THE TRANSACTIONS OF THE IEICE, V E73(1990)July, No.7, Tokyo, JP
2334 The Transactions of the Institute of Electronics, Information and Comm. Engineers
p. 1133-1146
PAPER <Special Issue on Cryptography and Infomation Security>

Superdistribution : The Concept and the Architecture

Ryoichi MORI† *and* Masaji KAWAHARA††, *Members*

SUMMARY Superdistribution is an approach to distributing software in which software is made available freely and without restriction but is protected from modifications and modes of usage not authorized by its vendor. By eliminating the need of software vendors to protect their products against piracy through copy protection and similar measures, superdistribution promotes unrestricted distribution of software. The superdistribution architecture we have developed provides three principal functions: administrative arrangements for collecting accounting information on software usage and fees for software usage; an accounting process that records and accumulates usage charges, payments, and the allocation of usage charges among different software vendors; and a defense mechanism, utilizing digitally protected modules, that protects the system against interference with its proper operation. Superdistribution software is distributed over public channels in encrypted form. In order to participate in superdistribution, a computer must be equipped with an S-box —a digitally protected module containing microprocessors, RAM, ROM, and a real-time clock. The S-box preserves secret information such as a deciphering key and manages the proprietary aspects of the superdistribution system. A Software Usage Monitor insures the integrity of the system and keeps track of accounting information. The S-box can be realized as a digitally protected module in the form of a three-dimensional integrated circuit.

1. Introduction

Superdistribution is an approach to distributing software in which software is made available freely and without restriction but is protected from modifications and modes of usage not authorized by its vendor. Superdistribution relies neither on law nor ethics to achieve these protections; instead it is achieved through a combination of electronic devices, software, and administrative arrangements whose global design we call the "Superdistribution Architecture". The concept was invented by Mori in 1983; it was first called the "Software Service System"[1],[2]. Since 1987, work on superdistribution has been carried out by a committee of the Japan Electronics Industry Development Association (JEIDA), a non-profit industrywide organization. That committee is now known as the Superdistribution Technology Research Committee.

Superdistribution of software has the following novel combination of desirable properties:

(1) Software products are freely distributed without restriction. The user of a software product pays for using that product, not for possessing it.

(2) The vendor of a software product can set the terms and conditions of its use and the schedule of fees, if any, to be charged for its use.

(3) Software products can be executed by any user having the proper equipment, provided only that the user adheres to the conditions of use set by the vendor and pays the fees charged by the vendor.

(4) The proper operation of the superdistribution system, including the enforcement of the conditions set by the vendors, is ensured by tamper-resistant electronic devices such as digitally protected modules.

From a different viewpoint, the needs of users and the needs of vendors have until now been in irreconcilable conflict because the protective measures needed by vendors have been viewed by users as an intolerable burden. The superdistribution architecture provides a resolution to that conflict that serves the interests of both parties. Wide distribution benefits vendors because it increases usage of their products at little added cost and thus brings them more income. It benefits users because it makes more software available and the lower unit costs lead to lower prices. It also creates the possibilities of additional value-added services to be provided by the software industry. Moreover, users themselves become distributors of programs that they like, since with superdistribution there is absolutely nothing wrong with giving a copy of a program to a friend or colleague.

It might seem at first that publicly distributed software such as freeware and shareware already solves the problem addressed by superdistribution. But the likelihood of the authors being paid is too small for public for public domain software to play a leading role in the software industry. Superdistribution software is much like public domain software for which physical measures are used to ensure that the software producer is fairly compensated and that the software is protected against modification. While public domain software might achieve the aims of superdistribution in an idealized world where all users paid for software voluntarily and none of them abused it, we see little hope that such an

Manuscript received February 14, 1990.
Manuscript revised April 17, 1990.

† The author is with the Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba-shi, 305 Japan.
†† The author is with Master's Degree Program in Sciences and Engineering, University of Tsukuba, Tsukuba-shi, 305 Japan.

Table 1 Levels of software protection technology.

level	key concept	remarks
4	superdistribution	SdA(Superdistribution Architecture) 1983 R.Mori(University of Tsukuba)
3	execution privileges	"Right-To-Execute":ABYSS(A Basic Yorktown Security System) 1987 S.R.White(IBM)[3]
2	customizing software with a computer ID	costomizing deciphering key (software is common) 1986 A.Herzberg PPS(Public Protection of Software)[4] 1984 D.J.Albert (Enciphering and key managemant)[5] 1982 G.B.Purdy SPS(Soltware Protection Scheme)[6]
		customizing each copy of the software
1	hardware protection	controling execution: hardware key, e.g., ADAPSO Key-ring[7]
		inhibiting duplication: copy protection, noncompatible ROM
0	no physical protection	laws & ethics

idealized world will ever come to exist.

Table 1 describes a hierarchy of levels of software protection[3]-[7]. The previous work most similar to superdistribution is the ABYSS architecture[3],[6] developed by White, Comerford, and Weingart at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. ABYSS is based on the notion of a use-once authorization mechanism called a "token" that provides the "right to execute" a software product. All or part of the software product is executed within a protected processor, and is distributed in encrypted form. Physical security for an ABYSS processor is provided by a dense cocoon of wires whose resistance is constantly monitored by the processor. A change in resistance indicates a likely attempt to penetrate the system. The chief difference between superdistribution and the ABYSS scheme is that superdistribution does not require the physical distribution of tokens or anything else to users of a software product. In other words, ABYSS requires that software be paid for in advance while superdistribution does not.

2. The Superdistribution Architecture

The superdistribution architecture we have developed provides three principal functions:

(1) Administrative arrangements for collecting accounting information on software usage and fees for software usage.

(2) An accounting process that records and accumulates usage charges, payments, and the allocation of usage charges among different software vendors.

(3) A defense mechanism, utilizing digitally protected modules, that protects the system against interference with its proper operation.

In order to participate in superdistribution, a computer must be equipped with a device known as an *S-box* (Superdistribution Box)†. An S-box is a protected module containing microprocessors, RAM, ROM, and a real-time clock. It preserves secret information such as a deciphering key and manages the proprietary aspects of a superdistribution system. An S-box can be installed on nearly any computer, although it must be specialized to the computer's CPU type. It is also possible to integrate the S-box directly into the design of a computer. We call a computer equipped with an S-box an *S-computer*.

Programs designed for use with superdistribution are known as *S-programs*. They can be distributed freely since they are maintained in an encrypted form.

In order to make it acceptable to users, software vendors, and hardware manufacturers, the superdistribution architecture has been designed to satisfy the following requirements:

(1) The presence of the S-box must not prevent the host computer from executing software not designed for the S-box. The presence of the S-box must be invisible while such software is active.

(2) The modifications needed to install an S-box in an existing computer must be simple and inexpensive.

(3) The initial investment required to make S-programs generally available must be small.

(4) The execution speed of an S-program must not suffer a noticeable performance penalty in comparison with an ordinary program.

(5) The S-box and its supporting protocols must be unobtrusive both to users and to programmers.

(6) The S-box must be compatible with multiprogramming environments, since we anticipate that such environments will become very common in personal computers.

In our current design a program can be written without considering the S-box, but the program must be explicitly encrypted before it is distributed if it is to gain the protections of superdistribution. This encryption can be done by a programmer, using the S-box itself.

† The S-box is unrelated to the S-boxes used in the Data Encryption Standard.

The design of a superdistribution architecture can be decomposed into four tasks:

(1) Design of the superdistribution network.
(2) Logical design of the S-box and its interfaces.
(3) Design of the supporting software and file structures, including appropriate cryptographic protocols and techniques.
(4) Design of the protection for the S-box.

Technology for accomplishing each of these tasks is now nearly within to the state of the art. In the rest of this paper we describe our approach to these tasks. We have already designed and constructed two prototype S-computers with their supporting software. Prototype II, the version we are now working with, satisfies the six requirements stated above.

3. Design of the Superdistribution Network

The technical innovations that we describe here are best understood in the context of a network that provides superdistribution of S-programs and the files needed in a user's computer in order to support the use of the superdistribution services. We emphasize, however, that distributing the S-software itself is not a function of the network since a user can obtain that software by any convenient means — from a bulletin board, making a copy of a friend's program, or perhaps purchasing a collection of S-programs at a nominal price. S-programs are stored and transmitted in encrypted form, so the transmission paths need not be protected in any way.

Figure 1 illustrates a typical software distribution system that utilizes the superdistribution architecture. Each S-computer — that is, an end-user computer supporting superdistribution — is equipped with an S-box. The S-box contains a metering program, the Software Usage Monitor (SUM), that enforces the terms set by each software vendor for executing that vendor's products and keeps track of how much the user owes to each vendor. The S-box generates a payment file that contains this information. The fees charged for software usage are measured in units that we call *S-credits*. Payment files are encrypted and transmitted to the collection agency through the network as shown in the figure.

The collection agencies receive the payment files from users, process those files, and transmit payments to vendors — both the authors of the S-programs and the manufacturers of the S-boxes. Each collection agency has an S-box in its computer. The payment files are decrypted and processed under control of this S-box so as to ensure the integrity of payments to the vendors of S-programs and of the supporting hardware as well.

The clearinghouse keeps track of funds transfers engendered by superdistribution. The clearinghouse can be either a new organization or an existing one, e. g. a credit card company, that is prepared to provide the necessary services. The advantage of creating a new organization is that it provides additional degrees of freedom in the systems design. The disadvantage is that it requires a large initial investment. Using an existing organization as the clearinghouse not only saves that investment but also gives that organization the opportunity to enlarge its market.

Identification numbers (ID's) are essential to this arrangement. Each user, each software vendor, each S-box manufacturer, and each collection agency has a unique ID. In addition, users can also have ID's, either as individuals or as organizations. The ID of a user is usually in a different name space than the ID of that user's machine. User ID's provide a convenient mechanism for establishing credit, since they are associated with the individual or organization who is actually

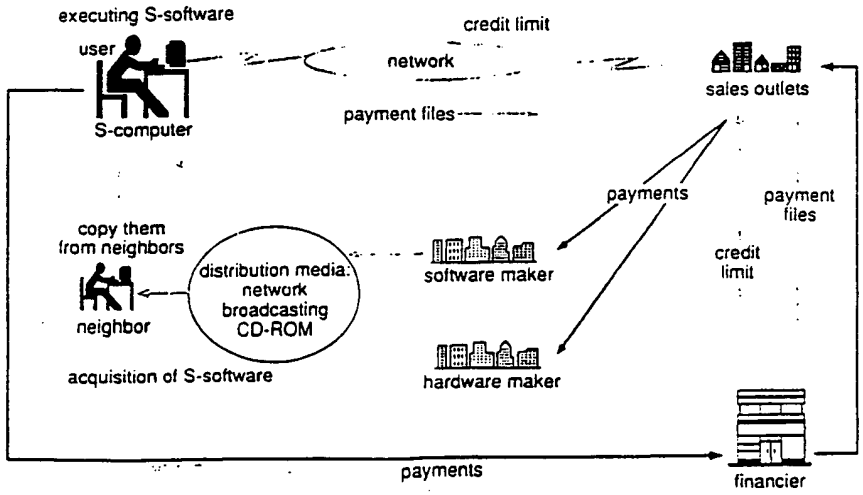

Fig. 1 Example of software distribution system utilizing the superdistribution architecture.

paying for the software. (An organization or even an individual may have more than one machine.) On the other hand, a software distribution system based purely on S-box ID's can provide a high degree of privacy for users.

Requiring prepayment for software usage avoids any risk of default, but such a requirement is unattractive to users and negates the social benefits that we hope to see provided through superdistribution. It is easy to understand why this should be so if we consider how people would react to being required to pay for all their water and electricity in advance.

The superdistribution scheme does not provide any absolute guarantee that users will pay what they owe, or even that they will return the necessary payment files to the collection agents. However, the S-box, which contains a real-time clock, will suspend its services (other than transmitting payment files) if the conditions, which are specified by the vendor or the system, are not met. Transmission can be either over a telecommunication link or with a memory card. In the event that a user transmits the payment files but does not remit the corresponding payment, a collection agent can apply appropriate sanctions.

Superdistribution does not rely on the honesty or goodwill of the manufacturer of the S-box. It is even possible to protect software designed for the S-box against attacks perpetrated by the manufacturer of the S-box. However, the methods of providing such protection are beyond the scope of this paper.

To join the system, a user need only insert an S-box, most likely in the form of a coprocessor chip, into his or her own personal computer. A collection agency can join the system in the same way. The collection agency needs a slightly different form of S-box, one that contains the software for decoding and processing the payment files. Processing at and between the collection agencies and the clearinghouse can be conveniently handled by a Credit Authorization Terminal (CAT) very similar to the ones now in use in many retail establishments.

3.1 Usage of the S-Box

Suppose that I am interested in using some S-programs. I obtain an S-box from an agent either in person or by mail. My agreement with the agent includes a credit authorization or a prepayment of, say, $100. I can now use up to $100 worth of S-programs. When I obtain the S-box, I can also obtain a memory card for it. The memory card is used to record accounting information that describes my usage of S-programs as well as the balance in my account.

To renew my usage, I must communicate the accounting information to the agent. I can do this either by establishing modem communication with the agent or by physically presenting the memory card (if I have one) to the agent. Suppose I use a modem. Then I transmit a message specifying my usage to date of each S-program that I have run. This message is generated by the S-box and enciphered so that it cannot be tampered with by anyone. The agent charges my credit card account for that usage and credits the vendors of the software that I have used. He then transmits back to me an enciphered message that resets my authorization to $100.

On the other hand, suppose that I physically present the memory card either in person or by mail. I can either have it renewed on the spot or (if I choose to mail it in) can have a second card sent to me in advance so that I am never without a card.

3.2 Customer Support and Manuals in Superdistribution

Superdistribution can also be helpful in conjunction with forms of support for users of S-programs that are not yet provided electronically.

Superdistribution changes the environment for telephone support. First, superdistribution eliminates the need to discriminate illicit and authorized users, because the incidence of illicit usage can be kept reasonably low. Second, superdistribution provides a way of charging for telephone support fairly. An S-program can generate a code number, with internal checking, and charge the user for that code number. The user then calls the vendor, asks his questions, and provides the code number. The vendor validates the code number.

Superdistributed software can be supported by hypertext and other forms of electronic documentation, but such documentation is still unlikely to replace printed manuals entirely. Currently, manuals serve both to educate the user and to provide an additional safeguard for the vendor against piracy (since a user can't buy the manuals separately from the software). With superdistribution, the safeguarding function is unnecessary and paper manuals can be sold as freely as books. In particular, a user can obtain several copies of the printed manuals if he wishes, and can even purchase the manuals without purchasing the software at all. Similar remarks apply to manuals distributed on CD-ROMs, which could be used to generate customized versions of the printed manual for particular configurations of the software, the hardware, or other aspects of the working environment.

4. Design of the S-Box and Its Interfaces

The function of the S-box is to execute the Software Usage Monitor in a secure way. The S-box also contains the cryptographic keys that ensure the integrity of the system. These keys are kept secret by storing them within a digitally protected modules (see Sect. 6).

An S-box can be added to an existing computer in any of several ways:

(1) Attaching it to an I/O port.
(2) Connecting it to the bus.
(3) Placing it between the CPU and the bus.
(4) Placing it on the same chip as the CPU or inside the CPU.

Methods (1) and (2) require no modification to the computer; our Prototype I used method (1). This method has the disadvantage that it introduces significant overhead in a multiprogramming environment. Method (4), though more difficult to implement initially, offers the best performance and the lowest cost. Because methods (3) and (4) require modifying the computer, we chose to use method (2), connection to the bus, for Prototype II.

This connection is best realized by making the S-box a coprocessor and using an existing coprocessor socket. Our current implementation uses a second computer connected to the coprocessor socket of the first one; our goal is to fabricate the S-box as a single chip that can be placed in that coprocessor socket. We are using an MC68020-based computer because of its compatibility with other equipment in our laboratory, but adapting the design to other types of processors is not difficult.

All the protection needed to ensure the integrity of the payment files and of the software is concentrated in the S-box. There is no need to protect any of the communication paths shown in Fig. 2. The signals sent over the communication paths are all encrypted, so dedicated networks are not necessary to ensure the security of the system. Any public network can be used, and the added cost of a dedicated network can be avoided. In fact, since the communications links need not be kept constantly active the system is even suitable for use with portable personal computers.

The S-box provides a limited form of protection against viruses, in that programs that are specifically designed to work with the S-box cannot be modified at the user's computer and so cannot be disabled or otherwise taken over by a virus. However, the S-box does not protect against programs, distributed through superdistribution or otherwise, whose effect is on parts of the system unrelated to the S-box.

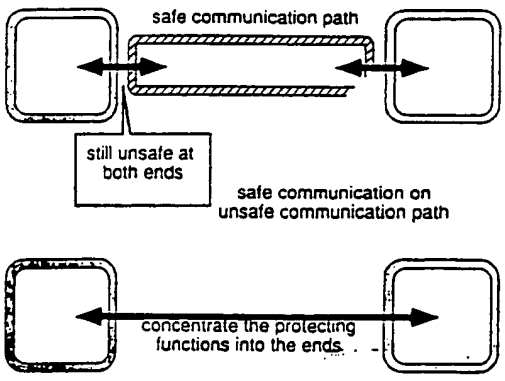

Fig. 2 No protection of communication paths results better performance/cost.

Some of the functions called for may appear to be expensive—in particular, the ability to provide the necessary physical protection and to erase the cryprographic keys recorded in the S-box in the event of an attack. However, these functions can be achieved at reasonable cost if the S-box is mass-produced as a single IC chip (or a small group of such chips).

The workings of the S-box, which we describe below, depend on the existence of cryptographic keys within the S-box. A potential problem could arise were the equipment manufacturing the S-box to be disabled and the values of the keys to be lost. This problem can be solved, although it requires the use of a different kind of protected module.

We have considered but rejected the idea of using simpler memory cards, similar to the farecards used in some transit systems, that would merely record an account balance. There are two problems with such cards. First, separate mechanisms would still be needed to record and transmit the accounting information described above. Second, the devices generally available for reading and writing such cards are insufficiently secure. A cheater could obtain such a device and modify the data on the card without using the S-box at all.

4.1 The Software Usage Monitor

The Software Usage Monitor is executed in the S-box. It performs the following functions:
(1) It monitors the execution of an S-program in order to make sure that it is only executed in the manner intended by its vendor.
(2) It encrypts and decrypts S-programs and other necessary information.
(3) It maintains an account of the charges associated with the execution of an S-program.

An S-program can issue instructions to the Software Usage Monitor. These instructions are realized as extended instructions of the CPU.

In Prototype II we have implemented the Software Usage Monitor through an emulation of the coprocessor functions. We have used an MC68020 board fitted with an MC68881 coprocessor socket, connecting a second computer (the emulator) to that socket through an interface circuit as shown in Fig. 3. The current implementation does not provide for multiprogramming because the Apple Macintosh we are using, to which an MC68020 board is plugged in, does not yet have that capability.

Further limitations on Prototype II are that it does not provide encryption and that it is not encapsulated as a digitally protected module. We consider those limitations acceptable because the purpose of our experiments at this stage is to validate the execution control and

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.