of section, line and path as entities in an SDH transmission network. Taking an STS-1 frame as an example, where is the information concerning these entities carried in the SDH signal? What is the nature of the information?

19.3. What mechanism is used to allow an SDH SPE to pass between SDH networks which are not synchronised? How does this differ from the mechanism used to allow tributaries from the old plesiochronous system (e.g. 2.048 Mbit/s) to be taken in and out of an SDH system?

19.4. Explain the term add-drop in the context of multiplexers. Draw block diagrams to show why this function is simpler to perform in the SDH system than in the older PDH system.

APPE

Tabulated val

e

x	erf x
0.00	0.000000
0.01	0.011283
0.02	0.022565
0.03	0.033841
0.04	0.045111
0.05	0.056372
0.06	0.067622
0.07	0.078858
0.08	0.090078
0.09	0.101282
0.10	0.112463
0.11	0.123623
0.12	0.134758
0.13	0.145867
0.14	0.156947
0.15	0.167996
0.16	0.179012
0.17	0.189992
0.18	0.200936
0.19	0.211840
0.20	0.222703
0.21	0.233522
0.22	0.244296
0.23	0.255023
0.24	0.265700
0.25	0.276326
0.26	0.286900
0.27	0.297418
0.28	0.307880
0.29	0.318283
0.30	0.328627
0.31	0.338908
0.32	0.349126
0.33	0.359279
0.34	0.369365
0.35	0.379382
0.36	0.389330
0.37	0.399206
0.38	0.409009
0.39	0.418739

Matt Blaze's Technical Papers

Last updated 6 August 2006

Many of my technical papers are available here. Newer papers are usually in Adobe PDF format; like it or not, PDF is the de facto standard format for scientific papers these days. Most of the older papers are in PostScript format; you'll need a PostScript printer or viewer (such as GhostView) to read them. Most of these files have also been converted to Adobe PDF format (using ps2pdf) and can be viewed or printed with a PDF viewer such as Acrobat, acroread4, or xpdf. If you have a choice, you'll probably find the PostScript version looks and works better than the PDF version does (ps2pdf doesn't do particularly well with some of the fonts). A few papers are available as plain ASCII text or LaTeX source.

Wiretapping, Surveillance and Countermeasures

The Trustworthy Network Eavesdropping and Countermeasures (TNEC) project studies the reliability of communications interception systems and technologies. A better understanding of the limitations of eavesdropping techniques could lead to more trustworthy law enforcement wiretap evidence (or at least more appropriate treatment of electronic evidence), networks with properties that inherently frustrate (or facilitate) interception, and new techniques for achieving communications security.

One of our first efforts is a comprehensive analysis of the wiretapping technologies used by law enforcement (for both voice and data). We have found serious exploitable weaknesses in fielded interception systems. For details, including audio demos of novel eavesdropping countermeasures, see the wiretapping web page here.

- M. Sherr. E. Cronin, S. Clark and M. Blaze.
  http://www.crypto.com/papers/wiretapping/ web page.

- M. Sherr, E. Cronin, S. Clark and M. Blaze. "Signaling Vulnerabilities in Wiretapping Systems." *IEEE Security and Privacy.* November/December 2005. [PDF].

Similar vulnerabilities exist in digital Internet eavedropping systems as well:

- E. Cronin, M. Sherr, and M. Blaze. "The Eavedsdropper's Dilemma." Technical Report MS-CIS-05-24. University of Pennsylvania. 2005. [PDF].

Another focus of the TNEC project examines local host-based surveillance. The *JitterBug* demonstrates a novel eavesdropping threat against typed keyboard input. Commercially-available hardware keyboard "sniffers" can easily capture and store an unsuspecting user's keystrokes. Because a subverted keyboard has no direct network connection, sniffer attacks are generally assumed to require either support software on the host or periodic in-person access by the attacker to retrieve the data. We show that this need not be the case. A new technique based on "JitterBugs" can exfiltrate captured data entirely through subtle perturbations in the precise times at which typed keystrokes are passed to the host. Whenever a user runs an interactive network application (such as SSH), an attacker can derive previously captured keystrokes entirely by observing the timing of network packets, even from across the

Internet or via encrypted wireless traffic. The JitterBug demonstrates that input devices must be scrutinized as part of any trusted computing base and, more generally, that simple "supply chain attacks" can represent a practical and serious threat to data confidentiality. (Gaurav Shah and Andres Molina won the Best Student Paper award at USENIX Security 2006 for this work.)

- G. Shah, A. Molina, and M. Blaze. "Keyboards and Covert Channels." *Proc. 15th USENIX Security Symposium.* Vancouver, BC. August 2006. [PDF].

- Gaurav Shah's JitterBug page: `http://www.cis.upenn.edu/~gauravsh/jitterbug.html`. JitterBug prototype code and PCB template.

Physical and "Human-Scale" Security

Cryptologic techniques can be applied outside of computers and networks, Perhaps surprisingly, the abstractions used in analyzing secure computing and communications systems turn out also to be useful for understanding mechnical locks and their keyspaces. Indeed, modeling master keyed locks as online authentication oracles leads directly to efficient solutions for what might naively seem like exponential problems for the attacker. In fact, it seems like almost a textbook example, as if master keying practices for locks were designed specifically to illustrate this class of weakness. We sometimes assume that hardware-based security is inherently superior to that based in software, but even the humble mechanical lock can be just as insecure as complex computing systems, and can fail in similar ways.

A widely circulated paper of mine describes attacks against master keyed mechanical locks. For an overview of the attack, which was described in the January 23rd 2003 *New York Times*, click here. For a brief commentary on the reaction to this paper, see my essay, "Keep it secret, stupid!" (click here), which was originally posted to comp.risks.

(Warning: there are embedded photos in this paper; they make the PS and PDF files very large. The GZIPed PostScript version is 5.7MB long (uncompresses to 14MB), and the PDF version is 4MB long.)

- M. Blaze. "Cryptology and Physical Security: Rights Amplification in Master-Keyed Mechanical Locks." March 2003. *IEEE Security and Privacy.* March/April 2003. [GZIPed PostScript], [PDF].

- My *Notes on Picking Pin Tumbler Locks*, intended primarily for use by students in my security seminar, can be found here [HTML].

While the security metrics and mechanical safeguards used in safes and vaults may not relay on the latest technology, they are often quite ingenious. They may have much to teach computer security. Some of what I understand about the subject is in the survey paper below (warning -- heavily illustrated 2.5MB .pdf file). And for a brief commentary on the reaction to *this* paper, see my essay, "the second sincerest form of flattery" (click here), which was originally posted to `interesting-people`.

- M. Blaze. "Safecracking for the Computer Scientist." *U. Penn CIS Department Technical Report.* 7 December 2004 (revised 20 December 2004). [PDF].

This position paper, presented at the Cambridge Security Protocols Workshop 2004, introduces and advocates the "Human Scale Security Project," which supports the above work.

- M. Blaze. "Toward a broader view of security protocols." *12th Cambridge International Workshop on Security Protocols*. Cambridge, UK. April 2004. [PDF].

Trust Management

These papers introduce the "trust management" approach to specifying and enforcing security policy.

- The Trust Management Web Page, updated regularly.
- M. Blaze, J. Ioannidis, A. Keromytis. "Offline Micropayments without Trusted Hardware." *Financial Cryptography 2001*. Grand Cayman, February 2001. [PostScript], [PDF].
- M. Blaze, J. Ioannidis, A. Keromytis. "Trust Management for IPSEC." *NDSS 2001*. San Diego, February 2001. [PDF].
- M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis. The KeyNote Trust Management System, Version 2. *RFC-2704*. IETF, September 1999. [ASCII Text].
- M. Blaze, J. Ioannidis, A. Keromytis. "Compliance Checking and IPSEC Policy Management." *Internet Draft*. `draft-blaze-ipsp-trustmgt-00.txt`. IETF, March 2000. [ASCII Text].
- M. Blaze, J. Ioannidis, A. Keromytis. "DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System." *RFC-2792*. IETF, March 2000. [ASCII Text]
- M. Blaze, J. Ioannidis, and A. Keromytis. " Trust Management and Network-Layer Security Protocols." *1999 Cambridge Protocols Workshop*. Cambridge, April 1999. [PostScript], [PDF], [LaTeX Source].
- M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. "The Role of Trust Management in Distributed Systems Security." Chapter in *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, (Vitek and Jensen, eds.) Springer-Verlag, 1999. [PostScript], [PDF].
- M. Blaze, J. Feigenbaum, M. Strauss. "Compliance-Checking in the PolicyMaker Trust-Management System." *Proc. 2nd Conference on Financial Cryptography*. Anguilla 1998. LNCS 1465, pp 251-265, Springer-Verlag, 1998. [PostScript], [PDF].
- M. Blaze, J. Feigenbaum and J. Lacy. "Decentralized Trust Management." *IEEE Symposium on Security and Privacy*, Oakland, CA. May 1996. [PostScript], [PDF].

Angelos Keromytis's KeyNote Trust Management toolkit and open-source reference

implementation is available here as a GZIPed TAR archive. The toolkit runs under most Unix-like (BSD, linux, etc.) platforms, with limited support for Win32 platforms.

Also see Angelos Keromytis' KeyNote web page for the latest details on the KeyNote implementation.

Remotely-Keyed Encryption

These papers introduce and formalize the notion of "remotely-keyed" encryption, in which a low-bandwidth, but trusted device (such as a smart card) assists a high-bandwidth, but untrusted host with bulk encryption.

- M. Blaze, J. Feigenbaum, and M. Naor. "A Formal Treatment of Remotely Keyed Encryption (Extended Abstract)". Eurocrypt '98, Helsinki. LNCS 1403 pp. 251-265. [PostScript], [PDF].
- M. Blaze. "High-Bandwidth Encryption with Low-Bandwidth Smartcards." January 18, 1996. *Cambridge Workshop on Fast Software Encryption*, February 1996. [PostScript], [PDF].

Key Escrow

These papers describe and evaluate various key escrow proposals, from a technical (as opposed to political) perspective.

- *The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption* (second edition). June 1998. [HTML], [PDF].
- *The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption* (first edition). May 1997. (OBSOLETE: superseded by second edition, above). [ASCII Text], [PDF], [PostScript].
- M. Blaze. "Oblivious Key Escrow." *First Cambridge Workshop on Information Hiding* May 1996. Springer 1997. [PostScript], [PDF], [LaTeX source].
- Memo from NSA regarding key length report, with comments from M. Blaze and W. Diffie. July 18, 1996. [ASCII Text].
- M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson and M. Wiener. "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security". Report of ad hoc panel of cryptographers and computer scientists. January 1996. [ASCII Text], [PDF], [PostScript].
- M. Blaze, J. Feigenbaum and F.T. Leighton. "Master-Key Cryptosytems." Abstract presented at *Crypto '95 (rump session)*, Santa Barbara, CA, August 1995. [PostScript], [PDF]
- M. Blaze. "Protocol Failure in the Escrowed Encryption Standard." *Proceedings of Second ACM Conference on Computer and Communications Security*, Fairfax, VA, November 1994. [PostScript], [PDF].

Network-Layer Security

These papers describe the design and implementation network-layer and related security protocols, including JFK, a secure key exchange protocol, and swIPe, a predecessor to the IPSEC standard. (At this point, swIPe is of primarily historical interest, although the USENIX paper should be of some value to IPSEC implementors. JFK is a useful key exchange protocol that should be especially valuable for IPSEC and network security key management).

- W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. "Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols." In Proc. ACM Computer and Communications Security (CCS) Conference. November 2002, Washington, DC. (pp 48-58). [PDF].

- J. Ioannidis and M. Blaze. "The swIPe IP Security Protocol." *Internet Draft*. December 1993. [ASCII Text].

- J. Ioannidis and M. Blaze. "Architecture and Implementation of Network Layer Security Under UNIX." *Proceedings of the Fourth USENIX Security Workshop*, October 1993. [PostScript], [PDF].

Cryptographic Applications

- R. Levein, L. McCarthy, M. Blaze. "Transparent Internet E-mail Security (DRAFT)". August 9, 1996. [PostScript], [PDF].

- M. Blaze and S.M. Bellovin. "Session-Layer Encryption." *Proceedings of the USENIX Security Workshop*, June 1995. [PostScript].

- M. Blaze. "Key Management in an Encrypting File System." *USENIX Summer 1994 Technical Conference*, Boston, MA, June 1994. [PostScript], [PDF].

- M. Blaze. "A Cryptographic File System for Unix." *Proceedings of the First ACM Conference on Computer and Communications Security*, Fairfax, VA, November 1993. [PostScript], [PDF].

  The latest CFS code can be found here.

Ciphers and Algorithms

- S. M. Bellovin, M. Blaze. "Cryptographic Modes of Operation for the Internet." NIST Workshop on AES Modes. Santa Barbara, CA. August 2001. [PDF].

- M. Blaze, M. Strauss. "Atomic Proxy Cryptography." Full version of our *EuroCrypt '98* paper. May 1997. [PostScript], [PDF].

- M. Blaze. "Efficient Symmetric-Key Ciphers Based on an NP-Complete Subproblem (DRAFT)". October 2, 1996. [PostScript], [PDF]

- M. Blaze and B. Schneier. "The MacGuffin Block Cipher Algorithm." *Leuven Workshop on Cryptographic Algorithms*, Leuven, Belgium, December 1994.

[PostScript], [PDF].

Cryptography Policy, Export Regulations, and Politics

- M. Blaze. Declaration in Felten, et al v. RIAA. 13 August 2001. [ASCII Text].
- S. Bellovin, M. Blaze, D. Farber, P. Neumann, E. Spafford. "Comments on the Carnivore System Technical Review." Formal comments to the US Department of Justice. 3 December 2000. [HTML].
- M. Blaze & S. M. Bellovin. "Tapping, Tapping on my Network Door." INSIDE RISKS 124. *CACM*, October 2000. [HTML].
- M. Blaze. "Cryptography Policy and the Information Economy." Draft. 17 December 1996. [PostScript], [PDF], [ASCII Text].
- My prepared testimony before the Senate Commerce Committee subcommittee on Science, Technology, and Space. June 26, 1996 [ASCII Text].
- M. Blaze. "My Life as an International Arms Courier." January, 1995. Adapated from posting to *comp.risks* [ASCII Text]

Peer-to-Peer Networking

My dissertation work, over ten years ago, anticipated and analyzed what we would now call "Peer-to-Peer" file distribution.

- M. Blaze. Caching in Large-Scale Distributed File Systems. PhD thesis. Princeton University Department of Computer Science. November 1992. [PostScript].

Other People's Papers

From time to time, I make available papers from other researchers that I didn't write myself but that are of wide interest and don't otherwise have a home. Here's what's available now:

- S. Fluhrer, I. Mantin and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. Preliminary Draft, July 25, 2001. [PostScript].
- A. Biryukov and A. Shamir. Real Time Cryptanalysis of the Alleged A5/1 on a PC. Preliminary Draft, December 9, 1999. [PostScript].

Click here to return to the crypto.com home page.

TechEncyclopedia More than 20,000 IT terms

Results found for: inverted file

inverted file

In data management, a file that is indexed on many of the attributes of the data itself. For example, in an employee file, an index could be maintained for all secretaries, another for managers. It is faster to search the indexes than every record. Also known as "inverted lists," inverted file indexes use a lot of disk space; searching is fast, updating is slower.

TERMS SIMILAR TO YOUR ENTRY

Entries before inverted file

- inverse addressing
- inverse kinematics
- inverse multiplexor
- inverse telecine
- inverse video

Entries after inverted file

- inverted list
- inverter
- InVircible
- invisible GIF
- invisible Web

DEFINE ANOTHER IT TERM

Define

Or get a random definition

Brian Cryer's Glossary of IT Terms with Links

Inverted File Index

Inverted File Index

A type of Inverted Index where each index provides a mapping from words to the documents that contain them. cf Fully Inverted Index.

Can you add to this definition? If so please Report an Observation. Do you know of a relevant link to add under this definition? If so please Add a Link.

Add a Term | Add a Link | Report an Observation | Glossary | Home

thefreedictionary.com

(database, information science) **Inverted index** - A sequence of (key, pointer) pairs where each pointer points to a record in a database which contains the key value in some particular field. The index is sorted on the key values to allow rapid searching for a particular key value, using e.g. binary search. The index is "inverted" in the sense that the key value is used to find the record rather than the other way round. For databases in which the records may be searched based on more than one field, multiple indices may be created that are sorted on those keys.

An index may contain gaps to allow for new entries to be added in the correct sort order without always requiring the following entries to be shifted out of the way.

inverted file index

(data structure)

Definition: An *inverted index* which only indicates the text in which a word appears, not where the word appears within the text.

See also *full inverted index*, *block addressing index*.

Note: See the example at inverted index.

Author: PEB

More information

Nivio Ziviani, Edleno Silva de Moura, Gonzalo Navarro, Ricardo Baeza-Yates, *Compression: A Key for Next-Generation Text Retrieval Systems*, IEEE Computer, 33(11):37-44, November 2000, (page 42).

Go to the Dictionary of Algorithms and Data Structures home page.

If you have suggestions, corrections, or comments, please get in touch with Paul E. Black.

Entry modified 17 December 2004.
HTML page formatted Wed Apr 19 12:22:45 2006.

Cite this as:
Paul E. Black, "inverted file index", in *Dictionary of Algorithms and Data Structures* [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. 17 December 2004. (accessed TODAY) Available from: http://www.nist.gov/dads/HTML/invertedFileIndex.html

Inverted index

From Wikipedia, the free encyclopedia

An **inverted index** is an index structure storing a mapping from words to their locations in a document or a set of documents, giving full text search. An inverted index is the most important data structure used in search engines. Such an associative array is a multimap, and can be implemented in many ways. It could be a hash table, where the keys are words (strings), and the values are arrays of locations.

There are two main variants of inverted indexes: An *inverted file index* contains for each word a list of references to all the documents in which it occurs. A *full inverted index* additionally contains information about where in the documents the words appear. This could be implemented in several ways. The simplest is perhaps a list of all pairs of document IDs and local positions. An *inverted file index* needs less space, but also has less functionality. You can do *term search* (what you usually do in a search engine), but not *phrase search* (what you usually get when you put quotes around your search query).

Contents

Example

Given the texts T_0 = `"it is what it is"`, T_1 = `"what is it"` and T_2 = `"it is a banana"`, we have the following *inverted file index*:

```
"a":      {2}
"banana": {2}
"is":     {0, 1, 2}
"it":     {0, 1, 2}
"what":   {0, 1}
```

A *term search* for the terms `"what"`, `"is"` and `"it"` would give the set $\{0,1\} \cap \{0,1,2\} \cap \{0,1,2\} = \{0,1\}$.

With the same texts, we get the following *full inverted index*, where the pairs are document numbers and local word numbers. Like the document numbers, local word numbers also begin with zero. So, `"banana": {(2, 3)}` means the word "banana" is in the third document (T_2), and it is the fourth word in that document (position 3).

```
"a":      {(2, 2)}
"banana": {(2, 3)}
"is":     {(0, 1), (0, 4), (1, 1), (2, 1)}
"it":     {(0, 0), (0, 3), (1, 2), (2, 0)}
"what":   {(0, 2), (1, 0)}
```

If we run a *phrase search* for `"what is it"` we get hits for all the words in both document 0 and 1. But the terms occur only consecutively in document 1.

References

- Donald Knuth. *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Pages 560–563 of section 6.5: Retrieval on Secondary Keys.
- Justin Zobel, Alistair Moffat and Kotagiri Ramamohanarao, *Inverted files versus signature files for text indexing*. ACM Transactions on Database Systems (TODS), Volume 23, Issue 4 (December 1998), Pages: 453 - 490.

See also

Vector space model

External links

- NIST's Dictionary of Algorithms and Data Structures: inverted index (http://www.nist.gov/dads/HTML/invertedIndex.html)
- Inverted index for search engine (http://www.rankcount.com/reverse-keyword-search.php)

Retrieved from "http://en.wikipedia.org/wiki/Inverted_index"

Categories: Data structures | Algorithms on strings

About CNI

Task Force Meetings

Conferences

Projects

CNI Collaborations

Site Map

Proceedings: Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment

Coalition for Networked Information

Interactive Multimedia Association

John F. Kennedy School of Government

Science, Technology & Public Policy Program

Massachusetts Institute of Technology

Program on Digital Open High-Resolution Systems

Rather than put a trademark symbol in every occurrence of other trademarked names, we state that we are using the names only in an editorial fashion, and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Published by:

Interactive Multimedia Association
Intellectual Property Project
3 Church Circle
Suite 800
Annapolis, MD 21401-1933
Phone: (410) 626-1380
FAX: (410) 263-0590

Table of Contents

Consortium for University Printing and Information Distribution (CUPID)

A Publishing and Royalty Model for Networked Documents

Theodor Holm Nelson

Acronyms List

Last updated Monday, July 2, 2001.

tiscali

From: www.tiscali.co.uk/reference/

Index

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

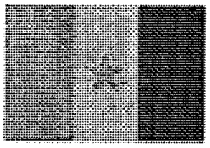
Interactive Multimedia Association
Organization founded in 1987 to promote the growth of the multimedia industry. Based in Anapolis, Maryland, USA, the IMA runs special interest groups, summit meetings, conferences, and trade shows for its member companies.

Dictionary Search

Search for:

Go

Senegal Flag



The star represents Islam and expresses peace, harmony, hope, and socialism. The tricolour is reminiscent of the flag of France, the former colonial power. The pan-African colours express unity with other African nations. Effective date: 25 August 1960. >>

About CNI

Task Force Meetings

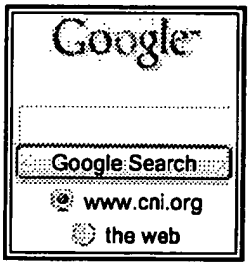
Conferences

Presentations/ Publications

Projects

CNI Collaborations

Site Map



CNI Membership and Task Force Information

Frequently Asked Questions about CNI Membership

List of CNI Members

Becoming a Member
The Coalition for Networked information (CNI) is an institutional membership organization. Membership dues are the primary financial resource for CNI's programs. Dues for the year (July 1, 2006-June 30, 2007) are $6,200, and membership is renewable on a year-by-year basis.

CNI's members include:

- Higher Education Institutions
- Publishers
- Network & Telecommunications Companies
- Scholarly Organizations
- Professional Organizations
- Libraries
- Information Technology Companies
- Government Agencies

To request more information or a membership application, write to Jackie Eudell at <jackie@cni.org>.

The Task Force
Each of CNI's member institutions appoints two representatives to the CNI Task Force, which holds semi-annual meetings. Higher education institutions are encouraged to appoint their heads of libraries and information technology. Other types of institutions generally appoint top administrators or directors of electronic publishing. All members of the Task Force have equal status; for example, there are no separate membership categories for corporate members. A Steering Committee guides the Coalition's activities.

Last updated Tuesday, May 2, 2006.

Table 10.4 Terminology

CCITT	USA	UK
Primary centre	Toll centre	Group switching centre
Secondary centre	Primary centre	District switching centre
Tertiary centre	Sectional centre	Main switching centre
Trunk exchange	Toll office	Trunk exchange
Trunk network	Toll network	Trunk network
Trunk circuit	Trunk	Trunk (circuit)
Local exchange	End (central) office	Local exchange
Junction circuit	Inter-office trunk	Junction

be used. The words used in North America and the UK differ significantly in their meaning. Here we will refer to the various levels of switching by using the CCITT recommended terms. Table 10.4 relates them to those commonly used in North America and the UK.

From Fig. 10.35 we can see that there are several levels of switching that combine to form the complete network. It is usual to think of the systems in two parts. The first is the junction network, serving the subscriber and consisting of the link from subscriber to local exchange, from local exchange to primary switching centre, and back to the called subscriber via another local exchange. The second part of the system is the trunk network, which is concerned only with calls passing at primary centre level and above. Thus the primary centre is associated with both parts of the network.

The structure shown by the solid lines represents the basic hierarchical network of the system, and telephone calls routed along these links are said to be travelling on a backbone or final route. When there is much traffic between exchanges, either belonging to the same cluster, or at different levels, direct routes may be installed to save a stage of switching. Such routes are shown as dotted lines in Fig. 10.35. These direct routes are dimensioned to a different grade of service from the final routes. Because a call finding all the direct-route circuits busy can be placed on the backbone route, the direct route can be dimensioned to a higher grade of service than the final route, and thus work more efficiently.

The number of exchanges at the various levels depends on several factors: the physical extent of the network, the number of subscribers, the amount of traffic, the forecast growth and the transmission methods used. Beyond the top level of the national system is a layer that gives access to the international network. This layer may consist of one or more international (usually called gateway) exchanges.

10.32 Numbering Schemes

In modern systems, the numbering scheme used by a telephone administration to allocate subscribers' numbers has an underlying plan, and there are a few constraints on the development of the plan that must be taken into account:

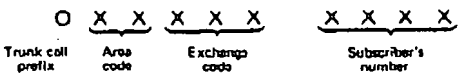


Fig. 10.36 National telephone number.

(i) It must provide each subscriber with a unique number within the national network.
(ii) The allocation to areas must be able to meet forecast growth for several decades.
(iii) The number of digits should not exceed that recommended by CCITT.

In principal, (i) is easy to satisfy if (ii) and (iii) have been met. The length of the number recommended by CCITT is $11 - n$ where n is the country code (see below). If, for example, n is 2, then the national number should not exceed 9 digits in length. These digits are used to denote the subscriber's number on the local exchange, the exchange within a given area, and the area within the national numbering scheme. In many local exchanges there is a maximum capacity of 10 000 lines, thus the last four digits of the national number are allocated to the subscriber's number in the exchange. Of the remaining five digits in our example, the first two would denote the area and the remaining three the exchange within the area. Thus the number has the form shown in Fig. 10.36.

The division of digits between area exchange and subscribers code may not be the same as that shown – but all three components exist in every number.

For automatic long-distance dialling a prefix is necessary to indicate to the exchange equipment that a trunk call is being made. In many countries a "0" is used, but any other digit would do. In calculating the length of the national number the prefix is not included.

International Number

In the introduction to this chapter we imagined a future when person-to-person calls could be made very easily via individual instruments and satellite links. For such availability of connections to be possible each subscriber in the world must have a unique number. To achieve that, some agreement between countries is essential; each country must be identified by a number different from that of all other countries. That is achieved by agreement through CCITT on the way in which these codes, called country codes, are allocated.

The first digit of the country code is the zone code; the world is divided up into nine zones and each country belongs to a zone. The relationship between zone number and geographical area is shown in Table 10.5.

In all but two of the zones, one or two digits are added to the zone number to produce the country code. For example, Brazil has a zone number 5 and an additional country code digit 5 to give its country code 55. Brunei is in zone 6 and two further digits are added to give a country code 673.

The two exceptions are zones 1 (North America and the Caribbean) and 7

Table 10.5 Zone numbers

1	North America	6	Australasia
2	Africa	7	USSR
3	Europe	8	Eastern Asia
4	Europe	9	Far East and Middle East
5	South America	0	Spare

(USSR). Throughout each of these zones there is a linked numbering scheme that means, for example, that no subscriber in Canada has the same national number as a subscriber in the USA. Consequently, to connect to anyone in zone 1 the digit 1 is followed by the national number. A similar situation exists in the USSR. Europe is at the other extreme; there are many countries with large national networks that have nine digits in their national numbers. For these, a two-digit country code is required, and that can only be achieved by having two zone numbers allocated to Europe.

The division of the world into the zones shown in Table 10.5 is intended to be satisfactory until early in the next century, but clearly, as some large countries develop their telephone networks, some adjustment will be necessary at some future time.

The national and international numbering schemes we have discussed above are the simplest. However, in several parts of the world there are small exceptions, particularly in regard to local calls. In the scheme where the national number is used for all calls within a country, it can lead to irritation on the part of the subscriber and long set-up times for the exchange equipment. Consequently, in many countries local calls use a shorter code. For calls within the same area, the area code is omitted, and for small single-exchange areas no exchange code is used for own-exchange calls. Coupled with this last arrangement will be a very short code for calls to adjacent exchanges; these arrangements are particularly well suited to rural areas. The disadvantage of short codes is that they change with the location of the calling subscriber, and therefore a short code directory must be available in each exchange area.

10.33 Routeing Calls

The early type of switching equipment, called step-by-step or Strowger, operated by using the dialled pulses to move the selectors to the position corresponding to the digit dialled. In many ways this was an excellent system, but one major disadvantage was that it allowed no flexibility in the way calls were routed – the route was predetermined by the dialled digits. Although some systems were modified to overcome this problem it was not until common-control equipment became widely used that the path a call took between calling and called subscribers could be chosen to allow the most efficient use of the available capacity in the system. The function of the common control in the routing process was to store the dialled digits in a register and then translate them into routing digits which would indicate to the switching system the path to take through the network. This register – translator combination is essential to automatic trunk and international dialling schemes; it allows the telephone administration to manage the system efficiently by changing routes as circumstances alter without having to change subscribers' numbers. This therefore separates the subscriber from the system. The subscriber dials the national number from any location and the register–translator automatically selects an appropriate route.

10.34 Digital Systems

Several factors have acted to push telephony from analogue to digital working. To make such a major change, telephone administrations and equipment manufacturers have been persuaded that it is in digital operation that the future lies, and the decade from the mid-seventies has been characterized in all equipment producing countries by huge investments of manpower and plant in a race to manufacture an efficient digital telecommunications system. There are more manufacturers involved than at any other time and great financial commitments have been made in the hope that a market will be available for the many products being developed.

Traditionally the two basic elements of a telephone system were transmission and

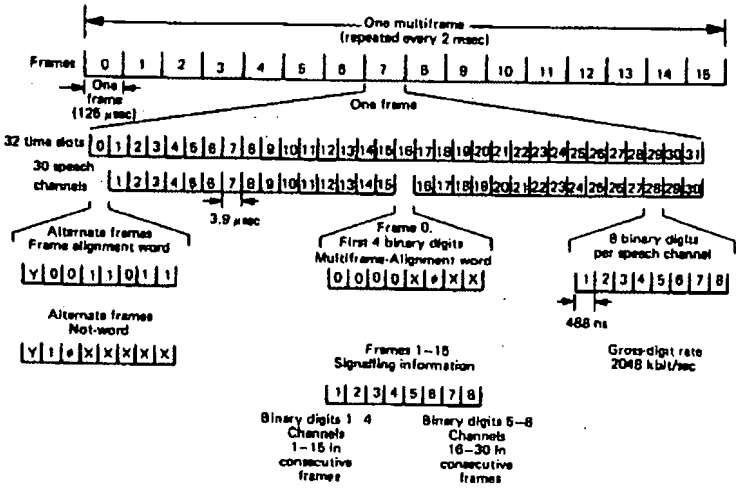


Fig. 10.37 32-frame PCM multiplex frame arrangement: *x*, digits not allocated to any particular function and set to state *one*; *y*, reserved for international use (normally set to state *one*); 0, digits normally *zero* but changed to *one* when loss of frame alignment occurs and/or system-fail alarm occurs (TS0 only) or when loss of multiframe alignment occurs (TS16 only).

switching. However, with digital operation the whole system is considered as an entity.

Following the development of PCM several countries installed PCM links between analogue exchanges. These worked in a basic 24- or 32-channel format with an encoder and decoder. We can see how the 32-channel system is formed with reference to Fig. 10.37. Strictly speaking, it should be referred to as a 30-channel, 32-time-slot system, because two of the time slots contain signalling and synchronizing information, not speech samples. The sampling rate of each speech channel determines the length of each time slot. For telephony the sampling rate used is 8 kHz, which means that the time between adjacent samples of the same channel is 125 μsec. One frame, consisting of 32 time-slots lasts for this time. A time slot is therefore approximately 3.9 μsec long. For reasons that we shall see in a moment, 16 frames are put together to form a multi-frame which has a time span of 2 msec. This is the basic unit of the PCM system applied to telephony.

Within a time slot there is the encoded information about the speech sample for that channel. In most systems it is coded into 256 ($= 2^8$) levels and there are therefore 8 binary digits within each time slot; each digit must be less than 0.48 μsec. long. The technique of PCM is given in detail in Section 3.5.

The channel digits bearing speech information must be sent to the right destination and monitored for release and clear-down. Signals must therefore be associated with each channel and in PCM telephony that is done by allocating a signalling word to each channel once per multi-frame. Sufficient information for signalling can be contained in a 4-digit word so that an 8 digit word can contain two signals. Hence a signalling time slot can contain enough information for two signals.

Figure 10.37 indicates that time slot sixteen (TS16) is used to carry the signalling. In the second frame it carries the signals related to speech channels 0 and 15, in the third frame those for channels 1 and 16 and so on until frame 16 when TS16 has signals for speech channels 14 and 29. The next frame is the first of the following multi-frame and the sequence is repeated.

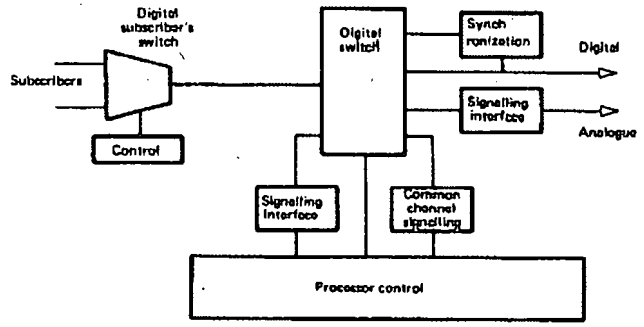


Fig. 10.38 Digital telephone exchange.

TS0 in each frame and TS16 in the first frame, carry synchronization and alignment words to ensure that the transmission and reception of the system is maintained in step.

Modern digital electronics, with its fast logic and complex integrated circuitry, has provided the techniques for producing digital switching and control systems at a cost comparable with analogue, coupled with an ability to provide better facilities, cheaper maintenance and more flexibility. The increasingly widespread use of data in various forms, and the need to send such information over large distances, has also encouraged the development of digital telecommunications and it has led to the intention to integrate together speech and data links.

Digital exchanges consist of the basic components showing in Fig. 10.38. Most of the control of the system is handled by microprocessor devices, singly or in clusters, which are driven by software. Here we are not able to discuss the huge new field of telecommunications software engineering, but it provides the most important challenge in modern system design. The software must be efficient, reliable, secure, understandable and well documented. In theory it affords a degree of flexibility in the operation of the system which is much higher than is possible in hard-wire control. However, such is the complexity of large telephone networks that software development presents the most difficult problems to designers, for it must last for tens of years and although made up of very long programs, it must be able to cope with dramatic changes in hardware technology.

In analogue exchanges the usual figure of merit used is the grade of service, or probability of blocking, but in digital switches the blocking is virtually zero. In these systems the major problems concern delay in the processing of calls caused by the processor units becoming overloaded. The analysis of such problems is very difficult and if simple queueing theory models do not apply resort must be made to computer simulation. One of the difficult tasks of the software engineer is to produce a satisfactory compromise between short efficient programs and those that are longer, more complex and more reliable and secure.

Referring again to Fig. 10.38, look first at the digital switch. It can have many structural forms, depending on the size of the system and the technology used, but as an example we will consider the common time–space–time (TST) configuration. TST is a shorthand for the time-switch, space-switch, time-switch arrangement shown in Fig. 10.39(a).

The time switch is split into several units, each having M PCM links of L channels, as Fig. 10.39(b). Consequently, if the time switch is non-blocking it will have an outlet highway of $N = ML$ time slots. The space switch is square with R inlet highways and R outlet highways. The purpose of the TST unit is to allow a particular call, which occupies a specific channel into one of the time switches, to be connected to a particular outlet channel. Basically the switching is between highways on either side of the space switch. Each highway has N time slots and in order for a particular call to be connected say from $H1$ to $H3$ it must find a time slot which is free in both highways. This slot may not be the same as the required incoming and outgoing slots for the call, and so some time delay, provided by the time switches, is necessary. The method used to produce the delay again depends on

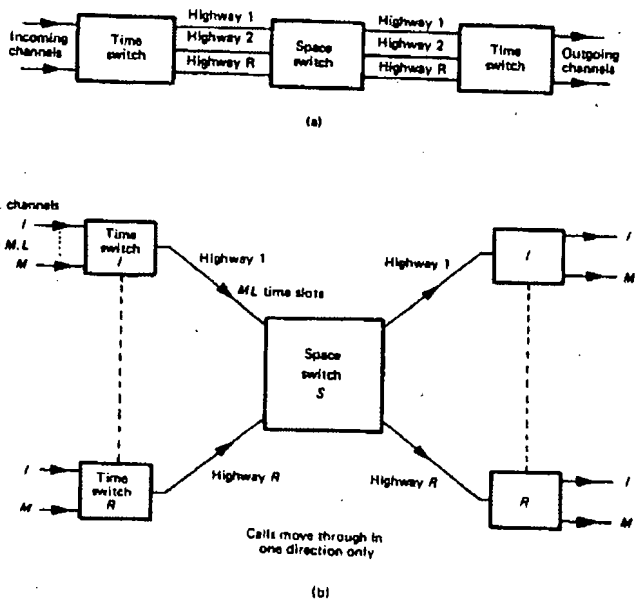


Fig. 10.39 Digital switch: (a) TST block diagram; (b) network representation.

the technology employed, but the delay may be from 1 to 31 time slots depending on the relative positions of the time slots in and out of the unit, and the chosen free time slot in the space switch.

To understand the behaviour of the TST switch in terms of the link systems considered earlier it is important to appreciate that for each time slot the interconnections in the space switch will be different; at each time slot there will be a different set of calls in progress and the connections between the highways will only last for one time slot period then new connections will be established. This can be represented by having N space switches (Fig. 10.40), one for each time slot.

Whether or not blocking occurs in the TST unit depends entirely on the dimensions of the space switch, and since in modern systems switches are comparatively inexpensive they are usually large enough to make blocking negligible. For total non-blocking there must be at least as many outlets as inlets on the time switches, and the space switch highways must have $2N - 1$ time slots, where N is the number of time slots in a link to a time switch.

Digital switches are uni-directional, and that implies that two paths are required to connect two channels X and Y, one for conversation from X to Y and the other

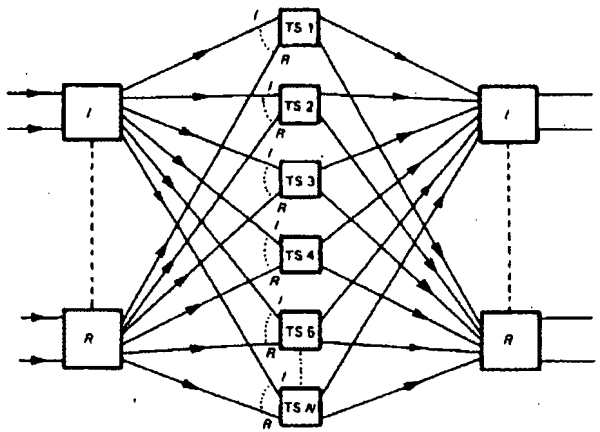


Fig. 10.40 Analogue equivalent of TST switch.

for conversation from Y to X. To reduce the control process, the X to Y slot is chosen according to whatever rules are used by the designer, and the Y to X interconnection is allocated a fixed number of time-slots from it, e.g. one, or half a frame. By this method, if the X to Y connection is available, the Y to X must also be free.

The digital switch just described is situated in a main exchange, forming part of the trunk network. Interconnections between exchanges are made, for the information paths, via PCM links. However, signalling is carried over a common channel, using signalling system CCITT No. 7 as described in Section 10.3. The inter-exchange signals will not only be concerned with setting up calls between exchanges, but with accounting, administration fault diagnosis and maintenance. As described earlier, the CCITT No. 7 system transmits signals as messages that can have variable length. Each message is preceded by labels that identify its origin and destination exchange, the type of message (call handling, fault, etc.) and includes error-detection and acknowledgement bits. If an error is detected in a message, that and all subsequent messages are retransmitted to ensure that the sequence received at the far end is in the correct order.

The error rate has an important bearing on the capacity of the signalling channel; re-transmission of incorrectly received messages obviously takes up time that could be used for new signals and consequently slows down the overall process. The capacity is specified in terms of number of messages per busy hour given that the delay from end to end is not greater than some predetermined value.

On the subscriber side of the digital switch there will be a local unit of some description. For large areas a local digital exchange would be used with mf signalling from subscriber to exchange where conversion to a PCM format would

take place before concentration through a digital switch. Alternatively for very small units no exchange facility would be available, but a simple digital concentrator would be used to take in the analogue channels, convert them to PCM and multiplex them onto a single highway to the nearest local exchange. Calls between subscribers on the same concentrator would then have to pass through the local exchange. Signalling in these PCM links would be on TS16 and the control, software and firmware at the main exchange would convert it to common channel if a trunk call was required.

The introduction of digital systems is rarely a starting point for the telephone system. Usually a system exists and the digital equipment has to be grafted on to it. Whatever method is used, interworking between the old and new systems is required, and one area of difficulty is the interfacing of various analogue signalling systems with the new equipment designed to operate on TS16 or common channel. This interfacing can be a severe problem if there are many existing signalling schemes in a particular network, and the development of satisfactory units can add considerably to system costs.

10.35 Conclusion

In some respects, the whole concept of telecommunications is changing, and with it the services and functions provided by telecommunications operating companies. The very rapid growth of information technology, with its demand for high-speed, large-capacity, data links, with video output, and the introduction of new facilities on telephone exchanges, are both causing manufacturers to rethink the whole philosophy of how communication systems should best be provided. In the next few years the main point at issue will be to decide whether future systems should be fully integrated, or not. If so, should they be based on a digital telephone system, with its centralized switching units, or on one of the many data networks, with distributed control? Conversely, economy and efficiency may dictate an extension of the present hybrid scheme in which both methods co-exist, with perhaps an element of inter-working between them. It is too early in the development of distributed systems to predict what changes will take place in the next decade, but that major redesign will occur is beyond doubt.

References

1. Brockmeyer, B., Halstron, H.L. and Jensen, A., *The Life and Works of A.K. Erlang*, Copenhagen Telephone Company, 1943.
2. Jacobaeus, C., "A study on congestion in link systems", *Ericsson Technics*, No. 48, 1950.
3. Cooper, R.B., *Introduction to Queueing Theory*, Edward Arnold, London, 1981, Chapter 5.
4. Fry, T.C., *Probability and its Engineering Uses*, Van Nostrand Reinhold, Wokingham, 1965.

5. Fishman, G.S., *Principles of Discrete Event Simulation*, Wiley, Chichester, 1978.
6. Flood, J.E., *Telecommunication Networks*, Peter Perigrinus, London, 1975.
7. Bear, D., *Telecommunication Traffic Engineering*, Peter Perigrinus, London, 1976.
8. Hills, M.T., *Telecommunication Switching Principles*, George Allen & Unwin, London, 1979.

Problems

10.1 A traffic-recording machine takes measurements of the number of busy devices in a group every three minutes during the busy hour. If the sum of the devices busy over that period is 600, what is the value of the traffic carried?

Answer: 30 erlangs.

10.2 A loss-system full availability group consists of five devices. If the mean call holding time is 180 sec, and the call intensity is 80 calls/hour, what is the mean load per device?

Answer: 0.8 erlangs.

10.3 In a particular system, it was found that during the busy hour, the average number of calls in progress simultaneously in a certain full availability group of circuits was 15. All circuits were busy for a total of 30 sec during the busy hour. Calculate the traffic offered to the group.

Answer: 15.13 erlangs.

10.4 A group of 8 circuits is offered 6 erlangs of traffic. Find the time congestion of the group, and calculate how much traffic is lost.

Answer: 0.122 erlangs; 0.73 erlangs.

10.5 A ninth circuit is added to the group in Question 10.1. What traffic will it carry?

Answer: 5.55 erlangs.

10.6 A system of six telephones has full availability access to six devices. Find the probability that $1, 2, \ldots, 6$ devices are busy. What is (a) the call, and (b) the time congestion of the system if the carried traffic is 2.4 erlangs?

Answer: (a) 0; (b) 0.004.

10.7 (a) Two erlangs of traffic are fed to three devices. What is the congestion, and how much traffic is lost?
(b) Two erlangs of traffic are fed to one device. The overflow is fed to a

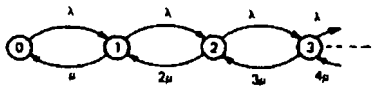
second device, and the overflow from that to a third. What is the overall congestion, and is the value of the traffic lost the same as in (a)?

Answer: (a) 0.2105, 0.421 erlangs; (b) 0.432, No.

10.8 Show that, if the assumption of statistical equilibrium is valid, the probability of a system being in state i is given in terms of the probability that it is in state 0 by

$$[i] = \frac{\prod_{j=0}^{i-1} \lambda_j}{\prod_{j=1}^{i} \mu_j} [0]$$

10.9 The state transition diagram below represents a system with an infinite number of devices subjected to calls arriving at random with fixed mean arrival rate, λ



If $\lambda/\mu = A$, the mean offered traffic, use the birth and death equations, and assume statistical equilibrium, to show that

$$[i] = \frac{A^i \exp(-A)}{i!}$$

10.10 An infinite number of sources feed 5 erlangs of traffic into 8 devices. Find the probability of the network being in each of its possible states, i, and check that $\sum_i [i] = 1$. Plot $[i]$ against i.

Answer: 0.0072, 0.0362, 0.0904, 0.1504, 0.1883, 0.1883, 0.1564, 0.1121, 0.0702.

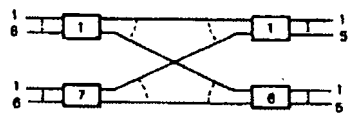
10.11 Repeat question 10 for (i) 2 erlangs, and (ii) 7 erlangs of traffic, noting the variation in both the congestion and the distribution of $[i]$, with increasing traffic.

Answer: (i) 0.1354, 0.2707, 0.2707, 0.1905, 0.0902, 0.0361, 0.0120, 0.0035, 0.0009; (ii) 0.0013, 0.0088, 0.0306, 0.0715, 0.1251, 0.1751, 0.2044, 0.2044, 0.1788.

10.12 A telephone route of n circuits has to carry a normal load of 3 erlangs. If the grade of service must not exceed 0.03, what is the smallest value that n can have? In an emergency, there is a 20% increase in offered traffic. What will be the grade of service in this overload condition?

Answer: $n = 7$, 0.045.

10.13 Draw the chicken diagram equivalent of the two-stage link network shown below, and calculate the internal blocking.

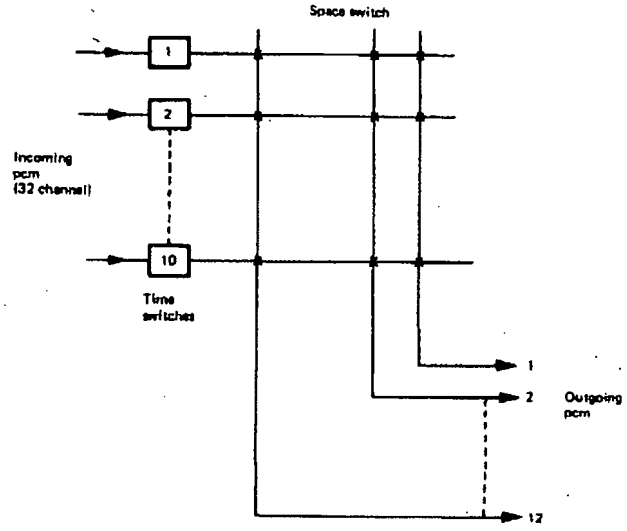


Corresponding outlets from second-stage switches form a route.

Traffic carried per route = 2.5 erlangs
Traffic carried per inlet switch = 1.8 erlangs

Answer: 0.0285.

10.14 A time–space digital switching configuration is shown. What are the functions of the time and space elements in relation to the speech channels on the PCM inlets and outlets.? Draw the analogue equivalent of this switch.



O10 An Open Architecture for Digital Library System and a Plan for its Development - CNRI (1998)

Corporation for National Research Initiatives

Programs/Activities List

Research Programs

- CORDRA
- D-Lib® and D-Lib® Magazine
- DVIA
- Digital Object Architecture
- Digital Object Identifier
- Digital Object Store
- Handle System
- Knowbot Programs
- MEMS Exchange
- Repository Architecture
- Speech and Language

Digital Object Architecture Project

Note: the Digital Object Architecture Project includes the CNRI Repository.

CNRI's program of research and development in digital libraries has a number of inter-related activities that overlap and build upon each other. The work includes development of core technology that is usec in several testbeds and implementation projects, with funding from a variety of sources.

The **Digital Object Architecture Project** continues the architectural work of the DARPA-funded **Computer Science Technical Reports Project** (CSTR).

The project focuses on the development of an infrastructure of services that provide access to distributed and secure digital objects. Digital objects are networked objects that are instantiated by an infrastructure service we call a repository. Digital objects provide access to their content using an extensible and secure dissemination mechanism. Disseminations can be thought of as high level types that are uniquely distinguished by a combination of operations, and types of data the latter are performed on. Disseminations consist of mobile code called Servlet that can be designed, implemented, and registered with the digital object infrastructure by anyone with the proper permissions. Any digital object with the appropriate rights can automatically use registered servlets. This extensible dissemination mechanism enables digital objects to accommodate a wide variety of possible content, from complex to simple, static or dynamic, and from permanent to real time data. Disseminations have few operational limits and enable digital objects to dynamically generate or acquire their content.

Current ongoing research includes the development of dissemination registry, infrastructure searching, security and scalability.

Support for the Digital Object Architecture project is provided by DARPA, the Library of Congress, and the Defense Technical Information Center (DTIC), through DARPA grant MDA972-92-J-1029.

Technology

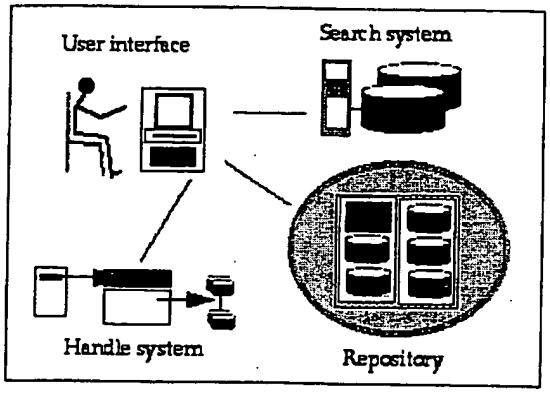


Figure 1

Figure 1 shows the principal system components. CNRI's research concentrates on the concept of digital objects, the Handle System for identifying digital objects, and the Repository for storing them and making them available over the Internet. The Registry is a specialized repository that is used to authenticate digital objects.

The Handle System is a system for providing persistent names for Internet resources. It is a highly reliable, high performance, distributed system.
The Repository provides network based storage and access to digital objects. All access to digital objects passes uses a simple repository access protocol and is subject to access controls established by the manager of the repository.
The Registry is a specialized repository that provides secure registration and authentication of digital objects.

Partners

CORDRA/ADL. CNRI is designing a registration system for the ADL CORDRA project, to be known as the ADL Registry or ADL-R, and deliver it for use at the Defense Technical Information Center (DTIC). This effort will enable the discovery and reuse of learning content held in repositories distributed across the DoD.

Defense Virtual Information Architecture. The Defense Technical Information Center (DTIC), the Defense Advanced Research Projects Agency (DARPA), and CNRI are working to extend and transfer CNRI's Digital Object Architecture work into a prototype digital library implementation.

Additional Information

- Kahn, Robert & Wilensky, Robert. "A framework for distributed digital object services"; *International Journal on Digital Libraries* (2006) 6(2). [**doi:10.1007/s00799-005-0128-x**]. **Reproduced** with permission of the publisher.

- Henry Jerez, Giridhar Manepalli, Christophe Blanchi, and Larry Lannom, "ADL-R: The First Instance of a CORDRA Registry", *D-Lib Magazine*, February 2006. [**doi:10.1045/february2006-**

jerez]

- Giridhar Manepalli, Henry Jerez, and Michael L. Nelson, "FeDCOR: An Institutional CORDRA Registry", *D-Lib Magazine*, February 2006. [**doi:10.1045/february2006-manepalli**]
- Talk by Dr. Robert Kahn on the **Digital Object Architecture and it's importance to the future of the Internet**. [Windows Media Format, 240 MB]
- Christophe Blanchi, Jason Petrone, "**Distributed Interoperable Metadata Registry**", *D-Lib Magazine*, December 2001. [doi:10.1045/december2001-blanchi]
- Christophe Blanchi, Jason Petrone, "**An Architecture for Digital Object Typing**", White Paper, CNRI, September 2001. [hdl:4263537/4096]
- Robert E. Kahn and Vinton G. Cerf, "**What is the Internet (And What Makes It Work)**", prepared by the authors at the request of the *Internet Policy Institute*, December 1999.
- Sun, Sam, "**Handle System Namespace and Service Definition**", TWIST '99, Irvine, California, August 19, 1999.
- "**Managing Access to Digital Information**" Cross Industry Working Team (XIWT), July 1999.
- Sandra Payette, Cornell University; Christophe Blanchi, CNRI; Carl Lagoze, Cornell University; Edward A. Overly, CNRI, "**Interoperability for Digital Objects and Repositories: The Cornell/CNRI Experiments**, *D-Lib Magazine*, May 1999. [doi:10.1045/may99-payette]
- Laurence Lannom. "**Handle System Overview**". ICSTI Forum, No. 30, April 1999.
- William Y. Arms, "A national library for undergraduate science, mathematics, engineering, and technology education: needs, options, and feasibility (technical considerations)". In: **National Research Council, "Developing a national digital library for undergraduate science, mathematics, engineering, and technology education".** Washington D.C.: National Academy Press. 1998.
- "**Implementing Policies for Access Management**" by William Y. Arms, *D-Lib Magazine*, February 1998. [doi:10.1045/february98-arms]
- William Y. Arms, "**Digital Object Identifiers (DOIs) and Clifford Lynch's five questions on identifiers**". ARL Newsletter October 1997.
- "**An Architecture for Information in Digital Libraries**" by William Y. Arms, Christophe Blanchi, Edward A. Overly. *D-Lib Magazine*, February 1997. [doi:10.1045/february97-arms]
- "**Uniform Resource Names: A Progress Report**" by the URN

Implementors. *D-Lib Magazine*, February 1996.
[hdl:cnri.dlib/february96-urn_implementors]

- "A Design for Inter-Operable Secure Object Stores (ISOS)" by Carl Lagoze, Robert McGrath, Ed Overly, Nancy Yeager. Cornell Computer TR95-1558.
- "Implementation Issues in an Open Architecture Framework for Digital Object Services" by Carl Lagoze and David Ely. Cornell Computer Science Technical Report TR95-1540.
- Lagoze, Carl, "**A Secure Repository Design for Digital Libraries**", *D-Lib Magazine*, Decmeber 1995.
[doi:10.1045/december95-lagoze]
- "**Key Concepts in the Architecture of the Digital Library**" by William Y. Arms, *D-Lib Magazine*, July 1995.
[doi:10.1045/july95-arms]
- "**A Framework for Distributed Digital Object Services**" by Robert Kahn and Robert Wilensky, May 1995.
[hdl:4263537/5001]

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G11B 20/00 — A2

(11) International Publication Number: WO 97/43761

(43) International Publication Date: 20 November 1997 (20.11.97)

(21) International Application Number: PCT/US97/08192

(22) International Filing Date: 15 May 1997 (15.05.97)

(30) Priority Data:

60/017,722	15 May 1996 (15.05.96)	US
60/018,132	22 May 1996 (22.05.96)	US
08/689,606	12 August 1996 (12.08.96)	US
08/689,754	12 August 1996 (12.08.96)	US
08/699,712	12 August 1996 (12.08.96)	US
PCT/US96/14262	4 September 1996 (04.09.96)	WO
(34) Countries for which the regional or international application was filed:		US et al.
60/037,931	14 February 1997 (14.02.97)	US

(71) Applicant *(for all designated States except US)*: INTERTRUST TECHNOLOGIES CORP. [US/US]; 460 Oakmead Parkway, Sunnyvale, CA 94086 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: SHEAR, Victor, H. [US/US]; 5203 Battery Lane, Bethesda, MD 20814 (US). SIBERT, Olin, W. [US/US]; 30 Ingleside Road, Lexington, MA 02173-2522 (US). VANWIE, David, M. [US/US]; Apartment 216, 965 E. El Camino Real, Sunnyvale, CA 94087 (US). WEBER, Robert, P. [US/US]; 215 Waverley Street #4, Menlo Park, CA 94025 (US).

(74) Agent: FARIS, Robert, W.; Nixon & Vanderhye P.C., 8th floor, 1100 North Glebe Road, Arlington, VA 22201-4714 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

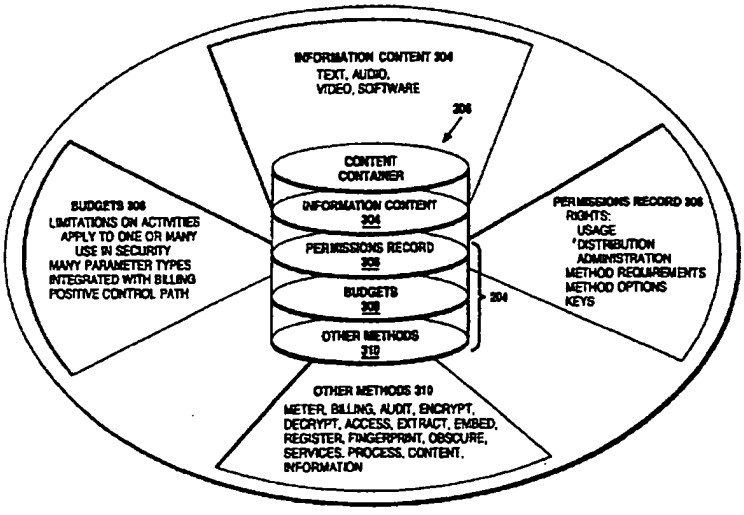
Published

Without international search report and to be republished upon receipt of that report.

(54) Title: CRYPTOGRAPHIC METHODS, APPARATUS AND SYSTEMS FOR STORAGE MEDIAELECTRONIC RIGHTS MANAGEMENT IN CLOSED AND CONNECTED APPLIANCES

(57) Abstract

A rights management arrangement for storage media such as optical digital video disks (DVDs, also called digital versatile disks) provides adequate copy protection in a limited, inexpensive mass-produceable, low-capability platform such as a dedicated home consumer disk player and also provides enhanced, more flexible security techniques and methods when the same media are used with platforms having higher security capabilities. A control object (or set) defines plural rights management rules for instance, price for performance or rules governing redistribution. Low capability platforms may enable only a subset of the control rules such as controls on copying or marking of played material. Higher capability platforms may enable all (or different subsets) of the rules. Cryptographically strong security is provided by encrypting at least some of the information carried by the media and enabling decryption based on the control set and/or other limitations. A secure "software container" can be used to protectively encapsulate (e.g., by cryptographic techniques) various digital property content (e.g., audio, video, game, etc.) and control object (i.e., set of rules) information. A standardized container format is provided for general use on/with various mediums and platforms. In addition, a special purpose container may be provided for DVD medium and appliances (e.g., recorders, players, etc.) that contains DVD program content (digital property) and DVD medium specific rules. The techniques, systems and methods disclosed herein are capable of achieving compatibility with other protection standards, such as for example, CGMA and Matsushita data protection standards adopted for DVDs. Cooperative rights management may also be provided, where plural networked rights management arrangements collectively control a rights management event on one or more of such arrangements.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

CRYPTOGRAPHIC METHODS, APPARATUS AND SYSTEMS FOR STORAGE MEDIA ELECTRONIC RIGHTS MANAGEMENT IN CLOSED AND CONNECTED APPLIANCES

Cross-Reference to Related Applications and Patents

The specifications and drawings of the following prior, commonly assigned published patent specifications are incorporated by reference into this patent specification:

PCT Publication No. WO 96/27155 dated 6 September 1996 entitled "Systems And Methods For Secure Transaction Management And Electronic Rights Protection", which is based on PCT application no. PCT/US96/02303 filed 13 February 1996 and U.S. patent application serial no. 08/388,107 of Ginter et al. entitled filed on February 13, 1995 (hereinafter "Ginter et al");

U.S. Patent No 4,827,508 entitled "Database Usage Metering and Protection System and Method" dated May 2, 1989;

U.S. Patent No. 4,977,594 entitled "Database Usage Metering and Protection System and Method" dated December 11, 1990;

U.S. Patent No. 5,050,213 entitled "Database Usage Metering and Protection System and Method" dated September 17, 1991; and

U.S. Patent No. 5,410,598 entitled "Database Usage Metering and Protection System and Method" dated April 25, 1995; and

European Patent No.EP 329681 entitled "Database Usage Metering and Protection System and Method" dated January 17, 1996.

In addition, the specifications and drawings of the following commonly-assigned prior-filed patent specifications are incorporated by reference into this patent application:

PCT Application No. PCT/US96/14262 filed 4 September 1996 entitled "Trusted Infrastructure Support Systems, Methods And Techniques For Secure Electronic Commerce, Electronic Transactions, Commerce Process Control And Automation, Distributed Computing, And Rights Management," which corresponds to U.S. patent application serial no. 08/699,712 filed on August 12, 1996 (hereinafter "Shear et al.");

PCT Application No. _____ filed _____, 1997 entitled "Steganographic Techniques For Securely Delivering Electronic Digital Rights Management Control Information Over Insecure Communications Channels," which corresponds to U.S. patent application serial no. 08/689,606 of Van Wie and Weber filed on August 12, 1996 (hereinafter "Van Wie and Weber"); and

PCT Application No. _____ filed _____, 1997 based on U.S. Patent Application serial no.08/689,754 entitled "Systems and Methods Using Cryptography To Protect Secure Computing Environments," of Sibert and Van Wie filed on August 12, 1996 (hereinafter "Sibert and Van Wie").

FIELD OF THE INVENTION

This invention relates to information protection techniques using cryptography, and more particularly to techniques using cryptography for managing rights to information stored on portable media -- one example being optical media such as Digital Video Disks (also known as "Digital Versatile Disks" and/or "DVDs"). This invention also relates to information protection and rights management techniques having selectable applicability depending upon, for example, the resources of the device being

used by the consumer (e.g., personal computer or standalone player), other attributes of the device (such as whether the device can be and/or typically is connected to an information network ("connected" versus "unconnected")), and available rights. This invention further relates, in part, to cooperative rights management -- where plural networked rights management arrangements collectively control a rights management event on one or more of such arrangements. Further, important aspects of this invention can be employed in rights management for electronic information made available through broadcast and/or network downloads and/or use of non-portable storage media, either independent of, or in combination with portable media.

BACKGROUND OF THE INVENTION

The entertainment industry has been transformed by the pervasiveness of home consumer electronic devices that can play video and/or audio from pre-recorded media. This transformation began in the early 1900s with the invention of the phonograph—which for the first time allowed a consumer to listen to his or her favorite band, orchestra or singer in his or her home whenever he or she wishes. The availability of inexpensive video

cassette recorders/players beginning in the early 1980s brought about a profound revolution in the movie and broadcast industries, creating an entirely new home consumer market for films, documentaries, music videos, exercise videos, etc.

The entertainment industry has long searched for optimal media for distributing content to home consumers. The original phonograph cylinders distributed by Thomas Edison and other phonograph pioneers had the advantage that they were difficult to copy, but suffered from various disadvantages such as high manufacturing costs, low resistance to breakage, very limited playback time, relatively low playback quality, and susceptibility to damage from wear, scratching or melting. Later-developed wax and vinyl disks could hold more music material but suffered from many of the same disadvantages. Magnetic tapes, on the other hand, could be manufactured very inexpensively and could hold a large amount of program material (e.g., 2, 4 or even 6 hours of video and/or audio). Such magnetic tapes could reproduce program material at relatively high quality, and were not as susceptible to damage or wearing out. However, despite the many clear advantages that magnetic tape provides over other media, the

entertainment industry has never regarded it as an ideal or optimum medium because of its great susceptibility to copying.

Magnetic tape has the very flexible characteristic that it can be relatively easily recorded on. Indeed, the process for recording a magnetic tape is nearly as straightforward as that required for playing back pre-recorded content. Because of the relative ease by which magnetic tape can be recorded, home consumer magnetic tape equipment manufacturers have historically provided dual mode equipment that can both record and play back magnetic tapes. Thus, home audio and video tape players have traditionally had a "record" button that allows a consumer to record his or her own program material on a blank (un-recorded) magnetic tape. While this recording ability has given consumers additional flexibility (e.g., the ability to record a child's first words for posterity, and the ability to capture afternoon soap operas for evening viewing), it has unfortunately also been the foundation of an illegal multi-billion dollar content pirating industry that produces millions of illegal, counterfeit copies every year. This illegal pirating operation—which is international in scope—leeches huge amounts of revenue every year from the world's major

entertainment content producers. The entertainment industry must pass along these losses to honest consumers—resulting in higher box office prices, and higher video and audio tape sales and rental prices.

In the mid 1980s, the audio entertainment industry developed the optical compact disk as an answer to some of these problems. The optical compact disk—a thin, silvery plastic platter a few inches in diameter—can hold an hour or more of music or other audio programming in digital form. Such disks were later also used for computer data. The disk can be manufactured very inexpensively, and provides extremely high quality playback that is resistant to noise because of the digital techniques used to record and recover the information. Because the optical disk can be made from plastic, it is light weight, virtually unbreakable, and highly resistant to damage from normal consumer handling (unlike the prior vinyl records that were easily scratched or worn down even by properly functioning phonographs). And, because recording on an optical disk is, so far, significantly more difficult than playing back an optical disk, home consumer equipment providing both recording and playback capabilities is unlikely, in

the near future, to be as cost-effective as play-only equipment—greatly reducing the potential for illicit copying. Because of these overwhelming advantages, the music industry has rapidly embraced the new digital compact disk technology—virtually replacing older audio vinyl disk media within the space of a few short years.

Indeed, the threat of widespread and easy unauthorized copying in the absence of rights management technologies apparently has been an important contributing factor to the demise of digital audio tape (DAT) as a media for music distribution and, more importantly, home audio recording. Rightsholders in recorded music vigorously opposed the widespread commercialization of inexpensive DAT technology that lacked rights management capabilities since the quality of the digital recording was completely faithful to the digital source on, for example, music CDs. Of course, the lack of rights management was not the only factor at work, since compared with optical media, tape format made random access difficult, for example, playing songs out of sequence.

The video entertainment industry is on the verge of a revolution similar to that wrought by music CDs based on movies in digital format distributed on high capacity read-only optical media. For example, digital optical disk technology has advanced to the point where it is now possible to digitally record, among other things, a full length motion picture (plus sound) on one side of a 5" plastic optical disk. This same optical disk can accommodate multiple high-quality digital audio channels (e.g., to record multi-channel "sensurround" sound for home theaters and/or to record film dialog in multiple different languages on the same disk). This same technology makes it possible to access each individual frame or image of a movie for still image reproduction or—even more exciting—to provide an unprecedented "random access" playback capability that has never before existed in home consumer equipment. This "random access" playback could be used, for example, to delete violence, foul language or nudity at time of playback so that parents could select a "PG" playback version of an "R" rated film at the press of a button. The "random access" capability also has exciting possibilities in terms of allowing viewers to interact with the pre-recorded content (e.g.,

allowing a health enthusiast to select only those portions of an exercise video helpful to a particular day's workout). See, for example, "Applications Requirements for Innovative Video Programming," DVD Conference Proceedings (Interactive Multimedia Association, 19-20 October 1995, Sheraton Universal Hotel, Universal City, California).

Non-limiting examples of the DVD family of optical media include:

- DVD (Digital Video Disk, Digital Versatile Disk), a non-limiting example of which includes consumer appliances that play movies recorded on DVD disks;

- DVD-ROM (DVD-Read Only Memory), a non-limiting example of which includes a DVD read-only drive and disk connected to a computer or other appliance;

- DVD-RAM (DVD Random Access Memory), a non-limiting example of which includes a read/write drive and optical media in, for example, consumer appliances for home recording and in a computer or other appliance

for the broadest range of specific applications; and

- Any other high capacity optical media presently known or unknown.

"DVDs" are, of course, not limited to use with movies. Like CDs, they may also be used for other kinds of information, for example:

- sound recordings
- software
- databases
- games
- karaoke
- multimedia
- distance learning
- documentation
- policies and manuals

- any kind of digital data or other information
- any combination of kinds of digital data or other information
- any other uses presently known or unknown.

The broad range of DVD uses presents a technical challenge: how can the information content distributed on such disks, which might be any kind or combination of video, sound, or other data or information broadly speaking, be adequately protected while preserving or even maximizing consumer flexibility? One widely proposed requirement for the new technology(mainly within the context of video), is, to the extent copying is permitted at all, to either: (a) allow a consumer to make a first generation copy of the program content for their own use, but prevent the consumer from making "copies of copies", or multi-generational copies of a given property (thus keeping honest people honest); or (b) to allow unlimited copying for those properties that rightsholders do not wish to protect against copying, or which consumers have made themselves.

However, providing only such simplistic and limited copy protection in a non-extensible manner may turn out to be extremely shortsighted—since more sophisticated protection and/or rights management objectives (e.g., more robust and selective application of copy protection and other protection techniques, enablement of pay-per-view models, the ability of the consumer to make use of enhanced functionality such as extracting material or interactivity upon paying extra charges, and receiving credit for redistribution, to name a few) could be very useful now or in the future. Moreover, in optimally approaching protection and rights management objectives, it is extremely useful to take differing business opportunities and threats into account that may relate to information delivered via DVD media, for example, depending upon available resources of the device and/or whether the device is connected or unconnected.

More sophisticated rights management capabilities will also allow studios and others who have rights in movies and/or sound recordings to better manage these important assets, in one example, to allow authorized parties to repurpose pieces of digital film, video and/or audio, whether specific and/or arbitrary pieces,

to create derivative works, multimedia games, in one non-limiting example. Solutions proposed to date for protecting DVD content have generally focused solely on limited copy protection objectives and have failed to adequately address or even recognize more sophisticated rights management objectives and requirements. More specifically, one copy protection scheme for the initial generation of DVD appliances and media is based on an encryption method developed initially by Matsushita and the simple CGMA control codes that indicate permitted copying: a one-generation copy, no copies, or unlimited copying.

SUMMARY OF THE INVENTIONS

Comprehensive solutions for protecting and managing information in systems that incorporate high capacity optical media such as DVD require, among other things, methods and systems that address two broad sets of problems: (a) digital to analog conversion (and vice versa); and (b) the use of such optical media in both connected and unconnected environments. The inventions disclosed herein address these and other problems. For example, in the context of analog to digital conversion (and vice versa), it is contemplated that, in accordance with the present

inventions, at least some of the information used to protect properties and/or describe rights management and/or control information in digital form could also be carried along with the analog signal. Devices that convert from one format and/or medium to another can, for example, incorporate some or all of the control and identifying information in the new context(s), or at least not actively delete such information during the conversion process. In addition, the present inventions provide control, rights management and/or identification solutions for the digital realm generally, and also critically important technologies that can be implemented in consumer appliances, computers, and other devices. One objective of the inventions is to provide powerful rights management techniques that are useful in both the consumer electronics and computer technology markets, and that also enable future evolution of technical capabilities and business models. Another non-limiting objective is to provide a comprehensive control, rights management and/or identification solution that remains compatible, where possible, with existing industry standards for limited function copy protection and for encryption.

The present inventions provide rights management and protection techniques that fully satisfy the limited copy protection objectives currently being voiced by the entertainment industry for movies while also flexibly and extensibly accommodating a wide range of more sophisticated rights management options and capabilities.

Some important aspects of the present inventions (that are more fully discussed elsewhere in this application) include:

- Selection of control information associated with information recorded on DVD media (for example, rules and usage consequence control information, that comprise non-limiting example elements of a Virtual Distribution Environment (VDE)) that is based at least in part on class of appliance, for example, type of appliance, available resources and/or rights;

- Enabling such selected control information to be, at least in part, a subset of control information used on other appliances and/or classes of appliance, or completely different control information;

- Protecting information output from a DVD device, such as applying rights management techniques disclosed in Ginter et al. and the present application to the signals transmitted using an IEEE 1394 port (or other serial interface) on a DVD player;

- Creation of protected digital content based on an analog source;

- Reflecting differing usage rights and/or content availability in different countries and/or regions of the world;

- Securely managing information on DVD media such that certain portions may be used on one or more classes of appliance (e.g., a standalone DVD player), while other portions may be used on the same or different classes of appliance (e.g., a standalone DVD player or a PC);

- Securely storing and/or transmitting information associated with payment, auditing, controlling and/or otherwise managing content recorded on DVD media, including techniques related to those disclosed in Ginter et al. and in Shear et al.;

- Updating and/or replacing encryption keys used in the course of appliance operation to modify the scope of information that may be used by appliances and/or classes of appliances;

- Protecting information throughout the creation, distribution, and usage process, for example, by initially protecting information collected by a digital camera, and continuing protection and rights management through the editing process, production, distribution, usage, and usage reporting.

- Allowing "virtual rights machines," consisting of multiple devices and/or other systems that participate and work together in a permanently or in a temporarily connected network to share some or all of the rights management for a single and/or multiple nodes including, for example, allowing resources available in plural such devices and/or other systems, and/or rights associated with plural parties and/or groups using and/or controlling such devices and/or other systems, to be employed in concert (according to rights related rules and controls) so as to govern one or more electronic

events on any one or more of such devices and/or other systems, such event governance including, for example: viewing, editing, subsetting, anthologizing, printing, copying, titling, extracting, saving, and/or redistributing rights protected digital content.

- Allowing for the exchange of rights among peer-to-peer relating devices and/or other systems, wherein such devices and/or other systems participate in a temporary or permanently connected network, and wherein such rights are bartered, sold for currency, and/or otherwise exchanged for value and/or consideration where such value and/or consideration is exchanged between such peer-to-peer participating commercial and/or consumer devices and/or other systems.

General Purpose DVD/Cost-effective Large Capacity Digital Media Rights Protection and Management

The inventions described herein can be used with any large capacity storage arrangement where cost-effective distribution media is used for commercial and/or consumer digital information delivery and DVD, as used herein, should be read to include any such system.

Copy protection and rights management are important in practical DVD systems and will continue to be important in other large capacity storage, playback, and recording systems, presently known or unknown, in the future. Protection is needed for some or all of the information delivered (or written) on most DVD media. Such protection against copying is only one aspect of rights management. Other aspects involve allowing rightsholders and others to manage their commercial interests (and to have them enforced, potentially at a distance in time and/or space) regardless of distribution media and/or channels, and the particular nature of the receiving appliance and/or device. Such rights management solutions that incorporate DVD will become even more significant as future generations of recordable DVD media and appliances come to market. Rightsholders will want to maintain and assert their rights as, for example, video, sound recordings, and other digital properties are transmitted from one device to another and as options for recording become available in the market.

The apparent convergence between consumer appliances and computers, increasing network and modem speeds, the declining cost of computer power and bandwidth, and the

increasing capacity of optical media will combine to create a world of hybrid business models in which digital content of all kinds may be distributed on optical media played on at least occasionally connected appliances and/or computers, in which the one-time purchase models common in music CDs and initial DVD movie offerings are augmented by other models, for example, lease, pay per view, and rent to own, to name just few. Consumers may be offered a choice among these and other models from the same or different distributors and/or other providers. Payment for use may happen over a network and/or other communications channel to some payment settlement service. Consumer usage and audit information may flow back to creators, distributors, and/or other participants. The elementary copy protection technologies for DVD now being introduced cannot support these and other sophisticated models.

As writable DVD appliances and media become available, additional hybrid models are possible, including, for example, the distribution of digital movies over satellite and cable systems. Having recorded a movie, a consumer may elect a lease, rental, pay-per-view, or other model if available. As digital television

comes to market, the ability of writable DVDs to make faithful copies of on-air programming creates additional model possibilities and/or rights management requirements. Here too, simplistic copy protection mechanisms currently being deployed for the initial read-only DVD technologies will not suffice.

Encryption Is A Means, Not An End

Encryption is useful in protecting intellectual properties in digital format, whether on optical media such as DVD, on magnetic media such as disk drives, in the active memory of a digital device and/or while being transmitted across computer, cable, satellite, and other kinds of networks or transmission means. Historically, encryption was used to send secret messages. With respect to DVD, a key purpose of encryption is to require the use of a copy control and rights management system in order to ensure that only those authorized to do so by rightsholders can indeed use the content.

But encryption is more of a means, rather than an end. A central issue is how to devise methods for ensuring, to the maximal extent possible, that only authorized devices and parties can decrypt the protected content and/or otherwise use information

only to the extent permitted by the rightsholder(s) and/or other relevant parties in the protected content.

The Present Inventions

The present inventions provide powerful right management capabilities. In accordance with one aspect provided by the present invention, encrypted digital properties can be put on a DVD in a tamper-resistant software "container" such as, for example, a "DigiBox" secure container, together with rules about "no copy" and/or "copy" and/or "numbers of permitted copies" that may apply and be enforced by consumer appliances. These same rules, and/or more flexible and/or different rules, can be enforced by computer devices or other systems that may provide more and/or different capabilities (e.g., editing, excerpting, one or more payment methods, increased storage capability for more detailed audit information, etc.). In addition, the "software container" such as for example, a "DigiBox" secure container, can store certain content in the "clear" (that is, in unencrypted form). For example, movie or music titles, copyright statements, audio samples, trailers, and/or advertising can be stored in the clear and/or could be displayed by any appropriate application or

device. Such information could be protected for authenticity (integrity) when available for viewing, copying, and/or other activities. At the same time, valuable digital properties of all kinds-film, video, image, text, software, and multimedia— may be stored at least partially encrypted to be used only by authorized devices and/or applications and only under permitted, for example rightsholder-approved, circumstances.

Another aspect provided in accordance with the present invention (in combination with certain capabilities disclosed in Ginter et al.) is that multiple sets of rules could be stored in the same "container" on a DVD disk. The software then applies rules depending on whether the movie, for example, was to be played by a consumer appliance or computer, whether the particular apparatus has a backchannel (e.g., an on-line connection), the national and/or other legal or geographic region in which the player is located and/or the movie is being displayed, and/or whether the apparatus has components capable of identifying and applying such rules. For example, some usage rules may apply when information is played by a consumer device, while other rules may apply when played by a computer. The choice of rules

may be left up to the rightsholder(s) and/or other participants-- or some rules may be predetermined (e.g., based on the particular environment or application). For example, film rightsholders may wish to limit copying and ensure that excerpts are not made regardless of the context in which the property is played. This limitation might be applied only in certain legal or geographic areas. Alternatively, rightsholders of sound recordings may wish to enable excerpts of predetermined duration (e.g., no more than 20 seconds) and that these excerpts are not used to construct a new commercial work. In some cases, governments may require that only "PG" versions of movies and/or the equivalent rating for TV programs may be played on equipment deployed in their jurisdiction, and/or that the applicable taxes, fees and the like are automatically calculated and/or collected if payments related to content recorded on DVD is requested and/or performed (e.g., pay-per-use of a movie, game, database, software product, etc.; and/or orders from a catalog stored at least in part on DVD media, etc.).

In a microprocessor controlled (or augmented) digital consumer appliance, such rules contemplated by the present

inventions can be enforced, for example, without requiring more than a relatively few additions to a central, controlling microprocessor (or other CPU, a IEEE 1394 port controller, or other content handling control circuitry), and/or making available some ROM or flash memory to hold the necessary software. In addition, each ROM (or flash or other memory, which such memory may be securely connected to, or incorporated into, such control circuitry in a single, manufactured component) can, in one example, contain one or more digital documents or "certificate(s)" that uniquely identifies a particular appliance, individual identity, jurisdiction, appliance class(es), and/or other chosen parameters. An appliance can, for example, be programmed to send a copy of a digital property to another digital device only in encrypted form and only inside a new, tamper-resistant "software container." The container may also, for example, carry with it a code indicating that it is a copy rather than an original that is being sent. The device may also put a unique identifier of a receiving device and/or class of devices in the same secure container. Consequently, for example, in one particular arrangement, the copy may be playable only on the intended receiving device,

class(es) of devices, and/or devices in a particular region in one non-limiting example and rights related to use of such copy may differ according to these and/or other variables.

The receiving device, upon detecting that the digital property is indeed a copy, can, for example, be programmed not to make any additional copies that can be played on a consumer device and/or other class(es) of devices. If a device detects that a digital property is about to be played on a device and/or other class(es) of devices other than the one it was intended for, it can be programmed to refuse to play that copy (if desired).

The same restrictions applied in a consumer appliance can, for example, be enforced on a computer equipped to provide rights management protection in accordance with the present inventions. In this example, rules may specify not to play a certain film and/or other content on any device other than a consumer appliance and/or classes of appliances, for example. Alternatively, these same powerful capabilities could be used to specify different usage rules and payment schemes that would apply when played on a computer (and/or in other appliances and/or classes of appliances), as the rightsholder(s) may desire, for example,

different pricing based upon different geographic or legal locales where content is played.

In addition, if "backchannels" are present—for example, set-top boxes with bi-directional communications or computers attached to networks—the present inventions contemplate electronic, independent delivery of new rules if desired or required for a given property. These new rules may, for example, specify discounts, time-limited sales, advertising subsidies, and/or other information if desired. As noted earlier, determination of these independently delivered rules is entirely up to the rightsholder(s) and/or others in a given model.

The following are two specific examples of a few aspects of the present invention discussed above:

1. An Analog To Digital Copying Example

a) Bob has a VHS tape he bought (or rented) and wants to make a copy for his own use. The analog film has copy control codes embedded so that they do not interfere with the quality of the signal. Bob has a writable DVD appliance

that is equipped to provide rights management protection in accordance with the present invention. Bob's DVD recorder detects the control codes embedded in the analog signal (for example, such recorder may detect watermarks and/or fingerprints carrying rights related control and/or usage information), creates a new secure container to hold the content rules and describe the encoded film, and creates new control rules (and/or delivers to a secure VDE system for storage and reporting certain usage history related information such as user name, time, etc.) based on the analog control codes and/or other information it detected and that are then placed in the DigiBox and/or into a secure VDE installation data store such as a secure data base. Bob can play that copy back on his DVD appliance whenever he chooses.

b) Bob gives the DVD disk he recorded to Jennifer who wishes to play it on computer that has a DVD drive. Her computer is equipped to provide rights management protection in accordance with the present invention. Her computer opens the "DigiBox," detects that this copy is being used on a device different from the one that recorded it (an unauthorized device) and refuses to play the copy.

c) Bob gives the DVD disk to Jennifer as before, but now Jennifer contacts electronically a source of new rules and usage consequences, which might be the studio, a distributor, and/or a rights and permissions clearinghouse, (or she may have sufficient rights already on her player to play the copy). The source sends a DigiBox container to Jennifer with rules and consequences that permit playing the movie on her

computer while at the same time charging her for use, even though the movie was recorded on DVD by Bob rather than by the studio or other value chain participant.

2. A Digital To Analog Copying Example

a) Jennifer comes home from work, inserts a rented or owned DVD into a player connected to, or an integral part of her TV, and plays the disk. In a completely transparent way, the film is decrypted, the format is converted from digital to analog, and displayed on her analog TV.

b) Jennifer wishes to make a copy for her own use. She plays the film on an DVD device incorporating rights management protection in accordance with the present invention, that opens the DigiBox secure container, accesses the control information, and decrypts the film.

She records the analog version on her VCR which records a high-quality copy.

c) Jennifer gives the VCR copy to Doug who wishes to make a copy of the analog tape for his own use, but the analog control information forces the recording VCR to make a lower-quality copy, or may prevent copying. In another non-limiting example, more comprehensive rights management information may be encoded in the analog output using the methods and/or systems described in more detail in the above referenced Van Wie and Weber patent application.

In accordance with one aspect provided by this invention, the same portable storage medium, such as a DVD, can be used with a range of different, scaled protection environments providing different protection capabilities. Each of the different environments may be enabled to use the information carried by the portable storage medium based on rights management techniques and/or capabilities supported by the particular environment. For

example, a simple, inexpensive home consumer disk player may support copy protection and ignore more sophisticated and complex content rights the player is not equipped to enable. A more technically capable and/or secure platform (e.g., a personal computer incorporating a secure processing component possibly supported by a network connection, or a "smarter" appliance or device) may, for example, use the same portable storage medium and provide enhanced usage rights related to use of the content carried by the medium based on more complicated rights management techniques (e.g., requiring payment of additional compensation, providing secure extraction of selected content portions for excerpting or anthologizing, etc.). For example, a control set associated with the portable storage medium may accommodate a wide variety of different usage capabilities—with the more advanced or sophisticated uses requiring correspondingly more advanced protection and rights management enablement found on some platforms and not others. Lower-capability environments can, as another example, ignore (or not enable or attempt to use) rights in the control set that they don't understand, while higher-capability environments (having awareness of the

overall capabilities they provide), may, for example, enable the rights and corresponding protection techniques ignored by the lower-capability environments.

In accordance with another aspect provided by the invention, a media- and platform-independent security component can be scaled in terms of functionality and performance such that the elementary rights management requirements of consumer electronics devices are subsets of a richer collection of functionality that may be employed by more advanced platforms. The security component can be either a physical, hardware component, or a "software emulation" of the component. In accordance with this feature, an instance of medium (or more correctly, one version of the content irrespective of media) can be delivered to customers independently of their appliance or platform type with the assurance that the content will be protected. Platforms less advanced in terms of security and/or technical capabilities may provide only limited rights to use the content, whereas more advanced platforms may provide more expansive rights based on correspondingly appropriate security conditions and safeguards.

In accordance with a further aspect provided by the present invention, mass-produced, inexpensive home consumer DVD players (such as those constructed, for example, with minimum complexity and parts count) can be made to be compatible with the same DVDs or other portable storage media used by more powerful and/or secure platforms (such as, for example, personal computers) without degrading advanced rights management functions the storage media may provide in combination with the more powerful and/or secure platforms. The rights management and protection arrangement provided and supported in accordance with this aspect of the invention thus supports inexpensive basic copy protection and can further serve as a commercial convergence technology supporting a bridging that allows usage in accordance with rights of the same content by a limited resource consumer device while adequately protecting the content and further supporting more sophisticated security levels and capabilities by (a) devices having greater resources for secure rights management, and/or (b) devices having connectivity with other devices or systems that can supply further secure rights management resources. This aspect of the invention allows

multiple devices and/or other systems that participate and work together in a permanently or temporarily connected network to share the rights management for at least one or more electronic events (e.g., managed through the use of protected processing environments such as described in Ginter et al.) occuring at a single, or across multiple nodes and further allows the rights associated with parties and/or groups using and/or controlling such multiple devices and/or other systems to be employed according to underlying rights related rules and controls, this allowing, for example, rights available through a corporate executive's device to be combined with or substitute for, in some manner, the rights of one or more subordinate corporate employees when their computing or other devices of these parties are coupled in a temporary networking relationship and operating in the appropriate context. In general, this aspect of the invention allows distributed rights management for DVD or otherwise packaged and delivered content that is protected by a distributed, peer-to-peer rights management.Such distributed rights management can operate whether the DVD appliance or other electronic information usage device is participating

permanently or temporarily connected network and whether or not the relationships among the devices and/or other systems participating in the distributed rights management arrangement are relating temporarily or have a more permanent operating relationship. In this way, the same device may have different rights available depending on the context in which that device is operating (e.g., in a corporate environment such as in collaboration with other individuals and/or with groups, in a home environment internally and/or in collaboration with external one or more specified individuals and/or other parties, in a retail environment, in a classroom setting as a student where a student's notebook might cooperate in rights management with a classroom server and/or instructor PC, in a library environment where multiple parties are collaboratively employing differing rights to use research materials, on a factory floor where a hand held device works in collaboration with control equipment to securely and appropriately perform proprietary functions, and so on).

For example, coupling a limited resource device arrangement, such as a DVD appliance, with an inexpensive network computer (NC), or a personal computer (PC), may allow

an augmenting (or replacing) of rights management capabilities and/or specific rights of parties and/or devices by permitting rights management to be a result of a combination of some or all of the rights and/or rights management capabilities of the DVD appliance and those of an Network or Personal Computer (NC or PC). Such rights may be further augmented, or otherwise modified or replaced by the availability of rights management capabilities provided by a trusted (secure) remote network rights authority.

These aspects of the present invention can allow the same device, in this example a DVD appliance, to support different arrays, e.g., degrees, of rights management capabilities, in disconnected and connected arrangements and may further allow available rights to result from the availability of rights and/or rights management capabilities resulting from the combination of rights management devices and/or other systems. This may include one or more combinations of some or all of the rights available through the use of a "less" secure and/or resource poor device or system which are augmented, replaced, or otherwise modified through connection with a device or system that is

"more" or "differently" secure and/or resource rich and/or possesses differing or different rights, wherein such connection employs rights and/or management capabilities of either and/or both devices as defined by rights related rules and controls that describe a shared rights management arrangement.

In the latter case, connectivity to a logically and/or physically remote rights management capability can expand (by, for example, increasing the available secure rights management resources) and/or change the character of the rights available to the user of the DVD appliance or a DVD appliance when such device is coupled with an NC, personal computer, local server, and/or remote rights authority. In this rights augmentation scenario, additional content portions may be available, pricing may change, redistribution rights may change (e.g., be expanded), content extraction rights may be increased, etc.

Such "networking rights management" can allow for a combination of rights management resources of plural devices and/or other systems in diverse logical and/or physical relationships, resulting in either greater or differing rights through the enhanced resources provided by connectivity with one or more

"remote" rights authorities. Further, while providing for increased and/or differing rights management capability and/or rights, such a connectivity based rights management arrangement can support multi-locational content availability, by providing for seamless integration of remotely available content, for example, content stored in remote, Internet world wide web-based, database supported content repositories, with locally available content on one or more DVD discs.

In this instance, a user may experience not only increased or differing rights but may use both local DVD content and supplementing content (i.e., content that is more current from a time standpoint, more costly, more diverse, or complementary in some other fashion, etc.). In such an instance, a DVD appliance and/or a user of a DVD appliance (or other device or system connected to such appliance) may have the same rights, differing, and/or different rights applied to locally and remotely available content, and portions of local and remotely available content may themselves be subject to differing or different rights when used by a user and/or appliance. This arrangement can support an overall, profound increase in user content opportunities that are seamlessly

integrated and efficiently available to users in a single content searching and/or usage activity by exploiting the rights management and content resources of plural, connected arrangements.

Such a rights augmenting remote authority may be directly coupled to a DVD appliance and/or other device by modem, or directly or indirectly coupled through the use of an I/O interface, such as a serial 1394 compatible controller (e.g., by communicating between a 1394 enabled DVD appliance and a local personal computer that functions as a smart synchronous or asynchronous information communications interface to such one or more remote authorities, including a local PC or NC or server that serves as a local rights management authority augmenting and/or supplying the rights management in a DVD appliance).

In accordance with yet another aspect provided by this invention, rights provided to, purchased, or otherwise acquired by a participant and/or participant DVD appliance or other system can be exchanged among such peer-to-peer relating devices and/or other systems through the use of one or more permenantly or temporarily networked arrangments. In such a case, rights may be

bartered, sold, for currency, otherwise exchanged for value, and/or loaned so long as such devices and/or other systems participate in a rights management system, for example, such as the Virtual Distribution Environment described in Ginter, et al., and employ rights transfer and other rights management capabilities described therein. For example, this aspect of the present invention allows parties to exchange games or movies in which they have purchased rights. Continuing the example, an individual might buy some of a neighbor's usage rights to watch a movie, or transfer to another party credit received from a game publisher for the successful superdistribution of the game to several acquaintances, where such credit is transferred (exchanged) to a friend to buy some of the friend's rights to play a different game a certain number of times, etc.In accordance with yet another aspect provided by this invention, content carried by a portable storage medium such as a DVD is associated with one or more encryption keys and a secure content identifier. The content itself (or information required to use the content) is at least partially cryptographically encrypted—with associated decryption keys being required to decrypt the content before the content can be

used. The decryption keys may themselves be encrypted in the form of an encrypted key block. Different key management and access techniques may be used, depending on the platform.

In accordance with still yet another aspect provided by this invention, electronic appliances that "create" digital content (or even analog content) —e.g., a digital camera/video recorder or audio recorder—can be readily equipped with appropriate hardware and/or software so as to produce content that is provided within a secure container at the outset. For example, content recorded by a digital camera could be immediately packaged in a secure container by the camera as it is recording. The camera could then output content already packaged in a secure container(s). This could preclude the need to encapsulate the content at a later point in time or at a later production stage, thus, saving at least one production-process step in the overall implementation of electronic rights management in accordance with the present invention. Moreover, it is contemplated that the very process of "reading" content for use in the rights management environment might occur at many steps along a conventional production and distribution process (such as during editing and/or

the so called "pressing" of a master DVD or audio disk, for example). Accordingly, another significant advantage of the present invention is that rights management of content essentially can be extended throughout and across each appropriate content creation, editing, distribution, and usage stages to provide a seamless content protection architecture that protects rights throughout an entire content life cycle.

In one example embodiment, the storage medium itself carries key block decryption key(s) in a hidden portion of the storage medium not normally accessible through typical access and/or copying techniques. This hidden key may be used by a drive to decrypt the encrypted key block—such decrypted key block then being used to selectively decrypt content and related information carried by the medium. The drive may be designed in a secure and tamper-resistant manner so that the hidden keys are never exposed outside of the drive to provide an additional security layer.

In accordance with another example embodiment, a video disk drive may store and maintain keys used to decrypt an encrypted key block. The key block decryption keys may be

stored in a drive key store, and may be updatable if the video disk drive may at least occasionally use a communications path provided, for example, by a set top box, network port or other communications route.

In accordance with a further example embodiment, a virtual distribution environment secure node including a protected processing environment such as a hardware-based secure processing unit may control the use of content carried by a portable storage medium such as a digital video disk in accordance with control rules and methods specified by one or more secure containers delivered to the secure node on the medium itself and/or over an independent communications path such as a network.

Certain conventional copy protection for DVD currently envisions CGMA copy protection control codes combined with certain encryption techniques first proposed apparently by Matsushita Corporation. Notwithstanding the limited benefits of this approach to digital property protection, the present invention is capable of providing a supplementary, compatible, and far more comprehensive rights management system while also providing

additional and/or different options and solutions. The following are some additional examples of advantageous features provided in accordance with the inventions:

- Strong security to fully answer content supplier needs.

- Value chain management automation and efficiencies including distributed rights protection, "piece of the tick" payment disaggregation to value chain participants, cost-effective micro-transaction management, and superdistribution, including offline micropayment and microtransaction support for at least occasionally connected devices.

- Simplified, more efficient channel management including support for the use of the same content deliverable on limited resource, greater resource, standalone, and/or connected devices.

- Can be used with any medium and application type and/or all forms of content and content models -- not just compressed video and sound as in some prior techniques and supports the use of copies of the same or materially the

same content containers across a wide variety of media delivery systems (e.g., broadcast, Internet repository, optical disc, etc) for operation on a wide variety of different electronic appliances (e.g., digital cameras, digital editing equipment, sound recorders, sound editing equipment, movie theater projectors, DVD appliances, broadcast tape players, personal computers, smart televisions, etc).

- Asset management and revenue and/or other consideration maximizing through important new content revenue and/or other consideration opportunities and the enhancement of value chain operating efficiencies.

- Is capable of providing 100% compatibility with the other protection techniques such as, for example, CGMA protection codes and/or Matsushita data scrambling approaches to DVD copy protection.

- Can be employed with a variety of existing data scrambling or protection systems to provide very high degrees of compatibility and/or level of functionality.

- Allows DVD technology to become a reusable, programmable, resource for an unlimited variety of entertainment, information commerce, and cyberspace business models.

- Enables DVD drive and/or semiconductor component manufacturers and/or distributors and/or other value adding participants to become providers of, and rights holders in, the physical infrastructure of the emerging, connected world of the Internet and Intranets where they may charge for the use of a portion (e.g., a portion they provided) of the distributed, physical infrastructure as that portion participates in commercial networks. Such manufacturers and/or distributors and/or other value adding participants can enjoy the revenue benefits resulting from participation in a "piece of the tick" by receiving a small portion of the revenue received as a result of a participating transaction.

- Provides automated internationalization, regionalization, and rights management in that:

  -- DVD content can be supplied with arrays of different rule sets for

automatic use depending on rights and identity of the user; and

-- Societal rights, including taxes, can be handled transparently.

In addition, the DVD rights management method and apparatus of the present invention provides added benefits to media recorders/publishers in that it:

- Works with a current "keep honest people honest" philosophy.
- Can provide 100% compatibility with other protection schemes such as for example, Matsushita data scrambling and/or CGMA encoded discs.
- Can work with and/or supplement other protection schemes to provide desired degree and/or functionality, or can be used in addition to or instead of other approaches to provide additional and/or different functionality and features.

- Provides powerful, extensible rights management that reaches beyond limited copy protection models to rights management for the digitally convergent world.

- Empowers recording/publishing studios to create sophisticated asset management tools.

- Creates important business opportunities through controlled use of studio properties in additional multimedia contexts.

- Uniquely ties internationalization, regionalization, superdistribution, repurposing, to content creation processes and/or usage control.

Other aspects of the present invention provide benefits to other types of rightsholders, such as for example:

- Persistent, transparent protection of digital content—globally, through value chain and process layers.

- Significant reduction in revenue loss from copying and pass-along.

- Converts "pass-along," copying, and many forms of copyright infringement from a strategic business threat to a fundamental business opportunity.

- A single standard for all digital content regardless of media and/or usage locality and other rights variables.

- Major economies of scale and/or scope across industries, distribution channels, media, and content type.

- Can support local usage governance and auditing within DVD players allowing for highly efficient micro-transaction support, including multiparty microtransactions and transparent multiparty microtransactions.

- Empowers rightsholders to employ the broadest range of pricing, business models, and market strategies—as they see fit.

Further aspects of the present invention which may prove beneficial to DVD and other digital medium appliance manufacturers are:

- Capable of providing bit for bit compatibility with existing discs.

- Content type independent.

- Media independent and programmable/reusable.

- Highly portable transition to next generation of appliances having higher density devices and/or a writable DVD and/or other optical media format(s).

- Participation in revenue flow generated using the appliance.

- Single extensible standard for all digital content appliances.

- Ready for the future "convergent" world in which many appliances are connected in the home using, as one example, IEEE 1394 interfaces or other means (e.g., some appliances will be very much like computers and some computers will be very much like appliances).

Aspects of the present inventions provide many benefits to computer and OS manufacturers such as for example:

- Implementation in computers as an extension to the operating system, via for example, at least one transparent plug-in, and does not require modifications to computer hardware and/or operating systems.

- Easy, seamless integration into operating systems and into applications.

- Extremely strong security, especially when augmented with "secure silicon" (i.e., hardware/firmware protection apparatus fabricated on chip).

- Transforms user devices into true electronic commerce appliances.

- Provides a platform for trusted, secure rights management and event processing.

- Programmable for customization to specialized requirements.

Additional features and advantages provided in accordance with the inventions include, for example:

- Information on the medium (for example, both properties and metadata) may be encrypted or not.

- Different information (for example, properties, metadata) may be encrypted using different keys. This provides greater protection against compromise, as well as supporting selective usage rights in the context of a sophisticated rights management system.

- There may be encrypted keys stored on the medium, although this is not required. These keys may be used to decrypt the protected properties and metadata. Encrypted keys are likely to be used because that allows more keying material for the information itself, while still keeping access under control of a single key.

- Multiple sets of encrypted keys may be stored on the medium, either to have different sets of keys associated with different information, or to allow multiple control regimes to use the

same information, where each control regime may use one or more different keys to decrypt the set of encrypted keys that it uses.

- To support the ability of the player to access rights managed containers and/or content, a decryption key for the encrypted keys may be hidden on the medium in one or more locations that are not normally accessible. The "not normally accessible" location(s) may be physically enabled for drives installed in players, and disabled for drives installed in computers. The enablement may be different firmware, a jumper on the drive, etc.

- The ability of the player to access rights managed containers and/or content may also be supported by one or more stored keys inside the player that decrypts certain encrypted keys on the medium.

- Keys in a player may allow some players to play different properties than others. Keys could be added to, and/or deleted from the player by a network connection (e.g., to a PC, a cable system, and/or a modem connection to a source of new and/or additional keys and/or

key revocation information) or automatically loaded by "playing" a key distribution DVD.

- Controlling computer use may be supported by some or all of the same techniques that control player use of content and/or rights managemnt information.

- Controlling computer use of content and/or rights management information may be supported by having a computer receive, through means of a trusted rights management system, one or more appropriate keys.

- A computer may receive additional keys that permit decryption of certain encrypted keys on the medium.

- A computer may receive additional keys that permit decryption of one or more portions of encrypted data directly. This may permit selective use of information on the medium without disclosing keys (e.g., a player key that decrypts any encrypted keys).

In accordance with further aspects provided by the present invention, a secure "software container" is provided that allows:

- Cryptographically protected encapsulation of content, rights rules, and usage controls.

- Persistent protection for transport, storage, and value chain management.

- Sophisticated rules interface architecture.

Elements can be delivered independently, such as new controls, for example, regarding discount pricing (e.g. sale pricing, specific customer or group discounts, pricing based on usage patterns, etc.) and/or other business model changes, can be delivered after the property has been distributed (this is especially beneficial for large properties or physical distribution media (e.g., DVD, CD-ROM) since redistribution costs may be avoided and consumers may continue to use their libraries of discs). In addition, encrypted data can be located "outside" the container. This can allow, for example, use of data stored independently from the controls and supports "streaming" content as well as "legacy" systems (e.g., CGMS).

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages provided in accordance with these inventions may be better and more completely understood by referring to the following detailed description of presently preferred examples in conjunction with the drawings, of which:

Figure 1A shows example home consumer electronics equipment for using portable storage media such as digital video disks;

Figure 1B shows example secure node equipment for using the same portable storage media but providing more advanced rights management capabilities;

Figure 1C shows an example process for manufacturing protected optical disks;

Figure 2A shows an example architecture of the Figure 1A consumer electronics equipment;

Figure 2B shows an example architecture for the Figure 1B secure node equipment;

Figure 3 shows example data structures used by the Figure 1A equipment;

Figure 3A and 3B show example control set definitions;

Figures 4A and 4B show example usage techniques provided by the Figure 1A appliance;

Figure 5 shows example data structures used by the Figure 1B secure node for accessing information on the storage medium;

Figure 6 shows an example usage technique performed by the Figure 1B secure node;

Figure 7 is a block diagram illustrating an example of a special secure software container contained on a DVD;

Figure 8 is a block diagram illustrating an example of a secure container along with the video property content stored on a DVD medium;

Figure 9 is a block diagram illustrating another example of a standard container stored on a DVD medium including an additional container having a more complex rule arrangement for use, for example, with a secure node;

Figure 10 shows an example use of a DVD having a container (i.e., stored on the medium) with a DVD player provided with a secure rights management node, and also shows use of the same DVD with a DVD player that does not have a secure rights management node;

Figure 11 is a block diagram illustrating use of a DVD that does not have a container on a DVD player that is provided with rights management secure node in accordance with the present invention as compared with use of the same DVD with a DVD player that does not have a secure node;

Figures 12-14 show example network configurations; and

Figures 15A-15C show an example virtual rights process.

DETAILED DESCRIPTION OF PRESENTLY PREFERRED EXAMPLE EMBODIMENTS

Overall Example Digital Video Disk Usage System

Figure 1A shows example inexpensive mass-produced home consumer electronics equipment 50 for using information stored on a storage medium 100 such as a portable digitally-encoded optical disk (e.g., a digital video disk or "DVD").

Consumer equipment 50 includes a dedicated disk player 52, that in some embodiments, may also have the capability to write optical media (writeable DVD disks, or "DVD-RAM") for example) as well, connected to a home color television set 54. A remote control unit 56 may be used to control the disk player 52 and/or television set 54.

In one example, disk 100 may store a feature length motion picture or other video content. Someone wishing to watch the content stored on disk 100 may purchase or rent the disk, insert the disk into player 52 and use remote control 56 (and/or controls 58 that may be provided on player 52) to control the player to play back the content via home television set 54.

In some embodiments, remote control 56 (and/or controls 58 that may be provided on device 52) may be used to control the recording of a movie, for example. Player 52 reads the digitized video and audio information carried by disk 100, converts it into signals compatible with home color television set 54, and provides those signals to the home color television set.

In some embodiments, television set 54 (and/or a set top box) provide the video signals to be recorded by device 52 on writable optical media, DVD-RAM in one non-limiting example. Television set 54 produces images on screen 54a and produces sounds through loudspeakers 54b based on the signals player 52 provides to the television set.

The same disk 100 may be used by a more advanced platform 60 shown in Figure 1B. Platform 60 may include, for example, a personal computer 62 connected to a display monitor 64, a keyboard 66, a mouse pointing device 68, and a loudspeaker 70. In this example, platform 60 may be able to play back the content stored on disk 100 in the same way as dedicated disk player 52, but may also be capable of more sophisticated and/or advanced uses of the content as enabled by the presence of secure node 72 within the platform. (In some embodiments, platform 60 may also be able to record content on writable optical media, DVD-RAM, in one non-limiting example.) For example, it may be possible, using platform 60 and its secure node 72, to interactively present the motion picture or other content such that the user may input choices via keyboard 66 and/or mouse pointing device 68

that, in real time, change the presentation provided via display 64 and loudspeaker 60.

As one example, the platform 60 user selects from options displayed on display 64 that cause the content presentation sequence to change (e.g., to provide one of a number of different endings, to allow the user to interactively control the flow of the images presented, etc.). Computer 62 may also be capable of using and manipulating digital data including for example computer programs and/or other information stored on disk 100 that player 52 cannot handle.

Secure node 72 provides a secure rights management facility that may, for example, permit more invasive or extensive use of the content stored on disk. For example, dedicated player 52 may prevent any copying of content stored by disk 100, or it may allow the content to be copied only once and never again. Platform 60 including secure node 72, on the other hand, may allow multiple copies of some or all of the same content—but only if certain conditions are met (e.g., the user of equipment 60 falls within a certain class of people, compensation at an agreed on rate is securely provided for each copy made, only certain excerpts of

the content are copied, a secure audit trail is maintained and reported for each copy so made, etc.). (In some embodiments, dedicated player 52 may send protected content only to devices authenticated as able to enforce securely rights management rules and usage consequences. In some embodiments, devices may authenticate using digital certificates, one non-limiting example being certificates conforming to the X.509 standard.) Hence, platform 60 including secure node 72 can, in this example, use the content provided by disk 100 in a variety of flexible, secure ways that are not possible using dedicated player 52—or any other appliance that does not include a secure node.

Example Secure Disk Creation and Distribution Process

Figure 1C shows an example secure process for creating a master multimedia DVD disk 100 for use with players 50, 60. In this example, a digital camera 350 converts light images (i.e., pictures) into digital information 351 representing one or a sequence of images. Digital camera 350 in this example includes a secure node 72A that protects the digital information 351 before it leaves camera 350. Such protection can be accomplished, for

example, by packaging the digital information within one or more containers and/or associating controls with the digital information.

In this example, digital camera 350 provides the protected digital image information 351 to a storage device such as, for example, a digital tape recorder 352. Tape recorder 352 stores the digital image information 351 (along with any associated controls) onto a storage medium such as magnetic tape cartridge 354 for example. Tape recorder 352 may also include a secure node 72B. Secure node 72B in this example can understand and enforce the controls that the digital camera secure node 72A applies to and/or associated with the digital information 351, and/or it may apply its own controls to the stored information.

The same or different tape recorder 352 may play back protected digital information 351 to a digital mixing board 356. Digital mixing board 356 may mix, edit, enhance or otherwise process the digital information 351 to generate processed digital information 358 representing one or a sequence of images. Digital mixing board 356 may receive additional inputs from other devices such as for example other tape recorders, other digital cameras, character generators, graphics generators, animators, or

any other image-based devices. Any or all of such devices may also include secure nodes 72 to protect the information they generate. In some embodiments, some of the digital information can be derived from equipment including a secure node, and other digital information can be derived from equipment that has no secure node. In still other embodiments, some of the digital information provided to digital mixer 356 is protected and some is not protected.

Digital mixing board 356 may also include a secure node 72C in this example. The digital mixing board secure node 72C may enforce controls applied by digital camera secure node 72A and/or tape recorder secure node 72B, and/or it may add its own protections to the digital information 358 it generates.

In this example, an audio microphone 361 receives sound and converts the sound into analog audio signals. The audio signals in this example are inputted to a digital audio tape recorder 362. In the example shown, tape recorder 362 and audio mixer 364 are digital devices. However, in other embodiments, one, the other or both of these devices may operate in the analog domain. In the example shown, digital audio tape recorder 362 converts the

analog audio signals into digital information representing the sounds, and stores the digital information (and any associated controls) onto a tape 362.

In this example, audio tape recorder 362 includes a secure node 72E that may associate controls with the information stored on tape 363. Such controls may be stored with the information on the tape 363. In another embodiment, microphone 361 may include its own internal secure node 72 that associates control information with the audio information (e.g., by steganographically encoding the audio information with control information). The tape recorder 362 may enforce such controls applied by microphone 361.

Alternatively, microphone 361 may operate in the digital domain and provide digital representations of audio, perhaps including control information supplied by secure node 72 optionally incorporated in microphone 361, directly to connected devices such as audio tape recorder 362. Digital representations may optionally be substituted for analog representations of any signals between the devices in the example Figure 1C.

The same or different tape recorder 362 may play back the information recorded on tape 363, and provide the information 366 to an audio mixer 364. Audio mixer 364 may edit, mix, or otherwise process the information 366 to produce information 368 representing one or a sequence of sounds. Audio mixer 364 may also receive inputs from other devices such as for example other tape recorders, other microphones, sound generators, musical synthesizers, or any other audio-based devices. Any or all of such devices may also include secure nodes 72 to protect the information they generate. In some embodiments, some of the digital information is derived from equipment including a secure node, and other digital information is derived from equipment that has no secure node. In still other embodiments, some of the digital information provided to audio mixer 364 is protected and some is not protected.

Audio mixer 364 in this example includes a secure node 72F that enforces the controls, if any, applied by audio tape recorder secure node 72E; and/or applies its own controls.

Digital image mixer 356 may provide digital information 358 to "DVD-RAM" equipment 360 that is capable of writing to

master disks 100 and/or to disks from which master dicks may be created Similarly, audio mixer 364 may provide digital information 368 to equipment 360. Equipment 360 records the image information 358 and audio information 368 onto master disk 100. In this example, equipment 360 may include a secure node 72D that enforces controls applied by digital camera secure node 72A, tape recorder secure node 72B, digital mixer secure node 72C audio tape recorder secure node 72E and/or audio mixer secure node 72F; and/or it may add its own protections to the digital information 358 it writes onto master disks 100. A disk manufacturer can then mass-produce disks 100(1)-100(N) based on the master disk 100 using conventional disk mass-production equipment for distribution through any channels (e.g., video and music stores, websites, movie theaters, etc.). Consumer appliances 50 shown in Figures 1A and 1B may play back the disks 100 – enforcing the controls applied to the information stored on the disks 100. Secure nodes 72 thus maintain end-to-end, persistent secure control over the images generated by digital camera 350 and the sounds generated by microphone 361 during the entire process of making, distributing and using disks 100.

In the Figure 1C example shown, the various devices may communicate with one another over so-called "IEEE 1394" high-speed digital serial busses. In this context, "IEEE 1394" refers to hardware and software standards set forth in the following standards specification incorporated by reference herein: 1394-1995 IEEE Standard for a High Performance Serial Bus, No. 1-55937-583-3 (Institute of Electrical and Electronics Engineers 1995). This specification describes a high-speed memory mapped digital serial bus that is self-configuring, hot pluggable, low cost and scalable. The bus supports isochronous and asynchronous transport at 100, 200 or 400 Mbps, and flexibly supports a number of different topologies. The specification describes a physical level including two power conductors and two twisted pairs for signalling. The specification further describes physical, link and transaction layer protocols including serial bus management. Alternatively, any other suitable electronic communication means may be substituted for the "IEEE 1394" medium shown in Figure 1C, including other wired media (e.g., Ethernet, universal serial bus), and/or wireless media based on radio-frequency (RF)

transmission, infra-red signals, and/or any other means and/or types of electronic communication.

Example Dedicated Player Architecture

Figure 2A shows an example architecture for dedicated player 52. In this example, player 52 includes a video disk drive 80, a controller 82 (e.g., including a microprocessor 84, a memory device such as a read only memory 86, and a user interface 88), and a video/audio processing block 90. Video disk drive 80 optically and physically cooperates with disk 100, and reads digital information from the disk. Controller 82 controls disk drive 80 based on program instructions executed by microprocessor 84 and stored in memory 86 (and further based on user inputs provided by user interface 88 which may be coupled to controls 58 and/or remote control unit 56). Video/audio processing block 90 converts digital video and audio information read by disk drive 80 into signals compatible with home color television set 54 using standard techniques such as video and audio decompression and the like. Video/audio processing block 90 may also insert a visual marking indicating the ownership and/or protection of the video program. Block 90 may also

introduce a digital marking indicating to a standard recording device that the content should not be recorded.

Example Secure Node Architecture

Figure 2B shows an example architecture for platform 60 shown in Figure 1B—which in this example is built around a personal computer 62 but could comprise any number of different types of appliances. In this example, personal computer 62 may be connected to an electronic network 150 such as the Internet via a communications block 152. Computer equipment 62 may include a video disk drive 80' (which may be similar or identical to the disk drive 80 included within example player 52). Computer equipment 62 may further include a microprocessor 154, a memory 156 (including for example random access memory and read only memory), a magnetic disk drive 158, and a video/audio processing block 160. Additionally, computer equipment 62 may include a tamper-resistant secure processing unit 164 or other protected processing environment. Secure node 72 shown in Figure 1B may thus be provided by a secure processing unit 164, software executing on microprocessor 154, or a combination of

the two. Different embodiments may provide secure node 72 using software-only, hardware-only, or hybrid arrangements.

Secure node 72 in this example may provide and support a a general purpose Rights Operating System employing reusable kernel and rights language components. Such a commerce-enabling Rights Operating System provides capabilities and integration for advanced commerce operating systems of the future. In the evolving electronic domain, general purpose, reusable electronic commerce capabilities that all participants can rely on will become as important as any other capability of operating systems. Moreover, a rights operating system that provides, among other things, rights and auditing operating system functions can securely handle a broad range of tasks that relate to a virtual distribution environment. A secure processing unit can, for example, provide or support many of the security functions of the rights and auditing operating system functions. The other operating system functions can, for example, handle general appliance functions. The overall operating system may, for example, be designed from the beginning to include the rights and auditing operating system functions plus the other operating

system functions, or the rights and auditing operating system functions may, in another example, be an add-on to a preexisting operating system providing the other operating system functions. Any or all of these features may be used in combination with the invention disclosed herein.

Example Disk Data Structures and Associated Protections

Figure 3 shows some example data structures stored on disk 100. In this example, disk 100 may store one or more properties or other content 200 in protected or unprotected form. Generally, in this example, a property 200 is protected if it is at least in part encrypted and/or associated information needed to use the property is at least in part encrypted and/or otherwise unusable without certain conditions having being met. For example, property 200(1) may be completely or partially encrypted using conventional secure cryptographic techniques. Another property 200(2) may be completely unprotected so that it can be used freely without any restriction. Thus, in accordance with this example, disk 100 could store both a movie as a protected property 200(1) and an unprotected interview with the actors and producers or a

"trailer" as unprotected property 200(2). As shown in this example, disk 100 may store any number of different properties 200 in protected or unprotected form as limited only by the storage capacity of the disk.

In one example, the protection mechanisms provided by disk 100 may use any or all of the protection (and/or other) structures and/or techniques described in the above-referenced Shear patents. The Shear patents describe, by way of non-exhaustive example, means for solving the problem of how to protect digital content from unauthorized use. For example, the Shear patent specifications describe, among other things, means for electronically "overseeing" -- through distributed control nodes present in client computers -- the use of digital content. This includes means and methods for fulfilling the consequences of any such use.

Non-limiting examples of certain elements described in the Shear patent specifications include:

(a) decryption of encrypted information,

(b) metering,

(c) usage control in response to a combination of derived metering information and rules set by content providers,

(d) securely reporting content usage information,

(e) use of database technology for protected information storage and delivery,

(f) local secure maintenance of budgets, including, for example, credit budgets,

(g) local, secure storage of encryption key and content usage information,

(h) local secure execution of control processes, and

(i) in many non-limiting instances, the use of optical media.

Any or all of these features may be used in combination in or with the inventions disclosed herein.

Certain of the issued Shear patents' specifications also involve database content being local and remote to users.

Database information that is stored locally at the end-user's system and complemented by remote, "on-line" database information, can, for example, be used to augment the local information, which in one example, may be stored on optical media (for example, DVD and/or CD-ROM). Special purpose semiconductor hardware can, for example, be used to provide a secure execution environment to ensure a safe and reliable setting for digital commerce activities.

The Shear patents also describe, among other things, database usage control enabled through the use of security, metering, and usage administration capabilities. The specifications describe, *inter alia*, a metering and control system in which a database, at least partially encrypted, is delivered to a user (e.g., on optical media). Non-limiting examples of such optical media may, for example, include DVD and CD-ROM. Subsequent usage can, for example, be metered and controlled in any of a variety of ways, and resulting usage information can be transmitted to a responsible party (as one example).

The Shear patent specifications also describe the generation of a bill in response to the transmitted information. Other

embodiments of the Shear patents provide, for example, unique information security inventions which involve, for example, digital content usage being limited based on patterns of usage such as the quantity of particular kinds of usage. These capabilities include monitoring the "contiguousness," and/or "logical relatedness" of used information to ensure that the electronic "conduct" of an individual does not exceed his or her licensed rights. Still other aspects of the Shear patents describe, among other things, capabilities for enabling organizations to securely and locally manage electronic information usage rights. When a database or a portion of a database is delivered to a client site, some embodiments of the Shear patents provide, for example, optical storage means (non-exhaustive examples of which include DVD and CD-ROM) as the mechanism of delivery. Such storage means can store, for example, a collection of video, audio, images, software programs, games, etc., in one example, on optical media, such as DVD and/or CD-ROM, in addition to other content such as a collection of textual documents, bibliographic records, parts catalogs, and copyrighted or uncopyrighted materials of all kinds.

Any or all of these features may be used in the embodiments herein.

One specific non-limiting embodiment could, for example, involve a provider who prepares a collection of games. The provider prepares a database "index" that stores information pertaining to the games, such as for example, the name, a description, a creator identifier, the billing rates, and the maximum number of times or total elapsed time each game may be used prior to a registration or re-registration requirement. Some or all of this information could be stored in encrypted form, in one example, on optical media, non-limiting examples of which include DVD and CD-ROM. The provider may then encrypt some or all portions of the games such that a game could not be used unless one or more encrypted portions were decrypted. Typically, decryption would not occur unless provider specified conditions were satisfied, in one example, unless credit was available to compensate for use and audit information reflecting game usage was being stored. The provider could determine, for example: which user activities he or she would allow, whether to meter such activities for audit and/or control purposes, and what, if any, limits

would be set for allowed activities. This might include, for example, the number of times that a game is played, and the duration of each play. Billing rates might be discounted, for example, based on total time of game usage, total number of games currently registered for use, or whether the customer was also registered for other services available from the same provider, etc.

In the non-limiting example discussed above, a provider might, for example, assemble all of the prepared games along with other, related information, and publish the collection on optical media, non-limiting examples of which include CD-ROM and/or DVD. The provider might then distribute this DVD disk to prospective customers. The customers could then select the games they wish to play, and contact the provider. The provider, based on its business model, could then send enabling information to each authorized customer, such as for example, including, or enabling for use, decryption keys for the encrypted portion of the selected games (alternatively, authorization to use the games may have arrived with the DVD and/or CD-ROM disk, or might be automatically determined, based on provider set criteria, by the

user's secure client system, for example, based on a user's participation in a certified user class). Using the user's client decryption and metering mechanism the customer could then make use of the games. The mechanism might then record usage information, such as for example, the number of times the game was used, and, for example, the duration of each play. It could periodically transmit this information the game provider, thus substantially reducing the administration overhead requirements of the provider's central servers. The game provider could receive compensation for use of the games based upon the received audit information. This information could be used to either bill their customers or, alternatively, receive compensation from a provider of credit.

Although games provide one convenient, non-limiting example, many of these same ideas can be easily applied to all kinds of content, all kinds of properties, including, by way of non-limiting examples:

- video,
- digitized movies,

- audio,
- images,
- multimedia,
- software,
- games,
- any other kind of property
- any combination of properties.

Other non-limiting embodiments of the Shear patent specifications support, for example, securely controlling different kinds of user activities, such as displaying, printing, saving electronically, communicating, etc. Certain aspects further apply different control criteria to these different usage activities. For example, information that is being browsed may be distinguished from information that is read into a host computer for the purpose of copying, modifying, or telecommunicating, with different cost rates being applied to the different activities (so that, for example,

the cost of browsing can be much less than the cost of copying or printing).

The Shear patent specifications also, for example, describe management of information inside of organizations by both publishers and the customer. For example, an optional security system can be used to allow an organization to prevent usage of all or a portion of an information base unless the user enters his security code. Multiple levels of security codes can be supported to allow restriction of an individual's use according to his security authorization level. One embodiment can, for example, use hardware in combination with software to improve tamper resistance, and another embodiment could employ an entirely software based system. Although a dedicated hardware/software system may under certain circumstances provide assurance against tampering, techniques which may be implemented in software executing on a non-dedicated system may provide sufficient tamper resistance for some applications. Any or all of these features may be used in combination with the technology disclosed in this patent specification.

Figures 3 Disks May Also Store Metadata, Controls and Other Information

In this example, disk 100 may also store "metadata" in protected and/or unprotected form. Player 52 uses metadata 202 to assist in using one or more of the properties 200 stored by disk 100. For example, disk 100 may store one metadata block 202(1) in unprotected form and another metadata block 202(2) in protected form. Any number of metadata blocks 202 in protected and/or unprotected form may be stored by disk 100 as limited only by the disk's storage capacity. In this example, metadata 202 comprises information used to access properties 200. Such metadata 202 may comprise, for example, frame sequence or other "navigational" information that controls the playback sequence of one or more of the properties 200 stored on disk 100. As one example, an unprotected metadata block 202 may access only selected portions of a protected property 200 to generate an abbreviated "trailer" presentation, while protected metadata block 202 may contain the frame playback sequence for the entire video presentation of the property 200. As another example, different metadata blocks 202 may be provided for different "cuts" of the

same motion picture property 200 (e.g., an R-rated version, a PG-rated version, a director's cut version, etc.).

In this example, disk 100 may store additional information for security purposes. For example, disk 100 may store control rules in the form of a control set 204—which may be packaged in the form of one or more secure containers 206. Commerce model participants can securely contribute electronic rules and controls that represent their respective "electronic" interests. These rules and controls extend a "Virtual Presence™" through which the commerce participants may govern remote value chain activities according to their respective, mutually agreed to rights. This Virtual Presence may take the form of participant specified electronic conditions (e.g., rules and controls) that must be satisfied before an electronic event may occur. These rules and controls can be used to enforce the party's rights during "downstream" electronic commerce activities. Control information delivered by, and/or otherwise available for use with, VDE content containers may, for example, constitute one or more "proposed" electronic agreements which manage the use and/or consequences of the use of such content and which can enact the

terms and conditions of agreements involving multiple parties and their various rights and obligations.

The rules and controls from multiple parties can be used, in one example, to form aggregate control sets ("Cooperative Virtual Presence™") that ensure that electronic commerce activities will be consistent with the agreements amongst value chain participants. These control sets may, for example, define the conditions which govern interaction with protected digital content (disseminated digital content, appliance control information, etc.). These conditions can, for example, be used to control not only digital information use itself, but also the consequences of such use. Consequently, the individual interests of commerce participants are protected and cooperative, efficient, and flexible electronic commerce business models can be formed. These models can be used in combination with the present invention.

Disks May Store Encrypted Information

Disk 100 may also store an encrypted key block 208. In this example, disk 100 may further store one or more hidden keys 210. In this example, encrypted key block 208 provides one or more cryptographic keys for use in decrypting one or more properties

200 and/or one or more metadata blocks 202. Key block 208 may provide different cryptographic keys for decrypting different properties 200 and/or metadata blocks 202, or different portions of the same property and/or metadata block. Thus, key block 208 may comprise a large number of cryptographic keys, all of which are or may be required if all of the content stored by disk 100 is to be used. Although key block 208 is shown in Figure 3 as being separate from container 206, it may be included within or as part of the container if desired.

Cryptographic key block 208 is itself encrypted using one or more additional cryptographic keys. In order for player 52 to use any of the protected information stored on disk 100, it must first decrypt corresponding keys within the encrypted key block 208—and then use the decrypted keys from the key block to decrypt the corresponding content.

In this example, the keys required to decrypt encrypted key block 208 may come from several different (possibly alternative) sources. In the example shown in Figure 3, disk 100 stores one or more decryption keys for decrypting key block 208 on the medium itself in the form of a hidden key(s) 210. Hidden key(s) 210 may

be stored, for example, in a location on disk 100 not normally accessible. This "not normally accessible" location could, for example, be physically enabled for drives 80 installed in players 52 and disabled for drives 80' installed in personal computers 62. Enablement could be provided by different firmware, a jumper on drive 80, etc. Hidden key(s) 210 could be arranged on disk 100 so that any attempt to physically copy the disk would result in a failure to copy the hidden key(s). In one example a hidden key(s) could be hidden in the bit stream coding sequences for one or more blocks as described by J. Hogan (Josh Hogan, "DVD Copy Protection," presentation to DVD copy protect technical meeting #4, 5/30/96, Burbank, CA.)

Alternatively, and/or in addition, keys required to decrypt encrypted key block 208 could be provided by disk drive 80. In this example, disk drive 80 might include a small decryption component such as, for example, an integrated circuit decryption engine including a small secure internal key store memory 212 having keys stored therein. Disk drive 80 could use this key store 212 in order to decrypt encrypted key block 208 without exposing either keys 212 or decrypted key block 208—and then use the

decrypted key from key block 208 to decrypt protected content 200, 202.

Disks May Store and/or Use Secure Containers

In yet another example, the key(s) required to decrypt protected content 200, 202 is provided within secure container 206. Figure 3A shows a possible example of a secure container 206 including information content 304 (properties 200 and metadata 202 may be external to the container—or alternatively, most or all of the data structures stored by video disk 100 may be included as part of a logical and/or actual protected container). The control set 204 shown in Figure 3 may comprise one or more permissions record 306, one or more budgets 308 and/or one or more methods 310 as shown in Figure 3A. Figure 3B shows an example control set 204 providing one or more encryption keys 208, one or more content identifiers 220, and one or more controls 222. In this example, different controls 222 may apply to different equipment and/or classes of equipment such as player 52 and/or computer equipment 62 depending upon the capabilities of the particular platform and/or class of platform. Additionally, controls 220 may apply to different ones of properties 200 and/or

different ones of metadata blocks 202. For example, a control 222(1) may allow property 200(1) to be copied only once for archival purposes by either player 52 or computer equipment 62. A control 222(2) (which may be completely ignored by player 52 because it has insufficient technical and/or security capabilities but which may be useable by computer equipment 62 with its secure node 72) may allow the user to request and permit a public performance of the same property 200(1) (e.g., for showing in a bar or other public place) and cause the user's credit or other account to be automatically debited by a certain amount of compensation for each showing. A third control 222(3) may, for example, allow secure node 72 (but not player 52) to permit certain classes of users (e.g., certified television advertisers and journalists) to extract or excerpt certain parts of protected property 200(1) for promotional uses. A further control 222(4) may, as another example, allow both video player 52 and secure node 72 to view certain still frames within property 200(1)—but might allow only secure node 72 to make copies of the still frames based on a certain compensation level.

Example Disks and/or System May Make Use of Trusted Infrastructure

Controls 222 may contain pointers to sources of additional control sets for one or more properties, controls, metadata, and/or other content on the optical disk. In one example, these additional controls may be obtained from a trusted third party, such as a rights and permissions clearinghouse and/or from any other value chain participant authorized by at least one rightsholder to provide at least one additional control set. This kind of rights and permissions clearinghouse is one of several distributed electronic administrative and support services that may be referred to as the "Distributed Commerce Utility," which, among other things, is an integrated, modular array of administrative and support services for electronic commerce and electronic rights and transaction management. These administrative and support services can be used to supply a secure foundation for conducting financial management, rights management, certificate authority, rules clearing, usage clearing, secure directory services, and other transaction related capabilities functioning over a vast electronic network such as the Internet and/or over organization internal Intranets, or even in-home networks of electronic appliances. Non-

limiting examples of these electronic appliances include at least occasionally connected optical media appliances, examples of which include read-only and/or writable DVD players and DVD drives in computers and convergent devices, including, for example, digital televisions and settop boxes incorporating DVD drives.

These administrative and support services can, for example, be adapted to the specific needs of electronic commerce value chains in any number of vertical markets, including a wide variety of entertainment applications. Electronic commerce participants can, for example, use these administrative and support services to support their interests, and/or they can shape and reuse these services in response to competitive business realities. Non-exhaustive examples of electronic commerce participants include individual creators, film and music studios, distributors, program aggregators, broadcasters, and cable and satellite operators.

The Distributed Commerce Utility can, for example, make optimally efficient use of commerce administration resources, and can, in at least some embodiments, scale in a practical fashion to

optimally accommodate the demands of electronic commerce growth.

The Distributed Commerce Utility may, for example, comprise a number of Commerce Utility Systems. These Commerce Utility Systems can provide a web of infrastructure support available to, and reusable by, the entire electronic community and/or many or all of its participants. Different support functions can, for example, be collected together in hierarchical and/or in networked relationships to suit various business models and/or other objectives. Modular support functions can, for example, be combined in different arrays to form different Commerce Utility Systems for different design implementations and purposes. These Commerce Utility Systems can, for example, be distributed across a large number of electronic appliances with varying degrees of distribution.

The "Distributed Commerce Utility" provides numerous additional capabilities and benefits that can be used in conjunction with the particular embodiments shown in the drawings of this application, non-exhaustive examples of which include:

- Enables practical and efficient electronic commerce and rights management.

- Provides services that securely administer and support electronic interactions and consequences.

- Provides infrastructure for electronic commerce and other forms of human electronic interaction and relationships.

- Optimally applies the efficiencies of modern distributed computing and networking.

- Provides electronic automation and distributed processing.

- Supports electronic commerce and communications infrastructure that is modular, programmable, distributed and optimally computerized.

- Provides a comprehensive array of capabilities that can be combined to support services that perform various administrative and support roles.

- Maximizes benefits from electronic automation and distributed processing to produce optimal allocation and use of resources across a system or network.

- Is efficient, flexible, cost effective, configurable, reusable, modifiable, and generalizable.

- Can economically reflect users' business and privacy requirements.

- Can optimally distribute processes -- allowing commerce models to be flexible, scaled to demand and to match user requirements.

- Can efficiently handle a full range of activities and service volumes.

- Can be fashioned and operated for each business model, as a mixture of distributed and centralized processes.

- Provides a blend of local, centralized and networked capabilities that can be uniquely shaped and reshaped to meet changing conditions.

- Supports general purpose resources and is reusable for many different models; in place infrastructure can be reused by different value chains having different requirements.

- Can support any number of commerce and communications models.

- Efficiently applies local, centralized and networked resources to match each value chain's requirements.

- Sharing of common resources spreads out costs and maximizes efficiency.

- Supports mixed, distributed, peer-to-peer and centralized networked capabilities.

- Can operate locally, remotely and/or centrally.

- Can operate synchronously, asynchronously, or support both modes of operation.

- Adapts easily and flexibly to the rapidly changing sea of commercial opportunities, relationships and constraints of "Cyberspace."

Any or all of these features may be used in combination with the inventions disclosed herein.

The Distributed Commerce Utility provides, among other advantages, comprehensive, integrated administrative and support services for secure electronic commerce and other forms of electronic interaction. These electronic interactions supported by the Distributed Commerce Utility may, in at least some embodiments, entail the broadest range of appliances and distribution media, non-limiting examples of which include networks and other communications channels, consumer appliances, computers, convergent devices such as WebTV, and optical media such as CD-ROM and DVD in all their current and future forms.

Example Access Techniques

Figures 3, 4A and 4B show example access techniques provided by player 52. In this example, upon disk 100 being loaded into player disk drive 80 (Figure 4A, block 400), the player controller 82 may direct drive 80 to fetch hidden keys 210 from disk 100 and use them to decrypt some or all of the encrypted key block 208 (Figure 4A, block 402). In this example, drive 80 may

store the keys so decrypted without exposing them to player controller 82 (e.g., by storing them within key store 212 within a secure decryption component such as an integrated circuit based decryption engine) (Figure 4A, block 404). The player 52 may control drive 80 to read the control set 204 (which may or may not be encrypted) from disk 100 (Figure 4A, block 406). The player microprocessor 82 may parse control set 204, ignore or discard those controls 222 that are beyond its capability, and maintain permissions and/or rights management information corresponding to the subset of controls that it can enforce (e.g., the "copy once" control 222(1)).

Player 52 may then wait for the user to provide a request via control inputs 58 and/or remote control unit 56. If the control input is a copy request ("yes" exit to Figure 4A, decision block 408), then player microprocessor 84 may query control 222(1) to determine whether copying is allowed, and if so, under what conditions (Figure 4A, decision block 410). Player 52 may refuse to copy the disk 100 if the corresponding control 222(1) forbids copying ("no" exit to Figure 4A, decision block 410), and may allow copying (e.g., by controlling drive 80 to sequentially access

all of the information on disk 100 and provide it to an output port not shown) if corresponding control 222(1) permits copying ("yes" exit to Figure 4A, decision block 410; block 412). In this example, player 52 may, upon making a copy, store an identifier associated with disk 100 within an internal, non-volatile memory (e.g., controller memory 86) or elsewhere if control 222(1) so requires. This stored disk identifier can be used by player 52 to enforce a "copy once" restriction (i.e., if the user tries to use the same player to copy the same disk more than once or otherwise as forbidden by control 222(1), the player can deny the request).

If the user requests one of properties 200 to be played or read ("yes" exit to Figure 4A, decision block 414), player controller 82 may control drive 80 to read the corresponding information from the selected property 200 (e.g., in a sequence as specified by metadata 202) and decrypt the read information as needed using the keys initially obtained from key block 208 and now stored within drive key storage 212 (Figure 4A, block 416).

Figure 4B is a variation on the Figure 4A process to accommodate a situation in which player 52 itself provides decryption keys for decrypting encrypted key block 208. In this

example, controller 82 may supply one or more decryption keys to drive 80 using a secure protocol such a Diffie-Hellman key agreement, or through use of a shared key known to both the drive and some other system or component to which the player 52 is or once was coupled (Figure 4B, block 403). The drive 80 may use these supplied keys to decrypt encrypted key block 208 as shown in Figure 4A, block 404, or it may use the supplied keys to directly decrypt content such as protected property 200 and/or protected metadata 202(2).

As a further example, the player 52 can be programmed to place a copy it makes of a digital property such as a film in encrypted form inside a tamper-resistant software container. The software container may carry with it a code indicating that the digital property is a copy rather than an original. The sending player 52 may also put its own unique identifier (or the unique identifier of an intended receiving device such as another player 52, a video cassette player or equipment 50) in the same secure container to enforce a requirement that the copy can be played only on the intended receiving device. Player 52 (or other receiving device) can be programmed to make no copies (or no

additional copies) upon detecting that the digital property is a copy rather than an original. If desired, a player 52 can be programmed to refuse to play a digital property that is not packaged with the player's unique ID.

Example Use of Analog Encoding Techniques

In another example, more comprehensive rights management information may be encoded by player 52 in the analog output using methods for watermarking and/or fingerprinting.Today, a substantial portion of the "real world" is analog rather than digital. Despite the pervasiveness of analog signals, existing methods for managing rights and protecting copyright in the analog realm are primitive or non-existent. For example:

- Quality degradation inherent in multigenerational analog copying has not prevented a multi-billion dollar pirating industry from flourishing.

- Some methods for video tape copy and pay per view protection attempt to prevent any copying at all of commercially released content, or allow only one

generation of copying. These methods can generally be easily circumvented.

- Not all existing devices respond appropriately to copy protection signals.

- Existing schemes are limited for example to "copy/no copy" controls.

- Copy protection for sound recordings has not been commercially implemented.

A related problem relates to the conversion of information between the analog and digital domains. Even if information is effectively protected and controlled initially using strong digital rights management techniques, an analog copy of the same information may no longer be securely protected.

For example, it is generally possible for someone to make an analog recording of program material initially delivered in digital form. Some analog recordings based on digital originals are of quite good quality. For example, a Digital Versatile Disk

("DVD") player may convert a movie from digital to analog format and provide the analog signal to a high quality analog home VCR. The home VCR records the analog signal. A consumer now has a high quality analog copy of the original digital property. A person could re-record the analog signal on a DVD-RAM. This recording will in many circumstances have substantial quality – and would no longer be subject to "pay per view" or other digital rights management controls associated with the digital form of the same content.

Since analog formats will be with us for a long time to come, rightsholders such as film studios, video rental and distribution companies, music studios and distributors, and other value chain participants would very much like to have significantly better rights management capabilities for analog film, video, sound recordings and other content. Solving this problem generally requires a way to securely associate rights management information with the content being protected.

In combination with other rights management capabilities, watermarking and/or fingerprinting, may provide "end to end"

secure rights management protection that allows content providers and rights holders to be sure their content will be adequately protected -- irrespective of the types of devices, signaling formats and nature of signal processing within the content distribution chain. This "end to end" protection also allows authorized analog appliances to be easily, seamlessly and cost-effectively integrated into a modern digital rights management architecture.

Watermarking and/or fingerprinting may carry, for example, control information that can be a basis for a Virtual Distribution Environment ("VDE") in which electronic rights management control information may be delivered over insecure (e.g., analog) communications channels. This Virtual Distribution Environment is highly flexible and convenient, accommodating existing and new business models while also providing an unprecedented degree of flexibility in facilitating ad hoc creation of new arrangements and relationships between electronic commerce and value chain participants -- regardless of whether content is distributed in digital and/or analog formats.

Watermarking together with distributed, peer-to-peer rights management technologies providers numerous advantages, including, but not limited to:

- An indelible and invisible, secure technique for providing rights management information.

- An indelible method of associating electronic commerce and/or rights management controls with analog content such as film, video, and sound recordings.

- Persistent association of the commerce and/or rights management controls with content from one end of a distribution system to the other -- regardless of the number and types of transformations between signaling formats (for example, analog to digital, and digital to analog).

- The ability to specify "no copy/ one copy/ many copies" rights management rules, and also more

complex rights and transaction pricing models (such as, for example, "pay per view" and others).

- The ability to fully and seamlessly integrate with comprehensive, general electronic rights management solutions.

- Secure control information delivery in conjunction with authorized analog and other non-digital and/or non-secure information signal delivery mechanisms.

- The ability to provide more complex and/or more flexible commerce and/or rights management rules as content moves from the analog to the digital realm and back.

- The flexible ability to communicate commerce and/or rights management rules implementing new, updated, or additional business models to authorized analog and/or digital devices.

Any or all of these features may be used in combination in and/or with the inventions disclosed in the present specification.

Briefly, watermarking and/or fingerprinting methods may, using "steganographical" techniques, substantially indelibly and substantially invisibly encode rights management and/or electronic commerce rules and controls within an information signal such as, for example, an analog signal or a digitized (for example, sampled) version of an analog signal, non-limiting examples of which may include video and/or audio data, that is then decoded and utilized by the local appliance. The analog information and stenographically encoded rights management information may be transmitted via many means, non-limiting examples of which may include broadcast, cable TV, and/or physical media, VCR tapes, to mention one non-limiting example. Any or all of these techniques may be used in combination in accordance with the inventions disclosed herein.

Watermarking and/or fingerprinting methods enable at least some rights management information to survive transformation of the video and/or other information from analog to digital and from

digital to analog format. Thus in one example, two or more analog and/or digital appliances may participate in an end-to-end fabric of trusted, secure rights management processes and/or events.

Example, More Capable Embodiments

As discussed above, the example control set shown in Figure 3B provides a comprehensive, flexible and extensible set of controls for use by both player 52 and computer equipment 62 (or other platform) depending upon the particular technical, security and other capabilities of the platform. In this example, player 52 has only limited technical and security capabilities in order to keep cost and complexity down in a mass-produced consumer item, and therefore may essentially ignore or fail to enable some or all of the controls 222 provided within control set 204. In another example, the cost of memory and/or processors may continue to decline and manufacturers may choose to expand the technical and security capabilities of player 52. A more capable player 52 will provide more powerful, robust, and flexible rights management capabilities.

Figure 5 shows an example arrangement permitting platform 60 including secure node 72 to have enhanced and/or different capabilities to use information and/or rights management information on disk 100, and Figure 6 shows an example access technique provided by the secure node. Referring to Figure 5, secure node 72 may be coupled to a network 150 whereas player 52 may not be—giving the secure node great additional flexibility in terms of communicating security related information such as audit trails, compensation related information such as payment requests or orders, etc. This connection of secure node 72 to network 150 (which may be replaced in any given application by some other communications technique such as insertion of a replaceable memory cartridge) allows secure node 72 to receive and securely maintain rights management control information such as an additional container 206' containing an additional control set 204'. Secure node 72 may use control set 204' in addition or in lieu of a control set 204 stored on disk 100. Secure node 72 may also maintain a secure cryptographic key store 212 that may provide cryptographic keys to be used in lieu of or in addition to any keys 208, 210 that may be stored on disk 100.

Because of its increased security and/or technical capabilities, secure node 72 may be able to use controls 222 within control set 204 that player 52 ignores or cannot use—and may be provided with further and/or enhanced rights and/or rights management capabilities based on control set 204' (which the user may, for example, order specially and which may apply to particular properties 200 stored on disk 100 and/or particular sets of disks).

Example Secure Node Access Techniques

The Figure 6 example access technique (which may be performed by platform 60 employing secure node 72, for example) involves, in this particular example, the secure node 72 fetching property identification information 220 from disk 100 (Figure 6, block 502), and then locating applicable control sets and/or rules 204 (which may be stored on disk 100, within secure node 72, within one or more repositories the secure node 72 accesses via network 150, and/or a combination of any or all of these techniques) (Figure 6, block 504). Secure node 72 then loads the necessary decryption keys and uses them to decrypt information as required (Figure 6, block 506). In one example, secure node 72 obtains the necessary keys from secure containers 206 and/or 206'

and maintains them within a protected processing environment such as SPU 164 or a software-emulated protected processing environment without exposing them externally of that environment. In another example, the secure node 72 may load the necessary keys (or a subset of them) into disk drive 82' using a secure key exchange protocol for use by the disk drive in decrypting information much in the same manner as would occur within player 52 in order to maintain complete compatibility in drive hardware.

Secure node 72 may monitor user inputs and perform requested actions based on the particular control set 204, 204'. For example, upon receiving a user request, secure node 72 may query the control set 204, 204' to determine whether it (they) permits the action the user has requested (Figure 6, block 508) and, if permitted, whether conditions for performing the requested operation have been satisfied (Figure 6, block 510). In this example, secure node 72 may effect the operations necessary to satisfy any such required conditions such as by, for example, debiting a user's locally-stored electronic cash wallet, securely requesting an account debit via network 150, obtaining and/or

checking user certificates to ensure that the user is within an appropriate class or is who he or she says he is, etc.—using network 150 as required (Figure 6, block 510). Upon all necessary conditions being satisfied, secure node 72 may perform the requested operation (and/or enable microprocessor 154 to perform the operation) (e.g., to release content) and may then generate secure audit records which can be maintained by the secure node and/or reported at the time or later via network 150 (Figure 6, block 512).

If the requested operation is to release content (e.g., make a copy of the content), platform 60 (or player 52 in the example above) may perform the requested operation based at least in part on the particular controls that enforce rights over the content. For example, the controls may prevent platform 60 from releasing content except to certain types of output devices that cannot be used to copy the content, or they may release the content in a way that discourages copying (e.g., by "fingerprinting" the copy with an embedded designation of who created the copy, by intentionally degrading the released content so that any copies made from it will be inferior, etc.). As one specific example, a

video cassette recorder (not shown) connected to platform 60 may be the output device used to make the copy. Because present generations of analog devices such as video cassette recorders are incapable of making multigenerational copies without significant loss in quality, the content provider may provide controls that permit content to be copied by such analog devices but not by digital devices (which can make an unlimited number of copies without quality loss). For example, platform 60 may, under control of digital controls maintained by secure node 72, release content to the video cassette recorder only after the video cassette recorder supplies the platform a digital ID that designates the output device as a video cassette recorder -- and may refuse to provide any output at all unless such a digital ID identifying the output device as a lower quality analog device is provided. Additionally or in the alternative, platform 60 may intentionally degrade the content it supplies to the video cassette recorder to ensure that no acceptable second -generation copies will be made. In another example, more comprehensive rights management information may be encoded by platform 60 in the analog output using watermarking and/or fingerprinting.

Additional Examples of Secure Container Usage

Figure 7 shows a basic example of a DVD medium 700 containing a kind of secure container 701 for use in DVDs in accordance with the present invention. As shown in this example, container 701 ("DigiBox for DVDs") could be a specialized version of a "standard" container tailored especially for use with DVD and/or other media, or it could, alternatively (in an arrangement shown later in Figure 8), be a fully "standard" container. As shown in this example, the specialized container 701 incorporates features that permit it to be used in conjunction with content information, metadata, and cryptographic and/or protection information that is stored on the DVD medium 700 in the same manner as would have been used had container 701 not been present. Thus, specialized container 701 provides compatibility with existing data formats and organizations used on DVDs and/or other media. In addition, a specialized container 701 can be tailored to support only those features necessary for use in support of DVD and/or other media, so that it can be processed and/or manipulated using less powerful or less expensive computing resources than would be required for complete support of a "standard" container object.

In this example, specialized "DVD only" container 701 includes a content object (a property) 703 which includes an "external reference" 705 to video title content 707, which may be stored on the DVD and/or other medium in the same manner as would have been used for a medium not including container 701. The video title content 707 may include MPEG-2 and/or AC-3 content 708, as well as scrambling (protection) information 710 and header, structure and/or meta data 711. External reference 705 contains information that "designates" (points to, identifies, and/or describes) specific external processes to be applied/executed in order to usecontent and other information not stored in container 701. In this example, external reference 705 designates video title content 707 and its components 708, 710, and 711. Alternatively, container 701 could store some or all of the video title content in the container itself, using a format and organization that is specific to container 701, rather than the standard format for the DVD and/or other medium 700.

In this example, container 701 also includes a control object (control set) 705 that specifies the rules that apply to use of video title content 707. As indicates by solid arrow 702, control object

705 "applies to" content object (property) 703. As shown in this example, rule 704 can specify that protection processes, for example CGMA or the Matsushita data scrambling process, be applied, and can designate, by external reference 709 contained in rule 704, data scrambling information 710 to be used in carrying out the protection scheme. The shorthand "do CGMA" description in rule 704 indicates that the rule requires that the standard CGMA protection scheme used for content on DVD media is to be used in conjunction with video title content 707, but a different example could specify arbitrary other rules in control object 705 in addition to or instead of the "do CGMA" rule, including other standard DVD protection mechanisms such as the Matsushita data scrambling scheme and/or other rights management mechanisms. External reference 709 permits rule 704 to be based on protection information 710 that is stored and manipulated in the same format and manner as for a DVD medium that does not incorporate container 701 and/or protection information that is meaningful only in the context of processing container 701.

Figure 8 shows a example of a DVD medium 800 containing a "standard" secure container 801. In this example, the

"standard" container provides all of the functionality (if desired) of the Figure 7 container, but may offer additional and/or more extensive rights management and/or content use capabilities than available on the "DVD only" container (e.g., the capacity to operate with various different platforms that use secure nodes).

Figure 9 shows a more complex example of DVD medium 800 having a standard container 901 that provides all of the functionality (if desired) of the Figure 7 container, and that can function in concert with other standard containers 902 located either on the same DVD medium or imported from another remote secure node or network. In this example, standard container 902 may include a supplementary control object 904 which applies to content object 903 of standard container 901. Also in this example, container 902 may provide an additional rule(s) such as, for example, a rule permitting/extending rights to allow up to a certain number (e.g., five) copies of the content available on DVD 900. This arrangement, for example, provides added flexibility in controlling rights management of DVD content between multiple platforms via access through "backchannels" such as via a set-top

box or other hardware having bi-directional communications capabilities with other networks or computers.

Additional Use of A DVD Disk With A Secure Container

Figure 10 illustrates the use of a "new" DVD disk—i.e., one that includes a special DVD secure container in the medium. This container may, in one example, be used or two possible use scenarios: a first situation in which the disk is used on an "old" player (DVD appliance, i.e., a DVD appliance that is not equipped with a secure node to provide rights management in accordance with the present invention; and a second situation in which the disk is used on a "new" player—i.e., a DVD appliance which is equipped with a secure node to provide rights management in accordance with the present invention. In this example, a secure node within the "new" player is configured with the necessary capabilities to process other copy protection information such as, for example, CGMA control codes and data scrambling formats developed and proposed principally by Matsushita.

For example, in the situation shown in Figure 10, the "new" player (which incorporates a secure node in accordance with the

present invention) can recognize the presence of a secure container on the disk. The player may then load the special DVD secure container from the disk into the resident secure node. The secure node opens the container, and implements and/or enforces appropriate rules and usage consequences associated with the content by applying rules from the control object. These rules are extremely flexible. In one example, the rules may, for example, call for use of other protection mechanisms (such as, for example, CGMA protection codes and Matsushita data scrambling) which can be found in the content (or property) portion of the container.

In another example shown in Figure 10, the special DVD container on the disk still allows the "old" player to use to a predetermined limited amount content material which may be used in accordance with conventional practices.

Example Use of A DVD Disk With No Secure Container

Referring now to Figure 11, a further scenario is discussed. Figure 11 illustrates use of an "old" DVD disk with two possible use examples: a first example in which the disk is used on an "old" player—i.e., a DVD appliance that is not equipped with a secure

node for providing rights management in accordance with the present invention—and a second example in which the disk is used on a "new" player (i.e., equipped with a secure node).

In the first case, the "old" player will play the DVD content in a conventional manner. In the second scenario, the "new" player will recognize that the disk does not have a container stored in the medium. It therefore constructs a "virtual" container in resident memory of the appliance. To do this, it constructs a container content object, and also constructs a control object containing the appropriate rules. In one particular example, the only applicable rule it need apply is to "do CGMA" -- but in other examples, additional and/or different rules could be employed. The virtual container is then provided to the secure node within the "new" player for implementing management of use rights in accordance with the present invention. Although not shown in Figures 10 and 11, use of "external references" may also be provided in both virtual and non-virtual containers used in the DVD context.

Example Illustrative Arrangements for Sharing, Brokering and Combining Rights When Operating in At Least Occasionally Connected Scenarios

As described above, the rights management resources of several different devices and/or other systems can be flexibly combined in diverse logical and/or physical relationships, resulting for example in greater and/or differing rights. Such rights management resource combinations can be effected through connection to one or more remote rights authorities. Figures 12-14 show some non-limiting examples of how rights authorities can be used in various contexts.

For example, Figure 12 shows a rights authority broker 1000 connected to a local area network (LAN) 1002. LAN 1002 may connect to wide area network if desired. LAN 1002 provides connectivity between rights authority broker 1000 and any number of appliances such as for example a player 50, a personal computer 60, a CD "tower" type server 1004. In the example shown, LAN 1002 includes a modem pool (and/or network

protocol server, not shown)1006 that allows a laptop computer 1008 to connect to the rights authority broker 1000 via dial-up lines 1010. Alternatively, laptop 1008 could communicate with rights authority broker 1000 using other network and/or communication means, such as the Internet and/or other Wide Area Networks (WANs). A disk player 50A may be coupled to laptop 1008 at the laptop location. In accordance with the teachings above, any or all of devices shown in Figure 12 may include one or more secure nodes 72.

Rights authority broker 1000 may act as an arbiter and/or negotiator of rights. For example, laptop 1008 and associated player 50A may have only limited usage rights when operating in a stand-alone configuration. However, when laptop 1008 connects to rights authority broker 1000 via modem pool 1006 and LAN 1002 and/or by other communication means, the laptop may acquire different and/or expanded rights to use disks 100 (e.g., availability of different content portions, different pricing, different extraction and/or redistribution rights, etc.) Similarly, player 50, equipment 60 and equipment 1004 may be provided with an enhanced and/or different set of disk usage rights through

communication with rights authority broker 1000 over LAN 1002. Communication to and from rights authority broker 1000 is preferably secured through use of containers of the type disclosed in the above-referenced Ginter et al. patent specification.

Figure 13 shows another example use of a rights authority broker 1000 within a home environment. In this example, the laptop computer 1008 may be connected to a home-based rights authority broker 1000 via a high speed serial IEEE 1394 bus and/or by other electronic communication means. In addition, rights authority broker 1000 can connect with any or all of:

- a high definition television 1100,
- one or more loudspeakers 1102 or other audio transducers,
- one or more personal computers 60,
- one or more set-top boxes 1030,
- one or more disk players 50,
- one or more other rights authority brokers 1000A-1000N and

- any other home or consumer equipment or appliances.

Any or all of the equipment listed above may include a secure node 72.

Figure 14 shows another example use of a rights authority broker 1000. In this example, rights authority broker 1000 is connected to a network 1020 such as a LAN, a WAN, the Internet, etc. Network 1020 may provide connectivity between rights authority broker 1000 and any or all of the following equipment:

- one or more connected or occasionally connected disk players 50A, 50B;
- one more networked computers 1022;
- one or more disk reader towers/servers 1004;
- one or more laptop computers 1008;
- one or more Commerce Utility Systems such as a rights and permissions clearinghouse 1024 (see Shear et al., "Trusted Infrastructure..." specification referenced above);

- one or more satellite or other communications uplinks 1026;
- one or more cable television head-ends 1028;
- one or more set-top boxes 1030 (which may be connected to satellite downlinks 1032 and/or disk players 50C);
- one or more personal computer equipment 60;
- one or more portable disk players 1034 (which may be connected through other equipment, directly, and/or occasionally unconnected;
- one or more other rights authority brokers 1000A-1000N; and
- any other desired equipment.

Any or all of the above-mentioned equipment may include one or more secure nodes 72. Rights authority broker 1000 can distribute and/or combine rights for use by any or all of the other components shown in Figure 14. For example, rights authority broker 100 can supply further

secure rights management resources to equipment connected to the broker via network 1020. Multiple equipment shown in Figure 14 can participate and work together in a permanently or temporarily connected network 1020 to share the rights management for a single node. Rights associated with parties and/or groups using and/or controlling such multiple devices and/or other systems can be employed according to underlying rights related rules and controls. As one example, rights available through a corporate executive's laptop computer 1008 might be combined with or substituted for, in some manner, the rights of one or more subordinate corporate employees when their computing or other devices 60 are coupled to network 1020 in a temporary networking relationship. In general, this aspect of the invention allows distributed rights management for DVD or otherwise packaged and delivered content that is protected by a distributed, peer-to-peer rights management. Such a distributed rights management can operate whether the DVD appliance or other content usage device is participating in a permanently or temporarily

connected network 1020, and whether or not the relationships among the devices and/or other systems participating in the distributed rights management arrangement are relating temporarily or have a more permanent operating relationship.

For example, laptop computer 1008 may have different rights available depending on the context in which that device is operating. For example, in a general corporate environment such as shown in Figure 12, the laptop 1008 may have one set of rights. However, the same laptop 1008 may be given a different set of rights when connected to a more general network 1020 in collaboration with specified individuals and/or groups in a corporation. The same laptop 1008 may be given a still different set of rights when connected in a general home environment such as shown by example in Figure 13. The same laptop 1008 could be given still different rights when connected in still other environments such as, by way of non-limiting example:

- a home environment in collaboration with specified individuals and/or groups,

- a retail environment,
- a classroom setting as a student,
- a classroom setting in collaboration with an instructor, in a library environment,
- on a factory floor,
- on a factory floor in collaboration with equipment enabled to perform proprietary functions, and so on.

As one more particular example, coupling a limited resource device arrangement such as a DVD appliance 50 shown in Figure 14 with an inexpensive network computer (NC) 1022 may allow an augmenting (or replacing) of rights management capabilities and/or specific rights of parties and/or devices by permitting rights management to be a result of a combination of some or all of the rights and/or rights management capabilities of the DVD appliance and those of an Network or Personal Computer (NC or PC). Such rights may be further augmented, or otherwise modified or replaced by the availability of rights management capabilities provided by a trusted (secure) remote network rights authority 1000.

The same device, in this example a DVD appliance 50, can thus support different arrays, e.g., degrees, of rights management capabilities, in disconnected and connected arrangements and may further allow available rights to result from the availability of rights and/or rights management capabilities resulting from the combination of rights management devices and/or other systems. This may include one or more combinations of some or all of the rights available through the use of a "less" secure and/or resource poor device or system which are augmented, replaced, or otherwise modified through connection with a device or system that is "more" or "differently" secure and/or resource rich and/or possesses differing or different rights, wherein such connection employs rights and/or management capabilities of either and/or both devices as defined by rights related rules and controls that describe a shared rights management arrangement.

In the latter case, connectivity to a logically and/or physically remote rights management capability can expand (by, for example, increasing the available secure rights management resources) and/or change the character of the rights available to the user of the DVD appliance 50 or a DVD appliance when such

device is coupled with an NC 1022, personal computer 60, and/or remote rights authority 1000. In this rights augmentation scenario, additional content portions may be available, pricing may change, redistribution rights may change (e.g., be expanded), content extraction rights may be increased, etc.

Such "networking rights management" can allow for a combination of rights management resources of plural devices and/or other systems in diverse logical and/or physical relationships, resulting in either greater or differing rights through the enhanced resources provided by connectivity with one or more "remote" rights authorities. Further, while providing for increased and/or differing rights management capability and/or rights, such a connectivity based rights management arrangement can support multi-locational content availability, by providing for seamless integration of remotely available content, for example, content stored in remote, Internet world wide web-based, database supported content repositories, with locally available content on one or more DVD discs 100.

In this instance, a user may experience not only increased or differing rights but may be able to use to both local DVD content

and supplementing content (i.e., content that is more current from a time standpoint, more costly, more diverse, or complementary in some other fashion, etc.). In such an instance, a DVD appliance 50 and/or a user of a DVD appliance (or other device or system connected to such appliance) may have the same rights, differing, and/or different rights applied to locally and remotely available content, and portions of local and remotely available content may themselves be subject to differing or different rights when used by a user and/or appliance. This arrangement can support an overall, profound increase in user content opportunities that are seamlessly integrated and efficiently available to users in a single content searching and/or usage activity.

Such a rights augmenting remote authority 1000 may be directly coupled to a DVD appliance 50 and/or other device by modem (see item 1006 in Figure 12) and/or directly or indirectly coupled through the use of an I/O interface, such as a serial 1394 compatible controller (e.g., by communicating between a 1394 enabled DVD appliance and a local personal computer that functions as a smart synchronous or asynchronous information communications interface to such one or more remote authorities,

including a local PC 60 or NC 1022 that serves as a local rights management authority augmenting and/or supplying the rights management in a DVD appliance) and/or by other digital communication means such as wired and/or wireless network connections.

Rights provided to, purchased, or otherwise acquired by a participant and/or participant DVD appliance 50 or other system can be exchanged among such peer-to-peer relating devices and/or other systems so long as they participate in a permanently or temporarily connected network. 1020. In such a case, rights may be bartered, sold, for currency, otherwise exchanged for value, and/or loaned so long as such devices and/or other systems participate in a rights management system, for example, such as the Virtual Distribution Environment described in Ginter, et al., and employ rights transfer and other rights management capabilities described therein. For example, this aspect of the present invention allows parties to exchange games or movies in which they have purchased rights. Continuing the example, an individual might buy some of a neighbor's usage rights to watch a movie, or transfer to another party credit received from a game

publisher for the successful superdistribution of the game to several acquaintances, where such credit is transferred (exchanged) to a friend to buy some of the friend's rights to play a different game a certain number of times, etc.

Example Virtual Rights Process

Figures 15A-15C shows an example of a process in which rights management components of two or more appliances or other devices establish a virtual rights machine environment associated with an event, operation and/or other action. The process may be initiated in a number of ways. In one example, an appliance user (and/or computer software acting on behalf of a user, group of users, and/or automated system for performing actions) performs an action with a first appliance (e.g., requesting the appliance to display the contents of a secure container, extract a portion of a content element, run a protected computer program, authorize a work flow process step, initiate an operation on a machine tool, play a song, etc.) that results in the activation of a rights management component associated with such first appliance (Figure 15A, block 1500). In other examples, the process may get started in response to an automatically generated event (e.g., based

on a time of day or the like), a random or pseudo-random event, and/or a combination of such events with a user-initiated event.

Once the process begins, a rights management component such as a secure node 72 (for example, an SPE and/or HPE as disclosed in Ginter et al.) determines which rights associated with such first appliance, if any, the user has available with respect to such an action (Figure 15A, block 1502). The rights management component also determines the coordinating and/or cooperating rights associated with such an action available to the user located in whole or in part on other appliances (Figure 15A, block 1502).

In one example, these steps may be performed by securely delivering a request to a rights authority server 1000 that identifies the first appliance, the nature of the proposed action, and other information required or desired by such a rights authority server. Such other information may include, for example:

- the date and time of the request,
- the identity of the user,
- the nature of the network connection,

- the acceptable latency of a response, etc.), and/or
- any other information.

In response to such a request, the rights authority server 1000 may return a list (or other appropriate structure) to the first appliance. This list may, for example, contain the identities of other appliances that do, or may, have rights and/or rights related information relevant to such a proposed action.

In another embodiment, the first appliance may communicate (e.g., poll) a network with requests to other appliances that do, or may, have rights and/or rights related information relevant to such proposed action. Polling may be desirable in cases where the number of appliances is relatively small and/or changes infrequently. Polling may also be useful, for example, in cases where functions of a rights authority server 1000 are distributed across several appliances.

The rights management component associated with the first appliance may then, in this example, check the security level(s) (and/or types) of devices and/or users of other appliances that do, or may, have rights and/or rights related information relevant to

such an action (Figure 15A, block 1506). This step may, for example, be performed in accordance with the security level(s) and/or device type management techniques disclosed in Sibert and Van Wie, and the user rights, secure name services and secure communications techniques disclosed in Ginter et al. Device and/or user security level determination may be based, for example, in whole or in part on device and/or user class.

The rights management component may then make a decision as to whether each of the other appliance devices and/or users have a sufficient security level to cooperate in forming the set of rights and/or rights related information associated with such an action (Figure 15A, block 1508). As each appliance is evaluated, some devices and/or users may have sufficient security levels, and others may not. In this example, if a sufficient security level is not available ("No" exit to decision block 1508), the rights management component may create an audit record (for example, an audit record of the form disclosed in Ginter et al.) (Figure 15A, block 1510), and may end the process (Figure 15A, block 1512). Such audit record may be for either immediate transmission to a responsible authority and/or for local storage and later

transmission, for example. The audit recording step may include, as one example, incrementing a counter that records security level failures (such as the counters associated with summary services in Ginter et al.)

If the devices and/or users provide the requisite security level ("Yes" exit to block 1508), the rights management component in this example may make a further determination based on the device and/or user class(es) and/or other configuration and/or characteristics (Figure 15B, block 1514). Such determination may be based on any number of factors such as for example:

- the device is accessible only through a network interface that has insufficient throughput;
- devices in such a class typically have insufficient resources to perform the action, or relevant portion of the action, at all or with acceptable performance, quality, or other characteristics;

- the user class is inappropriate due to various conditions (e.g., age, security clearance, citizenship, jurisdiction, or any other class-based or other user characteristic); and/or
- other factors.

In one example, decision block 1514 may be performed in part by presenting a choice to the user that the user declines.

If processes within the rights management component determines that such device and/or user class(es) are inappropriate("No" exit to block 1514), the rights management component may write an audit record if required or desired (Figure 15B, block 1516) and the process may end (Figure 15B, block 1518).

If, on the other hand, the rights management component determines that the device and/or user classes are appropriate to proceed ("Yes" exit to block 1514), the rights management component may determine the rights and resources available for performing the action on the first appliance and the other appliances acting together (Figure 15B, block 1520). This step may be performed, for example, using any or all of the method

processing techniques disclosed in Ginter et al. For example, method functions may include event processing capabilities that formulate a request to each relevant appliance that describes, in whole or in part, information related to the action, or portion of the action, potentially suitable for processing, in whole or in part, by such appliance. In this example, such requests, and associated responses, may be managed using the reciprocal method techniques disclosed in Ginter et al. If such interaction requires additional information, or results in ambiguity, the rights management component may, for example, communicate with the user and allow them to make a choice, such as making a choice among various available, functionally different options, and/or the rights management component may engage in a negotiation (for example, using the negotiation techniques disclosed in Ginter et al.) concerning resources, rights and/or rights related information.

The rights management component next determines whether there are sufficient rights and/or resources available to perform the requested action (Figure 15B, decision block 1522). If there are insufficient rights and/or resources available to perform the action ("No" exit to block 1522), the rights management component may

write an audit record (Figure 15B, block 1524), and end the process (Figure 15B, block 1526).

In this example, if sufficient rights and/or resources are available ("Yes" exit to block 1522), the rights management component may make a decision regarding whether additional events should be processed in order to complete the overall action (Figure 15B, block 1528). For example, it may not be desirable to perform only part of the overall action if the necessary rights and/or resources are not available to complete the action. If more events are necessary and/or desired ("Yes" exit to block 1528), the rights management component may repeat blocks 1520, 1522 (and potentially perform blocks 1524, 1526) for each such event.

If sufficient rights and/or resources are available for each of the events ("No" exit to block 1528), the rights management component may, if desired or required, present a user with a choice concerning the available alternatives for rights and/or resources for performing the action (Figure 15B, block 1530). Alternatively and/or in addition, the rights management component may rely on user preference information (and/or defaults) to "automatically" make such a determination on behalf

of the user (for example, based on the overall cost, performance, quality, etc.). In another embodiment, the user's class, or classes, may be used to filter or otherwise aid in selecting among available options. In still another embodiment, artificial intelligence (including, for example, expert systems techniques) may be used to aid in the selection among alternatives. In another embodiment, a mixture of any or all of the foregoing (and/or other) techniques may be used in the selection process.

If there are no acceptable alternatives for rights and/or resources, or because of other negative aspects of the selection process (e.g., a user presses a "Cancel" button in a graphical user interface, a user interaction process exceeds the available time to make such a selection, etc.), ("No" exit to block 1530) the rights management component may write an audit record (Figure 15B, block 1532), and end the process (Figure 15B, block 1534).

On the other hand, if a selection process identifies one or more acceptable sets of rights and/or resources for performing the action and the decision to proceed is affirmative ("Yes" exit to block 1530), the rights management component may perform the proposed action using the first appliance alone or in combination

with any additional appliances (e.g., a rights authority 1000, or any other connected appliance) based on the selected rights and/or resources (Figure 15C, block 1536). Such cooperative implementation of the proposed actions may include for example:

- performing some or all of the action with the first appliance;
- performing some or all of the action with one or more appliances other than the first appliance (e.g., a rights authority 1000 and/or some other appliance);
- performing part of the action with the first appliance and part of the action with one or more other appliances; or
- any combination of these.

For example, this step may be performed using the event processing techniques disclosed in Ginter et al.

As one illustrative example, the first appliance may have all of the resources necessary to perform a particular task (e.g., read certain information from an optical disk), but may lack the rights necessary to do so. In such an instance, the first appliance may

obtain the additional rights it requires to perform the task through the steps described above. In another illustrative example, the first appliance may have all of the rights required to perform a particular task, but it may not have the resources to do so. For example, the first appliance may not have sufficient hardware and/or software resources available to it for accessing, processing or otherwise using information in certain ways. In this example, step 1536 may be performed in whole or in part by some other appliance or appliances based in whole or in part on rights supplied by the first appliance. In still another example, the first appliance may lack both rights and resources necessary to perform a certain action, and may rely on one or more additional appliances to supply such resources and rights.

In this example, the rights management component may, upon completion of the action, write one or more audit records (Figure 15C, block 1538), and the process may end (Figure 15C, block 1540).

* * * * * *

An arrangement has been described which adequately satisfies current entertainment industry requirements for a low cost, mass-produceable digital video disk or other high capacity disc copy protection scheme but which also provides enhanced, extensible rights management capabilities for more advanced and/or secure platforms and for cooperative rights management between devices of lessor, greater, and/or differing rights resources. While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the invention.

We Claim:

1. An electronic appliance including:

a disk use arrangement for at least one of (a) reading information from, and (b) writing information to, a digital versatile disk optical storage medium; and

a secure node coupled to the disk use arrangement, the secure node providing at least one rights management process.

2. An electronic appliance including:

a disk use arrangement for at least one of (a) reading information from, and (b) writing information to, a digital versatile disk optical storage medium; and

at least one processing arrangement coupled to the disk use arrangement, the processing arrangement selecting at least some control information associated with information recorded on the storage medium based at least in part on the class of the appliance and/or the user of the appliance.

3. A system as in claim 2 wherein the processing arrangement selects a subset of control information used on another appliance and/or class of appliance.

4. A system as in claim 2 wherein the processing arrangement selects different control information from the information selected by another appliance and/or class of appliance.

5. A system as in claim 2 wherein at least some of the control information comprises an analog signal.

6. A system as in claim 2 wherein at least some of the control information comprises digitally encoded information.

7. In an appliance capable of using digital versatile disks, a method including the following steps:

at least one of (a) reading information from, and (b) writing information to, a digital versatile disk optical storage medium; and

selecting at least some control information associated with information recorded on the storage medium based at least in part on the class of the appliance and/or the user of the appliance.

8. A method as in claim 7 wherein the selecting step includes the step of selecting a subset of control information used on another appliance and/or class of appliance.

9. A method as in claim 7 wherein the selecting step includes the step of selecting, from control information stored on the storage medium, a different set of control information than the control information selected by another appliance and/or class of appliance.

10. An electronic appliance including:

a disk use arrangement for reading information from a digital versatile disk optical storage medium; and

at least one processing arrangement coupled to the disk use arrangement, the processing arrangement protecting information read from the storage medium.

11. An appliance as in claim 10 wherein the processing arrangement includes a rights management arrangement that applies at least one rights management technique to the read information.

12. An appliance as in claim 10 wherein the appliance further includes at least one port compliant at least in part with the IEEE 1394-1995 high speed serial bus standard, and the processing arrangement couples the protected information to the port.

13. In an electronic appliance, a method including the following steps:

reading information from a digital versatile disk optical storage medium; and

protecting the information read from the optical storage medium.

14. A method as in claim 13 wherein the protecting step includes the step of applying at least one rights management technique to the read information.

15. A method as in claim 13 further including the step of sending the protected information to an IEEE 1394 port.

16. An electronic appliance including:

a disk use arrangement for using information stored, or to be stored, on a digital versatile disk optical storage medium; and

at least one protecting arrangement coupled to the disk use arrangement and also coupled to receive at least one analog signal, the protecting arrangement creating protected digital information based at least in part on the analog signal.

17. In an electronic appliance, a method including the following steps:

receiving at least one analog signal; and

creating protected digital content based at least in part on the analog signal for storage on a digital versatile disk.

18. In an electronic appliance, a method including the following steps:

reading at least one analog signal from a digital versatile disk;

creating protected digital content based at least in part on the analog signal; and

outputting the protected digital content.

19. An electronic appliance including:

a disk use arrangement for using information stored, or to be stored, on a digital versatile disk optical storage medium; and

at least one rights management arrangement coupled to the disk use arrangement, the rights management arrangement treating the storage medium and/or information obtained from the storage medium differently depending on the geographical and/or jurisdictional context of the appliance.

20. In an electronic appliance, a method including the steps of:

reading information from at least one digital versatile disk; and

performing at least one rights management operation based at least in part on the geographical and/or jurisdictional context of the appliance.

21. An electronic appliance including:

a disk use arrangement for using at least one secure container stored on a digital versatile disk optical storage medium; and

at least one rights management arrangement coupled to the disk use arrangement, the rights management arrangement processing the secure container.

22. In an electronic appliance, a method including the following steps:

reading at least one secure container from at least one digital versatile disk; and

performing at least one rights management operation on the secure container.

23. An electronic appliance including:

at least one rights management arrangement for generating and/or modifying at least one secure container for storage onto a digital versatile disk optical storage medium.

24. In an electronic appliance, a method including the step of performing at least one rights management operation on at least one secure container for storage onto a digital versatile disk optical storage medium.

25. A digital versatile disk use system and/or method characterized in that the system and/or method uses at least one secure container.

26. A digital versatile disk use system and/or method characterized in that the system and/or method uses at least one

secure container of the type disclosed in PCT Publication No. WO 96/27155.

27. An electronic appliance including:

a disk use arrangement for writing information onto and/or reading information from a digital versatile disk optical storage medium; and

a secure arrangement that securely manages information on the storage medium such that at least a first portion of the information may be used on at least a first class of appliance while at least a second portion of the information may be used on at least a second class of appliance

28. In an electronic appliance, a method including the following steps:

reading information from and/or writing information to at least one digital versatile disk optical storage medium;

using at least a first portion of the information on at least a first class of appliance; and

using at least a second portion of the information on at least a second class of appliance.

29. A system including first and second classes of electronic appliances each including a secure processing arrangement, the first appliance class secure arrangement securely managing and/or using at least a first portion of the information, the second appliance class secure arrangement securely managing and/or using at least a second portion of the information.

30. A system as in claim 29 wherein the first and second information portions are different, and the second appliance class secure arrangement does not use the first information portion.

31. A system as in claim 29 wherein the first appliance class does not use the second information portion.

32. In a system including first and second classes of electronic appliances each including a secure arrangement, a method comprising:

(a) securely managing and/or using at least a first portion of the information with the first appliance class secure arrangement, and

(b) securely managing and/or using at least a second portion of the information with the second appliance class secure arrangement.

33. A method as in claim 32 wherein the first and second information portions are different, and step (b) does not use the first information portion.

34. A method as in claim 32 wherein step (a) does not use the second information portion.

35. An electronic appliance including:

a disk use arrangement for writing information onto and/or reading information from a digital versatile disk optical storage medium; and

a secure arrangement that securely stores and/or transmits information associated with at least one of payment, auditing, controlling and/or otherwise managing content recorded on the storage medium.

36. In an electronic appliance, a method including the following steps:

reading information from and/or writing information to at least one digital versatile disk optical storage medium; and

securely storing and/or transmitting information associated with at least one of payment, auditing, controlling and/or otherwise managing content recorded on the storage medium.

37. An electronic appliance including:

a disk use arrangement for writing information onto and/or reading information from a digital versatile disk optical storage medium;

a cryptographic engine coupled to the disk use arrangement, the engine using at least one cryptographic key; and

a secure arrangement that securely updates and/or replaces at least one cryptographic key used by the cryptographic engine to at least in part modify the scope of information usable by the appliance.

38. A method of operating an electronic appliance including:

writing information onto and/or reading information from a digital versatile disk optical storage medium;

using at least one cryptographic key in conjunction with said information; and

securely updating and/or replacing at least one cryptographic key used by the cryptographic engine to at least in part modify the scope of information useable by the appliance.

39. A digital versatile disk appliance characterized in that at least one cryptographic key used by the appliance is securely updated and/or replaced to at least in part modify the scope of information that can be used by the appliance.

40. An appliance as in claim 39 further characterized in that the key updating and/or replacing is based on class of appliance.

41. An electronic appliance having a class associated therewith, characterized in that at least one cryptographic key set used by the appliance class is selected to help ensure security of information released from at least one digital versatile disk.

42. A digital camera for generating at least one image to be written onto a digital versatile disk optical storage medium, characterized in that the camera includes at least one information protecting arrangement that at least in part protects the image so that the information is persistently protected through subsequent processes such as editing, production, writing onto a digital versatile disk, and/or reading from a digital versatile disk.

43. A digital camera for generating image information that can be written onto a digital versatile disk optical storage medium, a method comprising:

capturing at least one image with a digital camera; and

protecting information provided by the digital camera so that the information is selectively persistently protected through subsequent processes such as distribution, editing and/or production, writing onto the digital versatile disk optical storage medium, and/or reading from the digital versatile disk optical storage medium.

44. In an electronic appliance including a disk use arrangement, a method comprising:

reading information from at least one digital versatile disk optical storage medium; and

persistently protecting at least some of the read information through at least one subsequent editing and/or production process.

45. In an electronic appliance, a method including the following steps:

reading information from and/or writing information to at least one digital versatile disk optical storage medium; and

securely managing information on the storage medium, including the step of using at least a first portion of the information on at least a first class of appliance, and using at least a second portion of the information on at least a second class of appliance.

46. A method of providing copy protection and/or use rights management of at least one digital property content and/or secure container to be stored and/or distributed on a digital versatile disk medium, comprising the step(s) of:

providing a set of use control(s) within a cryptographically encapsulated data structure having a predetermined format, the data structure format defining at least one secure software container for providing use rights information for digital property content to be stored on the digital versatile disk medium.

47. A method as in claim 46 further including the step of using at least one digital property content stored on an optical disk in accordance with the use controls, including the step of using a prescribed secure cryptographic key or set of cryptographic keys for using rights information.

48. A method as in claim 46 further including the step of decrypting control rules and/or other selected encrypted

information content encapsulated in the software container using at least one set of cryptographic keys.

49. A method as in claim 46 further including the step of applying decrypted control rules to regulate use in accordance with control information contained within said control rules, so as to facilitate management of a diverse set of use and distribution rights which may be specific to different users and/or optical disk appliances.

50. A method of providing rights management of digital property stored on digital versatile disk according to claim 46 wherein said secure container data structure comprises:

one or more content objects comprising digital property content; and

one or more control objects comprising a set of control rules defining copy protection, use and distribution rights to digital property content stored on the optical disk.

51. A method of providing rights management of digital property stored on a digital versatile disk according to claim 46, wherein a content object further comprises one or more reference pointers to digital property content stored elsewhere on the digital versatile disk.

52. A method of providing rights management of digital property stored on a digital versatile disk according to claim 46, wherein a control object further comprises one or more reference pointers to control information stored elsewhere on the digital versatile disk.

53. A method of providing rights management of digital property stored on digital versatile disk according to claim 46, wherein digital information stored on said optical disk includes one or more metadata blocks comprising further information used in conjunction with the control rules to use digital property content stored elsewhere on the optical disk.

54. A method of providing rights management of digital property stored on digital versatile disk according to claim 46, wherein a metablock may be either of a protected type or of an unprotected type.

55. An arrangement for implementing a rights management system for controlling copy protection, use and/or distribution rights to multi-media digital property content stored or otherwise contained on a digital versatile disk, comprising:

an encrypted data structure defining a secure information container stored on an optical disk medium, the encrypted data structure including and/or referencing at least one content object and at least one control object associated with the content object, said content object comprising digital property content and said control object comprising rules defining use rights to the digital property content.

56. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a content object further comprises one or more reference pointers to digital property content stored elsewhere on the digital versatile disk.

57. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises one or more reference pointers to control information stored elsewhere on the digital versatile disk.

58. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein an control object further comprises information for controlling various operations of an optical disk appliance or computer.

59. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises information for controlling various operations of an optical disk appliance or computer.

60. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises a rule specifying decoding and/or enforcement of CGMA encoded copy protection rules associated with the digital content property.

61. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises a rule specifying at least one content scrambling system compatible encoding/decoding of digital property content.

62. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein said optical disk contains a block of stored information comprising encrypted keys used for decryption of said encrypted data structure.

63. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein said optical disk contains a block of stored information comprising hidden keys used for decryption of said encrypted keys.

64. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a content object further comprises one or more reference pointers to digital property content stored on a separate storage medium.

65. A rights management system for providing copy protection, use and/or distribution rights management for multi-media digital property content stored or otherwise contained on a digital versatile disk for access by an optical disk player device that uses digital property content stored on said optical disk medium, wherein said appliance includes a microprocessor controller for decrypting and using control rules and other selected encrypted information content encapsulated in the secure container by using a prescribed cryptographic key and applying said decrypted control rules to regulate use in accordance with control information contained within said control rules, so as to facilitate management of a diverse set of use and/or distribution rights which may be specific to different users and/or optical disk appliances, the system including:

an optical disk medium having stored thereon an encrypted data structure defining a secure information container, the encrypted data structure comprising and/or referencing at least one content object and at least one control object, said content object comprising digital property content, said control object

comprising rules defining use rights associated with the digital property.

66. A method for providing copy protection, use and distribution rights management of multi-media digital property stored on and/or distributed via digital versatile disk, said optical disk medium having stored thereon an encrypted data structure defining a secure container for housing rights and/or copy protection information pertaining to digital property content stored on the optical disk, wherein an optical disk player appliance for using digital property content stored on an optical disk must utilize a prescribed secure cryptographic key or set of keys to use the secure container, said data structure comprising one or more content objects comprising digital property content and one or more control objects comprising a set of rules defining use rights to digital property, comprising the steps of:

(a) decrypting control rules and other selected encrypted information content encapsulated in the secure container using one or more cryptographic keys; and

(b) applying decrypted control rules to regulate use and/or distribution of digital property content stored on the optical disk in accordance with control information contained within the control rules, so as to provide customized use and/or distribution rights that are specific to different optical disk user platforms and/or optical disk appliances.

67. A rights management system for providing copy protection, use and/or distribution rights management of digital property stored or otherwise contained on a digital versatile disk, comprising:

a secure container means provided on an optical disk medium for cryptographically encapsulating digital property content stored on the optical disk, said container means comprising a content object means for containing digital property content and a control object means for containing control rules for regulating use and/or distribution of said digital property content stored on the optical disk.

68. The rights management system of claim 67 wherein an optical disk player appliance for using information stored on an optical disk comprises a secure node means for using said secure container means provided on an optical disk and implementing said control rules to control operation of said player appliance to regulate use of said digital property content.

69. In a system including plural electronic appliances at least temporarily connected to one another, a rights authority broker that determines what appliances are connected and specifies at least one rights management context depending on said determination.

70. An electronic appliance at least temporarily connected to a rights authority broker, the electronic appliance receiving at least one rights context from the rights authority broker when the device is connected to the rights authority broker.

71. A first electronic appliance at least temporarily connected to a second electronic appliance, the first

electronic appliance selecting between at least first and second rights management contexts depending at least in part on whether the first appliance is connected to the second electronic appliance.

72. In a system including first and second electronic appliances that may be selectively coupled to communicate with one another, an arrangement for defining at least one different rights management control based at least in part on whether the first and second electronic appliances are connected.

73. A method of defining at least one rights management context comprising:

(a) determining whether a first electronic appliance is present; and

(b) defining at least one rights management control set based at least in part on the determining step (a).

74. A method of defining at least one rights management context including:

(a) coupling an optical disk storing information to an electronic appliance that can be selectively connected to a rights management broker;

(b) determining whether the electronic appliance is currently coupled to a rights management broker; and

(c) conditioning at least one aspect of use of at least some of the information stored on the optical disk based on whether the electronic appliance is coupled to the rights management broker.

75. A method as in claim 74 wherein step (c) includes the step of obtaining at least one rights management context from the rights management broker.

76. A method as in claim 74 wherein step (c) includes the step of obtaining at least one combined control set from the rights management broker.

77. A method of defining at least one rights management context including:

(a) coupling an optical disk storing information to an electronic appliance;

(b) using at least some of the information stored on the optical disk based on a first rights management context;

(c) coupling the electronic appliance to a rights management broker; and

(d) concurrently with step (c), using at least some of the information stored on the optical disk based on a second rights management context different from the first rights management context

78. An electronic appliance include a secure node and an optical disk reader, the electronic appliance applying different rights management contexts to protected information stored on an optical disk coupled to the optical disk reader depending at least in part on whether the electronic appliance is coupled to at least one additional secure node.

79. An electronic appliance including:

an optical disk reading and/or writing arrangement;

a secure node coupled to the optical disk reading and/or writing arrangement, the secure node performing at least one rights management related function with respect to at least some information read by the optical disk reading and/or writing arrangement; and

at least one serial bus port coupled to the secure node, the serial bus port for providing any or all of the functions, structures, protocols and/or methods of IEEE 1394-1995.

80. A digital versatile disk appliance including:

means for watermarking content; and

serial bus means for communicating the watermarked content,

wherein the serial bus means complies with IEEE 1394-1995.

81. An optical disk reading and/or writing device including:

at least one secure node capable of watermarking content and/or processing watermarked content; and

an IEEE 1394-1995 serial bus port.

82. An optical disk using device comprising:

a secure processing unit; and

an IEEE 1394-1995 serial bus port.

83. A device as in claim 82 wherein the secure processing unit includes a channel manager.

84. A device as in claim 82 wherein the secure processing unit executes a rights operating system in whole or in part.

85. A device as in claim 82 wherein the secure processing unit includes a tamper-resistant barrier.

86. A device as in claim 82 wherein the secure processing unit includes an encryption/decryption engine.

87. A rights cooperation method comprising:

(a) connecting an appliance to at least one further appliance;

(b) determining whether the first and/or further appliance and/or user(s) of said first and/or further appliance have appropriate rights and/or resources for performing an action; and

(c) selectively performing the action based at least in part on the determination.

88. A rights cooperation method comprising:

(a) connecting an appliance to at least one further appliance;

(b) determining whether the first and/or further appliance and/or user(s) of said first and/or further appliance have appropriate security for performing an action; and

(c) cooperating between the first and further appliance to selectively perform the action.

89. A cooperative rights management arrangement comprising:

a communications arrangement that allows at least first and second appliances to communicate; and

an arrangement that processes at least one event based at least in part on assessing and/or pooling rights and/or resources between the first and second appliances.

90. An optical disk using system and/or method including at least some of the elements shown in Figure 1A.

91. An optical disk using system and/or method including at least some of the elements shown in Figure 1B.

92. An optical disk using system and/or method including at least some of the elements shown in Figure 1C.

93. An optical disk using system and/or method including at least some of the elements shown in Figure 2A.

94. An optical disk using system and/or method including at least some of the elements shown in Figure 2B.

95. An optical disk using system and/or method including at least some of the elements shown in Figure 3.

96. An optical disk using system and/or method using at least some of the elements shown in Figure 3A.

97. An optical disk using system and/or method using at least some of the control set elements shown in Figure 3B.

98. An optical disk using system and/or method using at least some of the elements shown in Figure 4A.

99. An optical disk using system and/or method using at least some of the elements shown in Figure 4B.

100. An optical disk using system and/or method using at least some of the elements shown in Figure 5.

101. An optical disk using system and/or method using at least some of the elements shown in Figure 6.

102. An optical disk using system and/or method using at least some of the elements shown in Figure 7.

103. An optical disk using system and/or method using at least some of the elements shown in Figure 8.

104. An optical disk using system and/or method using at least some of the elements shown in Figure 9.

105. An optical disk using system and/or method using at least some of the elements shown in Figure 10.

106. An optical disk using system and/or method using at least some of the elements shown in Figure 11.

107. An optical disk using system and/or method including at least some of the elements shown in Figure 12.

108. An optical disk using system and/or method including at least some of the elements shown in Figure 13.

109. An optical disk using system and/or method including at least some of the elements shown in Figure 14.

110. A system and/or method including some or all of the elements shown in Figures 15A-15C.

111. A system and/or method as in any one of the preceding claims, further including, in combination, any element described in any one of the following prior patent specifications:

PCT Publication No. WO 96/27155;

European Patent No. EP 329681;

PCT Application No. PCT/US96/14262;

U.S. Patent Application Serial No. 08/689,606; and/or

U.S. Patent Application Serial No. 08/689,754.

112. A system or process as in any of the preceding claims wherein the phrase "high capacity optical disk" is substituted for "digital versatile disk."

113. A method of clearing or otherwise processing information resulting at least in part from one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

114. A system and/or method for defining rules for use in one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

115. A system and/or method for defining rules and associated content for use in one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

116. A system and/or method for producing an optical disk for use with one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

117. A system and/or method for clearing audit information from one or more appliances and/or methods as defined in any of the preceding claims.

118. In an network including at least one electronic appliance that reads information from and/or writes information to at least one digital versatile disk optical storage medium, and securely communicates information associated with at least one of

payment, auditing, usage, access, controlling and/or otherwise managing content recorded on the storage medium, a method of processing said communicated information including the step of generating at least one payment request and/or order based at least in part on the information.

119. A method of defining at least one control set for storage on a high capacity optical disk that can storage images, audio, text and/or other information, the high capacity optical disk for use by any of plural different electronic appliance types, the method including the step of specifying at least one control that provides different conditions and/or consequences depending upon at least one of the following:

electronic appliance class;

electronic appliance security;

electronic appliance user class;

electronic appliance connectivity;

electronic appliance resources;

electronic appliance access to resources; and

rights management cooperation between plural electronic appliances.

Fig.1B

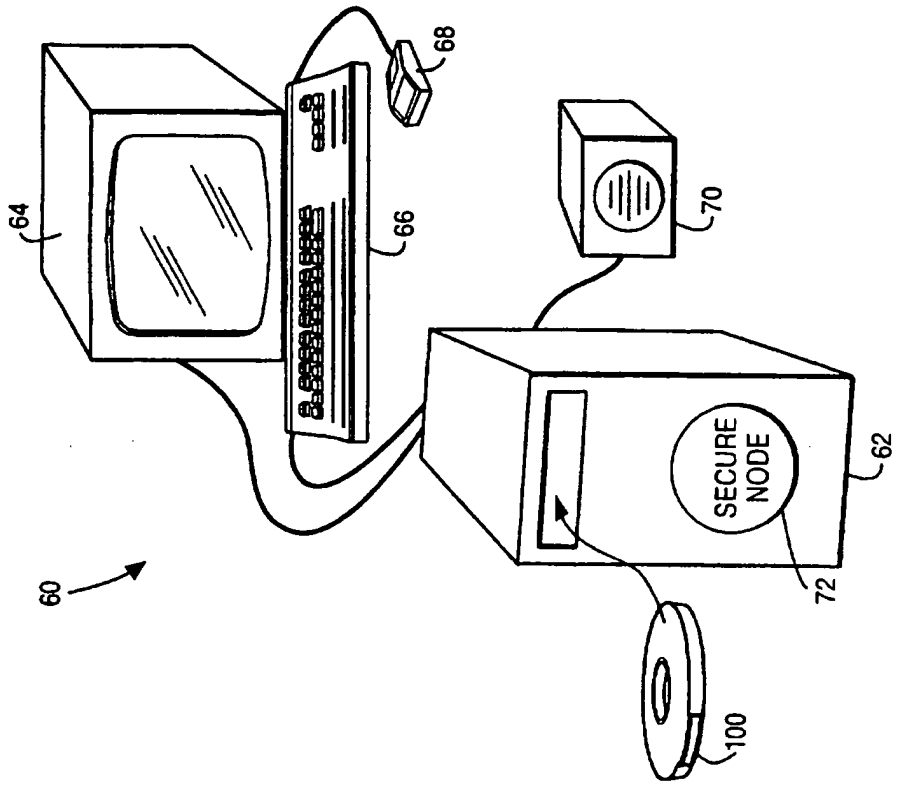
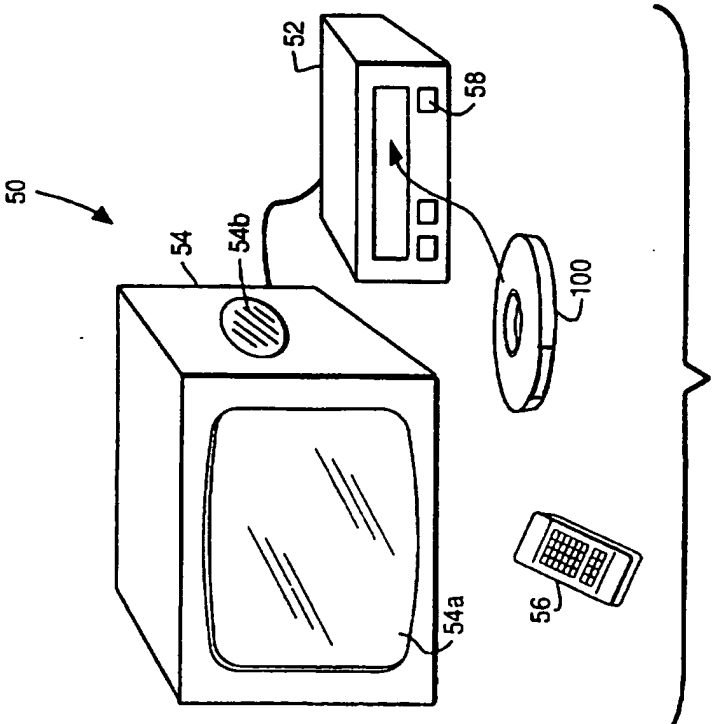


Fig.1A



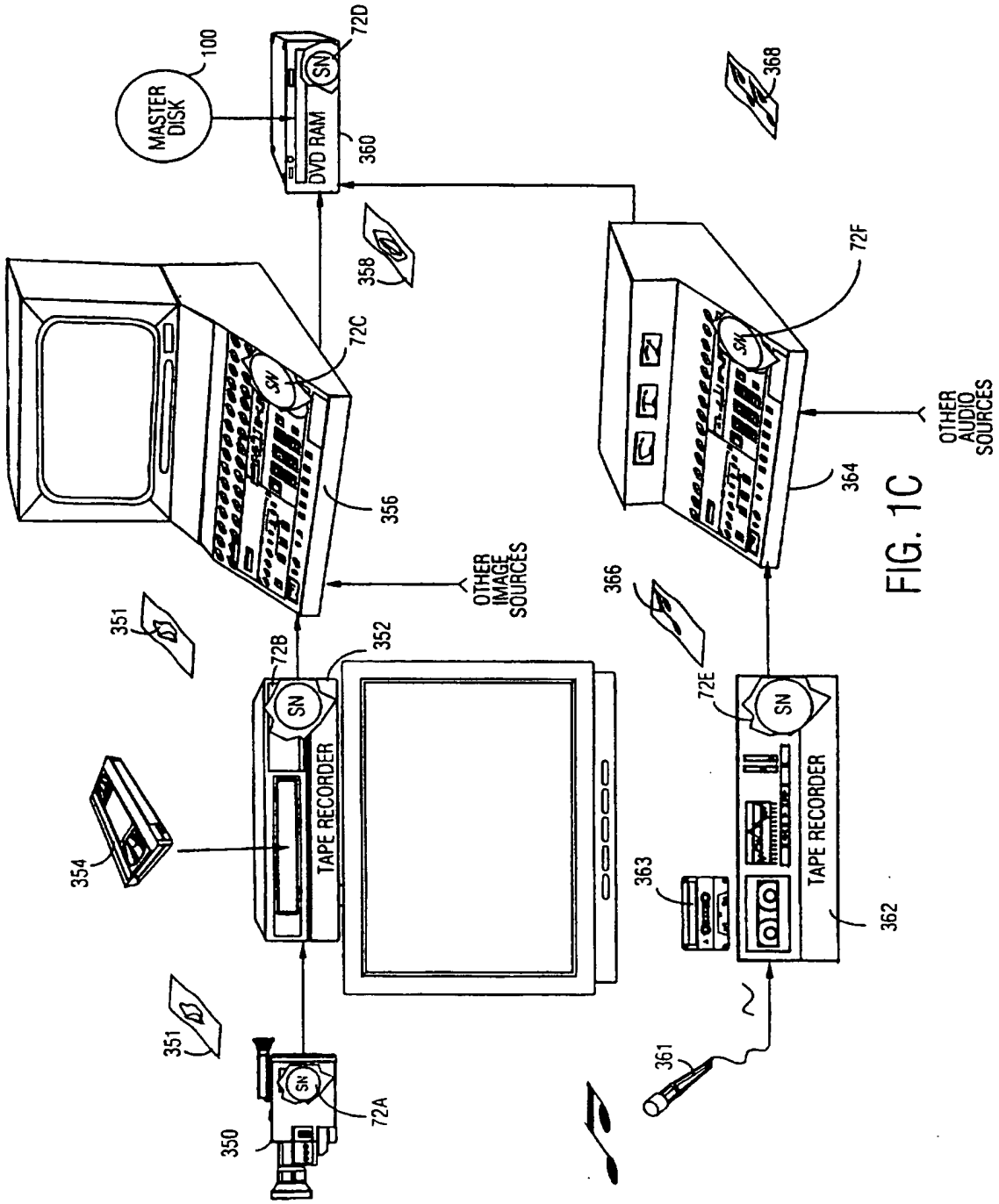


FIG. 1C

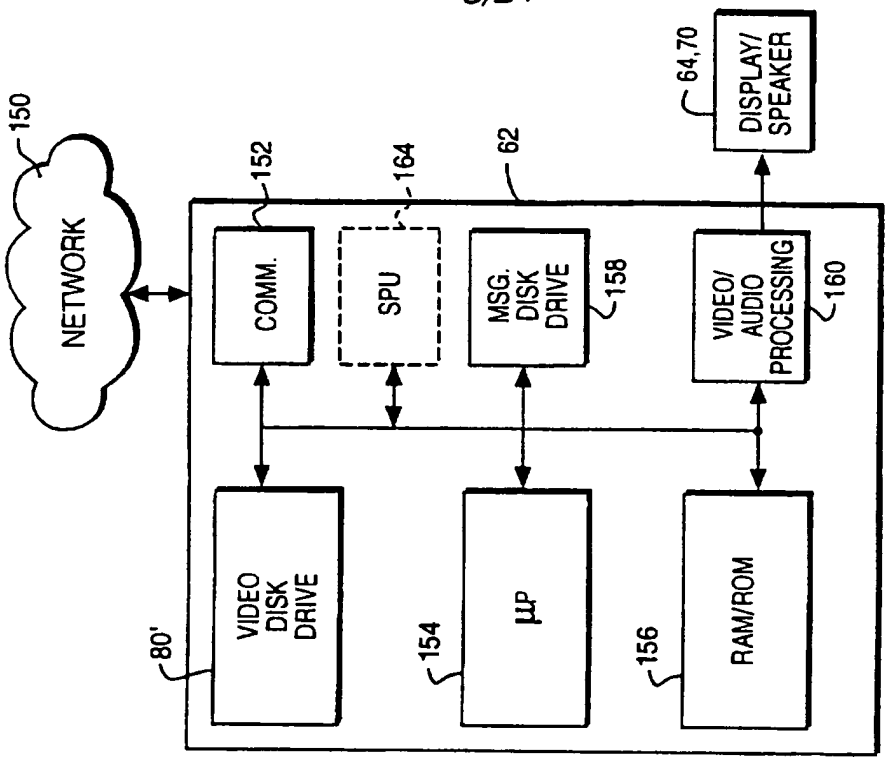


Fig.2B

EXAMPLE SECURE NODE ARCHITECTURE

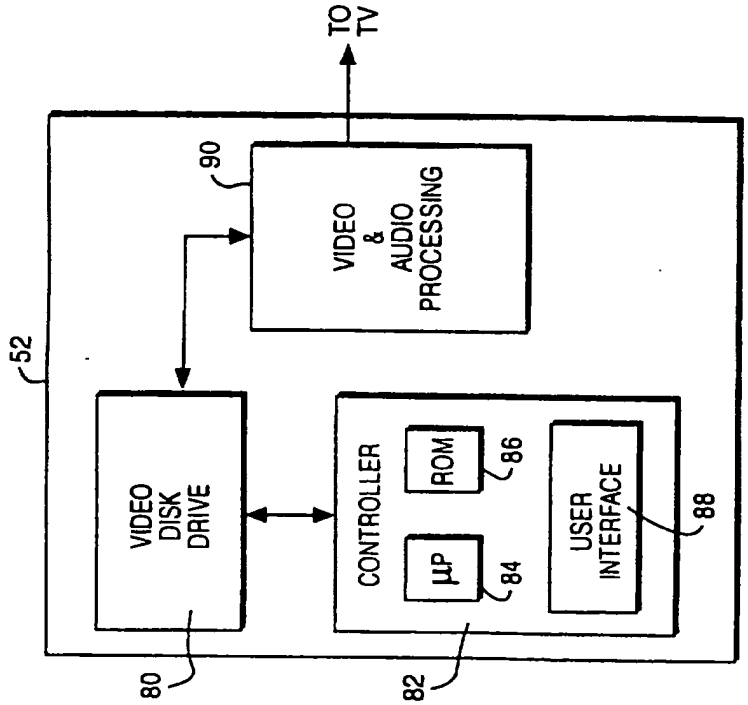


Fig.2A

EXAMPLE PLAYER ARCHITECTURE

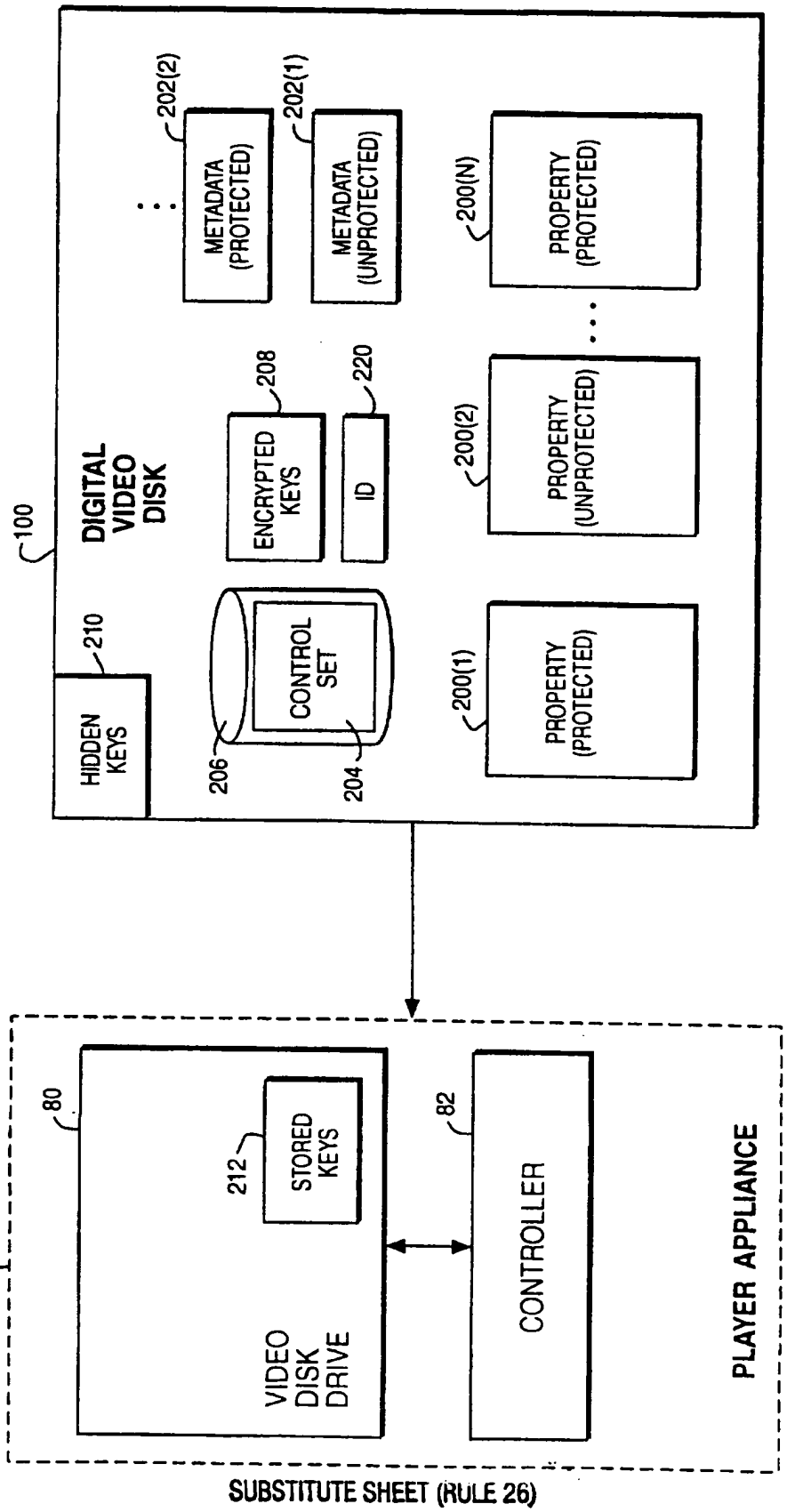


Fig.3

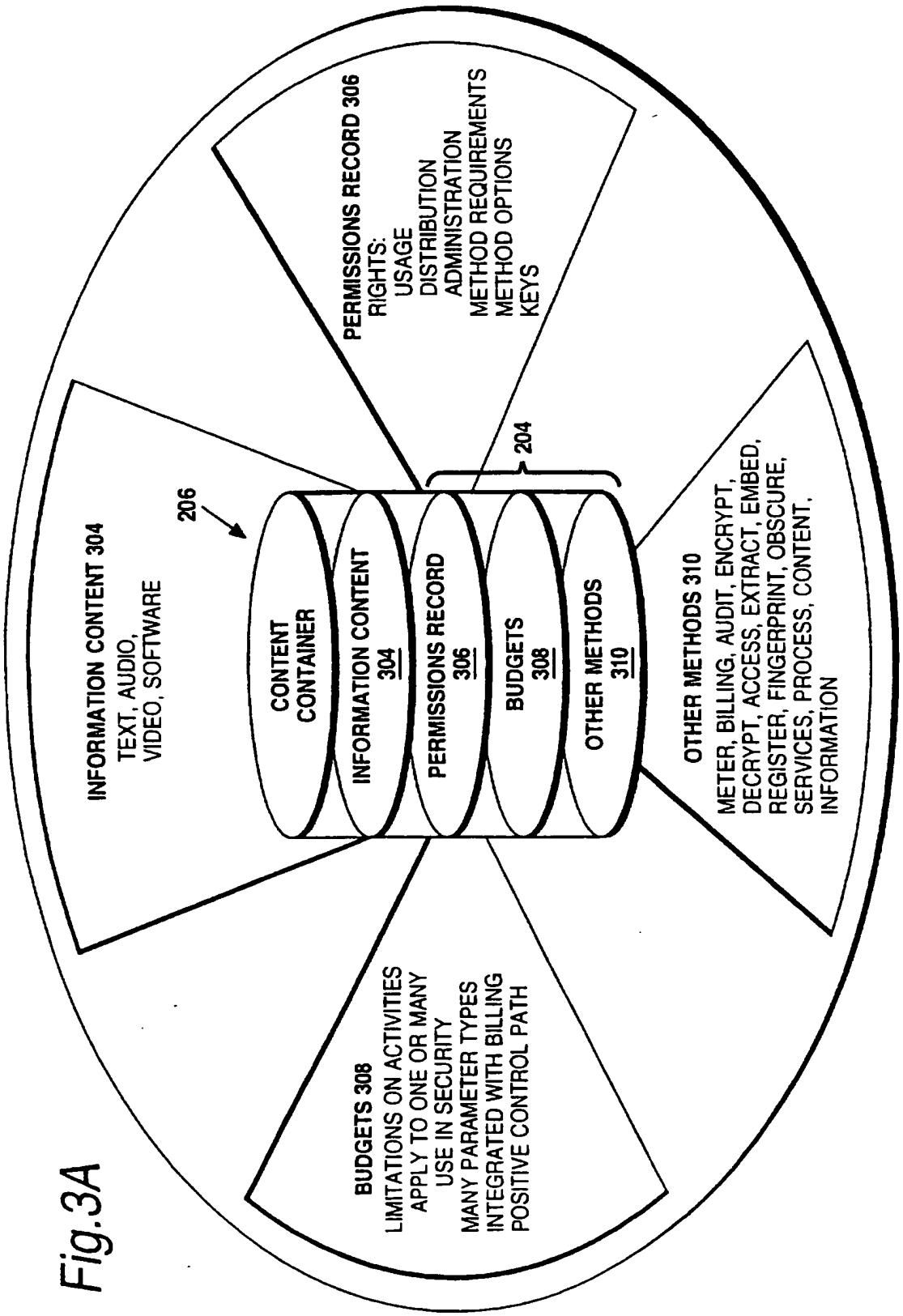


Fig.3A

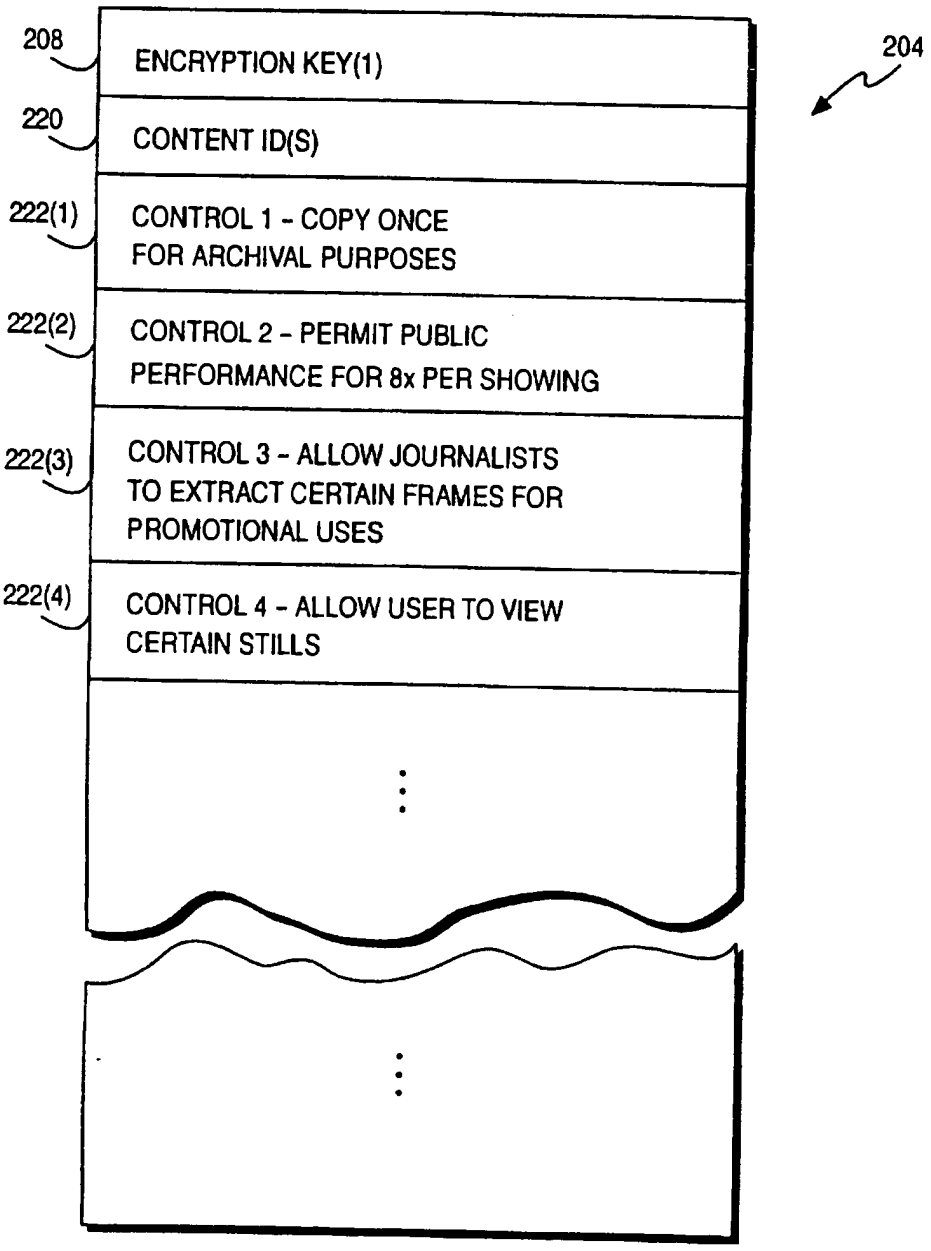


Fig.3B

EXAMPLE CONTROL SET

Fig.4A

EXAMPLE PLAYER STEPS

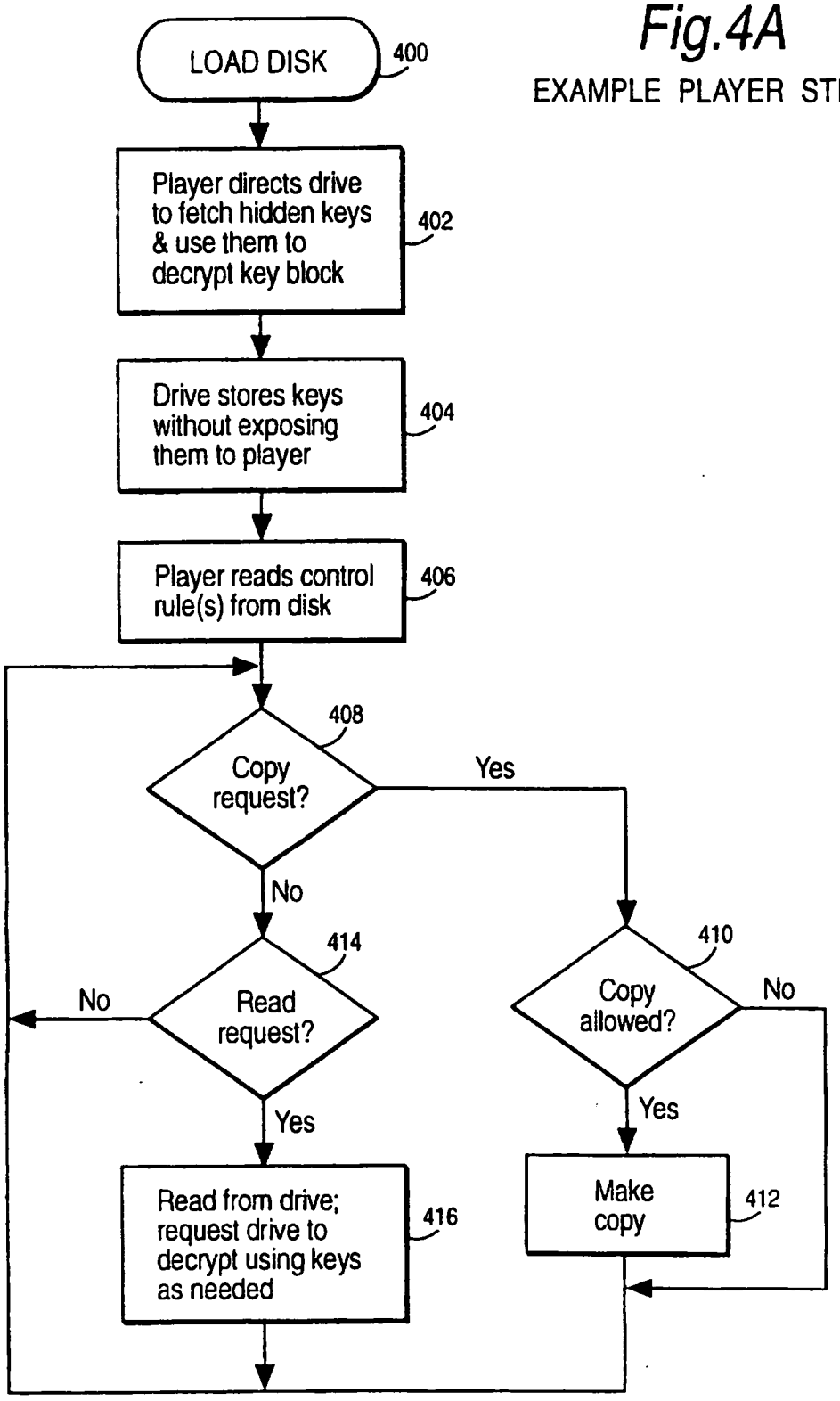
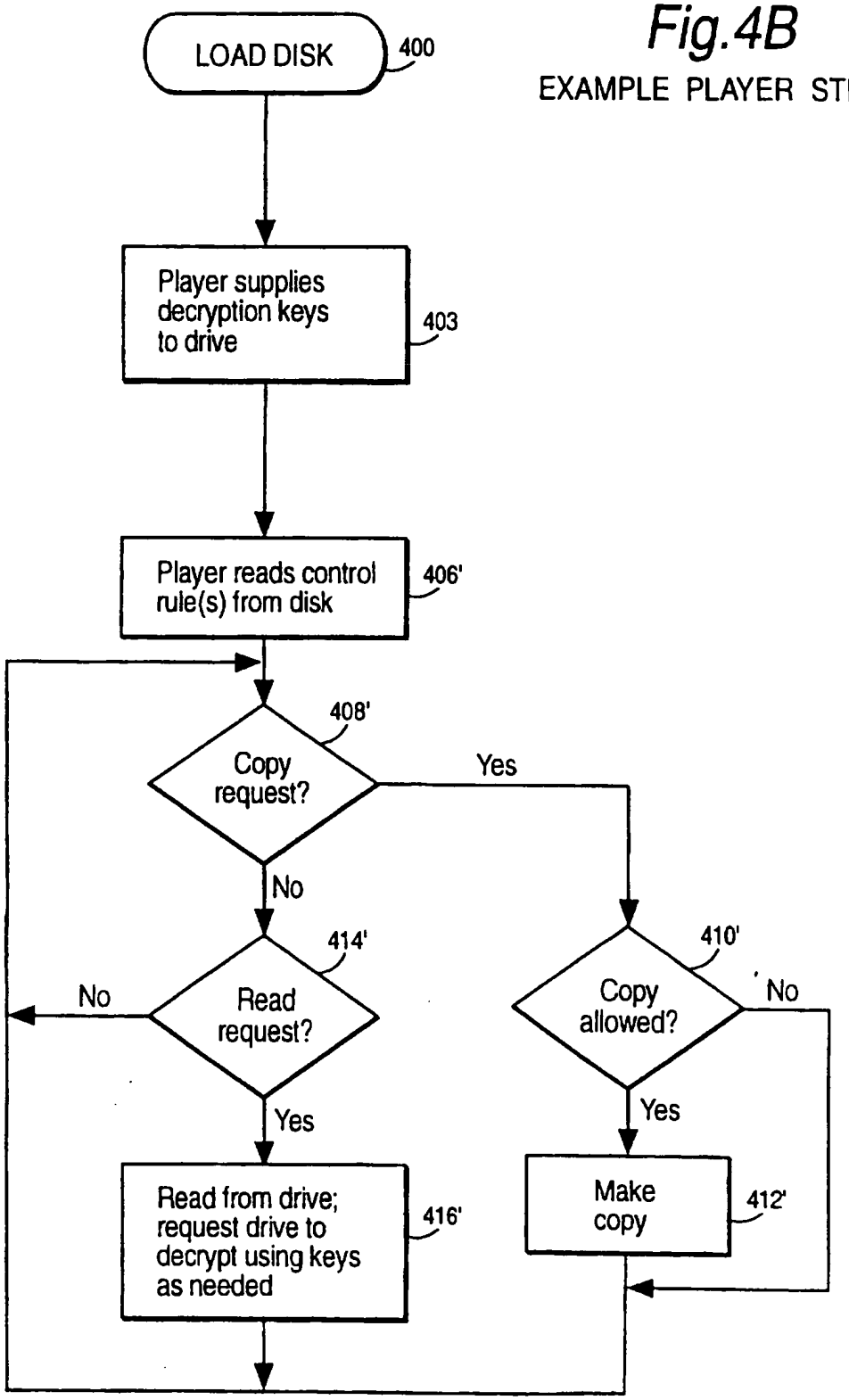


Fig.4B
EXAMPLE PLAYER STEPS



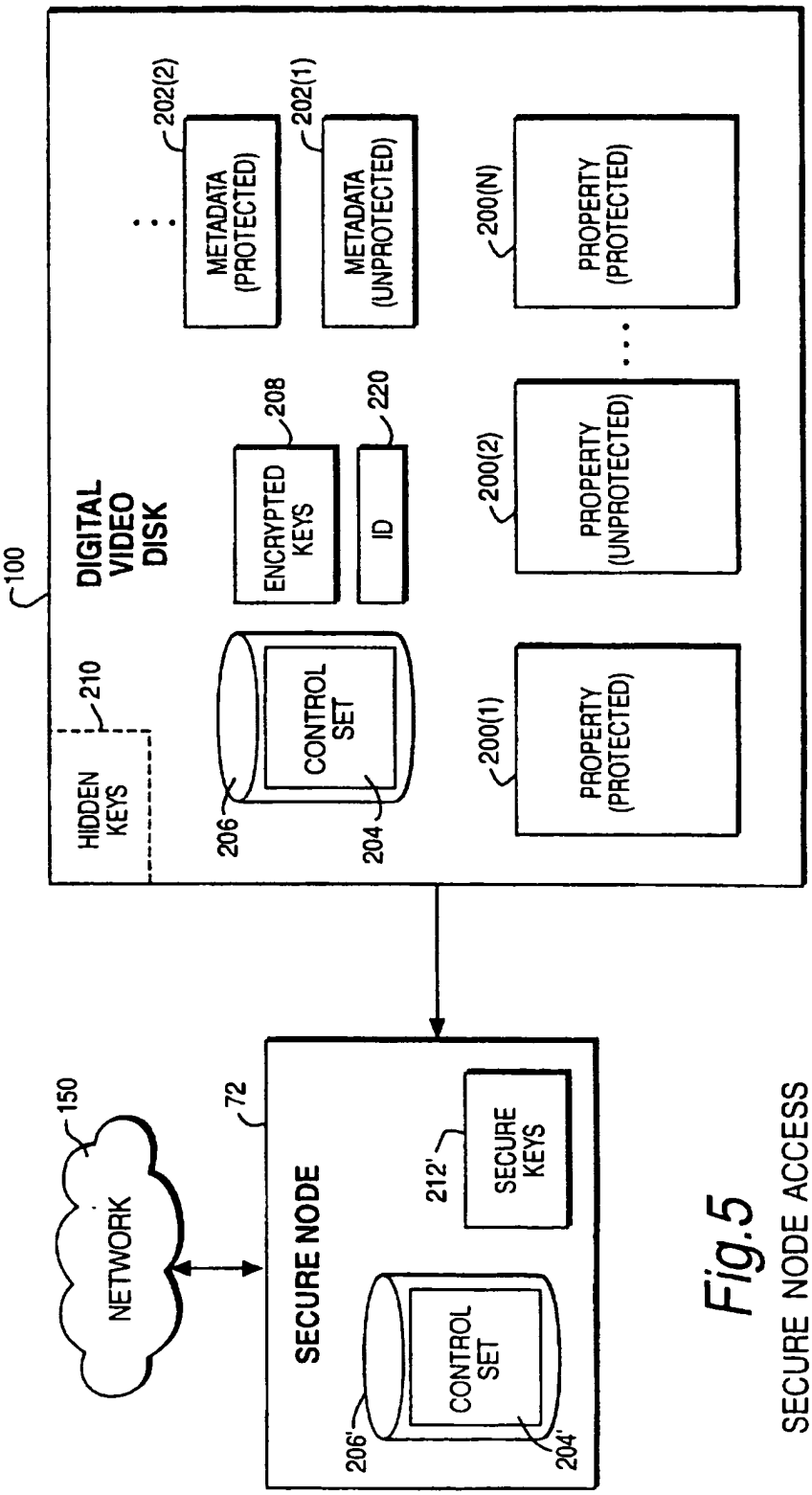


Fig.5
SECURE NODE ACCESS

Fig.6

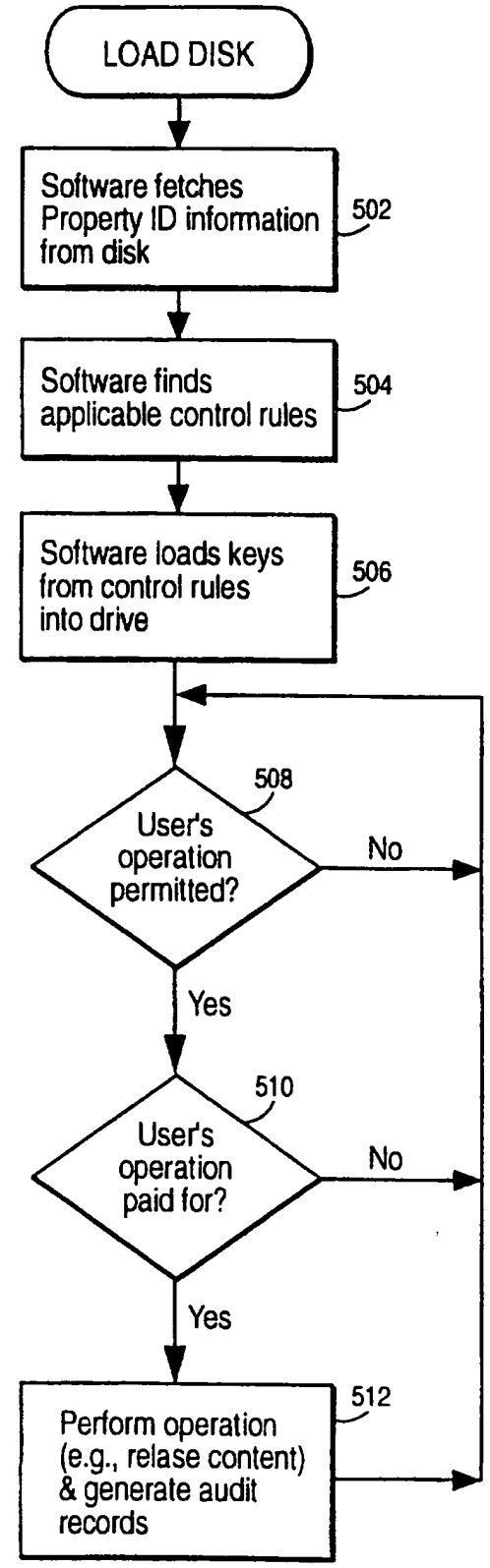


FIG. 7

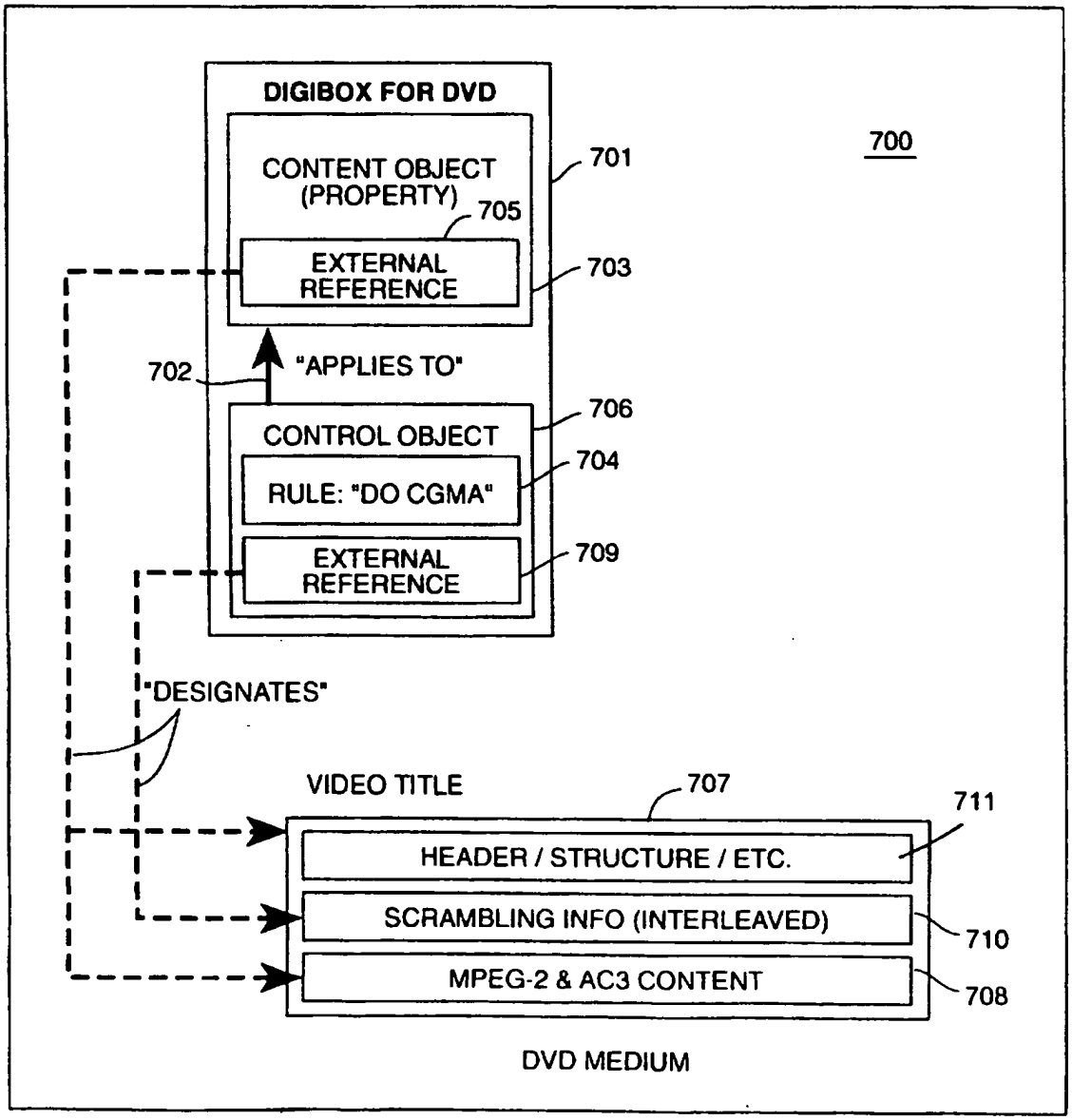


FIG. 8

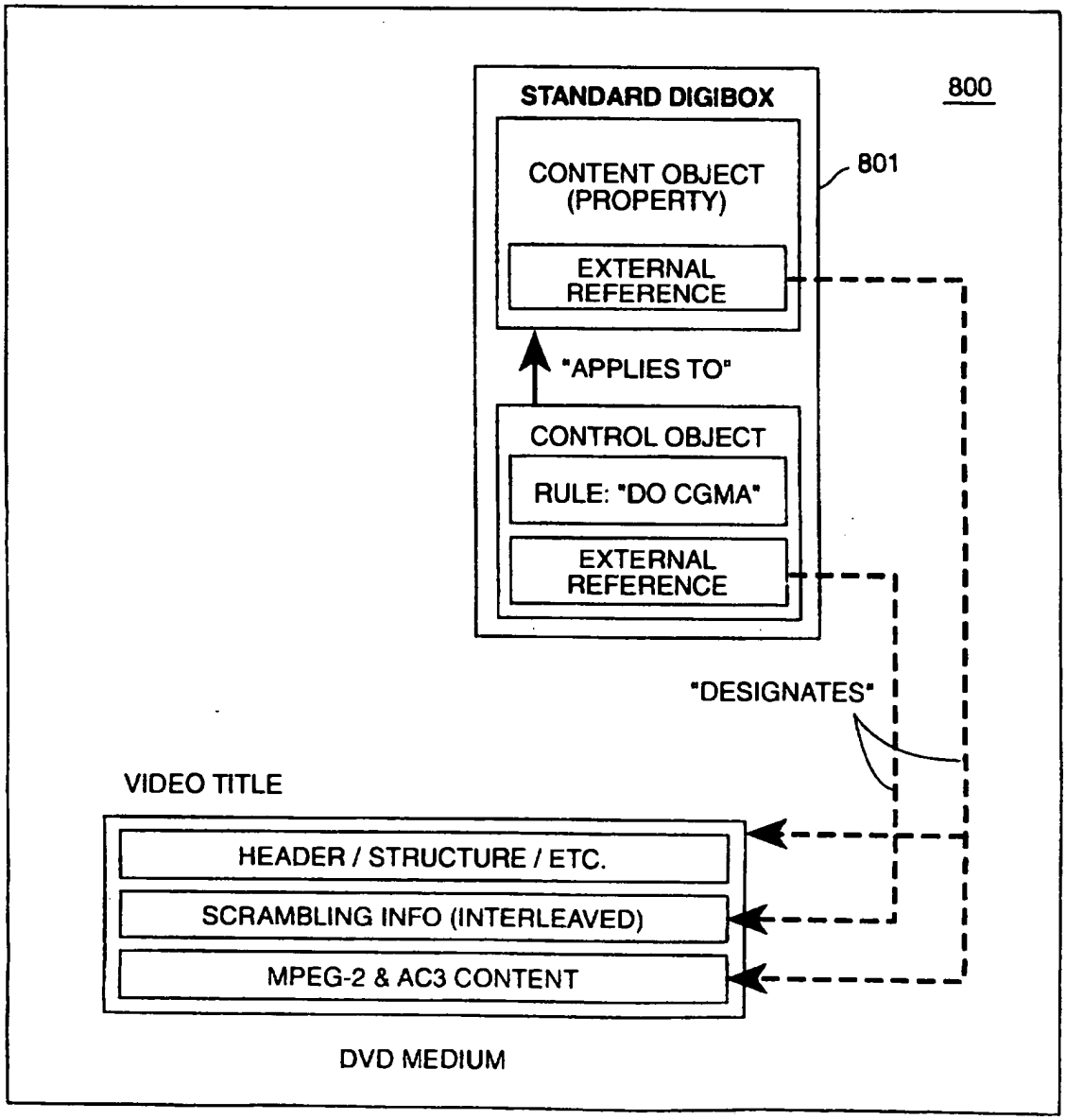


FIG. 9

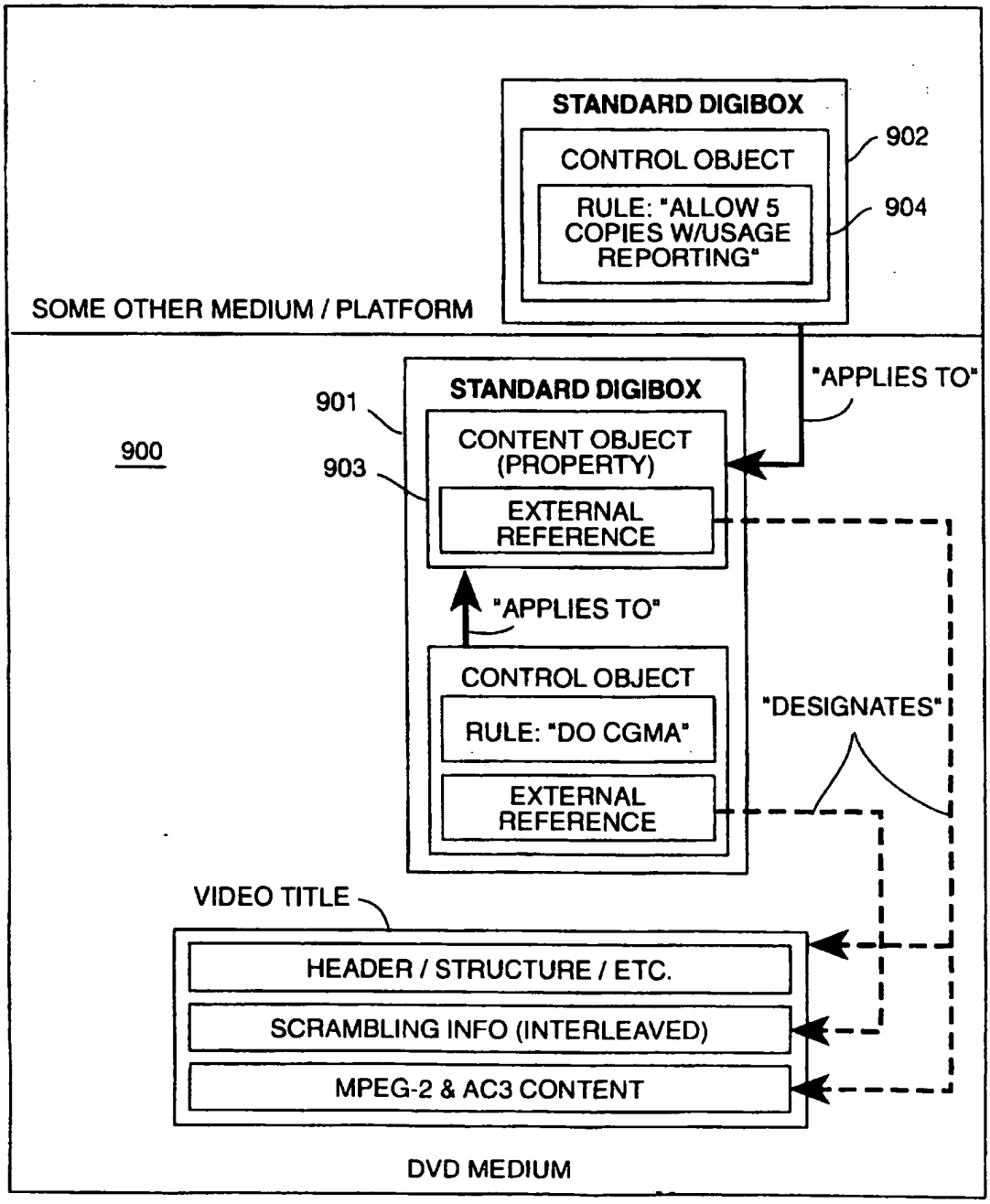


FIG. 10

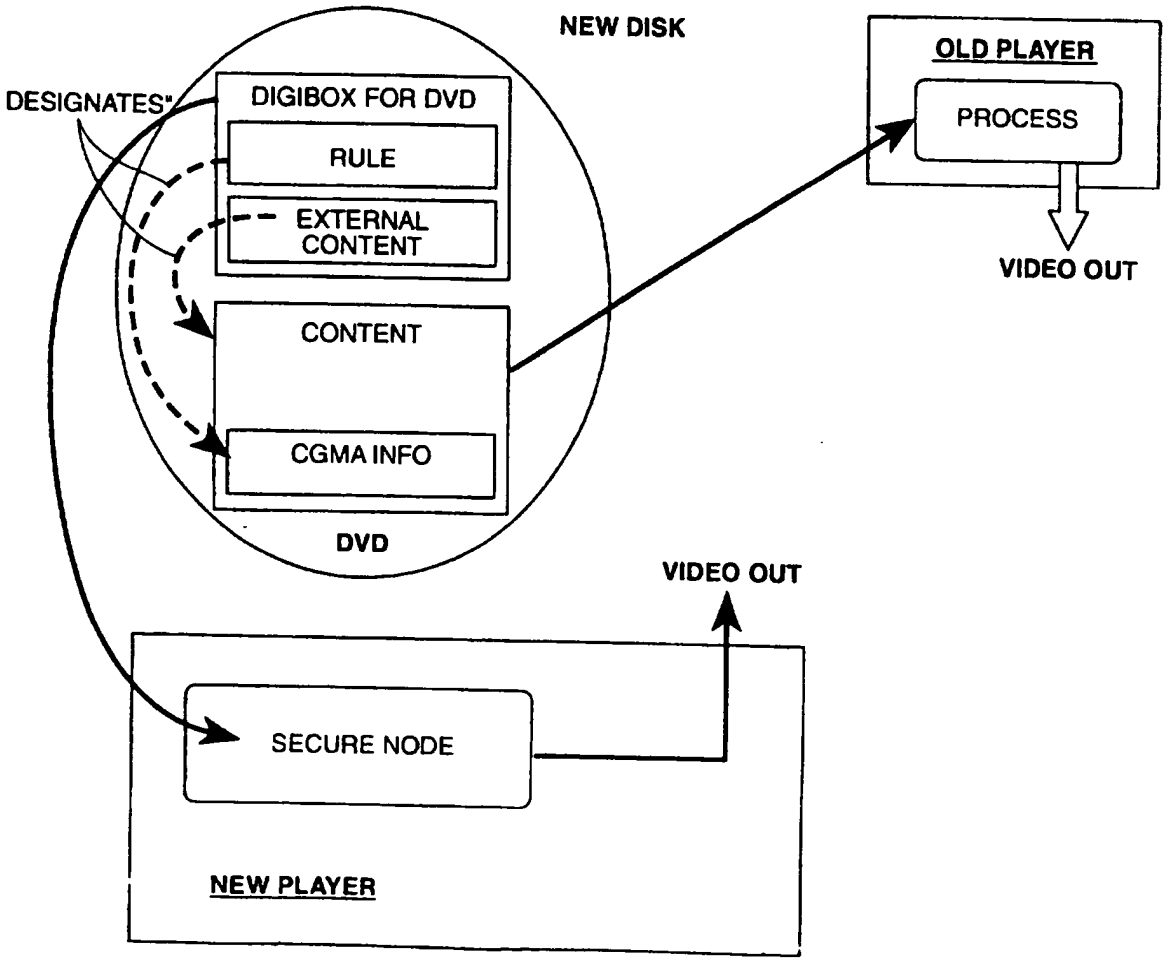
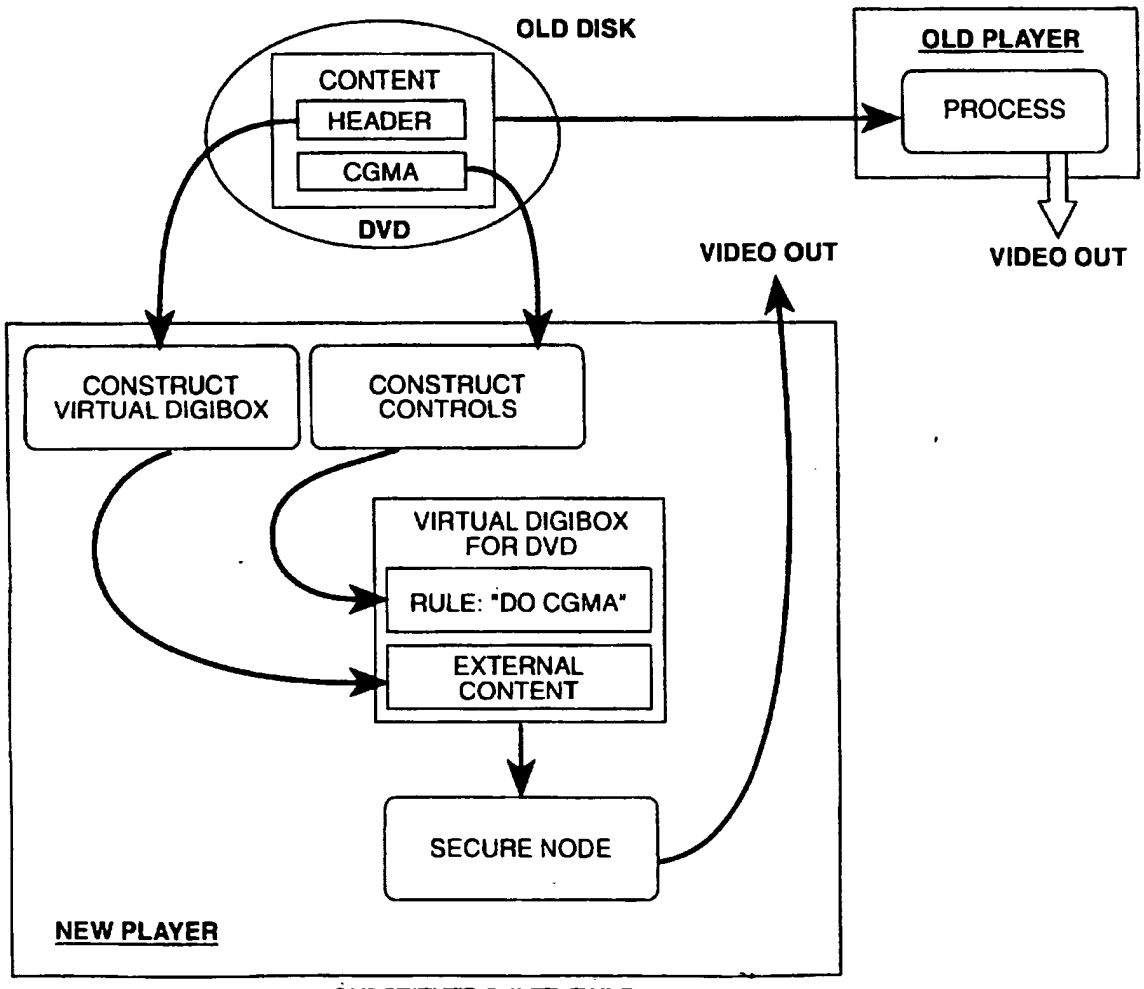


FIG. 11



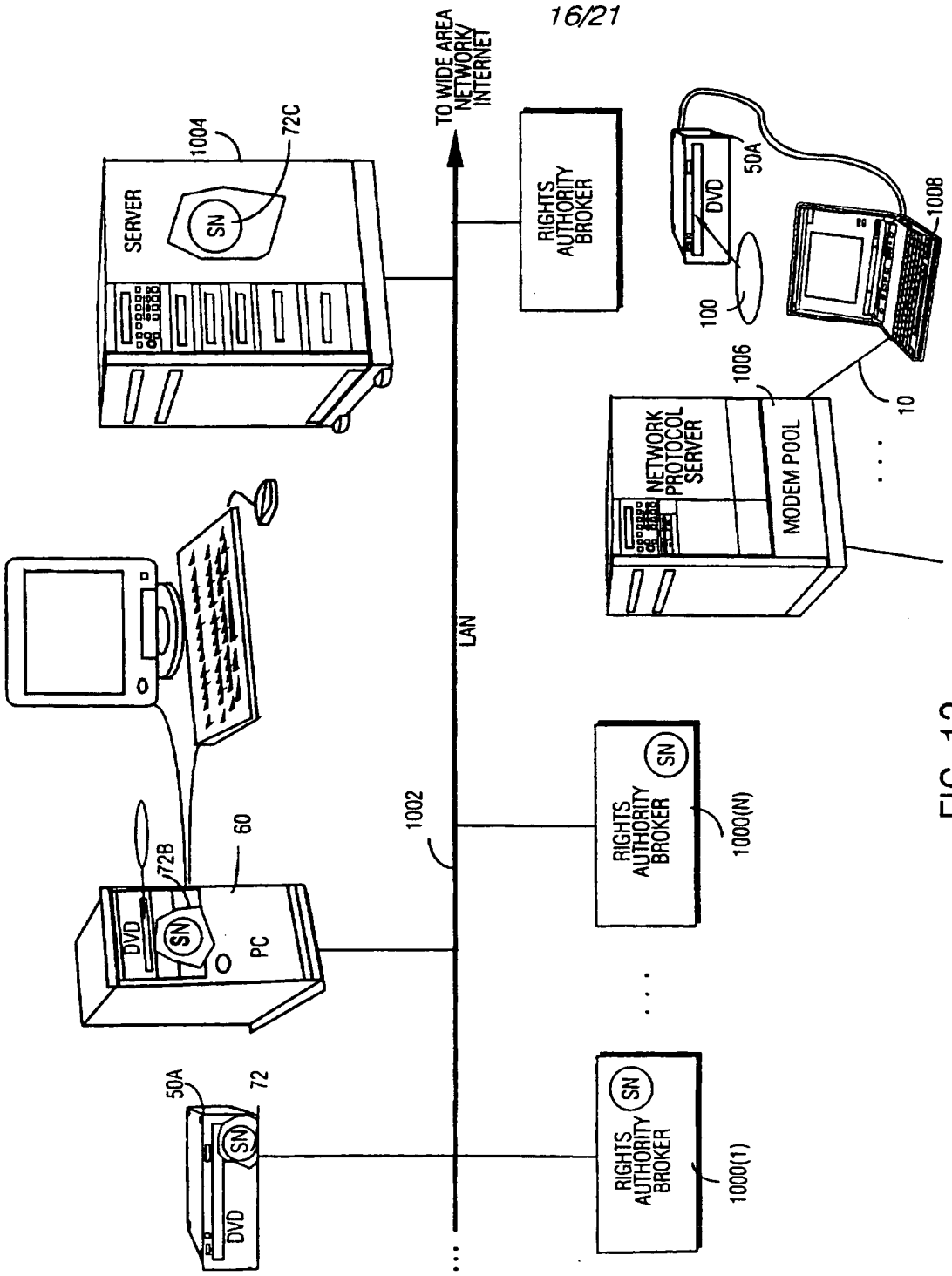


FIG. 12

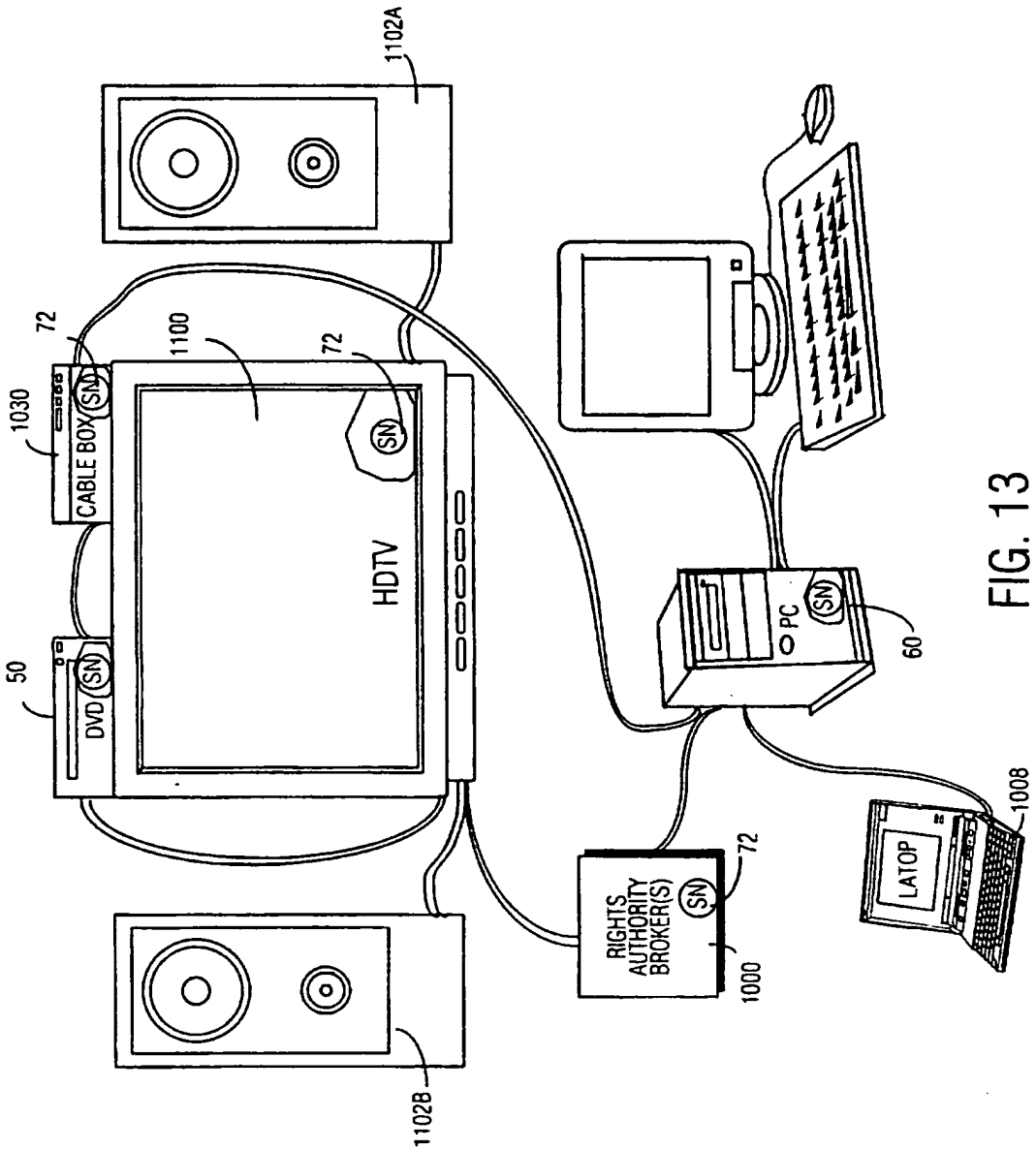


FIG. 13

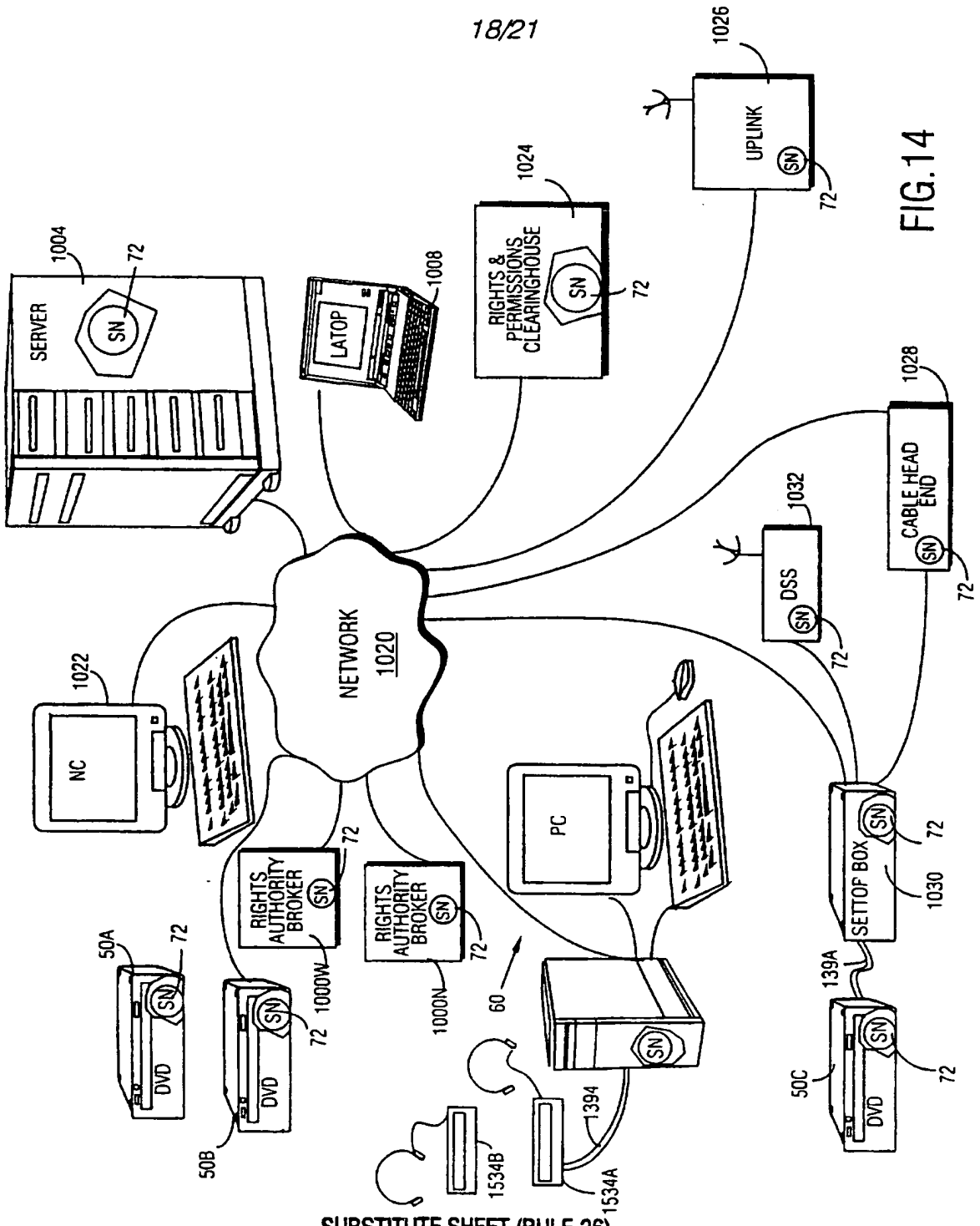


FIG.14

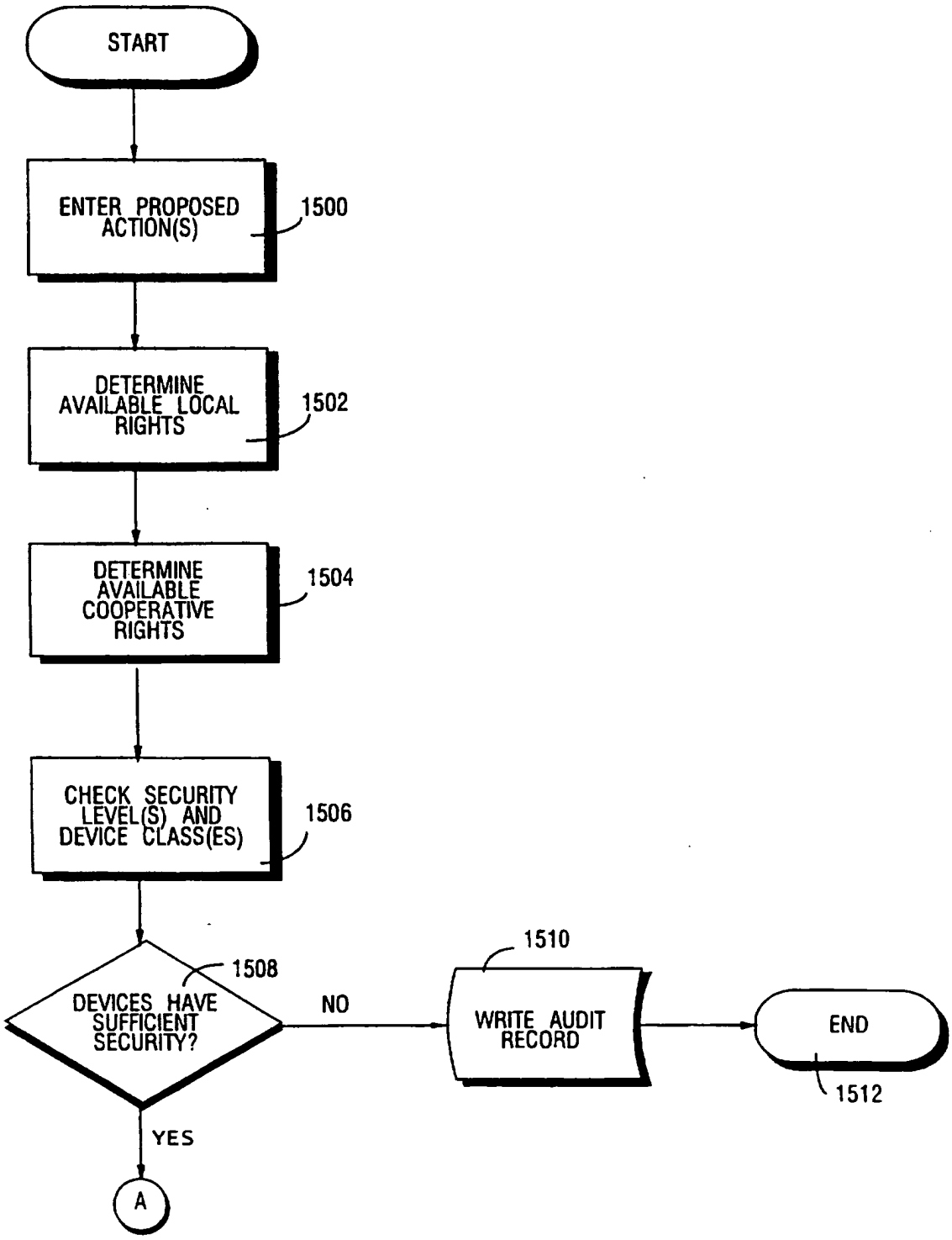


FIG.15A

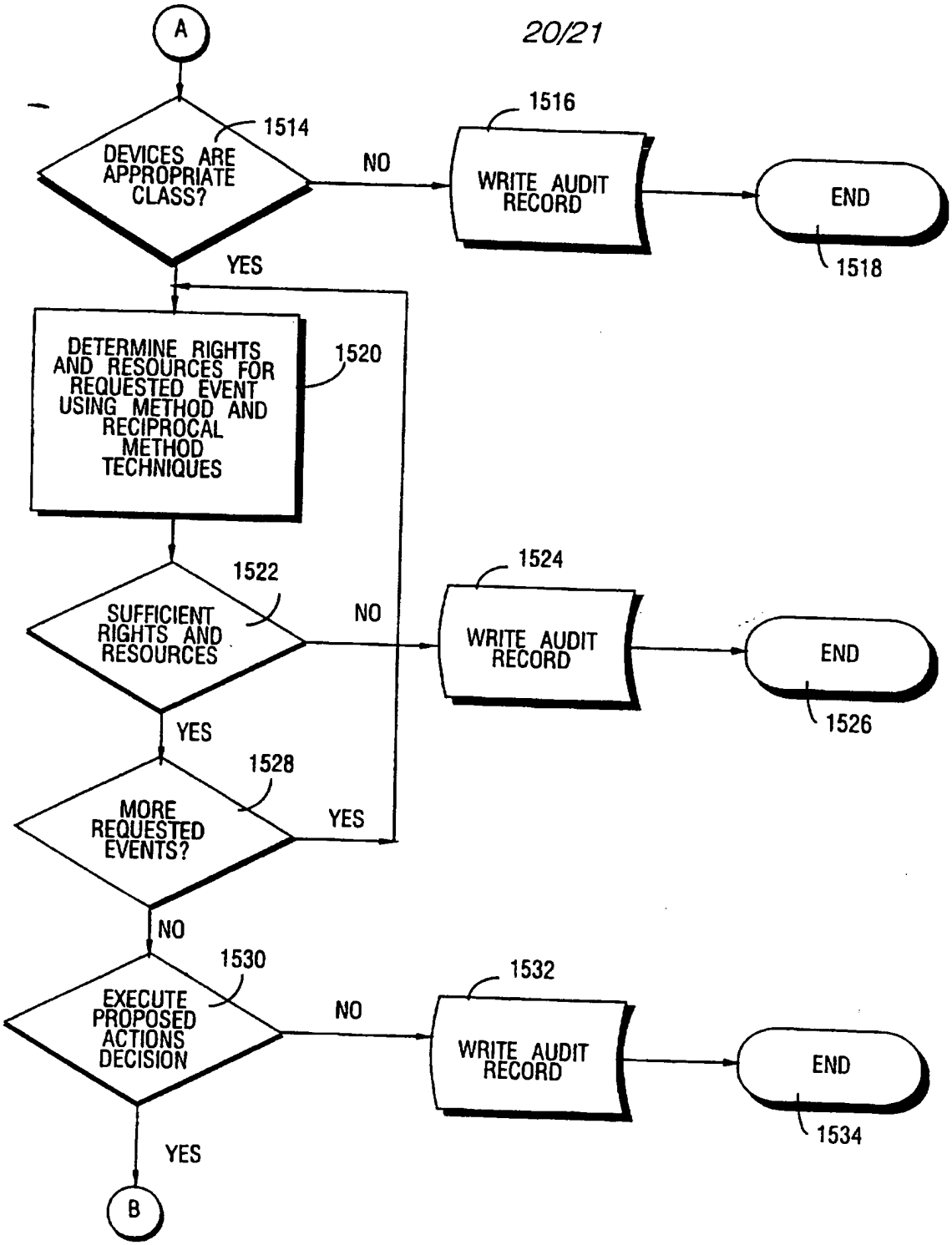
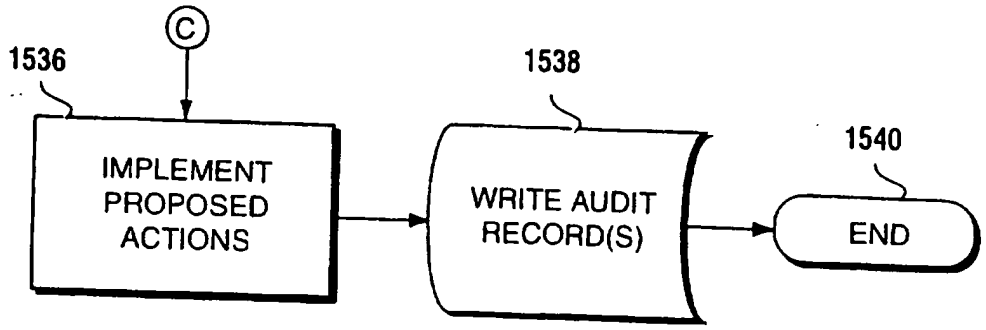


FIG.15B

FIG. 15C



PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 1/00 — A1

(11) International Publication Number: WO 98/09209

(43) International Publication Date: 5 March 1998 (05.03.98)

(21) International Application Number: PCT/US97/15243

(22) International Filing Date: 29 August 1997 (29.08.97)

(30) Priority Data:
08/706,206 30 August 1996 (30.08.96) US

(71) Applicant: INTERTRUST TECHNOLOGIES CORP. [US/US]; 460 Oakmead Parkway, Sunnyvale, CA 94086 (US).

(72) Inventors: GINTER, Karl, L.; 10404 43rd Avenue, Beltsville, MD 20705 (US). SHEAR, Victor, H.; 5203 Battery Lane, Bethesda, MD 20814 (US). SIBERT, W., Olin; 30 Ingleside Road, Lexington, MA 02173-2522 (US). SPAHN, Francis, J.; 2410 Edwards Avenue, El Cerrito, CA 94530 (US). VAN WIE, David, M.; 1250 Lakeside Drive, Sunnyvale, CA 94086 (US).

(74) Agent: FARIS, Robert, W.; Nixon & Vanderhye P.C., 8th floor, 1100 North Glebe Road, Arlington, VA 22201-4714 (US).

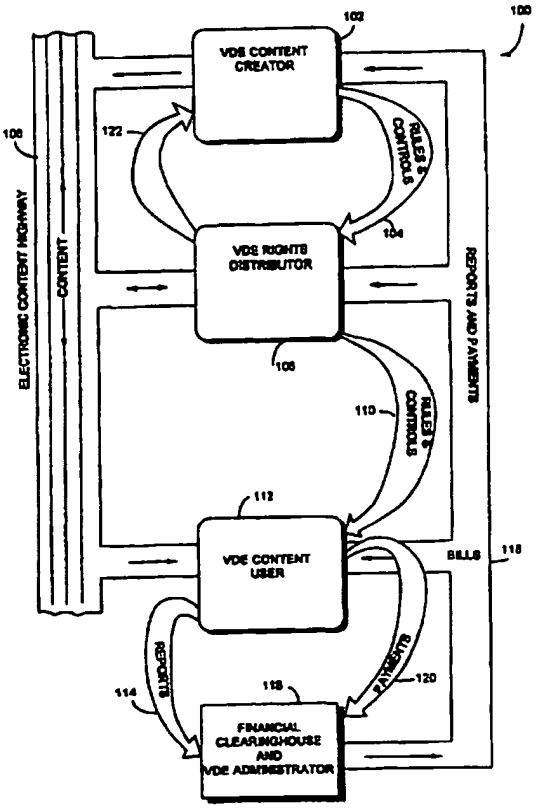
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

(57) Abstract

The present invention provides systems and methods for electronic commerce including secure transaction management and electronic rights protection. Electronic appliances such as computers employed in accordance with the present invention help to ensure that information is accessed and used only in authorized ways, and maintain the integrity, availability, and/or confidentiality of the information. Secure subsystems used with such electronic appliances provide a distributed virtual distribution environment (VDE) that may enforce a secure chain of handling and control, for example, to control and/or meter or otherwise monitor use of electronically stored or disseminated information. Such a virtual distribution environment may be used to protect rights of various participants in electronic commerce and other electronic or electronic-facilitated transactions. Secure distributed and other operating system environments and architectures, employing, for example, secure semiconductor processing arrangements that may establish secure, protected environments at each node. These techniques may be used to support an end-to-end electronic information distribution capability that may be used, for example, utilizing the "electronic highway".



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

Field(s) of the Invention(s)

This invention generally relates to computer and/or electronic security.

More particularly, this invention relates to systems and techniques for secure transaction management. This invention also relates to computer-based and other electronic appliance-based technologies that help to ensure that information is accessed and/or otherwise used only in authorized ways, and maintains the integrity, availability, and/or confidentiality of such information and processes related to such use.

The invention also relates to systems and methods for protecting rights of various participants in electronic commerce and other electronic or electronically-facilitated transactions.

The invention also relates to secure chains of handling and control for both information content and information employed to regulate the use of such content and consequences of such use. It also relates to systems and techniques that manage, including meter and/or limit and/or otherwise monitor use of electronically stored and/or disseminated information. The invention

particularly relates to transactions, conduct and arrangements that make use of, including consequences of use of, such systems and/or techniques.

The invention also relates to distributed and other operating systems, environments and architectures. It also generally relates to secure architectures, including, for example, tamper-resistant hardware-based processors, that can be used to establish security at each node of a distributed system.

Background and Summary of the Invention(s)

Telecommunications, financial transactions, government processes, business operations, entertainment, and personal business productivity all now depend on electronic appliances. Millions of these electronic appliances have been electronically connected together. These interconnected electronic appliances comprise what is increasingly called the "information highway." Many businesses, academicians, and government leaders are concerned about how to protect the rights of citizens and organizations who use this information (also "electronic" or "digital") highway.

Electronic Content

Today, virtually anything that can be represented by words, numbers, graphics, or system of commands and instructions can be formatted into electronic digital information. Television, cable, satellite transmissions, and on-line services transmitted over telephone lines, compete to distribute digital information and entertainment to homes and businesses. The owners and marketers of this content include software developers, motion picture and recording companies, publishers of books, magazines, and newspapers, and information database providers. The popularization of on-line services has also enabled the individual personal computer user to participate as a content provider. It is estimated that the worldwide market for electronic information in 1992 was approximately $40 billion and is expected to grow to $200 billion by 1997, according to Microsoft Corporation. The present invention can materially enhance the revenue of content providers, lower the distribution costs and the costs for content, better support advertising and usage information gathering, and better satisfy the needs of electronic information users. These improvements can lead to a significant increase in the amount and variety of electronic information and the methods by which such information is distributed.

The inability of conventional products to be shaped to the needs of electronic information providers and users is sharply in contrast to the present invention. Despite the attention devoted by a cross-section of America's largest telecommunications, computer, entertainment and information provider companies to some of the problems addressed by the present invention, only the present invention provides commercially secure, effective solutions for configurable, general purpose electronic commerce transaction/distribution control systems.

Controlling Electronic Content

The present invention provides a new kind of "virtual distribution environment" (called "VDE" in this document) that secures, administers, and audits electronic information use. VDE also features fundamentally important capabilities for managing content that travels "across" the "information highway." These capabilities comprise a rights protection solution that serves all electronic community members. These members include content creators and distributors, financial service providers, end-users, and others. VDE is the first general purpose, configurable, transaction control/rights protection solution for users of computers, other electronic appliances, networks, and the information highway.

A fundamental problem for electronic content providers is extending their ability to control the use of proprietary information. Content providers often need to limit use to authorized activities and amounts. Participants in a business model involving, for example, provision of movies and advertising on optical discs may include actors, directors, script and other writers, musicians, studios, publishers, distributors, retailers, advertisers, credit card services, and content end-users. These participants need the ability to embody their range of agreements and requirements, including use limitations, into an "extended" agreement comprising an overall electronic business model. This extended agreement is represented by electronic content control information that can automatically enforce agreed upon rights and obligations. Under VDE, such an extended agreement may comprise an electronic contract involving all business model participants. Such an agreement may alternatively, or in addition, be made up of electronic agreements between subsets of the business model participants. Through the use of VDE, electronic commerce can function in the same way as traditional commerce—that is commercial relationships regarding products and services can be shaped through the negotiation of one or more agreements between a variety of parties.

Commercial content providers are concerned with ensuring proper compensation for the use of their electronic information. Electronic digital information, for example a CD recording, can today be copied relatively easily and inexpensively. Similarly, unauthorized copying and use of software programs deprives rightful owners of billions of dollars in annual revenue according to the International Intellectual Property Alliance. Content providers and distributors have devised a number of limited function rights protection mechanisms to protect their rights. Authorization passwords and protocols, license servers, "lock/unlock" distribution methods, and non-electronic contractual limitations imposed on users of shrink-wrapped software are a few of the more prevalent content protection schemes. In a commercial context, these efforts are inefficient and limited solutions.

Providers of "electronic currency" have also created protections for their type of content. These systems are not sufficiently adaptable, efficient, nor flexible enough to support the generalized use of electronic currency. Furthermore, they do not provide sophisticated auditing and control configuration capabilities. This means that current electronic currency tools lack the sophistication needed for many real-world financial business models. VDE provides means for anonymous currency

and for "conditionally" anonymous currency, wherein currency related activities remain anonymous except under special circumstances.

VDE Control Capabilities

VDE allows the owners and distributors of electronic digital information to reliably bill for, and securely control, audit, and budget the use of, electronic information. It can reliably detect and monitor the use of commercial information products. VDE uses a wide variety of different electronic information delivery means: including, for example, digital networks, digital broadcast, and physical storage media such as optical and magnetic disks. VDE can be used by major network providers, hardware manufacturers, owners of electronic information, providers of such information, and clearinghouses that gather usage information regarding, and bill for the use of, electronic information.

VDE provides comprehensive and configurable transaction management, metering and monitoring technology. It can change how electronic information products are protected, marketed, packaged, and distributed. When used, VDE should result in higher revenues for information providers and greater

user satisfaction and value. Use of VDE will normally result in lower usage costs, decreased transaction costs, more efficient access to electronic information, re-usability of rights protection and other transaction management implementations, greatly improved flexibility in the use of secured information, and greater standardization of tools and processes for electronic transaction management. VDE can be used to create an adaptable environment that fulfills the needs of electronic information owners, distributors, and users; financial clearinghouses; and usage information analyzers and resellers.

Rights and Control Information

In general, the present invention can be used to protect the rights of parties who have:

(a) proprietary or confidentiality interests in electronic information. It can, for example, help ensure that information is used only in authorized ways;

(b) financial interests resulting from the use of electronically distributed information. It can help ensure that content providers will be paid for use of distributed information; and

(c) interests in electronic credit and electronic currency storage, communication, and/or use including electronic cash, banking, and purchasing.

Protecting the rights of electronic community members involves a broad range of technologies. VDE combines these technologies in a way that creates a "distributed" electronic rights protection "environment." This environment secures and protects transactions and other processes important for rights protection. VDE, for example, provides the ability to prevent, or impede, interference with and/or observation of, important rights related transactions and processes. VDE, in its preferred embodiment, uses special purpose tamper resistant Secure Processing Units (SPUs) to help provide a high level of security for VDE processes and information storage and communication.

The rights protection problems solved by the present invention are electronic versions of basic societal issues. These issues include protecting property rights, protecting privacy rights, properly compensating people and organizations for their work and risk, protecting money and credit, and generally protecting the security of information. VDE employs a system that uses a common set of processes to manage rights issues in an efficient, trusted, and cost-effective way.

VDE can be used to protect the rights of parties who create electronic content such as, for example: records, games, movies, newspapers, electronic books and reference materials, personal electronic mail, and confidential records and communications. The invention can also be used to protect the rights of parties who provide electronic products, such as publishers and distributors; the rights of parties who provide electronic credit and currency to pay for use of products, for example, credit clearinghouses and banks; the rights to privacy of parties who use electronic content (such as consumers, business people, governments); and the privacy rights of parties *described* by electronic information, such as privacy rights related to information contained in a medical record, tax record, or personnel record.

In general, the present invention can protect the rights of parties who have:

(a) commercial interests in electronically distributed information -- the present invention can help ensure, for example, that parties, will be paid for use of distributed information in a manner consistent with their agreement;

(b) proprietary and/or confidentiality interests in electronic information -- the present invention can, for example, help ensure that data is used only in authorized ways;

(c) interests in electronic credit and electronic currency storage, communication, and/or use -- this can include electronic cash, banking, and purchasing; and

(d) interests in electronic information derived, at least in part, from use of other electronic information.

VDE Functional Properties

VDE is a cost-effective and efficient rights protection solution that provides a unified, consistent system for securing and managing transaction processing. VDE can:

(a) audit and analyze the use of content,

(b) ensure that content is used only in authorized ways, and

(c) allow information regarding content usage to be used only in ways approved by content users.

In addition, VDE:

(a) is very configurable, modifiable, and re-usable;

(b) supports a wide range of useful capabilities that may be combined in different ways to accommodate most potential applications;

(c) operates on a wide variety of electronic appliances ranging from hand-held inexpensive devices to large mainframe computers;

(d) is able to ensure the various rights of a number of different parties, and a number of different rights protection schemes, simultaneously;

(e) is able to preserve the rights of parties through a series of transactions that may occur at different times and different locations;

(f) is able to flexibly accommodate different ways of securely delivering information and reporting usage; and

(g) provides for electronic analogues to "real" money and credit, including anonymous electronic cash, to pay for products and services and to support personal (including home) banking and other financial activities.

VDE economically and efficiently fulfills the rights protection needs of electronic community members. Users of VDE will not require additional rights protection systems for different information highway products and rights problems—nor will they be required to install and learn a new system for each new information highway application.

VDE provides a unified solution that allows all content creators, providers, and users to employ the same electronic rights protection solution. Under authorized circumstances, the participants can freely exchange content and associated content control sets. This means that a user of VDE may, if allowed, use the same electronic system to work with different kinds of content having different sets of content control information. The

content and control information supplied by one group can be used by people who normally use content and control information supplied by a different group. VDE can allow content to be exchanged "universally" and users of an implementation of the present invention can interact electronically without fear of incompatibilities in content control, violation of rights, or the need to get, install, or learn a new content control system.

The VDE securely administers transactions that specify protection of rights. It can protect electronic rights including, for example:

(a) the property rights of authors of electronic content,

(b) the commercial rights of distributors of content,

(c) the rights of any parties who facilitated the distribution of content,

(d) the privacy rights of users of content,

(e) the privacy rights of parties portrayed by stored and/or distributed content, and

(f) any other rights regarding enforcement of electronic agreements.

VDE can enable a very broad variety of electronically enforced commercial and societal agreements. These agreements can include electronically implemented contracts, licenses, laws, regulations, and tax collection.

Contrast With Traditional Solutions

Traditional content control mechanisms often require users to purchase more electronic information than the user needs or desires. For example, infrequent users of shrink-wrapped software are required to purchase a program at the same price as frequent users, even though they may receive much less value from their less frequent use. Traditional systems do not scale cost according to the extent or character of usage and traditional systems can not attract potential customers who find that a fixed price is too high. Systems using traditional mechanisms are also not normally particularly secure. For example, shrink-wrapping does not prevent the constant illegal pirating of software once removed from either its physical or electronic package.

Traditional electronic information rights protection systems are often inflexible and inefficient and may cause a content provider to choose costly distribution channels that increase a product's price. In general these mechanisms restrict product pricing, configuration, and marketing flexibility. These compromises are the result of techniques for controlling information which cannot accommodate both different content models and content models which reflect the many, varied requirements, such as content delivery strategies, of the model participants. This can limit a provider's ability to deliver sufficient overall value to justify a given product's cost in the eyes of many potential users. VDE allows content providers and distributors to create applications and distribution networks that reflect content providers' and users' preferred business models. It offers users a uniquely cost effective and feature rich system that supports the ways providers *want* to distribute information and the ways users *want* to use such information. VDE supports content control models that ensure rights and allow content delivery strategies to be shaped for maximum commercial results.

Chain of Handling and Control

VDE can protect a collection of rights belonging to various parties having in rights in, or to, electronic information. This

information may be at one location or dispersed across (and/or moving between) multiple locations. The information may pass through a "chain" of distributors and a "chain" of users. Usage information may also be reported through one or more "chains" of parties. In general, VDE enables parties that (a) have rights in electronic information, and/or (b) act as direct or indirect agents for parties who have rights in electronic information, to ensure that the moving, accessing, modifying, or otherwise using of information can be securely controlled by rules regarding how, when, where, and by whom such activities can be performed.

VDE Applications and Software

VDE is a secure system for regulating electronic conduct and commerce. Regulation is ensured by control information put in place by one or more parties. These parties may include content providers, electronic hardware manufacturers, financial service providers, or electronic "infrastructure" companies such as cable or telecommunications companies. The control information implements "Rights Applications." Rights applications "run on" the "base software" of the preferred embodiment. This base software serves as a secure, flexible, general purpose foundation that can accommodate many different rights applications, that is, many different business models and their respective participant requirements.

A rights application under VDE is made up of special purpose pieces, each of which can correspond to one or more basic electronic processes needed for a rights protection environment. These processes can be combined together like building blocks to create electronic agreements that can protect the rights, and may enforce fulfillment of the obligations, of electronic information users and providers. One or more providers of electronic information can easily combine selected building blocks to create a rights application that is unique to a specific content distribution model. A group of these pieces can represent the capabilities needed to fulfill the agreement(s) between users and providers. These pieces accommodate many requirements of electronic commerce including:

- the distribution of permissions to use electronic information;

- the persistence of the control information and sets of control information managing these permissions;

- configurable control set information that can be selected by users for use with such information;

- data security and usage auditing of electronic information; and

- a secure system for currency, compensation and debit management.

For electronic commerce, a rights application, under the preferred embodiment of the present invention, can provide electronic enforcement of the business agreements between all participants. Since different groups of components can be put together for different applications, the present invention can provide electronic control information for a wide variety of different products and markets. This means the present invention can provide a "unified," efficient, secure, and cost-effective system for electronic commerce and data security. This allows VDE to serve as a single standard for electronic rights protection, data security, and electronic currency and banking.

In a VDE, the separation between a rights application and its foundation permits the efficient selection of sets of control information that are appropriate for each of many different types of applications and uses. These control sets can reflect both rights of electronic community members, as well as obligations

(such as providing a history of one's use of a product or paying taxes on one's electronic purchases). VDE flexibility allows its users to electronically implement and enforce common social and commercial ethics and practices. By providing a unified control system, the present invention supports a vast range of possible transaction related interests and concerns of individuals, communities, businesses, and governments. Due to its open design, VDE allows (normally under securely controlled circumstances) applications using technology independently created by users to be "added" to the system and used in conjunction with the foundation of the invention. In sum, VDE provides a system that can fairly reflect and enforce agreements among parties. It is a broad ranging and systematic solution that answers the pressing need for a secure, cost-effective, and fair electronic environment.

VDE Implementation

The preferred embodiment of the present invention includes various tools that enable system designers to directly insert VDE capabilities into their products. These tools include an Application Programmer's Interface ("API") and a Rights Permissioning and Management Language ("RPML"). The RPML provides comprehensive and detailed control over the use of the invention's features. VDE also includes certain user

interface subsystems for satisfying the needs of content providers, distributors, and users.

Information distributed using VDE may take many forms. It may, for example, be "distributed" for use on an individual's own computer, that is the present invention can be used to provide security for locally stored data. Alternatively, VDE may be used with information that is dispersed by authors and/or publishers to one or more recipients. This information may take many forms including: movies, audio recordings, games, electronic catalog shopping, multimedia, training materials, E-mail and personal documents, object oriented libraries, software programming resources, and reference/record keeping information resources (such as business, medical, legal, scientific, governmental, and consumer databases).

Electronic rights protection provided by the present invention will also provide an important foundation for trusted and efficient home and commercial banking, electronic credit processes, electronic purchasing, true or conditionally anonymous electronic cash, and EDI (Electronic Data Interchange). VDE provides important enhancements for improving data security in organizations by providing "smart"

transaction management features that can be far more effective than key and password based "go/no go" technology.

VDE normally employs an integration of cryptographic and other security technologies (e.g. encryption, digital signatures, etc.), with other technologies including: component, distributed, and event driven operating system technology, and related communications, object container, database, smart agent, smart card, and semiconductor design technologies.

I. Overview

A. VDE Solves Important Problems and Fills Critical Needs

The world is moving towards an integration of electronic information appliances. This interconnection of appliances provides a foundation for much greater electronic interaction and the evolution of electronic commerce. A variety of capabilities are required to implement an electronic commerce environment. VDE is the first system that provides many of these capabilities and therefore solves fundamental problems related to electronic dissemination of information.

Electronic Content

VDE allows electronic arrangements to be created involving two or more parties. These agreements can themselves comprise a collection of agreements between participants in a commercial value chain and/or a data security chain model for handling, auditing, reporting, and payment. It can provide efficient, reusable, modifiable, and consistent means for secure electronic content: distribution, usage control, usage payment, usage auditing, and usage reporting. Content may, for example, include:

- financial information such as electronic currency and credit;
- commercially distributed electronic information such as reference databases, movies, games, and advertising; and
- electronic properties produced by persons and organizations, such as documents, e-mail, and proprietary database information.

VDE enables an electronic commerce marketplace that supports differing, competitive business partnerships, agreements, and evolving overall business models.

The features of VDE allow it to function as the first trusted electronic information control environment that can conform to, and support, the bulk of conventional electronic commerce and data security requirements. In particular, VDE enables the participants in a business value chain model to create an electronic version of traditional business agreement terms and conditions and further enables these participants to shape and evolve their electronic commerce models as they believe appropriate to their business requirements.

VDE offers an architecture that avoids reflecting specific distribution biases, administrative and control perspectives, and content types. Instead, VDE provides a broad-spectrum, fundamentally configurable and portable, electronic transaction control, distributing, usage, auditing, reporting, and payment operating environment. VDE is not limited to being an application or application specific toolset that covers only a limited subset of electronic interaction activities and participants. Rather, VDE supports systems by which such applications can be created, modified, and/or reused. As a result,

the present invention answers pressing, unsolved needs by offering a system that supports a standardized control environment which facilitates interoperability of electronic appliances, interoperability of content containers, and efficient creation of electronic commerce applications and models through the use of a programmable, secure electronic transactions management foundation and reusable and extensible executable components. VDE can support a single electronic "world" within which most forms of electronic transaction activities can be managed.

To answer the developing needs of rights owners and content providers and to provide a system that can accommodate the requirements and agreements of all parties that may be involved in electronic business models (creators, distributors, administrators, users, credit providers, etc.), VDE supplies an efficient, largely transparent, low cost and sufficiently secure system (supporting both hardware/ software and software only models). VDE provides the widely varying secure control and administration capabilities required for:

1. Different types of electronic content,

2. Differing electronic content delivery schemes,

3. Differing electronic content usage schemes,

4. Different content usage platforms, and

5. Differing content marketing and model strategies.

VDE may be combined with, or integrated into, many separate computers and/or other electronic appliances. These appliances typically include a secure subsystem that can enable control of content use such as displaying, encrypting, decrypting, printing, copying, saving, extracting, embedding, distributing, auditing usage, etc. The secure subsystem in the preferred embodiment comprises one or more "protected processing environments", one or more secure databases, and secure "component assemblies" and other items and processes that need to be kept secured. VDE can, for example, securely control electronic currency, payments, and/or credit management (including electronic credit and/or currency receipt, disbursement, encumbering, and/or allocation) using such a "secure subsystem."

VDE provides a secure, distributed electronic transaction management system for controlling the distribution and/or other usage of electronically provided and/or stored information. VDE

controls auditing and reporting of electronic content and/or appliance usage. Users of VDE may include content creators who apply content usage, usage reporting, and/or usage payment related control information to electronic content and/or appliances for users such as end-user organizations, individuals, and content and/or appliance distributors. VDE also securely supports the payment of money owed (including money owed for content and/or appliance usage) by one or more parties to one or more other parties, in the form of electronic credit and/or currency.

Electronic appliances under control of VDE represent VDE 'nodes' that securely process and control; distributed electronic information and/or appliance usage, control information formulation, and related transactions. VDE can securely manage the integration of control information provided by two or more parties. As a result, VDE can construct an electronic agreement between VDE participants that represent a "negotiation" between, the control requirements of, two or more parties and enacts terms and conditions of a resulting agreement. VDE ensures the rights of each party to an electronic agreement regarding a wide range of electronic activities related to electronic information and/or appliance usage.

Through use of VDE's control system, traditional content providers and users can create electronic relationships that reflect traditional, non-electronic relationships. They can shape and modify commercial relationships to accommodate the evolving needs of, and agreements among, themselves. VDE does not require electronic content providers and users to modify their business practices and personal preferences to conform to a metering and control application program that supports limited, largely fixed functionality. Furthermore, VDE permits participants to develop business models not feasible with non-electronic commerce, for example, involving detailed reporting of content usage information, large numbers of distinct transactions at hitherto infeasibly low price points, "pass-along" control information that is enforced without involvement or advance knowledge of the participants, etc.

The present invention allows content providers and users to formulate their transaction environment to accommodate:

(1) desired content models, content control models, and content usage information pathways,

(2) a complete range of electronic media and distribution means,

(3) a broad range of pricing, payment, and auditing strategies,

(4) very flexible privacy and/or reporting models,

(5) practical and effective security architectures, and

(6) other administrative procedures that together with steps (1) through (5) can enable most "real world" electronic commerce and data security models, including models unique to the electronic world.

VDE's transaction management capabilities can enforce:

(1) privacy rights of users related to information regarding their usage of electronic information and/or appliances,

(2) societal policy such as laws that protect rights of content users or require the collection of taxes derived from electronic transaction revenue, and

(3) the proprietary and/or other rights of parties related to ownership of, distribution of, and/or other commercial rights related to, electronic information.

VDE can support "real" commerce in an electronic form, that is the progressive creation of commercial relationships that form, over time, a network of interrelated agreements representing a value chain business model. This is achieved in part by enabling content control information to develop through the interaction of (negotiation between) securely created and independently submitted sets of content and/or appliance control information. Different sets of content and/or appliance control information can be submitted by different parties in an electronic business value chain enabled by the present invention. These parties create control information sets through the use of their respective VDE installations. Independently, securely deliverable, component based control information allows efficient interaction among control information sets supplied by different parties.

VDE permits multiple, separate electronic arrangements to be formed between subsets of parties in a VDE supported electronic value chain model. These multiple agreements together comprise a VDE value chain "extended" agreement.

VDE allows such constituent electronic agreements, and therefore overall VDE extended agreements, to evolve and reshape over time as additional VDE participants become involved in VDE content and/or appliance control information handling. VDE electronic agreements may also be extended as new control information is submitted by existing participants. With VDE, electronic commerce participants are free to structure and restructure their electronic commerce business activities and relationships. As a result, the present invention allows a competitive electronic commerce marketplace to develop since the use of VDE enables different, widely varying business models using the same or shared content.

A significant facet of the present invention's ability to broadly support electronic commerce is its ability to securely manage independently delivered VDE component objects containing control information (normally in the form of VDE objects containing one or more methods, data, or load module VDE components). This independently delivered control information can be integrated with senior and other pre-existing content control information to securely form derived control information using the negotiation mechanisms of the present invention. All requirements specified by this derived control information must be satisfied before VDE controlled content can

be accessed or otherwise used. This means that, for example, all load modules and any mediating data which are listed by the derived control information as required must be available and securely perform their required function. In combination with other aspects of the present invention, securely, independently delivered control components allow electronic commerce participants to freely stipulate their business requirements and trade offs. As a result, much as with traditional, non-electronic commerce, the present invention allows electronic commerce (through a progressive stipulation of various control requirements by VDE participants) to evolve into forms of business that are the most efficient, competitive and useful.

VDE provides capabilities that rationalize the support of electronic commerce and electronic transaction management. This rationalization stems from the reusability of control structures and user interfaces for a wide variety of transaction management related activities. As a result, content usage control, data security, information auditing, and electronic financial activities, can be supported with tools that are reusable, convenient, consistent, and familiar. In addition, a rational approach—a transaction/distribution control standard—allows all participants in VDE the same foundation set of hardware control and security, authoring, administration,

and management tools to support widely varying types of information, business market model, and/or personal objectives.

Employing VDE as a general purpose electronic transaction/distribution control system allows users to maintain a single transaction management control arrangement on each of their computers, networks, communication nodes, and/or other electronic appliances. Such a general purpose system can serve the needs of many electronic transaction management applications without requiring distinct, different installations for different purposes. As a result, users of VDE can avoid the confusion and expense and other inefficiencies of different, limited purpose transaction control applications for each different content and/or business model. For example, VDE allows content creators to use the same VDE foundation control arrangement for both content authoring and for licensing content from other content creators for inclusion into their products or for other use. Clearinghouses, distributors, content creators, and other VDE users can all interact, both with the applications running on their VDE installations, and with each other, in an entirely consistent manner, using and reusing (largely transparently) the same distributed tools, mechanisms, and consistent user interfaces, regardless of the type of VDE activity.

VDE prevents many forms of unauthorized use of electronic information, by controlling and auditing (and other administration of use) electronically stored and/or disseminated information. This includes, for example, commercially distributed content, electronic currency, electronic credit, business transactions (such as EDI), confidential communications, and the like. VDE can further be used to enable commercially provided electronic content to be made available to users in user defined portions, rather than constraining the user to use portions of content that were "predetermined" by a content creator and/or other provider for billing purposes.

VDE, for example, can employ:

(1) Secure metering means for budgeting and/or auditing electronic content and/or appliance usage;

(2) Secure flexible means for enabling compensation and/or billing rates for content and/or appliance usage, including electronic credit and/or currency mechanisms for payment means;

(3) Secure distributed database means for storing control and usage related information (and employing validated compartmentalization and tagging schemes);

(4) Secure electronic appliance control means;

(5) A distributed, secure, "virtual black box" comprised of nodes located at every user (including VDE content container creators, other content providers, client users, and recipients of secure VDE content usage information) site. The nodes of said virtual black box normally include a secure subsystem having at least one secure hardware element (a semiconductor element or other hardware module for securely executing VDE control processes), said secure subsystems being distributed at nodes along a pathway of information storage, distribution, payment, usage, and/or auditing. In some embodiments, the functions of said hardware element, for certain or all nodes, may be performed by software, for example, in host processing environments of electronic appliances;

(6) Encryption and decryption means;

(7) Secure communications means employing authentication, digital signaturing, and encrypted transmissions. The secure subsystems at said user nodes utilize a protocol that establishes and authenticates each node's and/or participant's identity, and establishes one or more secure host-to-host encryption keys for communications between the secure subsystems; and

(8) Secure control means that can allow each VDE installation to perform VDE content authoring (placing content into VDE containers with associated control information), content distribution, and content usage; as well as clearinghouse and other administrative and analysis activities employing content usage information.

VDE may be used to migrate most non-electronic, traditional information delivery models (including entertainment, reference materials, catalog shopping, etc.) into an adequately secure digital distribution and usage management

and payment context. The distribution and financial pathways managed by a VDE arrangement may include:

- content creator(s),
- distributor(s),
- redistributor(s),
- client administrator(s),
- client user(s),
- financial and/or other clearinghouse(s),
- and/or government agencies.

These distribution and financial pathways may also include:

- advertisers,
- market survey organizations, and/or
- other parties interested in the user usage of information securely delivered and/or stored using VDE.

Normally, participants in a VDE arrangement will employ the same secure VDE foundation. Alternate embodiments support VDE arrangements employing differing VDE foundations. Such alternate embodiments may employ procedures to ensure certain interoperability requirements are met.

Secure VDE hardware (also known as SPUs for Secure Processing Units), or VDE installations that use software to substitute for, or complement, said hardware (provided by Host Processing Environments (HPEs)), operate in conjunction with secure communications, systems integration software, and distributed software control information and support structures, to achieve the electronic contract/rights protection environment of the present invention. Together, these VDE components comprise a secure, virtual, distributed content and/or appliance control, auditing (and other administration), reporting, and payment environment. In some embodiments and where commercially acceptable, certain VDE participants, such as clearinghouses that normally maintain sufficiently physically secure non-VDE processing environments, may be allowed to employ HPEs rather VDE hardware elements and interoperate, for example, with VDE end-users and content providers. VDE components together comprise a configurable, consistent, secure and "trusted" architecture for distributed, asynchronous control of electronic content and/or appliance usage. VDE supports a "universe wide" environment for electronic content delivery, broad dissemination, usage reporting, and usage related payment activities.

VDE provides generalized configurability. This results, in part, from decomposition of generalized requirements for supporting electronic commerce and data security into a broad range of constituent "atomic" and higher level components (such as load modules, data elements, and methods) that may be variously aggregated together to form control methods for electronic commerce applications, commercial electronic agreements, and data security arrangements. VDE provides a secure operating environment employing VDE foundation elements along with secure independently deliverable VDE components that enable electronic commerce models and relationships to develop. VDE specifically supports the unfolding of distribution models in which content providers, over time, can expressly agree to, or allow, subsequent content providers and/or users to participate in shaping the control information for, and consequences of, use of electronic content and/or appliances. A very broad range of the functional attributes important for supporting simple to very complex electronic commerce and data security activities are supported by capabilities of the present invention. As a result, VDE supports most types of electronic information and/or appliance: usage control (including distribution), security, usage auditing, reporting, other administration, and payment arrangements.

VDE, in its preferred embodiment, employs object software technology and uses object technology to form "containers" for delivery of information that is (at least in part) encrypted or otherwise secured. These containers may contain electronic content products or other electronic information and some or all of their associated permissions (control) information. These container objects may be distributed along pathways involving content providers and/or content users. They may be securely moved among nodes of a Virtual Distribution Environment (VDE) arrangement, which nodes operate VDE foundation software and execute control methods to enact electronic information usage control and/or administration models. The containers delivered through use of the preferred embodiment of the present invention may be employed both for distributing VDE control instructions (information) and/or to encapsulate and electronically distribute content that has been at least partially secured.

Content providers who employ the present invention may include, for example, software application and game publishers, database publishers, cable, television, and radio broadcasters, electronic shopping vendors, and distributors of information in electronic document, book, periodical, e-mail and/or other forms. Corporations, government agencies, and/or individual

"end-users" who act as storers of, and/or distributors of, electronic information, may also be VDE content providers (in a restricted model, a user provides content only to himself and employs VDE to secure his own confidential information against unauthorized use by other parties). Electronic information may include proprietary and/or confidential information for personal or internal organization use, as well as information, such as software applications, documents, entertainment materials, and/or reference information, which may be provided to other parties. Distribution may be by, for example, physical media delivery, broadcast and/or telecommunication means, and in the form of "static" files and/or streams of data. VDE may also be used, for example, for multi-site "real-time" interaction such as teleconferencing, interactive games, or on-line bulletin boards, where restrictions on, and/or auditing of, the use of all or portions of communicated information is enforced.

VDE provides important mechanisms for both enforcing commercial agreements and enabling the protection of privacy rights. VDE can securely deliver information from one party to another concerning the use of commercially distributed electronic content. Even if parties are separated by several "steps" in a chain (pathway) of handling for such content usage information, such information is protected by VDE through encryption and/or

other secure processing. Because of that protection, the accuracy of such information is guaranteed by VDE, and the information can be trusted by all parties to whom it is delivered. Furthermore, VDE guarantees that all parties can trust that such information cannot be received by anyone other than the intended, authorized, party(ies) because it is encrypted such that only an authorized party, or her agents, can decrypt it. Such information may also be derived through a secure VDE process at a previous pathway-of-handling location to produce secure VDE reporting information that is then communicated securely to its intended recipient's VDE secure subsystem. Because VDE can deliver such information securely, parties to an electronic agreement need not trust the accuracy of commercial usage and/or other information delivered through means other than those under control of VDE.

VDE participants in a commercial value chain can be "commercially" confident (that is, sufficiently confident for commercial purposes) that the direct (constituent) and/or "extended" electronic agreements they entered into through the use of VDE can be enforced reliably. These agreements may have both "dynamic" transaction management related aspects, such as content usage control information enforced through budgeting, metering, and/or reporting of electronic information

and/or appliance use, and/or they may include "static" electronic assertions, such as an end-user using the system to assert his or her agreement to pay for services, not to pass to unauthorized parties electronic information derived from usage of content or systems, and/or agreeing to observe copyright laws. Not only can electronically reported transaction related information be trusted under the present invention, but payment may be automated by the passing of payment tokens through a pathway of payment (which may or may not be the same as a pathway for reporting). Such payment can be contained within a VDE container created automatically by a VDE installation in response to control information (located, in the preferred embodiment, in one or more permissions records) stipulating the "withdrawal" of credit or electronic currency (such as tokens) from an electronic account (for example, an account securely maintained by a user's VDE installation secure subsystem) based upon usage of VDE controlled electronic content and/or appliances (such as governments, financial credit providers, and users).

VDE allows the needs of electronic commerce participants to be served and it can bind such participants together in a universe wide, trusted commercial network that can be secure enough to support very large amounts of commerce. VDE's security and metering secure subsystem core will be present at

all physical locations where VDE related content is (a) assigned usage related control information (rules and mediating data), and/or (b) used. This core can perform security and auditing functions (including metering) that operate within a "virtual black box," a collection of distributed, very secure VDE related hardware instances that are interconnected by secured information exchange (for example, telecommunication) processes and distributed database means. VDE further includes highly configurable transaction operating system technology, one or more associated libraries of load modules along with affiliated data, VDE related administration, data preparation, and analysis applications, as well as system software designed to enable VDE integration into host environments and applications. VDE's usage control information, for example, provide for property content and/or appliance related: usage authorization, usage auditing (which may include audit reduction), usage billing, usage payment, privacy filtering, reporting, and security related communication and encryption techniques.

VDE extensively employs methods in the form of software objects to augment configurability, portability, and security of the VDE environment. It also employs a software object architecture for VDE content containers that carries protected

content and may also carry both freely available information (e.g, summary, table of contents) and secured content control information which ensures the performance of control information. Content control information governs content usage according to criteria set by holders of rights to an object's contents and/or according to parties who otherwise have rights associated with distributing such content (such as governments, financial credit providers, and users).

In part, security is enhanced by object methods employed by the present invention because the encryption schemes used to protect an object can efficiently be further used to protect the associated content control information (software control information and relevant data) from modification. Said object techniques also enhance portability between various computer and/or other appliance environments because electronic information in the form of content can be inserted along with (for example, in the same object container as) content control information (for said content) to produce a "published" object. As a result, various portions of said control information may be specifically adapted for different environments, such as for diverse computer platforms and operating systems, and said various portions may all be carried by a VDE container.

An objective of VDE is supporting a transaction/distribution control standard. Development of such a standard has many obstacles, given the security requirements and related hardware and communications issues, widely differing environments, information types, types of information usage, business and/or data security goals, varieties of participants, and properties of delivered information. A significant feature of VDE accommodates the many, varying distribution and other transaction variables by, in part, decomposing electronic commerce and data security functions into generalized capability modules executable within a secure hardware SPU and/or corresponding software subsystem and further allowing extensive flexibility in assembling, modifying, and/or replacing, such modules (e.g. load modules and/or methods) in applications run on a VDE installation foundation. This configurability and reconfigurability allows electronic commerce and data security participants to reflect their priorities and requirements through a process of iteratively shaping an evolving extended electronic agreement (electronic control model). This shaping can occur as content control information passes from one VDE participant to another and to the extent allowed by "in place" content control information. This process allows users of VDE to recast existing control

information and/or add new control information as necessary (including the elimination of no longer required elements).

VDE supports trusted (sufficiently secure) electronic information distribution and usage control models for both commercial electronic content distribution and data security applications. It can be configured to meet the diverse requirements of a network of interrelated participants that may include content creators, content distributors, client administrators, end users, and/or clearinghouses and/or other content usage information users. These parties may constitute a network of participants involved in simple to complex electronic content dissemination, usage control, usage reporting, and/or usage payment. Disseminated content may include both originally provided and VDE generated information (such as content usage information) and content control information may persist through both chains (one or more pathways) of content and content control information handling, as well as the direct usage of content. The configurability provided by the present invention is particularly critical for supporting electronic commerce, that is enabling businesses to create relationships and evolve strategies that offer competitive value. Electronic commerce tools that are not inherently configurable and interoperable will ultimately fail to produce products (and

services) that meet both basic requirements and evolving needs of most commerce applications.

VDE's fundamental configurability will allow a broad range of competitive electronic commerce business models to flourish. It allows business models to be shaped to maximize revenues sources, end-user product value, and operating efficiencies. VDE can be employed to support multiple, differing models, take advantage of new revenue opportunities, and deliver product configurations most desired by users. Electronic commerce technologies that do not, as the present invention does:

- support a broad range of possible, complementary revenue activities,
- offer a flexible array of content usage features most desired by customers, and
- exploit opportunities for operating efficiencies,

will result in products that are often intrinsically more costly and less appealing and therefore less competitive in the marketplace.

Some of the key factors contributing to the configurability intrinsic to the present invention include:

(a) integration into the fundamental control environment of a broad range of electronic appliances through portable API and programming language tools that efficiently support merging of control and auditing capabilities in nearly any electronic appliance environment while maintaining overall system security;

(b) modular data structures;

(c) generic content model;

(d) general modularity and independence of foundation architectural components;

(e) modular security structures;

(f) variable length and multiple branching chains of control; and

(g) independent, modular control structures in the form of executable load modules that can be maintained in one or more libraries, and assembled into control methods and models, and where such model control

schemes can "evolve" as control information passes through the VDE installations of participants of a pathway of VDE content control information handling.

Because of the breadth of issues resolved by the present invention, it can provide the emerging "electronic highway" with a single transaction/distribution control system that can, for a very broad range of commercial and data security models, ensure against unauthorized use of confidential and/or proprietary information and commercial electronic transactions. VDE's electronic transaction management mechanisms can enforce the electronic rights and agreements of all parties participating in widely varying business and data security models, and this can be efficiently achieved through a single VDE implementation within each VDE participant's electronic appliance. VDE supports widely varying business and/or data security models that can involve a broad range of participants at various "levels" of VDE content and/or content control information pathways of handling. Different content control and/or auditing models and agreements may be available on the same VDE installation. These models and agreements may control content in relationship to, for example, VDE installations and/or users in general; certain specific users, installations, classes and/or other

groupings of installations and/or users; as well as to electronic content generally on a given installation, to specific properties, property portions, classes and/or other groupings of content.

Distribution using VDE may package both the electronic content and control information into the same VDE container, and/or may involve the delivery to an end-user site of different pieces of the same VDE managed property from plural separate remote locations and/or in plural separate VDE content containers and/or employing plural different delivery means. Content control information may be partially or fully delivered separately from its associated content to a user VDE installation in one or more VDE administrative objects. Portions of said control information may be delivered from one or more sources. Control information may also be available for use by access from a user's VDE installation secure sub-system to one or more remote VDE secure sub-systems and/or VDE compatible, certified secure remote locations. VDE control processes such as metering, budgeting, decrypting and/or fingerprinting, may as relates to a certain user content usage activity, be performed in a user's local VDE installation secure subsystem, or said processes may be divided amongst plural secure subsystems which may be located in the same user VDE installations and/or in a network server and in the user installation. For example, a local VDE

installation may perform decryption and save any, or all of, usage metering information related to content and/or electronic appliance usage at such user installation could be performed at the server employing secure (e.g., encrypted) communications between said secure subsystems. Said server location may also be used for near real time, frequent, or more periodic secure receipt of content usage information from said user installation, with, for example, metered information being maintained only temporarily at a local user installation.

Delivery means for VDE managed content may include electronic data storage means such as optical disks for delivering one portion of said information and broadcasting and/or telecommunicating means for other portions of said information. Electronic data storage means may include magnetic media, optical media, combined magneto-optical systems, flash RAM memory, bubble memory, and/or other memory storage means such as huge capacity optical storage systems employing holographic, frequency, and/or polarity data storage techniques. Data storage means may also employ layered disc techniques, such as the use of generally transparent and/or translucent materials that pass light through layers of data carrying discs which themselves are physically packaged together as one

thicker disc. Data carrying locations on such discs may be, at least in part, opaque.

VDE supports a general purpose foundation for secure transaction management, including usage control, auditing, reporting, and/or payment. This general purpose foundation is called "VDE Functions" ("VDEFs"). VDE also supports a collection of "atomic" application elements (e.g., load modules) that can be selectively aggregated together to form various VDEF capabilities called control methods and which serve as VDEF applications and operating system functions. When a host operating environment of an electronic appliance includes VDEF capabilities, it is called a "Rights Operating System" (ROS). VDEF load modules, associated data, and methods form a body of information that for the purposes of the present invention are called "control information." VDEF control information may be specifically associated with one or more pieces of electronic content and/or it may be employed as a general component of the operating system capabilities of a VDE installation.

VDEF transaction control elements reflect and enact content specific and/or more generalized administrative (for example, general operating system) control information. VDEF capabilities which can generally take the form of applications

(application models) that have more or less configurability which can be shaped by VDE participants, through the use, for example, of VDE templates, to employ specific capabilities, along, for example, with capability parameter data to reflect the elements of one or more express electronic agreements between VDE participants in regards to the use of electronic content such as commercially distributed products. These control capabilities manage the use of, and/or auditing of use of, electronic content, as well as reporting information based upon content use, and any payment for said use. VDEF capabilities may "evolve" to reflect the requirements of one or more successive parties who receive or otherwise contribute to a given set of control information. Frequently, for a VDE application for a given content model (such as distribution of entertainment on CD-ROM, content delivery from an Internet repository, or electronic catalog shopping and advertising, or some combination of the above) participants would be able to securely select from amongst available, alternative control methods and apply related parameter data, wherein such selection of control method and/or submission of data would constitute their "contribution" of control information. Alternatively, or in addition, certain control methods that have been expressly certified as securely interoperable and compatible with said application may be independently submitted by a participant as part of such a

contribution. In the most general example, a generally certified load module (certified for a given VDE arrangement and/or content class) may be used with many or any VDE application that operates in nodes of said arrangement. These parties, to the extent they are allowed, can independently and securely add, delete, and/or otherwise modify the specification of load modules and methods, as well as add, delete or otherwise modify related information.

Normally the party who creates a VDE content container defines the general nature of the VDEF capabilities that will and/or may apply to certain electronic information. A VDE content container is an object that contains both content (for example, commercially distributed electronic information products such as computer software programs, movies, electronic publications or reference materials, etc.) and certain control information related to the use of the object's content. A creating party may make a VDE container available to other parties. Control information delivered by, and/or otherwise available for use with, VDE content containers comprise (for commercial content distribution purposes) VDEF control capabilities (and any associated parameter data) for electronic content. These capabilities may constitute one or more "proposed" electronic agreements (and/or agreement functions available for selection

and/or use with parameter data) that manage the use and/or the consequences of use of such content and which can enact the terms and conditions of agreements involving multiple parties and their various rights and obligations.

A VDE electronic agreement may be explicit, through a user interface acceptance by one or more parties, for example by a "junior" party who has received control information from a "senior" party, or it may be a process amongst equal parties who individually assert their agreement. Agreement may also result from an automated electronic process during which terms and conditions are "evaluated" by certain VDE participant control information that assesses whether certain other electronic terms and conditions attached to content and/or submitted by another party are acceptable (do not violate acceptable control information criteria). Such an evaluation process may be quite simple, for example a comparison to ensure compatibility between a portion of, or all senior, control terms and conditions in a table of terms and conditions and the submitted control information of a subsequent participant in a pathway of content control information handling, or it may be a more elaborate process that evaluates the potential outcome of, and/or implements a negotiation process between, two or more sets of control information submitted by two or more parties. VDE also

accommodates a semi-automated process during which one or more VDE participants directly, through user interface means, resolve "disagreements" between control information sets by accepting and/or proposing certain control information that may be acceptable to control information representing one or more other parties interests and/or responds to certain user interface queries for selection of certain alternative choices and/or for certain parameter information, the responses being adopted if acceptable to applicable senior control information.

When another party (other than the first applier of rules), perhaps through a negotiation process, accepts, and/or adds to and/or otherwise modifies, "in place" content control information, a VDE agreement between two or more parties related to the use of such electronic content may be created (so long as any modifications are consistent with senior control information). Acceptance of terms and conditions related to certain electronic content may be direct and express, or it may be implicit as a result of use of content (depending, for example, on legal requirements, previous exposure to such terms and conditions, and requirements of in place control information).

VDEF capabilities may be employed, and a VDE agreement may be entered into, by a plurality of parties without

the VDEF capabilities being directly associated with the controlling of certain, specific electronic information. For example, certain one or more VDEF capabilities may be present at a VDE installation, and certain VDE agreements may have been entered into during the registration process for a content distribution application, to be used by such installation for securely controlling VDE content usage, auditing, reporting and/or payment. Similarly, a specific VDE participant may enter into a VDE user agreement with a VDE content or electronic appliance provider when the user and/or her appliance register with such provider as a VDE installation and/or user. In such events, VDEF in place control information available to the user VDE installation may require that certain VDEF methods are employed, for example in a certain sequence, in order to be able to use all and/or certain classes, of electronic content and/or VDE applications.

VDE ensures that certain prerequisites necessary for a given transaction to occur are met. This includes the secure execution of any required load modules and the availability of any required, associated data. For example, required load modules and data (e.g. in the form of a method) might specify that sufficient credit from an authorized source must be confirmed as available. It might further require certain one or

more load modules execute as processes at an appropriate time to ensure that such credit will be used in order to pay for user use of the content. A certain content provider might, for example, require metering the number of copies made for distribution to employees of a given software program (a portion of the program might be maintained in encrypted form and require the presence of a VDE installation to run). This would require the execution of a metering method for copying of the property each time a copy was made for another employee. This same provider might also charge fees based on the total number of different properties licensed from them by the user and a metering history of their licensing of properties might be required to maintain this information.

VDE provides organization, community, and/or universe wide secure environments whose integrity is assured by processes securely controlled in VDE participant user installations (nodes). VDE installations, in the preferred embodiment, may include both software and tamper resistant hardware semiconductor elements. Such a semiconductor arrangement comprises, at least in part, special purpose circuitry that has been designed to protect against tampering with, or unauthorized observation of, the information and functions used in performing the VDE's control functions. The special purpose

secure circuitry provided by the present invention includes at least one of: a dedicated semiconductor arrangement known as a Secure Processing Unit (SPU) and/or a standard microprocessor, microcontroller, and/or other processing logic that accommodates the requirements of the present invention and functions as an SPU. VDE's secure hardware may be found incorporated into, for example, a fax/modem chip or chip pack, I/O controller, video display controller, and/or other available digital processing arrangements. It is anticipated that portions of the present invention's VDE secure hardware capabilities may ultimately be standard design elements of central processing units (CPUs) for computers and various other electronic devices.

Designing VDE capabilities into one or more standard microprocessor, microcontroller and/or other digital processing components may materially reduce VDE related hardware costs by employing the same hardware resources for both the transaction management uses contemplated by the present invention and for other, host electronic appliance functions. This means that a VDE SPU can employ (share) circuitry elements of a "standard" CPU. For example, if a "standard" processor can operate in protected mode and can execute VDE related instructions as a protected activity, then such an embodiment may provide sufficient hardware security for a variety of

applications and the expense of a special purpose processor might be avoided. Under one preferred embodiment of the present invention, certain memory (e.g., RAM, ROM, NVRAM) is maintained during VDE related instruction processing in a protected mode (for example, as supported by protected mode microprocessors). This memory is located in the same package as the processing logic (e.g. processor). Desirably, the packaging and memory of such a processor would be designed using security techniques that enhance its resistance to tampering.

The degree of overall security of the VDE system is primarily dependent on the degree of tamper resistance and concealment of VDE control process execution and related data storage activities. Employing special purpose semiconductor packaging techniques can significantly contribute to the degree of security. Concealment and tamper-resistance in semiconductor memory (e.g., RAM, ROM, NVRAM) can be achieved, in part, by employing such memory within an SPU package, by encrypting data before it is sent to external memory (such as an external RAM package) and decrypting encrypted data within the CPU/RAM package before it is executed. This process is used for important VDE related data when such data is stored on unprotected media, for example, standard host storage, such as random access memory, mass storage, etc. In

that event, a VDE SPU would encrypt data that results from a secure VDE execution before such data was stored in external memory.

Summary of Some Important Features Provided by VDE in Accordance With the Present Invention

VDE employs a variety of capabilities that serve as a foundation for a general purpose, sufficiently secure distributed electronic commerce solution. VDE enables an electronic commerce marketplace that supports divergent, competitive business partnerships, agreements, and evolving overall business models. For example, VDE includes features that:

"sufficiently" impede unauthorized and/or uncompensated use of electronic information and/or appliances through the use of secure communication, storage, and transaction management technologies. VDE supports a model wide, distributed security implementation which creates a single secure "virtual" transaction processing and information storage environment. VDE enables distributed VDE installations to securely store and communicate information and remotely control the execution processes and the

character of use of electronic information at other VDE installations and in a wide variety of ways;

- support low-cost, efficient, and effective security architectures for transaction control, auditing, reporting, and related communications and information storage. VDE may employ tagging related security techniques, the time-ageing of encryption keys, the compartmentalization of both stored control information (including differentially tagging such stored information to ensure against substitution and tampering) and distributed content (to, for many content applications, employ one or more content encryption keys that are unique to the specific VDE installation and/or user), private key techniques such as triple DES to encrypt content, public key techniques such as RSA to protect communications and to provide the benefits of digital signature and authentication to securely bind together the nodes of a VDE arrangement, secure processing of important transaction management executable code, and a combining of a small amount of highly secure, hardware protected storage space with a much larger "exposed" mass media storage

space storing secured (normally encrypted and tagged) control and audit information. VDE employs special purpose hardware distributed throughout some or all locations of a VDE implementation: a) said hardware controlling important elements of: content preparation (such as causing such content to be placed in a VDE content container and associating content control information with said content), content and/or electronic appliance usage auditing, content usage analysis, as well as content usage control; and b) said hardware having been designed to securely handle processing load module control activities, wherein said control processing activities may involve a sequence of required control factors;

- support dynamic user selection of information subsets of a VDE electronic information product (VDE controlled content). This contrasts with the constraints of having to use a few high level individual, pre-defined content provider information increments such as being required to select a whole information product or product section in order to acquire or otherwise use a portion of such product or

section. VDE supports metering and usage control over a variety of increments (including "atomic" increments, and combinations of different increment types) that are selected ad hoc by a user and represent a collection of pre-identified one or more increments (such as one or more blocks of a preidentified nature, e.g., bytes, images, logically related blocks) that form a generally arbitrary, but logical to a user, content "deliverable." VDE control information (including budgeting, pricing and metering) can be configured so that it can specifically apply, as appropriate, to ad hoc selection of different, unanticipated variable user selected aggregations of information increments and pricing levels can be, at least in part, based on quantities and/or nature of mixed increment selections (for example, a certain quantity of certain text could mean associated images might be discounted by 15%; a greater quantity of text in the "mixed" increment selection might mean the images are discounted 20%). Such user selected aggregated information increments can reflect the actual requirements of a user for information and is more flexible than being limited to a single, or a few, high

level, (e.g. product, document, database record) predetermined increments. Such high level increments may include quantities of information not desired by the user and as a result be more costly than the subset of information needed by the user if such a subset was available. In sum, the present invention allows information contained in electronic information products to be supplied according to user specification. Tailoring to user specification allows the present invention to provide the greatest value to users, which in turn will generate the greatest amount of electronic commerce activity. The user, for example, would be able to define an aggregation of content derived from various portions of an available content product, but which, as a deliverable for use by the user, is an entirely unique aggregated increment. The user may, for example, select certain numbers of bytes of information from various portions of an information product, such as a reference work, and copy them to disc in unencrypted form and be billed based on total number of bytes plus a surcharge on the number of "articles" that provided the bytes. A content provider might reasonably charge less for

such a user defined information increment since the user does not require all of the content from all of the articles that contained desired information. This process of defining a user desired information increment may involve artificial intelligence database search tools that contribute to the location of the most relevant portions of information from an information product and cause the automatic display to the user of information describing search criteria hits for user selection or the automatic extraction and delivery of such portions to the user. VDE further supports a wide variety of predefined increment types including:

- bytes,
- images,
- content over time for audio or video, or any

other increment that can be identified by content provider data mapping efforts, such as:

- sentences,
- paragraphs,
- articles,
- database records, and
- byte offsets representing increments of logically related information.

VDE supports as many simultaneous predefined increment types as may be practical for a given type of content and business model.

- securely store at a user's site potentially highly detailed information reflective of a user's usage of a variety of different content segment types and employing both inexpensive "exposed" host mass storage for maintaining detailed information in the form of encrypted data and maintaining summary information for security testing in highly secure special purpose VDE installation nonvolatile memory (if available).

- support trusted chain of handling capabilities for pathways of distributed electronic information and/or for content usage related information. Such chains may extend, for example, from a content creator, to a distributor, a redistributor, a client user, and then may provide a pathway for securely reporting the same and/or differing usage information to one or more auditors, such as to one or more independent clearinghouses and then back to the content providers, including content creators.

The same and/or different pathways employed for certain content handling, and related content control information and reporting information handling, may also be employed as one or more pathways for electronic payment handling (payment is characterized in the present invention as administrative content) for electronic content and/or appliance usage. These pathways are used for conveyance of all or portions of content, and/or content related control information. Content creators and other providers can specify the pathways that, partially or fully, must be used to disseminate commercially distributed property content, content control information, payment administrative content, and/or associated usage reporting information. Control information specified by content providers may also specify which specific parties must or may (including, for example, a group of eligible parties from which a selection may be made) handle conveyed information. It may also specify what transmission means (for example telecommunication carriers or media types) and transmission hubs must or may be used.

- support flexible auditing mechanisms, such as employing "bitmap meters," that achieve a high degree of efficiency of operation and throughput and allow, in a practical manner, the retention and ready recall of information related to previous usage activities and related patterns. This flexibility is adaptable to a wide variety of billing and security control strategies such as:
  - upgrade pricing (e.g. suite purchases),
  - pricing discounts (including quantity discounts),
  - billing related time duration variables such as discounting new purchases based on the timing of past purchases, and
  - security budgets based on quantity of different, logically related units of electronic information used over an interval of time.

Use of bitmap meters (including "regular" and "wide" bitmap meters) to record usage and/or purchase of information, in conjunction with other elements of the preferred embodiment of the present invention, uniquely supports efficient maintenance of usage history for: (a) rental, (b) flat fee licensing

or purchase, (c) licensing or purchase discounts based upon historical usage variables, and (d) reporting to users in a manner enabling users to determine whether a certain item was acquired, or acquired within a certain time period (without requiring the use of conventional database mechanisms, which are highly inefficient for these applications). Bitmap meter methods record activities associated with electronic appliances, properties, objects, or portions thereof, and/or administrative activities that are independent of specific properties, objects, etc., performed by a user and/or electronic appliance such that a content and/or appliance provider and/or controller of an administrative activity can determine whether a certain activity has occurred at some point, or during a certain period, in the past (for example, certain use of a commercial electronic content product and/or appliance). Such determinations can then be used as part of pricing and/or control strategies of a content and/or appliance provider, and/or controller of an administrative activity. For example, the content provider may choose to charge only once for access to a portion of a property,

regardless of the number of times that portion of the property is accessed by a user.

- support "launchable" content, that is content that can be provided by a content provider to an end-user, who can then copy or pass along the content to other end-user parties without requiring the direct participation of a content provider to register and/or otherwise initialize the content for use. This content goes "out of (the traditional distribution) channel" in the form of a "traveling object." Traveling objects are containers that securely carry at least some permissions information and/or methods that are required for their use (such methods need not be carried by traveling objects if the required methods will be available at, or directly available to, a destination VDE installation). Certain travelling objects may be used at some or all VDE installations of a given VDE arrangement since they can make available the content control information necessary for content use without requiring the involvement of a commercial VDE value chain participant or data security administrator (e.g. a control officer or network

administrator). As long as traveling object control information requirements are available at the user VDE installation secure subsystem (such as the presence of a sufficient quantity of financial credit from an authorized credit provider), at least some travelling object content may be used by a receiving party without the need to establish a connection with a remote VDE authority (until, for example, budgets are exhausted or a time content usage reporting interval has occurred). Traveling objects can travel "out-of-channel," allowing, for example, a user to give a copy of a traveling object whose content is a software program, a movie or a game, to a neighbor, the neighbor being able to use the traveling object if appropriate credit (e.g. an electronic clearinghouse account from a clearinghouse such as VISA or AT&T) is available. Similarly, electronic information that is generally available on an Internet, or a similar network, repository might be provided in the form of a traveling object that can be downloaded and subsequently copied by the initial downloader and then passed along to other parties who may pass the object on to additional parties.

- provide very flexible and extensible user identification according to individuals, installations, by groups such as classes, and by function and hierarchical identification employing a hierarchy of levels of client identification (for example, client organization ID, client department ID, client network ID, client project ID, and client employee ID, or any appropriate subset of the above).

- provide a general purpose, secure, component based content control and distribution system that functions as a foundation transaction operating system environment that employs executable code pieces crafted for transaction control and auditing. These code pieces can be reused to optimize efficiency in creation and operation of trusted, distributed transaction management arrangements. VDE supports providing such executable code in the form of "atomic" load modules and associated data. Many such load modules are inherently configurable, aggregatable, portable, and extensible and singularly, or in combination (along with associated data), run as control methods under the VDE transaction operating environment. VDE can

satisfy the requirements of widely differing electronic commerce and data security applications by, in part, employing this general purpose transaction management foundation to securely process VDE transaction related control methods. Control methods are created primarily through the use of one or more of said executable, reusable load module code pieces (normally in the form of executable object components) and associated data. The component nature of control methods allows the present invention to efficiently operate as a highly configurable content control system. Under the present invention, content control models can be iteratively and asynchronously shaped, and otherwise updated to accommodate the needs of VDE participants to the extent that such shaping and otherwise updating conforms to constraints applied by a VDE application, if any (e.g., whether new component assemblies are accepted and, if so, what certification requirements exist for such component assemblies or whether any or certain participants may shape any or certain control information by selection amongst optional control information (permissions record) control methods.

This iterative (or concurrent) multiple participant process occurs as a result of the submission and use of secure, control information components (executable code such as load modules and/or methods, and/or associated data). These components may be contributed independently by secure communication between each control information influencing VDE participant's VDE installation and may require certification for use with a given application, where such certification was provided by a certification service manager for the VDE arrangement who ensures secure interoperability and/or reliability (e.g., bug control resulting from interaction) between appliances and submitted control methods. The transaction management control functions of a VDE electronic appliance transaction operating environment interact with non-secure transaction management operating system functions to properly direct transaction processes and data related to electronic information security, usage control, auditing, and usage reporting. VDE provides the capability to manages resources related to secure VDE content

and/or appliance control information execution and data storage.

- facilitate creation of application and/or system functionality under VDE and to facilitate integration into electronic appliance environments of load modules and methods created under the present invention. To achieve this, VDE employs an Application Programmer's Interface (API) and/or a transaction operating system (such as a ROS) programming language with incorporated functions, both of which support the use of capabilities and can be used to efficiently and tightly integrate VDE functionality into commercial and user applications.

- support user interaction through: (a) "Pop-Up" applications which, for example, provide messages to users and enable users to take specific actions such as approving a transaction, (b) stand-alone VDE applications that provide administrative environments for user activities such as: end-user preference specifications for limiting the price per transaction, unit of time, and/or session, for

accessing history information concerning previous transactions, for reviewing financial information such as budgets, expenditures (e.g. detailed and/or summary) and usage analysis information, and (c) VDE aware applications which, as a result of the use of a VDE API and/or a transaction management (for example, ROS based) programming language embeds VDE "awareness" into commercial or internal software (application programs, games, etc.) so that VDE user control information and services are seamlessly integrated into such software and can be directly accessed by a user since the underlying functionality has been integrated into the commercial software's native design. For example, in a VDE aware word processor application, a user may be able to "print" a document into a VDE content container object, applying specific control information by selecting from amongst a series of different menu templates for different purposes (for example, a confidential memo template for internal organization purposes may restrict the ability to "keep," that is to make an electronic copy of the memo).

- employ "templates" to ease the process of configuring capabilities of the present invention as they relate to specific industries or businesses. Templates are applications or application add-ons under the present invention. Templates support the efficient specification and/or manipulation of criteria related to specific content types, distribution approaches, pricing mechanisms, user interactions with content and/or administrative activities, and/or the like. Given the very large range of capabilities and configurations supported by the present invention, reducing the range of configuration opportunities to a manageable subset particularly appropriate for a given business model allows the full configurable power of the present invention to be easily employed by "typical" users who would be otherwise burdened with complex programming and/or configuration design responsibilities template applications can also help ensure that VDE related processes are secure and optimally bug free by reducing the risks associated with the contribution of independently developed load modules, including unpredictable aspects of code interaction between independent modules and applications, as well as security risks

associated with possible presence of viruses in such modules. VDE, through the use of templates, reduces typical user configuration responsibilities to an appropriately focused set of activities including selection of method types (e.g. functionality) through menu choices such as multiple choice, icon selection, and/or prompting for method parameter data (such as identification information, prices, budget limits, dates, periods of time, access rights to specific content, etc.) that supply appropriate and/or necessary data for control information purposes. By limiting the typical (non-programming) user to a limited subset of configuration activities whose general configuration environment (template) has been preset to reflect general requirements corresponding to that user, or a content or other business model can very substantially limit difficulties associated with content containerization (including placing initial control information on content), distribution, client administration, electronic agreement implementation, end-user interaction, and clearinghouse activities, including associated interoperability problems (such as conflicts resulting from security, operating system,

and/or certification incompatibilities). Use of appropriate VDE templates can assure users that their activities related to content VDE containerization, contribution of other control information, communications, encryption techniques and/or keys, etc. will be in compliance with specifications for their distributed VDE arrangement. VDE templates constitute preset configurations that can normally be reconfigurable to allow for new and/or modified templates that reflect adaptation into new industries as they evolve or to reflect the evolution or other change of an existing industry. For example, the template concept may be used to provide individual, overall frameworks for organizations and individuals that create, modify, market, distribute, consume, and/or otherwise use movies, audio recordings and live performances, magazines, telephony based retail sales, catalogs, computer software, information data bases, multimedia, commercial communications, advertisements, market surveys, infomercials, games, CAD/CAM services for numerically controlled machines, and the like. As the context surrounding these templates changes or evolves,

template applications provided under the present invention may be modified to meet these changes for broad use, or for more focused activities. A given VDE participant may have a plurality of templates available for different tasks. A party that places content in its initial VDE container may have a variety of different, configurable templates depending on the type of content and/or business model related to the content. An end-user may have different configurable templates that can be applied to different document types (e-mail, secure internal documents, database records, etc.) and/or subsets of users (applying differing general sets of control information to different bodies of users, for example, selecting a list of users who may, under certain preset criteria, use a certain document). Of course, templates may, under certain circumstances have fixed control information and not provide for user selections or parameter data entry.

- support plural, different control models regulating the use and/or auditing of either the same specific copy of electronic information content and/or differently regulating different copies (occurrences)

of the same electronic information content. Differing models for billing, auditing, and security can be applied to the same piece of electronic information content and such differing sets of control information may employ, for control purposes, the same, or differing, granularities of electronic information control increments. This includes supporting variable control information for budgeting and auditing usage as applied to a variety of predefined increments of electronic information, including employing a variety of different budgets and/or metering increments for a given electronic information deliverable for: billing units of measure, credit limit. security budget limit and security content metering increments, and/or market surveying and customer profiling content metering increments. For example, a CD-ROM disk with a database of scientific articles might be in part billed according to a formula based on the number of bytes decrypted, number of articles containing said bytes decrypted, while a security budget might limit the use of said database to no more than 5% of the database per month for users on the wide area network it is installed on.

- provide mechanisms to persistently maintain trusted content usage and reporting control information through both a sufficiently secure chain of handling of content and content control information and through various forms of usage of such content wherein said persistence of control may survive such use. Persistence of control includes the ability to extract information from a VDE container object by creating a new container whose contents are at least in part secured and that contains both the extracted content and at least a portion of the control information which control information of the original container and/or are at least in part produced by control information of the original container for this purpose and/or VDE installation control information stipulates should persist and/or control usage of content in the newly formed container. Such control information can continue to manage usage of container content if the container is "embedded" into another VDE managed object, such as an object which contains plural embedded VDE containers, each of which contains content derived (extracted) from a different source.

- enables users, other value chain participants (such as clearinghouses and government agencies), and/or user organizations, to specify preferences or requirements related to their use of electronic content and/or appliances. Content users, such as end-user customers using commercially distributed content (games, information resources, software programs, etc.), can define, if allowed by senior control information, budgets, and/or other control information, to manage their own internal use of content. Uses include, for example, a user setting a limit on the price for electronic documents that the user is willing to pay without prior express user authorization, and the user establishing the character of metering information he or she is willing to allow to be collected (privacy protection). This includes providing the means for content users to protect the privacy of information derived from their use of a VDE installation and content and/or appliance usage auditing. In particular, VDE can prevent information related to a participant's usage of electronic content from being provided to other parties without the participant's tacit or explicit agreement.

- provide mechanisms that allow control information to "evolve" and be modified according, at least in part, to independently, securely delivered further control information. Said control information may include executable code (e.g., load modules) that has been certified as acceptable (e.g., reliable and trusted) for use with a specific VDE application, class of applications, and/or a VDE distributed arrangement. This modification (evolution) of control information can occur upon content control information (load modules and any associated data) circulating to one or more VDE participants in a pathway of handling of control information, or it may occur upon control information being received from a VDE participant. Handlers in a pathway of handling of content control information, to the extent each is authorized, can establish, modify, and/or contribute to, permission, auditing, payment, and reporting control information related to controlling, analyzing, paying for, and/or reporting usage of, electronic content and/or appliances (for example, as related to usage of VDE controlled property content). Independently delivered (from an independent source which is independent except in

regards to certification), at least in part secure, control information can be employed to securely modify content control information when content control information has flowed from one party to another party in a sequence of VDE content control information handling. This modification employs, for example, one or more VDE component assemblies being securely processed in a VDE secure subsystem. In an alternate embodiment, control information may be modified by a senior party through use of their VDE installation secure sub-system after receiving submitted, at least in part secured, control information from a "junior" party, normally in the form of a VDE administrative object. Control information passing along VDE pathways can represent a mixed control set, in that it may include: control information that persisted through a sequence of control information handlers, other control information that was allowed to be modified, and further control information representing new control information and/or mediating data. Such a control set represents an evolution of control information for disseminated content. In this example the overall content control

set for a VDE content container is "evolving" as it securely (e.g. communicated in encrypted form and using authentication and digital signaturing techniques) passes, at least in part, to a new participant's VDE installation where the proposed control information is securely received and handled. The received control information may be integrated (through use of the receiving parties' VDE installation secure sub-system) with in-place control information through a negotiation process involving both control information sets. For example, the modification, within the secure sub-system of a content provider's VDE installation, of content control information for a certain VDE content container may have occurred as a result of the incorporation of required control information provided by a financial credit provider. Said credit provider may have employed their VDE installation to prepare and securely communicate (directly or indirectly) said required control information to said content provider. Incorporating said required control information enables a content provider to allow the credit provider's credit to be employed by a content end-user to compensate for the end-user's

use of VDE controlled content and/or appliances, so long as said end-user has a credit account with said financial credit provider and said credit account has sufficient credit available. Similarly, control information requiring the payment of taxes and/or the provision of revenue information resulting from electronic commerce activities may be securely received by a content provider. This control information may be received, for example, from a government agency. Content providers might be required by law to incorporate such control information into the control information for commercially distributed content and/or services related to appliance usage. Proposed control information is used to an extent allowed by senior control information and as determined by any negotiation trade-offs that satisfy priorities stipulated by each set (the received set and the proposed set). VDE also accommodates different control schemes specifically applying to different participants (e.g., individual participants and/or participant classes (types)) in a network of VDE content handling participants.

- support multiple simultaneous control models for the same content property and/or property portion. This allows, for example, for concurrent business activities which are dependent on electronic commercial product content distribution, such as acquiring detailed market survey information and/or supporting advertising, both of which can increase revenue and result in lower content costs to users and greater value to content providers. Such control information and/or overall control models may be applied, as determined or allowed by control information, in differing manners to different participants in a pathway of content, reporting, payment, and/or related control information handling. VDE supports applying different content control information to the same and/or different content and/or appliance usage related activities, and/or to different parties in a content and/or appliance usage model, such that different parties (or classes of VDE users, for example) are subject to differing control information managing their use of electronic information content. For example, differing control models based on the category of a user as a distributor of a VDE controlled content

object or an end-user of such content may result in different budgets being applied. Alternatively, for example, a one distributor may have the right to distribute a different array of properties than another distributor (from a common content collection provided, for example, on optical disc). An individual, and/or a class or other grouping of end-users, may have different costs (for example, a student, senior citizen, and/or poor citizen user of content who may be provided with the same or differing discounts) than a "typical" content user.

- support provider revenue information resulting from customer use of content and/or appliances, and/or provider and/or end-user payment of taxes, through the transfer of credit and/or electronic currency from said end-user and/or provider to a government agency, might occur "automatically" as a result of such received control information causing the generation of a VDE content container whose content includes customer content usage information reflecting secure, trusted revenue summary information and/or detailed user transaction listings (level of detail might depend, for example on type or

size of transaction—information regarding a bank interest payment to a customer or a transfer of a large (e.g. over $10,000) might be, by law, automatically reported to the government). Such summary and/or detailed information related to taxable events and/or currency, and/or creditor currency transfer, may be passed along a pathway of reporting and/or payment to the government in a VDE container. Such a container may also be used for other VDE related content usage reporting information.

- support the flowing of content control information through different "branches" of content control information handling so as to accommodate, under the present invention's preferred embodiment, diverse controlled distributions of VDE controlled content. This allows different parties to employ the same initial electronic content with differing (perhaps competitive) control strategies. In this instance, a party who first placed control information on content can make certain control assumptions and these assumptions would evolve into more specific and/or extensive control

assumptions. These control assumptions can evolve during the branching sequence upon content model participants submitting control information changes, for example, for use in "negotiating" with "in place" content control information. This can result in new or modified content control information and/or it might involve the selection of certain one or more already "in-place" content usage control methods over in-place alternative methods, as well as the submission of relevant control information parameter data. This form of evolution of different control information sets applied to different copies of the same electronic property content and/or appliance results from VDE control information flowing "down" through different branches in an overall pathway of handling and control and being modified differently as it diverges down these different pathway branches. This ability of the present invention to support multiple pathway branches for the flow of both VDE content control information and VDE managed content enables an electronic commerce marketplace which supports diverging, competitive business partnerships, agreements, and evolving overall business models

which can employ the same content properties combined, for example, in differing collections of content representing differing at least in part competitive products.

- enable a user to securely extract, through the use of the secure subsystem at the user's VDE installation, at least a portion of the content included within a VDE content container to produce a new, secure object (content container), such that the extracted information is maintained in a continually secure manner through the extraction process. Formation of the new VDE container containing such extracted content shall result in control information consistent with, or specified by, the source VDE content container, and/or local VDE installation secure subsystem as appropriate, content control information. Relevant control information, such as security and administrative information, derived, at least in part, from the parent (source) object's control information, will normally be automatically inserted into a new VDE content container object containing extracted VDE content. This process typically occurs under the control framework of a

parent object and/or VDE installation control information executing at the user's VDE installation secure subsystem (with, for example, at least a portion of this inserted control information being stored securely in encrypted form in one or more permissions records). In an alternative embodiment, the derived content control information applied to extracted content may be in part or whole derived from, or employ, content control information stored remotely from the VDE installation that performed the secure extraction such as at a remote server location. As with the content control information for most VDE managed content, features of the present invention allows the content's control information to:

(a) "evolve," for example, the extractor of content may add new control methods and/or modify control parameter data, such as VDE application compliant methods, to the extent allowed by the content's in-place control information. Such new control information might specify, for example, who may use at least a portion of the new object, and/or how said at least a portion of said extracted

content may be used (e.g. when at least a portion may be used, or what portion or quantity of portions may be used);

(b) allow a user to combine additional content with at least a portion of said extracted content, such as material authored by the extractor and/or content (for example, images, video, audio, and/or text) extracted from one or more other VDE container objects for placement directly into the new container;

(c) allow a user to securely edit at least a portion of said content while maintaining said content in a secure form within said VDE content container;

(d) append extracted content to a pre-existing VDE content container object and attach associated control information -- in these cases, user added information may be secured, e.g., encrypted, in part or as a whole, and may be subject to usage and/or auditing control

information that differs from the those applied to previously in place object content;

(e) preserve VDE control over one or more portions of extracted content after various forms of usage of said portions, for example, maintain content in securely stored form while allowing "temporary" on screen display of content or allowing a software program to be maintained in secure form but transiently decrypt any encrypted executing portion of said program (all, or only a portion, of said program may be encrypted to secure the program).

Generally, the extraction features of the present invention allow users to aggregate and/or disseminate and/or otherwise use protected electronic content information extracted from content container sources while maintaining secure VDE capabilities thus preserving the rights of providers in said content information after various content usage processes.

- support the aggregation of portions of VDE controlled content, such portions being subject to differing VDE content container control information, wherein various of said portions may have been provided by independent, different content providers from one or more different locations remote to the user performing the aggregation. Such aggregation, in the preferred embodiment of the present invention, may involve preserving at least a portion of the control information (e.g., executable code such as load modules) for each of various of said portions by, for example, embedding some or all of such portions individually as VDE content container objects within an overall VDE content container and/or embedding some or all of such portions directly into a VDE content container. In the latter case, content control information of said content container may apply differing control information sets to various of such portions based upon said portions original control information requirements before aggregation. Each of such embedded VDE content containers may have its own control information in the form of one or more permissions records. Alternatively, a negotiation between

control information associated with various aggregated portions of electronic content, may produce a control information set that would govern some or all of the aggregated content portions. The VDE content control information produced by the negotiation may be uniform (such as having the same load modules and/or component assemblies, and/or it may apply differing such content control information to two or more portions that constitute an aggregation of VDE controlled content such as differing metering, budgeting, billing and/or payment models. For example, content usage payment may be automatically made, either through a clearinghouse, or directly, to different content providers for different potions.

- enable flexible metering of, or other collection of information related to, use of electronic content and/or electronic appliances. A feature of the present invention enables such flexibility of metering control mechanisms to accommodate a simultaneous, broad array of: (a) different parameters related to electronic information content use; (b) different increment units (bytes, documents,

properties, paragraphs, images, etc.) and/or other organizations of such electronic content; and/or (c) different categories of user and/or VDE installation types, such as client organizations, departments, projects, networks, and/or individual users, etc. This feature of the present invention can be employed for content security, usage analysis (for example, market surveying), and/or compensation based upon the use and/or exposure to VDE managed content. Such metering is a flexible basis for ensuring payment for content royalties, licensing, purchasing, and/or advertising. A feature of the present invention provides for payment means supporting flexible electronic currency and credit mechanisms, including the ability to securely maintain audit trails reflecting information related to use of such currency or credit. VDE supports multiple differing hierarchies of client organization control information wherein an organization client administrator distributes control information specifying the usage rights of departments, users, and/or projects. Likewise, a department (division) network manager can function as a distributor

(budgets, access rights, etc.) for department networks, projects, and/or users, etc.

- provide scalable, integratable, standardized control means for use on electronic appliances ranging from inexpensive consumer (for example, television set-top appliances) and professional devices (and hand-held PDAs) to servers, mainframes, communication switches, etc. The scalable transaction management/auditing technology of the present invention will result in more efficient and reliable interoperability amongst devices functioning in electronic commerce and/or data security environments. As standardized physical containers have become essential to the shipping of physical goods around the world, allowing these physical containers to universally "fit" unloading equipment, efficiently use truck and train space, and accommodate known arrays of objects (for example, boxes) in an efficient manner, so VDE electronic content containers may, as provided by the present invention, be able to efficiently move electronic information content (such as commercially published properties, electronic currency and credit, and

content audit information), and associated content control information, around the world. Interoperability is fundamental to efficient electronic commerce. The design of the VDE foundation, VDE load modules, and VDE containers, are important features that enable the VDE node operating environment to be compatible with a very broad range of electronic appliances. The ability, for example, for control methods based on load modules to execute in very "small" and inexpensive secure sub-system environments, such as environments with very little read/write memory, while also being able to execute in large memory sub-systems that may be used in more expensive electronic appliances, supports consistency across many machines. This consistent VDE operating environment, including its control structures and container architecture, enables the use of standardized VDE content containers across a broad range of device types and host operating environments. Since VDE capabilities can be seamlessly integrated as extensions, additions, and/or modifications to fundamental capabilities of electronic appliances and host operating systems,

VDE containers, content control information, and the VDE foundation will be able to work with many device types and these device types will be able to consistently and efficiently interpret and enforce VDE control information. Through this integration users can also benefit from a transparent interaction with many of the capabilities of VDE. VDE integration with software operating on a host electronic appliance supports a variety of capabilities that would be unavailable or less secure without such integration. Through integration with one or more device applications and/or device operating environments, many capabilities of the present invention can be presented as inherent capabilities of a given electronic appliance, operating system, or appliance application. For example, features of the present invention include: (a) VDE system software to in part extend and/or modify host operating systems such that they possesses VDE capabilities, such as enabling secure transaction processing and electronic information storage; (b) one or more application programs that in part represent tools associated with VDE operation; and/or (c) code to be integrated into application

programs, wherein such code incorporates references into VDE system software to integrate VDE capabilities and makes such applications VDE aware (for example, word processors, database retrieval applications, spreadsheets, multimedia presentation authoring tools, film editing software, music editing software such as MIDI applications and the like, robotics control systems such as those associated with CAD/CAM environments and NCM software and the like, electronic mail systems, teleconferencing software, and other data authoring, creating, handling, and/or usage applications including combinations of the above). These one or more features (which may also be implemented in firmware or hardware) may be employed in conjunction with a VDE node secure hardware processing capability, such as a microcontroller(s), microprocessor(s), other CPU(s) or other digital processing logic.

- employ audit reconciliation and usage pattern evaluation processes that assess, through certain, normally network based, transaction processing reconciliation and threshold checking activities,

whether certain violations of security of a VDE arrangement have occurred. These processes are performed remote to VDE controlled content end-user VDE locations by assessing, for example, purchases, and/or requests, for electronic properties by a given VDE installation. Applications for such reconciliation activities include assessing whether the quantity of remotely delivered VDE controlled content corresponds to the amount of financial credit and/or electronic currency employed for the use of such content. A trusted organization can acquire information from content providers concerning the cost for content provided to a given VDE installation and/or user and compare this cost for content with the credit and/or electronic currency disbursements for that installation and/or user. Inconsistencies in the amount of content delivered versus the amount of disbursement can prove, and/or indicate, depending on the circumstances, whether the local VDE installation has been, at least to some degree, compromised (for example, certain important system security functions, such as breaking encryption for at least some portion of the secure subsystem and/or VDE controlled content by uncovering one or more

keys). Determining whether irregular patterns (e.g. unusually high demand) of content usage, or requests for delivery of certain kinds of VDE controlled information during a certain time period by one or more VDE installations and/or users (including, for example, groups of related users whose aggregate pattern of usage is suspicious) may also be useful in determining whether security at such one or more installations, and/or by such one or more users, has been compromised, particularly when used in combination with an assessment of electronic credit and/or currency provided to one or more VDE users and/or installations, by some or all of their credit and/or currency suppliers, compared with the disbursements made by such users and/or installations.

- support security techniques that materially increase the time required to "break" a system's integrity. This includes using a collection of techniques that minimizes the damage resulting from comprising some aspect of the security features of the present inventions.

- provide a family of authoring, administrative, reporting, payment, and billing tool user applications that comprise components of the present invention's trusted/secure, universe wide, distributed transaction control and administration system. These components support VDE related: object creation (including placing control information on content), secure object distribution and management (including distribution control information, financial related, and other usage analysis), client internal VDE activities administration and control, security management, user interfaces, payment disbursement, and clearinghouse related functions. These components are designed to support highly secure, uniform, consistent, and standardized: electronic commerce and/or data security pathway(s) of handling, reporting, and/or payment; content control and administration; and human factors (e.g. user interfaces).

- support the operation of a plurality of clearinghouses, including, for example, both financial and user clearinghouse activities, such as

those performed by a client administrator in a large organization to assist in the organization's use of a VDE arrangement, including usage information analysis, and control of VDE activities by individuals and groups of employees such as specifying budgets and the character of usage rights available under VDE for certain groups of and/or individual, client personnel, subject to control information series to control information submitted by the client administrator. At a clearinghouse, one or more VDE installations may operate together with a trusted distributed database environment (which may include concurrent database processing means). A financial clearinghouse normally receives at its location securely delivered content usage information, and user requests (such as requests for further credit, electronic currency, and/or higher credit limit). Reporting of usage information and user requests can be used for supporting electronic currency, billing, payment and credit related activities, and/or for user profile analysis and/or broader market survey analysis and marketing (consolidated) list generation or other information derived, at least in part, from said usage

information. this information can be provided to content providers or other parties, through secure, authenticated encrypted communication to the VDE installation secure subsystems. Clearinghouse processing means would normally be connected to specialized I/O means, which may include high speed telecommunication switching means that may be used for secure communications between a clearinghouse and other VDE pathway participants.

- securely support electronic currency and credit usage control, storage, and communication at, and between, VDE installations. VDE further supports automated passing of electronic currency and/or credit information, including payment tokens (such as in the form of electronic currency or credit) or other payment information, through a pathway of payment, which said pathway may or may not be the same as a pathway for content usage information reporting. Such payment may be placed into a VDE container created automatically by a VDE installation in response to control information stipulating the "withdrawal" of credit or electronic currency from an electronic credit or currency

account based upon an amount owed resulting from usage of VDE controlled electronic content and/or appliances. Payment credit or currency may then be automatically communicated in protected (at least in part encrypted) form through telecommunication of a VDE container to an appropriate party such as a clearinghouse, provider of original property content or appliance, or an agent for such provider (other than a clearinghouse). Payment information may be packaged in said VDE content container with, or without, related content usage information, such as metering information. An aspect of the present invention further enables certain information regarding currency use to be specified as unavailable to certain, some, or all VDE parties ("conditionally" to fully anonymous currency) and/or further can regulate certain content information, such as currency and/or credit use related information (and/or other electronic information usage data) to be available only under certain strict circumstances, such as a court order (which may itself require authorization through the use of a court controlled VDE installation that may be required to securely access "conditionally"

anonymous information). Currency and credit information, under the preferred embodiment of the present invention, is treated as administrative content;

- support fingerprinting (also known as watermarking) for embedding in content such that when content protected under the present invention is released in clear form from a VDE object (displayed, printed, communicated, extracted, and/or saved), information representing the identification of the user and/or VDE installation responsible for transforming the content into clear form is embedded into the released content. Fingerprinting is useful in providing an ability to identify who extracted information in clear form a VDE container, or who made a copy of a VDE object or a portion of its contents. Since the identity of the user and/or other identifying information may be embedded in an obscure or generally concealed manner, in VDE container content and/or control information, potential copyright violators may be deterred from unauthorized extraction or copying. Fingerprinting normally is embedded into

unencrypted electronic content or control information, though it can be embedded into encrypted content and later placed in unencrypted content in a secure VDE installation sub-system as the encrypted content carrying the fingerprinting information is decrypted. Electronic information, such as the content of a VDE container, may be fingerprinted as it leaves a network (such as Internet) location bound for a receiving party. Such repository information may be maintained in unencrypted form prior to communication and be encrypted as it leaves the repository. Fingerprinting would preferably take place as the content leaves the repository, but before the encryption step. Encrypted repository content can be decrypted, for example in a secure VDE sub-system, fingerprint information can be inserted, and then the content can be re-encrypted for transmission. Embedding identification information of the intended recipient user and/or VDE installation into content as it leaves, for example, an Internet repository, would provide important information that would identify or assist in identifying any party that managed to compromise the security of a VDE installation or the

delivered content. If a party produces an authorized clear form copy of VDE controlled content, including making unauthorized copies of an authorized clear form copy, fingerprint information would point back to that individual and/or his or her VDE installation. Such hidden information will act as a strong disincentive that should dissuade a substantial portion of potential content "pirates" from stealing other parties electronic information. Fingerprint information identifying a receiving party and/or VDE installation can be embedded into a VDE object before, or during, decryption, replication, or communication of VDE content objects to receivers. Fingerprinting electronic content before it is encrypted for transfer to a customer or other user provides information that can be very useful for identifying who received certain content which may have then been distributed or made available in unencrypted form. This information would be useful in tracking who may have "broken" the security of a VDE installation and was illegally making certain electronic content available to others. Fingerprinting may provide additional, available

information such as time and/or date of the release (for example extraction) of said content information. Locations for inserting fingerprints may be specified by VDE installation and/or content container control information. This information may specify that certain areas and/or precise locations within properties should be used for fingerprinting, such as one or more certain fields of information or information types. Fingerprinting information may be incorporated into a property by modifying in a normally undetectable way color frequency and/or the brightness of certain image pixels, by slightly modifying certain audio signals as to frequency, by modifying font character formation, etc. Fingerprint information, itself, should be encrypted so as to make it particularly difficult for tampered fingerprints to be interpreted as valid. Variations in fingerprint locations for different copies of the same property; "false" fingerprint information; and multiple copies of fingerprint information within a specific property or other content which copies employ different fingerprinting techniques such as information distribution patterns, frequency and/or brightness manipulation, and encryption related

techniques, are features of the present invention for increasing the difficulty of an unauthorized individual identifying fingerprint locations and erasing and/or modifying fingerprint information.

- provide smart object agents that can carry requests, data, and/or methods, including budgets, authorizations, credit or currency, and content. For example, smart objects may travel to and/or from remote information resource locations and fulfill requests for electronic information content. Smart objects can, for example, be transmitted to a remote location to perform a specified database search on behalf of a user or otherwise "intelligently" search remote one or more repositories of information for user desired information. After identifying desired information at one or more remote locations, by for example, performing one or more database searches, a smart object may return via communication to the user in the form of a secure "return object" containing retrieved information. A user may be charged for the remote retrieving of information, the returning of information to the user's VDE installation, and/or the use of such information. In

the latter case, a user may be charged only for the information in the return object that the user actually uses. Smart objects may have the means to request use of one or more services and/or resources. Services include locating other services and/or resources such as information resources, language or format translation, processing, credit (or additional credit) authorization, etc. Resources include reference databases, networks, high powered or specialized computing resources (the smart object may carry information to another computer to be efficiently processed and then return the information to the sending VDE installation), remote object repositories, etc. Smart objects can make efficient use of remote resources (e.g. centralized databases, super computers, etc.) while providing a secure means for charging users based on information and/or resources actually used.

- support both "translations" of VDE electronic agreements elements into modern language printed agreement elements (such as English language agreements) and translations of electronic rights protection/transaction management modern

language agreement elements to electronic VDE agreement elements. This feature requires maintaining a library of textual language that corresponds to VDE load modules and/or methods and/or component assemblies. As VDE methods are proposed and/or employed for VDE agreements, a listing of textual terms and conditions can be produced by a VDE user application which, in a preferred embodiment, provides phrases, sentences and/or paragraphs that have been stored and correspond to said methods and/or assemblies. This feature preferably employs artificial intelligence capabilities to analyze and automatically determine, and/or assist one or more users to determine, the proper order and relationship between the library elements corresponding to the chosen methods and/or assemblies so as to compose some or all portions of a legal or descriptive document. One or more users, and/or preferably an attorney (if the document a legal, binding agreement), would review the generated document material upon completion and employ such additional textual information and/or editing as necessary to describe non electronic transaction elements of the agreement

and make any other improvements that may be necessary. These features further support employing modern language tools that allow one or more users to make selections from choices and provide answers to questions and to produce a VDE electronic agreement from such a process. This process can be interactive and the VDE agreement formulation process may employ artificial intelligence expert system technology that learns from responses and, where appropriate and based at least in part on said responses, provides further choices and/or questions which "evolves" the desired VDE electronic agreement.

- support the use of multiple VDE secure subsystems in a single VDE installation. Various security and/or performance advantages may be realized by employing a distributed VDE design within a single VDE installation. For example, designing a hardware based VDE secure subsystem into an electronic appliance VDE display device, and designing said subsystem's integration with said display device so that it is as close as possible to the point of display, will increase the security for video

materials by making it materially more difficult to "steal" decrypted video information as it moves from outside to inside the video system. Ideally, for example, a VDE secure hardware module would be in the same physical package as the actual display monitor, such as within the packaging of a video monitor or other display device, and such device would be designed, to the extent commercially practical, to be as tamper resistant as reasonable. As another example, embedding a VDE hardware module into an I/O peripheral may have certain advantages from the standpoint of overall system throughput. If multiple VDE instances are employed within the same VDE installation, these instances will ideally share resources to the extent practical, such as VDE instances storing certain control information and content and/or appliance usage information on the same mass storage device and in the same VDE management database.

- requiring reporting and payment compliance by employing exhaustion of budgets and time ageing of keys. For example, a VDE commercial arrangement and associated content control information may

involve a content provider's content and the use of clearinghouse credit for payment for end-user usage of said content. Control information regarding said arrangement may be delivered to a user's (of said content) VDE installation and/or said financial clearinghouse's VDE installation. Said control information might require said clearinghouse to prepare and telecommunicate to said content provider both content usage based information in a certain form, and content usage payment in the form of electronic credit (such credit might be "owned" by the provider after receipt and used in lieu of the availability or adequacy of electronic currency) and/or electronic currency. This delivery of information and payment may employ trusted VDE installation secure subsystems to securely, and in some embodiments, automatically, provide in the manner specified by said control information, said usage information and payment content. Features of the present invention help ensure that a requirement that a clearinghouse report such usage information and payment content will be observed. For example, if one participant to a VDE electronic agreement fails to observe such information

reporting and/or paying obligation, another participant can stop the delinquent party from successfully participating in VDE activities related to such agreement. For example, if required usage information and payment was not reported as specified by content control information, the "injured" party can fail to provide, through failing to securely communicate from his VDE installation secure subsystem, one or more pieces of secure information necessary for the continuance of one or more critical processes. For example, failure to report information and/or payment from a clearinghouse to a content provider (as well as any security failures or other disturbing irregularities) can result in the content provider not providing key and/or budget refresh information to the clearinghouse, which information can be necessary to authorize use of the clearinghouse's credit for usage of the provider's content and which the clearinghouse would communicate to end-user's during a content usage reporting communication between the clearinghouse and end-user. As another example, a distributor that failed to make payments and/or report usage information to a

content provider might find that their budget for creating permissions records to distribute the content provider's content to users, and/or a security budget limiting one or more other aspect of their use of the provider's content, are not being refreshed by the content provider, once exhausted or timed-out (for example, at a predetermined date). In these and other cases, the offended party might decide not to refresh time ageing keys that had "aged out." Such a use of time aged keys has a similar impact as failing to refresh budgets or time-aged authorizations.

- support smart card implementations of the present invention in the form of portable electronic appliances, including cards that can be employed as secure credit, banking, and/or money cards. A feature of the present invention is the use of portable VDEs as transaction cards at retail and other establishments, wherein such cards can "dock" with an establishment terminal that has a VDE secure sub-system and/or an online connection to a VDE secure and/or otherwise secure and compatible subsystem, such as a "trusted" financial

clearinghouse (e.g., VISA, Mastercard). The VDE card and the terminal (and/or online connection) can securely exchange information related to a transaction, with credit and/or electronic currency being transferred to a merchant and/or clearinghouse and transaction information flowing back to the card. Such a card can be used for transaction activities of all sorts. A docking station, such as a PCMCIA connector on an electronic appliance, such as a personal computer, can receive a consumer's VDE card at home. Such a station/card combination can be used for on-line transactions in the same manner as a VDE installation that is permanently installed in such an electronic appliance. The card can be used as an "electronic wallet" and contain electronic currency as well as credit provided by a clearinghouse. The card can act as a convergence point for financial activities of a consumer regarding many, if not all, merchant, banking, and on-line financial transactions, including supporting home banking activities. A consumer can receive his paycheck and/or investment earnings and/or "authentic" VDE content container secured detailed information on such

receipts, through on-line connections. A user can send digital currency to another party with a VDE arrangement, including giving away such currency. A VDE card can retain details of transactions in a highly secure and database organized fashion so that financially related information is both consolidated and very easily retrieved and/or analyzed. Because of the VDE security, including use of effective encryption, authentication, digital signaturing, and secure database structures, the records contained within a VDE card arrangement may be accepted as valid transaction records for government and/or corporate recordkeeping requirements. In some embodiments of the present invention a VDE card may employ docking station and/or electronic appliance storage means and/or share other VDE arrangement means local to said appliance and/or available across a network, to augment the information storage capacity of the VDE card, by for example, storing dated, and/or archived, backup information. Taxes relating to some or all of an individual's financial activities may be automatically computed based on "authentic" information securely stored and available to said

VDE card. Said information may be stored in said card, in said docking station, in an associated electronic appliance, and/or other device operatively attached thereto, and/or remotely, such as at a remote server site. A card's data, e.g. transaction history, can be backed up to an individual's personal computer or other electronic appliance and such an appliance may have an integrated VDE installation of its own. A current transaction, recent transactions (for redundancy), or all or other selected card data may be backed up to a remote backup repository, such a VDE compatible repository at a financial clearinghouse, during each or periodic docking for a financial transaction and/or information communication such as a user/merchant transaction. Backing up at least the current transaction during a connection with another party's VDE installation (for example a VDE installation that is also on a financial or general purpose electronic network), by posting transaction information to a remote clearinghouse and/or bank, can ensure that sufficient backup is conducted to enable complete reconstruction of VDE card internal information in the event of a card failure or loss.

- support certification processes that ensure authorized interoperability between various VDE installations so as to prevent VDE arrangements and/or installations that unacceptably deviate in specification protocols from other VDE arrangements and/or installations from interoperating in a manner that may introduce security (integrity and/or confidentiality of VDE secured information), process control, and/or software compatibility problems. Certification validates the identity of VDE installations and/or their components, as well as VDE users. Certification data can also serve as information that contributes to determining the decommissioning or other change related to VDE sites.

- support the separation of fundamental transaction control processes through the use of event (triggered) based method control mechanisms. These event methods trigger one or more other VDE methods (which are available to a secure VDE sub-system) and are used to carry out VDE managed transaction related processing. These triggered methods include independently (separably) and

securely processable component billing management methods, budgeting management methods, metering management methods, and related auditing management processes. As a result of this feature of the present invention, independent triggering of metering, auditing, billing, and budgeting methods, the present invention is able to efficiently, concurrently support multiple financial currencies (e.g. dollars, marks, yen) and content related budgets, and/or billing increments as well as very flexible content distribution models.

- support, complete, modular separation of the control structures related to (1) content event triggering, (2) auditing, (3) budgeting (including specifying no right of use or unlimited right of use), (4) billing, and (5) user identity (VDE installation, client name, department, network, and/or user, etc.). The independence of these VDE control structures provides a flexible system which allows plural relationships between two or more of these structures, for example, the ability to associate a financial budget with different event trigger structures (that are put in place to enable

controlling content based on its logical portions). Without such separation between these basic VDE capabilities, it would be more difficult to efficiently maintain separate metering, budgeting, identification, and/or billing activities which involve the same, differing (including overlapping), or entirely different, portions of content for metering, billing, budgeting, and user identification, for example, paying fees associated with usage of content, performing home banking, managing advertising services, etc. VDE modular separation of these basic capabilities supports the programming of plural, "arbitrary" relationships between one or differing content portions (and/or portion units) and budgeting, auditing, and/or billing control information. For example, under VDE, a budget limit of $200 dollars or 300 German Marks a month may be enforced for decryption of a certain database and 2 U.S. Dollars or 3 German Marks may be charged for each record of said database decrypted (depending on user selected currency). Such usage can be metered while an additional audit for user profile purposes can be prepared recording the identity of each filed displayed. Additionally,

further metering can be conducted regarding the number of said database bytes that have been decrypted, and a related security budget may prevent the decrypting of more than 5% of the total bytes of said database per year. The user may also, under VDE (if allowed by senior control information), collect audit information reflecting usage of database fields by different individuals and client organization departments and ensure that differing rights of access and differing budgets limiting database usage can be applied to these client individuals and groups. Enabling content providers and users to practically employ such diverse sets of user identification, metering, budgeting, and billing control information results, in part, from the use of such independent control capabilities. As a result, VDE can support great configurability in creation of plural control models applied to the same electronic property and the same and/or plural control models applied to differing or entirely different content models (for example, home banking versus electronic shopping).

Methods, Other Control Information, and VDE Objects

VDE control information (e.g., methods) that collectively control use of VDE managed properties (database, document, individual commercial product), are either shipped with the content itself (for example, in a content container) and/or one or more portions of such control information is shipped to distributors and/or other users in separably deliverable "administrative objects." A subset of the methods for a property may in part be delivered with each property while one or more other subsets of methods can be delivered separately to a user or otherwise made available for use (such as being available remotely by telecommunication means). Required methods (methods listed as required for property and/or appliance use) must be available as specified if VDE controlled content (such as intellectual property distributed within a VDE content container) is to be used. Methods that control content may apply to a plurality of VDE container objects, such as a class or other grouping of such objects. Methods may also be required by certain users or classes of users and/or VDE installations and/or classes of installations for such parties to use one or more specific, or classes of, objects.

A feature of VDE provided by the present invention is that certain one or more methods can be specified as required in order

for a VDE installation and/or user to be able to use certain and/or all content. For example, a distributor of a certain type of content might be allowed by "senior" participants (by content creators, for example) to require a method which prohibits end-users from electronically saving decrypted content, a provider of credit for VDE transactions might require an audit method that records the time of an electronic purchase, and/or a user might require a method that summarizes usage information for reporting to a clearinghouse (e.g. billing information) in a way that does not convey confidential, personal information regarding detailed usage behavior.

A further feature of VDE provided by the present invention is that creators, distributors, and users of content can select from among a set of predefined methods (if available) to control container content usage and distribution functions and/or they may have the right to provide new customized methods to control at least certain usage functions (such "new" methods may be required to be certified for trustedness and interoperability to the VDE installation and/or for of a group of VDE applications). As a result, VDE provides a very high degree of configurability with respect to how the distribution and other usage of each property or object (or one or more portions of objects or properties as desired and/or applicable) will be controlled. Each VDE

participant in a VDE pathway of content control information may set methods for some or all of the content in a VDE container, so long as such control information does not conflict with senior control information already in place with respect to:

(1) certain or all VDE managed content,

(2) certain one or more VDE users and/or groupings of users,

(3) certain one or more VDE nodes and/or groupings of nodes, and/or

(4) certain one or more VDE applications and/or arrangements.

For example, a content creator's VDE control information for certain content can take precedence over other submitted VDE participant control information and, for example, if allowed by senior control information, a content distributor's control information may itself take precedence over a client administrator's control information, which may take precedence over an end-user's control information. A path of distribution participant's ability to set such electronic content control

information can be limited to certain control information (for example, method mediating data such as pricing and/or sales dates) or it may be limited only to the extent that one or more of the participant's proposed control information conflicts with control information set by senior control information submitted previously by participants in a chain of handling of the property, or managed in said participant's VDE secure subsystem.

VDE control information may, in part or in full, (a) represent control information directly put in place by VDE content control information pathway participants, and/or (b) comprise control information put in place by such a participant on behalf of a party who does not directly handle electronic content (or electronic appliance) permissions records information (for example control information inserted by a participant on behalf of a financial clearinghouse or government agency). Such control information methods (and/or load modules and/or mediating data and/or component assemblies) may also be put in place by either an electronic automated, or a semi-automated and human assisted, control information (control set) negotiating process that assesses whether the use of one or more pieces of submitted control information will be integrated into and/or replace existing control information (and/or chooses between alternative control information based upon interaction with

in-place control information) and how such control information may be used.

Control information may be provided by a party who does not directly participate in the handling of electronic content (and/or appliance) and/or control information for such content (and/or appliance). Such control information may be provided in secure form using VDE installation secure sub-system managed communications (including, for example, authenticating the deliverer of at least in part encrypted control information) between such not directly participating one or more parties' VDE installation secure subsystems, and a pathway of VDE content control information participant's VDE installation secure subsystem. This control information may relate to, for example, the right to access credit supplied by a financial services provider, the enforcement of regulations or laws enacted by a government agency, or the requirements of a customer of VDE managed content usage information (reflecting usage of content by one or more parties other than such customer) relating to the creation, handling and/or manner of reporting of usage information received by such customer. Such control information may, for example, enforce societal requirements such as laws related to electronic commerce.

VDE content control information may apply differently to different pathway of content and/or control information handling participants. Furthermore, permissions records rights may be added, altered, and/or removed by a VDE participant if they are allowed to take such action. Rights of VDE participants may be defined in relation to specific parties and/or categories of parties and/or other groups of parties in a chain of handling of content and/or content control information (e.g., permissions records). Modifications to control information that may be made by a given, eligible party or parties, may be limited in the number of modifications, and/or degree of modification, they may make.

At least one secure subsystem in electronic appliances of creators, distributors, auditors, clearinghouses, client administrators, and end-users (understanding that two or more of the above classifications may describe a single user) provides a "sufficiently" secure (for the intended applications) environment for:

1. Decrypting properties and control information;

2. Storing control and metering related information;

3. Managing communications;

4. Processing core control programs, along with associated data, that constitute control information for electronic content and/or appliance rights protection, including the enforcing of preferences and requirements of VDE participants.

Normally, most usage, audit, reporting, payment, and distribution control methods are themselves at least in part encrypted and are executed by the secure subsystem of a VDE installation. Thus, for example, billing and metering records can be securely generated and updated, and encryption and decryption keys are securely utilized, within a secure subsystem. Since VDE also employs secure (e.g. encrypted and authenticated) communications when passing information between the participant location (nodes) secure subsystems of a VDE arrangement, important components of a VDE electronic agreement can be reliably enforced with sufficient security (sufficiently trusted) for the intended commercial purposes. A VDE electronic agreement for a value chain can be composed, at least in part, of one or more subagreements between one or more subsets of the value chain participants. These subagreements are comprised of one or more electronic contract "compliance" elements (methods including associated parameter data) that ensure the protection of the rights of VDE participants.

The degree of trustedness of a VDE arrangement will be primarily based on whether hardware SPUs are employed at participant location secure subsystems and the effectiveness of the SPU hardware security architecture, software security techniques when an SPU is emulated in software, and the encryption algorithm(s) and keys that are employed for securing content, control information, communications, and access to VDE node (VDE installation) secure subsystems. Physical facility and user identity authentication security procedures may be used instead of hardware SPUs at certain nodes, such as at an established financial clearinghouse, where such procedures may provide sufficient security for trusted interoperability with a VDE arrangement employing hardware SPUs at user nodes.

The updating of property management files at each location of a VDE arrangement, to accommodate new or modified control information, is performed in the VDE secure subsystem and under the control of secure management file updating programs executed by the protected subsystem. Since all secure communications are at least in part encrypted and the processing inside the secure subsystem is concealed from outside observation and interference, the present invention ensures that content control information can be enforced. As a result, the creator and/or distributor and/or client administrator and/or

other contributor of secure control information for each property (for example, an end-user restricting the kind of audit information he or she will allow to be reported and/or a financial clearinghouse establishing certain criteria for use of its credit for payment for use of distributed content) can be confident that their contributed and accepted control information will be enforced (within the security limitations of a given VDE security implementation design). This control information can determine, for example:

(1) How and/or to whom electronic content can be provided, for example, how an electronic property can be distributed;

(2) How one or more objects and/or properties, or portions of an object or property, can be directly used, such as decrypted, displayed, printed, etc;

(3) How payment for usage of such content and/or content portions may or must be handled; and

(4) How audit information about usage information related to at least a portion of a property should be collected, reported, and/or used.

Seniority of contributed control information, including resolution of conflicts between content control information submitted by multiple parties, is normally established by:

(1) the sequence in which control information is put in place by various parties (in place control information normally takes precedence over subsequently submitted control information),

(2) the specifics of VDE content and/or appliance control information. For example, in-place control information can stipulate which subsequent one or more piece of control from one or more parties or class of parties will take precedence over control information submitted by one or more yet different parties and/or classes of parties, and/or

(3) negotiation between control information sets from plural parties, which negotiation establishes what control information shall constitute the resulting control information set for a given piece of VDE managed content and/or VDE installation.

Electronic Agreements and Rights Protection

An important feature of VDE is that it can be used to assure the administration of, and adequacy of security and rights protection for, electronic agreements implemented through the use of the present invention. Such agreements may involve one or more of:

(1) creators, publishers, and other distributors, of electronic information,

(2) financial service (e.g. credit) providers,

(3) users of (other than financial service providers) information arising from content usage such as content specific demographic information and user specific descriptive information. Such users may include market analysts, marketing list compilers for direct and directed marketing, and government agencies,

(4) end users of content,

(5) infrastructure service and device providers such as telecommunication companies and hardware

manufacturers (semiconductor and electronic appliance and/or other computer system manufacturers) who receive compensation based upon the use of their services and/or devices, and

(6) certain parties described by electronic information.

VDE supports commercially secure "extended" value chain electronic agreements. VDE can be configured to support the various underlying agreements between parties that comprise this extended agreement. These agreements can define important electronic commerce considerations including:

(1) security,

(2) content use control, including electronic distribution,

(3) privacy (regarding, for example, information concerning parties described by medical, credit, tax, personal, and/or of other forms of confidential information),

(4) management of financial processes, and

(5) pathways of handling for electronic content, content and/or appliance control information, electronic content and/or appliance usage information and payment and/or credit.

VDE agreements may define the electronic commerce relationship of two or more parties of a value chain, but such agreements may, at times, not directly obligate or otherwise directly involve other VDE value chain participants. For example, an electronic agreement between a content creator and a distributor may establish both the price to the distributor for a creator's content (such as for a property distributed in a VDE container object) and the number of copies of this object that this distributor may distribute to end-users over a given period of time. In a second agreement, a value chain end-user may be involved in a three party agreement in which the end-user agrees to certain requirements for using the distributed product such as accepting distributor charges for content use and agreeing to observe the copyright rights of the creator. A third agreement might exist between the distributor and a financial clearinghouse that allows the distributor to employ the clearinghouse's credit for payment for the product if the end-user has a separate (fourth) agreement directly with the clearinghouse extending credit to the end-user. A fifth, evolving

agreement may develop between all value chain participants as content control information passes along its chain of handling. This evolving agreement can establish the rights of all parties to content usage information, including, for example, the nature of information to be received by each party and the pathway of handling of content usage information and related procedures. A sixth agreement in this example, may involve all parties to the agreement and establishes certain general assumptions, such as security techniques and degree of trustedness (for example, commercial integrity of the system may require each VDE installation secure subsystem to electronically warrant that their VDE node meets certain interoperability requirements). In the above example, these six agreements could comprise agreements of an extended agreement for this commercial value chain instance.

VDE agreements support evolving ("living") electronic agreement arrangements that can be modified by current and/or new participants through very simple to sophisticated "negotiations" between newly proposed content control information interacting with control information already in place and/or by negotiation between concurrently proposed content control information submitted by a plurality of parties. A given model may be asynchronously and progressively modified over

time in accordance with existing senior rules and such modification may be applied to all, to classes of, and/or to specific content, and/or to classes and/or specific users and/or user nodes. A given piece of content may be subject to different control information at different times or places of handling, depending on the evolution of its content control information (and/or on differing, applicable VDE installation content control information). The evolution of control information can occur during the passing along of one or more VDE control information containing objects, that is control information may be modified at one or more points along a chain of control information handling, so long as such modification is allowed. As a result, VDE managed content may have different control information applied at both different "locations" in a chain of content handling and at similar locations in differing chains of the handling of such content. Such different application of control information may also result from content control information specifying that a certain party or group of parties shall be subject to content control information that differs from another party or group of parties. For example, content control information for a given piece of content may be stipulated as senior information and therefore not changeable, might be put in place by a content creator and might stipulate that national distributors of a given piece of their content may be permitted to make 100,000 copies

per calendar quarter, so long as such copies are provided to boni fide end-users, but may pass only a single copy of such content to a local retailers and the control information limits such a retailer to making no more than 1,000 copies per month for retail sales to end-users. In addition, for example, an end-user of such content might be limited by the same content control information to making three copies of such content, one for each of three different computers he or she uses (one desktop computer at work, one for a desktop computer at home, and one for a portable computer).

Electronic agreements supported by the preferred embodiment of the present invention can vary from very simple to very elaborate. They can support widely diverse information management models that provide for electronic information security, usage administration, and communication and may support:

(a) secure electronic distribution of information, for example commercial literary properties,

(b) secure electronic information usage monitoring and reporting,

(c) secure financial transaction capabilities related to both electronic information and/or appliance usage and other electronic credit and/or currency usage and administration capabilities,

(d) privacy protection for usage information a user does not wish to release, and

(e) "living" electronic information content dissemination models that flexibly accommodate:

(1) a breadth of participants,

(2) one or more pathways (chains) for: the handling of content, content and/or appliance control information, reporting of content and/or appliance usage related information, and/or payment,

(3) supporting an evolution of terms and conditions incorporated into content control information, including use of electronic negotiation capabilities,

(4) support the combination of multiple pieces of content to form new content aggregations, and

(5) multiple concurrent models.

Secure Processing Units

An important part of VDE provided by the present invention is the core secure transaction control arrangement, herein called an SPU (or SPUs), that typically must be present in each user's computer, other electronic appliance, or network. SPUs provide a trusted environment for generating decryption keys, encrypting and decrypting information, managing the secure communication of keys and other information between electronic appliances (i.e. between VDE installations and/or between plural VDE instances within a single VDE installation), securely accumulating and managing audit trail, reporting, and budget information in secure and/or non-secure non-volatile memory, maintaining a secure database of control information management instructions, and providing a secure environment for performing certain other control and administrative functions.

A hardware SPU (rather than a software emulation) within a VDE node is necessary if a highly trusted environment

for performing certain VDE activities is required. Such a trusted environment may be created through the use of certain control software, one or more tamper resistant hardware modules such as a semiconductor or semiconductor chipset (including, for example, a tamper resistant hardware electronic appliance peripheral device), for use within, and/or operatively connected to, an electronic appliance. With the present invention, the trustedness of a hardware SPU can be enhanced by enclosing some or all of its hardware elements within tamper resistant packaging and/or by employing other tamper resisting techniques (e.g. microfusing and/or thin wire detection techniques). A trusted environment of the present invention implemented, in part, through the use of tamper resistant semiconductor design, contains control logic, such as a microprocessor, that securely executes VDE processes.

A VDE node's hardware SPU is a core component of a VDE secure subsystem and may employ some or all of an electronic appliance's primary control logic, such as a microcontroller, microcomputer or other CPU arrangement. This primary control logic may be otherwise employed for non VDE purposes such as the control of some or all of an electronic appliance's non-VDE functions. When operating in a hardware SPU mode, said primary control logic must be sufficiently secure so as to protect

and conceal important VDE processes. For example, a hardware SPU may employ a host electronic appliance microcomputer operating in protected mode while performing VDE related activities, thus allowing portions of VDE processes to execute with a certain degree of security. This alternate embodiment is in contrast to the preferred embodiment wherein a trusted environment is created using a combination of one or more tamper resistant semiconductors that are not part of said primary control logic. In either embodiment, certain control information (software and parameter data) must be securely maintained within the SPU, and further control information can be stored externally and securely (e.g. in encrypted and tagged form) and loaded into said hardware SPU when needed. In many cases, and in particular with microcomputers, the preferred embodiment approach of employing special purpose secure hardware for executing said VDE processes, rather than using said primary control logic, may be more secure and efficient. The level of security and tamper resistance required for trusted SPU hardware processes depends on the commercial requirements of particular markets or market niches, and may vary widely.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages provided by the present invention(s) may be better and more completely understood by referring to the following detailed description of presently preferred example embodiments in connection with the drawings, of which:

FIGURE 1 illustrates an example of a "Virtual Distribution Environment" provided in accordance with a preferred example/embodiment of this invention;

FIGURE 1A is a more detailed illustration of an example of the "Information Utility" shown in FIGURE 1;

FIGURE 2 illustrates an example of a chain of handling and control;

FIGURE 2A illustrates one example of how rules and control information may persist from one participant to another in the Figure 2 chain of handling and control;

FIGURE 3 shows one example of different control information that may be provided;

FIGURE 4 illustrates examples of some different types of rules and/or control information;

FIGURES 5A and 5B show an example of an "object";

FIGURE 6 shows an example of a Secure Processing Unit ("SPU");

FIGURE 7 shows an example of an electronic appliance;

FIGURE 8 is a more detailed block diagram of an example of the electronic appliance shown in FIGURE 7;

FIGURE 9 is a detailed view of an example of the Secure Processing Unit (SPU) shown in FIGURES 6 and 8;

Figure 9A shows an example combined secure processing unit and control processing unit;

Figure 9B shows an example secure processing unit integrated with a standard CPU;

FIGURE 10 shows an example of a "Rights Operating System" ("ROS") architecture provided by the Virtual Distribution Environment;

FIGURES 11A-11C show examples of functional relationship(s) between applications and the Rights Operating System;

FIGURES 11D-11J show examples of "components" and "component assemblies";

FIGURE 12 is a more detailed diagram of an example of the Rights Operating System shown in FIGURE 10;

FIGURE 12A shows an example of how "objects" can be created;

FIGURE 13 is a detailed block diagram of an example the software architecture for a "protected processing environment" shown in FIGURE 12;

FIGURES 14A-14C are examples of SPU memory maps provided by the protected processing environment shown in FIGURE 13;

FIGURE 15 illustrates an example of how the channel services manager and load module execution manager of FIGURE 13 can support a channel;

FIGURE 15A is an example of a channel header and channel detail records shown in FIGURE 15;

FIGURE 15B is a flowchart of an example of program control steps that may be performed by the FIGURE 13 protected processing environment to create a channel;

FIGURE 16 is a block diagram of an example of a secure data base structure;

FIGURE 17 is an illustration of an example of a logical object structure;

FIGURE 18 shows an example of a stationary object structure;

FIGURE 19 shows an example of a traveling object structure;

FIGURE 20 shows an example of a content object structure;

FIGURE 21 shows an example of an administrative object structure;

FIGURE 22 shows an example of a method core structure;

FIGURE 23 shows an example of a load module structure;

FIGURE 24 shows an example of a User Data Element (UDE) and/or Method Data Element (MDE) structure;

FIGURES 25A-25C show examples of "map meters";

FIGURE 26 shows an example of a permissions record (PERC) structure;

FIGURES 26A and 26B together show a more detailed example of a permissions record structure;

FIGURE 27 shows an example of a shipping table structure;

FIGURE 28 shows an example of a receiving table structure;

FIGURE 29 shows an example of an administrative event log structure;

FIGURE 30 shows an example inter-relationship between and use of the object registration table, subject table and user rights table shown in the FIGURE 16 secure database;

FIGURE 31 is a more detailed example of an object registration table shown in FIGURE 16;

FIGURE 32 is a more detailed example of subject table shown in FIGURE 16;

FIGURE 33 is a more detailed example of a user rights table shown in FIGURE 16;

FIGURE 34 shows a specific example of how a site record table and group record table may track portions of the secure database shown in FIGURE 16;

FIGURE 34A is an example of a FIGURE 34 site record table structure;

FIGURE 34B is an example of a FIGURE 34 group record table structure;

FIGURE 35 shows an example of a process for updating the secure database;

FIGURE 36 shows an example of how new elements may be inserted into the FIGURE 16 secure data base;

FIGURE 37 shows an example of how an element of the secure database may be accessed;

FIGURE 38 is a flowchart example of how to protect a secure database element;

FIGURE 39 is a flowchart example of how to back up a secure database;

FIGURE 40 is a flowchart example of how to recover a secure database from a backup;

FIGURES 41A-41D are a set of examples showing how a "chain of handling and control" may be enabled using "reciprocal methods";

FIGURES 42A-42D show an example of a "reciprocal" BUDGET method;

FIGURES 43A-43D show an example of a "reciprocal" REGISTER method;

FIGURES 44A-44C show an example of a "reciprocal" AUDIT method;

FIGURES 45-48 show examples of several methods being used together to control release of content or other information;

FIGURES 49, 49A-49F show an example OPEN method;

FIGURES 50, 50A-50F show an example of a READ method;

FIGURES 51, 51A-51F show an example of a WRITE method;

FIGURE 52 shows an example of a CLOSE method;

FIGURES 53A-53B show an example of an EVENT method;

FIGURE 53C shows an example of a BILLING method;

FIGURE 54 shows an example of an ACCESS method;

FIGURES 55A-55B show examples of DECRYPT and ENCRYPT methods;

FIGURE 56 shows an example of a CONTENT method;

FIGURES 57A and 57B show examples of EXTRACT and EMBED methods;

FIGURE 58A shows an example of an OBSCURE method;

FIGURES 58B, 58C show examples of a FINGERPRINT method;

FIGURE 59 shows an example of a DESTROY method;

FIGURE 60 shows an example of a PANIC method;

FIGURE 61 shows an example of a METER method;

FIGURE 62 shows an example of a key "convolution" process;

FIGURE 63 shows an example of how different keys may be generated using a key convolution process to determine a "true" key;

FIGURES 64 and 65 show an example of how protected processing environment keys may be initialized;

FIGURES 66 and 67 show example processes for decrypting information contained within stationary and traveling objects, respectively;

Figures 67A and 67B show example techniques for cracking a software-based protected processing environment;

FIGURE 68 shows an example of how a protected processing environment may be initialized;

FIGURE 69 shows an example of how firmware may be downloaded into a protected processing environment;

Figure 69A shows an example technique for distributing protected processing environment software;

Figure 69B-69C show an example installation routine for installing a software-based protected processing environment;

Figure 69D shows example techniques for embedding cryptographic keys at random locations within structure-based protected processing environment operational materials;

Figure 69E shows example locations for PPE operational materials random modifications and/or digital fingerprints;

Figure 69F shows an example customized static storage layout for PPE operational materials;

Figure 69G shows example electronic appliance signature locations;

Figure 69H shows example sequence dependent and independent processes;

Figures 69I and 69J show example static code and data storage organizations;

Figures 69K-69L together show example steps for providing dynamic protection mechanisms;

Figure 69M shows an example initialization time check routine;

Figure 69N shows an example time check routine;

Figure 69O shows example time check data structures;

FIGURE 70 shows an example of multiple VDE electronic appliances connected together with a network or other communications means;

Figure 70A shows how content may be prepared for printing and encrypted inside a PPE, then decrypted inside a printer;

Figure 70B shows how characters may be selected from slightly different fonts in order to place an electronic fingerprint or watermark into printed output;

Figure 70C shows how characters in a font may be permuted to render a printed page unusable without the corresponding scrambled font;

FIGURE 71 shows an example of a portable VDE electronic appliance;

FIGURES 72A-72D show examples of "pop-up" displays that may be generated by the user notification and exception interface;

FIGURE 73 shows an example of a "smart object";

FIGURE 74 shows an example of a process using "smart objects";

FIGURES 75A-75D show examples of data structures used for electronic negotiation;

FIGURES 75E-75F show example structures relating to an electronic agreement;

FIGURES 76A-76B show examples of electronic negotiation processes;

FIGURE 77 shows a further example of a chain of handling and control;

FIGURE 78 shows an example of a VDE "repository";

FIGURES 79-83 show an example illustrating a chain of handling and control to evolve and transform VDE managed content and control information;

FIGURE 84 shows a further example of a chain of handling and control involving several categories of VDE participants;

FIGURE 85 shows a further example of a chain of distribution and handling within an organization;

Figures 86 and 86A show a further example of a chain of handling and control; and

Figure 87 shows an example of a virtual silicon container model.

MORE DETAILED DESCRIPTION

Figures 1-7 and the discussion below provides an overview of some aspects of features provided by this invention. Following this overview is a more technical "detail description" of example embodiments in accordance with the invention.

Overview

Figure 1 shows a "Virtual Distribution Environment" ("VDE") 100 that may be provided in accordance with this invention. In Figure 1, an information utility 200 connects to communications means 202 such as telephone or cable TV lines for example. Telephone or cable TV lines 202 may be part of an "electronic highway" that carries electronic information from place to place. Lines 202 connect information utility 200 to other people such as for example a consumer 208, an office 210, a video production studio 204, and a publishing house 214. Each of the people connected to information utility 200 may be called a "VDE participant" because they can participate in transactions occurring within the virtual distribution environment 100.

Almost any sort of transaction you can think of can be supported by virtual distribution environment 100. A few of many examples of transactions that can be supported by virtual distribution environment 100 include:

- home banking and electronic payments;
- electronic legal contracts;
- distribution of "content" such as electronic printed matter, video, audio, images and computer programs; and
- secure communication of private information such as medical records and financial information.

Virtual distribution environment 100 is "virtual" because it does not require many of the physical "things" that used to be necessary to protect rights, ensure reliable and predictable distribution, and ensure proper compensation to content creators and distributors. For example, in the past, information was distributed on records or disks that were difficult to copy. In the past, private or secret content was distributed in sealed envelopes or locked briefcases delivered by courier. To ensure appropriate compensation, consumers received goods and services only after they handed cash over to a seller. Although information utility 200 may deliver information by transferring physical "things" such as electronic storage media, the virtual distribution environment 100 facilitates a completely electronic "chain of handling and control."

VDE Flexibility Supports Transactions

Information utility 200 flexibly supports many different kinds of information transactions. Different VDE participants may define and/or participate in different parts of a transaction. Information utility 200 may assist with delivering information about a transaction, or it may be one of the transaction participants.

For example, the video production studio 204 in the upper right-hand corner of Figure 1 may create video/television programs. Video production studio 204 may send these programs over lines 202, or may use other paths such as satellite link 205 and CD ROM delivery service 216. Video production studio 204 can send the programs directly to consumers 206, 208, 210, or it can send the programs to information utility 200 which may store and later send them to the consumers, for example. Consumers 206, 208, 210 are each capable of receiving and using the programs created by video production studio 204—assuming, that is, that the video production studio or information utility 200 has arranged for these consumers to have appropriate "rules and controls" (control information) that give the consumers rights to use the programs.

Even if a consumer has a copy of a video program, she cannot watch or copy the program unless she has "rules and

controls" that authorize use of the program. She can use the program only as permitted by the "rules and controls."

For example, video production studio 204 might release a half-hour exercise video in the hope that as many viewers as possible will view it. The video production studio 204 wishes to receive $2.00 per viewing. Video production studio 204 may, through information utility 200, make the exercise video available in "protected" form to all consumers 206, 208, 210. Video production studio 204 may also provide "rules and controls" for the video. These "rules and controls" may specify for example:

(1) any consumer who has good credit of at least $2.00 based on a credit account with independent financial provider 212 (such as Mastercard or VISA) may watch the video,

(2) virtual distribution environment 100 will "meter" each time a consumer watches the video, and report usage to video production studio 204 from time to time, and

(3) financial provider 212 may electronically collect payment ($2.00) from the credit account of each consumer

who watches the video, and transfer these payments to the video production studio 204.

Information utility 200 allows even a small video production studio to market videos to consumers and receive compensation for its efforts. Moreover, the videos can, with appropriate payment to the video production studio, be made available to other video publishers who may add value and/or act as repackagers or redistributors.

Figure 1 also shows a publishing house 214. Publishing house 214 may act as a distributor for an author 206. The publishing house 214 may distribute rights to use "content" (such as computer software, electronic newspapers, the video produced by publishing house 214, audio, or any other data) to consumers such as office 210. The use rights may be defined by "rules and controls" distributed by publishing house 216. Publishing house 216 may distribute these "rules and controls" with the content, but this is not necessary. Because the content can be used only by consumers that have the appropriate "rules and controls," content and its associated "rules and controls" may be distributed at different times, in different ways, by different VDE participants. The ability of VDE to securely distribute and

enforce "rules and controls" separately from the content they apply to provides great advantages.

Use rights distributed by publishing house 214 may, for example, permit office 210 to make and distribute copies of the content to its employees. Office 210 may act as a redistributor by extending a "chain of handling and control" to its employees. The office 210 may add or modify "rules and controls" (consistent with the "rules and controls" it receives from publishing house 214) to provide office-internal control information and mechanisms. For example, office 210 may set a maximum usage budget for each individual user and/or group within the office, or it may permit only specified employees and/or groups to access certain information.

Figure 1 also shows an information delivery service 216 delivering electronic storage media such as "CD ROM" disks to consumers 206. Even though the electronic storage media themselves are not delivered electronically by information utility 200 over lines 202, they are still part of the virtual distribution environment 100. The electronic storage media may be used to distribute content, "rules and controls," or other information.

Example of What's Inside Information Utility 200

"Information utility" 200 in Figure 1 can be a collection of participants that may act as distributors, financial clearinghouses, and administrators. Figure 1A shows an example of what may be inside one example of information utility 200. Information utility participants 200a-200g could each be an independent organization/business. There can be any number of each of participants 200a-200g. In this example, electronic "switch" 200a connects internal parts of information utility 200 to each other and to outside participants, and may also connect outside participants to one another.

Information utility 200 may include a "transaction processor" 200b that processes transactions (to transfer electronic funds, for example) based on requests from participants and/or report receiver 200e. It may also include a "usage analyst" 200c that analyzes reported usage information. A "report creator" 200d may create reports based on usage for example, and may provide these reports to outside participants and/or to participants within information utility 200. A "report receiver" 200e may receive reports such as usage reports from content users. A "permissioning agent" 200f may distribute "rules and controls" granting usage or distribution permissions based on a profile of a consumer's credit worthiness, for example.

An administrator 200h may provide information that keeps the virtual distribution environment 100 operating properly. A content and message storage 200g may store information for use by participants within or outside of information utility 200.

Example of Distributing Content" Using A Chain of Handling and Control"

As explained above, virtual distribution environment 100 can be used to manage almost any sort of transaction. One type of important transaction that virtual distribution environment 100 may be used to manage is the distribution or communication of "content" or other important information. Figure 2 more abstractly shows a "model" of how the Figure 1 virtual distribution environment 100 may be used to provide a "chain of handling and control" for distributing content. Each of the blocks in Figure 2 may correspond to one or more of the VDE participants shown in Figure 1.

In the Figure 2 example, a VDE content creator 102 creates "content." The content creator 102 may also specify "rules and controls" for distributing the content. These distribution-related "rules and controls" can specify who has permission to distribute the rights to use content, and how many users are allowed to use the content.

Arrow 104 shows the content creator 102 sending the "rules and controls" associated with the content to a VDE rights distributor 106 ("distributor") over an electronic highway 108 (or by some other path such as an optical disk sent by a delivery service such as U. S. mail). The content can be distributed over the same or different path used to send the "rules and controls." The distributor 106 generates her own "rules and controls" that relate to usage of the content. The usage-related "rules and controls" may, for example, specify what a user can and can't do with the content and how much it costs to use the content. These usage-related "rules and controls" must be consistent with the "rules and controls" specified by content creator 102.

Arrow 110 shows the distributor 106 distributing rights to use the content by sending the content's "rules and controls" to a content user 112 such as a consumer. The content user 112 uses the content in accordance with the usage-related "rules and controls."

In this Figure 2 example, information relating to content use is, as shown by arrow 114, reported to a financial clearinghouse 116. Based on this "reporting," the financial clearinghouse 116 may generate a bill and send it to the content user 112 over a "reports and payments" network 118. Arrow 120

shows the content user 112 providing payments for content usage to the financial clearinghouse 116. Based on the reports and payments it receives, the financial clearinghouse 116 may provide reports and/or payments to the distributor 106. The distributor 106 may, as shown by arrow 122, provide reports and/or payments to the content creator 102. The clearinghouse 116 may provide reports and payments directly to the creator 102. Reporting and/or payments may be done differently. For example, clearinghouse 116 may directly or through an agent, provide reports and/or payments to each of VDE content creators 102, and rights distributor 106, as well as reports to content user 112.

The distributor 106 and the content creator 102 may be the same person, or they may be different people. For example, a musical performing group may act as both content creator 102 and distributor 106 by creating and distributing its own musical recordings. As another example, a publishing house may act as a distributor 106 to distribute rights to use works created by an author content creator 102. Content creators 102 may use a distributor 106 to efficiently manage the financial end of content distribution.

The "financial clearinghouse" 116 shown in Figure 2 may also be a "VDE administrator." Financial clearinghouse 116 in its VDE administrator role sends "administrative" information to the VDE participants. This administrative information helps to keep the virtual distribution environment 100 operating properly. The "VDE administrator" and financial clearinghouse roles may be performed by different people or companies, and there can be more than one of each.

More about Rules and Controls"

The virtual distribution environment 100 prevents use of protected information except as permitted by the "rules and controls" (control information). For example, the "rules and controls" shown in Figure 2 may grant specific individuals or classes of content users 112 "permission" to use certain content. They may specify what kinds of content usage are permitted, and what kinds are not. They may specify how content usage is to be paid for and how much it costs. As another example, "rules and controls" may require content usage information to be reported back to the distributor 106 and/or content creator 102.

Every VDE participant in "chain of handling and control" is normally subject to "rules and controls." "Rules and controls" define the respective rights and obligations of each of the various

VDE participants. "Rules and controls" provide information and mechanisms that may establish interdependencies and relationships between the participants. "Rules and controls" are flexible, and permit "virtual distribution environment" 100 to support most "traditional" business transactions. For example:

- "Rules and controls" may specify which financial clearinghouse(s) 116 may process payments,
- "Rules and controls" may specify which participant(s) receive what kind of usage report, and
- "Rules and controls" may specify that certain information is revealed to certain participants, and that other information is kept secret from them.

"Rules and controls" may self limit if and how they may be changed. Often, "rules and controls" specified by one VDE participant cannot be changed by another VDE participant. For example, a content user 112 generally can't change "rules and controls" specified by a distributor 106 that require the user to pay for content usage at a certain rate. "Rules and controls" may "persist" as they pass through a "chain of handling and control," and may be "inherited" as they are passed down from one VDE participant to the next.

Depending upon their needs, VDE participants can specify that their "rules and controls" can be changed under conditions specified by the same or other "rules and controls." For example, "rules and controls" specified by the content creator 102 may permit the distributor 106 to "mark up" the usage price just as retail stores "mark up" the wholesale price of goods. Figure 2A shows an example in which certain "rules and controls" persist unchanged from content creator 102 to content user 112; other "rules and controls" are modified or deleted by distributor 106; and still other "rules and controls" are added by the distributor.

"Rules and controls" can be used to protect the content user's privacy by limiting the information that is reported to other VDE participants. As one example, "rules and controls" can cause content usage information to be reported anonymously without revealing content user identity, or it can reveal only certain information to certain participants (for example, information derived from usage) with appropriate permission, if required. This ability to securely control what information is revealed and what VDE participant(s) it is revealed to allows the privacy rights of all VDE participants to be protected.

Rules and Contents" Can Be Separately Delivered

As mentioned above, virtual distribution environment 100 "associates" content with corresponding "rules and controls," and prevents the content from being used or accessed unless a set of corresponding "rules and controls" is available. The distributor 106 doesn't need to deliver content to control the content's distribution. The preferred embodiment can securely protect content by protecting corresponding, usage enabling "rules and controls" against unauthorized distribution and use.

In some examples, "rules and controls" may travel with the content they apply to. Virtual distribution environment 100 also allows "rules and controls" to be delivered separately from content. Since no one can use or access protected content without "permission" from corresponding "rules and controls," the distributor 106 can control use of content that has already been (or will in the future be) delivered. "Rules and controls" may be delivered over a path different from the one used for content delivery. "Rules and controls" may also be delivered at some other time. The content creator 102 might deliver content to content user 112 over the electronic highway 108, or could make the content available to anyone on the highway. Content may be used at the time it is delivered, or it may be stored for later use or reuse.

The virtual distribution environment 100 also allows payment and reporting means to be delivered separately. For example, the content user 112 may have a virtual "credit card" that extends credit (up to a certain limit) to pay for usage of any content. A "credit transaction" can take place at the user's site without requiring any "online" connection or further authorization. This invention can be used to help securely protect the virtual "credit card" against unauthorized use.

Rules and Contents" Define Processes

Figure 3 shows an example of an overall process based on "rules and controls." It includes an "events" process 402, a meter process 404, a billing process 406, and a budget process 408. Not all of the processes shown in Figure 3 will be used for every set of "rules and controls."

The "events process" 402 detects things that happen ("events") and determines which of those "events" need action by the other "processes." The "events" may include, for example, a request to use content or generate a usage permission. Some events may need additional processing, and others may not. Whether an "event" needs more processing depends on the "rules and controls" corresponding to the content. For example, a user who lacks permission will not have her request satisfied ("No

Go"). As another example, each user request to turn to a new page of an electronic book may be satisfied ("Go"), but it may not be necessary to meter, bill or budget those requests. A user who has purchased a copy of a novel may be permitted to open and read the novel as many times as she wants to without any further metering, billing or budgeting. In this simple example, the "event process" 402 may request metering, billing and/or budgeting processes the first time the user asks to open the protected novel (so the purchase price can be charged to the user), and treat all later requests to open the same novel as "insignificant events." Other content (for example, searching an electronic telephone directory) may require the user to pay a fee for each access.

"Meter" process 404 keeps track of events, and may report usage to distributor 106 and/or other appropriate VDE participant(s). Figure 4 shows that process 404 can be based on a number of different factors such as:

(a) type of usage to charge for,

(b) what kind of unit to base charges on,

(c) how much to charge per unit,

(d) when to report, and

(e) how to pay.

These factors may be specified by the "rules and controls" that control the meter process.

Billing process 406 determines how much to charge for events. It records and reports payment information.

Budget process 408 limits how much content usage is permitted. For example, budget process 408 may limit the number of times content may be accessed or copied, or it may limit the number of pages or other amount of content that can be used based on, for example, the number of dollars available in a credit account. Budget process 408 records and reports financial and other transaction information associated with such limits.

Content may be supplied to the user once these processes have been successfully performed.

Containers and Objects"

Figure 5A shows how the virtual distribution environment 100, in a preferred embodiment, may package information elements (content) into a "container" 302 so the information can't be accessed except as provided by its "rules and controls."

Normally, the container 302 is electronic rather than physical. Electronic container 302 in one example comprises "digital" information having a well defined structure. Container 302 and its contents can be called an "object 300."

The Figure 5A example shows items "within" and enclosed by container 302. However, container 302 may "contain" items without those items actually being stored within the container. For example, the container 302 may reference items that are available elsewhere such as in other containers at remote sites. Container 302 may reference items available at different times or only during limited times. Some items may be too large to store within container 302. Items may, for example, be delivered to the user in the form of a "live feed" of video at a certain time. Even then, the container 302 "contains" the live feed (by reference) in this example.

Container 302 may contain information content 304 in electronic (such as "digital") form. Information content 304 could be the text of a novel, a picture, sound such as a musical performance or a reading, a movie or other video, computer software, or just about any other kind of electronic information you can think of. Other types of "objects" 300 (such as

"administrative objects") may contain "administrative" or other information instead of or in addition to information content 304.

In the Figure 5A example, container 302 may also contain "rules and controls" in the form of:

(a) a "permissions record" 808;

(b) "budgets" 308; and

(c) "other methods" 1000.

Figure 5B gives some additional detail about permissions record 808, budgets 308 and other methods 1000. The "permissions record" 808 specifies the rights associated with the object 300 such as, for example, who can open the container 302, who can use the object's contents, who can distribute the object, and what other control mechanisms must be active. For example, permissions record 808 may specify a user's rights to use, distribute and/or administer the container 302 and its content. Permissions record 808 may also specify requirements to be applied by the budgets 308 and "other methods" 1000. Permissions record 808 may also contain security related information such as scrambling and descrambling "keys."

"Budgets" 308 shown in Figure 5B are a special type of "method" 1000 that may specify, among other things, limitations

on usage of information content 304, and how usage will be paid for. Budgets 308 can specify, for example, how much of the total information content 304 can be used and/or copied. The methods 310 may prevent use of more than the amount specified by a specific budget.

"Other methods" 1000 define basic operations used by "rules and controls." Such "methods" 1000 may include, for example, how usage is to be "metered," if and how content 304 and other information is to be scrambled and descrambled, and other processes associated with handling and controlling information content 304. For example, methods 1000 may record the identity of anyone who opens the electronic container 302, and can also control how information content is to be charged based on "metering." Methods 1000 may apply to one or several different information contents 304 and associated containers 302, as well as to all or specific portions of information content 304.

Secure Processing Unit (SPU)

The "VDE participants" may each have an "electronic appliance." The appliance may be or contain a computer. The appliances may communicate over the electronic highway 108. Figure 6 shows a secure processing unit ("SPU") 500 portion of

the "electronic appliance" used in this example by each VDE participant. SPU 500 processes information in a secure processing environment 503, and stores important information securely. SPU 500 may be emulated by software operating in a host electronic appliance.

SPU 500 is enclosed within and protected by a "tamper resistant security barrier" 502. Security barrier 502 separates the secure environment 503 from the rest of the world. It prevents information and processes within the secure environment 503 from being observed, interfered with and leaving except under appropriate secure conditions. Barrier 502 also controls external access to secure resources, processes and information within SPU 500. In one example, tamper resistant security barrier 502 is formed by security features such as "encryption," and hardware that detects tampering and/or destroys sensitive information within secure environment 503 when tampering is detected.

SPU 500 in this example is an integrated circuit ("IC") "chip" 504 including "hardware" 506 and "firmware" 508. SPU 500 connects to the rest of the electronic appliance through an "appliance link" 510. SPU "firmware" 508 in this example is "software" such as a "computer program(s)" "embedded" within

chip 504. Firmware 508 makes the hardware 506 work. Hardware 506 preferably contains a processor to perform instructions specified by firmware 508. "Hardware" 506 also contains long-term and short-term memories to store information securely so it can't be tampered with. SPU 500 may also have a protected clock/calendar used for timing events. The SPU hardware 506 in this example may include special purpose electronic circuits that are specially designed to perform certain processes (such as "encryption" and "decryption") rapidly and efficiently.

The particular context in which SPU 500 is being used will determine how much processing capabilities SPU 500 should have. SPU hardware 506, in this example, provides at least enough processing capabilities to support the secure parts of processes shown in Figure 3. In some contexts, the functions of SPU 500 may be increased so the SPU can perform all the electronic appliance processing, and can be incorporated into a general purpose processor. In other contexts, SPU 500 may work alongside a general purpose processor, and therefore only needs to have enough processing capabilities to handle secure processes.

VDE Electronic Appliance and Rights Operating System"

Figure 7 shows an example of an electronic appliance 600 including SPU 500. Electronic appliance 600 may be practically any kind of electrical or electronic device, such as:

- a computer
- a T.V. "set top" control box
- a pager
- a telephone
- a sound system
- a video reproduction system
- a video game player
- a "smart" credit card

Electronic appliance 600 in this example may include a keyboard or keypad 612, a voice recognizer 613, and a display 614. A human user can input commands through keyboard 612 and/or voice recognizer 613, and may view information on display 614. Appliance 600 may communicate with the outside world through any of the connections/devices normally used within an electronic appliance. The connections/devices shown along the bottom of the drawing are examples:

a "modem" 618 or other telecommunications link;

a CD ROM disk 620 or other storage medium or device;

a printer 622;

broadcast reception 624;

a document scanner 626; and

a "cable" 628 connecting the appliance with a "network."

Virtual distribution environment 100 provides a "rights operating system" 602 that manages appliance 600 and SPU 500 by controlling their hardware resources. The operating system 602 may also support at least one "application" 608. Generally, "application" 608 is hardware and/or software specific to the context of appliance 600. For example, if appliance 600 is a personal computer, then "application" 608 could be a program loaded by the user, for instance, a word processor, a communications system or a sound recorder. If appliance 600 is a television controller box, then application 608 might be hardware or software that allows a user to order videos on demand and perform other functions such as fast forward and rewind. In this example, operating system 602 provides a standardized, well defined, generalized "interface" that could support and work with many different "applications" 608.

Operating system 602 in this example provides "rights and auditing operating system functions" 604 and "other operating system functions" 606. The "rights and auditing operating

system functions" 604 securely handle tasks that relate to virtual distribution environment 100. SPU 500 provides or supports many of the security functions of the "rights and auditing operating system functions" 402. The "other operating system functions" 606 handle general appliance functions. Overall operating system 602 may be designed from the beginning to include the "rights and auditing operating system functions" 604 plus the "other operating system functions" 606, or the "rights and auditing operating system functions" may be an add-on to a preexisting operating system providing the "other operating system functions."

"Rights operating system" 602 in this example can work with many different types of appliances 600. For example, it can work with large mainframe computers, "minicomputers" and "microcomputers" such as personal computers and portable computing devices. It can also work in control boxes on the top of television sets, small portable "pagers," desktop radios, stereo sound systems, telephones, telephone switches, or any other electronic appliance. This ability to work on big appliances as well as little appliances is called "scalable." A "scalable" operating system 602 means that there can be a standardized interface across many different appliances performing a wide variety of tasks.

The "rights operating system functions" 604 are "services-based" in this example. For example, "rights operating system functions" 604 handle summary requests from application 608 rather than requiring the application to always make more detailed "subrequests" or otherwise get involved with the underlying complexities involved in satisfying a summary request. For example, application 608 may simply ask to read specified information; "rights operating system functions" 604 can then decide whether the desired information is VDE-protected content and, if it is, perform processes needed to make the information available. This feature is called "transparency." "Transparency" makes tasks easy for the application 608. "Rights operating system functions" 604 can support applications 608 that "know" nothing about virtual distribution environment 100. Applications 608 that are "aware" of virtual distribution environment 100 may be able to make more detailed use of virtual distribution environment 100.

In this example, "rights operating system functions" 604 are "event driven". Rather than repeatedly examining the state of electronic appliance 600 to determine whether a condition has arisen, the "rights operating system functions" 604 may respond directly to "events" or "happenings" within appliance 600.

In this example, some of the services performed by "rights operating system functions" 604 may be extended based on additional "components" delivered to operating system 602. "Rights operating system functions" 604 can collect together and use "components" sent by different participants at different times. The "components" help to make the operating system 602 "scalable." Some components can change how services work on little appliances versus how they work on big appliances (e.g., multi-user). Other components are designed to work with specific applications or classes of applications (e.g., some types of meters and some types of budgets).

Electronic Appliance 600

An electronic appliance 600 provided by the preferred embodiment may, for example, be any electronic apparatus that contains one or more microprocessors and/or microcontrollers and/or other devices which perform logical and/or mathematical calculations. This may include computers; computer terminals; device controllers for use with computers; peripheral devices for use with computers; digital display devices; televisions; video and audio/video projection systems; channel selectors and/or decoders for use with broadcast and/or cable transmissions; remote control devices; video and/or audio recorders; media players including compact disc players, videodisc players and

tape players; audio and/or video amplifiers; virtual reality machines; electronic game players; multimedia players; radios; telephones; videophones; facsimile machines; robots; numerically controlled machines including machine tools and the like; and other devices containing one or more microcomputers and/or microcontrollers and/or other CPUs, including those not yet in existence.

Figure 8 shows an example of an electronic appliance 600. This example of electronic appliance 600 includes a system bus 653. In this example, one or more conventional general purpose central processing units ("CPUs") 654 are connected to bus 653. Bus 653 connects CPU(s) 654 to RAM 656, ROM 658, and I/O controller 660. One or more SPUs 500 may also be connected to system bus 653. System bus 653 may permit SPU(s) 500 to communicate with CPU(s) 654, and also may allow both the CPU(s) and the SPU(s) to communicate (e.g., over shared address and data lines) with RAM 656, ROM 658 and I/O controller 660. A power supply 659 may provide power to SPU 500, CPU 654 and the other system components shown.

In the example shown, I/O controller 660 is connected to secondary storage device 652, a keyboard/display 612,614, a communications controller 666, and a backup storage device 668.

Backup storage device 668 may, for example, store information on mass media such as a tape 670, a floppy disk, a removable memory card, etc. Communications controller 666 may allow electronic appliance 600 to communicate with other electronic appliances via network 672 or other telecommunications links. Different electronic appliances 600 may interoperate even if they use different CPUs and different instances of ROS 602, so long as they typically use compatible communication protocols and/or security methods. In this example, I/O controller 660 permits CPU 654 and SPU 500 to read from and write to secondary storage 662, keyboard/display 612, 614, communications controller 666, and backup storage device 668.

Secondary storage 662 may comprise the same one or more non-secure secondary storage devices (such as a magnetic disk and a CD-ROM drive as one example) that electronic appliance 600 uses for general secondary storage functions. In some implementations, part or all of secondary storage 652 may comprise a secondary storage device(s) that is physically enclosed within a secure enclosure. However, since it may not be practical or cost-effective to physically secure secondary storage 652 in many implementations, secondary storage 652 may be used to store information in a secure manner by encrypting information before storing it in secondary storage 652. If information is

encrypted before it is stored, physical access to secondary storage 652 or its contents does not readily reveal or compromise the information.

Secondary storage 652 in this example stores code and data used by CPU 654 and/or SPU 500 to control the overall operation of electronic appliance 600. For example, Figure 8 shows that "Rights Operating System" ("ROS") 602 (including a portion 604 of ROS that provides VDE functions and a portion 606 that provides other OS functions) shown in Figure 7 may be stored on secondary storage 652. Secondary storage 652 may also store one or more VDE objects 300. Figure 8 also shows that the secure files 610 shown in Figure 7 may be stored on secondary storage 652 in the form of a "secure database" or management file system 610. This secure database 610 may store and organize information used by ROS 602 to perform VDE functions 604. Thus, the code that is executed to perform VDE and other OS functions 604, 606, and secure files 610 (as well as VDE objects 300) associated with those functions may be stored in secondary storage 652. Secondary storage 652 may also store "other information" 673 such as, for example, information used by other operating system functions 606 for task management, non-VDE files, etc. Portions of the elements indicated in secondary storage 652 may also be stored in ROM 658, so long as

those elements do not require changes (except when ROM 658 is replaced). Portions of ROS 602 in particular may desirably be included in ROM 658 (e.g., "bootstrap" routines, POST routines, etc. for use in establishing an operating environment for electronic appliance 600 when power is applied).

Figure 8 shows that secondary storage 652 may also be used to store code ("application programs") providing user application(s) 608 shown in Figure 7. Figure 8 shows that there may be two general types of application programs 608: "VDE aware" applications 608a, and Non-VDE aware applications 608b. VDE aware applications 608a may have been at least in part designed specifically with VDE 100 in mind to access and take detailed advantage of VDE functions 604. Because of the "transparency" features of ROS 602, non-VDE aware applications 608b (e.g., applications not specifically designed for VDE 100) can also access and take advantage of VDE functions 604.

SECURE PROCESSING UNIT 500

Each VDE node or other electronic appliance 600 in the preferred embodiment may include one or more SPUs 500. SPUs 500 may be used to perform all secure processing for VDE 100. For example, SPU 500 is used for decrypting (or otherwise

unsecuring) VDE protected objects 300. It is also used for managing encrypted and/or otherwise secured communication (such as by employing authentication and/or error-correction validation of information). SPU 500 may also perform secure data management processes including governing usage of, auditing of, and where appropriate, payment for VDE objects 300 (through the use of prepayments, credits, real-time electronic debits from bank accounts and/or VDE node currency token deposit accounts). SPU 500 may perform other transactions related to such VDE objects 300.

SPU Physical Packaging and Security Barrier 502

As shown Figure 6, in the preferred embodiment, an SPU 500 may be implemented as a single integrated circuit "chip" 505 to provide a secure processing environment in which confidential and/or commercially valuable information can be safely processed, encrypted and/or decrypted. IC chip 505 may, for example, comprise a small semiconductor "die" about the size of a thumbnail. This semiconductor die may include semiconductor and metal conductive pathways. These pathways define the circuitry, and thus the functionality, of SPU 500. Some of these pathways are electrically connected to the external "pins" 504 of the chip 505.

As shown in Figures 6 and 9, SPU 500 may be surrounded by a tamper-resistant hardware security barrier 502. Part of this security barrier 502 is formed by a plastic or other package in which an SPU "die" is encased. Because the processing occurring within, and information stored by, SPU 500 are not easily accessible to the outside world, they are relatively secure from unauthorized access and tampering. All signals cross barrier 502 through a secure, controlled path provided by BIU 530 that restricts the outside world's access to the internal components within SPU 500. This secure, controlled path resists attempts from the outside world to access secret information and resources within SPU 500.

It is possible to remove the plastic package of an IC chip and gain access to the "die." It is also possible to analyze and "reverse engineer" the "die" itself (e.g., using various types of logic analyzers and microprobes to collect and analyze signals on the die while the circuitry is operating, using acid etching or other techniques to remove semiconductor layers to expose other layers, viewing and photographing the die using an electron microscope, etc.) Although no system or circuit is absolutely impervious to such attacks, SPU barrier 502 may include additional hardware protections that make successful attacks exceedingly costly and time consuming. For example, ion

implantation and/or other fabrication techniques may be used to make it very difficult to visually discern SPU die conductive pathways, and SPU internal circuitry may be fabricated in such a way that it "self-destructs" when exposed to air and/or light. SPU 500 may store secret information in internal memory that loses its contents when power is lost. Circuitry may be incorporated within SPU 500 that detects microprobing or other tampering, and self-destructs (or destroys other parts of the SPU) when tampering is detected. These and other hardware-based physical security techniques contribute to tamper-resistant hardware security barrier 502.

To increase the security of security barrier 502 even further, it is possible to encase or include SPU 500 in one or more further physical enclosures such as, for example: epoxy or other "potting compound"; further module enclosures including additional self-destruct, self-disabling or other features activated when tampering is detected; further modules providing additional security protections such as requiring password or other authentication to operate; and the like. In addition, further layers of metal may be added to the die to complicate acid etching, micro probing, and the like; circuitry designed to "zeroize" memory may be included as an aspect of self-destruct processes; the plastic package itself may be designed to resist

chemical as well as physical "attacks"; and memories internal to SPU 500 may have specialized addressing and refresh circuitry that "shuffles" the location of bits to complicate efforts to electrically determine the value of memory locations. These and other techniques may contribute to the security of barrier 502.

In some electronic appliances 600, SPU 500 may be integrated together with the device microcontroller or equivalent or with a device I/O or communications microcontroller into a common chip (or chip set) 505. For example, in one preferred embodiment, SPU 500 may be integrated together with one or more other CPU(s) (e.g., a CPU 654 of an electronic appliance) in a single component or package. The other CPU(s) 654 may be any centrally controlling logic arrangement, such as for example, a microprocessor, other microcontroller, and/or array or other parallel processor. This integrated configuration may result in lower overall cost, smaller overall size, and potentially faster interaction between an SPU 500 and a CPU 654. Integration may also provide wider distribution if an integrated SPU/CPU component is a standard feature of a widely distributed microprocessor line. Merging an SPU 500 into a main CPU 654 of an electronic appliance 600 (or into another appliance or appliance peripheral microcomputer or other microcontroller) may substantially reduce the overhead cost of implementing

VDE 100. Integration considerations may include cost of implementation, cost of manufacture, desired degree of security, and value of compactness.

SPU 500 may also be integrated with devices other than CPUs. For example, for video and multimedia applications, some performance and/or security advantages (depending on overall design) could result from integrating an SPU 500 into a video controller chip or chipset. SPU 500 can also be integrated directly into a network communications chip or chipset or the like. Certain performance advantages in high speed communications applications may also result from integrating an SPU 500 with a modem chip or chipset. This may facilitate incorporation of an SPU 500 into communication appliances such as stand-alone fax machines. SPU 500 may also be integrated into other peripheral devices, such as CD-ROM devices, set-top cable devices, game devices, and a wide variety of other electronic appliances that use, allow access to, perform transactions related to, or consume, distributed information.

SPU 500 Internal Architecture

Figure 9 is a detailed diagram of the internal structure within an example of SPU 500. SPU 500 in this example includes a single microprocessor 520 and a limited amount of

memory configured as ROM 532 and RAM 534. In more detail, this example of SPU 500 includes microprocessor 520, an encrypt/decrypt engine 522, a DMA controller 526, a real-time clock 528, a bus interface unit ("BIU") 530, a read only memory (ROM) 532, a random access memory (RAM) 534, and a memory management unit ("MMU") 540. DMA controller 526 and MMU 540 are optional, but the performance of SPU 500 may suffer if they are not present. SPU 500 may also include an optional pattern matching engine 524, an optional random number generator 542, an optional arithmetic accelerator circuit 544, and optional compression/decompression circuit 546. A shared address/data bus arrangement 536 may transfer information between these various components under control of microprocessor 520 and/or DMA controller 526. Additional or alternate dedicated paths 538 may connect microprocessor 520 to the other components (e.g., encrypt/decrypt engine 522 via line 538a, real-time clock 528 via line 538b, bus interface unit 530 via line 538c, DMA controller via line 538d, and memory management unit (MMU) 540 via line 538e).

The following section discusses each of these SPU components in more detail.

Microprocessor 520

Microprocessor 520 is the "brain" of SPU 500. In this example, it executes a sequence of steps specified by code stored (at least temporarily) within ROM 532 and/or RAM 534. Microprocessor 520 in the preferred embodiment comprises a dedicated central processing arrangement (e.g., a RISC and/or CISC processor unit, a microcontroller, and/or other central processing means or, less desirably in most applications, process specific dedicated control logic) for executing instructions stored in the ROM 532 and/or other memory. Microprocessor 520 may be separate elements of a circuitry layout, or may be separate packages within a secure SPU 500.

In the preferred embodiment, microprocessor 520 normally handles the most security sensitive aspects of the operation of electronic appliance 600. For example, microprocessor 520 may manage VDE decrypting, encrypting, certain content and/or appliance usage control information, keeping track of usage of VDE secured content, and other VDE usage control related functions.

Stored in each SPU 500 and/or electronic appliance secondary memory 652 may be, for example, an instance of ROS 602 software, application programs 608, objects 300 containing

VDE controlled property content and related information, and management database 610 that stores both information associated with objects and VDE control information. ROS 602 includes software intended for execution by SPU microprocessor 520 for, in part, controlling usage of VDE related objects 300 by electronic appliance 600. As will be explained, these SPU programs include "load modules" for performing basic control functions. These various programs and associated data are executed and manipulated primarily by microprocessor 520.

Real Time Clock (RTC) 528

In the preferred embodiment, SPU 500 includes a real time clock circuit ("RTC") 528 that serves as a reliable, tamper resistant time base for the SPU. RTC 528 keeps track of time of day and date (e.g., month, day and year) in the preferred embodiment, and thus may comprise a combination calendar and clock. A reliable time base is important for implementing time based usage metering methods, "time aged decryption keys," and other time based SPU functions.

The RTC 528 must receive power in order to operate. Optimally, the RTC 528 power source could comprise a small battery located within SPU 500 or other secure enclosure. However, the RTC 528 may employ a power source such as an

externally located battery that is external to the SPU 500. Such an externally located battery may provide relatively uninterrupted power to RTC 528, and may also maintain as non-volatile at least a portion of the otherwise volatile RAM 534 within SPU 500.

In one implementation, electronic appliance power supply 659 is also used to power SPU 500. Using any external power supply as the only power source for RTC 528 may significantly reduce the usefulness of time based security techniques unless, at minimum, SPU 500 recognizes any interruption (or any material interruption) of the supply of external power, records such interruption, and responds as may be appropriate such as disabling the ability of the SPU 500 to perform certain or all VDE processes. Recognizing a power interruption may, for example, be accomplished by employing a circuit which is activated by power failure. The power failure sensing circuit may power another circuit that includes associated logic for recording one or more power fail events. Capacitor discharge circuitry may provide the necessary temporary power to operate this logic. In addition or alternatively, SPU 500 may from time to time compare an output of RTC 528 to a clock output of a host electronic appliance 600, if available. In the event a discrepancy is detected, SPU 500 may respond as appropriate, including

recording the discrepancy and/or disabling at least some portion of processes performed by SPU 500 under at least some circumstances.

If a power failure and/or RTC 528 discrepancy and/or other event indicates the possibility of tampering, SPU 500 may automatically destroy, or render inaccessible without privileged intervention, one or more portions of sensitive information it stores, such as execution related information and/or encryption key related information. To provide further SPU operation, such destroyed information would have to be replaced by a VDE clearinghouse, administrator and/or distributor, as may be appropriate. This may be achieved by remotely downloading update and/or replacement data and/or code. In the event of a disabling and/or destruction of processes and/or information as described above, the electronic appliance 600 may require a secure VDE communication with an administrator, clearinghouse, and/or distributor as appropriate in order to reinitialize the RTC 528. Some or all secure SPU 500 processes may not operate until then.

It may be desirable to provide a mechanism for setting and/or synchronizing RTC 528. In the preferred embodiment, when communication occurs between VDE electronic appliance

600 and another VDE appliance, an output of RTC 528 may be compared to a controlled RTC 528 output time under control of the party authorized to be "senior" and controlling. In the event of a discrepancy, appropriate action may be taken, including resetting the RTC 528 of the "junior" controlled participant in the communication.

SPU Encrypt/Decrypt Engine 522

In the preferred embodiment, SPU encrypt/decrypt engine 522 provides special purpose hardware (e.g., a hardware state machine) for rapidly and efficiently encrypting and/or decrypting data. In some implementations, the encrypt/decrypt functions may be performed instead by microprocessor 520 under software control, but providing special purpose encrypt/decrypt hardware engine 522 will, in general, provide increased performance. Microprocessor 520 may, if desired, comprise a combination of processor circuitry and dedicated encryption/decryption logic that may be integrated together in the same circuitry layout so as to, for example, optimally share one or more circuit elements.

Generally, it is preferable that a computationally efficient but highly secure "bulk" encryption/decryption technique should be used to protect most of the data and objects handled by SPU 500. It is preferable that an extremely secure

encryption/decryption technique be used as an aspect of authenticating the identity of electronic appliances 600 that are establishing a communication channel and securing any transferred permission, method, and administrative information. In the preferred embodiment, the encrypt/decrypt engine 522 includes both a symmetric key encryption/decryption circuit (e.g. DES, Skipjack/Clipper, IDEA, RC-2, RC-4, etc.) and an antisymmetric (asymmetric) or Public Key ("PK") encryption/decryption circuit. The public/private key encryption/decryption circuit is used principally as an aspect of secure communications between an SPU 500 and VDE administrators, or other electronic appliances 600, that is between VDE secure subsystems. A symmetric encryption/decryption circuit may be used for "bulk" encrypting and decrypting most data stored in secondary storage 662 of electronic appliance 600 in which SPU 500 resides. The symmetric key encryption/decryption circuit may also be used for encrypting and decrypting content stored within VDE objects 300.

DES or public/private key methods may be used for all encryption functions. In alternate embodiments, encryption and decryption methods other than the DES and public/private key methods could be used for the various encryption related

functions. For instance, other types of symmetric encryption/decryption techniques in which the same key is used for encryption and decryption could be used in place of DES encryption and decryption. The preferred embodiment can support a plurality of decryption/encryption techniques using multiple dedicated circuits within encrypt/decrypt engine 522 and/or the processing arrangement within SPU 500.

Pattern Matching Engine 524

Optional pattern matching engine 524 may provide special purpose hardware for performing pattern matching functions. One of the functions SPU 500 may perform is to validate/authenticate VDE objects 300 and other items. Validation/authentication often involves comparing long data strings to determine whether they compare in a predetermined way. In addition, certain forms of usage (such as logical and/or physical (contiguous) relatedness of accessed elements) may require searching potentially long strings of data for certain bit patterns or other significant pattern related metrics. Although pattern matching can be performed by SPU microprocessor 520 under software control, providing special purpose hardware pattern matching engine 524 may speed up the pattern matching process.

Compression/Decompression Engine 546

An optional compression/decompression engine 546 may be provided within an SPU 500 to, for example, compress and/or decompress content stored in, or released from, VDE objects 300. Compression/decompression engine 546 may implement one or more compression algorithms using hardware circuitry to improve the performance of compression/decompression operations that would otherwise be performed by software operating on microprocessor 520, or outside SPU 500. Decompression is important in the release of data such as video and audio that is usually compressed before distribution and whose decompression speed is important. In some cases, information that is useful for usage monitoring purposes (such as record separators or other delimiters) is "hidden" under a compression layer that must be removed before this information can be detected and used inside SPU 500.

Random Number Generator 542

Optional random number generator 542 may provide specialized hardware circuitry for generating random values (e.g., from inherently unpredictable physical processes such as quantum noise). Such random values are particularly useful for constructing encryption keys or unique identifiers, and for initializing the generation of pseudo-random sequences.

Random number generator 542 may produce values of any convenient length, including as small as a single bit per use. A random number of arbitrary size may be constructed by concatenating values produced by random number generator 542. A cryptographically strong pseudo-random sequence may be generated from a random key and seed generated with random number generator 542 and repeated encryption either with the encrypt/decrypt engine 522 or cryptographic algorithms in SPU 500. Such sequences may be used, for example, in private headers to frustrate efforts to determine an encryption key through cryptoanalysis.

Arithmetic Accelerator 544

An optional arithmetic accelerator 544 may be provided within an SPU 500 in the form of hardware circuitry that can rapidly perform mathematical calculations such as multiplication and exponentiation involving large numbers. These calculations can, for example, be requested by microprocessor 520 or encrypt/decrypt engine 522, to assist in the computations required for certain asymmetric encryption/decryption operations. Such arithmetic accelerators are well-known to those skilled in the art. In some implementations, a separate arithmetic accelerator 544 may be

omitted and any necessary calculations may be performed by microprocessor 520 under software control.

DMA Controller 526

DMA controller 526 controls information transfers over address/data bus 536 without requiring microprocessor 520 to process each individual data transfer. Typically, microprocessor 520 may write to DMA controller 526 target and destination addresses and the number of bytes to transfer, and DMA controller 526 may then automatically transfer a block of data between components of SPU 500 (e.g., from ROM 532 to RAM 534, between encrypt/decrypt engine 522 and RAM 534, between bus interface unit 530 and RAM 534, etc.). DMA controller 526 may have multiple channels to handle multiple transfers simultaneously. In some implementations, a separate DMA controller 526 may be omitted, and any necessary data movements may be performed by microprocessor 520 under software control.

Bus Interface Unit (BIU) 530

Bus interface unit (BIU) 530 communicates information between SPU 500 and the outside world across the security barrier 502. BIU 530 shown in Figure 9 plus appropriate driver software may comprise the "appliance link" 510 shown in Figure

6. Bus interface unit 530 may be modelled after a USART or PCI bus interface in the preferred embodiment. In this example, BIU 530 connects SPU 500 to electronic appliance system bus 653 shown in Figure 8. BIU 530 is designed to prevent unauthorized access to internal components within SPU 500 and their contents. It does this by only allowing signals associated with an SPU 500 to be processed by control programs running on microprocessor 520 and not supporting direct access to the internal elements of an SPU 500.

Memory Management Unit 540

Memory Management Unit (MMU) 540, if present, provides hardware support for memory management and virtual memory management functions. It may also provide heightened security by enforcing hardware compartmentalization of the secure execution space (e.g., to prevent a less trusted task from modifying a more trusted task). More details are provided below in connection with a discussion of the architecture of a Secure Processing Environment ("SPE") 503 supported by SPU 500.

MMU 540 may also provide hardware-level support functions related to memory management such as, for example, address mapping.

SPU Memory Architecture

In the preferred embodiment, SPU 500 uses three general kinds of memory:

(1) internal ROM 532;

(2) internal RAM 534; and

(3) external memory (typically RAM and/or disk supplied by a host electronic appliance).

The internal ROM 532 and RAM 534 within SPU 500 provide a secure operating environment and execution space. Because of cost limitations, chip fabrication size, complexity and other limitations, it may not be possible to provide sufficient memory within SPU 500 to store all information that an SPU needs to process in a secure manner. Due to the practical limits on the amount of ROM 532 and RAM 534 that may be included within SPU 500, SPU 500 may store information in memory external to it, and move this information into and out of its secure internal memory space on an as needed basis. In these cases, secure processing steps performed by an SPU typically must be segmented into small, securely packaged elements that may be "paged in" and "paged out" of the limited available internal memory space. Memory external to an SPU 500 may not be secure. Since the external memory may not be secure, SPU 500 may encrypt and cryptographically seal code and other

information before storing it in external memory. Similarly, SPU 500 must typically decrypt code and other information obtained from external memory in encrypted form before processing (e.g., executing) based on it. In the preferred embodiment, there are two general approaches used to address potential memory limitations in a SPU 500. In the first case, the small, securely packaged elements represent information contained in secure database 610. In the second case, such elements may represent protected (e.g., encrypted) virtual memory pages. Although virtual memory pages may correspond to information elements stored in secure database 610, this is not required in this example of a SPU memory architecture.

The following is a more detailed discussion of each of these three SPU memory resources.

SPU Internal ROM

SPU 500 read only memory (ROM) 532 or comparable purpose device provides secure internal non-volatile storage for certain programs and other information. For example, ROM 532 may store "kernel" programs such as SPU control firmware 508 and, if desired, encryption key information and certain fundamental "load modules." The "kernel" programs, load module information, and encryption key information enable the

control of certain basic functions of the SPU 500. Those components that are at least in part dependent on device configuration (e.g., POST, memory allocation, and a dispatcher) may be loaded in ROM 532 along with additional load modules that have been determined to be required for specific installations or applications.

In the preferred embodiment, ROM 532 may comprise a combination of a masked ROM 532a and an EEPROM and/or equivalent "flash" memory 532b. EEPROM or flash memory 532b is used to store items that need to be updated and/or initialized, such as for example, certain encryption keys. An additional benefit of providing EEPROM and/or flash memory 532b is the ability to optimize any load modules and library functions persistently stored within SPU 500 based on typical usage at a specific site. Although these items could also be stored in NVRAM 534b, EEPROM and/or flash memory 532b may be more cost effective.

Masked ROM 532a may cost less than flash and/or EEPROM 532b, and can be used to store permanent portions of SPU software/firmware. Such permanent portions may include, for example, code that interfaces to hardware elements such as the RTC 528, encryption/decryption engine 522, interrupt

handlers, key generators, etc. Some of the operating system, library calls, libraries, and many of the core services provided by SPU 500 may also be in masked ROM 532a. In addition, some of the more commonly used executables are also good candidates for inclusion in masked ROM 532a. Items that need to be updated or that need to disappear when power is removed from SPU 500 should not be stored in masked ROM 532a.

Under some circumstances, RAM 534a and/or NVRAM 534b (NVRAM 534b may, for example, be constantly powered conventional RAM) may perform at least part of the role of ROM 532.

SPU Internal RAM

SPU 500 general purpose RAM 534 provides, among other things, secure execution space for secure processes. In the preferred embodiment, RAM 534 is comprised of different types of RAM such as a combination of high-speed RAM 534a and an NVRAM ("non-volatile RAM") 534b. RAM 534a may be volatile, while NVRAM 534b is preferably battery backed or otherwise arranged so as to be non-volatile (i.e., it does not lose its contents when power is turned off).

High-speed RAM 534a stores active code to be executed and associated data structures.

NVRAM 534b preferably contains certain keys and summary values that are preloaded as part of an initialization process in which SPU 500 communicates with a VDE administrator, and may also store changeable or changing information associated with the operation of SPU 500. For security reasons, certain highly sensitive information (e.g., certain load modules and certain encryption key related information such as internally generated private keys) needs to be loaded into or generated internally by SPU 500 from time to time but, once loaded or generated internally, should never leave the SPU. In this preferred embodiment, the SPU 500 non-volatile random access memory (NVRAM) 534b may be used for securely storing such highly sensitive information. NVRAM 534b is also used by SPU 500 to store data that may change frequently but which preferably should not be lost in a power down or power fail mode.

NVRAM 534b is preferably a flash memory array, but may in addition or alternatively be electrically erasable programmable read only memory (EEPROM), static RAM (SRAM), bubble memory, three dimensional holographic or other

electro-optical memory, or the like, or any other writable (e.g., randomly accessible) non-volatile memory of sufficient speed and cost-effectiveness.

SPU External Memory

The SPU 500 can store certain information on memory devices external to the SPU. If available, electronic appliance 600 memory can also be used to support any device external portions of SPU 500 software. Certain advantages may be gained by allowing the SPU 500 to use external memory. As one example, memory internal to SPU 500 may be reduced in size by using non-volatile read/write memory in the host electronic appliance 600 such as a non-volatile portion of RAM 656 and/or ROM 658.

Such external memory may be used to store SPU programs, data and/or other information. For example, a VDE control program may be, at least in part, loaded into the memory and communicated to and decrypted within SPU 500 prior to execution. Such control programs may be re-encrypted and communicated back to external memory where they may be stored for later execution by SPU 500. "Kernel" programs and/or some or all of the non-kernel "load modules" may be stored by SPU 500 in memory external to it. Since a secure database 610

may be relatively large, SPU 500 can store some or all of secure database 610 in external memory and call portions into the SPU 500 as needed.

As mentioned above, memory external to SPU 500 may not be secure. Therefore, when security is required, SPU 500 must encrypt secure information before writing it to external memory, and decrypt secure information read from external memory before using it. Inasmuch as the encryption layer relies on secure processes and information (e.g., encryption algorithms and keys) present within SPU 500, the encryption layer effectively "extends" the SPU security barrier 502 to protect information the SPU 500 stores in memory external to it.

SPU 500 can use a wide variety of different types of external memory. For example, external memory may comprise electronic appliance secondary storage 652 such as a disk; external EEPROM or flash memory 658; and/or external RAM 656. External RAM 656 may comprise an external nonvolatile (e.g. constantly powered) RAM and/or cache RAM.

Using external RAM local to SPU 500 can significantly improve access times to information stored externally to an SPU. For example, external RAM may be used:

- to buffer memory image pages and data structures prior to their storage in flash memory or on an external hard disk (assuming transfer to flash or hard disk can occur in significant power or system failure cases);
- provide encryption and decryption buffers for data being released from VDE objects 300.
- to cache "swap blocks" and VDE data structures currently in use as an aspect of providing a secure virtual memory environment for SPU 500.
- to cache other information in order to, for example, reduce frequency of access by an SPU to secondary storage 652 and/or for other reasons.

Dual ported external RAM can be particularly effective in improving SPU 500 performance, since it can decrease the data movement overhead of the SPU bus interface unit 530 and SPU microprocessor 520.

Using external flash memory local to SPU 500 can be used to significantly improve access times to virtually all data structures. Since most available flash storage devices have limited write lifetimes, flash storage needs to take into account the number of writes that will occur during the lifetime of the flash memory. Hence, flash storage of frequently written temporary items is not recommended. If external RAM is non-

volatile, then transfer to flash (or hard disk) may not be necessary.

External memory used by SPU 500 may include two categories:

- external memory dedicated to SPU 500, and
- memory shared with electronic appliance 600.

For some VDE implementations, sharing memory (e.g., electronic appliance RAM 656, ROM 658 and/or secondary storage 652) with CPU 654 or other elements of an electronic appliance 600 may be the most cost effective way to store VDE secure database management files 610 and information that needs to be stored external to SPU 500. A host system hard disk secondary memory 652 used for general purpose file storage can, for example, also be used to store VDE management files 610. SPU 500 may be given exclusive access to the external memory (e.g., over a local bus high speed connection provided by BIU 530). Both dedicated and shared external memory may be provided.

SPU Integrated Within CPU

As discussed above, it may be desirable to integrate CPU 654 and SPU 500 into the same integrated circuit and/or device.

SPU 500 shown in Figure 9 includes a microprocessor 520 that may be similar or identical to a standard microprocessor available off-the-shelf from a variety of manufacturers. Similarly, the SPU DMA controller 526 and certain other microprocessor support circuitry may be standard implementations available in off-the-shelf microprocessor and/or microcomputer chips. Since many of the general control and processing requirements provided by SPU 500 in the preferred embodiment can be satisfied using certain generic CPU and/or microcontroller components, it may be desirable to integrate SPU VDE functionality into a standard generic CPU or microcontroller chip. Such an integrated solution can result in a very cost-effective "dual mode" component that is capable of performing all of the generic processing of a standard CPU as well as the secure processing of an SPU. Many of the control logic functions performed by the preferred embodiment SPU can be performed by generic CPU and/or micro-controller logic so that at least a portion of the control logic does not have to be duplicated. Additional cost savings (e.g., in terms of reducing manufacturing costs, inventory costs and printed circuit board real estate requirements) may also be obtained by not requiring an additional, separate physical SPU 500 device or package. Figure 9A shows one example architecture of a combination CPU/SPU 2650. CPU/SPU 2650 may include a standard

microprocessor or microcontroller 2652, a standard bus interface unit (BIU) 2656, and a standard (optional) DMA controller 2654, as well as various other standard I/O controllers, computation circuitry, etc. as may be found in a typical off-the-shelf microprocessor/microcontroller. Real time clock 528 may be added to the standard architecture to give the CPU/SPU 2650 access to the real time clock functions as discussed above in connection with Figure 9. Real-time clock 528 must be protected from tampering in order to be secure. Such protections may include internal or external backup power, an indication that its power (and thus its operation) has been interrupted, and/or an indication that the external clock signal(s) from which it derives its timing have been interfered with (e.g., sped up, slowed down). Similarly, an encrypt/decrypt engine 522, pattern matching engine 524, compression/decompression engine 546 and/or arithmetic accelerator 544 may be added if desired to provide greater efficiencies, or the functions performed by these components could be provided instead by software executing on microprocessor 2652. An optional memory management unit 540 may also be provided if desired. A true random number generator 542 may be provided also if desired. Connections shown between mode interface switch 2658 and other components can carry both data and control information, specifically control information that determines what security-

relevant aspects of the other components are available for access and/or manipulation.

In addition, secure ROM 532 and/or secure RAM 534 may be provided within CPU/SPU 2650 along with a "mode interface switch" 2658a, 2658b. Mode interface switch 2658 selectively provides microprocessor 2652 with access to secure memory 532, 534 and other secure components (blocks 522, 546, 524, 542, 544, 528) depending upon the "mode" CPU/SPU 2650 is operating in. CPU/SPU 2650 in this example may operate in two different modes:

- an "SPU" mode, or
- a "normal" mode.

In the "normal" mode, CPU/SPU 2650 operates substantially identically to a standard off-the-shelf CPU while also protecting the security of the content, state, and operations of security-relevant components included in CPU/SPU 2650. Such security-relevant components may include the secure memories 532, 534; the encrypt/decrypt engine 522, the optional pattern-matching engine 524, random number generator 542, arithmetic accelerator 544, the SPU-not-initialized flag 2671, the secure mode interface switch 2658, the real-time clock 528, the DMA controller 2654, the MMU 540, compress/decompress block

546, and/or any other components that may affect security of the operation of the CPU/SPU in "SPU" mode.

In this example, CPU/SPU 2650 operating in the "normal" mode controls mode interface switch 2658 to effectively "disconnect" (i.e., block unsecure access to) the security-relevant components, or to the security-relevant aspects of the operations of such components as have a function for both "normal" and "SPU" mode. In the "normal" mode, for example, microprocessor 2652 could access information from standard registers or other internal RAM and/or ROM (not shown), execute instructions in a "normal" way, and perform any other tasks as are provided within a standard CPU -- but could not access or compromise the contents of secure memory 532, 534 or access blocks 522, 524, 542, 544, 546. In this example "normal" mode, mode interface switch 2658 would effectively prevent any access (e.g., both read and write access) to secure memory 532, 534 so as to prevent the information stored within that secure memory from being compromised.

When CPU/SPU 2650 operates in the "SPU" mode, mode interface switch 2658 allows microprocessor 2652 to access secure memory 532, 534, and to control security-relevant aspects of other components in the CPU/SPU. The "SPU" mode in this

example requires all instructions executed by microprocessor 2652 to be fetched from secure memory 532, 534 -- preventing execution based on "mixed" secure and non-secure instructions. In the "SPU" mode, mode interface switch 2658 may, in one example embodiment, disconnect or otherwise block external accesses carried over bus 652 from outside CPU/SPU 2650 (e.g., DMA accesses, cache coherency control accesses) to ensure that the microprocessor 2652 is controlled entirely by instructions carried within or derived from the secure memory 532, 534. Mode interface switch 2658 may also disconnect or otherwise block access by microprocessor 2652 to some external memory and/or other functions carried over bus 652. Mode interface switch 2658 in this example prevents other CPU operations/instructions from exposing the contents of secure memory 532, 534.

In the example shown in Figure 9A, the mode control of mode interface switch 2658 is based on a "mode" control signal provided by microprocessor 2652. In this example, microprocessor 2652 may be slightly modified so it can execute two "new" instructions:

- "enable 'SPU' mode" instruction, and
- "disable 'SPU' mode" instruction.

When microprocessor 2652 executes the "enable 'SPU' mode" instruction, it sends an appropriate "mode" control signal to mode interface switch 2658 to "switch" the interface switch into the "SPU" mode of operation. When microprocessor 2652 executes the "disable 'SPU' mode" instruction, it sends an appropriate "mode" control signal to mode interface switch 2658 to disable the "SPU" mode of operation.

When CPU/SPU 2650 begins operating in the "SPU" mode (based on microprocessor 2652 executing the "enable "SPU" mode" instruction), mode interface switch 2658 forces microprocessor 2652 to begin fetching instructions from secure memory 532, 534 (e.g., beginning at some fixed address) in one example. When CPU/SPU 2650 begins operating in this example "SPU" mode, mode interface switch 2658 may force microprocessor 2652 to load its registers from some fixed address in secure memory 532, 534 and may begin execution based on such register content. Once operating in the "SPU" mode, microprocessor 2652 may provide encryption/decryption and other control capabilities based upon the code and other content of secure memory 532, 534 needed to provide the VDE

functionality of SPU 500 described above. For example, microprocessor 2652 operating under control of information within secure memory 532, 534 may read encrypted information from bus 652 via bus interface unit 2656, write decrypted information to the bus interface unit, and meter and limit decryption of such information based on values stored in the secure memory.

At the end of secure processing, execution by microprocessor 2652 of the "disable SPU mode" instruction may cause the contents of all registers and other temporary storage locations used by microprocessor 2652 that are not within secure memory 532, 534 to be destroyed or copied into secure memory 532, 534 before "opening" mode interface switch 2658. Once mode interface switch 2658 is "open," the microprocessor 2652 no longer has access to secure memory 532, 534 or the information it contained, or to control or modify the state of any other security-relevant components or functions contained within CPU/SPU 2650 to which access is controlled by mode interface switch 2658.

Whenever CPU/SPU 2650 enters or leaves the "SPU" mode, the transition is performed in such a way that no information contained in the secure memory 532, 534 or derived

from it (e.g., stored in registers or a cache memory associated with microprocessor 2652) while in the "SPU" mode can be exposed by microprocessor 2652 operations that occur in the "normal" mode. This may be accomplished either by hardware mechanisms that protect against such exposure, software instructions executed in "SPU" mode that clear, reinitialize, and otherwise reset during such transitions, or a combination of both.

In some example implementations, interrupts may be enabled while CPU/SPU 2650 is operating in the "SPU" mode similarly interrupts and returns from interrupts while in the "SPU" mode may allow transitions from "SPU" mode to "normal" mode and back to "SPU" mode without exposing the content of secure memory 532, 534 or the content of registers or other memory associated with microprocessor 2652 that may contain information derived from secure mode operation.

In some example implementations, there may be CPU/SPU activities such as DMA transfers between external memory and/or devices and secure memory 532, 534 that are initiated by microprocessor 2652 but involve autonomous activity by DMA controller 2654 and, optionally, encrypt/decrypt engine 522 and/or compress/decompress engine 546. In such implementations, mode interface switch 2658 and its associated

control signals may be configured to permit such pending activities (e.g. DMA transfers) to continue to completion even after CPU/SPU 2650 leaves "SPU" mode, provided that upon completion, all required clearing, reinitialization, and/or reset activities occur, and provided that no access or interference is permitted with the pending activities except when CPU/SPU 2650 is operating in "SPU" mode.

In an additional example embodiment, encryption/decryption logic may be connected between microprocessor 2652 and secure memory 532, 354. This additional encryption/decryption logic may be connected "in parallel" to mode interface switch 2658. The additional encryption/decryption logic may allow certain accesses by microprocessor 2652 to the secure memory 532, 534 when CPU/SPU 2650 is operating in the "normal" mode. In this alternate embodiment, reads from secure memory 532, 534 when CPU/SPU 2650 is operating in the "normal" mode automatically result in the read information being encrypted before it is delivered to microprocessor 2652 (and similarly, and writes to the secure memory may result in the written information being decrypted before it is deposited into the secure memory). This alternative embodiment may permit access to secure memory 532, 534 (which may in this example store the information in

"clear" form) by microprocessor 2652 when CPU/SPU 2650 is operating in the "non-secure normal" mode, but only reveals the secure memory contents to microprocessor 2652 in unencrypted form when the CPU/SPU is operating in the "SPU" mode. Such access may also be protected by cryptographic authentication techniques (e.g., message authentication codes) to prevent modification or replay attacks that modify encrypted data stored in secure memory 532, 534. Such protection may be performed utilizing either or both of software and/or hardware cryptographic techniques.

All of the components shown in Figure 9A may be disposed within a single integrated circuit package. Alternatively, mode interface switch 2658 and secure memory 532, 534, and other security-relevant components might be placed within an integrated circuit chip package and/or other package separate from the rest of CPU/SPU 2650. In this two-package version, a private bus could be used to connect microprocessor 2652 to the mode interface switch 2658 and associated secure memory 532, 534. To maintain security in such multi-package versions, it may be necessary to enclose all the packages and their interconnections in an external physical tamper-resistant barrier.

Initialization of Integrated CPU/SPU

Instructions and/or data may need to be loaded into CPU/SPU 2650 before it can operate effectively as an SPU 500. This may occur during the manufacture of CPU/SPU 2650 or subsequently at a CPU/SPU initialization facility. Security of such initialization may depend on physical control of access to the CPU/SPU component(s), on cryptographic means, or on some combination of both. Secure initialization may be performed in plural steps under the control of different parties, such that an initialization step to be performed by party B is preconditioned on successful performance of a step by party A. Different initialization steps may be protected using different security techniques (e.g. physical access, cryptography).

In this example, switch 2658 may expose an external control signal 2670 that requests operation in "SPU" mode rather than "normal" mode after a power-on reset. This signal would be combined (e.g., by a logical AND 2672) with a non-volatile storage element 2671 internal to CPU/SPU 2650. If both of these signals are asserted, AND gate 2672 would cause CPU/SPU 2650 to begin operating in SPU mode, either executing existing instructions from an address in SPU memory 532, executing instructions from main memory 2665 or otherwise external to the

CPU/SPU. The instructions thus executed would permit arbitrary initialization and other functions to be performed in "SPU" mode without necessarily requiring any instructions to be previously resident in the SPU memory 532.

Once initialized, the SPU would, under control of its initialization program, indicate to switch 2658 that the flag 2671 is to be cleared. Clearing flag 2671 would permanently disable this initialization capability because no mechanism would be provided to set flag 2671 back to its initial value.

If flag 2671 is clear, or control signal 2670 is not asserted, CPU/SPU 2650 would behave precisely as does microprocessor 2652 with respect to power-on reset and other external conditions. Under such conditions, only execution of the "enable SPU mode" instruction or otherwise requesting SPU mode under program control would cause "SPU" mode to be entered.

Additionally, a mechanism could be provided to permit microprocessor 2652 and/or control signal 2672 to reinitialize the flag 2671. Such reinitialization would be performed in a manner that cleared secure memory 532, 534 of any security-relevant information and reinitialized the state of all security-relevant components. This reinitialization mechanism would permit CPU/SPU 2650 to be initialized several times, facilitating testing

and/or re-use for different applications, while protecting all security-relevant aspects of its operation.

In the preferred embodiment, CPU/SPU 2650 would, when SPU mode has not yet been established, begin operating in SPU mode by fetching instructions from secure non-volatile memory 532, thereby ensuring a consistent initialization sequence and preventing SPU dependence on any information held outside CPU/SPU 2650. This approach permits secret initialization information (e.g., keys for validating digital signatures on additional information to be loaded into secure memory 532, 534) to be held internally to CPU/SPU 2650 so that it is never exposed to outside access. Such information could even be supplied by a hardware "mask" used in the semiconductor fabrication process.

CPU/SPU Integrated With Unmodified Microprocessor

Figure 9B shows an additional example embodiment, in which a completely standard microprocessor 2652 integrated circuit chip could be transformed into a CPU/SPU 2650 by adding an SPU chip 2660 that mediates access to external I/O devices and memory. In such an embodiment, the microprocessor 2652 would be connected to the SPU chip 2660 by a private memory bus 2661, and all three such components

would be contained within hardware tamper-resistant barrier 502.

In this embodiment, SPU chip 2660 may have the same secure components as in Figure 9, i.e., it may have a ROM/EEPROM 532, a RAM 532, an RTC 528, an (optional) encryption/decryption engine 522, an (optional) random number generator (RNG) 542, an (optional) arithmetic accelerator 544, and a (optional) compression/decompression engine 546, and a (optional) pattern matching circuit 524. Microprocessor 520 is omitted from SPU chip 2660 since the standard microprocessor 2650 performs the processing functions instead. In addition, SPU chip 2660 may include a flag 2671 and AND gate logic 2672 for the initialization purposes discussed above.

In addition, SPU chip 2660 includes an enhanced switch 2663 that provides the same overall (bus enhanced) functionality performed by the switch 2658 in the Figure 9A embodiment.

Enhanced switch 2663 would perform the functions of a bus repeater, mediator and interpreter. For example, enhanced switch 2663 may act as a bus repeater that enables microprocessor 2652's memory accesses made over internal memory bus 2661 to be reflected to external memory bus 2664

and performed on main memory 2665. Enhanced switch 2663 may also act as a bus repeater similarly for internal I/O bus 2662 to external I/O bus 2665 in the event that microprocessor 2652 performs I/O operations distinctly from memory operations. Enhanced switch 2663 may also perform the function of a mediator for microprocessor control functions 2666 (e.g., non-maskable interrupt, reset) with respect to externally requested control functions 2667. Enhanced switch 2663 may also provide mediation for access to SPU-protected resources such as ROM 532, RAM 534, encrypt/decrypt engine 522 (if present), random number generator 542 (if present), arithmetic accelerator 544 (if present), pattern matching engine 524 (if present), and real-time clock 528 (if present). Enhanced switch 2663 may also act as an interpreter of control signals received from microprocessor 2652 indicating entry to, exit from, and control of SPU mode.

Switch 2663 in this example recognizes a specific indication (e.g., an instruction fetch access to a designated address in the secure memory 532) as the equivalent to the "enable 'SPU' mode" instruction. Upon recognizing such an indication, it may isolate the CPU/SPU 2650 from external buses and interfaces 2664, 2665, and 2667 such that any external activity, such as DMA cycles, would be "held" until the switch

2663 permits access again. After this, switch 2663 permits a single access to a specific location in secure memory 532 to complete.

The single instruction fetched from the designated location performs a control operation (a cache flush, for example), that can only be performed in microprocessor 2652's most privileged operating mode, and that has an effect visible to switch 2663. Switch 2663 awaits the occurrence of this event, and if it does not occur within the expected number of cycles, does not enter "SPU" mode.

Occurrence of the control operation demonstrates that microprocessor 2652 is executing in its most privileged "normal" mode and therefore can be trusted to execute successfully the "enter 'SPU' mode" sequence of instructions stored in secure memory 532. If microprocessor 2652 were not executing in its most privileged mode, there would be no assurance that those instructions would execute successfully. Because switch 2663 isolates microprocessor 2652 from external signals (e.g., interrupts) until "SPU" mode is successfully initialized, the entry instructions can be guaranteed to complete successfully.

Following the initial instruction, switch 2663 can enter "partial SPU mode," in which a restricted area of ROM 532 and RAM 534 may be accessible. Subsequent instructions in secure memory 532 may then be executed by microprocessor 2652 to place it into a known state such that it can perform SPU functions -- saving any previous state in the restricted area of RAM 534 that is accessible. After the known state is established, an instruction may be executed to deliver a further indication (e.g., a reference to another designated memory location) to switch 2663, which would enter "SPU" mode. If this further indication is not received within the expected interval, switch 2663 will not enter "SPU" mode. Once in "SPU" mode, switch 2663 permits access to all of ROM 532, RAM 534, and other devices in SPU chip 2660.

The instructions executed during "partial SPU" mode must be carefully selected to ensure that no similar combination of instructions and processor state could result in a control transfer out of the protected SPU code in ROM 532 or RAM 534. For example, internal debugging features of microprocessor 2652 must be disabled to ensure that a malicious program could not set up a breakpoint later within protected SPU code and receive control. Similarly, all address translation must be disabled or reinitialized to ensure that previously created MMU data

structures would not permit SPU memory accesses to be compromised. The requirement that the instructions for "partial SPU mode" run in the microprocessor 2652's most privileged mode is necessary to ensure that all its processor control functions can be effectively disabled.

The switch 2663 provides additional protection against tampering by ensuring that the expected control signals occur after an appropriate number of clock cycles. Because the "partial SPU" initialization sequence is entirely deterministic, it is not feasible for malicious software to interfere with it and still retain the same timing characteristics, even if malicious software is running in microprocessor 2652's most privileged mode.

Once in "SPU" mode, switch 2663 may respond to additional indications or signals generated by microprocessor 2652 (e.g., references to specific memory addresses) controlling features of SPU mode. These might include enabling access to external buses 2664 and 2665 so that SPU-protected code could reference external memory or devices. Any attempts by components outside CPU/SPU 2650 to perform operations (e.g., accesses to memory, interrupts, or other control functions) may be prevented by switch 2663 unless they had been explicitly enabled by instructions executed after "SPU" mode is entered.

To leave SPU mode and return to normal operation, the instructions executing in "SPU" mode may provide a specific indication to switch 2663 (e.g., a transfer to a designated memory address). This indication may be recognized by switch 2663 as indicating a return to "normal mode," and it may again restrict access to ROM 532, RAM 534, and all other devices within SPU chip 2660, while re-enabling external buses and control lines 2664, 2665, and 2667. The instructions executed subsequently may restore the CPU state to that which was saved on entry to SPU mode, so that microprocessor 2652 may continue to perform functions in progress when the SPU was invoked.

In an alternate embodiment, the entry into SPU mode may be conditioned on an indication recognized by switch 2663, but the switch may then use a hardware mechanism (e.g., the processor's RESET signal) to reinitialize microprocessor 2562. In such an embodiment, switch 2663 may not implement partial SPU mode, but may instead enter SPU mode directly and ensure that the address from which instructions would be fetched by microprocessor 2652 (specific to microprocessor 2652's architecture) results in accesses to appropriate locations in the SPU memory 532. This could reduce the complexity of the SPU mode entry mechanisms in switch 2663, but could incur an

additional processing cost from using a different reinitialization mechanism for microprocessor 2652.

SPU chip 2660 may be customized to operate in conjunction with a particular commercial microprocessor. In this example, the SPU may be customized to contain at least the specialized "enter SPU mode" instruction sequences to reinitialize the processor's state and, to recognize special indications for SPU control operations. SPU chip 2660 may also be made electrically compatible with microprocessor 2652's external bus interfaces. This compatibility would permit CPU/SPU 2650 to be substituted for microprocessor 2652 without change either to software or hardware elsewhere in a computer system.

In other alternate embodiments, the functions described above for SPU chip 2600, microprocessor 2652, and internal buses 2661, 2662, and 2666 could all be combined within a single integrated circuit package, and/or on a single silicon die. This could reduce packaging complexity and/or simplify establishment of the hardware tamper-resistant barrier 502.

* * * * * *

The hardware configuration of an example of electronic appliance 600 has been described above. The following section describes an example of the software architecture of electronic appliance 600 provided by the preferred embodiment, including the structure and operation of preferred embodiment "Rights Operating System" ("ROS") 602.

Rights Operating System 602

Rights Operating System ("ROS") 602 in the preferred embodiment is a compact, secure, event-driven, services-based, "component" oriented, distributed multiprocessing operating system environment that integrates VDE information security control information, components and protocols with traditional operating system concepts. Like traditional operating systems, ROS 602 provided by the preferred embodiment is a piece of software that manages hardware resources of a computer system and extends management functions to input and/or output devices, including communications devices. Also like traditional operating systems, preferred embodiment ROS 602 provides a coherent set of basic functions and abstraction layers for hiding the differences between, and many of the detailed complexities of, particular hardware implementations. In addition to these characteristics found in many or most operating systems, ROS 602 provides secure VDE transaction management and other

advantageous features not found in other operating systems. The following is a non-exhaustive list of some of the advantageous features provided by ROS 602 in the preferred embodiment:

Standardized interface provides coherent set of basic functions

- simplifies programming
- the same application can run on many different platforms

Event driven

- eases functional decomposition
- extendible
- accommodates state transition and/or process oriented events
- simplifies task management
- simplifies inter-process communications

Services based

- allows simplified and transparent scalability
- simplifies multiprocessor support
- hides machine dependencies
- eases network management and support

Component Based Architecture

- processing based on independently deliverable secure components

- component model of processing control allows different sequential steps that are reconfigurable based on requirements
- components can be added, deleted or modified (subject to permissioning)
- full control information over pre-defined and user-defined application events
- events can be individually controlled with independent executables

Secure

- secure communications
- secure control functions
- secure virtual memory management
- information control structures protected from exposure
- data elements are validated, correlated and access controlled
- components are encrypted and validated independently
- components are tightly correlated to prevent unauthorized use of elements
- control structures and secured executables are validated prior to use to protect against tampering
- integrates security considerations at the I/O level
- provides on-the-fly decryption of information at release time

- enables a secure commercial transaction network
- flexible key management features

Scalaeble

- highly scalaeble across many different platforms
- supports concurrent processing in a multiprocessor environment
- supports multiple cooperating processors
- any number of host or security processors can be supported
- control structures and kernel are easily portable to various host platforms and to different processors within a target platform without recompilation
- supports remote processing
- Remote Procedure Calls may be used for internal OS communications

Highly Integratable

- can be highly integrated with host platforms as an additional operating system layer
- permits non-secure storage of secured components and information using an OS layer "on top of" traditional OS platforms
- can be seamlessly integrated with a host operating system to provide a common usage paradigm for transaction management and content access

- integration may take many forms: operating system layers for desktops (e.g., DOS, Windows, Macintosh); device drivers and operating system interfaces for network services (e.g, Unix and Netware); and dedicated component drivers for "low end" set tops are a few of many examples
- can be integrated in traditional and real time operating systems

Distributed

- provides distribution of control information and reciprocal control information and mechanisms
- supports conditional execution of controlled processes within any VDE node in a distributed, asynchronous arrangement
- controlled delegation of rights in a distributed environment
- supports chains of handling and control
- management environment for distributed, occasionally connected but otherwise asynchronous networked database
- real time and time independent data management
- supports "agent" processes

Transparent

- can be seamlessly integrated into existing operating systems

- can support applications not specifically written to use it

Network friendly

- internal OS structures may use RPCs to distribute processing
- subnets may seamlessly operate as a single node or independently

General Background Regarding Operating Systems

An "operating system" provides a control mechanism for organizing computer system resources that allows programmers to create applications for computer systems more easily. An operating system does this by providing commonly used functions, and by helping to ensure compatibility between different computer hardware and architectures (which may, for example, be manufactured by different vendors). Operating systems also enable computer "peripheral device" manufacturers to far more easily supply compatible equipment to computer manufacturers and users.

Computer systems are usually made up of several different hardware components. These hardware components include, for example:

a central processing unit (CPU) for executing instructions;

an array of main memory cells (e.g., "RAM" or "ROM") for storing instructions for execution and data acted upon or parameterizing those instructions; and

one or more secondary storage devices (e.g., hard disk drive, floppy disk drive, CD-ROM drive, tape reader, card reader, or "flash" memory) organized to reflect named elements (a "file system") for storing images of main memory cells.

Most computer systems also include input/output devices such as keyboards, mice, video systems, printers, scanners and communications devices.

To organize the CPU's execution capabilities with available RAM, ROM and secondary storage devices, and to provide commonly used functions for use by programmers, a piece of software called an "operating system" is usually included with the other components. Typically, this piece of software is designed to begin executing after power is applied to the computer system and hardware diagnostics are completed. Thereafter, all use of the CPU, main memory and secondary memory devices is normally managed by this "operating system" software. Most computer operating systems also typically include a mechanism for extending their management functions

to I/O and other peripheral devices, including commonly used functions associated with these devices.

By managing the CPU, memory and peripheral devices through the operating system, a coherent set of basic functions and abstraction layers for hiding hardware details allows programmers to more easily create sophisticated applications. In addition, managing the computer's hardware resources with an operating system allows many differences in design and equipment requirements between different manufacturers to be hidden. Furthermore, applications can be more easily shared with other computer users who have the same operating system, with significantly less work to support different manufacturers' base hardware and peripheral devices.

ROS 602 is an Operating System Providing Significant Advantages

ROS 602 is an "operating system." It manages the resources of electronic appliance 600, and provides a commonly used set of functions for programmers writing applications 608 for the electronic appliance. ROS 602 in the preferred embodiment manages the hardware (e.g., CPU(s), memory(ies), secure RTC(s), and encrypt/decrypt engines) within SPU 500. ROS may also manage the hardware (e.g., CPU(s) and

memory(ies)) within one or more general purpose processors within electronic appliance 600. ROS 602 also manages other electronic appliance hardware resources, such as peripheral devices attached to an electronic appliance. For example, referring to Figure 7, ROS 602 may manage keyboard 612, display 614, modem 618, disk drive 620, printer 622, scanner 624. ROS 602 may also manage secure database 610 and a storage device (e.g., "secondary storage" 652) used to store secure database 610.

ROS 602 supports multiple processors. ROS 602 in the preferred embodiment supports any number of local and/or remote processors. Supported processors may include at least two types: one or more electronic appliance processors 654, and/or one or more SPUs 500. A host processor CPU 654 may provide storage, database, and communications services. SPU 500 may provide cryptographic and secured process execution services. Diverse control and execution structures supported by ROS 602 may require that processing of control information occur within a controllable execution space -- this controllable execution space may be provided by SPU 500. Additional host and/or SPU processors may increase efficiencies and/or capabilities. ROS 602 may access, coordinate and/or manage further processors remote to an electronic appliance 600 (e.g., via

network or other communications link) to provide additional processor resources and/or capabilities.

ROS 602 is services based. The ROS services provided using a host processor 654 and/or a secure processor (SPU 500) are linked in the preferred embodiment using a "Remote Procedure Call" ("RPC") internal processing request structure. Cooperating processors may request interprocess services using a RPC mechanism, which is minimally time dependent and can be distributed over cooperating processors on a network of hosts. The multi-processor architecture provided by ROS 602 is easily extensible to support any number of host or security processors. This extensibility supports high levels of scalability. Services also allow functions to be implemented differently on different equipment. For example, a small appliance that typically has low levels of usage by one user may implement a database service using very different techniques than a very large appliance with high levels of usage by many users. This is another aspect of scalability.

ROS 602 provides a distributed processing environment. For example, it permits information and control structures to automatically, securely pass between sites as required to fulfill a user's requests. Communications between VDE nodes under the

distributed processing features of ROS 602 may include interprocess service requests as discussed above. ROS 602 supports conditional and/or state dependent execution of controlled processors within any VDE node. The location that the process executes and the control structures used may be locally resident, remotely accessible, or carried along by the process to support execution on a remote system.

ROS 602 provides distribution of control information, including for example the distribution of control structures required to permit "agents" to operate in remote environments. Thus, ROS 602 provides facilities for passing execution and/or information control as part of emerging requirements for "agent" processes.

If desired, ROS 602 may independently distribute control information over very low bandwidth connections that may or may not be "real time" connections. ROS 602 provided by the preferred embodiment is "network friendly," and can be implemented with any level of networking protocol. Some examples include e-mail and direct connection at approximately "Layer 5" of the ISO model.

The ROS 602 distribution process (and the associated auditing of distributed information) is a controlled event that itself uses such control structures. This "reflective" distributed processing mechanism permits ROS 602 to securely distribute rights and permissions in a controlled manner, and effectively restrict the characteristics of use of information content. The controlled delegation of rights in a distributed environment and the secure processing techniques used by ROS 602 to support this approach provide significant advantages.

Certain control mechanisms within ROS 602 are "reciprocal." Reciprocal control mechanisms place one or more control components at one or more locations that interact with one or more components at the same or other locations in a controlled way. For example, a usage control associated with object content at a user's location may have a reciprocal control at a distributor's location that governs distribution of the usage control, auditing of the usage control, and logic to process user requests associated with the usage control. A usage control at a user's location (in addition to controlling one or more aspects of usage) may prepare audits for a distributor and format requests associated with the usage control for processing by a distributor. Processes at either end of a reciprocal control may be further controlled by other processes (e.g., a distributor may be limited

by a budget for the number of usage control mechanisms they may produce). Reciprocal control mechanisms may extend over many sites and many levels (e.g., a creator to a distributor to a user) and may take any relationship into account (e.g., creator/distributor, distributor/user, user/user, user/creator, user/creator/distributor, etc.) Reciprocal control mechanisms have many uses in VDE 100 in representing relationships and agreements in a distributed environment.

ROS 602 is scalable. Many portions of ROS 602 control structures and kernel(s) are easily portable to various host platforms without recompilation. Any control structure may be distributed (or redistributed) if a granting authority permits this type of activity. The executable references within ROS 602 are portable within a target platform. Different instances of ROS 602 may execute the references using different resources. For example, one instance of ROS 602 may perform a task using an SPU 500, while another instance of ROS 602 might perform the same task using a host processing environment running in protected memory that is emulating an SPU in software. ROS 602 control informationis similarly portable; in many cases the event processing structures may be passed between machines and host platforms as easily as between cooperative processors in a single computer. Appliances with different levels of usage

and/or resources available for ROS 602 functions may implement those functions in very different ways. Some services may be omitted entirely if insufficient resources exist. As described elsewhere, ROS 602 "knows" what services are available, and how to proceed based on any given event. Not all events may be processable if resources are missing or inadequate.

ROS 602 is component based. Much of the functionality provided by ROS 602 in the preferred embodiment may be based on "components" that can be securely, independently deliverable, replaceable and capable of being modified (e.g., under appropriately secure conditions and authorizations). Moreover, the "components" may themselves be made of independently deliverable elements. ROS 602 may assemble these elements together (using a construct provided by the preferred embodiment called a "channel") at execution time. For example, a "load module" for execution by SPU 500 may reference one or more "method cores," method parameters and other associated data structures that ROS 602 may collect and assemble together to perform a task such as billing or metering. Different users may have different combinations of elements, and some of the elements may be customizable by users with appropriate authorization. This increases flexibility, allows elements to be reused, and has other advantages.

ROS 602 is highly secure. ROS 602 provides mechanisms to protect information control structures from exposure by end users and conduit hosts. ROS 602 can protect information, VDE control structures and control executables using strong encryption and validation mechanisms. These encryption and validation mechanisms are designed to make them highly resistant to undetected tampering. ROS 602 encrypts information stored on secondary storage device(s) 652 to inhibit tampering. ROS 602 also separately encrypts and validates its various components. ROS 602 correlates control and data structure components to prevent unauthorized use of elements. These features permit ROS 602 to independently distribute elements, and also allows integration of VDE functions 604 with non-secure "other" OS functions 606.

ROS 602 provided by the preferred embodiment extends conventional capabilities such as, for example, Access Control List (ACL) structures, to user and process defined events, including state transitions. ROS 602 may provide full control information over pre-defined and user-defined application events. These control mechanisms include "go/no-go" permissions, and also include optional event-specific executables that permit complete flexibility in the processing and/or controlling of events. This structure permits events to be

individually controlled so that, for example, metering and budgeting may be provided using independent executables. For example, ROS 602 extends ACL structures to control arbitrary granularity of information. Traditional operating systems provide static "go-no go" control mechanisms at a file or resource level; ROS 602 extends the control concept in a general way from the largest to the smallest sub-element using a flexible control structure. ROS 602 can, for example, control the printing of a single paragraph out of a document file.

ROS 602 provided by the preferred embodiment permits secure modification and update of control information governing each component. The control information may be provided in a template format such as method options to an end-user. An end-user may then customize the actual control information used within guidelines provided by a distributor or content creator. Modification and update of existing control structures is preferably also a controllable event subject to auditing and control information.

ROS 602 provided by the preferred embodiment validates control structures and secured executables prior to use. This validation provides assurance that control structures and executables have not been tampered with by end-users. The

validation also permits ROS 602 to securely implement components that include fragments of files and other operating system structures. ROS 602 provided by the preferred embodiment integrates security considerations at the operating system I/O level (which is below the access level), and provides "on-the-fly" decryption of information at release time. These features permit non-secure storage of ROS 602 secured components and information using an OS layer "on top of" traditional operating system platforms.

ROS 602 is highly integratable with host platforms as an additional operating system layer. Thus, ROS 602 may be created by "adding on" to existing operating systems. This involves hooking VDE "add ons" to the host operating system at the device driver and network interface levels. Alternatively, ROS 602 may comprise a wholly new operating system that integrates both VDE functions and other operating system functions.

Indeed, there are at least three general approaches to integrating VDE functions into a new operating system, potentially based on an existing operating system, to create a Rights Operating System 602 including:

(1) Redesign the operating system based on VDE transaction management requirements;

(2) Compile VDE API functions into an existing operating systems; and

(3) Integrate a VDE Interpreter into an existing operating system.

The first approach could be most effectively applied when a new operating system is being designed, or if a significant upgrade to an existing operating system is planned. The transaction management and security requirements provided by the VDE functions could be added to the design requirements list for the design of a new operating system that provides, in an optimally efficient manner, an integration of "traditional" operating system capabilities and VDE capabilities. For example, the engineers responsible for the design of the new version or instance of an operating system would include the requirements of VDE metering/transaction management in addition to other requirements (if any) that they use to form their design approach, specifications, and actual implementations. This approach could lead to a "seamless" integration of VDE functions and capabilities by threading metering/transaction management functionality throughout the system design and implementation.

The second approach would involve taking an existing set of API (Application Programmer Interface) functions, and incorporating references in the operating system code to VDE function calls. This is similar to the way that the current Windows operating system is integrated with DOS, wherein DOS serves as both the launch point and as a significant portion of the kernel underpinning of the Windows operating system. This approach would be also provide a high degree of "seamless" integration (although not quite as "seamless" as the first approach). The benefits of this approach include the possibility that the incorporation of metering/transaction management functionality into the new version or instance of an operating system may be accomplished with lower cost (by making use of the existing code embodied in an API, and also using the design implications of the API functional approach to influence the design of the elements into which the metering/transaction management functionality is incorporated).

The third approach is distinct from the first two in that it does not incorporate VDE functionality associated with metering/transaction management and data security directly into the operating system code, but instead adds a new generalized capability to the operating system for executing metering/transaction management functionality. In this case, an

interpreter including metering/transaction management functions would be integrated with other operating system code in a "stand alone" mode. This interpreter might take scripts or other inputs to determine what metering/transaction management functions should be performed, and in what order and under which circumstances or conditions they should be performed.

Instead of (or in addition to) integrating VDE functions into/with an electronic appliance operating system, it would be possible to provide certain VDE functionality available as an application running on a conventional operating system.

ROS Software Architecture

Figure 10 is a block diagram of one example of a software structure/architecture for Rights Operating System ("ROS") 602 provided by the preferred embodiment. In this example, ROS 602 includes an operating system ("OS") "core" 679, a user Application Program Interface ("API") 682, a "redirector" 684, an "intercept" 692, a User Notification/Exception Interface 686, and a file system 687. ROS 602 in this example also includes one or more Host Event Processing Environments ("HPEs") 655 and/or one or more Secure Event Processing Environments ("SPEs") 503

(these environments may be generically referred to as "Protected Processing Environments" 650).

HPE(s) 655 and SPE(s) 503 are self-contained computing and processing environments that may include their own operating system kernel 688 including code and data processing resources. A given electronic appliance 600 may include any number of SPE(s) 503 and/or any number of HPE(s) 655. HPE(s) 655 and SPE(s) 503 may process information in a secure way, and provide secure processing support for ROS 602. For example, they may each perform secure processing based on one or more VDE component assemblies 690, and they may each offer secure processing services to OS kernel 680.

In the preferred embodiment, SPE 503 is a secure processing environment provided at least in part by an SPU 500. Thus, SPU 500 provides the hardware tamper-resistant barrier 503 surrounding SPE 503. SPE 503 provided by the preferred embodiment is preferably:

- small and compact
- loadable into resource constrained environments such as for example minimally configured SPUs 500
- dynamically updatable

- extensible by authorized users
- integratable into object or procedural environments
- secure.

In the preferred embodiment, HPE 655 is a secure processing environment supported by a processor other than an SPU, such as for example an electronic appliance CPU 654 general-purpose microprocessor or other processing system or device. In the preferred embodiment, HPE 655 may be considered to "emulate" an SPU 500 in the sense that it may use software to provide some or all of the processing resources provided in hardware and/or firmware by an SPU. HPE 655 in one preferred embodiment of the present invention is full-featured and fully compatible with SPE 503—that is, HPE 655 can handle each and every service call SPE 503 can handle such that the SPE and the HPE are "plug compatible" from an outside interface standpoint (with the exception that the HPE may not provide as much security as the SPE).

HPEs 655 may be provided in two types: secure and not secure. For example, it may be desirable to provide non-secure versions of HPE 655 to allow electronic appliance 600 to efficiently run non-sensitive VDE tasks using the full resources

of a fast general purpose processor or computer. Such non-secure versions of HPE 655 may run under supervision of an instance of ROS 602 that also includes an SPE 503. In this way, ROS 602 may run all secure processes within SPE 503, and only use HPE 655 for processes that do not require security but that may require (or run more efficiently) under potentially greater resources provided by a general purpose computer or processor supporting HPE 655. Non-secure and secure HPE 655 may operate together with a secure SPE 503.

HPEs 655 may (as shown in Figure 10) be provided with a software-based tamper resistant barrier 674 that makes them more secure. Such a software-based tamper resistant barrier 674 may be created by software executing on general-purpose CPU 654. Such a "secure" HPE 655 can be used by ROS 602 to execute processes that, while still needing security, may not require the degree of security provided by SPU 500. This can be especially beneficial in architectures providing both an SPE 503 and an HPE 655. The SPU 502 may be used to perform all truly secure processing, whereas one or more HPEs 655 may be used to provide additional secure (albeit possibly less secure than the SPE) processing using host processor or other general purpose resources that may be available within an electronic appliance 600. Any service may be provided by such a secure HPE 655. In

the preferred embodiment, certain aspects of "channel processing" appears to be a candidate that could be readily exported from SPE 503 to HPE 655.

The software-based tamper resistant barrier 674 provided by HPE 655 may be provided, for example, by: introducing time checks and/or code modifications to complicate the process of stepping through code comprising a portion of kernel 688a and/or a portion of component assemblies 690 using a debugger; using a map of defects on a storage device (e.g., a hard disk, memory card, etc.) to form internal test values to impede moving and/or copying HPE 655 to other electronic appliances 600; using kernel code that contains false branches and other complications in flow of control to disguise internal processes to some degree from disassembly or other efforts to discover details of processes; using "self-generating" code (based on the output of a co-sine transform, for example) such that detailed and/or complete instruction sequences are not stored explicitly on storage devices and/or in active memory but rather are generated as needed; using code that "shuffles" memory locations used for data values based on operational parameters to complicate efforts to manipulate such values; using any software and/or hardware memory management resources of electronic appliance 600 to "protect" the operation of HPE 655 from other processes,

functions, etc. Although such a software-based tamper resistant barrier 674 may provide a fair degree of security, it typically will not be as secure as the hardware-based tamper resistant barrier 502 provided (at least in part) by SPU 500. Because security may be better/more effectively enforced with the assistance of hardware security features such as those provided by SPU 500 (and because of other factors such as increased performance provided by special purpose circuitry within SPU 500), at least one SPE 503 is preferred for many or most higher security applications. However, in applications where lesser security can be tolerated and/or the cost of an SPU 500 cannot be tolerated, the SPE 503 may be omitted and all secure processing may instead be performed by one or more secure HPEs 655 executing on general-purpose CPUs 654. Some VDE processes may not be allowed to proceed on reduced-security electronic appliances of this type if insufficient security is provided for the particular process involved.

Only those processes that execute completely within SPEs 503 (and in some cases, HPEs 655) may be considered to be truly secure. Memory and other resources external to SPE 503 and HPEs 655 used to store and/or process code and/or data to be used in secure processes should only receive and handle that information in encrypted form unless SPE 503/HPE 655 can

protect secure process code and/or data from non-secure processes.

OS "core" 679 in the preferred embodiment includes a kernel 680, an RPC manager 732, and an "object switch" 734. API 682, HPE 655 and SPE 503 may communicate "event" messages with one another via OS "core" 679. They may also communicate messages directly with one another without messages going through OS "core" 679.

Kernel 680 may manage the hardware of an electronic appliance 600. For example, it may provide appropriate drivers and hardware managers for interacting with input/output and/or peripheral devices such as keyboard 612, display 614, other devices such as a "mouse" pointing device and speech recognizer 613, modem 618, printer 622, and an adapter for network 672. Kernel 680 may also be responsible for initially loading the remainder of ROS 602, and may manage the various ROS tasks (and associated underlying hardware resources) during execution. OS kernel 680 may also manage and access secure database 610 and file system 687. OS kernel 680 also provides execution services for applications 608a(1), 608a(2), etc. and other applications.

RPC manager 732 performs messaging routing and resource management/integration for ROS 680. It receives and routes "calls" from/to API 682, HPE 655 and SPE 503, for example.

Object switch 734 may manage construction, deconstruction and other manipulation of VDE objects 300.

User Notification/Exception Interface 686 in the preferred embodiment (which may be considered part of API 682 or another application coupled to the API) provides "pop up" windows/displays on display 614. This allows ROS 602 to communicate directly with a user without having to pass information to be communicated through applications 608. For applications that are not "VDE aware," user notification/exception interface 686 may provide communications between ROS 602 and the user.

API 682 in the preferred embodiment provides a standardized, documented software interface to applications 608. In part, API 682 may translate operating system "calls" generated by applications 608 into Remote Procedure Calls ("RPCs") specifying "events." RPC manager 732 may route these RPCs to kernel 680 or elsewhere (e.g., to HPE(s) 655 and/or

SPE(s) 503, or to remote electronic appliances 600, processors, or VDE participants) for processing. The API 682 may also service RPC requests by passing them to applications 608 that register to receive and process specific requests.

API 682 provides an "Applications Programming Interface" that is preferably standardized and documented. It provides a concise set of function calls an application program can use to access services provided by ROS 602. In at least one preferred example, API 682 will include two parts: an application program interface to VDE functions 604; and an application program interface to other OS functions 606. These parts may be interwoven into the same software, or they may be provided as two or more discrete pieces of software (for example).

Some applications, such as application 608a(1) shown in Figure 11, may be "VDE aware" and may therefore directly access both of these parts of API 682. Figure 11A shows an example of this. A "VDE aware" application may, for example, include explicit calls to ROS 602 requesting the creation of new VDE objects 300, metering usage of VDE objects, storing information in VDE-protected form, etc. Thus, a "VDE aware" application can initiate (and, in some examples, enhance and/or extend) VDE functionality provided by ROS 602. In addition,

"VDE aware" applications may provide a more direct interface between a user and ROS 602 (e.g., by suppressing or otherwise dispensing with "pop up" displays otherwise provided by user notification/exception interface 686 and instead providing a more "seamless" interface that integrates application and ROS messages).

Other applications, such as application 608b shown in Figure 11B, may not be "VDE Aware" and therefore may not "know" how to directly access an interface to VDE functions 604 provided by API 682. To provide for this, ROS 602 may include a "redirector" 684 that allows such "non-VDE aware" applications 608(b) to access VDE objects 300 and functions 604. Redirector 684, in the preferred embodiment, translates OS calls directed to the "other OS functions" 606 into calls to the "VDE functions" 604. As one simple example, redirector 684 may intercept a "file open" call from application 608(b), determine whether the file to be opened is contained within a VDE container 300, and if it is, generate appropriate VDE function call(s) to file system 687 to open the VDE container (and potentially generate events to HPE 655 and/or SPE 503 to determine the name(s) of file(s) that may be stored in a VDE object 300, establish a control structure associated with a VDE object 300, perform a registration for a VDE object 300, etc.). Without redirector 684 in this example, a

non-VDE aware application such as 608b could access only the part of API 682 that provides an interface to other OS functions 606, and therefore could not access any VDE functions.

This "translation" feature of redirector 684 provides "transparency." It allows VDE functions to be provided to the application 608(b) in a "transparent" way without requiring the application to become involved in the complexity and details associated with generating the one or more calls to VDE functions 604. This aspect of the "transparency" features of ROS 602 has at least two important advantages:

(a) it allows applications not written specifically for VDE functions 604 ("non-VDE aware applications") to nevertheless access critical VDE functions; and

(b) it reduces the complexity of the interface between an application and ROS 602.

Since the second advantage (reducing complexity) makes it easier for an application creator to produce applications, even "VDE aware" applications 608a(2) may be designed so that some calls invoking VDE functions 604 are requested at the level of an "other OS functions" call and then "translated" by redirector 684 into a VDE function call (in this sense, redirector 684 may be considered a part of API 682). Figure 11C shows an example of

this. Other calls invoking VDE functions 604 may be passed directly without translation by redirector 684.

Referring again to Figure 10, ROS 620 may also include an "interceptor" 692 that transmits and/or receives one or more real time data feeds 694 (this may be provided over cable(s) 628 for example), and routes one or more such data feeds appropriately while providing "translation" functions for real time data sent and/or received by electronic appliance 600 to allow "transparency" for this type of information analogous to the transparency provided by redirector 684 (and/or it may generate one or more real time data feeds).

Secure ROS Components and Component Assemblies

As discussed above, ROS 602 in the preferred embodiment is a component-based architecture. ROS VDE functions 604 may be based on segmented, independently loadable executable "component assemblies" 690. These component assemblies 690 are independently securely deliverable. The component assemblies 690 provided by the preferred embodiment comprise code and data elements that are themselves independently deliverable. Thus, each component assembly 690 provided by the preferred embodiment is comprised of independently securely deliverable elements which may be communicated using VDE

secure communication techniques, between VDE secure subsystems.

These component assemblies 690 are the basic functional unit provided by ROS 602. The component assemblies 690 are executed to perform operating system or application tasks. Thus, some component assemblies 690 may be considered to be part of the ROS operating system 602, while other component assemblies may be considered to be "applications" that run under the support of the operating system. As with any system incorporating "applications" and "operating systems," the boundary between these aspects of an overall system can be ambiguous. For example, commonly used "application" functions (such as determining the structure and/or other attributes of a content container) may be incorporated into an operating system. Furthermore, "operating system" functions (such as task management, or memory allocation) may be modified and/or replaced by an application. A common thread in the preferred embodiment's ROS 602 is that component assemblies 690 provide functions needed for a user to fulfill her intended activities, some of which may be "application-like" and some of which may be "operating system-like."

Components 690 are preferably designed to be easily separable and individually loadable. ROS 602 assembles these elements together into an executable component assembly 690 prior to loading and executing the component assembly (e.g., in a secure operating environment such as SPE 503 and/or HPE 655). ROS 602 provides an element identification and referencing mechanism that includes information necessary to automatically assemble elements into a component assembly 690 in a secure manner prior to, and/or during, execution.

ROS 602 application structures and control parameters used to form component assemblies 690 can be provided by different parties. Because the components forming component assemblies 690 are independently securely deliverable, they may be delivered at different times and/or by different parties ("delivery" may take place within a local VDE secure subsystem, that is submission through the use of such a secure subsystem of control information by a chain of content control information handling participant for the preparation of a modified control information set constitutes independent, secure delivery). For example, a content creator can produce a ROS 602 application that defines the circumstances required for licensing content contained within a VDE object 300. This application may reference structures provided by other parties. Such references

might, for example, take the form of a control path that uses content creator structures to meter user activities; and structures created/owned by a financial provider to handle financial parts of a content distribution transaction (e.g., defining a credit budget that must be present in a control structure to establish creditworthiness, audit processes which must be performed by the licensee, etc.). As another example, a distributor may give one user more favorable pricing than another user by delivering different data elements defining pricing to different users. This attribute of supporting multiple party securely, independently deliverable control information is fundamental to enabling electronic commerce, that is, defining of a content and/or appliance control information set that represents the requirements of a collection of independent parties such as content creators, other content providers, financial service providers, and/or users.

In the preferred embodiment, ROS 602 assembles securely independently deliverable elements into a component assembly 690 based in part on context parameters (e.g., object, user). Thus, for example, ROS 602 may securely assemble different elements together to form different component assemblies 690 for different users performing the same task on the same VDE object 300. Similarly, ROS 602 may assemble differing element

sets which may include, that is reuse, one or more of the same components to form different component assemblies 690 for the same user performing the same task on different VDE objects 300.

The component assembly organization provided by ROS 602 is "recursive" in that a component assembly 690 may comprise one or more component "subassemblies" that are themselves independently loadable and executable component assemblies 690. These component "subassemblies" may, in turn, be made of one or more component "sub-sub-assemblies." In the general case, a component assembly 690 may include N levels of component subassemblies.

Thus, for example, a component assembly 690(k) that may includes a component subassembly 690(k + 1). Component subassembly 690(k + 1), in turn, may include a component sub-sub-assembly 690(3), ... and so on to N-level subassembly 690(k + N). The ability of ROS 602 to build component assemblies 690 out of other component assemblies provides great advantages in terms of, for example, code/data reusability, and the ability to allow different parties to manage different parts of an overall component.

Each component assembly 690 in the preferred embodiment is made of distinct components. Figures 11D-11H are abstract depictions of various distinct components that may be assembled to form a component assembly 690(k) showing Figure 11I. These same components can be combined in different ways (e.g., with more or less components) to form different component assemblies 690 providing completely different functional behavior. Figure 11J is an abstract depiction of the same components being put together in a different way (e.g., with additional components) to form a different component assembly 690(j). The component assemblies 690(k) and 690(j) each include a common feature 691 that interlocks with a "channel" 594 defined by ROS 602. This "channel" 594 assembles component assemblies 690 and interfaces them with the (rest of) ROS 602.

ROS 602 generates component assemblies 690 in a secure manner. As shown graphically in Figures 11I and 11J, the different elements comprising a component assembly 690 may be "interlocking" in the sense that they can only go together in ways that are intended by the VDE participants who created the elements and/or specified the component assemblies. ROS 602 includes security protections that can prevent an unauthorized person from modifying elements, and also prevent an

unauthorized person from substituting elements. One can picture an unauthorized person making a new element having the same "shape" as the one of the elements shown in Figures 11D-11H, and then attempting to substitute the new element in place of the original element. Suppose one of the elements shown in Figure 11H establishes the price for using content within a VDE object 300. If an unauthorized person could substitute her own "price" element for the price element intended by the VDE content distributor, then the person could establish a price of zero instead of the price the content distributor intended to charge. Similarly, if the element establishes an electronic credit card, then an ability to substitute a different element could have disastrous consequences in terms of allowing a person to charge her usage to someone else's (or a non-existent) credit card. These are merely a few simple examples demonstrating the importance of ROS 602 ensuring that certain component assemblies 690 are formed in a secure manner. ROS 602 provides a wide range of protections against a wide range of "threats" to the secure handling and execution of component assemblies 690.

In the preferred embodiment, ROS 602 assembles component assemblies 690 based on the following types of elements:

Permissions Records ("PERC"s) 808;

Method "Cores" 1000;

Load Modules 1100;

Data Elements (e.g., User Data Elements ("UDEs") 1200 and Method Data Elements ("MDEs") 1202); and

Other component assemblies 690.

Briefly, a PERC 808 provided by the preferred embodiment is a record corresponding to a VDE object 300 that identifies to ROS 602, among other things, the elements ROS is to assemble together to form a component assembly 690. Thus PERC 808 in effect contains a "list of assembly instructions" or a "plan" specifying what elements ROS 602 is to assemble together into a component assembly and how the elements are to be connected together. PERC 808 may itself contain data or other elements that are to become part of the component assembly 690.

The PERC 808 may reference one or more method "cores" 1000'. A method core 1000' may define a basic "method" 1000 (e.g., "control," "billing," "metering," etc.)

In the preferred embodiment, a "method" 1000 is a collection of basic instructions, and information related to basic instructions, that provides context, data, requirements, and/or

relationships for use in performing, and/or preparing to perform, basic instructions in relation to the operation of one or more electronic appliances 600. Basic instructions may be comprised of, for example:

- machine code of the type commonly used in the programming of computers; pseudo-code for use by an interpreter or other instruction processing program operating on a computer;
- a sequence of electronically represented logical operations for use with an electronic appliance 600;
- or other electronic representations of instructions, source code, object code, and/or pseudo code as those terms are commonly understood in the arts.

Information relating to said basic instructions may comprise, for example, data associated intrinsically with basic instructions such as for example, an identifier for the combined basic instructions and intrinsic data, addresses, constants, and/or the like. The information may also, for example, include one or more of the following:

- information that identifies associated basic instructions and said intrinsic data for access, correlation and/or validation purposes;
- required and/or optional parameters for use with basic instructions and said intrinsic data;
- information defining relationships to other methods;
- data elements that may comprise data values, fields of information, and/or the like;
- information specifying and/or defining relationships among data elements, basic instructions and/or intrinsic data;
- information specifying relationships to external data elements;
- information specifying relationships between and among internal and external data elements, methods, and/or the like, if any exist; and

- additional information required in the operation of basic instructions and intrinsic data to complete, or attempt to complete, a purpose intended by a user of a method, where required, including additional instructions and/or intrinsic data.

Such information associated with a method may be stored, in part or whole, separately from basic instructions and intrinsic data. When these components are stored separately, a method may nevertheless include and encompass the other information and one or more sets of basic instructions and intrinsic data (the latter being included because of said other information's reference to one or more sets of basic instructions and intrinsic data), whether or not said one or more sets of basic instructions and intrinsic data are accessible at any given point in time.

Method core 1000' may be parameterized by an "event code" to permit it to respond to different events in different ways. For example, a METER method may respond to a "use" event by storing usage information in a meter data structure. The same METER method may respond to an "administrative" event by reporting the meter data structure to a VDE clearinghouse or other VDE participant.

In the preferred embodiment, method core 1000' may "contain," either explicitly or by reference, one or more "load modules" 1100 and one or more data elements (UDEs 1200, MDEs 1202). In the preferred embodiment, a "load module" 1100 is a portion of a method that reflects basic instructions and intrinsic data. Load modules 1100 in the preferred embodiment

contain executable code, and may also contain data elements ("DTDs" 1108) associated with the executable code. In the preferred embodiment, load modules 1100 supply the program instructions that are actually "executed" by hardware to perform the process defined by the method. Load modules 1100 may contain or reference other load modules.

Load modules 1100 in the preferred embodiment are modular and "code pure" so that individual load modules may be reenterable and reusable. In order for components 690 to be dynamically updatable, they may be individually addressable within a global public name space. In view of these design goals, load modules 1100 are preferably small, code (and code-like) pure modules that are individually named and addressable. A single method may provide different load modules 1100 that perform the same or similar functions on different platforms, thereby making the method scalable and/or portable across a wide range of different electronic appliances.

UDEs 1200 and MDEs 1202 may store data for input to or output from executable component assembly 690 (or data describing such inputs and/or outputs). In the preferred embodiment, UDEs 1200 may be user dependent, whereas MDEs 1202 may be user independent.

The component assembly example 690(k) shown in Figure 11E comprises a method core 1000', UDEs 1200a & 1200b, an MDE 1202, load modules 1100a-1100d, and a further component assembly 690(k+1). As mentioned above, a PERC 808(k) defines, among other things, the "assembly instructions" for component assembly 690(k), and may contain or reference parts of some or all of the components that are to be assembled to create a component assembly.

One of the load modules 1100b shown in this example is itself comprised of plural load modules 1100c, 1100d. Some of the load modules (e.g., 1100a, 1100d) in this example include one or more "DTD" data elements 1108 (e.g., 1108a, 1108b). "DTD" data elements 1108 may be used, for example, to inform load module 1100a of the data elements included in MDE 1202 and/or UDEs 1200a, 1200b. Furthermore, DTDs 1108 may be used as an aspect of forming a portion of an application used to inform a user as to the information required and/or manipulated by one or more load modules 1100, or other component elements. Such an application program may also include functions for creating and/or manipulating UDE(s) 1200, MDE(s) 1202, or other component elements, subassemblies, etc.

Components within component assemblies 690 may be "reused" to form different component assemblies. As mentioned above, figure 11F is an abstract depiction of one example of the same components used for assembling component assembly 690(k) to be reused (e.g., with some additional components specified by a different set of "assembly instructions" provided in a different PERC 808(l)) to form a different component assembly 690(l). Even though component assembly 690(l) is formed from some of the same components used to form component assembly 690(k), these two component assemblies may perform completely different processes in complete different ways.

As mentioned above, ROS 602 provides several layers of security to ensure the security of component assemblies 690. One important security layer involves ensuring that certain component assemblies 690 are formed, loaded and executed only in secure execution space such as provided within an SPU 500. Components 690 and/or elements comprising them may be stored on external media encrypted using local SPU 500 generated and/or distributor provided keys.

ROS 602 also provides a tagging and sequencing scheme that may be used within the loadable component assemblies 690 to detect tampering by substitution. Each element comprising a

component assembly 690 may be loaded into an SPU 500, decrypted using encrypt/decrypt engine 522, and then tested/compared to ensure that the proper element has been loaded. Several independent comparisons may be used to ensure there has been no unauthorized substitution. For example, the public and private copies of the element ID may be compared to ensure that they are the same, thereby preventing gross substitution of elements. In addition, a validation/correlation tag stored under the encrypted layer of the loadable element may be compared to make sure it matches one or more tags provided by a requesting process. This prevents unauthorized use of information. As a third protection, a device assigned tag (e.g., a sequence number) stored under an encryption layer of a loadable element may be checked to make sure it matches a corresponding tag value expected by SPU 500. This prevents substitution of older elements. Validation/correlation tags are typically passed only in secure wrappers to prevent plaintext exposure of this information outside of SPU 500.

The secure component based architecture of ROS 602 has important advantages. For example, it accommodates limited resource execution environments such as provided by a lower cost SPU 500. It also provides an extremely high level of configurability. In fact, ROS 602 will accommodate an almost

unlimited diversity of content types, content provider objectives, transaction types and client requirements. In addition, the ability to dynamically assemble independently deliverable components at execution time based on particular objects and users provides a high degree of flexibility, and facilitates or enables a distributed database, processing, and execution environment.

One aspect of an advantage of the component-based architecture provided by ROS 602 relates to the ability to "stage" functionality and capabilities over time. As designed, implementation of ROS 602 is a finite task. Aspects of its wealth of functionality can remain unexploited until market realities dictate the implementation of corresponding VDE application functionality. As a result, initial product implementation investment and complexity may be limited. The process of "surfacing" the full range of capabilities provided by ROS 602 in terms of authoring, administrative, and artificial intelligence applications may take place over time. Moreover, already-designed functionality of ROS 602 may be changed or enhanced at any time to adapt to changing needs or requirements.

More Detailed Discussion of Rights Operating System 602 Architecture

Figure 12 shows an example of a detailed architecture of ROS 602 shown in Figure 10. ROS 602 may include a file system 687 that includes a commercial database manager 730 and external object repositories 728. Commercial database manager 730 may maintain secure database 610. Object repository 728 may store, provide access to, and/or maintain VDE objects 300.

Figure 12 also shows that ROS 602 may provide one or more SPEs 503 and/or one or more HPEs 655. As discussed above, HPE 655 may "emulate" an SPU 500 device, and such HPEs 655 may be integrated in lieu of (or in addition to) physical SPUs 500 for systems that need higher throughput. Some security may be lost since HPEs 655 are typically protected by operating system security and may not provide truly secure processing. Thus, in the preferred embodiment, for high security applications at least, all secure processing should take place within an SPE 503 having an execution space within a physical SPU 500 rather than a HPE 655 using software operating elsewhere in electronic appliance 600.

As mentioned above, three basic components of ROS 602 are a kernel 680, a Remote Procedure Call (RPC) manager 732

and an object switch 734. These components, and the way they interact with other portions of ROS 602, will be discussed below.

Kernel 680

Kernel 680 manages the basic hardware resources of electronic appliance 600, and controls the basic tasking provided by ROS 602. Kernel 680 in the preferred embodiment may include a memory manager 680a, a task manager 680b, and an I/O manager 680c. Task manager 680b may initiate and/or manage initiation of executable tasks and schedule them to be executed by a processor on which ROS 602 runs (e.g., CPU 654 shown in Figure 8). For example, Task manager 680b may include or be associated with a "bootstrap loader" that loads other parts of ROS 602. Task manager 680b may manage all tasking related to ROS 602, including tasks associated with application program(s) 608. Memory manager 680a may manage allocation, deallocation, sharing and/or use of memory (e.g., RAM 656 shown in Figure 8) of electronic appliance 600, and may for example provide virtual memory capabilities as required by an electronic appliance and/or associated application(s). I/O manager 680c may manage all input to and output from ROS 602, and may interact with drivers and other hardware managers that provide communications and interactivity with physical devices.

RPC Manager 732

ROS 602 in a preferred embodiment is designed around a "services based" Remote Procedure Call architecture/interface. All functions performed by ROS 602 may use this common interface to request services and share information. For example, SPE(s) 503 provide processing for one or more RPC based services. In addition to supporting SPUs 500, the RPC interface permits the dynamic integration of external services and provides an array of configuration options using existing operating system components. ROS 602 also communicates with external services through the RPC interface to seamlessly provide distributed and/or remote processing. In smaller scale instances of ROS 602, a simpler message passing IPC protocol may be used to conserve resources. This may limit the configurability of ROS 602 services, but this possible limitation may be acceptable in some electronic appliances.

The RPC structure allows services to be called/requested without the calling process having to know or specify where the service is physically provided, what system or device will service the request, or how the service request will be fulfilled. This feature supports families of services that may be scaled and/or customized for specific applications. Service requests can be forwarded and serviced by different processors and/or different

sites as easily as they can be forwarded and serviced by a local service system. Since the same RPC interface is used by ROS 602 in the preferred embodiment to request services within and outside of the operating system, a request for distributed and/or remote processing incurs substantially no additional operating system overhead. Remote processing is easily and simply integrated as part of the same service calls used by ROS 602 for requesting local-based services. In addition, the use of a standard RPC interface ("RSI") allows ROS 602 to be modularized, with the different modules presenting a standardized interface to the remainder of the operating system. Such modularization and standardized interfacing permits different vendors/operating system programmers to create different portions of the operating system independently, and also allows the functionality of ROS 602 to be flexibly updated and/or changed based on different requirements and/or platforms.

RPC manager 732 manages the RPC interface. It receives service requests in the form of one or more "Remote Procedure Calls" (RPCs) from a service requestor, and routes the service requests to a service provider(s) that can service the request. For example, when rights operating system 602 receives a request from a user application via user API 682, RPC manager 732 may

route the service request to an appropriate service through the "RPC service interface" ("RSI"). The RSI is an interface between RPC manager 732, service requestors, and a resource that will accept and service requests.

The RPC interface (RSI) is used for several major ROS 602 subsystems in the preferred embodiment.

RPC services provided by ROS 602 in the preferred embodiment are divided into subservices, i.e., individual instances of a specific service each of which may be tracked individually by the RPC manager 732. This mechanism permits multiple instances of a specific service on higher throughput systems while maintaining a common interface across a spectrum of implementations. The subservice concept extends to supporting multiple processors, multiple SPEs 503, multiple HPEs 655, and multiple communications services.

The preferred embodiment ROS 602 provides the following RPC based service providers/requestors (each of which have an RPC interface or "RSI" that communicates with RPC manager 732):

SPE device driver 736 (this SPE device driver is connected to an SPE 503 in the preferred embodiment);

HPE Device Driver 738 (this HPE device driver is connected to an HPE 738 in the preferred embodiment);

Notification Service 740 (this notification service is connected to user notification interface 686 in the preferred embodiment);

API Service 742 (this API service is connected to user API 682 in the preferred embodiment;

Redirector 684;

Secure Database (File) Manager 744 (this secure database or file manager 744 may connect to and interact with commercial database manager 730 and secure files 610 through a cache manager 746, a database interface 748, and a database driver 750);

Name Services Manager 752;

Outgoing Administrative Objects Manager 754;

Incoming Administrative Objects Manager 756;

a Gateway 734 to object switch 734 (this is a path used to allow direct communication between RPC manager 732 and Object Switch 734); and

Communications Manager 776.

The types of services provided by HPE 655, SPE 503, User Notification 686, API 742 and Redirector 684 have already been

described above. Here is a brief description of the type(s) of services provided by OS resources 744, 752, 754, 756 and 776:

Secure Database Manager 744 services requests for access to secure database 610;

Name Services Manager 752 services requests relating to user, host, or service identification;

Outgoing Administrative Objects Manager 754 services requests relating to outgoing administrative objects;

Incoming Administrative Objects Manager 756 services requests relating to incoming administrative objects; and

Communications Manager 776 services requests relating to communications between electronic appliance 600 and the outside world.

Object Switch 734

Object switch 734 handles, controls and communicates (both locally and remotely) VDE objects 300. In the preferred embodiment, the object switch may include the following elements:

a stream router 758;

a real time stream interface(s) 760 (which may be connected to real time data feed(s) 694);

a time dependent stream interface(s) 762;

a intercept 692;

a container manager 764;

one or more routing tables 766; and

buffering/storage 768.

Stream router 758 routes to/from "real time" and "time independent" data streams handled respectively by real time stream interface(s) 760 and time dependent stream interface(s) 762. Intercept 692 intercepts I/O requests that involve real-time information streams such as, for example, real time feed 694. The routing performed by stream router 758 may be determined by routing tables 766. Buffering/storage 768 provides temporary store-and-forward, buffering and related services. Container manager 764 may (typically in conjunction with SPE 503) perform processes on VDE objects 300 such as constructing, deconstructing, and locating portions of objects.

Object switch 734 communicates through an Object Switch Interface ("OSI") with other parts of ROS 602. The Object Switch Interface may resemble, for example, the interface for a Unix socket in the preferred embodiment. Each of the "OSI" interfaces shown in Figure 12 have the ability to communicate with object switch 734.

ROS 602 includes the following object switch service providers/resources (each of which can communicate with the object switch 734 through an "OSI"):

Outgoing Administrative Objects Manager 754;

Incoming Administrative Objects Manager 756;

Gateway 734 (which may translate RPC calls into object switch calls and vice versa so RPC manager 732 may communicate with object switch 734 or any other element having an OSI to, for example, provide and/or request services);

External Services Manager 772;

Object Submittal Manager 774; and

Communications Manager 776.

Briefly,

Object Repository Manager 770 provides services relating to access to object repository 728;

External Services Manager 772 provides services relating to requesting and receiving services externally, such as from a network resource or another site;

Object Submittal Manager 774 provides services relating to how a user application may interact with object switch 734 (since the object submittal manager

provides an interface to an application program 608, it could be considered part of user API 682); and Communications Manager 776 provides services relating to communicating with the outside world.

In the preferred embodiment, communications manager 776 may include a network manager 780 and a mail gateway (manager) 782. Mail gateway 782 may include one or more mail filters 784 to, for example, automatically route VDE related electronic mail between object switch 734 and the outside world electronic mail services. External Services Manager 772 may interface to communications manager 776 through a Service Transport Layer 786. Service Transport Layer 786a may enable External Services Manager 772 to communicate with external computers and systems using various protocols managed using the service transport layer 786.

The characteristics of and interfaces to the various subsystems of ROS 680 shown in Figure 12 are described in more detail below.

RPC Manager 732 and Its RPC Services Interface

As discussed above, the basic system services provided by ROS 602 are invoked by using an RPC service interface (RSI).

This RPC service interface provides a generic, standardized interface for different services systems and subsystems provided by ROS 602.

RPC Manager 732 routes RPCs requesting services to an appropriate RPC service interface. In the preferred embodiment, upon receiving an RPC call, RPC manager 732 determines one or more service managers that are to service the request. RPC manager 732 then routes a service request to the appropriate service(s) (via a RSI associated with a service) for action by the appropriate service manager(s).

For example, if a SPE 503 is to service a request, the RPC Manager 732 routes the request to RSI 736a, which passes the request on to SPE device driver 736 for forwarding to the SPE. Similarly, if HPE 655 is to service the request, RPC Manager 732 routes the request to RSI 738a for forwarding to a HPE. In one preferred embodiment, SPE 503 and HPE 655 may perform essentially the same services so that RSIs 736a, 738a are different instances of the same RSI. Once a service request has been received by SPE 503 (or HPE 655), the SPE (or HPE) typically dispatches the request internally using its own internal RPC manager (as will be discussed shortly). Processes within SPEs 503 and HPEs 655 can also generate RPC requests. These

requests may be processed internally by a SPE/HPE, or if not internally serviceable, passed out of the SPE/HPE for dispatch by RPC Manager 732.

Remote (and local) procedure calls may be dispatched by a RPC Manager 732 using an "RPC Services Table." An RPC Services Table describes where requests for specific services are to be routed for processing. Each row of an RPC Services Table in the preferred embodiment contains a services ID, the location of the service, and an address to which control will be passed to service a request. An RPC Services Table may also include control information that indicates which instance of the RPC dispatcher controls the service. Both RPC Manager 732 and any attached SPEs 503 and HPEs 655 may have symmetric copies of the RPC Services Table. If an RPC service is not found in the RPC services tables, it is either rejected or passed to external services manager 772 for remote servicing.

Assuming RPC manager 732 finds a row corresponding to the request in an RPC Services Table, it may dispatch the request to an appropriate RSI. The receiving RSI accepts a request from the RPC manager 732 (which may have looked up the request in an RPC service table), and processes that request

in accordance with internal priorities associated with the specific service.

In the preferred embodiment, RPC Service Interface(s) supported by RPC Manager 732 may be standardized and published to support add-on service modules developed by third party vendors, and to facilitate scalability by making it easier to program ROS 602. The preferred embodiment RSI closely follows the DOS and Unix device driver models for block devices so that common code may be developed for many platforms with minimum effort. An example of one possible set of common entry points are listed below in the table.

Interface call	Description
SVC_LOAD	Load a service manager and return its status.
SVC_UNLOAD	Unload a service manager.
SVC_MOUNT	Mount (load) a dynamically loaded subservice and return its status.
SVC_UNMOUNT	Unmount (unload) a dynamically loaded subservice.
SVC_OPEN	Open a mounted subservice.
SVC_CLOSE	Close a mounted subservice.
SVC_READ	Read a block from an opened subservice.

SVC_WRITE	Write a block to an opened subservice.
SVC_IOCTL	Control a subservice or a service manager.

Load

In the preferred embodiment, services (and the associated RSIs they present to RPC manager 732) may be activated during boot by an installation boot process that issues an RPC LOAD. This process reads an RPC Services Table from a configuration file, loads the service module if it is run time loadable (as opposed to being a kernel linked device driver), and then calls the LOAD entry point for the service. A successful return from the LOAD entry point will indicate that the service has properly loaded and is ready to accept requests.

RPC LOAD Call Example: SVC_LOAD (long service_id)

This LOAD interface call is called by the RPC manager 732 during rights operating system 602 initialization. It permits a service manager to load any dynamically loadable components and to initialize any device and memory required by the service. The service number that the service is loaded as is passed in as *service_id* parameter. In the preferred embodiment, the service

returns 0 is the initialization process was completed successfully or an error number if some error occurred.

Mount

Once a service has been loaded, it may not be fully functional for all subservices. Some subservices (e.g., communications based services) may require the establishment of additional connections, or they may require additional modules to be loaded. If the service is defined as "mountable," a RPC manager 732 will call the MOUNT subservice entry point with the requested subservice ID prior to opening an instance of a subservice.

RPC MOUNT Call Example:

SVC_MOUNT (long service_id, long subservice_id, BYTE *buffer)

This MOUNT interface call instructs a service to make a specific subservice ready. This may include services related to networking, communications, other system services, or external resources. The *service_id* and *subservice_id* parameters may be specific to the specific service being requested. The *buffer* parameter is a memory address that references a control structure appropriate to a specific service.

Open

Once a service is loaded and "mounted," specific instances of a service may be "opened" for use. "Opening" an instance of a service may allocate memory to store control and status information. For example, in a BSD socket based network connection, a LOAD call will initialize the software and protocol control tables, a MOUNT call will specify networks and hardware resources, and an OPEN will actually open a socket to a remote installation.

Some services, such as commercial database manager 730 that underlies the secure database service, may not be "mountable." In this case, a LOAD call will make a connection to a database manager 730 and ensure that records are readable. An OPEN call may create instances of internal cache manager 746 for various classes of records.

RPC OPEN Call Example:

SVC_OPEN (long service_id, long subservice_id, BYTE *buffer, int (*receive) (long request_id))

This OPEN interface call instructs a service to open a specific subservice. The *service_id* and *subservice_id* parameters are specific to the specific service being requested,

and the *buffer* parameter is a memory address that references a control structure appropriate to a specific service.

The optional *receive* parameter is the address of a notification callback function that is called by a service whenever a message is ready for the service to retrieve it. One call to this address is made for each incoming message received. If the caller passes a NULL to the interface, the software will not generate a callback for each message.

Close, Unmount and Unload

The converse of the OPEN, MOUNT, and LOAD calls are CLOSE, UNMOUNT, and UNLOAD. These interface calls release any allocated resources back to ROS 602 (e.g., memory manager 680a).

RPC CLOSE Call Example: SVC_CLOSE (long svc_handle)

This LOAD interface call closes an open service "handle." A service "handle" describes a service and subservice that a user wants to close. The call returns 0 if the CLOSE request succeeds (and the handle is no longer valid) or an error number.

RPC UNLOAD Call Example: SVC_UNLOAD (void)

This UNLOAD interface call is called by a RPC manager 732 during shutdown or resource reallocation of rights operating system 602. It permits a service to close any open connections, flush buffers, and to release any operating system resources that it may have allocated. The service returns 0.

RPC UNMOUNT Call Example: SVC_UNMOUNT (long service_id, long subservice_id)

This UNMOUNT interface call instructs a service to deactivate a specific subservice. The *service_id* and *subservice_id* parameters are specific to the specific service being requested, and must have been previously mounted using the *SVC_MOUNT()* request. The call releases all system resources associated with the subservice before it returns.

Read and Write

The READ and WRITE calls provide a basic mechanism for sending information to and receiving responses from a mounted and opened service. For example, a service has requests written to it in the form of an RPC request, and makes its response available to be read by RPC Manager 732 as they become available.

RPC READ Call Example:

SVC_READ (long svc_handle, long request_id, BYTE *buffer, long size)

This READ call reads a message response from a service. The *svc_handle* and *request_id* parameters uniquely identify a request. The results of a request will be stored in the user specified *buffer* up to *size* bytes. If the buffer is too small, the first size bytes of the message will be stored in the buffer and an error will be returned.

If a message response was returned to the caller's buffer correctly, the function will return 0. Otherwise, an error message will be returned.

RPC WRITE Call Example:

SVC_write (long service_id, long subservice_id, BYTE *buffer, long size, int (*receive) (long request_id)

This WRITE call writes a message to a service and subservice specified by the *service_id/subservice_id* parameter pair. The message is stored in buffer (and usually conforms to the VDE RPC message format) and is *size* bytes long. The function returns the *request id* for the message (if it was accepted for sending) or an error number. If a user specifies the *receive* callback functions, all messages regarding a request will

be sent to the request specific callback routine instead of the generalized message callback.

Input/Output Control

The IOCTL ("Input/Output ConTroL") call provides a mechanism for querying the status of and controlling a loaded service. Each service type will respond to specific general IOCTL requests, all required class IOCTL requests, and service specific IOCTL requests.

RPC IOCTL Call Example: ROI_SVC_IOCTL (long service_id, long subservice_id.
int command. BYTE *buffer)

This IOCTL function provides a generalized control interface for a RSI. A user specifies the *service_id* parameter and an optional *subservice_id* parameter that they wish to control. They specify the control *command* parameter(s), and a *buffer* into/from which the *command* parameters may be written/read. An example of a list of commands and the appropriate buffer structures are given below.

Command	Structure	Description
GET_INFO	SVC_INFO	Returns information about a service/subservice.
GET_STATS	SVC_STATS	Returns current statistics about a service/subservice.
CLR_STATS	None	Clears the statistics about a service/subservice.

* * * * *

Now that a generic RPC Service Interface provided by the preferred embodiment has been described, the following description relates to particular examples of services provided by ROS 602.

SPE Device Driver 736

SPE device driver 736 provides an interface between ROS 602 and SPE 503. Since SPE 503 in the preferred embodiment runs within the confines of an SPU 500, one aspect of this device driver 736 is to provide low level communications services with the SPU 500 hardware. Another aspect of SPE device driver 736 is to provide an RPC service interface (RSI) 736a particular to

SPE 503 (this same RSI may be used to communicate with HPE 655 through HPE device driver 738).

SPE RSI 736a and driver 736 isolates calling processes within ROS 602 (or external to the ROS) from the detailed service provided by the SPE 503 by providing a set of basic interface points providing a concise function set. This has several advantages. For example, it permits a full line of scaled SPUs 500 that all provide common functionality to the outside world but which may differ in detailed internal structure and architecture. SPU 500 characteristics such as the amount of memory resident in the device, processor speed, and the number of services supported within SPU 500 may be the decision of the specific SPU manufacturer, and in any event may differ from one SPU configuration to another. To maintain compatibility, SPE device driver 736 and the RSI 736a it provides conform to a basic common RPC interface standard that "hides" differences between detailed configurations of SPUs 500 and/or the SPEs 503 they may support.

To provide for such compatibility, SPE RSI 736a in the preferred embodiment follows a simple block based standard. In the preferred embodiment, an SPE RSI 736a may be modeled after the packet interfaces for network Ethernet cards. This

standard closely models the block mode interface characteristics of SPUs 500 in the preferred embodiment.

An SPE RSI 736a allows RPC calls from RPC manager 732 to access specific services provided by an SPE 736. To do this, SPE RSI 736a provides a set of "service notification address interfaces." These provide interfaces to individual services provided by SPE 503 to the outside world. Any calling process within ROS 602 may access these SPE-provided services by directing an RPC call to SPE RSI 736a and specifying a corresponding "service notification address" in an RPC call. The specified "service notification address" causes SPE 503 to internally route an RPC call to a particular service within an SPE. The following is a listing of one example of a SPE service breakdown for which individual service notification addresses may be provided:

Channel Services Manager

Authentication Manager/Secure Communications Manager

Secure Database Manager

The Channel Services Manager is the principal service provider and access point to SPE 503 for the rest of ROS 602. Event processing, as will be discussed later, is primarily managed (from the point of view of processes outside SPE 503)

by this service. The Authentication Manager/Secure Communications Manager may provide login/logout services for users of ROS 602, and provide a direct service for managing communications (typically encrypted or otherwise protected) related to component assemblies 690, VDE objects 300, etc. Requests for display of information (e.g., value remaining in a financial budget) may be provided by a direct service request to a Secure Database Manager inside SPE 503. The instances of Authentication Manager/Secure Communications Manager and Secure Database Manager, if available at all, may provide only a subset of the information and/or capabilities available to processes operating inside SPE 503. As stated above, most (potentially all) service requests entering SPE are routed to a Channel Services Manager for processing. As will be discussed in more detail later on, most control structures and event processing logic is associated with component assemblies 690 under the management of a Channel Services Manager.

The SPE 503 must be accessed through its associated SPE driver 736 in this example. Generally, calls to SPE driver 736 are made in response to RPC calls. In this example, SPE driver RSI 736a may translate RPC calls directed to control or ascertain information about SPE driver 736 into driver calls. SPE driver

RSI 736a in conjunction with driver 736 may pass RPC calls directed to SPE 503 through to the SPE.

The following table shows one example of SPE device driver 736 calls:

Entry Point	Description
SPE_info()	Returns summary information about the SPE driver 736 (and SPE 503)
SPE_initialize_interface()	Initializes SPE driver 736, and sets the default notification address for received packets.
SPE_terminate_interface()	Terminates SPE driver 736 and resets SPU 500 and the driver 736.
SPE_reset_interface()	Resets driver 736 without resetting SPU 500.
SPE_get_stats()	Return statistics for notification addresses and/or an entire driver 736.
SPE_clear_stats()	Clears statistics for a specific notification address and/or an entire driver 736.
SPE_set_notify()	Sets a notification address for a specific service ID.
SPE_get_notify()	Returns a notification address for a specific service ID.
SPE_tx_pkt()	Sends a packet (e.g., containing an RPC call) to SPE 503 for processing.

The following are more detailed examples of each of the SPE driver calls set forth in the table above.

Example of an "SPE Information" Driver Call: SPE_info (void)

This function returns a pointer to an SPE_INFO data structure that defines the SPE device driver 736a. This data structure may provide certain information about SPE device driver 736, RSI 736a and/or SPU 500. An example of a SPE_INFO structure is described below:

Version Number/ID for SPE Device Driver 736
Version Number/ID for SPE Device Driver RSI 736
Pointer to name of SPE Device Driver 736
Pointer to ID name of SPU 500
Functionality Code Describing SPE Capabilities/functionality

Example of an SPE "Initialize Interface" Driver Call:

SPE_initialize_interface (int (fcn *receiver)(void))

A receiver function passed in by way of a parameter will be called for all packets received from SPE 503 unless their destination service is over-ridden using the *set_notify()* call. A receiver function allows ROS 602 to specify a format for packet communication between RPC manager 732 and SPE 503.

This function returns "0" in the preferred embodiment if the initialization of the interface succeeds and non-zero if it fails. If the function fails, it will return a code that describes the reason for the failure as the value of the function.

Example of an SPE "Terminate Interface" Driver Call:

SPE_terminate_interface (void)

In the preferred embodiment, this function shuts down SPE Driver 736, clears all notification addresses, and terminates all outstanding requests between an SPE and an ROS RPC manager 732. It also resets an SPE 503 (e.g., by a warm reboot of SPU 500) after all requests are resolved.

Termination of driver 736 should be performed by ROS 602 when the operating system is starting to shut down. It may also be necessary to issue this call if an SPE 503 and ROS 602 get so far out of synchronization that all processing in an SPE must be reset to a known state.

Example of an SPE "Reset Interface" Driver Call:

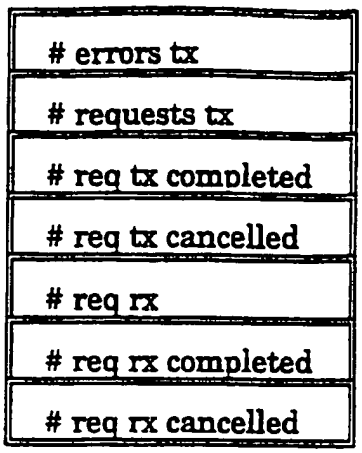
SPE_reset_interface (void)

This function resets driver 736, terminates all outstanding requests between SPE 503 and an ROS RPC manager 732, and clears all statistics counts. It does not reset the SPU 500, but simply restores driver 736 to a known stable state.

Example of an SPE "Get Statistics" Driver Call: SPE_get_stats (long service_id)

This function returns statistics for a specific service notification interface or for the SPE driver 736 in general. It returns a pointer to a static buffer that contains these statistics or NULL if statistics are unavailable (either because an interface is not initialized or because a receiver address was not specified). An example of the SPE_STATS structure may have the following definition:

Service id
packets rx
packets tx
bytes rx
bytes tx
errors rx



If a user specifies a service ID, statistics associated with packets sent by that service are returned. If a user specified 0 as the parameter, the total packet statistics for the interface are returned.

Example of an SPE "Clear Statistics" Driver Call:

SPE_clear_stats (long service_id)

This function clears statistics associated with the SPE service_id specified. If no service_id is specified (i.e., the caller passes in 0), global statistics will be cleared. The function returns 0 if statistics are successfully cleared or an error number if an error occurs.

Example of an SPE "Set Notification Address" Driver Call:

SPE_set_notify (long service_id, int (fcn*receiver) (void))

This function sets a notification address (receiver) for a specified service. If the notification address is set to NULL, SPE device driver 736 will send notifications for packets to the specified service to the default notification address.

Example of a SPE "Get Notification Address" Driver Call:

SPE_get_notify (long service_id)

This function returns a notification address associated with the named service or NULL if no specific notification address has been specified.

Example of an SPE "Send Packet" Driver Call:

send_pkt (BYTE *buffer, long size, int (far *receive) (void))

This function sends a packet stored in buffer of "length" size. It returns 0 if the packet is sent successfully, or returns an error code associated with the failure.

Redirector Service Manager 684

The redirector 684 is a piece of systems integration software used principally when ROS 602 is provided by "adding on" to a pre-existing operating system or when "transparent" operation is desired for some VDE functions, as described earlier.

In one embodiment the kernel 680, part of communications manager 776, file system 687, and part of API service 742 may be part of a pre-existing operating system such as DOS, Windows, UNIX, Macintosh System, OS9, PSOS, OS/2, or other operating system platform. The remainder of ROS 602 subsystems shown in Figure 12 may be provided as an "add on" to a preexisting operating system. Once these ROS subsystems have been supplied and "added on," the integrated whole comprises the ROS 602 shown in Figure 12.

In a scenario of this type of integration, ROS 602 will continue to be supported by a preexisting OS kernel 680, but may supplement (or even substitute) many of its functions by providing additional add-on pieces such as, for example, a virtual memory manager.

Also in this integration scenario, an add-on portion of API service 742 that integrates readily with a preexisting API service is provided to support VDE function calls. A pre-existing API service integrated with an add-on portion supports an enhanced set of operating system calls including both calls to VDE functions 604 and calls to functions 606 other than VDE functions (see Figure 11A). The add-on portion of API service

742 may translate VDE function calls into RPC calls for routing by RPC manager 732.

ROS 602 may use a standard communications manager 776 provided by the preexisting operating system, or it may provide "add ons" and/or substitutions to it that may be readily integrated into it. Redirector 684 may provide this integration function.

This leaves a requirement for ROS 602 to integrate with a preexisting file system 687. Redirector 684 provides this integration function.

In this integration scenario, file system 687 of the preexisting operating system is used for all accesses to secondary storage. However, VDE objects 300 may be stored on secondary storage in the form of external object repository 728, file system 687, or remotely accessible through communications manager 776. When object switch 734 wants to access external object repository 728, it makes a request to the object repository manager 770 that then routes the request to object repository 728 or to redirector 692 (which in turn accesses the object in file system 687).

Generally, redirector 684 maps VDE object repository 728 content into preexisting calls to file system 687. The redirector 684 provides preexisting OS level information about a VDE object 300, including mapping the object into a preexisting OS's name space. This permits seamless access to VDE protected content using "normal" file system 687 access techniques provided by a preexisting operating system.

In the integration scenarios discussed above, each preexisting target OS file system 687 has different interface requirements by which the redirector mechanism 684 may be "hooked." In general, since all commercially viable operating systems today provide support for network based volumes, file systems, and other devices (e.g., printers, modems, etc.), the redirector 684 may use low level network and file access "hooks" to integrate with a preexisting operating system. "Add-ons" for supporting VDE functions 602 may use these existing hooks to integrate with a preexisting operating system.

User Notification Service Manager 740

User Notification Service Manager 740 and associated user notification exception interface ("pop up") 686 provides ROS 602 with an enhanced ability to communicate with a user of electronic appliance 600. Not all applications 608 may be

designed to respond to messaging from ROS 602 passed through API 682, and it may in any event be important or desirable to give ROS 602 the ability to communicate with a user no matter what state an application is in. User notification services manager 740 and interface 686 provides ROS 602 with a mechanism to communicate directly with a user, instead of or in addition to passing a return call through API 682 and an application 608. This is similar, for example, to the ability of the Windows operating system to display a user message in a "dialog box" that displays "on top of" a running application irrespective of the state of the application.

The User Notification 686 block in the preferred embodiment may be implemented as application code. The implementation of interface 740a is preferably built over notification service manager 740, which may be implemented as part of API service manager 742. Notification services manager 740 in the preferred embodiment provides notification support to dispatch specific notifications to an appropriate user process via the appropriate API return, or by another path. This mechanism permits notifications to be routed to any authorized process—not just back to a process that specified a notification mechanism.

API Service Manager 742

The preferred embodiment API Service Manager 742 is implemented as a service interface to the RPC service manager 732. All user API requests are built on top of this basic interface. The API Service Manager 742 preferably provides a service instance for each running user application 608.

Most RPC calls to ROS functions supported by API Service Manager 742 in the preferred embodiment may map directly to service calls with some additional parameter checking. This mechanism permits developers to create their own extended API libraries with additional or changed functionality.

In the scenario discussed above in which ROS 602 is formed by integrating "add ons" with a preexisting operating system, the API service 742 code may be shared (e.g., resident in a host environment like a Windows DLL), or it may be directly linked with an applications's code— depending on an application programmer's implementation decision, and/or the type of electronic appliance 600. The Notification Service Manager 740 may be implemented within API 682. These components interface with Notification Service component 686 to provide a transition between system and user space.

Secure Database Service Manager ("SDSM") 744

There are at least two ways that may be used for managing secure database 600:

- a commercial database approach, and
- a site record number approach.

Which way is chosen may be based on the number of records that a VDE site stores in the secure database 610.

The commercial database approach uses a commercial database to store securely wrappered records in a commercial database. This way may be preferred when there are a large number of records that are stored in the secure database 610. This way provides high speed access, efficient updates, and easy integration to host systems at the cost of resource usage (most commercial database managers use many system resources).

The site record number approach uses a "site record number" ("SRN") to locate records in the system. This scheme is preferred when the number of records stored in the secure database 610 is small and is not expected to change extensively over time. This way provides efficient resources use with limited update capabilities. SRNs permit further grouping of similar data records to speed access and increase performance.

Since VDE 100 is highly scalable, different electronic appliances 600 may suggest one way more than the other. For example, in limited environments like a set top, PDA, or other low end electronic appliance, the SRN scheme may be preferred because it limits the amount of resources (memory and processor) required. When VDE is deployed on more capable electronic appliances 600 such as desktop computers, servers and at clearinghouses, the commercial database scheme may be more desirable because it provides high performance in environments where resources are not limited.

One difference between the database records in the two approaches is whether the records are specified using a full VDE ID or SRN. To translate between the two schemes, a SRN reference may be replaced with a VDE ID database reference wherever it occurs. Similarly, VDE IDs that are used as indices or references to other items may be replaced by the appropriate SRN value.

In the preferred embodiment, a commercially available database manager 730 is used to maintain secure database 610. ROS 602 interacts with commercial database manager 730 through a database driver 750 and a database interface 748. The database interface 748 between ROS 602 and external, third

party database vendors' commercial database manager 730 may be an open standard to permit any database vendor to implement a VDE compliant database driver 750 for their products.

ROS 602 may encrypt each secure database 610 record so that a VDE-provided security layer is "on top of" the commercial database structure. In other words, SPE 736 may write secure records in sizes and formats that may be stored within a database record structure supported by commercial database manager 730. Commercial database manager 730 may then be used to organize, store, and retrieve the records. In some embodiments, it may be desirable to use a proprietary and/or newly created database manager in place of commercial database manager 730. However, the use of commercial database manager 730 may provide certain advantages such as, for example, an ability to use already existing database management product(s).

The Secure Database Services Manager ("SDSM") 744 makes calls to an underlying commercial database manager 730 to obtain, modify, and store records in secure database 610. In the preferred embodiment, "SDSM" 744 provides a layer "on top of" the structure of commercial database manager 730. For example, all VDE-secure information is sent to commercial

database manager 730 in encrypted form. SDSM 744 in conjunction with cache manager 746 and database interface 748 may provide record management, caching (using cache manager 746), and related services (on top of) commercial database systems 730 and/or record managers. Database Interface 748 and cache manager 746 in the preferred embodiment do not present their own RSI, but rather the RPC Manager 732 communicates to them through the Secure Database Manager RSI 744a.

Name Services Manager 752

The Name Services Manager 752 supports three subservices: user name services, host name services, and services name services. User name services provides mapping and lookup between user name and user ID numbers, and may also support other aspects of user-based resource and information security. Host name services provides mapping and lookup between the names (and other information, such as for example address, communications connection/routing information, etc.) of other processing resources (e.g., other host electronic appliances) and VDE node IDs. Services name service provides a mapping and lookup between services names and other pertinent information such as connection information (e.g.,

remotely available service routing and contact information) and service IDs.

Name Services Manager 752 in the preferred embodiment is connected to External Services Manager 772 so that it may provide external service routing information directly to the external services manager. Name services manager 752 is also connected to secure database manager 744 to permit the name services manager 752 to access name services records stored within secure database 610.

External Services Manager 772 & Services Transport Layer 786

The External Services Manager 772 provides protocol support capabilities to interface to external service providers. External services manager 772 may, for example, obtain external service routing information from name services manager 752, and then initiate contact to a particular external service (e.g., another VDE electronic appliance 600, a financial clearinghouse, etc.) through communications manager 776. External services manager 772 uses a service transport layer 786 to supply communications protocols and other information necessary to provide communications.

There are several important examples of the use of External Services Manager 772. Some VDE objects may have some or all of their content stored at an Object Repository 728 on an electronic appliance 600 other than the one operated by a user who has, or wishes to obtain, some usage rights to such VDE objects. In this case, External Services Manager 772 may manage a connection to the electronic appliance 600 where the VDE objects desired (or their content) is stored. In addition, file system 687 may be a network file system (e.g., Netware, LANtastic, NFS, etc.) that allows access to VDE objects using redirecter 684. Object switch 734 also supports this capability.

If External Services Manager 772 is used to access VDE objects, many different techniques are possible. For example, the VDE objects may be formatted for use with the World Wide Web protocols (HTML, HTTP, and URL) by including relevant headers, content tags, host ID to URL conversion (e.g., using Name Services Manager 752) and an HTTP-aware instance of Services Transport Layer 786.

In other examples, External Services Manager 772 may be used to locate, connect to, and utilize remote event processing services; smart agent execution services (both to provide these services and locate them); certification services for Public Keys;

remote Name Services; and other remote functions either supported by ROS 602 RPCs (e.g., have RSIs), or using protocols supported by Services Transport Layer 786.

Outgoing Administrative Object Manager 754

Outgoing administrative object manager 754 receives administrative objects from object switch 734, object repository manager 770 or other source for transmission to another VDE electronic appliance. Outgoing administrative object manager 754 takes care of sending the outgoing object to its proper destination. Outgoing administrative object manager 754 may obtain routing information from name services manager 752, and may use communications service 776 to send the object. Outgoing administrative object manager 754 typically maintains records (in concert with SPE 503) in secure database 610 (e.g., shipping table 444) that reflect when objects have been successfully transmitted, when an object should be transmitted, and other information related to transmission of objects.

Incoming Administrative Object Manager 756

Incoming administrative object manager 756 receives administrative objects from other VDE electronic appliances 600 via communications manager 776. It may route the object to object repository manager 770, object switch 734 or other

destination. Incoming administrative object manager 756 typically maintains records (in concert with SPE 503) in secure database 610 (e.g., receiving table 446) that record which objects have been received, objects expected for receipt, and other information related to received and/or expected objects.

Object Repository Manager 770

Object repository manager 770 is a form of database or file manager. It manages the storage of VDE objects 300 in object repository 728, in a database, or in the file system 687. Object repository manager 770 may also provide the ability to browse and/or search information related to objects (such as summaries of content, abstracts, reviewers' commentary, schedules, promotional materials, etc.), for example, by using INFORMATION methods associated with VDE objects 300.

Object Submittal Manager 774

Object submittal manager 774 in the preferred embodiment provides an interface between an application 608 and object switch 734, and thus may be considered in some respects part of API 682. For example, it may allow a user application to create new VDE objects 300. It may also allow incoming/outgoing administrative object managers 756, 754 to create VDE objects 300 (administrative objects).

Figure 12A shows how object submittal manager 774 may be used to communicate with a user of electronic appliance 600 to help to create a new VDE object 300. Figure 12A shows that object creation may occur in two stages in the preferred embodiment: an object definition stage 1220, and an object creation stage 1230. The role of object submittal manager 774 is indicated by the two different "user input" depictions (774(1), 774(2)) shown in Figure 12A.

In one of its roles or instances, object submittal manager 774 provides a user interface 774a that allows the user to create an object configuration file 1240 specifying certain characteristics of a VDE object 300 to be created. This user interface 774a may, for example, allow the user to specify that she wants to create an object, allow the user to designate the content the object will contain, and allow the user to specify certain other aspects of the information to be contained within the object (e.g., rules and control information, identifying information, etc.).

Part of the object definition task 1220 in the preferred embodiment may be to analyze the content or other information to be placed within an object. Object definition user interface 774a may issue calls to object switch 734 to analyze "content" or

other information that is to be included within the object to be created in order to define or organize the content into "atomic elements" specified by the user. As explained elsewhere herein, such "atomic element" organizations might, for example, break up the content into paragraphs, pages or other subdivisions specified by the user, and might be explicit (e.g., inserting a control character between each "atomic element") or implicit. Object switch 734 may receive static and dynamic content (e.g., by way of time independent stream interface 762 and real time stream interface 760), and is capable of accessing and retrieving stored content or other information stored within file system 687.

The result of object definition 1240 may be an object configuration file 1240 specifying certain parameters relating to the object to be created. Such parameters may include, for example, map tables, key management specifications, and event method parameters. The object construction stage 1230 may take the object configuration file 1240 and the information or content to be included within the new object as input, construct an object based on these inputs, and store the object within object repository 728.

Object construction stage 1230 may use information in object configuration file 1240 to assemble or modify a container.

This process typically involves communicating a series of events to SPE 503 to create one or more PERCs 808, public headers, private headers, and to encrypt content, all for storage in the new object 300 (or within secure database 610 within records associated with the new object).

The object configuration file 1240 may be passed to container manager 764 within object switch 734. Container manager 734 is responsible for constructing an object 300 based on the object configuration file 1240 and further user input. The user may interact with the object construction 1230 through another instance 774(2) of object submittal manager 774. In this further user interaction provided by object submittal manager 774, the user may specify permissions, rules and/or control information to be applied to or associated with the new object 300. To specify permissions, rules and control information, object submittal manager 774 and/or container manager 764 within object switch 734 generally will, as mentioned above, need to issue calls to SPE 503 (e.g., through gateway 734) to cause the SPE to obtain appropriate information from secure database 610, generate appropriate database items, and store the database items into the secure database 610 and/or provide them in encrypted, protected form to the object switch for incorporation into the object. Such information provided by SPE 503 may

include, in addition to encrypted content or other information, one or more PERCs 808, one or more method cores 1000', one or more load modules 1100, one or more data structures such as UDEs 1200 and/or MDEs 1202, along with various key blocks, tags, public and private headers. and error correction information.

The container manager 764 may, in cooperation with SPE 503. construct an object container 302 based at least in part on parameters about new object content or other information as specified by object configuration file 1240. Container manager 764 may then insert into the container 302 the content or other information (as encrypted by SPE 503) to be included in the new object. Container manager 764 may also insert appropriate permissions, rules and/or control information into the container 302 (this permissions, rules and/or control information may be defined at least in part by user interaction through object submittal manager 774, and may be processed at least in part by SPE 503 to create secure data control structures). Container manager 764 may then write the new object to object repository 687, and the user or the electronic appliance may "register" the new object by including appropriate information within secure database 610.

Communications Subsystem 776

Communications subsystem 776, as discussed above, may be a conventional communications service that provides a network manager 780 and a mail gateway manager 782. Mail filters 784 may be provided to automatically route objects 300 and other VDE information to/from the outside world. Communications subsystem 776 may support a real time content feed 684 from a cable, satellite or other telecommunications link.

Secure Processing Environment 503

As discussed above in connection with Figure 12, each electronic appliance 600 in the preferred embodiment includes one or more SPEs 503 and/or one or more HPEs 655. These secure processing environments each provide a protected execution space for performing tasks in a secure manner. They may fulfill service requests passed to them by ROS 602, and they may themselves generate service requests to be satisfied by other services within ROS 602 or by services provided by another VDE electronic appliance 600 or computer.

In the preferred embodiment, an SPE 503 is supported by the hardware resources of an SPU 500. An HPE 655 may be supported by general purpose processor resources and rely on software techniques for security/protection. HPE 655 thus gives

ROS 602 the capability of assembling and executing certain component assemblies 690 on a general purpose CPU such as a microcomputer, minicomputer, mainframe computer or supercomputer processor. In the preferred embodiment, the overall software architecture of an SPE 503 may be the same as the software architecture of an HPE 655. An HPE 655 can "emulate" SPE 503 and associated SPU 500, i.e., each may include services and resources needed to support an identical set of service requests from ROS 602 (although ROS 602 may be restricted from sending to an HPE certain highly secure tasks to be executed only within an SPU 500).

Some electronic appliance 600 configurations might include both an SPE 503 and an HPE 655. For example, the HPE 655 could perform tasks that need lesser (or no) security protections, and the SPE 503 could perform all tasks that require a high degree of security. This ability to provide serial or concurrent processing using multiple SPE and/or HPE arrangements provides additional flexibility, and may overcome limitations imposed by limited resources that can practically or cost-effectively be provided within an SPU 500. The cooperation of an SPE 503 and an HPE 655 may, in a particular application, lead to a more efficient and cost effective but nevertheless secure overall processing environment for supporting and providing the

secure processing required by VDE 100. As one example, an HPE 655 could provide overall processing for allowing a user to manipulate released object 300 'contents,' but use SPE 503 to access the secure object and release the information from the object.

Figure 13 shows the software architecture of the preferred embodiment Secure Processing Environment (SPE) 503. This architecture may also apply to the preferred embodiment Host Processing Environment (HPE) 655. "Protected Processing Environment" ("PPE") 650 may refer generally to SPE 503 and/or HPE 655. Hereinafter, unless context indicates otherwise, references to any of "PPE 650," "HPE 655" and "SPE 503" may refer to each of them.

As shown in Figure 13, SPE 503 (PPE 650) includes the following service managers/major functional blocks in the preferred embodiment:

Kernel/Dispatcher 552

- Channel Services Manager 562
- SPE RPC Manager 550
- Time Base Manager 554
- Encryption/Decryption Manager 556
- Key and Tag Manager 558

- Summary Services Manager 560
- Authentication Manager/Service Communications Manager 564
- Random Value Generator 565
- Secure Database Manager 566
- Other Services 592.

Each of the major functional blocks of PPE 650 is discussed in detail below.

I. **SPE Kernel/Dispatcher 552**

The Kernel/Dispatcher 552 provides an operating system "kernel" that runs on and manages the hardware resources of SPU 500. This operating system "kernel" 552 provides a self-contained operating system for SPU 500; it is also a part of overall ROS 602 (which may include multiple OS kernels, including one for each SPE and HPE ROS is controlling/managing). Kernel/dispatcher 552 provides SPU task and memory management, supports internal SPU hardware interrupts, provides certain "low level services," manages "DTD" data structures, and manages the SPU bus interface unit 530. Kernel/dispatcher 552 also includes a load module execution manager 568 that can load programs into secure execution space for execution by SPU 500.

In the preferred embodiment, kernel/dispatcher 552 may include the following software/functional components:

load module execution manager 568

task manager 576

memory manager 578

virtual memory manager 580

"low level" services manager 582

internal interrupt handlers 584

BIU handler 586 (may not be present in HPE 655)

Service interrupt queues 588

DTD Interpreter 590.

At least parts of the kernel/dispatcher 552 are preferably stored in SPU firmware loaded into SPU ROM 532. An example of a memory map of SPU ROM 532 is shown in Figure 14A. This memory map shows the various components of kernel/dispatcher 552 (as well as the other SPE services shown in Figure 13) residing in SPU ROM 532a and/or EEPROM 532b. The Figure 14B example of an NVRAM 534b memory map shows the task manager 576 and other information loaded into NVRAM.

One of the functions performed by kernel/dispatcher 552 is to receive RPC calls from ROS RPC manager 732. As explained above, the ROS Kernel RPC manager 732 can route RPC calls to

the SPE 503 (via SPE Device Driver 736 and its associated RSI 736a) for action by the SPE. The SPE kernel/dispatcher 552 receives these calls and either handles them or passes them on to SPE RPC manager 550 for routing internally to SPE 503. SPE 503 based processes can also generate RPC requests. Some of these requests can be processed internally by the SPE 503. If they are not internally serviceable, they may be passed out of the SPE 503 through SPE kernel/dispatcher 552 to ROS RPC manager 732 for routing to services external to SPE 503.

A. Kernel/Dispatcher Task Management

Kernel dispatcher task manager 576 schedules and oversees tasks executing within SPE 503 (PPE 650). SPE 503 supports many types of tasks. A "channel" (a special type of task that controls execution of component assemblies 690 in the preferred embodiment) is treated by task manager 576 as one type of task. Tasks are submitted to the task manager 576 for execution. Task manager 576 in turn ensures that the SPE 503/SPU 500 resources necessary to execute the tasks are made available, and then arranges for the SPU microprocessor 520 to execute the task.

Any call to kernel/dispatcher 552 gives the kernel an opportunity to take control of SPE 503 and to change the task or

tasks that are currently executing. Thus, in the preferred embodiment kernel/dispatcher task manager 576 may (in conjunction with virtual memory manager 580 and/or memory manager 578) "swap out" of the execution space any or all of the tasks that are currently active, and "swap in" additional or different tasks.

SPE tasking managed by task manager 576 may be either "single tasking" (meaning that only one task may be active at a time) or "multi-tasking" (meaning that multiple tasks may be active at once). SPE 503 may support single tasking or multi-tasking in the preferred embodiment. For example, "high end" implementations of SPE 503 (e.g., in server devices) should preferably include multi-tasking with "preemptive scheduling." Desktop applications may be able to use a simpler SPE 503, although they may still require concurrent execution of several tasks. Set top applications may be able to use a relatively simple implementation of SPE 503, supporting execution of only one task at a time. For example, a typical set top implementation of SPU 500 may perform simple metering, budgeting and billing using subsets of VDE methods combined into single "aggregate" load modules to permit the various methods to execute in a single tasking environment. However, an execution environment that supports only single tasking may limit use with more

complex control structures. Such single tasking versions of SPE 503 trade flexibility in the number and types of metering and budgeting operations for smaller run time RAM size requirements. Such implementations of SPE 503 may (depending upon memory limitations) also be limited to metering a single object 300 at a time. Of course, variations or combinations are possible to increase capabilities beyond a simple single tasking environment without incurring the additional cost required to support "full multitasking."

In the preferred embodiment, each task in SPE 503 is represented by a "swap block," which may be considered a "task" in a traditional multitasking architecture. A "swap block" in the preferred embodiment is a bookkeeping mechanism used by task manager 576 to keep track of tasks and subtasks. It corresponds to a chunk of code and associated references that "fits" within the secure execution environment provided by SPU 500. In the preferred embodiment, it contains a list of references to shared data elements (e.g., load modules 1100 and UDEs 1200), private data elements (e.g., method data and local stack), and swapped process "context" information (e.g., the register set for the process when it is not processing). Figure 14C shows an example of a snapshot of SPU RAM 532 storing several examples of "swap blocks" for a number of different tasks/methods such as a

"channel" task, a "control" task, an "event" task, a "meter" task, a "budget" task, and a "billing" task. Depending on the size of SPU RAM 532, "swap blocks" may be swapped out of RAM and stored temporarily on secondary storage 652 until their execution can be continued. Thus, SPE 503 operating in a multi-tasking mode may have one or more tasks "sleeping." In the simplest form, this involves an active task that is currently processing, and another task (e.g., a control task under which the active task may be running) that is "sleeping" and is "swapped out" of active execution space. Kernel dispatcher 522 may swap out tasks at any time.

Task manager 576 may use Memory Manager 578 to help it perform this swapping operation. Tasks may be swapped out of the secure execution space by reading appropriate information from RAM and other storage internal to SPU 500, for example, and writing a "swap block" to secondary storage 652. Kernel 552 may swap a task back into the secure execution space by reading the swap block from secondary storage 652 and writing the appropriate information back into SPU RAM 532. Because secondary storage 652 is not secure, SPE 503 must encrypt and cryptographically seal (e.g., using a one-way hash function initialized with a secret value known only inside the SPU 500) each swap block before it writes it to secondary storage. The

SPE 503 must decrypt and verify the cryptographic seal for each swap block read from secondary storage 652 before the swap block can be returned to the secure execution space for further execution.

Loading a "swap block" into SPU memory may require one or more "paging operations" to possibly first save, and then flush, any "dirty pages" (i.e., pages changed by SPE 503) associated with the previously loaded swap blocks, and to load all required pages for the new swap block context.

Kernel dispatcher 522 preferably manages the "swap blocks" using service interrupt queues 588. These service interrupt queues 588 allow kernel/dispatcher 552 to track tasks (swap blocks) and their status (running, "swapped out," or "asleep"). The kernel/dispatcher 552 in the preferred embodiment may maintain the following service interrupt queues 588 to help it manage the "swap blocks":

RUN queue

SWAP queue

SLEEP queue.

Those tasks that are completely loaded in the execution space and are waiting for and/or using execution cycles from microprocessor 502 are in the RUN queue. Those tasks that are

"swapped" out (e.g., because they are waiting for other swappable components to be loaded) are referenced in the SWAP queue. Those tasks that are "asleep" (e.g., because they are "blocked" on some resource other than processor cycles or are not needed at the moment) are referenced in the SLEEP queue. Kernel/dispatcher task manager 576 may, for example, transition tasks between the RUN and SWAP queues based upon a "round-robin" scheduling algorithm that selects the next task waiting for service, swaps in any pieces that need to be paged in, and executes the task. Kernel/dispatcher 552 task manager 576 may transition tasks between the SLEEP queue and the "awake" (i.e., RUN or SWAP) queues as needed.

When two or more tasks try to write to the same data structure in a multi-tasking environment, a situation exists that may result in "deadly embrace" or "task starvation." A "multi-threaded" tasking arrangement may be used to prevent "deadly embrace" or "task starvation" from happening. The preferred embodiment kernel/dispatcher 552 may support "single threaded" or "multi-threaded" tasking.

In single threaded applications, the kernel/dispatcher 552 "locks" individual data structures as they are loaded. Once locked, no other SPE 503 task may load them and will "block"

waiting for the data structure to become available. Using a single threaded SPE 503 may, as a practical matter, limit the ability of outside vendors to create load modules 1100 since there can be no assurance that they will not cause a "deadly embrace" with other VDE processes about which outside vendors may know little or nothing. Moreover, the context swapping of a partially updated record might destroy the integrity of the system, permit unmetered use, and/or lead to deadlock. In addition, such "locking" imposes a potentially indeterminate delay into a typically time critical process, may limit SPE 503 throughput, and may increase overhead.

This issue notwithstanding, there are other significant processing issues related to building single-threaded versions of SPE 503 that may limit its usefulness or capabilities under some circumstances. For example, multiple concurrently executing tasks may not be able to process using the same often-needed data structure in a single-threaded SPE 503. This may effectively limit the number of concurrent tasks to one. Additionally, single-threadedness may eliminate the capability of producing accurate summary budgets based on a number of concurrent tasks since multiple concurrent tasks may not be able to effectively share the same summary budget data structure. Single-threadedness may also eliminate the capability to support

audit processing concurrently with other processing. For example, real-time feed processing might have to be shut down in order to audit budgets and meters associated with the monitoring process.

One way to provide a more workable "single-threaded" capability is for kernel/dispatcher 552 to use virtual page handling algorithms to track "dirty pages" as data areas are written to. The "dirty pages" can be swapped in and out with the task swap block as part of local data associated with the swap block. When a task exits, the "dirty pages" can be merged with the current data structure (possibly updated by another task for SPU 500) using a three-way merge algorithm (i.e., merging the original data structure, the current data structure, and the "dirty pages" to form a new current data structure). During the update process, the data structure can be locked as the pages are compared and swapped. Even though this virtual paging solution might be workable for allowing single threading in some applications, the vendor limitations mentioned above may limit the use of such single threaded implementations in some cases to dedicated hardware. Any implementation that supports multiple users (e.g., "smart home" set tops, many desk tops and certain PDA applications, etc.) may hit limitations of a single threaded device in certain circumstances.

It is preferable when these limitations are unacceptable to use a full "multi-threaded" data structure write capabilities. For example, a type of "two-phase commit" processing of the type used by database vendors may be used to allow data structure sharing between processes. To implement this "two-phase commit" process, each swap block may contain page addresses for additional memory blocks that will be used to store changed information. A change page is a local copy of a piece of a data element that has been written by an SPE process. The changed page(s) references associated with a specific data structure are stored locally to the swap block in the preferred embodiment.

For example, SPE 503 may support two (change pages) per data structure. This limit is easily alterable by changing the size of the swap block structure and allowing the update algorithm to process all of the changed pages. The "commit" process can be invoked when a swap block that references changed pages is about to be discarded. The commit process takes the original data element that was originally loaded (e.g., UDE_0), the current data element (e.g., UDE_n) and the changed pages, and merges them to create a new copy of the data element (e.g., UDE_{n+1}). Differences can be resolved by the DTD interpreter 590 using a DTD for the data element. The original data element is

discarded (e.g., as determined by its DTD use count) if no other swap block references it.

B. Kernel/Dispatcher Memory Management

Memory manager 578 and virtual memory manager 580 in the preferred embodiment manage ROM 532 and RAM 534 memory within SPU 500 in the preferred embodiment. Virtual memory manager 580 provides a fully "virtual" memory system to increase the amount of "virtual" RAM available in the SPE secure execution space beyond the amount of physical RAM 534a provided by SPU 500. Memory manager 578 manages the memory in the secure execution space, controlling how it is accessed, allocated and deallocated. SPU MMU 540, if present, supports virtual memory manager 580 and memory manager 578 in the preferred embodiment. In some "minimal" configurations of SPU 500 there may be no virtual memory capability and all memory management functions will be handled by memory manager 578. Memory management can also be used to help enforce the security provided by SPE 503. In some classes of SPUs 500, for example, the kernel memory manager 578 may use hardware memory management unit (MMU) 540 to provide page level protection within the SPU 500. Such a hardware-based memory management system provides an effective

mechanism for protecting VDE component assemblies 690 from compromise by "rogue" load modules.

In addition, memory management provided by memory manager 578 operating at least in part based on hardware-based MMU 540 may securely implement and enforce a memory architecture providing multiple protection domains. In such an architecture, memory is divided into a plurality of domains that are largely isolated from each other and share only specific memory areas under the control of the memory manager 578. An executing process cannot access memory outside its domain and can only communicate with other processes through services provided by and mediated by privileged kernel/dispatcher software 552 within the SPU 500. Such an architecture is more secure if it is enforced at least in part by hardware within MMU 540 that cannot be modified by any software-based process executing within SPU 500.

In the preferred embodiment, access to services implemented in the ROM 532 and to physical resources such as NVRAM 534b and RTC 528 are mediated by the combination of privileged kernel/dispatcher software 552 and hardware within MMU 540. ROM 532 and RTC 528 requests are privileged in

order to protect access to critical system component routines (e.g., RTC 528).

Memory manager 578 is responsible for allocating and deallocating memory; supervising sharing of memory resources between processes: and enforcing memory access/use restriction. The SPE kernel/dispatcher memory manager 578 typically initially allocates all memory to kernel 552, and may be configured to permit only process-level access to pages as they are loaded by a specific process. In one example SPE operating system configuration. memory manager 578 allocates memory using a simplified allocation mechanism. A list of each memory page accessible in SPE 503 may be represented using a bit map allocation vector. for example. In a memory block, a group of contiguous memory pages may start at a specific page number. The size of the block is measured by the number of memory pages it spans. Memory allocation may be recorded by setting/clearing the appropriate bits in the allocation vector.

To assist in memory management functions, a "dope vector" may be prepended to a memory block. The "dope vector" may contain information allowing memory manager 578 to manage that memory block. In its simplest form, a memory block may be structured as a "dope vector" followed by the actual

memory area of the block. This "dope vector" may include the block number, support for dynamic paging of data elements, and a marker to detect memory overwrites. Memory manager 578 may track memory blocks by their block number and convert the block number to an address before use. All accesses to the memory area can be automatically offset by the size of the "dope vector" during conversion from a block memory to a physical address. "Dope vectors" can also be used by virtual memory manager 580 to help manage virtual memory.

The ROM 532 memory management task performed by memory manager 578 is relatively simple in the preferred embodiment. All ROM 532 pages may be flagged as "read only" and as "non-pagable." EEPROM 532B memory management may be slightly more complex since the "burn count" for each EEPROM page may need to be retained. SPU EEPROM 532B may need to be protected from all uncontrolled writes to conserve the limited writable lifetime of certain types of this memory. Furthermore, EEPROM pages may in some cases not be the same size as memory management address pages.

SPU NVRAM 534b is preferably battery backed RAM that has a few access restrictions. Memory manager 578 can ensure control structures that must be located in NVRAM 534b are not

relocated during "garbage collection" processes. As discussed above, memory manager 578 (and MMU 540 if present) may protect NVRAM 534b and RAM 534a at a page level to prevent tampering by other processes.

Virtual memory manager 580 provides paging for programs and data between SPU external memory and SPU internal RAM 534a. It is likely that data structures and executable processes will exceed the limits of any SPU 500 internal memory. For example, PERCs 808 and other fundamental control structures may be fairly large, and "bit map meters" may be, or become, very large. This eventuality may be addressed in two ways:

(1) subdividing load modules 1100; and

(2) supporting virtual paging.

Load modules 1100 can be "subdivided" in that in many instances they can be broken up into separate components only a subset of which must be loaded for execution. Load modules 1100 are the smallest pagable executable element in this example. Such load modules 1100 can be broken up into separate components (e.g., executable code and plural data description blocks), only one of which must be loaded for simple load modules to execute. This structure permits a load module

1100 to initially load only the executable code and to load the data description blocks into the other system pages on a demand basis. Many load modules 1100 that have executable sections that are too large to fit into SPU 500 can be restructured into two or more smaller independent load modules. Large load modules may be manually "split" into multiple load modules that are "chained" together using explicit load module references.

Although "demand paging" can be used to relax some of these restrictions, the preferred embodiment uses virtual paging to manage large data structures and executables. Virtual Memory Manager 580 "swaps" information (e.g., executable code and/or data structures) into and out of SPU RAM 534a, and provides other related virtual memory management services to allow a full virtual memory management capability. Virtual memory management may be important to allow limited resource SPU 500 configurations to execute large and/or multiple tasks.

C. SPE Load Module Execution Manager 568

The SPE (HPE) load module execution manager ("LMEM") 568 loads executables into the memory managed by memory manager 578 and executes them. LMEM 568 provides mechanisms for tracking load modules that are currently loaded inside the protected execution environment. LMEM 568 also

provides access to basic load modules and code fragments stored within, and thus always available to, SPE 503. LMEM 568 may be called, for example, by load modules 1100 that want to execute other load modules.

In the preferred embodiment, the load module execution manager 568 includes a load module executor ("program loader") 570, one or more internal load modules 572, and library routines 574. Load module executor 570 loads executables into memory (e.g., after receiving a memory allocation from memory manager 578) for execution. Internal load module library 572 may provide a set of commonly used basic load modules 1100 (stored in ROM 532 or NVRAM 534b, for example). Library routines 574 may provide a set of commonly used code fragments/routines (e.g., bootstrap routines) for execution by SPE 503.

Library routines 574 may provide a standard set of library functions in ROM 532. A standard list of such library functions along with their entry points and parameters may be used. Load modules 1100 may call these routines (e.g., using an interrupt reserved for this purpose). Library calls may reduce the size of load modules by moving commonly used code into a central location and permitting a higher degree of code reuse. All load modules 1100 for use by SPE 503 are preferably referenced by a

load module execution manager 568 that maintains and scans a list of available load modules and selects the appropriate load module for execution. If the load module is not present within SPE 503, the task is "slept" and LMEM 568 may request that the load module 1100 be loaded from secondary storage 562. This request may be in the form of an RPC call to secure database manager 566 to retrieve the load module and associated data structures, and a call to encrypt/decrypt manager 556 to decrypt the load module before storing it in memory allocated by memory manager 578.

In somewhat more detail, the preferred embodiment executes a load module 1100 by passing the load module execution manager 568 the name (e.g., VDE ID) of the desired load module 1100. LMEM 568 first searches the list of "in memory" and "built-in" load modules 572. If it cannot find the desired load module 1100 in the list, it requests a copy from the secure database 610 by issuing an RPC request that may be handled by ROS secure database manager 744 shown in Figure 12. Load module execution manager 568 may then request memory manager 578 to allocate a memory page to store the load module 1100. The load module execution manager 568 may copy the load module into that memory page, and queue the page for decryption and security checks by encrypt/decrypt manager 556

and key and tag manager 558. Once the page is decrypted and checked, the load module execution manager 568 checks the validation tag and inserts the load module into the list of paged in modules and returns the page address to the caller. The caller may then call the load module 1100 directly or allow the load module execution module 570 to make the call for it.

Figure 15a shows a detailed example of a possible format for a channel header 596 and a channel 594 containing channel detail records 594(1), 594(2), . . . 594(N). Channel header 596 may include a channel ID field 597(1), a user ID field 597(2), an object ID field 597(3), a field containing a reference or other identification to a "right" (i.e., a collection of events supported by methods referenced in a PERC 808 and/or "user rights table" 464) 597(4), an event queue 597(5), and one or more fields 598 that cross-reference particular event codes with channel detail records ("CDRs"). Channel header 596 may also include a "jump" or reference table 599 that permits addressing of elements within an associated component assembly or assemblies 690. Each CDR 594(1), . . . 594(N) corresponds to a specific event (event code) to which channel 594 may respond. In the preferred embodiment, these CDRs may include explicitly and/or by reference each method core 1000' (or fragment thereof), load module 1100 and data structure(s), (e.g., URT, UDE 1200 and/or

MDE 1202) needed to process the corresponding event. In the preferred embodiment, one or more of the CDRs (e.g., 594(1)) may reference a control method and a URT 464 as a data structure.

Figure 15b shows an example of program control steps performed by SPE 503 to "open" a channel 594 in the preferred embodiment. In the preferred embodiment, a channel 594 provides event processing for a particular VDE object 300, a particular authorized user, and a particular "right" (i.e., type of event). These three parameters may be passed to SPE 503. Part of SPE kernel/dispatcher 552 executing within a "channel 0" constructed by low level services 582 during a "bootstrap" routine may respond initially to this "open channel" event by allocating an available channel supported by the processing resources of SPE 503 (block 1125). This "channel 0" "open channel" task may then issue a series of requests to secure database manager 566 to obtain the "blueprint" for constructing one or more component assemblies 690 to be associated with channel 594 (block 1127). In the preferred embodiment, this "blueprint" may comprise a PERC 808 and/or URT 464. In may be obtained by using the "Object, User, Right" parameters passed to the "open channel" routine to "chain" together object registration table 460 records, user/object table 462 records, URT 464 records, and PERC 808

records. This "open channel" task may preferably place calls to key and tag manager 558 to validate and correlate the tags associated with these various records to ensure that they are authentic and match. The preferred embodiment process then may write appropriate information to channel header 596 (block 1129). Such information may include, for example, User ID, Object ID, and a reference to the "right" that the channel will process. The preferred embodiment process may next use the "blueprint" to access (e.g, the secure database manager 566 and/or from load module execution manager library(ies) 568) the appropriate "control method" that may be used to, in effect, supervise execution of all of the other methods 1000 within the channel 594 (block 1131). The process may next "bind" this control method to the channel (block 1133), which step may include binding information from a URT 464 into the channel as a data structure for the control method. The process may then pass an "initialization" event into channel 594 (block 1135). This "initialization" event may be created by the channel services manager 562, the process that issued the original call requesting a service being fulfilled by the channel being built, or the control method just bound to the channel could itself possibly generate an initialization event which it would in effect pass to itself.

In response to this "initialization" event, the control method may construct the channel detail records 594(1), . . . 594(N) used to handle further events other than the "initialization" event. The control method executing "within" the channel may access the various components it needs to construct associated component assemblies 690 based on the "blueprint" accessed at step 1127 (block 1137). Each of these components is bound to the channel 594 (block 1139) by constructing an associated channel detail record specifying the method core(s) 1000', load module(s) 1100, and associated data structure(s) (e.g., UDE(s) 1200 and/or MDE(s) 1202) needed to respond to the event. The number of channel detail records will depend on the number of events that can be serviced by the "right," as specified by the "blueprint" (i.e., URT 464). During this process, the control method will construct "swap blocks" to, in effect, set up all required tasks and obtain necessary memory allocations from kernel 562. The control method will, as necessary, issue calls to secure database manager 566 to retrieve necessary components from secure database 610, issue calls to encrypt/decrypt manager 556 to decrypt retrieved encrypted information, and issue calls to key and tag manager 558 to ensure that all retrieved components are validated. Each of the various component assemblies 690 so constructed are "bound" to the channel through the channel header event code/pointer records 598 and by constructing

appropriate swap blocks referenced by channel detail records 594(1), . . . 594(N). When this process is complete, the channel 594 has been completely constructed and is ready to respond to further events. As a last step, the Figure 15b process may, if desired, deallocate the "initialization" event task in order to free up resources.

Once a channel 594 has been constructed in this fashion, it will respond to events as they arrive. Channel services manager 562 is responsible for dispatching events to channel 594. Each time a new event arrives (e.g., via an RPC call), channel services manager 562 examines the event to determine whether a channel already exists that is capable of processing it. If a channel does exist, then the channel services manager 562 passes the event to that channel. To process the event, it may be necessary for task manager 576 to "swap in" certain "swappable blocks" defined by the channel detail records as active tasks. In this way, executable component assemblies 690 formed during the channel open process shown in Figure 15b are placed into active secure execution space, the particular component assembly that is activated being selected in response to the received event code. The activated task will then perform its desired function in response to the event.

To destroy a channel, the various swap blocks defined by the channel detail records are destroyed, the identification information in the channel header 596 is wiped clean, and the channel is made available for re-allocation by the "channel 0" "open channel" task.

D. SPE Interrupt Handlers 584

As shown in Figure 13, kernel/dispatcher 552 also provides internal interrupt handler(s) 584. These help to manage the resources of SPU 500. SPU 500 preferably executes in either "interrupt" or "polling" mode for all significant components. In polling mode, kernel/dispatcher 552 may poll each of the sections/circuits within SPU 500 and emulate an interrupt for them. The following interrupts are preferably supported by SPU 500 in the preferred embodiment:

- "tick" of RTC 528
- interrupt from bus interface 530
- power fail interrupt
- watchdog timer interrupt
- interrupt from encrypt/decrypt engine 522
- memory interrupt (e.g., from MMU 540).

When an interrupt occurs, an interrupt controller within microprocessor 520 may cause the microprocessor to begin

executing an appropriate interrupt handler. An interrupt handler is a piece of software/firmware provided by kernel/dispatcher 552 that allows microprocessor 520 to perform particular functions upon the occurrence of an interrupt. The interrupts may be "vectored" so that different interrupt sources may effectively cause different interrupt handlers to be executed.

A "timer tick" interrupt is generated when the real-time RTC 528 "pulses." The timer tick interrupt is processed by a timer tick interrupt handler to calculate internal device date/time and to generate timer events for channel processing.

The bus interface unit 530 may generate a series of interrupts. In the preferred embodiment, bus interface 530, modeled after a USART, generates interrupts for various conditions (e.g., "receive buffer full," "transmitter buffer empty," and "status word change"). Kernel/dispatcher 552 services the transmitter buffer empty interrupt by sending the next character from the transmit queue to the bus interface 530. Kernel/dispatcher interrupt handler 584 may service the received buffer full interrupt by reading a character, appending it to the current buffer, and processing the buffer based on the state of the service engine for the bus interface 530. Kernel/dispatcher 552 preferably processes a status word change

interrupt and addresses the appropriate send/receive buffers accordingly.

SPU 500 generates a power fail interrupt when it detects an imminent power fail condition. This may require immediate action to prevent loss of information. For example, in the preferred embodiment, a power fail interrupt moves all recently written information (i.e., "dirty pages") into non-volatile NVRAM 534b, marks all swap blocks as "swapped out," and sets the appropriate power fail flag to facilitate recovery processing. Kernel/dispatcher 552 may then periodically poll the "power fail bit" in a status word until the data is cleared or the power is removed completely.

SPU 500 in the example includes a conventional watchdog timer that generates watchdog timer interrupts on a regular basis. A watchdog timer interrupt handler performs internal device checks to ensure that tampering is not occurring. The internal clocks of the watchdog timer and RTC 528 are compared to ensure SPU 500 is not being paused or probed, and other internal checks on the operation of SPU 500 are made to detect tampering.

The encryption/decryption engine 522 generates an interrupt when a block of data has been processed. The kernel interrupt handler 584 adjusts the processing status of the block being encrypted or decrypted, and passes the block to the next stage of processing. The next block scheduled for the encryption service then has its key moved into the encrypt/decrypt engine 522, and the next cryptographic process started.

A memory management unit 540 interrupt is generated when a task attempts to access memory outside the areas assigned to it. A memory management interrupt handler traps the request, and takes the necessary action (e.g., by initiating a control transfer to memory manager 578 and/or virtual memory manager 580). Generally, the task will be failed, a page fault exception will be generated, or appropriate virtual memory page(s) will be paged in.

E. Kernel/Dispatcher Low Level Services 582

Low level services 582 in the preferred embodiment provide "low level" functions. These functions in the preferred embodiment may include, for example, power-on initialization, device POST, and failure recovery routines. Low level services 582 may also in the preferred embodiment provide (either by themselves or in combination with authentication

manager/service communications manager 564) download response-challenge and authentication communication protocols, and may provide for certain low level management of SPU 500 memory devices such as EEPROM and FLASH memory (either alone or in combination with memory manager 578 and/or virtual memory manager 580).

F. Kernel/Dispatcher BIU handler 586

BIU handler 586 in the preferred embodiment manages the bus interface unit 530 (if present). It may, for example, maintain read and write buffers for the BIU 530, provide BIU startup initialization, etc.

G. Kernel/Dispatcher DTD Interpreter 590

DTD interpreter 590 in the preferred embodiment handles data formatting issues. For example, the DTD interpreter 590 may automatically open data structures such as UDEs 1200 based on formatting instructions contained within DTDs.

The SPE kernel/dispatcher 552 discussed above supports all of the other services provided by SPE 503. Those other services are discussed below.

II. SPU Channel Services Manager 562

"Channels" are the basic task processing mechanism of SPE 503 (HPE 655) in the preferred embodiment. ROS 602 provides an event-driven interface for "methods." A "channel" allows component assemblies 690 to service events. A "channel" is a conduit for passing "events" from services supported by SPE 503 (HPE 655) to the various methods and load modules that have been specified to process these events, and also supports the assembly of component assemblies 690 and interaction between component assemblies. In more detail, "channel" 594 is a data structure maintained by channel manager 593 that "binds" together one or more load modules 1100 and data structures (e.g., UDEs 1200 and/or MDEs 1202) into a component assembly 690. Channel services manager 562 causes load module execution manager 569 to load the component assembly 690 for execution, and may also be responsible for passing events into the channel 594 for response by a component assembly 690. In the preferred embodiment, event processing is handled as a message to the channel service manager 562.

Figure 15 is a diagram showing how the preferred embodiment channel services manager 562 constructs a "channel" 594, and also shows the relationship between the channel and component assemblies 690. Briefly, the SPE

channel manager 562 establishes a "channel" 594 and an associated "channel header" 596. The channel 594 and its header 596 comprise a data structure that "binds" or references elements of one or more component assemblies 690. Thus, the channel 594 is the mechanism in the preferred embodiment that collects together or assembles the elements shown in Figure 11E into a component assembly 690 that may be used for event processing.

The channel 594 is set up by the channel services manager 562 in response to the occurrence of an event. Once the channel is created, the channel services manager 562 may issue function calls to load module execution manager 568 based on the channel 594. The load module execution manager 568 loads the load modules 1100 referenced by a channel 594, and requests execution services by the kernel/dispatcher task manager 576. The kernel/dispatcher 552 treats the event processing request as a task, and executes it by executing the code within the load modules 1100 referenced by the channel.

The channel services manager 562 may be passed an identification of the event (e.g., the "event code"). The channel services manager 562 parses one or more method cores 1000' that are part of the component assembly(ies) 690 the channel

services manager is to assemble. It performs this parsing to determine which method(s) and data structure(s) are invoked by the type of event. Channel manager 562 then issues calls (e.g., to secure database manager 566) to obtain the methods and data structure(s) needed to build the component assembly 690. These called-for method(s) and data structure(s) (e.g., load modules 1100, UDEs 1200 and/or MDEs 1202) are each decrypted using encrypt/decrypt manager 556 (if necessary), and are then each validated using key and tag manager 558. Channel manager 562 constructs any necessary "jump table" references to, in effect, "link" or "bind" the elements into a single cohesive executable so the load module(s) can reference data structures and any other load module(s) in the component assembly. Channel manager 562 may then issue calls to LMEM 568 to load the executable as an active task.

Figure 15 shows that a channel 594 may reference another channel. An arbitrary number of channels 594 may be created by channel manager 594 to interact with one another.

"Channel header" 596 in the preferred embodiment is (or references) the data structure(s) and associated control program(s) that queues events from channel event sources, processes these events, and releases the appropriate tasks

specified in the "channel detail record" for processing. A "channel detail record" in the preferred embodiment links an event to a "swap block" (i.e., task) associated with that event. The "swap block" may reference one or more load modules 1100, UDEs 1200 and private data areas required to properly process the event. One swap block and a corresponding channel detail item is created for each different event the channel can respond to.

In the preferred embodiment, Channel Services Manager 562 may support the following (internal) calls to support the creation and maintenance of channels 562:

Call Name	Source	Description
"Write Event"	Write	Writes an event to the channel for response by the channel. The Write Event call thus permit the caller to insert an event into the event queue associated with the channel. The event will be processed in turn by the channel 594.

"Bind Item"	Ioctl	Binds an item to a channel with the appropriate processing algorithm. The Bind Item call permits the caller to bind a VDE item ID to a channel (e.g., to create one or more swap blocks associated with a channel). This call may manipulate the contents of individual swap blocks.
"Unbind Item"	Ioctl	Unbinds an item from a channel with the appropriate processing algorithm. The Unbind Item call permits the caller to break the binding of an item to a swap block. This call may manipulate the contents of individual swap blocks.

SPE RPC Manager 550

As described in connection with Figure 12, the architecture of ROS 602 is based on remote procedure calls in the preferred embodiment. ROS 602 includes an RPC Manager 732 that passes RPC calls between services each of which present an RPC service interface ("RSI") to the RPC manager. In the preferred embodiment, SPE 503 (HPE 655) is also built around the same RPC concept. The SPE 503 (HPE 655) may include a number of internal modular service providers each presenting an RSI to an RPC manager 550 internal to the SPE (HPE). These internal

service providers may communicate with each other and/or with ROS RPC manager 732 (and thus, with any other service provided by ROS 602 and with external services), using RPC service requests.

RPC manager 550 within SPE 503 (HPE 655) is not the same as RPC manager 732 shown in Figure 12, but it performs a similar function within the SPE (HPE): it receives RPC requests and passes them to the RSI presented by the service that is to fulfill the request. In the preferred embodiment, requests are passed between ROS RPC manager 732 and the outside world (i.e., SPE device driver 736) via the SPE (HPE) Kernel/Dispatcher 552. Kernel/Dispatcher 552 may be able to service certain RPC requests itself, but in general it passes received requests to RPC manager 550 for routing to the appropriate service internal to the SPE (HPE). In an alternate embodiment, requests may be passed directly between the HPE, SPE, API, Notification interface, and other external services instead of routing them through the ROS RPC manager 732. The decision on which embodiment to use is part of the scalability of the system; some embodiments are more efficient than others under various traffic loads and system configurations. Responses by the services (and additional service requests they may themselves generate) are provided to RPC

Manager 550 for routing to other service(s) internal or external to SPE 503 (HPE 655).

SPE RPC Manager 550 and its integrated service manager uses two tables to dispatch remote procedure calls: an RPC services table, and an optional RPC dispatch table. The RPC services table describes where requests for specific services are to be routed for processing. In the preferred embodiment, this table is constructed in SPU RAM 534a or NVRAM 534b, and lists each RPC service "registered" within SPU 500. Each row of the RPC services table contains a service ID, its location and address, and a control byte. In simple implementations, the control byte indicates only that the service is provided internally or externally. In more complex implementations, the control byte can indicate an instance of the service (e.g., each service may have multiple "instances" in a multi-tasking environment). ROS RPC manager 732 and SPE 503 may have symmetric copies of the RPC services table in the preferred embodiment. If an RPC service is not found in the RPC services table, SPE 503 may either reject it or pass it to ROS RPC manager 732 for service.

The SPE RPC manager 550 accepts the request from the RPC service table and processes that request in accordance with the internal priorities associated with the specific service. In