Some Cryptographic Techniques

For Secure Data Communication

By

Vijayaraghavan Varadharajan

This thesis is submitted to the Council
for National Academic Awards in partial
fulfilment of the requirements for the
degree of Doctor of Philosophy

Department of Communication Engineering

Plymouth Polytechnic

August 1984

Collaboration:

British Telecom Research Labs.,

Martlesham

DECLARATION

I hereby declare that while registered as a candidate for the degree of Doctor of Philosophy with the Council for National Academic Awards I have not been a registered candidate for another award of the CNAA or other academic or professional institution.

Signed: *Vijayaraghavan Varadharajan*

Acknowledgements

I wish to express my grateful thanks to Dr C T Stockel, Director of Studies, (Dept of Mathematics, Statistics and Computing, Plymouth Polytechnic) and to my supervisors Mr P W Sanders, (Dept of Communication Engineering, Plymouth Polytechnic) and Dr R W K Odoni (Dept of Mathematics, Exeter University) for their valuable guidance and advice at all stages of this research project. My thanks are also due to Dr G Wade (Dept of Communication Engineering, Plymouth Polytechnic) who was previously the Director of Studies. I would like to mention in particular that Mr Sanders had always been a source of encouragement. A special word of thanks should go to Dr Odoni for readily accepting to be one of the supervisors having acted as an advisor in the early stages of the project. I am greatly indebted to him for the long hours of useful discussions we had on numerous occasions.

My sincere thanks should also go to Prof S D Cohen (Dept of Mathematics, Glasgow University), Prof J Massey (Institute of Communication Engineering, ETH, Zurich) and Prof W Ledermann (Dept of Mathematics, Sussex University) who had unhesitatingly spared their time and offered valuable suggestions and advice in the course of the project.

I take this opportunity to acknowledge with thanks the various facilities made available to me by the Plymouth Polytechnic. I am grateful to the members of staff and technicians, in particular to Mr A Santillo and L Roberts, of the Dept of Communication Engineering for their kind co-operation and technical assistance as well as to the Library staff members.

I should also like to express my gratitude to the Devon County Council and the British Telecom Research Labs, Martlesham for the financial support and facilities provided for the project.

I have pleasure in thanking Mrs W Happer for her accurate and neat typing of the thesis.

Finally I shall be failing in my duty if I do not mention the encouragement and support received from my parents at every stage of the accomplishment of my task.

- iii -

C O N T E N T S

Contents continued....   <u>Page</u>

– v –

Contents condinued...   <u>Page</u>

– vi –

- vii -

Contents continued...   <u>Page</u>

- viii -

Contents continued...

– x –

Contents continued...

Papers Published/Submitted

- xi -

.. .

Some Cryptographic Techniques For Secure Data Communication

V Varadharajan

Abstract

This thesis investigates conventional and public key cryptographic techniques for secure data communication.

Block and stream cipher methods to provide secure communication over an insecure channel are discussed with particular reference to the Data Encryption Standard (DES) algorithm. A microprocessor based data encryption interface unit has been designed and constructed using the DES to provide both communication and file security. Several chaining techniques using the system have also been investigated enabling a study of their error characteristics, speed of operation, level of security and their ability to overcome difficulties due to data redundancy and structure. A statistical analysis of the randomness of the output sequences in each of these techniques has been made. Furthermore, the developed system can be used on the Prestel public network allowing storage and retrieval of completely and partly encrypted frames of information on the Prestel database.

The use of such a DES based encryption system in a communication network poses problems with regard to key distribution since the keys used need to be distributed to the users over a secure, separate channel. Several methods of key distribution including the use of public key systems are discussed.

Extensions of the Rivest-Shamir-Adleman (RSA) public key scheme to matrix rings, polynomial rings and algebraic number fields have been proposed. These extensions indicate that rings other than the ring of rational integers can be used to construct public key systems with the factorization trapdoor property. The security of such systems again relies on the difficulty of factorizing a large integer.

An extension of the Diffie-Hellman public key distribution system to matrix rings is proposed. Short cycling attacks against the exponentiation system in $GF(2^n)$ have been analysed and are shown to be equivalent to a random search procedure. A hybrid system using exponentiation in $GF(2^n)$ for key distribution and the DES for data security has been implemented and the advantage of normal basis representation in the computation of the exponentiation in $GF(2^n)$ is examined.

The role of permutation polynomials in the design of public key systems has also been investigated. In particular, it is shown that secure public key systems can be designed using Dickson permutation polynomials and Rédei rational functions. Further the complexity of public key systems can be increased by combining the permutation polynomials under the law of composition.

- xii -

C H A P T E R   1

INTRODUCTION

1.1   General

The concept of data security is becoming increasingly
significant owing to the expanding role of distributed computation,
distributed data bases and telecommunications applications such as
electronic mail and electronic funds transfer. The computer and
communications technologies have resulted in a dramatic increase
in the volume and speed of information collection, distribution and
storage. Greater information transfer and storage in turn imply
greater risk of exposure of sensitive or confidential information to
unauthorised users due to the ready availability of inexpensive
miniature intercepting devices. These have resulted in an increased
interest in computer data security not only in the military and
political areas but also in the field of commerce, where a single
transaction may involve millions of pounds. This has motivated
research particularly in the art of cryptography, which forms the
central technique of communication security.

Cryptography is the science and study of secret writing [1].
Cryptography can be defined as the transformation of a message or a
data stream by means of an algorithm so that anyone observing the
transformed data cannot deduce the hidden information. Such
transformations provide solutions to two major problems of data
security namely the privacy problem and the authentication problem [2].
In some environments the message can be transmitted in clear text as
long as its integrity is safeguarded. A common example where the
problem of authentication predominates is in telephone communication
where the called party cannot determine who is calling. Other
environments may require that the contents of the message be concealed
during transmission from unauthorised observation and this is a privacy
problem. The problems of privacy and authentication are closely
related and techniques for solving one can frequently be applied to
the other. Data encryption is recognised [3] as the most reliable
method for not only protecting vital information from eavesdroppers
but also a technique of message authentication preventing injection of
false information into a communication system by illegitimate users.

- 1 -

This thesis is mainly concerned with the data privacy problem.

On one hand, the easy availability of enormous computer power enables the cryptographer to design complicated algorithms. But on the other hand, the computer technology also helps the code breakers to be more effective in cracking the system. So it is a never ending struggle between code makers and code breakers.

Recent developments in encryption techniques for computer communication network security including the Data Encryption Standard (DES) [11] and the evolution of public key cryptosystems provided the major thrust of the present research work. Essentially the thesis can be divided into two parts. In the first part (Chapters 2 to 9), the use of encryption and decryption techniques in communication systems is investigated. The design and operation of an encryption interface unit incorporating the DES to provide communication and file security and different key distribution schemes are discussed. The second part (Chapters 10 to 15) is mainly concerned with the design of public key cryptosystems with a particular emphasis on the extensions of the Rivest-Shamir-Adleman (RSA) [12] type factorization trapdoor systems.

1.2   Thesis Organization

In Chapter 2, basic concepts of symmetric and asymmetric cryptosystems and major cryptographic techniques are briefly reviewed.

An analysis of the DES is presented in Chapter 3 which includes a software implementation of the Standard, its possible weaknesses, some of its underlying design criteria and its cryptographic strength.

The design of a microprocessor based   data encryption interface unit using the DES is described in Chapter 4.

The operation of the interface unit to provide a two-way secure data transfer in a two-node Apple microcomputer network is the subject of Chapter 5. Four different stream and block chaining techniques of the DES have been investigated using the developed interface unit.

In Chapter 6, a statistical analysis of the randomness characteristic of the output sequences produced by the DES under different modes has been carried out.

The use of the developed encryption interface unit has been

- 2 -

extended in Chapter 7 to allow file security in Apple disk systems.

Chapter 8 is concerned with the incorporation of DES based encryption system in Prestel Viewdata network. This enables transfer and storage of encrypted as well as plain data between an Apple microcomputer and the Prestel database.

Different methods of key distribution for communication and file security are investigated in Chapter 9. It includes a brief review of the RSA and the Knapsack public key cryptosystems.

In Chapter 10, the prototype RSA system over rational integers has been extended to matrix and polynomial rings.

Chapter 11 discusses the notion of ideal theory and considers the RSA type factorization trapdoor systems from an ideal point of view.

The factorization trapdoor concept in some quadratic algebraic number fields and the design of public key systems in such fields are investigated in Chapter 12.

The implementation of a hybrid system using the DES and the Diffie-Hellman public key distribution [35] system is investigated in Chapter 13. An extension of the Diffie-Hellman system to matrix rings is proposed.

The role of permutation polynomials in the design of public key systems forms the subject of Chapter 14. In particular, the use of Dickson permutation polynomials and certain Rédei functions in the construction of public key systems is discussed.

In Chapter 15, the use of chaining techniques in the matrix public key system and some precautions which must be taken when the RSA system or its extensions are used in a broadcasting type situation are described.

Chapter 16 contains the main conclusions.

- 3 -

CRYPTOGRAPHIC CONCEPTS

2.1   Cryptographic Systems

Detailed treatment of cryptographic principles can be found in [2, 3, 4]. A basic cryptographic privacy system is shown in figure 2.1. The transmitter or the sender generates a plaintext message M which is to be communicated to a legitimate receiver over an insecure channel monitored by an eavesdropper. To prevent the eavesdropper from learning the contents of M, the sender encrypts M, with an invertible transformation to produce the cryptogram or ciphertext, $C = T(M)$. When the legitimate receiver obtains C, it is deciphered with the inverse transformation to obtain the plaintext message, $M = T^{-1}(C)$.

The transformation T applied at the sending and receiving ends is a key dependent mapping from a set of messages in the plaintext to a set of ciphertext messages and vice versa. The particular transformation used is chosen from a family of transformations. The parameter that selects the individual transformation to be employed is called the key. Note that there may be more than one key involved. Assuming that the same key is used in both encryption and decryption, then $C = T_k (M)$ and $M = T_k^{-1}(C)$.

Thus a general cryptosystem consists of the following components:

1.   A plaintext message space M ;
2.   A ciphertext message space C ;
3.   A key space K;
4.   A family of encryption transformations E_k : M→ C where $k \in K$.
5.   A family of decryption transformations D_k : C→ M where $k \in K$.

The encryption and decryption transformations E_k and D_k are defined by the encrypting and decrypting algorithms E and D which may be a set of instructions, a piece of hardware or a computer program and is common to every transformation in the family. Different values of the key (s.) result in totally different transformations of plaintexts and cipher-texts. This implies that the family of transformations, that is, the general cryptographic system, can be made public information without

- 4 -
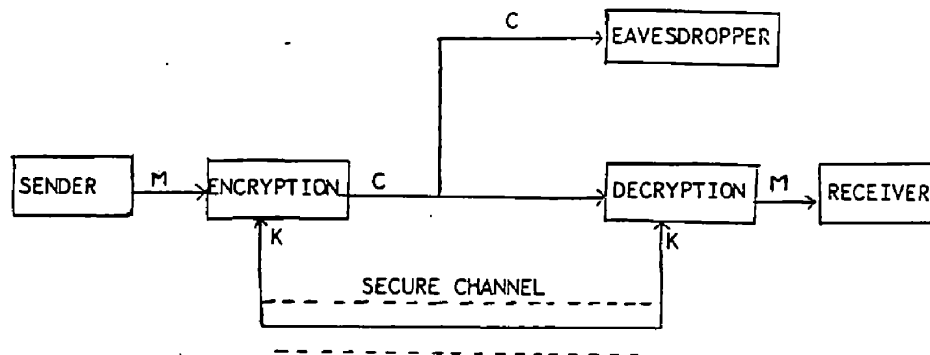
Fig. 2.1 - Basic Cryptographic Privacy System
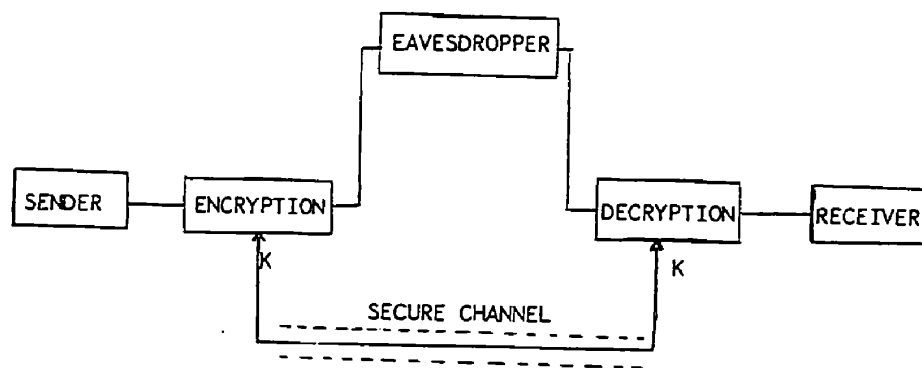
Fig. 2.2 - Basic Cryptographic Authentication System

- 5 -

compromising the security of the system. Only the key(s) needs to be kept secret. This satisfies one of the general requirements of a cryptosystem that the security must not depend on the secrecy of something like a cryptographic algorithm which cannot be easily changed if it is compromised. In addition, a publicly known system is necessary for standardization among commercial users. Even though the opponent knows the set of all possible keys, that is, the key space, he may still be unable to discover the correct set of keys required if the key space is large.

Simmons [5] classifies cryptosystems as symmetric (one key) and asymmetric (two key). In symmetric cryptosystems, the enciphering and deciphering keys are the same or easily determined from each other. Because the general method of encryption and decryption is known, this means that the transformations E_k and D_k are also easily derived from each other. Thus if both E_k and D_k are protected both secrecy and authenticity are achieved. However secrecy cannot be separated from authenticity because making either E_k or D_k available, exposes the other. Thus for secure communication, such a system requires the key to be transmitted to the receiver via some secure channel. Figure 2.2 illustrates how such a cryptographic system can be used to solve the authentication problem. In this case, the opponent not only sees all ciphertexts flowing on the channel but can alter them at will. The legitimate receiver protects himself from being deceived by an altered or injected message, by decrypting all the messages he receives and accepting only those encrypted with the correct key.

In asymmetric cryptosystems, the enciphering and deciphering keys differ in such a way that at least one key is computationally infeasible to determine from the other.. Thus one of the transformations E_k or D_k can be revealed without endangering the other. Secrecy and authenticity are provided by protecting the separate transformations namely D_k for secrecy and E_k for authenticity. Such asymmetric systems are often referred to as public key systems as in addition to E and D, the encryption key is made public. Only the decrypting key is kept secret by the receiver. The use of such systems thus avoid the necessity to transmit the key used in the algorithm over a secure channel among the communicators. Moreover such systems can be used to transmit the secret key required for conventional symmetric systems.

Such asymmetric systems are also able to deal with the problem of dispute that may arise between the sender and the receiver over the actual message sent in an authentication system. The

inability. of the symmetric system to deal with this type of problem
limits its application. This can be seen as. follows.

The validity of contracts and agreements is usually
guaranteed by signatures. The essence of a signature is that only one
person can produce it but anybody can recognize it. For this, each
user must be able to produce messages whose authenticity can be checked
by anyone, but which could not have been produced by anyone else
especially the intended recipient. In a symmetric system, the receiver
authenticates any message that he receives from the sender by
deciphering it with the key which the two hold in common. Because this
key is held in common, however, the receiver has the ability to produce
any ciphertext that could have been produced by the sender and hence
the receiver cannot prove that it is the sender who actually sent him
the disputed message. The asymmetric system provides a direct elegant
solution to this signature problem. If user A wishes to send a
signed message M to user B, he signs the message by producing
$S = D_A$ (M). When user B receives S, he can recover M by operating on
S with E_A, that is, $M = E_A$ (S). B keeps S as the proof that user A
has sent him the particular message M as only the user A could have
generated S because he is the only one who knows D_A. To obtain
secrecy of communication as well as authentication, user A sends
E_B (S) instead of S to user B. As only B knows D_B, he is the only
one who can recover S and hence M.

2.2   Cryptosystem Security and Complexity Theory

Any attempt by the eavesdropper either to decrypt a
cryptogram C to get the plaintext M or to encrypt an inauthentic plain-
text M´ to get an acceptable cryptogram C´ without prior knowledge of the
key is called cryptanalysis [2]. If cryptanalysis is impossible so
that a cryptanalyst cannot deduce M from C or C´ from M´ without prior
knowledge of the key, the cryptographic system can be said to be secure.

In order to measure the security of a cryptosystem, Diffie and
Hellman [2] have defined at least three types of attack which the
system should withstand when being subjected.

(a)   A'ciphertext only' attack is the weakest form of attack
       which the cryptographic system must withstand. In this
       attack, the cryptanalyst attempts to decipher the
       cryptogram using only the statistical properties of the

- 7 -

message source. As an example, consider a letter written in English. Not all characters or words occur equally often; for instance, the letter 'E' occurs approximately 13% of the time [6] . Such non-uniformities in the frequency distribution of the alphabet are used to give clues about the message. There is also probably a heading which contains a date and address and a closing such as 'sincerely'. With the aid of statistical tables, the cryptanalyst uses each of these facts to determine which message was most likely sent.

(b) Under a 'known plaintext' attack, the cryptanalyst is assumed to have a substantial amount of corresponding message - cryptogram pairs and tries to determine the key used in the algorithm. This form of attack is a significant threat as frequently messages are enciphered under the same key. Hence if a system cannot withstand such an attack, all messages which have been encrypted under a common key needs to be kept secret as long as any of the messages is to be kept secret. Such an attack is quite common in practice. For instance, a typical example is when information may be transmitted in secrecy which is intended for public release at a later date.

(c) A 'chosen text' attack generally occurs less frequently than a known plaintext attack. In this case, the cryptanalyst is assumed to choose messages to be enciphered or ciphertexts to be deciphered in an attempt to determine the key.

For the purpose of certifying systems as secure, it is necessary to consider more formidable cryptanalytical threats. These not only give more realistic models of the working invironment of a cryptographic system but also make the assessment of the system's strength easier.

There are two fundamentally different ways in which crypto-graphic systems may be considered secure.

A cryptosystem is said to be unconditionally secure under a

- 8 -

given form of attack if the amount of information available to the cryptanalyst is actually insufficient to determine the solution, which may be the key or the plaintext, whatever be the computing power the cryptanalyst has at his disposal [7] . As an example, consider a cryptosystem where a message-cryptogram pair uniquely determines the key. This system is not unconditionally secure under a chosen text attack or a known plaintext attack. However if the information content of the message plus the information content of the key is greater than the maximum possible amount of information in the cryptogram, then this system is unconditionally secure under a ciphertext only attack. The cryptanalyst cannot determine the complete message and key from the cryptogram alone, since he would obtain more information than that provided by the cryptogram.

Unfortunately, unconditionally secure systems require either perfect source coding or a key whose length grows linearly with respect to the sum of the lengths of all messages enciphered [7] . This requirement is not practical in most applications. Thus computationally secure systems are usually used in cryptography. A system is said to be computationally secure under a given form of attack if the amount of computation required to compute the solution exceeds the cryptanalyst's abilities or the economic value of the message to him. A measure called the work factor is often associated with a cryptosystem which gives an expression of the minimum amount of work necessary for a successful attack. In practice, there is no universally accepted fixed set of parameters used to express the work factor. Frequently, however, it is measured in one or more of the following ways: cryptanalyst hours, number of mathematical or logical operations, computing resources such as data storage and processing requirements, special hardware and calender time or more generally the cost in some money units such as dollars. This idea of computationally secure system is also related to the concept of one-way functions and complexity theory.

Algorithmic complexity theory is concerned with the comp-utational requirements (both time and space) as a function of the size of the problem solved by a particular algorithm. Complexity theory is essentially a collection of results in computer science that attempts to quantify the statement 'Problem A is 'harder' than problem B' [6] . There is a class of problems called NP problems [8] and in particular a distinguished subclass of NP called the class of NP-complete problems

- 9 -

which are regarded to be the 'hardest' problems. The class NP-complete is thought to be a source of problems that can be adapted to cryptographic applications and will by virtue of their computational complexity produce strong cryptographic systems. The cryptanalyst is then required to solve an NP-complete problem to break the cryptosystem which theoretically should require an exponential time algorithm.

However it is generally argued that [9] the complexity theory deals often with the worst case behaviour whereas in cryptography, the cryptanalytical task must be hard for almost all the instances. It will be a poor cryptosystem if the system allows easy decryption of all but a few cryptograms by the opponent. In addition computationally hard problems are not necessarily cryptographically hard problems since the cryptanalyst generally possesses additional side information and often tries to solve several instances of the same problem [3] . Recently a conventional and a public key cryptosystem based on NP-complete problems have been solved using polynomial time algorithms [6, 10] . This goes to show that merely starting with a computationally hard problem may not be enough to provide secure cryptosystems.

2.3   Cryptographic Techniques

There are two fundamental cryptographic techniques that can be used to design strong encryption-based protection schemes, namely, the block cipher technique and the stream cipher technique [3] . The suitability of either of these two techniques for use in cryptosystems depends on the nature of the application.

2.3.1   Block Cipher

Let the message be divided into blocks of fixed length. A block cipher then transforms these input blocks into output blocks using the same key. For instance, considering a binary system a string of input bits of fixed length is transformed into a string of output bits of fixed length using a block cipher as shown in Figure 2.3. The encryption and decryption functions are such that every bit in the output block depends on every bit in the input block as well as on every bit of the key. In the binary system, if the blocksize is n, then the size of the plaintext space and the size of the ciphertext space is 2^n. In order that the deciphering of a ciphertext block yields an unambiguous plaintext block, the mapping must be invertible and hence

- 10 -

injective and in this case surjective as the size of the input and output spaces are equal. Thus one can view a block cipher as defining one of the 2^n ! transformations on the set of n-bit blocks. In practice, it is not feasible to implement a block cipher that realizes all the possible combinations because of the size of the key required and the logical complexity of the cipher. Usually a key of n-bits is employed to select one out of a subset of 2^n functions.

PLAINTEXT BLOCK → KEY —K→ ENCIPHER → CIPHERTEXT BLOCK

CIPHERTEXT BLOCK → KEY —K→ DECIPHER → PLAINTEXT BLOCK
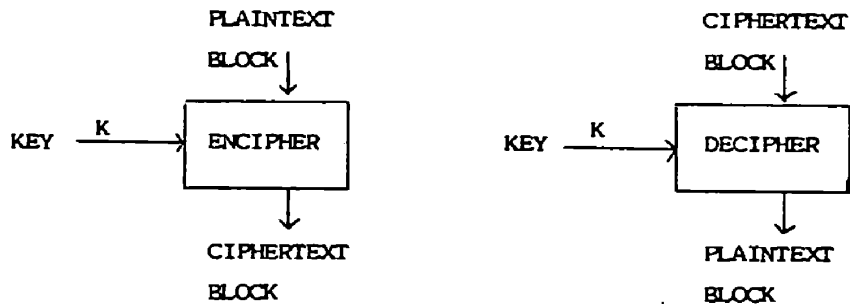
Figure 2.3 - Block Cipher.

     A fundamental property of this type of cipher is that the blocksize plays an important part in determining the cryptanalytical strength of the cipher. The blocksize must be chosen large enough to foil simple message exhaustion attacks. This attack consists of encrypting all 2^n possible plaintexts with a given key thus building a dictionary of ciphertexts and corresponding plaintexts. A message can then be recovered by searching the dictionary and relating each intercepted ciphertext block to its corresponding plaintext block. However if the blocksize is made large enough, the dictionary can be made too large to construct or store. Other attacks must also be considered before arriving at an acceptable blocksize. An attack called block frequency analysis based on frequency of occurrence of blocks is quite common. It is similar to the analysis performed on a simple substitution cipher by taking into account letter frequencies. Analytical or deterministic attack - which consists of expressing cipher operations in mathematical form as a set of equations and solving for the unknown variables directly using analytical methods - can be thwarted by making every bit of the output block a complex mathematical function of every bit of input block and key thus giving a strong intersymbol dependence property. The block ciphers also

- 11 -

exhibit error propagation properties which are suitable for error detection and authentication purposes.

The Data Encryption Standard [11] is an example of a symmetric block cipher of length 64 bits which well withstands the above mentioned and more sophisticated attacks. The algorithm will be discussed in Chapter 3 and is used in the design of a encryption interface unit in Chapters 4 and 5. Examples of public key block ciphers include the RSA system [12] and the trapdoor knapsack system [13]. They are discussed in Chapter 9.

2.3.2   Stream Cipher

A stream cipher divides the message M into successive characters or bits. It then uses a character or bit stream generator to produce a cryptographic key stream which is then combined with the plaintext message characters or bits to produce the ciphertext characters or bits. A similar procedure is carried out to recover the plaintext characters or bits by combining the key stream with the ciphertext characters or bits. The stream cipher concept is illustrated in Figures 2.4 and 2.5 where the ciphertext Y is produced from plaintext X by Exclusive-oring it with a secret binary stream R.

Let us now assume that the key to the bit stream generator is fixed and that the cryptographic key stream R produced at each iteration depends only on this key. This then implies that R does not change from one iteration to the other. Now if an opponent knows a plaintext-ciphertext pair, then he can recover the key stream R by forming $R = X \oplus Y$. Having obtained R, the opponent can decipher any intercepted ciphertext without even knowing the key to the generator which is unacceptable. Further, repetitions on the plaintext would be reflected in the ciphertext even if he did not know a ciphertext-plaintext pair. Hence to overcome this problem, the key stream must be made to change for every iteration of the ciphering algorithm. A stream cipher is said to be periodic if the key stream changes such that it repeats itself after d characters or bits for some fixed d; otherwise it is said to be non-periodic. Ideally, one would want the key stream to have a long period and to vary in a random manner. If the key stream were truly random and its length is equal to the length of the message, then this would produce an unbreakable cipher. Because the key stream is random, then it must be provided to the users in advance via some independent and secure channel which causes insurmountable logistical problems when the intended data traffic is

- 12 -
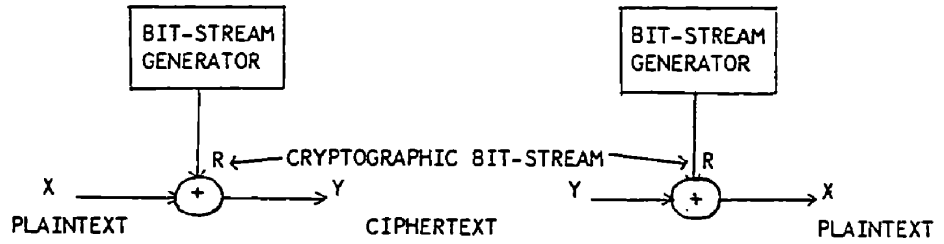
Fig. 2.4 - Stream Cipher Concept

$$Y = X \oplus R$$
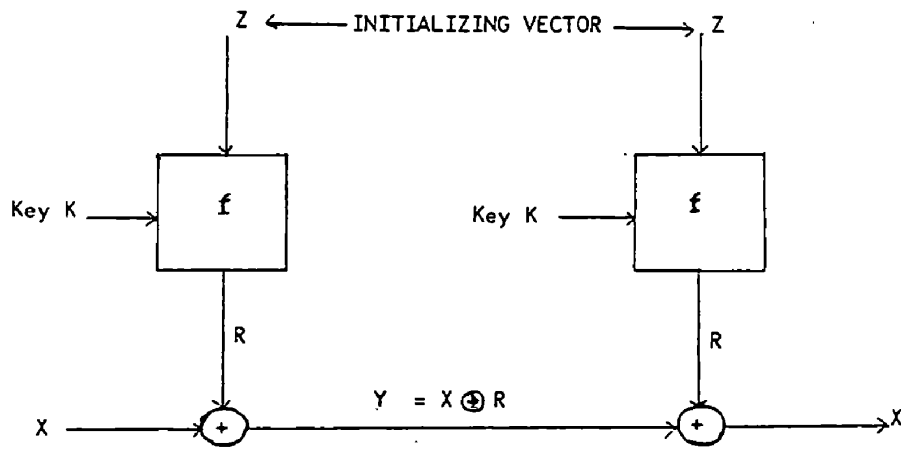
Fig. 2.5 - Stream Cipher

— 13 —

very large. Hence for practical reasons, the key stream must be implemented as an algorithmic procedure so that the key stream can be produced by users at both ends. Ciphers generated by rotor and Hagelin machines [1] are periodic whereas the Vernam cipher (one time pad) [16] and running key ciphers are non-periodic ciphers.

A system in which the key stream is generated independently of the message stream is said to be a synchronous stream cipher. In such a modulo 2 addition based system, each bit in the output cipher-text is dependent upon the corresponding bit in the input plaintext but not upon any other bits in the input plaintext. This is in con-trast to the block cipher which exhibits a much more complex relationship between the bits in the plaintext and the bits in the ciphertext. Hence the stream ciphers can be made to have non-error propagating property. Both approaches however have comparable strength.

Various techniques may be used to generate the key stream in stream ciphers. Not only the key stream generated must have good pseudo-random properties but also the generation process must be non-linear. This limits the direct use of linear feedback shift registers for the generation of these key streams because with such generators, the cryptanalyst can derive the entire key stream given a relatively small amount of plaintext-ciphertext pair [4, 14]. It is also important that the complexity of the linear equivalent of any non-linear generating process be estimated [14]. It has been suggested in [15] that non-linear substitution-permutation functions when combined with a shift register produces cryptographically strong key streams. Since the key streams can be generated in blocks, it is also possible for a block cipher to be used to obtain a stream cipher. Because both the sender and the receiver must generate key streams that are equal and secret, it is necessary that the keys used in the algorithm must also be equal and secret. This implies that a public key block cipher algorithm can be used to obtain a stream cipher if and only if it is used as a conventional algorithm, that is, both the sender and the receiver use the same algorithm (encryption E or decryption D) and the same secret key.

But as noted earlier, a fixed key even though it is kept secret does not ensure an unpredictable cryptographic key stream. To avoid producing the same key stream at each iteration of the algorithm, another parameter called the initialization vector (IV) is

- 14 -

introduced in the ciphering process. Different initialization vectors are generated in a pseudo-random non-repeating manner which in turn produce different cryptographic key streams. That is, R is generated using $R = f_k (Z)$ where Z is the initialization vector, f_k defines the block cipher algorithm under key k. Encryption and decryption operations are then given by $Y = X \oplus f_k (Z)$ and $X = Y \oplus f_k (Z)$. This is shown in Figure 2.5. For the stream cipher to be cryptographically strong, the initialization vector needs to be varied in a pseudo-random manner. One way to do this is to generate independently a new initialization vector for each iteration of the ciphering algorithm. This has the disadvantage of increasing the amount of transmitted data since the initialization vectors are now added to each block of cipher-text. A more efficient approach is as follows: at the first iteration of the cipher algorithm, the initialization vector is used as before to produce a block of key stream bits which can be used to encipher the first block of plaintext (assuming the blocksize of the cipher algorithm is same as the size of the plaintext block). At all subsequent iterations of ciphering algorithm, the initialization vector is altered or determined using one of many feedback chaining techniques. Thus chaining eliminates the problem of transmitting or storing a separate initialization vector value for each ciphertext. A feedback can be obtained from several places namely the key stream itself, the plaintext, the ciphertext or some combination thereof. Each of these approaches gives rise to cryptographic systems with different characteristics with respect to recovery from errors.

Note that the initialization vector in addition to providing cryptographic strength also establishes synchoronization between communicating cryptographic devices. It assures that the same cryptographic key streams are generated at the both ends of the link. Once the initial state of the system has been set, only the current state of the system needs to be remembered to maintain synchronization.

In general feedback chaining techniques increase the overall strength of a cryptographic system. The chaining techniques when used during ciphering process make an output dependent not only on the current input and key but also on earlier input(s) and/or output(s). In effect it introduces noise into the ciphering process. This helps to eliminate the undesirable effects of redundancy and structure present within the plaintext data. Several chaining techniques will be discussed in Chapter 5 with special reference to the Data Encryption Standard.

- 15 -

CHAPTER 3

DATA ENCRYPTION ALGORITHMS

3.1   Underline General

In cryptography, two main operations have been used for
centuries and they still form the main elements of modern encryption
algorithms. They are transpositions and substitutions and may be
applied to words, symbols, letters and binary bits or groups of bits.
There is also another technique, that of concealment where the symbols
of the message are mixed up with many other symbols which carry no
important information at all although they may appear to. This method
can be used to give considerable security but it expands the message
by a great amount. These operations lead to three different classes
of ciphers namely the transposition ciphers, the substitution ciphers
and a combination of both called the product ciphers.

3.1.1   Transposition Cipher

A transposition cipher consists of rearrangement of the
characters (bits) in a block of plaintext; the characters retain their
identity but lose their position. If the transposition is one-to-one
then the process is reversible. On the other hand, if the trans-
position is not one-to-one then the operation becomes irreversible.
Consider, for instance, the transposition which maps an 8-bit block
to a 6-bit one, say by discarding bits 3 and 6 and rearranging the
others. Here the total number of zeroes and ones are no longer
preserved. Transposition by itself is not a very secure type of
encipherment because unless every message has a unique form of
transposition, the acquisition of several plaintext-ciphertext pairs
allows the cryptanalyst to discover the permutation statistically.

3.1.2   Substitution Cipher

A substitution cipher consists of the replacement of
characters of the plaintext with characters from another alphabet.
In the case of binary operations, a look-up table characterises the
substitution operation. The bits are divided into small groups
which are then replaced by the contents of the look-up table addressed

- 16 -

by each group. For example, if the message is divided into 4 bit groups, there will be 16 possible combinations for each group, so that the table will require 16 entries numbered 0 to 15. In general, the number of ones and zeroes is not preserved and a change in one bit of the input may affect several bits of the output. The Data Encryption Standard (Section 3.2) uses eight such look-up tables commonly referred to as S-boxes, each converting a six-bit input into a four bit output. An important advantage of the substitution cipher is that the contents of the look-up table can be changed frequently and implementation of such an operation can be readily done with read only memories (ROMs). A substitution may or may not be reversible depending on the form of the look-up tables. One of the oldest substitution cipher is the Caeser cipher which is a monoalphabetical substitution cipher. This cipher can be broken in ciphertext-only attack with approximately 30 alphabet characters using letter frequency analysis [14]. An important substitution cipher is the one-time pad in which the key is random, non-repeating and used only once. One-time pads are unbreakable as there is not enough information in the ciphertext to determine the key or message uniquely. The first implementation of the one-time pad cipher was the Vernam cipher [16] in which the key bits were added modulo 2 to the plaintext bits. One major problem with this cipher is that the key length grows linearly with the length of the message.

3.1.3 Product Cipher

A product cipher involves both the steps of substitution and transposition. Shannon [7] suggested the use of product ciphers to build a strong system out of individually weak components. He suggested that the product cipher be formed using substitution and permutation ciphers in an alternating manner. The permutation shuffles the digits providing 'diffusion' and non-linear substitutions provide 'confusion'. Confusion makes the relationship between the ciphertext and the plaintext as complicated as possible, that is, it hides the key and diffusion spreads the statistics of the plaintext into the ciphertext. This formed the basis of the Lucifer system designed by the IBM [17]. The Data Encryption Standard which is considered next is based on the Lucifer system.

- 17 -

3.2   Data Encryption Standard

The cryptographic algorithm used in the design of the
encryption unit is the Data Encryption Standard (DES) [11]. This
algorithm is now regarded as the US Federal Standard recommended for
use by non-military Government Agencies. It has also been adopted by
the American National Standards Institute (ANSI) and is recommended for
use by the American Bankers Association, (ABA). The adoption of DES
as a Standard for encrypting data contributed to the surge in interest
in this algorithm. Before considering the development of an encryption
system to provide communication security using a hardware implement-
ation of the Standard in the next chapter, in this chapter the DES
algorithm is analysed to provide a deeper insight into the design of
practical encryption algorithms. The algorithm has been implemented
by software to study some of its characteristics. In particular, the
software approach enables the study of intermediate outputs during each
round, whereas the hardware (LSI) implementation only gives the final
ciphertext output. The software implementation is also found to be
useful when performing statistical tests on the randomness of the
output obtained from the algorithm. This forms the subject of
Chapter 6.

First an overview of the algorithm is given. Then the
software implementation is described together with some performance
figures. Some of the characteristics of the algorithm together with
some of the design criteria underlying the choice of parameters in
the algorithm are presented. The controversy surrounding the DES
and possible weaknesses of the algorithm are then considered. Finally
the complexity of the algorithm and its security are investigated.

3.2.1   DES Algorithm - An Overview

The Data Encryption Standard algorithm is a block product
cipher system. Block because it transforms more than one character
at a time. Product because it is composed of a series of trans-
positions, substitutions and additive encodings combined by a sequence
of feedback cycles.

It is a complex non-linear algorithm which enciphers a
64-bit block of plaintext into a 64-bit block of ciphertext under the
control of a 56-bit cryptographic key. DES can be regarded as a huge
key-controlled substitution box (S-box) with a 64-bit input and
output. With such an S-box a total of $(2^{64})!$ different transformations

- 18 -

or functions from plaintext to ciphertext are possible. The 56-bit
key thus selects only a small subset (2^{56}) of the total set's
possible functions. As a single huge S-box is difficult to construct,
DES is implemented by using several smaller S-boxes and permuting
their concatenated outputs. Repetition of the substitution and
permutation processes several times increases the cryptographic
strength.

The complete DES algorithm is given in [11]. The three major
steps in the algorithm are summarized in Figure 3.1.

1.   A transposition operation, referred to as the initial
        permutation (IP). This fixed transposition does not utilize
        the 64-bit key and operates solely on the 64 data bits.

2.   A complex key dependent product transformation that uses
        block ciphering to increase the number of substitutional
        and reordering patterns.

3.   A final transposition operation referred to as the inverse
        initial permutation (IP^{-1}) which is actually the reversal of
        the transformation performed in the first step.

The second step is the most important step out of the three
and it consists of 16 separate rounds of encipherment; each round
using a product cipher approach or cipher function. The steps
performed in each round shown in Figure 3.2 are summarized below:

(i)   The 64-bit input block is divided into two parts, a left half
        (L) and a right half (R), each 32-bits long.

(ii)   The right half of the input block becomes the left half of
        the output block. This is denoted in Figure 3.2 by an arrow
        going from R_{i-1} to L_i.

The steps (iii) to (vii) which follow can be regarded as a complex
cipher function, f, operating on both key and right half of the input
block.

(iii)   The 32-bits long right half (R) undergoes an expansion

- 19 -

. . . .

PLAINTEXT   64 DATA BITS   INPUT

INITIAL PERMUTATION

PRODUCT

TRANSFORMATION

INVERSE
INITIAL PERMUTATION

CIPHERTEXT   64 DATA BITS   OUTPUT
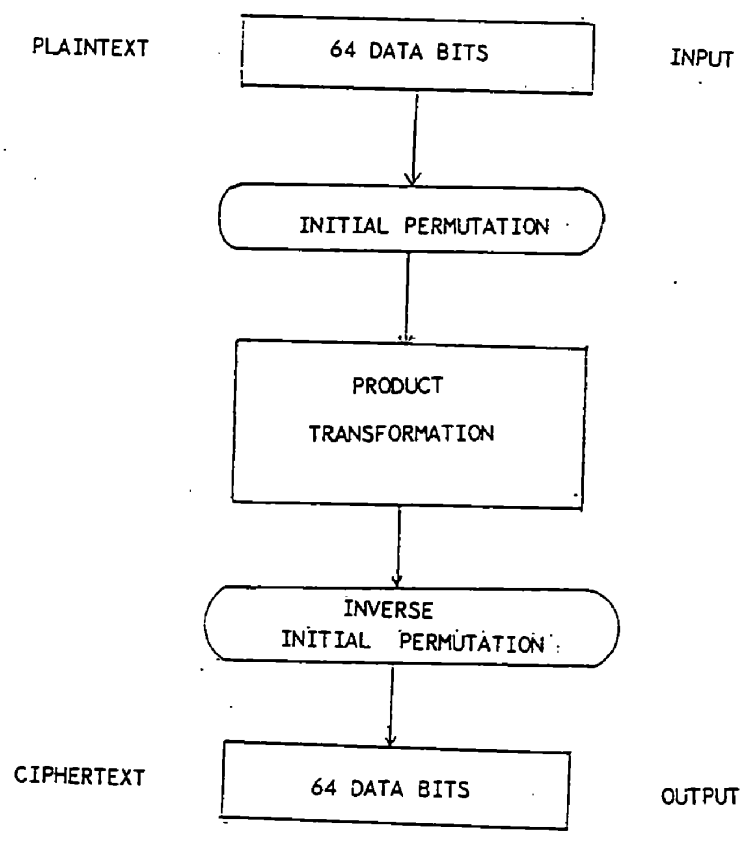
Fig. 3.1 —Overview of Enciphering process for Data Encryption Standard.
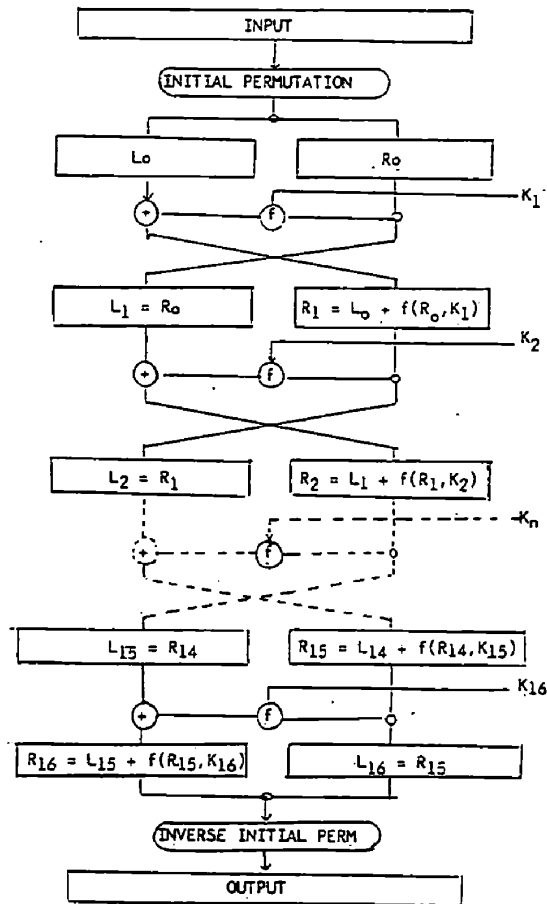
Fig.3.2(a) – Enciphering operation.
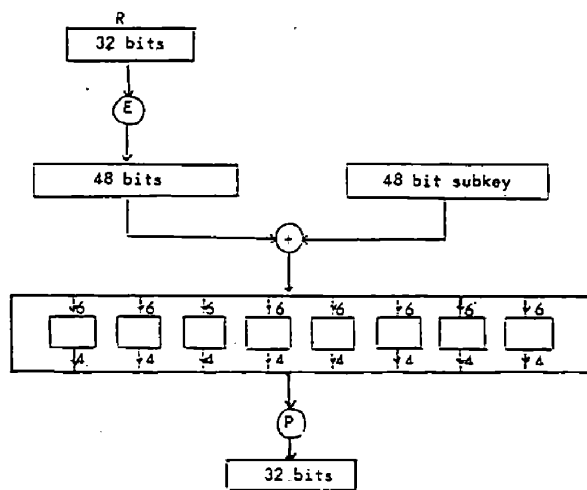
Fig.3.2(b) –Details of function f(R,K).

Page 21

process (E), yielding a 48-bit data block. This is a fixed expansion permutation and not key-dependent.

(iv)   The 64-bit key is used to generate a 48-bit subkey through a key scheduling procedure described in the next section.

(v)   This 48-bit subkey K_i is exclusive-ored with the expanded right half $E(R_i)$ yielding a 48-bit result.

(vi)   This 48-bit data is divided into eight 6-bit groups each of which is subjected to a 6-to-4-bit non-reversible substitution function (S_i). Six groups of 4-bits are then concatenated to form a 32-bit output.

(vii)   The 32-bit output is permuted to produce a 32-bit block by the fixed permutation P.

(viii)   The 32-bit output of step (vii) is combined via the exclusive-or operation with the left half of the input block to form the right half of the 64-bit output block.

Details of the permutations IP, IP^{-1}, P and substitution boxes S_i can be found in [11].

3.2.2   The Key Schedule Procedure

The key schedule procedure is used to enlarge the keyspace by expanding the externally supplied key into internal subkeys. The DES key schedule operation derives its sixteen 48-bit subkeys required for 16 rounds from the 56-bit key entered externally, by simple repetition. For reasons of security, all of these keys must be different. This is achieved by selecting a different subset of 48-bits from the 56-bit key. (Note that 8-bits out of 64-bits key are used as parity bits). The procedure is based on a shifting and bit selection algorithm.

Figure 3.3 illustrates the key schedule calculation used for encipherment. It begins with an initial permutation defined by Permuted Choice 1, (PC-1). PC-1 is the same for encipherment and decipherment and it selects 56 of the 64 external key bits (stripping off the 8 parity bits) and loads them into two 28-bit shift registers C and D. The parity checking of the external key is performed prior

– 22 –
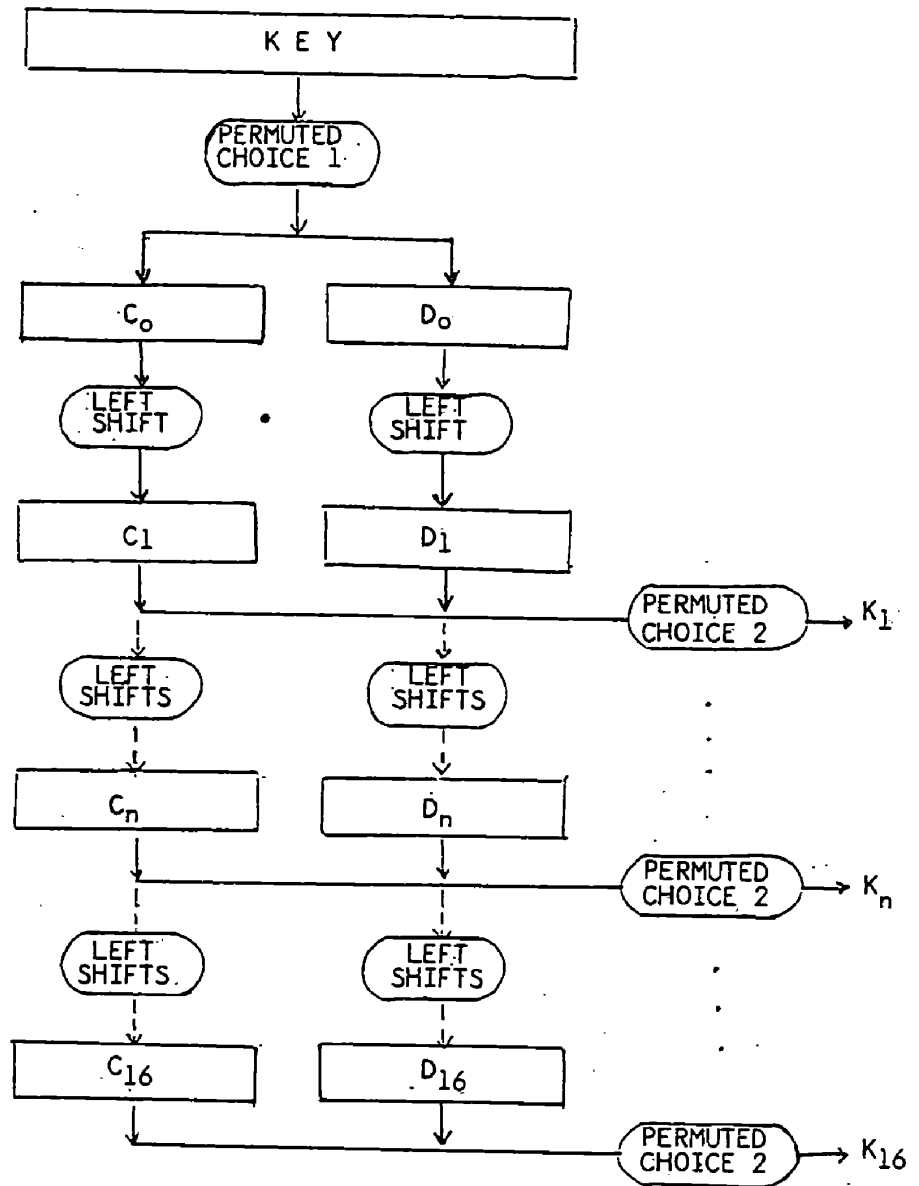
Fig.3.3 -Key Schedule Calculation.

to PC-1. During encryption, the contents of the registers C_{i-1} and D_{i-1} are shifted one or two positions to the left according to the schedule of circular shift operations shown in Table 3.1, Figure 3.4. The key K(i) is then derived from (C_i, D_i) through a second permutation defined by Permuted Choice, PC-2. Moreover the shift schedule is such that $C_{16} = C_o$ and $D_{16} = D_o$. It is to be noted here that PC-2 does not mix the contents of C_i and D_i registers. This property together with the circular shift operations allows certain values of keys K to generate identical internal subkeys. This is an inherent weakness of the algorithm and is discussed further in Section 3.6.4.

During the deciphering operation the key K(16) is used in round one and K(15) in round two and so on. But the contents of C_o and D_o are the same for enciphering and deciphering, since the external key is loaded in both cases via the Permuted Choice PC-1. This means that the key K(16) can be created at the first round merely by omitting the first shift operation and K(15) can be created at the second round by shifting C_o and D_o one bit to the right. The remainder of the internal keys are obtained in the same manner using the shift schedule in Table 3.1, Figure 3.4, in reverse order except that left shifts are changed to right shifts.

3.2.3 DES Encryption and Decryption

It is seen that the same algorithm can be used to perform encryption and decryption with minor changes. An outline of the proof of this property is as follows:

Let the contents of L and R registers on the ith round be L(i) and R(i). Let the output of the permutation P be P(i) and the output of PC-2 be K(i). Let the operation of the expansion permutation, S-boxes, permutation P be represented by f. Then during encryption,

$$L(i) = R(i-1)$$
$$R(i) = L(i-1) \oplus P(i)$$
$$P(i) = f[R(i-1), K(i)]$$

where $\oplus$ denotes Exclusive-or operation.

– 24 –

Round Number	Number of Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Figure 3.4 - Table 3.1 : Schedule of Shift operations

Rewriting these as

$$L(i) = R(i-1) \qquad (3.1)$$
$$R(i) = L(i-1) \oplus f[R(i-1), K(i)] \qquad (3.2)$$

Hence the ith round is completely determined in terms of the $(i-1)^{th}$ round provided the appropriate bits of key K are available. Similarly $(i-1)^{th}$ round can be completely expressed in terms of the ith round. This can be done because of the reversible property of exclusive-or operation: if $A = B \oplus C$ then $B = A \oplus C$. Therefore rewriting (3.1) as

$$R(i-1) = L(i)$$

and rewriting (3.2) as

$$L(i-1) = R(i) \oplus f[R(i-1), K(i)]$$
$$ie, L(i-1) = R(i) \oplus f[L(i), K(i)] \qquad (3.3)$$

it is seen that the same algorithm can be used for decryption. The involute structure of the DES can be seen by regarding the DES as a product of 33 mappings [6] as shown below

$$IP^{-1} \times \Pi_{16} \times \cdots \times \Theta \times \Pi_1 \times IP$$

where

IP is the initial permutation and IP^{-1} is its inverse Π_i, ($1 \leq i \leq 16$) are involutions given by

$\Pi_i: (x, x') \longrightarrow (x \oplus f(x'), x')$ where x is the left half and x' is the right half

Θ is the interchange involution given by

$\Theta: (x, x') \longrightarrow (x', x)$

The inverse of the DES is then given by

$$(DES)^{-1} = (IP)^{-1} \times \Pi_1 \times \Theta \times \cdots \times \Theta \times \Pi_{16} \times IP$$

- 26 -

That is, decryption is essentially achieved by operating the DES backwards.

3.3   Software DES Implementation

The DES algorithm has been implemented using a 6502 based Apple microcomputer system. This was done to gain a deeper insight into the working of the algorithm and to allow a comparison between hardware and software implementation with regard to performance characteristics and the results produced. The two implementations may not be compatible and the results may not agree with each other as the FIPS publication [11]gives only the recommended values for the permutations and S-boxes.

The description of the program together with a partial listing is given in Appendix 1. The two important factors to be considered in the software design are the storage and the speed of the implementation. In this design, a mixed approach has been adopted as there was no real constraint on memory or speed. In some places storage space increases are traded to obtain speed. A typical example of such an instance would be the use of in-line code to replace loops. On the other hand, most of the transpositions and substitutions are implemented as functions instead of using tables. Consider for instance, the 48-bit key select function involving the 56-bit initial permutation and then one or two left shifts of the 56-bit string and a 48-bit permutation to produce each of the 16 operational keys required during a cycle through the algorithm. Here these subkeys are generated at the time required. But this whole process could have been implemented using sixteen 48 byte tables, listing equivalent permutations of the current keys bit positions. In the beginning of the program, all the 16 operational keys can be calculated and stored in a table. Then each round through the algorithm can obtain the operational keys directly from the storage table.

The storage space used by this program is approximately 2 kilo-bytes. The encryption times are of the order of 100 milli-seconds. This would allow a maximum throughput of about 640 bits per second. Therefore the algorithm can be implemented with reasonable performance. But it is found that 6502 instruction set is not suitable for high speed implementation of the algorithm. If bit test instructions are available similar to those of Z80, encryption time can be

- 27 -

much reduced.

A numerical example showing the output from intermediate stages within an encryption round together with different key and data values after every round is shown in Figure 3.5. It is also found that the results produced by this program and the hardware implementation (Chapter 4) agreed with each other.

Although the algorithm can be implemented in software, it suffers from two major disadvantages. Firstly as seen from above the operation is slow compared to LSI implementation of the algorithm which allows of the order of 1 M bits per second or above. Secondly, in the software approach, the key is stored within the computer system memory which is accessible when the system is in use. With the LSI hardware implementation the key becomes unavailable after it has been entered into the device and hence the security of the overall system has been greatly enhanced. In the next section a possible advantage of the software approach is discussed.

3.3.1 A Possible Advantage of DES Software

If the internal functions of the DES algorithm are generated using tables, then it seems that there exists a mechanism to increase the difficulty of cryptanalyzing a series of encrypted messages given an encrypted message and a copy of the corresponding plaintext. Assuming that there is no fatal flaw which is dependent on these internal tables, then it is shown that these tables (ie. permutations and substitutions) can be altered in software implementations (in contrast to the hardware approach) to produce a significantly larger effective key space. One may argue that changing the values of these tables may strengthen the algorithm, but does not reduce the throughput from the rate achieved with the specified standard. Therefore each of these tables can be used as variables, like the key, agreed between senders and receivers. (Note that this idea may not be valid from subsequent sections where it is argued that carefully chosen permutations and substitutions strengthen the DES algorithm and that randomly chosen parameters may introduce weaknesses into the algorithm).

The Initial Permutation (IP) and its inverse(IP^{-1}) operate on 64-bit data blocks and hence there are 64! different settings possible for this permutation. Note that given IP, IP^{-1} is fixed. Here any bit ordering including an identity transform is assumed to be acceptable but each bit must be used. The tables are defined by a series of 64 unique one byte addresses of bits in the Initial Permutation.

– 28 –

All data and key values are represented in hexadecimal form.

Plaintext, X: FEDCBA9876543210
Key,   K: FEDCBA9876543210

Round 1

Plaintext   : L(0) = FEDCBA98
               R(0) = 76543210

Plaintext after Initial Permutation: 33FF33000F550F55

Key after Permutation choice 1   : 0E98D4BE5498C2FE

Key after 1 left shift   : 1E32AA7CAA3286FC

Key after Permutation choice 2   : 7A1E6C3032163234

Output from S-boxes, S1-S8   : 0C216D50

Output after Permutation P   : 921C209C

Ciphertext after round 1   : 0F550F55A1E3139C

        i.e.   L(1) = 0F550F55

               R(1) = A1E3139C


```
Round 1 -  Data:  0F550F55A1E3139C
           Key :  7A1E6C3032163234
Round 2 -  Data:  A1E3139C7D1EC3B6
           Key :  4A4A4C4C6C522E32
Round 3 -  Data:  7D1EC3B6B66ABD06
           Key :  5C445A6A247A385A
Round 4 -  Data:  B66ABD06F229B504
           Key :  462E305A2C4E6C50
Round 5 -  Data:  F229B5048FC9FBC4
           Key :  60623E78740A487A
Round 6 -  Data:  8FC9FBC40EB8F43D
           Key :  6E1A5642606E5474
Round 7 -  Data:  0EB8F43DC6B370BA
           Key :  487E5A547A225872
Round 8 -  Data:  C6B370BACB7238B9
           Key :  540E3E0E4862527C
Round 9 -  Data:  CB7238B90C8EFF39
           Key :  1E66302C6C283E0C
Round 10-  Data:  0C8EFF39C3DD5634
           Key :  364266704E1C141A
Round 11-  Data:  C3DD5634F5C83C96
           Key :  6E5CC0278784E360A
Round 12-  Data:  F5C83C96A38DA004
           Key :  465878345C541C56
Round 13-  Data:  A38DA0040B8B7529
           Key :  3646585E5A4E7806
Round 14-  Data:  0B8B75293393CBF1
           Key :  547662123E441A64
Round 15-  Data:  3393CBF145770966
           Key :  242C76747A3C4A14
Round 16-  Data:  4577096604DE0463
           Key :  1A38167822707E1A
Ciphertext = A933F61830233B310
```

Figure 3.5 - Data and Key values after each DES round

- 29 -

To change IP or IP^{-1} requires an input of 64 bytes. There are 32! different P permutations possible and a 32 byte input vector is required to specify the ordering of 32-bits. Also there are $\binom{48}{32} \doteq$ 2.25 x 10^{12} different possible expansion (E) permutations for mapping 32-bits into 48-bits. The 16 intermediate keys K(1) to K(16) can be obtained using a 16 x 48 table. Assuming that the 16 key select vectors must be unique, the number of unique combinations is equal to $\prod_{i=0}^{15} \binom{56}{48} - i$ which is approximately 2.77 x 10^{146}. Changing the key selection tables requires an input of 768 bytes. Further, there are $\binom{48}{32}$ = 2.25 x 10^{12} possible combinations for S-boxes which reduce the 48-bit exclusive-ored product to 32-bits.

Thus the number of possible control variables using this table structure is equal to (number of IP/IP^{-1} combinations) x (number of 48-bit expansion combinations) x (number of P permutation combinations) x (number of 6 bits to 4 bits reduction combination) x (number of key select combinations). This is of the order 10^{295}. For each of these parameter settings, there are 2^{56} possible values of key which must be tested if the DES is to be broken by exhaustive attack. For this extended algorithm, an input of 1232 bytes is required. Changing the key merely requires changing any of these 1232 bytes.

The above argument shows how the permutations and substitutions in the DES algorithm can themselves be regarded as 'keys'. But one should be careful in determining the security of such a system. In the above analysis, it has been assumed for simplicity that any one set of permutations and substitutions can be used in the algorithm. But the security of the algorithm may be heavily dependent on the particular values of the S-boxes and the permutations [3]. Thus although the above process shows how the keysize to the DES algorithm can be extended, it must be realized that not all these 'keys' may provide the same security.

3.4   Some Characteristics of DES Algorithm

3.4.1   Avalanche Effect

If a small change in the key or plaintext were to produce a corresponding small change in the ciphertext, then this might effectively reduce the size of the plaintext or key space to be searched. Hence one of the fundamental requirements of a good

- 30 -

cryptographic algorithm is to produce a 'large' change in the cipher-text for a 'small'change in the plaintext or key. The DES exhibits this property referred to as the 'avalanche' effect by Horst Feistel [6]. Actually Meyer has shown in [18] that after five rounds, each ciphertext bit depends on all plaintext bits and key bits. This inter-symbol dependence property can be used for error detection and authentication purposes.

Table 3.2 in Figure 3.6 shows the effect of change of a single input bit change in the plaintext. The plaintexts X and X´

X = (11111111 11111111 11111111 11111111 11111111 11111111
     11111111 11111111)

X´= (01111111 11111111 11111111 11111111 11111111 11111111
     11111111 11111111)

are enciphered with a randomly chosen key and the effect of changing a single bit as a function of the number of rounds is seen. In Table 3.3 in Figure 3.7, the procedure is repeated, now fixing the plaintext and changing a single bit in the key.

A similar effect is seen to occur for changes of single bit in ciphertext and key on decryption.

3.4.2 Complementary Property

The DES is invariant under complementation of plaintext, key and ciphertext. The relationship called the complementary property of DES can be expressed as:

$$E_K (X) = \overline{E_{\bar{K}} (\bar{X})}$$

where E_K stands for encryption under key K
        X is the plaintext
and the bars represent complementation, that is, bit-inversion.

This property arises because of the way in which the internal subkeys are used in each round. This can be shown as follows:

Section 3.2.1 shows that the expanded version of right half

– 31 –

Key

Round

Plaintext 1	0 :	10000101 11001101 11001011 00011100 10011011 11010000 01000110 00011010
Plaintext 2	0 :	11111111 11111111 11111111 11111111 01111111 11111111 11111111
	1 :	11111111 11111111 11111111 11111111 01000100 01111101 00010000
	1 :	11111111 11111111 11111111 11111110 01110100 11111110 00000000
	2 :	01000100 01111101 11111111 00010000 00101010 10101000 01010111
	2 :	01000000 01110100 11111110 00000000 01100010 10100010 10110111
	3 :	00101010 11011000 01010111 00110111 01010000 01000110 11001000
	3 :	01100010 10100011 10110101 10100101 01001000 01100111 10001100
	4 :	00110101 01010000 01000110 10011011 10110111 01100010 01010001
	4 :	11010101 01001000 01100111 10001100 00000111 10100001 10011000
	5 :	10011011 10110000 01100010 00111001 00001001 00001000 10010001
	5 :	01110110 00001100 10100001 11011011 10000111 01010010 11111011
	6 :	00110011 00011110 01011001 11001101 00001111 11101001 00010010
	6 :	11011011 10000111 01010010 11111011 01101111 10101110 11001001
	7 :	11001101 00001111 11101001 00010010 10000011 01010000 00110011
	7 :	01101111 01111111 10101110 11001001 00111111 11101100 01000110
	8 :	10000011 01010000 00110110 10111010 10011011 11000100 00010110
	8 :	00111111 10110100 11110111 01000100 11111110 10001001 00110000

(continued on next page)

- 32 -

```
 9 :  10111010 10011110 11000100 00010110 01000111 10011000 11001001 00011001
 9 :  01000100 11111110 10001001 00110000 10100100 11010100 11001111 01100110

10 :  01000111 10011000 11001001 00011001 00010110 10110011 11001011 01010100
10 :  10100101 11010100 11001111 01100110 00010110 10100001 11101011 01000100

11 :  00101110 10100011 11000011 01010100 11000010 00101011 11110111 01001101
11 :  00010110 10100001 11101011 01000100 10111010 01111110 00000010 01010111

12 :  11000010 00100101 11110111 01001101 11011100 00000101 01101100 10011001
12 :  10111110 01111010 00000010 01010111 01000010 10110010 11000111 01110111

13 :  11011100 00000101 01101100 11011001 10100111 10100010 00000010 10110011
13 :  01000110 10110010 11000111 01110111 00101011 10010111 11010000 00100010

14 :  01001111 10100110 00100010 10111001 01101110 10010001 01100010 10101100
14 :  00101101 11011011 10010000 00100100 11000010 10100111 00100000 00011111

15 :  01101110 10010001 01100010 10101100 10101010 11100001 00101111 10000010
15 :  11000010 10100111 00100100 00011111 01110011 01000001 11000011 00101011

16 :  01001010 11100001 10101110 00100110 00011000 11110001 01101110 01101100
16 :  01110011 01000011 11000011 00101011 00101001 11011011 01001100 01011101

Ciphertext 1 :  00101100 11001011 01001011 10011001 00010100 01101101 10100101 00100110
Ciphertext 2 :  11111011 10111010 00011101 01010111 11000010 11101101 11011000
```

Figure 3.6 - Table 3.2 ; Avalanche Effect in DES; change in plaintext

Key 1	:	10000101 11001101 11001101 00011100 00011100 11010000 01000110 00011010
Key 2	:	00000101 11001101 11001101 00011100 00011100 11010000 01000110 00011010

Round		
0 :	11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
0 :	11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
1 :	11111111 11111111 11111111 11111111 01000100 01111101 11111101 00010000
1 :	11111111 11111111 11111111 11111111 01000100 00111101 11111101 01010000
2 :	01000100 01111101 11111111 00010000 00101010 11011000 10101000 01010010
2 :	01000100 00111101 11111111 01010000 00101110 11011001 11011100 01000100
3 :	00101010 11011000 10101000 01010010 00010011 01010000 01000110 11001000
3 :	00101110 11011001 11011100 01000100 01111011 11111101 01111001 00101101
4 :	00110101 01010000 01000110 00100011 10110000 01100010 01010000 10100001
4 :	11111011 11101110 01111001 00101001 11100010 11100101 01111101 11001000
5 :	10011011 10110000 01100010 01010001 00011001 00001001 01011001
5 :	01001010 11100010 01111101 11001000 01000001 00010011 00101011
6 :	00110101 00001110 00001001 01011101 11001101 00010010
6 :	00100001 11110000 00010011 10001110 01000111 01111010
7 :	11001101 00001111 11101001 00010010 10000001 01010000 00110110 00110011
7 :	10001110 00110110 01000111 01111010 10100001 00001100 00110001 01110110
8 :	10000011 01010000 00110010 10111010 10011100 11000100 00010110
8 :	10100001 00001100 00110001 01110110 01011111 01011011 10111001 00110010

(Continued on next page)

– 34 –

```
 9 :  10111010 10011110 11000100 00010110 01000111 10011001 11001001 00011001
 9 :  00111011 01011111 10111001 00110010 10100011 00001110 11000010 01011110

10 :  01000111 10011000 11001001 00011001 00010110 10100011 11000011 01010100
10 :  10100011 00001110 11000010 01011010 00000010 10000011 11011111

11 :  00101110 11010011 11000011 01010100 11000010 00100101 11110111 01001101
11 :  00000010 00010100 10000011 11011111 01011111 11001101 01001100 01000011

12 :  11000010 00100101 11110111 01001101 11001100 00000101 01101100 11011001
12 :  01011110 11001011 01011011 01000100 01000111 01010001 01111100 00011111

13 :  11011100 00000101 01101100 11011001 01001111 10100110 00000010 10111001
13 :  01000111 01010001 01111100 00011111 10111011 00011011 10101010 01110000

14 :  01001111 10100110 00000010 10111001 01101110 10011001 01100010 10101100
14 :  10111011 01011101 11001010 01110010 01101010 10101101 10001111

15 :  01101110 10011001 01100010 10101100 01001010 11100001 00101111 10000110
15 :  11110010 01101010 10101101 10000011 11111000 01010001 01010001

16 :  01001010 11100001 00101111 10000110 00011000 00011000 11110001 01101110
16 :  11011010 01000011 11111000 01010011 00111110 01001010 01111010

Ciphertext 1 :  00101100 11001011 01001011 10011001 00010100 01101101 10100101 00100110
Ciphertext 2 :  00110010 11100111 01010000 11001101 01001001 10111111 10001000
```

Figure 3.7 - Table 3.3 : Avalanche Effect in DES; Change in Key

- 35 -

of the plaintext undergoes a complicated process together with the
key vectors K(i) at each round i. Let this process be denoted by
function f. Then,

$$f \; [R(i), \; K(i+1)] \; = \; f \; [E(R(i)) \oplus K(i+1)]$$

where $E(R(i))$ denotes the expanded right half of plaintext at round i

$\oplus$ denotes Exclusive-or operation

Complementing both $R(i)$ and $K(i+1)$ therefore does not change the value
of f and

$$f \; [R(i), \; K(i+1)] \; = \; f \; [\bar{R}(i), \; \bar{K}(i+1)]$$

Because

$$L(1) = R(o)$$
$$R(1) = L(o) \oplus f \; [R(o), \; K(1)]$$

complementing plaintext X means complementing $L(o)$ and $R(o)$.
Complementing key K means complementing $K(1), \ldots, K(16)$. Therefore
this results in complementation of $L(1)$ and $R(1)$. By induction, this
can be extended to $L(2) \; R(2), \ldots, L(16)R(16)$ and hence to ciphertext
C.

Because of the complementary property of DES, if a
cryptanalyst could obtain $Y_1 = E_K(X)$ and $Y_2 = E_K(\bar{X})$ for an arbitrary
X, he could reduce the size of the key space he must search from
2^{56} to 2^{55}. The cryptanalyst enciphers X with all keys K that start
with a 'o'. The resultant ciphertext is compared with Y_1 and Y_2. If
$C \neq Y_1$ the key in use is not K, if $\bar{C} \neq Y_2$ the key in use is not $\bar{K}$
(which starts with a '1'). That is, effectively this symmetry reduces the
search effort by 50 percent under a partially chosen plaintext attack.
An example illustrating this complementary principle of the DES
obtained using the DES software is shown in Figure 3.8. This
complementary behaviour can be avoided by selecting the S-boxes
using one or more key bits directly, instead of employing the modulo
2 operation. On the other hand, this complementary property can
be used advantageously for testing purposes.

- 36 -

Page 36

Key: 1000100010001000100010001000100010001000100010001000

Plaintext: 0011011101010011010100110000100011100101110100001111111110

Ciphertext 1: 1011000101100001011011011001001010111111110011110011

Complementation of key: 0111011011010000110111011011011011011011011011010111

Complementation of Plaintext: 1100100001001011010011001001110101011010100010101110000001

Ciphertext 2: 0100110100011110010010011001101101110101010100000110001100

Ciphertext 2: Complement of Ciphertext 1.

Figure 3.8 : Complementation in DES

- 37 -

3.5   Design Criteria

Although the DES algorithm has been made public, the design criteria behind the specific choice of some permutations and substitutions remain still classified. For example, one design criterion for the DES was that the permutation schedule must ensure that each output (ciphertext) bit is a function of all input bits (plaintext and key) after a minimum number of rounds. It is reported that [3] the initial approach of random selection of the parameters such as the permutations and substitutions had to be abandoned because they introduced weaknesses into the algorithm. Instead the parameters were randomly generated and tested against the design criteria. It is believed that a significant portion of the random designs were rejected in this process. One of the most important parameters in the algorithm is the S-boxes and they are considered in the next section.

3.5.1   S-Boxes

All the operations involved in the DES algorithm except the S-box mappings are linear in binary arithmetic. Therefore the S-boxes play an important role in the security of the DES. It is therefore crucial that the S-boxes not be affine or the whole algorithm would be affine. A study by Hellman et al [19] found that none of the S-boxes are affine. But they point out that there are a number of questionable quasi-linear structures which may tend to weaken the algorithm. But currently no methods are found which could exploit these quasi-linearities in reducing the search effort. The designers of DES argue that it should not be surprising that certain parameters contain some structural properties different from those expected to result from the use of purely random selection. Further they claim that [3] their design effort showed that carefully selected S-box functions produce a much stronger algorithm than one based on random designs.

The DES S-boxes are non-invertible 6 to 4 non-linear mappings. Each S-box is found to have the property that changing any single bit of the input while keeping the others constant always changes at least two bits in the output. This property results in the error propagation property and is essential to avoid the key clustering attack [2]. This property plays an important role in causing the avalanche effect mentioned earlier. On average it is

- 38 -

found that the number of output bits that change in the DES for a
single input is above 2 (2.5 - 2.6) [19].

        Another principle underlying the design of the S-boxes is
that the difference between the number of zeroes and the number of
ones in the output when any one of the input bits is held constant
is a minimum. This design criteria is believed [20] to strengthen
the algorithm.

3.5.2 Initial and Final Permutations

        The plaintext input to the DES is first permuted using IP
before transformed by the sixteen rounds. The output of the final
round undergoes an inverse permutation IP^{-1} where IP^{-1}. IP = I, the
identity permutation. It is important to note that in a single
encryption process, IP and IP^{-1} do not cancel each other as IP
operates on the plaintext and IP^{-1} operates on the ciphertext produced
by the 16th round. On the other hand, the permutation IP^{-1} performed
during the encryption process is cancelled by the permutation IP
performed at the beginning of the decryption process.

        It seems that the permutations IP and IP^{-1} do not have any
cryptographic significance. Possible reasons for their inclusion in
the algorithm are that they facilitate the implementation of the
algorithm for a particular IC layout or they result from the way the
data is loaded into the DES chip. These permutations IP and IP^{-1} can
be absorbed by proper layout into the main algorithm and they do not
slow down the algorithm. The algorithm designers claim that the
permutations do have cryptographic value and they contribute to the
security of the DES. This could be possible as follows: In many
applications, ASCII characters are used for the plaintext. Because
of the particular ASCII coding and the frequency distribution of the
English alphabets, the distribution of '1's and 'o's will not be
uniform. The IP permutation groups every i^{th} bit of each input byte
into j^{th} byte of the permuted output. This preprocessing of the
plaintext may redistribute the '1's and 'o's in a uniform way before
inputting to the 16 rounds and thus may enhance the security.

3.5.3 P-Permutation

        The function of the P-permutation in the DES as in all
substitution-permutation ciphers is to provide the element of
diffusion [7]. This concept was first introduced by Shannon to

- 39 -

spread the dependencies of output bits on input bits through
successive rounds to achieve total mixing.

3.6   Criticism and Weaknesses of DES

The controversy surrounding the DES essentially comes from
two main objections given below:

(i)   the small size of the key space   and

(ii)   the algorithm's unpublished design principles.

Other criticisms of the algorithm include the small number of rounds,
the two 'redundant' permutations IP and IP^{-1} and a relatively simple
key schedule operation.

To investigate these criticisms, two workshops were
organized by the National Bureau of Standards (NBS). The first
workshop analysed the complexity of the algorithm and restated that
no short cut methods were found [20]. The second workshop concluded
that a key searching machine cannot be built before 1990 and will
cost several tens of millions of dollars with a probability factor
of being available even then of about 0.1 to 0.2 [21]. Let us now
consider the major criticisms of DES in turn.

3.6.1 The Key Length

By far the greatest source of controversy has been the choice
of 56-bits as the key length. The length of the key determines the
feasibility of key trial. Critics argued that the length of the key
is too small and that the key space is amenable to exhaustive search.
Diffie and Hellman [22] disputed the NBS claim that trying all
possible keys is not economically feasible and estimated that a DES
key can be recovered by exhaustive search for approximately $5000
worth of computation time on a special purpose machine. The special
purpose machine would consist of a million LSI chips and could try all
the 2^{56} keys in one day. The cost of such a machine using the semi-
conductor technology of the mid-seventies was estimated to be in
20 million dollar range. More recently [84], the cost of a 2-day
average search time machine is estimated to be around 50 million
dollars. This meant that it is out of reach of most groups with the
possible exception of government security agencies. It was predicted
that in 10 years, the machine will cost approximately 200,000 dollars

- 40 -

and recovery of one key will be around 50 dollars. Such attacks assume that the DES is being used in the standard code book mode.

Alternative brute force attack on the DES algorithm is to use a table look-up approach. In this approach all the ciphertexts resulting from enciphering a chosen plaintext are stored and sorted using all the 2^{56} keys. The amount of memory required for this attack is of the order of 56 $(2^{56}) \doteq 4 \times 10^{18}$ bits which is enormous. This would require 4 billion magnetic tapes (about 10^9 bits per tape), costing about 80 billion dollars ($20 per tape) [4]. In [23], Hellman proposed a trade off between time and memory. Like table look up approach, this technique requires precomputation and storage of a table. However, with this technique, the table size is only of the order $O(n^{2/3})$ instead of the previously $O(n)$ where n = $56(2^{56})$ bits. This technique also requires searching and the search time is of the order $O(n^{2/3})$ instead of $O(n)$ where n = 2^{56}. Hellman predicts that with proper combination of precomputation and searching, a machine may be constructed that would recover the key with high probability in one day. This in turn led to the suggestion that a bigger key length of 128 bits is required.

Further the key space is reduced if the key is not chosen as a 56 randomly selected bits. For instance, if the key is selected as eight characters, each character being one of say 64 possibilities then the number of distinct keys which need to be tested is $(64)^8$ = 2.8×10^{14} which may be quite feasible.

The key space is also reduced by the invariant property of DES under complementation. As seen in Section 3.4.2, this symmetry can be used to reduce search effort by 50% under a partially chosen plaintext attack.

3.6.2 Unpublished Design Principles

The second major criticism of the DES algorithm is that the design criteria behind the choice of some of the parameters are not disclosed publicly. Those parameters are the S-boxes, fixed permutation P and the key schedule operation.

The classified design principles led some critics to argue that deliberate trapdoors might have been incorporated in the algorithm by the designer to his advantage. This argument is based on the fact that it is possible to design 'innocent-looking' S-boxes which contained trapdoors [19]. Further, regular structures were

– 41 –

discovered in certain parts of the DES algorithm which were surpris-
ingly similar to a type that can be used to build a trapdoor into the
system [19, 24]. Like the complementation property, it may be possible
to save another factor of two with carefully chosen S-boxes thus
saving a total of 75% over exhaustive search. Lexar Corporation [24]
examined the properties of the substitutions and permutations and
reported some 'peculiar' properties of the S-boxes but to date no
feasible cryptanalytic technique for DES has been found.

3.6.3 Number of Rounds

DES is also criticized for having a small number of rounds
and it is suggested that the number of rounds should at least be
increased to 32 [19]. The number of rounds controls the mixing of
input bit values, that is, the amount of diffusion introduced by the
co-ordinate permutation. An increase in the number of rounds will
produce a greater avalanche effect. A two round DES was successfully
cryptanalyzed [19] by exploiting the correlation between L_2R_2 and the
input. Such correlation is believed to disappear after eight
rounds [6].

Some statistical analysis to test the randomness of the
DES output sequences and any correlation between plaintext input and
ciphertext output is considered in Chapter 6.

3.6.4 Key Schedule Algorithm - Weak and Semiweak Keys

The cryptographic strength of the DES will be reduced if
some of the internal subkeys derived from the external key are the
same. In the extreme case, if all the internal keys are the same
then the key space is reduced to 2^{48}. Referring to the key schedule
algorithm given in Figure 3.3, this situation occurs whenever all bits
in the register C are all ones or zeroes and the bits in register D
are all ones or zeroes. There are four such weak keys altogether [3].
These weak keys also have the property that there is no difference
between the operations of encipherment and decipherment, that is,
$E_kE_k(X) = X = D_kD_k(X)$ for weak keys. There is another set of keys
defined as semiweak which have the property that only two different
internal keys are produced each occurring 8 times. There are six
such semiweak keys and they have the property $E_kE_{k'}(X) = X =$
$E_{k'}E_k(X)$, where k and k' are distinct semiweak keys. Further there
are 48 other external keys which produce only 4 different internal

- 42 -

keys.

These special keys do not pose any threat to the security of the algorithm because the number of such keys is small compared to the total set. Provided that the keys are randomly selected the probability of choosing such a key is very small. Further these keys can easily be avoided during key generation.

The key schedule algorithm can be improved by incorporating non-linearity in the internal key generation process [39]. A suitably shifted external key can be inputted to a substitution-permutation process in each round to produce the internal key as shown in Figure 3.9. The scheme allows, in addition to the enlargement of the key space, other cryptographically desirable properties such as error propagation. As the substitution permutation function is already available in the algorithm, this requires minimum additional hardware.

3.7   Cryptanalysis of DES

Cryptanalytical methods can be divided into two subcategories namely deterministic or analytical and statistical methods.

In a deterministic approach, the cryptanalyst attempts to express the desired unknown quantity (such as the key or the message) in terms of some other known quantity or quantities (such as given ciphertext or given plaintext and corresponding ciphertext). One method of attack is therefore to represent the 64 ciphertext output bits as functions of the 56 key bits in a known plaintext attack and try to solve the 64 non-linear equations over GF(2) for 56 unknowns. This problem is an NP-complete problem [8] and hence is difficult to solve in general. Such an analytical attack of the DES was proposed in [25]. The attack proved to be manageable with affine S-boxes but infeasible when the real DES was considered. IBM and NSA also reported to have conducted similar attacks as part of their validation process with no short cut solutions being found.

In the statistical approach, the cryptanalyst attempts to exploit statistical relationships between plaintext, ciphertext and key. To thwart statistical attacks, the algorithm's output should be pseudo-random even for highly structured inputs. In other words, for a large set of plaintext and key inputs one must not be able, on the basis of statistical analysis, to reject the hypothesis that the

- 43 -
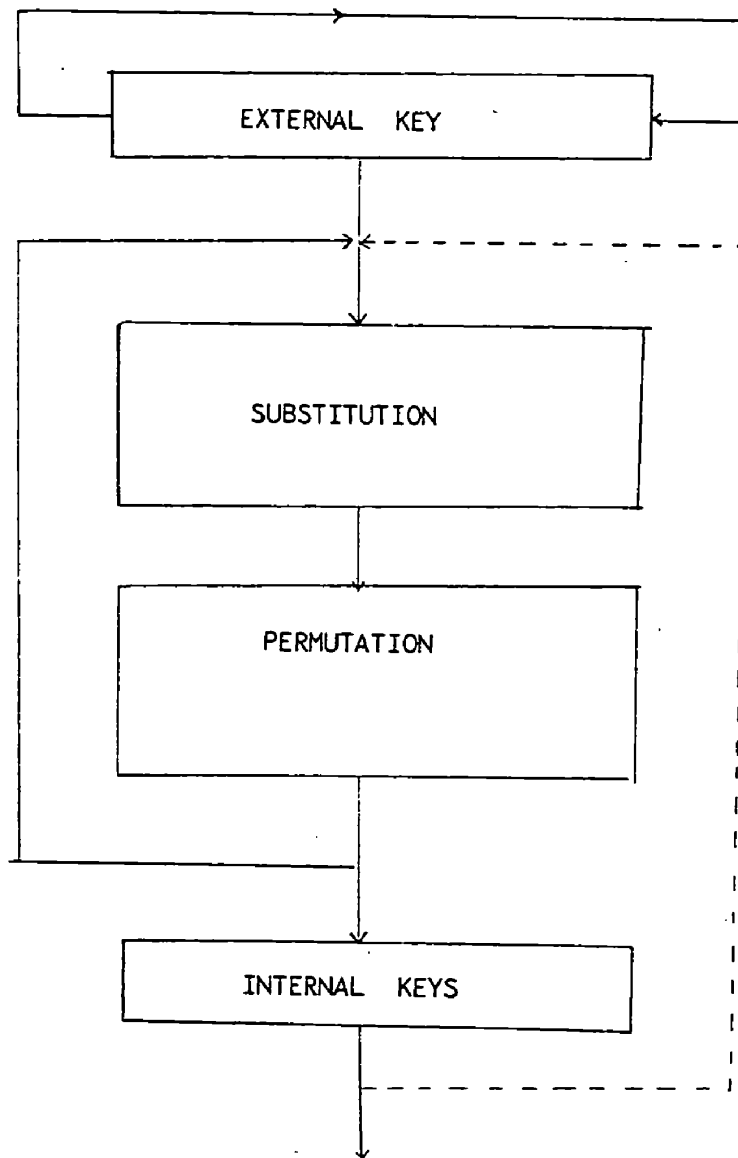
Figure 3.9 - Non-linear Key Schedule Algorithm

output bit stream is random. Some fundamental statistical tests have been performed on the DES output streams and they are discussed in Chapter 6.

Finally one should mention the brute force attack to find the unknown quantity such as the message or key by using a direct search method, trial and error or exhaustion. As mentioned earlier the key length is an important factor on this type of attack and greater the key length bigger the work factor to perform the key exhaustion.

3.8   Discussion

If an algorithm is defined to be cryptographically strong when the algorithm complexity is such that,

(i)   using all known shortcut solutions, it is not practically possible to solve for the key or for the message

(ii)   it is too costly to employ simple methods such as key exhaustion because too much time and/or hardware are required

then,despite a lot of controversy surrounding the DES, DES can be considered to be a cryptographically strong algorithm. As far as the author is aware not a single successful 'break' has been developed that can produce or determine the unknown key. Further,the life of the DES can be extended (until a shortcut solution is found) by enciphering the data two or more times with different keys. Double encryption will significantly increase the difficulty of intrusion requiring of the order of 2^{56} words of memory and 2^{55} operations [3]. However it is preferred that such a multiple encryption process is specified as part of the algorithm itself. The security can also be improved by employing DES under various chaining modes. They are considered in subsequent chapters. In account of this, the DES algorithm has been chosen to be employed in the design of an encryption interface unit to be used in a communication network. This forms the subject of the next chapter. This chapter is concluded by noting that the DES can be used as a building block for designing more sophisticated algorithms. For instance,the DES itself can be used as

- 45 -

the cipher function f in Figure 3.2. This gives rise to the scheme
shown in Figure 3.10 where   a   block of 128 bits forms the
plaintext.
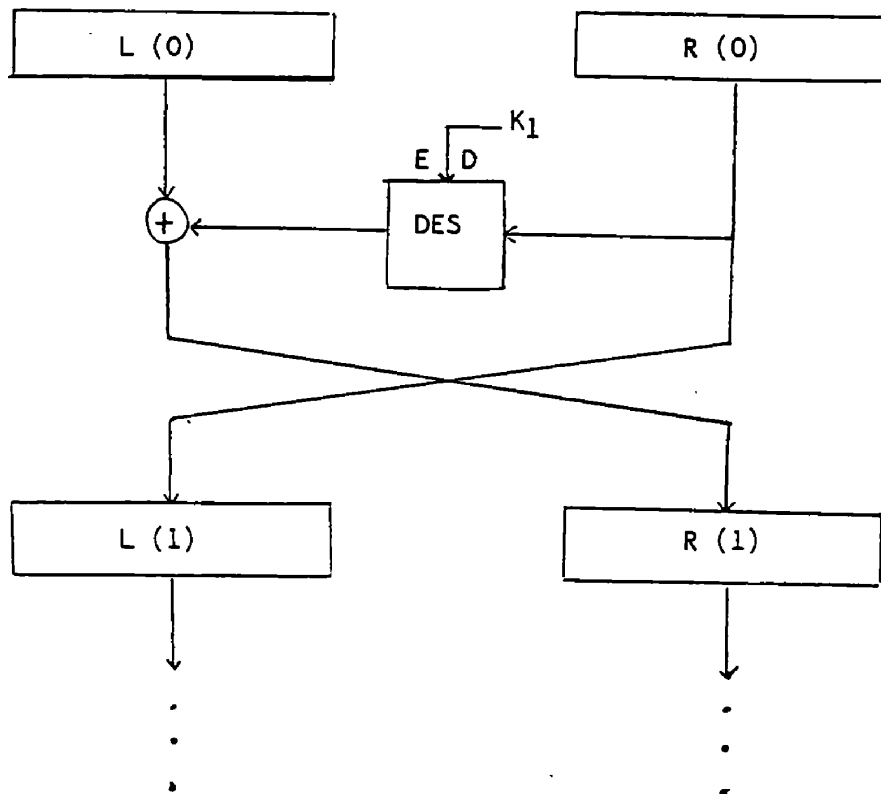
Figure 3.10 - Extended DES Algorithm

- 46 -

CHAPTER 4

DESIGN OF ENCRYPTION SYSTEM: SYSTEM HARDWARE

4.1 System Requirements

The requirements for the encryption system may be class-
ified into three groups namely

(i)   security requirements

(ii)   Operator requirements and

(iii)   technical requirements.

4.1.1 Security Requirements

Briefly, the security requirements are as follows:

(a)   The encryption algorithm used in the system should be
        cryptographically strong. That is, it should withstand at
        least all the major cryptanalytic attacks described in
        Section 2.2.

(b)   There should be a large number of user selectable keys to
        prevent the opponent employing exhaustion techniques to
        determine the key selected.

The DES algorithm outlined in Chapter 3 meets these conditions very
well.

(c)   The use of the algorithm within the system should be carried
        out in such a way that the system as a whole is at least as
        strong as the initial strength of the algorithm itself.

(d)   At no time, keys used by the algorithm must be stored in
        plain form within the system as otherwise it would enable
        any unauthorized user to its recovery.

(e)   Key changes should be made easy to implement.

(f)   A high degree of physical security measures must be provided
        to protect the system against intentional or accidental

- 47 -

threats.

4.1.2 Operator_Requirements

The system should be made 'user friendly' thus allowing any
user to operate it efficiently with little training. This aspect is
very important in the case of large networks. The unit should also
be easy to install.

4.1.3 Technical Requirements_

The technical requirements for the encryption system depend
to a great extent upon the use to which the system will be put. This
particular encryption system is designed to be used in the following
applications:

(a)   transmission/reception of encrypted/plain data in a point-
        to-point communication security system not containing a
        host computer;

(b)   a local encrypted/plain data storage/retrieval system using
        floppy disks;

(c)   transmission/reception of encrypted/plain data over the
        communication link to a host computer and storage/retrieval
        of encrypted/plain data using a host computer, in particular
        with the Prestel Viewdata computer.

The requirements for a storage/retrieval encryption system
are quite different from those necessary in a point-to-point
communication system. Some of the aspects to be considered are:

1.   Real time processing and transmission delays: These are
        particularly important in point-to-point communication
        where information flow is irregular and messages may not be
        of fixed format.

2.   Transmission data rates: This aspect is important in
        determining whether asynchronous or synchronous transmission
        is needed. We are concerned here with only asynchronous
        transmission of data rates up to 1200 bauds. Further the

- 48 -

communication with Prestel computer is asymmetric full
duplex in the sense that it requires 1200 bauds for
transmission and 75 bauds for reception.

3.   Where a mixture of plain and encrypted texts is required to
        be sent, switching between encrypted text mode and plain-
        text mode should be easy. This should be carried out within
        the message so that the receiving unit can follow identical
        mode changes.

4.   In the case of storage/retrieval system with a host computer,
        since the encrypted information is likely to pass through
        the host computer control unit where certain control
        characters are generated and detected, there is a need to
        prevent the occurrence of these special characters in the
        encrypted information. This implies that the encryption
        system should be made transparent to the host computer
        protocols. This is particularly relevant when the system is
        used in Prestel network.

5.   Standard editing facilities such as character delete, line
        delete etc. must be available to the system users.

4.2   <u>General System Description</u>

        A symmetric encryption-decryption system in an end-to-end
communication configuration is shown schematically in Figure 4.1.
The operation of the system can be summarised as follows. Referring
to Figure 4.1, the plaintext from an Apple terminal forms the input to
the encryption system. The plaintext is a combination of any of the
alphanumerical characters from the keyboard. The key required for the
encryption process is entered from the terminal keyboard under the
user control prior to the commencement of the session and is stored
in an inaccessible area within the encryption device. The encryption
system operates on the input sequence using the key to produce a
ciphertext sequence. This cipher is transmitted through input-output
communication controller to the modem. The modem converts the binary
data pulses to analog frequencies using Frequency Shift Keying (FSK)
scheme for transmission over the public switched telephone network.

- 49 -
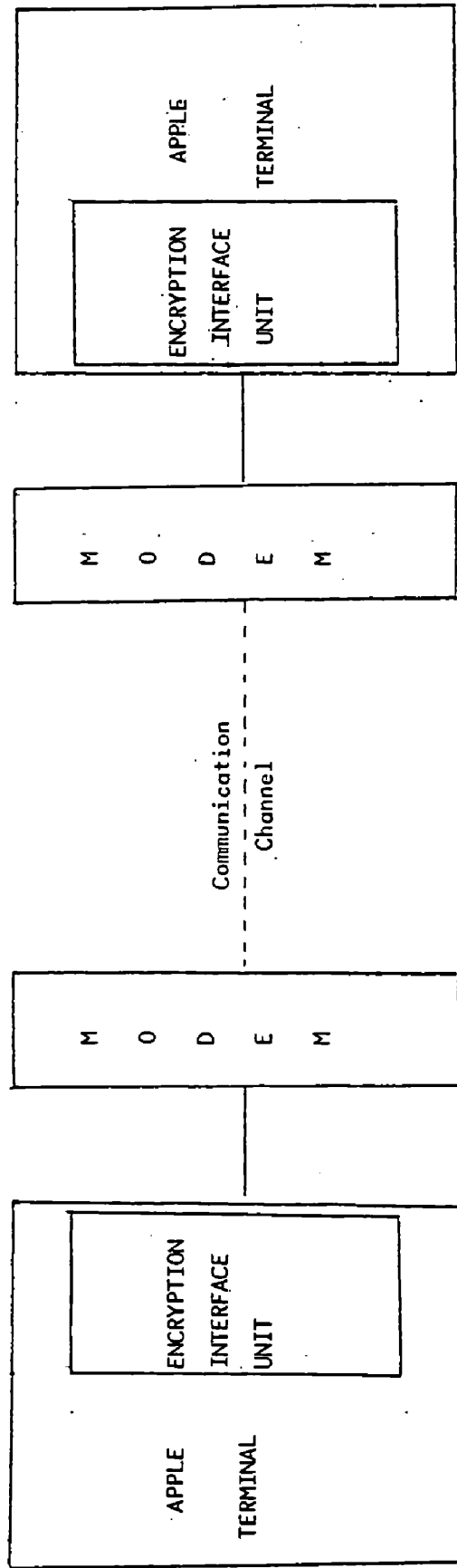
Fig. 4.1 - Point to Point System Configuration.

- 50 -

At the receiving end, another modem converts the FSK signals back to digital form which is then deciphered to regenerate the original plaintext. This deciphered information can then be displayed on a Visual Display Unit or printed on a line printer.

The communication link between Apple microcomputers is a half-duplex one thus allowing transfer of data in either direction but not at the same time.

From Figure 4.1 it is seen that in this implementation, the encryption system is incorporated as an in-built feature of the terminal, rather than as a separate unit. The built-in implementation is superior to stand-alone implementation in the area of access prevention as the former technique greatly reduces the chances of detection between the terminal and the encryption unit where the text is in plain form. However this technique may be difficult to implement in existing systems and can require major redesigns. This is where the stand-alone unit has the advantage in that it can simply be inserted into existing networks with little or no impact to existing equipment.

The different sections which constitute the system are:

1.   Apple microcomputers - comprising 6502 microprocessor, memory, diskette interface and Language card.

2.   The encryption interface unit.

3.   The modulator-demodulator equipment.

4.2.1 Apple Microcomputer

In this project, Apple microcomputers have been used to form a two mode network. The Apple microcomputer is a 6502 microprocessor based system with diskette interfaces, disk driving units, Integer card and a standard keyboard. The processor can directly reference up to 64 kilobytes of memory. Along the back of the Apple's main board, is a row of eight 'slots' or peripheral connectors. In seven of these slots, peripheral interface boards designed for Apple system can be installed. In slot 0, Integer firmware card is installed. Each slot has a 2K-byte of common shared memory associated with it, with a view to holding programs or driving subroutines of the peripheral interface card. The designed encryption interface card is

- 51 -

installed in slot number 2 providing a memory range of C800 to CFFF.

4.2.2 Encryption_Interface_Unit

The unit is designed utilizing the 6502 microprocessor of the Apple microcomputer system. A block diagram showing the various sections of the unit is illustrated in Figure 4.2. The sections are:

1.   Data Security Device.

2.   Serial Input-Output Controller.

3.   RS-232C Interface Circuits.

4.   Timing Circuits.

5.   Decoding Circuits.   •

6.   Memory.   ·

4.2.2.1 Circuit Description and Operation

A complete circuit diagram is shown in Figure 4.3. Some important aspects of the circuit are now briefly considered.

4.2.2.2 Data_Security Device_

As mentioned earlier, it has been decided to use the DES algorithm in hardware in the design of the interface unit. The choice of the data security device incorporating the standard was restricted due to limitations of not only their availability but also their access. A Western Digital device has been used for this purpose because of its ready availability. The device is made in n-channel silicon gate MOS technology and is TTL compatible on all inputs and outputs. The device can be interfaced to a wide variety of processors though it is tailored to the Intel 8080A class microprocessor. The device performs 64-bit block encryption and decryption using 56-bit key. The block diagram, shown in Figure 4.4, illustrates the internal structure of the device. It contains 64-bit data register, 56-bit key register, an 8-bit command/status register plus the necessary logic to check key parity and implement the DES algorithm. The device has a single 8-bit data bus buffer with tri-state operation through which

- 52 -
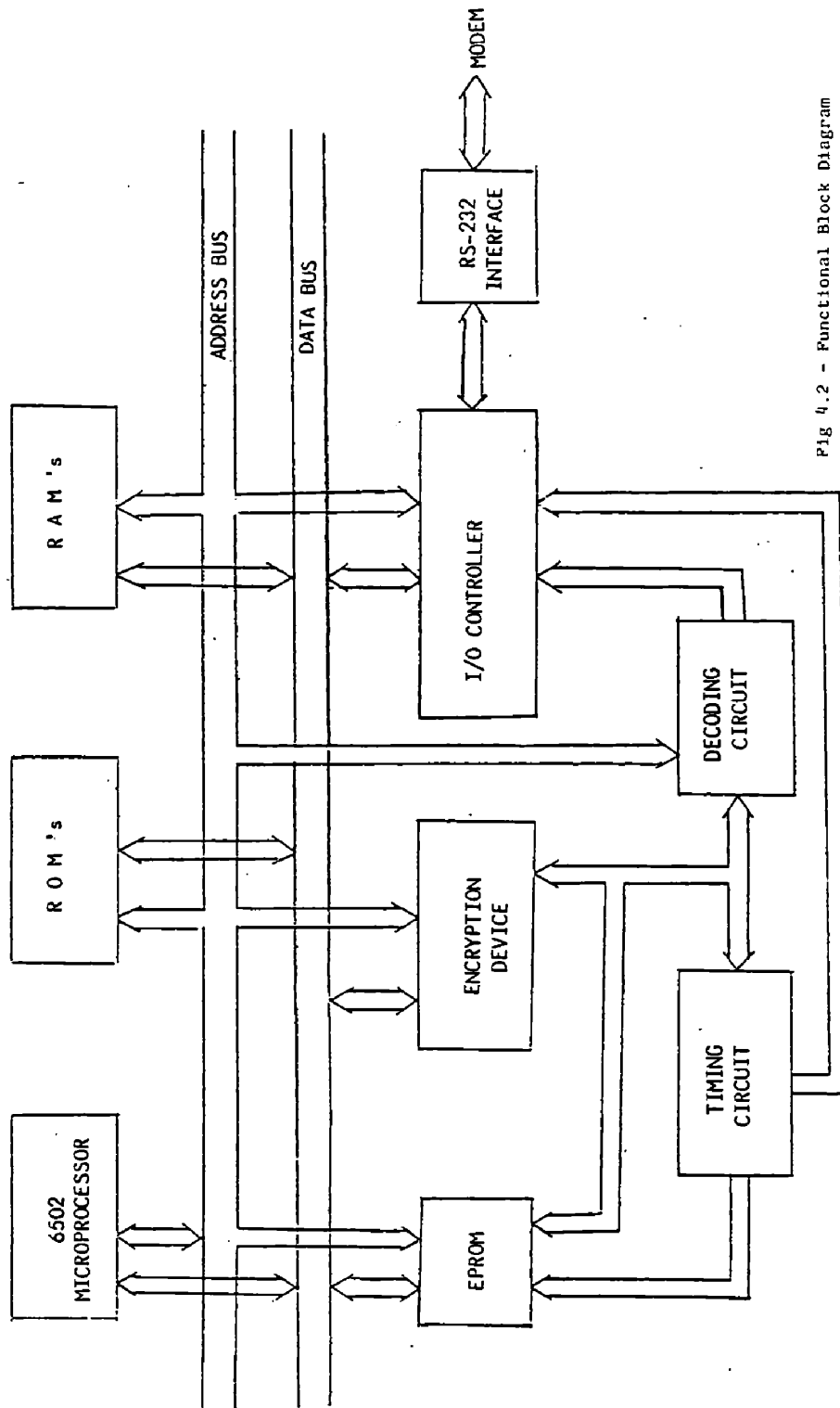
Fig 4.2 - Functional Block Diagram

- 53 -
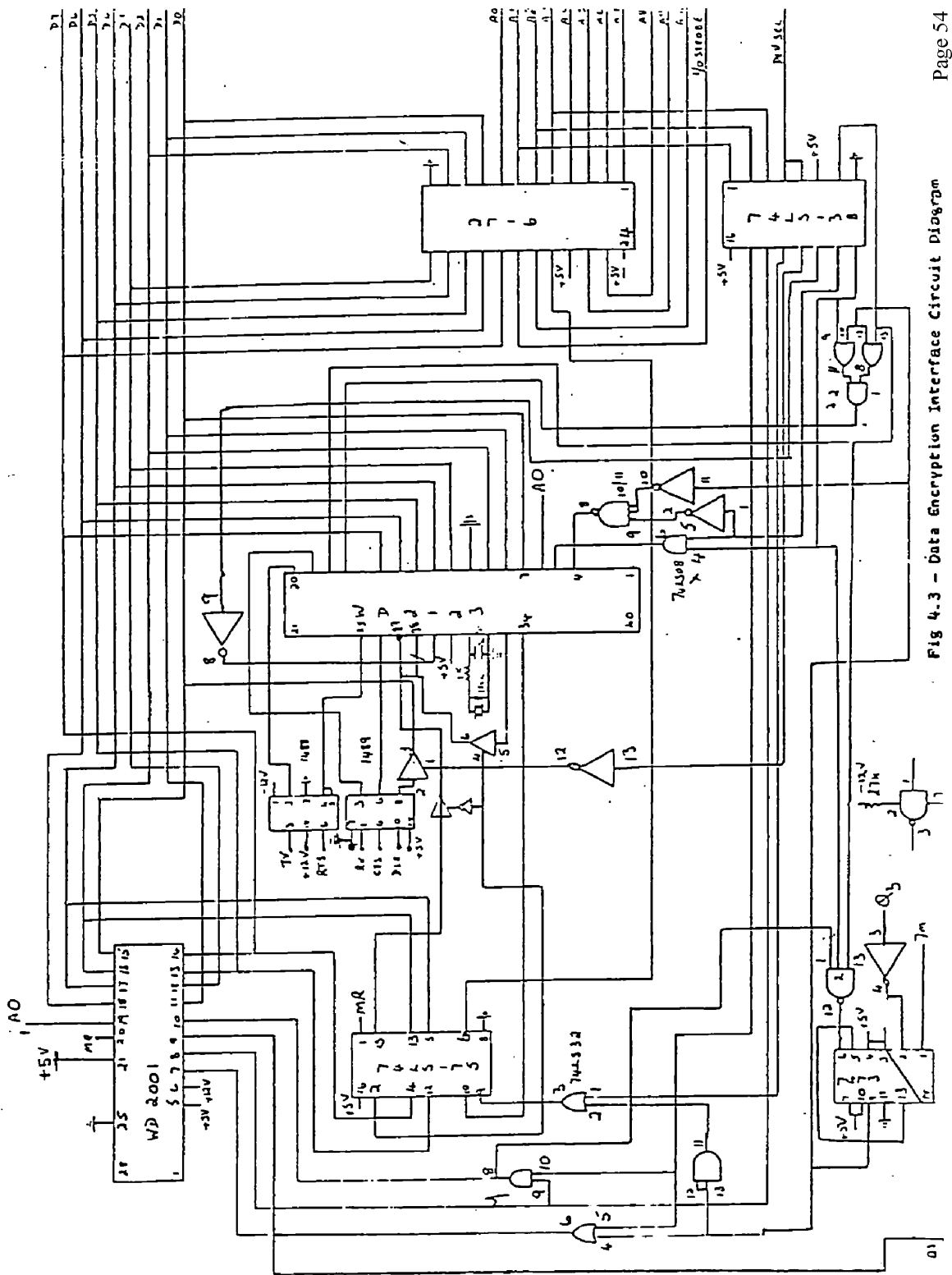
Fig 4.3 — Data Encryption Interface Circuit Diagram
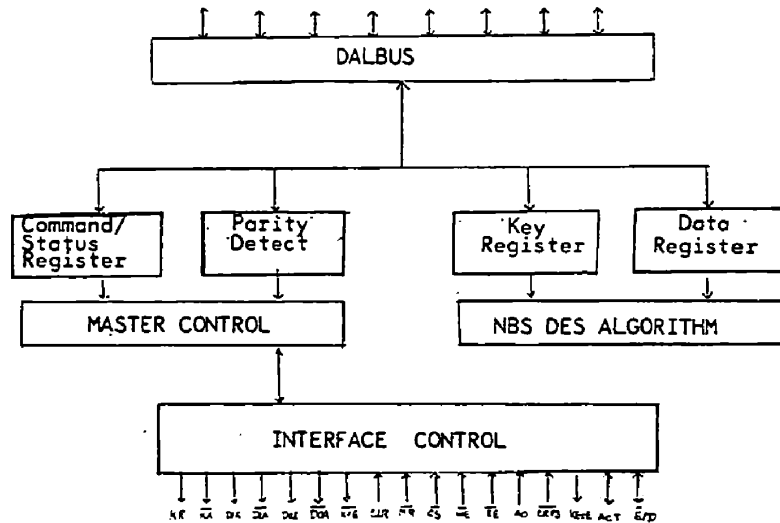
Fig. 4.4 Internal Architecture of Data Security Device
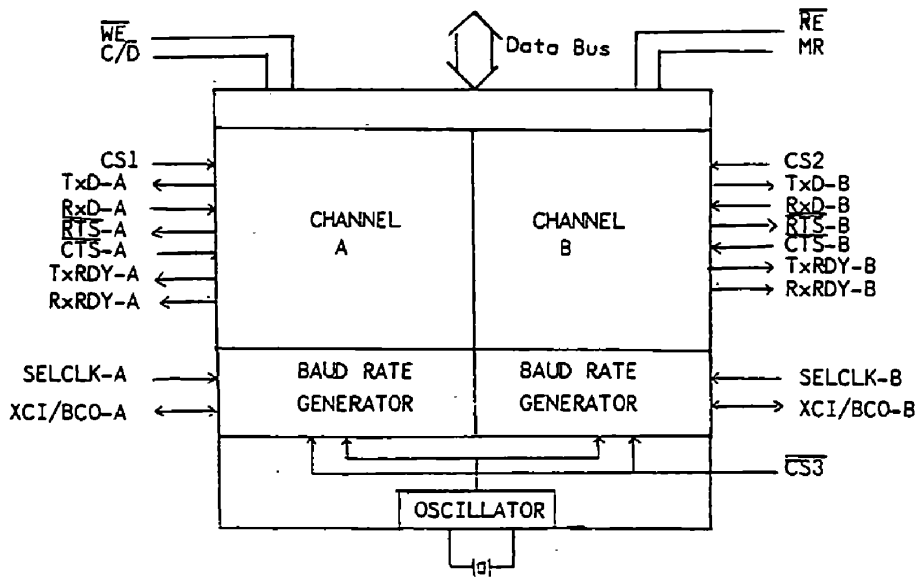
Fig. 4.5 - Internal Architecture of Serial I/O Controller

– 55 –

data may be entered into the key register or the data register. Output data from the status register or the data register is also switched through the data bus buffer. The device can be programmed either via the input lines or via the data bus. In this design, the second approach has been adopted. The programming of the device will be briefly discussed in system software (Chapter 5).

4.2.2.3 Serial_Input-Output Controller

There are a number of such devices which would be suitable for transmission and reception of plain and enciphered data. Again a Western Digital device, dual enhanced communications element (DEUCE) has been used. It is primarily designed to operate in an 8-bit microprocessor environment.

The controller contains two independent full-duplex asynchronous receiver/transmitter channels and two internal baud rate generators associated with the two channels. A block diagram of the internal architecture of the device is shown in Figure 4.5. Communications between the controlling CPU and the two receiver/ transmitter channels or the two baud rate generators occurs via the 8-bit data bus through a common set of bus transceivers. The use of this complex device has enabled the development of a compact encryption system.

Each channel has associated with it a number of registers such as the Transmit Holding Register, Receive Holding Register, Command Register, Mode Register, Status Register etc,which can be programmed to transmit and receive asynchronous serial data. It performs serial to parallel conversion on data characters received from the modem and parallel to serial conversion on data characters received from the CPU. The CPU can read the status of either channel at any time. Status information includes the type and condition of transfer operations being performed by the device as well as any transmission error conditions. Internal control of each channel is achieved by means of two internal microcontrollers one for transmit and one for receive. A block diagram of one of two communication controllers is shown in Figure 4.6. The control registers,various counters and external signals provide inputs to the microcontrollers, which generate the necessary control signals to send and receive serial data according to the programmed protocol.

The device also contains Baud Rate Registers which can be

- 56 -
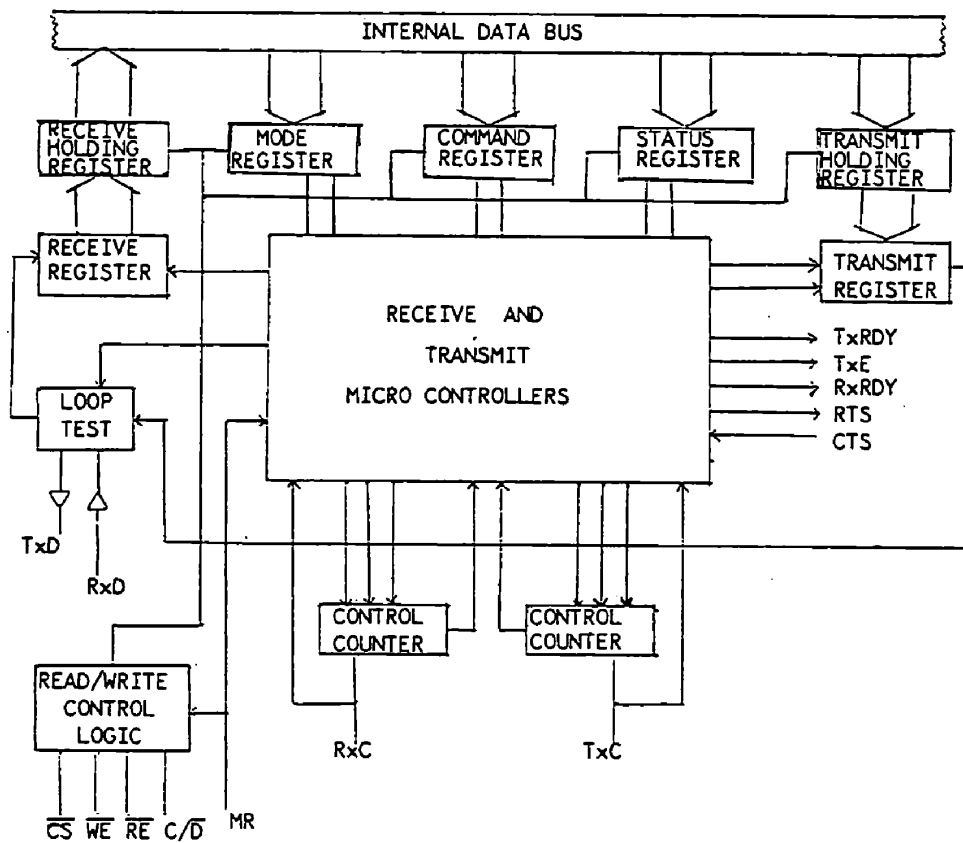
Fig. 4.6 - Block Diagram of a Receive/Transmit Communication Controller

programmed to desired data rates by loading them with the appropriate
number of bauds. The contents of these registers are then decoded and
then addressed to a frequency select ROM for the generation of proper
frequency by the divider circuitry and the control logic during
operation. A diagram of one of the two Baud Rate generators is shown
in Figure 4.7. The baud clock source is selected using select clock
pin (SELCLK) in conjunction with the clock select bit (CR1) in the
Command Register (see Figure 4.8). When the bit CR1 is high, the
external clock mode is selected. This means that the Transmit and
Receive clocks are internally tied together and SELCLK input determines
whether those clocks are driven from the internal baud rate generator
(SELCLK high) or from the external clock input XCI/BCO (SELCLK low).
If the internal baud rate generator is selected, then the external
clock input becomes a baud rate generator clock output. When the
bit CR1 is a logic '0' then the internal clock select mode is chosen.
The transmit clock is driven by the internal baud rate generator clock
and the receive clock is driven by the SELCLK input. The XCI/BCO pin
then becomes the baud clock output, the transmit clock. The inputs
to the SELCLK and XCI/BCO pins are derived using the simple logic
circuit shown in Figure 4.8(a).

When the signal present on the line marked X is low and the
signal at Y is high, this enables the tristate Z1 and sets the SELCLK
to be high. With the bit CR1 at logic '1', this would result in both
the transmit and receive clocks tied together and driven by internal
baud rate generator. This mode shown in Figure 4.8(b) is adopted for
point-to-point communication between two Apple microcomputers.

When the signal present on the line marked X is high and
the bit CR1 is at logic '0' then the tristate Z2 is enabled. The
XCI/BCO output is then connected to the SELCLK input which drives
the receive clock. The transmit clock is driven by the internal baud
rate generator. This configuration thus allows to transmit and
receive at different frequencies using only one channel without using
an external baud rate generator. This is one of the main reasons for
using this I/O controller. This mode of operation, shown in Figure
4.8(c), is adopted when the interface is used in the Prestel network
where the transmission data rate is 75 bauds and receive data rate
is 1200 bauds.

The signals Clear To Send (CTS) and Request To Send (RTS)
to and from the device are used in controlling the modem. These

- 58 -
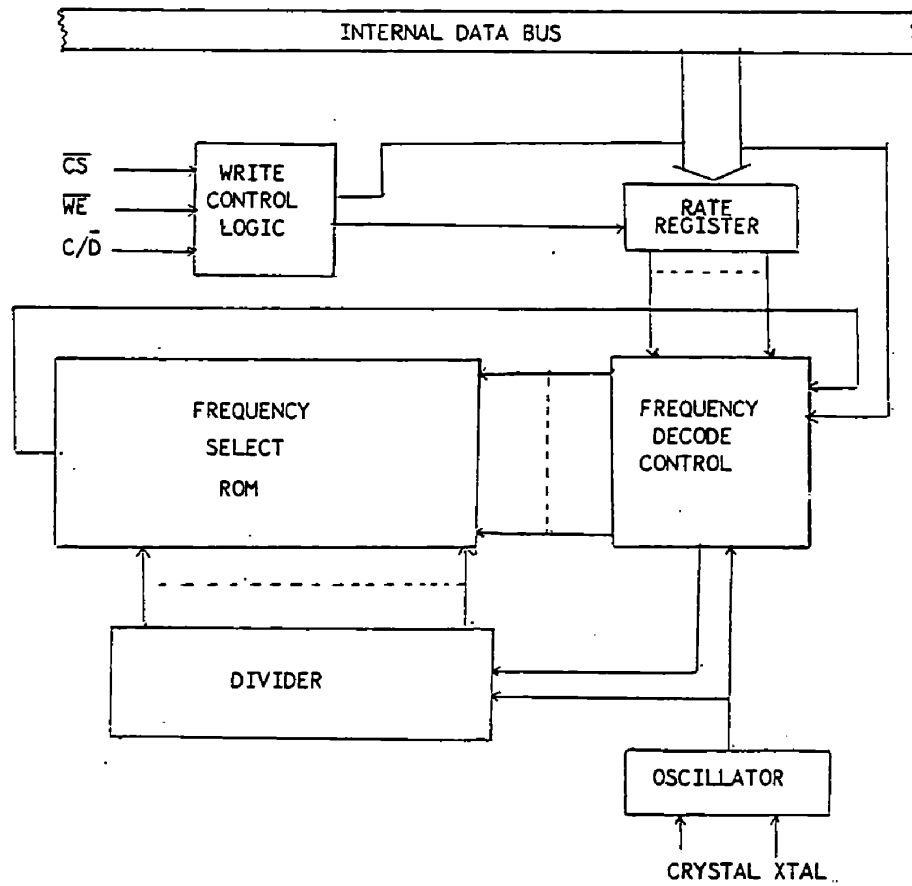
Fig. 4.7 – Baud Rate Generator Block Diagram

(a)

(b)

(c)

Fig. 4.8

– 60 –
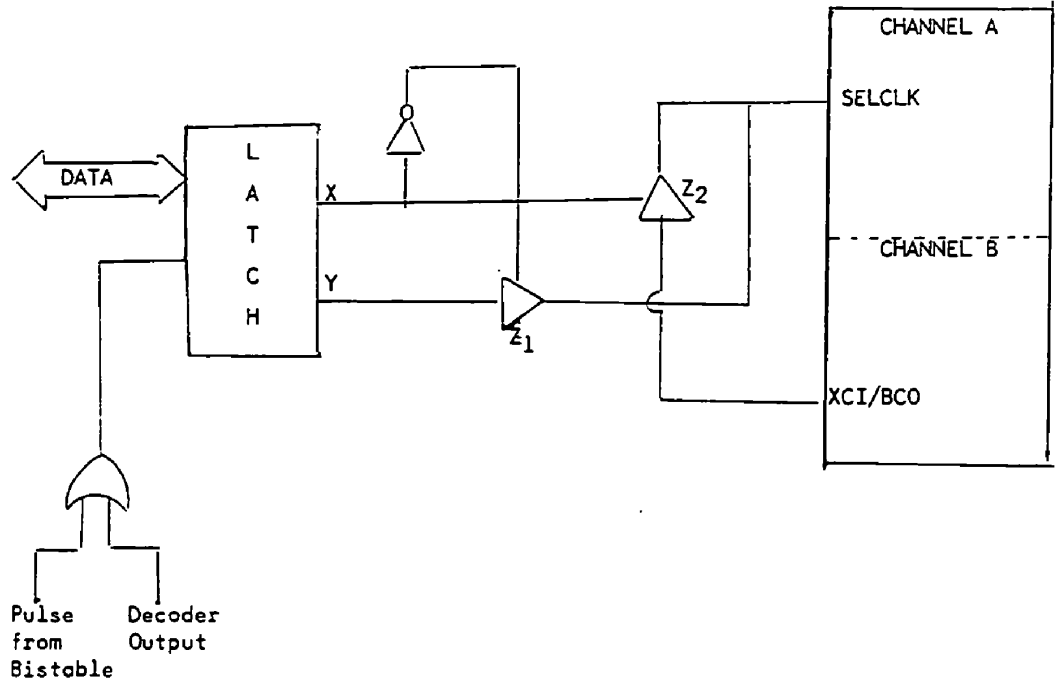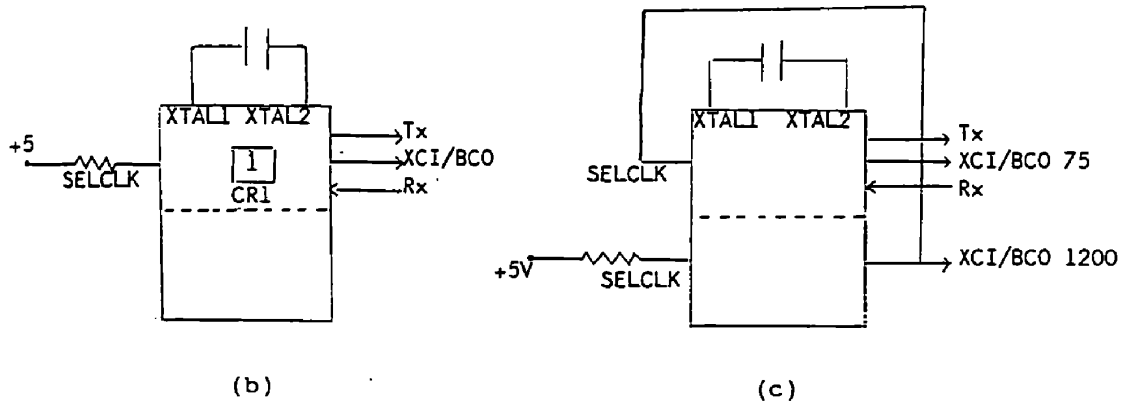
signals are discussed in the next section.

4.2.2.4   RS-232C Interface Circuits

These circuits are used to interface between Data Terminal
Equipment (eg Apple computer terminal) and Data Communications
Equipment (eg Modem) in conformity with the specification of EIA stand-
ard number RS-232C. The RS-232C standard defines all the control and
data signals (including signal levels) used to enable the modem and
the terminal to operate together and transfer data. For this purpose,
a quad line driver (MC 1488) and its companion circuit (MC 1489) quad
line receiver are used to provide a complete interface system between
TTL level signals from the system to RS-232C levels.

The interface control lines can be divided into two groups,
one concerned with the control sequences for connecting the modem to
the line and the other concerned with the sequences controlling the
application (and detection) of signal to the line.

The signals Data Terminal Ready (DTR) and Data Set Ready
(DSR) belong to the first group. The DTR signal from the terminal is
used to tell the modem to answer the telephone (ON) and to hang it up
(OFF). In this design, this signal is used to indicate whether the
terminal power is ON or OFF. The DSR signal is used to indicate to
the terminal that the modem is in a state in which it is capable of
transmitting data.

The group of RS-232C control signals that relate to the
application and detection of time signal are: Request To Send (RTS),
Clear To Send (CTS) and Data Carrier Detect (DCD). The logical
interaction of these signals is simple when they operate in a two-wire
half-duplex mode and complex when terminals operate a four-wire full-
duplex mode. In this design, the point-to-point communication between
two Apple microcomputers is half-duplex whereas the communication with
the Prestel computer is asymmetric full-duplex.

The Request To Send signal from the terminal is used to
indicate to the modem that the terminal wishes to transmit data. The
modem then responds by initiating its transmit mode. When the modem
has reached a steady state carrier condition, and is ready to transmit,
it sends a Clear To Send signal to the terminal. However, CTS does
not imply a positive verification that communication with the other end
has been established. Therefore the Data Carrier Detect signal is
used to indicate that the receiving modem has detected a line signal

- 61 -

(carrier) from the other end.

In point-to-point communication, initially when both ends are waiting for transmission to begin, the RTS signals of the serial input-output controller are at logic '1' and the modem will be holding the CTS inputs to the serial I/O device high. The terminal which wishes to transmit activates the RTS by making it low and does not start transmitting until CTS is turned ON. The RTS/CTS time delay is determined by specific hardware settings in the modem and is vitally important in determining the overall efficiency of data link use. Actually, the designed encryption system has been tested with two different modems - Modular Technology and the standard BT[†] ones. It is found that the former produced large amount of noise spikes when the modem switched from one mode to the other. This necessitated some complex delay routines to be introduced to reject the noise spikes. A typical interface timing diagram is shown in Figure 4.9. When connected to asymmetric full-duplex Prestel system, the RTS is kept ON continuously and the CTS ON is used to indicate that the transmission can start. This is an easy approach to ensure that the modem will operate properly in the full-duplex mode.

4.2.2.5   Timing Circuits

This block in Figure 4.2 consists of logic circuitry to provide the correct timing signals such as CHIP SELECT, READ, WRITE etc for the operation of the interface unit. The inputs to this block include amongst others, the clock signals from the Apple microcomputer system for proper synchronization.

The 6502 microprocessor has two clock signals Φ_o and Φ_1 of 1.023 MHz which are complementary to each other. In addition, a general purpose timing signal, twice the frequency of the system clocks but asymmetrical and an intermediate timing signal of 7.159 MHz are also available. The microprocessor uses its address and data buses only during the timing period when Φ_o is active. When Φ_o is low, the microprocessor is carrying out internal operations and does not need the address buses. These timing signals are shown in Figure 4.10.

Let us now very briefly consider the main timing signals required for the data security device and the serial input-output controller.

The timing diagram of a typical READ cycle of the encryption device is shown in Figure 4.11. The Data Output Request (DOR) bit in the Status Register of the device is used to indicate whether the

---

† British Telecom

- 62 -
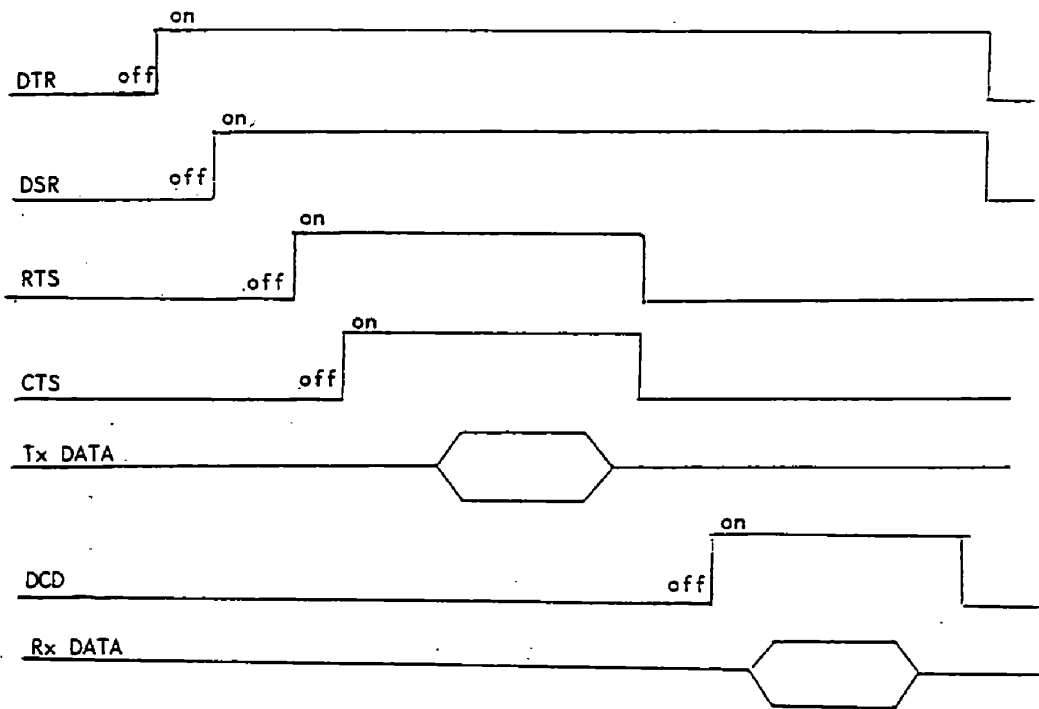
Fig. 4.9 - Interface Control Timing.

7M

ϕ_0

ϕ_1

Q3

ADDRESS

DATA (READ)
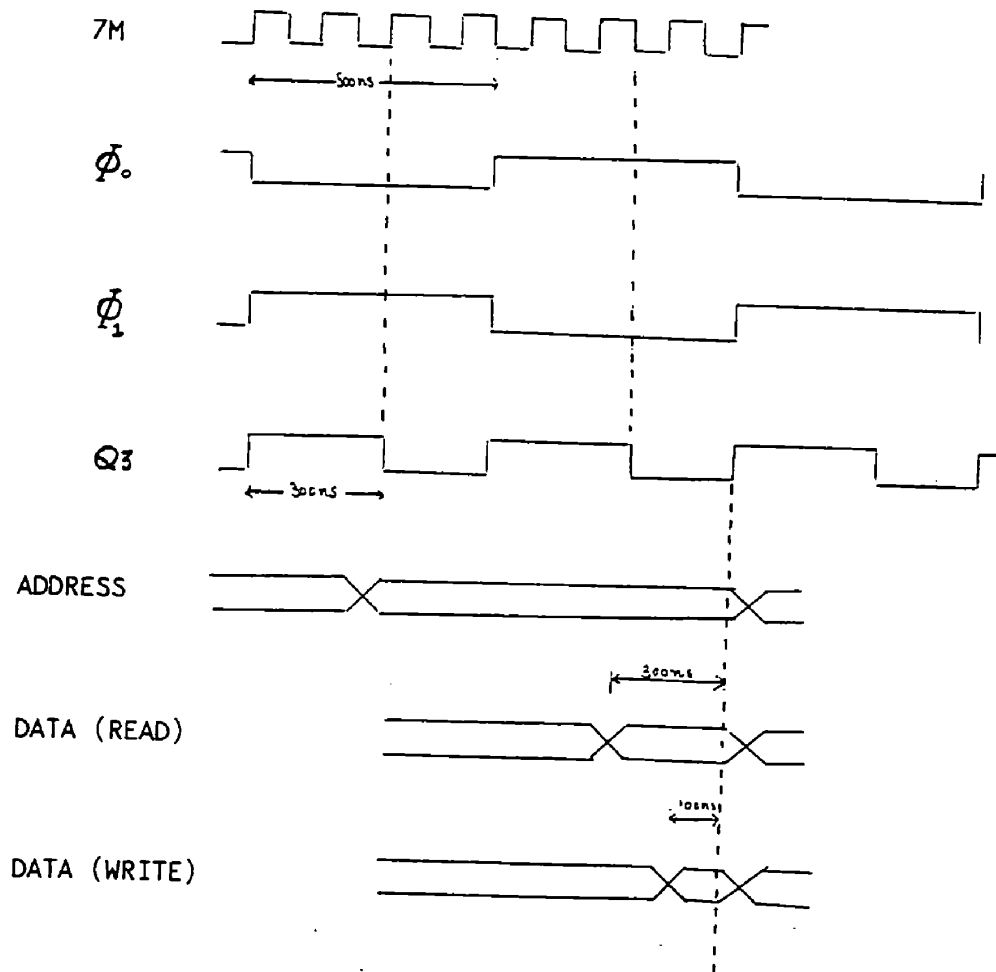
DATA (WRITE)

Fig. 4.10 – System Clock Signals and Timing Relationships

— 64 —
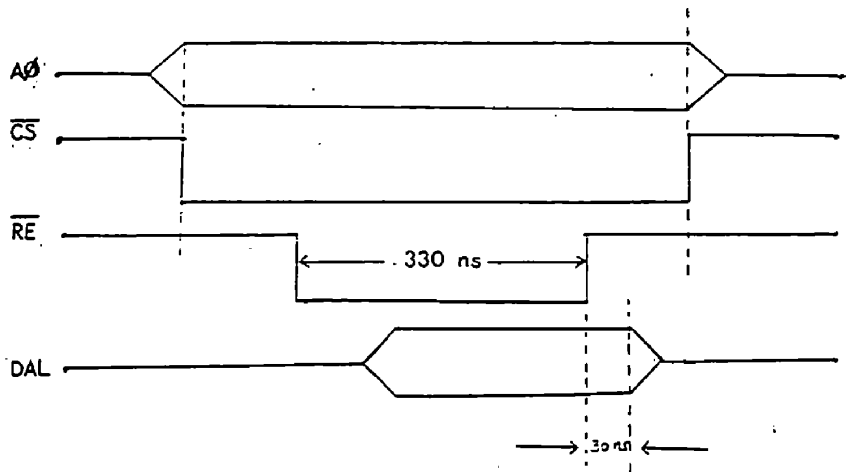
Fig.4.11 - READ Timing of Data Security Device.
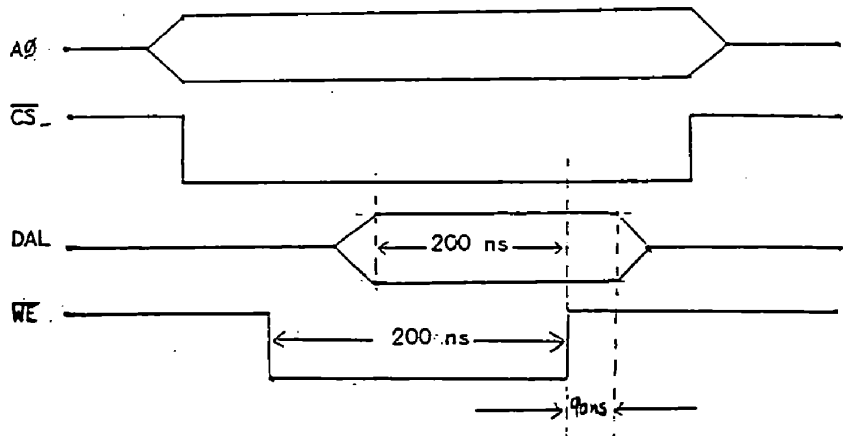
Fig. 4.12- WRITE Timing of Data Security Device.

- 65 -

DATA WORD present on the data bus is valid or not. When it is valid, the device is selected by making the CHIP SELECT ($\overline{CS}$) active low. This signal remains active low for at least 410 nano-seconds (ns). Then the READ signal is made active low to read the data from the bus. From the timing diagram, it is seen that the READ signal needs to be active low for 330 ns and the data must remain stable at least 30 ns after the rising edge of the READ pulse. The READ timing is not critical and it only needs to be greater than 330 ns. The READ signal is therefore derived using a standard decoder which produces an active low pulse of about 450 ns wide. This signal is also suitable for CHIP SELECT signals of both the security device and the serial I/O controller.

The WRITE timing diagram of the encryption device is shown in Figure 4.12. The device is selected by enabling the CHIP SELECT active low and then the DATA WORD or KEY WORD to be written is transferred to the data bus and a WRITE signal is produced. From the timing diagram, the data is to remain stable on the bus at least 200 ns before the rising edge of the WRITE pulse and 90 ns after the same edge. As the CHIP SELECT signal obtained from the decoder is 450 ns wide, the required pulse can be obtained by chopping off up to 200 ns from the rising edge of the CHIP SELECT signal. Actually, only a 90 ns pulse is subtracted as a 90 ns pulse is needed anyway in deriving the WRITE signal for the serial input-output controller. This pulse is then ored with the CHIP SELECT to give the WRITE signal of the security device. In Figure 4.13, this signal is denoted by Q2 + CS. The pulse is produced using a simple bistable circuit as shown in Figure 4.13 together with the timing diagram. The clock signals 7 MHz and Q3 are obtained from the Apple system. Note that the CHIP SELECT signal from the decoder has been used to control the rising edge of the WRITE signal by using it as the CLEAR input signal to the second bistable. The signal Q3 could not be used because it changes to the high state slightly before the CHIP SELECT thus producing a spike.

Now let us consider the READ, WRITE and CHIP SELECT signals required for the serial input-output controller.

The READ timing diagram of the serial input-output controller is shown in Figure 4.14. This signal is derived using a simple logic circuit which combines the pulse produced by the bistable circuit with another output line from the decoder circuit to produce the READ

Fig. 4.13

- 67 -
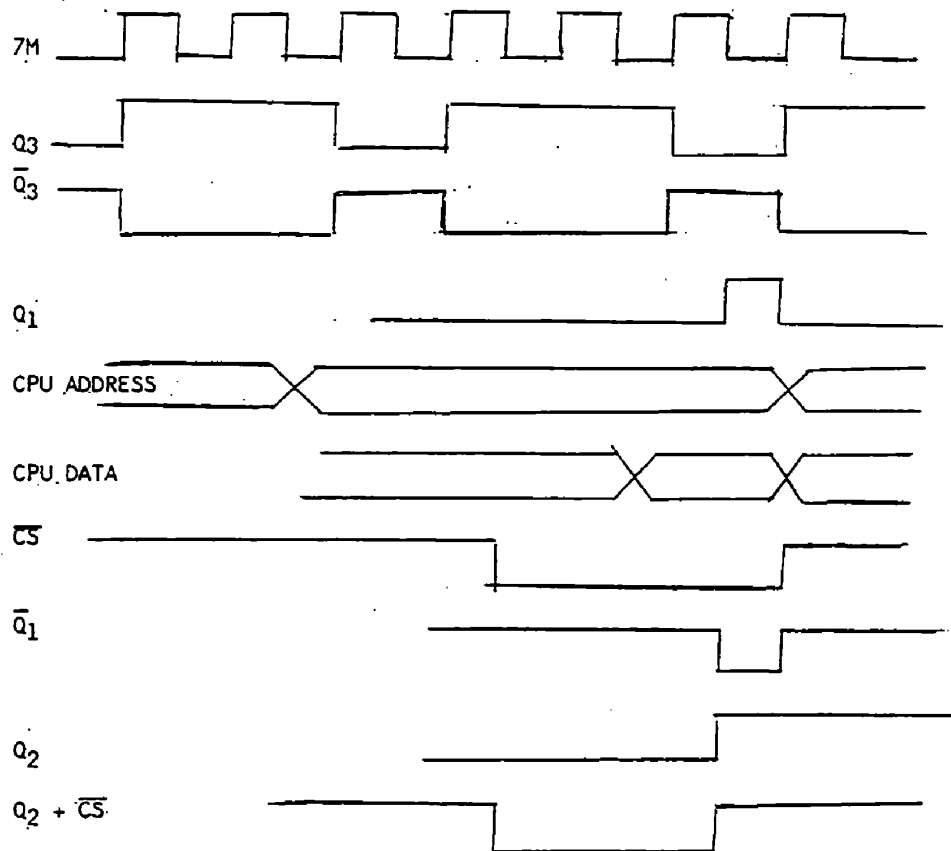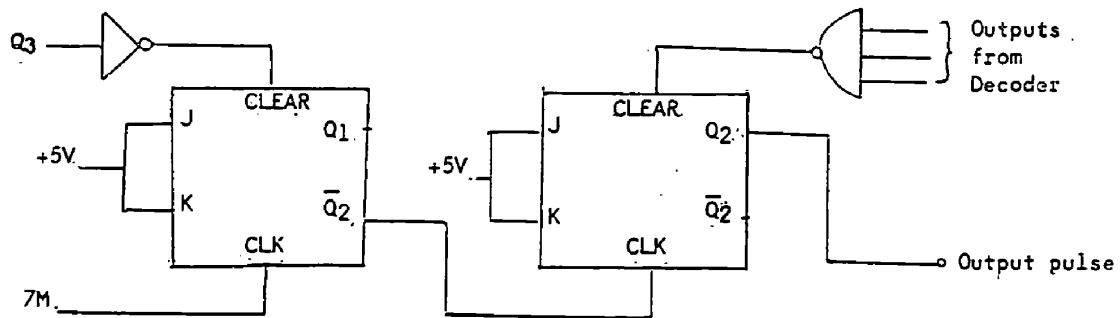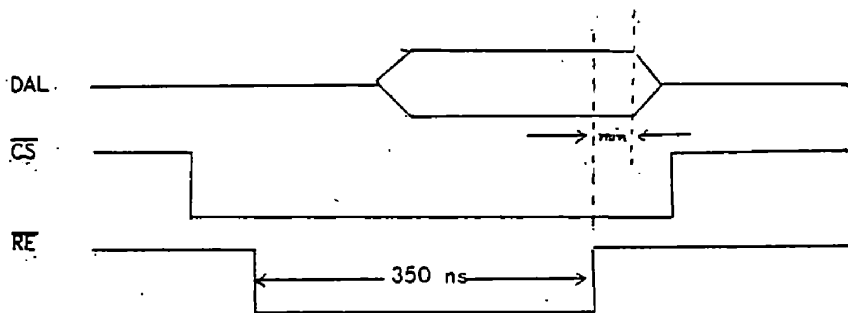
Fig. 4.14- READ Timing of I/O Controller.
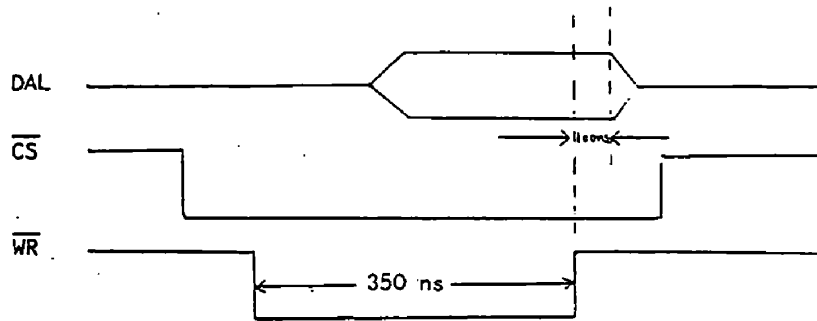
Fig.4.15 -WRITE Timing of I/O Controller.

- 68 -

signal. This produces a READ pulse of about 360 ns wide, with a
rising edge some 90 ns from the valid data edge. These agree well
with the required READ pulse in the timing diagram.

      The WRITE timing diagram of the serial input-output
controller is shown in Figure 4.15. In comparing this WRITE signal
with that of the one required by the security device, it is seen
that the controller requires the data to be stable on the bus for a
longer time of at least 350 ns before the rising edge. Also the
data is required to remain stable for a minimum time of 100 ns after
the edge, which is 10 ns greater than that required for the encryption
device. This signal is produced again by oring the 90 ns pulse with
an output line from the decoder. Although this produces a WRITE
pulse of width 360 ns with 90 ns of stable data after the rising
edge, it is found that these timings do satisfy the requirements
and does not cause any problem.

      The two CHIP SELECT signals required for channel A and
baud rate generator of the I/O controller are directly derived from
the output of the decoder.

4.2.2.6 Decoding_Circuits

      This block in Figure 4.2 provides signals which together
with the output signals from the timing block are used in selecting
different devices when appropriate addresses are present on the
address bus. In a system where there are many devices to run, it is
necessary to decode the address bus down into individual addresses.
This allows only one device and in particular only one mode of
operation of the device to be selected. For instance, say when a
control word has to be written into the serial input-output controller,
three operations are required to be carried out namely

(i)   only the input-output device must be selected,

(ii)   only WRITE signal must be activated and

(iii)   a distinction has to be made between a control word and
             a data word.

      The inputs to the decoder include the Device Select line
from the Apple microcomputer system which indicates which one of the
peripheral connectors is active. Whenever the interface is selected
this line will be active low. The Table 4.1 in Figure 4.16 shows
the decoded address and the corresponding operations carried out by

- 69 -

the system.

Address	Operation Performed
C0A0	WRITE Data Word to data security device.
C0A1	WRITE Control Word to data security device.
C0A2	READ Data Word from data security device.
C0A3	READ Status of data security device.
C0A4	ACTIVATE latch.
C0A6	ACTIVATE Tristate to test modem control signals.
C0A8	RESET I/O Controller
C0AA	RECEIVE (READ) Data from I/O Controller.
C0AB	READ Status of I/O Controller.
C0AC	TRANSMIT (WRITE) Data to I/O Controller.
C0AD	WRITE Control Word to I/O Controller.
C0AE	WRITE to Baud Rate Register of I/O Controller.

Figure 4.16 - Table 4.1: Decoding Arrangement

4.2.2.7 Memory_

The operation of the encryption unit is completely controlled
by software. Hence the proposed interface unit must have some non-
volatile memory to hold the program. For this reason, a 2K-byte
PROM is incorporated on the interface. The reasons for the choice are
two fold. Firstly, a 2K-byte is the maximum amount of memory that
can be associated with a peripheral slot in the Apple microcomputer
system. Secondly, it is estimated that a 2K-byte memory should be
sufficient for each of the required system tasks.

4.2.3 Modem

To transmit digital signals over Public Switched Telephone
network, which passes frequencies in the range 300-3000 Hz, it is
necessary for a data transmitter to modulate the digital stream by
superimposing the '1's and '0's onto a carrier signal. The data
receiver at the other end demodulates this signal. In addition to
its basic function of translating between the binary digital signals
of the data terminal equipment and the modulated voice frequency
signals of the communication channel, as seen in section 4.2.2.4,
the modem also performs a number of control functions which co-
ordinate the flow of data between terminal equipments.

- 70 -

The digital information can be encoded by systematically changing either the amplitude, the frequency, the phase or some combination of these characteristics of the carrier signal. The type of modulation used depends on the specific application. Over the public telephone network, the maximum data rate is limited to 1200 bits per second. For speeds up to 1800 bits per second, Frequency Shift Keying (FSK) is the common choice and hence this is used. With FSK, a pair of tones f_1 and f_2 are alternatively sent over the line for the binary '1's and '0's of the asynchronous data stream. The receiving section essentially consists of two filters that sense the frequency f_1 or f_2. Whenever f_1 is sensed, a binary '1' is produced and whenever f_2 is sensed a binary '0' is output to the data terminal equipment.

Modular Technology (12 CV) mini modems and standard BT modems employing FSK scheme have been used with this system.

- 71 -

CHAPTER 5

POINT TO POINT COMMUNICATION SYSTEM : SYSTEM SOFTWARE (1)

5.1   General

        The software has been written for the encryption interface
unit to perform the following tasks

(a)   Point-to-point communication between Apple microcomputers.

(b)   Storage/retrieval of encrypted information with floppy
        disk system.

(c)   Storage/retrieval of encrypted information with Prestel
        Viewdata system.

All the programs   are written in 6502 Assembly language to minimize
the execution time and memory space required. Before considering
these programs, two basic I/O techniques for handling transfer of
asynchronous data are briefly examined.

5.1.1 Polling Technique
        The polling method is one of the simplest ways to handle
asynchronous events. In this type of I/O, all operations are controlled
by the CPU program. The processor interrogates flags associated with
each possible event to determine whether any service is required.
That is, it polls the peripheral device periodically to determine its
readiness and hence the name polling technique. The CPU resources
are tied up during the time of transfer and the time of polling and
hence cannot be used for other tasks. This technique is mainly used
with low speed devices.

5.1.2 Interrupt Driven Technique
        In non-polling systems, the asynchronous event generates
an interrupt request which is passed on to the processor. The
processor in turn suspends the execution of the current process and
starts execution of the interrupt service routine which say performs
the data transfer. When the interrupt service routine is completed,
the processor resumes execution of the suspended process. The response

- 72 -

time is faster with such a system because no time is spent on interrogating the other non-active interrupts which in turn increases the system throughput.

The decision to adopt one technique or other depends on the nature of the application. In this project, the polling technique has been chosen for two reasons. Firstly in this encryption system, there are only two low speed devices which result in enough spare time for the processor to examine the status flags in a repeated fashion. Secondly the Apple 6502 system is not particularly suitable for an interrupt driven technique.

In this chapter, only the point-to-point communication program will be discussed. The other two tasks will be considered in Chapters 7 and 8 respectively.

5.1.3 Point-to-Point_Communication_

The encryption interface allows data transfer between Apple terminals (via a public switched telephone network) in either plain or encrypted or a mixture of plain and encrypted formats. Five different modes of the Data Encryption Standard have been investi-gated [3, 26]. They are:

1.   Block Encryption (ECB)
2.   Chained Block Cipher (CBC)
3.   Stream Cipher Feedback (CFB)
4.   Chained Block Cipher with Plaintext Feedback (CBCP)
5.   Stream Cipher with Ciphertext and Vector Feedback   (CFBV)

Each of these modes has been implemented using the encryption inter-face. They are described briefly in succeeding sections.

5.2   Block Encryption Mode

5.2.1   Principle

This is the most basic mode of operation of the Data
Encryption Standard which transforms 64-bits of input to 64-bits of
output as mentioned in Section 3.2.   The analogy to Electronic Code
Book (ECB) arises because the same plaintext block always produces
the same ciphertext for a given cryptographic key.   Thus it should be
theoretically possible to construct a codebook of plaintext blocks
and corresponding ciphertext blocks for any given key.

The ECB encryption-decryption scheme is shown in Figure 5.1.
In encryption, the plaintext block (D1, D2, ..., D64) is used directly
as the DES input block (I1, I2, ..., I64).   The input block is
processed through the DES device in the encrypt state which has been
preloaded with the appropriate cryptographic key.   The resultant
output block (O1, O2, ..., O64) is used directly as the ciphertext
(C1, C2, ..., C64).   The decryption process is same as the encryption
process except that the decrypt state of the DES device is used
rather than the encrypt state.   That is, the key schedule selection
is reversed.

Mathematically, the operations of encipherment and decipherment can be described as follows:

Let the cryptographic function f define the relationship between the
plaintext X and the ciphertext Y.   Then,

$$Y = f_k (X)$$
and
$$X = f_k^{-1} (Y)$$

where the subscript k designates the particular key (and hence the
particular function f_k) which is selected out of the set of all
possible keys.

5.2.2   Implementation

Only a brief summary of the program is given here.   The
structure of the program can be divided mainly into five parts,
namely, the initialisation routine, key input routine, data input
routine, transmission routine, which includes encryption, and receive

- 74 -

```
┌─────────────────────────┐          ┌─────────────────────────┐
│       PLAINTEXT         │          │       CIPHERTEXT        │
│   (D1, D2, .......D64)  │          │   (C1, C2, .......C64)  │
└─────────────────────────┘          └─────────────────────────┘
            ⇓                                    ⇓
┌─────────────────────────┐          ┌─────────────────────────┐
│    INPUT   BLOCK        │          │    INPUT   BLOCK        │
│   (I1, I2, ........I64) │          │   (I1, I2, .......I64)  │
│                         │          │                         │
│                         │          │                         │
│     DES ENCRYPTION      │          │     DES DECRYPTION      │
│                         │          │                         │
│                         │          │                         │
│    OUTPUT   BLOCK       │          │    OUTPUT   BLOCK       │
│   (O1, O2, ........O64) │          │   (O1, O2, .........O64)│
└─────────────────────────┘          └─────────────────────────┘
            ⇓                                    ⇓
┌─────────────────────────┐          ┌─────────────────────────┐
│       CIPHERTEXT        │          │       PLAINTEXT         │
│   (C1, C2, ........C64) │          │   (D1, D2, .........D64)│
└─────────────────────────┘          └─────────────────────────┘
```
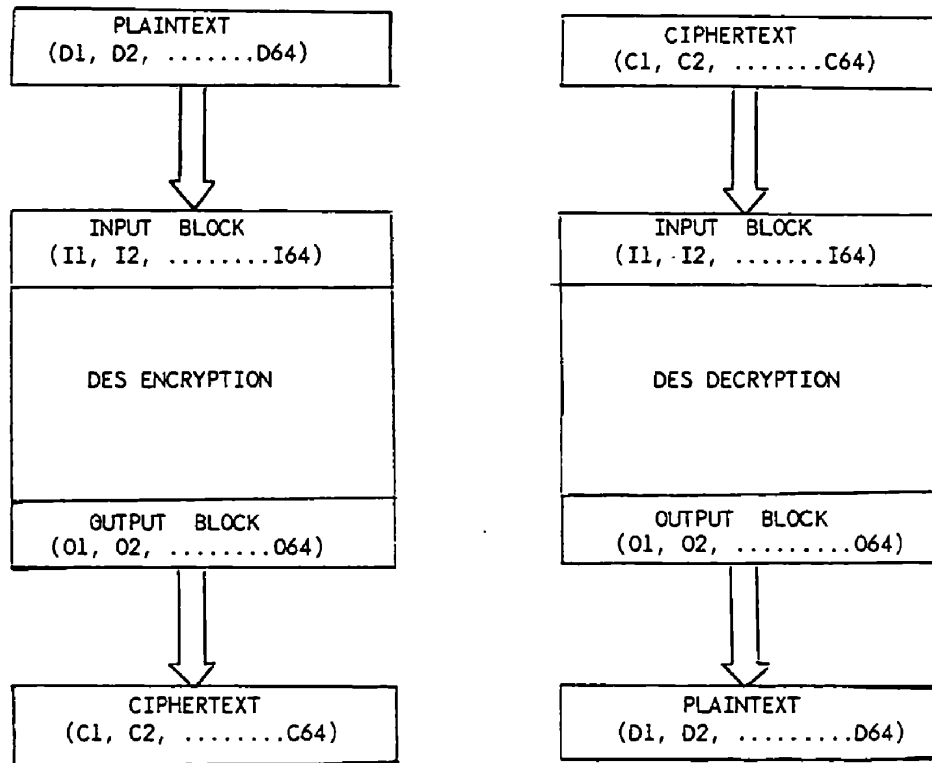
Fig. 5.1 - Block Encryption Codebook Mode (ECB)

routine, which includes decryption and display routines. The basic
flowchart of the program and the complete listing can be found in
Appendix 2.

5.2.2.1 Initialisation Routine

The initialisation routine itself can be divided into two
subsections. The first subsection contains instructions which are
executed without any user interaction whereas in the second subsection
the user chooses the parameters for setting a particular mode of
operation of the interface.

The first subsection consists of instructions to initialise
the CPU, to set the ports of the serial I/O controller, to ensure the
correct state of control signals from the modem, to set the state of
the latch in the interface and to initialize some memory locations
which would be later used as counters and flags in the program. The
Data Set Ready (DSR) signal from the modem is tested to check whether
the modem is ON and is not in the test mode. If it is not set, an
error message 'NO LINK ERROR' is displayed. The format of the
character to be transferred is defined to consist of 1 stop bit, 8
data bits and an even parity bit. A baud rate factor of 16 is used.
The external clock mode is selected in the I/O controller. As
explained in Section 4.2.2.3, this mode has been selected in point-
to-point communication to drive the transmit and receive clocks with
the same source. The error flags have been reset to allow error
detection in subsequent transfer of data. These error flags include
parity error, overflow error and framing error. The serial I/O is
set to a null mode, that is, neither a transmission mode nor a
receive mode.

In the second subsection, to begin with the user selects the
desired data transmission rate. A choice of baud rates ranging from
50 to 1200 bits per second are available. Then the user is asked to
select one of the three formats in which the interface may operate
namely plain or encrypted or a mixture of plain and encrypted. If
plain format is selected, no further initialization is required. If
either of the other two formats is chosen, then the data security
device is activated to encryption state.

5.2.2.2 Key Input Routine

If one of the two encryption formats is selected, then the

- 76 -

secret DES key is entered from the keyboard by the user. It is
assumed that both the parties concerned have the preknowledge of the
key which is a prerequisite for proper communication of encrypted data.
In this program, 8 characters are used to form the 64-bit key required
by the DES algorithm. Any of the 88 alphanumerical characters of the
keyboard can be used to form the 8 character key. As mentioned in
Section 3.6.1, the total number of distinct keys possible is then
equal to $(88)^8 \doteq 3.6 \times 10^{15}$ which is less than the maximum
$2^{56} \doteq 7.2 \times 10^{16}$. The program can be very easily changed to accept
sixteen 4-bit characters as the key thus giving the total possible key
space. Here the 8 character key is chosen to enable the user to
remember the key without actually recording it somewhere. It is
essential that the key should be chosen randomly so that it may not be
easily guessed by the opponent. Before loading every key byte into
the KEY REGISTER of the device, the key parity is tested. If a parity
error is detected, then an appropriate error message is displayed on
the screen. The interface unit displays the entered key on the screen
to enable the user to verify the correctness of the key. However the
display is immediately erased to avoid detection by others during
subsequent communication. The majority of users would probably use
some easy to remember phrase or number combination for developing the
key. In such cases, the long phrases can be converted to a form suit-
able for DES using a good hashing function. It should be sufficiently
complicated to produce essentially unbiassed and statistically
independent bits in the DES key. The program can also be modified to
provide for multiple DES key encryption with different keys to
achieve higher levels of security.

5.2.2.3 Data_Input_Routine_

        In point-to-point communication mode, the data to be
enciphered is assumed to be input from the keyboard of the terminal.
File transfer is considered in Chapter 7 where file security is
discussed. This routine determines whether the data to be processed
is coming from the telephone line or from the keyboard of the terminal.
This operation is carried out using the polling technique. The
processor is allowed to poll the Data Carrier Detect (DCD) flag from
the modem and the Keyboard Data input Flag (KDF). The KDF is
provided by the Apple system and if set indicates that a key has been
pressed. This implies that the data from the keyboard needs to be

transmitted over the line in plain or encrypted or mixture format depending on the selection made earlier and hence the control is transferred to Transmission routine. On the other hand, if the DCD flag is set indicating the presence of data on the line, Receive routine is invoked to decrypt and display the received data.

5.2.2.4 Transmission_Routine_

At a given time, up to a maximum of 256 bytes of message can be input to the interface from the keyboard. A Return character is used to mark the end of the message. At any time, the key CNTRL-X can be used to cancel the entire message. All editing facilities of the Apple system such as cursor movements can be used. The program operates on the whole message (up to 256 characters) at a time. After a whole message is processed and transmitted, the program returns to fetch the next message from the keyboard. An important point to note is that with block encryption mode (ECB), the message needs to be divided into blocks of 64 bits (8 bytes) before encryption can be done.

Initially, the I/O controller is set to transmit mode. Each 64 bit block of plaintext is then input to the data security device in successive 8 bytes after testing for the correct status of the Data Input Request (DIR) flag of the device. Then the ciphertext block is sent from the device byte by byte after testing the control signals, Clear To Send (CTS) and Transmit Ready (TXRDY). Appropriate error messages are displayed if the flags are not set properly and the system is resynchronized. Special routines are developed to deal with control characters CNTRL-G (Bell), CNTRL-J (line feed) and repeated Return characters in the plaintext block. When the encrypted version of these characters are transmitted, a certain amount of delay (approximately 80 milliseconds) is included to allow time for the generation of these characters at the receiving end after decryption.

However the message need not necessarily contain an exact number of 64 bit blocks. Hence it is likely that the message will end before the last block of 64 bits is complete. Therefore some padding is required at the end of the message to fill the last block to exactly 64 bits long. This has been done by padding the last block with some random characters after the Return character . These random characters are generated as part of the key input routine. The padding results in the cryptogram expansion of up to a maximum of 7 bytes compared to the orignal plaintext message if the last block is

− 78 −

not of full size.

5.2.2.5 Receive Routine

The Receive routine is much simpler than its counterpart Transmit routine. It is known that the incoming data is encrypted by dividing it up into blocks of 64-bits at the transmitting end. Hence this routine should simply fetch the 64-bits of ciphertext at a time from the link after testing the Receive Ready flag (RXRDY) of the I/O controller. The ciphertext is then input to the data security device which is now set to decryption mode. Then the deciphered block of data is read from the device and is displayed on the screen using Display routine. The Display routine is same as the one used by the Apple system. It displays uppercase and lowercase alphanumeric characters in either normal or inverse or flashing modes. The control characters in the decrypted block are not displayed. The block of decrypted characters is tested for the presence of the Return character. If it is not present, then it is known that another block of encrypted data must be following the current block. Hence the program loops back to test the RXRDY flag to obtain more data from the link. On the other hand if a Return character is detected within the block of deciphered data, it indicates that the current block is the last block of the message. The random characters after the Return character are treated as dummy characters and hence they are ignored. The decrypted information will be the same as the original message provided the same key has been used at both ends of the link.

The routine also performs three types of error tests on the received characters. They are parity error, framing error and overrun error. In each case, appropriate error messages are displayed. A framing error is detected when the receiver finds a logic '0' occurring at the time when stop bit, logic '1', should occur. A overrun error is detected when a new character is loaded into the Receive Holding Register of the serial I/O controller before the processor has had time to read the previous one. In all these three cases the program loops back to the start of the data input routine thus abandoning the current ciphertext block and automatically synchronises.

5.2.2.6 Plain Data Communication

The routines which allow plain data transmission and

– 79 –

reception are very much similar to the ones described earlier except
in this case there is no need to perform encryption and decryption.
Hence the message is processed byte by byte rather than in blocks.

This facility of communicating in plain format is provided
to allow the transfer of non-sensitive information over the link. For
instance, one could envisage a situation where the users initially
communicate in plain format to set various parameters needed for
establishing a secure link. This immediately led to the idea of developing
a program which could handle a mixture of plain and encrypted data
and this is considered in the next section.

5.2.2.7 Mixture of Plain and Encrypted Data Communication

With this format, only parts of the message are in encrypted
form while the rest of the message is transmitted in plain form.
This mode is useful in many applications where it is not necessary to
encrypt the whole message.

The receiver must be able to identify which parts of the
received data are in encrypted form. The data transmitted is always
in plain form until the change to encryption mode is initiated. This
is done by typing a special character (CNTRL-A) on the keyboard that
has been designated for this purpose. Characters following this
character are encrypted by the sending terminal. The interface unit
is automatically returned to the plain format after 8 characters.
Alternatively, another special character (CNTRL-B) can be used to
return to the plain format. The receiving end then checks for a
CNTRL-A character and starts to decrypt when it is found. When a
CNTRL-B is received, it switches back to plain mode reception.
However it is necessary to ensure that the CNTRL-B character does not
occur within the enciphered data to ensure unambiguous decryption
at the receiving end. This can be achieved by using multiple
CNTRL-B characters to indicate the end of encrypted text. The greater
the number of such characters, the smaller the probability that they
occur in the enciphered data and hence the smaller the ambiguity
in decryption, but this increases the number of redundant characters
in the transmitted data. The key required for the encryption
algorithm is entered as before at the beginning of the communication.

Again with the block encryption mode, if the encrypted parts
of the message are not integral multiples of 64 bits, they require
padding and this results in cryptogram expansion. The difference

- 80 -

with the mixture format is that this expansion does not occur only at the end of the message (last block), as in the complete encryption format discussed earlier, but it may occur anywhere within the message.

When implementing this mixture format in the Apple system, additional problems are encountered compared to the complete encryption format. One such problem arises from the requirement that in the multi-user network the plain information transmitted may need to be received correctly by all users whereas the encrypted information must only be deciphered correctly by the user with the right key. There may be cases where the encryption algorithm transforms a non-control character to a control character and vice versa. As a control character is not displayed by the Apple system and the screen cursor does not move, this results in a line of text, with parts of it encrypted at the transmitting end, not producing a line of text at the receiving end with the wrong key. As a certain amount of delay is required for some special characters such as CNTRL-G (Bell), CNTRL-J (Line feed), this can cause overrun error at the receiving end with the wrong key. This results in errors in the subsequent reception of data even in plain format at the receiving end. This is further complicated in this block encryption mode due to the padding with random numbers to fill the block.

A modified version of the earlier complete encryption format program is developed for this mixture format which overcomes the problems mentioned above.

5.2.3 Results and Discussion

An example of plaintext containing some data structure and patterns has been chosen to study the various characteristics of different encryption modes. Such an example is provided by the assembly language program shown in Figure 5.2. This whole message is enciphered using ECB encryption with the key 3131313131313131 in hexadecimal form. Note that here a non-random key has been chosen to allow the ciphertext produced to be used in some statistical tests in Chapter 6. A ciphertext character produced can be any one of the 256 possible combinations (2^8). To display the ciphertext, it is therefore necessary to extend the standard ASCII character set from 128 to 256. This has been done using Hershey characters [27]. The complete character set is given in Appendix 3. Note that this extended

– 81 –

Fig. 5.2 - Plaintext Example

character set does not in itself introduce any secure cryptographic transformation. The ciphertext produced is shown in Figure 5.3. The program used to produce this graphical display is given in Appendix 4.

From Figure 5.3, it can be seen that a potential weakness of this block mode is that the same plaintext always produces the same ciphertext under a fixed key. This in turn implies that if the plaintext contains patterns, they will be reflected in the ciphertext as seen in Figure 5.3. Consider for instance, the asterisks in lines 1 and 5 and sequences of blanks at the beginning of each line in the assembly language program. Thus the compromise of the plaintext block underlying any ciphertext block results in the compromise of all repetitions of this same text for the remainder of the cryptographic period. Thus this block encryption mode is more susceptible to code book analysis compared to the other modes considered later. Further if the plaintext information is highly redundant then block encryption may not prevent cryptanalysis using block frequency analysis. Block frequency analysis determines the frequency of each ciphertext block from a large sample of intercepted ciphertext. By relating the observed frequencies of the ciphertext blocks to the expected frequencies of the plaintext blocks, the cryptanalyst may be able to draw certain inferences concerning the nature of the plaintext corresponding to a given ciphertext. Also if the data transmitted is highly redundant, the number of possible meaningful plaintext blocks may be small enough to construct a dictionary.

This block encryption mode is also susceptible to replay. As each block is independently enciphered with the same key, one block can be replayed for another. For instance, consider the transaction shown below

CREDIT	JONES	£5000	CREDIT	SMITH	£10
C1	C2	C3	C4	C5	C6

This can be changed to

CREDIT	JONES	£5000	CREDIT	SMITH	£5000
C1	C2	C3	C4	C5	C3

- 83 -

Fig. 5.3 – ECB Ciphertext Example

Fig. 5.4 - ECB Error Characteristic

by replaying the block containing the ciphertext for £5000. This
type of replay is not possible with stream cipher and chained block
cipher modes which are considered later. One simple solution to this
problem is to append checksums to the end of messages.

This block encryption mode is also vulnerable to insertion
and deletion of blocks because these changes to the message do not
affect surrounding blocks. Again use of error detecting codes before
encryption and checksums protects against this threat.

As mentioned in the implementation section, in this mode,
the information is encrypted in integral multiples of 64 bits. This
resulted in padding of last block with random characters. This causes
cipher extension and therefore may be unacceptable in some applications.
The padding effect can be seen in Figure 5.3 at the end producing a
longer ciphertext message than the original plaintext. If the
padding is done with blanks or zeroes, instead of random numbers,
then this may make them vulnerable to crypanalysis.

Since each bit of the ECB output block is a complex
function of every bit in the input block and the key, a single bit
change in either the key or the plaintext results in a ciphertext
block in which each bit is changed with approximately equal
probability. Conversely, a change in 1 bit of either the key or
ciphertext will produce changes in an average of fifty percent of
the bits of deciphered plaintext. Although this error propagation
within the block is extensive, it is strictly limited to the block
in which the error occurs and the decryption of other blocks is
unaffected. This can be seen from Figure 5.4 where a number of
errors have been introduced in the ciphertext prior to decryption.
Thus the ECB mode does not provide error extension between blocks.
If block boundaries are lost between sender and receiver, then ECB
cryptographic synchronization will also be lost until correct block
boundaries are re-established. This may happen for instance when a
bit slip occurs.

5.3   Cipher Block Chaining Mode

5.3.1 Principle
Cipher block chaining is again a block cipher in which the
plaintext is exclusive-ored with a block of pseudo-random data prior
to being processed through the DES device.

- 86 -

This scheme is shown in Figure 5.5. In order to commence the CBC encryption, the first DES input block is formed by exclusive-oring the first data block with a 64-bit initialization vector (IV). That is,

(I1, I2, ... I64) = (IV1⊕D1, IV2⊕D2, ..., IV64⊕D64).

This first CBC input block is processed through the DES device in the encrypt state, producing a 64-bit DES output block which defines the ciphertext. This first ciphertext block is then exclusive-ored with the second plaintext block to produce the second DES input block. This second input block is enciphered using the DES device to produce the second ciphertext block. This encryption process continues to chain successive ciphertext and plaintext blocks together until the last plaintext block of the message is encrypted. In CBC decryption, the first ciphertext block is processed through the DES device in the decrypt state. The first output block is then exclusive-ored with the CBC initialization vector producing the first plaintext block. The second ciphertext block is then entered into the DES device and the resultant output block is exclusive-ored with the first ciphertext block to produce the second plaintext block. The CBC decryption process continues to exclusive-or the ciphertext block at time t-1 (t>1) with the DES output block to obtain plaintext at time t until the end of the message.

Mathematically, the scheme can be expressed as follows: Let the cryptographic function f_k define the relationship between the DES input block and the DES output block under the chosen key k. Let function h define how the input to the DES is altered through the introduction of the initialization vector and the feedback or intermediate initialization vectors U(1), U(2),..., U(n-1) at time t = 1 to n-1. Note that the function h may be a many-to-one function since identical inputs to this function will be available during both encipherment and decipherment. Then

$$U(1) = Z = \text{Initialization Vector}$$
$$U(i) = h[U(i-1), \text{feedback quantity}], i > 1$$

In the CBC mode, U(i) is equal to the previous ciphertext block Y(i-1), the feedback quantity. That is, U(i) = Y(i-1). Hence the encipherment and decipherment operations can be expressed as

- 87 -
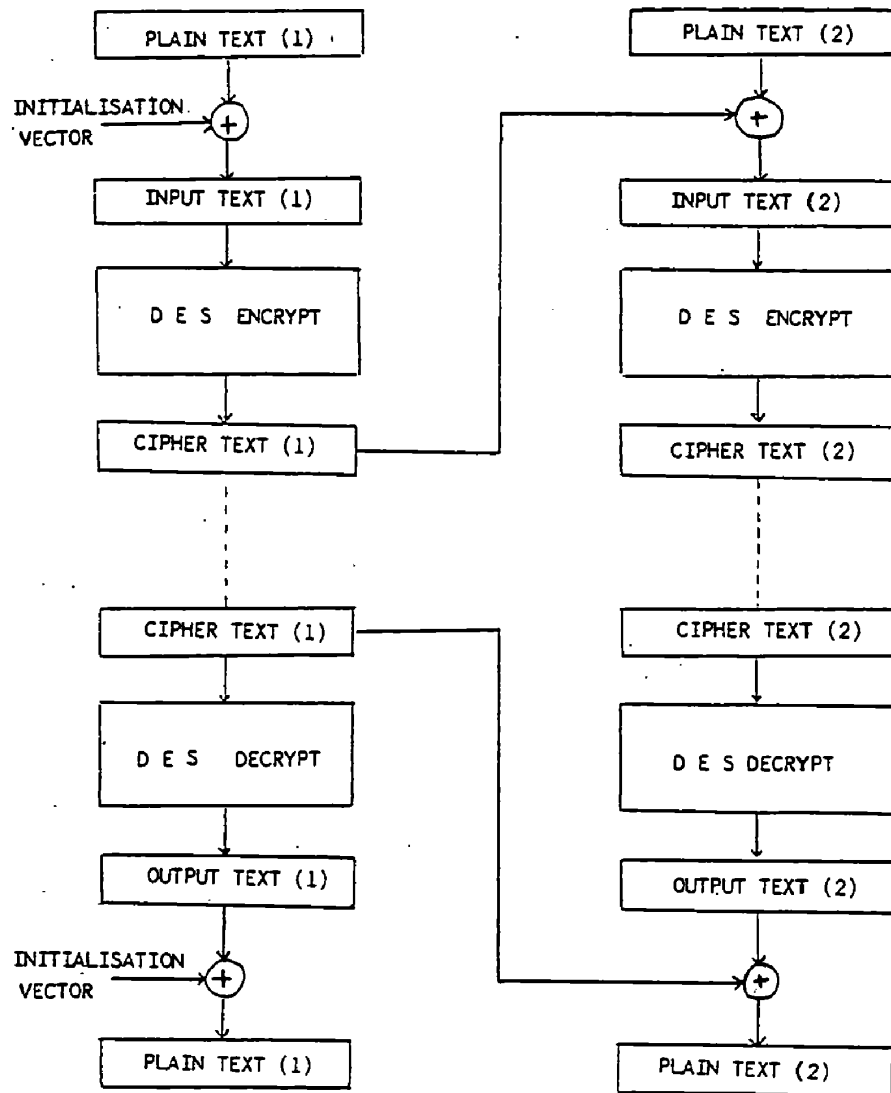
Fig. 5.5 - Cipher Block Chaining Mode. (CBC)

- 88 -

$$Y(i) = f_k [X(i) \oplus Y(i-1)] \quad i \geqslant 1$$

and

$$X(i) = f_k^{-1} [Y(i)] \oplus Y(i-1) \quad i \geqslant 1$$

where

$$X(o) \equiv Y(o) \equiv Z$$

From the recursive nature of the above equations, there exists functions H_1, H_2..., H_i such that

$$Y(i) = H_i [k, X(o), X(1), ... X(i)], i \geqslant 1 \qquad (5.1)$$

Similarly there exists functions G_1, G_2, ..., G_i such that

$$X(i) = G_i [k, Y(i-1), Y(i)], i \geqslant 1 \qquad (5.2)$$

From equations (5.1) and (5.2), it follows that patterns within the input are masked since the ciphertext block $Y(i)$ depends on plaintext blocks $X(1)$, $X(2)$, ..., $X(i)$. However since the received plaintext block $X(i)$ does not depend on all ciphertext blocks $Y(1)$, $Y(2)$, ..., $Y(i)$, the scheme does not represent a general block cipher (see Section 5.5).

5.3.2 Implementation

This program is a modified version of the ECB mode program described in Section 5.2.2. The flow chart of the program together with the listing can be found in Appendix 5. Here only the differences between this program and the ECB mode program are briefly discussed.

As in the ECB program, the user initially selects the data rate, the format of data transfer (plain, encrypted or mixture) and enters the DES secret key. Then the transmission end generates a 64 bit random block and sends it to the receiving end. The block is encrypted under the ECB mode at both the transmitting and receivng ends to form the initialization vector (IV). This vector is stored in a set of memory locations named TEMP 1 at the transmitting end and TEMP 2 at the receiving end. It is again necessary to divide the message into blocks of 64 bits. Hence padding of the last block

- 89 -

with random characters is required to make the blocksize equal to 64. The transmission routine fetches the plaintext block which is then exclusive-ored with the memory locations TEMP 1 before inputting it to the DES device. The ciphertext block produced by the device is transmitted to the receiver as in the ECB mode. The ciphertext is also stored back into the memory locations TEMP 1 for use in the next encryption cycle. The receive routine inputs the ciphertext to the DES device and also stores it in memory locations TEMP 3. The output from the DES device is exclusive-ored with the memory locations TEMP 2 to produce the plaintext. The memory locations TEMP 3 are then transferred to TEMP 2 to form the new IV for exclusive-or operation in the decryption of the next ciphertext block.

Finally, note that in the case of mixture of plain and encrypted data communication, only the encrypted blocks are chained together.

5.3.3 Results and Discussion

The plaintext example shown in Figure 5.2 is enciphered under CBC encryption with the same key as in the ECB mode. The initialization vector used is 0202020202020202. The ciphertext produced is shown in Figure 5.6.

From Figure 5.6, it is seen that the CBC mode does not produce the same ciphertext even when the plaintext is the same and hence the pattern exposure problems associated with the ECB mode have been eliminated. This is because the ciphertext produced for a plaintext block with a given key is dependent on the plaintext as well as the feedback (intermediate) vector used in the process which is different at different times. So the CBC mode reproduces the same ciphertext whenever the same plaintext is encrypted under a fixed key and initialization vector. Thus with the CBC mode, ciphertext repetition occurs at the message level whereas with the ECB mode, ciphertext repetition is found to occur at block level. Thus the code book analysis problem has been reduced.

CBC is therefore less susceptible to replay than the ECB mode of encryption. The type of replay mentioned in Section 5.2.3 is not possible with CBC mode as different parts of the message are enciphered with different feedback vectors. It is also less vulnerable to insertion and deletion of blocks as these changes to the message affect the surrounding blocks.

- 90 -

Fig. 5.6 – Ciphertext Example

CBC also protects a block cipher against the time-memory trade off attack [23] as follows.

Let Y(i) be the ciphertext corresponding to the chosen plaintext X(i). Since f_k^{-1} (Y(i)) = X(i) $\oplus$ Y(i-1), to determine the key k, a cryptanalyst would have to generate tables of starting and ending points using X(i) $\oplus$ Y(i-1) rather than X(i). But this would rule out the possibility of precomputing the tables or of using the same tables to break more than one cipher.

As noted in the implementation section, the problem of padding still exists with the CBC mode. In the complete encryption format, this is confined to the last block of the message. The receiver scans the decrypted block and discards the pseudo-random bits after the Return character code in the block. One way to eliminate padding is to switch to stream cipher feedback (CFB) mode (see Section 5.4) to encipher the short block at the end of the message .

In addition, the security of the CBC mode depends among other things upon the management of the CBC initialization vectors. It is important that the initialization vector (IV) is pseudo-randomly selected. Further the vector must be protected from disclosure. In the above implementation, this is carried out by using the ECB encrypted version of the random block generated as the IV. It is also advisable to change the vectors frequently to avoid the cryptanalyst using the ciphertext search attack.

Within the ciphertext, some errors are introduced and then deciphered with the same key and initialization vector to study the error extension characteristics of the CBC mode. From Figure 5.7 one or more bit errors within a single ciphertext block are found to affect the decryption of two blocks, namely, the block in which the error occurs and the succeeding block. If the errors occur in the ith ciphertext block, then each bit of the ith plaintext block has an error rate of about fifty percent. The (i+1)-th plaintext block has only those bits in error which correspond directly to the ciphertext bits in error. Further it is seen from Figure 5.7 that after two blocks, cryptographic synchronization is automatically established and the subsequent deciphered blocks are unaffected. Compared to the ECB mode, this has extended the error propagation to two blocks. This can also be seen from equations (5.1) and (5.2)

- 92 -

Fig. 5.7 - CBC Error Characteristic

where an error in the ciphertext block Y(i-1) can affect every bit
in the recovered plaintext block X(i-1) but it will affect only the
corresponding bit positions in the received plaintext block X(i).
Subsequent plaintext blocks X(i+1), X(i+2), ... are unaffected.

If bits are added or lost in a ciphertext block so that
block boundaries are lost between sender and receiver, then
synchronization is lost. However cryptographic synchronization will
automatically be re-established 64 bits after block boundaries have
been established.

This self synchronizing scheme may be useful when small
amounts of noise are present on the data communication links.

5.4   Stream Cipher Feedback

5.4.1   Principle

The cipher feedback mode (CFB) is an additive stream
cipher technique in which the DES is used to generate a pseudo-
random binary stream. This stream is exclusive-ored with the binary
plaintext to form the ciphertext which is fed back to form the next
DES input block. The pseudo-random binary stream is sometimes
referred to as the key stream and the DES the key generator.

This mode is schematically shown in Figure 5.8. One
through to sixty four bit CFB operation may be used unlike the ECB
mode where the message is required to be divided into blocks of a
given blocksize namely 64. A 64-bit initialization vector (IV) is
used as a starter input block to begin the CFB operation. (Note
that if the size of IV is chosen to be less than 64 bits, then it can
be padded with '0's to form 64 bits). This vector is processed
through the DES device in the encrypt state to produce a pseudo-
random output block. The message is divided into characters of
s-bit size where $0 < s < 65$. The DES algorithm is operated once for
each new s-bit character. Then s-bits of the DES pseudo-random
output block (O1, O2, ..., Os) are used in the exclusive-or operation
with the s-bits of the plaintext (D1, D2, ..., Ds) to form the
ciphertext (C1, C2, ..., Cs). That is, (C1, C2, ..., Cs) =
(D1 ⊕ O1, D2 ⊕ O2, ..., Ds ⊕ Os). This operation may be defined
when the length of the plaintext character to be encrypted is less than

— 94 —
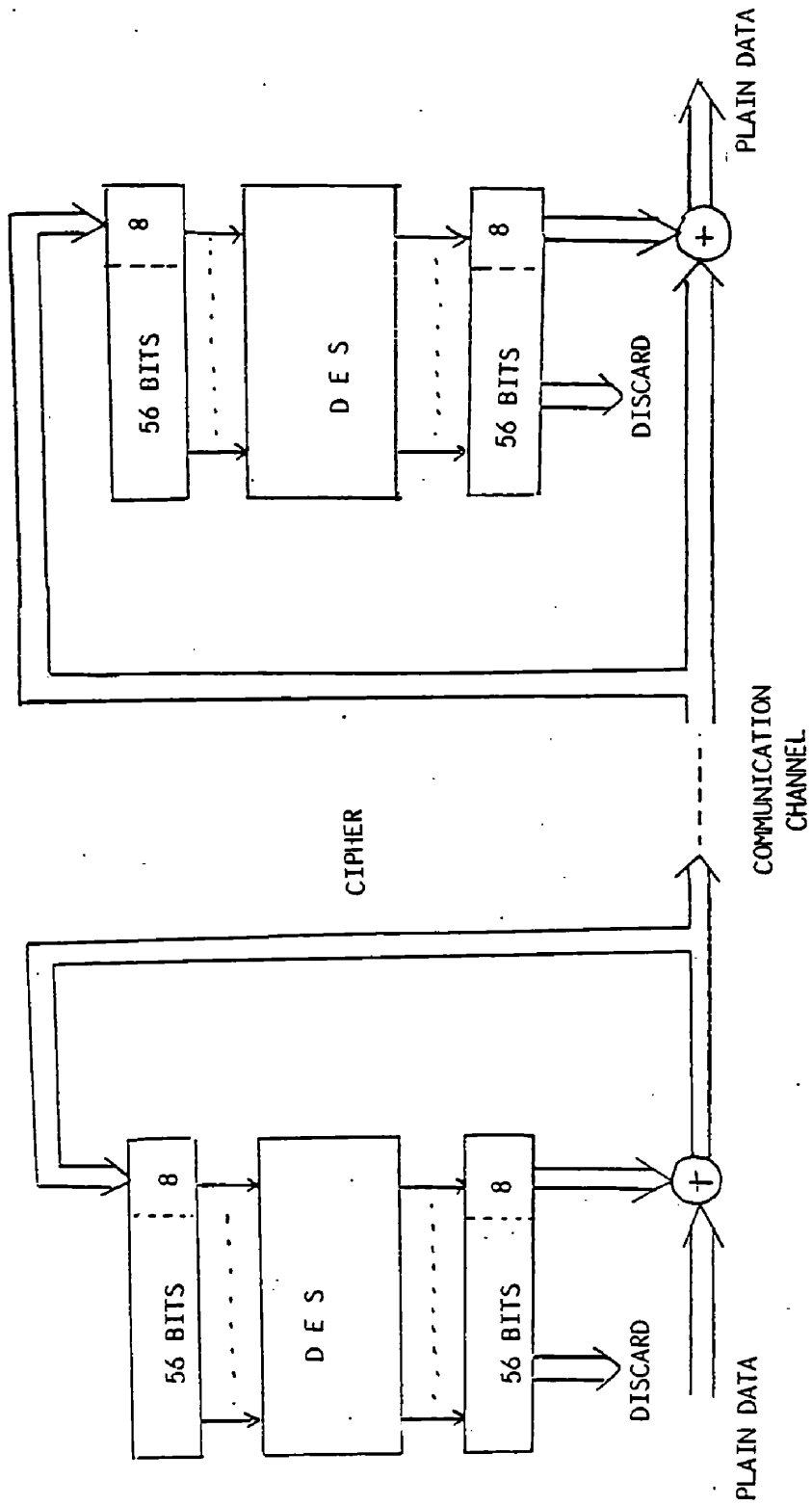
Fig. 5.8 – Cipher Feedback Mode. (CFB)

s-bits by concatenating zeroes to the left hand side or most
significant bits of the plaintext character. Similarly during
decryption, plaintext is produced by exclusive-oring a s-bit
ciphertext character with the s-bits of DES output block. That is,

$$(D1, D2, ..., Ds) = (C1 \oplus O1, C2 \oplus O2, ..., Cs \oplus Os)$$

In both cases, the same s-bits of the DES output block are used and
the unused bits are discarded. At both ends, the next input block
is created by discarding the most significant s-bits of the previous
input block, shifting the remaining bits s positions to the left and
then inserting the s-bit ciphertext character just produced in the
encryption operation or just used in the decrypt operation into the
least significant bit positions as shown in Figure 5.8. That is,
the input block (I1, I2, ..., I64) is given by,

$$(I1, I2, ..., I64) = (Is+1, Is+2, ..., I64, C1, C2, ..., Cs)$$

This input block is then processed through the DES device in the
encrypt state to produce the next output block. An important
difference compared with the two modes considered in Sections 5.2 and
5.3 is that even in decryption, the DES is used in its encryption
state. This is because in CFB mode, the DES algorithm has been used
as a pseudo-random number generator rather than as a cryptographic
transformation.

Mathematically, this mode can be expressed as follows:
Let $X(i)$ be the ith plaintext input data and $Y(i)$ be the ith
ciphertext output data. Let $U(i)$ be the intermediate initialization
vector at time i. In general, the length of the intermediate
initialization vector may not be equal to the length of the initial-
ization vector Z. Let the function h' define how $U(1)$ is obtained
from Z. That is, $U(1) = h'(Z)$.
Encipherment of the first s-bit plaintext is given by:

$$Y(1) = X(1) \oplus f_k [U(1)]$$

Let the function h define the dependency of the intermediate
initialization vector at time i on the previous initialization vector
$U(i-1)$ as well as the additional feedback quantity. That is,

– 96 –

$$U(i) = h \, [U(i-1), \text{ feedback quantity}]$$

where the feedback quantity = $Y(i-1)$.

In CFB mode with $s<64$,

$$U(i) = U(i-1) \| Y(i-1), \ i>1$$

where $\|$ denotes concatenation of $U(i-1)$ and $Y(i-1)$.

With $s=64$,

$$U(i) = Y(i-1), \ i>1$$

That is, effectively the ciphertext at time $i-1$ is fed back as input to the DES algorithm. Defining $Y(0) \equiv Z$, with $s=64$, it follows that

$$f_k [U(i)] = f_k [Y(i-1)], \ i \geq 1$$

Therefore equations of encipherment and decipherment can be expressed as

$$Y(i) = X(i) \oplus f_k [U(i)], \ i \geq 1 \qquad (5.3)$$

and

$$X(i) = Y(i) \oplus f_k [U(i)], \ i \geq 1 \qquad (5.4)$$

respectively where $U(1) = Z$.

5.4.2 Implementation

An 8-bit cipher feedback mode has been implemented on the encryption system. The flowchart of the program can be found in Appendix 6. The differences between this program and the ECB and CBC programs are now very briefly mentioned.

As in the case of the CBC program, a 64-bit initialization vector is generated at the transmitting end and sent to the receiver for proper synchronization. The major difference compared to ECB

and CBC programs is that in this mode, the message is encrypted byte by byte and not in blocks. This implies that there is no need for padding at the end of the message and consequently no cryptogram extension. One additional routine SHIFT is required which shifts the intermediate initialization vector (memory locations TEMP 1 and TEMP 2) to the left and appends the 8 bits of the ciphertext to the right side of the shifted input to produce the next DES input. Finally at both the transmitting and the receiving ends, the data security device is programmed to encryption state.

5.4.3. Results and Discussion

The plaintext example shown in Figure 5.2 is enciphered under the CFB mode using the same key and the initialization vector as in the CBC mode. The ciphertext produced is shown in Figure 5.9.

From Figure 5.9, it is seen that as in the case of the CBC mode, the CFB mode does not produce the same ciphertext even when the input plaintext is the same. This is because the intermediate initialization vectors are different in each case. Thus chaining has again eliminated patterns occurring in the ciphertext For this mode to produce the same ciphertext when the same plaintext is encrypted, both the key and the initialization vector must be identical in the two cases. This mode is similar to CBC in its resistance to forms of attack such as ciphertext searching, replay, insertion and deletion.

While both the CBC and ECB modes discussed earlier required padding of the last block of the message (if its length is not equal to an integral multiple of 64 bits), such problems do not arise in this stream cipher mode. That is, the key stream length can be matched exactly to the length of the plaintext to be enciphered. So this mode allows easy encryption of 'non-block type' messages such as character by character or even bit by bit encryption. Hence the CFB mode can be used for instance to encipher the short last block occurring at the end of the message in CBC mode (Section 5.3.3). However, CFB is less efficient than CBC in that it requires for each plaintext character one execution of the encryption algorithm. For example, the throughput of the DES operating on 8-bit CFB is reduced by a factor of 8 or more compared to the CBC mode. This can be overcome by enciphering n plaintext characters using the pseudo-random key stream produced by one DES cycle where n is the largest

- 98 -

Fig. 5.9 - CFB Ciphertext Example

Fig. 5.10 - CFB Error Characteristic

integer which is less than $[\frac{\text{block size of DES}}{\text{plaintext character size}}]$. In the case of 8-bit CFB for instance, one can encipher 8 plaintext characters (bytes) using the pseudo-random sequence produced by one DES cycle. This will produce a throughput comparable to that of the CBC without the need for padding.

Errors have been introduced within the ciphertext to study the error extension characteristics of the CFB mode. From Figure 5.10 it is seen that bit errors within one CFB ciphertext character affects not only the decryption of the garbled ciphertext character but also the decryption of the succeeding characters until the bit errors are shifted out of the CFB input block. The first affected plaintext character is garbled in exactly those places where the ciphertext character is in error. Successive plaintext characters experience an average error rate of 50% until all errors have been shifted out of the DES input block. In this 8-bit CFB case, errors in one ciphertext character is seen to affect decryption of nine characters. Further it is seen from Figure 5.10 that the CFB decryption automatically regains cryptographic synchronization. This self-synchronization property is also reflected in equations (5.3)and (5.4), Section 5.4.1. From equation (5.4), an error in ciphertext Y(i-1) can potentially affect every bit in the computed quantity f_k [U(i)] and hence can cause every bit in the recovered plaintext X(i) to be in error. From equation (5.4), an error in ciphertext Y(i-1) causes only the corresponding bit position in the recovered plaintext X(i-1) to be in error. The system synchronizes when U(i) becomes equal at both ends. Thus like the CBC mode, CFB mode provides limited error extension.

The recommendations given in Section 5.3.3 regarding the management of the CBC initialization vectors and its influence on the security are also applicable to CFB.

5.5   Cipher Block Chaining with Plaintext Feedback

5.5.1   Principle
            Cipher block chaining with plaintext feedback (CBCP) is a block cipher in which the plaintext is exclusive-ored not only with the previous ciphertext block but also with the previous plaintext block prior to being processed by the DES.
            This scheme is illustrated in Figure 5.11. The first DES

- 101 -
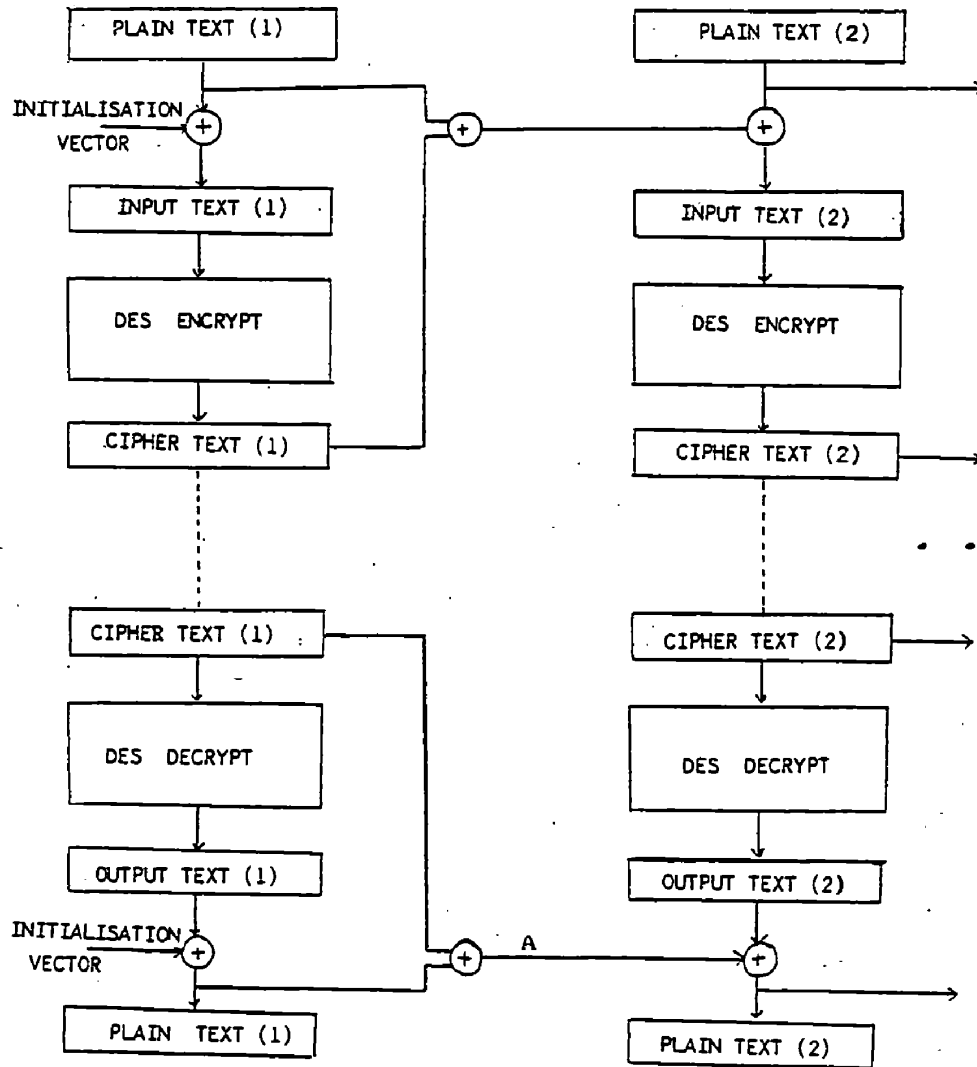
Fig 5.11 - Cipher Block Chaining with Plaintext Feedback (CBCP)

input block is formed by exclusive—oring the first plaintext block with a 64—bit initialization vector (IV). This input block is processed through the DES device producing a 64—bit DES output block which defines the first ciphertext block. Hence processing of the first plaintext block is same as in the CBC mode. This first ciphertext block is exclusive—ored with the second plaintext block as well as the first plaintext block to construct the second DES input block. The next DES operation produces the second ciphertext block. This chaining process is continued until the end of the message when the last block of ciphertext is obtained by encrypting the last input block formed by exclusive—oring last plaintext block with the (last—1) ciphertext and plaintext blocks.

In CBCP decryption, the first ciphertext block received is processed through the DES device to produce the DES output block. This first output block is exclusive—ored with the same initialization vector (IV) to produce the first plaintext block. Again the deciphering process of the first ciphertext block is same as in the CBC mode. The second ciphertext block is processed through the DES to yield the second output block which is then exclusive—ored with the first ciphertext and plaintext blocks to produce the second plaintext block. This process is continued until the end when the last block of plaintext is obtained by exclusive—oring the last DES output block with the (last—1) plaintext and ciphertext blocks.

Mathematically, the scheme can be expressed as follows:

Encipherment and decipherment procedures are given by:

$$Y(i) = f_k \left[X(i) \oplus U(i) \right] \quad i \geqslant 1 \qquad (5.5)$$

and

$$X(i) = f_k^{-1} \left[Y(i) \right] \oplus U(i) \quad i \geqslant 1 \qquad (5.6)$$

where

X(i) is the ith plaintext block

Y(i) is the corresponding ith ciphertext block

and

— 103 —

$$U(i) = \begin{cases} Z, \text{ the initialization vector} & i = 1 \\ \\ h[\, X(i-1),\ Y(i-1)\,] & i > 1 \end{cases}$$

Here h represents a simple exclusive–or function.

Therefore,

$$U(i) = X(i-1) \oplus Y(i-1)\ ,\ i > 1 \qquad\qquad (5.7)$$

Substituting (5.7) for U(i) into (5.5) gives

$$Y(i) = f_k\, [X(i) \oplus X(i-1) \oplus Y(i-1)]\ ,\ i > 1$$

From the recursive nature of the equations (5.5) and (5.7) it follows that there exist functions H_1, H_2, ..., H_i such that

$$Y(i) = H_i\, [k,\ X(0),\ X(1),\ ...,\ X(i)]\ ,\ i \geqslant 1 \qquad (5.8)$$

where $X(0) \equiv Z$.

Similarly from (5.6) and (5.7) it follows that,

$$X(i) = \begin{cases} f_k^{-1}\, [Y(i)] \oplus Z\ \cdot & i = 1 \\ \\ f_k^{-1}\, [Y(i)] \oplus Y(i-1) \oplus X(i-1) & i > 1 \end{cases}$$

Thus there exist functions G_1, G_2, ..., G_i such that

$$X(i) = G_i\, [k,\ Y(0),\ Y(1),\ ...,\ Y(i)]\ ,\ i \geqslant 1 \qquad (5.9)$$

where $Y(0) \equiv Z$.

From equation (5.8), one can see that the enciphering process is entirely deterministic and the output ciphertext block at time i, Y(i),is dependent only on the inputs to the ciphering process from time 1 through time i, namely, the key (k), the initialization vector (Z) and all the plaintext blocks X(1) through to X(i). Furthermore, since ciphertext block Y(i) is dependent on

– 104 –

the initial conditions established at the beginning of the ciphering process, at time 1, it is said to be origin dependent[3].

From equation (5.9), it can be seen that the recovered plaintext block at time i, X(i), depends only on the key (k), the initialization vector (Z) and all ciphertext blocks Y(1) through to Y(i). Thus X(i) is also origin dependent.

This scheme is defined to be a general block cipher[3].

5.5.2 Implementation

The program implementing this mode of operation is very much similar to the one used for the CBC mode. The only difference is that in this case, a plaintext block is not only exclusive-ored with the previous ciphertext block (TEMP 1) but also with the previous plaintext block prior to DES encryption. A similar difference occurs after DES decryption.

5.5.3 Results and Discussion

The plaintext example shown in Figure 5.2 is enciphered under this mode using the same key and initialization vector as before. The ciphertext produced is shown in Figure 5.12.

Again from Figure 5.12 it is seen that there is no repetition of ciphertext even when the plaintext is repetitive. This mode is similar to CBC and CFB in its resistance to forms of attack such as ciphertext searching, replay, insertion and deletion. The most interesting property of this mode however is that of error extension. The deciphered version of the example with some errors introduced in the ciphertext prior to decryption is shown in Figure 5.13. Two points are worth to be noted. The first one is that an error in the ciphertext affects the decryption of all subsequent blocks until the end of the message. That is, the scheme exhibits the property of error propagation. This agrees with equation (5.9) where every bit of the recovered plaintext block X(i) is a function of every bit in the ciphertext blocks Y(1) through to Y(i). The only case in which the error is not propagated occurs when the corrupted ciphertext block Y(i)* and the deciphered value of Y(i)* under key k obey the equality

$$Y(i)^* \oplus f_k^{-1}(Y(i)^*) = Y(i) \oplus f_k^{-1}(Y(i))$$

- 105 -

Fig. 5.12 - CBCP Ciphertext Example

Fig. 5.13 - CBCP Error Characteristic

That is, the feedback value at point A(Figure 5.11)input to the next cycle, is unchanged. Assuming that the probability that an error in $Y(i)$ causing each bit in $f_k^{-1}(Y(i)*)$ to differ from the corresponding bit in $f_k^{-1}(Y(i))$ is approximately equal to 0.5, the probability that an error cancellation occurs is approximately equal to 2^{-64}.

Secondly, when deciphering the ciphertext containing errors with the right key and the right initialization vector, it is seen that patterns or repetitions in the garbled deciphered text are revealed which correspond to the patterns in the plaintext. The reason for this occurrence is given as follows:

First consider the case where there are no errors in the ciphertext prior to decryption. Referring to the diagram shown below, assume that the plaintext blocks 2 and 3 are the same (but their corresponding ciphertexts C_1 and C_2 will be different due to chaining). Then, for the deciphered block 3 to be equal to block 2, P_2, one must have

$$O_2^2 \oplus C_2^1 \oplus P_2 = P_2$$

ie ,

$$O_2^2 \oplus C_2^1 = (0 \dots 0) = \text{Zero block}$$
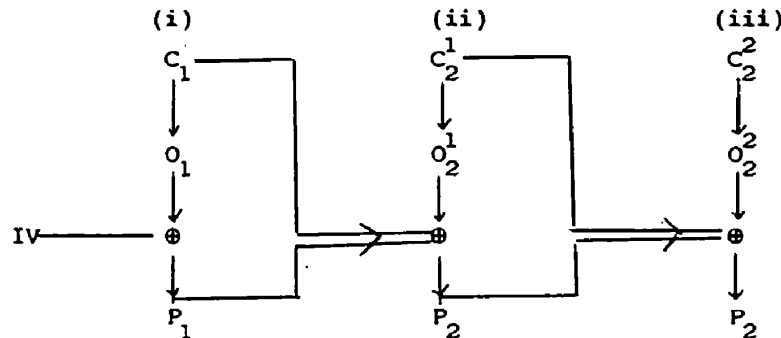
Now if the block C_1 has an error, C_1^*, then this decryption results in plaintext block, P_1^*. The second block P_2^* is then equal to $C_1^* \oplus P_1^* \oplus O_2^1$. The third block is given by $O_2^2 \oplus C_2^1 \oplus P_2^*$ and this is equal to P_2^* because $O_2^2 \oplus C_2^1$ is still equal to the zero block as it is unaffected by error in C_1^* . Thus this repetition in the deciphered version will occur as long as the two successive blocks are the same in the plaintext. When P_2 and P_3 are different, then

- 108 -

$P_2 \oplus C_2^1 \oplus O_2^2 = P_3$ and $C_2^1 \oplus O_2^2$ is not equal to the zero block. Therefore when an error occurs in C_1^* , $P_2^* \oplus C_2^1 \oplus O_2^2 = P_3^* \neq P_2^*$.

This pattern occurrence is not of great concern as far as the security of this scheme is concerned because here we are talking about the legitimate receiver with the right key and the right initialization vector detecting patterns in the decrypted version of the garbled ciphertext. This scheme is not at all suitable for communication links prone to noise. On the other hand, this error propagation property can be used to prevent 'spoofing' attack. This scheme is very much suitable for message authentication purposes where one needs to determine with a high level of confidence whether the message has been altered.

As CBCP is a block cipher padding is again required at the end of the message like the ECB and the CBC modes.

5.6   Stream Cipher Feedback with Vector   Feedback

5.6.1 Principle

This is an additive stream cipher technique similar to CFB in which the DES is used to generate a pseudo-random binary stream. This stream is exclusived-ored with the plaintext to form the ciphertext. The ciphertext together with the initialization vector is then fed back to form the next DES input block. The feedback from the initialization vector is the feature which differentiates it from the CFB mode. It is referred to as CFBV.

This scheme is illustrated in Figure 5.14. The initialization vector forms the first DES input block. This is encrypted by the DES device producing a 64-bit pseudo-random output block. The rightmost or the least significant s-bits ($1 \leqslant s \leqslant 64$) of the DES output block are exclusive-ored with the s-bits of the plaintext to form s-bits of ciphertext. These s-bits of the ciphertext are expanded to form a 64-bit block by repetition. This block is then exclusive-ored with the previous initialization vector and the result is shifted by s-bits to form the next DES input block, the new initialization vector. This is then used in the encryption of the next s-bit plaintext character. This process is repeated until the end of the message.

In CFBV decryption, the first output block, produced by encrypting the same initialization vector, is exclusive-ored with the

- 109 -
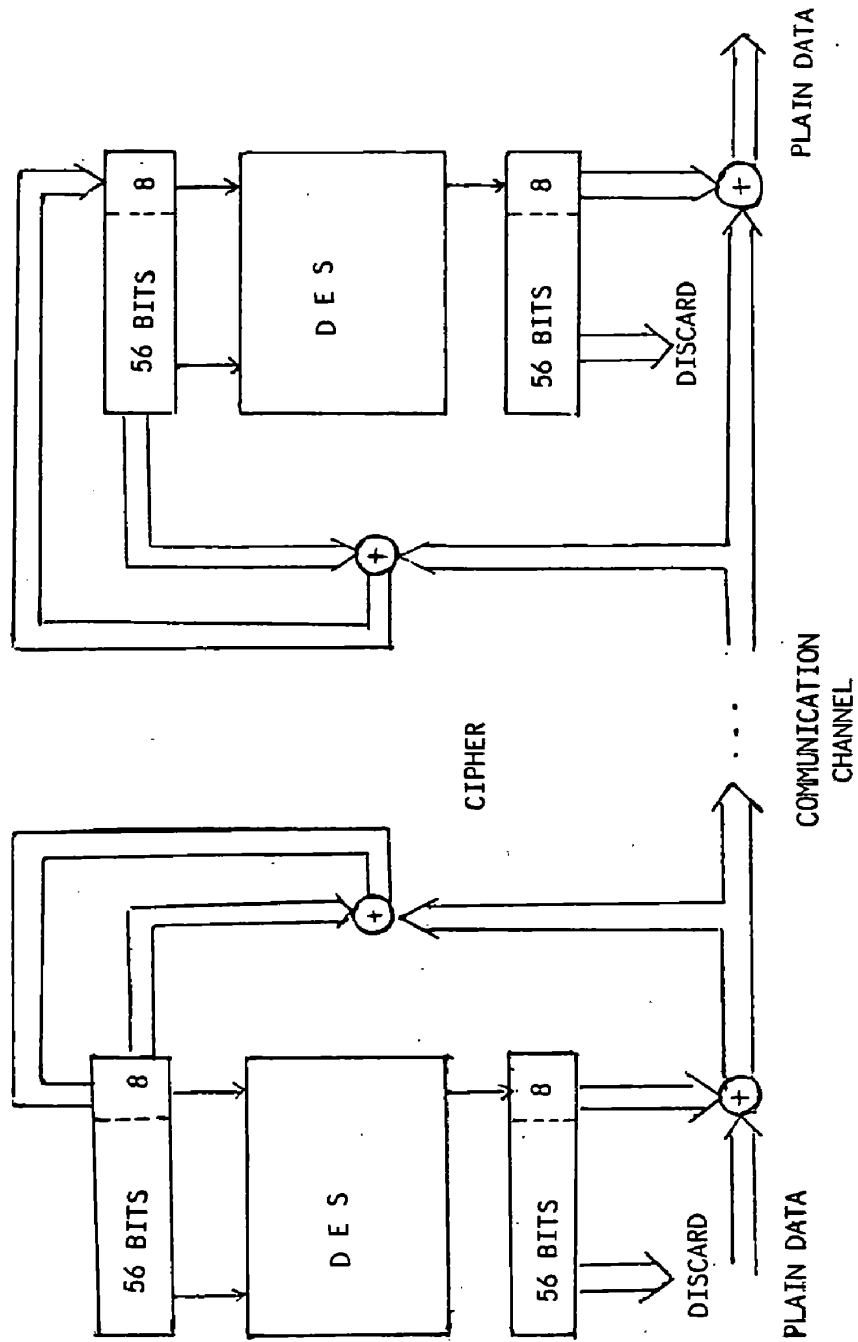
Fig. 5.14. -Stream Cipher with Ciphertext and Vector Feedback (CFBV)

- 110 -

first s-bit ciphertext character to produce the first s-bit plaintext character. The first s-bit ciphertext character is then expanded to form a 64-bit block which is exclusive-ored with the previous initialization vector. The result is shifted by s-bits to form the next DES input block. This process is repeated until the end of the message.

Note that the first cycle of this scheme is exactly the same as in CFB. Furthermore as in CFB, the DES device is used in its encryption state at both ends.

Mathematically, this scheme can be described as follows: The encipherment and decipherment can be expressed as

$$Y(i) = X(i) \oplus f_k [U(i)] \qquad i \geq 1 \qquad (5.10)$$

and

$$X(i) = Y(i) \oplus f_k [U(i)] \qquad i \geq 1 \qquad (5.11)$$

respectively

where

X(i) is the ith plaintext character

Y(i) is the corresponding ith ciphertext character

$$U(i) = \begin{cases} Z, \text{ initialization vector, } i = 1 \\ \\ h [U(i-1), Y(i-1)] , \qquad i > 1 \end{cases} \qquad (5.12)$$

Here the function h is given by

$$h [U(i-1), Y(i-1)] = E (Y(i-1)) \oplus U(i-1), i > 1$$

where

E (Y(i-1)) represents expansion of s-bit ciphertext characters Y(i-1) repetitively to form a 64-bit block.

From the recursive nature of the equations (5.10), (5.11) and (5.12), it follows that there exist functions H_1, H_2, ..., H_i and G_1, G_2, ...,

− 111 −

Page 111

G_i, such that

$$Y(i) = X(i) + H_i [k, X(0), X(1), \ldots, X(i-1)] \quad i \geq 1 \quad (5.13)$$

and

$$X(i) = Y(i) + G_i [k, Y(0), Y(1), \ldots, Y(i-1)] \quad i \geq 1 \quad (5.14)$$

where

$$X(0) \equiv Y(0) \equiv Z$$

The equations (5.13) and (5.14) are counterparts of the equations (5.8) and (5.9) given in Section 5.5.1. Hence this scheme represents a general stream cipher [3].

From equation (5.13), it follows that the jth bit in the ciphertext character $Y(i)$ is directly affected by only the jth bit in the plaintext character $X(i)$ whereas it is potentially affected by every bit in plaintext characters $X(1)$ through to $X(i-1)$. In like manner, from equation (5.14), it follows that the jth bit in the received plaintext character $X(i)$ is directly affected by only the jth bit in the ciphertext character $Y(i)$ whereas it is potentially affected by every bit in ciphertext characters $Y(1)$ through to $Y(i-1)$.

5.6.2 Implementation

The program implementing this mode of operation with s=8 is very much similar to the one used for the CFB mode. The only difference compared to the CFB mode is that in this case, the new initialization vector is formed by exclusive-oring the previous initialization vector with the expanded previous 8-bit ciphertext character.

5.6.3 Results and Discussion

The plaintext example shown in Figure 5.2 is enciphered under this mode using the same key and initialization vector as before. The ciphertext produced is shown in Figure 5.15.

Like all the other modes except the ECB discussed earlier this mode masks the patterns in the plaintext thus reducing the code book analysis, replay, insertion and deletion attacks. It is

- 112 -

Fig. 5.15 - CFBV Ciphertext Example

- 113 -

Fig. 5.16 – CFBV Error Characteristic

similar to CFB in that the messages can be processed character by character thus avoiding the padding required in ECB, CBC and CBCP modes.

The most interesting property of this stream cipher mode is that of error extension. The deciphered version of the example with some errors introduced in the ciphertext prior to decryption is shown in Figure 5.16. Two points are worth mentioning. The first one is that an error in a ciphertext character is found to affect the decryption of all subsequent ciphertext characters until the end of the message. That is, like CBCP, this stream cipher exhibits the property of error propagation. This can also be seen from equation (5.14). Since the recovered plaintext $X(i)$ is potentially affected by every bit in the ciphertexts $Y(1)$ through to $Y(i-1)$, error propagation is achieved. However because the jth bit in the plaintext $X(i)$ depends only on the jth bit in the ciphertext $Y(i)$, the intersymbol dependence can be achieved for all but the final plaintext. On the other hand with the CBCP, there is intersymbol dependence throughout all blocks.

Secondly, when deciphering this ciphertext containing errors with correct key and initialization vector, it is seen from Figure 5.16 that there is no pattern or repetition in the garbled decrypted text. This is in contrast to the CBCP scheme considered in Section 5.5.3. So even the legitimate user with right key and right initialization vector gets a completely garbled text when the errors are introduced in the ciphertext. This means that this scheme is not suitable for links prone to noise but is very useful for message authentication purposes.

C H A P T E R   6

STATISTICAL TESTS ON DES OUTPUT SEQUENCES

6.1   General

Some statistical tests are applied to the output sequences
obtained using the DES algorithm under different modes (see Chapter 5)
to test for their randomness properties. Some simple statistical tests
are also considered with a view to detecting the dependence or
correlation between the output and inputs to the DES and to determining
whether plaintext-ciphertext pairs could be used to predict the bits of
the key.

6.2   Statistical Tests for Randomness

Strictly speaking, no finite sequence is ever truly random.
The best that can be done is to single out certain properties as
being associated with randomness and to accept any sequence which has
these properties as a random sequence. In particular, it is assumed
that the opponent intercepts sections of ciphertext sequence and attempts
to exploit the statistical properties of the sequence in his crypt-
analytical attack. Therefore it is necessary to apply the statistical
tests to sections of ciphertext sequence to check their randomness
characteristics. This type of randomness is often referred to as local
randomness [14].

There are several statistical tests which can be applied to
a sequence. Here four fundamental tests have been considered which
can be used to provide a quantitative measure of randomness [14]. They
are the frequency test, the serial test, the runs test and the auto-
correlation test. All these tests measure the relative frequencies of
certain patterns of 'O's and'l's in the sequence considered, in one way
or another. The sequence under consideration is then regarded to be
random if the sequence passes the test. Levels of confidence are set so
as to decide if the sequence is random enough for our purpose.

Initially, it is necessary to choose the length of the
section or sample to be tested. The sample size must not be too large
to swamp local variations but at the same time it must not be too
small preventing any reasonable conclusions.   Accordingly, for

the test in question, a sample size of n = 1024 is chosen. As the next step, the properties of the input to the DES algorithm are defined. The procedure adopted is as follows: In each case, 'non-random' inputs are applied to the DES and the output is tested for its randomness. For this reason, the inputs are chosen to be periodic sequences with different period lengths, thus allowing a variable number of cycles to be present within the selected sample size.

Six different cycle lengths are selected namely 5, 8, 10, 20, 40 and 64 digits. Within each category, five input samples are chosen. Each of these inputs is encrypted using DES under the five different modes namely, the ECB, the CBC, the CFB, the CBCP and the CFBV. This procedure is carried out using five different DES keys. Among the chosen keys are included a weak key (see Section 4.6.4) and a semi-weak key (see Section 4.6.4), a 'non-random' key and two arbitrarily selected 'random' keys. The input samples and the keys used are given in Appendix 7, Section A7.1.

The tests are performed in two parts. In the first part, the tests are applied to output samples produced by encryption under the different DES modes for a fixed key. This is done to investigate the effect of different modes of encryption on randomness of the output. In the second part, the key to the DES is varied to find the effect of key on the randomness of the output.

The four tests and the confidence levels which indicate whether a sequence is random or non-random are now briefly outlined [14].

6.2.1 Test 1: The Frequency Test

The frequency test checks whether there is approximately the same number of '0's and '1's in the sequence.

Let the length of the sequence be n and let it contain n_0 zeroes and n_1 ones. Defining,

$$\chi^2 = \frac{(n_0 - n_1)^2}{n}$$

$\chi^2 = 0$ when $n_0 = n_1$. Larger the value of χ^2, greater the discrepancy between the observed and expected frequencies. This is a χ^2- test with one degree of freedom. Thus if the value of χ^2 is

– 117 –

not greater than 3.84, then from the table of χ^2 distribution given in [28], the sequence is passed at 5% significance level. (Note that if $\chi^2 = 0$, the sequence might also be rejected on the grounds of it being too good!).

6.2.2 Test 2: The Serial Test

The serial test checks whether the transition probabilities are reasonable, that is, the probability of consecutive entries being equal or different is about the same. This then gives some level of confidence that each bit is independent of its predecessor.

Supposing that the sequences 00 occurs n_{00} times, 01 occurs n_{01} times, 10 occurs n_{10} times and 11 occurs n_{11} times, then

$$n_{01} + n_{00} = n_0 \quad \text{or} \quad n_0 - 1$$
$$n_{10} + n_{11} = n_1 \quad \text{or} \quad n_1 - 1$$

and

$$n_{00} + n_{01} + n_{10} + n_{11} = n-1$$

(Note that n-1 occurs because in a sequence length of n bits, there are only n-1 transitions).

Ideally, we want $n_{00} = n_{01} = n_{10} = n_{11} \triangleq \frac{n-1}{4}$. Good [29] has shown that

$$\frac{4}{n-1} \sum_{i=0}^{1} \sum_{j=0}^{1} (n_{ij})^2 - \frac{2}{n} \sum_{i=0}^{1} (n_i)^2 + 1 \qquad (6.1)$$

is approximately distributed as χ^2 with two degrees of freedom. The value of χ^2 corresponding to a 5% significance level with two degrees of freedom is 5.99. Hence the sequence is rejected if the value of (6.1) is greater than 5.99.

6.2.3 Test 3: The Runs Test

The runs test is based on the theory of runs where a run is a succession of identical letters (zeroes or ones) which is followed and preceded by different letters. The total number of runs is often a good indication of a possible lack of randomness.

To find the probability that n_0 zeroes and n_1 ones will

- 118 -

form u runs when each of $\binom{n_0 + n_1}{n_1}$ possible arrangement of these letters is equally likely, first consider the case where u is even, namely, u = 2t, for some positive integer t. There are $\binom{n_0-1}{t-1}$ ways in which the n_0 zeroes can form t runs and $\binom{n_1-1}{t-1}$ ways in which the n_1 ones can form t runs. It follows then that there are altogether $2\binom{n_1-1}{t-1}\binom{n_0-1}{t-1}$ ways in which those $n_1 + n_0$ letters can form 2t runs. The factor 2 is accounted for by the fact that when the two kinds of runs are combined so that they alternate, we can begin with a run of zero or with a run of one. Thus when u = 2t, the probability of getting u runs is

$$f(u) = \frac{2 \binom{n_0-1}{t-1} \binom{n_1-1}{t-1}}{\binom{n_0 + n_1}{n_1}}$$

When u = 2t + 1, similar arguments lead to

$$f(u) = \frac{\binom{n_0-1}{t} \binom{n_1-1}{t-1} + \binom{n_0-1}{t-1} \binom{n_1-1}{t}}{\binom{n_0 + n_0}{n_1}}$$

When n_0 and n_1 are both greater than 10 or more, the sampling distribution of u can be approximated with a normal distribution. Making use of this distribution, Gibbons [30] shows that, the expected value E(u) and variance Var (u) are as follows:

$$E(u) = \frac{2 n_0 n_1}{n_0 + n_1} + 1$$

and

$$Var(u) = \frac{2 n_0 n_1 (2n_0 n_1 - n_0 - n_1)}{(n_0 + n_1)^2 (n_0 + n_1 - 1)}$$

Thus for sufficiently large value of n_0 and n_1, the normal test variable Z is given by

$$Z = \frac{u - E(u)}{\sqrt{Var(u)}}$$

The null hypothesis that the sequence is random is rejected if

– 119 –

$|Z| \geqslant 1.96$ at 5% significance level.

6.2.4 Test 4: The Autocorrelation Test

The autocorrelation function of the sequence is an important element when testing for randomness. Random sequences possess a special kind of autocorrelation function namely peaked in the middle and tapering off rapidly at the ends. Autocorrelation also reflects the periodicity within the sequence.

If $\{ x_0, x_1, \ldots, x_n \}$ is a binary sequence, then its autocorrelation function can be defined as

$$A(r) = \frac{1}{n-r} \sum_{i=1}^{n-r} x_i \cdot x_{i+r} \qquad \text{for } r = 0, 1, \ldots, m$$

Here a slightly modified version of $A(r)$ has been used. The operation between x_i and x_{i+r} is defined as one of matching (comparison) instead of direct multiplication. That is,

$$A(r) = \frac{1}{n-r} \sum_{i=1}^{n-r} z_{ir}$$

where
$$z_{ir} = \begin{cases} 1 & \text{if } x_i = x_{i+r} \\ 0 & \text{if } x_i \neq x_{i+r} \end{cases}$$

That is, $A(r)$ is some sort of a measure of number of times the shifted and original sequences match, both x_i and x_{i+r} are equal to one or zero. This operation is more sensitive than direct multiplication which just tests for matching ones.

6.2.4.1 Expected Value of $A(r)$, $E(A(r))$

The probability that x_i and x_{i+r} match is given by $\text{Prob}[z_{ir} = 1]$.

For $r \neq 0$

$$\text{Prob}[z_{ir} = 1] = \text{Prob}(x_i = 1 \text{ and } x_{i+r} = 1) + \text{Prob}(x_i = 0 \text{ and } x_{i+r} = 0)$$

- 120 -

Let the probability that $x_i = 1$ be p and the probability that $x_i = 0$ be q. Then,

$$\text{Prob } [Z_{ir} = 1] = p^2 + q^2$$

Therefore $p_r[Z_{ir} = 0] = 1 - (p^2 + q^2) = 2pq$

Hence $E(Z_{ir}) = (p^2 + q^2) \cdot 1 + (2pq) \cdot 0$
$$= p^2 + q^2$$

But $E(A(r)) = \frac{1}{n-r} E\left(\sum_{i=1}^{n-r} Z_{ir}\right)$

$$= \frac{1}{n-r} (n-r) E(Z_{ir})$$

$$E(A(r)) = p^2 + q^2$$

For a random sequence, assuming that the probabilities p and q is a ½ this yields

$$E(A(r)) \longrightarrow \tfrac{1}{2}$$

This is to be expected as in a random sequence, the probability of observing a match of zeroes or ones is equal to that of not observing such a match. The mean value itself is an indication of non-randomness. From this, if the sequence is random, its autocorrelation function should vary around the 0·5 mark.

6.2.4.2 $\underline{\text{Variance of } A(r), \text{ Var } A(r)}$
$\text{Var}(A(r)) = E(A^2(r)) - [E(A(r)]^2$

Considering first $E(A^2(r))$, we have

$$E(A^2(r)) = \frac{1}{(n-r)^2} E\left(\sum_{i=1}^{n-r} \sum_{j=1}^{n-r} (x_i x_{i+r} x_j x_{j+r})\right)$$

$$= \frac{1}{(n-r)^2} E\left(\sum_{i=1}^{n-r} \sum_{j=1}^{n-r} (Z_{ir} Z_{jr})\right)$$

$$= \frac{1}{(n-r)^2} \sum_{i=1}^{n-r} \sum_{j=1}^{n-r} E(Z_{ir} Z_{jr}) \qquad (6.2)$$

$$- 121 -$$

Dividing the expression (6.2) into three cases namely (i) i=j, (ii) i≠j and i=j+r or j=i+r and (iii) i≠j and i≠j+r or j≠i+r, we have

$$E(A^2(r)) = \frac{1}{(n-r)^2} \sum_{i=1}^{n-r} E(Z_{ir}\ Z_{ir}) + \frac{2}{(n-r)^2} \sum_{i=r+1}^{n-r} E(Z_{ir}\ Z_{i-r,r})$$

$$+ \frac{1}{(n-r)^2} \sum_i \sum_j E(Z_{ir}\ Z_{jr})$$

$$E(Z_{ir}\ Z_{ir}) = E(Z_{ir}) = p^2 + q^2$$

Now consider the second term $E(Z_{ir}\ Z_{i-r,r})$

$$Z_{ir}\ Z_{i-r,r} = \begin{cases} 1 & \text{if } Z_{ir} = Z_{i-r,r} \\ 0 & \text{otherwise} \end{cases}$$

Note that here Z_{ir} and $Z_{i-r,r}$ are not independent because

$$Z_{ir} = x_i \odot x_{i+r} \quad \text{and} \quad Z_{i-r,r} = x_{i-r} \odot x_i$$

Where $\odot$ refers to matching.

Therefore,

$$\text{Prob}(Z_{ir},\ Z_{i-r,r} = 1) = \text{Prob}(x_i \odot x_{i+r} \odot x_{i-r} = 1) +$$

$$\text{Prob}(x_i \odot x_{i+r} \odot x_{i-r} = 0)$$

$$= p^3 + q^3$$

Therefore,

$$\frac{2}{(n-r)^2} \sum_{i=r+1}^{n-r} E(Z_{ir}\ Z_{i-r,r}) = \frac{2(n-2r)}{(n-r)^2}\ (p^3 + q^3) \qquad (6.3)$$

n-2r is equal to the number of terms for which i ≠ j and i = j+r. By symmetry, the number of terms for which, i ≠ j and j = i+r is n-2r. Hence the factor 2 in the expression (6.3).

- 122 -

Now consider the third term $\frac{1}{(n-r)}^2 \sum_i \sum_j E(Z_{ir} Z_{jr})$ where $i \neq j$, $i \neq j+r$ and $j \neq i+r$.

$$Z_{ir}Z_{jr} = \begin{cases} 1 & Z_{ir} = Z_{jr} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$\text{Prob}(Z_{ir} Z_{jr} = 1) = \text{prob}(Z_{ir} = 1)$ and $\text{prob}(Z_{jr} = 1)$

$$= (p^2 + q^2)(p^2 + q^2)$$

$$= (p^2 + q^2)^2$$

The number of terms in the double sum $\sum_i \sum_j$ is given by

$$= \text{total number of terms} - [(n-r) + 2(n-2r)]$$

$$= (n-r)^2 - (n-r) - 2(n-2r)$$

Therefore

$$\frac{1}{(n-r)}^2 \sum_i \sum_j E(Z_{ir} Z_{jr}) = \left(1 - \frac{2(n-2r)}{(n-r)^2} - \frac{1}{n-r}\right)(p^2 + q^2)^2$$

Hence,

$$E(A^2(r)) = \frac{p^2 + q^2}{(n-r)} + \frac{2(n-2r)}{(n-r)^2}(p^3 + q^3) + \left[1 - \frac{2(n-2r)}{(n-r)^2} - \frac{1}{n-r}\right](p^2 + q^2)^2$$

For a random sequence, it is assumed that $p = q = \frac{1}{2}$. Substituting this in the above expression for $E(A^2(r))$ gives,

$$E(A^2(r)) = \frac{1}{2(n-r)} + \frac{n-2r}{2(n-r)^2} + \frac{1}{4} - \frac{(n-2r)}{2(n-r)^2} - \frac{1}{4(n-r)}$$

$$= \frac{1}{4(n-r)} + \frac{1}{4}$$

$$\text{Var } A(r) = E(A^2(r)) - [E(A(r))]^2$$

$$= \frac{1}{4(n-r)} + \frac{1}{4} - [\tfrac{1}{2}]^2 = \frac{1}{4(n-r)}$$

$- 123 -$

Thus the variance is inversely proportional to (n-r). That is, greater the sample size n, smaller the variation from the mean and as the lag r increases, the variance increases. Having calculated the expected value and the variance of A(r), one can approximate the distribution with a normal distribution for sufficiently large value of n. Thus the normal test variable N(r) is given by

$$N(r) = \left| \frac{A(r) - E(A(r))}{\sqrt{Var\ A(r)}} \right|$$

Thus at 5% significance level if $|N(r)| < 1 \cdot 96$, the sequence is said to be random. In this test, the autocorrelation of the ciphertext sequences are computed and the number of $|N(r)|$ values which exceed $1 \cdot 96$ is used as a measure of non-randomness of the sequence.

6.3   Results and Discussion

        Section A7.2 in Appendix 7 gives the results of the frequency, serial and runs tests on the ciphertext sequences produced using the ECB, CFB, CBC, CBCP and CFBV modes of DES under a fixed key namely 3131313131313131. Five input samples for each cycle length have been encrypted under the five chosen modes of DES. The notation used in Section A7.2 is : (Encryption Mode r.s), where r indicates the cycle length of the input sample and s indicates the number of the input sample within the cycle length category, r. (see Section A7.1). For instance CFB 1.2 refers to the mode CFB, input sample number 2, having a cycle length of 5 digits. The figures marked with (*) indicate that the values are in proximity to the 5% significance level and the figures marked with (**) indicate the values beyond the 5% significance level, thus showing a possible lack of randomness of the sequence under consideration.

        It is seen from the results in Section A 7.2 that in general the DES under the ECB mode shows the most non-randomness characteristics out of the selected five modes. Section A7.2.1 shows, under each mode, the number of sequences which are classified as non-random by each of the three tests. In Section A7.2.2 are listed the sequences which are classified as non-random by more than one test. Again from Sections A7.2.1 and A7.2.2, it is seen that the ECB mode seems to produce the most non-random sequences out of the five modes

- 124 -

considered.

Section A7.3 in Appendix 7 gives the results of the
frequency, serial and runs tests on the ciphertext sequences produced
using the five different chosen keys (see Section A7.1) under the
five DES modes. In this case, one input sample from each cycle length
category has been chosen for testing purposes. The input samples
selected are themselves classified as non-random by more than one of
the three tests. The chosen input sequences are : 1.5, 2.1, 3.3,
4.1, 5.2 and 6.5 (see Section A7.1). These six inputs are renumbered
as (i) to (vi) respectively. The five keys used are labelled I to V
(see Section A7.1). It is seen from the results in Section A7.3
that the variation of the key does not seem to produce any appreciable
difference in the randomness characteristic of the output sequences.
Section A7.3.1 shows, under each key, the number of sequences which
are classified as non-random by each of the three tests. In Section
A7.3.2 are listed the sequences which are classified as non-random
by more than one test. Again from Sections A7.3.1 and A7.3.2, there
seems to be no great significant effect on randomness due to change
in the DES keys. This seems to suggest that the sequences produced
are more or less random like for any key being used. If so, this
may be regarded as an important positive aspect of the DES crypto-
graphic algorithm. If there were some keys which produced significant
non-random sequences, then this might be used in cryptanalytical
attacks and hence may be considered as a weakness.

Two input samples having cycle lengths of 5 and 10 digits
(input samples 1.5 and 3.3), encrypted using the keys I and V, have
been used in the autocorrelation test. Only the autocorrelation
function curves of the input sample 1.5 (denoted as sample (i))and
its five ciphertext output sequences produced using the five DES
modes with key V, are shown in Section A7.4. The autocorrelation
function of the input reflects the periodic nature of the input
sample (repetition of 5 digits). Further the minimum value of the
input autocorrelation curve depends on the relative proportion of
zeroes and ones present in the input sequence. Greater the proportion
of '1', higher the mean value and hence higher the minimum value of
the autocorrelation curve. From the ciphertext autocorrelation curves,
it is seen that their mean value is around 0.5 which agrees with the
expected value $E(A(r))$ derived earlier for a random sequence. It is
also seen that the variation of the curve around the mean value seems

– 125 –

to be the greatest in the case of ECB mode. Section A7.4.1 gives
the number of points M which lie beyond the 5% significance level,
calculated using normal distribution approximation given in Section
6.2.4. The value of M can be used as a measure of randomness;
greater the value of M, less is the randomness of the sequence. It
is seen that for the input sample (i), with ECB mode, about 19% of
the points lie outside the 5% significant level thus indicating some
degree of non-randomness.

To sum up, it can be said that, from the four tests -
frequency, serial, runs and autocorrelation, the DES algorithm seems
to be a very good pseudo-random number generator. Out of the five
modes considered - ECB, CFB, CBC, CBCP and CFBV - the ECB mode seems
to produce the most non-randomness characteristics. There does not
seem to be any great difference between the other modes from the
point of view of randomness of output sequences produced.

Having examined the randomness characteristics of the final
ciphertext output from the DES algorithm, the next step is to apply
these tests to the intermediate outputs namely the outputs of the 16
rounds of the DES operated in the standard ECB mode. The degree of
randomness is expected to increase as the number of rounds increases.
It is found that the first round shows a high degree of non-randomness
and as the number of rounds increases the outputs become more and
more random. Some of the results obtained are given in Section
A7.5.

6.4   Other Statistical Tests

In this section, some other statistical tests which are
carried out to detect dependence or correlation between the inputs
and the output of DES are briefly mentioned.

6.4.1 Cross-Correlation Test

The aim of this test is to find out whether there is any
significant correlation between the ciphertext sequences produced
using some special inputs and whether it can be of any use in a
cryptanalytical attack on DES.

A key consisting of a '1' in the most significant position
and sixty three '0's is chosen to encrypt a plaintext block of all
'0's. Then the bits are shifted one position to right and the key

- 126 -

is used to encrypt the same plaintext. This procedure is repeated
64 times, by shifting each time the key bits one position to the
right. Out of the 64 ciphertexts produced, only 56 ciphertext
sequences are distinct, since 8 bits of the key are used as parity
bits. In order to determine whether there is any correlation between
these 56 sequences, it is necessary to examine the cross correlation
of each sequence with the other 55 sequences. As this seemed to be
impracticable, two sequences are chosen and they are cross correlated
with the rest. The chosen sequences are:

(i)   the first ciphertext sequence corresponding to the key
            100... 0
(ii)   the 29th ciphertext sequence corresponding to key $0...1...0$

The results obtained showed cases with high degree of correlation;
three such cases are given below:

(a)   first output cross-correlated with the third output with
            lag $r = 1$
(b)   first output cross-correlated with the fourth output with
            lag $r = 1$
(c)   first output cross-correlated with the eleventh output with
            lag $r = 0$

To investigate these cases further, four other plaintext blocks are
encrypted under the same key using the above procedure. The results
are then examined for correlation for the three cases stated above.
Further these four plaintexts are encrypted using four different keys
following the above procedure and the results examined for correlation
for the three cases (a), (b) and (c) above. However, these
investigations showed that no conclusive evidence can be found of
the existence of any systematic correlation between these 56 cipher-
text sequences.

6.4.2 χ^2-Test to Detect Dependence Between Output and Input

A χ^2-test [6] has been used to detect whether there is any
dependence of an output sub-block on an input sub-block for a fixed
key.

Let the plaintext block $X = (x_0...x_{63})$ be enciphered by
DES with key $K = (k_0,..., k_{55})$ into ciphertext $Y = (y_0,...,y_{63})$.

- 127 -

Page 127

In DES, each output bit co-ordinate y_i is a function of the 64 input bit co-ordinates x_i and the 56 key co-ordinates k_i. The χ^2 test enables to check whether some set of the output bit co-ordinates CO_{out} are dependent on some set of the input bit co-ordinates CO_{in} for a fixed key. Note that the test can also be used [6] to test for the dependency of some set of output bit co-ordinates on some set of key co-ordinates for a fixed input. Dependence might be used to estimate the key or plaintext; for example, if k_i is dependent on some set of output or input bit positions, one could make this the basis of the recovery of k_i from corresponding plaintext and ciphertext.

The χ^2 test used is explained below [6]:

(i)   Small subsets of CO_{in} and CO_{out} of sizes Nin and Nout are chosen where $CO_{in} = (i_0, \ldots, i_{Nin-1})$, $CO_{out} = (j_0, \ldots, j_{Nout-1})$

(ii)   A key $K = (k_0, \ldots k_{55})$ is chosen

(iii)   A set of plaintexts is encrypted under the key K and the 2^{Nin} by 2^{Nout} contingency table is formed where the (s,t) entry is the number of times

$$(x_{i,i_0}, \ldots x_{i,i_{Nin}}) = (s_0, \ldots, s_{Nin-1})$$

$$(y_{i,j_0}, \ldots, y_{i,j_{Nout}}) = (t_0, \ldots, t_{Nout-1})$$

where $(s_0, \ldots, s_{Nin-1})$, $(t_0, \ldots, t_{Nout-1})$ are the base 2 representations of s and t.

(iv)   Then the χ^2 statistic with $2^{Nin+Nout-1}$ degrees of freedom is computed where

$$\chi^2 = \sum_{0 \leq s < 2^{Nin}} \sum_{0 \leq t < 2^{Nout}} \frac{\left(N_{s,t}(X) - N_{samples} \, 2^{-(Nout+Nin)}\right)^2}{2^{-(Nout+Nin)} \, N_{samples}}$$

In our case, $N_{samples} = 250$ and the key is 31313131313131. The subsets are chosen to be $CO_{in} = CO_{out} = (3,4)$. The contingency table for a two-tailed χ^2-test at 1% and 99% confidence levels is given as follows:

- 128 -

	00	01	10	11
00	20	21	11	9
01	19	10	18	16
10	25	17	13	12
11	22	10	16	11

The computed χ^2-value is 23.408. Using the two-tailed χ^2-test, the 1% and 99% confidence levels are given by χ^2_{lower} = 5.81 and χ^2_{upper} = 32 respectively. This leads us to accept the null hypothesis that the output bit positions in CO_{out} are independent of the input bit positions in CO_{in} with the chosen key. In practice, the acceptance or rejection of the null hypothesis must be based upon the results of several independent χ^2-tests. The evaluation of multiple χ^2-tests is often made using Kolmogorov-Smirnov test [6]. Even with multiple χ^2-tests for the correlation to be of any value in cryptanalyzing the DES, either the correlation is present for only a limited number of pairs (which can be predetermined) or correlation is present in a relatively large number of pairs which can be determined by random sampling. Thus it is necessary to carry out these tests on all pairs (CO_{in}, CO_{out}). The application of χ^2-test is reported [6] to have been carried out by IBM and NSA as part of internal validation of DES.

CHAPTER 7

LOCAL FILE SECURITY: SYSTEM SOFTWARE (2)

7.1   General

The second application of the developed encryption interface
unit is the encryption and decryption of files stored locally in the
Apple disk system. This application offers the system off-line
encryption facility whereas the point-to-point configuration considered
in Chapter 5 provides on-line encryption facility.

7.2   Choice of DES Mode

Theoretically any of the DES modes previously discussed
can be used for this application. However when a file is encrypted,
recovery from an error must be effected with ciphertext alone. If
a ciphering procedure with error propagation is used for file security,
subsequent inability to read a portion of the ciphertext because of
damage either to the physical medium or to the recorded bits, may
prevent all the following ciphertext from being deciphered. Therefore,
a self-synchronizing approach is desirable for file encryption. This
constraint therefore eliminates the use of the last two of the five
modes discussed in Chapter 5, leaving CBC, CFB and ECB modes. The
ECB mode is to be avoided as it is the least secure of the three
because of its vulnerability to the code book analysis problem. The
remaining two chaining modes are the CBC, a block cipher and the CFB,
a stream cipher. Any one of these two can be used. If stream
cipher feedback on eight bit character is used, then the maximum
speed will be one-eighth of the speed that can be achieved using
the block mode. That is, if the 8-bit CFB is used,the throughput
is very much reduced. Hence it is decided to adopt the CBC mode
for this file security program. The limited error extension property
of the CBC mode may be useful in such an application even though
complete error propagation property is not suitable. Consider for
instance, encryption of a database containing personnel records.
Suppose a figure in the salary field of the ciphertext file is
changed accidentally or deliberately, then the limited error extension
property will cause two blocks of characters to be in error when
decrypted. This would enable easy error detection.

- 130 -

On the other hand, with the CBC, the problem of padding the end of the file with pseudo-random numbers exists as it is a block cipher. Also when padding is used, an additional character called the pad count needs to be included as part of the padding characters. The pad count specifies the number of pad characters including itself which have been appended to the end of the file. This information needs to be preserved for future decipherment. Due to padding, the ciphertext will be longer than the original plaintext. This may be undesirable if the ciphertext is to replace the plaintext in some previously allocated file space. One way to avoid ciphertext expansion would be to use a stream cipher mode of operation to handle the special situation of short blocks. In this mixed mode of operation, the block cipher mode is used for ciphering standard blocks and the stream cipher mode is used for ciphering the short blocks at the end of file. Alternatively, the short blocks can be enciphered without increasing their length using the following method. To encipher the last short block of ℓ bytes ($\ell < 8$) the preceding full block of ciphertext is reenciphered and the first ℓ bytes of the result are then exclusive-ored with the plaintext short block. The preceding full block of ciphertext depends on all the preceding blocks of the file and thus is sufficiently variable. But as it is visible to the opponent, reencipherment of it provides the necessary secrecy. Thus this method provides the last short block the full strength of a standard DES encryption.

For this Apple system, as there is no stringent constraint preventing the ciphertext expansion, the padding technique has been adopted. It will be seen in the next chapter that such an approach is not possible when considering the Prestel Viewdata System and a stream cipher technique needs to be used.

7.3   Implementation

As far as the implementation is concerned, there is to be no change in the hardware of the encryption system. On the other hand, a different program has been developed for this purpose. The program can be divided into two sections. The first section performs encryption and decryption of files stored locally in the Apple disk system. The files can be APPLESOFT or INTEGER BASIC files. The second section invokes some of the routines developed in the point-to-

– 131 –

point communication system to transfer the encrypted files to a
remote Apple terminal where they can be automatically stored onto
floppy disks. The flowchart of the first section of the program
together with the listing is given in Appendix 8. The two sections
of the program are now very briefly described.

The program initially asks the user to enter the name of
the file to be encrypted or decrypted. It then fetches the file
from the local Apple disk system automatically and stores the file
in a prespecified part of the memory in the Apple microcomputer system.
The APPLESOFT files are stored starting from memory locations
0800 (hex) upwards whereas the INTEGER files are stored from 9600
(hex) downwards. These addresses are referred to as 'start-of-file'
addresses. Each Basic instruction stored in machine code consists
of a two-byte next instruction pointer, a two-byte instruction number,
a sequence of bytes representing the original source line of
instructions and a byte containing the 'end-of-file' marker. The
Apple system also provides an 'end-of-file' pointer. Briefly, the
encryption and decryption program is described as follows. In
encryption, the plain file from the start-of-file address to end-of-
file address forms the input to the program. Then the file is
divided into blocks of 64-bits and encrypted under cipher block
chaining mode. Padding of the last block of the file is done with
random characters in the usual way. The encrypted file is then
stored back into the same memory locations writing on top of the
plain file in the Apple system memory. Then an automatic transfer
of the encrypted file from the system memory to a floppy disk is
performed under the filename provided by the user. The encrypted
file can be loaded back from the disk at a later time and decrypted
to give the original file provided the same key and initialization
vectors are used. The decryption program requires the 'end-of-file"
address to be able to stop the decipherment process. This in turn
implies that the 'end-of-file' address must be stored along with the
cipher file during encryption. One can store this end-of-file
address either at the end of the cipher file or at the head of the
cipher file. If the address is stored at the end of the cipher file,
the decryption program will be unable to find it as the end of file
depends on the length of the cipher file which varies. The decryption
program cannot distinguish between the actual ciphertext and the
information containing the 'end-of-file' address; so it is stored

– 132 –

Page 132

at the head of the cipher file in plain format. In addition, the
count of the number of random characters padded to the end of file,
pad count, is also stored in a similar fashion. Having obtained
the 'end-of-file' address and the pad count, the decryption program
can find the initialization vector stored at the end of the file.
Alternatively, the user may be asked to enter the initialization
vector along with the key at the beginning of the program. The
decryption program then deciphers the cipher file using CBC in the
normal way discarding the dummy random characters at the end.

The second section uses modified versions of the Transmission
routine and Receive routine (Section 5.2.2) to transfer files between
two users in a point-to-point system. The user who wishes to
transmit a file initially sends some plaintext to the receiving end
using the terminal keyboard which contains information about his
identity, the identity of the intended receiver, the type of file
(APPLESOFT or INTEGER), the time at which it is sent etc. The
cipher file is then sent over the communication link using the
Transmission routine. The receiving end fetches the file and stores
it onto a floppy disk automatically under the file name provided
by the sender. The intended receiver can then decrypt the file at
a later time in an off-line manner.

- 133 -

CHAPTER 8

SECURITY IN PRESTEL VIEWDATA SYSTEM: SYSTEM SOFTWARE (3)

8.1   General

As the need and the common use of large data bases to store sensitive information increases, the requirement to maintain secrecy becomes more and more important. The Apple microcomputer system together with the designed encryption unit is interfaced to the Prestel network, the British Telecom Viewdata System, thus allowing the Apple to act as an intelligent viewdata terminal. This enables transfer and storage of encrypted as well as plain data between Apple and Prestel computer.

8.2   Brief Review of Prestel Viewdata System

The Prestel system consists of a network of GEC 4082 computers linked together by high speed data links. There are two types of computer centres namely the Information Retrieval Centres (IRCs) and Update Centres (UDCs). Currently the network consists of one UDC linked to a number of IRCs.

The basic unit of information on Prestel is a frame which consists up to a maximum of 960 characters. One or more frames are linked together to form a page. These pages of information form the Prestel database. Each page is uniquely identified by a number of up to 9 digits. Frames are further identified by letters of the alphabet a to z. Frames and pages are linked together by means of pointers and they form a tree structure. Detailed information on Prestel system can be found in [31].

8.3   Encryption/Decryption in Prestel System

As the basic unit of information is a frame, a natural choice would therefore be to encipher a complete frame at a time. However, there may be instances where encipherment of sections of a frame may be required. So in our system, we should be able to encrypt parts of a frame. At the start of each frame, it is to be indicated whether encipherment has been used.

– 134 –

As discussed in Chapter 7, the two modes of DES which are suitable for this data base application are the cipher block chaining (CBC) and the stream cipher feedback (CFB). Since it is required to encrypt parts of a page which may be small pieces of data such as individual characters, the stream cipher feedback appears to be more suitable. Further if the CBC mode is used, when parts of a frame are encrypted, this is likely to require padding for the encryption portion. This in turn will result in cryptogram extension and pose a problem when storing the enciphered frame on the Prestel data base as each frame is limited to a maximum of 960 characters. This constraint leads us to consider the use of the CFB mode in this application. As the backward channel, that is, from the user to the Prestel computer, has a speed of only 75 bits per second the reduction in speed resulting from the use of the CFB mode does not affect the throughput of the system.

The data format of each character transferred to the Prestel computer consists of 7 data bits. For transmission down the line, these 7 data bits are sent in an asynchronous start-stop format comprising 1 start bit, 1 stop bit, 1 even parity bit and 7 data bits. If a block cipher mode such as the CBC mode is used then it is required to transmit 64-bits of ciphertext in the above 10-bit format. One way to do this is to break the block into nine seven-bit groups and a single bit group. The nine 7-bit groups can be transmitted in the normal fashion. The last bit can be grouped with the next block. This needs to be done each time a block is enciphered and this process continues until the end of the frame or page. Alternatively, the last bit can be padded with 6 other bits to form an extra character. But this results in an extra character for every block encrypted and causes problems in storage of enciphered frame as mentioned earlier. Further since all the 64-bits of a ciphertext block are required to decipher correctly, the last bit must be received before decryption can begin. In account of this, the simple approach of stream cipher feedback mode has been adopted.

The encrypted information passes through the Prestel computer control unit which rejects any of the control characters present in the ciphertext. Referring to the coding table given in Figure 8.1, the codes belonging to the columns 1 and 0 are not accepted by Prestel computer as data. Therefore there is a need to prevent the occurrence of these control characters in the ciphertext. That is, the encryption system is to be made transparent to Prestel control

- 135 -

Fig. 8.1 - Viewdata Transmission Codes

Row	b4 b3 b2 b1	Col 0 (000)	Col 1 (001)	Col 2 (010)	Col 3 (011)	3b	Col 4 (100)	4b	Col 5 (101)	5b	Col 6 (110)	Col 7 (111)
0	0 0 0 0	NUL		Sp	0	Set Verify Mode(1)	@		P		—	p
1	0 0 0 1		Cursor On DC1	!	1	Set Verify (2)	A	Alpha^n Red	Q	Graphics Red	a	q
2	0 0 1 0	STX	DC2	"	2	Set Verify (3)	B	Alpha^n Green	R	Graphics Green	b	r
3	0 0 1 1	ETX	DC3	£	3	Set Programme Mode	C	Alpha^n Yellow	S	Graphics Yellow	c	s
4	0 1 0 0	ENQ	Cursor Off DC4	$	4	Tape/Pause on Playback	D	Alpha^n Blue	T	Graphics Blue	d	t
5	0 1 0 1	ACK	NAK	%	5	Tape Start	E	Alpha^n Magenta	U	Graphics Magenta	e	u
6	0 1 1 0			&	6	Tape Stop	F	Alpha^n Cyan	V	Graphics Cyan	f	v
7	0 1 1 1		CAN	'	7		G	Alpha^n White	W	Graphics White	g	w
8	1 0 0 0	Cursor BS		(	8		H	Flash	X	Conceal Display	h	x
9	1 0 0 1	Cursor HT	ESC	)	9		I	Steady	Y	Contig Graphics	i	y
10	1 0 1 0	Cursor LF	SS2	*	:		J		Z	Separated Graphics	j	z
11	1 0 1 1	Cursor VT	SS3	+	;		K		↓		k	¼
12	1 1 0 0	Cursor Home & Clear FF	Cursor Home RS	,	<		L	Normal Height	½	Black Background	l	‖
13	1 1 0 1	Cursor CR		-	=		M	Double Height	↑	New Background	m	¾
14	1 1 1 0	SO		.	>		N		←	Hold Graphics	n	÷
15	1 1 1 1	SI		/	?		O		#	Release Graphics	o	█

unit. A simple way of achieving this is to use a 6-bit cipher feed-
back technique. With this technique it is always possible to ensure
that the ciphertext belongs to the set of accepted codes. But this
allows only 64 different possible characters that can be enciphered.
In this design, these 64 input codes are 0-9, A-Z, a-z, space and
period. All other codes are transparent and bypass encryption. Thus
the output codes are reformed into the same range as the inputs thus
preserving the one to one relationship between transmission and
reception. As we are mainly interested in enciphering alphanumerical
characters present in the text, the above set of input codes is found
to be adequate for our purpose.

This can be extended to 96 codes (32 out of the possible
128 codes being control codes) using the 'breaking-up' technique
mentioned above. First consider the case where the plaintext
(ciphertext) characters are 8-bits long. In this case, the encryption
process can either be in block cipher mode or in stream cipher mode.
The cipher is first broken into 6-bit groups and then each 6-bit
group is expanded to form a 7-bit character by adding a '1' in the
most significant position, on transmission. This process removes any
unwanted control codes from the transmitted ciphertext character. This
is shown below
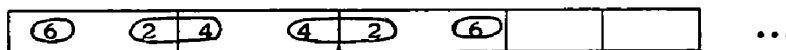
$\bigcirc$ - denotes the 6-bit ciphertext character which is expanded to
7-bits by adding a '1' in the most significant position on transmission

$\square$ - denotes the original 8-bit ciphertext character

Thus it is seen that to transmit 3 bytes of information, one needs to
send 4 encrypted 7-bit characters. If the length of the plaintext
to be enciphered is n-bytes long then this method will result in
$\lceil \frac{8n}{6} \rceil$ + 1 or $\frac{8n}{6}$ 7-bit characters depending $8n \not\equiv o \pmod{6}$ or
$8n \equiv o \pmod{6}$ where $\lceil \frac{8n}{6} \rceil$ indicates the largest integer less than
$\frac{8n}{6}$.

Now consider the case where the plaintext characters are
7-bits long. In this case, as mentioned earlier, stream cipher feed-
back mode seems to be more suitable than block cipher mode. To allow
encryption of all 96 codes, again the breaking up technique can be

- 137 -

used. In this case,cipher is first broken into groups of 6-bits and then each 6-bit group is expanded back to 7 bits by adding a '1' in the most significant position, on transmission. This process ensures that the transmitted character is in the normal ASCII code range and is acceptable to any host computer.

| ⑥ | ① | 5 | ② | 4 | ③ | 3 | ④ | 2 | ⑤ | 1 | ⑥ .. |
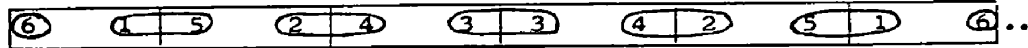
Thus if the plaintext to be enciphered consists of n 7-bit characters then this method will result in $\lceil \frac{7n}{6} \rceil$ + 1 or $\frac{7n}{6}$ characters on transmission depending on $7n \not\equiv 0 \pmod{6}$ or $7n \equiv 0 \pmod{6}$.

Let us now briefly consider the connection protocol involved when using the Apple encryption system with the Prestel computer. In the usual way, the system is connected to the public switched telephone network via the modem. A call to the Prestel computer is initiated using the telephone connected to the modem. The Prestel computer responds by sending a continuous tone of high frequency. At this point, the DATA switch on the telephone is pressed thus allowing the modem to get control of the line. That is, the modem aknowledges by sending a low frequency tone to the Prestel computer. Now the terminal is ready for data transfer.

The system software essentially carries out two distinctive tasks. Firstly, it emulates the Prestel terminal keyboard using Apple keyboard. That is, for the system to make use of some extra facilities provided by Prestel, the Apple keyboard is effectively extended to include some special characters. Secondly it incorporates encryption-decryption facility into the Prestel system. The software is lengthy and complicated. Hence only some important aspects are briefly considered here.

Prestel system can be used in two modes, namely, the user mode and the information provider (IP) mode.

In the user mode, two main facilities provided by the system are:

(a)   Reception of plain and encrypted frames from the Prestel database.

(b)   Transmission of commands such as choosing a frame etc from the Apple keyboard to the Prestel computer. Note that in this case, the only keys used are 0-9, # and *.

- 138 -

In the editing mode, in addition to (a) and (b), facilities
are provided for entering, amending, copying and deleting plain or
encrypted frames of Prestel. The Prestel editing terminal keyboard
is given in Appendix 9. It is seen that this keyboard has additional
facilities compared to the Apple keyboard which are required to
provide necessary control signals. Two important ones among these
are: Start Edit and End Edit. In addition,special functions for
encryption and decryption are required.

Having entered the secret DES key in the normal fashion, at
the beginning of the communication, in the user mode, the user must
have the choice as to when to set the interface into the decryption
mode. This enables him to decipher only those pages which are in
enciphered form and to read the other Prestel pages in plain form.
This is carried out by pressing the key CNTRL-Q. Now if any of the
enciphered pages is read from the Prestel database, it is displayed
on the terminal in plain form. Only the user with the right key and
the correct initialization vector will be able to obtain the complete
original plain frame. The interface unit is set back to normal plain
mode by pressing the key CNTRL-R. This software implementation allows
changes in initialization vector during communication whereas to
change the DES key, the system needs to be reset and restarted again.
This has been done because every user is expected to have a single
secret key although he may use any number of different initialization
vectors. This is particularly important when a user needs to encrypt
same portions of text in different frames. Changing the initialization
vector allows different ciphertext representations of the same
plaintext under a fixed key. As it stands, the user needs to keep
a record of the frame number. together with the initialization vector
he used to encrypt that frame and his single secret DES key. An
improved scheme would be to generate a pseudo-random key, called the
frame key, dynamically and encrypt the frame using this key. The
frame key can then be enciphered under the user's secret key using
ECB mode and stored at the head of the frame. The initialization
vector is again generated using a pseudo-random process and can be
enciphered under the frame key using ECB mode and is also stored at
the top of the frame. Using this method one has effectively chained
the frame key and the initialization vector used in the encryption of
a frame. The decryption process can automatically recover the
enciphered frame key from the top of the frame and then decipher it

- 139 -

Page 139

using the user's secret key under ECB mode to produce the frame key.
The frame key is now used to decipher the next 8 bytes at the head
of the frame under ECB mode to produce the initialization vector.
The initialization vector and the frame key can now be used to decipher
the frame under the CFB mode. This method would allow different
frames and different pages to be enciphered under different keys
without having to reset the system. Further, the user does not need
to keep a record of each frame key and the corresponding initial-
ization vector used in the encipherment of that frame. This is being
done automatically. More on such key management aspects will be
considered in Chapter 9.

In the editing mode, the user is able to enter and amend
the encrypted as well as plain frames in Prestel. From the user
point of view, it is essential that the operations that need to be
done for encryption and decryption must be as simple as possible.
Start (?) and stop (/) markers are used to indicate the beginning
and end of enciphered data in the frame. The key CNTRL-A is pressed
to set the interface unit to encryption state. All subsequent
characters typed are automatically encrypted under the 6-bit CFB
mode. The key CNTRL-B is used to return the interface unit to the
plain mode. This allows encryption of even single bytes of data.
The system initially produces upper case letters. Lower case letters
are obtained by pressing the key CNTRL-V. All characters typed
are now in lower case until the upper case shift, CNTRL-W is typed.
Start Edit and End Edit needed to work the Prestel Editor System
are obtained using the keys CNTRL-T and CNTRL-E respectively. Most
of the cursor control movements such as backspace, forward, downward,
upward are included in the editor facilities. The Return key behaves
slightly differently compared to the normal Apple mode in that the
cursor returns to the beginning of the same line. So to move to the
beginning of the following line, one needs to press the Return
followed by Line feed (CNTRL-J). Note that the graphic and colour
keys are not included in this Prestel encryption system.

This software implementation provides on-line editing/user
facilities on the Prestel Viewdata system. It is possible to merge
this program and the one discussed in Chapter 7 to perform local
off-line editing and encryption of a Prestel frame and then transfer
the created Prestel frame to the Prestel computer.

- 140 -

An example of a completely encrypted frame and a partly encrypted frame together with the corresponding plain frame is shown in Figures 8.2, 8.3 and 8.4.

. . HERE IS A GAME THAT CAN BE PROGRAMMED
FOR PLAY ON A DIGITAL COMPUTER . . .
    A polyomino is a figure formed by joini
ng unit squares along their edges. Pentom
inoes are 5 square polyominoes and it is
possible to construct 12 different pentominoes.
A pentomino game is played by ar
ranging the 12 pentominoes into various
size rectangular boxes . . . 3 by 20 or 4 b
y 15 or 5 by 12 or 6 by 10. Computers hav
e been used to generate many solutions. A
computer program produced two solutions
for 3 by 20 configuration and 2339 for
the most popular size 6 by 10 rectangula
r configuration.

Fig. 8.2 - Plain PRESTEL Page

- 141 -

Scratchpad   651314b   Op

?OnLih^{L}cKwGbZHjxea JgVwBCLJ BVz bbvDW Ev
 o' y w PFZQsNNaBK txvOp hV TBB@xUmJ' K
D YnvBISJ YwQKFtrTmoBCLMv okHXebhN GXqq
'KZdvI erePmTQax@ kyKmC Rcfi'F 'JiNaXt
@nQbwYg xkTRLsODN IuCvSzLZvXNZb WNpYPB
X HdIEzgOnPtm jDGtOaUVIG OdMF OgiYmdqRYA
sdTdriHkue jTd QQs CnxkbuVgziKiokxbKtpdj
juwOC 'G nH yMIg@MvI dhae RHX'DbQHo 'yF
xx@y Kz S UZg amqmVB'EfozyFjUIPQFcQF TR
JkzFtFA AVsoGpciLp!Wewo wkBPJ JyiDHEz Z
uLrwEu oRPgWzOLDIWMr ^{L} Rkqs f YRAXZXS!V
 PR CS2CIVUet DtPC^{L}c MhTEyQByjfALbp rq^L
G Ip psp/yee B@It! Yxo abb TxftGQpWgjLZ
yyaLmfEJvrOLzfSBwXsf ByiX wFvg h bH Art
vbuoraZdf'xfvG!xS HRztdsp!gNuaDQ BsIyuK/

? indicates start of encryption.
/ indicates end of encryption.

KEY 3131313131313131
IV   0000000000000000

MODE: 6-bit CFB

Fig. 8.3 - Completely Encrypted PRESTEL page

Scratchpad   651314 e   Op

. .   HERE IS A ?@FJI/ THAT CAN BE PROGRAMMED
FOR ? KWt/ ON A DIGITAL COMPUTER . . .
   A ?gVDy   NEG/ is a figure formed by joini
ng unit squares along their edges. ?1 Yhab
@aZ / are 5 square ?DtNxLC kIP / and it is
possible to construct 12 different ?jXgl Pa pUzA/
A ?KS   MhiD/ game is played by ar
ranging the 12 ? Smw@YzLZc/ into various
size rectangular boxes . . . 3 by 20 4 b
y 15 or 5 by 12 or 6 by 10. Computers hav
e been used to generate many solutions. A
computer program produced ? jF/ solutions
for ?C KoLgq/ configuration and ?huYv/ for
the most popular size ?bScLSto/ rectangula
r configuration.

? indicates start of encryption.
/ indicates end of encryption.

         ,
KEY : 3131313131313131
IV : 0000000000000000

MODE: 6-bit CFB

   Fig. 8.4 - Partly Encrypted PRESTEL page

- 143 -

C H A P T E R   9

KEY DISTRIBUTION AND PUBLIC KEY CRYPTOGRAPHY

9.1 General

Until now, the use of DES cryptographic algorithm in
protecting the data during transfer between users has been considered.
However the security of the DES depends on the secrecy of its keys.
Thus protecting the data depends on protecting the keys because they
are the means by which the data can be decrypted. Any key controlled
cryptographic algorithm thus requires a protocol for safely handling
and controlling its cryptographic keys. Keys must be produced and
distributed not once but constantly. In some systems they must be
changed with the passage of time, or with the amount of traffic and
in all systems, they must be changed when they are feared compromised.
Frequent key changes limit the amount of data compromised if an
opponent does learn a key. Keys must be provided to new users of
the system and old keys must be retired as users withdraw. The
consideration of all these aspects forms the subject of key management.

There are essentially three ways to incorporate cryptography
into a communication system namely link-by-link, node-by-node and
end-to-end encryption [3].

In link-by-link encryption, data is encrypted across the
medium connecting two directly communicating nodes. Link-by-link
encryption is independent of the system and does not necessarily
imply that the cryptographic capability is integrated into the
communicating nodes. It may be regarded as being implemented by a
pair of cryptographic devices bracketing the line between two
communicating nodes and situated between the nodes and their modems
as shown in Figure 9.1.

Node-by-node encryption is similar to link-by-link encryption
in that each link is protected by a unique key. However data passing
through an intermediate node are not in the clear as would be the
case with link encryption. Rather at an intermediate node, the
enciphered data are transformed from encipherment under one key to
encipherment under another key (that is, deciphered and reenciphered)
within a security module which may be a peripheral device attached
to the node. That is, the plaintext occurs only within the security

– 144 –
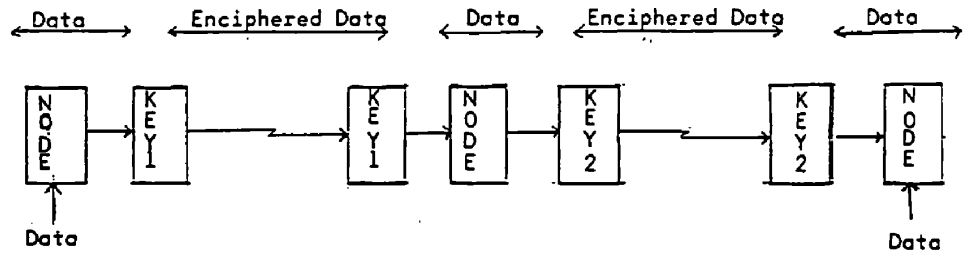
Fig. 9.1 Link Encryption
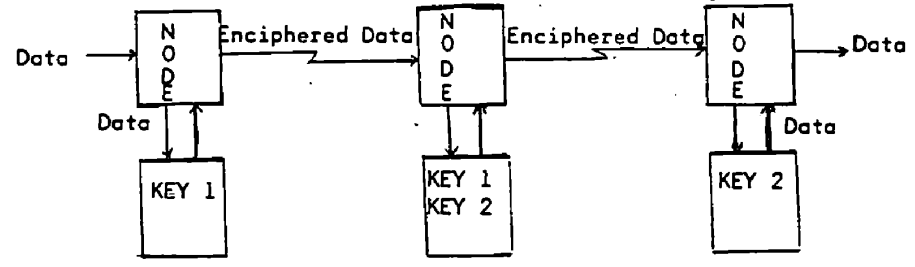
Fig. 9.2 Node Encryption
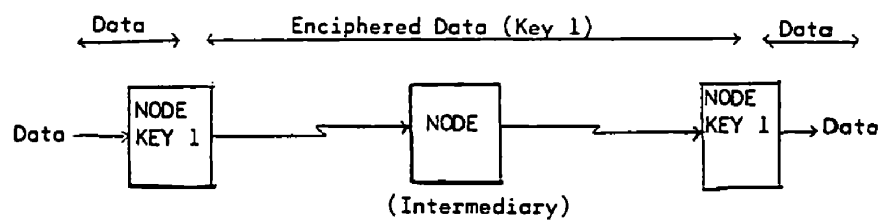
Fig. 9.3 - End Encryption

module and not within the node (Figure 9.2).

In end-to-end encryption, data encrypted at the originating node is not decrypted until it arrives at its final destination. Thus this method continuously protects data during transmission between users. Unlike link and node encryption, end-to-end encryption allows each user to have several keys, one key for each user who uses encryption (Figure 9.3)

It appears that in terms of security, cost and flexibility, end-to-end encryption seems to be the most attractive for systems requiring many protected links [3] .

The Apple encryption system discussed earlier is a simple end-to-end encryption system. More exactly, it can be referred to as a private end-to-end cryptographic system as the user needs to request for cryptography and its use is not transparent.

Some key management schemes which allow the DES interface unit to be integrated into data processing systems to provide protection for communications between individual users in an end-to-end encryption network are discussed.

9.2   Key Management Using Key Centre

This approach uses a Key Centre (KC) which acts as a source of session keys for encrypted calls using the DES algorithm. A detailed description of the functioning of such a centre is given in [3, 32].

Key centre can be operated manually in which the keys are sent by mail or couriers. If such an arrangement is trusted, that is, whether the risk of untoward disclosure either accidently or as a result of deliberate attempts is acceptable, it could work out very well. At least, this may be possible when the network is small and traffic volume is low. On the other hand, if the network handles large traffic volumes,the need to change the keys often demands that large amounts of keys to be distributed. In large networks, the number of possible interconnections grows as $n(n-1)/2$ where n is the number of users. This may become an expensive venture because the manual systems have to be guarded against security leaks by conventional methods and the persons involved have to be trustworthy.

In an automatic KC, data network is used to distribute and generate the keys automatically. Consider the connection protocols

involved when two users wish to communicate in a secure fashion in a single Key Centre environment.

Essentially two types of keys namely data encrypting keys and key encrypting keys can be identified. The data encrypting key is active only for a duration of a single communication session and therefore is referred to as a session key (KS). The session key is protected by enciphering it under key encrypting key which varies from user to user. Therefore the Key Centre is required to store one key encrypting key for each user. These keys are themselves stored within the centre in enciphered form using the Centre's master key (KCM). Hence the problem of providing secrecy for cipher keys is reduced to providing secrecy for only one key namely the master key. This type of approach is referred to as the master key concept [3]. It is assumed that the master key is stored in some non volatile storage in an inaccessible area in the Centre referred to as the cryptographic facility so that it need be loaded into the cryptographic facility only once. Furthermore, each user is required to store only his user key (KU).

9.3   Communication Security

Let KS_1, KS_2, ..., KS_n represent the time variant, dynamically changing data encrypting keys used for enciphering and deciphering data. It is assumed that KS is operational for the duration of a communications session. Let KCM represent the master key of the Centre and KU represent the user (or terminal) key.

To begin with, the user i requests the Centre KC for a session key (KS) to communicate with user j. The request is accompanied with a verifiable identification of the user i. The whole message is enciphered under the user i's key KU_i. That is,

$$i \rightarrow KC : i, (i, j, r_o)_{KU_i}$$

where r_o is a random number chosen by user i. It is used to prevent an intruder impersonating the KC by replaying some previously recorded reply containing an earlier session key which the intruder would like the user i to use again. Upon receiving the request the Centre fetches the user i's key KU_i which is stored in its memory under the master key KCM. Then it deciphers the request and checks against its stored information to see if the request is legitimate

- 147 -

and if it is, it issues the session key (KS) to the user i. The session key KS is generated within the Centre using a pseudo-random procedure. The reply from the Centre to the user i is given by

$$KC \rightarrow i \; : \; (KS, r_o, (KS, i)_{KU_j})_{KU_i}$$

The random number r_o is returned by the KC for the user i to verify that the reply is coming from the KC and not from an intruder. Further as the session key KS is encrypted under KU_i, it allows only the user i to decrypt and obtain KS and not any intruder. The session key together with the identification of user i encrypted under KU_j is also sent to user i. The user i cannot decrypt this portion of the reply as he does not possess KU_j. The user i then sends this cipher portion to user j, that is,

$$i \rightarrow j \; : \; (KS, i)_{KU_j}$$

The user j responds by sending a random number r_1 to user i, encrypted under the session key KS

$$j \rightarrow i \; : \; (r_1, j)_{KS}$$

The user j does this to ensure that it is indeed user i who is requesting the call and not any intruder using parameters of a previous call. The user i then checks j's identity and modifies the random number r_1 in some prearranged fashion to result in r_2, which he returns to user j under KS

$$i \rightarrow j \; : \; (r_2)_{KS}$$

Now the users i and j can be almost certain that they are talking to each other and can communicate with each other in a secure manner using KS as the DES secret key. Most of the above steps can be made transparent to the users in the network.

A variation of the above method consists of the Key Centre KC sending the session key directly to the user j instead of sending

- 148 -

it via user i. That is, the distinction lies in the path taken by
the session key from the Centre to user j. With this approach, two
possibilities may occur - either the session key has already arrived
at user j when the latter receives the call request or it has not yet
arrived. In the latter case, one must ensure that an old key is not
used mistakenly. Further the case where user j needs to wait for the
session key increases the complexity of the connection protocols.
Thus the above outlined method where the session key arrives to user j
via user i seems to have some advantage over the other method.

To further improve the integrity of the conversation and
reduce the problem of impersonation, timestamps, T, [33] can be added
to the key distribution protocol. The first three steps of the above
procedure are then modified to become:

$$i \rightarrow KC \quad : \quad i, (i, j, r_o, T)_{KU_i}$$

$$KC \rightarrow i \quad : \quad (KS, r_o, (KS, i, T)_{KU_j}, T)_{KU_i}$$

$$i \rightarrow j \quad : \quad (KS, i, T)_{KU_j}$$

The users i and j can then verify that their messages are not replays
by checking $|clock - T| < \Delta t$ where clock gives the local time, Δt
gives some time error which includes the network delay time and the
time discrepancy between the sender's clock and the local clock.
This requires some form of time synchronization among the users of
the network.

With the above schemes, it is seen that if the session key
is somehow lost within the user's system, then a fresh call is to be
made by the user i to the KC to establish a new session key. It is
preferable that the KC generates a new session key even when the
user i did not actually use the old KS for any conversation. If on
the other hand, the KC does keep a record of session keys issued to
different users over a small period of time (say one day) then these
keys need to be stored in enciphered form within the Centre. Rather
than using the same master key for this purpose, it is advisable to
use another master key KCM1 to encrypt these temporarily stored
session keys.

- 149 -

9.4   File Security

Let us now consider a key management scheme for file security where one wishes to protect the stored data in the same way as the communicated data [3]. It is assumed that the encrypted files are to be stored in a database in the host processor (HP) in the network. (This could be for instance the Key Centre mentioned earlier). It is also assumed that the users in the network have distinct secret keys KU which are also stored in the host processor in enciphered form under the master key KCM. Consider the case where the user i wishes to store a file in encrypted form in the database under the name CIPHERFILE. Let the corresponding file in clear form be PLAINFILE. To begin with the procedure followed is very similar to the one outlined for communication security given earlier. A call is made to the host processor

$$i \rightarrow HP \ : \ i, \ (i, \ r_o)_{KU_i}$$

where r_o is a random number chosen by user i.

The host processor responds by generating a file key (KF) using a pseudo-random process and encrypting it under the user i's key KU_i. This is then sent to user i

$$HP \rightarrow i \ : \ (KF, \ r_o)_{KU_i}$$

The user i decrypts the message to obtain KF and verifies the random number r_o to ensure that the reply is coming from HP and not from an intruder. Then the user can encrypt his PLAINFILE using KF as the DES secret key to produce CIPHERFILE. This is then transmitted to the host processor to be stored under the same name. To be able to recover the PLAINFILE, it is necessary for the host processor or the user i to record the information that the file has been encrypted under KF. In a large system with a number of users and with each user having a number of files, it may not be a good idea for the Centre to keep a separate file containing the name of the data file and the corresponding file key. (If this is done, then this separate file needs to be enciphered under some master key). A better arrangement would be to store the information at the header of the file itself. The file key KF can be stored in the header in encrypted form under a master key KCM2. (KCM2 rather than KCM is chosen to achieve separation from communication security). With this method if a user r wishes to

– 150 –

decipher the CIPHERFILE, he requests the host processor for the file key KF.

$$r \rightarrow HP \; : \; r,(CIPHERFILE)$$

The host processor reads the header of CIPHERFILE, decrypts it using KCM2 to recover KF. This is then reenciphered under KU_r and transmitted to user r. This procedure does not allow the host processor to differentiate between users. For instance, the user i may wish that PLAINFILE not be available to user r. This can be achieved if in addition to KF, the identification of the owner of the CIPHERFILE is recorded on the file header, that is, the header contains the information $[i,(KF)_{KCM2}]$. Further, the host processor is required to maintain a record of which users are allowed by user i to obtain the PLAINFILE. Then if user r requests the host processor for the file key of CIPHERFILE, the host processor first reads the header to find the owner of the CIPHERFILE. Having found the owner, i, it checks whether the requesting user belongs to the group of users who are allowed to read PLAINFILE. If user r belongs to this group, it recovers the file key KF from the header, encrypts it under KU_r and sends it to user r. If user r does not belong to this group, the above step will not be carried out by the host processor and access to the file key is prohibited.

9.5   Key Distribution for Groups of Users

Consider a more general case where a user in the network wishes to broadcast a message to several users [34]. Assume that a group G is a non-empty subset of n users and members of G wish to broadcast and receive messages from other members of G and to access and update files private to G. A given user may be a member of as many as 2^{n-1} groups and there are at most 2^n-1 non-empty groups in the system. Again it is assumed that all aspects of key distribution for any given group is managed by a single Key Centre. One method of key distribution among the group of users is considered in this section and three other methods employing public key concept are described in Section 9.7.

– 151 –

9.5.1 Method 1

In this method, the Key Centre is assumed to keep a list of personal keys of all users. In addition, the Key Centre also keeps a record of all group keys for the groups it manages.

To establish a group G, a member i of G registers the group with the Key Centre. The Centre returns a group identifier IG to the user i who then distributes to the members of the group G using the method described earlier. The KC also generates a group key KG and creates a record identified by IG that contains KG and the users who belong to G. This record itself is stored in enciphered form under a master key.

Whenever a user j belonging to G wishes to communicate with other users or store a file to be read by other users, he obtains the group key KG from the Centre. The key distribution protocol can be described as follows:

$$j \rightarrow KC \;:\; j, \; IG$$

The user j sends to KC his identification and his group identification and requests for the group key KG from KC. KC fetches the group record identified by IG, checks whether j is a member of the group and returns KG to user j enciphered under j's personal key.

$$KC \rightarrow j \;:\; (IG, \; KG, \; T)_{KU_j}$$

where T is a timestamp used to protect against replay of previous keys. Because the group key KG is enciphered under user j's personal key it is not possible for an intruder either to intercept KG or to impersonate j and acquire a group key for a group to which he does not belong. User j can now use the group key KG to encrypt a file to be read by other users of the group or to decrypt an encrypted file created by any other user of the group.

An user i of the group can obtain the group key KG from the Centre in a similar fashion and hence the users i and j can communicate with each other in a secure manner.

- 152 -

The primary disadvantage of this approach is the storage requirements for the group keys. KC may need to store up to $2^n - 1$ group keys. Secondly there is no identification between group members and hence no discrimination between group members.

The idea of one Key Centre in the above schemes can be extended to many such Centres and a group of m users 'belonging' to each Centre. In such a situation, each Key Centre is required to possess a shared secret key with each of the other Key Centres. (It could be two such keys, for instance, one for communication security and the other for file security). If there are n such Centres, each Centre therefore has n-1 (or 2n-2) such keys. Then for instance, if a user i belonging to Centre I wishes to communicate with a user j belonging to Centre J, then the Centre I generates the session key and sends it to user i in the usual fashion. As the Centre I does not know the user j's secret key, it sends the session key to Centre J enciphered under the shared communication secret key between Centres I and J. The Centre J recovers the session key and reenciphers it under the user j's key KU_j and sends it to user j. A similar procedure can be envisaged when a user i belonging to Centre I wishes to read a file of user j belonging to Centre J.

The problem of key distribution can also be overcome using public key cryptography concept proposed by Diffie and Hellman [35]. In the next section, the underlying principles of public key cryptography are considered to see how such systems can be used to solve the key distribution problem in an elegant way.

9.6   Public Key Systems

Public key systems allow two users to communicate securely over an insecure channel without any prearrangement. Cryptosystems which allow this type of communication are asymmetric (see Section 2.2) in the sense that the sender and receiver have different keys

- 153 -

at least one of which is computationally infeasible to derive from the other. These systems separate enciphering and deciphering capabilities and privacy is achieved without keeping the enciphering key secret because it is no longer used for deciphering. Hence the enciphering key is published in addition to the enciphering and deciphering algorithms without compromising the security of the system. The concept of a public key cryptosystem is shown in Figure 9.4. User i encrypts the message M using the public enciphering key of user j and sends the cipher to user j over an insecure channel. Only the user j will be able to decrypt the cipher to recover M as he is the only one who knows his secret deciphering key. The encryption (E) and decryption (D) algorithms in such a system have the following properties:

(a)   Deciphering the enciphered form of a message M yields M, that is, $D(E(M)) = M$.

(b)   Both E and D are easy to compute.

(c)   By publicly revealing E, the user does not reveal an easy way to compute D. This means that only the receiver (designer) can decrypt messages encrypted with E or compute D efficiently.

(d)   If a message M is first deciphered and then enciphered, then M is the result, that is $E(D(M)) = M$.

The property (d) is not necessary for a public key cryptosystem but if it is obeyed then it is possible to obtain the digital signature feature (see Section 2.2) [34].

The public key concept gives rise to a new class of cryptographic algorithms. One application of such algorithms is to solve the problem of key distribution in systems employing symmetric cryptosystems. The public key cryptosystems would in many instances be the ultimate solution to the key distribution problem. This can be done as follows: User i can encrypt the session key KS using the public key of user j and then send it to user j over an insecure channel. Because the deciphering key is only known to user j, he is the only one who can decrypt the cipher and obtain the session key.

- 154 -
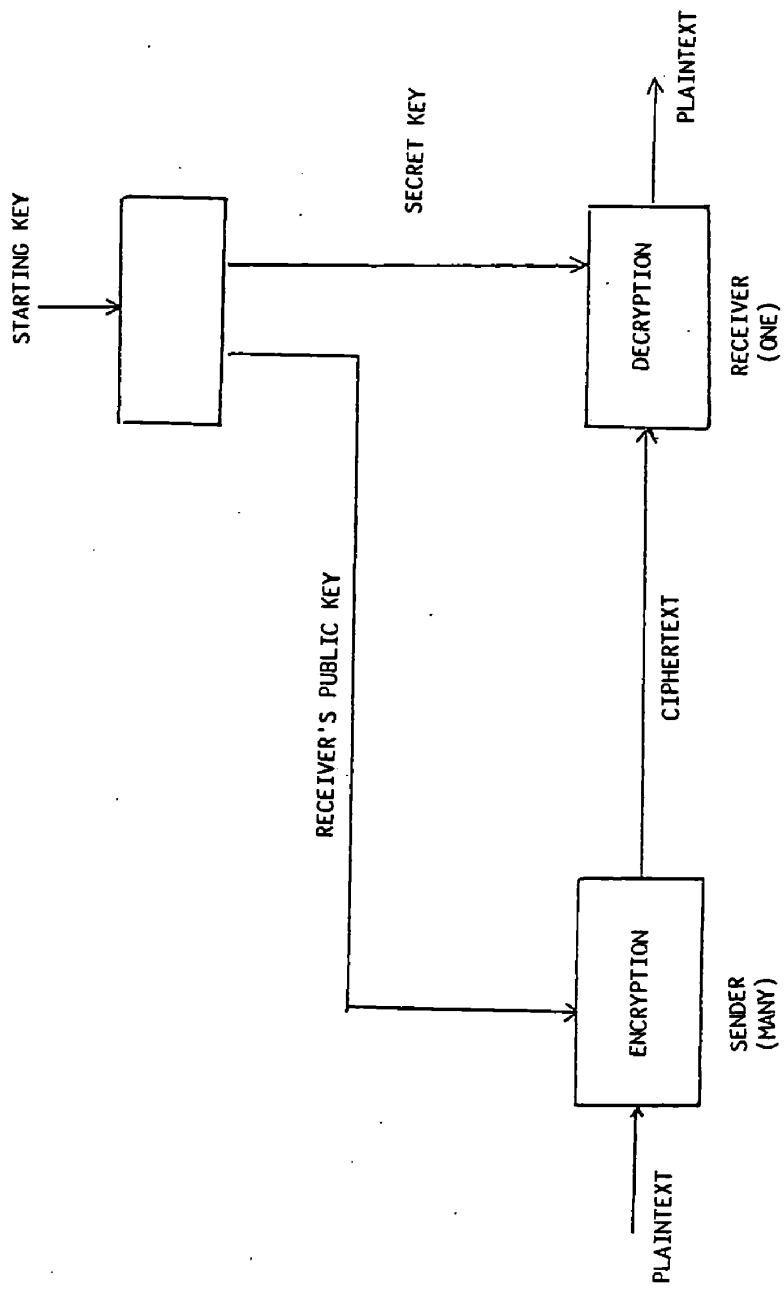
Fig. 9.4 — Public-Key Cryptosystem.

- 155 -

The users i and j can then communicate with each other using a symmetric cryptosystem such as the DES under KS. The protocols as described above pose other problems associated with the integrity of the public keys and false impersonations by an opponent. Again a trustworthy third party such as the Key Centre, KC, may be required for the maintenance of the public keys. The above set up can now be modified using a Public Key Centre (PKC) which supplies and maintains public keys of all users in the network. One can further assume that the Centre PKC has a public key (P_k) and a secret key (S_k) pair and that the key P_k is known to every user in the network.

1. $i \rightarrow$ PKC : $i, (i, j, r_o)_{P_k}$

where r_o is a random number chosen by user i. As in Section 9.3, this is used to prevent an opponent impersonating the PKC by replaying some previously recorded reply.

The PKC upon receiving the request from user i for user j's public key, encrypts the user j's public key P_j using the public key of user i, P_i, and sends it to user i along with the random number r_o (or some modified r_o using a publicly arranged function).

2.   PKC $\rightarrow i$: $(i, P_j, r_o)_{P_i}$

Note that the cipher in step 1 can only be decrypted by the Centre and no one else and the cipher in step 2 can only be decrypted by user i and no one else. A similar procedure can be followed by user j if he wishes to obtain the public key of user i. From now on, user i can communicate with user j in a secure manner, either by generating a session key KS and transmitting it to user j enciphered under P_j as mentioned above or using public key approach, that is, by encrypting the messages under the public key of the receiver j.

Note that in the set up procedure, the Centre PKC is not used for generating the session keys and it does not know the secret keys of users but is used as a distributor of public keys. Thus for the integrity to be maintained, it is crucial that the public key file be protected from unauthorized modification.

The public key concept can also be employed in a different way to the one described above to provide a solution to the key

- 156 -

distribution problem [34]. With this method, there is no need for decryption as such at the receiving end. It is not a public key cryptosystem but it is a public key distribution system. Two users wishing to exchange a key, communicate back and forth until they arrive at a key which is common. Then this common key can be used as a session key in a symmetric cryptosystem. The opponent eavesdropping on this exchange finds it computationally infeasible to compute the key from the information overheard.

Both the public key cryptosystems and the public key distribution systems are based on one-way functions of one form or another. For instance, it is said that in public key systems, it is infeasible to determine the secret key from the knowledge of public key. (Property (c)). A one-way function has the properties that

(i)   It is an easily computed function from x to y, that is, $y = f(x)$.

(ii)   It has an inverse function.

(iii)   It is computationally infeasible to discover the inverse function.

A precise definition of a one-way function therefore depends on a specific measure of complexity as it varies with time and technology. As mentioned in Section 2.3, the complexity measures are often defined in terms of time or storage required or as a time-memory product. If the number of operations to be done in computing the inverse is taken as a measure, then thermodynamics places a limit of approximately 10^{70} on the number of operations that can be performed even if the entire energy of the Sun could be harvested [36, 37]. As the legal receiver has to decrypt the cipher, the public key systems are based on 'trapdoor' one-way functions rather than one-way functions. A trapdoor one-way function is a one-way function which has the additional property that:

- it is computationally infeasible to discover what the inverse function is, unless certain specific information (trapdoor inform- ation) that is employed in the design of the function is known. A trapdoor one-way function becomes a trapdoor one-way permutation if

- 157 -

it satisfies the property that:

— decryption algorithm followed by encryption algorithm produces the original plaintext (property (d)).

In this case, the mapping between ciphertext and plaintext becomes both injective and surjective. This is essential for implementing digital signatures.

A brief review of two well known public key cryptosystems and a public key distribution system is now presented.

9.6.1 Merkle-Hellman Trapdoor Knapsack Public Key Cryptosystem [13]

The knapsack problem is a combinatorial problem in which given a vector $\underline{a}$ of n elements, it is required to select a subset of these which add up to a given sum S. The problem is to determine which a_i for i = 1 to n are to be included in forming S, that is, determining whether $x_i = o$ or $x_i = 1$ for i = 1 to n in the following equation:

$$S = \underline{a} \cdot \underline{x} = \sum_{i=1}^{n} a_i x_i \qquad (9.1)$$

It is seen that there are 2^n possible ways of selecting the quantities x_i and this exponential function increases very rapidly as n increases whereas it is easy to test whether a particular combination is a solution. But there are some instances in which the equation (9.1) is easy to solve. One such instance is when the elements $\{a_i'\}$ form a superincreasing sequence. That is,

$$a_i' > \sum_{j=1}^{i-1} a_j' \qquad \text{for all } i > 1 \qquad (9.2)$$

When this occurs, then $x_n = 1$ and only if $S \geqslant a_n'$ and similarly for i = n-1, n-2, ..., 1 $x_i = 1$ if and only if

$$S - \sum_{j=i+1}^{n} x_j \cdot a_j' \geqslant a_i'$$

Hence the legal receiver designs the system in such a way that only he can transform the hard knapsack (9.1) into an easy knapsack using the above procedure whereas the opponents are forced to solving (9.1). He does this by choosing two numbers w and m which are relatively

— 158 —

prime to each other and a superincreasing sequence a_i' for $i = 1$ to n. This vector is then transformed to form the sequence $\{a_i\}$ using w and m as follows:

$$a_i = a_i' \cdot w (\text{mod } m)$$

The vector $\underset{\sim}{a}$ is published and it forms the encrypting key of the public key system. The vector $\underset{\sim}{a}'$ and integers w and m are kept secret and form the deciphering key. The encryption procedure consists of taking a plaintext $\dot{x}$ in the form of a vector $\underset{\sim}{x} = (x_1, \ldots, x_n)$ where $x_i \in \{0, 1\}$ and forming $S = \underset{\sim}{a} \cdot \underset{\sim}{x}$. The ciphertext S is then sent over the insecure channel to the receiver. The decryption process uses w and m and $\underset{\sim}{a}'$ as follows:

$$S' = w^{-1} \cdot S \ (\text{mod } m)$$

$$= w^{-1} \cdot \sum_{i=1}^{n} x_i \cdot a_i \ (\text{mod } m)$$

$$= w^{-1} \cdot \sum_{i=1}^{n} x_i \cdot w \cdot a_i' \ (\text{mod } m)$$

$$S' = \sum_{i=1}^{n} x_i \cdot a_i' \qquad (\text{mod } m)$$

If m is chosen such that $m > \sum_{i=1}^{n} a_i'$ then

$$S' = \sum_{i=1}^{n} x_i \cdot a_i'$$

This knapsack is easily solved for $\underset{\sim}{x}$ which is also the solution to the apparently difficult trapdoor knapsack $S = \underset{\sim}{a} \cdot \underset{\sim}{x}$. It is also possible to iterate the basic transformation by generating several pairs of (w, m). For instance, rather than requiring $\underset{\sim}{a}'$ to satisfy (9.2), $\underset{\sim}{a}'$ can be transformed to a new problem $\underset{\sim}{a}''$ using

$$a_i'' = w'^{-1} \cdot a_i' \ (\text{mod } m')$$

where $\underset{\sim}{a}''$ satisfies (9.2) and is easy to solve. With each such successive transformation, the structure in the publicly known vector

– 159 –

a is made more amd more obscure.

9.6.2 Rivest-Shamir-Adleman(RSA) Public Key Cryptosystem [12]

The trapdoor in this asymmetric scheme is based on the difference in computational difficulty in finding large primes as opposed to factoring large numbers.

Briefly, the RSA system can be described as follows: The receiver chooses two large primes p and q so large that factoring m = p.q is beyond all projected computational capabilities. The plaintext message M can be chosen from the range $1 \leqslant M < m$. The ciphertext C corresponding to M is derived from the permutation

$$C \equiv M^e \pmod{m}$$

The plaintext M is retrieved from C by applying the inverse transformation

$$M \equiv C^d \pmod{m}$$

The receiver chooses e and d such that

(a) $\gcd^\dagger (e, \phi (m)) = 1$ where $\phi (m)$ is the Euler-totient function and in this case, it is equal to $(p-1)(q-1)$

(b) $e \, d \equiv 1 \pmod{\phi (m)}$ (9.3)

In other words, e and d are multiplcative inverses in the group formed by residue classes mod $\phi (m)$.

The reason why this encryption-decryption scheme works is based on the Euler-Fermat theorem [38] which states that for any integer M which is relatively prime to m,

$$M^{\phi (m)} \equiv 1 \pmod{m} \qquad (9.4)$$

Using (9.3) gives,

$$M^{ed} \equiv M^{K \phi (m) + 1} \pmod{m} \text{ for some integer K.}$$

---

† greatest common divisor – 160 –

From (9.4), for all M such that p does not divide M

$$M^{p-1} \equiv 1 \quad (\text{mod } p)$$

and since p-1 divides $\emptyset$ (m)

$$M^{K \emptyset (m) + 1} \equiv M \quad (\text{mod } p) \qquad (9.5)$$

When $M \equiv 0$ (mod p), the equation (9.5) is obeyed trivially. Similarly for q

$$M^{K \emptyset (m) + 1} \equiv M \quad (\text{mod } q) \qquad (9.6)$$

Using Chinese Remainder Theorem (Appendix 10), equations (9.5) and (9.6) imply that for all M,

$$M^{ed} \equiv M^{K \emptyset (m) + 1} \equiv M \quad (\text{mod } m)$$

In this system, the numbers e and m are made public and they form the encrypting key. The numbers d, p and q are kept secret and form the decrypting key. If m can be easily factorized to p and q then the cryptanalyst can find $\emptyset$ (m) and d and hence can crack the system.

This cryptosystem and its possible extensions form a major part of this thesis and hence this system is considered in detail in subsequent chapters.

9.6.3 Diffie-Hellman Public Key Distribution System [35]

This public key distribution system makes use of the apparent difficulty of computing logarithms over a finite (Galois) field GF(q) where q is a prime.

Let 'a' be a primitive element of GF(q) and let

$$y \equiv a^x \quad (\text{mod } q) \quad \text{for} \quad 1 \leqslant x \leqslant q - 1$$

x is referred to as the logarithm of y to the base 'a' over GF(q)

$$x = \log_a y \text{ over GF(q)}$$

– 161 –

Calculation of y from x is easy whereas computation of x from y is much more difficult, that is, it is an one-way function. This problem is called the logarithm problem whereas the RSA system is based on the root problem.

The key distribution occurs as follows. User i generates a random number x_i chosen uniformly from the set of integers 1, 2, ..., q-1. He then computes

$$y_i \equiv a^{x_i} \pmod{q}$$

and publishes y_i and keeps x_i secret. Similarly user j publishes y_j and keeps x_j secret where

$$y_j \equiv a^{x_j} \pmod{q}$$

The private session key, K_{ij}, is established by forming

$$K_{ij} \equiv a^{x_i x_j} \pmod{q}$$

User i computes K_{ij} by obtaining y_j from the public file and forming

$$K_{ij} \equiv y_j^{x_i} \pmod{q}$$

$$\equiv a^{x_i x_j} \equiv a^{x_j x_i} \pmod{q}$$

User j similarly computes K_{ij} using $y_i^{x_j} \pmod{q}$. For the opponent to form K_{ij}, he must compute

$$K_{ij} \equiv y_i^{(\log_a y_j)} \pmod{q} \equiv y_j^{(\log_a y_i)} \pmod{q}$$

Therefore if logarithms over GF(q) are easily computed, the system can be broken. A possible extension of this system together with a practical implementation are described in Chapter 13.

9.7   Key Distribution Using Public Key For Groups of Users

Let us now return to the situation where members of a group G wish to broadcast and receive messages from other members of G and to access and update files private to G and consider three other methods of key distribution for such an arrangement. Method 1 is given in Section 9.5.1.

- 162 -

9.7.1 Method 2

This approach uses a public key distribution method such as the Diffie–Hellman exponentiation method to distribute the group key.

Here each user i registers with the Key Centre KC a public key $y_i \equiv a^{x_i} \pmod p$ where x_i is only known to user i. The primitive root 'a' and the prime p are known to all users in the network. The user i transmits to KC a list of members of the group G.

$$i \rightarrow KC: \quad i, \quad G = \{u_1, u_2, \ldots, u_n\}$$

If the user i belongs to the group, then the Key Centre generates a number x_G and sends it to user i enciphered under his public key. That is,

$$KC \rightarrow i : K_{i,G} \equiv y_i^{x_G} \pmod p$$

The user i upon receiving the above message computes

$$(K_{i,G})^{x_i^{-1}} \pmod p$$
$$\equiv a^{x_i x_G x_i^{-1}} \pmod p$$
$$\equiv a^{x_G} \pmod p$$

That is, the group key KG is given by $a^{x_G} \pmod p$. With this method, storage of up to 2^n-1 secret values of x_G is necessary either by the Key Centre or by the users, where n is the total number of users and the KC does not need to know the personal keys of the users x_i $(1 \leqslant i \leqslant n)$.

9.7.2 Method 3

If the KC is however given access to users' personal keys, a modification to the above public key distribution method is as follows:

The group key KG is now made equal to

$$KG = a^{x_1 \cdot x_2 \cdots \cdots x_n} \pmod p \qquad\qquad (9.7)$$

When the ith member of G requests KG from the Centre, KC returns KG

— 163 —

enciphered under the personal key x_i. The master key is represented
by the list of personal keys. Another member of G may be able to
determine $a^{x_i} \pmod{p}$ but he cannot compute x_i without computing a
discrete logarithm. If p is chosen to be a large prime number, this
is not feasible.

Here the key centre KC needs to store only n personal keys. Although
the method uses a one-way function, it is not a public key distribution
method because the KC must have access to secret personal keys of the
users.

Note that in both the schemes 2 and 3, for suitably chosen primes p
of the form $p = 4q + 1$ where q is also a prime, $a = 2$ is a primitive
root.

9.7.3 Method 4

This method considers the establishment of the group key
KG given by (9.7) without the use of the Key Centre. It assumes a
special situation where the n users (0 to n-1) are linked together in
a circular fashion, thus forming a ring. That is, user i always sends
messages to user i + 1 and user n-1 sends message to user 0. The table
9.1 given in Figure 9.5 shows the messages received and transmitted by
user i at various time instants.

Time Instant	Transmitted Message	Received Message
1	$a^{x_i} \pmod{p}$	$a^{x_{i-1}} \pmod{p}$
2	$a^{x_{i-1}x_i} \pmod{p}$	$a^{x_{i-2}x_{i-1}} \pmod{p}$
$\vdots$		
r	$a^{x_{i-(r-1)}\cdots x_i} \pmod{p}$	$a^{x_{i-r}\cdots x_{i-1}} \pmod{p}$
$\vdots$		
n-1	$a^{x_{i-(n-2)}\cdots x_i} \pmod{p}$	$a^{x_{i-(n-1)}\cdots x_{i-1}} \pmod{p}$

Figure 9.5 - Table 9.1: Messages received and transmitted by user i.

At time instant $t = r$, the user i raises the message received from
user i - 1 at time instant $t = r-1$ to the power x_i and transmits it to
user i + 1. The user i forms the group key KG by raising the
received message at time instant $t = n - 1$ to the power x_i. That is,

$$KG = a^{x_{i-(n-1)}\cdots x_{i-1}\,x_i} \pmod{p}$$

— 164 —

Every user in the network can arrive at this common group key by raising the message received at the (n-1)th time instant to the power of his secret key. Any intruder who does not belong to this ring network cannot determine the group key by monitoring the transmitted messages at all links in the network. A more generalized version of this method has recently been published in [41].

9.8   Key Distribution Schemes for Prestel Encryption System

        This chapter is concluded by considering possible key management schemes for Prestel Viewdata system with encryption facility.

        Initially consider the case where a user i (information provider) wishes to store a frame (or frames) on Prestel database which should only be read by user j. This is referred to as the 'Postbox system'. In this case, it is assumed that there is no direct network link between the two users other than via the database. If the user systems only allow DES type symmetric cryptography then the frame key (the key with which a Prestel frame has been enciphered) must be exchanged via some secure courier. This is the only method possible if it is assumed that Prestel is only used as a database and does not play an interactive role in the distribution of keys. However, if the user systems support public key cryptography, then one of the following two approaches can be adopted.

        The first approach uses a public key cryptosystem such as the RSA system. Here the user i can encrypt the frame using the public key of user j (this is assuming that the public file containing enciphering keys of users is available, say, in the form of a telephone directory). User j will then be the only person who would be able to read the frame and not even user i can read the frame he had entered. Alternatively, user i can generate a random frame key which he can use to encrypt the frame using a symmetric algorithm such as the DES. User i then encrypts this frame key using the public key of user j and stores the result at the top of the frame. User j can recover the frame key using his secret RSA key and then read the Prestel frame. User i will also be able to read the frame as he can keep a copy of the frame key which he has generated (say enciphered under his own public key).

- 165 -

The second approach uses a public key distribution system such as the Diffie-Hellman exponentiation system. Here the user i obtains the public key information y_j of user j from the public file and performs $y_j{}^{x_i}$, where x_i is the secret key of user i, to form the common key K_{ij}. Then he generates a random frame key and uses it to encrypt the Prestel frame. The difference between this approach and the one above is that in this case, the frame key is enciphered under the common key and is stored at the top of the frame. User j will be able to read the frame as he can obtain the common key and hence the frame key. Here again only users i and j will be able to read the Prestel frame.

Consider now the case where a group of n users who wish to communicate with each other via the Prestel database. That is, each one of the n users should be able to read the frames entered by any one of the others in the group. Amongst the members of the group, existence of a privileged member referred to as the Manager is envisaged. [The group could represent typically a small company or an organization]. Further it is assumed that every member of the group is linked to the Manager. In such a situation, the Manager assumes the role of the Key Centre described earlier. Any user i who wishes to enter an encrypted frame on Prestel initially establishes a frame key with the Manager following the procedure outlined in Section 9.3. This frame key can then be used to encipher the frame and stored at the top of the frame enciphered under a master key of the Manager. A user j belonging to the group can read the enciphered frame key from the top of the frame and send it to the Manager who returns to him the frame key enciphered under the user j's personal key. Thus user j can decipher the frame entered by user i. With this approach, in addition to the master key, the Manager is required to keep a record of the personal keys of all n users and if there is more than one group, a record of members in each group. One can also use any one of the methods 1 to 3 given in Sections 9.5 and 9.7 to establish a group key among the n users using the Manager as the Key Centre. Having established the group key, the user i can follow the usual procedure of generating a random frame key and storing this at the top of the frame, this time enciphered under the group key. Thus any user belonging to the group can read the frame. Method 3 (see Section 9.7.2) is seen to be the most attractive method as the Manager needs to store a list of keys which is a polynomial function of the number of users n.

- 166 -

Page 166

CHAPTER 10

EXTENSIONS OF THE RSA CRYPTOSYSTEM

10.1   General

As seen in the last chapter, the concept of public key cryptosystems recently proposed by Merkle and Hellman [35] not only provides a novel way of distributing the keys required for the symmetric cryptosystems but it also gives rise to a class of asymmetric public key cryptosystems with digital signature capabilities. From this stage onwards, the thesis mainly concentrates on the analysis and the design of public key systems; in particular, the RSA public key cryptosystem and the Diffie-Hellman public key distribution system. Both these systems are of immense interest among the cryptographic community at the present time.

Before considering some possible extensions of the RSA system to matrix and polynomial rings, it is useful to consider some design aspects of the RSA system over the rational integers. These aspects are more or less applicable to the extended RSA systems considered in subsequent sections.

10.2   Some Design Aspects of RSA System

As mentioned in Section 9.6.2, in the RSA system the encryption is performed by raising the message x ($1 \leqslant x < m$) to the eth power modulo m and the decryption is performed by raising the cipher y to the power d modulo m. That is

$$y \equiv x^e \ (\text{mod } m)$$

and

$$x \equiv y^d \ (\text{mod } m)$$

where m is equal to the product of two large distinct primes p and q. The public encryption key is the pair of integers (e, m) and the secret decryption key is (d, m). The coding exponents are chosen such that they are multiplicative inverse of each other modulo $\emptyset$ (m) and that x^e is a permutation of the residue classes modulo m. That is,

- 167 -

$$e\ d \equiv 1\ (mod\ \emptyset\ (m))\quad where\ \emptyset\ (m) = (p-1)\ (q-1)$$

To design the RSA system, the user needs to choose two large primes p and q to form the modulus m. The magnitude of the modulus m is determined by the required difficulty of breaking the designed system. The table shown in Figure 10.1, taken from [12], gives an indication of the magnitude of m. The number of operations computed for each value of m is based on the best known factoring algorithm for large intergers. (see Section 10.3.1).

$log_{10}m$	Number of operations	Remarks
50	1.4×10^{10}	
100	2.3×10^{15}	At the limits of current technology
200	1.2×10^{23}	Beyond current technology
400	2.7×10^{34}	Requires significant advances in technology
800	1.3×10^{51}	

Figure 10.1   Effort required to factor modulus m

Once an approximate idea of the magnitude of m is decided, then the two primes p and q need to be selected randomly. The prime number theorem states that the primes near m are spaced on the average one every (ln m) integers. Thus even for large primes several hundred digits long, only a few hundred candidates must be tested before finding a prime. Hence one needs to test whether a chosen large number is a prime. There exists elegant probabilistic algorithms [42] which decide with an arbitrarily small uncertainty if the chosen number is a prime. The probabilistic algorithm consists of making many independent tests and declaring the number to be composite when any one test fails. If a test mistakes a composite for being prime with probability f^{-1} then by using k such tests the probability that the algorithm will incorrectly declare a composite to be prime is f^{-k}. Three tests of primality are commonly used [43] namely, the Fermat

– 168 –

test, the Solovay-Strassen test and the Rabin test. In all these cases, it is assumed that the tests are applied to the number b.

10.2.1 Primality Tests

10.2.1.1 Fermat Test

This test is based on the Euler-Fermat theorem previously mentioned. The theorem states that if p is a prime then

$$a^{p-1} \equiv 1 \pmod{p} \text{ for all } a, 1 \leqslant a < p$$

Thus the test consists of choosing 'a' less than the number b and accepting b to be prime if $a^{b-1} \pmod{b}$ is congruent to 1 (mod b). In practice, only a few values of 'a' need to be tried and not all 'a' as indicated above. (Only a small number of composites pass this test even a few times). It is recommended that approximately one hundred tests be made to reliably conclude that the selected number is prime. It is generally believed that choosing a = 3 will identify virtually all composites. The Carmichael numbers (eg.561 = 3.11.17) are known to pass this test even though they are composite.

10.2.1.2 Solovay-Strassen Test [44]

This test identifies the Carmichael numbers as being composite. It picks a random number 'a' between 1 and b-1 and tests whether

$$\gcd(a,b) = 1 \text{ and } J(a,b) = a^{(b-1)/2} \pmod{b} \qquad (10.1)$$

where J(a,b) is the Jacobi symbol and gcd denotes the greatest common divisor. The Jacobi symbol is defined as

$$J(a,p) = \begin{cases} 1 & \text{when } x^2 \equiv a \pmod{p} \text{ has a solution in } Z/pZ \\ -1 & \text{when } x^2 \equiv a \pmod{p} \text{ has no solution in } Z/pZ \end{cases}$$

where Z/pZ denotes the ring of integers modulo p.

If b is prime, then (10.1) is always true and if b is composite (10.1) is false with a probability of a $\frac{1}{2}$. Therefore making k tests yield an answer that is wrong with a probability of 2^{-k} when claiming the input is prime. (When claiming the number is composite, it is always

- 169 -

correct).

10.2.1.3 Rabin Test [42]

As b is necessarily an odd integer, representing b as $b = 2^r s + 1$ where s is odd, this test then chooses a number of values of 'a' randomly in the range 1 to b-1 and accepts b to be prime if either $a^s \equiv 1 \pmod{b}$ or $a^{2^j s} \equiv -1 \pmod{b}$ for some j where $0 \le j < r$. Otherwise b is rejected.

All the three tests can be carried out to check whether the chosen number b is a prime or not. The failure probabilities of each test are discussed in length in [43]. Note that this primality testing procedure needs to be done only once by any system designer and all these tests require computational effort of the order of $O(\log_2 b)$ operations on large integers.

To confound certain factoring algorithms [45], it is desirable to choose primes p and q such that p-1 and q-1 have a large prime factor. This can be done by generating a large prime number b and then letting p (or q) be the first prime in the sequence bi + 1 for i = 2, 4, 6 ... Additional security can be provided if b-1 also has a large prime factor. Furthermore, it is also advisable not to choose p and q too close to each other. If $p \stackrel{.}{=} q$, then $2\sqrt{m}$ is a good approximation of p + q. Knowing p + q gives immediately $\phi(m)$ since $\phi(m) = (p-1)(q-1) = m + 1 - (p+q)$. Further, the primes should not be chosen to be any special primes such as the Fermat primes or Mersenne primes as these are well studied and the resulting product may be more likely to yield to attacks of factorization.

10.2.2 Choice of Coding Exponents

Having chosen the primes p and q, and hence m, the next step is to choose the coding exponents e and d. To do this, the user chooses a number d which is relatively prime to $\phi(m)$. This is done by selecting a number d (mod m) and computing gcd (d, $\phi(m)$). This is done using the Euclid's algorithm given in the Appendix 11. This not only checks whether d and $\phi(m)$ are relatively prime but also gives the multiplicative inverse e. It is known that the gcd function is computable in $O(\log_2 m)$ time [45]. It is necessary to choose both e and d such that they are greater than $\log_2 m$. If $e < \log_2 m$, then small messages will not be disguised by the modulo

- 170 -

reduction process. That is, x^e (mod m) = x^e and the cipher is breakable by brute force attack. If d is smaller than $\log_2 m$, again the system can be broken by a random search by the opponent to determine its value. There is a further condition on the choice of e. As mentioned previously, e needs to be chosen relatively prime to $\emptyset$ (m) or more exactly to $\ell cm^{\dagger}$ (p-1,q-1), for the function x^e (mod m) to permute the residue classes (mod m) (see Section 14.1). But this permutation has fixed messages, that is, there are residue classes x (mod m) which satisfy the congruence

$$x^e \equiv x \text{ (mod m) where } e \geqslant 3 \text{ is odd and m = p.q} \qquad (10.2)$$

It is known [46] that any solution of the congruence

$$x^3 \equiv x \text{ (mod m)} \qquad (10.3)$$

also satisfies (10.2). Note that the congruence (10.3) has exactly 9 solutions in the range $1 \leqslant x \leqslant m-1$, (m = pq). Thus the congruence (10.2) will have at least 9 fixed messages. Blakely [47] has shown that for the congruence to have only 9 fixed messages e must be chosen such that gcd $\{e-1, \ell cm (p-1,q-1)\}$ = 2.

10.3   Cryptanalysis of RSA System

The main cryptanalytical attack seems to be the determination of the secret coding exponent d. There are basically three ways a cryptanalyst might try to determine d from the publicly revealed information (e, m).

10.3.1   Factorization of m

The factorization of m would allow the opponent to compute $\emptyset$ (m) and hence the secret coding exponent d using ed $\equiv$ 1 (mod $\emptyset$ (m)). A large number of factoring algorithms exist [45]. The fastest algorithm known at the present time can factor m in approximately $(\ln m)^{(\ln m/\ln \ln m)^{\frac{1}{2}}}$ steps and is due to R. Schroeppel (unpublished). The table given in Section 10.1 is based on Schroeppel's method and it shows that a number m of 200 digits (decimal) long would provide a margin of safety against future developments.

$\dagger$ ℓcm : least common multiple   − 171 −

10.3.2 Computation of $\emptyset$ (m) Without Factorization of m

Another method of cryptanalysis would be to somehow directly determine $\emptyset$ (m) without factorizing m. It is shown in [12] that the approach of computing $\emptyset$ (m) directly is no easier than factoring m since finding $\emptyset$ (m) enables the opponent to easily factor m. This can be seen as follows:

$$\text{Let } x = p + q = m + 1 - \emptyset \text{ (m)}$$
$$\text{and } y = (p-q)^2 = x^2 - 4m$$

Knowing $\emptyset$ (m), one can determine x and hence y. Using x and y, the primes p and q are given by $p = \dfrac{x + \sqrt{y}}{2}$ and $q = \dfrac{x - \sqrt{y}}{2}$.

10.3.3. Determining d Without Factoring m or Computing $\emptyset$ (m)

The third method of cryptanalysis consists of computing the secret exponent d without factorizing m or determining $\emptyset$ (m). Again it is argued in [12] that provided d is chosen large enough to make a direct search attack infeasible, computing d is no easier than factoring m, since once d is known, m could be factorized easily. This can be seen as follows: If d is known, then it is possible to calculate some multiple of $\emptyset$ (m) using

$$ed - 1 = k \emptyset \text{ (m) for some integer k}$$

Miller [40] has demonstrated that m can be factored using any multiple of $\emptyset$ (m). The opponent, on the other hand, may hope to find a d´ such that it is equivalent to the secret exponent d of the designer. If there are a lot of such d´, then one could use a brute force search method to break the system. But all the d´ differ by the least common multiple of (p-1) and (q-1) and if one is found then m can be factored. Thus finding any such d´ is as difficult as factoring m.

Now some possible extensions of the RSA system are considered.

10.4   Extension of RSA System to Matrix Rings

10.4.1 Trapdoor Rings

Assume R is a finite ring [48] with 1 which is associative

- 172 -

but not necessarily commutative. Suppose that members of the ring R are used as messages and that $r \in R$ is enciphered as r^e where e is the published encrypting exponent. The trapdoor property can now be stated as follows:- there exists some integer $n > 0$ such that $r^{n+1} = r$ for all $r \in R$. These rings are to be referred to as trapdoor rings. For instance in Z/pZ, $r^p = r$ for all $r \in R$. More generally, if we let $R = F_q = GF(q)$, the field of q elements where q is a prime power (p^k) then $r^q = r$ for all $r \in R$. Further, if R and S are any two such trapdoor rings, then the direct sum $R \oplus S$ consists of vectors (r,s) with $r \in R$ and $s \in S$ is another trapdoor ring say T. The number of elements in the ring T is equal to the product of the number of elements in R and S. This above process can be applied repeatedly taking vectors of arbitrarily many components, each taken from some finite field. Considering finite fields F_{q_i} for $1 \leq i \leq j$ where q_i's can be the same or different, the trapdoor ring R is formed by all vectors $x = (x_1, \ldots, x_j)$ where $x_i \in F_{q_i}$ for $1 \leq i \leq j$. The ring R consists of $q_1 \cdot q_2 \cdot \ldots \cdot q_j$ elements and the equality $r^{n+1} = r$ is obeyed for all $r \in R$ where n is equal to $(q_1 - 1)(q_2 - 1) \ldots (q_j - 1)$ or any multiple of it.

There are many finite rings which are not trapdoor rings. Consider for instance, $R = Z/p^2 Z$ where p is a prime. Then, $p^2 = p^3 = \ldots = 0$ in the ring R but $p \neq 0$ in ring R. So the property that $p^{n+1} = p$ is not satisfied for any $n > 0$. More generally, for a ring R to be a trapdoor ring, it is necessary that R have no nilpotent elements except zero. (An element, x, is said to be nilpotent if $x^a = 0$ and $x^{a-1} \neq 0$ for some $a > 0$). However, if we take an integer m to be a square free positive integer say $m = p_1 \cdot \ldots \cdot p_j$, where all the p_i's are distinct primes, then the ring $R = Z/mZ$ is a trapdoor ring. This ring can in fact be regarded as a direct sum of finite fields $F_{p_1} \oplus F_{p_2} \oplus \ldots \oplus F_{p_j}$ as described above. If $j = 2$, then this becomes the standard trapdoor ring used by the RSA cryptosystem.

It has been suggested by Dr R Odoni that every finite trapdoor ring is isomorphic to a direct sum of finite fields. The argument relies on the use of Wedderburn's structure theory [49] for semisimple rings. The main steps involved are as follows:

1.   The ring R is trapdoor implies that R has no nilpotent elements except O.

2.   A finite ring without nonzero nilpotent elements must have a 1 [50].

– 173 –

3. A *finite* ring with 1 and lacking nilpotents ($\neq 0$) is a direct sum of matrix rings with entries in a division algebra (skew field) (Wedderburn's theorem).

4. If any of these matrices is not 1 x 1 then there will be non-zero nilpotent elements in R.

5. Hence R is a direct sum of finite skew fields.

6. A finite skew field is necessarily commutative.

Thus a trapdoor ring R is a commutative ring with 1 which is isomorphic to a direct sum of finite fields. (Note that two rings R and S are isomorphic if there exists a function $f : R \rightarrow S$ which is one-to-one and onto and satisfies $f(r_1 \pm r_2) = f(r_1) \pm f(r_2)$, $f(r_1 r_2) = f(r_1) f(r_2)$ for all $r_1, r_2 \in R$)

The original RSA scheme derived its message space from Z/mZ, the ring of integers modulo m where m is the product of two large distinct primes p and q. Here other finite systems that might serve as a basis for an extended RSA cryptosystem are investigated.

10.4.2 Non-Singular Matrices Over Z/mZ

If the ring of all n x n matrices over the ring $R = Z/mZ$ is considered, it is seen that the ring contains nilpotent elements when $n > 1$. Consider for instance $M = \begin{pmatrix} 0 & 3 \\ 0 & 0 \end{pmatrix}$ over $Z/6Z$; $M^2 \equiv 0$ (mod 6) but $M \not\equiv 0$ (mod 6). To overcome this problem, initially only the group M_n formed by the non-singular matrices of order n, is selected to form the message space of this extended system.

Let us first consider the finite group formed by matrices of order n whose determinants are relatively prime to p and whose elements are in Z/pZ (p prime). That is, the non-singular matrices over Z/pZ are considered.

To begin with, the non-singular matrices over Z/pZ form a finite group because (i) the product of any two members of the set is congruent (mod p) with some member of the set, (ii) every member has its reciprocal, that is, for any two members A and B, there exist members C and D such that $AB \equiv C$ (mod p) and $AD \equiv I$ (mod p) and (iii) the unit matrix is the 'identical' member of the group.

The order of the group formed by these elements can be

– 174 –

shown to be equal to N_p where

$$N_p = (p^n-1) \cdot (p^n-p) \cdots (p^n - p^{n-1}) \qquad (10.4)$$

The argument goes as follows: The elements of the first row vector $\underset{\sim}{U}_1$ of the matrix may consist of any of the p numbers except that they cannot all be zero since this would make the matrix singular. There are therefore p^n-1 ways of selecting this vector. When $\underset{\sim}{U}_1$ has been selected, then $\underset{\sim}{U}_2$ may be any of the p^n possible vectors except those which are congruent with $k_1 \underset{\sim}{U}_1$ (mod p) for $k_1 = \{0, 1, \ldots, p-1\}$. This will be the case if and only if $\underset{\sim}{U}_1$ and $\underset{\sim}{U}_2$ are chosen to be linearly independent. Therefore there are p^n-p ways of selecting $\underset{\sim}{U}_2$. Continuing this procedure, the expression (10.4) for the order N_p is obtained.

If non-singular matrices with elements over Z/pZ are used as messages, then one can form a conventional cryptographic system where the secret key contains the modulus p itself. The encrypting (e) and decrypting (d) exponents can then be determined using

$$ed \equiv 1 \ (\text{mod } N_p) \qquad (10.5)$$

The encrypting key is therefore (e, p, n) and the decrypting key is (d, p, n). None of these keys can be made public and the encryption and decryption procedures are as in the RSA system.

$$\left. \begin{array}{l} M^e \equiv C \ (\text{mod } p) \\[2mm] \text{and} \\[2mm] C^d \equiv M \ (\text{mod } p) \end{array} \right\} \qquad (10.6)$$

where M, C $\in M_n$ (Z/pZ), non-singular matrices over Z/pZ

The above system can be modified to include the public key property as follows: Suppose the modulus is a composite number m whose factorization is

$$m = \prod_{j=1}^{s} p_j^{r_j} \qquad (10.7)$$

Then the order N_m of the multiplicative group formed by non-singular matrices of order n over Z/mZ is given by

– 175 –

$$N_m = \prod_{j=1}^{s} N_{p_j^{r_j}} \qquad (10.8)$$

This ia a consequence of the Chinese Remainder Theorem (Appendix 10).

Let us first determine the order N_{p^r} of the group of non-singular n x n matrices over $Z/p^r Z$ when p is prime and $r > 1$.

Let θ be the homomorphism [48], mapping an n x n matrix A over $Z/p^{r+1}Z$ to A', a matrix over $Z/p^r Z$, via a_{ij} (mod p^{r+1}) $\longmapsto$ a_{ij} (mod p^r).

$$\theta : \qquad A \longmapsto A'$$

$$\theta : \quad M_n(Z/p^{r+1}Z) \longmapsto M_n(Z/p^r Z)$$

This induces a surjective homomorphism between the linear groups formed by these matrices, that is,

$$\theta' : \quad GL(n, p^{r+1}) \longrightarrow\!\!\!\!\!\rightarrow GL(n, p^r)$$

Therefore using group theory [48]

$$\frac{GL(n, p^{r+1})}{Kernel(\theta')} \cong GL(n, p^r)$$

(where $\cong$ denotes isomorphic to)

The kernel consists of the set of matrices which are mapped to the identity matrix I (mod p^r), ie,

$$a_{ii} \equiv 1 \pmod{p^r} \quad \text{for } 1 \leq i \leq n \qquad (10.9)$$

$$a_{ij} \equiv 0 \pmod{p^r} \quad \text{for } i \neq j \qquad (10.10)$$

There are p possibilities for each of the equations (10.9) and (10.10) giving rise to p^{n^2} possibilities. Therefore using group theory, and denoting order by the symbol #,

$$\# GL(n, p^{r+1}) = p^{n^2} \# GL(n, p^r) = \ldots =$$

$$= p^{rn^2} \# GL(n, p)$$

- 176 -

Page 176

But $\#\,GL\,(n,\,p)\;=\;N_p$

$$= (p^n-1)\,(p^n-p)\,\ldots\,(p^n-p^{n-1})\quad \text{from}\quad (10.4)$$

Therefore

$$\#\,GL\,(n,\,p^r)\;=\;p^{(r-1)n^2}\,(p^n-1)\,\ldots\,(p^n-p^{n-1})$$

$$=\;N_{p^r}$$

$$\therefore\quad N_{p_j^{\,r_j}}\;=\;p_j^{(r_j-1)n^2}\,(p_j^n-1)\,(p_j^n-p_j)\,\ldots\,(p_j^n-p_j^{n-1})\qquad (10.11)$$

Substituting (10.11) into (10.8) gives the order N_m. Now as in the
RSA cryptosystem if m is made to be the product of two distinct
primes p and q this simplifies to

$$N_m\;=\;N_p\cdot N_q$$

$$=\;(p^n-1)\,(p^n-p)\,\ldots\,(p^n-p^{n-1})\,(q^n-1)\,(q^n-q)\ldots\,(q^n-q^{n-1})$$

Therefore for a message M

$$M^{N_m}\;\equiv\;I\;(\text{mod } m)$$

and the coding exponents e and d are determined using

$$ed\;\equiv\;1\;(\text{mod } N_m)\qquad\qquad\qquad (10.12)$$

The expression of the order N_m depends on the structure of m, that is,
on its prime factors. This therefore can be used to form a public
key cryptosystem by choosing m to be a large integer (say 200
decimal digits) whose security is the same as that of the RSA system.
Although the order N_m can be used in finding e and d as in
(10.12), it is usually a large number. For instance, even for small
primes such as p = 13 and q = 23, the order is approximately
1.6×10^{22} for 3 x 3 non-singular matrices. Therefore it is
desirable to find the exponent, EXP, of the group, that is, the least
integer greater than zero such that

$$M^{EXP}\;\equiv\;I\;(\text{mod } m)\;\text{for all } M.$$

EXP is a divisor of the order N_m of the group.

$$-\,177\,-$$

Let us first consider the exponent of the group formed by the non-singular matrices over Z/pZ, $M_n(Z/pZ)$.

Let the exponent be ℓ such that

$$A^\ell \equiv I \text{ for all } A \in M_n(Z/pZ)$$

Assume that $p > n$. $A^\ell \equiv I \pmod{p}$ implies that $x^\ell - 1$ is divisible by the minimum polynomial of A. As A ranges over the non-singular matrices of order n over Z/pZ, $x^\ell - 1$ must be divisible by every monic irreducible polynomial $P(x)$ ($\neq x$) of degree $\leq n$ in Z/pZ. Every irreducible polynomial $P(x)_u$ ($\neq x$) of degree u divides $x^{p^u - 1} - 1$. Thus $x^\ell - 1$ must be divisible by $x^{p^u - 1} - 1$.

But

$$x^b - 1 \equiv 0 \pmod{x^a - 1}$$

implies that $a \mid b$

Hence,

$$\ell \equiv 0 \pmod{p^u - 1} \text{ for } 1 \leq u \leq n$$

Therefore,

$$\ell \equiv 0 \pmod{\ell cm \{p-1, \ldots, p^n - 1\}} \text{ certainly} \qquad (10.14)$$

Furthermore, the matrix A given by

$$A = I + \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \vdots & & & & 0 \\ 0 & \cdots & & & 0 \end{bmatrix}$$

satisfies $A^p = I \neq A$ $(p > n)$

That is, A has order p and hence $p \mid \ell$

Hence the exponent of $GL(n, p)$, $p > n$, is given by

$$\ell = p \; \ell cm \{p-1, p^2-1, \ldots, p^n - 1\} \qquad (10.15)$$

Now for any $A \in M_n(Z/pZ)$, using Jordan's Canonical form, there exists a non-singular matrix E such that

- 178 -

$$E^{-1}AE = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_r \end{bmatrix}$$

$$= D$$

Each block B_i is of the form

$$B_i = \begin{bmatrix} \lambda_i & 1 & & O \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ O & & & \lambda_i \end{bmatrix} .$$

ie, $B_i = \lambda_i I_i + N_i$ for some upper triangular nilpotent matrix N_i

where λ_i's are non-zero in $F_{p^{r_i}}$ for some $r_i \leqslant n$.

If the order of D is k, that is, $D^k \equiv I$ (mod p), then as $D = E^{-1}AE$, this gives

$$(E^{-1}AE)^k = (E^{-1}AE)(E^{-1}AE) \ldots k \text{ times}$$

$$= E^{-1}A^k E$$

Thus $A^k \equiv I$ (mod p)

Order of A = order of D = k = ℓcm of orders of B_i. If $N_i = 0$, then order of B_i is a divisor of $p^{r_i}-1$. Hence the order of A divides $\ell cm \{p-1, \ldots, p^n-1\}$. Otherwise, $B_i^p = \lambda_i^p I_i$ will have such an order and hence the order of A divides $p \, \ell cm \{p-1, \ldots, p^n-1\}$.

A multiple of p in the expression (10.15) for the exponent ℓ is expected as the order given by (10.4) contains multiples of p.

Similarly,

$$- 179 -$$

$$A^s \equiv I \pmod{q} \text{ for all } A \text{ in } M_n(Z/qZ)$$

where

$$s = q \, \ell cm \, \{q-1, q^2-1, \ldots, q^n-1\}$$

Therefore exponent EXP of the group GL(n,m) where m = pq is given by

$$\ell cm \, (p \, \ell cm \, \{p-1, p^2-1, \ldots p^n-1\} \, , \, q \, \ell cm \, \{q-1, q^2-1, \ldots q^n-1\} \,)$$

Let us now extend this argument to non-square free modulus m. First consider a matrix A in $M_n(Z/p^2Z)$. Let θ be the natural homomorphism from Z/p^2Z onto Z/pZ (p prime). From the above argument

$$\theta(A)^{EXP} \equiv I \text{ in } M_n(Z/pZ)$$
$$\equiv \theta(I)$$

Therefore,

$$\theta(A^t - I) \equiv 0 \pmod{p} \text{ where } t = EXP$$

This means that every entry in $A^t - I$ is some multiple of prime p and hence

$$A^t - I = p B \text{ for some matrix B.}$$

That is,

$$A^t = I + p B$$
$$A^{tp} = (I + p B)^p$$

Using the binomial theorem,

$$A^{tp} = I + \binom{p}{1} p B + \binom{p}{2} p^2 B^2 + \ldots$$
$$\equiv I \pmod{p}$$

Therefore considering in general a matrix in $M_n(Z/p^kZ)$

$$M_n(Z/p^kZ) \rightarrow M_n(Z/pZ)$$
$$\theta: \qquad A \rightarrow \theta(A)$$

If $\theta(A)$ has order t then A has order t or pt or $p^2 t \ldots$ or $p^{k-1}t$.

If

$$m = \prod_{i=1}^{s} p_i^{r_i}$$

then

$$EXP = \ell cm \{v_1, v_2, \ldots, v_s\}$$

where

$$v_i = p_i^{r_i-1} (w_i)$$

and

$$w_i = p_i \ \ell cm \ (p_i-1, \ p_i^2-1, \ \ldots \ p_i^n-1)$$

$$(10.16)$$

(assuming p_i is greater than n for all i)

Again from (10.16), it is clearly seen that the exponent EXP depends upon the prime factor decomposition of m.

The message space has so far been restricted to only non-singular elements , $M_n(Z/mZ)$. Theoretically, the message space can also include some singular non-nilpotent elements. Consider for instance, $M_2(Z/3Z)$. There are 48 non-singular matrices using (10.4). But there are in addition, some non-nilpotent singular matrices which could also be used as messages. Consider such a matrix X

$$X = \begin{pmatrix} a & b \\ \lambda a & \lambda b \end{pmatrix} \qquad \text{where } \lambda, \ a, \ b \text{ are in } Z/3Z$$

From the Cayley-Hamilton theorem [51], this matrix satisfies a quadratic characteristic equation of the form

$$X^2 - (\text{trace } X) \ X + (\det X) \ I = 0$$

In the singular case, det X = 0. If trace X = 0, then $X^2 = 0$ and so X is nilpotent. If trace X $\neq$ 0, then $X^2 = \lambda X$, $X^4 = \lambda^2 X^2 = X^2$ as $\lambda^2 = 1$ for $0 \neq \lambda \in Z/3Z$; thus $X^3 = \lambda X^2 = \lambda^2 X = X$. Therefore $X^{1+2k} = X$ for k $\geqslant$ 0. Therefore for such non-nilpotent matrices X in $Z/3Z$, $X^{49} \equiv X$.

Note that when n > 2, there may be singular non-nilpotent matrices which do not satisfy an equation of the form $X^k \equiv X$. This happens for instance when the minimal equation is $X^4 - X^2 = 0$. Or even

– 181 –

consider a non-nilpotent matrix X which contains a nilpotent block as shown below

$$X = \left(\begin{array}{c|c} \text{nilpotent} & 0 \\ \hline 0 & \text{non-singular} \end{array}\right) = \left(\begin{array}{c|c} N & 0 \\ \hline 0 & B \end{array}\right)$$

Then

$$X^k = \left(\begin{array}{c|c} N^k & 0 \\ \hline 0 & B^k \end{array}\right)$$

As N is nilpotent for some k, $N^k = 0$ and hence

$$X^k = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & B^k \end{array}\right) \neq X$$

Thus in such cases, this would imply that it is not possible to recover the message. These cases need to be avoided if one decides to include such singular non-nilpotent matrices.

From cryptography point of view, the use of such singular non-nilpotent matrices as messages may seem impractical as the sender cannot easily recognize such matrices. In practice, one would like to determine easily whether a message is within the acceptable set or not. Even the restriction of messages to arbitrary non-singular matrices may pose problems as the sender has no control over the matrix elements but must accept what the plaintext dictates. That is, the sender cannot ensure that his messages will always form non-singular matrices (over any modulus). The sender is faced with the problem of determining whether a plaintext message matrix is non-singular or not. This involves finding the determinant of the n×n matrix and then checking whether the determinant is relatively prime to the modulus using the Euclid's algorithm. (Appendix 11). However it will be seen in Section 10.4.6 that for large m, the probability that an arbitrarily selected matrix is non-singular is very much close to 1.

One common approach to obtain an arbitrary non-singular matrix over the reals is to have the diagonal entries of the matrix message much bigger than the corresponding entries in the row and

– 182 –

column but this 'diagonal dominance' does not always ensure that the matrix will be non-singular when working over finite rings. For instance, consider the matrix A below which is 'diagonally dominant'

$$A = \begin{pmatrix} p-1 & 1 \\ 1 & p-1 \end{pmatrix}$$

Det $A = p^2 - 2p = 0$ over Z/pZ. Hence it is seen that 'diagonal dominance' is not applicable over finite rings.

A simple method to obtain arbitrary non-singular matrices is described in Section 10.4.8 where system implementation is considered.

10.4.3 Orthogonal Matrices Over Z/mZ

Now let us consider the use of a special set of non-singular matrices, namely the set of orthogonal matrices, as the message space of the matrix based RSA system.

The set of orthogonal matrices over a finite field of p elements forms a group as shown below.

Let M_i be a member of the set. Then $M_i^t M_i = I$.
As $(M_i M_j)^t = M_j^t M_i^t$, we have

$$(M_i M_j)^t M_i M_j = M_j^t M_i^t M_i M_j = I$$

Hence the closure property is obeyed. Further $(M_i^{-1})^t = (M_i^t)^{-1} = (M_i^{-1})^{-1} = M_i$ and $(M_i^{-1})^t M_i^{-1} = M_i M_i^{-1} = I$. Let $J = M_i^{-1}$ where $J^t J = I$ and J is orthogonal. Hence the set of orthogonal matrices over Z/pZ forms a group.

The order of the group formed by the nxn orthogonal matrices has been worked out by J. MacWilliams [51].
For odd n, ie, n = 2a + 1 for some integer a, the order is given by

$$2p^a \prod_{i=0}^{a-1} (p^{2a} - p^{2i})$$

For even n, ie n = 2a, the order is given by

- 183 -

$$\frac{2}{p^a + 1} \prod_{i=0}^{a-1} (p^{2a} - p^{2i}) = 2(p^a - 1) \prod_{i=1}^{a-1} (p^{2a} - p^{2i})$$

if -1 is a square in GF (p)

and the order is equal to

$$2(p^a + (-1)^{a+1}) \prod_{i=1}^{a-1} (p^{2a} - p^{2i})$$

if -1 is a non-square in GF (p).

Using the Chinese Remainder Theorem, the order of the group of the orthogonal matrices over Z/mZ, where $m = p_1 p_2$, a square free integer, is equal to

(order of orthogonal matrices over $Z/p_1 Z$) x (order of orthogonal matrices over $Z/p_2 Z$). As the factorization of the modulus m is required to calculate the order, one can use the group of orthogonal matrices in the matrix based public key system.

Now the question how easy it is to construct an orthogonal matrix message needs to be looked at. Cayley's theorem [52] gives an easy way of constructing an orthogonal matrix using a skew-symmetric matrix over the reals. If S is a real skew-symmetric matrix then I + S is non-singular because the characteristic roots of S are purely imaginary and the matrix $A = \frac{I-S}{I+S}$ is an orthogonal matrix. But this is not applicable to finite fields as the determinant of I + S can be equal to 0 (mod $p_1 p_2$). Thus one can use the Cayley's technique to construct orthogonal matrix messages provided one ensures that the determinant of (I + S) is relatively prime to $p_1 p_2$. Note that if I + S is non-singular over Z/mZ (m = $p_1 p_2$), then I - S is also non-singular. This can be seen as follows:

Let W be the inverse of I + S (mod m). That is, (I + S)W $\equiv$ I (mod m).

Then $((I + S)W)^t = W^t(I + S)^t = W^t (I - S) \equiv I$. Hence W^t is the inverse of (I - S) (mod m).

Hence to construct an orthogonal matrix message over Z/mZ, an arbitrary skew-symmetric matrix S over Z/mZ is chosen and I + S is

- 184 -

formed. If I + S is non-singular over Z/mZ, then $(I - S)/(I + S)$ is constructed to form the message. But again the sender is faced with the problem of determining whether the matrix I + S is non-singular or not.

10.4.4 Upper Triangular Matrices Over Z/mZ

Alternatively, let us now consider the set of upper triangular matrices as a possible choice for the message space. If the diagonal entries of an upper triangular matrix are made unity, this ensures that the matrix is invertible over any modulus m as the determinant is equal to 1.

Let M represent such an n x n upper triangular message matrix. One can partition M into I + N where N is a nilpotent matrix and I is the identity matrix. If M is in U_n (Z/pZ) then $(I + N)^P \equiv I$ as $N^P = 0$ assuming p ≥ n-1. The order of the group formed by these upper triangular matrices U_n over Z/pZ is $p^{n(n-1)/2}$. The order becomes $m^{n(n-1)/2}$ when considering matrices U_n over Z/mZ. To determine the exponent EXP of the group formed by such upper triangular matrices over Z/mZ,

let

$$M = \begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix} \quad \text{where a, b, c } \varepsilon \text{ Z/mZ}$$

and let

$$M^k = \begin{pmatrix} 1 & a_k & b_k \\ 0 & 1 & c_k \\ 0 & 0 & 1 \end{pmatrix}.$$

Then

$$M^{k+1} = \begin{pmatrix} 1 & a_k & b_k \\ 0 & 1 & c_k \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$$

- 185 -

$$= \begin{pmatrix} 1 & a+a_k & b+ca_k+b_k \\ 0 & 1 & c_k+c \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore $a_{k+1} = a+a_k$ implies that $a_k = ka$

$c_{k+1} = c+c_k$ implies that $c_k = kc$

and

$$b_{k+1} = b+kca+b_k$$

$\therefore$ $$b_N-b_0 = \sum_{k=0}^{N-1} b_{k+1}-b_k$$

$$= \sum_{k=0}^{N-1} (b+kca)$$

$$b_N = Nb+ca \frac{(N-1)}{2} N$$

Therefore
$$M^k = \begin{pmatrix} 1 & ka & kb + \frac{k(k-1)}{2} ac \\ 0 & 1 & kc \\ 0 & 0 & 1 \end{pmatrix}$$

For $M^k \equiv I \pmod{m}$ to be obeyed, the following conditions must be satisfied:

(i) $ka \equiv 0 \pmod{m}$

(ii) $kc \equiv 0 \pmod{m}$

(iii) $kb+\frac{k(k-1)ac}{2} \equiv 0 \pmod{m}$

To satisfy (i) and (ii) $k \equiv 0 \pmod{m}$

To satisfy (iii) $\frac{k(k-1)}{2} \equiv 0 \pmod{m}$

ie , $k(k-1) \equiv 0 \pmod{2m}$

Therefore if m is even, the exponent, EXP, is 2m and if m is odd then EXP is m. Therefore it is seen that both the order and the exponent do not depend on the prime factors of m. Hence this cannot be used in a public key cryptosystem but again one can use it in a

- 186 -

conventional cryptosystem with the secret key (e, d, m, n) where
ed $\equiv$ 1 mod (2m or m).

On the other hand, if the message space is altered to contain upper triangular matrices with diagonal entries prime to m, then such messages are invertible modulo m. This is not a serious problem as in practice m is a product of large primes and the diagonal elements can be chosen to be relatively small integers. Now the order of the group formed by such matrices is determined as follows.

Considering a nxn matrix, it is required that all the n diagonal entries must be coprime to m. The number of integers less than m and coprime to m is given by the Euler totient function $\phi(m)$. The remaining $\frac{1}{2}$ n(n-1) superdiagonal entries of the matrix may take any value modulo m. Therefore the order is equal to $m^{n(n-1)/2}\phi(m)^n$. The vital difference between this order and the one calculated above is that now the order of the group is dependent on the prime factors of m. Hence the modulus m needs to be factorized before the decryption exponent d can be calculated using ed $\equiv$ 1(mod(order)). As for the set of non-singular matrices, one can determine the exponent of the group formed by these upper triangular matrices with diagonal entries prime to m. The exponent can be used instead of the order in finding e and d.

Initially, consider a square free modulus. Let

$$m = \prod_{j=1}^{s} P_j$$

Consider first a message M in Z/p_1Z whose diagonal elements are relatively prime to p_1

$$M = \begin{pmatrix} a_{11} & ? \\ & \ddots & \\ O & & a_{nn} \end{pmatrix} \quad \text{where } (a_{ii}, p_1) = 1, \forall i \\ 1 \leqslant i \leqslant n$$

Partitioning the matrix M into a diagonal matrix D_1 and an upper triangular nilpotent matrix U_1, that is, $M = D_1 + U_1$, then it is seen that $D_1 \cdot U_1$, $U_1 \cdot D_1$ and U_1^2 are also upper triangular nilpotent. Then inductively,

- 187 -

if $M^r = D_1^r + U_r$, we have

$$M^{r+1} = (D_1 + U_1)(D_1^r + U_r)$$

$$= D_1^{r+1} + \underbrace{U_1 D_1^r + D_1 U_r + U_1 U_r}_{\text{nilpotent upper triangular}}$$

$$= D_1^{r+1} + U_{r+1}$$

Hence

$$M^{\phi(p_1)} \equiv I + U_\phi \pmod{p_1}$$

But

$$(I + U_\phi)^{p_1} \equiv I + U_\phi^{p_1} \pmod{p_1}$$

$$(I + U_\phi)^{p_1^2} \equiv I + U_\phi^{p_1^2} \pmod{p_1}$$

$$\vdots \qquad\qquad \vdots$$

Thus $M^{p_1^t \phi(p_1)} \equiv I \pmod{p_1}$ for some t.

If $p_1 \geqslant n-1$ then $t = 1$. Therefore the exponent of upper triangular matrices with coprime elements along the diagonal is $\phi(p_1) \cdot p_1$

If $m = \displaystyle\prod_{j=1}^{s} p_j$ then the exponent divides

$$\ell cm \left\{ \phi(p_1) \cdot p_1, \ \phi(p_2) \cdot p_2, \ldots, \ \phi(p_s) \cdot p_s \right\}$$

If m is made to be a non-square free modulus given by

$$m = \prod_{j=1}^{s} p_j^{r_j}$$

then a bound for the exponent can be calculated as follows:

First consider an n×n upper triangular non-singular matrix M over $Z/p^r Z$, $(p \geqslant n)$. Again let $M = D_1 + U_1$ where D_1 is a diagonal matrix and U_1 is an upper triangular nilpotent matrix over $Z/p^r Z$. Following the argument given above, it is seen that

$$M^{\phi(p^r)} \equiv I + U_\phi \pmod{p^r}$$

– 188 –

where $U_\emptyset$ is some upper triangular nilpotent matrix.

Hence

$$(I + U_\emptyset)^P \equiv I + p \cdot U_{\emptyset_1} \pmod{p^r}$$

$$(I + p \cdot U_{\emptyset_1})^P \equiv I + p^2 \cdot U_{\emptyset_2} \pmod{p^r}$$

$$\vdots \qquad\qquad \vdots$$

$$(I + p \cdot U_{\emptyset_{r-1}})^P \equiv (I + U_\emptyset)^{p^r} \equiv I + p^r \cdot U_{\emptyset_r} \pmod{p^r}$$

$$\equiv I \pmod{p^r}$$

Therefore

$$_M \emptyset(p^r) \; p^r \equiv I \pmod{p^r}$$

Hence the exponent of the group formed by upper triangular non-singular matrices over Z/mZ, where m is a non-square free modulus given above, divides

$$\ell cm \left\{ \emptyset(p_1^{r_1}) p_1^{r_1}, \; \emptyset(p_2^{r_2}) p_2^{r_2}, \; \ldots, \; \emptyset(p_s^{r_s}) p_s^{r_s} \right\}$$

10.4.5 Linear Fractional Group [53]

Finally, let us consider the group of linear fractional substitutions over Z/mZ, where m is equal to the product of r distinct primes to see if such a group is suitable for a trapdoor system. The linear fractional group LF(n,m) is very much similar to the linear homogenous group GL(n,m) considered earlier. Here only the case n = 2 is examined. First consider the group LF(2,p) where p is a prime. These substitutions are of the form

$$u : x \to \frac{ax + b}{cx + d} \quad \text{over } Z/p*Z = \{ 0, 1, \ldots, p-1, \infty \}$$

The symbol ∞ is adjoined to represent any formal quotient y/0 where y is a non-zero element of the field. The linear fractional substitutions LF(2,p) can be obtained from the linear homogenous substitution GL(2,p) on variables x_1 and x_2 by setting $x \equiv x_1/x_2$. The number of transformations of the form u is finite and they form a group as the product of any two substitutions is also a substitution of the same form. The order of the group LF(2,m) can be determined as follows, where $m = \prod_{i=1}^{r} p_i$.

$$- 189 -$$

For u to represent a substitution, it is required that the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be invertible, that is, the determinant $ad-bc$ prime to m. Now let us consider the two matrices

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \lambda a & \lambda b \\ \lambda c & \lambda d \end{pmatrix} \quad \text{where } \lambda \varepsilon \ Z/mZ$$

The corresponding two substitutions represented by

$$\frac{ax + b}{cx + d} \quad \text{and} \quad \frac{\lambda \ (ax + b)}{\lambda \ (cx + d)} \quad \text{over } Z/mZ$$

are identical if $(\lambda, \ m) = 1$

Consider the matrix M_p given by

$$M_p = \begin{pmatrix} a_p & b_p \\ c_p & d_p \end{pmatrix} \quad \text{where } a_p d_p - b_p c_p \not\equiv 0 \ (\text{mod } p)$$

If the condition that $\det M_p \not\equiv 0 \ (\text{mod } p)$ can be satisfied by choosing a_p, b_p, c_p and d_p appropriately, then the Chinese Remainder Theorem (Appendix 10) allows us to find a, b, c and d over Z/mZ such that:

$$a \equiv a_{p_1} \ (\text{mod } p_1), \ a \equiv a_{p_2} \ (\text{mod } p_2), \ldots \ a \equiv a_{p_r} \ (\text{mod } p_r)$$

and similarly for b, c and d.

Thus the order of LF(2,m) can be found by finding the orders of LF(2,p_i) for $1 \leqslant i \leqslant r$. The number of possible values of λ which result in identical substitutions is given by the Euler totient function $\emptyset(p_i)$ for each p_i. Thus the order is given by

$$\# \ LF(2,m) \ = \ \frac{\displaystyle\prod_{i=1}^{r} \ \# \ LF^{\prime}(2,p_i)}{\displaystyle\prod_{i=1}^{r} \ \emptyset(p_i)}$$

But

$$\# \ LF^{\prime}(2,p) \ = \ \# \ GL(2,p)$$
$$= (p^2-1)(p^2-p) \quad \text{from Section 10.4.2 with n=2}$$

- 190 -

$$\therefore \quad \# \, LF(2,m) \;=\; \frac{\prod\limits_{i=1}^{r} (p^2{}_i - 1) \cdot (p_i{}^2 - p_i)}{\prod\limits_{i=1}^{r} (p_i - 1)}$$

$$\# \, LF(2,m) \;=\; \prod\limits_{i=1}^{r} (p_i + 1)\, p_i\, (p_i - 1)$$

Thus as the order depends on the factorization of the modulus m, this is suitable for a trapdoor system. The messages are of the form $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with a, b, c, $d \in Z/mZ$ and $ad - bc \not\equiv 0 \pmod{m}$. The encrypting and decrypting exponents e and d can be determined using $ed \equiv 1 \pmod{\# LF(2,m)}$.

10.4.6 Proportion of Non-Singular Matrices Over Z/mZ

The extended RSA system described above requires the message space to consist of either arbitrary non-singular matrices or orthogonal matrices or invertible upper triangular matrices including the diagonal elements over Z/mZ. In the first two cases it is required to choose arbitrary message matrices which are non-singular. In the case of orthogonal matrices, recall that one method of forming a message requires the matrix $(I+S)$ to be non-singular. Hence some bound on the fraction of $n \times n$ matrices which are invertible modulo m will be useful. If the elements of the $n \times n$ matrix are chosen uniformly from the integers modulo m, then these bounds can be taken as the bounds on the probability that the matrix is non-singular.

An $n \times n$ matrix M has an inverse modulo m if and only if the rows (columns) of the matrix form a linearly independent set of vectors modulo m. Using the argument given in Section 10.4.2, if $m = p$, a prime, then the number of invertible matrices $(\bmod\ p)$ is given by

$$(p^n - 1)\,(p^n - p)\cdots(p^n - p^{n-1}) \;=\; \prod\limits_{i=1}^{n-1} (p^n - p^i)$$

The total number of $n \times n$ matrices with elements over Z/pZ is equal to p^{n^2}. Thus the proportion of invertible matrices $(\bmod\ p)$ is equal to

$$\frac{\prod\limits_{i=1}^{n-1} (p^n - p^i)}{p^{n^2}}$$

- 191 -

$$= \quad (1-1/_p n) \; (1-p/_p n) \; \dots \; (1-p^{n-1}/_p n)$$

$$= \quad \prod_{i=1}^{n} (1-p^{-i})$$

Thus as the size of the prime increases, the probability that a chosen matrix is non-singular approaches 1. If the proportion of non-singular matrices over Z/mZ is considered, where $m = \prod_{j=1}^{r} p_j$, then it is equal to

$$\frac{\displaystyle\prod_{j=1}^{r} \; \prod_{i=1}^{n} (p_j^{\,n}-p_j^{\,i})}{\left(\displaystyle\prod_{j=1}^{r} p_j\right)^{n^2}}$$

$$= \quad \prod_{j=1}^{r} \; \prod_{i=1}^{n} (1-p_j^{-i})$$

Similar expressions can be found for the proportion of non-singular matrices over Z/mZ, where m is a non-square free integer, using the expression for the order N_m derived in Section 10.4.2. The proportion is then given by N_m/m^{n^2}. From the cryptography point of view where m is a large integer composed of a few large prime factors, this proportion is very much close to 1. Even for small primes $p_1=13$ and $p_2=23$, the proportion is approximately 0.87, for n=2.

10.4.7 System Design and Operation

The designer randomly chooses large primes p_1 to p_r for some $r \geqslant 2$ following the guidelines suggested in Section 10.2 and these are kept secret. But the primes need not be necessarily distinct. Then he specifies what his message space is going to be, whether it consists of arbitrary non-singular matrices or orthogonal matrices or upper triangular matrices with invertible elements along the diagonal. This information along with the dimension n of the matrix message is made public. Then he determines the coding exponents e and d using the equation $ed \equiv 1 \bmod (Order \; or \; Exponent)$. The order and the exponent expressions for the different message spaces are given in the earlier sections. The public encryption key is therefore given by (e, m, n) and the secret decryption key is

– 192 –

(d, m, n). The system is also suitable for authentication purposes like the RSA system.

A general procedure to construct a matrix message is as follows. The plaintext message is divided into blocks of integers less than the modulus m and a sequence of nxn message matrices M_i is constructed by arranging the integers in order as they occur, left to right and top to bottom. The encryption procedure consists of raising each of these plaintext matrices to the power e modulo m. The ciphertext matrices, C_i, produced are then transmitted to the receiver by sending the ciphertext matrix elements in order as they occur in the matrix, left to right, top to bottom, with a space separating the elements. The receiver recovers the original message by first reconstructing the sequence of nxn ciphertext matrices and then raising each matrix C_i in the sequence to the power d modulo m. Thus the main operation in both the encryption and decryption procedures consists of raising an nxn matrix to some power modulo m. This can be performed using the Square and Multiply Technique [45] which is now briefly described.

10.4.7.1 Square and Multiply Technique

Consider the binary representation of the encryption exponent e. Each '1' in the representation is now replaced by a pair of letters SX and each '0' is replaced by the letter S. The symbol SX which appears at the left is now crossed off. The result is a rule for computing M^e if S is interpreted as the squaring operation and X is interpreted as the multiplying operation by M. Each of these operations is performed modulo m. The flowchart of the algorithm is given in Figure 10.2. If the encryption exponent is chosen to have only a few '1' digits in its binary representation, this will make this algorithm run faster than if the exponent has a random binary representation. A similar procedure can be carried out for decryption using the exponent d.

Further, the intermediate computations need not be reduced over Z/mZ but rather over Z/kmZ for any positive integer k. The final answer is computed by reducing the answer over Z/kmZ into Z/mZ. This gives the correct answer because $(a_1 \cdot a_2 \pmod{km}) \equiv a_1 \cdot a_2 \pmod{m}$.

- 193 -

Let $e = (e_k, \ldots, e_2, e_1)$ $\qquad$ $e_i \in GF(2)$


```
              ( START )
                  |
                  v
        +-------------------+
        |  INITIALIZATION   |
        |    i <--- 1       |
        |    C <--- I       |
        +-------------------+
                  |
                  v
        /-------------------\
   NO  <    e(i) = 1 ?       >
   |    \-------------------/
   |              | YES
   |              v
   |    +-------------------+
   |    |    C = C * M       |
   |    +-------------------+
   |              |
   |              v
   |    /-------------------\   YES   +----------+
   +--><     i = k ?         >------->|  C = M^e |
        \-------------------/         +----------+
                  | NO                     |
                  v                        v
        +-------------------+          ( STOP )
        |   M <--- M * M     |
        +-------------------+
                  |
                  v
        +-------------------+
        |   i <--- i + 1     |
        +-------------------+
```
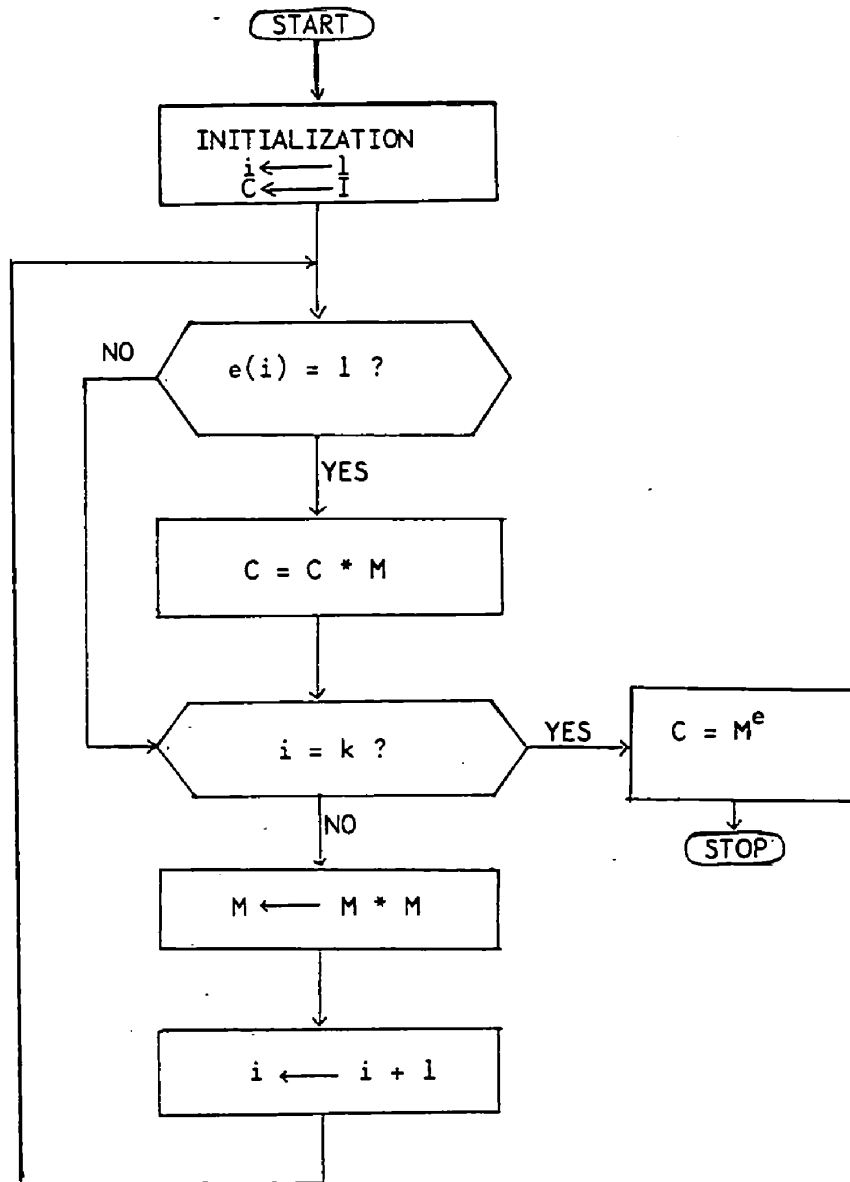
All operations are modulo m

Fig. 10.2 - Algorithm for computing M^e (mod m)

$-$ 194 $-$

10.4.8 System Implementation

This extended RSA system using matrix messages has been simulated on the Prime Computer. The encryption and the decryption of message matrices have been performed using the Square and Multiply Technique given above.

10.4.8.1 Non-Singular Matrix Messages

In the case of non-singular matrix message space, the message matrix M is constructed according to the general procedure given in Section 10.4.7 and its determinant is calculated using the program DETMOD.F77 given in Appendix 12. Then Euclid's algorithm is used to test whether the determinant is relatively prime to the modulus m. If so, the message is raised to the power e over Z/mZ using the program MATEXP.FTN given in Appendix 13. The same program is used to decrypt the ciphertext matrices with exponent d. Recall from Section 10.4.6 that the probability of an arbitrarily chosen matrix message M over Z/mZ, where m = pq and p and q are large primes, is non-singular, is very much close to 1.

One can also construct an arbitrary non-singular matrix M over Z/mZ by multiplying together an upper triangular matrix U with unit diagonal and a lower triangular matrix L with unit diagonal over Z/mZ. The elements other than the diagonal ones in U and L can be arbitrarily chosen modulo m. As both U and L are non-singular over Z/mZ, M = LU is non-singular over Z/mZ. Further the non-commutativity property of matrices ($UL \neq LU$ in general) ensures that the opponent still needs to factorize m to be able to calculate the decrypting exponent d, in contrast to the case of upper triangular matrices with unit diagonal considered in Section 10.4.4. That is, $M^e = (LU)^e \neq U^e L^e$. Thus although $U^{ed}_1 \equiv U$ (mod m), $L^{ed}_1 \equiv L$ (mod m) where $ed_1 \equiv 1$ (mod m or 2m) (see Section 10.4.4), $M^{ed}_1 \not\equiv M$ (mod m). However $M^{ed} \equiv M$ (mod m) where $ed \equiv 1$ mod ($\#GL(n,m)$ or EXP $GL(n,m)$) (see Section 10.4.2). The receiver can obtain the matrices L and U uniquely given the matrix M.

10.4.8.2 Upper Triangular Matrix Messages

In the case of upper triangular matrix messages with invertible diagonal elements, the message matrix is again constructed according to the general procedure given in Section 10.4.7. The lower triangular section of the message (excluding the diagonal) is

- 195 -

set to zero. The diagonal elements are almost arbitrarily chosen to be relatively prime to the modulus m. This can be done provided they are relatively small, as the prime factors of m are normally very large. Again these messages are encrypted/decrypted using the program MATEXP.FTN given in Appendix 13. Note that in this case, only the upper triangular entries of the ciphertext matrices (ie, $\frac{n(n+1)}{2}$ elements of each ciphertext matrix) need to be transmitted to the receiver.

An example in each of these two message spaces showing the various parameters involved is given below.

10.4.8.3 Example 1 : 2 x 2 Non-Singular Matrix Messages

Let the modulus be m = $p_1^2 p_2 = 3^2 5 = 45$

Exponent of the group formed by 2 x 2 non-singular matrices over $Z/45Z$ divides $\ell cm \{ v_1, v_2 \}$

where

$$v_1 = 3^{2-1} \ell cm \{3, 3-1, 9-1\} = 3 \ell cm \{3, 2, 8\}$$
$$v_2 = \ell cm \{5, 5-1, 25-1\} = \ell cm \{5, 4, 24\}$$

Thus EXP is a divisor of 1080. Hence the coding exponents e and d are given by

ed $\equiv$ 1 (mod 1080)

One pair (e, d) which satisfies the above congruence is e = 23, d = 47.

Let us assume that the plaintext message to be encrypted is 81334. Dividing the message as (8 13 3 4), the plaintext message matrix can be constructed as

$$P = \begin{pmatrix} 8 & 13 \\ 3 & 4 \end{pmatrix}$$

The determinant of P is equal to $-7 \equiv 38$ (mod 45) and (det P, 45) = 1. Hence P is non-singular (mod m).
The ciphertext matrix is then given by

— 196 —

$$C \equiv P^e \equiv \begin{pmatrix} 8 & 13 \\ 3 & 4 \end{pmatrix}^{23} \quad (\text{mod } 45)$$

$$\equiv \begin{pmatrix} 38 & 34 \\ 39 & 31 \end{pmatrix} \quad (\text{mod } 45)$$

This ciphertext matrix is transmitted to the receiver as (38 34 39 31). The receiver reconstructs the matrix C and computes C^d (mod m) to obtain P. That is,

$$C^d \ (\text{mod } 45) \equiv \begin{pmatrix} 38 & 34 \\ 39 & 31 \end{pmatrix}^{47} \quad (\text{mod } 45)$$

$$\equiv \begin{pmatrix} 8 & 13 \\ 3 & 4 \end{pmatrix} \quad (\text{mod } 45)$$

$$\equiv P$$

10.4.8.4 Example 2 - 3 x 3 Upper Triangular Matrix Messages
        Let the modulus m = p.q = 41.29 = 1189
Exponent of the group formed by 3 x 3 upper triangular non-singular matrices over Z/1189Z divides

$$\text{lcm} \ \{40.41, \ 28.29\} \ = 332920$$

Choosing the encrypting exponent, e = 1317, the decrypting exponent d can be calculated using Euclid's Algorithm (Appendix 11) and is equal to 117293. That is,

$$1317. \ 117293 \equiv 1 \ (\text{mod } 332920)$$

Let us assume that the plaintext message to be encrypted is 2326772051 41. In this example, the message is divided into 2-digit blocks of integers less than m starting from right to left as

(23 26 77 20 51 41)

The upper triangular plaintext message matrices then become

$$P_1 = \begin{pmatrix} 90 & 23 & 26 \\ 0 & 50 & 77 \\ 0 & 0 & 48 \end{pmatrix}, \ P_2 = \begin{pmatrix} 31 & 20 & 51 \\ 0 & 215 & 41 \\ 0 & 0 & 289 \end{pmatrix}$$

- 197 -

where the diagonal elements are arbitrarily chosen to be relatively prime to m (=1189).

The ciphertext message matrices are then given by

$$C_1 \equiv \begin{pmatrix} 90 & 23 & 26 \\ 0 & 50 & 77 \\ 0 & 0 & 48 \end{pmatrix}^{1317} \equiv \begin{pmatrix} 1105 & 458 & 1070 \\ 0 & 50 & 251 \\ 0 & 0 & 831 \end{pmatrix} \pmod{1189}$$

$$C_2 \equiv \begin{pmatrix} 31 & 20 & 51 \\ 0 & 215 & 41 \\ 0 & 0 & 289 \end{pmatrix}^{1317} \equiv \begin{pmatrix} 843 & 774 & 660 \\ 0 & 592 & 41 \\ 0 & 0 & 405 \end{pmatrix} \pmod{1189}$$

These ciphertext matrices are transmitted to the receiver as

(1105 458 1070 50 251 831 843 774 660 592 41 405).

The receiver reconstructs the matrices C_1 and C_2 and computes $C_1{}^d$ (mod m) and $C_2{}^d$ (mod m) to obtain P_1 and P_2 and hence the plaintext message. That is,

$$C_1{}^d \equiv \begin{pmatrix} 1105 & 458 & 1070 \\ 0 & 50 & 251 \\ 0 & 0 & 831 \end{pmatrix}^{117293} \equiv \begin{pmatrix} 90 & 23 & 26 \\ 0 & 50 & 77 \\ 0 & 0 & 48 \end{pmatrix} \pmod{1189}$$

$$C_2{}^d \equiv \begin{pmatrix} 843 & 774 & 660 \\ 0 & 592 & 41 \\ 0 & 0 & 405 \end{pmatrix}^{117293} \equiv \begin{pmatrix} 31 & 20 & 51 \\ 0 & 215 & 41 \\ 0 & 0 & 289 \end{pmatrix} \pmod{1189}$$

10.4.9 Discussion

Thus one can see that the RSA system can be generalized to matrix rings provided the message space is restricted to avoid nilpotent elements. The group of non-singular matrices over Z/mZ, the group of orthogonal matrices over Z/mZ and the group of upper triangular matrices over Z/mZ with diagonal elements coprime to m have been investigated. From a practical implementation point of view, the upper triangular (non-singular) matrix message space seems to be the better candidate as the messages can be constructed in an almost arbitrary manner. In the case of non-singular and orthogonal matrices, an additional procedure to find the determinant of the message matrix is required. From Section 10.4.6, it is seen that for a large enough modulus, the probability that an arbitrarily chosen

– 198 –

message matrix is non-singular is very much close to 1. Further if the non-singular matrix is constructed as a product of upper and lower triangular matrices, then it seems that arbitrary non-singular message matrices can be easily constructed.

For a cryptanalyst to break the system by computing the secret decoding exponent d, he needs to find the order (or the exponent) of the group. In all the three cases, it is seen that the factorization of the modulus is required to compute the order (or the exponent) of the group. Provided m is chosen to be a large integer (say 200 decimal digits long, see Figure 10.1), this provides a secure system. Other attacks such as the computation of the order without finding the prime factors of m and determining the secret coding exponent d without factoring m or calculating the order can be shown to be as difficult as factoring m using similar arguments to those given in Section 10.3. Thus this extended matrix RSA system provides a similar level of security as the RSA system over integers.

Further two points are worth mentioning regarding this extended system. Firstly it is seen that a non-square free modulus can be used with this system which is not possible with the RSA system over integers. That is, powers of primes can be used to form the modulus m. Secondly, the use of a matrix as a message allows large amounts of data to be processed within one encryption/decryption cycle. Whether this is an advantage depends upon the ease with which matrix manipulation can be carried out in real time.

Also the use of a matrix as a message may provide some extra features to the system. Consider, for instance, a non-singular message matrix divided into blocks as shown below

$$M = \left(\begin{array}{c|c} M_1 & X_1 \\ \hline X_2 & M_2 \end{array} \right)$$

where M_1 and M_2 represent genuine message blocks whereas X_1 and X_2 are redundant random blocks. Encryption of such a message M would truly 'internally' mix the message elements and the random elements in the matrix. In the case of RSA system over integers, a similar effect can be achieved by sending a sequence of ciphers which contains some random elements. For instance, the sequence may be $c_1, x_1, c_2, x_2, \ldots$

- 199 -

where c_i represents genuine cipher and x_i represents random cipher.
In the former case, the internal structure of the matrix and in the
latter case, the pattern of the sequence need to be prearranged
between the sender and the receiver. In the case of the matrix
system, the knowledge of the internal structure of the matrix does
not give any help to the opponent as the message is enciphered
'internally' and hence this information can be made public. In the
latter case, however, the knowledge of the pattern sequence helps the
opponent to discard the random ciphers and hence this information
cannot be made public. The use of chaining techniques in such matrix
systems is considered in Chapter 15.

10.5   Extension of RSA System to Polynomial Rings

        Another ring of special interest is the polynomial ring
R [x] which consists of polynomials whose coefficients are elements
of an arbitrary ring R. A possible extension of the factorization
trapdoor system in the ring of polynomials R [x] is considered. In
particular, to begin with, the ring F [x] where F is a finite field
is looked at.

10.5.1 Concept of Galois Field
        Let P(x) be a given polynomial in x of degree n with
integral coefficients not all divisible by a given prime p. Let
F(x) be any polynomial in x with integral coefficients. On dividing
F(x) by P(x), a quotient Q(x) and a remainder of degree n-1 at most
is obtained, where the remainder may be written in the form f(x) +
p q(x), and

$$f(x) \equiv a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

Each of the coefficients a_i belong to the set $\{0, 1, 2, \ldots, p-1\}$ and
q(x) is a polynomial with integral coefficients. That is,

$$F(x) = f(x) + p q(x) + P(x) \cdot Q(x)$$

The totality of functions F(x) which can be obtained by holding f(x)
fixed and varying q(x) and Q(x) in all possible ways, subject to
maintaining the named properties of q(x) and Q(x), constitute a

- 200 -

class of residues which is completely determined by f(x), the given prime p and P(x). Two polynomials in x with integral coefficients are congruent moduli p and P(x) if and only if they belong to the same class of residues moduli p and P(x). The number of distinct residue classes moduli p and P(x) is equal to the number of functions of the form f(x). Since each of the n independent coefficients a_i can take any one of the p values, there are p^n possible residue classes. These residues constitute a finite field called the Galois field [54]. If P(x) is not irreducible modulo p or if p is not a prime, then the residues do not form a field since at least two non-zero residues can be found whose product is 0 moduli p and P(x).

10.5.2 A Polynomial Based RSA System

Using this concept, one can design a conventional crypto-graphic system as follows. Since the number of residue classes is finite, it must be possible to find two numbers r and s such that

$$(f(x))^r \equiv (f(x))^s \mod (p, P(x))$$

The non-zero elements f(x) of the extension field $GF(p^n)$ form a multiplicative group. The order of this group is equal to p^n-1, that is,

$$(f(x))^{1 + k(p^n-1)} \equiv f(x) \mod (p, P(x))$$

where k is an arbitrary non-negative integer.

The system designer chooses randomly a large prime number p (see Section 10.2) and a high degree irreducible polynomial P(x) over GF(p) of degree n. (See Section 10.5.2.1) The message space of this cryptographic system consists of polynomials $\{m(x)\}$ belonging to F[x] where F is the finite field GF(p). The messages are represented as blocks of sequences of p-ary digits each of size n associated with a polynomial m(x) over GF(p) of degree less than n. The encryption and decryption coding exponents are determined using

$$ed \equiv 1 \mod (\text{order})$$

ie , $$ed \equiv 1 \mod (p^n-1) \qquad (10.17)$$

- 201 -

The encryption procedure then consists of raising the message polynomial to the power e to form the cipher polynomial, $c(x)$

$$c(x) \equiv (m(x))^e \mod (p, P(x))$$

The decryption procedure consists of raising the cipher polynomial to the power d to obtain the message polynomial using

$$m(x) \equiv (c(x))^d \mod (p, P(x))$$

The encryption key is $(e, p, P(x))$ and the decryption key is $(d, p, P(x))$. Both these keys need to be kept secret since the knowledge of the encryption key would allow the opponent to calculate d using (10.17) and vice versa. Hence this does not give rise to a public key system as such.

Note further that the probability that $c(x) \equiv m(x) \mod (p, P(x))$ is equal to $\gcd(e-1, p^n-1)/p^n$ and this proportion can be made small by the system designer through appropriate choice of e, p and n.

This actually is the generalization of the Pohlig–Hellman secret key scheme over GF(p) to arbitrary finite fields $GF(p^n)$ where p is a prime and n is a positive integer. Note that the field $GF(p^n)$ is isomorphic to $Z/pZ [x] /P(x)$ where $P(x)$ is an irreducible polynomial of degree n over Z/pZ.

This can be further extended to the case where the system designer chooses high degree irreducible polynomials $P_i(x)$ and large distinct primes p_i for $1 \leq i \leq s$. Let the degree of irreducible polynomials be n_i for $1 \leq i \leq s$. The coding exponents e_i and d_i can then be determined using

$$e_i d_i \equiv 1 \mod(p^{n_i}-1) \quad \text{for } 1 \leq i \leq s$$

The message is divided into groups of s blocks where the ith block is of size n_i and consists of a sequence of p_i-ary digits. The ith plain block is associated with a polynomial $m_i(x)$ of degree less than n_i. The ith cipher block $c_i(x)$ is produced using

$$c_i(x) \equiv (m_i(x))^e \mod (p_i, P_i(x)) \quad \text{for } 1 \leq i \leq s$$

– 202 –

Now let n_{max} = maximum of $(n_1, \ldots n_s)$. Each $c_i(x)$ is now extended to form a polynomial of degree $n_{max} - 1$ by randomly generating the extra coefficients required over $Z/p_i Z$ for $1 \leqslant i \leqslant s$. Thus there are s polynomials each of degree $n_{max} - 1$ with coefficients over $Z/p_i Z$ for $1 \leqslant i \leqslant s$. These polynomials are combined to form a single polynomial $c(x)$ over Z/mZ where $m = \prod_{i=1}^{s} p_i$ using the Chinese Remainder Theorem (Appendix 10). $c(x)$ is the cipher polynomial which is transmitted to the receiver. The receiver reduces the polynomial $c(x)$ modd (p_i, x^{n_i}) to obtain $c_i(x)$ for $1 \leqslant i \leqslant s$. Then he uses the coding exponents d_i, to obtain $m_i(x)$ using $m_i(x) \equiv (c_i(x))^{d_i} \mod(p_i, P_i(x))$ for $1 \leqslant i \leqslant s$. Hence the encryption key consists of $(e_i, p_i, P_i(x))$ and the decryption key consists of $(d_i, p_i, P_i(x))$ for $1 \leqslant i \leqslant s$. Note that this is still a conventional (symmetric) cryptosystem.

The above method can be modified to give a public key system. This has been first proposed by Kravitz [55]. This is based on the difficulty of obtaining the degrees of the irreducible factors of a polynomial of large degree over a finite field, $F = Z/pZ$, p prime.

Let $f(x)$ be a polynomial of degree s which is equal to the product of r distinct irreducible polynomials over Z/pZ. That is,

$$f(x) = \Pi_1(x) . \Pi_2(x) \ldots \Pi_r(x) \quad (\mod p)$$

where $\Pi_i(x), 1 \leqslant i \leqslant s$, are distinct irreducible polynomials of degree s_i. Then $s = s_1 + s_2 + \ldots + s_r = \sum_{i=1}^{r} s_i$

Let the message space consists of $\{m(x)\}$ where $m(x)$ is a polynomial over Z/pZ of degree less than s. Then the set of polynomials of degree less than s and relatively prime to $f(x)$ form a multi-plicative group modulo $f(x)$. [Two polynomials are said to be relatively prime if their greatest common divisor has degree 0 (ie, a constant polynomial). That is, two polynomials h_1 and h_2 are relatively prime if $ah_1 + bh_2 = 1$ for some polynomials a, b in $Z/pZ [x]$].

Hence

$$(m(x))^{order} \equiv 1 \mod d (p, f(x))$$

The order of the group is determined as follows.

First let us consider a slightly different case where

- 203 -

$f(x) = \Pi_1{}^{g_1}(x)$, a non square free polynomial (although it is said earlier that $f(x)$ consists of distinct irreducible factors mod p). Here the degree of the message polynomial $m(x)$ is less than the degree of $\Pi_1{}^{g_1}(x)$, that is, less than $g_1 s_1$. In addition, if the greatest common divisor of the degrees of $m(x)$ and $\Pi_1(x)$ is greater than 0, then

$$m(x) = \Pi_1(x) \cdot h(x)$$

where the degree of $h(x)$ is less than $g_1 s_1 - s_1 = s_1(g_1 - 1)$.
Total number of polynomials $\{m(x)\}$ of degree less than $g_1 s_1$ is equal to $|F|^{g_1 s_1} = p^{g_1 s_1}$. Within this set, the number of polynomials $m(x)$ which are not relatively prime to $\Pi_1(x)$ (that is, which are multiples of $\Pi_1(x)$ and for which $h(x)$ exists) is equal to the number of such polynomials $h(x)$ possible. There are $|F|^{s_1(g_1-1)}$ such polynomials $h(x)$ and hence there are $|F|^{s_1(g_1-1)}$ polynomials $m(x)$ which are multiples of $\Pi_1(x)$. Therefore, the order of the group is given by

$$|F|^{s_1 g_1} - |F|^{s_1(g_1-1)}$$

Thus if

$$f(x) = \prod_{i=1}^{r} \Pi_i{}^{g_i}(x) \quad (\text{mod } p)$$

then, the order of the group becomes

$$\prod_{i=1}^{r} \left(|F|^{g_i s_i} - |F|^{s_i(g_i-1)} \right) \tag{10.18}$$

In this case $|F| = p$. Substituting this into the above expression gives

$$\prod_{i=1}^{r} \left(p^{g_i s_i} - p^{s_i(g_i-1)} \right) \tag{10.19}$$

In particular, if $g_i = 1$ for all $1 \leqslant i \leqslant r$, (that is, $f(x)$ consists of distinct irreducible factors) then the expression (10.19) becomes

$$\prod_{i=1}^{r} \left(p^{s_i} - 1 \right)$$

As will be seen later in Section 10.5.3, the polynomial $f(x)$ needs

to be square free for cryptographic application.

10.5.2.1 System Design and Operation

The system designer randomly chooses a large prime p and r distinct high-degree irreducible polynomials over Z/pZ, $\Pi_1(x)$ to $\Pi_r(x)$, with degrees s_1 to s_r.

The chosen number can be tested to check if it is a prime using the primality tests given in Section 10.2. A chosen polynomial can be tested to see if it is irreducible over Z/pZ as follows: From [56], a polynomial $P(x)$ of degree n is irreducible in Z/pZ if and only if $P(x) \mid (x^{p^n} - x)$ and $gcd(P(x), x^{p^{n_i}} - x) = 1$ for $1 \leqslant i \leqslant k$ where $n = u_1 \ldots u_k$ and u_i's are primes and $n_i = n/u_i$. Thus to test whether a chosen polynomial $P(x)$ with degree n is irreducible over Z/pZ, the designer computes $gcd(P(x), x^{p^{n_i}} - x)$ and tests whether it is equal to 1 for all i, $1 \leqslant i \leqslant k$. Berlekamp [56] shows that a randomly chosen polynomial of degree s_i is irreducible with probability $1/s_i$. Thus an average of s_i tries is required to find an s_ith degree irreducible polynomial.

Having selected the irreducible polynomials $\Pi_1(x)$ to $\Pi_r(x)$, the designer forms their product $f(x)$ which is of degree s where $s = \sum_{i=1}^{r} s_i$. The polynomial $f(x)$ and the prime p are then made public information. The r distinct irreducible factors of $f(x)$ are kept secret. The coding exponents e and d are calculated using

$$ed \equiv 1 \mod (order) \qquad\qquad (10.20)$$

$$where \quad order = \prod_{i=1}^{r} (p^{s_i} - 1) \qquad\qquad (10.21)$$

The public encryption key is $(e, p, f(x))$ and the secret decryption key is $(d, p, f(x))$. The message is divided into blocks of size s consisting of sequences of p-ary digits. Each such plaintext block is associated with a polynomial $m(x)$ of degree less than s. The encryption procedure then consists of raising the polynomial $m(x)$ to the power e using

$$c(x) \equiv (m(x))^e \mod (p, f(x))$$

and the decryption procedure is given by

$$m(x) \equiv (c(x))^d \mod (p, f(x))$$

- 205 -

The above scheme can be used as a public key system as the order is found to depend upon the degrees of the irreducible polynomial factors of the composite modulus polynomial thus providing the trapdoor property.

10.5.2.2 Security of The System

The above scheme is not as secure as the RSA system over integers or the matrix based RSA system considered earlier due to the following reasons.

Following Section 10.3, the main cryptanalytical attacks consist of finding the order (expression (10.21)). Once the order is found, the cryptanalyst can determine the secret decoding exponent d using (10.20). The order expression contains the degrees of the irreducible factors of f(x) and the prime p. As the prime p is made public, the security of the system relies on the difficulty of factorizing the polynomial f(x) into its irreducible factors. Furthermore, the same decoding exponent d works for all sets of $\Pi_i(x)$ for i = 1 to r with same degrees s_i, $1 \leq i \leq r$.

It is generally true that factorization of a polynomial over a finite field is not a hard problem in sharp contrast with the factorization problem of a large integer. Berlekamp [57] proposed an efficient algorithm for factoring polynomials in Z/pZ where p is prime. For large prime p, Knuth [45] suggests some modifications to the Berlekamp's procedure. Here the Berlekamp's procedure is briefly described. The basic strategy of Berlekamp's method of factoring a polynomial in Z/pZ is to translate the problem into that of solving a system of linear equations with coefficients in Z/pZ and finding greatest common divisors. Both of these steps can be done using known methods in a finite number of steps. The idea behind the algorithm is now briefly outlined.

Suppose that f(x) has degree s and suppose that a polynomial h(x) can be found in Z/pZ of degree greater than or equal to 1 but less than s such that

$$f(x) \ \Big| \ (h(x))^P - h(x)$$

If degree $(h(x)) = k \geq 1$, then $(h(x))^P - h(x) \neq 0$ for the coefficient of x^{pk} is non-zero. By Fermat's theorem, the polynomial $u^P - u$ has p roots in Z/pZ namely u = 0, 1, 2,..., p-1. Thus $u^P - u$ factors (mod p)

– 206 –

into

$$u^P - u = u(u-1)(u-2) \ldots (u-(p-1))$$

Setting $u = h(x)$ gives

$$(h(x))^P - h(x) = h(x)(h(x)-1)(h(x)-2) \ldots (h(x)-(p-1)) \text{ in } Z/pZ \quad (10.22)$$

Since $f(x)$ divides $(h(x))^P - h(x)$, $f(x)$ is the greatest common divisor of $f(x)$ and $(h(x))^P - h(x)$. Since $h(x)-v_1$ and $h(x)-v_2$ are relatively prime if $v_1 \neq v_2$, from (10.22) it is seen that

$$f(x) = \gcd \ (f(x), (h(x))^P - h(x))$$

$$= \prod_{j=0}^{p-1} \gcd \ (f(x), h(x) - j) \quad (10.23)$$

Each factor on the right hand side has degree at most that of $h(x)$ which is in turn less than s, the degree of $f(x)$. Thus there must be at least two non-trivial factors of $f(x)$ on the right hand side of (10.23), that is, at least two factors of f which have degree $\geqslant 1$ and hence (10.23) is a non-trivial factorization of $f(x)$.

To factor $f(x)$, thus one needs to find a polynomial $h(x)$ such that $f(x)$ divides $(h(x))^P - h(x)$. This is done by solving a set of linear equations for the coefficients of $h(x)$ as follows.

Suppose $h(x) = b_0 + b_1 x + \ldots + b_{s-1} x^{s-1}$ where $b_0, \ldots, b_{s-1}$ are the coefficients to be determined. To see whether $f(x)$ divides $(h(x))^P - h(x)$, consider first $(h(x))^P$. Using the equality

$$(a + b)^P = a^P + b^P \text{ over } Z/pZ$$

we have

$$(h(x))^P = b_0^P + b_1^P x^P + \ldots + b_{s-1}^P x^{(s-1)p}$$

Applying Fermat's theorem to each of $b_0, b_1, \ldots, b_{s-1}$, $b_i^P = b_i$ in Z/pZ for all i, and hence

$$(h(x))^P = b_0 + b_1 x^P + \ldots + b_{s-1} x^{(s-1)p} \quad (10.24)$$

Now using the division algorithm and dividing $f(x)$ into x^{ip} for $i = 0, 1, \ldots, s-1$, gives

$$- 207 -$$

$$x^{ip} = f(x) q_i(x) + r_i(x) \qquad (10.25)$$

where

$$r_i(x) = r_{i,0} + r_{i,1} x + \ldots + r_{i,s-1} x^{s-1} \qquad (10.26)$$

Substituting (10.25) into (10.24) yields

$$(h(x))^P = b_0 r_0(x) + b_1 r_1(x) + \ldots + b_{s-1} r_{s-1}(x)$$
$$+ (\text{multiple of } f(x))$$

Thus $f(x)$ divides $(h(x))^P - h(x)$ if and only if $f(x)$ divides the polynomial

$$b_0 r_0(x) + b_1 r_1(x) + \ldots + b_{s-1} r_{s-1}(x) - (b_0 + b_1 x \ldots +$$
$$b_{s-1} x^{s-1})$$

But this polynomial has degree $\leqslant$ s-1 hence is divisible by $f(x)$ (of degree s) if and only if it is equal to 0.

Thus $f(x)$ divides $(h(x))^P - h(x)$ if and only if the coefficients b_0, b_1, ... b_s of $h(x)$ satisfy

$$b_0 r_0(x) + b_1 r_1(x) + \ldots + b_{s-1} r_{s-1}(x)$$
$$- (b_0 + b_1 x + \ldots + b_{s-1} x^{s-1}) = 0 \qquad (10.27)$$

Collecting coefficients of 1, x, x^2,..., x^{s-1} in (10.27), s simultaneous linear equations in the s unknowns b_0, b_1,..., b_{s-1} are obtained. These equations can be solved for b_0, ..., b_{s-1} and coefficients of a polynomial $h(x)$ such that $f(x)$ divides $(h(x))^P - h(x)$ can be obtained.

To calculate the decoding exponent d in (10.20), one does not need to obtain the irreducible factors but only need to obtain their degrees. In particular, in a system where the modulus $f(x)$ consists of two distinct irreducible factors $\Pi_1(x)$ and $\Pi_2(x)$ (analogous to the RSA system over integers), one can follow a simple technique to obtain the degrees of the irreducible factors. Without loss of generality, it can be assumed that polynomials are all monic. From finite fields theory [56], it is known that in a field of order p, $x^{p^\ell} - x$ factors into the product of all monic irreducible polynomials whose degrees divide ℓ. Thus the cryptanalyst computes $h_\ell(x) = $ gcd $(f(x), x^{p^\ell} - x)$ for $\ell = 1, 2,..., s/2$, successively until he finds a $h_\ell(x) \neq 1$. In such an instance, $h_\ell(x) = \Pi_1(x)$ or $\Pi_2(x)$ or $f(x)$.

The first two cases give the degrees $\ell = s_1$ or s_2 and in the third case $\ell = s_1 = s_2$. Thus in all the three cases, the opponent has obtained the degrees of irreducible factors even though in the third case, he has not actually factorized $f(x)$. Thus he can calculate the order (10.21) and can determine the decoding exponent using (10.20).

To compute the gcd $(f(x), x^{p^\ell} - x)$ for $\ell = 1, 2, \ldots, s/2$ requires approximately

(i) $s/2$ operations of gcd $(f(x), x^{p^\ell} - x)$

(ii)   each such operation requires approximately $\log_2 p^{s/2}$ multiplications mod $f(x)$, giving a total of $O(s^2 \log_2 p)$ multiplications mod $f(x)$ in Z/pZ.

The encryption (or decryption) procedure requires a maximum of $\log_2 w$ multiplications mod $f(x)$ where $w = (p^{s_1}-1)(p^{s_2}-1) - 1$ because the maximum value of e (or d) is $(p^{s_1}-1)(p^{s_2}-1) - 1$. As $w < p^s$, the computational effort is $O(s \log_2 p)$ multiplications mod $f(x)$. Thus the work factor is approximately $O(s)$. Thus if the cryptanalyst is required to make about 2^{60} trials before finding the decoding exponent d, then the degree of $f(x) = s_1 + s_2 = s = 2^{60}$. In such a case, the length of the cipher polynomial also becomes 2^{60} introducing a long delay in both encrypting and decrypting devices.

Further Gait [59] points out a short cycling attack based on superenciphering any non-trivial polynomial $t(x)$ with encrypting exponent e equal to any non-trivial power of p. He shows that the period of the sequence of powers of $t(x)^e$ mod $f(x)$ is equal to $s_1 s_2$. It seems then the work required to recover the decoding exponent depends on the difficulty of factorizing $s_1 s_2$. Hence for the system to be secure $s_1 s_2$ needs to be very large. Short cycling attacks against public key systems are considered in Chapter 13.

10.5.3 An Improved Polynomial Based RSA System

Now let us extend this polynomial scheme further to the case where $f(x)$ is a composite polynomial over Z/mZ, where m is a composite number, in an attempt to increase the security of the above mentioned public key system.

Let

$$m = \prod_{i=1}^{\ell} p_i^{\alpha_i} \quad \text{where } p_i \text{ are primes} \tag{10.28}$$

and the degree of the polynomial $f(x)$ be s.

Then the message space consists of polynomials $\{m(x)\}$ whose degrees are less than s and whose coefficients are elements of the ring Z/mZ. The first step is to find the order of the group formed by the polynomials of degree less than s and relatively prime to $f(x)$ over Z/mZ. Let us denote this order using the Euler totient function symbol $\emptyset_m(f(x))$. From the expression derived for the finite field $F = Z/pZ$ in Section 10.5.2, the order of the group formed by the polynomials which are relatively prime to $f(x)$ in Z/pZ is given by

$$\emptyset_p(f(x)) = \prod_{i=1}^{r} (p^{s_i b_i} - p^{s_i(b_i-1)}) \tag{10.29}$$

where $f(x) \equiv \prod_{i=1}^{r} g_i^{b_i}(x) \pmod{p}$ and $g_i(x)$ are irreducible polynomials of degrees s_i in Z/pZ.

Note that the commutative ring $R=Z/pZ[x]/f(x)$ under consideration is isomorphic to $Z[x]/(p,f(x))$. Now let us consider the order of the group formed by the units in the ring $Z/p^t Z[x]/f(x) \cong Z[x]/(p^t, f(x))$. Let Θ be a surjective ring homomorphism between the rings given below

$$\Theta : \quad \frac{Z[x]}{(p^{t+1}, f(x))} \longrightarrow\!\!\!\!\!\longrightarrow \frac{Z[x]}{(p^t, f(x))}$$

If restricted to units, then there is a surjective homomorphism Θ' such that

$$\Theta' : \quad U\left(\frac{Z[x]}{(p^{t+1}, f(x))}\right) \longrightarrow\!\!\!\!\!\longrightarrow U\left(\frac{Z[x]}{(p^t, f(x))}\right)$$

The kernel of the mapping Θ' consists of those sets of units of the left hand side which are congruent to 1 mod $(p^t, f(x))$, that is,

kernel $\Theta' = \{h(x)\}$ such that $h(x) \equiv 1 \bmod (p^t, f(x))$

or,

$$h(x) = 1 + p^t v(x) + f(x) k(x)$$

and $h(x)$ is a unit in $\dfrac{Z[x]}{(p^{t+1}, f(x))}$. Let us now calculate the number of such classes mod $(p^{t+1}, f(x))$. The standard form for representatives of classes mod $(p^t, f(x))$ is given by

- 210 -

$$u_0 + u_1 x + \ldots + u_{s-1} x^{s-1} \quad \text{where } 0 \leqslant u_i < p^t.$$

The standard form for representatives of classes mod $(p^{t+1}, f(x))$ is given by

$$u_0 + u_1 x + \ldots + u_{s-1} x^{s-1} \quad \text{where } 0 \leqslant u_i < p^{t+1}$$

As $f(x)$ is same in both cases, $h(x) \equiv 1 + p^t v(x) \pmod{f(x)}$.

The number of such $v(x)$ possible in mod $(p^{t+1}, f(x))$ is p^s because:
$v(x) = v_0 + v_1 x + \ldots + v_{s-1} x^{s-1}$ and each coefficient v_i can take
values in the range 0 to $p-1$. If v_i is greater than $p-1$ then it will
get reduced mod p^{t+1} as $p^t . p = p^{t+1}$. Thus the total number of such
$v(x)$ possible is equal to $p^{\deg f(x)} = p^s$.

Therefore

$$\text{\# kernal } \theta' = p^s$$

Using group theory,

$$\frac{U\left(\dfrac{Z[x]}{(p^{t+1}, f(x))}\right)}{\text{Kernel } \theta'} \cong U\left(\dfrac{Z[x]}{(p^t, f(x))}\right)$$

and

$$\# U\left(\dfrac{Z[x]}{(p^{t+1}, f(x))}\right) = \# (\text{kernel } \bar{\theta}') \ \# U\left(\dfrac{Z[x]}{(p^t, f(x))}\right)$$

where # denotes the order.

Therefore,

$$\phi_{p^{t+1}}(f(x)) = p^s \, \phi_{p^t}(f(x)) \qquad t \geqslant 1$$

$$= p^{2s} \, \phi_{p^{t-1}}(f(x))$$

$$= \ldots = \ldots$$

and $\therefore \quad \phi_{p^t}(f(x)) = p^{(t-1)s} \, \phi_p(f(x)) \qquad (10.30)$

The next stage is to consider the case where $m = \prod_{i=1}^{\ell} p_i^{\alpha_i}$ and to
evaluate the required expression $\phi_m(f(x))$.

Using the Chinese Remainder Theorem it is seen that

$$\frac{Z[x]}{(m, f(x))} \cong \frac{Z[x]}{(p_1^{\alpha_1}, f(x))} \oplus \frac{Z[x]}{(p_2^{\alpha_2}, f(x))} \oplus \ldots \oplus \frac{Z[x]}{(p_\ell^{\alpha_\ell}, f(x))}$$

- 211 -

where $\oplus$ denotes the direct sum.

Thus

$$\emptyset_m(f(x)) = \prod_{i=1}^{\ell} \emptyset_{p_i^{\alpha_i}}(f_i(x)) \tag{10.31}$$

where $f_i(x) \equiv f(x) \pmod{p_i}$ for $i=1, 2\dots,\ell$.

Considering the factorization of $f_i(x)$ rather than $f(x)$ over Z, let the factorization of $f_i(x)$ be

$$f_i(x) = \prod_{j=1}^{r_i} g_{ij}^{b_{ij}}(x) \pmod{p_i} \tag{10.32}$$

where the degree of irreducible polynomial $g_{ij}^{b_{ij}}(x)$ is $s_{ij} b_{ij}$. Note that the upper limit of the product term in the expression (10.32) goes up to r_i, that is, it is a function of to which prime p_i the polynomial $f(x)$ is being factored. This is because in general $f(x)$ (mod p_i) will factorize into some r_i distinct irreducible polynomials as i varies.

Subsituting (10.30) into (10.31) yields

$$\emptyset_m(f(x)) = \prod_{i=1}^{\ell} p_i^{(\alpha_i - 1)s} \emptyset_{p_i}(f_i(x)) \tag{10.33}$$

From (10.29), $\emptyset_{p_i}(f_i(x))$ is given by

$$\emptyset_{p_i}(f_i(x)) = \prod_{j=1}^{r_i} (p_i^{s_{ij}b_{ij}} - p_i^{s_{ij}(b_{ij}-1)}) \tag{10.34}$$

Substituting (10.34) into (10.33) gives

$$\emptyset_m(f(x)) = \prod_{i=1}^{\ell} p_i^{(\alpha_i - 1)s} \prod_{j=1}^{r_i} (p_i^{s_{ij}b_{ij}} - p_i^{s_{ij}(b_{ij}-1)})$$

Substituting $m = \prod_{i=1}^{\ell} p_i^{\alpha_i}$, the above expression becomes

$$\emptyset_m(f(x)) = \left(\frac{m}{\prod_{i=1}^{\ell} p_i}\right)^s \prod_{i=1}^{\ell} \prod_{j=1}^{r_i} (p_i^{s_{ij}b_{ij}} - p_i^{s_{ij}(b_{ij}-1)}) \tag{10.35}$$

Although the general expression (10.35) for the order of the group formed by polynomials $\{m(x)\}$ which are relatively prime to $f(x)$ over Z/mZ has been derived, from cryptography point of view, only square free modulus m and square free $f(x)$ are allowed. If for instance, m is chosen to be a non-square free integer then there will be nilpotent residue classes mod $f(x)$ for any $f(x)$ and proper

- 212 -

decryption will not be possible in such cases. Similar arguments also apply to the case where $f(x)$ is non-square free. Consider the two examples given below which illustrate the two cases m and $f(x)$ non-square free.

Example 1 : non-square free m

Let $m = p_1^2 \cdot p_2 = 3^2 \cdot 5 = 45$ and let $f(x) = x$
$f(x) \ (\text{mod } 3) \equiv f_1(x) = x$
$f(x) \ (\text{mod } 5) \equiv f_2(x) = x$
$\emptyset_m(f(x)) = (\frac{45}{15}) \cdot 2 \cdot 4 = 24$
choosing $e = d = 5$ so that $ed \equiv 1 \ (\text{mod } 24)$

Let the message $m(x) = 6$. Then $(m(x))^{ed} \equiv 6^{ed} \equiv 36 \not\equiv 6 \ (\text{mod}(45,x))$.
The message $m(x) = 36$ gives $36^{ed} \equiv 36 \ (\text{mod}(45,x))$.
Thus there is ambiguous decryption as both messages 6 and 36 produce identical output after decryption.

Example 2 : non-square free $f(x)$

Let $m = p_1 \cdot p_2 = 3 \cdot 5 = 15$ and let $f(x) = 4x^2 + 3$
$f(x) \ (\text{mod } 3) \equiv f_1(x) \equiv x^2 \equiv (\pi_{11}(x))^2$
$f(x) \ (\text{mod } 5) \equiv f_2(x) \equiv 4x^2 + 3 \equiv \pi_{21}(x)$
$\emptyset_m(f(x)) = 24$
choosing $e = d = 5$ so that $ed \equiv 1 \ (\text{mod } 24)$

Let the message $m(x) = x$. Then $(m(x))^{ed} \equiv 6x \not\equiv x \ (\text{mod}(15, 4x^2 + 3))$.
The message $m(x) = 6x$ gives $(6x)^{ed} \equiv 6x \ (\text{mod}(15, 4x^2 + 3))$.
Thus there is ambiguous decryption as both messages x and 6x produce identical output after decryption.

Thus if m and $f(x)$ are made square free, then the expression (10.35) becomes

$$\emptyset_m(f(x)) = \prod_{i=1}^{\ell} \ \prod_{j=1}^{r_i} \ (p_i^{s_{ij}} - 1) \qquad\qquad (10.36)$$

Thus the order $\emptyset_m(f(x))$ not only depends on the degrees of the irreducible factors (s_{ij}) of $f(x)$ but also on the prime factors of the integer m. So if the original polynomial scheme is operated over Z/mZ, then this seems to give rise to a secure public key cryptosystem like

– 213 –

the RSA system over integers.

10.5.3.1 System Design and Operation

The designer randomly chooses large primes p_1 to p_ℓ for some $\ell, \ell \geqslant 2$ following the guidelines suggested in Section 10.2. He then needs to choose a modulus polynomial $f(x)$ with the property that $f_i(x) \equiv f(x) \pmod{p_i}$, $1 \leqslant i \leqslant \ell$, are square free polynomials of degree s. Two methods for designing such a polynomial $f(x)$ are now outlined.

In method 1, the designer chooses randomly irreducible polynomials $h_{ij}(x)$ of degrees s_{ij} over $Z/p_i Z$ for $i = 1$ to ℓ. As noted in Section 10.5.2.1, an average of s_{ij} tries is required to find an irreducible polynomial of degree s_{ij} in $Z/p_i Z$. The polynomials are chosen in such a way that the degrees of the factors modulo p_i for a given i add up to form s. That is, the distinct factors $h_{ij}(x)$, $1 \leqslant j \leqslant r_i$, are combined to form $f_i(x) \pmod{p_i}$ of degree s. Thus the designer now has the values s_{ij} and p_i for all $1 \leqslant i \leqslant \ell$ and $1 \leqslant j \leqslant r_i$.

In method 2, the designer randomly chooses polynomials $f_i(x)$ of degree s and accepts them if they are square free over $Z/p_i Z$ respectively. This is done by choosing a polynomial $f_i(x)$ and computing the gcd $(f_i(x), f_i'(x))$. If the gcd $(f_i(x), f_i'(x)) = 1$, then $f_i(x)$ is square free. Knuth [45] estimates that a randomly chosen polynomial $f_i(x)$ will be square free over $Z/p_i Z$ with a probability of $1 - 1/p_i$. Hence the expected number of trials for finding a square free polynomial $f_i(x)$ is less than 2 for each i. An alternative way of finding square free polynomials is to start off with irreducible polynomials $f(x)$ over Z. The discriminant of $f(x)$ can be calculated using standard formula [85, p 451] and the prime p is then chosen so that p does not divide the discriminant. Then $f(x)$ is square free over Z/pZ. This procedure can be repeated for $p_i, 1 \leqslant i \leqslant \ell$. Having obtained the square free polynomials $f_i(x)$ over $Z/p_i Z$, $1 \leqslant i \leqslant \ell$, through one way or other, the designer can use the Chinese Remainder Theorem to form a unique sth degree composite polynomial $f(x)$ over Z/mZ where $m = \prod_{i=1}^{\ell} p_i$ and $f(x) \equiv f_i(x) \pmod{p_i}$ for $1 \leqslant i \leqslant \ell$. The degrees of the irreducible factors of $f_i(x)$ over $Z/p_i Z$ can be determined using the polynomial factorization algorithm of Berlekamp (Section 10.5.2.2) or one of the modified techniques given by Knuth [45]. Thus the designer now has the values s_{ij} and p_i for all $1 \leqslant i \leqslant \ell$ and $1 \leqslant j \leqslant r_i$.

These s_{ij} and p_i values can be substituted into the

- 214 -

expression of $\phi_m(f(x))$ given by (10.36). Recall that for the encryption and decryption to work properly,

$$(m(x))^{1+k\phi_m(f(x))} \equiv m(x) \quad \text{modd } (m,f(x))$$

for any m(x) with coefficients in Z/mZ and whose degree is less than s. The coding exponents e and d can therefore be determined using

$$ed \equiv 1 \quad \text{mod } \phi_m(f(x))$$

where e and d are multiplicative inverse of each other mod $\phi_m(f(x))$. The public encryption key is given by (e, m, f(x)) and the secret decryption key is equal to (d, m, f(x)).

The message is divided into blocks of size s (degree of f(x)), each block consisting of integers over Z/mZ. Each such block is associated with a polynomial m(x) of degree less than s. The encryption procedure then consists of raising m(x) to the power e to form the cipher polynomial c(x) using

$$c(x) \equiv (m(x))^e \quad \text{modd } (m, f(x))$$

The decryption procedure uses

$$m(x) \equiv (c(x))^d \quad \text{modd } (m, f(x))$$

to obtain the message polynomial back.

10.5.3.2 Sy̲s̲t̲e̲m̲_I̲m̲p̲l̲e̲m̲e̲n̲t̲a̲t̲i̲o̲n̲

This extended polynomial based RSA system has been simulated on the Prime Computer. The encryption/decryption of message/cipher polynomials are performed using the Square and Multiply technique given in Section 10.4.7.1. The listing of the program is given in Appendix 14. The cipher polynomial is transmitted to the receiver by the sender in the form of a vector. The coefficients of the cipher polynomial are sent to receiver, separated by a space starting from the lowest power coefficient. The receiver reconstructs the cipher polynomial and decrypts it to recover the message. An example showing the various parameters involved is given below:

- 215 -

Example

Let $m = p_1 p_2 = 5.7 = 35$.   Hence $\ell = 2$

Let $f(x) = x^4 + x^2 + 1$

$f(x) \ (\bmod \ 5) \equiv f_1(x) = (x^2 + 4x + 1)(x^2 + x + 1)$

$f(x) \ (\bmod \ 7) \equiv f_2(x) = (x^2 + 5)(x^2 + 3)$

Therefore, $\phi_m(f(x)) = (5^2 - 1)^2 (7^2 - 1)^2 = 1327104$

The coding exponents e and d are chosen to be $e = 6005$ and $d = 221$ where $ed \equiv 1 \ (\bmod \ 1327104)$

Let the 35—ary representation of the message to be encrypted be (18 9 30 23), (4 21 13 7) where the message is broken into blocks of size 4. That is,

$m_1(x) = 4x^3 + 21x^2 + 13x + 7$ and

$m_2(x) = 18x^3 + 9x^2 + 30x + 23$

Encryption:

cipher polynomial $c_1(x) \equiv (4x^3 + 21x^2 + 13x + 7)^{221} \ \text{modd}(35, x^4+x^2+1)$

$ \equiv x^3+14x^2+12x+21$

cipher polynomial $c_2(x) \equiv (18x^3+9x^2+30x+23)^{221} \ \text{modd}(35, x^4+x^2+1)$

$ \equiv 8x^3+6x^2+20x+14$

Decryption:

Message polynomial $m_1(x) \equiv (x^3+14x^2+12x+21)^{6005} \ \text{modd}(35, x^4+x^2+1)$

$ \equiv 4x^3+21x^2+13x+7$

Message polynomial $m_2(x) \equiv (8x^3+6x^2+20x+14)^{6005} \ \text{modd}(35, x^4+x^2+1)$

$ \equiv 18x^3+9x^2+30x+23$

10.5.4 Discussion

Thus it is seen that the RSA cryptosystem can be extended to polynomial rings. The difficulty of factorization of a polynomial into its irreducible factors over a finite field does not in itself provide the necessary security required for a public key cryptosystem. It is necessary to incorporate the factorization of an integer into its primes by considering the modulus polynomial $f(x)$ over Z/mZ (m—square free composite integer) to enhance the security

- 216 -

of the system. This gives a system which is comparable in strength
to that of the RSA system over integers. Furthermore, from crypto-
graphy point of view, it is required that both the modulus polynomial
$f(x)$ and the modulus integer m be square free. This is very much
like the RSA system over integers in contrast to the matrix RSA
system discussed in Section 10.4.

10.6   Extension of RSA System to Matrix Rings with Polynomial
         Elements

In this section, the aim is to combine the two systems
discussed in Sections 10.4 and 10.5, thus extending the factorization
trapdoor concept to matrix messages containing polynomials over a
ring as their elements. The approach adopted in this section
parallels with that used in Section 10.4.2. If the ring of all nxn
matrices over R is considered, then it is seen that the ring contains
nilpotent elements when $n > 1$. As mentioned in Section 10.4.2, from
cryptography point of view, it is necessary to avoid such nilpotent
elements to satisfy the condition $M^{r+1} \equiv M$ for some $r > 0$. As in
Section 10.4.2, to begin with the set of non-singular matrices is
investigated; then the set of upper triangular matrices is considered.

10.6.1 Non-singular Matrices Over $R = Z[x]/(m,f(x))$
The matrix messages are represented as $M(x)$ instead of
M, to indicate that their elements are polynomials in a single
indeterminate x. Hence $M(x)$ is written as

$$M(x) = \begin{bmatrix} m_{11}(x) & \cdots & m_{1n}(x) \\ m_{12}(x) & \cdots & m_{2n}(x) \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ m_{n1}(x) & \cdots & m_{nn}(x) \end{bmatrix} \qquad (10.37)$$

The complete public key system using such messages is developed in
four separate stages.

10.6.1.1 Stage 1
In this stage, the elements $m_{ij}(x)$ in (10.37) for all i, j

- 217 -

PMC Exhibit 2117
Apple v. PMC
IPR2016-00755
Page 230

$1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$, are to belong to the ring $R_1 = \frac{Z/pZ[x]}{(\Pi_1(x))}$ where $\Pi_1(x)$ is an irreducible polynomial in Z/pZ of degree s_1 (more precisely, here R_1 is a field). That is, the group M_n formed by non-singular message matrices of order n with elements in R_1 is considered. In this instance, non-singularity implies that the determinant of $M(x)$ is relatively prime to p and $\Pi_1(x)$. The order of the group formed by such nxn non-singular matrices can be obtained in a similar fashion as in the case of non-singular matrices over Z/pZ discussed in Section 10.4.2.

The order of the group formed by these matrices is given by the expression below, where # denotes the order.

$$\# \ GL(n, \frac{Z/pZ[x]}{(\Pi_1(x))}) \ = \ \# \ GL \ (n, \ F_{p^{s_1}})$$

because

$$\frac{Z/pZ[x]}{(\Pi_1(x))} \quad \text{is a finite field of } p^{s_1} \text{ elements}$$

And

$$\# \ GL \ (n, \ F_{p^{s_1}}) = \left[(p^{s_1})^n - 1 \right] \left[(p^{s_1})^n - (p^{s_1}) \right] \cdots$$
$$\left[(p^{s_1})^n - (p^{s_1})^{n-1} \right] \quad (10.38)$$

using (10.4) in Section 10.4.2.

If such non-singular matrices are used as messages then one can form a conventional cryptographic system where the secret key contains the modulus p itself. The encrypting (e) and decrypting (d) exponents can then be determined using

$$ed \equiv 1 \mod (\# \ GL(n, F_{p^{s_1}}))$$

The encrypting key is therefore $(e, p, \Pi_1(x), n)$ and the decrypting key is $(d, p, \Pi_1(x), n)$. None of these keys can be made public and the encryption and the decryption procedures are given by

$$C(x) = (M(x))^e \ modd \ (p, \Pi_1(x))$$

and

$$M(x) = (C(x))^d \ modd \ (p, \Pi_1(x))$$

– 218 –

respectively where $M(x)$, $C(x) \in M_n(R_1)$

10.6.1.2 Stage 2

The above system can be modified to include the public key property as follows:

To begin with, the modulus polynomial is allowed to be equal to the product of distinct irreducible polynomials. This will result in a public key system analogous to the original polynomial scheme considered in Section 10.5.2 whose security depends upon the difficulty of factorizing a composite polynomial to irreducible factors. Let

$$f(x) = \prod_{i=1}^{r} g_i(x) \quad (\text{mod } p)$$

where the polynomials $g_i(x)$ for $1 \leqslant i \leqslant r$ are irreducible in Z/pZ. The polynomials $g_i(x)$ are of degrees s_i for $1 \leqslant i \leqslant r$ respectively.

The order of the group formed by the non-singular message matrices $M(x)$ with elements in the ring $R_2 = \dfrac{Z/pZ[x]}{(f(x))}$ can be obtained by the application of the Chinese Remainder Theorem as follows:

$$\# \, GL(n, \frac{Z/pZ[x]}{(f(x))}) = \# \, GL(n, \frac{Z/pZ[x]}{(g_1(x))} \oplus \cdots \oplus \frac{Z/pZ[x]}{(g_r(x))})$$

$$= \# \, GL(n, F_{p^{s_1}} \oplus \cdots \oplus F_{p^{s_r}})$$

$$= \prod_{i=1}^{r} \# \, GL(n, F_{p^{s_i}}) \tag{10.39}$$

Substituting (10.38) into (10.39) gives

$$\# \, GL(n, \frac{Z/pZ[x]}{(f(x))}) = \prod_{i=1}^{r} \left[(p^{s_i})^n - 1 \right] \left[(p^{s_i})^n - (p^{s_i}) \right] \cdots \left[(p^{s_i})^n - (p^{s_i})^{n-1} \right]$$

Thus

$$(M(x))^{order} \equiv 1 \quad \text{modd } (p, f(x))$$

and hence the coding exponents e and d can be calculated using

$$ed \equiv 1 \ (\text{mod } (\# \, GL(n, \frac{Z/pZ[x]}{(f(x))})))$$

and

$$(M(x))^{ed} \equiv M(x) \quad \text{modd}(p, f(x))$$

- 219 -

It is seen that in this system, the order depends on the degrees of the irreducible factors $g_i(x)$. (cf. Section 10.5.2). Thus this system can be used as a public key system with the public encryption key $(e,p,f(x),n)$ and the secret decryption key $(d,p,f(x),n)$. The encryption and decryption procedures are given by

$$C(x) \equiv (M(x))^e \text{ modd } (p,f(x))$$

and

$$M(x) \equiv (C(x))^d \text{ modd } (p,f(x))$$

respectively, where $M(x)$, $C(x) \in M_n(R_2)$.

The security of this system is dependent on the difficulty of factorizing the modulus polynomial $f(x)$ in Z/pZ. This is so because if the degrees of the irreducible factors of $f(x)$ can be found then the order of the group can be evaluated and hence the secret decoding exponent d can be determined. For the reasons mentioned in Section 10.5.2.2, this does not give rise to a secure public key system.

10.6.1.3 Stage 3

The next logical step is to consider the case where the modulus polynomial $f(x)$ consists of powers of irreducible polynomials as its factors, that is, $f(x)$ is a non-square free polynomial. Note that in contrast to the system considered in Section 10.5, here it is allowed to have a non-square free $f(x)$ because the nilpotent elements are being eliminated by considering only the non-singular matrices over the ring $R_3 = \frac{Z/pZ[x]}{(f(x))}$. Considering the factors of $f(x)$ in $Z/pZ[x]$ rather than in $Z[x]$, let

$$\bar{f}(x) \equiv f(x) \pmod{p}$$

and

$$\bar{f}(x) \equiv \prod_{i=1}^{r} g_i^{b_i}(x) \pmod{p}$$

where $g_i(x)$, $1 \leq i \leq r$, are irreducible polynomials in Z/pZ with degrees s_i respectively.

The order of the group formed by non-singular matrices $M(x)$ over the ring R_3 is evaluated as follows:

$$- 220 -$$

Using the Chinese Remainder Theorem,

$$\frac{Z/pZ[x]}{(\bar{f}(x))} \cong \frac{Z/pZ[x]}{(g_1^{b_1}(x))} \oplus \ldots \oplus \frac{Z/pZ[x]}{(g_r^{b_r}(x))}$$

Letting $T = Z/pZ[x]$ and $\rho_i = (g_i(x))$
then,

$$GL(n, T/_{\bar{f}(x)}) \cong GL(n, T/\rho_1^{b_1}) \times \ldots \times GL(n, T/\rho_r^{b_r})$$

where $\times$ denotes direct product.

To determine the order, $\# GL(n, T/\rho_i^{b_i})$, consider the sequence of surjective homomorphisms, θ_i.

$$\theta_i : GL(n, T/_\rho i) \longrightarrow\!\!\!\!\!\rightarrow GL(n, T/_\rho i-1) \qquad\qquad i \geqslant 2$$

Under such a mapping an $n \times n$ matrix $M(x)$ $(mod \rho^i)$ becomes $M'(x)$ $(mod \rho^{i-1})$ as shown below:

$$M(x) \ (mod \ \rho^i) \longrightarrow M'(x) \ (mod \rho^{i-1})$$

Using group theory, for such an onto mapping θ_i,

$$\# GL(n, T/_\rho i) = \# GL(n, T/_\rho i-1) \cdot \# (Kernel \ \theta_i)$$

The kernel θ_i consists of the set of matrices which are mapped to the identity matrix I in $(mod \rho^{i-1})$, that is,

$$m_{\ell j}(x) \equiv 0 \quad (mod \ \rho^{i-1}) \qquad\qquad \ell \neq j \qquad\qquad (10.40)$$

$$m_{\ell \ell}(x) \equiv 1 \quad (mod \ \rho^{i-1}) \qquad\qquad 1 \leqslant \ell \leqslant n \qquad\qquad (10.41)$$

There are $p^{\deg g_i(x)} = p^{s_i}$ possibilities for each of the equations (10.40) and (10.41) giving rise to $(p^{s_i})^{n^2}$ total possibilities. Hence,

$$\# GL(n, T/_\rho i) = (p^{s_i})^{n^2} \ \# GL(n, T/_\rho i-1)$$

$$= \ldots \qquad = \ldots$$

$-$ 221 $-$

Therefore,

$$\#\mathrm{GL}(n, T/_{\rho}b_i) = (p^{s}i)^{n^2(b_i-1)} \ \# \mathrm{GL}(n, T/\rho) \qquad (10.42)$$

But $T/\rho = \dfrac{Z/pZ[x]}{(g_i(x))}$ is a finite field of $p^{s}i$ elements.

Therefore using (10.38)

$$\# \mathrm{GL}(n, T/\rho) = [(p^{s}i)^n - 1] \ [(p^{s}i)^n - (p^{s}i)] \ \cdots \ [(p^{s}i)^n - (p^{s}i)^{n-1}]$$

Substituting this into (10.42) yields

$$\# \mathrm{GL}\left(n, \frac{Z/pZ[x]}{(g_i^{b_i}(x))}\right) = (p^{s}i)^{n^2(b_i-1)} \left[(p^{s}i)^n - 1\right] \cdots \left[(p^{s}i)^n - (p^{s}i)^{n-1}\right] \qquad (10.43)$$

Thus

$$\# \mathrm{GL}(n, \frac{Z/pZ[x]}{(f(x))}) = \prod_{i=1}^{r} \# \mathrm{GL}(n, \frac{Z/pZ[x]}{(g_i^{b_i}(x))}) \qquad (10.44)$$

It is necessary to consider one further stage to complete the design of a secure public key system using such non-singular matrices as messages.

10.6.1.4   Stage 4

The final stage is to consider the elements $m_{ij}(x)$ in the message matrix $M(x)$ in (10.37) to belong to the ring $R_4 = \dfrac{Z/mZ[x]}{(f(x))}$ where

$$m = \prod_{i=1}^{t} p_i \quad \text{and } p_i, \ 1 \leqslant i \leqslant t, \text{ are distinct primes}$$

and the factorization of $f(x)$ is given by

$$f(x) \ (\mathrm{mod} \ p_i) \equiv f_i(x) = \prod_{j=1}^{r_i} g_{ij}^{b_{ij}}(x) \quad \text{for } 1 \leqslant i \leqslant t$$

The degrees of irreducible polynomials $g_{ij}(x)$ over Z/p_iZ are s_{ij} respectively.

The order of the group formed by the $n \times n$ non-singular matrices $M(x)$ over the ring R_4, denoted by $\# \mathrm{GL}(n, \dfrac{Z/mZ[x]}{(f(x))})$, is evaluated as follows:

- 222 -

The ring $\dfrac{Z/mZ[x]}{(f(x))}$ is isomorphic to the ring $\dfrac{Z[x]}{(m,f(x))}$. Using the Chinese Remainder Theorem,

$$\frac{Z[x]}{(m,f(x))} \cong \frac{Z[x]}{(P_1,f(x))} \oplus \frac{Z[x]}{(P_2,f(x))} \oplus \cdots \oplus \frac{Z[x]}{(P_t,f(x))}$$

Therefore,

$$GL\left(n, \frac{Z[x]}{(m,f(x))}\right) \cong GL\left(n, \frac{Z[x]}{(P_1,f(x))}\right) \times \cdots \times GL\left(n, \frac{Z[x]}{(P_t,f(x))}\right)$$

But

$$GL\left(n, \frac{Z[x]}{(P_i, f(x))}\right) \cong GL\left(n, \frac{F_{P_i}[x]}{(f(x))}\right)$$

where F_{P_i} is a finite field of p_i elements.

The order $\#\, GL\left(n, \dfrac{F_{P_i}[x]}{(f(x))}\right)$ has already been evaluated and is given by (10.44) as

$$\#\, GL\left(n, \frac{F_{P_i}[x]}{(f(x))}\right) = \prod_{j=1}^{r_i} (p_i^{s_{ij}})^{n^2(b_{ij}-1)} \left[(p_i^{s_{ij}})^n - 1\right] \cdots \left[(p_i^{s_{ij}})^n - (p_i^{s_{ij}})^{n-1}\right]$$

Hence

$$\#\, GL\left(n, \frac{Z[x]}{(m,f(x))}\right) = \prod_{i=1}^{t} \prod_{j=1}^{r_i} (p_i^{s_{ij}})^{n^2(b_{ij}-1)} \left[(p_i^{s_{ij}})^n - 1\right] \cdots \left[(p_i^{s_{ij}})^n - (p_i^{s_{ij}})^{n-1}\right] \quad (10.45)$$

The order given by the expression (10.45) above shows that it is dependent on the degrees of the irreducible factors of $f(x)$, namely, s_{ij} b_{ij}, as well as on the prime factors of the modulus m, namely p_i, for all i and j. Thus when used as a public key system, this provides considerably more security than the system considered in Stage 2, provided that the modulus m is chosen large enough. The encryption and the decryption of messages are carried out in the normal fashion using

$$C(x) \equiv (M(x))^e \quad \text{modd } (m,f(x))$$

and

- 223 -

$$M(x) \equiv (C(x))^d \mod (m, f(x))$$

where $M(x), C(x) \in M_n (R_4)$

and $ed \equiv 1 \pmod{order}$

and the order is given by (10.45).

Note that this system is a combined version of the systems developed in Section 10.4 and 10.5.

10.6.2 Upper Triangular Matrices Over $R = \dfrac{Z[x]}{(m, f(x))}$

For the systems developed in the previous section, the message space is restricted to contain only non-singular matrices. But as mentioned in Section 10.4.2, the sender is faced with the problem of determining whether his message matrix is non-singular or not, to find whether it belongs to the message space. This poses problems as the sender has no control over the matrix elements but must accept what the plaintext dictates. Alternatively, as in Section 10.4.4, one can consider the set of upper triangular matrices with invertible elements along the diagonal as the message space. This makes the construction of non-singular message matrices easier especially when m is a product of a few large primes and $f(x)$ is a product of a few high degree irreducible polynomials. In such a situation one can almost arbitrarily choose the diagonal elements of the upper triangular message matrix to satisfy the condition that they are relatively prime to $f(x)$ modulo m. In particular, the diagonal elements can be chosen to be elements in the ring Z/mZ which are relatively prime to m. If necessary, one can also use the Euclid's algorithm to evaluate the gcd to test the relative primeness.

In this section, the set of upper triangular matrices whose elements are polynomials over a chosen ring R forms the message space of the developed cryptosystem. The four stages involved in developing the complete public key system, similar to those discussed in the previous section, are now considered in turn. Note that in all these stages, it is assumed that the diagonal elements are invertible elements over the chosen ring R. This ensures that the message matrix is invertible. A typical message matrix $M_\Delta(x)$ is written as

- 224 -

$$
M_{\Delta}(x) = \begin{bmatrix} m_{11}(x) & \cdot & \cdot & \cdot & & m_{1n}(x) \\ 0 & \cdot & & & & \cdot \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \bigcirc & & \cdot \\ \cdot & & & & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & 0 & m_{nn}(x) \end{bmatrix}
$$

where Δ indicates that the matrix is triangular.

10.6.2.1 Stage 1

In this stage, the message space consists of upper triangular matrices whose non-zero elements $m_{ij}(x)$ belong to the ring $R_1 = \dfrac{Z/pZ[x]}{(\Pi_1(x))}$ where $\Pi_1(x)$ is an irreducible polynomial in Z/pZ of degree s_1. (That is, R_1 is actually a field).

The order of the group formed by such upper triangular matrices over R_1 can be found as follows:

Each diagonal entry may be any one of the polynomials which is relatively prime to $\Pi_1(x)$ over Z/pZ. The number of such polynomials has already been evaluated in Section 10.5.2 and is given by the Euler totient function $\emptyset_p(\Pi_1(x))$ where

$$\emptyset_p(\Pi_1(x)) = (p^{s_1}-1)$$

The remaining $\frac{1}{2}n(n-1)$ superdiagonal entries of the upper triangular matrix may take any value in the field $\dfrac{Z/pZ[x]}{(\Pi_1(x))}$. Thus each of these entries has p^{s_1} possibilities as $\dfrac{Z/pZ[x]}{(\Pi_1(x))}$ is a finite field of p^{s_1} elements.

Thus the order of this group $U\left(n, \dfrac{Z/pZ[x]}{(\Pi_1(x))}\right)$ is given by

$$\# \, U\left(n, \frac{Z/pZ[x]}{(\Pi_1(x))}\right) = (p^{s_1}-1)^n \, (p^{s_1})^{n(n-1)/2}$$

This can be used to form a conventional cryptographic system where the secret coding exponents e and d can be found using

$$ed \equiv 1 \pmod{\# \, U\left(n, \frac{Z/pZ[x]}{(\Pi_1(x))}\right)}$$

- 225 -

and the encryption and the decryption procedures are given by

$$C_\Delta(x) \equiv (M_\Delta(x))^e \quad \text{modd } (p, \Pi_1(x))$$

and

$$M_\Delta(x) \equiv (C_\Delta(x))^d \quad \text{modd } (p, \Pi_1(x))$$

respectively, where $M_\Delta(x)$, $C_\Delta(x) \in U_n(R_1)$

10.6.2.2 Stage 2

The above system can be modified to include the public key property as follows:

Initially, it is assumed that the modulus polynomial consists of a product of distinct irreducible polynomials modulo p. That is,

$$f(x) = \prod_{i=1}^{r} g_i(x) \pmod{p}$$

where $g_i(x)$, $1 \leq i \leq r$ are irreducible in Z/pZ.

The polynomials $g_i(x)$ are of degrees s_i, $1 \leq i \leq r$, respectively.

The order of the group formed by upper triangular matrices over the ring $R_2 = \frac{Z/pZ[x]}{(f(x))}$ with invertible diagonal elements is evaluated as follows:

Using the Chinese Remainder Theorem,

$$\# U\left(n, \frac{Z/pZ[x]}{(f(x))}\right) = \# U\left(n, \frac{Z/pZ[x]}{(g_1(x))} \oplus \cdots \oplus \frac{Z/pZ[x]}{(g_r(x))}\right)$$

$$= \# U\left(n, F_{p^{s_1}} \oplus \cdots \oplus F_{p^{s_r}}\right)$$

Each of the diagonal entries can be any one of the polynomials relatively prime to $f(x)$ over Z/pZ and is given by $\phi_p(f(x))$ where

$$\phi_p(f(x)) = \prod_{i=1}^{r} (p^{s_i} - 1) \quad \text{(See Section 10.5.2)}$$

The remaining $\frac{n(n-1)}{2}$ superdiagonal entries have p^s possibilities each where

$$s = \sum_{i=1}^{r} s_i$$

Therefore the order is given by

– 226 –

Page 226

$$\# \, U \left(n, \frac{Z/pZ[x]}{(f(x))}\right) = \left[\phi_p \, (f(x))\right]^n \left[(p^s)\right]^{n(n-1)/2}$$

It is seen that in this system, the order depends on the degrees of the irreducible factors of the modulus polynomial $f(x)$ (cf Sections 10.5.2 and 10.6.1.2). Thus this system can be used as a public key system with the public encryption key $(e,p,f(x),n)$ and the secret decryption key $(d,p,f(x),n)$. The coding exponents e and d are calculated using

$$ed \equiv 1 \; \left(\text{mod} \, \# \, U \left(n, \frac{Z/pZ[x]}{(f(x))}\right)\right)$$

and

$$(M_\Delta(x))^{ed} \equiv M_\Delta(x) \quad \text{modd} \; (p,f(x))$$

where $M_\Delta(x) \in U_n(R_2)$

For the reasons mentioned in Section 10.5.2.2, this does not offer a secure public key system.

10.6.2.3 Stage 3

The next step is to consider the case where the modulus polynomial $f(x)$ consists of powers of irreducible polynomials as its factors, that is, $f(x)$ is a non-square free polynomial in Z/pZ.

Let

$$f(x) \equiv \prod_{i=1}^{r} g_i^{b_i}(x) \quad (\text{mod} \; p)$$

where $g_i(x)$, $1 \leqslant i \leqslant r$, are irreducible polynomials in Z/pZ with degrees s_i respectively.

The order of the group formed by upper triangular matrices over the ring $R_3 = \dfrac{Z/pZ[x]}{(f(x))}$ is now evaluated.

Using the Chinese Remainder Theorem,

$$\frac{Z/pZ[x]}{(f(x))} \cong \frac{Z/pZ[x]}{(g_1^{b_1}(x))} \oplus \dots \oplus \frac{Z/pZ[x]}{(g_r^{b_r}(x))}$$

and hence

$$- \; 227 \; -$$

$$\# \, U \, (n, \, \frac{Z/pZ[x]}{(f(x))}) \quad = \, \# \, U \, (n, \, \frac{Z/pZ[x]}{(g_1^{b_1}(x))}) \, \oplus \, \ldots \, \oplus \, \frac{Z/pZ[x]}{(g_r^{b_r}(x))})$$

Using arguments similar to the ones given in Section 10.6.2.2,

$$\# \, U \, (n, \, \frac{Z/pZ[x]}{(f(x))}) \quad = \, \left[\phi_p \, (f(x))\right]^n \, (p^s)^{n(n-1)/2}$$

where

$$s \, = \, \sum_{i=1}^{r} \, s_i b_i$$

and

$$\phi_p(f(x)) = \prod_{i=1}^{r} \, \left[p^{s_i b_i} - p^{s_i(b_i-1)}\right] \quad \text{(see Section 10.5.2)}$$

10.6.2.4 $\underline{Stage \; 4}$

Finally the group formed by the upper triangular matrices with invertible diagonal elements over the ring $R_4 = \frac{Z/mZ[x]}{(f(x))}$ is considered. The modulus $m = \prod_{i=1}^{t} p_i^{\alpha_i}$, where p_i, $1 \leqslant i \leqslant t$, are distinct primes and the factorization of $f(x)$ is given by

$$f(x) \, (\text{mod } p_i) \equiv f_i(x) = \prod_{j=1}^{r_i} \, g_{ij}^{b_{ij}}(x) \quad \text{for } 1 \leqslant i \leqslant t$$

where the degrees of the irreducible polynomials $g_{ij}(x)$ are s_{ij} respectively. Using arguments similar to the ones given in Section 10.6.2.2, the order of the group is given by

$$\# \, U \, (n, \, \frac{Z/mZ[x]}{(f(x))}) \quad = \, \left[\phi_m(f(x))\right]^n \, (m^s)^{n(n-1)/2}$$

where $\phi_m \, (f(x))$ is obtained from Section 10.5.3 as

$$\phi_m(f(x)) = \left(\frac{m}{\prod_{i=1}^{t} p_i}\right)^s \, \prod_{i=1}^{t} \, \prod_{j=1}^{r_i} \, (p_i^{s_{ij}b_{ij}} - p_i^{s_{ij}(b_{ij}-1)})$$

Note also that by substituting n=1 in the expression (10.45) gives

$$\# \, GL \, (1, \frac{Z[x]}{(m,f(x))}) \quad = \, \phi_m \, (f(x)) \qquad (\alpha_i = 1 \, , \, \forall \, i)$$

$$- \, 228 \, -$$

Thus the order is seen to be dependent on the degrees of the irreducible factors of $f(x)$ as well as on the prime factors of m. Thus this system can be used to provide a secure public key system provided the modulus m is chosen large enough (say 200 decimal digits, see Section 10.2). The encryption and the decryption procedures are given by

$$C_\Delta(x) \equiv (M_\Delta(x))^e \quad \text{modd } (m, f(x))$$

and

$$M_\Delta(x) \equiv (C_\Delta(x))^d \quad \text{modd } (m, f(x))$$

where $M_\Delta(x)$, $C_\Delta(x) \in U_n(R_4)$

and $ed \equiv 1 \pmod{\text{order}}$.

The public encryption key is given by $(e, m, f(x), n)$ and the secret decryption key is equal to $(d, m, f(x), n)$.

10.7   Discussion

In this chapter, the factorization trapdoor concept has been extended to some matrix and polynomial rings which are isomorphic to a direct sum of finite fields. This has resulted in a generalization of the RSA cryptostructure to matrix and polynomial ring message space. It is seen that some of these extended systems can be made at least as secure as the original RSA system over integers modulo m. Other features such as the use of non-square free moduli seem to be possible with some of these extended systems in contrast to the original RSA system over Z/mZ. Investigation of such systems indicate that factorization trapdoor structures required for the design of public key system can be found in rings other than the ring of integers modulo m. From a practical point of view, it seems that the high complexity of such systems may favour the implementation of the factorization trapdoor in the ring of integers.

- 229 -

C H A P T E R   11

FACTORIZATION TRAPDOOR FROM IDEAL POINT OF VIEW

11.1 General

In addition to considering the factorization trapdoor
system from an 'element' point of view, the trapdoor concept can
also be treated from the more general 'ideal' point of view. In
particular the integer, polynomial and matrix based RSA factorization
trapdoor schemes considered in the previous chapter are briefly
re-examined from the ideal point of view. Some of the principles
of the ideal theory are used in the next chapter in further extending
the trapdoor concept to such algebraic number fields as the ring
of Gaussian integers and some other quadratic fields.

To begin with, some basic definitions and principles of
the ideal theory are stated (without proofs) which will be required
in subsequent sections. A detailed treatment of ideal theory can be
found in a number of mathematical textbooks in particular in [60, 61].

11.2 Basic Concepts

11.2.1 Ideal

A set J of one or more elements of a ring R is called an
ideal in R if and only if it has the following properties:

(i)   If i and j are elements of J, then $i \pm j$ is an element of
J.

(ii)   If i is an element of J, then for every element r of R,
ir and ri are elements of J.

11.2.2 Congruence

Let J be an ideal. Two elements a and b are defined to be
congruent modulo the ideal J if a–b is in J, denoted by $a \equiv b \pmod{J}$.

11.2.3 Principal Ideal

Let R be a commutative ring with 1 and 'a' be a non-zero
element of R. If A is any ideal which contains the element 'a', then

– 230 –

A must also contain all the elements of the form ra where $r \in R$. An ideal generated by a single element 'a' of R is called a principal ideal and is denoted by <a> . An integral domain in which all ideals are principal is called a principal ideal domain (PID).

11.2.4 Prime Ideal

If R is a commutative ring with 1, an ideal ρ in R is said to be a prime ideal if and only if $ab \equiv 0 \pmod{\rho}$ implies that $a \equiv 0 \pmod{\rho}$ or $b \equiv 0 \pmod{\rho}$. An alternative definition of prime ideal is that it is an ideal ρ other than the unit ideal with the property that for any two ideals A and B if $\rho | AB$ then $\rho | A$ or $\rho | B$. Note that an ideal ρ is said to 'divide' an ideal A if there exists an ideal C such that $A = \rho C$.

11.2.5 Product of Ideals

The product AB of two ideals A and B is defined as the ideal C 'generated by all products' ab where $a \in A$ and $b \in B$.

11.2.6 Unique Factorization of Ideals

Every ideal in a Dedekind domain [60] can be factored into the product of a finite number of prime ideals and this representation is unique.

11.2.7 Factorization Trapdoor

Using the ideal factorization theorem (Section 11.2.6), let the decomposition of a non-square free ideal A be

$$A = \rho_1^{s_1} \ldots \rho_r^{s_r}$$

The number of residues modulo the ideal A is given by the norm of the ideal, N(A). The number of invertible residue classes modulo the ideal A is denoted by $\bar{\emptyset}(A)$ in a similar manner to the Euler totient function $\emptyset$.

Theorem 1

If A and B are relatively prime ideals then

$$\bar{\emptyset}(AB) = \bar{\emptyset}(A) \cdot \bar{\emptyset}(B)$$

- 231 -

Page 231

The congruences $a \equiv c \pmod{A}$ and $a \equiv d \pmod{B}$ establish a one to one correspondence $a \rightleftarrows (c,d)$ between the residues 'a' prime to AB and the pairs (c,d) whose two members c,d range over the residues prime to A and B respectively. Hence the theorem 1 follows.

Theorem 2

If ρ is a prime ideal then,

$$\bar{\phi}(\rho^S) = (N\rho)^S \{1 - 1/N\rho\}$$
$$= (N\rho - 1)(N\rho)^{S-1}$$

A complete proof is given in [61]. But it can be proved using a method similar to that which is employed to calculate the Euler totient function $\phi(p^S)$ where p is a prime in Z. The complete system of residues with respect to ρ is represented using $N\rho$ integers $0,1,\ldots p-1$. Of these only 0 is not prime to p. Hence $\bar{\phi}(\rho) = N\rho - 1 = N\rho(1 - 1/N\rho)$. As $N(AB) = N(A)N(B)$ for any two ideals A and B, it is seen that $N(\rho^2) = (N\rho)^2$. With respect to ρ^2 there are $p^2 - p$ incongruent classes that are relatively prime to ρ and hence $\bar{\phi}(\rho^2) = (N\rho)^2 [1 - \frac{1}{N\rho}]$. Use of induction, gives Theorem 2.

Using theorems 1 and 2, if

$$A = \rho_1^{S_1} \ldots \rho_r^{S_r}$$

then

$$\bar{\phi}(A) = N(A) \prod_{i=1}^{r} (1 - \frac{1}{N\rho_i}) \qquad (11.1)$$

The expression (11.1) can be used to give the generalized version of Fermat's theorem for ideals, namely,

if 'a' is prime to an ideal A then $a^{\bar{\phi}(A)} \equiv 1 \pmod{A}$

If A is a prime ideal, ie, $A = \rho$, then

$$a^{N\rho - 1} \equiv 1 \pmod{\rho}$$

Thus the relatively prime residues modulo A form a group of order $\bar{\phi}(A)$.

- 232 -

Some of these concepts are now applied to the specific cases already looked at such as the ring integers, the ring of polynomials and matrix rings.

11.3   Ring of Integers

Let Z be the ring of integers. If m is a fixed integer in the ring Z then the ideal $<m>$ contains just the integers which are divisible by m.

Theorem 3

In the ring Z, every ideal is a principal ideal.

Let A be an ideal in Z. If A is the zero ideal then A is the principal ideal $<0>$. If $A \neq <0>$ let m be the smallest integer in A. If n is an integer then using Euclid's algorithm, we have

$$n = qm + r$$

where q and r are integers and $0 \leqslant r < m$.

Now if n is an element of A, then n−qm is also in A. That is, r is in A. If r is greater than zero, then it contradicts the assumption that m is the least positive integer. Therefore r = 0, that is, n = qm. So all elements of A are of this form and hence $A = <m>$.

The prime ideals in Z are therefore precisely the ideals $<p>$ where p is a prime (together with the ideal $<0>$ and the ideal Z).

If the unique factorization of m into primes over Z is given by

$$m = p_1 \ldots p_r \quad \text{for some } r \geqslant 2 \quad (\text{m assumed to be square free})$$

then one can view the ideal decomposition of $<m>$ into prime ideals $<p_i>$ as

$$<m> = <p_1> \ldots <p_r>$$

The order of the group formed by relatively prime residues modulo $<m>$ is given by the Euler totient function for the ideal $<m>$, $\phi<m>$, where

– 233 –

$$\phi \langle m \rangle = \phi \langle p_1 \rangle \cdots \phi \langle p_r \rangle$$

and ·

$$\phi \langle p_i \rangle = N \langle p_i \rangle - 1$$

$N \langle p_i \rangle$ is the norm of the ideal $\langle p_i \rangle$ and is equal to the number of residue classes modulo $\langle p_i \rangle$ given by p_i . Hence

$$\phi \langle m \rangle = (p_1 - 1) \cdots (p_r - 1)$$

and

$$a^{\phi \langle m \rangle} \equiv 1 \pmod{\langle m \rangle}$$

This gives the familiar RSA system over the integers looked at from the ideal point of view.

11.4   Polynomial Rings

If F is a field and x is an indeterminate, then again every ideal in $F[x]$ is a principal ideal. The proof is very much similar to the one given for the ring of integers Z (cf. Theorem 3). In this case, $m(x)$ is chosen to be a polynomial of least degree in a given ideal and the Euclidean division algorithm for polynomials is used. Hence a non-trivial ideal A of $F[x]$ is of the form $A = \langle f(x) \rangle$ where $f(x)$ is a non-zero monic polynomial of minimal degree in A.

The prime ideals in this ring are those which are generated by irreducible polynomials $f(x)$ over F. If $f(x)$ is composite, then let the unique factorization of $f(x)$ into irreducible polynomials $g_i(x)$ be

$$f(x) = g_1^{b_1}(x) \cdots g_r^{b_r}(x)$$

where the degree of polynomial $g_i(x)$ is s_i.

Then one can view the decomposition of the ideal generated by $f(x)$ into ideals generated by $g_i^{b_i}(x)$ as

- 234 -

$$\langle f(x) \rangle = \langle g_1^{b_1}(x) \rangle \cdot \langle g_2^{b_2}(x) \rangle \ldots \langle g_r^{b_r}(x) \rangle$$

From Theorem 2, we have

$$\mathscr{G}(\rho^b) = (N_\rho)^{b-1} (N_\rho - 1)$$

Here

$$N\langle g_i(x) \rangle = \text{ number of elements in the residue class ring } F[x]/\langle g_i(x) \rangle$$

$$= \text{ number of polynomials in } F[x] \text{ of degree less than } s_i$$

$$= |F|^{s_i}$$

Hence

$$\mathscr{G}\langle g_i^{b_i}(x) \rangle = |F|^{s_i(b_i-1)} \left\{ |F|^{s_i} - 1 \right\}$$

$$= |F|^{s_i b_i} - |F|^{s_i(b_i-1)}$$

This expression is same as the one obtained earlier in Section 10.5.2 by considering elements in the field.

The order of the group formed by residue classes modulo $\langle f(x) \rangle$ is obtained using Theorem 1,

$$\mathscr{G}\langle f(x) \rangle = \mathscr{G}\langle g_1^{b_1}(x) \rangle \ldots \mathscr{G}\langle g_r^{b_r}(x) \rangle$$

$$= \prod_{i=1}^{r} \left\{ |F|^{s_i b_i} - |F|^{s_i(b_i-1)} \right\}$$

11.5 Matrix Rings

Let us now reconsider the matrix system discussed in Section 10.4 to see whether it is possible to improve it using some ideal theory principles.

The ring of $n \times n$ matrices over a field F, $M_n^*(F)$, has no non-trivial ideals. The only ideals are the zero ideal and the whole ring itself. Hence one cannot consider the ring $M_n^*(F)$ for our purpose. One can consider two possible alternatives.

(a) The first approach is to consider a subring, say the ring of $n \times n$ upper triangular matrices (including the diagonal

---

M_n^* - See note on p. 243   — 235 —

elements) over F, $U_n(F)$. This ring has many ideals. For instance,

$$J = \left\{ \begin{pmatrix} 0 & ? \\ 0 & 0 \end{pmatrix} \right\} \text{ is an ideal in } U_n(F)$$

? - denotes any arbitrary elements in F.

      (b)   The second approach is to consider matrices over a commutative ring R rather than a field. Then there is one to one correspondence between the two-sided ideals in R and the two-sided ideals in $M_n^*(R)$. This may give rise to a trapdoor coding system in $M_n^*(R)$. As will be seen later, the system obtained using this approach is same as that already discussed in Section 10.4.2.

11.5.1 Approach (a)

      Consider the ring of 2x2 upper triangular matrices with arbitrary diagonal elements over a finite field, $R = U_2(F_p)$ where p is a prime. Let J be an ideal generated by an element M. Then J is the smallest ideal containing the element and by definition is equal to

$$J = \left\{ \sum_i r_i M r_i^{'} \quad \begin{array}{l} \text{all finite combinations} \\ \text{as } r_i, r_i^{'} \text{ vary over R} \end{array} \right\}$$

The element M must be chosen to be non-invertible otherwise it is possible to choose $r_i = M^{-1}$ and $r_i^{'} = I$ (identity). This results in the ideal J containing $(M^{-1} M I) = I$. Hence the ideal must contain $r_i I$ for all $r_i \in R$. That is, J = R. This contradicts the assumption that J is a proper subset of R. Hence no element of the ideal must possess multiplicative inverse. Let us consider X to be a generator matrix in $U_2(F_p)$ where X is equal to

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

The ideal J is

$$J = \left\{ \sum \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u & v \\ 0 & w \end{pmatrix} \right\}$$

where a, b, c, u, v, w $\in$ Z/pZ, p prime.
That is,

- 236 -

$$J = \left\{ \sum \begin{pmatrix} au & av \\ 0 & 0 \end{pmatrix} \right\}$$

Let $x = au$ and $y = av$, $x, y \in Z/pZ$, then

$$J = \left\{ \sum \begin{pmatrix} x & y \\ 0 & 0 \end{pmatrix} \right\} \tag{11.2}$$

It is seen that if $r \in R$ and $j \in J$, then

$$rj = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} x & y \\ 0 & 0 \end{pmatrix} \in J$$

$$jr = \begin{pmatrix} x & y \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u & v \\ 0 & w \end{pmatrix} \in J$$

The number of elements in the ring $R = U_2(Z/pZ)$ is equal to p^3 as each of a, b and c has p possible choices. The order of the ideal J is p^2. Hence the order of the quotient ring R/J is equal to p. Thus the ring R/J is isomorphic to Z/pZ. From group theory, in general, the order of the ideal J is a factor of the order of the ring R. If the ring $R = U_2(Z/mZ)$ where m is equal to product of primes is now considered, then R/J is isomorphic to the ring Z/mZ. If m is square free, R/J forms a trapdoor system.

Let X be a generator matrix of the form given below

$$X = \begin{pmatrix} g & 0 \\ 0 & 0 \end{pmatrix} \tag{11.3}$$

If gcd $(g,m) = 1$, then the ideal J becomes $\left\{ \begin{pmatrix} x & y \\ 0 & 0 \end{pmatrix} \right\}$ which is same as the one given in (11.2). Hence assume that gcd $(g,m) = \ell \neq 1$. For instance, m could be equal to the product of two large primes p and q and some ℓ, that is, $m = \ell.p.q$. Then g could be equal to $\ell.s$ where gcd $(s,p) = 1$ and gcd $(s,q) = 1$. The ideal J then consists of elements of the form given below

$$J = \left\{ \begin{pmatrix} gx & gy \\ 0 & 0 \end{pmatrix} \right\}$$

- 237 -

where x, y ε T = Z/mZ

The residue classes of the quotient ring R/J are of the form

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} = \begin{pmatrix} a \ (\text{mod } g) & b \ (\text{mod } g) \\ 0 & c \ (\text{mod } m) \end{pmatrix}$$

Proposition 1

The number of elements in the quotient ring R/J is given by

$$\# R/J = m\ell^2$$

Proof

If two members of the ring R are congruent to each other modulo J, that is,

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \equiv \begin{pmatrix} a' & b' \\ 0 & c' \end{pmatrix} \ (\text{mod } J)$$

where a, b, c, a', b', c' ε T = Z/mZ

then this implies that

$$a \equiv a' \ (\text{mod } g)$$
$$b \equiv b' \ (\text{mod } g)$$
$$c \equiv c'$$

The number of possible choices for the element c is equal to m.
Let k be equal to the number of possible choices for elements a or b.
Then k is equal to the number of residue classes mod gT which are distinct in T. In other words, k is equal to #T/gT. Consider the mappings

$$Z \xrightarrow{\ \alpha\ } T \longrightarrow T/gT$$

and

$$Z \xrightarrow{\ \theta\ } T/gT$$

Then

$$T/gT \cong Z/\text{Kernel } \theta$$
$$Z/\text{Kernel } \theta = Z/\{x \in Z; \ \alpha(x) \in gT\}$$
$$= Z/\{x \in Z; \ x \equiv gy \ (\text{mod } m) \text{ is soluble for}$$
$$y \in Z\}$$
$$= Z/\{x \in Z; \ x \equiv 0 \text{ mod } (\gcd \ (g,m))\}$$

- 238 -

That is,

$$T/gT \cong Z/\ell Z \quad \text{where } \ell = \gcd(g,m)$$

$$\therefore \quad \#T/gT = \ell$$

Hence the number of residue classes of the ring R/J is $m\ell^2$.

As the order does not depend on the factorization of the modulus m, there is no trapdoor as such yet. But let us now consider the invertible residue classes modulo the ideal J, that is, the diagonal elements a and c are chosen to be invertible in the ring T/gT and T respectively.

Proposition 2

The number of invertible residue classes modulo J is given by

$$\phi(\ell) \cdot \ell \cdot \phi(m)$$

Proof

The invertible residue classes of the ring R/J are of the form

$$\begin{pmatrix} a \ (\text{mod } g) & b \ (\text{mod } g) \\ 0 & c \end{pmatrix}$$

where a, b, c ϵ T = Z/mZ
and such that a is invertible (mod g) and c is invertible (mod m). The number of invertible residue classes mod m is given by the Euler totient function $\phi(m)$. Hence the number of possible choices for the element c is equal to $\phi(m)$. The element b can be arbitrarily chosen (mod g) and the number of possible choices is equal to ℓ from Proposition 1 above. Let k be equal to the number of possible choices for the element a. From above, it is known that $T/gT \cong Z/\ell Z$. Therefore, k is equal to the number of invertible elements in $Z/\ell Z$, that is, $\phi(\ell)$. Hence the number of invertible residue classes in $R/J = \phi(\ell) \cdot \ell \cdot \phi(m)$.

Thus in practice if m is equal to the product of a *few* large primes
and ℓ which is itself equal to the product of a *few* reasonable size
prime integers, then the elements a and c ($\neq$0) can be chosen arbitrarily
provided they are relatively small compared to m and ℓ .

The above arguments can be extended to the case where the
ideal J is of the form

$$
J = \left\{ \begin{pmatrix} g^{x_1}, & g^{x_2}, & \cdots & g^{x_n} \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{pmatrix} \right\} \tag{11.4}
$$

The order of the quotient ring R/J, that is, the number of residue
classes modulo J is now given by

$$
\ell^n \, m^{n \, (n-1)/2}
$$

The number of invertible residue classes modulo J is given by

$$
\text{Ord.} = \phi(\ell) \, \{\phi(m)\}^{n-1} \, \ell^{n-1} \, m^{(n-2)(n-1)/2} \tag{11.5}
$$

11.5.1.1 Choice_of_Generator_Matrix_

When deciding on a generator matrix X, the following points
must be taken into account.

(i)   The smaller the generated ideal J, the larger the quotient
          ring R/J. That is, the smaller the order of the ideal J
          is, the greater the number of residue classes in R/J thus
          giving rise to a larger number of possible messages that
          can be used in the system.

– 240 –

(ii)   For the encryption and decryption procedures to be relatively easy, the generator matrix must have a simple form; hence the choice of the matrix X in (11.3). For instance, another simple generator matrix X is given by

$$X = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

The ideal J is then given by

$$J = \left\{ \begin{pmatrix} 0 & x \\ 0 & 0 \end{pmatrix} \right\}$$

where $x \in Z/mZ$

Hence the residue classes of R modulo J are of the form

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \equiv \begin{pmatrix} a & 0 \\ 0 & c \end{pmatrix} \quad (\text{mod } J)$$

That is,

$$R/J \cong Z/mZ \oplus Z/mZ$$

This is equivalent to 2 scalar RSA prototype systems.

(iii)   More generally, one can use more than one generator matrix to generate the ideal J but this makes the whole system more complicated without increasing the security.

$$J = \left\{ \sum \begin{pmatrix} \text{arbitrary} \\ \text{element} \\ \text{in R} \end{pmatrix} \left(\underbrace{X_1 \dots X_k}_{\text{Generator matrices}} \right) \begin{pmatrix} \text{arbitrary} \\ \text{element} \\ \text{in R} \end{pmatrix} \right\}$$

11.5.1.2 Sy̲s̲t̲e̲m̲_̲D̲e̲s̲i̲g̲n̲

A factorization trapdoor system can therefore be constructed as follows:

The message space consists of nxn invertible upper triangular matrices (including the diagonal elements) modd (m, J) , where $m = \prod_{i=1}^{r} p_i^{a_i}$ and the ideal J is given by (11.4). The encryption procedure raises the message matrix to the power e giving the cipher

- 241 -

matrix C as

$$C \equiv M^e \pmod{(m,J)}$$

The decryption procedure is given by

$$M \equiv C^d \pmod{(m,J)}$$

The coding exponents e and d can be determined using

$$ed \equiv 1 \pmod{ord.}$$

where ord. is given by (11.5).

The public encryption key is given by (e,m,n,J) and the secret decryption key is (d,m,n,J).

As the order depends on the structure of m, the security of the system again lies in the difficulty of factorization of the modulus m. But note that if the size of ℓ is is made large then although the opponent needs to factorize ℓ to be able to calculate $\phi(\ell)$ in (11.5), the factorization of m and hence the computation of $\phi(m)$ is made that much easier. This is because as J is made public, the opponent knows ℓ, the gcd (m,g) and hence he achieves partial factorization of $\frac{m}{\ell} = pq$. Larger the size of ℓ smaller the value of $\frac{m}{\ell}$.

Thus it seems that the process of performing modulo an ideal as indicated above does not appear to increase the security of the trapdoor system but instead disguises further the basic trapdoor system.

11.5.2 Approach (b)

Let us first state an important result about two-sided ideals in complete matrix rings. For proof refer to [62].

If M is a two-sided ideal in the ring R, then the ring M_n^* of all matrices of order n over M is a two-sided ideal in the ring M_n^* of all matrices of order n over R.

In our case, R = Z and assume m = pq where p and q are distinct primes in Z. Then $<m>$, $<p>$ and $<q>$ are ideals in Z and $M_n^*(mZ)$, $M_n^*(pZ)$ and $M_n^*(qZ)$ are all ideals of $M_n^*(Z)$ using the above

- 242 -

result. Further the ideal factorization of $<m>$ in R is given by

$$<m> \; = <p> \, . \, <q>$$

Using the one-to-one correspondence between the two-sided ideals in R and in $M_n^*(R)$, gives

$$J_m \; = \; J_p \, . \, J_q$$

where J_p, J_q and J_m denote the ideals formed by $M_n^*(pZ)$, $M_n^*(qZ)$ and $M_n^*(mZ)$ in the ring $M_n^*(Z)$.

Although a trapdoor system is possible in Z/mZ when m is square free, there is no corresponding system in the case of $M_n^*(R)$. This is due to the fact that the quotient ring given by $M_n^*(Z/mZ)$,

$$\frac{M_n^*(Z)}{M_n^*(mZ)} \; \cong \; M_n^*(Z/mZ) \cong M_n^*(Z/pZ) \; \oplus \; M_n^*(Z/qZ)$$

is not isomorphic to a direct sum of finite fields whereas the quotient ring $Z/<m>$ is isomorphic to $Z/<p>$ and $Z/<q>$. As seen earlier in Section 10.4.2, $M_n^*(Z/pZ)$ is not a finite field as it contains nilpotent elements. Thus a corresponding trapdoor system is possible if and only if the nilpotent elements are eliminated. This has already been considered and it gave rise to the trapdoor system in matrix rings discussed in Section 10.4.2.

Note: $M_n^*(R)$ – Ring of all nxn matrices over R

$M_n(R)$ – Ring of non-singular nxn matrices over R

FACTORIZATION TRAPDOOR IN ALGEBRAIC NUMBER FIELDS

12.1   General

The ring of Gaussian integers is initially considered with
a view to extending the factorization trapdoor concept to algebraic
number fields other than rational integers Z.   Then the more general
quadratic fields are briefly investigated.

12.2   Factorization Trapdoor System in Gaussian Integers

12.2.1 Ring of Gaussian Integers

Before considering the design of the factorization trapdoor
system in this ring, it is useful to describe very briefly some of
the properties of Gaussian integers. A detailed treatment of
Gaussian integers can be found in [63, 64, 65].

Let $i = \sqrt{-1}$ and consider the set of complex numbers $Z[i]$
defined by $\{a+bi \mid a,b \in Z\}$. This set is closed under addition and
subtraction. Moreover if $a+bi$, $c+di \in Z[i]$, then $(a+bi)(c+di) =
ac+adi+bci+bdi^2 = (ac-bd) + (ad+bc)i \in Z[i]$. Thus $Z[i]$ is closed under
multiplication and is a ring.

The norm of an element, $\alpha = a+bi$, in $Z[i]$ is defined to be
equal to $\alpha \cdot \bar{\alpha}$ where $\bar{\alpha}$ is the complex conjugate of α. That is,
$N\alpha = (a+bi)(a-bi)$ and hence $N\alpha = a^2+b^2$. Further it is seen that
$N(\alpha\beta) = (N\alpha) \cdot (N\beta)$ as $N(\alpha\beta) = (\alpha\beta) \cdot (\overline{\alpha\beta}) = (\alpha \bar{\alpha})(\beta \bar{\beta})$.

An integer in $Z[i]$, α, is called a unit if 1 is divisible by
α. Hence if α is a unit both α and $1/\alpha$ are integers in $Z[i]$.

Lemma 1

The norm of a unit is 1 and any integer whose norm is 1 is
a unit.

If α is a unit then $\alpha \mid 1$, that is, $1 = \alpha\beta$ and so $1 =
N\alpha N\beta$. This means that $N\alpha \mid 1$ and hence $N\alpha = 1$. If $\alpha = a+bi$, then
$1 = a^2+b^2 = (a+bi)(a-bi)$, that is, $(a+bi) \mid 1$. So $(a+bi)$ is a unit.
The units in $Z[i]$ are ± 1, $\pm i$ as the only solutions of $a^2+b^2 =1$ are
$a = \pm 1$, $b = 0$; $a = 0$, $b = \pm 1$.

A prime Π is an integer, not 0 or a unit, divisible only

- 244 -

by the numbers associated with itself or with 1. In $Z[i]$, a prime has no divisors except the eight trivial ones, namely, $1, \Pi, -1, -\Pi, i, i\Pi, -i$ and $-i\Pi$.

Lemma 2

An integer whose norm is a rational prime is also a prime. For suppose that $N\alpha = p$ and that $\alpha = \beta_1 \beta_2$ where $\alpha, \beta_1, \beta_2 \in Z[i]$. Then $p = N\alpha = N\beta_1 N\beta_2$. Hence either $N\beta_1 = 1$ or $N\beta_2 = 1$ and either β_1 or β_2 is a unit; and therefore α is a prime.

Lemma 3

If Π is a prime in $Z[i]$, then it divides exactly one positive rational prime p.

$N\Pi = \Pi \bar{\Pi}$ and so $\Pi | N\Pi$. Let the prime decomposition of $N\Pi$ in Z be $N\Pi = p_1 \cdots p_r$ where p_i's $(1 \leq i \leq r)$ are distinct positive primes. Then $\Pi | p_1 \cdots p_r$. That is, Π divides one of the primes p_i. It cannot divide two primes p_j and p_k. If so, then one can find two rational integers ℓ_j and ℓ_k using Euclid's algorithm such that $\ell_j p_j + \ell_k p_k = 1$. If $\Pi | p_j$ and $\Pi | p_k$ then $\Pi | 1$. So Π is a unit not a prime contrary to the hypothesis.

Lemma 4

Any integer, not zero or unit, is divisible by a prime and can be expressed as a product of primes.

If α is an integer, not a prime, then

$$\alpha = \beta_1 \beta_2, \quad N\beta_1 > 1, \quad N\beta_2 > 1, \quad N\alpha = N\beta_1 N\beta_2 \text{ and } 1 < N\beta_1 < N\alpha$$

If β_1 is not a prime, then

$$\beta_1 = \beta_3 \beta_4, \quad N\beta_3 > 1, \quad N\beta_4 > 1, \quad N\beta_1 = N\beta_3 N\beta_4 \text{ and } 1 < N\beta_3 < N\beta_1$$

This process can be continued so long as β_r is not a prime. Since $N\alpha, N\beta_1, N\beta_2 \cdots$ form a decreasing sequence of positive rational integers, this must come to a prime β_r. Then $\alpha = \beta_2 \beta_1 = \beta_2 \beta_3 \beta_4 = \cdots = \beta_2 \beta_3 \cdots \beta_r$ and so $\beta_r | \alpha$. Therefore now it is assumed that α is divisible by a prime Π_1 and $\alpha = \Pi_1 \alpha_1$ and the above process can be repeated for α_1.

- 245 -

In Z[i], the representation of α as a product of primes is unique as in Z (except from trivial variations).

The unique factorization of integers in Z[i] is equivalent to the principality of all ideals in it [63]. That is, Z[i] is a principal ideal domain. The argument to show this is very much similar to the one used to prove all ideals are principal in the ring of integers Z. Here instead of choosing the least positive number, the element of least positive norm is used. The prime ideals in Z[i] are therefore the ideals generated by primes in Z[i].

12.2.2 Design of Trapdoor Coding System in Z[i]

As before, $\alpha \equiv \beta$ mod $< \gamma >$, where $\alpha, \beta, \gamma \in Z[i]$, is defined to imply that $\alpha - \beta$ is in the principal ideal $< \gamma >$, that is, $\alpha - \beta$ is divisible by γ.

Fermat's theorem in Z[i] can be stated as follows:
If Π_1 and Π_2 are relatively prime then

$$\Pi_1^{\phi<\Pi_2>} \equiv 1 \mod <\Pi_2> \tag{12.1}$$

where the Euler totient function $\phi<\Pi_2> = N\Pi_2 - 1$

Now if an ideal $<m>$ generated by a Gaussian integer m is considered whose decomposition is

$$m = \Pi_1 \ldots \Pi_r$$

where Π_i for $1 \leqslant i \leqslant r$ are distinct primes in Z[i], then the number of invertible residue classes modulo $<m>$ is given by $\phi<m>$

$$\phi<m> = \phi<\Pi_1> \cdot \phi<\Pi_2> \cdots \phi<\Pi_r> \tag{12.2}$$

In order to calculate the order $\phi<m>$, one needs to compute $N\Pi_i$ for all i.

Using Lemma 3, let the Gaussian prime Π divide a rational prime p, ie, $\Pi | p$. Then $N\Pi | Np$. But $Np = p^2$. Therefore $N\Pi = p$ or $N\Pi = p^2$. That is, if $\Pi = a + bi$, then $a^2 + b^2 = p$ or p^2.

Case 1 : p $\equiv$ 3 (mod 4)

– 246 –

Since p is odd, one of a or b must be even, the other odd. Otherwise, the sum of their squares would be even. Let a = 2x and b = 2y + 1. If $a^2 + b^2 = p$ then

$$p = 4 x^2 + (2y + 1)^2$$
$$= 4(x^2 + y^2 + y) + 1$$
$$\equiv 1 \pmod 4$$

whereas to begin with, the assumption was $p \equiv 3 \pmod 4$. So in this case, only $a^2 + b^2 = p^2$ is possible and $N\Pi = Np = p^2$. That is, the rational prime p stays prime in Z[i].

Case 2 : $p \equiv 2 \pmod 4$

p = 2 is the only prime which falls into this class and from cryptography point of view, this case is not interesting. ($N\Pi = 2$).

Case 3 : $p \equiv 1 \pmod 4$

Here p is of the form 4k + 1 where k is any rational integer. Then $p | n^2 + 1$ for some rational integer n. But $(n^2 + 1) = (n + i)(n - i)$ and as $\Pi | p$, Π divides $(n + i)(n - i)$. But p does not divide $(n + i)$ or $(n - i)$ for otherwise one of $n/p \pm 1/p$ i would be a Gaussian integer; this cannot be possible as $1/p$ is not a rational integer. Hence Π and p are not associated and $N\Pi \neq Np$. So $a^2 + b^2 \neq p^2$ and hence only $a^2 + b^2 = p$ is possible, ie $N\Pi = p$.

The system designer first chooses primes $p_1, p_2, \ldots, p_r$ randomly for some $r \geq 2$ such that $p_j \equiv 1 \pmod 4$ or $p_j \equiv 3 \pmod 4$, for $1 \leq j \leq r$. Then he computes the norm of each of the Gaussian primes Π_j using $N\Pi_j = p_j$ or p_j^2. The order $\emptyset_{<m>}$ can then be calculated using

$$\emptyset <m> = \prod_{j=1}^{r} (N\Pi_j - 1) \qquad (12.3)$$

The coding exponents e and d required for encryption and decryption procedures can be determined using

$$ed \equiv 1 \pmod{\emptyset <m>} \qquad (12.4)$$

The public encryption key consists of (e, m) and the secret decryption key is (d, m) where m ϵ Z[i]. As the designer is required to make

– 247 –

m public, to obtain m, he needs to solve equations of the form

$$a_j^2 + b_j^2 = p_j \text{ or } p_j^2 \quad \text{for } 1 \leqslant j \leqslant r$$

and $\Pi_j = a_j + b_j i.$

Consider the problem of finding a and b in $a^2 + b^2 = p$ or p^2 given p (system designer knows p). The subscript j has been dropped for convenience.

Case 1 : $a^2 + b^2 = p^2$ where $p \equiv 3 \pmod 4$

This implies that

$$a^2 + b^2 \equiv 0 \pmod p \tag{12.5}$$

Suppose $a \not\equiv 0 \pmod p$, then

$$1 + \left(\frac{b}{a}\right)^2 \equiv 0 \pmod p$$

Letting $X = b/a$, then

$$1 + X^2 \equiv 0 \pmod p$$

ie $X^2 \equiv -1 \pmod p \tag{12.6}$

For the congruence (12.6) to have a solution, -1 must be a quadratic residue modulo p (Section 12.3.2). That is, the Legendre Symbol $\left(\frac{-1}{p}\right)$ must be equal to 1.

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = 1$$

As $p \equiv 3 \pmod 4$, $p = 4k + 3$ for some rational integer k

       ie, $p-1 = 4k + 2$

Therefore ,

$$(-1)^{(p-1)/2} = (-1)^{2k + 1} = -1$$

Thus when $p \equiv 3 \pmod 4$, the congruence (12.5) cannot be solved for a and b if $a \not\equiv 0 \pmod p$. Thus 'a' must be equal to 0 (mod p). This implies that the only solutions of (12.5) that are possible are:

$$a = 0, \ b^2 = p^2; \ b = 0, \ a^2 = p^2$$

− 248 −

<u>Case 2 :</u> $a^2 + b^2 = p$ where $p \equiv 1 \pmod 4$ (12.7)

<u>Method 1</u>

Again assuming $a \not\equiv 0 \pmod p$ and letting $X = b/a$,

$$1 + X^2 \equiv 0 \pmod p \qquad (12.8)$$

For the above congruence to have a solution, -1 must be a quadratic residue modulo p, ie,

$$(-1)^{(p-1)/2} = 1$$

As $p \equiv 1 \pmod 4$, $p - 1 = 4k$ for some rational integer k. Therefore, $(-1)^{(p-1)/2} = 1$.

Hence there are solutions of (12.7) for which a, $b \not\equiv 0 \pmod p$. The solutions to the congruence (12.8) are given by

$$X = \pm \left[\frac{(p-1)}{2} \right]! \pmod p$$

This can be seen as follows:

$$\left\{ [\tfrac{p-1}{2}]! \right\}^2 \equiv (1.2 \ldots \tfrac{p-1}{2})(1.2 \ldots \tfrac{p-1}{2}) \pmod p$$

$$\equiv (1.2 \ldots \tfrac{p-1}{2})(-(p-1) \ldots -(\tfrac{p+1}{2})) \pmod p$$

$$\equiv (1.2 \ldots \tfrac{p-1}{2})(\tfrac{p+1}{2} \ldots p-1)(-1)^{(p-1)/2}$$

$$\left\{ [\tfrac{p-1}{2}]! \right\}^2 \equiv (p-1)! \equiv -1 \pmod p \text{ using Wilson's theorem [38].}$$

Hence $\{[\tfrac{p-1}{2}]!\}$ is a solution of (12.8).

As $b \equiv a X \pmod p$, 'a' is allowed to vary in the range 0 to $\sqrt{p}$ and the least positive residue $aX \pmod p$ is tested to see whether it is less than $\sqrt{p}$. If so, then this value can be used for b because then $0 < a^2 + b^2 < 2p$ and $a^2 + b^2 \equiv 0 \pmod p$ imply that $a^2 + b^2 = p$.

<u>Example</u>

Using the above method to find a and b such that

- 249 -

$$a^2 + b^2 = p = 29 \tag{12.9}$$

Then

$$X = \pm \ (\frac{29-1}{2}) \ !$$

$$X \equiv 12 \ (\text{mod } 29)$$

Considering $0 < a < \sqrt{29}$, ie, $0 < a \leqslant 5$, gives

$$b_1 \equiv 1.12 \equiv 12 \ (\text{mod } 29)$$
$$b_2 \equiv 2.12 \equiv 24 \ (\text{mod } 29)$$
$$b_3 \equiv 3.12 \equiv \ 7 \ (\text{mod } 29)$$
$$b_4 \equiv 4.12 \equiv 19 \ (\text{mod } 29)$$
$$b_5 \equiv 5.12 \equiv \ 2 \ (\text{mod } 29)$$

Hence $b = b_5 = 2$ is the only one which is less than 5 and therefore $a = 5$, $b = 2$ would satisfy (12.9).

For large prime p, this method does not appear to be feasible as one needs to evaluate $(\frac{p-1}{2})!$ and test values of 'a' upto $\sqrt{p}$. This requires of the order of $(\frac{p}{2}+\sqrt{p})$ multiplications (mod p) in the worst case.

<u>Method 2</u>

This method involves the use of Jacobsthal sum, $S(c)$,[66] in determining a and b in (12.7). $S(c)$ is given by

$$s(c) = \sum_{n=1}^{p} \ (\frac{n}{p}) \ (\frac{n^2 - c}{p})$$

where $p \equiv 1 \ (\text{mod } 4)$ and $c \not\equiv 0 \ (\text{mod } p)$. (Note that $s(c) = 0$ if $p \equiv 3 \ (\text{mod } 4)$).

Considering

$$\sum_{c=1}^{p} s^2(c) = \sum_{c=1}^{p} \sum_{m=1}^{p} (\frac{m}{p}) (\frac{m^2-c}{p}) \sum_{n=1}^{p} (\frac{n}{p}) (\frac{n^2-c}{p})$$

$$= \sum_{m=1}^{p} \sum_{n=1}^{p} (\frac{m}{p}) (\frac{n}{p}) \left\{ \sum_{c=1}^{p} (\frac{m^2-c}{p}) (\frac{n^2-c}{p}) \right\}$$

Consider the inner sum $\{ \ \}$.

$$- \ 250 \ -$$

if $m^2 \equiv n^2 \pmod{p}$, then it is equal to $\sum\limits_{c=1}^{p} (\frac{m^2-c}{p})^2 = p-1$

if $m^2 \not\equiv n^2 \pmod{p}$, then it is equal to -1.

$$\therefore \quad \sum_{c=1}^{p} s^2(c) = (p-1) \sum_{\substack{m=1 \\ m^2 \equiv n^2}}^{p} \sum_{n=1}^{p} (\frac{m}{p}) (\frac{n}{p}) - \sum_{\substack{m=1 \\ m^2 \not\equiv n^2}}^{p} \sum_{n=1}^{p} (\frac{m}{p}) (\frac{n}{p})$$

That is,

$$\sum_{c=1}^{p} s^2(c) = 2(p-1)(p-1) - \sum_{n=1}^{p} (\frac{n}{p}) (0 - \frac{n}{p} - (\frac{-n}{p}))$$

$$= 2(p-1)^2 + 2(p-1)$$

$$\sum_{c=1}^{p} s^2(c) = 2p(p-1)$$

But

$$\sum_{c=1}^{p} s^2(c) = (\frac{p-1}{2}) s^2(1) + (\frac{p-1}{2}) s^2(k)$$

where k is any quadratic non-residue [66].

That is, $2p(p-1) = \frac{p-1}{2} s^2(1) + \frac{p-1}{2} s^2(k)$

$$p = (\frac{s(1)}{2})^2 + (\frac{s(k)}{2})^2$$

Thus to calculate a and b in $a^2 + b^2 = p$, one finds s(1) where

$$s(1) = \sum_{n=1}^{p} (\frac{n}{p}) (\frac{n^2-1}{p})$$

and then computes $p - (\frac{s(1)}{2})^2$ giving

$$a^2 = (\frac{s(1)}{2})^2 \quad \text{and} \quad b^2 = p - (\frac{s(1)}{2})^2$$

Considering the same example as before to find a and b in $a^2 + b^2 = p = 29$, using this method

$$s(1) = \sum_{n=1}^{29} (\frac{n}{29}) (\frac{n^2-1}{29}) = 10$$

- 251 -

Page 251

$$(\frac{s(1)}{2})^2 = 25 \text{ hence } a = 5 \text{ and } b = \sqrt{p - (\frac{s(1)}{2})^2} = 2$$

Again this method may not seem to be attractive when p is large.

Method 3

Legendre's method [86] is based on the continued fraction expansion of $\sqrt{p}$. He showed that if p is of the form $4k + 1$, then the expansion of $\sqrt{p}$ is of the form

$$\sqrt{p} = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cdots \cfrac{1}{q_2 + \cfrac{1}{q_1 + \cfrac{1}{2q_0 + \cdots}}}}}$$

It is seen that there is a symmetrical part q_1, q_2, ..., q_2, q_1 followed by $2q_0$ and there is no central term. Now let c be the particular complete quotient which begins at the middle of the period, that is,

$$c = c_m = q_m + \cfrac{1}{q_{m-1} + \cdots \cfrac{1}{q_1 + \cfrac{1}{2q_0 + \cfrac{1}{q_1 + \cdots}}}}$$

This is a purely periodic continued fraction whose period consists of q_m, ..., q_1, $2q_0$, q_1, ..., q_m. Since this period is symmetrical, $\bar{c} = -\frac{1}{c}$, where $\bar{c}$ denotes the conjugate of c. c is now expressible in the form $c = \frac{b + \sqrt{p}}{a}$ where a and b are integers. The equation $c\bar{c} = -1$ gives $(\frac{b + \sqrt{p}}{a}) \cdot (\frac{b - \sqrt{p}}{a}) = -1$ or $a^2 + b^2 = p$.

Again consider the example $p = 29$. The process for developing $\sqrt{29}$ in a continued fraction is

$$\sqrt{29} = 5 + \frac{1}{c_1}$$

$$c_1 = \frac{1}{4}(5 + \sqrt{29}) = 2 + \frac{1}{c_2}$$

$$c_2 = \frac{1}{5}(3 + \sqrt{29}) = 1 + \frac{1}{c_3}$$

$$c_3 = \frac{1}{5}(2 + \sqrt{29}) = 1 + \frac{1}{c_4}$$

$$c_4 = \frac{1}{4}(3 + \sqrt{29}) = 2 + \frac{1}{c_5}$$

$$c_5 = 5 + \sqrt{29}$$

The continued fraction is 5, $\overline{2, 1, 1, 2, 10}$. The appropriate complete quotient to take is $c = c_3$ giving $a = 5$ and $b = 2$.

Method 4

The cryptographer can also design the system starting with primes Π_j in $Z[i]$ rather than primes p_j in Z. That is, he selects Gaussian integers Π_j, $1 \leqslant j \leqslant r$, arbitrarily by choosing say a_j to be an even integer and b_j to be an odd integer in $\Pi_j = a_j + b_j i$ so that $a_j^2 + b_j^2$ is always odd. Then he can test the norms $N\Pi_j$ for $1 \leqslant j \leqslant r$, to find whether they are primes or squares of primes in Z. The primality testing can be done using the probabilistic algorithm mentioned in Section 10.2. If $N\Pi_j$ is a prime or a square of a prime in Z, then Π_j must be a prime in $Z[i]$. If $N\Pi_j$ is not a prime or a prime square in Z, then he chooses another pair of a_j and b_j. As this procedure needs to be done only once by each user at the beginning, this could be a feasible approach especially when $r = 2$ as in the prototype RSA system.

Choice of primes

Although the designer can choose any primes of the form $p \equiv 3 \pmod 4$ or $p \equiv 1 \pmod 4$ (except the special ones such as the Mersenne primes), now it is shown that certain combinations will result in the easy factorization of the composite Gaussian integer m. If m is considered to be a product of two Gaussian primes Π_1 and Π_2 where $\Pi_1 | p_1$ and $\Pi_2 | p_2$, then there are four possible combinations. They are:

(i) $p_1 \equiv 1 \pmod 4$, $p_2 \equiv 1 \pmod 4$

(ii) $p_1 \equiv 3 \pmod 4$, $p_2 \equiv 3 \pmod 4$

(iii) $p_1 \equiv 1 \pmod 4$, $p_2 \equiv 3 \pmod 4$

(iv) $p_1 \equiv 3 \pmod 4$, $p_2 \equiv 1 \pmod 4$

Case (i) results in $\Pi_1 = a_1 + b_1 i$ and $\Pi_2 = a_2 + b_2 i$ where a_1, a_2, b_1, $b_2 \neq 0$ in Z and $m = \Pi_1 \Pi_2 \in Z[i]$. The order of the group formed by the invertible residue classes modulo $< m >$ is therefore equal to $\phi < m > = (N\Pi_1 - 1)(N\Pi_2 - 1) = (p_1 - 1)(p_2 - 1)$.

Case (ii) yields $\Pi_1 = a_1$ or $b_1 i$ and $\Pi_2 = a_2$ or $b_2 i$ as one of each pair (a,b) is equal to zero. Letting $m = a_1 a_2$, it is seen that m

— 253 —

is a rational integer and the order $\tilde{\phi}<m> = (p_1^2-1)(p_2^2-1)$.

Case (iii) and (iv) will yield an m of the form cd+cfi. As m is made public, the opponent can easily spot the factor c and hence factorize m.

Hence it is seen from cryptographic considerations that, only the schemes where both p_1 and p_2 are chosen to be of same type (either 1(mod 4) or 3(mod 4))will provide secure systems. This idea extends to all primes p_j, $1 \leqslant j \leqslant r$, when m is a product of r Gaussian primes.

12.2.3 Security of the System in Z[i]

As in the case of the prototype RSA system over Z, the publicly available information for the opponent consists of the encrypting exponent e and the composite modulus m. If the rational primes are chosen such that $p_j \equiv 3$(mod 4) for all j, then the modulus m can be made to be a rational integer and the norm $Nm = m^2$. If the primes are chosen such that $p_j \equiv 1$(mod 4) for all j, then m is of the form c+di $\in$ Z[i]. In this case, the opponent can easily calculate the norm $Nm = c^2+d^2$. Hence in either case, the security of the system essentially lies in the difficulty of factoring a large rational integer, the norm Nm. The problem of factoring Nm is similar to that of factoring the modulus m ($\in$ Z) in the prototype RSA system. Thus the security of this system is same as that of its predecessor. Once the norm Nm is factorized into r primes q_j, then the order $\tilde{\phi}<m>$ can be found using $\prod_{j=1}^{r}(N\Pi_j-1)$ where $N\Pi_j = q_j$ or q_j^2. Then the opponent can compute the secret decoding exponent d using (12.4). Note that the cryptanalyst does not need to know the Gaussian primes $\Pi_1,...,\Pi_r$ but only needs to know their respective norms. In other words, the opponent will be working over Z not over Z[i] and does not need to solve the equation $a^2+b^2 = p$ for a and b.

12.2.4 Representation of Messages and System Operation

The messages are to be represented using the residue classes modulo the ideal generated by the modulus m. The number of distinct messages possible is equal to the number of incongruent residues mod <m> and is given by the norm of the ideal < m >. In the case of a principal ideal domain $N<m> = Nm$, the norm of the element m.

. In performing encryption or decryption using this system in

– 254 –

Z[i], there is a problem with regard to the message representation.
First consider the process of reduction mod<m> which is required in
both encryption and decryption procedures. This can be carried out
using the Euclidean algorithm as Z[i] is a PID. Let $y = x^e$ where e
is the encrypting exponent and consider the operation y mod <m> where
x, y and m ϵ Z[i].

Using Euclid's algorithm, there exists two integers u and v in Z[i]
such that

$$y = um + v \quad \text{where } Nv < Nm$$

Consider

$$y/m = A + Bi \quad \text{where A and B are rational numbers.}$$

Choose rational integers s and t such that

$$|A-s| \leqslant \tfrac{1}{2}, \ |B-t| \leqslant \tfrac{1}{2}$$

This can be done by choosing s and t as rational integers nearest to
A and B respectively (see Figure 12.1). Now let $u = s+ti$ and $v = y-um$, then it is seen that $Nv < Nm$ as

$$|v| = |y-um| = |y-(s+ti)m|$$
$$= |m| \ |\tfrac{y}{m} - s - ti|$$

$$|v| = |m| \ |(A-s) + (B-t)i|$$
$$= |m| \ \{(A-s)^2 + (B-t)^2\}^{\tfrac{1}{2}}$$

$$\therefore \ |v| \leqslant |m| \ \{\tfrac{1}{2^2} + \tfrac{1}{2^2}\}^{\tfrac{1}{2}} < |m|.$$

Since $Nv = |v|^2 < |m|^2 = Nm$, the inequality is established.

Fig 12.1

– 255 –
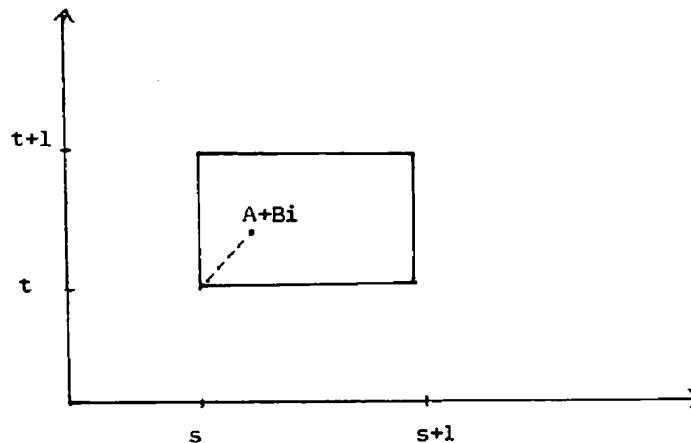
But the u and v are not uniquely determined. Thus if a message
$M = x+iy$ is encrypted to form $g+hi \equiv (x+iy)^e \bmod \langle m \rangle$, the decrypted
message, M' is given by

$$M' \equiv (g+hi)^d \bmod \langle m \rangle$$
$$\equiv k+\ell i \qquad \bmod \langle m \rangle$$

The decrypted message, $k+\ell i \equiv x+iy \bmod \langle m \rangle$ but it is of 'different
representation'. For instance, consider the example given below.

Let the modulus $m = 8+i \in Z[i]$

The norm, $\quad Nm = 65$

$$= N\Pi_1 \, N\Pi_2 \quad = 5.13$$

As $p_1 \equiv 1 \pmod 4$ and $p_2 \equiv 1 \pmod 4$, $\phi \langle m \rangle$ is given by

$$\phi \langle m \rangle = 4.12$$
$$= 48$$

Choosing $e = d = 7$ where $ed \equiv 1 \pmod{48}$, consider the message
$M = 5+3i$. Using the Euclidean algorithm,

$$(5+3i)^7 \equiv -2-i \bmod \langle m \rangle \quad \equiv g+hi$$

Thus

$$M' \equiv (-2-i)^7 \qquad \bmod \langle m \rangle$$

ie , $\qquad M' \equiv (-2-i)^7 \equiv -2-6i \bmod \langle m \rangle \equiv k+\ell i$

But

$$(-2-6i) \equiv 5+3i \equiv 2i-3 \bmod \langle m \rangle$$

The receiver would not be able to differentiate say between $-2-6i$ or
$5+3i$ or $2i-3$ although they are congruent to each other mod $\langle m \rangle$. Thus
there is not a unique representation of messages unlike in the case
of the RSA system over Z where a message is uniquely represented by
taking the least positive residue modulo m. In $Z[i]$, one approach
could be to use the norm of the modulus, Nm, to construct a standard
set of representatives. But this would not work as two elements α
and β can be congruent to each other and their norms $N\alpha$ and $N\beta$ be less
than Nm. Thus some form of standard set of representatives is essential.

Case 1

      First consider the case where the primes Π_j which form m
divide rational primes p_j of the form $p_j \equiv 1 \pmod 4$. Then the norm
is a square free rational integer given by

− 256 −

$$Nm = \prod_{j=1}^{r} p_j \qquad \text{where } m = \prod_{j=1}^{r} \Pi_j$$

The residue class ring $Z[i]/<m>$ is isomorphic to a direct sum of finite fields $Z[i]/<\Pi_j>$, ie,

$$Z[i]/<m> \cong Z[i]/<\Pi_1> \oplus \ldots \oplus Z[i]/<\Pi_r>$$

The field $Z[i]/<\Pi_j>$ contains $N\Pi_j = p_j$ elements. Therefore one standard method of representing the messages mod $<m>$ would be to use the integers in the ring Z/NmZ, that is, 0 to $Nm-1$. And every element of $Z[i]/<m>$ is congruent to an element in Z/NmZ. This is same as the message space of the RSA system over the rationals.

For example if $p_1 = 13$ and $p_2 = 5$, $p_1 \equiv p_2 \equiv 1 \pmod 4$ then $Nm = 65$ and $\phi <m> = 48$. The message space is therefore equal to $\{0, 1, \ldots 64\}$. The encryption and decryption processes are carried out using $C \equiv M^e \pmod{Nm}$ and $M \equiv C^d \pmod{Nm}$ where $ed \equiv 1 \pmod{\phi<m>}$. The coding exponent e and Nm are made public.

Now consider the case where the message space still consists of the integers in Z/NmZ but now the encryption and decryption procedures are calculated modulo m where $m = a+bi \in Z[i]$. In this case, e, m and Nm are made public. Let the message be $M \in Z/NmZ$, then the encryption procedure results in

$$M^e \bmod (a+bi) \equiv g+hi = cipher$$

Decryption produces

$$(g+hi)^d \bmod (a+bi) \equiv k+\ell i$$

That is, the recovered message M is equal to $k+\ell i$

$$M \equiv k+\ell i \pmod{a+bi} \qquad\qquad (12.10)$$

Conjugating both sides of (12.10)

$$M \equiv k-\ell i \pmod{a-bi} \qquad\qquad (12.11)$$

Using Chinese Remainder Theorem, the original M can be formed where

$$M \equiv \gamma_1 (k-\ell i) + \gamma_2 (k+\ell i) \pmod{(a+bi)(a-bi)}$$

where $\gamma_1 + \gamma_2 = 1$ and $M \in Z/NmZ$, $\gamma_1, \gamma_2 \in Z[i]$

A standard set of representatives of the ring $Z[i]/<m>$ can also be obtained using elementary divisor theory [65].

The ideal $<m>$ in $Z[i]$ is generated by the element $m = a+bi$

– 257 –

and it consists of the set of Gaussian integers

$$\langle m \rangle = \{ (a+bi)\lambda \} \quad \text{where } \lambda \; \varepsilon \; Z[i]$$

The ideal $\langle m \rangle$ can be regarded as a Z—module generated by [a+bi, ai—b] and the Z—basis of the ideal $\langle m \rangle$ is therefore given by (a+bi, ai—b). The integral basis of Z[i] is (1,i) [65].

Hence one can associate a matrix A with the pair of bases as follows

$$\text{Z—basis of ideal } \langle m \rangle = A \; (\text{Z—basis of } Z[i])$$

ie,

$$\begin{pmatrix} a + bi \\ ai - b \end{pmatrix} = A \begin{pmatrix} 1 \\ i \end{pmatrix}$$

where

$$A = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

Let

$$\underset{\sim}{t} = \begin{pmatrix} a + bi \\ ai - b \end{pmatrix} \quad \text{and} \quad \underset{\sim}{s} = \begin{pmatrix} 1 \\ i \end{pmatrix}$$

Therefore

$$\underset{\sim}{t} = A \; \underset{\sim}{s} \tag{12.12}$$

$\underset{\sim}{s}$ and $\underset{\sim}{t}$ can be replaced by $\underset{\sim}{w}$ and $\underset{\sim}{v}$ where $\underset{\sim}{w}$ and $\underset{\sim}{v}$ are equal to $\underset{\sim}{s}$ and $\underset{\sim}{t}$ multiplied by some unimodular matrices respectively.

Let

$$D = VAU \text{ where U and V are suitably chosen unimodular}$$

transformations such that $D = \text{diag}(d_1, d_2)$ and $d_1 \mid d_2$. That is,

$$\begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix} = VAU$$

Multiplying $\underset{\sim}{s}$ by U^{-1}, gives

$$\underset{\sim}{w} = U^{-1} \underset{\sim}{s}$$

and hence

$$\underset{\sim}{t} = AU \; \underset{\sim}{w} \tag{12.13}$$

Multiplying (12.13) by V gives

$$V\underset{\sim}{t} = VAU \; \underset{\sim}{w}$$

ie

$$\underset{\sim}{v} = D \; \underset{\sim}{w}$$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

— 258 —

The new basis of $Z[i]$ is (w_1, w_2) and the new basis of $\langle m \rangle$ is $(d_1 w_1, d_2 w_2)$.

Therefore, the residue classes mod $\langle m \rangle$ are represented as $x_1 w_1 + x_2 w_2$ where x_i goes from 0 to $d_i - 1$ for $i = 1, 2$.

Hence the system designer is required to reduce the matrix A to its diagonal form. The steps involved in this reduction algorithm of an nxn matrix A are now considered [67].

First Stage of Reduction

The aim is to reduce the nxn matrix A to an equivalent nxn matrix C of the special form

$$
C = \begin{bmatrix} \begin{array}{c|ccc} d_1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & C^* & \\ \vdots & & & \\ 0 & & & \end{array} \end{bmatrix} \tag{12.14}
$$

where d_1 divides each entry of C^*.

A finite sequence of elementary row and column operations is considered which when performed on A either yields a matrix of the form (12.14) or else leads to an nxn matrix $B = (b_{ij})$ satisfying the condition

$$
b_{11} < a_{11} \tag{12.15}
$$

In the latter case, one goes back to the beginning and applies the sequence of operations again. Either the form (12.14) is achieved in which case, this stage ends or (12.15) is reached in which case the leading entry is reduced still further and the process continues. After a finite number of steps, the form (12.14) will be reached.

The sequence of operations is as follows:
If A is the zero matrix, then it is already of the form (12.14); otherwise, A has a non-zero entry and by suitable interchanges of rows and columns this can be moved to the leading position. Therefore assume $a_{11} \neq 0$ and consider the following three possibilities:

Case (i)

There is an entry a_{1j} in the first row such that $a_{11} \nmid a_{1j}$. By the properties of Euclidean domain,

$$
a_{1j} = a_{11} q + r
$$

– 259 –

where either $r = 0$ or $r < a_{11}$

Since $a_{11} \nmid a_{1j}$, one must have $r \neq 0$ and so $r < a_{11}$. Subtracting q times the first column from the jth column and then interchanging the first and the jth columns, the leading entry a_{11} is replaced by r and (12.15) is achieved.

Case (ii)

There is an entry a_{i1} in the first column such that $a_{11} \nmid a_{i1}$. In this case, proceeding as in Case(i) but operating with rows instead of columns, (12.15) is reached.

Case (iii)

a_{11} divides every entry in the first row and first column. In this case, by subtracting suitable multiples of the first column from the other columns, one can replace all the entries in the first row other than a_{11} itself by zeroes. Similarly, subtracting multiples of the first row from the others, a matrix is obtained which is of the form,

$$D = \begin{bmatrix} a_{11} & 0 \ \ldots \ 0 \\ 0 & \\ \vdots & D^* \\ 0 & \end{bmatrix}$$

If a_{11} divides every entry of D^*, (12.14) has been reached; if not there is an entry, say, d_{ij} such that $a_{11} \nmid d_{ij}$. In that case, by adding the ith row to the top row leads to case (i).

Repeated application of these procedures will result in the form (12.14) after a finite number of steps, thereby completing the first stage of reduction.

End of Reduction

By applying the above process to the submatrix C^*, one can reduce its size still further, leaving a trail of diagonal elements. Any elementary operation on C^* corresponds to an elementary operation on C which does not affect the first row and column. Also any elementary operation on C^* gives a new matrix whose entries are linear combinations of the old ones and hence the new entries will therefore still be divisible by d_1. Thus $d_1 \mid d_2 \mid \ldots \mid d_n$.

- 260 -

In particular in the case of a 2×2 matrix A, the unimodular transformations can be found as follows

Let $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $b > a$ $\quad a \nmid \gcd(b,c,d)$

It is first shown that the matrix A is equivalent to a matrix of the form

$$\begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix} \quad \text{where} \quad a' \mid \gcd(b',c',d')$$

Using induction on a: the case $a = 1$ is trivial; when $a > 1$ and $a \nmid b$, choosing q so that $0 < aq + b < a$

and consider

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} aq+b & * \\ * & * \end{pmatrix}$$

where the leading element is a positive integer less than a. If $a \mid b$ and $a \nmid c$, then choosing q' such that $0 < aq' + c < a$ and consider

$$\begin{pmatrix} q' & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} aq'+c & * \\ * & * \end{pmatrix}$$

where the leading element is once again a positive integer less than a. Finally if $a \mid \gcd(b,c)$ but $a \nmid d$, let $c = c_1 a$ so that

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -c_1 & 1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & (1-c_1)b+d \\ * & * \end{pmatrix}$$

and $a \nmid \{(1-c_1)b+d\}$ which reduces back to the case when $a \nmid b$. The inductive argument is now complete. Now $a' \mid \gcd(b',c',d')$. Letting $b' = a'b''$, $c' = a'c''$ and $d' = a'd''$ and considering

$$\begin{pmatrix} 1 & 0 \\ -c'' & 1 \end{pmatrix} \begin{pmatrix} a' & a'b'' \\ a'c'' & a'd'' \end{pmatrix} \begin{pmatrix} 1 & -b'' \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a' & 0 \\ 0 & a'(d''-b''c'') \end{pmatrix} \quad (12.16)$$

the desired result is obtained.

With $A = \begin{pmatrix} a & +b \\ -b & a \end{pmatrix}$, (12.16) reduces to

$$D = \begin{pmatrix} a' & 0 \\ 0 & a'(1+b''^2) \end{pmatrix}$$

– 261 –

Further as $d_1 \mid d_2$ and $Nm = d_1 d_2$ where $Nm = \prod_{j=1}^{r} P_j$, the only allowed values for d_1 and d_2 are 1 and Nm respectively. So the messages can be represented as $x_2 w_2$ where x_2 goes from 0 to $Nm-1$. In this case, the elementary divisor d_2, the encrypting exponent e and the matrix U (see 12.13) are made public. The decrypting exponent d is kept secret. The security lies in the difficulty of factoring $d_2 = Nm$. The encryption procedure is carried out as follows:

Let $M = x_2 w_2$

$$C \equiv M^e \bmod <m>$$
$$\equiv (x_2 w_2)^e \bmod <m>$$
$$\equiv x_2^{e} w_2^{e} \bmod <m>$$

Using

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = U^{-1} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \quad \text{where} \quad \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 1 \\ i \end{pmatrix}$$

and

$$i^2 + 1 = 0$$

$C' \equiv y_2' w_2$ is obtained. Now reducing y_2' (mod d_2) the cipher C is obtained

$$C \equiv y_2 w_2 \bmod <m>$$

A similar procedure can be carried out for decryption to recover the original message using the decrypting exponent d. An example using this method is now given below.

Let $\Pi_1 = 2+3i$ and $\Pi_2 = 1+2i$ where Π_1 and Π_2 are primes in $Z[i]$. The modulus $m = 7i-4$ and $Nm = 65$. $\phi <m> = 48$. Let $e = d = 7$ where $ed \equiv 1 \pmod{48}$.

$$A = \begin{pmatrix} -4 & 7 \\ -7 & -4 \end{pmatrix}$$

Using the method given above to reduce A to diagonal matrix D,

$$D = \begin{pmatrix} 1 & 0 \\ -18 & 1 \end{pmatrix} \begin{pmatrix} -4 & 7 \\ -7 & -4 \end{pmatrix} \begin{pmatrix} 2 & -7 \\ 1 & -4 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 65 \end{pmatrix}$$

$$\underset{\sim}{w} = \begin{pmatrix} 4 - 7i \\ 1 - 2i \end{pmatrix} = \begin{pmatrix} 4 & -7 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ i \end{pmatrix}$$

where

$$U^{-1} = \begin{pmatrix} 4 & -7 \\ 1 & -2 \end{pmatrix}$$

Hence

$$\begin{pmatrix} 1 \\ i \end{pmatrix} = \begin{pmatrix} 2 & -7 \\ 1 & -4 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \tag{12.17}$$

Let the message be $M = 4w_2$

The cipher is given by

$$C \equiv M^7 \bmod \langle 7i-4 \rangle$$
$$\equiv 4^7 \, w_2^7 \bmod \langle 7i-4 \rangle$$

Using (12.17), $w_2 = 1-2i$

and $w_2^7 = 29-278i$

Using (12.17),

$$29-278i = -220w_1 + 909w_2 \tag{12.18}$$

Now reducing (12.18) (mod d_1) and (mod d_2), the cipher is given by

$$C \equiv 61w_2 \bmod \langle 7i-4 \rangle$$

A similar process for decryption with decoding exponent $d = 7$ gives back the original message M.

Case 2

Now consider the case where the primes Π_j which form the modulus m divide the rational primes p_j of the form $p_j \equiv 3 \pmod 4$. Then the norm Nm is a non-square free rational integer given by

$$Nm = \prod_{j=1}^{r} p_j^2$$

In this case although $Z[i]/\langle \Pi_j \rangle$ is a finite field of p_j^2 elements, one cannot represent the residue classes modulo $\langle \Pi_j \rangle$ using the integers $Z/p_j^2 Z$ as the latter does not form a field. On the other hand, one can represent the messages in the form $x+iy$ where $x, y \in Z$. As mentioned in 12.2.1, the Nm can be made to equal m^2 by appropriately choosing m to be a rational integer. Therefore, one can represent the distinct residue classes mod $\langle m \rangle$ as $x+iy$ where $0 \leqslant x, y \leqslant |\sqrt{Nm}| - 1$, thus giving rise to Nm residue classes. Using the elementary divisor theory described above, this corresponds to the case where $d_1 = \prod_{j=1}^{r} p_j$ and $d_2 = \prod_{j=1}^{r} p_j$. Again these are the only allowed values for d_1 and d_2 as the following conditions must hold:

(i) $d_1 \mid d_2$

(ii) $d_1 \cdot d_2 = Nm$

(iii) $Z/d_i Z$ is isomorphic to a direct sum of finite fields, for
       $i = 1,2$.

$$- 263 -$$

The divisors d_1 and d_2 and the exponent e are made public. The encryption and decryption procedures are very much simplified in this case. If a message M is raised to the power e then to perform mod <m> , it is only necessary to reduce the coefficients by the corresponding elementary divisors. Let M=x+iy be the message then, the cipher C is obtained by

$$C \equiv M^e \text{ mod } <m>$$
$$\equiv (x+iy)^e \text{ mod } <m>$$
$$\equiv g(\text{mod } d_1) + h (\text{mod } d_2) i$$

where $d_1 = d_2 = |\sqrt{Nm}|$

A similar procedure can be used for decryption.

12.3   Factorization Trapdoor System in Other Quadratic Fields

12.3.1   Quadratic Fields R (√D)

Note that when D=-1, $R(\sqrt{D})$, where R is the set of rational numbers, includes the ring of Gaussian integers considered in the previous section. Initially some of the properties of the integers in $R(\sqrt{D})$ which are required in the design of a trapdoor system are briefly examined. A detailed treatment of quadratic fields can be found in [61, 63, 65] .

A quadratic field is a field of degree 2 over the rationals. Such a field is of the form $R(\theta)$ where θ is a root of a quadratic polynomial which is irreducible over the rationals. Let θ satisfy an equation

$$x^2 + 2ax + b = 0 \quad \text{where a, b } \epsilon \text{ Z}$$

Then $\theta = -a \pm \sqrt{a^2 - b}$. Removing from $a^2 - b$ all square factors so that $a^2 - b = s^2 D$ where D has no factor higher than the first power, then $R(\theta)$ is equivalent to $R(\sqrt{D})$. That is, every quadratic field is of the form $R(\sqrt{D})$ where D is a rational integer free of square factors.

The ring of integers of $R(\sqrt{D})$ for square free D depends on the arithmetic properties of D. It is shown in [61] that the integers in $R(\sqrt{D})$ fall into two categories, namely,

(i)   if $D \not\equiv 1$ (mod 4), then the integers are of the form $x+y\sqrt{D}$

- 264 -

where x, y ε Z, ie, Z [$\sqrt{D}$]

(ii)   if D ≡ 1 (mod 4), then the integers are of the form x+y $(\frac{1+\sqrt{D}}{2})$ where x, y ε Z, ie, $Z\left[\frac{1+\sqrt{D}}{2}\right]$

Note that the Gaussian integers considered in the last section falls into the first category.

Using Lemma 1 (Section 12.2.1), it is seen that an integer in R($\sqrt{D}$) is a unit if its norm is $\pm$ 1. When D $\not\equiv$ 1 (mod 4), the norm of α is Nα = α $\bar{\alpha}$ = (a+b$\sqrt{D}$) (a-b$\sqrt{D}$) = $a^2 - Db^2$.
When D ≡ 1 (mod 4), the norm of α is given by

$$N\alpha = [a + \frac{b}{2}(1+\sqrt{D})] \ [a + \frac{b}{2}(1-\sqrt{D})] = (a + \tfrac{1}{2}b)^2 - \frac{D}{4}b^2.$$

Note that the norms are positive in complex quadratic fields (ie, D is negative) but not necessarily positive in real quadratic fields (ie, D is positive). Thus α is a unit if

$$a^2 - Db^2 = \pm 1$$

or

$$(a + \tfrac{1}{2}b)^2 - \frac{D}{4}b^2 = \pm 1$$

$$\left. \vphantom{\begin{array}{c}1\\1\\1\\1\end{array}}\right\} \qquad (12.19)$$

When D<0, the equations (12.19) have only a finite number of solutions [63]. When D = -1, as seen in Section 12.2.1 the ring of Gaussian integers Z[i] has four units namely $\pm$ 1, $\pm$ i. When D = -3, there are six solutions to the equation (12.19) namely $\pm$ 1, $\pm$ w, $\pm$ w^2 where w is the cube root of unity, w = (-1 + $\sqrt{-3}$)/2. For all other complex fields, the only units are $\pm$ 1. In the case of real fields, there exists an infinite number of solutions to the equation (12.19) and hence an infinite number of units [63]. These units however may be expressed in the form of $\pm$ ϵ^n where n takes all positive and negative rational values. The quantity ϵ is called the fundamental unit.

Using the definition of a prime element given in Section 12.2.1 it is seen that the Lemmas 2, 3 and 4 are also applicable in R($\sqrt{D}$). Although every integer in R($\sqrt{D}$) can be expressed as a product of primes, it does not necessarily imply that the factorization is unique like in Z or Z[i]. Consider for instance the factorizations of the integer 6 in R($\sqrt{10}$) expressed as

$$6 = 2.3 = (4 + \sqrt{10})(4 - \sqrt{10})$$

or of the integer 21 in R($\sqrt{-5}$)

$$21 = 3.7 = (1 + 2\sqrt{-5})(1 - 2\sqrt{-5}) = (4 + \sqrt{-5})(4 - \sqrt{-5})$$
$$(12.20)$$

- 265 -

Considering (12.20), we have the following situation: the integer 3, a Prime, divides $(1 + 2\sqrt{-5})(1 - 2\sqrt{-5})$ but fails to divide either factor in $R(\sqrt{-5})$. Such a situation does not arise for instance in Z or $Z[i]$. Hence it is seen that in $R(\sqrt{-5})$, Prime integers which are not associated can have a common factor which is not in $R(\sqrt{-5})$. It appears then that in such algebraic number fields the Primes are not necessarily the atoms from which all the integers are constructed.

It is in such cases the factorization of an ideal into a unique set of prime ideals (see Section 11.2.6)comes into use. The rings where the unique factorization of integers fails,correspond to non-principal ideal domains. The theory of non-principal ideal domains is considered to be beyond the scope of this thesis and hence the design of factorization trapdoor systems has been confined to principal ideal domains.

12.3.2 Design of Trapdoor Coding System: Complex Euclidean Quadratic Fields

The fields which possess the unique factorization of elements property obey Euclidean algorithm of one form or other. There are just five complex Euclidean fields namely when $D = -1, -2, -3, -7$ and -11. (There are 4 other complex fields which have the property of unique factorization of elements but obey a slightly different form of Euclidean algorithm [38]). For these nine cases the ring $R(\sqrt{D})$ is a principal ideal domain. The prime ideals are therefore the ideals generated by the prime integers in $R(\sqrt{D})$. From the point of view of designing a factorization trapdoor system, the primes in $R(\sqrt{D})$ and the relationship between the primes in Z and the primes in $R(\sqrt{D})$ need to be considered. More exactly, it is necessary to know whether a rational prime splits in $R(\sqrt{D})$ and if so how does it split.

From Kummer's theorem [68], the decomposition of an ideal $<p>$, where p is a prime in Z, into prime ideals in $R(\sqrt{D})$ is determined by the factorization of the polynomial $f(x) = x^2 - D$ in Z/pZ. Over Z/pZ, the factorizations of $f(x)$ are

$$x^2 - D = \begin{cases} x^2 & \text{if } p|D \text{ or } 4D \\ x^2 - D & \text{if } D \text{ is not a square (mod p)} \\ (x-a)(x+a) & \text{if } D \equiv a^2 \text{ (mod p)}, \ a \in Z \end{cases}$$

The three cases therefore correspond to:

$$- 266 -$$

1. If a prime p in Z divides the discriminant [68] of the field $R(\sqrt{D})$ (if $D \equiv 1$ (mod 4), discriminant $= D$ and if $D \not\equiv 1$ (mod 4), discriminant $= 4D$), then the ideal $<p>$ is factorized into the square of a prime ideal in $R(\sqrt{D})$. That is, $<p> = \rho^2$ where ρ denotes a prime ideal in $R(\sqrt{D})$, and $N\rho = p$. (12.21)

2. An odd prime p which does not divide the discriminant generates a prime ideal of degree 2 in $R(\sqrt{D})$ if $x^2 - D \equiv 0$ (mod p) does not have an integral solution. That is, p is irreducible in $R(\sqrt{D})$ and $<p> = \rho$ and $N\rho = p^2$. (12.22)

3. On the other hand, if $x^2 - D \equiv 0$ (mod p) (or $y^2 - 4D \equiv 0$ (mod p)) has a solution then $<p>$ decomposes into two distinct conjugate prime ideals, $\langle p \rangle = \rho_1 \rho_2$ where $N\rho_1 = N\rho_2 = p$ (12.23)

(Note that the prime $p = 2$ is of no cryptographic significance)

As an example, consider the non-principal ideal domain $R(\sqrt{-5})$ whose discriminant is equal to $4D = -20$. 2 and 5 are the only prime factors of the discriminant and consequently are factorable into squares of prime ideals as $<2> = <2, 1 + \sqrt{-5}>^2$ and $<5> = <\sqrt{-5}>^2$

The congruence $x^2 + 5 \equiv 0$ (mod p) has solution for $p = 3, 7, 23, \ldots$ but cannot be solved for $p = 11, 13, 17, \ldots$ Therefore
$<3> = <3, 1 + \sqrt{-5}>$ $<3, 1 - \sqrt{-5}>$
$<7> = <7, 3 + \sqrt{-5}>$ $<7, 3 - \sqrt{-5}>$ $\ldots$
while $<11>, <13> \ldots$ are prime ideals.

The designer chooses primes $P_1, \ldots, P_r$ in Z which give rise to prime ideals in $R(\sqrt{D})$. If the primes p_i are chosen so that $x^2 - D \equiv 0$ (mod p_i) have integral solutions then letting

$$<P_i> = \rho_{i1} \cdot \rho_{i2} \quad \text{where } \rho_{i1}, \rho_{i2} \text{ are prime ideals in } R(\sqrt{D})$$

the composite ideal $<m>$ is equal to, say,

$$<m> = \prod_{i=1}^{r} \rho_{i1}$$

The number of incongruent residues with respect to the ideal $<m>$ and relatively prime to $<m>$ is given by

$$\bar{\phi}<m> = \prod_{i=1}^{r} \bar{\phi}(\rho_{i1})$$
$$= \prod_{i=1}^{r} N\rho_{i1} - 1 \qquad (12.24)$$

And hence using (12.23),

– 267 –

$$\not\phi \langle m \rangle = \prod_{i=1}^{r} (p_i - 1) \qquad (12.25)$$

If the primes p_i, $1 \leqslant i \leqslant r$, are chosen such that $x^2 - D \equiv 0 \pmod{p_i}$ do not have an integral solution for all i, then the ideals $\langle p_i \rangle$ are irreducible in $R(\sqrt{D})$ and

$$\not\phi \langle m \rangle = \prod_{i=1}^{r} (p_i^2 - 1) \qquad (12.26)$$

using (12.22).

The designer can decide whether $x^2 - D \equiv 0 \pmod{p}$ for a chosen prime p has solution or not using the law of quadratic reprocity. In general if $x^2 \equiv a \pmod{n}$, where $\gcd(a,n) = 1$, has a solution then 'a' is said to be a quadratic residue modulo n. Here n=p and a=D. To determine whether D is a quadratic residue or not modulo p, the designer computes the Legendre Symbol $(\frac{D}{p})$. If $(\frac{D}{p}) = 1$ then D is a quadratic residue and if it is equal to -1 then D is a quadratic non-residue. Calculating the Legendre symbol is not much different from evaluating gcd of two numbers using Euclid's algorithm and can be done by repeated divisions in polynomial time. (Note that $|D|$ is of small value).

The Fermat theorem in this case is given by

$$\alpha^{\not\phi \langle m \rangle} \equiv 1 \pmod{\langle m \rangle}$$

where α is an arbitrary integer in $R(\sqrt{D})$ relatively prime to $\langle m \rangle$.

The encryption and decryption exponents e and d can be calculated using

$$ed \equiv 1 \pmod{\not\phi \langle m \rangle}$$

where $\not\phi \langle m \rangle$ is given by (12.25) or (12.26)

The public encryption key is (e,m) and the secret decryption key is (d,m). Having generated the rational primes p_i, calculated $\not\phi \langle m \rangle$, e and d, the designer needs to obtain m to make it public. That is, given the primes $p_1, \ldots, p_r$ in Z, he needs to calculate m where m = $\Pi_1 \ldots \Pi_r$ and Π's are primes in $R(\sqrt{D})$, and Π_i is of the form $\Pi_i = a_i + b_i\sqrt{D}$ if $D \not\equiv 1 \pmod{4}$ and $\Pi_i = a_i + b_i(\frac{1+\sqrt{D}}{2})$ if $D \equiv 1 \pmod{4}$. To obtain m, he needs to find $\Pi_1, \ldots, \Pi_r$, that is, he needs to calculate a_i and b_i for all i, $1 \leqslant i \leqslant r$ by solving either

$$N\Pi_i = a_i^2 - b_i^2 D = p_i \text{ or } p_i^2, \text{ if } D \not\equiv 1 \pmod{4} \qquad (12.27)$$

or

$$N\Pi_i = a_i^2 + a_i b_i - \frac{D-1}{4} b_i^2 = p_i \text{ or } p_i^2, \text{ if } D \equiv 1 \pmod{4}$$

$$(12.28)$$

Note that (12.28) can be written as

$$A_i^2 - Db_i^2 = 4p_i \text{ or } 4p_i^2 \qquad (12.29)$$

where $A_i = (2a_i + b_i)$

For solving (12.27) or (12.29), the method 4 of Section 12.2.2, which consists of arbitrarily choosing a_i and b_i to form integers Π_i and then checking whether their norms are primes or squares of primes, seems to be the most attractive one for the system designer.

Again it may be advisable for the designer to choose all the primes p_i, $1 \leq i \leq r$, to be of the same type, that is, either they all decompose into distinct conjugate prime ideals in $R(\sqrt{D})$ or they are irreducible in $R(\sqrt{D})$. If one of $a_i(A_i)$ or b_i is zero and the other is $p_i(2p_i)$, then factorization of m would be made easy as seen in Section 12.2.2.

12.3.3 Security of The System in $R(\sqrt{D})$

The security of the system seems to be the same as that of the RSA system. The opponent needs to find the decrypting exponent d to break the system. One way of finding this is to obtain $\phi<m>$. To calculate $\phi<m>$, he needs to factorize Nm given the modulus m. If the designer had chosen the rational primes such that congruence $x^2 - D \equiv 0$ (mod p_i) have a solution for all i, then the norm Nm is given by

$$Nm = \prod_{i=1}^{r} p_i$$

On the other hand, if the primes have been chosen such that $x^2 - D \equiv 0$ (mod p_i) have no solution for all i, then

$$Nm = \prod_{i=1}^{r} p_i^2$$

As in the case of Gaussian integers, note that the cryptanalyst does not require to find the primes Π_i in $R(\sqrt{D})$ to break the system. Thus unlike the designer, he is not faced with the problem of solving (12.27) or (12.29).

12.3.4 Representation of Messages and System Operation

If the primes p_i are chosen such that $x^2 - D \equiv 0$ (mod p_i) have solution for all i, then every integer of $R(\sqrt{D})/<m>$ is congruent to an element in the range 0 to $N<m> - 1$ as in Section 12.2.4. Thus the messages can be represented using integers in Z in the range 0 to $N<m> - 1$. As in Section 12.2.4, the encryption and decryption procedures

– 269 –

can be performed modulo Nm or m, where $Nm \in Z$ and $m \in R(\sqrt{D})$. The messages can also be represented in the form $x + y\sqrt{D}$ ($D \not\equiv 1 \pmod 4$)) or $x+y \frac{(1+\sqrt{D})}{2}$ ($D \equiv 1 \pmod 4$)), $x,y \in Z$, using the elementary divisor method given in Section 12.2.4.

If $D \not\equiv 1 \pmod 4$ then the integral basis of $R(\sqrt{D})$ is $(1,\sqrt{D})$ and the matrix A in Section 12.2.4 is equal to

$$A = \begin{pmatrix} a & b \\ bD & a \end{pmatrix}$$

If $D \equiv 1 \pmod 4$, then the integral basis of $R(\sqrt{D})$ is $(1, \frac{1+\sqrt{D}}{2})$ and the matrix A is equal to

$$A = \begin{pmatrix} a & b \\ \frac{b(D-1)}{4} & a+b \end{pmatrix}$$

Using the method given in Section 12.2.4, the matrix A can be reduced to diagonal form (d_1, d_2) and the modified basis of the ideal $< m >$ is obtained as $(d_1 w_1, d_2 w_2)$. The residue classes mod $<m>$ are therefore given by $x_1 w_1 + x_2 w_2$ where $0 \leq x_1 < d_1$ and $0 \leq x_2 < d_2$. As Nm is equal to $\prod_{i=1}^{r} p_i$ in this case, the only allowed values for d_1 and d_2 are given by $d_1 = 1$ and $d_2 = \prod_{i=1}^{r} p_i$. The operation modulo $< m >$ required in encryption and decryption is performed in the same way as that given in Section 12.2.4.

If the primes p_i are chosen such that $x^2 - D \equiv 0 \pmod{p_i}$ have no solution for all i, then the norm is a non-square free rational integer and as in Section 12.2.4, one cannot represent the residue classes modulo $< \Pi_i >$ using the integers $Z/p_i^2 Z$. This case corresponds to the situation when $d_1 = d_2 = \prod_{i=1}^{r} p_i$ in the elementary divisor method. The messages are hence represented in the form $x_1 + x_2 \sqrt{D}$ for $D \not\equiv 1 \pmod 4$ and $x_1 + x_2 (\frac{1+\sqrt{D}}{2})$ for $D \equiv 1 \pmod 4$ where $0 \leq x_1, x_2 < \sqrt{Nm} = d_1 = d_2$. The reduction mod $<m>$ is performed by simply reducing each of the components x_1 and x_2 mod $\sqrt{Nm}$.

Again, the elementary divisors d_1 and d_2 are made public together with the encrypting exponent, e.

12.3.5 Real Quadratic Fields

There are 16 real quadratic fields which obey the Euclid's algorithm with respect to the field norm and hence possess the unique factorization of integers property [65]. They occur when D is equal to

- 270 -

2,3,5,6,7,11,13,17,19,21,29,33,37,41,57 and 73. One major difference
between the complex quadratic fields and real quadratic fields is that
the latter has infinitely many units (Section 12.3.1). However this
does not cause any serious problems provided we choose the message
representation within the allowed standard set mod $<m>$. This is done
in any case, by either using the elementary divisor method or using
the standard set of messages modulo Nm. Thus it seems that in this
case, there are not any major changes to the trapdoor system described
above for the Euclidean complex quadratic fields.

12.4   Discussion

The design of factorization trapdoor systems in some
quadratic fields which are principal ideal domains has been considered.
However majority of the quadratic fields are non-principal ideal
domains and they do not possess the unique factorization of elements
property. But the unique factorization of a non-zero ideal into prime
ideals still applies in such fields. Factorization trapdoor system
seems possible if the chosen ring modulo the ideal is isomorphic to
a direct sum of finite fields. Choosing a square free ideal A, not
necessarily a principal ideal, in $R(\sqrt{D})$ and let

Then $A = \rho_1 ... \rho_r$ where ρ_i are prime ideals in $R(\sqrt{D})$

$$R(\sqrt{D})/A \cong R(\sqrt{D})/\rho_1 \oplus ... \oplus R(\sqrt{D})/\rho_r$$

where $R(\sqrt{D})/\rho_i$ is a finite field of $N\rho_i$ elements

The order of the group formed by residues relatively prime to A is
given by

$$\underline{\phi}(A) = \prod_{i=1}^{r} \underline{\phi}(\rho_i) = \prod_{i=1}^{r} (N\rho_i - 1) \quad \text{using (11.1)}$$

The coding exponents e and d can be found using $ed \equiv 1 \ (\text{mod } \underline{\phi}(A))$.

From Lemma 3, every prime ideal divides a rational prime p which is
unique [63]. If the rational primes p_i are chosen such that $x^2 - D \equiv 0$
(mod p_i) have solution for all i, then the residue classes mod ρ_i can be
represented using rational integers mod p_i. Hence the messages modulo
A can be represented using integers 0 to N(A)-1. The elementary divisor
method of Section 12.2.4 can also be employed to represent the messages.
If the primes p_i are chosen such that $x^2 - D \equiv 0$ (mod p_i) have no solution

- 271 -

Page 271

for all i, then both elementary divisors d_1 and d_2 are equal to $\sqrt{N(A)}$.

Let us conclude this chapter by considering a small example which shows some of the calculations involved in the design of a trapdoor system in $R(\sqrt{-5})$, a non-principal ideal domain.

Choosing rational primes $p_1=3$ and $p_2=7$, their decomposition in $R(\sqrt{-5})$ is given by

$$<3> \;=\; (3,\, 1+\sqrt{-5})\; (3,\, 1-\sqrt{-5}) \;=\; \rho_{11}\, \rho_{12}$$

$$<7> \;=\; (7,\, 3+\sqrt{-5})\; (7,\, 3-\sqrt{-5}) \;=\; \rho_{21}\, \rho_{22}$$

Let

$$A = \rho_1\, \rho_2 \;=\; (3,\, 1+\sqrt{-5})\; (7, 3-\sqrt{-5})$$

Then

$$A = (21,\, 7+7\sqrt{-5},\; 9-3\sqrt{-5},\; 8+2\sqrt{-5})$$

Any ideal can be represented using a two-element basis over the ring [65]. Using standard rules for transforming the ideal basis [65],

$$A = (21,\, 4+\sqrt{-5})$$
$$N(A)= 21$$

The integral basis of $R(\sqrt{-5})$ is $(1,\sqrt{-5})$ as $-5 \not\equiv 1 \pmod 4$
Representing the ideal A as a Z-module

$$[21,\, 21\theta,\; 4+\theta,\; -5+4\theta\,] \qquad \text{where } \theta = \sqrt{-5}$$

or

$$\begin{pmatrix} 21 \\ 21\theta \\ 4+\theta \\ -5+4\theta \end{pmatrix} = \begin{pmatrix} 21 & 0 \\ 0 & 21 \\ 4 & 1 \\ -5 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ \theta \end{pmatrix}$$

Let

$$B = \begin{pmatrix} 21 & 0 \\ 0 & 21 \\ 4 & 1 \\ -5 & 4 \end{pmatrix}$$

Using the algorithm given in Section 12.2.4, this matrix is reduced to a diagonal form (d_1, d_2) where $d_1 \mid d_2$.

$$VBU = \begin{pmatrix} 1 & 0 \\ 0 & 21 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

- 272 -

Page 272

where

$$V = \begin{bmatrix} 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & -5 & -4 \end{bmatrix} \quad \text{and } U = \begin{pmatrix} 1 & 5 \\ 0 & 1 \end{pmatrix}$$

Now the messages can be represented as

$$M = x_1 w_1 + x_2 w_2 \qquad 1 \leqslant x_i < d_i \qquad i=1,2 \qquad (12.30)$$

where

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = U^{-1} \begin{pmatrix} 1 \\ \theta \end{pmatrix} = \begin{pmatrix} 1 & -5 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \theta \end{pmatrix}$$

In this case, as $d_1 = 1, M = x_2 w_2$, $1 \leqslant x_2 < d_2$. However if the rational primes are chosen such that they stay as primes in the higher field $R(\sqrt{D})$, then $d_1 = d_2 = \sqrt{N(A)}$ and the messages can be represented using (12.30). $\not\!\phi(A) = (NP_1 - 1)(NP_2 - 1) = 12$. One set of coding exponents e and d is $e = 5$, $d = 5$. The messages can be encrypted using a similar procedure to that given in Section 12.2.4 for Gaussian integers, except in this case, the recursive equation is $f(\theta) = \theta^2 + 5 = 0$ instead of $i^2 + 1 = 0$. The elementary divisors d_1, d_2, the encrypting exponent e and the matrix U are made public and the decrypting exponent d is kept secret.

- 273 -

CONVENTIONAL CRYPTOSYSTEM WITH PUBLIC KEY DISTRIBUTION

13.1   General

        This chapter focusses on the concept of public key
distribution (PKD) mentioned in Section 9.6, whereby the 'public key
idea' is solely used to transfer a key between two users over an
insecure channel. In this case, there is no message as such which
gets encrypted at the sending end and decrypted at the receiving end.
In such a system, two users who wish to exchange a key communicate
back and forth until they arrive at a key in common. The opponent
eavesdropping on this exchange must therefore find it computationally
infeasible to derive the key from the information overheard. This
type of arrangement is used in conjunction with the conventional
cryptographic DES interface unit to form a DES/PKD hybrid system. An
implementation of such a hybrid system is discussed in Section 13.5.
        Diffie and Hellman [35] proposed such a key distribution
system based on exponentiation over finite field. This technique
briefly described in Section 9.5.3 makes use of the apparent difficulty
of computing logarithms over a finite field GF(q) where q is a very
large prime number. Each user generates an independent random number
x_i chosen uniformly from the set of integers $\{1,2,\ldots q-1\}$, and computes
$y_i = a^{x_i}(\bmod\ q)$ where 'a' is a primitive element of GF(q). The number
y_i is made public and the number x_i is kept secret. When users i and
j wish to communicate privately, they can use the common key K_{ij}, given
by $K_{ij} \equiv a^{x_i x_j} \equiv (y_i)^{x_j} \equiv (y_j)^{x_i}(\bmod\ q)$.
        For the system to be secure, the key K_{ij} must be difficult
to compute for anyone who knows y_i and y_j but does not know either x_i
or x_j. In order to ensure that this computation is difficult, it is
necessary that logarithms over GF(q) be difficult to compute. Other-
wise an opponent could compute x_i from y_i and impersonate user i.
Some currently known algorithms for calculating the logarithms over
finite fields are now briefly looked at.

13.2   Logarithms Over Finite Fields

        Knuth's algorithm [45] to compute logarithm over GF(q)

– 274 –

requires $2 \lceil q^{\frac{1}{2}} \rceil$ multiplications (mod q) in addition to other operations of comparable complexity. This algorithm requires $2 \lceil q^{\frac{1}{2}} \rceil$ words of memory, each $\lceil \log_2 q \rceil$ bits long. This algorithm can be generalized to allow a time-memory tradeoff in which time is almost proportional to q^r and memory to q^{1-r} for any $0 \leqslant r \leqslant 1$. Knuth's algorithm corresponds to the case where $r=\frac{1}{2}$. Allowing arbitrary values of r enables the algorithm to be adjusted to particular time-memory requirements. The algorithm to compute x from $y \equiv a^x \pmod{q}$ works as follows:

Let $m = \lceil q^r \rceil$

Then there exist integers c and d such that

$x = cm+d$ with $0 \leqslant c < \lceil q/m \rceil \doteq q^{1-r}$

and $0 \leqslant d < m \doteq q^r$

Substituting for x in a^x, gives

$y = a^{cm+d} \pmod{q}$

ie, $a^d \equiv y \ a^{-cm} \pmod{q}$

In order to determine c and d, the values of $a^d \pmod{q}$ are precomputed for $d=0,1,\ldots m-1$ in $O(q^r)$ operations and the results stored in a table in $O(q^r \log_2 q^r)$ operations. Then $y, ya^{-m}, ya^{-2m} \ldots \pmod{q}$ are each computed and compared with the sorted table of $\{a^d\}$ until a match is found. Each value of c tried requires 1 multiplication (mod q) and $\log_2 q^r$ comparisons, thus giving a total of $(1+\log_2 q^r)$ operations. There are $O(q^{1-r})$ values of c to be tried. When $r=1$, this algorithm is a look up table and when $r=0$, the algorithm reduces to an exhaustive search. Neglecting the logarithmic factors it is seen that the time-memory product is constant (since $q^r \cdot q^{1-r} = q$) as the algorithm ranges between the extremes of a look up table and an exhaustive search.

Pohlig and Hellman [69] proposed an improved algorithm to compute logarithms over GF(q) when the prime q is chosen such that q-1 has only small prime factors. On the other hand, it has long been known that a disproportionately large portion of numbers are entirely composed of small prime factors and it is precisely this fact on which Adleman based his very recent discrete logarithm algorithm [70]. Assuming a microsecond per operation machine, this new algorithm could be expected to compromise a system based on a 200-bit prime q in 2.6 days rather than the 3×10^{16} years using the best previously published method due to Shanks [45]. For a large enough prime q, however, Adleman's algorithm is also infeasible. A sketch of this subexponential Adleman's solution to the logarithm problem is now given following [6].

- 275 -

Problem: Given a, y and q (a prime) find x that satisfies $y \equiv a^x \pmod q$ (where 'a' is a primitive element in GF(q) in the PKD system).

The algorithm requires the following preliminary definition: a number α is said to be 'smooth' with respect to a bound BD, if the factorization of α into primes, $\alpha = \prod_{i=1}^{r} p_i^{e_i}$ is such that all p_i, $1 \leq i \leq r$, satisfy $p_i \leq BD$.

Step 1   Find by random sampling (and checking) a positive integer R such that $B \equiv y^R \pmod q$, $1 \leq B \leq q-1$ and B is smooth with respect to the bound $BD(q) = e^{(\ln q \ln \ln q)^{\frac{1}{2}}}$ and gcd $(R, q-1) = 1$.

Step 2   Let $q_1 \ldots q_m$ be all the primes $\leq BD(q)$. Find by random sampling (and checking) positive integers R_i for $1 \leq i \leq m$ such that $A_i \equiv a^{R_i} \pmod q$ where $1 \leq A_i \leq q-1$ and A_i is smooth with respect to BD(q) and the vectors $\vec{A}_i = (e_{i1}, \ldots e_{im})$, where $A_i = \prod_{j=1}^{m} q_j^{e_{ij}}$, span the m-dimensional module over $Z/(q-1)Z$, the ring of integers modulo (q-1).

Step 3   By Gaussian elimination express $\vec{B} = (f_1, \ldots, f_m)$, where $B = \prod_{j=1}^{m} q_j^{f_j}$, as a linear combination of the $\vec{A}_i$, namely $\vec{B} = n_1 \vec{A}_1 + n_2 \vec{A}_2 + \ldots + n_m \vec{A}_m \pmod{(q-1)}$ where $0 \leq n_i \leq q-2$ for $1 \leq i \leq m$.

Now
$$B = \prod_{j=1}^{m} q_j^{f_j}$$

and
$$\ln B = \sum_{j=1}^{m} f_j \ln q_j$$

$$\ln B = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} n_i e_{ij} + k_j (q-1) \right) \ln q_j \qquad (13.1)$$

as
$$f_j \equiv \sum_{i=1}^{m} n_i e_{ij} \pmod{(q-1)}$$

Exponentiating (13.1) gives,
$$B \equiv \prod_{i=1}^{m} \left(\prod_{j=1}^{m} q_j^{e_{ij}} \right)^{n_i} \pmod q$$

That is,
$$B \equiv \prod_{i=1}^{m} A_i^{n_i} \pmod q$$

Where A_i can be replaced by a^{R_i}.

Step 4   Calculate $R^{-1} \pmod{q-1}$. Then $y = \prod_{i=1}^{m} a^{n_i R_i R^{-1}} \pmod q$

That is, $y \equiv a^{R^{-1}(\sum_{i=1}^{m} n_i R_i)} \pmod q$.   That is, $x \equiv R^{-1} \sum_{i=1}^{m} n_i R_i \pmod q$

Now briefly consider the execution time of this algorithm.

- 276 -

Let $\psi(x,y)/x = \text{Prob}(z$ is smooth with respect to $y/z \leqslant x)$. Using Erdos's result [6], $\psi(x,y)/x = e^{-(\ln x/\ln y)(\ln\ln x - 1)}$. Thus the expected number of tries needed to obtain a smooth number is approximately $e^{(\ln x/\ln y)(\ln\ln x - 1)}$. This search must be carried out $\delta(y)$ times where $\delta(y)$ is the number of primes less than or equal to y and $\delta(y) \doteq y/\ln y = e^{\ln y - \ln\ln y}$. Thus the total computational effort is approximately $e^{(\ln x/\ln y)(\ln\ln x - 1) + \ln y - \ln\ln y}$ which is approximately $e^{c/d + d}$ with $c = \ln x \ln\ln x$ and $d = \ln y$. But $e^{c/d + d}$ $(c > 0)$ is maximized when $d = c^{\frac{1}{2}}$, yielding a computational effort of order $e^{2(\ln x \ln\ln x)^{\frac{1}{2}}}$. A sharper analysis [6] results in an upper bound for the logarithm problem of $e^{(\ln q \ln\ln q)^{\frac{1}{2}}}$ thus removing the factor 2 from the exponent.

System designer's avoidance of q such that $q-1$ has only small prime factors cannot be accomplished by first choosing a prime q such that $e^{(\ln q \ln\ln q)^{\frac{1}{2}}}$ is prohibitively large from the cryptanalyst point of view and then determining the prime power factorization of $q-1$. This is because the most efficient known factorization algorithm due to Schroeppell (unpublished) also makes use of the concept of smoothness. It has an expected running time for factoring $q-1$ of $e^{(\ln(q-1)\ln\ln(q-1))^{\frac{1}{2}}}$ One way to overcome this problem is. to generate a large random prime number u and let q to be the first prime in the sequence $iu+1$ for $i=2,4,6\ldots$ as mentioned in Section 10.2.

13.3   Public Key Distribution in GF(2^n)

While it is possible to implement the exponentiation public key distribution system as above, some implementation difficulties can be overcome by considering the exponentiation system in the extension fields GF(2^n).

First consider the exponentiation over GF(q) from a system design point of view. Initially one needs a suitable set of routines to generate and test for large prime numbers. The tests mentioned in Section 10.2 can be used for this purpose. Further it is required that the routine must generate a prime q such that $q-1$ has no small factors. Blakely [71] refers to such primes of the form $q=2p+1$ where p is a prime as 'safe' primes. Then one must choose a primitive element 'a' in GF(q). The number of primitive elements (mod q) is given by the Euler totient function $\phi(q-1)$ and hence the probability that an arbitrarily chosen element 'a' is primitive is given by $\phi(q-1)/q-1$. If q is of the form $4p+1$ where p is a prime then 2^n(mod q) is a primitive element (mod q) for any n relatively prime to $q-1$ and hence the problem

of determining a primitive element is simplified. Otherwise finding a primitive root of an arbitrary prime q may be more cumbersome requiring the knowledge of the factors of q-1. In addition, multiple precision arithmetic is required which is cumbersome and slows down the response time.

To overcome some of these problems associated with the GF(q) implementation, Berkovits [72] proposed to perform the exponentiation in the extension field $GF(2^n)$, instead of GF(q). As this is an extension field of GF(2) all operations are based on modulo 2 arithmetic which is easily implementable using digital logic systems. Addition and sub-traction are performed with exclusive-or operation alone, while oper-ations in GF(q) require carry and borrow propagation. Further advan-tages arise from the choice of n to be a prime so that 2^n-1 is a Mersenne prime. Since there are no subgroups within the multiplicative group of such a field, the logarithmic attack reduces to an exhaustive search. Furthermore, since the order of the multiplicative group is a prime, every element except one is primitive. Thus the selection of 'a' becomes arbitrary. Finally, since all the resulting y's ($y \equiv a^x$ in $GF(2^p)$) are also primitive, the number of possible keys is maximized.

As seen in Section 10.5, the elements of $GF(p^n)$ can be represented using polynomials of degree less than n whose coefficients are in GF(p). In this system p=2 and hence the coefficients are all either 0 or 1. Using the earlier notation, the field $Z/2Z[x]/f(x)$, where f(x) is an irreducible polynomial of degree n over $Z/2Z$, is isomorphic to $GF(2^n)$. Multiplication of two elements in $GF(2^n)$ are programmed as multiplication of two polynomials and the terms in the product with exponent n or higher are reduced modulo the generating irreducible polynomial f(x). Let f(x) be a monic polynomial and is equal to

$$f(x) = x^n + \sum_{i=0}^{n-1} b_i x^i \qquad \text{where } b_i \in \{0,1\}$$

Then

$$x^n \equiv \sum_{i=0}^{n-1} b_i x^i \qquad (\text{mod } f(x))$$

Then the reduction process of a polynomial P(x) modulo f(x) is performed as follows: (cf Section 10.5)

Let

$$P(x) \equiv Q(x) + R(x) x^n$$

where the degree of Q(x) is less than n.

- 278 -

Then
$$P(x) \equiv Q(x) + R(x) \sum_{i=0}^{n-1} b_i x^i$$

$$\equiv Q(x) + \sum_{b_i=1} R(x) x^i \quad (\text{mod } f(x))$$

This result may still have terms with exponents equal to n or greater and the process is repeated. The reduction process continues until the result is of degree less than n.

As 2^n-1 is prime, any number 'a' less than 2^n-1 (other than 1) can be used as the base (primitive element in $GF(2^n)$) of the PKD system. The base 'a' can be represented as a vector whose value as a binary number is equal to 'a'. That is, let

$$a = \sum_{i=0}^{n-1} a_i 2^i \qquad \text{where } a_i \in Z/2Z$$

Let the secret keys of users A and B be X_A and X_B which are integers less than 2^n-1 (except 1). Then,

$$y_A(x) = \left(\sum_{i=0}^{n-1} a_i x^i \right)^{X_A} \quad (\text{mod } f(x))$$

where $y_A(x)$ is a polynomial of degree less than n with coefficients over GF(2). $y_A(x)$ can be represented as an integer Y_A in the range $0 < Y_A < 2^n-1$ by calculating the value of the polynomial $y_A(x)$ in binary. Similarly the user B calculates Y_B. Thus the public key and secret key pairs of users A and B are given by (Y_A, X_A) and (Y_B, X_B) respectively. The common key K_{AB} is derived by users A and B using

$$K_{AB} \equiv (y_A(x))^{X_B} \equiv (y_B(x))^{X_A} \quad (\text{mod } f(x))$$

The choice of the extension field $GF(2^n)$ depends on the required difficulty of computing logarithms over the field. The particular choice of the Mersenne prime $2^{127}-1$, that is, n=127 is very attractive from implementation point of view. In $GF(2^{127})$, manipulation of 127-bit blocks are conveniently performed in most computers which have 8,16,32 or 64-bit architectures. A further attraction is that there exists a particularly simple irreducible polynomial, a trinomial, over GF(2) namely $f(x) = x^{127} + x + 1$ [73] which can be used to generate all the elements of $GF(2^{127})$. But with the advent of the subexponential algorithm for computing logarithm over finite fields by Adleman, it is necessary to work in higher extension fields to offer a similar amount of work factor as the DES to break the system. The

- 279 -

next Mersenne prime occurs when n=521 and $2^{521}-1$ is a prime. Thus one can work in $GF(2^{521})$ to overcome the attack using Adleman's subexponential logarithm algorithm. In this field, the corresponding irreducible trinomial over GF(2) is given by $f(x) = x^{521} + x^{32} + 1$ [73]. Tore Herlestam [74] has proposed a heuristic method for computing logarithms over $GF(2^p)$ where p is a prime. When 2^p-1 is a Mersenne prime less than or equal to $2^{31}-1$, the method is reported to work in very short running times on a general purpose computer [74]. Although the numerical results obtained so far and the complexity of the problem does not allow to assess the security of the corresponding public key distribution systems with p=127, 521 and larger, this may induce doubts that such systems could be considered secure.

Even so, this system, referred to as the Mitre system [75], is probably the most practical of the public key algorithms that have been proposed so far. However there is another attack called the short cycling attack which can be used with any public key system discussed so far. This attack may enable 'backdoor' penetrations, for example, in the Mitre system, a penetrator could use his knowledge of the system parameters namely the system base 'a' and the modulus polynomial f(x) to superencipher intercepted cipher until a cycle occurs. The effectiveness of such an attack on the Mitre system is now considered. This attack is of the same type as the one used by Simmons and Norris [76] against the RSA system.

13.4   Short Cycling Attack

Suppose g is a function of a set S into itself. Then given an x in S, the sequence defined by $x_0=x$ and $x_i=g(x_{i-1})$ is called the path of the element x under the action of g. If S is a finite set, then the path of each element must eventually repeat itself. When g is a one to one map, then repetition begins with $x_k=x_0=x$ for some minimal k. Under these circumstances, the path of x cycles around the same k elements. This circular path is called the orbit of x under g. The orbit of any element not in the orbit of x is completely disjoint from the orbit of x. Thus the set S is partitioned into disjoint orbits and hence the sum of the numbers of elements in the distinct orbits is the cardinality of S which is equal to the size of the largest orbit possible.

Now consider the cycling in the PKD system based on exponentiation in $GF(2^n)$. Let

- 280 -

$$P = \{x : x \text{ an integer between } 1 \text{ and } 2^n - 1\}$$

and
$$C = \{y : y \text{ a non-zero element of } GF(2^n)\}$$

Let P denote the set of plaintexts (secret keys) and C the set of ciphertexts (public keys) under the encrypting function E. Let us define another function DEC from C to P as follows: If y is an element of C, it is a polynomial of degree less than n. Writing y as the n-tuple of its coefficients and evaluating that n-tuple as the binary expansion of an integer, the result DEC(y) is obtained. Since both E and DEC are one to one functions, their composition F is a one to one map of the finite set P onto itself. The set P is partitioned into disjoint orbits under the F-map. These orbits are the cycles. A schematic diagram of the mappings is shown in Figure 13.1.
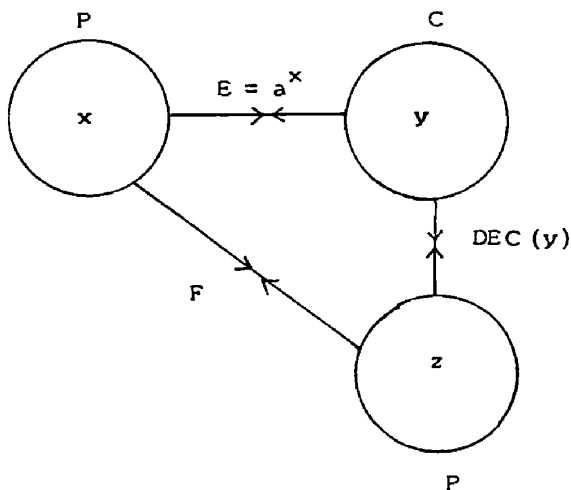
Fig 13.1

The threat of short cycling arises as follows:

Let
$$y_0 \equiv a^{x_0} \pmod{f(x)}$$

Then
$$x_1 \equiv DEC(y_0)$$

$$y_1 \equiv a^{x_1} \pmod{f(x)}$$

$$x_2 \equiv DEC(y_1)$$

$$\cdot \qquad \cdot$$
$$\cdot \qquad \cdot$$
$$\cdot \qquad \cdot$$

$$y_i \equiv a^{x_i} \pmod{f(x)}$$

- 281 -

$$x_{i+1} \equiv DEC(y_i)$$

Hence at some point r, as the set is finite

$$y_r \equiv a^{x_r} \pmod{f(x)}$$

$$x_{r+1} \equiv DEC(y_r)$$

$$\equiv x_0$$

and

$$y_{r+1} \equiv a^{x_0} \pmod{f(x)}$$

$$x_1 \equiv DEC(y_{r+1})$$

When this point is reached, the opponent realizes that the penultimate number of the sequence gives the original secret key x_0.

This type of short cycling analysis has been carried out in small extension fields using irreducible polynomials of degrees 3 and 7. The primitive polynomials used are $f(x) = x^3+x+1$ in $GF(2^3)$ and $f(x) = x^7+x+1$ in $GF(2^7)$ respectively. The system base 'a' is allowed to vary from 2 to 7 and 2 to 127 respectively. Then the cycle lengths are determined for various values of the secret exponent x, using the program CYCLE.FTN given in Appendix 15. The cycle lengths obtained in $GF(2^7)$ for several values of x are given in Appendix 17 (Section A17.1). The complete set of results shows that in the case of $GF(2^7)$ with f(x) $=x^7+x+1$, the base a=38 (evaluated using x=2 in x^5+x^2+x, an element of $GF(2^7)$) gives the maximum cycle length of 127. [In the case of $GF(2^3)$ with $f(x)=x^3+x+1$, a=5 (x^2+1 in $GF(2^3)$) gave the maximum cycle length of 7]. For all the other values of the base, the cycle lengths are less than 127. This can be explained by the reasoning that the choice of the system base 'a' partitions the set P into disjoint orbits; in the case of a=38 with the primitive polynomial x^7+x+1, this has created a partition consisting only of the entire set P. Thus every element of P is in the same orbit and hence every value x gives the maximum cycle length with a=38. The other choices of 'a' partitioned P into several disjoint orbits with different cycle lengths. Two different exponent values of x having the same cycle length (for instance a=9, $x_1=15$, $x_2=32$, cycle length =116) may be due to two or more orbits having the same cardinality or it may be that x_1 and x_2 lie in the same orbit.

These results may indicate that certain bases and certain generating polynomials are superior to others thus giving rise to maximum cycle lengths irrespective of the exponent x. If so, the system parameters should not be chosen at random, for instance, the system base 'a' should not be chosen randomly among all primitive

- 282 -

elements. To determine whether an optimum set of parameters is possible, further analysis is carried out in the extension field $GF(2^7)$.

Considering the mappings shown in Figure 13.1, it is seen that an opponent is not restricted to choosing the function DEC to get back from C to P. Any one to one map of C to P will serve his purpose and the system designer cannot guard against all possible choices of the opponent. Consider for instance, the function $G(y)$ where

$$G(y) = DEC \ (b \cdot y)$$

Let b take all non-zero values in the set C. That is, b can be any one of the 127 polynomials of degree less than 7 in $GF(2^7)$. Consider the following algorithm.

1.   Choose a system base 'a'.

2.   Choose a particular value for 'b' in $G(y) = DEC(b \cdot y)$

3.   Vary the exponent values x (secret keys in PKD) from 2 to 127 and in each case, calculate the corresponding cycle length.

4.   Do steps 2 and 3 for 127 values of 'b'.

This algorithm has been implemented using the program RANDCYCLE.FTN given in Appendix 16. Several values for the system base 'a' have been tried; only the results for a=38 are given in Appendix 17, Section A17.2. The results show the expected cycle lengths obtained as the secret exponent x varies from 1 to 127 for different values of the polynomial b (evaluated as a binary vector). From the results, it is seen that by varying b in $G(y) = DEC(b \cdot y)$ the expected cycle length can be changed for a fixed base 'a'. Hence the opponent can obtain a shorter cycle length than the maximum, by appropriately choosing the value of b in $G(y)$ for any system base 'a'. Section A17.2 shows that with a=38, even though the expected cycle length is equal to the maximum 127, when b=1, with b=2, the expected cycle length $\doteq$ 38.87, with b=13, expected cycle length $\doteq$ 27.18, with b=125, expected cycle length $\doteq$ 32.64 and so on. Hence even if the system designer had chosen the 'best' system base, a=38, in his PKD system in $GF(2^7)$, if the opponent chooses b=13 in his $G(y) = DEC(b \cdot y)$ then he only needs to superencipher on average 28 times before obtaining the secret exponent x. The system designer has no control over the opponent choice of 'b'. Thus it seems that there is no best choice for the system base 'a'.

Section A17.3 of Appendix 17 gives the average expected cycle lengths for several values of the system base 'a'. That is, having

- 283 -

selected a system base 'a', the expected cycle lengths are calculated
for the 127 non-zero specialized values of the polynomial b. Then the
average of these expected cycle lengths is determined. From the results
it seems that the average expected cycle length for a chosen system
base is around 63.5. Recalling that in the algorithm given above, one
has considered only 127 functions of a specialized form for the
polynomial 'b', it appears that the average expected cycle length will
approach 63.5 if one has averaged over all the 127! possible functions
from the set C to the set P. Dr R Odoni has in fact explained using a
heuristic argument that the cycle length will be n/2 when all the
n! functions are taken into account.

Thus once the opponent chooses the function from C to P,
that is, the value of b in G(y), then he has actually chosen the order
in which he will try elements of P to search for the one that encrypts
into cipher y. That is, he has effectively decided on the elements

$$G(y_1) = DEC \ (b \cdot y_1)$$
$$G(y_2) = DEC \ (b \cdot y_2)$$
$$\vdots \qquad \qquad \vdots$$
$$G(y_n) = DEC \ (b \cdot y_n)$$

The opponent tries each of the elements in the above sequence in turn
in the equation $a^{G(y_i)} = y_{i+1}$ to find the one which matches with the
given y_1. For different values of b, the order of the sequence of
elements changes and hence the number of elements to be searched to
find the match with y_1 changes. The average number of such elements
to be tried is about 63.5. This implies that for a randomly chosen
system base, the expected cycle length of an arbitrary cycle is about
half the number of non-zero field elements. (That is, $(2^{127}-1)/2$ and
$(2^{521}-1)/2$ for $GF(2^{127})$ and $GF(2^{521})$ respectively). Thus although the
cycling attack will eventually find a solution, the work required
appears to be equivalent to a "random" exhaustive key search and hence
confirms [77].

In the next section, the implementation of this PKD system
in conjunction with the conventional DES system, using Apple micro-
computers is considered.

13.5   DES/PKD Hybrid System

A hybrid DES/PKD demonstration system has been developed

- 284 -

using the DES interface unit (Chapter 4) and the exponentiation system over $GF(2^{127})$. The system has been experimented using two Apple microcomputers forming a link. Each Apple microcomputer is assumed to be shared by n users (here n = 5). Each of these n users can compose and send a message to any other user at the other end in a secure way. This system is primarily intended as a testbed for investigating the problems of developing an application system incorporating DES/PKD techniques on a microprocessor based system.

The DES interface card is used to encrypt messages under one of the three modes namely the ECB, CFB or CBC (Chapter 5). The PKD function performs the basic $GF(2^{127})$* exponentiation system using $f(x) = x^{127} + x + 1$ as the modulus irreducible polynomial. This is essentially used to transfer the session keys securely between the two Apple microcomputers over a public telephone network. With 6502 microprocessor running at 1 MHZ, an average time of 4 seconds with a worst case of 6 seconds is required to perform the exponentiation in $GF(2^{127})$. The PKD program size is approximately 400 bytes and the listing of the program is given in Appendix 18 (see Section 13.6).

During system operation, the user can communicate with an user at the other end under DES or DES/PKD modes or generate a new secret/public key pair for the PKD system. The system base 'a' and the modulus polynomial $f(x)$ are input once to the program and they are assumed to be fixed. The public key generated is stored in some preallocated memory location depending upon the user ID, n. That is, in this simple system, each terminal stores its own list of public keys. This is sometimes referred to as the local Public Key File (PKF) mode.

A secure connection is established between users i and j through a simple connection protocol sequence. This sequence allows to establish a DES session key and initialization vector

- -

*As mentioned in Section 13.2, this can be changed to $GF(2^{521})$ with $f(x) = x^{521} + x^{32} + 1$ to overcome Adleman's algorithm.

- 285 -

required for secure communication between the users i and j. It also enables each user to authenticate the other's identity. A new session key and a new initialization vector, derived using the common key K_{ij}, are established for each session between i and j. This sequence is used with the local PKF mode mentioned above.

When user i wishes to establish a secure connection with user j, i sends his user ID(I) along with the ID of user j(J) to the other end. That is,

$$i \rightarrow j \; : \; I, J$$

The user j is informed at the receiving end that a communication has been requested by user i and the system asks the user j to input his secret key via the keyboard of the terminal. As soon as user j has entered his 16 character (127-bit) secret key, the system requests user i at the sending end to input his secret key. At this point, users i and j fetch the other user's locally stored public key value from their memories and compute the common key K_{ij} using their own secret keys independently at their respective ends. Then the sending end generates a pseudo-random number, R, and encrypts this number under the ECB mode of DES using the first 64 bits of K_{ij} as the DES key. (K_{ij1}). This cipher is then transmitted to the receiving end. That is,

$$i \rightarrow j \; : \; (R)_{K_{ij1}}$$

The receiving end decrypts the cipher using K_{ji1} as the DES key in ECB mode to obtain the pseudo-random number R. Then he modifies the number R by adding 1 to it and encrypts (R+1) under the ECB mode of DES with K_{ji2} and transmits to user i.
That is,

$$j \rightarrow i \; : \; (R+1)_{K_{ji2}}$$

This operation is done by user j to prove his identity to user i which requires the knowledge of the common key $K_{ij} = K_{ji}$. The user i at the sending end decrypts the information and tests to see if the message is equal to R+1. If this is the case, then the user at the receiving end

- 286 -

must have computed the key $K_{ji} = K_{ij}$ and hence it must be user j as he is the only one who knows the user j's secret key. Hence user i can start a conversation with user j at the other end. If they do not match, then the user at the receiving end is informed that the conversation cannot begin. This may either be due to some error during the communication or more seriously due to the false identity of user j at the receiving end.

In this simple procedure, the dynamically generated pseudo-random number is used as the session key to be used in the DES algorithm for encrypting data. Note that the number R is never actually transferred over the link between users i and j in its plain form. If the DES algorithm is used in its ECB mode for encrypting data messages no more initialization procedure is required. The users i and j can communicate with each other in a secure manner using the DES system with R as the secret DES key. On the other hand, if the DES is used in either CFB or CBC modes, then the initialization vector needs to be transferred from the sender to the receiver. This is done using the normal procedure explained in Section 5.3.2, by generating another pseudo-random number, encrypting it under the ECB mode using the session key and transmitting it to the receiver.

To authenticate the identity of user i and establish an initialization vector for transfers from j to i, user j generates the pseudo-random number which he then sends to i in encrypted from. Establishment of authentication of user i proceeds in a similar fashion as described above by the modification of the random number by user i. Note that user authentication in such a public key distribution system is based on the possession of the secret key which the user employs to compute the common key for any other user. It is the calculation of the common key that results in user authentication and DES protection.

A set of 10 pairs of public/secret keys used in this demonstration system is given in Figure 13.2. The public keys although they do not have to be kept secret, they still have to be authentic. Otherwise an imposter could manufacture his own public key, claim it belonged to another user and then employ it to impersonate him. Thus public keys have to be public in the broadest sense - not only non-secret, but guaranteed,'accessible to or shared by all users of the network' [78]. Three possible methods by which the public keys can be distributed in this hybrid DES/PKD system are considered [75].

- 287 -

System Base   a = 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

System Polynomial, f(x) : $x^{127} + x + 1$

User No.	Secret key (Bytes)	Public key (Hexadecimal code)
1	ASDFGHJKLMNBVCXZ	FEDB15A175Ø2AF4E13C716788663ØCD41
2	8765432187654321	9F6AC7C3DD227686F39749ØCFA53DE96
3	QWERTYUIOPLKJHGF	C82C8F4781B7CC85DED399F7Ø7DED4DØ
4	ØPL,MKO98IJNBHU7	9Ø52E6CØE3DCAEØ326F82DB99FC9EAØ9
5	1QAZXSW23EDCVFR4	228Ø521679EBCE31F29227A3194BA2F2
6	5TGB6YHN7UJM8ZA1	18EDF7ØF67CF4F21158EE37BE2F8Ø725
7	!"#$&'()Ø*=@+<>	91314E65CF6D5231D258A8BAØ6D3E3AØ
8	1!2"3#4$5%6&7'8(	12872666521B7B15Ø338581F6EBB83ØC
9	AZSXDCFVGBHNJMKL	619Ø5Ø8319F9Ø5A1CØAD2ØA7FF92BBCA
10	W28DF6&N^MTG!RAM	4877538FFEØ3934F7B8DE1Ø924CØFE94

Fig. 13.2 – Public & Secret Key Pairs in $GF(2^{127})$

13.5.1 Central Public Key File

In this approach, as in Section 9.6, the existence of a
Public Key Distribution Centre (PKDC) is envisaged which controls the
formation of user connections. The connection protocol sequence may
be described as follows. Each user registers his public key with the
PKDC and each user knows the public key of the Centre. When a user i
wishes to communicate with user j, he first enters his secret key on
the terminal. This key can then be combined with the PKDC's public
key, which is assumed to be locally stored on the terminal, to form a
common key between the user and the PKDC. Then the user can connect
with the PKDC, using the common key in his DES based system, to
transmit an encrypted request to the PKDC for user j's public key (or
the entire list of public keys). The PKDC decrypts the request and
encrypts and transmits the desired public key(s) to the user i. Hence
the connection to the PKDC is authentic and private so that an opponent
cannot modify the public keys transmitted or impersonate the PKDC
without detection. This approach insures that the PKDC controls the
access of users to the system and centralizes the dissemination of
public keys. Note that a connection to the PKDC is required to obtain

- 288 -

a list of public keys but once the user has stored the public keys,
the PKDC is no longer needed unless the user wishes to update his
public key list.

13.5.2 Local Public_Key_File

This method has been used in the simple demonstration system
described earlier and is very attractive when only a small number of
users are involved. Each user possesses an authentic list of public
keys. The distribution of this local directory can be accomplished
manually through hardcopy or electronic storage media such as
programmable memory PROM. Precautions must be taken to protect the
public keys from modification or substitution but not from privacy.

13.5.3 No_Public Key File_

The third approach assumes a more benign environment in which
the opponent is content to passively eavesdrop. If this applies, keys
do not need to be stored. To establish a connection, each user
generates a new secret key, computes a new public key, exchanges it and
then calculates the secret session key. An active opponent however
can interpose between the two users, can mirror each half of the
scenario and establish an imposter connection with each. This method
is mentioned here for completeness sake and it is recommended that it
should be avoided in practice.

Thus this arrangement shows that the implementation of the
PKD algorithm and its use in a hybrid system using the DES is entirely
feasible. The value of combining the protection provided by the
conventional cryptosystem with the user authentication attributes of
a public key system is most advantageous. The integration of these two
methods gives the designer of secure systems flexibility in these areas:

1.   The distribution and management of keys; while most con-
        ventional cryptosystems require centralized key management
        and connection establishment, integration of public keys
        systems with conventional cryptosystems promises centralized
        control of key management functions with distributed connection
        establishment.

2.   Decentralized user authentication; since a user's identity
        can be confirmed using public parameters and a single secret
        parameter known only to the user.

3.   Document or file protection; the conventional cryptosystem

- 289 -

can be used to protect against information release by encryption using document or file key while access to the file key is restricted by the public key system.

13.6   Exponentiation in $GF(2^n)$

In this section two different methods of performing exponentiation in $GF(2^n)$ are considered. These indicate that fast exponentiation in $GF(2^n)$ is possible using dedicated hardware and make the DES/PKD hybrid system described above a practical way of providing both security and authentication.

Both the methods, use the well known 'square and multiply' technique to perform exponentiation in $GF(2^n)$. This technique has been used throughout Chapters 10, 11 and 12 and the flowchart is given in Figure 10.2. But in this case, the operations are performed modulo 2 which makes the implementation easy and reduces the running time of the algorithm. The two methods differ in the way they perform multiplication and squaring in $GF(2^n)$.

13.6.1 Method 1

13.6.1.1 Squaring

Squaring operation in Galois field can be performed very efficiently if the irreducible generator polynomial $f(x)$ is fixed as shown below.

Let the irreducible polynomial be a trinomial, $f(x) = x^n + x + 1$. (In the demonstration hybrid DES/PKD system described in Section 13.5, $f(x) = x^{127} + x + 1$). Let 'a' represent an element of the field $GF(2^n)$ (for example, the system base) and representing 'a' as a binary polynomial

$$a = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \qquad a_i \in GF(2)$$

As $GF(2^n)$ is a field with characteristic 2, the property that

$$(a+b)^2 = a^2 + b^2$$

holds true for the operations in the field. Thus the representation of a^2 is given by

$$a^2 = a_0 + a_1 x^2 + a_2 x^4 + \ldots + a_{n-1} x^{2(n-1)}$$

Now reducing the powers greater than n using the recursive function $x^{n+j} = x^{j+1} + x^j$ for $j \geqslant 0$, gives

$$a^2 = a_0 + a_{n/2} x + (a_1 + a_{n/2}) x^2 + \ldots + a_{n-1} x^{n-2} + (a_{n/2} + a_{n-1}) x^{n-1}$$

where $n/2$ denotes $\lceil n/2 \rceil$, the least integer greater than $n/2$.

- 290 -

As the characteristic of the field is 2, one can represent a^2 using exclusive-or operator $\oplus$

$$a^2 = a_0 + a_{n/2}\, x + (a_1 \oplus a_{n/2})x^2 + \ldots + a_{n-1}x^{n-2} + (a_{n/2} \oplus a_{n-1})\, x^{n-1}.$$

This expression can be rewritten as

$$a^2 = a_0 + \sum_{i=1}^{g} \{a_{i+g}x^{2i-1} + (a_i \oplus a_{i+g})\, x^{2i}\} \tag{13.2}$$

where $g = n/2 - 1$.

Implementation of the equation (13.2) from the hardware point of view gives
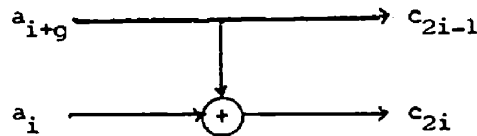
where

$$(c_0, c_1, \ldots, c_{n-1}) = (a_0, a_1, \ldots, a_{n-1})^2$$

Thus squaring can be very efficiently accomplished using exclusive-or gates alone. Such an implementation in $GF(2^7)$ using 7-bit vector is shown in Figure 13.3.
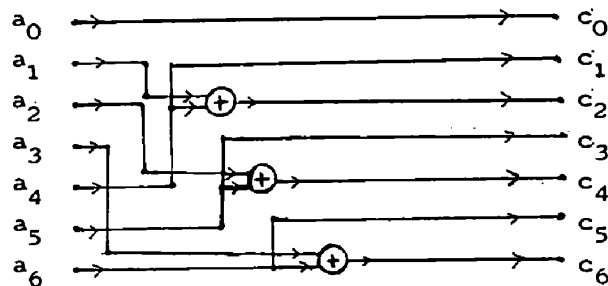
Fig 13.3-Squaring in $GF(2^7)$

13.6.1.2 Multiplication

The multiplication in $GF(2^n)$ can be performed using the standard canonical basis representation of elements. That is, the elements of $GF(2^n)$ are expressed in terms of a canonical basis for $GF(2^n)$ over $GF(2)$ and the multiplication rule is derived as follows:

Suppose that U and V in $GF(2^n)$ have the representation $(u_0, u_1, \ldots u_{n-1})$ and $(v_0, v_1, \ldots, v_{n-1})$ respectively in terms of the canonical basis $(1, \alpha, \ldots, \alpha^{n-1})$ where α is a root of an irreducible polynomial of degree n over $GF(2)$. This means that

$$U = [u_0, u_1, u_2, \ldots, u_{n-1}] \begin{bmatrix} 1 \\ \alpha \\ \vdots \\ \alpha^{n-1} \end{bmatrix}$$

- 291 -

and

$$V = [v_0, v_1, \ldots, v_{n-1}] \begin{bmatrix} 1 \\ \alpha \\ \cdot \\ \cdot \\ \alpha^{n-1} \end{bmatrix} = [1, \alpha, \ldots, \alpha^{n-1}] \begin{bmatrix} v_0 \\ v_1 \\ \cdot \\ \cdot \\ v_{n-1} \end{bmatrix}$$

Hence $Z = U.V$

$$= [u_0, u_1, \ldots, u_{n-1}] \begin{bmatrix} 1 \\ \alpha \\ \cdot \\ \cdot \\ \alpha^{n-1} \end{bmatrix} [1, \alpha, \ldots, \alpha^{n-1}] \begin{bmatrix} v_0 \\ v_1 \\ \cdot \\ \cdot \\ v_{n-1} \end{bmatrix}$$

$$Z = [u_0, u_1, \ldots, u_{n-1}] \begin{bmatrix} 1 & \cdots & \alpha^{n-1} \\ \alpha & & \cdot \\ \cdot & & \cdot \\ \alpha^{n-1} & \cdots & \alpha^{2n-2} \end{bmatrix} \begin{bmatrix} v_0 \\ \cdot \\ \cdot \\ v_{n-1} \end{bmatrix} \qquad (13.3)$$

Now expanding the $n \times n$ matrix in (13.3) as

$$\begin{bmatrix} 1 & \cdots & \alpha^{n-1} \\ \alpha & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \alpha^{n-1} & \cdots & \alpha^{2n-2} \end{bmatrix} = M_0 + M_1 \alpha + M_2 \alpha^2 + \ldots + M_{n-1} \alpha^{n-1}$$

where M_k is the GF(2) matrix whose entry in row i and column j is the coefficient of α^k when α^{i+j-2} is expanded in the canonical basis $(1, \alpha, \ldots, \alpha^{n-1})$. Hence if $(z_0, \ldots, z_{n-1})$ is the representation of Z we have

$$z_0 + z_1 \alpha + \ldots + z_{n-1} \alpha^{n-1} = \underset{\sim}{u}^t M_0 \underset{\sim}{v} + \underset{\sim}{u}^t M_1 \underset{\sim}{v} \alpha + \ldots + \underset{\sim}{u}^t M_{n-1} \underset{\sim}{v} \alpha^{n-1}$$

where

$\underset{\sim}{u} = (u_0, u_1, \ldots, u_{n-1})$ and $\underset{\sim}{v} = (v_0, v_1, \ldots, v_{n-1})$.

It now follows from the uniqueness of the representation in terms of a fixed basis that

$$z_k = \underset{\sim}{u}^t M_k \underset{\sim}{v} \qquad \text{for } k = 0, 1, \ldots, n-1 \qquad (13.4)$$

The right hand side of (13.4) is sometimes referred to as the bilinear form in the vectors $\underset{\sim}{u}$ and $\underset{\sim}{v}$.

To avoid the need to store the matrices M_k in the calculation of the product of two elements, the algorithm given by Berlekamp in [56] has been used.

Let the multiplicand be U and the multiplier be V and assume

- 292 -

that they are stored in two n-bit registers U and V respectively. Let the partial product register be called Z. To perform multiplication, the Z register is initially set to zero. Depending on the lowest bit of the multiplicand v_0, U is either added or not added into Z (modulo 2 addition). If $v_0 = 1$, then U is added into Z; if $v_0 = 0$, Z is left unchanged. The V register is then shifted right, the U register is multiplied by α and the process is repeated. Multiplication by α is done according to the recursive equation $\alpha^n = \alpha + 1$. At the mth step, U contains α^m times the original multiplicand, v_0 contains v_m, the mth bit of the original multiplier and Z contains $\sum_{i=0}^{m} v_i(U\alpha^i)$. After n such steps, the multiplication is complete. Z contains the product of the original U and V registers. The V register has been cycled completely around to its original position but the U register now contains $U\alpha^{n-1}$. A schematic diagram using feedback shift registers is shown in Figure 13.4 for the field $GF(2^7)$ with $f(x) = x^7 + x + 1$ as the generating irreducible polynomial.
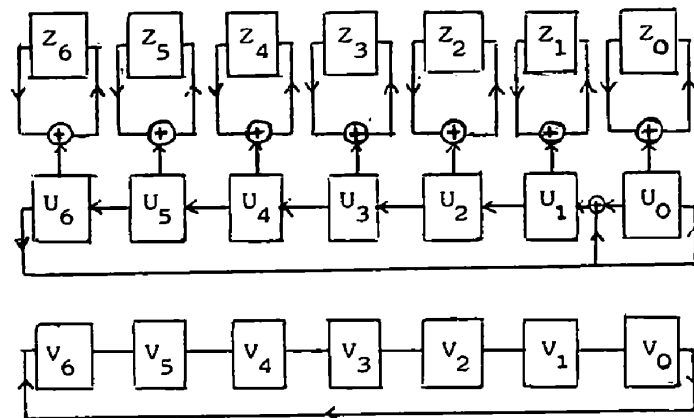
Fig 13.4 – Multiplication in $GF(2^7)$

This method of multiplying and squaring elements in $GF(2^n)$ has been used in the software implementation of the PKD algorithm in the hybrid system. With dedicated hardware, the exponentiation algorithm would take approximately 16256 (=127 x 128) clock cycles in the worst case. Hence using 1MHZ clock, this gives a worst running time of less than 20 milliseconds compared to 6 seconds when carried out in software using 6502 machine code programming.

13.6.2 Method_2_

Here exponentiation is performed in $GF(2^n)$ using a novel

– 293 –

technique based on normal basis representation of elements [79]. This section contains some of the notes communicated by Prof J Massey, ETH, Zurich, in private correspondence. The method has been used to design a hardware exponentiation system in $GF(2^7)$.

13.6.2.1 Multiplication

The multiplication rule given in method 1 can be used with any basis not only with the canonical basis.

Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be any basis for $GF(2^n)$ over $GF(2)$ and let $\underline{u} = (u_1, \ldots, u_n)$, $\underline{v} = (v_1, v_2, \ldots, v_n)$ and $Z = (z_0, z_1, \ldots, z_n)$ be the representations of U, V and Z in terms of this basis. It follows that if Z = UV, then

$$Z = \begin{bmatrix} u_1, & \ldots, u_n \end{bmatrix} \begin{bmatrix} \alpha_1^2 & \ldots & \alpha_1\alpha_n \\ \alpha_2\alpha_1 & & \bullet \\ \bullet & & \bullet \\ \bullet & & \bullet \\ \bullet & & \bullet \\ \alpha_n\alpha_1 & \ldots & \alpha_n^2 \end{bmatrix} \begin{bmatrix} v_1 \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ v_n \end{bmatrix} \qquad (13.5)$$

The matrix in (13.5) can be expanded as

$$\begin{bmatrix} \alpha_1^2 & \ldots & \alpha_1\alpha_n \\ \bullet & & \bullet \\ \bullet & & \bullet \\ \bullet & & \bullet \\ \alpha_n\alpha_1 & \ldots & \alpha_n^2 \end{bmatrix} = M_1\alpha_1 + M_2\alpha_2 + \ldots + M_n\alpha_n$$

where M_k is the GF(2) matrix whose entry in row i and column j is the coefficient of α_k when $\alpha_i\alpha_j$ is expanded in the basis $\alpha_1, \alpha_2, \ldots, \alpha_n$. It follows that

$$z_k = \underline{u}^t M_k \underline{v} \qquad \text{for } k = 1, 2, \ldots, n.$$

In particular let us now consider a special basis called the normal basis.

Suppose that E is an extension of the field F and E is a vector space of dimension n over F. Then $\alpha_1, \alpha_2, \ldots, \alpha_n$ is said to be a normal basis for E over F if $\alpha_1, \ldots, \alpha_n$ are a basis for E over F and are also roots of the same irreducible polynomial in $F[x]$. For F = GF(2) and $E = GF(2^n)$, this is equivalent to saying that for some element α of $GF(2^m)$, $\alpha_i = \alpha^{2^{i-1}}$ for $i = 1, 2, \ldots, n$ are linearly independent over GF(2). Now letting $[u_0, u_1, \ldots, u_{n-1}]$ be the representation of $U \in GF(2^n)$ in terms of the normal basis $\alpha, \alpha^2, \ldots \alpha^{2^{n-1}}$, then

– 294 –

$$U = u_0\alpha + u_1\alpha^2 + \cdots + u_{n-2}\alpha^{2^{n-2}} + u_{n-1}\alpha^{2^{n-1}}$$

so that

$$U^2 = u_0\alpha^2 + u_1\alpha^4 + \cdots + u_{n-2}\alpha^{2^{n-1}} + u_{n-1}\alpha$$

where the identity $u^2 \equiv u$ for all $u \in GF(2)$ and the fact that cross-term products vanish when raising to the power 2 in $GF(2)$ have been used. Thus it is seen that squaring an element U in $GF(2^n)$ merely consists of a right cyclic shift of its representation in terms of a normal basis for $GF(2^n)$ over $GF(2)$. In a general extension field $GF(q^n)$ where q is a prime, raising to the qth power an element of $GF(q^n)$ corresponds to a right cyclical shift of its representation in terms of a normal basis for $GF(q^n)$ over $GF(q)$. Thus the implementation of squaring in 'multiply and square' technique exponentiation is made very simple using normal basis.

Now letting $\underset{\sim}{u} = (u_0, \ldots, u_{n-1})$, $\underset{\sim}{v} = (v_0, v_1, \ldots, v_{n-1})$ and $\underset{\sim}{z} = (z_0, z_1, \ldots, z_{n-1})$ be the normal basis representation of U, V and Z respectively and $Z = U \cdot V$, then

$$z_{n-1} = (u_0, u_1, \ldots, u_{n-1})^t \ M \ (v_0, v_1, \ldots, v_{n-1}) \qquad (13.6)$$

where M is the $GF(2)$ matrix whose entry in row i and column j is the coefficient of $\alpha^{2^{n-1}}$ when $\alpha^{2^{i-1}} \ \alpha^{2^{j-1}} = \alpha^{2^{i+j-2}}$ is expanded in the normal basis $\alpha, \alpha^2, \ldots, \alpha^{2^{n-1}}$. But $Z^2 = U^2 V^2$ and the elements U^2, V^2 and Z^2 have the normal basis representations $(u_{n-1}, u_0, \ldots, u_{n-2})$, $(v_{n-1}, v_0, \ldots, v_{n-2})$ and $(z_{n-1}, z_0, \ldots, z_{n-2})$ respectively. Thus it follows that for the same matrix M as in (13.6)

$$z_{n-2} = (u_{n-1}, u_0, \ldots, u_{n-2})^t \ M \ (v_{n-1}, v_0, \ldots v_{n-2})$$

and in general that, for this same $GF(2)$ matrix M

$$z_k = (u_{k+1}, \ldots, u_{n-1}, u_0, \ldots, u_k) \ M \ (v_{k+1}, \ldots v_{n-1}, v_0, \ldots v_k)$$

for $0 \leq k \leq n-1$.

Thus when a normal basis representation of $GF(2^n)$ over $GF(2)$ is employed, each digit in the product is given by the same bilinear form with appropriate cyclic shifting of the representations of the factors. This property is the key to construction of simple multipliers for finite fields.

In the PKD system, it is necessary to implement the equation (13.6) to perform multiplication in $GF(2^n)$. The expression (13.6) can be written as

$$z_{n-1} = u_{n-2} v_{n-2} + (u_0, \ldots, u_{n-1})^t \ \Lambda \ (v_0, v_1, \ldots, v_{n-1}) \qquad (13.7)$$

where the binary matrix Λ is symmetric with an all zero main diagonal.

- 295 -

This can be seen as follows:

Writing m_{ij} for the entries in M,

$$z_{n-1} = \sum_i \sum_j u_i v_j m_{ij}$$

Suppose $u_k = v_k = 1$ and $u_s = v_s = 0$ for $s \neq k$. Then $z_{n-1} = m_{kk}$. But for this case $U = V = \alpha^{2^k}$ which implies that $Z = UV = \alpha^{2^{k+1}}$. Thus $z_{n-1} = 1$ if and only if $k = n-2$. Hence $m_{kk} = 1$ if and only if $k = n-2$, so that the next to last entry on the main diagonal of M is the only one which is non-zero.

Because of its symmetry and zero diagonal, the nxn matrix Λ can be written as

$$\Lambda = \begin{bmatrix} 0 & & & \\ \cdot & & & \\ \cdot & T & \\ \cdot & & & \\ 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ & & \cdot \\ & T^t & \cdot \\ & & \cdot \\ & & 0 \end{bmatrix}$$

where T is the binary upper triangular $(n-1) \times (n-1)$ matrix.

$$T = \begin{bmatrix} t_{01} & t_{02} & \cdots & t_{0,n-1} \\ 0 & t_{12} & & t_{1,n-1} \\ \cdot & & \ddots & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ 0 & \cdots & & t_{n-2,n-1} \end{bmatrix}$$

Thus,

$$z_{n-1} = u_{n-2} v_{n-2} + (u_0, \ldots, u_{n-2})^t \, T \, (v_1, \ldots, v_{n-1})$$

$$+ [T(u_1, \ldots, u_{n-1})]^t \, (v_0, v_1 \ldots v_{n-2}) \tag{13.8}$$

Using (13.8) one particular implementation of the multiplication function is shown in Figure 13.5. The boxes labelled T contain Exclusive-or gates only. Each component of the output is that sum of input bits corresponding to the locations of the '1's in the corresponding row of the matrix T. The outputs of this box are then And-ed with the appropriate components of the other factor. The outputs of these n-1 And gates are then summed by a tree of n-2 Exclusive-or gates. The outputs of the two Exclusive-or trees are summed and added to the term $u_{n-2} v_{n-2}$. Alternatively, the two Exclusive-or trees and the Exclusive-or gate that combines their outputs can be replaced by a single Exclusive-or tree with 2n-2 gates having 2n-1 inputs. The circuit shown in Figure 13.5 computes z_{n-1}. If U and V are shifted as shown in the diagram, the same circuit will compute $z_{n-2}, z_{n-3}, \ldots, z_1, z_0$ during the

- 296 -
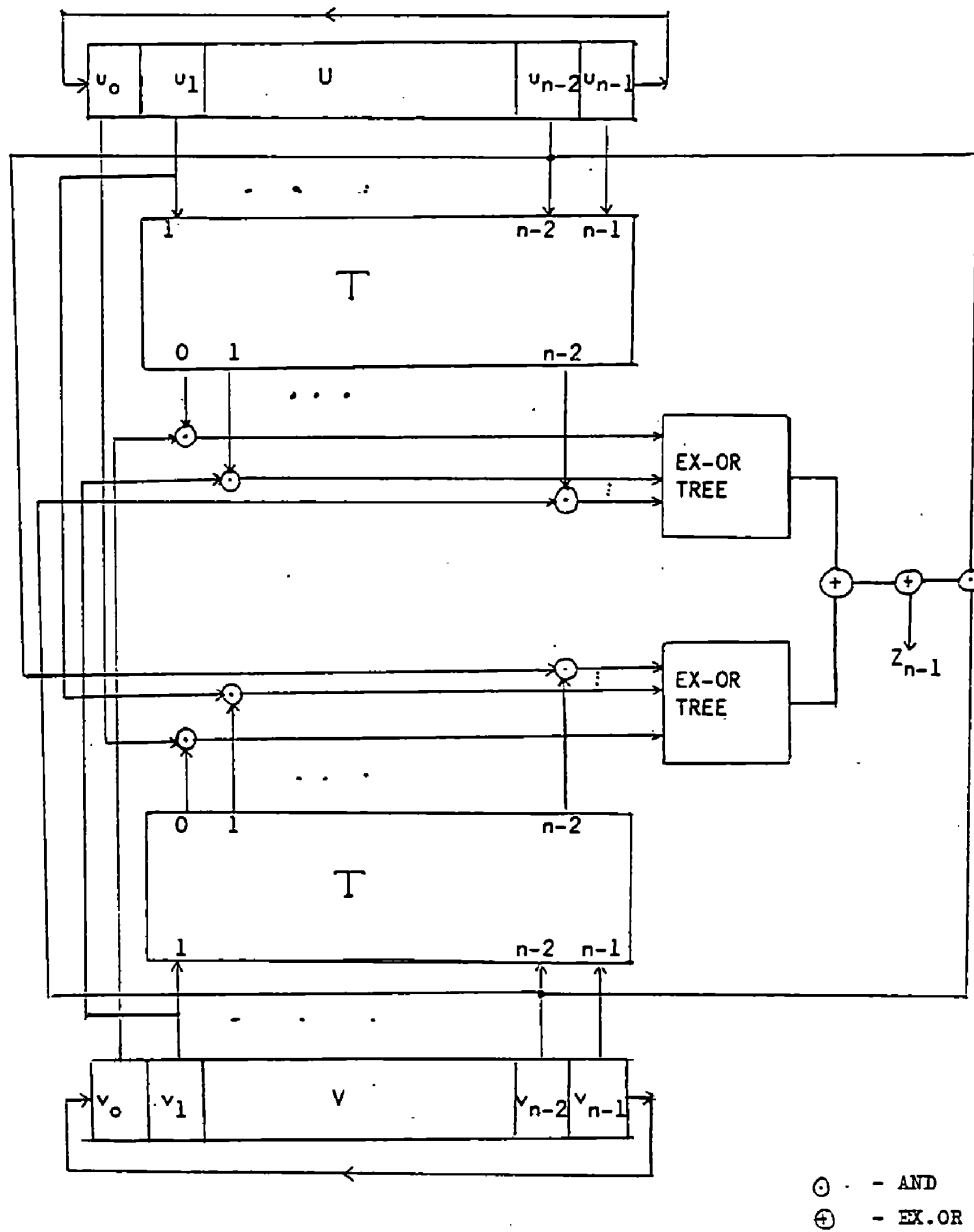
Fig. 13.5 - Multiplier Configuration using T-matrix

⊙ · – AND

⊕ – EX.OR

– 297 –

next n-1 subsequent shifts.

A different implementation of the multiplication function is based on the alternating property of Λ.

Let B denote any n×n matrix in any field F and let $\underset{\sim}{u}$ and $\underset{\sim}{v}$ be any vectors in F^n. Then the bilinear form $\underset{\sim}{u}^t B \underset{\sim}{v}$ defined by B is said to be alternating if it vanishes whenever $\underset{\sim}{u} = \underset{\sim}{v}$, that is, if $\underset{\sim}{u}^t B \underset{\sim}{u} = 0$ for all $\underset{\sim}{u}$ in F^n. Two n×n matrices over F, B and D, are said to be congruent if there is an invertible matrix P such that

$$B = P^t D P$$

Then

$$\underset{\sim}{u}^t B \underset{\sim}{v} = (P\underset{\sim}{u})^t D (P\underset{\sim}{v})$$

Thus the bilinear form $\underset{\sim}{u}^t B \underset{\sim}{v}$ can be evaluated using the matrix D if the basis is changed by the transformation, P^{-1}. When D has only a small number of non-zero entries, the evaluation of $(P\underset{\sim}{u})^t D (P\underset{\sim}{v})$ requires only a small number of multiplications and additions beyond those needed to form $P\underset{\sim}{u}$ and $P\underset{\sim}{v}$.

In [48], a binary matrix B which is symmetric and has an all zero main diagonal is shown to be alternating. Thus the matrix Λ which defines the normal basis multiplier is alternating. The rank of the matrix Λ is n when n is even and is n-1 when n is odd. The matrix Λ can be reduced to a diagonal matrix D using the elementary divisor method given in Section 12.2.4. Thus for the n×n matrix Λ, there exists elementary matrices E_1, E_2, ..., E_m (for some m) such that

$$E_m \ldots E_1 \Lambda E_1^t \ldots E_m^t = D$$

where D is a diagonal matrix with n non-zero entries when n is even or n-1 non-zero entries when n is odd. Thus

$$P^t = E_m \ldots E_1$$

The number of '1's present in the P-matrix is indicative of the number of Exclusive-or gates required to implement this matrix. Hence using this P-box, one can implement the multiplier function as

$$z_{n-1} = u_{n-2} v_{n-2} + (P\underset{\sim}{u})^t D (P\underset{\sim}{v})$$

This configuration is shown in Figure 13.6. The connections in the P-box are determined by the P-matrix and the connections to the And gates are determined by the D-matrix. Subsequent shifts of $\underset{\sim}{u}$ and $\underset{\sim}{v}$ vectors will produce $z_{n-2}, \ldots, z_0$.

As a final method of implementing the multiplier function $z_{n-1} = \underset{\sim}{u}^t M \underset{\sim}{v}$, the 'brute-force' approach is considered. This approach consists of first forming the vector $M\underset{\sim}{v}$ and then And-ing

- 298 -
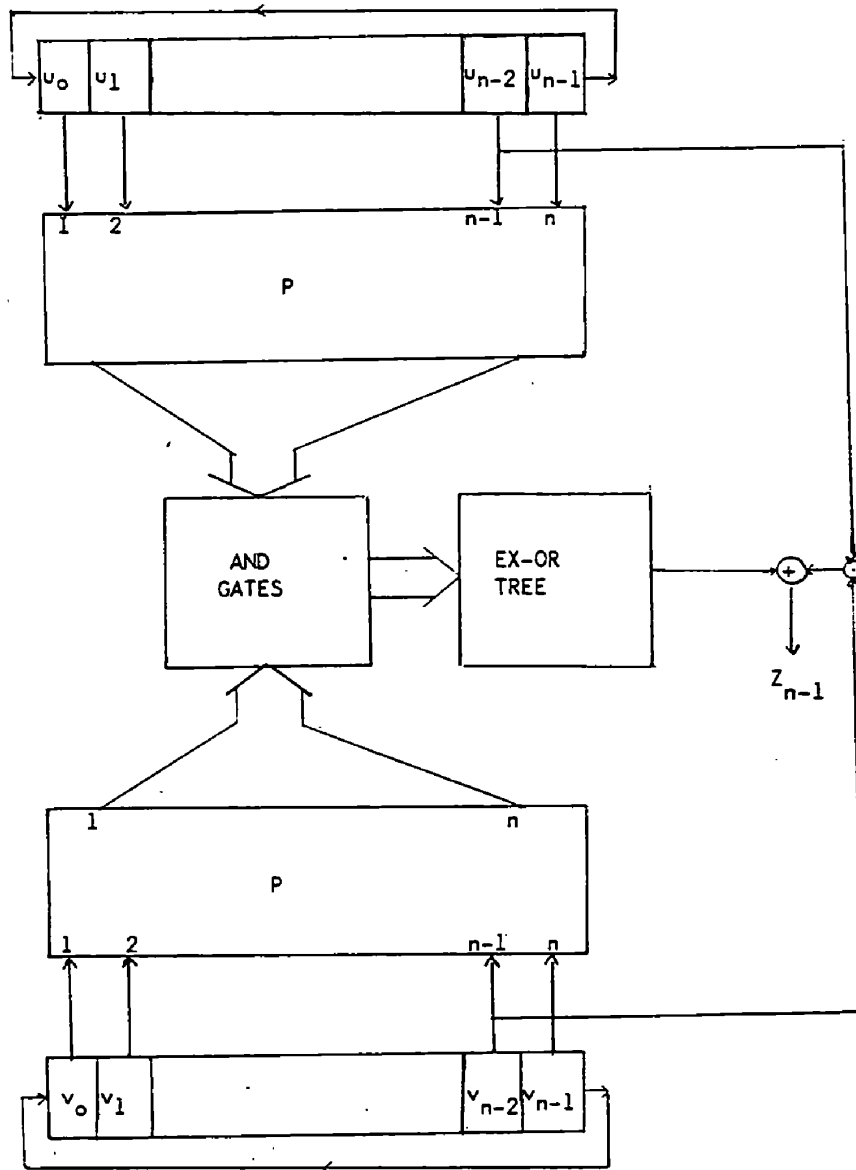
Fig. 13.6 - Multiplier Configuration using P matrix

- 299 -

each of its components with the corresponding component of $\underline{u}$, then finally summing the outputs of the n And gates. This approach requires n And gates and n-1 Exclusive-or gates in addition to the Exclusive-or gates needed to form $M\underline{v}$. Compared to the previous two methods, this one lacks the 'modularity' - several sub-circuits of the same kind.

13.7   Hardware Design of An Exponentiator in GF(2^7)

A hardware exponentiation system in GF(2^7) has been designed using the second method which involves the normal basis representation. To begin with, a program is written to search for a generator of a normal basis in GF(2^7). This consists of finding seven conjugates in GF(2^7) which are linearly independent over GF(2). A complete listing of the program is given in Appendix 19. The steps involved in this algorithm are given as follows:

1.   Choose an element $\alpha \in$ GF(2^n)

2.   Compute $\alpha_i = \alpha^{2^{i-1}}$ for i=1,..., n in GF(2^n) using the irreducible polynomial $f(x) = x^n + x + 1$.

3.   Form a n×n matrix using the n-bit vectors $\alpha_1,..., \alpha_n$.

4.   Calculate the determinant of the matrix over GF(2).

5.   If determinant $\equiv$ 0 in GF(2), Go to step 1.

6.   If determinant $\not\equiv$ 0 in GF(2), then α generates a normal basis.

In this case n= 7. Initially, it is decided to look at the elements α of the form $\alpha = x^a + 1$ for $1 \leq a \leq n-1$. One such generator α is found to be $\alpha_1 = x^3 + 1$. Then the powers $\alpha_i = \alpha^{2^{i-1}}$ for i = 1,2,..., 7 modulo ($x^7 + x + 1$) are given by:

$\alpha_1 = x^3 + 1, \alpha_2 = x^6 + 1, \alpha_3 = x^6 + x^5 + 1, \alpha_4 = x^6 + x^5 + x^4 + x^3 + 1,$

$\alpha_5 = x^5 + x^4 + x^3 + x^2 + x + 1, \alpha_6 = x^6 + x^3 + x + 1$ and $\alpha_7 = x^5 + x^2 + 1.$

The symmetric matrix of the bilinear form is then given by

x^6+1	x^6+x^2+1	x^6+x^5+x+1	$x^5+x^4+x^3$	x^6+x	$x^4+x^3+x^2+x+1$	x^3+x+1
*	x^6+x^5+1	$x^6+x^5+x^4+1$	$x^6+x^5+x^4+x^3+x^2+1$	$x^6+x^4+x^3+x^2+x$	$x^6+x^5+x^2$	x^6+x^4+x+1
*	*	$x^6+x^5+x^4+x^3+1$	$x^6+x^5+x^4+x^3+x^2+x+1$	$x^5+x^3+x^2+x+1$	x^5+x^4+x	$x^6+x^5+x^3$
*	*	*	$x^5+x^4+x^3+x^2+x+1$	x^5+x^3+x+1	$x^6+x^3+x^2+1$	x^4+x^3+x
*	*	*	*	x^6+x^3+x+1	$x^6+x^4+x^3+x^2+1$	x^5+x^4+1
*	*	*	*	*	x^5+x^2+1	$x^5+x^4+x^2+x+1$
*	*	*	*	*	*	x^3+1

* - indicates symmetry

- 300 -

Rewriting the above matrix in terms of the normal basis, gives

$$\begin{bmatrix} \alpha_2 & \alpha_4+\alpha_5+\alpha_6+\alpha_1+\alpha_2 & \alpha_5+\alpha_7+\alpha_4 & \alpha_4+\alpha_2 & \alpha_1+\alpha_6 & \alpha_2+\alpha_3+\alpha_5 & \alpha_1+\alpha_3+\alpha_4+\alpha_5+\alpha_7 \\ \bullet & \alpha_3 & \alpha_2+\alpha_3+\alpha_5+\alpha_6+\alpha_7 & \alpha_1+\alpha_6+\alpha_5 & \alpha_5+\alpha_3 & \alpha_2+\alpha_7 & \alpha_3+\alpha_4+\alpha_6 \\ \bullet & \bullet & \alpha_4 & \alpha_1+\alpha_3+\alpha_4+\alpha_6+\alpha_7 & \alpha_2+\alpha_6+\alpha_7 & \alpha_4+\alpha_6 & \alpha_1+\alpha_3 \\ \bullet & \bullet & \bullet & \alpha_5 & \alpha_1+\alpha_2+\alpha_4+\alpha_5+\alpha_7 & \alpha_1+\alpha_3+\alpha_7 & \alpha_5+\alpha_7 \\ \bullet & \bullet & \bullet & \bullet & \alpha_6 & \alpha_1+\alpha_2+\alpha_3+\alpha_5+\alpha_6 & \alpha_1+\alpha_2+\alpha_4 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \alpha_7 & \alpha_2+\alpha_6+\alpha_3+\alpha_4+\alpha_7 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \alpha_1 \end{bmatrix}$$

• - indicates symmetry

Expanding the above matrix in terms of $\alpha_1, \alpha_2, \ldots, \alpha_7$, yields

$$M = M_1\,\alpha_1 + M_2\,\alpha_2 + M_3\,\alpha_3 + M_4\,\alpha_4 + M_5\,\alpha_5 + M_6\,\alpha_6 + M_7\,\alpha_7$$

where

$$M_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

and

M_2 is obtained by rotating the rows of M_1 downward by 1 position. and then rotating the columns right by 1 position. Similar operations on M_2 yield M_3 and so on. In the implementation of the multiplier function

$M = M_7$ is used. The T-matrix is therefore given by

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The T-box can therefore be realized as
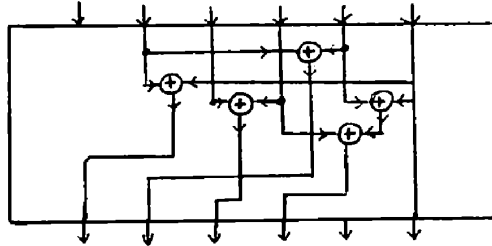
Note that one of the six outputs is not used and is identically equal
to 0. The And gate is therefore unnecessary that is fed by this 0
output. Thus the complete circuit requires 2 T-boxes with 5 Exclusive-
or gates each, 2 sets of 5 And gates operating on their outputs, 2
Exclusive-or trees with 4 Exclusive-or gates each and 2 additional
Exclusive-or gates and 1 And gate for producing the final output. The
total gate count for the multiplier function is equal to 20 Exclusive-
or gates and 11 And gates.

Using the brute force method the multiplier function can be
implemented as follows:
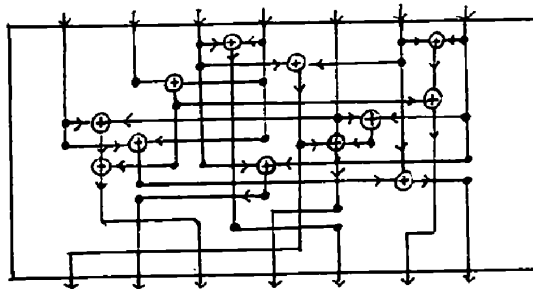
The number of gates required using this approach is equal to 18
Exclusive-or gates and 7 And gates. It appears to give the minimum
number of gates of the three realizations; but the realization using
the T-matrix or the P-matrix is preferrable for large extension fields
because then the circuit is composed of two identical subcircuits, that
is, a modular design is achieved.

- 302 -

The circuit diagram of the designed complete exponentiation system in $GF(2^7)$ is given in Appendix 20. Here the multiplier function has been implemented using the 'brute-force' approach.

As the number of '1's present in the M-matrix give an indication of the number of Exclusive-or gates required to implement the multiplier function, to minimise the number of gates, one needs to reduce the number of '1's in the M-matrix to a minimum. In an effort to obtain such an optimum M-matrix, various normal basis generators α with four different irreducible polynomials are tried in $GF(2^7)$ and the number of '1's in their corresponding M-matrices are calculated. The results are given in Figure 13.7. From the results it is seen that

a) the difference between the minimum and the maximum number of '1's for the tried cases is not large, that is, the variance does not appear to be high.

b) The average number of '1's in the M-matrix is approximately equal to 23 which is roughly equal to $n^2/2 = \frac{49}{2} \doteq 24$.

This may imply that for a randomly chosen normal basis generator and an irreducible polynomial, the number of '1's is approximately equal to half the entries in the M-matrix. It appears that the random choice of normal basis generators and different irreducible polynomials does not seem to yield any substantial reduction in the number of '1's in the M-matrix. The above claim should be read with caution as this is based on a small number of trials in a small extension field $GF(2^7)$.

13.8   Normal Basis Generators in $GF(2^{127})$

From cryptography point of view, one is interested in large extension fields namely $GF(2^{127})$ or $GF(2^{521})$. The next step is therefore to determine the generators of normal basis in these extension fields which can then be used in the PKD exponentiation system. Albert [48] proved that if F is a subfield of E and E is normal over F, then E has a normal basis over F. Thus, $GF(2^n)$ has a normal basis over GF(2) for all n. Although the theorem by Albert establishes the existence of a normal basis for any field $GF(2^n)$, it does not give any help to determine the generators of normal basis in practice. As no systematic method for finding these generators was apparently evident, it is decided to resort to the trial and error procedure using the algorithm given in Section 13.7. However, let us first consider the probability of finding a normal basis using this random search procedure.

- 303 -

Irreducible polynomial of degree 7 over GF(2)	Normal Basis Generator	No. of '1's in the M-matrix	
X^7+X+1	X^3+1	21	(1)†
X^7+X+1	X^5+1	27	(2)
X^7+X+1	X^3+1	25	(3)
X^7+X+1	X^5+X+1	27	(4)
X^7+X+1	X^6+X+1	19	(5)
X^7+X+1	X^6+X^2+1	25	(6)
$X^7+X^6+X^5+X^6+X^2+X+1$	X^4+X+1	21	(7)
X^7+X+1	X^6+1	21	(1)
X^7+X+1	X^5X^2+1	21	(1)
X^7+X+1	X^3+X^2+1	21	(7)
$X^7+X^6+X^5+X^4+X^2+X+1$	X^2+X+1	19	(5)
	X^6+X+1	27	(4)
	X	27	(2)
	X^2	27	(2)
	X^4	27	(2)
	X^3	19	(5)
	X^6	19	(5)
$X^7+X^3+X^2+X+1$	X^5	27	(4)
	$X+1$	27	(2)
	X^2+1	27	(2)
	X^3+1	25	(3)
	X^4+1	27	(2)
	X^6+1	25	(3)
	X^2+X+1	19	(5)
	X^3+X+1	25	(3)
	X^4+X+1	21	(7)
	X^6+X+1	19	(5)

† Same numbers in brackets indicate same M-matrix

Fig. 13.7 - Table showing some normal basis generators
in $GF(2^7)$ and the number of '1's in
the corresponding M-matrices

- 304 -

PMC Exhibit 2117
Apple v. PMC
IPR2016-00755
Page 317

From [56, Theorem 11.39], the number of elements in $GF(q^m)$ which have m linearly independent conjugates over $GF(q)$ is given by

$$q^m \prod_k (1 - q^{-d_k})$$

where the d_k are the degrees of the distinct irreducible factors of x^m-1 over $GF(q)$.

For the extension field $GF(2^{127})$

$$x^{127}-1 = \prod_{d|127} Q^{(d)}(x)$$

where $Q^{(d)}(x)$ denotes the cyclotomic polynomial.

That is,

$$x^{127}-1 = Q^{(1)}(x)\ Q^{(127)}(x)$$

$Q^{(1)}(x)$ is an irreducible polynomial of degree 1. The degrees of the irreducible factors of $Q^{(127)}(x)$ are determined using the following result [56] :

Since every element of order n has the same number of conjugates with respect to $GF(2)$, every irreducible factor of the cyclotomic polynomial $Q^{(n)}(x)$ has the same degree over $GF(2)$. This degree is the multiplicative order of 2 modulo n.

For the case n = 127, the multiplicative order of 2 modulo 127 is 7 as $2^7 \equiv 1$ (mod 127). Thus every irreducible factor of $Q^{(127)}(x)$ has degree 7 over $GF(2)$. Hence the probability of finding a normal basis in $GF(2^{127}) = (1-\frac{1}{2})(1-\frac{1}{2^7})^{18} \doteq 0.434$.

The program FINDNORBAS.F77 in Appendix 19 is used to determine a normal basis generator in $GF(2^{127})$ using the random search algorithm. Initially it is again decided to consider the elements of the form $\alpha = x^a+1$ for $1 \leq a \leq 126$. The operations are performed modulo the irreducible polynomial $x^{127}+x+1$. The first such generator of normal basis found in $GF(2^{127})$ is $\alpha = x^{119}+1$. In $GF(2^{521})$, with irreducible polynomial $x^{521}+x^{32}+1$, the first generator of the form x^a+1 is found to be $\alpha = x+1$.

The M-matrix and the T-matrix required for the implementation of the multiplier function are also determined, using programs M-MATRIX. F77 and T-MATRIX.F77. The listing of these two programs are given in Appendices 21 and 22. The number of Exclusive-or gates needed to implement the multiplier function using the T-matrix approach is calculated using the program EXORNO.F77 (Appendix 23). Without employing any optimization techniques, the T-matrix required 3794 Exclusive-or gates. Thus a rough estimate of the total number of

- 305 -

Exclusive—or and And gates required to implement the multiplier and hence the basic exponentiation function in $GF(2^{127})$ is about $2 \times 3794 + 2$ $(125) + 2 = 7840$ Exclusive—or gates and $2(126) + 1 = 253$ And gates. This can be conveniently manufactured using very large scale integration (VLSI) techniques.

13.9   Extension of Diffie—Hellman System to Matrix Rings

This chapter on public key distribution system is concluded by presenting an extension of the Diffie—Hellman system to matrix rings.

As the ring of all nxn matrices over a finite field contains nilpotent elements when $n>1$ (Section 10.4), again the group formed by only the non—singular matrices of order n, M_n, is considered. In particular, the group of non—singular matrices over Z/pZ where p is a prime is considered. To form a public key distribution system, it is required to choose an element $A \in M_n(Z/pZ)$ where p is a very large prime such that

$$A^r \equiv I \pmod{p}$$

where r is the order of A, the base matrix.
The base matrix A, the prime p and the order r are to be made public. Each user chooses a secret random number x_i less than r and generates a public matrix C_i where

$$C_i \equiv A^{x_i} \pmod{p}$$

Two users can arrive at the common key in the same way as in the Diffie—Hellman system. For instance, if user 1 wishes to initiate an interchange of secret information with user 2, he extracts the public matrix C_2 of user 2 and computes $C_2^{x_1} \pmod{p}$. Similarly user 2 computes $C_1^{x_2} \pmod{p}$ and the process yields the common key K where

$$K = K_{12} = K_{21} \equiv C_1^{x_2} \pmod{p} \equiv C_2^{x_1} \pmod{p} \equiv A^{x_1 x_2} \pmod{p}$$

With the Diffie—Hellman system operated in Z, the maximum number of secret keys possible is limited to $p-1$ whereas with this extended system it depends on the order of the base matrix, r. The larger the value of r, greater the number of users that the system can support. Again the security of this system is dependent on the difficulty of computing logarithms modulo p.

13.9.1 Design of Base Matrix

The system designer needs to construct a base matrix A in M_n

— 306 —

(Z/pZ) and determine its order r. One method of construction of A with a given order is outlined below.

Consider an irreducible polynomial $f(x)$ of degree m for which λ is a root where $\lambda \in F_q$ and $q = p^m$.

$$f(x) = a_0 + a_1 x + \ldots + a_{m-1} x^{m-1} \qquad a_i \in F_p$$

Regarding F_q as a m-dimensional vector space over F_p with basis $(1, \lambda, \lambda^2 \ldots, \lambda^{m-1})$, let T represent the following linear transformation on F_q.

$$T : x \to \lambda x.$$

Under T, then $1 \longrightarrow \lambda, \lambda \longrightarrow \lambda^2, \ldots, \lambda^m \longrightarrow -a_{m-1} \lambda^{m-1} - \ldots -a_0$.

Hence the matrix representation of the linear transformation T relative to the basis $(1, \lambda, \lambda^2, \ldots, \lambda^{m-1})$ is given by the companion matrix

$$B = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ -a_{m-1} & -a_{m-2} & -a_{m-3} & \ldots & -a_0 \end{bmatrix}$$

Linear independence of $I, T, T^2, \ldots, T^{m-1}$ implies that $I, B, B^2, \ldots,$ B^{m-1} are linearly independent. Since $f(x) = 0$, we have $f(B) = 0$. But $f(x)$ has degree m and so the linear independence implies that $f(x)$ is the minimum function of B.

Hence the order of the matrix B is equal to $p^m - 1$ and

$$B^{p^m - 1} \equiv I \pmod{p}$$

Thus the system designer can choose irreducible polynomials of degrees $m_1, m_2, \ldots, m_s$ in Z/pZ (see Section 10.5.2.1) and form the composite matrix B as shown below:

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_s \end{bmatrix}$$

where the order of B_i is equal to $p^{m_i} - 1$ for $1 \leq i \leq s$. The order of the matrix B is then given by the expression

$$\ell cm \left\{ (p^{m_1} - 1), (p^{m_2} - 1), \ldots, (p^{m_s} - 1) \right\}$$

The matrix A to be used in the public key distribution can then be obtained by conjugating B with an arbitrary non-singular matrix Y belonging to $M_n(Z/pZ)$. That is,

- 307 -

$$A = Y B Y^{-1}$$

The order of A is the same as that of B and they are of dimension n where n is given by:

$$n = \sum_{i=1}^{s} m_i$$

This above method has been implemented on the Prime computer system and the program used is given in Appendix 25. A small example of such a public key distribution system is considered below:

13.9.2 Example

Let $p = 5$

Let $f_1(x) = x^2 + x + 1$ where $f_1(x)$ is irreducible over $Z/5Z$.
The matrix B_1 is therefore given by

$$B_1 = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \equiv \begin{bmatrix} 0 & 1 \\ 4 & 4 \end{bmatrix} \pmod 5$$

and

$$\begin{bmatrix} 0 & 1 \\ 4 & 4 \end{bmatrix}^{5^2-1} \equiv I \pmod 5$$

Let $f_2(x) = x^3 + 3x^2 + x + 2$ which is irreducible over $Z/5Z$.
Hence the matrix B_2 is given by

$$B_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -1 & -3 \end{bmatrix} \equiv \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 3 & 4 & 2 \end{bmatrix} \pmod 5$$

and

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 3 & 4 & 2 \end{bmatrix}^{5^3-1} \equiv I \pmod 5$$

Now it is necessary to choose Y and Y^{-1} such that

$$Y \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} Y^{-1} = A \text{ where } A \in M_n(Z/5Z)$$

Let Y be an arbitrary 5x5 non-singular matrix given below:

$$Y = \begin{bmatrix} 2 & 3 & 1 & 0 & 1 \\ 1 & 2 & 2 & 3 & 1 \\ 1 & 0 & 3 & 2 & 4 \\ 1 & 2 & 1 & 4 & 3 \\ 2 & 4 & 0 & 1 & 1 \end{bmatrix}$$

- 308 -

The determinant of Y is calculated using program DETMOD.F77 (Appendix 12) and is equal to 4 (mod 5), hence Y is non-singular. The inverse of Y is calculated using program INVMOD.F77 (Appendix 24) and is given by

$$Y^{-1} = \begin{bmatrix} 0 & 2 & 1 & 3 & 0 \\ 0 & 1 & 2 & 2 & 0 \\ 3 & 3 & 1 & 4 & 3 \\ 2 & 2 & 4 & 2 & 4 \\ 3 & 0 & 1 & 4 & 2 \end{bmatrix}$$

and hence

$$A = \begin{bmatrix} 0 & 2 & 0 & 4 & 3 \\ 1 & 1 & 3 & 1 & 3 \\ 4 & 0 & 0 & 3 & 2 \\ 3 & 3 & 2 & 4 & 4 \\ 1 & 2 & 4 & 1 & 1 \end{bmatrix}$$

The order of A is equal to ℓcm $\{(5^2-1)(5^3-1)\}$ = 744 and $A^{744} \equiv I$ (mod 5), verified using program MATEXP.FTN (Appendix 13). Hence the key space is $2 \leqslant x \leqslant 743$ where $A^x \equiv C$ (mod 5) compared to the key space $2 \leqslant x \leqslant 4$ in the Diffie-Hellman system with p = 5.

13.9.3 Use of Upper Triangular Matrices Over Z/pZ

On the other hand, the ring of upper triangular matrices over Z/pZ (p prime) can also be used. If the base matrix A is chosen from this ring, then the maximum order of such a matrix is equal to p(p-1) which can be obtained by having non-zero elements along the main super-diagonal. This can be shown as follows.

Partitioning A into a diagonal matrix D and an upper triangular nilpotent matrix U, that is, A = U+D, then it is seen that D.U, U.D and U^2 are also upper triangular nilpotent matrices. In Section 10.4.4, it is shown that

$(D + U)^{\emptyset(p)} \equiv (I + U_\emptyset)$ (mod p) where $\emptyset$ is the Euler totient function and $U_\emptyset$ is some upper triangular nilpotent matrix.

and $(I + U_\emptyset)^{p^t} \equiv I$ for some t

If p is assumed to be greater than n-1 (which is valid in a PKD system as the prime p is very large), then t = 1. Thus the order of A is $p\emptyset(p) = p(p-1)$. Hence in this case the key space for x is $2 \leqslant x \leqslant p(p-1)$.

- 309 -

PERMUTATION POLYNOMIALS IN THE DESIGN OF PUBLIC KEY SYSTEMS

14.1   General

The fact that permutation polynomials can be used in the
construction of cryptographic systems of a general mathematical nature
should not be surprising since they determine the permutation of
elements of the set, which is the essential basis of a cipher system.
The RSA permutation polynomials and some others which may be used in
the design of public key systems are investigated.

A polynomial $f(x)$ with coefficients in a finite field F_q is
called a permutation polynomial if the numbers $f(a)$ where $a \in F_q$ are a
permutation of the a's. An equivalent statement is that the equation

$$f(x) = a$$

is solvable in F_q for every 'a' in F_q and that the equation has a unique
solution in F_q for each $a \in F_q$ [80].

14.2   Polynomial $y \equiv x^n \pmod{m}$

First consider the 'famous' power polynomial $y = x^n$ which has
been used in the RSA public key cryptosystem and the Diffie–Hellman
public key distribution system.

Lemma 1

The polynomial

$y \equiv x^n$ in Z/pZ where p is a prime

represents a permutation if and only if n is prime to p-1.

Let d be the greatest common divisor of n and p-1. Then letting
$y \equiv x^n \pmod{p}$ and raising y to the power (p-1)/d, gives,

$$(x^{n/d})^{p-1} \equiv 1 \pmod{p}$$

ie $$y^{(p-1)/d} \equiv 1 \pmod{p}$$

The above congruence has (p-1)/d roots in Z/pZ and each root is a nth
power in Z/pZ. If d = 1, that is, n is relatively prime to p-1, then
there exists only one nth root of each element in Z/pZ and hence $y \equiv x^n$
is a permutation modulo p.

Now the above argument can be extended to the case where
$y = x^n$ is a permutation in Z/mZ where $m = \prod_{i=1}^{r} p_i$ and the p_i, $1 \leq i \leq r$,
are distinct primes and n is a positive integer such that gcd $(n, (p_1-1)$

- 310 -

$(p_2-1)\ldots(p_r-1))= 1.$

To prove that $y \equiv x^n$ (mod m) is a permutation one needs to show that $x^n \equiv y^n$ (mod m) implies that $x \equiv y$ (mod m). Following [81], the proof is considered in three steps.

(i) x and y are relatively prime to m.

(ii) $x = ap_i$ and $y = bp_i$ for $1 \leqslant a, b \leqslant m/p_i$

(iii) $x = ap_i$ and $y = bp_j$ for $i \neq j$, $1 \leqslant a \leqslant m/p_i$, $1 \leqslant b \leqslant m/p_j$

(i)   The proof is by induction on r. From Lemma 1, if r=1, then $y \equiv x^n$ is a permutation as gcd $(n,p_1-1) = 1$. Assume it holds for $r \leqslant k-1$. Let $\{a_1, \ldots, a_{\emptyset(m)}\}$ be a complete reduced residue system mod m, that is, gcd $(a_i,m) = 1$. If $a_i^n \equiv a_j^n$ (mod m), then $(a_i/a_j)^n \equiv x^n \equiv 1$ (mod m) and hence $x^n \equiv 1$ (mod p_i), $1 \leqslant i \leqslant r$. The gcd $(n,p_i-1) = 1$ and Lemma 1 imply that $x \equiv 1$ (mod p_i). But then $x \equiv 1$ (mod m). Thus $\{a_1^n, \ldots, a_{\emptyset(m)}^n\}$ are distinct mod m.

(ii)   Now suppose $a^n p_i^n \equiv b^n p_i^n$ (mod m) where $1 \leqslant a, b \leqslant m/p_i$. Then $a^n p_i^{n-1} \equiv b^n p_i^{n-1}$ (mod m/p_i) and so $a^n \equiv b^n$ (mod m/p_i) since p_i is invertible mod m/p_i. By induction hypothesis, a = b. Consequently, $\{a^n p_i^n\}$, $1 \leqslant a \leqslant m/p_i$ are distinct mod m. Moreover for any i, the sets $\{a^n p_i^n\}$, $1 \leqslant a \leqslant m/p_i$ and $\{a_1^n, \ldots a_{\emptyset(m)}^n\}$ are distinct since $a_j^n \not\equiv 0$ (mod p_i), $1 \leqslant j \leqslant \emptyset(m)$.

(iii)   If $a^n p_i^n \equiv b^n p_j^n$ (mod m) where $1 \leqslant a \leqslant m/p_i$, $1 \leqslant b \leqslant m/p_j$, $i \neq j$, then $p_i | b$, $p_j | a$. Let $a = p_j \bar{a}$ and $b = p_i \bar{b}$, then

$$\bar{a}^n (p_i p_j)^{n-1} \equiv \bar{b}^n (p_i p_j)^{n-1} \pmod{\frac{m}{p_i p_j}}$$

and

$p_i p_j$ invertible mod $\frac{m}{p_i p_j}$ implies that $\bar{a}^n \equiv \bar{b}^n \pmod{\frac{m}{p_i p_j}}$

By the induction hypothesis and by $1 \leqslant \bar{a}, \bar{b} \leqslant m/p_i p_j$, $\bar{a} = \bar{b}$ and so $ap_i = bp_j$. The only intersection of the sets $\{ap_k^n\}$ $1 \leqslant a \leqslant m/p_k$, k = i,j have then, is at the common multiples of $p_i p_j$.

Suppose now that $x^n \equiv y^n$ (mod m) with $x \not\equiv y$ (mod m). From the first case, neither x nor y can be relatively prime to m. But then $x = ap_i$, $y = bp_j$, for some i,j with $1 \leqslant i, j \leqslant r$. Again by the above, this implies $x \equiv y$ (mod m). Thus $\{x^n\}$ $1 \leqslant x \leqslant m$ are distinct (mod m). This completes the induction.

Conversely, if the proposition is not true, then there is at least one i such that gcd $(n,p_i-1) \neq 1$ and hence x^n does not yield a permutation (mod p_i). If $x \not\equiv y$ (mod p_i) and $x^n \equiv y^n$ (mod p_i), then

- 311 -

for $z = m/p_i$ $xz \not\equiv yz$ (mod m) and $(xz)^n \equiv (yz)^n$ (mod m). So x^n is not a permutation (mod m).

The above paragraphs thus show why in the RSA system which uses the polynomials x^e and x^d (mod m), one needs to choose the coding exponents e and d relatively prime to $(p_1-1)(p_2-1)$ where, $m = p_1 p_2$.

14.3   Polynomial $y \equiv ax + b$ (mod m)

Consider the linear permutation polynomial f(x) given by

$$f(x) \equiv ax + b \text{ (mod m)} \tag{14.1}$$

where a and b are elements in Z/mZ and gcd (a,m) = 1 and m is a square free integer. Assume that the message is x, $1 \leqslant x \leqslant m$ and the encryption procedure consists of evaluating f(x). The decryption procedure is given by the inverse polynomial, $f^{-1}(x)$. Rewriting (14.1) as

$$y \equiv ax + b \text{ (mod m)}$$

ie, $$ax \equiv y - b \text{ (mod m)} \tag{14.2}$$

Letting $y - b = y'$, gives

$$ax \equiv y' \text{ (mod m)} \tag{14.3}$$

The congruence (14.3) has as solution

$$x \equiv y' \, a^{\phi(m)-1} \text{ (mod m)}$$

where $\phi(m)$ is the Euler totient function. But the opponent does not actually need to calculate $\phi(m)$ to find x given y. The congruence (14.3) can be easily solved using Euclid's algorithm without the knowledge of $\phi(m)$ as follows.

One can find a^{-1} using Euclid's algorithm where

$$a \, a^{-1} \equiv 1 \text{ (mod m)}$$

Hence $x \equiv y' \, a^{-1}$ (mod m)

Thus as expected, the linear congruence (14.1), does not provide a secure public key system since the opponent can easily recover the message x without factoring the modulus m to its prime factors. On the other hand, this polynomial can be used to form a conventional crypto-system where the parameters a, b and m are kept secret. Further the parameters a and b can be varied in some prearranged manner resulting in a variable substitution such as

$$a_i = g_1(a_{i-1}, \ldots, a_1)$$

and

$$b_i = g_2(b_{i-1}, \ldots, b_1)$$

- 312 -

14.4   Linear Fractional Substitution

Now consider the linear fractional substitution function

$$f(x) \equiv \frac{ax+b}{cx+d} \quad (\text{mod } p)$$

From Section 10.4.5, the above function permutes the elements in Z/pZ if $ad - bc \not\equiv 0$ (mod p). In Section 10.4.5, the elements a,b,c and d were used to form the message. Here the element x is considered to be the message. For designing a public key system which is to be based on the difficulty of factoring a large integer, consider

$$y \equiv \frac{ax+b}{cx+d} \quad (\text{mod } m)$$

where say $m = p_1 p_2$ and p_1, p_2 are distinct primes. y is the cipher. A symbol ∞ is adjoined to Z/mZ where $\infty = 1/0$, $0 = 1/\infty$ and for $x \in Z/mZ$, $\infty + x = \infty$, $\infty x \equiv \infty$ for $x \not\equiv 0$. The legitimate receiver can decrypt the cipher by finding the inverse operations modulo p_1 and modulo p_2 separately and then use the Chinese Remainder Theorem to obtain the message $x(\text{mod } p_1 p_2)$. Actually this system has no security at all because the cryptanalyst can also find the message very easily with the knowledge of the parameters a,b,c,d and m. That is, he does not need to factorize m into its prime factors. This is because the inverse of the matrix $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ (mod m) can be found even without the knowledge of the prime factors of m. This can be done using a process similar to the Gauss-Jordan elimination process over the real numbers except in this case, the elementary operations are chosen to ensure that the determinant of the resulting matrix is relatively prime to m (assuming determinant of $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is relatively prime to m). This type of algorithm would require a similar number of operations as its counterpart over the reals, that is, something like $O(n^3)$ operations for a n×n matrix [45]. Thus the opponent can find the A,B,C,D such that

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \equiv \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \quad (\text{mod } m)$$

If the greatest common divisor of cipher y (mod m) and m is greater than 1, then this gives one of the prime factors of m. The probability of this occurring must be small for large m, as the factorization of a large m is known to be hard. Therefore neglecting this case and considering gcd(y,m) = 1, the opponent can choose two elements w and z mod m such that

$$zy \equiv w \quad (\text{mod } m)$$

Once such a pair (w,z) is found, he can easily solve for the message x as

- 313 -

$$\begin{pmatrix} x \\ 1 \end{pmatrix} \equiv \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} w \\ z \end{pmatrix}$$

and

$$x \equiv Aw + Bz \quad (\text{mod } m) \tag{14.4}$$

Thus the message x can be recovered without having to factorize m. An example illustrating the above method is given below:

Let $m = p_1 p_2 = 3.7 = 21$.

Let

$$y \equiv \frac{3x + 5}{x + 2} \quad (\text{mod } 21) \quad \det \begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix} \equiv 1 \ (\text{mod } 21)$$

Let the message to be encrypted is, $x = 6$. Then the cipher y is given by

$$y \equiv \frac{3.6 + 5}{6 + 2} \equiv 16 \ (\text{mod } 21)$$

Using the modified Gauss-Jordan method, the opponent calculates the inverse to be

$$\begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 2 & 16 \\ 20 & 3 \end{pmatrix} \quad (\text{mod } 21)$$

Choosing $w \equiv 2 \ (\text{mod } 21)$, gives

$$16z \equiv 2 \ (\text{mod } 21)$$

Using Euclid's algorithm, $z \equiv 8 \ (\text{mod } 21)$. Now using equation (14.4), x is calculated to be

$$x \equiv 2.2 + 16.8 \ (\text{mod } 21)$$
$$\equiv 6 \ (\text{mod } 21)$$

Again such a system can be used in the design of a conventional symmetric cryptosystem. One can vary the parameters a_i, b_i, c_i, d_i such that $a_i d_i - b_i c_i \not\equiv 0 \ (\text{mod } m)$, by some prearranged manner. For instance, one can initially select two matrices M_1 and M_2 such that $\det (M_1) \not\equiv 0$ (mod m) and $\det(M_2) \not\equiv 0$ (mod m) and $M_{i+2} = M_{i+1} M_i$, where $M_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$, for subsequent i's.

14.5   <u>Rédei Rational Functions</u>

Now consider a general rational function $f(x) = \frac{g(x)}{h(x)}$, a quotient of polynomials over Z where g(x) and h(x) are relatively prime in Z[x]. Then f(x) is a permutation function modulo m if h(a) (mod m) is a prime residue class (mod m) for any $a \in Z$ and the mapping $\theta: Z/mZ \to Z/mZ$, $\theta (a) \equiv h(a)^{-1} g(a)$ is a permutation [80]. If $m = p_1 p_2$, then f(x) is a permutation function (mod m) if and only if it is a permutation function (mod p_1) and (mod p_2). Again a symbol ∞ is adjoined to Z/mZ

- 314 -

where $\infty = 1/0$ and $0 = 1/\infty$ and for $a \in Z/mZ, \infty + a = \infty$, $a\infty = \infty$ for a $\neq 0$. If the quantities $f(a)$ are distinct for all $a \in Z/mZ \cup \{\infty\}$, then $f(x)$ is a permutation function over $Z/mZ \cup \{\infty\}$.

Rédei [82] considered certain particular functions $f_n(x)$ which could be used in the design of public key cryptosystems as shown below:

Let α be a residue modulo p (p, a prime $\neq$ 2) such that $(\frac{\alpha}{p}) = -1$ (ie, α is a fixed non-square). Rédei then proved that the function $f_n(x)$ given by

$$\left(\frac{x + \sqrt{\alpha}}{x - \sqrt{\alpha}}\right)^n = \frac{f_n(x) + \sqrt{\alpha}}{f_n(x) - \sqrt{\alpha}}$$

ie ,

$$f_n(x) = \sqrt{\alpha} \ \frac{(x+\sqrt{\alpha})^n + (x-\sqrt{\alpha})^n}{(x+\sqrt{\alpha})^n - (x-\sqrt{\alpha})^n}$$

is a permutation (mod p) if n is odd and gcd $(n,p+1) = 1$ and $p \nmid n$. Further he showed that

$$f_{ed}(x) = f_e(f_d(x)) \qquad \text{over } Z/pZ$$

as

$$\frac{f_{ed}(x) + \sqrt{\alpha}}{f_{ed}(x) - \sqrt{\alpha}} = \frac{f_e(f_d(x)) + \sqrt{\alpha}}{f_e(f_d(x)) - \sqrt{\alpha}}$$

Thus f_d is the inverse permutation of f_e if d is chosen such that

$$ed \equiv 1 \ (\text{mod } p+1)$$

and

$$f_e(f_d(x)) = f_d(f_e(x)) = x$$

The proof can be found in [82].

One can use such a function f in the design of public key systems as follows.

If the ring Z/mZ is considered where say $m = p_1 p_2$ (p_1, p_2 are primes $\neq$ 2) then the encrypting exponent e can be chosen such that gcd$(e, p_1 +1) = 1$, gcd$(e, p_2 +1) = 1$ and $e \nmid p_1$ and $e \nmid p_2$, then $f_e(x)$ is a permutation function over Z/mZ. This is a consequence of the Chinese Remainder Therorem. Now one can determine the decoding exponent d such that

$$ed \equiv 1 \ (\text{mod } (p_1+1)(p_2+1)) \qquad\qquad (14.5)$$

The encryption procedure transforms the message $x \in Z/mZ$ using the function $y = f_e(x)$ and the decryption procedure recovers the message by evaluating $f_d(y)$ in Z/mZ. The parameter α in the function f is chosen such that $(\frac{\alpha}{p_1}) = -1$ and $(\frac{\alpha}{p_2}) = -1$. The public key is therefore equal to (e, f_e, m, α) and the secret key is (d, f_d, m, α).

– 315 –

From (14.5), it is seen that to calculate the decrypting exponent d, the opponent needs to factorize m into its prime factors. Thus this function seems to be suitable for a public key cryptosystem whose security again depends on the difficulty of factorizing a large integer m.

14.6   Dickson Polynomial Based Public Key System

Let us now consider a special class of polynomials called the Dickson polynomials [53]. As personally suggested by Prof S D Cohen of Glasgow University, such Dickson polynomials are used in the design of public key cryptosystems.

For $\alpha \in F_p$ and any odd positive integer k, the Dickson polynomial $g_k(x,\alpha)$ is given by

$$g_k(x,\alpha) = \sum_{r=0}^{(k-1)/2} \frac{k}{k-r} \binom{k-r}{r} (-\alpha)^r x^{k-2r} = \left(\frac{x+\sqrt{x^2-4\alpha}}{2}\right)^k + \left(\frac{x-\sqrt{x^2-4\alpha}}{2}\right)^k$$

For $\alpha = 0$, $g_k(x,0)$ is equal to the familiar power function x^k used by the RSA system.

From [53], it is known that if $gcd(k,p^2-1) = 1$, then $g_k(x,\alpha)$ is a permutation polynomial in F_p. Further

$$g_{ed}(x,\alpha) = g_e(g_d(x,\alpha),\alpha^d) = g_d(g_e(x,\alpha),\alpha^e)$$

Thus the inverse permutation of $g_e(x,\alpha)$, with $gcd(e,p^2-1) = 1$ can be found using

$$ed \equiv 1 \ (mod(p^2-1))$$

and

$$g_e^{-1}(x,\alpha) = g_d(x,\alpha^e) \quad \text{over } Z/pZ$$

Now consider the polynomial $g_e(x,\alpha)$ over Z/mZ where say $m=p_1p_2$, a product of distinct primes. Then $g_e(x,\alpha)$ represents a permutation if and only if

$$gcd(e,(p_1^2-1)(p_2^2-1)) = 1$$

Thus one can design a public key system using the Dickson polynomial $g_k(x,\alpha)$ as follows:

1.   Choose large random primes p_1 and p_2 and let $m = p_1p_2$.

2.   Choose the encryption exponent e such that $gcd(e,(p_1^2-1)(p_2^2-1)) = 1$.

3.   Calculate the decrypting exponent d such that
$$ed \equiv 1 \ (mod \ (p_1^2-1)(p_2^2-1)) \tag{14.6}$$

– 316 –

4.   Choose an arbitrary element α in Z/mZ.

The encryption procedure then consists of transforming the message $x \in Z/mZ$ using $g_e(x, \alpha)$ and the decryption procedure recovers the message using $g_d(x, \alpha^e)$. The public key is therefore given by $(e, m, g_e(x, \alpha), \alpha)$ and the secret key consists of $(d, m, g_d(x, \alpha^e), \alpha)$. The security of this system again lies in the difficulty of factoring the modulus m to its prime factors to be able to calculate d using (14.6).

Thus Dickson polynomials can be used to design secure public key systems like the RSA system. It has been simulated on the Prime Computer system using the program DPOLY.F77 given in Appendix 26. An example showing the various parameters is given below.

Example

Let $p_1 = 5$, $p_2 = 7$ and $m = p_1 p_2 = 35$. Let $\alpha = 1$

Choosing the coding exponents e and d such that

$$ed \equiv 1 \pmod{(5^2-1)(7^2-1)}$$

$$ed \equiv 1 \pmod{1152}$$

Therefore e = 11 and d = 419 are suitable.

Thus the encryption Dickson polynomial function is given by $g_{11}(x,1)$:

polynomial degree = 11

polynomial coefficients =

x^{11} x^0

(1   0   24   0   9   0   28   0   20   0   24   0   )

The decryption Dickson polynomial function is given by $g_{419}(x,1)$:

polynomial degree = 419

polynomial coefficients =

x^{419}

(1	0	1	0	2	0	5	0	14	0
28	0	13	0	26	0	5	0	31	0
25	0	0	0	0	0	15	0	0	0
0	0	0	0	0	0	0	0	14	0
21	0	21	0	7	0	0	0	0	0
0	0	0	0	20	0	15	0	15	0
30	0	5	0	0	0	0	0	30	0
20	0	20	0	5	0	30	0	0	0
0	0	0	0	0	0	0	0	14	0
21	0	21	0	7	0	0	0	0	0
21	0	21	0	7	0	0	0	14	0
28	0	28	0	21	0	0	0	21	0
0	0	0	0	0	0	0	0	0	0

```
O    O    O    O    O    O    O    O    7    O
28   O   28    O   21    O    O    O    O    O
14   O   34    O    8    O   15    O   16    O
12   O    7    O   14    O   30    O   34    O
20   O    5    O   30    O    O    O    O    O
 O   O    O    O    O    O    O    O    O    O
 O   O    O    O    O    O   15    O   15    O
30   O    5    O    O    O    O    O   20    O
 5   O    5    O   10    O   25    O    O    O
 O   O   15    O    O    O    O    O    O    O
 O   O    O    O    O    O    O    O    O    O
 O   O    O    O    O    O    O    O    O    O
18   O    3    O   31    O   20    O   22    O
14   O   14    O   13    O   25    O   18    O
15   O   20    O    O    O    O    O    O    O
 O   O    O    O    O    O    O    O    7    O
28   O   28    O   11    O   25    O   15    O
20   O    O    O    O    O   10    O   20    O
20   O    5    O   30    O    O    O    O    O
25   O    O    O    O    O    O    O    O    O
 O   O    O    O    O    O    O    O   21    O
14   O   14    O   28    O    O    O   25    O
24   O   24    O   13    O   15    O   21    O
 7   O   27    O    4    O   25    O   29    O
20   O    O    O    O    O    O    O    O    O
 O   O    O    O    O    O    O    O   28    O
 7   O   22    O   29    O   30    O    5    O
21   O   21    O   27    O    5    O   19    O
 3   O   18    O   21    O    O    O    1    O)
                                           O
                                           x
```

Message x ≡ 10 (mod 35)

Encryption : cipher y = $g_{11}(x,1) \equiv 25$ (mod 35)

Decryption : $g_{419}(y,1) \equiv 10$ (mod 35) = Message

14.7   Discussion

Some permutation polynomials for which the inverse permutations
are easy to construct have been considered. The linear polynomial
ax+b and the linear fractional substitution function (ax+b)/(cx+d) are

– 318 –

found to be insecure when used as public key systems. The Rèdei function and the Dickson polynomials seem to offer a similar level of security as the prototype RSA system. In general, it is not easy to find the inverse permutation for an arbitrary permutation polynomial. One way of finding the inverse permutation of a polynomial $g(x)$ in Z/pZ consists of raising the function $g(x)$ successively to powers $2,3,4,\ldots$ and combining an appropriate set of them to give the inverse function. Using this method, both the system designer and the cryptanalyst have the same work factor. On the other hand, if a permutation polynomial $g(x)$ can be found whose inverses over Z/p_iZ for $i=1,\ldots,$ r are easy to calculate but whose inverse over Z/mZ where $m = \prod_{i=1}^{r} p_i$ is difficult to calculate without knowing the prime factors p_i, then one can design a public key system based on the factorization trapdoor as follows:

1.   Choose large primes $p_1,\ldots,$ p_r and let $m = \prod_{i=1}^{r} p_i$.

2.   The permutation polynomial $g(x)$ is made public along with m.

3.   The encryption procedure consists of transforming a message $x_0 \in Z/mZ \to g(x_0) \in Z/mZ$.

4.   The decryption procedure finds $g^{-1}(x_0)$ in Z/p_iZ for $i=1,\ldots r$ and then uses the Chinese Remainder Theorem to recover the message x_0 in Z/mZ.

Note that the decryption procedure should require the knowledge of prime factors of m to be able to provide security.

Public key systems can also be designed using permutation polynomials based on the law of composition. One can combine the permutation polynomials, for which inverses can be found if some 'extra' information is known, under the law of composition to construct public key systems which can be more secure than the individual polynomial based systems. Consider for instance, the encryption procedure given by the composite permutation function g where

$$g = y_1 \ O \ y_2 \ O \ y_3 \ O \ \ldots \ O \ y_n$$

and

y_i, $1 \leq i \leq n$, are permutation functions.

(O — denotes composition)

The decryption procedure is then obtained by the composition of the inverses of the composition factors of g in the opposite order given by

$$h = y_n^{-1} \ O \ y_{n-1}^{-1} \ O \ \ldots \ O \ y_1^{-1}$$

Hence g O h = h O g = Identity, I.

The system designer can easily obtain the inverse composite permutation function h as he knows the factors of g. But the opponent only knows g

— 319 —

and its inverse is very difficult to calculate since it is very difficult to find the decomposition of g with respect to composition.

As an illustration of this scheme, consider the following simple example given below:

Let $y_1 \equiv x^{e_1} \pmod{m}$ where $m = p_1 p_2$ and $e_1 d_1 \equiv 1 \pmod{\phi(m)}$

Let $y_2 \equiv a_1 y_1 + b_1 \pmod{m}$ where $\gcd(a_1, m) = 1$ and $b_1 \in Z/mZ$

Let $y_3 \equiv y_2^{e_2} \pmod{m}$ where $e_2 d_2 \equiv 1 \pmod{\phi(m)}$

Let $y_4 \equiv a_2 y_3 + b_2 \pmod{m}$ where $\gcd(a_2, m) = 1$ and $b_2 \in Z/mZ$

Then the composite permutation g is given by

$$g(x) = a_2(a_1 x^{e_1} + b_1)^{e_2} + b_2$$

Letting $e_1 = 3$ and $e_2 = 5$, gives

$$g(x) = \sum_{i=0}^{15} c_i x^i \pmod{m} \text{ where } c_i \in Z/mZ.$$

The system designer would make the function $g(x)$ and m public and keep the prime factors p_1 and p_2 and the factors y_1, y_2, y_3 and y_4 of the composition secret. If someone only knows $g(x)$ and m, then it is very difficult to find the inverse permutation. As seen earlier, one way is to try g, g^2, ... until the inverse permutation can be constructed by trial and error procedure. On the other hand, the legal receiver can find the inverse permutation very easily by calculating y_4^{-1}, followed by y_3^{-1}, followed by y_2^{-1}, followed by y_1^{-1} to recover the message.

The complexity of the composite system can be dramatically increased using such an approach and there are numerous ways of constructing such composite systems. For instance, one could combine the RSA system and the knapsack system. The message can be encrypted using the RSA system or one of its extensions and the encrypted message can be interpreted as a string of numbers in say the binary system, which can be encrypted using knapsack system. This gives rise to a number of possibilities to increase the security.

- 320 -

C H A P T E R   15

CHAINING TECHNIQUES AND BROADCASTING WITH PUBLIC KEY SYSTEMS

15.1   General

In the following section, some of the chaining techniques
are applied to the generalized RSA matrix system discussed in Section
10.4. It may be recalled that some of these techniques have been used
with the symmetric DES system (see Chapter 5). Such techniques can
also be applied to the RSA polynomial system described in Section 10.5.
By these techniques, the undesirable effects of redundancy and structure
present in the plaintext data are. eliminated.

Finally, a precaution which should be taken when using the
RSA system or any one of its extensions in a broadcasting situation is
mentioned, where a single message is encrypted under several public keys
to be sent to several users.

15.2   Chaining Techniques

Let E and D denote the encryption and decryption of a message
M under the generalized RSA system. Initially, consider the inter-
symbol dependence in an individual block using the RSA matrix system.

First consider the messages to be non-singular matrices over
Z/mZ, where $m = \prod_{i=1}^{r} p_i^{a_i}$, under the normal encryption mode $C \equiv M^e$
(mod m). If there is a single error in cipher matrix C then the
recovered plain matrix will be completely in error. An example
illustrating this is shown in Figure 15.1. This indicates that there
is a strong intersymbol dependence within an individual ciphertext block.

On the other hand, with upper triangular matrix messages
(including the diagonal elements), this is not the case. Consider a
message M, a 3x3 matrix over Z/mZ, given by

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \quad \text{where } gcd(a_{ii}, m) = 1, \ 1 \leqslant i \leqslant 3$$

and the corresponding cipher matrix C

$$C \equiv M^e \pmod{m}$$

- 321 -

Let $m = 5.3^2 = 45$

Let $e = d = 5$

$$M \equiv \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \equiv \begin{pmatrix} 8 & 13 \\ 3 & 4 \end{pmatrix} \pmod{45}$$

$$C \equiv M^5 \equiv \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} \equiv \begin{pmatrix} 29 & 7 \\ 12 & 13 \end{pmatrix} \pmod{45}$$

An error in c_1 affects decryption of all elements m_1, m_2, m_3 and m_4.

Denoting this as

$(c \rightarrow c_1^*) \longrightarrow (m_1^*, m_2^*, m_3^*, m_4^*)$

$(29 \rightarrow 30) \longrightarrow (39, 40, 30, 19)$

Similarly,

$(c_2 \rightarrow c_2^*) \longrightarrow (m_1^*, m_2^*, m_3^*, m_4^*)$

$(7 \rightarrow 8) \longrightarrow (11, 5, 30, 1)$

$(c_3 \rightarrow c_3^*) \longrightarrow (m_1^*, m_2^*, m_3^*, m_4^*)$

$(12 \rightarrow 13) \longrightarrow (11, 10, 25, 1)$

$(c_4 \rightarrow c_4^*) \longrightarrow (m_1^*, m_2^*, m_3^*, m_4^*)$

$(13 \rightarrow 14) \longrightarrow (29, 2, 42, 44)$

Fig 15.1 – Intersymbol Dependence in Non-singular

Matrix Message Space.

- 322 -

Page 322

$$C = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ 0 & c_{22} & c_{23} \\ 0 & 0 & c_{33} \end{pmatrix}$$

If there is an error in c_{11}, denoted by c_{11}^*, then after decryption it is found that

$$\begin{pmatrix} c_{11}^* & c_{12} & c_{13} \\ 0 & c_{22} & c_{23} \\ 0 & 0 & c_{33} \end{pmatrix}^d \xrightarrow{\ D\ } \begin{pmatrix} a_{11}^* & a_{12}^* & a_{13}^* \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} = M^*$$

It is seen that only the elements a_{11}, a_{12} and a_{13} are affected due to an error in c_{11}. That is, there is not a strong intersymbol dependence between the elements as in the case of the non-singular matrix messages. More generally, it is possible to work out which elements are being affected by errors. Considering again 3x3 matrices, let the message be

$$M = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}$$

Then

$$M^{n+1} = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}^{n+1}$$

$$= \begin{pmatrix} a_n & b_n & c_n \\ 0 & d_n & e_n \\ 0 & 0 & f_n \end{pmatrix} \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}$$

Thus

$$a_{n+1} = a^{n+1} \qquad (15.1)$$
$$d_{n+1} = d^{n+1} \qquad (15.2)$$
$$f_{n+1} = f^{n+1} \qquad (15.3)$$

and

$$b_{n+1} = a\, b_n + b\, d^n$$
$$e_{n+1} = d\, e_n + e\, f^n$$
$$c_{n+1} = a\, c_n + b\, e_n + e\, f^n$$

$- 323 -$

Hence

$$b_n = \frac{b}{a-d} \, (a^n - d^n)$$

$$e_n = \frac{e}{d-f} \, (d^n - f^n)$$

$$c_n = \frac{be}{d-f} \, \frac{(a^n - d^n)}{(a-d)} + \left(c - \frac{be}{d-f}\right) \left(\frac{a^n - f^n}{a-f}\right)$$

Now if there is an error in 'a', that is, a^*, this results in error in a_n, b_n and c_n. Denoting this as,

$$a^* \longrightarrow a_n^*, \ b_n^*, \ c_n^*$$

then

$$b^* \longrightarrow b_n^*, \ c_n^*$$

$$c^* \longrightarrow c_n^*$$

$$d^* \longrightarrow d_n^*, \ b_n^*, \ e_n^*, \ c_n^*$$

$$e^* \longrightarrow e_n^*, \ c_n^*$$

$$f^* \longrightarrow f_n^*, \ e_n^*, \ c_n^*$$

Thus from the point of view of intersymbol dependence within a block, it is preferrable to use the set of non-singular matrices as messages.

Now consider some chaining techniques using this RSA matrix system. Suppose an user A wishes to communicate to an user B a total of r messages M_1, ..., M_r. Let the corresponding ciphers be C_1, ..., C_r.

One way of chaining would be to modify the first cipher C_1 using some function f_1 before transmitting to user B. The function f_1 can be communicated to user B under some secure means. For the subsequent ciphers C_2, ..., C_r, both users can derive the functions f_2, ..., f_r using some publicly known function g,

$$f_i = g(f_{i-1}, \ ..., \ f_1, \ M_{i-1}, ..., \ M_1, \ C_{i-1}, \ ..., \ C_1), \ i > 1$$

One could also transmit the function f_1 to begin with using a public key distribution system such as the extended matrix version of the Diffie-Hellman public key distribution system discussed in Chapter 13.

Now consider the case where the function f_1 is transmitted to the user B via the first message M_1. Two such possible methods are looked at.

15.2.1   Method 1

The operation of this method follows the pattern given below:

- 324 -

$$M_1 \xrightarrow{E} C_1 \qquad\qquad C_1 \xrightarrow{D} M_1$$

$$M_2+C_1 \xrightarrow{E} C_2 \qquad\qquad C_2 \xrightarrow{D} M_2+C_1 \xrightarrow{-C_1} M_2$$

$$M_3+C_2 \xrightarrow{E} C_3 \qquad\qquad C_3 \xrightarrow{D} M_3+C_2 \xrightarrow{-C_2} M_3$$
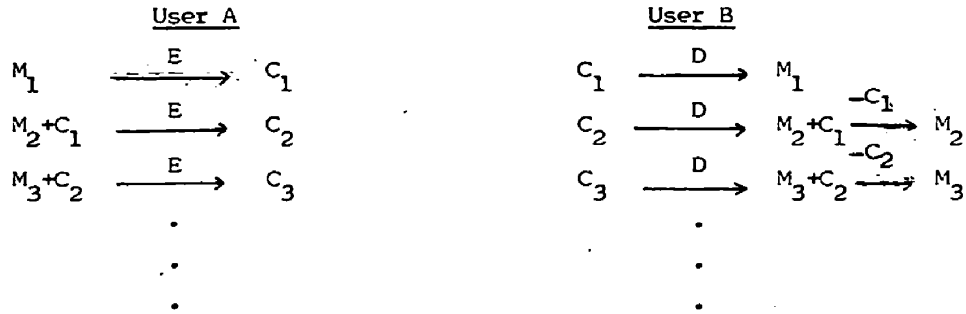
In this method, the previous cipher matrix is simply added with the next message matrix before encryption. Here a simple modulo m addition has been suggested. More generally, one could use a publicly known function g to form $g(M_i, C_{i-1})$. With this method, an error in C_i will cause an error in the decryption of M_i and M_{i+1} and then the system will synchronize. That is,

$$M_i + C_{i-1} \xrightarrow{E} C_i \qquad C_i^* \xrightarrow{D} M_i^* + C_{i-1}^* \xrightarrow{-C_{i-1}} M_i^{**}$$

$$M_{i+1} + C_i \xrightarrow{E} C_{i+1} \qquad C_{i+1} \xrightarrow{D} M_{i+1} + C_i \xrightarrow{-C_i^*} M_{i+1}^*$$
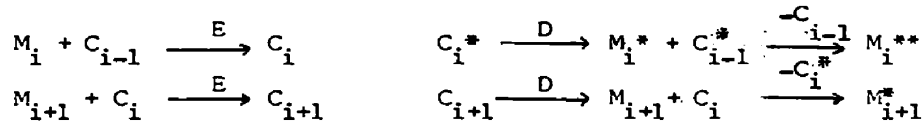
This system has the same error characteristic as the stream cipher feedback (CFB) or the cipher block chaining (CBC) modes of DES. Note that this method does not provide authentication as anyone can encrypt a message.

15.2.2   Method_2_

The operation of this method follows the pattern given below:

$$M_1 \xrightarrow{E} C_1 \qquad\qquad C_1 \xrightarrow{D} M_1$$

$$M_1+C_1+M_2 \xrightarrow{E} C_2 \qquad\qquad C_2 \xrightarrow{D} M_1+C_1+M_2 \xrightarrow{-M_1-C_1} M_2$$
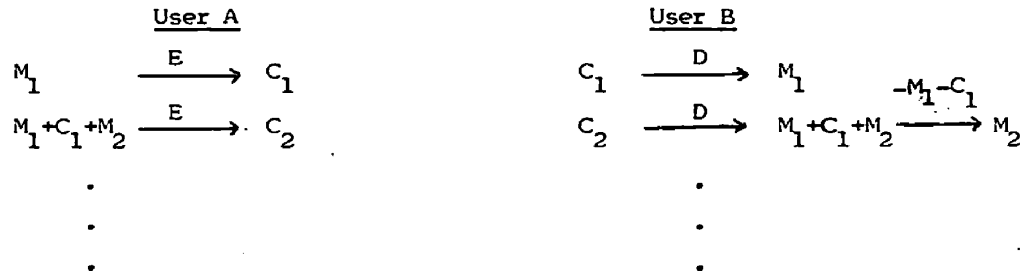
In this method, the previous cipher matrix as well as the previous message matrix are added modulo m to the next message matrix prior to encryption. More generally, again a publicly known function g can be used instead of modulo m addition where the arguments of g are M_i, M_{i-1} and C_{i-1}. With this method, an error in C_i, will cause error in the decryption of all subsequent message matrices.

– 325 –

User A   User B

$M_{i-1} + C_{i-1} + M_i \xrightarrow{\ E\ } C_i$ $\qquad$ $C_i^* \xrightarrow{\ D\ } M_i^* + M_{i-1}^* + C_{i-1}^* \xrightarrow{\ -C_{i-1}\ \ -M_{i-1}\ } M_i^{**}$

$M_i + C_i + M_{i+1} \xrightarrow{\ E\ } C_{i+1}$ $\qquad$ $C_{i+1} \xrightarrow{\ D\ } M_i^* + C_i^* + M_{i+1} \xrightarrow{\ -C_i^*\ \ -M_i^{**}\ } M_{i+1}^*$

. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$.

. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$.

. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$.

This system has the same error characteristic as the CBCP or the CFBV modes of DES discussed in Chapter 5. It possesses the error propagation property.

Note that in both these methods, the first message—cipher pair (M_1, C_1) is essentially used to set up some form of 'initialization vector' (Section 2.3.2) and it is a direct block encryption.

15.3 $\qquad$ Broadcasting of Messages

In many applications, it is necessary to transmit the same set of messages to a group of users in the network. Simmons has shown that [83] in such broadcasting situations, special precautions should be taken to avoid the message being recovered by a cryptanalyst without having to 'break' the underlying cryptosystem. His argument given below applies to the prototype RSA system (over Z) or any of its extensions proposed earlier. Thus the messages M and ciphers C in the argument can be rational integers, polynomials, matrices or algebraic integers.

Let the modulus of the system be m ($=p_1 p_2$) and two public encryption exponents be e_1 and e_2. Let a message M be encrypted with each of these exponents to form C_1 and C_2. That is,

$$C_1 \equiv M^{e_1} \pmod{m}$$
$$C_2 \equiv M^{e_2} \pmod{m}$$

where $M, C_1, C_2 \in Z/mZ$

Now if it is assumed that $\gcd(e_1, e_2) = 1$, then there exists a and b such that

$$ae_1 + be_2 = 1$$

The values of a and b can be found using the Euclid's algorithm. One of the coefficients is positive and the other is negative. If a is

- 326 -

negative, then using Euclid's algorithm, the multiplicative inverse of C_1 (mod m) is given by

$$C_1 C_1^{-1} \equiv 1 (\text{mod } m)$$

If $\gcd(C_1, m) = 1$, then C_1^{-1} exists and

$$(C_1^{-1})^{|a|} (C_2)^b = (M^{e_1})^{-|a|} (M^{e_2})^b$$
$$\equiv M^{ae_1 + be_2} (\text{mod } m)$$
$$\equiv M (\text{mod } m)$$

Thus the message can be recovered even without factoring m into p_1 and p_2. The only information needed are the integer values of m, e_1, e_2, C_1 and C_2 all of which are assumed to be publicly available. Thus precautions must be taken to avoid such situations.

An example illustrating such a situation is given below:

Let m = 13.23 = 299

Let e_1 = 5 d_1 = 53

Let e_2 = 7 d_2 = 151

Let the message M = 4.   Then,

$$C_1 \equiv (4)^5 \equiv 127 (\text{mod } 299)$$
$$C_2 \equiv (4)^7 \equiv 238 (\text{mod } 299)$$

Now using Euclid's algorithm

$$5a + 7b = 1$$
$$\Rightarrow a = 3, b = -2$$

As b is negative, we form

$$C_2 C_2^{-1} \equiv 1 (\text{mod } 299)$$
$$(238) C_2^{-1} \equiv 1 (\text{mod } 299)$$

Using Euclid's algorithm $C_2^{-1} \equiv 49 (\text{mod } 299)$

Hence

$$(49)^2 (127)^3 = 4918167583$$
$$\equiv 4 (\text{mod } 299)$$
$$\equiv M$$

- 327 -

Page 327

C H A P T E R 16

CONCLUSIONS

Some cryptographic techniques for secure data communication over an insecure channel have been investigated in this thesis. The first part has been primarily concerned with conventional crypto-systems, in particular, the Data Encryption Standard (DES) whereas the second part focussed on public key cryptosystems.. The main results and conclusions are summarized below [87-90].

Part 1

A software implementation of the DES algorithm has been carried out using an Apple microcomputer which allowed a study of some of the properties of the DES such as the complementary property and the avalanche effect. It has been found however that such a software implementation is too slow for many real time applications and since it is necessary to store the secret key within the computer system, there is a possibility of its recovery by an unauthorized user. These problems can be overcome by a hardware LSI implementation of the DES and this has been used in the design and construction of a micro-processor based data encryption interface unit.

The use of the interface in a point to point communication system allowed secure data transfer between Apple microcomputers in either plain or encrypted format. The interface has been satisfactorily tested over the public switched telephone network with data rates up to 1200 bits per second. If required, the security can be increased by performing multiple encryption with independent keys or using chaining techniques. Several chaining techniques have been investigated using the developed system namely Cipher Block Chaining (CBC), Stream Cipher Feedback (CFB), Cipher Block Chaining with Plaintext Feedback (CBCP) and stream Cipher Feedback with Vector Feedback (CFBV). Each of these schemes gave rise to a cryptographic system with different error characteristics, speed of operation and level of security and hence is suitable for different applications. The error propagation property of CBCP and CFBV schemes made them particularly suitable for message authentication purposes but unsuitable for use in

communication links prone to noise. The self synchronizing property
of CFB and CBC made them more useful for links prone to small amounts
of noise. The four chaining modes are found to be less susceptible
to attacks of replay, insertion, deletion and code book analysis in
comparison to the standard Electronic Code Book (ECB) mode. Chaining
also helped to eliminate the undesirable effects of data redundancy
and structure. A statistical analysis of the randomness of the output
sequences produced under these different chaining modes confirmed the
good pseudo-random generator property of the DES.

The use of the developed encryption interface has been
extended to provide off-line file security using the Apple disk system.
A self synchronizing mode is found to be more suitable for file
encryption as in this case, recovery from an error must be effected
with ciphertext alone. If a ciphering procedure with error prop-
agation property is used for file security, subsequent inability to read
a portion of the ciphertext because of damage to the physical medium
or the recorded bits, may prevent all the following ciphertext being
deciphered. With communication security, it is possible to recover
from an error by retransmitting the original message.

Further the developed system can be used on the Prestel
public network allowing storage and retrieval of completely and partly
encrypted frames of information on the Prestel database. A 6-bit
cipher feedback technique has been found to be suitable for such an
application. This technique prevented the occurrence of control
characters in the ciphertext which are not acceptable to the Prestel
control unit.

The use of such a DES based encryption system in a
communication network requires the keys be distributed to the users
over a separate secure channel. Several methods of key distribution
using Key Centres and public key systems have been discussed.

Part 2

A generalization of the RSA system in the ring of matrices
over Z/mZ, where m is a composite integer, is proposed. It is shown
that a factorization of the modulus m is needed to compute the
exponent of the group formed by either non-singular matrix messages or

— 329 —

upper triangular matrices with invertible diagonal elements over Z/mZ, thus offering a similar level of security as the prototype RSA system. This system allows the use of a non-square free modulus which is not possible with the RSA system over the integers. This scheme is as suitable for both privacy and authentication as its predecessor. The use of chaining techniques in this generalized system to overcome the difficulties of data redundancy has also been demonstrated.

An extension of the RSA system to polynomial rings has been considered. It is found that the difficulty of factorization of a polynomial into its irreducible factors over a finite field does not in itself provide a secure public key cryptosystem. However, if the difficulty of factorizing an integer is compounded with the difficulty of factorizing a polynomial, then a secure RSA type system in the ring of polynomials is seen to be possible. For cryptographic application, both the modulus polynomial and the modulus integer need to be square free to enable proper decryption.

The design of public key systems in some quadratic algebraic number fields using the factorization trapdoor concept has been presented. The security of such systems is found to be dependent on the difficulty of factorizing the norm of the modulus. Thus a similar level of security as the prototype RSA system can be achieved if the norm is made to be sufficiently large. One method of message representation in such systems involves the use of the elementary divisor theory to choose a standard set of representatives.

The investigation of such extensions indicate that rings other than the ring of rational integers, can be used to construct public key systems with the factorization trapdoor property. From a practical point of view, it seems that the complexity of such systems may favour the implementation of the factorization trapdoor in the ring of rational integers.

The Diffie-Hellman public key distribution has been implemented in the Galois extension field $GF(2^n)$ with $n = 127$, where the computations required for exponentiation can be easily performed using digital logic. To withstand the most recently published Adleman's subexponential algorithm to compute logarithms, it is necessary to work in a higher extension field of $GF(2^{521})$ to maintain a work factor equivalent to that of DES key exhaustion.

- 330 -

Short cycling attacks consisting of repeated encipherings have been carried out against the exponentiation system in $GF(2^n)$ with $n = 3$ and 7. It is found that if the primitive element is chosen at random, then the expected cycle length of an arbitrary cycle is very close to half the number of non-zero elements in the field. Thus this type of attack appears to be very much analogous to a random search procedure.

The exponentiation system in $GF(2^{127})$ has been used in conjunction with the DES encryption to form a hybrid system which combines the protection provided by the conventional cryptosystem with the user authentication attributes of a public key system. Such a hybrid arrangement is found to be feasible in practice.

A dedicated hardware exponentiation system in $GF(2^7)$ has been designed and constructed. The use of normal basis representation in the design allows a modular construction which is very useful in large scale integrated circuit design.

An extension of the Diffie-Hellman public key distribution system to matrix rings is proposed. Using rings of non-singular matrices over Z/pZ (p prime) and upper triangular matrices with invertible elements along the diagonal over Z/pZ, it is shown that the number of possible secret keys is much greater for a given prime p compared to the original system.

The role of permutation polynomials in the design of public key systems has been investigated and it is shown that the class of Dickson permutation polynomials and certain Rèdei rational functions can be used to construct public systems with a similar level of security as the prototype RSA system. Further, a method of designing public key systems using permutation polynomials under the law of composition has been presented. The complexity of such composite systems can be dramatically increased to provide high security.

- 331 -

REFERENCES

1.   Kahn, D., The Codebreakers, MacMillan, New York 1972.

2.   Diffie, W. and Hellman, M. E., 'Privacy and Authentication:
      An introduction to cryptography', Proc. IEEE, Vol.67,
      No.3, 1979.

3.   Meyer, C. and Matyas, S., Cryptography: A new dimension
      in Computer Data Security, John Wiley, 1982

4.   Denning, D. E. R., Cryptography and Data Security,
      Addison-Wesley, 1982.

5.   Simmons, G. J., 'Symmetric and Asymmetric Encryption',
      Computing Surveys, Vol.11, No.4, 1979.

6.   Konheim, A. G., Cryptography: A Primer, John Wiley, 1981.

7.   Shannon, C., 'Communication Theory of Secrecy Systems',
      Bell System Tech.Journal, Vol.28, 1949, pp 656-715.

8.   Garey, M. R. and Johnson, D. S., Computers and Intractability,
      A Guide to the Theory of NP-Completeness, W. H. Freeman
      and Co., 1979.

9.   Brassard, G., 'A note on the complexity of cryptography',
      IEEE Trans. on Information Theory, Vol.IT-25, No.2, 1979.

10.   Shamir, A., 'A polynomial Time Algorithm for breaking the
      basic Merkle-Hellman Cryptosystem', Proc. 23rd Annual
      Symp. Found. Computers Sci, 1982, pp 145-152.

11.   Data Encryption Standard (DES), FIPS publication 46,
      NBS, US Dept. of Commerce, 1977.

12.   Rivest, R. L., Shamir, A. and Adleman, L., 'A method for
      obtaining Digital Signatures and Public Key Cryptosystems',
      Commun. ACM, Vol.21, No.2, 1978, pp 120-126.

- 332 -

13. Merkle, R. C. and Hellman, M. E., 'Hiding Information and Signatures in Trapdoor Knapsacks', Vol.IT-24, No.5, 1978.

14. Beker, H. and Piper, F. C., Cipher Systems: The Protection of Communications, Northwood Pub., 1982.

15. Rivest, R., 'The Impact of Technology on Cryptography', Proc. IEEE Int. Comm. Conf., Toronto, 1978.

16. Vernam, G., 'Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications', Journal AIEE, Vol.45, 1926, pp 109-115.

17. Smith, J. L., 'The Design of Lucifer, a Cryptographic Device for Data Communications', IBM, Watson Research Centre, Yorktown Heights, New York, 1971.

18. Meyer, C. H., 'Ciphertext/Plaintext and Ciphertext/Key Dependence vs. number of rounds for the DES', AFIPS Conference Proc., 47, 1978, pp 1119-1126.

19. Hellman, M. E., Merkle, R., Schroeppel, R., Washington, L., Diffie, W., Pohlig, S. and Schweitzer, P., 'Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard', Elec. Engg. Dept., Stanford University, CA, SEL 76-042, 1976.

20. Branstad, D. K., Gait, J. and Katzke, S., 'Report of the Workshop on Cryptography in support of Computer Security', NBS, USA, IR 77-291, 1976.

21. NBS, 'Report of the 1976 Workshop on Estimation of Significant Advances in Computer Technology', NBS, USA, 1976.

22. Diffie, W. and Hellman, M. E., 'A Critique of the proposed Data Encryption Standard', Comm. ACM, Vol.19, No.3, 1976, pp 164-165.

- 333 -

Page 333

23.   Hellman, M. E., 'A cryptanalytic Time-Memory Trade off', IEEE Trans. on Information Theory, Vol.IT-26, No.4, 1980, pp 401-406.

24.   LEXAR Corporation, 'An Evaluation of the NBS Data Encryption Standard', Los Angeles, USA, 1976.

25.   Bichl, I., Biermeier, J., Gollmann, D. and Pichler, F., 'Cryptanalysis of the Data Encryption Standard by Formal Coding', Proc. IEEE International Symposium on Information Theory, USA, 1981.

26.   DES modes of operation FIPS Pub. 81, NBS, USA, 1980.

27.   Buckley, D. C., 'Hershey Characters', Plymouth Polytechnic Computer Centre Internal Handout No.U5. 8-14, 1983.

28.   Chatfield, C., Statistics for Technology, Chapman and Hall, 1978.

29.   Good, I. J., 'On the Serial Test for random sequences', Ann. Math. Statist., 1957, pp 262-264.

30.   Gibbons, J. D., Non-Parametric Statistical Inference, McGraw Hill, New York, 1971.

31.   Prestel Viewdata System Manuals, British Telecom, 1979.

32.   Price, W. L. and Davies, D. W., 'Issues in the design of a Key Distribution Centre', NPL Report, DNACS 43/81, 1981.

33.   Denning, D. E. and Sacco, G. M., 'Timestamps in Key Distribution Protocols', Comm. ACM, Vol.24, No.8, 1981, pp 533-536.

34.   Denning, D. E. and Schneider, F. B., 'Master keys for group sharing', Information Processing letters, Vol.12, No.1, 1981, pp 23-25.

35.   Diffie, W. and Hellman, M. E., 'New Directions in cryptography', IEEE Transactions on Information Theory, Vol. IT-22, 1976, pp 644-654.

36.   Keyes, R. W., 'Physical limits in digital electronics', Proc. IEEE, Vol.63, 1975, pp 740-767.

37.   Landauer, R. W., 'Irreversibility and heat generation in the computing process', IBM Journal of Research and Development, Vol.5, 1961, pp 183-191.

38.   Hardy, G. H. and Wright, E. M., An Introduction to the Theory of Numbers, Fifth edition, Oxford, 1983.

39.   Ayoub, F. N., 'Some aspects of the design of secure encryption algorithms', Council for National Academic Awards, Hatfield Polytechnic, Thesis, March 1983.

40.   Miller, G. L., 'Riemann's Hypothesis and Tests for Primality', Proc. of the Seventh Annual ACM Symposium on the Theory of Computing, pp 234-239.

41.   Ingemarsson, I., Tang, D. T. and Wong, C. K., 'A Conference Key Distribution System', IEEE Trans. on Information Theory, Vol.IT-28, No.5, 1982, pp 714-720.

42.   Rabin, M. O., 'Probabilistic algorithms', in Symposium on New Directions and Recent Results in Algorithms and Complexity' Academic Press, Ed. J. F. Traub, 1976.

43.   Bond, D. J. 'Practical Primality Testing', International Conference on Secure Communication Systems, IEE, London, 1984.

44.   Solovay, R. and Strassen, V., 'A Fast Monte-Carlo test for Primality', SIAM Journal of Computing, Vol.6, 1977, pp 84-85.

45.   Knuth, D. E., The Art of Computer Programming, Vol.2: Seminumerical Algorithms, 2nd Edition, Addison-Wesley, 1981.

46.   Smith, D. E. and Palmer, J. T., 'Universal fixed messages and the RSA Cryptosystem', Mathematika, Vol.26, 1979, pp 44-52.

47.   Blakely, G. R. and Borosh, I., 'RSA public key cryptosystems do not always conceal messages', Computers and Maths with Appls., Vol.5, pp 169-178.

48.   Albert, A. A., Fundamental Concepts of Higher Algebra, The University of Chicago Press, 1956.

49.   Burton, D. M., Rings and Ideals Addison-Wesley Pub., 1968.

50.   Van der Waerden, B. L., Modern Algebra, Vol.2, 1949.

51.   MacWilliams, J., 'Orthogonal matrices over finite fields', American Mathematical Monthly, Feb 1969.

52.   Turnbull, H. W. and Aitken, A. C., An Introduction to the Theory of Canonical Matrices, Blackie and Son Pub., 1932.

53.   Dickson, L. E., Linear Groups with an Exposition of The Galois Field Theory, Dover Pub., 1958.

54.   Postnikov, M. M., Foundations of Galois Theory, Translated by A. Swinfen and P. J. Hilton, Pergamon Press, International Series of Monographs on Pure and Applied Mathematics, Vol.29, 1962.

55.   Kravitz, D. W. and Reed, I. S., 'Extension of RSA Cryptostructure: A Galois Approach', IEE Electronic letters, Vol.18, No.6, 1982, pp 255-256.

56.   Berlekamp, E. R., Algebraic Coding Theory, McGraw Hill, New York, 1968.

57.   Berlekamp, E. R., 'Factoring polynomials over large finite fields', Mathematics of Computation, Vol.24, No.111, 1970, pp 713-735.

58.   Rabin, M. O., 'Probabilistic Algorithms in Finite Fields',
        Siam Journal of Computing, Vol.9, No.2, 1980, pp 273-280.

59.   Gait, J., 'Short cycling in the Kravitz-Reed Public Key
        Encryption System', Electronic letters, Vol.18, No.16,
        1982, pp 706-707.

60.   Northcott, D. G., Ideal Theory, Cambridge Tracts in
        Mathematics and Mathematical Physics, Cambridge Univ.
        Press, 1953.

61.   Hancock, H., Foundations of the Theory of Algebraic
        Numbers, Vol.1, Dover 1931.

62.   McCoy, N. H., Theory of Rings, MacMillan Co., New York,
        1964.

63.   Pollard, H., The Theory of Algebraic Numbers, The Carus
        Mathematical Monographs, No.9, Pub. by Mathematical Assoc.
        of America, John Wiley, 1950.

64.   Rosen, M. and Ireland, K., A Classical Introduction to Modern
        Number Theory, Springer-Verlag, 1980.

65.   Cohn, H., A Second Course in Number Theory, John Wiley,
        1962.

66.   Andrews, G. E., Number Theory, W. B. Saunders Co., 1971.

67.   Hartley, B. and Hawkes, T. O., Rings, Modules and Linear
        Algebra, Chapman and Hall, 1980 (Reprint).

68.   Weyl, H., Algebraic Theory of Numbers, Princeton University
        Press, 1940.

69.   Pohlig, S. C. and Hellman, M. E., 'An Improved Algorithm for
        Computing Logarithms over GF(p) and its Cryptographic
        Significance', IEEE Trans. On Information Theory, Vol.
        IT-24, No.1, 1978, pp 106-110.

70. Adleman, L., 'A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography', MIT, Dept. of Maths. and Lab. of Computer Science Report, 1980.

71. Blakely, B. and Blakley, G. R., 'Security of Number Theoretic Public Key Cryptosystems against Random Attack I', Cryptologia, Vol.2, No.4, 1978, pp 305-321.

72. Berkovits, S., Kowalchuk, J., and Schanning, B., 'Implementing Public Key Scheme', IEEE Communications Magazine, Vol.17, No.3, 1979, pp 2-3.

73. Zierler, N., 'Primitive Trinomials Whose Degree is a Mersenne Exponent', Information and Control, Vol.15, 1969, pp 67-69.

74. Herlestam, T., and Johanneson, R., 'On Computing Logarithms Over $GF(2^p)$,' BIT, Vol.21, 1981, pp 326-334.

75. Schanning, B. P., 'Data Encryption with Public Key Distribution, IEEE Conference EASCON, 1979.

76. Simmons, G. and Norris, M. J., 'Preliminary Comments on the MIT Public Key Cryptosystem', Cryptologia, Vol.1, 1977, pp 406-414.

77. Berkovits, S., 'Cycling is just a Random Search', IEEE Communication Magazine Vol.17, Nov. 1979, pp 2-3.

78. Webster's New Collegiate Dictionary, G & C Merriam Co., 1975.

79. Massey, J. L. and Omwia, J. K., 'A new multiplication method for $GF(2^m)$ and its application in Public Key Cryptography', Presented at Eurocrypt '83, Udine, Italy, March 1983.

80. Carlitz, L., 'Some theorems on permutation polynomials', Bulletin Amer. Math. Soc., Vol.68, 1962, pp 120-122.

- 338 -

81.   Cordes, C. M., 'Permutations mod m in the form x^n', American Mathematical Monthly, Jan. 1976.

82.   Rèdei, L., 'Über eindeutig umkehrpare Polynome in endlichen Körpern', Acta. Sci. Math., Vol.11, 1946-48, pp 85-92.

83.   Simmons, G. J., 'A Weak Privacy Protocol Using the RSA Cryptoalgorithm,'Vol.7, No.2, 1983, pp 180-182.

84.   Diffie, W., 'Cryptographic Technology: Fifteen Years Forecast', BNR Inc., Mountain View, California, USA, 1981.

85.   Rèdei, L., Algebra, Volume 1, International Series of Monographs in Pure and Applied Mathematics, Pergamon Press, 1967.

86.   Davenport, H., The Higher Arithmetic, An Introduction to The Theory of Numbers, Hutchinson and Co. Pub. Ltd., 1968.

87.   Sanders, P. W. and Varadharajan, V., 'Secure Communications between Microcomputer Systems', Computer Communications, Vol.6, No.5, Oct. 1983.

88.   Odoni, R. W. K., Varadharajan, V. and Sanders, P. W., 'Public Key Distribution in Matrix Rings', IEE Electronic Letters, Vol.20, No.9, Apr. 1984.

89.   Varadharajan, V. and Odoni, R. W. K., 'Extension of RSA System to Matrix Rings', submitted to Cryptologia for Publication.

90.   Sanders, P. W. and Varadharajan, V. 'DES can keep your data safe', Apple User (Windfall), Vol.2, No.12, June 1983.

A P P E N D I C E S

Appendix 1

Data_Encryption Standard_ : _A_Software_Design

       A description of the Data Encryption Standard program together with a partial listing is given in this Appendix.

       The program is written as a subroutine which can be called at hexadecimal address 1E∅∅. In order to use the routine, three things must be supplied: mode, plaintext and key. The user selects the encryption or decryption mode by inputting E or D. In the case of encryption, the input plaintext can be provided either from the keyboard or from a file stored in memory whereas in the case of decryption, the input ciphertext is expected to be present in the memory. The program is almost the same for both encryption and decryption. In the case of decryption, the left shifts are replaced by the right shifts and the shift schedule operation is carried out in the reverse order to that of encryption. Hence only the encryption program is considered here.

A 1.1 _ Main_Program_ : _Encryption

```
        RND CNT    :    Iteration count
        IE∅∅       :    JSR    (INIT - 1)
                        JSR    (E1)
                        JSR    (E2)
                        LDA @   00
                        STA    RND   CNT
        L3         :    LDA    RND   CNT
                        CMP @ 10
                        BEQ    L1
                        CMP @ 08
                        BEQ    L2
                        CMP @ OF
                        BEQ    L2
                        CMP @ 00
                        BEQ    L2
                        CMP @ 01
                        BEQ    L2
```

@ : Hexadecimal data

A-1

```
        L2          JSR         (LEFT SHIFT)
                    JSR         (LEFT SHIFT)
                    JSR         (PC - 2)
                    JSR         (E4)
                    INC         RND  CNT
                    JSR         (INIT - 2)
                    JMP         L3

        L1          JSR         L5
                    HALT
```

A 1.2 _Subroutines_ : _Encryption_

1.   INIT.1, INIT.2 : The subroutine INIT-1 initializes
      the key register, the data register and all the
      temporary registers to Zero whereas the routine INIT.2
      only clears the temporary registers.

2.   E1 : This subroutine takes 64 bits of key as input
      block and carries out the permutation PC-1. The
      output is stored in memory locations 9000 to 9007
      (Key Register).

3.   E2 : This subroutine takes 64 bits of data as input
      block and carries out the Initial Permutation (IP).
      The result is stored in memory locations 7000 to 7007.
      (Data Register).

4.   LEFT SHIFT : The input to this routine is found in
      locations 9000 to 9007. It performs a single circular
      left shift on locations 9000 to 9003 and on 9004 to
      9007. The valid output bits in locations 9000 to 9007
      are the seven most significant bits, i.e.,

      9000 to 9007 : | V A L I D //|
                              ←——— 7 ———→

5.   PC-2 : This routine performs the permutation PC-2 on
      the memory locations 9000 to 9007. (Only the seven most
      significant bits are used). The result is stored in 4000
      to 4007 where the six middle bits are the valid bits i.e.,

      4000 to 4007 : |// V A L I D //|
                              ←——— 6 ———→

                              A-2

6. E4 : This routine carried out a number of tasks. First,
    the right half of the original data containing 32 bits
    into 48 bits using permutation E. Input to this operation
    is found in locations 7004 to 7007 and the output is stored
    in 3500 to 3507. In each location, the valid bits are the
    six least significant ones, i.e.,

    3500 to 3507 : ░░░V A L I D
                     ←——6——→

    The next part of the routine combines this expanded
    version with the key bits after PC-2 (in 4000 to 4007)
    using exclusive-or operation. The result is stored in
    locations 4500 to 4507. In each location, the valid bits
    are the six least significant ones, i.e.,

    4500 to 4507 : ░░░V A L I D
                     ←——6——→

    Then the routine used the 6-bit blocks in 4500 to 4507 as
    input to the S-boxes and yields a 4-bit blocks as outputs.
    The first and sixth bits in each location are combined to
    represent a number, i, in the range $\emptyset$ to 3 in base 2.
    The middle four bits are used to represent a number, j,
    in the range 0 to 15 in base 2. The output is the entry
    (i, j) in the corresponding S-box. These S-box outputs
    are stored in locations 4600 to 4607. Then, the
    Permutation P is carried out using the values in 4600 to
    4607 as its input. The result is stored in 5700 to 5703
    where all 8 bits in each location are valid. They are
    exclusive-ored with the left half of the original data
    in 7000 to 7003 to form the right half of the next
    iteration. Finally the right half of the original data
    is transferred to the left half.

7. E5 : This routine carried out the Inverse Initial
    Permutation (IP^{-1}) on 7000 to 7007. The output of this
    routine is stored in 6100 to 6107 which is the final
    ciphertext output.

A-3

A 1.3 _ Use of the DES Program

*   1EØØ G

*   ENTER ENCRYPT (E) or DECRYPT (D)

    E

*   ENTER KEY PLEASE

*   ENTER DATA BLOCK PLEASE

*   ENCRYPTED DATA BLOCK IS FOUND IN (6100 - 6107)

    - - - - - -

*   1EØØ G

*   ENTER ENCRYPT (E) or DECRYPT (D)

    D

*   DECRYPTED DATA BLOCK IS FOUND IN (3800 - 3807)

    - - - - - -

A-4

Appendix 2

Point-to-point Communication: Block Encryption Program (ECB)

A 2.1 Basic Flowchart

START

```
┌─────────────────────────┐
│   Initialize temporary  │
│   Registers e.g. CNTRL-G│
│   Counter, CNTRL-J      │
│   Counter, etc.         │
└─────────────────────────┘
             │
             ▼
┌─────────────────────────┐
│   Set Mode Register,    │
│   Command Register of   │
│   I/O Controller        │
└─────────────────────────┘
             │
             ▼
┌─────────────────────────┐
│   Request the user for  │
│   the desired Baud Rate │
│   and set accordingly   │
└─────────────────────────┘
             │
             ▼
     Test DSR signal    ──No──▶  ╱ NO LINK ╱
       from Modem                ╱ ERROR   ╱
             │ Yes
             ▼
   Plain or Encrypted or ──Plain──▶ 1
     Mixture Format
             │ Mix or Enc.
             ▼
┌─────────────────────────┐
│   Activate and Initialize│
│   Data Security Device  │
└─────────────────────────┘
             │
             ▼
     Key Request ON?    ──No──▶  ╱ RESET  ╱
             │                   ╱ Please ╱
             │ Yes
             ▼
             2
```

A-5

```
                              2

              ┌─────────────────────────┐
              │  Display  'Enter Key     │
              │  Please'                 │
              └─────────────────────────┘
                            │
                            ▼
              ┌─────────────────────────┐
      ┌──────▶│  Fetch 1 Key Byte        │
      │       └─────────────────────────┘
      │                     │
      │                     ▼
      │       ┌─────────────────────────┐
      │       │  Parity  Check  Routine  │
      │       └─────────────────────────┘
      │                     │
      │                     ▼
      │       ┌─────────────────────────┐
      │       │  Load Key Byte in        │
      │       │  KEY Register            │
      │       └─────────────────────────┘
      │                     │
      │                     ▼
      │ No     ╱─────────────────────────╲
      └───────<      64   Key   Bits?      >
              ╲─────────────────────────╱
                            │
                            ▼ Yes

              ┌─────────────────────────┐
              │  Display 'Enter Data     │
              │  Please'                 │
              └─────────────────────────┘
                            │
                            ▼
              ╱─────────────────────────╲   Yes
             <   Test for Mixture Format  >─────────▶  1
              ╲─────────────────────────╱

      6 ──────────────▶    │ No
                            ▼
              ╱─────────────────────────╲   Yes
      ┌──────<    Check DCD signal        >─────────▶  3
      │       ╲   from Modem             ╱
      │        ╲───────────────────────╱
      │                     │
      │                     │ No
      │                     ▼
      │ No     ╱─────────────────────────╲
      └───────<   Check  Keyboard  Flag    >
              ╲─────────────────────────╱
                            │
                            ▼ Yes

                            4
```

4

Tx Mode

```
┌─────────────────────────────┐
│   Data   Input   Routine    │
└─────────────────────────────┘
                │
                ▼
```

Only Return Character ⟩ Yes →

```
┌─────────────────────────────┐
│ Fill block with pseudo-     │
│ random characters and       │
│ provide Delay for           │
│ repeated Return             │
└─────────────────────────────┘
```

10 ────────→ No

16

End   of   Message? ⟩ Yes→ 5

End of a Block? ⟩ Yes

No

```
┌─────────────────────────────┐
│    Fetch a Data Byte        │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Test for Control characters │
│ including CNTRL-G, CNTRL-J   │
│ and increment respective    │
│ counters if present         │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Store the output in the     │
│ Data Security Device        │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Read Encrypted Data and     │
│ Transmit it to Modem        │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Provide Delay for CNTRL-G,  │
│ CNTRL-J and Line feed characters │
└─────────────────────────────┘
```

A-7

5

```
/ Exactly 8 bytes = 1 Block? \——No———————————┐
\                            /                │
            │                                 │
           Yes                                ▼
            ▼                    ┌──────────────────────────┐
┌──────────────────────┐        │ Fill the Block with a    │
│ Encrypt and Send it to│       │ Return Character and     │
│ Modem                 │       │ pseudo-random characters │
└──────────────────────┘        └──────────────────────────┘
            │                                 │
            ▼                                 │
┌──────────────────────┐                      │
│ Pad with 7 bytes of  │                      │
│ pseudo-random characters│                   │
│ and a Return character │                    │
└──────────────────────┘                      │
            │                                 │
            ▼                                 │
┌──────────────────────┐                      │
│ Encrypt and Send the Block │◄───────────────┘
└──────────────────────┘
            │
            ▼
┌──────────────────────┐
│   Delay   Routines    │
└──────────────────────┘
            │
            ▼
┌──────────────────────┐
│   Reset   Counters    │
└──────────────────────┘
            │
            ▼
/ Test for Mixture format? \——No——————→  6
\                          /
            │
           Yes
            ▼
            1
```

3

8 ————————→   Rx mode

```
┌──────────────────────────┐
│ Fetch  Data  from  Link  │
└──────────────────────────┘
            │
            ▼
            7
```

A÷8

8

7

```
┌─────────────────────────────────┐
│      Error  Check Routines      │
└─────────────────────────────────┘
                │
No ◄────────────┤ 8 bytes of Data = 1 Block? │
                │
               Yes
                │
┌─────────────────────────────────┐
│      Decrypt  Data  Block       │
└─────────────────────────────────┘
                │
                │
┌─────────────────────────────────┐
│      Display the plain          │
│      Data                       │
└─────────────────────────────────┘
                │
No ◄────────────┤      End  of  message?      │
                │
               Yes
                │
6 ─── No ◄──────┤   Test for Mixture Format?  │
                │
               Yes
                │
                1
                1
                │
                │
┌─────────────────────────────────┐
│   Check DCD signal from         │──── Yes ────►
│   Modem                         │
└─────────────────────────────────┘
                │ No
                │
No ◄────────────┤   Check  Keyboard Flag      │
                │
               Yes
                │
               23
              A-9
```

9

23

```
┌─────────────────────────────┐
│    Data   Input   Routine   │
└─────────────────────────────┘
```

Only Return Character? 〉Yes → Provide Delay for Repeated Return

→ Increment Character Counter

End of Message ? 〉Yes → Reset Counters → 1

No

Test for Mixture Format? 〉Yes → 10

No

```
┌─────────────────────────────┐
│    Fetch   a   data   byte   │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│  Test Routines for Control  │
│  Characters including       │
│  CNRTL-G and CNTRL-J        │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│    Transmit Data to Modem   │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│       Delay  Routines       │
└─────────────────────────────┘
```

9

Test for Mixture Format 〉Yes → 8

No

24

A-9a

24

```
┌─────────────────────────────┐
│   Fetch  a  data  byte      │
└─────────────────────────────┘
              │ No
              ▼
┌─────────────────────────────┐
│   Error  Test  Routines     │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│   Display Received Data      │
└─────────────────────────────┘
              │
              ▼
```

9

⟨ End of Message? ⟩ Yes ──────► 1

              │ No

A-9b

Comments

```
30AE  :   LDA  C051   Set Text Mode
          STA  C0A8   Reset I/O Controller WD 2123
          LDX @ 00
          LDY @ 00
          LDA @ 00
          STA   1B    Counter:  No. of CNTRL-G within a block.
          STA   F9    Counter:  No. of CNTRL-J within a block.
          STA   06    Initialize memory location (06) to be
                      used in determining end of a line of
                      display on the VDU.
          LDA @ 01
          STA   08    Counter:  No. of characters in a line.
          LDA @ 10
          STA   07    Counter used in delay routine to avoid
                      noise spikes created during relay change-
                      over in the modem.
          LDA  C0A6
          AND @ 01    Check DSR = 1 ?
          BEQ   L1
          JMP NOLINK  Display No Link Error Message.

L1        LDA @ 6E    Set MODE Register of I/O Controller
          STA  C0AD   WD 2123.
          LDA @ 12    Set COMMAND Register of I/O Controller
                      WD 2123.
```

A- 10

```
        STA    COAD
        LDA  @  14        Set SELCLK = 1
        STA    COA4
        JSR BAUD          Subroutine to set appropriate Baud Rate.
        JSR PLAENC        Subroutine displaying the message 'Plain
                          or Encrypt'.
        JSR    FD6A       Subroutine to input data into the 6502
                          Apple System from keyboard.
        LDA    0200
        CMP  @  DO
        BNE    L2         Encryption format requested.
        JSR    DATA       Display Message - 'Enter Data Please'.
        JSR    R7
        JMP  PLAIN        Jump to plain data communication routines.
L2      JSR    DSD        Subroutine which handles the secret key.
                          Fetches key from keyboard, checks for
                          correct parity and stores in KEY Register
                          in Security Device WD 2001.
        JSR    DATA
        JSR    R7


L4      LDA    COA6       Check keyboard and the communication link
        AND  @  03        for the presence of valid data in an
        NOP               alternating manner.
        BNE    L3
        JMP    RX
L3      LDA    C000
        BPL    L4
```

A-11

```
        LDA @ 33        Set COMMAND Register of I/O Controller
        STA  COAD       WD 2123 to Transmit Mode.
        LDA @ 06        Set Encryption Mode in Security Device
        STA  COA1       WD 2001.

        JSR  FD6A        Data Input Subroutine.
        CPX @ 00
        BEQ  L5
        JMP  L6
L5      LDA @ FF
        JSR  FCA8
        JSR  END         Subroutine to encrypt and send a block
                         consisting of a Return character and
                         seven random  characters.
        JSR SWITCH       Switch the I/O device to Receive Mode.
        JMP  L4
L6      TXA
        LDY @ 00
        STA  1D
L10     LDX @ 00
L9      LDA @ 00
        CMP  1D          Check end of message.
        BEQ  L7
        CPX  08          Check completion of a block of 64 bits.
        BEQ  L8
        JSR  DIR         Check Data Input Request.
        LDA 0200,y
        JSR TEST         Test for CNTRL-G or CNTRL-J character.
        DEC  1D
        INY
        INX
        JMP  L9
L8      JSR READSEND  Subroutine to send an encrypted block.
        JSR DELAY LINE Subroutine to provide delay for some
                         special characters.
        JMP  L10
```

A-12

```
L7        CPX @ 08
          BEQ   L11
          JMP   L12

L11       JSR   READSEND
          JSR   DELAYLINE
          LDX @ 00

L12       JSR   END
          JSR   DELAYLINE
          JSR   SWITCH
          JMP   L4

RX        LDA @ 02      )
          JSR   FCA8    )
          DEC   07      )
          LDA   07      )   Check DCD repeatedly to avoid
          CMP @ 00      )   noise spikes.
          BEQ   Z1      )
          JMP   L4      )

Z1        LDA @ 10
          STA   07
          LDA   COAA
          LDA @ 16          Set Receive Mode in the I/O Controller.
          STA   COAD
          LDA @ OE          Set Security Device to Decryption Mode.
          STA   COA1

L18       LDA @ 00          Counter: To determine whether 8 bytes
                            have been received.
          STA   FC

L17       LDA   COAB
          JSR   DELAY       Test TxRDY of I/O Controller.
          AND @ 02
          BEQ   L17
          JSR   DIR
          INC   FC
```

A-13

```
            LDA   COAB      )
            AND @ 10        )
            BNE   L23       )
            LDA   COAB      )      Check for parity,
            AND @ 08        )      framing and overrun errors.
            BNE   L23       )
            LDA   COAB      )
            AND   20        )
            BNE   L23       )
            JMP   B1        )
      L23   JSR   ERR       )
            JMP   L4


      B1    LDA   COAA
            STA   COAO             Input the received cipher block
                                   to the Security Device.
            LDA   FC
            CMP @ 08
            BNE   L17
            JSR READPRINT          Decrypt the cipher and display the
                                   message on the VDU screen.
                                   If end of message, discard any random
                                   bytes in the last block.
            CMP @ 8D
            BNE   L18
            LDX ·@ 00
      L21   CPX @ 08
            BEQ   L19
            LDA 9500, x
            CMP @ 87
            BEQ   L20
            CMP @ 8A.
            BEQ   L20
            INx
            JMP   L21
```

A-14

```
L19     JSR     END1 .
        JMP     L4

L20     LDA  @  FF
        JSR     FCA8
        JMP     L19
```

A 2.3 _ _Subroutines

        Some of the important subroutines called by the main
program are now briefly described.

A 2.3.1 _READSEND

        This subroutine first checks the TxRDy and the CTS
status bits to see if they are in the correct state. Then it
checks the DOR status bit and reads data bytes from the Security
Device one by one until a whole block is formed. Then the block
is sent down the line.

```
        LDX @ 00

L25     CPX @ 08
        BEQ  L22

L23     LDA COAB ·
        BPL  L23
        LDA COAB
        JSR DELAY
        AND @ 01
        BEQ  L23
        INX
        LDA COA2
        STA COAC
        JMP  L25

L22     RTS
```

A 2.3.2 _READPRINT

        This subroutine initially checks the DOR status
bit of the Security Device and then reads the data bytes.
The decrypted data block is stored in 9500 to 9507 for later
use and is displayed on the VDU screen.

```
        LDX @ 00

L27     CPX @ 08
        BEQ  L24
```

A-15

```
L26     LDA   COA3
        BPL   L26
        INX
        LDA   COA2
        STA   9500,x
        JSR   FDF0
        JMP   L27

L24     RTS.
```

A 2.3.3 _ END

This subroutine is concerned with the last block of
the message in the Transmission Mode. First it checks whether
the number of characters present in the last block which is
not completely full is equal to seven. If so, it adds the
Return character at the end and encrypts this block and sends
the cipher over the link using READSEND. If however the
block contains less than seven characters, it fills the block
with a Return and random characters, encrypts the block and
transmits the cipher over the link. In both cases, the
subroutine SUB1 is used to increment the appropriate CNTRL-G,
CNTRL-J   character counters. The routine SUB1 is part of
the TEST subroutine described in Section A 2.3.4.

```
        JSR   DIR
        LDA @ 8D
        STA   COA0
        JSR   SUB1
        INX
M12     CPX @ 08
        BEQ   M14
        LDA   02F8,x
        STA   COA0
        JSR   SUB1
        INX
        JMP   M12
M14     JSR   READSEND
        RTS.
```

A-16

A 2.3.4 _ TEST

This subroutine checks whether the input character is a CNTRL-G or a CNTRL-J or any other control character and increments the appropriate counters accordingly.

```
          STA    COAO
          CMP  @  8A
          BEQ    R11
          CMP  @  87
          BEQ    R12
          CMP  @  AO
          BCS    R13
          RTS

R11       INC    F9
          RTS

R12       INC    1B
          RTS

R13       JSR    SUB1
          RTS

SUB1      INC    ·08
          LDA    08
          CMP  @  29
          BEQ    R14
          RTS

R14       LDA  @  FF
          STA    06
          LDA  @  01
          STA    08
          RTS
```

A 2.3.5 _ DELAYLINE

This subroutine provides the necessary delay required after sending a CNTRL-G, CNTRL-J or a linefeed (end of line) character.

```
          LDA    06
          CMP  @  FF
          BEQ    R16
          JMP    R17
```

A-17

```
R16      LDA @ F0
         JSR FCA8
         LDA @ 00
         STA   06

R17      LDA   1B
         CMP @ 00
         BEQ R18
         LDA @ FF
         JSR FCA8
         DEC   1B
         JMP R17

R18      LDA   F9
         CMP @ 00
         BEQ R19
         LDA @ 80
         JSR FCA8
         DEC   F9
         JMP R18

R19      RTS.
```

A 2.3.6 _ BAUD

     This subroutine takes a number between 0 to 8 from the
keyboard as its input and sets the Baud Rate in the I/O
controller accordingly in the range 50 to 1200 bits per
second (bauds).

```
         LDA   0200
         AND @ OF
         STA   COAE
         ASL
         CMP @ 06
         BCS   R20
         CLC
         ADC @ 04
R20      STA   09
         SEC
         LDA @ 11
         SBC   09
         STA   09
         RTS.
```

A-18

A 2.3.7 _ ERR

       This subroutine prints an ERROR MESSAGE and helps
the system to resynchronize after an error has occurred. This
is done by resetting DIR and DOR status flags and initializing
the character counter (08) and the memory location (06) to zero.

```
            JSR   FF2D
R21         LDA   COA3
            AND  @ 40
            BNE    R23

R22         LDA   COA3
            BPL    R22
            LDA   COA2
            JMP    R22

R23         STA   COA0
            JMP    R21

R24         LDA  @ 01
            STA    08
            LDA  @ 00
            STA    06
            LDX  @ 08

R25         CPX  @ 00
            BEQ    R26
            LDA   COA2
            DEX
            JMP    R25

R26         RTS.
```

A 2.3.8 _ PLAIN

       This subroutine allows plain data communication
over the link.

```
L29         LDA   COA6
            AND  @ 03
            BNE    L28
            JMP    RX1
```

A-19

```
L28         LDA   C000
            BPL     L29
            LDA @   33
            STA   C0AD
            JSR   FD6A
            CPX @   00
            BEQ     L30

L32         INX
            JMP     L31

L30         LDA @   FF
            JSR   FCA8
            JMP     L32

L31         TXA
            STA     ID
            LDY @   00

L33         CPX @   00
            BEQ     L34

L35         LDA   C0AB
            BPL     L35
            LDA   C0AB
            JSR   DELAY
            AND @   01
            BEQ     L35
            LDA 0200,Y
            STA   C0AC
            JSR   TEST
            JSR   DELAYLINE
            INY
            DEX
            JMP     L33

L34         JSR   SWITCH
            JMP     L29

RX1         LDA @   02
            JSR   FCA8
            DEC     07
            LDA     07
            CMP @   00
            BEQ     Z3
            JMP     L29
```

A-20

```
Z3          LDA  @  10
            STA     07
            LDA  COAA
            LDA  @  16
            STA  COAD
            LDA  @  00
            STA  6500

L40         LDA  COAB
            AND  @  02
            BEQ  L40
            LDA  COAB
            AND  @  20
            BNE  L23
            LDA  COAB
            AND  @  10
            BNE  L23
            LDA  COAB
            AND  @  08
            BNE  L23
            JSR  FF2D
            JMP  L29

L23         LDA  COAA
            STA  6501
            JSR  FDF0
            LDA  6501
            CMP  @  8D
            BEQ  L41
            LDA  6501
            STA  6500
            JMP  L40

L41         LDA  @  12
            STA  COAD
            LDA  @  FF
            JSR  FCA8
            LDA  6500
            CMP  @  87
            BEQ  L42
            CMP  @  8A
            BEQ  L42
            JMP  L29
```

A- 21

```
L42        LDA @ FO
           JSR  FCA8
           JMP. L29
```

A 2.3.9 _ DSD

        This subroutine initializes the Data Security Device,
WD 2001, and enters the DES secret key into the KEY Register
of the device after checking/correcting its parity. This
routine also generates a 64-bit pseudo-random number.

```
           LDA  0200
           CMP @ C5
           BEQ   S1
           LDA  0200
          .CMP @ C4
           BEQ   S2
           JMP  FF2D

S2         LDA @ 01
           STA   FC
           LDA @ OE
           JMP   S3

S1         LDA @ 00
           STA  .FC
           LDA @ 06

S3         STA  COA1
           LDA  COA3
           AND @ 10
           BNE   S4
           JMP  RESET

S4         JSR ENTER KEY PLEASE        Display 'Enter Key Please'.
           LDA @ 00
           STA   FA
           LDA @ F8
           STA   1D
           LDA @ 02
           STA   1E

S6         JSR  FDOC
           JSR  FDFO
           JSR  PARITYCHECK     Parity check subroutine called.
```

A-22

```
                    LDA     FB
                    STA     C0A0
                    LDA     C0A3
                    AND  @  20
                    BEQ     S5
                    JMP     FF2D

S5                  LDY  @  00
                    LDA     4E
                    STA  (1D),Y
                    INC     1D
                    INC     FA
                    LDA     FA
                    CMP  @  08
                    BNE     S6
                    LDY  @  08

S8                  CPY  @  00
                    BEQ     S7
                    LDA  @  88
                    JSR     FDF0
                    DEY
                    JMP     S8

S7                  JSR     FD8B
                    RTS

PARITYCHECK:        STA     FB
                    LDX  @  00
                    LDY  @  00

S10                 ROL
                    BCC     S9
                    INY

S9                  INX
                    CPX  @  08
                    BNE     S10
                    TYA
                    AND  @  01
                    ROR
                    BCS     S11
                    LDA     FB
                    CLC
                    ROL
                    STA     FB

S11                 RTS
```

A-23

Appendix 3

Extended Character Set

A-24

<u>Appendix 4</u>

<u>Graphics Display Program</u> (<u>RESTRY.F77</u>)

```
C  x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
C  This program prints the plaintext and ciphertext examples
C  using Hershey character set.  The file VVV contains the 96
C  possible ASCII characters (32 out of 128 are control
C  characters and hence not printed).  The program used GINO
C  library subroutines for drawing characters and it also
C  invokes special Hershey subroutines.
C  x x x x x x x x x x x x x x x x x x x x x x x x

      INTEGER TABLE (256), N, TOTAL1 (20)
C  Select the graphics printer as the output device.
          CALL CC906
C  Select a new paper for a new set of axis
          CALL PICCLE
C  Set the window size, horizontal and vertical axes scale.
          CALL UNITS (2.94)
          CALL WINDOW (2)
          CALL CHASIZ (1.0, 1.0).

C  Read the character set from file VVV.
          OPEN (7, FILE = 'VVV')
          READ (7,*)  (TABLE (I), I=1,256)
          CLOSE (7).

C  Input the example number and mode of encryption number.
          K = 0
          READ (1,*) IP1, IP2
          IPP1 = 1200 + IP1
          IPP2 = 1200 + IP2
          N = 0
          Y = 61.2

C  Draw the characters using GINO and HERSHEY subroutines.
          CALL MOVTO2 (1.0, Y)
          CALL HERCEN (IPP1)
          CALL MOVBY2 (1.0, 0.0)
          CALL HERCEN (IPP2)
          Y = Y-1.6
```

A-25

```
              CALL MOVTO2 (1.0,Y)
40            CALL CHOSE (TOTAL1, L,IK)
              1F (IK  .EQ. 1)  Go to 50
              DO 101  JJ = 1, L
              TOTAL1 (JJ) = TOTAL1(JJ) + 1
101           CONTINUE
              II = 1
              L1 = L + 1
30            1F (II.EQ.L1) Go to 10
              1F (N.EQ.40) Go to 20
              NO = TOTAL1 (II)
              NOO = TABLE (NO)
              1F (NOO.EQ.0) NOO = 161
              CALL HERCEN (NOO)
              CALL MOVBY2 (2.1,0.0)
              N = N + 1
              II = II + 1
              Go to 30
.10           Go to 40
20            Y = Y - 1.6
              IF (Y. EQ.0) Go to 50
              CALL MOVTO2 (1.0, Y)
              N = 0
              Go to 30
50            CALL DEVEND
              STOP
              END
c  The subroutine CHOSE transforms a character into its
c  corresponding Hershey character.  Hence this allows a
c  possible representation of all 256 codes using the
c  extended character set (Appendix 3).  It calls another
c  subroutine CONV2.

              SUBROUTINE CHOSE  (TOTAL1, L, IK)
              CHARACTER * 1  A(38)
              INTEGER I, NUM1, NUM2, NUM, TOTAL, TOTAL1 (20),
       *           LENGT, L
              READ (5,1) (A(I), I = 1, 38)
1             FORMAT (38 A1)
```

A-26

Page 368

```
              IK = 0
              I  = 1
300           IF (A(I). EQ. ' ' ) Go To 200
              I = I + 1
              Go to 300
200           LENGT = I
              1F (LENGT.Lt.5) Go To 211
              J = 6
              L = 0
201           L = L + 1
              CALL CONV2 (A(J), NUM1)
              J = J + 1
              CALL CONV2 (A(J), NUM2)
              TOTAL = 16 * NUM1 + NUM2
              TOTAL1 (L) = TOTAL
              J = J + 1
              IF (J.NE. LENGT) Go to 201
              RETURN
211           IK = 1
              RETURN
              END
              SUBROUTINE CONV2 (CHAR, NUM)
              CHARACTER * 1 CHAR
              INTEGER NUM
              IF (CHAR. EQ. '0') NUM = 0
              IF (CHAR. EQ. 'A') NUM = 10
              IF (CHAR. EQ. 'B') NUM = 11
              IF (CHAR. EQ. 'C') NUM = 12
              IF (CHAR. EQ. 'D') NUM = 13
              IF (CHAR. EQ. 'E') NUM = 14
              IF (CHAR. EQ. 'F') NUM = 15
              RETURN
              END
```

A- 27

Page 369

Appendix 5

Point to Point Communication : Cipher Block Chaining Program (CBC)

A 5.1 - Basic Flowchart

START

```
┌─────────────────────────┐
│ Initialize Temporary    │
│ Registers e.g. CNTRL-G  │
│ Counter, CNTRL-J        │
│ Counter, etc.           │
└─────────────────────────┘

┌─────────────────────────┐
│ Set Mode Register,      │
│ Command Register of     │
│ I/O Controller          │
└─────────────────────────┘

┌─────────────────────────┐
│ Request the user for the│
│ desired Baud Rate and   │
│ set accordingly         │
└─────────────────────────┘
```

Test DSR signal
from Modem ──No──> No LINK ERROR

Yes

Plain or Encrypted or
Mixture Format ──Plain──> 1

Mix or Enc.

```
┌─────────────────────────┐
│ Activate and Initialize │
│ Data Security Device    │
└─────────────────────────┘
```

Key Request ON ? ──No──> RESET Please

Yes

2

A-28

```
                                    2

                        ┌─────────────────────┐
                        │ Display 'Enter Key   │
                        │ Please'              │
                        └─────────────────────┘
                                    │
                        ┌─────────────────────┐
              ┌────────▶│ Fetch 1 Key Byte     │
              │         └─────────────────────┘
              │                     │
              │         ┌─────────────────────┐
              │         │ Parity Check Routine │
              │         └─────────────────────┘
              │                     │
              │         ┌─────────────────────┐
              │         │ Load Key Byte in     │
              │         │ KEY Register         │
              │         └─────────────────────┘
              │                     │
              │   No   ╱───────────────────────╲
              └───────⟨      64 Key Bits ?      ⟩
                       ╲───────────────────────╱
                                    │ Yes
                        ┌───────────────────────────────────────┐
                        │ Generate 64 bit                        │
                        │ pseudo random number                   │
                        │ and store in a specified               │
                        │ memory space - Initialization          │
                        │ Vector - and send it to the receiving end │
                        └───────────────────────────────────────┘
                                    │
                        ┌─────────────────────────────┐
                        │ Display 'Enter Data Please'  │
                        └─────────────────────────────┘
                                    │
                       ╱───────────────────────╲   Yes
                      ⟨  Test for Mixture Format ⟩────────▶  1
                       ╲───────────────────────╱
                                    │ No
     6. ────────────────────────────▶
                                    │
                       ╱───────────────────────╲
              ┌───────⟨ Check DCD signal from    ⟩   Yes
              │       ⟨ Modem                     ⟩────────▶  3
              │        ╲───────────────────────╱
              │                     │ No
              │       ╱───────────────────────╲
              │      ⟨  Check keyboard flag      ⟩
              └── No  ╲───────────────────────╱
                                    │ Yes

                                    4
```

A-29

4

↓ Tx mode

```
Data Input routine
```

Only return character? → Yes → Fill block with pseudo-random characters and provide Delay for repeated Return

↓ No

10 →

22 → End of Message ? → Yes → 5

↓ No   16

End of a Block? → Yes

↓ No

```
Fetch a Data Byte
```

```
Test for Control characters
including CNTRL-G, CNTRL-J
and increment respective
counters if present
```

```
Exclusive-or data byte with
Initialization Vector byte
```

```
Store the Output in the
Data Security Device
```

```
Read Encrypted Data and Transmit
it to Modem.  Store the encrypted
data as Intermediate Initialization Vector
```

↓

21

A-30

22   21
| ↓

```
┌─────────────────────────┐
│     Provide Delay for    │
│   CNTRL-G, CNTRL-J and   │
│   Line feed Characters   │
└─────────────────────────┘
              ↓
              5
              ↓
```

⟨ Exactly 8 bytes = 1 Block ? ⟩ → No ──────┐

Yes ↓   ↓

```
┌─────────────────────────┐    ┌─────────────────────────┐
│  Exclusive-or with the   │    │  Fill the Block with     │
│ Intermediate Initialization│   │  a Return character      │
│  Vector                  │    │  and pseudo random       │
└─────────────────────────┘    │  characters              │
              ↓                 └─────────────────────────┘
┌─────────────────────────┐
│  Encrypt and Send it     │
│  to Modem.  Store encrypted data │
│  as intermediate Initialization Vector │
└─────────────────────────┘
              ↓
┌─────────────────────────┐
│  Pad with 7 bytes of pseudo- │
│  random characters and a │
│  Return character        │
└─────────────────────────┘
```

16 ──────────────────→ ↓

```
┌─────────────────────────┐
│  Encrypt and Send and Store │
│  Encrypted data as intermediate │
│  Initialization Vector   │
└─────────────────────────┘
              ↓
┌─────────────────────────┐
│     Delay   Routines     │
└─────────────────────────┘
              ↓
┌─────────────────────────┐
│     Reset   Counters     │
└─────────────────────────┘
              ↓
```

⟨ Test for Mixture Format ⟩ → No   6

Yes ↓

1

A- 31

3

Rx Mode

```
Fetch Initialization Vector from Link
and store it in a specified memory
```

8 ──────────────────────────────→

```
Fetch Data from Link
```

```
Error Check Routines
```

No ←─── 8 Bytes of Data = 1 Block?

Yes

```
Decrypt Data Block
```

```
Exclusive-or with Intermediate
Initialization Vector
```

```
Display the resultant plain
data
```

No ←─── End of message?

Yes

6   No ←─── Test   for Mixture Format?

Yes

1

1

Check DCD signal from   Yes
Modem

No

No ←─── Check Keyboard Flag

Yes   9

23

A-32

23

```
┌─────────────────────────┐
│   Data  Input  Routine  │
└─────────────────────────┘
              │
              ▼
    ╱─────────────────────────╲      ┌──────────────────────┐
   ╱   Only  Return  Character? ╲ Yes │    Provide Delay     │
   ╲                            ╱────→│ for Repeated Return  │
    ╲─────────────────────────╱       └──────────────────────┘
              │                                   │
              │                                   ▼
              │                       ┌──────────────────────┐
              │                       │ Increment Character  │
              │◀──────────────────────│ Counter              │
              │                       └──────────────────────┘
              ▼
    ╱─────────────────────────╲
   ╱     End  of  Message ?     ╲─────────────────┐
   ╲                            ╱                  │
    ╲─────────────────────────╱                   ▼
              │                          ┌──────────────────┐
              ▼                          │ Reset Counters   │
    ╱─────────────────────────╲ Yes      └──────────────────┘
   ╱  Test for Mixture Format? ╲──────┐            │
   ╲                            ╱      │            │
    ╲─────────────────────────╱       ▼            ▼
              │ No                    10           1
              ▼
    ┌─────────────────────────┐
    │   Fetch  a  data  byte  │
    └─────────────────────────┘
              │
              ▼
    ┌─────────────────────────┐
    │ Test Routines for Control│
    │ Characters including     │
    │ CNRTL-G and CNTRL-J      │
    └─────────────────────────┘
              │
              ▼
    ┌─────────────────────────┐
    │  Transmit Data to Modem │
    └─────────────────────────┘
              │
              ▼
    ┌─────────────────────────┐
    │    Delay  Routines      │
    └─────────────────────────┘
              │
              ▼
```

9

```
    ╱─────────────────────────╲
   ╱   Test for Mixture Format  ╲ Yes     8
   ╲                            ╱────→
    ╲─────────────────────────╱
              │
              No
              ▼
```

24

A -33

24

```
┌─────────────────────────────┐
│   Fetch  a  data  byte      │
└─────────────────────────────┘
                │ No
                ▼
┌─────────────────────────────┐
│   Error  Test  Routines     │
└─────────────────────────────┘
                │
                ▼
┌─────────────────────────────┐
│   Display Received Data      │
└─────────────────────────────┘
                │
                ▼
        ╱───────────────────╲
9      ╱   End  of  Message?  ╲      Yes        1
       ╲                      ╱  ──────────►
        ╲───────────────────╱
                │ No
                ▼
```

A- 34

A 5.2 _ Program Listing

A 5.2.1 _Main Program

        Most comments are similar to the ones given in the
Block Encryption Program listing in Appendix 2.

```
30AE        :  LDA  C051
               STA  COA8
               LDX  @ 00
               LDY  @ 00
               LDA  @ 00
               STA    1B
               STA    F9
               STA    06
               LDA  @ 01
               STA    08
               LDA  @ 10
               STA    07
               LDA  COA6
               AND  @ 01
               BEQ    L1
               JMP NOLINK

L1             LDA  @ 6E
               STA  COAD
               LDA  @ 12
               STA  COAD
               LDA  @ 14
               STA  COA4
               JSR  BAUD
               LDA  COAA
               JSR PLAENC
               JSR  FD6A
               LDA  0200
               CMP  @ DO
               BNE    L2
               JSR  DATA
               JSR    R7
               JMP PLAIN
```

A-35

```
L2          JSR   DSD

P3          LDA   COA6
            AND @ 03
            BNE   P1
            JMP   P2
P1          LDA   C000
            BPL   P3

            LDA @ 33
            STA   COAD
            LDA @ 06
            STA   COA1
            JSR   FD6A
            LDY @ 00

P6          CPY @ 08
            BEQ   P4

P5          LDA   COAB
            BPL   P5
            LDA 02F8,Y          Pseudo random numbers are
            STA 3000,Y          stored in 3000 to 3007 which
            STA   COAC          act as the Initialization
            INY                 Vector (IV).  IV is
            JMP   P6            transmitted to the other end.
P4          JMP   P7

L4          LDA   COA6
            AND @ 03
            BNE   L3
            JMP   RX
            LDA   C000
            BPL   L4
            LDA @ 33
            STA   COAD
            LDA @ 06
            STA   COA1
            JSR   FD6A
P7          CPX @ 00
            BEQ   L5
            JMP   L6
```

A- 36

Page 378

```
L5              LDA @ FF
                JSR  FCA8
                JSR   END
                JSR SWITCH
                JMP    L4
L6              TXA
                LDY @ 00
                STA   1D

L10             LDX @ 00

L9              LDA @ 00
                CMP   1D
                BEQ   L7
                CPX @ 08
                BEQ   L8
                JSR   DIR
                LDA 0200,Y
                EOR 3000,X
                JSR  TEST
                DEC   1D
                INY
                INX
                JMP    L9

L8              JSR READSEND MOD
                JSR DELAYLINE
                JMP   L10

L7              CPX @ 08
                BEQ   L11
                JMP   L12

L11             JSR READSEND MOD
                JSR DELAYLINE
                LDX @ 00

L12             JSR ENDMOD
                JSR DELAYLINE
                JSR SWITCH
                JMP    L4
```

A-37

```
RX          LDA @  02
            JSR  FCA8
            DEC     07
            LDA     07
            CMP @  00
            BEQ    Z1
            JMP    L4
Z1          LDA @  10
            STA     07
            LDA  COAA
            LDA @  16
            STA  COAD
            LDA @  OE
            STA  COA1

L18         LDY @  00

L17         LDA  COAB
            JSR  DELAY
            AND @  02
            BEQ   L17
            JSR   DIR
            INY
            LDA  COAB
            AND @  10
            BNE   L23
            LDA  COAB
            AND @  08
            BNE   L23
            LDA  COAB
            AND     20
            BNE   L23
            JMP    B1

L23         JSR   ERR
            JMP    L4

B1          LDA  COAA
            STA  COAO
            STA 3010,Y
            CPY @  08
            BNE   L17
```

A-38

```
                JSR  READPRINTMOD
                CMP  @  8D
                BNE    L18
                LDX  @  00
L21             CPX  @  08
                BEQ    L19
                LDA  9500,X
                CMP  @  87
                BEQ    L20
                CMP  @  8A
                BEQ    L20
                INX
                JMP    L21

L19             JSR   END 1
                JMP    L4

L20             LDA  @  FF
                JSR  FCA8
                JMP    L19
```

A 5.2.2   Subroutines

Most of the subroutines used in the program are the
same as the ones used in the Block Encryption Program
given in Appendix 2.   The subroutine READSENDMOD is a slightly
modified version of READSEND given in Appendix 2 (A 2.3.1).
The modified section in READSENDMOD is given below.

```
                   .
                   .
                   .
                LDA   COA2
                STA   COAC
                STA   3000, X
                JMP    L25
L22             RTS
```

A-39

The subroutine READPRINTMOD is a slightly modified version of READPRINT given in Appendix 2 (A 2.3.2). The modified section in READPRINTMOD is given below.

```
            .
            .
            .
            LDA  COA2
            EOR  3000,X
            STA  9500,X
            JSR  FDF0
            LDA  3010,X
            STA  3000,X
            JMP  L27
L24         RTS
```

The subroutine ENDMOD is the same as the subroutine END of Appendix 2 (A 2.3.3) except that ENDMOD calls the subroutine READSENDMOD instead of READSEND.

A-40

Appendix 6

Point to Point Communication : Stream Cipher Feedback Program (CFB)

A 6.1 Basic Flowchart

```
                          START
                            │
                            ▼
              ┌───────────────────────────────────┐
              │ Initialize Temporary registers    │
              │ e.g. CNTRL-G, CNTRL-J Counters etc.│
              └───────────────────────────────────┘
                            │
                            ▼
              ┌───────────────────────────────────┐
              │ Set Mode register, Command         │
              │ register of I/O Controller         │
              └───────────────────────────────────┘
                            │
                            ▼
              ┌───────────────────────────────────┐
              │ Request the user for the desired   │
              │ Baud Rate and set accordingly      │
              └───────────────────────────────────┘
                            │
                            ▼                                    ┌──────────┐
              ◁───────────────────────────────▷   No            │ NO LINK  │
              │ Test DSR Signal from Modem     │ ─────────────▶  │ ERROR    │
              ◁───────────────────────────────▷                 └──────────┘
                            │ Yes
                            ▼
              ┌───────────────────────────────────┐
              │ Only complete Encryption format    │
              │ program is given                   │
              └───────────────────────────────────┘
                            │
                            ▼
              ┌───────────────────────────────────┐
              │ Activate and Initialize the        │
              │ Data Security Device               │
              └───────────────────────────────────┘
                            │
                            ▼                                    ┌──────────┐
              ◁───────────────────────────────▷   No            │ RESET    │
              │ Key Request ON ?               │ ─────────────▶  │ please   │
              ◁───────────────────────────────▷                 └──────────┘
                            │ Yes
                            ▼
                            2
```

A- 41

2

```
┌─────────────────────────┐
│ Display 'Enter key      │
│ Please'                 │
└─────────────────────────┘
┌─────────────────────────┐
│ Fetch 1 Key Byte        │
└─────────────────────────┘
┌─────────────────────────┐
│ Parity Check Routine    │
└─────────────────────────┘
┌─────────────────────────┐
│ Load Key Byte in KEY    │
│ Register                │
└─────────────────────────┘
```

No   ⟨ 64 Key Bits ? ⟩

```
┌───────────────────────────────┐
│ Generate 64-bit pseudo-       │
│ random number and store       │
│ in a specified memory         │
│ space - Initialization Vector │
│ and send it to the receiving end │
└───────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Display 'Enter Data Please' │
└─────────────────────────────┘
```

5 ⟶

⟨ Check DCD signal from Modem ⟩   Yes ⟶ 3

No

No ⟨ Check Keyboard Flag ⟩

Yes

TX Mode

```
┌─────────────────────────┐
│ Data Input Routine      │
└─────────────────────────┘
```

6 ⟶ ⟨ End of message ? ⟩   Yes ⟶ 5

No

4

A-42

4

↓

```
┌─────────────────────────────────────┐
│ Load 8 bytes of Initialization      │
│ Vector into Data Security Device     │
└─────────────────────────────────────┘
```

↓

```
┌─────────────────────────────────┐
│    Fetch a plain data byte      │
└─────────────────────────────────┘
```

↓

```
┌─────────────────────────────────────┐
│ Check whether the input             │
│ character is a control character     │
│ such as CNTRL-G or CNTRL-J           │
└─────────────────────────────────────┘
```

↓

```
┌─────────────────────────────────────┐
│ Fetch the 8th encrypted byte        │
│ and Exclusive-or it with the         │
│ plain data byte                      │
└─────────────────────────────────────┘
```

↓

```
┌─────────────────────────────────────┐
│ Transmit cipher byte to             │
│ Modem                                │
└─────────────────────────────────────┘
```

↓

```
┌─────────────────────────────────────┐
│ Provide Delay for some              │
│ special Control characters           │
└─────────────────────────────────────┘
```

↓

```
┌─────────────────────────────────────┐
│ Shift the input to the Data         │
│ Security Device by 8 places to       │
│ form new initialization vector       │
└─────────────────────────────────────┘
```

6 ←————————————

3

↓ RX Mode

```
┌─────────────────────────────────────┐
│ Fetch 8 bytes of initialization     │
│ vector from the link and store       │
│ it in a specified memory space       │
└─────────────────────────────────────┘
```

↓

7

A-43

7

```
┌──────────────────────────────────────┐
│ Load 8 bytes of Initialization       │
│ Vector into the Data Security Device │
└──────────────────────────────────────┘
```

```
┌──────────────────────────────────────┐
│ Fetch a data byte from the link      │
└──────────────────────────────────────┘
```

```
┌──────────────────────────────────────┐
│ Fetch the 8th encrypted byte         │
│ from the Data Security Device        │
│ and Exclusive - or it with the       │
│ received cipher data byte            │
└──────────────────────────────────────┘
```

```
┌──────────────────────────────────────┐
│ Display the deciphered data          │
│ byte on the VDU screen               │
└──────────────────────────────────────┘
```

```
┌──────────────────────────────────────┐
│ Shift the input to the Data          │
│ Security Device by 8 places to form  │
│ new Initialization Vector            │
└──────────────────────────────────────┘
```

< End of Message >   Yes →   5

No

A-44

A 6.2   Program Listing

A 6.2.1   Main Program

Most comments are similar to the ones given in the
Block Encryption Program listing in Appendix 2.

```
30AE        :   LDA   CO51
                STA   COA8
                LDX   @  00
                LDY   @  00
                LDA   @  00
                STA      1B
                STA      F9
                STA      06
                LDA   @  01
                STA      08
                LDA   @  10
                STA      07
                LDA   COA6
                AND   @  01
                BEQ      L1
                JMP   NOLINK

L1              LDA   @  6E
                STA   COAD
                LDA   @  12
                STA   COAD
                LDA   @  14
                STA   COA4
                JSR   BAUD
                LDA   COAA
                JSR   PLAENC
                JSR   FD6A
                LDA   0200
                CMP   @  D0
                BNE      L2
                JSR   DATA
                JSR      R7
                JMP   PLAIN
```

A-45

```
L2          JSR  DSD
P3          LDA  COA6
            AND @ 03
            BNE    P1
            JMP    P2
P1          LDA  C000
            BPL    P3
            LDA @ 33
            STA  COAD
            LDA @ 06
            STA  COA1
            JSR  FD6A
            LDY @ 00

P6          CPY @ 08
            BEQ    P4

P5          LDA  COAB
            BPL    P5
            LDA  COAB
            JSR  DELAY
            AND @ 01
            BEQ    P5
            LDA  02F8,Y
            STA  3000,Y
            STA  COAC
            INY
            JMP    P6

P4          JMP    P7

L4          LDA  COA6
            AND @ 03
            BNE    L3
            JMP    RX
            LDA  C000
            BPL    L4
            LDA @ 33
            STA  COAD
            LDA @ 06
            STA  COA1
            JSR  FD6A
```

A-46

```
P7          CPX  @  00
            BEQ     L5
            JMP     L6
            LDA  @  FF
            JSR  FCA8
            INX
            TXA
            LDY  @  00
            STA     1D

L9          LDX  @  00
            LDA  @  00
            CMP     1D
            BEQ     L7
            JSR  SUB3
            LDA  0200,Y
            STA     EC
            JSR  TEST
            JSR  SUB4

L8          LDA  COAB
            BPL     L8
            LDA  COAB
            JSR  DELAY
            AND  @  01
            BEQ     L8
            LDA     EC
            STA  COAC
            JSR  DELAYLINE
            JSR  SHIFT 1
            DEC     ID
            INY
            JMP     L9
L7          JSR  SWITCH
            JMP     L4

P2          LDA  @  02
            JSR  FCA8
            DEC     07
            LDA     07
            CMP  @  00
```

A-47

```
              BEQ    P11
              JMP    P3
P11           LDA  @  10
              STA     07
              LDA    COAA
              LDA  @  16
              STA    COAD
              LDA     06
              STA    COA1
              LDY  @  00
P14           CPY     08
              BEQ    P12
P13           LDA    COAB
              JSR    DELAY
              AND  @ ´02
              BEQ    P13
              LDA    COAA
              STA    3000,Y
              INY
              JMP    P14
P12           JMP    P15

RX            LDA     02
              JSR    FCA8
              DEC     07
              LDA     07
              CMP  @  00
              BEQ     Z1
              JMP     L4
Z1            LDA    COAA
              LDA     16
              STA    COAD
              LDA  @  06
              STA    COA1
P15           LDA    COAB
              JSR    DELAY
              AND  @  02
              BEQ    P15
              LDA    COAB
              AND  @  10
              BEQ    L23
```

A-48

```
              LDA   COAB
              AND  @ 08
              BEQ   L23
              LDA   COAB
              AND  @ 20
              BEQ   L23
              JMP   L24
L23           JMP   ERR
L24           JSR   SUB3
              LDA   COAA
              STA    EE
              STA    EC
              JSR   SUB4
              JSR   FDF0
              JSR   SHIFT2
              LDA    EC
              CMP  @ 8D
              BEQ   L25
              JMP   P15
L25           JSR   END
              JMP   L4
```

A 6.2.2 _ Subroutines

        Most of the subroutines are the same as the ones
given in the Block Encryption Program in Appendix 2.
Some additional subroutines called by this program are now
briefly described.

SUB3 : This subroutine inputs the data stored in the memory
locations 3000 to 3007 (Initialization Vector) into the Data
Security device after testing the DIR status flag each time.

```
        K3       CPX  @ 08
                 BEQ    K1
                 LDA   COA3
                 AND  @ 40
                 BNE    K2
                 JMP   FF2D
        K2       LDA   3000,X
                 STA   COA0
```

A-49

```
            INX
            JMP    K3
      K1    RTS
```

SUB4 : This subroutine reads the encrypted Initialization
Vector from the Data Security Device and exclusive-ores the
8th encrypted data byte with the plain data byte (the cipher
data byte) to produce the cipher data byte (the plain data
byte).

```
            LDX  @  00
      K6    CPX  @  07
            BEQ    K4
      K5    LDA  COA3
            BPL    K5
            LDA  COA2
            INX
            JMP    K6
            LDA  COA2
            EOR    EC
            STA    EC
            RTS
```

SHIFT 1, SHIFT 2 : These two subroutines are essentially
the same. They shift the memory locations 3000 to 3007
(Initialization Vector) sequentially as follows: 3007 —→ 3006,
3006 —→ 3005, - - -, 3001 —→ 3000 and (cipher data byte)
—→ 3007.

```
            LDX  @  01
            LDA  @  00
            STA    ED
      K7    LDA  3000,X
            DEX
            STA  3000,X
            INX
            INX
            INC    ED
            LDA  @  07
            CMP    ED
```

A-50

```
BNE     K7
LDA     EC
DEX
STA   3000,X
LDX @   00
RTS
```

A- 51

<u>Appendix 7</u>

<u>Results of some Statistical tests on DES Output Sequences</u>

<u>A 7.1 _ Input Samples and Keys used in Statistical Tests</u>

<u>A 7.1.1 _ Input Samples</u>

Input sample period in digits		Input sample (binary/hex)		Abbreviation
1.	5-digits	01110 (binary)		1.1
		10010 "		1.2
		01011 "		1.3
		00011 "		1.4
		10111 "		1.5
2.	8-digits	00000001 (binary)		2.1
		10101010 "		2.2
		11111111 "		2.3
		01000110 "		2.4
		11000011 "		2.5
3.	10-digits	1011111111 (binary)		3.1
		0100010000 "		3.2
		0000011111 "		3.3
		0110000111 "		3.4
		1100111000 "		3.5
4.	20-digits	FFFFC (hex)		4.1
		003FF "		4.2
		E3C3F "		4.3
		68D47 "		4.4
		BBB0F "		4.5
5.	40-digits	FFFFFFFFFC (hex)		5.1
		CE3C3E0FC0 "		5.2
		00000FFFFF "		5.3
		999EEE1116 "		5.4
		62B5A45197 "		5.5
6.	64-digits	FFFFFFFFFFFFFFF0 (hex)		6.1
		00000000FFFFFFFF "		6.2
		C4C4C433CCC4C4C4 "		6.3
		B1B2B3B4B5B6B7B8 "		6.4
		3141592654138547 "		6.5

A-52

A 7.1.2   Keys

I.   1F1F1F1F0E0E0E0E   –   'weak'

II.   1FE01FE00EF10EF1   –   'semiweak'

III.   039649C539317965   –   'random'

IV.   85CDCB1C9BD0851A   –   'random'

V.   3131313131313131   –   'non-random'

A 7.2   Results of Frequency, Serial and Runs tests on
         ciphertext sequences: using different modes under
         a fixed key

         5% Significant levels : Frequency (F) test : 3.84
                                   Serial   (S) test : 5.99
                                   Runs   (R) test : 1.96

         Key : 3131313131313131 (hex)

Input Sample	Mode	F	S	R
1.1	ECB	0.19	3.65	1.19
	CFB	0.77	2.90	-0.29
	CBC	0.66	2.61	-0.04
	CBCP	2.25	4.17	-0.06
	CFBV	0.39	7.36 (**)	-2.24 (**)
1.2	ECP	4.00 (**)	7.69 (**)	-1.26
	CFB	0.25	3.71	-1.24
	CBC	0.39	4.16	1.33
	CBCP	2.06	4.98	0.94
	CFBV	0.77	3.22	-0.73
1.3	ECB	20.81 (**)	23.09 (**)	1.37
	CFB	0.004	2.52	-0.75
	CBC	0.14	2.38	0.44
	CBCP	0.32	2.40	0.20
	CFBV	0.04	2.66	-0.81

A-53

Input Sample	Mode	F	S	R
1.4	ECB	8.63 (**)	15.22 (**)	-1.87 (*)
	CFB	0.47	3.37	0.95
	CBC	0.25	2.89	0.76
	CBCP	0.06	2.33	-0.56
	CFBV	0.02	2.72	0.81
1.5	ECB	4.25 (**)	6.18 (**)	-0.12
	CFB	2.64	4.87	0.40
	CBC	7.22 (**)	9.69 (**)	-0.34
	CBCP	0.04	4.79	1.63 (*)
	CFBV	2.44	4.35	0.01
2.1	ECB	9.00 (**)	10.83 (**)	0.28
	CFB	2.25	4.35	0.07
	CBC	3.52 (*)	5.63 (*)	0.55
	CBCP	0.32	5.75 (*)	-1.87 (*)
	CFBV	0.88	5.48	-1.60 (*)
2.2	ECB	1.00	3.06	-0.03
	CFB	0.77	3.43	0.78
	CBC	1.56	4.00	0.74
	CBCP	1.41	4.68	-1.08
	CFBV	4.52 (**)	6.61 (**)	-0.36
2.3	ECB	25.00 (**)	30.62 (**)	-1.25
	CFB	0.47	5.61 (*)	-1.80 (*)
	CBC	0.02	2.72	-0.88
	CBCP	0.10	2.09	-0.12
	CFBV	0.004	2.95	-1.00
2.4	ECB	9.00 (**)	14.96 (**)	2.30 (**)
	CFB	4.79 (**)	7.27 (**)	-0.67
	CBC	2.85	4.83	0.34
	CBCP	0.25	3.77	-1.24
	CFBV	0.14	2.33	-0.43
2.5	ECB	0	18.27 (**)	4.00 (**)
	CFB	2.44	6.33 (**)	1.46
	CBC	1.72	8.03 (**)	2.12 (**)
	CBCP	0.39	2.45	-0.18
	CFBV	1.27	3.50	0.42

A–54

Input Sample	Mode	F	S	R
3.1	ECB	0.19	5.18	1.70 (*)
	CFB	3.06	7.50 (**)	-1.53 (*)
	CBC	1	3.10	0.41
	CBCP	1.27	3.28	0.29
	CFBV	2.25	4.16	0.07
3.2	ECB	2.64	5.88 (*)	-1.11
	CFB	0.47	2.65	0.45
	CBC	4.00 (**)	7.33 (**)	1.19
	CBCP	0.02	2.52	0.69
	CFBV	0.32	4.33	1.39
3.3	ECB	1.72	4.58	0.99
	CFB	0.004	2.52	0.69
	CBC	1	3.22	0.53
	CBCP	0.004	3.99	1.38
	CFBV	0.035	2.48	-0.69
3.4	ECB	2.07	19.25 (**)	-3.88 (**)
	CFB	0.02	2.17	-0.44
	CBC	3.29	5.34	-0.34
	CBCP	0.77	2.79	-0.29
	CFBV	1.89	3.82	-0.07
3.5	ECB	4.52 (**)	6.82 (**)	0.77
	CFB	0.66	2.93	0.46
	CBC	2.44	5.17	-0.86
	CBCP	0.88	6.95 (**)	2.03 (**)
	CFBV	0.04	2.96	-1.00
4.1	ECB	0.004	2.62	0.75
	CFB	1.89	7.45 (**)	1.94 (*)
	CBC	0.56	2.68	-0.42
	CBCP	1.13	3.71	0.724
	CFBV	0.66	2.88	-0.42
4.2	ECB	0.04	10.14 (**)	2.81 (**)
	CFB	0.02	2.10	0.25
	CBC	0.04	2.27	0.44
	CBCP	0.53	7.92 (**)	-0.71
	CFBV	0.004	2.13	0.31

A- 55

Input Sample	Mode	F	S	R
4.3	ECB	0.00	2.01	0.06
	CFB	1.41	5.50 (*)	1.48
	CBC	1.56	5.65 (*)	-1.45
	CBCP	0.32	2.80	-0.74
	CFBV	1.27	6.98 (**)	-1.90 (*)
4.4	ECB	2.64	5.06	0.77
	CFB	0.19	2.38	0.38
	CBC	2.44	5.37	0.95
	CBCP	0.14	2.28	0.38
	CFBV	0.004	2.94	-1.00
4.5	ECB	3.50 (*)	5.43	0.24
	CFB	0.04	3.09	-1.06
	CBC	0.10	2.47	0.57
	CBCP	0.00	2.00	0.00
	CFBV	0.00	2.94	0.94
5.1	ECB	0.02	2.03	0.06
	CFB	0.32	2.63	0.51
	CBC	1.00	4.26	-1.10
	CBCP	2.07	4.15	-0.37
	CFBV	0.14	3.92	1.32
5.2	ECB	0.56	5.69 (*)	-1.80 (*)
	CFB	0.02	2.00	-0.06
	CBC	0.88	4.80	-1.41
	CBCP	0.02	3.65	1.25
	CFBV	0.47	2.63	0.328
5.3	ECB	0.39	3.07	0.825
	CFB	0.19	2.24	0.131
	CBC	0.04	2.76	-0.87
	CBCP	0.25	3.28	-1.06
	CFBV	1.72	4.16	0.74
5.4	ECB	2.64	25.3 (**)	-4.49 (**)
	CFB	1.12	3.21	-0.10
	CBC	0.14	2.33	-0.43
	CBCP	0.85	5.40	-1.76 (*)
	CFBV	0.04	2.31	0.50

A-56

Input Sample	Mode	F	S	R
5.5	ECB	0.04	4.57	-1.63 (*)
	CFB	0.06	3.53	-1.25
	CBC	0.04	2.05	1.10
	CBCP	0.56	2.80	0.52
	CFBV	0.00	2.00	0.06
6.1	ECB	36.00 (**)	37.70 (**)	1.17
	CFB	0.25	2.80	0.70
	CBC	0.32	2.28	0.01
	CBCP	2.44	4.98	0.70
	CFBV	0.39	4.59	1.45
6.2	ECB	1.00	3.07	0.03
	CFB	0.77	3.76	0.96
	CBC	0.06	2.16	0.25
	CBCP	0.10	2.30	0.44
	CFBV	0.47	2.68	-0.42
6.3	ECB	4.00 (**)	105.60 (**)	10.11 (**)
	CFB	0.56	2.74	0.46
	CBC	1.56	3.92	0.68
	CBCP	0.56	8.11 (**)	-2.36 (**)
	CFBV	0.19	4.38	-1.50 (*)
6.4	ECB	4.00 (**)	9.76 (**)	-1.88 (*)
	CFB	2.44	4.63	-0.49
	CBC	4.52 (**)	8.00 (**)	-1.05
	CBCP	0.04	3.09	-1.00
	CFBV	0.02	3.36	-1.19
6.5	ECB	36.00 (**)	53.46 (**)	-2.98 (**)
	CFB	1.56	3.85	-0.08
	CBC	2.25	5.17	-0.87
	CBCP	0.39	2.48	0.20
	CFBV	4.25 (**)	7.00 (**)	-0.68

A-57

A 7.2.1 _ _Number of sequences classified as non-random by each of
        the three tests

Let the number of sequences classified as non-random by
Frequency test be nF
Let the number of sequences classified as non-random by
Serial test be nS
Let the number of sequences classified as non-random by
Runs test be nR

Mode	nF	nS	nR
ECB	12	16	7
CFB	1	4	0
CBC	3	4	1
CBCP	0	3	2
CFBV	2	4	1

A 7.2.2 _ Number of sequences classified as non-random by more
         than one test

No.	Mode	Input Sample	Tests which indicate non-randomness
1	ECB	1.2	F, S
2		1.3	F, S
3		1.4	F, S
4		1.5	F, S
5		2.1	F, S
6		2.3	F, S
7		2.4	F, S, R
8		2.5	S, R
9		3.4	S, R
10		3.5	F, S
11		4.2	S, R
12		5.4	S, R
13		6.1	F, S
14		6.3	F, S, R
15		6.4	F, S
16		6.5	F, S, R
17	CBC	1.5	F, S
18		2.5	S, R

A-58

No.	Mode	Input Sample	Tests which indicate non-randomness
19		3.2	F, S
20		6.4	F, S
21	CBCP	3.5	S, R
22		6.3	S, R
23	CFBV	1.1	S, R
24		2.2	F, S
25		6.5	F, S

A 7.3   Results of Frequency, Serial and Runs tests on
        Ciphertext sequences; using different keys

5% Significant levels :   Frequency (F) test : 3.84

                           Serial (S) test   : 5.99

                           Runs (R) test   : 1.96

Sample No.	Input Sample
(i) (1.5)	10111
(ii) (2.1)	00000001
(iii) (3.3)	0000011111
(iv) (4.1)	FFFFC (hex)
(v) (5.2)	CE3C3E0FC0 (hex)
(vi) (6.5)	3141592654138547 (hex)

Input Sample No.	Mode	Key	F	S	R
(i)	ECB	I	0.004	3.07	1.00
		II	1.56	4.26	-0.71
		III	7.22 (**)	12.03 (**)	1.93 (*)
		IV	0.25	8.32 (**)	2.45 (**)
		V	4.25	6.18 (**)	-0.12
(i)	CFB	I	1.13	3.28	0.29
		II	4.00 (**)	6.95 (**)	-0.82
		III	0.004	2.23	0.44
		IV	4.25 (**)	6.56 (**)	-0.56
		V	2.64	4.87	0.40

A- 59

Input Sample No.	Mode	Key	F	S	R
(i)	CBC	I	2.25	5.10	1.01
		II	0.06	3.53	-1.25
		III	1.89	4.10	-0.32
		IV	2.25	4.35	0.07
		V	7.22 (**)	9.69 (**)	-0.34
(i)	CBCP	I	0.06	2.90	-0.94
		II	1.13	3.68	0.79
		III	0.56	2.78	0.39
		IV	1.27	3.50	0.42
		V	0.04	4.79	1.63 (*)
(i)	CFBV	I	0.77	2.93	0.46
		II	0.88	8.73 (**)	-2.41 (**)
		III	1.72	4.00	0.62
		IV	0.47	3.50	-1.05
		V	2.44	4.35	0.01
(ii)	ECB	I	25.00 (**)	4.30	-3.37 (**)
		II	1.00	39.35 (**)	6.04 (**)
		III	4.00 (**)	22.15 (**)	-3.96 (**)
		IV	2.07	9.19 (**)	-2.25 (**)
		V	9.00 (**)	10.83 (**)	0.28
(ii)	CFB	I	1.27	4.54	-1.15
		II	1.27	3.91	-0.84
		III	0.66	4.05	-1.17
		IV	0.10	2.24	0.32
		V	2.25	4.35	0.07
(ii)	CBC	I	1.27	3.28	-0.27
		II	1.00	5.41	1.53 (*)
		III	0.56	5.36	-1.67 (*)
		IV	0.47	4.41	-1.42
		V	3.52 (*)	5.63 (*)	0.55
(ii)	CBCP	I	4.00 (**)	15.33 (**)	3.14 (**)
		II	1.13	3.50	-0.65
		III	1.41	5.29	1.36
		IV	0.19	3.81	1.26
		V	0.32	5.75 (*)	-1.87 (*)

A- 60

Input Sample No.	Mode	Key	F	S	R
(ii)	CFBV	I	0.004	2.61	0.75
		II	0.39	2.57	0.45
		III	0.10	3.46	-1.19
		IV	3.06	4.98	0.22
		V	0.88	5.48	-1.60 (*)
(iii)	ECB	I	1.27	3.20	-0.02
		II	3.29	10.33 (**)	-2.16 (**)
		III	6.25 (**)	24.37 (**)	4.22 (**)
		IV	2.25	4.35	0.07
		V	1.72	4.58	0.99
(iii)	CFB	I	0.004	2.03	0.13
		II	1.27	3.21	-0.09
		III	0.04	3.85	1.31
		IV	1.41	8.00 (**)	-2.15 (**)
		V	0.004	2.52	0.69
(iii)	CBC	I	1.27	5.01	-1.34
		II	1.72	3.89	0.30
		III	0.004	2.44	-0.69
		IV	1.41	3.92	-0.65
		V	1.00	3.22	0.53
(iii)	CBCP	I	1.27	3.32	-0.34
		II	0.32	2.31	0.14
		III	0.39	2.40	-0.24
		IV	0.06	3.69	-1.31
		V	0.004	3.99	1.38
(iii)	CFBV	I	2.44	4.47	0.39
		II	1.27	4.83	-1.21
		III	0.14	2.14	-0.18
		IV	2.25	4.63	-0.49
		V	0.04	2.48	-0.69
(iv)	ECB	I	0.004	2.05	0.19
		II	5.94 (**)	11.06 (**)	-1.57 (*)
		III	0.04	7.52 (**)	2.32 (**)
		IV	0.56	3.68	1.02
		V	0.004	2.62	0.75

A-61

Input Sample No.	Mode	Key	F	S	R
(iv)	CFB	I	1.41	4.10	0.80
		II	1.89	6.94 (**)	1.75 (*)
		III	1.13	3.14	0.29
		IV	0.56	2.61	-0.05
		V	1.89	7.45 (**)	1.94 (*)
(iv)	CBC	I	1.41	4.83	-1.21
		II	0.04	4.59	-1.63 (*)
		III	0.32	3.07	-0.87
		IV	2.07	4.26	-0.50
		V	0.56	2.68	-0.42
(iv)	CBCP	I	0.47	2.44	0.08
		II	0.19	2.22	0.006
		III	0.06	2.90	-0.94
		IV	0.77	4.02	1.09
		V	1.13	3.71	0.72
(iv)	CFBV	I	4.00 (**)	5.89 (*)	0.19
		II	0.56	2.68	0.39
		III	0.10	2.14	-0.18
		IV	0.56	3.95	1.14
		V	0.66	2.88	-0.42
(v)	ECB	I	1.56	4.43	1.00
		II	2.07	4.87	0.88
		III	4.52 (**)	6.91 (**)	-0.61
		IV	2.07	9.19 (**)	-2.25 (**)
		V	0.56	5.69 (*)	-1.80 (*)
(v)	CFB	I	5.35 (**)	9.02 (**)	1.49
		II	0.06	2.90	0.88
		III	0.39	3.37	0.95
		IV	0.10	3.57	-1.25
		V	0.02	2.00	-0.06

A-62

Input Sample No.	Mode	Key	F	S	R
(v)	CBC	I	1.13	3.25	-0.22
		II	2.64	6.73 (**)	1.46
		III	0.77	3.07	0.59
		IV	1.00	3.14	0.28
		V	0.88	4.80	-1.41
(v)	CBCP	I	4.25 (**)	6.74 (**)	0.70
		II	4.52 (**)	6.66 (**)	0.14
		III	0.66	2.66	-0.23
		IV	1.89	5.18	1.12
		V	0.02	3.65	1.25
(v)	CFBV	I	2.64	4.76	0.52
		II	0.04	2.07	0.19
		III	0.88	6.58 (**)	-1.91 (*)
		IV	0.14	2.45	0.51
		V	0.47	2.63	0.328
(vi)	ECB	I	9.00 (**)	14.96 (**)	2.30 (**)
		II	1.00	7.07 (**)	2.03 (**)
		III	16.00 (**)	53.94 (**)	-5.59 (**)
		IV	64.00 (**)	65.63 (**)	2.07 (**)
		V	36.00 (**)	53.46 (**)	-2.98 (**)
(vi)	CFB	I	0.14	2.33	-0.43
		II	0.06	2.36	-0.56
		III	0.10	2.14	0.13
		IV	0.47	2.71	-0.55
		V	1.56	3.85	-0.08
(vi)	CBC	I	5.94 (**)	11.28 (**)	1.95 (*)
		II	1.89	4.26	0.56
		III	1.89	3.47	-0.38
		IV	1.00	2.96	-0.16
		V	2.25	5.17	-0.87
(vi)	CBCP	I	30.25 (**)	81.45 (**)	-6.31 (**)
		II	0.77	6.70 (**)	-1.98 (**)
		III	3.16	2.36	-0.30
		IV	0.04	3.39	-1.19
		V	0.39	2.48	0.20

A-63

Input Sample No.	Mode	Key	F	S	R
(vi)	CFBV	I	0.77	2.74	0.15
		II	0.04	2.03	0.06
		III	0.14	2.28	0.32
		IV	0.04	3.83	-1.38
		V	4.25 (**)	7.00 (**)	-0.68

A 7.3.1   Number of sequences classified as non-random by each of the three tests

Let the number of sequences classified as non-random by Frequency test be nF

Let the number of sequences classified as non-random by Serial test be nS

Let the number of sequences classified as non-random by Runs test be nR

Key	nF	nS	nR
I	8	6	4
II	3	10	5
III	5	7	4
IV	2	6	5
V	4	6	1

A 7.3.2   Number of sequences classified as non-random by more than one test

Key	Input Sample	Mode	Tests which indicate non-randomness
I	(ii)	ECB	F, R
	(ii)	CBCP	F, S, R
	(v)	CFB	F, S
	(v)	CBCP	F, S
	(vi)	ECB	F, S, R
	(vi)	CBC	F, S
	(vi)	CBCP	F, S, R

A-64

Key	Input Sample	Mode	Tests which indicate non-randomness
II	(i)	CFB	F, S
	(ii)	ECB	S, R
	(iii)	ECB	S, R
	(iv)	ECB	F, S
	(v)	CBCP	F, S
	(vi)	ECB	S, R
	(vi)	CBCP	S, R
III	(i)	ECB	F, S
	(ii)	ECB	F, S, R
	(iii)	ECB	F, S, R
	(iv)	ECB	S, R
	(v)	ECB	F, S
	(vi)	ECB	F, S, R
IV	(i)	ECB	S, R
	(i)	CFB	F, S
	(ii)	ECB	S, R
	(iii)	CFB	S, R
	(v)	ECB	S, R
	(vi)	ECB	F, S, R
V	(i)	CBC	F, S
	(ii)	ECB	F, S
	(vi)	ECB	F, S, R
	(vi)	CFBV	F, S

A 7.4 Autocorrelation graphs (See pages A-65a to A-65f)

A 7.4.1 Autocorrelation test : a measure of randomness

Let M be equal to the number of points which lie beyond the
5% significance level calculated using normal approximation
(section 6.2.4) Total number of points = 400.

Key	Input Sample	Mode	M/400
I	(i)	ECB	76/400
		CFB	14/400
		CBC	21/400
		CBCP	18/400
		CFBV	15/400

Page 407

A-65
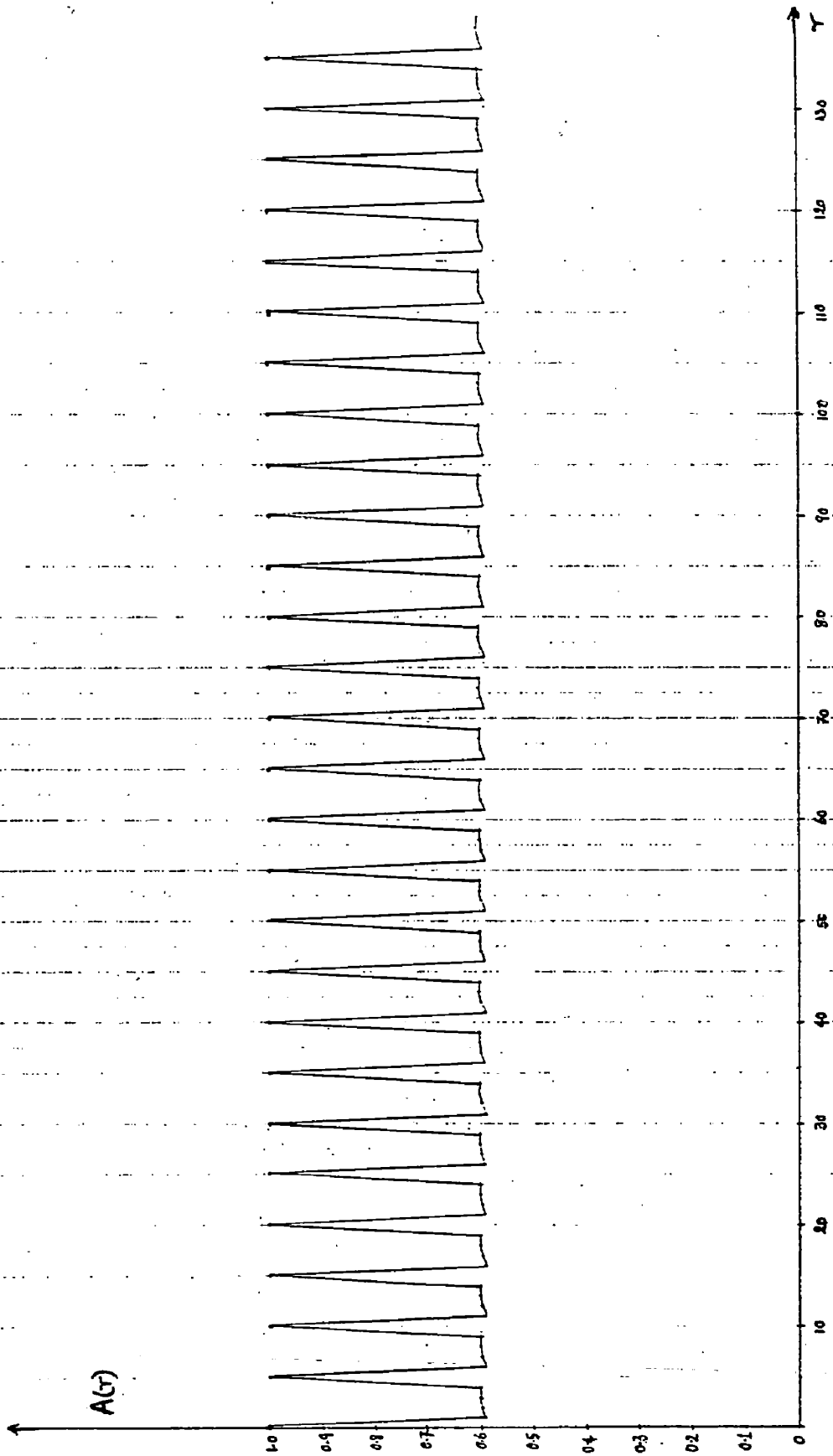
A 7.4 (a) Autocorrelation function of input sample (i)

A-65a

Page 408
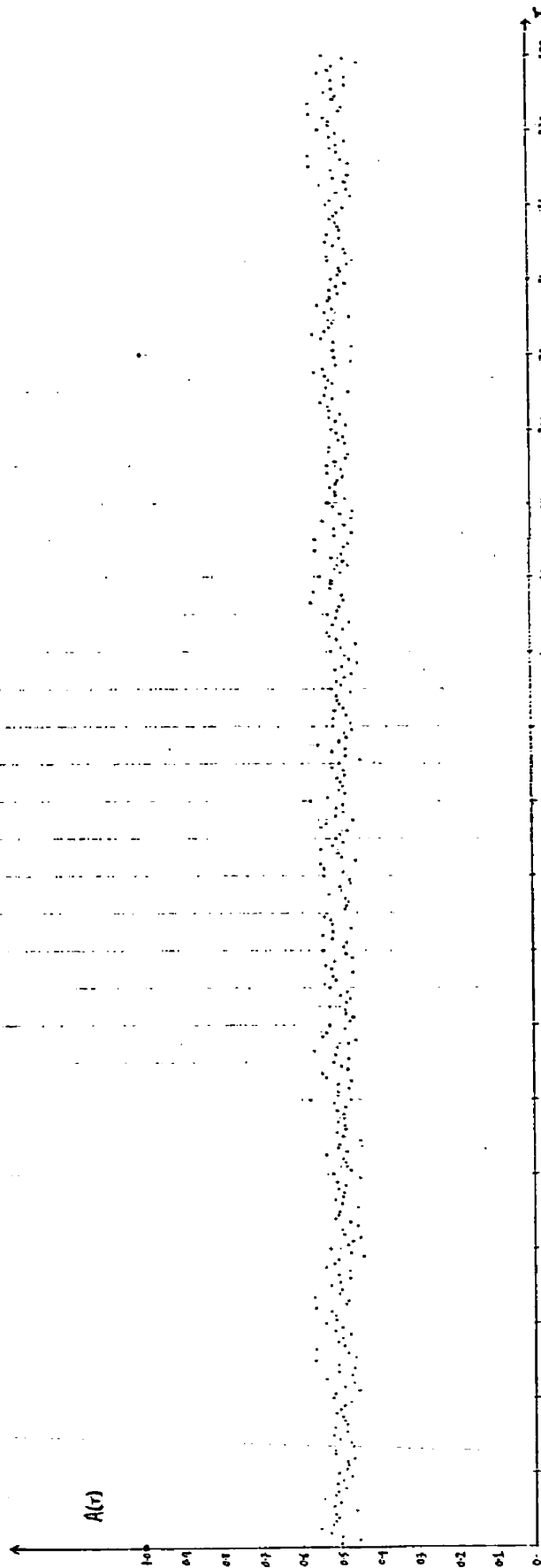
A 7.4 (b) Autocorrelation of ECB output sequence of input sample (1)

A-65b
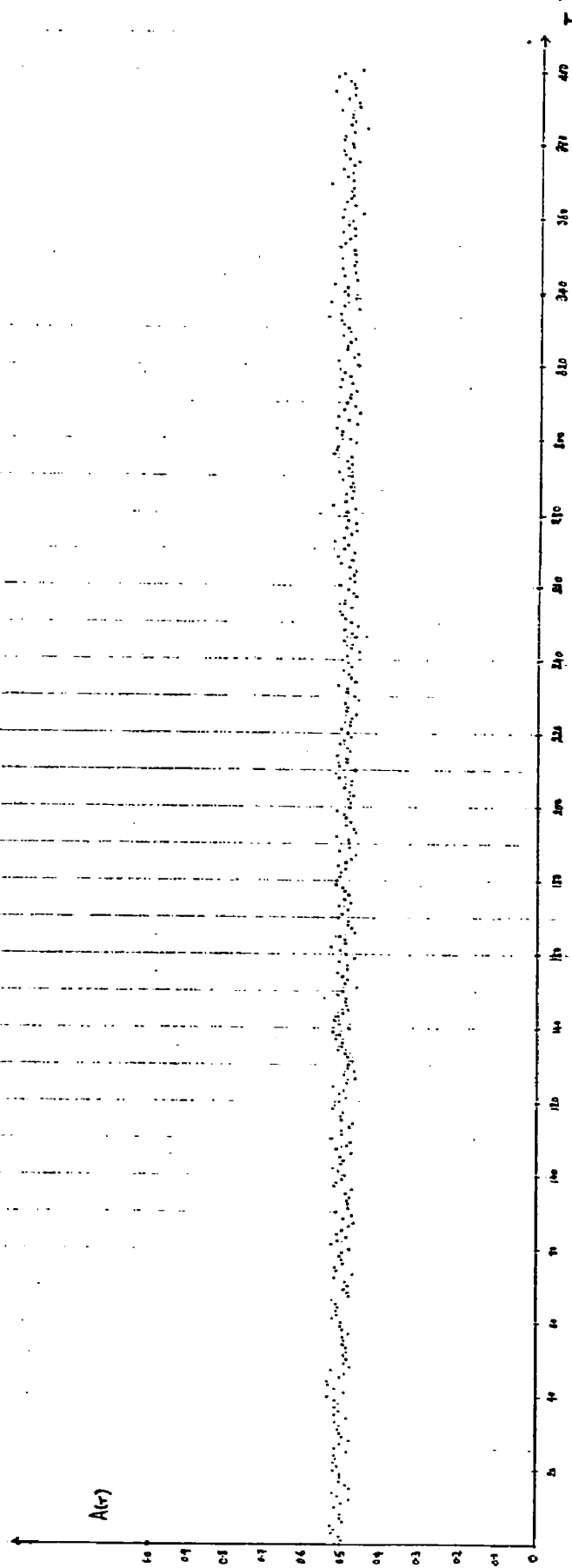
A 7.4 (a) Autocorrelation function of CFB output sequence of input sample (i)

A-65c
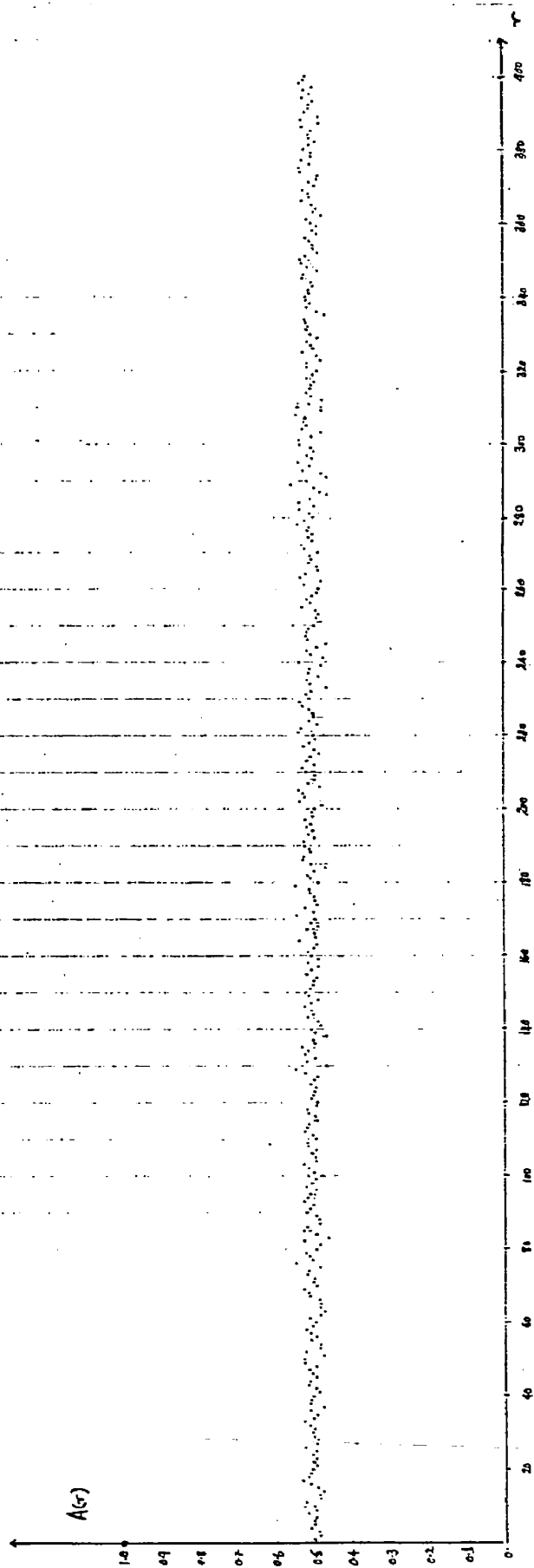
A.7.4 (d) Autocorrelation function of CBC output sequence of input sample (1)

A-65d
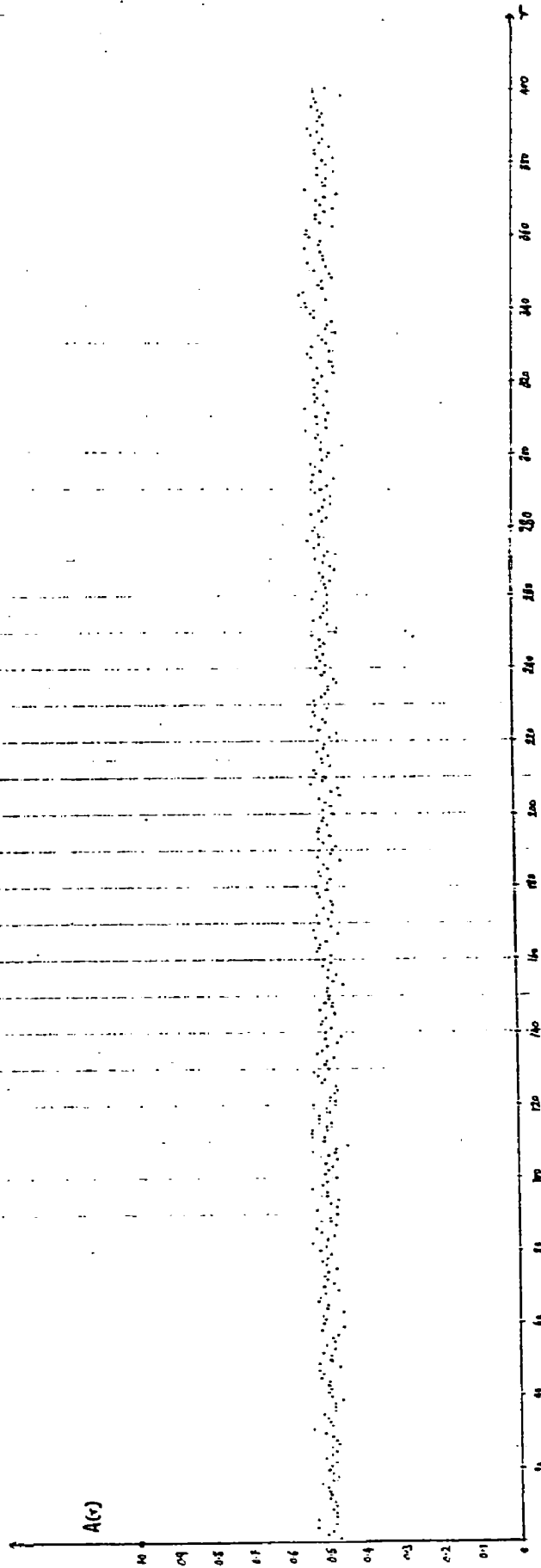
A 7.4 (e) Autocorrelation function of GBCP output sequence of input sample (1)

A-650
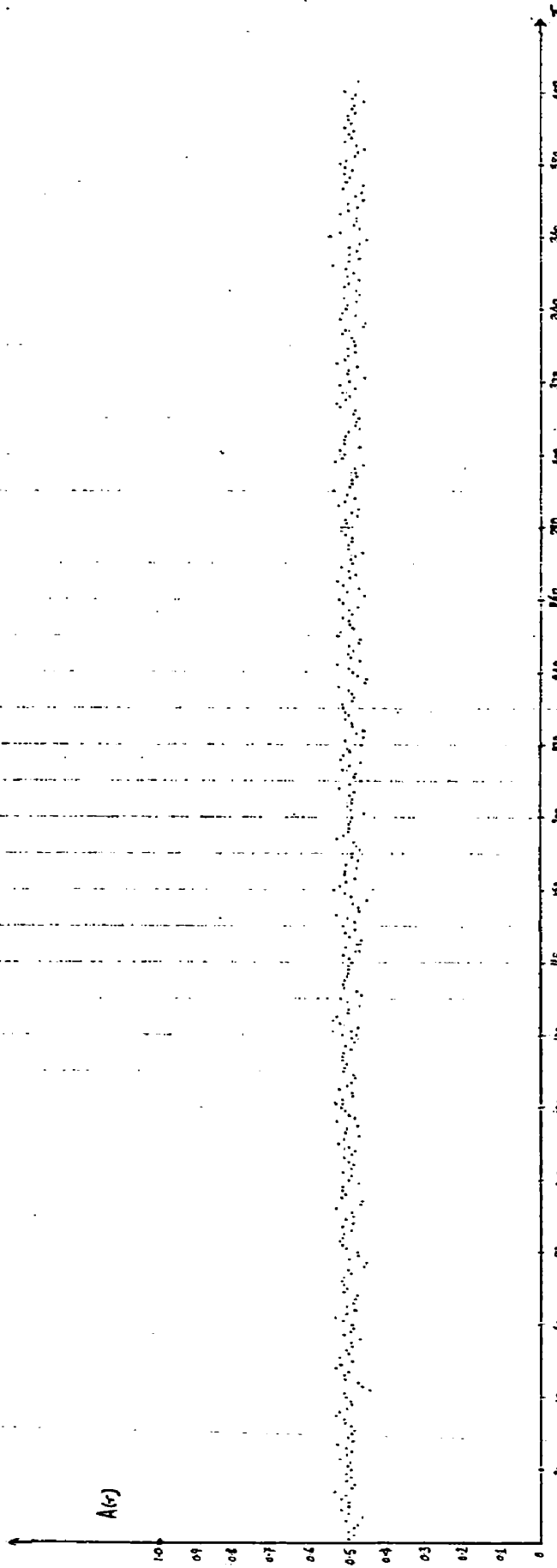
A 7.4 (f) Autocorrelation function of CFBV output sequence of input sample (1)

Λ-65f

Key	Input Sample	Mode	M/400
I	(iii)	ECB	65/400
		CFB	17/400
		CBC	13/400
		CBCP	10/400
		CFBV	21/400
V	(i)	ECB	76/400
		CFB	10/400
		CBC	18/400
		CBCP	37/400
		CFBV	29/400
	(iii)	ECB	77/400
		CFB	17/400
		CBC	20/400
		CBCP	22/400
		CFBV	21/400

A 7.5   Results of Frequency, Serial and Runs tests on
intermediate DES output sequences

5% Significant levels : Frequency (F) test : 3.84
                         Serial   (S) test : 5.99
                         Runs   (R) test : 1.96

(a)   Plaintext : FFFFFFFFFFFFFFFF   (hex)
       Key   : 1F1F1F1F0E0E0E0E   (hex) -   ('Weak')

Round No.	F	S	R
1	14.06	30.80	-2.91
2	0.06	2.40	0.51
3	3.06	6.01	0.94
4	2.25	4.72	0.29
5	0.25	2.40	0.03
6	0.06	2.40	0.51
7	0.25	2.15	-0.22
8	0.06	3.41	-1.25
9	0.00	2.78	-1.01
10	1.56	4.08	-0.31

A-66

Round No.	F	S	R
11	1.00	3.31	0.13
12	0.25	3.16	0.79
13	0.00	2.02	0.00
14	3.06	5.99	-0.39
15	2.25	4.71	-0.75
16	0.06	2.02	0.01
Ciphertext	0.06	2.15	0.01

(b)   Plaintext : FFFFFFFFFFFFFFFF (hex)
       Key   : 85CDCB1C9BD0461A (hex)

Round No.	F	S	R
1	18.06	46.26	-4.23
2	0.00	2.02	-0.25
3	1.56	7.88	2.27
4	2.25	4.72	1.08
5	1.00	3.31	0.13
6	0.25	2.27	-0.47
7	0.56	2.92	-0.44
8	0.56	2.78	-0.18
9	0.25	2.40	0.03
10	0.06	2.40	0.51
11	0.25	3.16	0.79
12	0.06	2.02	0.01
13	0.06	2.15	0.01
14	0.25	3.42	1.04
15	0.25	3.42	1.04
16	0.00	2.02	0.00
Ciphertext	0.25	6.72	2.06

(c)   Plaintext : FFFFFFFFFFFFFFFE (hex)
       Key   : 3131313131313131 (hex)

Round No.	F	S	R
1	10.56	27.31	-3.25
2	0.25	2.27	-0.47
3	0.56	3.67	1.09
4	1.56	4.70	0.98
5	1.56	10.93	-2.64

A-67

Round No.	F	S	R
6	0.06	4.69	-1.76
7	0.56	3.17	0.83
8	4.00	5.63	0.54
9	5.06	7.95	-0.68
10	3.06	4.74	0.14
11	1.56	4.07	0.46
12	2.25	5.98	-1.01
13	1.56	5.21	-1.35
14	0.00	6.59	-2.27
15	1.56	6.61	-1.61
16	2.25	4.72	0.29
Ciphertext	2.25	4.34	-0.49

(d) Plaintext : 016319E1C3C04780 (hex)
     Key   : 3131313131313131 (hex)

Round No.	F	S	R
1	1.56	4.32	-0.58
2	0.25	2.27	-0.47
3	0.25	2.53	0.54
4	2.25	5.09	-0.49
5	1.00	2.79	0.13
6	1.00	4.06	1.15
7	0.56	3.55	0.83
8	0.06	2.02	7.88
9	0.56	2.78	-0.18
10	2.25	4.34	-0.49
11	0.25	2.78	-0.73
12	3.06	4.72	0.41
13	0.25	3.42	-1.23
14	0.25	2.27	-0.47
15	0.00	3.29	1.01
16	1.56	3.43	-0.31
Ciphertext	1.56	4.07	0.46

A-68

Appendix 8

File Security : Cipher Block Chaining Program (CBC)

A 8.1 Basic Flowchart

```
                            START
                              │
                              ▼
                 ┌──────────────────────────┐
                 │ Enter name of file to be │
                 │ Encrypted or Decrypted   │
                 └──────────────────────────┘
                              │
                              ▼
                 ┌──────────────────────────┐
                 │ Fetch file from diskette │
                 └──────────────────────────┘
                              │
                              ▼                              D
          ⟨ Select Encryption (E) or Decryption (D) ⟩─────► 2
                              │ E
                              ▼
                 ┌──────────────────────────┐
                 │ Enter secret key and check/ │
                 │ Correct its parity       │
                 └──────────────────────────┘
                              │
                              ▼
                 ┌──────────────────────────┐
                 │ Determine whether File is │
                 │ Applesoft or Integer Basic│
                 └──────────────────────────┘
                              │
                              ▼
                 ┌──────────────────────────┐
                 │ Load corresponding End of │
                 │ File pointer positions   │
                 └──────────────────────────┘
                              │
                              ▼
                 ┌──────────────────────────┐
                 │ Generate 8 pseudo random │
                 │ characters to form the   │
                 │ Initialization Vector    │
                 └──────────────────────────┘
                              │
                              ▼                   Yes
              ⟨    End  of  File?    ⟩───────────► 1
                              │ No
                              │
        9                     8
```

A-69

8

9

```
┌─────────────────────────────┐
│   Call    SUB5              │
└─────────────────────────────┘
```

1

```
┌─────────────────────────────┐
│   Call    SUB6              │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│  Automatic Transfer         │
│  of Encrypted File onto     │
│  diskette under the file    │
│  name provided by the       │
│  user                       │
└─────────────────────────────┘
```

STOP

2

```
┌─────────────────────────────────┐
│  Determine whether the file     │
│  is Applesoft or Integer Basic  │
│  and load appropriate           │
│  End of File pointer positions  │
└─────────────────────────────────┘
```

```
┌─────────────────────────────────┐
│  Fetch Initialization Vector from│
│  end of file                    │
└─────────────────────────────────┘
```

< End   of   File ? >   Yes   3

No

```
┌─────────────────────────────┐
│   Call    SUB7              │
└─────────────────────────────┘
```

```
┌──────────────────────────────────────┐
│  Initialization Vector = Cipher Block │
└──────────────────────────────────────┘
```

A-70

3

```
┌─────────────────────────────┐
│ Modify the pointer locations │
│ to allow for LIST command    │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Return to appropriate Basic  │
│ prompt signal                │
└─────────────────────────────┘
```

SUB5

```
┌─────────────────────────────┐
│ COUNT1    = 0                │
└─────────────────────────────┘
```

COUNT1   = 8?   Yes →   5

No

```
┌─────────────────────────────┐
│ Exclusive-or Initialization  │
│ Vector byte with file data   │
│ byte                         │
└─────────────────────────────┘
```

Check DIR Status flag   No →   Display
                                       ERROR Message

Yes

```
┌─────────────────────────────┐
│ Input the Exclusive-ored     │
│ result into the Data         │
│ Security Device              │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│ Increment COUNT1 taking      │
│ into account of page         │
│ boundary crossing            │
└─────────────────────────────┘
```

A-71

5

```
┌─────────────────────────────────┐
│ Reset pointers to allow storage │
│ of encrypted file on top of the │
│ original plain file             │
└─────────────────────────────────┘
              │
              ▼
    ┌───────────────────┐
    │   COUNT2  = 0     │
    └───────────────────┘
              │
              ▼
```

COUNT2 = 8? ──Yes──▶ Return

No ◀── Check DOR Status

Yes

```
┌─────────────────────────────┐
│ Read the output from the    │
│ Data Security Device and    │
│ store the output back in    │
│ the same place in memory    │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│ Increment COUNT2 taking into│
│ account of page boundary    │
│ crossing                    │
└─────────────────────────────┘
```

SUB6

```
┌─────────────────────────────┐
│ Generate 8 bytes of         │
│ pseudorandom characters     │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│ Fill the last block by      │
│ padding it with pseudorandom│
│ characters                  │
└─────────────────────────────┘
```

6

A-72

6

```
Encrypt the padded block
and store it back in the
memory
```

```
Record the positions of
pointers
```

Return

SUB7

```
COUNT3  =  0
```

COUNT 3 = 8? ——— Yes ————→

No

Check DIR Status ——— No ——→ Display ERROR MESSAGE

Yes

```
Input a data byte
from Cipher File to
Data Security Device
```

```
Increment COUNT3 taking
into account of page
boundary crossing
```

```
COUNT4  = 0
```

10 ————→ COUNT4 = 8 ? ——— Yes ——→ Return

No

11

A-73

11

```
        ┌────────────────────────────────────┐
   No   <         Check  DOR  Status          >
        └────────────────────────────────────┘
                        │
                        ↓ Yes

        ┌────────────────────────────────────┐
        │  Read the data bytes from          │
        │  the Data Security Device          │
        │  and Exclusive-or it with          │
        │  the Initialization Vector to      │
        │  produce the original plain        │
        │  data                              │
        └────────────────────────────────────┘
                        │
                        ↓

        ┌────────────────────────────────────┐
        │  Store the decrypted data          │
        │  bytes back in the same            │
        │  memory locations as the           │
        │  Cipher data                       │
        └────────────────────────────────────┘
                        │
                        ↓

10      ┌────────────────────────────────────┐
        │  Increment COUNT4 taking           │
        │  into account of page boundary     │
        │  crossing                          │
        └────────────────────────────────────┘
```

A-74

A 8.2 _ _Program_Listing

A 8.2.1 _ Main_Program   Comments

```
5000     :    JSR   FILESECR        Displays 'File Security System'.
              LDA  @ 00
              STA    06              Locations 6000 to 6007 contain
              LDA  @ 60              the Initialization Vector.
              STA    07
              JSR   FILENAME         Displays 'Enter Filename Please'.
              JSR   LOAD             Loads file into system memory
              JSR   ENCDEC           Displays 'Encrypt (E) or Decrypt (D)'.
              JSR   DSD              Initializes  the Data Security
                                     Device and requests and stores the
                                     secret key in the KEY Register
                                     of the device.
                                     Same subroutine as the one given
                                     in Appendix 2.

              LDA    FC
              CMP  @ 00
              BEQ    L1              L1 - Encryption
              JMP    L2              L2 - Decryption
       L1     LDA   AAB6
              BEQ    L3
              LDA    69
              STA    1B
              LDA    6A
              STA    1C
              JMP    L4
       L3     LDA    CA
              STA    1B
              LDA    CB
              STA    1C
       L4     JSR   SUB5
              JSR   SUB6
              JSR   FILETRANSFER

       L2     LDA   AAB6
              BEQ    L5
              LDA   07FE
              STA    1B
```

                                A-75

```
                    STA     69
                    LDA     07FF
                    STA     1C
                    STA     6A
                    JMP     L6

          L5        LDA     9600
                    CMP     FF
                    BNE     L7
                    LDA     00
                    STA     1B
                    LDA     9601
                    TAX
                    INX
                    STX     1C
                    JMP     L6

          L7        TAX
                    INX
                    STX     1B
                    LDA     9601
                    STA     1C


          L6        JSR     SUB7
                    JSR     PROMPT
```

A 8.2.2 _ _Subroutines

        Some of the subroutines used by this program are the
same as the ones used by the Block Encryption Program in
Appendix 2. (for instance, subroutine DSD). Three
additional subroutines namely a merged version of subroutines
SUB5 and SUB7, SUB6 and FILETRANSFER are given in this section.

```
Merged SUB5 and SUB7  :     LDY @   00
              S22           LDX @   00
              S6            LDA     1B
                            STA     FC
                            LDA     19
                            CMP     FC
                            BNE     S1
                            LDA     1C
```

A-76

```
                    STA    FC
                    LDA    1A
                    CMP    FC
                    BNE    S1
                    RTS

S1                  CPX @  08
                    BEQ    S19
                    LDA    C0A3
                    AND @  40
                    BNE    S2
                    JMP    FF2D
S2                  LDA    08
                    CMP @  00
                    BEQ    S3
                    JMP    S4
S3                  LDA    (19),Y
                    EOR    6000,X
                    JMP    S5

S4                  LDA    (19),Y
                    STA    6010,X

S5                  STA    C0A0
                    INX
                    JSR    S10
                    JMP    S6

S19                 JSR    S18
                    JMP    S22

S10                 LDA    AAB6
                    BEQ    S7
                    LDA    19
                    CMP @  FF
                    BEQ    S8
                    INC    19
                    RTS
```

A-77

```
S8              LDA  @  00
                STA     19
                INC     1A
                RTS

S7              LDA     19
                CMP  @  00
                BEQ     S9
                DEC     19
                RTS

S9              LDA  @  FF
                STA     19
                DEC     1A
                RTS

S18             CPX  @  00
                BEQ     S12
                LDA  AAB6
                BEQ     S13
                LDA     19
                CMP  @  00
                BEQ     S14
                DEC     19
                JMP     S15

S14             LDA  @  FF
                STA     19
                DEC     1A
                JMP     S15

S13             LDA     19
                CMP  @  FF
                BEQ     S16
                INC     19
                JMP     S15

S16             LDA  @  00
                STA     19
                INC     1A

S15             DEX
                JMP     S18
```

A-78

```
S12             LDX  @ 00

S21             CPX  @ 08
                BNE    S23
                RTS

S23             LDA  COA3
                BPL    S23
                LDA    08
                CMP  @ 00
                BEQ    S24
                LDA  COA2
                EOR  6000,X
                STA  (19),Y
                LDA  6010,X
                STA  6000,X
                JMP    S20

    S24         LDA  COA2
                STA  (19),Y
                STA  6000,X

    S20         INX
                JSR    S10
                JMP    S21


SUB6   :        LDA  @   00
                STA     FC
                CPX  @   00
                BEQ    S23

    S26         CPX  @   08
                BEQ    S24
                LDA  (1D),Y
                EOR  6000,X
                STA  COA0
                LDA  AAB6
                BNE    S25
                INC    FC

    S25         JSR    S10
                INX
                DEC     1D
                JMP    S26
```

A-79

```
S12             LDX  @ 00

S21             CPX  @ 08
                BNE    S23
                RTS

S23             LDA  COA3
                BPL    S23
                LDA    08
                CMP  @ 00
                BEQ    S24
                LDA  COA2
                EOR  6000,X
                STA  (19),Y
                LDA  6010,X
                STA  6000,X
                JMP    S20

    S24         LDA  COA2
                STA  (19),Y
                STA  6000,X

    S20         INX
                JSR    S10
                JMP    S21


SUB6   :        LDA  @   00
                STA     FC
                CPX  @   00
                BEQ    S23

    S26         CPX  @   08
                BEQ    S24
                LDA  (1D),Y
                EOR  6000,X
                STA  COA0
                LDA  AAB6
                BNE    S25
                INC    FC

    S25         JSR    S10
                INX
                DEC     1D
                JMP    S26
```

A-79

```
S24                JSR   S18
S23                LDA   AAB6
                   BEQ   S27
                   LDA   19
                   STA   07FE
                   LDA   1A
                   STA   07FF
                   JMP   S28

S27                LDA @ 00
                   CMP   19
                   BNE   S29
                   LDA @ FF
                   STA   19
                   DEC   1A
                   JMP   S30

S29                DEC   19

S30                LDA   FC
                   STA  (19),Y
                   LDA  19
                   STA  9600
                   LDA   1A
                   STA  9601

S28                RTS


FILETRANSFER  :    JSR   FE93
                   JSR   FE89
                   CLD
                   LDA @ 30
                   STA   AA5F
                   LDA   03
                   STA   AA64
                   LDA @ 09
                   STA   AA65
                   LDA   AAB6
                   BEQ   S31
                   LDA @ FE
                   STA   AA72
                   LDA @ 07
```

A-80

```
                        STA   AA73
                        SEC
                        LDA    19
                        SBC    FE
                        STA    FC
                        LDA    1A
                        SBC @  07

        S32             STA   AA6D
                        LDA    FC
                        STA   AA6C
                        JSR   FD8E
                        LDA    C6
                        JSR   FDF0
                        LDA @  C9
                        JSR   FDF0
                        LDA @  CC
                        JSR   FDF0
                        LDA @  C5
                        JSR   FDF0
                        LDA @ .FF
                        JSR   FDF0
                        JSR   FD8E
                        JSR   FD6A
                        JSR    S33
                        JSR   A331
                        JSR   FD8E
                        JSR   03D0


        S31             LDA    1A
                        STA   AA73
                        LDA    19
                        STA   AA72
                        SEC
                        LDA @  02
                        SBC    19
                        STA    FC
                        LDA @  96
                        SBC    1A
                        JMP    S32
```

A-81

```
S33              LDY @  00
                 TXA
                 STA    06
                 LDX @  00

S34              LDA @  00
                 CMP    06
                 BEQ    L35
                 LDA    0200,Y
                 STA    AA75,X
                 INX
                 INY
                 DEC    06
                 JMP    S34

S35              CPY @  15
                 BEQ    S36
                 LDA @  A0
                 STA    AA75,Y
                 INY
                 JMP    S35

S36              RTS
```

A-82

Appendix 9

PRESTEL Editing Keyboard

A-83
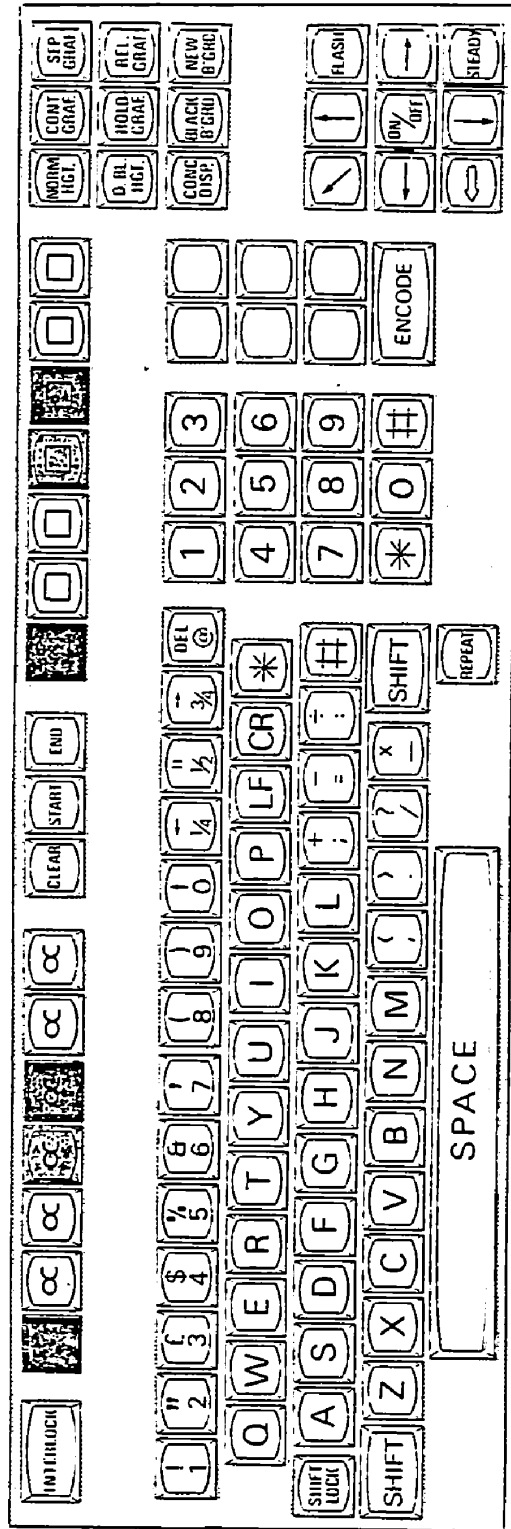
Appendix 10

Chinese _Remainder_Theorem

      The Chinese Remainder Theorem can be stated as follows
[38] :

Let m_1, m_2, ...m_r denote r positive integers that are
relatively prime in pairs and let a_1, a_2,..., a_r denote any r
integers. Then the congruences $x = a_i \pmod{m_i}$ i = 1, 2,...,r,
have common solutions. Any two solutions are congruent modulo
$m_1 \cdot m_2 \cdots m_r$.

Let us now consider an algorithm to implement the Chinese
Remainder Theorem to compute a (mod m) where $m = m_1 \cdots m_r$

Algorithm

1.   For i = 1 to r compute $M_i = m/m_i$

2.   For i = 1 to r compute $M_i^{-1} \pmod{m_i}$

3.   Compute $a = \sum_{i=1}^{r} a_i \, M_i \, M_i^{-1}$

$M_i^{-1} \pmod{m_i}$, $1 \le i \le r$, can be calculated by r
applications of the Euclid's algorithm (Appendix 11). Thus
step 2 requires at most $0 \ (r \log n)$ operations where $m \le n$.
Computation of 'a' in step 3 requires at most $0 \ (r)$ operations.
As $M_i < n$, the amount of storage required for M_i and M_i^{-1},
$1 \le i \le r$, is at most $0 \ (r \log n)$.

A-84

Appendix 11

Euclid's Algorithm

Euclid's algorithm is a rapid method of testing relative primeness and it can be used to compute inverses modulo m. In this thesis, the Euclid's algorithm has been frequently assumed to have been used in determining inverses modulo m. Here the basic algorithm is given. For more detailed treatment refer to [45]

Given positive integers a and b calculate the greatest common divisor, gcd $(a,b) = d$ and x and y such that $ax + by = d$.

This is carried out using the Division Theorem which can be stated as

Given integers $b > 0$ and $a \geq 0$, there exist unique integers $q \geq 0$ and r, $0 \leq r < b$ such that $a = bq + r$

Now applying the Division Theorem successively, it is seen that

$$a = bq + r_0 \qquad \text{(dividing b into a)}$$
$$b = r_0 q_0 + r_1 \qquad \text{(dividing } r_0 \text{ into b)}$$
$$r_0 = r_1 q_1 + r_2 \qquad \text{(dividing } r_1 \text{ into } r_0\text{)}$$
$$r_1 = r_2 q_2 + r_3 \qquad \text{(etc.)}$$

$$\vdots \qquad \vdots$$

$$r_{n-2} = q_{n-1} \cdot r_{n-1} + r_n$$

$$r_{n-1} = q_n \cdot r_n + 0$$

where $r_n < r_{n-1} < r_{n-2} \ldots < r_0 < b$

and r_n is the greatest common divisor of a and b. That is, $d = r_n = \text{gcd}(a,b)$

A-85

The numbers x and   y   can be calculated as follows. To begin with, let x = o,   u = 1   and v = o. Then at each division step, form

$$s \longleftarrow x$$

$$t \longleftarrow y$$

$$x \longleftarrow u \quad - \quad q_{i-1} \; x$$

$$y \longleftarrow v \quad - \quad q_{i-1} \; y$$

$$u \longleftarrow s$$

$$v \longleftarrow t$$

When the repeated division algorithm ends, $d = r_n$ and

$r_n = ax + by$.

Let us now sketch a more formal proof.

Theorem : If r_n is the last non-zero remainder in

Euclid's algorithm for a   and b,   then r_n is the gcd (a,b)

and $r_n = ax + by$ for some x and y.

Proof : If r_n is the last non-zero ramainder in Euclid's

algorithm for a and b,   then the number of steps in the

algorithm is n+1. If   n = o,   then b divides a and the

theorem is trivial. If n = 1,   then the Euclid's algorithm

for a and b has the form

$$a = bq_1 + r_1$$

$$b = r_1 \; q_2 + 0$$

r_1 is the gcd (a,b) and $r_1 = a.1 + b \cdot (-q_1)$

Using induction, assume that the theorem is true for N = n-1.

That is, the theorem is true for any two numbers whose Euclid's

algorithm takes n steps. Suppose the Euclid's algorithm

takes n + 1 steps for a and b. If $a = bq_1 + r_1$ then

the rest of the algorithm for a and b is the Euclid's

algorithm for r_1 and b. So the Euclid's algorithm takes n

steps for r_1 and b. If r_n is the last non-zero remainder

A-86

of Euclid's algorithm applied to a and b, it is the last

non-zero remainder for Euclid's algorithm applied to r_1

and b. By induction r_n is the gcd (b, r_1) and r_n =

$bu + r_1 v \cdot$ Now a = $bq_1 + r_1$ and hence r_n, the gcd (b, r_1)

is also the gcd (a, b). Substituting for r_1 in

the expression $r_n = b u + r_1 v$ gives $r_n = bu + v (a - bq_1)$

ie, $r_n = b (u - vq_1) + a v$. Thus the theorem is true

by induction.

To compute inverse of an integer a (mod m) where gcd (a,m) = 1,

let b = m. The repeated division algorithm can

then be used to find x and y such that ax + my = d.

Hence $ax \equiv d$ (mod m). As d = 1, x is the inverse of

a (mod m). Knuth $\boxed{45}$ shows that inverses (mod m) can be

computed in 0 (log m) operations. Furthermore, note that the

factorization of m into primes is not required to determine

the relative primeness and calculate the inverse.

A- 87

Determinant of a Matrix over GF(2) Program (DETMOD.F77)

```
C x x x x x x x x x x x x x x x x x x x x x x x x
C  This program calculates the determinant of a given NxN matrix
C  over G. F (2) by reducing the matrix to upper triangular form using
C  elementary operations.
C x x x x x x x x x x x x x x x x x x x x x x x x


         INTEGER  B(127, 127),  TEMP(127),  XM1, XM,
       +          X, XM2, XM3, XM4, XM5, SIGN

           N = 127
           READ (1,* ) ((B(I,J),  J = 1,N), I = 1, N)
           L = 1
           SIGN = 1
           K  =  2
           XM1  = 1
70         L1 = L
           XM = B (L,L)
           XM = ABS (XM)
           XM = MOD (XM,2)
           IF (XM. EQ. 0) GO TO 215
214        DO 95 J  = L, N
           XM2  =  B (L, J)
           XM2 = ABS (XM2)
           XM2 = MOD (XM2, 2)
           XM2 = XM2/XM
95         CONTINUE
           DO 141  I = K, N
           X = B (I, L)
           X = ABS (X)
           X = MOD (X, 2)
           IF (X. EQ. 0) GO TO 141
           DO  140 J = L, N
           XM3 = B (I, J)
           XM4 = B (L, J)
           XM3 = ABS (XM3)
           XM4 = ABS (XM4)
           XM3 = MOD (XM3, 2)
           XM4 = MOD (XM4, 2)
```

A- 88

Page 436

```
                      B (I, J)  =  XM3 - XM4 * X
                      IF (B(I,J). EQ. -1) B (I,J) = 1
        140           CONTINUE
        141           CONTINUE
                      L = L + 1
                      K = K + 1
                      XM1 = XM1 * XM
                      IF (L - N) 70, 190, 190
        190           XM5 = B (N,N)
                      XM5 = ABS (XM5)
                      XM5 = MOD (XM5, 2)
                      XM1 = XM1 * XM5
                      XM1 = XM1 * SIGN

        211           WRITE (1, 210) XM1
        210           FORMAT (1X, 'DETERMINANT =   ' , I10 )
                      STOP

        215           L1 = L1 + 1
                      IF (L1. EQ. 128) GO TO 212
                      XM = B (L1, L)
                      XM = ABS (XM)
                      XM = MOD (XM, 2)
                      IF (XM.EQ.0) GO TO 215
                      DO 96 J = L, N
                      TEMP (J) = B(L,J)
                      B (L,J) = B(L1, J)
                      B (L1,J) = TEMP (J)
        96            CONTINUE
                      SIGN = (-1) * SIGN
                      GO TO 214
        212           XM1 = 0
                      WRITE (1,813)
        813           FORMAT (1X, 'MATRIX IS SINGULAR')
                      GO TO 211

                      END
```

A-89

Appendix 13

Matrix Exponentiation Program (MATEXP.FTN)

```
C   x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x
C   This program is used to encrypt and decrypt matrix messages
C   using the extended RSA matrix system.  Exponentiation of the
C   matrix message is performed using the square and multiply
C   technique.
C   x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x

            INTEGER  M(3,3), C(3,3), Z(3,3), U(3,3), V(3,3), X(100), EXP
            REAL Q

            N = 3
            WRITE (1,1)
1           FORMAT (1 H, 'INPUT THE MESSAGE/CIPHER MATRIX PLEASE')
            READ (1,*) ((M(I,J), J = 1, N), I = 1, N)
            WRITE (1,2)

2           FORMAT (1 H, 'ENTER THE ENCRYPTING/DECRYPTING
                            EXPONENT PLEASE')
            READ (1,*) EXP

C           Initialize the C matrix to identity matrix
            DO 3  I = 1, N
            DO 3  J = 1, N
            C (I,J) = 0
3           C (I,I) = 1

C           Convert EXP to a k-bit binary vector (K ≤ 100)
            DO 4 L1 = 1, 100
            Q = EXP/2
            IR = EXP - INT (Q) * 2
            L2 = 101 - L1
            X (L2) = IR
            EXP = INT (Q)
4           CONTINUE
C         Raise the matrix M to the power EXP
            II = 100
9           IF (X(II) .EQ.0) Go to 5
            CALL MULT (M, C, Z)
            DO 6 I = 1, N
            DO 6 J = 1, N
```

A- 90

```
6           C(I,J)  =  Z (I,J)
5           IF (II.EQ.1) GO TO 7
            CALL MULT (M, M, Z)
            DO  8  I = 1, N
            DO  8  J = 1, N
8           M (I,J)  = Z (I,J)
            II = II - 1
            GO TO 9

7           WRITE (1, 10)
10          FORMAT (1H, 'THE CIPHER/MESSAGE MATRIX IS')
            WRITE (1, 11) ((M(I,J), J = 1, N), I = 1, N)
11          FORMAT (1H,  3 (I6, 2X))
            STOP
            END


c   The subroutine MULT multiplies the matrix U by matrix V
c   Over Z/mZ where  m  is the modulus of the public key
c   system.  In this case  m = 299.
            SUBROUTINE MULT (U, V, Z)
            INTEGER U (3,3), V(3,3), Z(3,3)

            DO  12  I1 = 1, N
            DO  12  J1 = 1, N
            Z (I1, J1) = 0
            DO  12  K1 = 1, N
            IP = Z (I1,J1) + U (I1, K1) * V (K1, J1)
            Z (I1, J1) = MOD (IP, 299)
12          CONTINUE
            RETURN
            END.
```

A-91

Appendix 14

Polynomial Exponentiation Program (POLYEXT. F 77)

```
c   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x
c   This program is used to encrypt and decrypt polynomial
c   messages using the extended RSA polynomial system.
c   Exponentiation of the polynomial message is performed using
c   the square and multiply technique.
c   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x


              INTEGER  M(4), V(4), Z(4), EXP, X(100)
              REAL Q
              N = 4
              WRITE (1,1)
1             FORMAT (1H, 'INPUT THE MESSAGE/CIPHER
                        POLYNOMIAL PLEASE')
              READ (1,*) (M(J), J = 1, N)
              WRITE (1,2)
2             FORMAT (1 H, 'ENTER THE ENCRYPTING/
                        DECRYPTING EXPONENT PLEASE')
              READ (1,*) EXP

c         Initialize the V polynomial to identity vector.
              DO 3  II = 1, N
3             V (II) = 0
              V (1) = 1

c         Convert EXP to a K-bit binary vector (K≤100)
              DO  4  L1 = 1, 100
              Q = EXP/2
              IR = EXP - INT(Q) * 2
              L2  = 101 - L1
              X(L2) = IR
              EXP = INT(Q)
4             CONTINUE
c         Raise the polynomial M to the power EXP
              II = 100
9             IF (X (II). EQ.0) GO TO 5
              CALL MULT (M, V, Z)
              DO  6 I = 1, N
```

A- 92

```
6              V (I) = Z (I)
5              1F (II.EQ.1) GO TO 7
               CALL MULT (M, M, Z)
               DO  8 I = 1, N
8              M (I) = Z (I)
               II = II - 1
               GO TO 9

7              WRITE (1, 10)
10             FORMAT (1 H, 'THE CIPHER/MESSAGE POLYNOMIAL 1S')
               WRITE (1, 11)  (M (J), J = 1, N)
11             FORMAT (1H, 4 (I6, 2X))
               STOP
               END

C  The subroutine multiplies the polynomial U by polynomial V
C  over Z/mZ where m is the modulus of the public key
C  system.  The product polynomial is reduced modulo an
C  irreducible polynomial.  In this subroutine, the irreducible
C  polynomial is f (x) = x * * 4  + x * * 2  + 1  and
C  m is equal to 35

               SUBROUTINE MULT (U, V, Z)
               INTEGER U (4), V(4), Z(4)
    Z(1) = U(1) * V(1) - U(4) * V(2) - U(3) * V(3)-U(2)*V(4)+U(4)*V(4)
    Z(2) = U(2)*V(1)+U(0)*V(2)-U(4)*V(3) - U(3)*V(4)
    Z(3) = U(3)*V(1)+U(2)*V(2)+U(1)*V(3)-U(4)*V(2)-U(3)*V(3)-U(2)*V(4)
    Z(4) = U(4)*V(1)+U(3)*V(2)+U(2)*V(3)+U(1)*V(4)-U(4)*V(3)-U(3)*V(4)
               DO 12  I = 1, N
               IP = Z(I)
               Z(I) = MOD (IP, 35)
12             CONTINUE
               STOP
               END
```

A-93

Appendix 15

Cycle Length Calculation Program in GF (2 * * 7) (CYCLE.F77)

```
C x x x x x x x x x x x x x x x x x x x x x x
C  This program is used to calculate the cycle lengths in
C  GF (2**7) for a given secret exponent key when the system
C  base polynomial, evaluated  as a binary vector, is varied
C  from 2 to 127   .  The generator irreducible polynomial used in
C  the system is x * * 7 + x + 1.
C x x x x x x x x x x x x x x x x x x x x x

       INTEGER  M(7,7), S(7,7), U(7), V(7), Z1(7), X(7),
                TEMP(7)
       REAL     Q, Q1

C  Read the multiplication matrix (M) and square matrix (S)
C  from the data file MITDATA
       READ (5, *) ((S(I,J), J = 1,7), I = 1,7)
       READ (5, *) ((M(I,J), J = 1,7), I = 1,7)

       WRITE (1,1)
1      FORMAT (1H, 'INPUT THE BASE AND SECRET KEY PLEASE')
       READ (1,*) BASE, KEY
501    BASE 1 = BASE
C  Convert BASE and KEY to 7-bit binary vectors.
       DO  2  L1 = 1, 7
       Q  = BASE/2
       Q1 = KEY/2
       IR = BASE - INT (Q) * 2
       IR1 = KEY - INT (Q1) * 2
       L2  = 8 - L1
       U (L2) = IR
       X (L2) = IR1
       BASE = INT (Q)
       KEY = INT (Q1)
2      CONTINUE

C  Store the BASE in temporary vector TEMP
       DO  3  I = 1, 7
       TEMP (I) = U (I)
3      CONTINUE
```

A-94

```
c    Initialize the V - vector to 1.
        DO  4  I = 1, 6
        V (I) = 0
4       V (7) = 1

c    Compute the cycle length for a given base.
        ICL = 1
        I = 7
9       1F (X (I).EQ.0) GO TO 5
        CALL MULT (U, V, M)
        DO  6 II = 1, 7
        U (II) = TEMP (II)
6       CONTINUE
5       1F (I.EQ.1) GO TO 7
        CALL SQUARE (U, S)
        DO  8 II = 1, 7
        TEMP  (II) = U (II)
8       CONTINUE
        I = I -1
        GO TO 9
        DO 51 JJ = 1, 7
        CHECK (IC, JJ) = V(JJ)
51      CONTINUE
        ICL = ICL -1
        1F (ICL .EQ. 0) GO TO 56
        DO  521  II = 1, IC
        DO  52  IJ = 1, 7
        1F (V(JJ) .EQ. CHECK (II, JJ)) FLAG = 1
        1F (V(JJ) .NE. CHECK (II, JJ)) FLAG = 0
52      CONTINUE
        1F (FLAG.EQ.1) GO TO 53
521     CONTINUE
56      DO  54 L = 1, 7
        X (L) = V(L)
        U (L) = TEMP1 (L)
54      CONTINUE
        ICL = ICL + 2
        1F (ICL.EQ. 350) GO TO 55
        GO TO 50
```

A-95

```
53        ICCL = ICL + 1
          WRITE (6,400) ICCL
400       FORMAT (1H, 'THE BASE IS =', 2 x, 7 (I4, 1X))
          GO TO 502
55        ICL = ICL -1
          WRITE (6,402) IC, (U(L), L = 1,7)
402       FORMAT (1H, 'MAXIMUM CYCLE', I4, 2X, 7 (I4,1X))
502       BASE = BASE1 + 1
          BASE1 = BASE
          IF (BASE .EQ.128) GO TO 503
          WRITE (6,504)
504       FORMAT (' * * * * * * * * * * * * *  ')
          GO TO 501
503       STOP
          END

C   The subroutine MULT multiplies the polynomial U by the
C   polynomial V over GF (2 * * 7) using the irreducible
C   trinomial x * * 7 + x + 1

          SUBROUTINE MULT (U, V, M)
          INTEGER P(7), P1(7), U(7), P2(7), V(7), M(7,7)

          DO  12  I = 1,7
12        P (I) = 0
          IJ = 7
20        IF (V(IJ).EQ.0) GO TO 13
          DO  14 I = 1,7
          P (I) = U(I) + P(I)
          N = P(I)
          N1 = MOD (N,2)
          P (I) = N1
14        CONTINUE
13        DO  15 K = 1,7
          DO  16 I = 1,7
          P1 (I) = U(I) * M (I,K)
16        CONTINUE
          P2 (K) = 0
          DO 17  I = 1,7
          P2 (K) = P2(K) + P1 (I)
          N = P2(K)
          N1 = MOD (N,2)
```

A-96

```
            P2(K) = N1
17       CONTINUE
15       CONTINUE
         DO  18 L = 1,7
18       U(L) = P2(L)
         IF (IJ.EQ.1)  GO TO 19
         IJ = IJ -1
         GO TO 20
19       DO  21 L = 1,7
21       V (L) = P (L)
         RETURN
         END


c  The subroutine SQUARE squares the polynomial U over
c  GF (2 * * 7) using the irreducible trinomial X * * 7 + X + 1.

         SUBROUTINE SQUARE (U, S)
         INTEGER Z(7), Z1(7), U(7), S(7,7)

         DO  22 J = 1,7
         DO  23 I = 1,7
         Z(I) = U(I) * S(I,J)
23       CONTINUE
         Z1(J) = 0
         DO  24 I = 1,7
         Z1 (J) = Z1(J) + Z(I)
         NUM = Z1 (J)
         NUM1 = MOD (NUM, 2)
         Z1 (J) = NUM1
24       CONTINUE
22       CONTINUE
         DO  25 I = 1,7
25       U(I) = Z1(I)
         RETURN
         END
```

A- 97

Appendix 16

Short Cycling Attack in PKD System over GF (2**7) (RANDCYCLE.F77)

```
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
c  This program carries out short cycling attack on the public
c  key distribution over GF (2**7) with irreducible trinomial
c  x ** 7 + x + 1.  The attack consists of repeated encipherings
c  to determine the cycle length and the secret key.  For a
c  fixed base polynomial, it calculates the average cycle length
c  when the secret keys vary over the range 1 to 127.
c  This procedure is repeated for the 127 possible DEC functions
c  of the form DEC (b * y) where b is any one of the 127
c  polynomials over GF (2 **7).  Then, the expected cycle length
c  for the 127 DEC functions is determined.
c  This program can be run with different system base
c  polynomials.
c
c  The program utilizes the subroutines MULT and SQUARE
c  given in CYCLE.F77.  (Appendix 15).
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

         INTEGER  M(7,7), S(7,7), U(7), V(7), Z1(7), X(7), IC
      +           TEMP1(7), CHECK(350,7),TEMP(7), FLAG, BASE,
      +           TEMP2(7), KEY, U1(7), V1(7), B(7), DEC, KEY1
      +           DEC1, BASE1
         REAL     Q, Q1, QK, SUM, AVR, EXPAVR, EXP

         SUM  = 0.0
         EXP  = 0.0

         WRITE (1,1)
1        FORMAT (1H, 'INPUT THE BASE, SECRET KEY
      + AND DEC POLYNOMIAL B PLEASE')
         READ (1,*) BASE, KEY, DEC
c  Convert BASE, KEY and DEC to 7-bit binary vectors.
         DO   2 L1 = 1,7
         Q = BASE/2
         IR = BASE - INT (Q) * 2
         L2 = 8 - L1
         U (L2) = IR
         BASE = INT (Q)
```

A-98

```
2         CONTINUE
101       DEC1 = DEC
501       KEY1 = KEY
          DO  3  L1 = 1, 7
          QK = KEY/2
          Q1 = DEC/2
          IRK = KEY - INT (QK) *2
          IR1 = DEC - INT (Q1) *2
          L2 = 8 - L1
          X(L2) = IRK
          B (L2) = IR1
          KEY = INT (QK)
          DEC = INT (Q1)
3         CONTINUE

C  Compute the cycle length ICC for a given KEY and DEC
          IC = 1
          DO  38 I = 1,7
          TEMP1 (I) = U(I)
38        CONTINUE
          DO 10 I = 1,6
          V (I) = 0
          V (7) = 1
          I = 7
          DO  40 I = 1,7
          TEMP (I) = U (I)
40        CONTINUE
31        IF (X(I).EQ.0)  GO TO 32
          CALL MULT (U, V, M)
          DO  41  II = 1,7
          U (II) = TEMP (II)
41        CONTINUE
32        IF (I.EQ.1) GO TO 33
          CALL SQUARE (U,S)
          DO  42 II = 1,7
          TEMP (II) = U(II)
42        CONTINUE
          I = I -1
          GO TO 31
```

A-99

```
33         CALL MULT (B, V, M)
           DO  82  I = 1,7
           B (I) = TEMP2(I)
82         CONTINUE
           DO 51 JJ = 1,7
           CHECK (IC, JJ) = V(JJ)
51         CONTINUE
           IC = IC -1
           IF (IC.EQ.0) GO TO 56
           DO 521 II = 1, IC
           DO 52 JJ = 1,7
           IF (V(JJ).EQ.CHECK (II,JJ)) FLAG = 1
           IF (V(JJ).NE.CHECK (II,JJ)) FLAG = 0
           IF (FLAG.EQ.0) GO TO 56
52         CONTINUE
           IF (FLAG.EQ.1) GO TO 53
521        CONTINUE
56         DO  54 L = 1,7
           X (L) = V(L)
           U(L) = TEMP1(L)
54         CONTINUE
           IC = IC + 2
           IF (IC .EQ.350) GO TO 55
           GO to 50
53         ICC = IC + 1
           DO 499 III = 1, 7
           U (III) = TEMP1(III)
499        CONTINUE
           SUM = SUM + ICC -1
           GO TO 502
           SUM = SUM + IC -1

c   Increment the KEY by 1 and repeat the process.
502        KEY = KEY1 + 1
           KEY1 = KEY
           IF (KEY.EQ.128) GO TO 503
           GO TO 501
```

A- 100

```
c   When all the 127 possible values for the KEY have been
c   tried, compute the average cycle length for a given DEC
c   and a given base polynomial.

503         AVR = SUM/127
            WRITE (1,477) SUM, AVR
477         FORMAT (1X, 'SUM =', F12.5, 3X, 'AVR = ', F10.5)
            WRITE (1,504)
504         FORMAT (' * * * * * * * * * * * * * *')

c   Increment the DEC value by 1 and repeat the process of
c   finding different cycle lengths when the KEY takes all the
c   127 possible values

            DEC = DEC1 + 1
            DEC1 = DEC
            KEY  = 1
            KEY1 = KEY
            SUM  = 0.0
            EXP  = EXP + AVR
            IF (DEC.EQ.128) GO TO 102
            GO TO 101

c   When the 127 DEC values of the special form are tried,
c   compute the expected cycle length for the given base
c   polynomial.

102         EXPAVR  =  EXP/127
            WRITE  (1,103) EXPAVR
103         FORMAT (1X, 'THE EXPECTED CYCLE LENGTH IS' F10.5)
            STOP
            END
```

A-101

Appendix 17

Results of Cycling in PKD System over GF (2 ** 7)

A 17.1   Cycle lengths in GF (2 ** 7) for several values of Secret Key, x

Base a	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
2	74	74	74	74	11	74	74	74	42	42
3	118	118	118	118	2	118	118	118	118	118
4	27	53	53	27	53	53	53	27	27	27
5	59	63	59	59	59	63	59	59	63	59
6	75	75	38	75	75	75	75	75	11	38
7	63	13	9	63	9	63	20	63	13	9
8	11	45	65	45	65	6	65	11	65	65
9	1	116	116	116	116	116	116	1	116	116
10	43	3	75	75	43	75	43	43	43	3
11	12	92	10	92	10	12	2	12	12	92
12	53	12	34	4	34	24	4	53	12	12
13	101	8	101	101	13	101	101	101	1	101
14	113	113	113	113	113	113	113	113	113	12
15	47	54	54	47	14	47	54	47	47	54
16	68	9	68	14	9	33	33	68	68	33

Page 450

A-102

Base a	x = 2	x = 4	x = 15	x = 32	x = 64	x =67	x = 68	x = 73	x = 105	x = 115
17	28	89	28	89	28	28	09	28	89	89
18	40	1	40	40	14	16	11	40	40	14
19	106	106	106	106	106	106	8	107	106	106
20	22	30	15	30	30	17	17	22	5	22
21	107	107	107	107	107	107	107	107	107	7
22	52	62	11	62	62	61	62	52	52	62
23	91	91	91	91	91	7	91	91	91	91
24	25	45	45	25	25	45	50	25	25	50
25	18	48	51	18	51	51	51	18	18	48
26	10	50	36	50	26	26	50	10	50	50
27	58	39	58	39	58	58	58	58	58	3
28	80	80	80	43	43	43	80	80	80	80
29	111	111	111	111	111	111	111	111	111	111
30	49	49	16	16	49	46	51	49	7	51
31	27	27	58	27	17	5	58	27	27	9

A-103

Base a	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
32	72	19	72	72	72	72	72	72	72	72
33	66	24	3	66	66	21	21	66	5	66
34	116	116	116	116	116	116	5	116	116	116
35	33	2	56	56	56	34	56	33	56	34
36	92	92	92	92	92	7	21	92	92	92
37	18	18	69	69	33	33	69	18	69	33
38	127	127	127	127	127	127	127	127	127	127
39	50	50	50	50	37	50	50	50	50	14
40	109	109	109	10	109	109	109	109	109	109
41	40	69	69	69	40	69	69	40	69	69
42	57	57	11	50	57	50	57	57	50	50
43	30	79	30	80	5	30	79	30	30	79
44	20	44	20	44	26	44	29	20	44	44
45	42	42	63	63	63	63	63	42	63	63
46	87	87	36	87	36	87	87	87	87	36
47	61	12	3	24	24	61	61	61	12	61

A—104

Base	a	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
48		91	51	91	91	26	91	91	91	26	26
49		78	43	78	78	78	78	78	78	78	78
50		104	18	18	4	104	18	104	104	104	104
51		72	72	18	48	72	72	48	72	48	72
52		2	72	21	13	13	21	13	2	72	72
53		82	15	82	18	82	18	82	82	15	82
54		31	2	52	52	52	52	2	31	52	52
55		37	9	21	8	32	21	8	37	37	32
56		21	21	21	72	72	72	72	21	72	21
57		40	40	49	49	4	49	49	40	4	49
58		14	49	63	14	49	63	49	14	14	49
59		38	27	27	31	38	38	9	38	13	31
60		17	17	13	26	5	31	31	17	17	5
61		12	47	22	37	47	87	22	12	37	12
62		45	45	4	45	66	66	66	45	45	4
63		126	126	126	126	126	126	126	126	126	126

A-105

Base a	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
64	7	33	46	46	33	46	7	7	10	12
65	60	54	58	58	58	58	60	60	60	60
66	75	16	75	16	75	16	75	75	16	3
67	16	87	87	87	16	87	6	16	87	9
68	97	97	97	97	15	97	97	97	97	97
69	12	108	108	108	104	108	108	12	108	108
70	5	79	10	1	79	79	79	5	8	79
71	118	114	118	118	9	118	118	118	118	118
72	88	88	88	88	24	88	88	88	88	88
73	104	104	104	104	104	104	104	104	104	104
74	117	117	117	117	117	117	117	117	117	117
75	103	103	22	103	103	103	103	103	103	103
76	120	120	120	120	120	120	120	120	120	120
77	84	84	84	84	84	84	84	84	84	37
78	28	28	77	77	16	77	28	28	77	77
79	8	28	28	91	91	91	91	8	91	28

A-106

Base	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
80	92	8	92	18	92	9	92	92	7	92
81	121	121	121	121	121	121	121	121	121	121
82	26	66	29	66	66	26	4	26	66	29
83	79	40	79	40	40	79	79	79	79	40
84	21	33	21	64	33	33	64	21	21	64
85	79	79	79	31	79	11	11	79	79	79
86	26	46	46	46	51	46	51	26	4	46
87	81	81	81	81	34	4	34	81	81	81
88	93	93	93	93	19	93	93	93	93	93
89	121	121	121	121	121	121	121	121	121	121
90	92	92	92	15	92	92	92	92	92	92
91	80	80	13	80	80	31	31	80	80	31
92	30	79	5	79	79	30	30	30	8	79
93	118	118	121	118	8	118	118	118	118	118
94	6	47	47	47	43	47	14	6	14	43
95	43	14	47	47	43	43	14	43	47	10

A-107

Base a	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
96	17	71	71	21	1	21	71	17	21	71
97	23	23	27	23	66	23	66	23	66	66
98	97	26	97	97	26	97	26	97	4	97
99	15	111	111	111	111	111	111	15	111	111
100	18	90	90	90	90	15	15	18	18	90
101	84	84	35	6	84	84	84	84	84	35
102	34	40	25	25	40	25	25	34	3	40
103	125	125	125	125	125	125	125	125	125	125
104	24	24	24	24	95	95	95	24	95	24
105	48	27	46	48	48	48	48	48	48	48
106	54	49	49	49	49	23	54	54	49	23
107	24	40	63	63	63	63	40	24	40	63
108	60	44	44	60	44	60	44	60	6	1
109	96	18	96	96	96	18	96	96	96	96
110	108	108	108	108	108	108	108	108	8	8
111	57	57	61	57	61	8	61	57	57	61

A-108

Base a	x = 2	x = 4	x = 15	x = 32	x = 64	x = 67	x = 68	x = 73	x = 105	x = 115
112	42	41	41	22	41	42	41	42	42	16
113	3	90	90	3	90	90	28	3	90	28
114	123	123	123	123	123	123	123	123	123	124
115	96	96	8	96	23	96	96	96	96	8
116	27	25	18	27	2	1	27	27	33	25
117	104	104	104	2	9	104	104	104	104	104
118	66	66	66	15	66	66	66	66	15	88
119	13	97	13	13	97	97	97	13	97	97
120	115	115	115	115	115	115	115	115	115	115
121	35	5	35	19	22	22	19	35	35	22
122	126	126	126	126	126	126	126	126	126	126
123	52	6	26	52	15	52	23	52	52	15
124	82	82	82	82	82	82	1	82	9	8
125	2	66	66	66	13	13	66	2	66	66
126	75	75	75	75	5	75	75	75	75	75
127	110	110	110	110	110	110	110	110	110	110

A–109

Expected cycle lengths for the System Base a = 38,
        using DEC (b * y)

Polynomial b   Expected
(evaluated as   Cycle length
a binary vector)

Polynomial b	Expected Cycle length	
1	127.00	(Refer to Section A 17.1,
2	38.87	a = 38 cycle length = 127)
3	54.48	
4	66.39	
5	94.67	
6	43.80	
7	35.87	
8	22.75	
9	47.87	
10	121.11	
11	44.50	
12	87.55	
13	27.17	
14	29.77	
15	113.61	
16	39.87	
17	43.36	
18	64.50	
19	103.19	
20	49.77	
21	109.85	
22	66.54	
23	53.09	
24	127.00	
25	109.95	
26	68.83	
27	76.54	
28	35.76	
29	52.15	
30	51.93	
31	47.80	
32	99.28	

A-110

33	73.16
34	53.17
35	29.39
36	46.40
37	121.11
38	45.44
39	53.02
40	71.49
41	52.94
42	64.42
43	70.09
44	60.95
45	127.00
46	57.27
47	45.50
48	44.00
49	123.05
50	43.02
51	75.60
52	27.30
53	54.87
54	29.71
55	35.60
56	38.28
57	64.00
58	35.71
59	67.99
60	48.39
61	46.89
62	54.40
63	77.71
64	77.71
65	69.69
66	61.60
67	82.09
68	49.79
69	60.18
70	78.09

A-111

Page 459

71	62.91
72	59.52
73	108.20
74	115.38
75	36.04
76	85.96
77	63.08
78	65.25
79	119.19
80	54.83
81	44.69
82	111.77
83	53.17
84	81.11
85	37.35
86	43.36
87	52.97
88	52.26
89	42.43
90	51.00
91	65.90
92	67.20
93	33.47
94	54.69
95	77.88
96	98.20
97	34.20
98	38.67
99	56.65
100	26.26
101	43.66
102	60.75
103	68.02
104	37.00
105	60.56
106	48.98
107	49.60
108	106.53

109	87.68
110	55.31
111	54.35
112	78.11
113	33.91
114	48.41
115	52.91
116	43.87
117	42.75
118	39.53
119	40.97
120	75.69
121	61.03
122	60.71
123	41.87
124	70.98
125	32.64
126	32.24
127	69.25

A.17.3 Average Expected Cycle lengths for several values of System Base

System Base	Average Expected Cycle Length
8	67.65
15	65.44
28	63.28
38	61.16
42	65.59
53	64.16
64	66.41
79	61.91
83	66.61
90	63.12
98	64.11
104	64.17
117	63.81
124	61.51

A-113

Appendix 18

Public Key Distribution Program Listing

A 18.1 Main Program

```
                                        Comments

     PKD     :     LDA   CO51
                   STA   COA8
                   LDA @ 66
                   STA    FA
                   LDA   COA6
                   AND @ 01
                   BEQ    L1
                   JMP NOLINK        Displays NOLINK Error
     L1            LDA @ 6E
                   STA   COAD
                   LDA @ 14
                   STA   COA4
                   JSR   BAUD        Sets Baud Rate
                   LDA   COAA
                   JSR   TRANS       Displays 'If wishing to Transmit
                                     enter your user ID and the
                                     receiver's user ID'.
     L4            LDA   COA6
                   AND @ 03
                   BNE    L3
                   JMP    RX
     L3            LDA   C000
                   BPL    L4
                   JSR   FD6A
                   LDA @ 33
                   STA   COAD
                   INX
                   LDY @ 00
     L2            CPX @ 00
                   BEQ    L5
     K1            LDA   COAB
                   BPL    K1
                   LDA   COAB
                   JSR   DELAY
                   AND @ 01
```

A- 114

```
          BEQ    K1
          LDA    0200,Y
          STA    COAC
          INY
          DEX
          JMP    L2
          LDA    0202
          AND    @ OF
          ASL
          ASL
          ASL
          ASL
          STA    F9
          LDA    @ 16
          STA    COAD
K2        LDA    COAB
          JSR    DELAY
          AND    @ 02
          BEQ    K2
          LDA    COAA
          CMP    0202
          BEQ    K3
          JMP    K4
K3        LDA    @ 33
          STA    COAD
          JSR    SUB8       Stores the public key of B, $y_B$,
                            in 6100 to 610F
          JSR    INPUTKEY   Displays 'Input Secret Key'
          JSR    SUB9       Stores the secret key, $x_A$, in
                            6200 to 620F.
          JSR    EXP        Forms $K_{AB} = (y_B)^{x_A}$ in GF(2**127)
                            and clears 6200 to 620F. $K_{AB}$
                            stored in 6000 to 600F.
          JSR    SUB10      Stores the common key $K_{AB}$ in the
                            KEY Register of the Data Security
                            Device.
          JSR    SUB11      Stores the session key, $K_S$, 6700
                            to 6707 in the DATA Register of
                            the Data Security Device.
```

A- 115

```
              JSR  SUB12      Forms E_{K_{AB}} (K_s) and transmits it
                             to the receiver B.
              INC  6700
              JSR  SUB11      Forms E_{K_{AB}} (K_s +1) for
                             authentication purposes and stores
              JSR  SUB13      it in 6710 to 671F
              LDY  @ 00
              LDA  @ 16
              STA  COAD
       L22    LDA  COAB
              JSR  DELAY
              AND  @ 02
              BEQ  L22
              LDA  COAA
              STA  6720,Y
              INY
              CPY  @ 08
              BEQ  L23
              JMP  L22
       L23    CPY  @ 00
              BEQ  L24
              LDA  6720,Y     Verify received E_{K_{AB}} (K_s +1)
              CMP  6710,Y
              BNE  L25
              DEY
              JMP  L23
       L24    JSR  SUB14      Load K_s into KEY Register of
              JSR  SUB10      Data Security Device
              JSR  SUB15      Clear all temporary memory locations.


              Jump to Block Encryption or
              Stream cipher Feedback or
              Cipher Block Chaining Program
              for DES encrypted data communication.


       RX     LDA  @ 16
              STA  COAD
       R1     LDA  COAB
              JSR  DELAY
              AND  @ 02
```

$$E_{K_{AB}} (K_s)$$

$$E_{K_{AB}} (K_s +1)$$

$$E_{K_{AB}} (K_s +1)$$

A-116

```
          BEQ    R1
          LDA    COAA
          AND    @ OF
          ASL
          ASL
          ASL
          ASL
          STA    F9
          LDX    @ 02
T3        CPX    @ 00
          BEQ    T1
T2        LDA    COAB
          JSR    DELAY
          AND    @ 02
          BEQ    T2
          LDA    COAA
          STA    FD
          DEX
          JMP    T3
T1        LDA    33
          STA    COAD
          JSR    INPUTKEY
          JSR    SUB9         Stores the secret key of B, $x_B$, in
                              6200 to 620F.
R2        LDA    COAB
          BPL    R2
          LDA    COAB
          JSR    DELAY
          AND    @ 01
          BEQ    R2
          LDA    FD
          STA    COAC
          JSR    SUB8         Stores the public key of A, $y_A$, in
                              6100 to 610F.
          JSR    EXP          Forms $K_{AB} = \left(y_A\right)^{x_B}$ in GF (2 **127)
                              and stores it in 6000 to 600F.
          LDA    @ 16
          STA    COAD
          LDA    COAA
```

A-117

```
        LDY @  00
R6      CPY @  08
        BEQ    R5
R4      LDA  COAB
        JSR  DELAY
        AND @  02
        BEQ    R4
        LDA  COAA
        STA  6700,Y
        INY
        JMP    R6
R5      LDA @  OE
        STA  COA1
        JSR  SUB10        Stores the common key $K_{AB}$ in the
                          KEY Register of the Data Security Device.
        JSR  SUB11        Stores the received encrypted session
                          key $K_S$ in the DATA Register of the
                          Data Security Device.
        JSR  SUB13        Decrypted $K_S$ is stored in 6710
                          to 671F.
        LDY @  00
R9      CPY @  08
        BEQ    R8
        LDA  6710,Y
        EOR  6008,Y
        STA  6700,Y
        INY
        JMP    R9
R8      LDA @  33
        STA  COAD
        INC  6700
        LDA @  06
        STA  COA1
        JSR  SUB11        Encrypts $K_S+1$ using $K_{AB}$ and
        JSR  SUB12        transmits the cipher to the sender A.
        JSR  SUB14        Load the session key $K_S$ into the
        LDA @  00         KEY Register of the Data Security
        STA  COA1         Device.
        JSR  SUB10
```

A-118

```
                JSR   SUB15          Clear all temporary memory
                                     locations

                Jump to Block Encryption
                or Cipher Feedback program
                for encrypted data communication.

A 18.2  Subroutines

        Listings of some important subroutines used by the
main program are given below.

                                          Comments
SUB8        :   LDY  @  00       Stores the public key in memory
        S2      CPY  @  10       locations 6100 to 610F and in
                BEQ     S1       6500 to 650F
                LDA    (F9),Y
                STA    6100,Y
                STA    6500,Y
                INY
                JMP     S2
        S1      RTS


SUB9        :   LDY  @  00       Stores the secret key in memory
        S4      CPY  @  10       locations 6200 to 620F
                BEQ     S3
                JSR    FD0C
                STA    6200,Y
                JSR    FDF0
                LDA     4E
                STA    6700,Y
                LDA  @  00
                STA    6000,Y
                STA    6400,Y
                INY
                JMP     S4
        S3      LDA  @  01
                STA    600F
        S6      CPY  @  00
                BEQ     S5
                LDA  @  88
```

A-119

```
                    JSR    FDF0
                    DEY
                    JMP    S6
          S5        JSR    FD8B
                    RTS


SUB10               LDA  @ 06        Stores the common key K_AB in
                    STA    C0A1      the KEY Register of the Data
                    LDA    C0A3      Security Device.
                    AND  @ 10
                    BNE    S7
                    JMP    FF2D
          S7        LDY  @ 00
          S9        CPY  @ 08
                    BEQ    S8
                    LDA    6000,Y
                    JSR    PARITY CHECK
                    LDA    FB
                    STA    C0A0
                    LDA    C0A3
                    AND  @ 20
                    BEQ    S10
                    JMP    FF2D
          S10       INY
                    JMP    S9
          S8        RTS          ·
SUB11       :       LDY  @ 00        Encrypts K_s +1 under the
                    CPY  @ 08        key K_AB using the DES
                    BEQ    S11       Block Encryption mode
                    LDA    C0A3
                    AND  @ 40
                    BNE    S12
                    JMP    FF2D
          S12       LDA    6700,Y
                    EOR    6008,Y
                    STA    C0A0
                    JMP    S13
          S11       RTS
```

A-120

```
SUB13     :        LDY  @ 00        Stores the encrypted K_s +1 under
          S17      CPY  @ 08        the key K_AB in memory locations
                   BEQ   S15        6710 to 671F.
          S16      LDA  COA3
                   BPL   S16
                   LDA  COA2
                   STA  6710,Y
                   INY
                   JMP   S17
          S15      RTS


SUB14     :        DEC  6700        Stores the session key in the
                   LDY  @ 00        memory locations 6000 to 600F
          S22      CPY  @ 08        which is then transferred to the
                   BEQ   S23        KEY Register of the Data Security
                                    Device.
                   LDA  6700,Y
                   STA  6000,Y
                   INY    .
                   JMP   S22
          S23      RTS


SUB15     :        LDY  @ 00        Clears all temporary memory
          S24      CPY  @ 10        locations used in this PKD program.
                   BEQ   S25
                   LDA  @ 00
                   STA  6200,Y
                   STA  6700,Y
                   STA  6710,Y
                   STA  6720,Y
                   STA  6000,Y
                   INY
                   JMP   S24
          S25      RTS

SUB12     :        LDY  @ 00        Used to encrypt either K_s under
          S18      CPY  @ 08        K_AB  or K_s+1 under K_AB which
                   BEQ   S21        is then transmitted to the other
          S19      LDA  COAB        end.
```

A-121

```
               BPL    S19
               LDA    COAB
               JSR    DELAY
               AND  @ 01
               BEQ    S19
       S20     LDA    COA3
               BPL    S20
               LDA    COA2
               STA    COAC
               INC
               JMP    S18
       S21     RTS
```

A-122

Appendix 19

Determination of A Normal Basis Generator in GF(2 * * 127)
(FINDNORBAS. F77)

```
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
c  This program selects elements of the form v = x * * ILL + 1
c  over GF (2 * * 127) where ILL varies from 1 to 126 and
c  computes the conjugates v * * 2, v * * 4, . . . , v * * (2 * *
c  (127-1)) over GF(2).  All operations are done modulo
c  the irreducible polynomial x * * 127 + x + 1.  Then it
c  determines whether these 127 conjugates are linearly
c  independent by calling the program DETMOD. F77 (Appendix 12).
c  DETMOD. F77 finds the determinant of the 127 x 127 matrix
c  formed by the 127 conjugates.  If the determinant is not
c  equal to zero, then the conjugates form a normal basis
c  over GF (2 * * 127).
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

              INTEGER  TEMP(127), B(127, 127), V(127), C(127)
              ILL = 1

c  Choose an element over GF (2 * * 127) of the form
c  x * * ILL + 1
42            DO  1  I = 1, 127
1             V (I)  = 0
              V (ILL) = 1
              V (1)  =  1

c  Compute the 127 conjugates and form the matrix B
              IJ  = 1
              DO  6 IL = 1, 127
              B (IJ, IL) = V(IL)
6             CONTINUE
              IJ = IJ +1
100           I = 1
              C (1) = V(1)
              DO  2 I = 1,63
              J = 64 + I
              K = I + 1
              JJ = 2 * I
              KK = 2 * I + 1
              C (JJ) = V(J)
```

A-123

```fortran
                    L = V(K) + V(J)
                    C (KK) = MOD (L,2)
2                   CONTINUE
                    DO 3 IK = 1, 127
                    V (IK) = C(IK)
                    B (IJ, IK) = C(IK)
3                   CONTINUE
                    IJ = IJ + 1
                    IF (IJ.EQ.128) GO TO 110
                    GO TO 100

C  Find the determinant of the matrix B using DETMOD. F77
                    CALL  DETMOD (B, IDET)
                    WRITE (1,210) ILL, IDET
210                 FORMAT (1X, 'DET = ', I10, 2X, 'ILL = ', I5)

C  If the matrix B is singular over GF(2), increment ILL
C  by one and repeat the procedure until ILL = 128.  If the
C  matrix B is non-singular stop.
                    IF (IDET. EQ. 0)  GO TO 810
                    STOP
810                 ILL = ILL + 1
                    IF (ILL.EQ.128) GO TO 221
                    GO TO 42
221                 STOP
                    END
```

A-124

Appendix 20

Normal Basis Exponentiator - Circuit Diagram
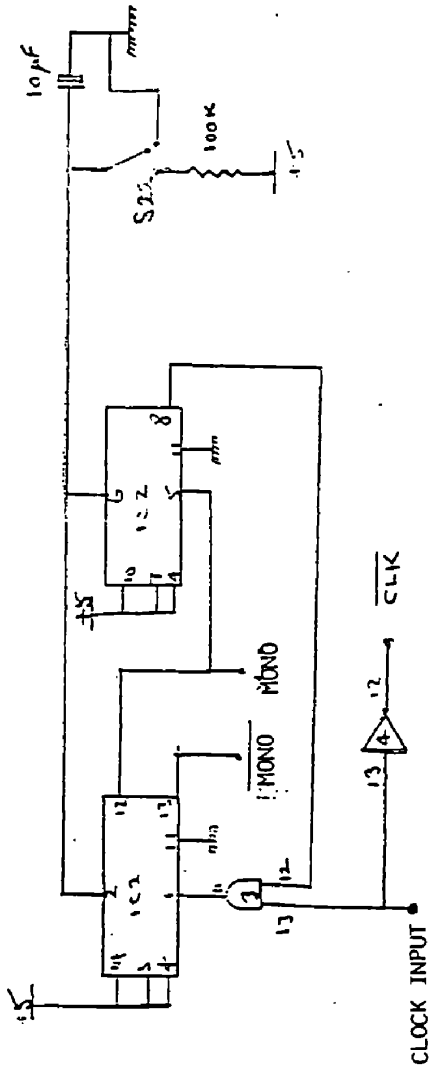
SWITCH PANEL LAYOUT (FRONT)

A-125

A-126
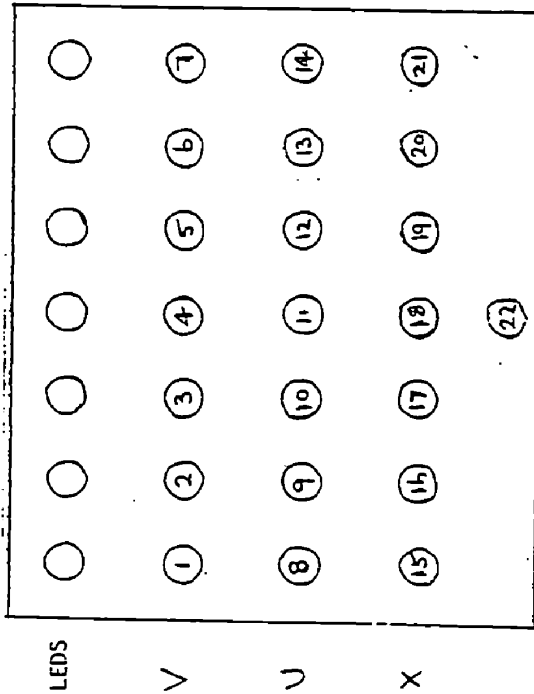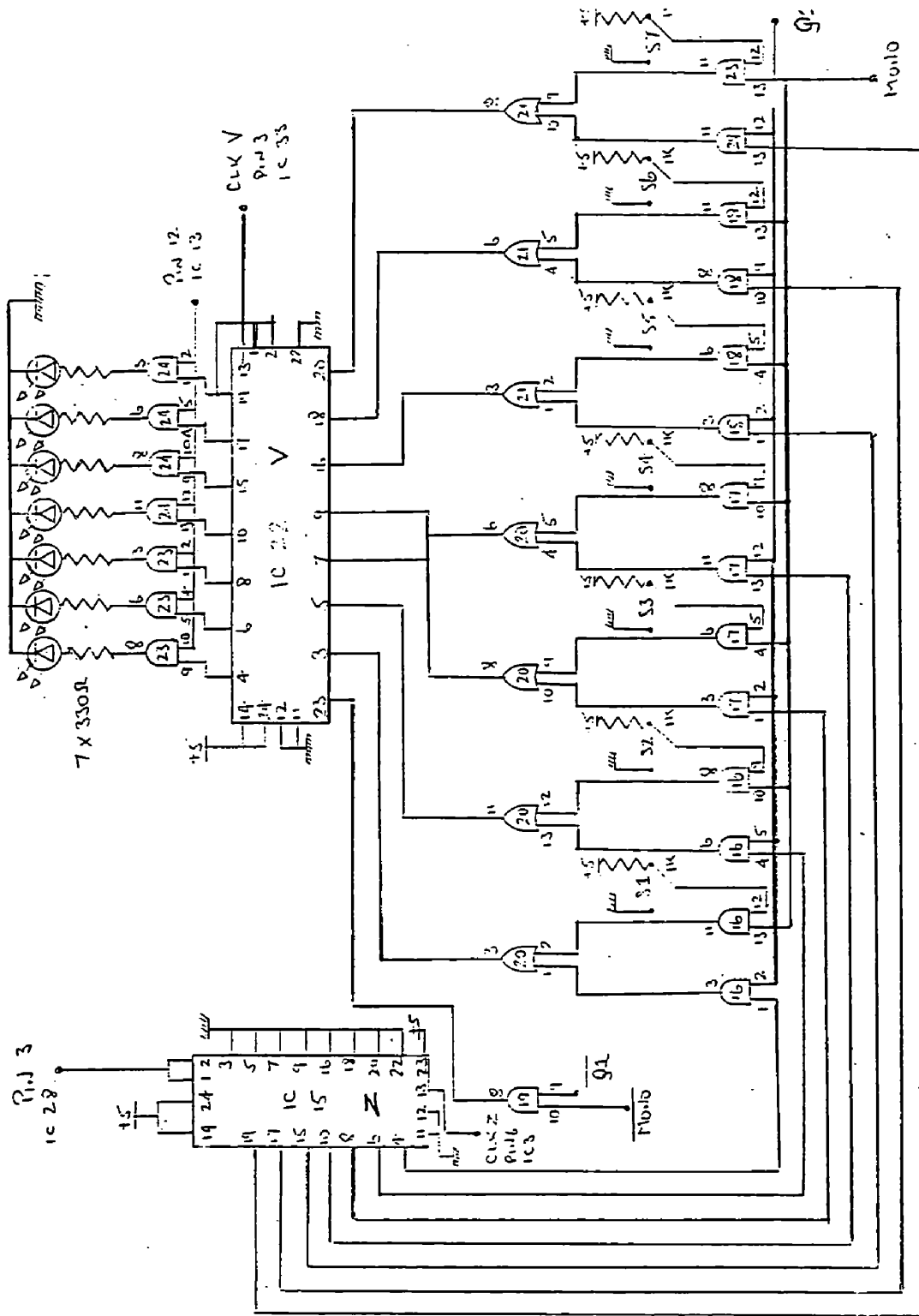
A-127
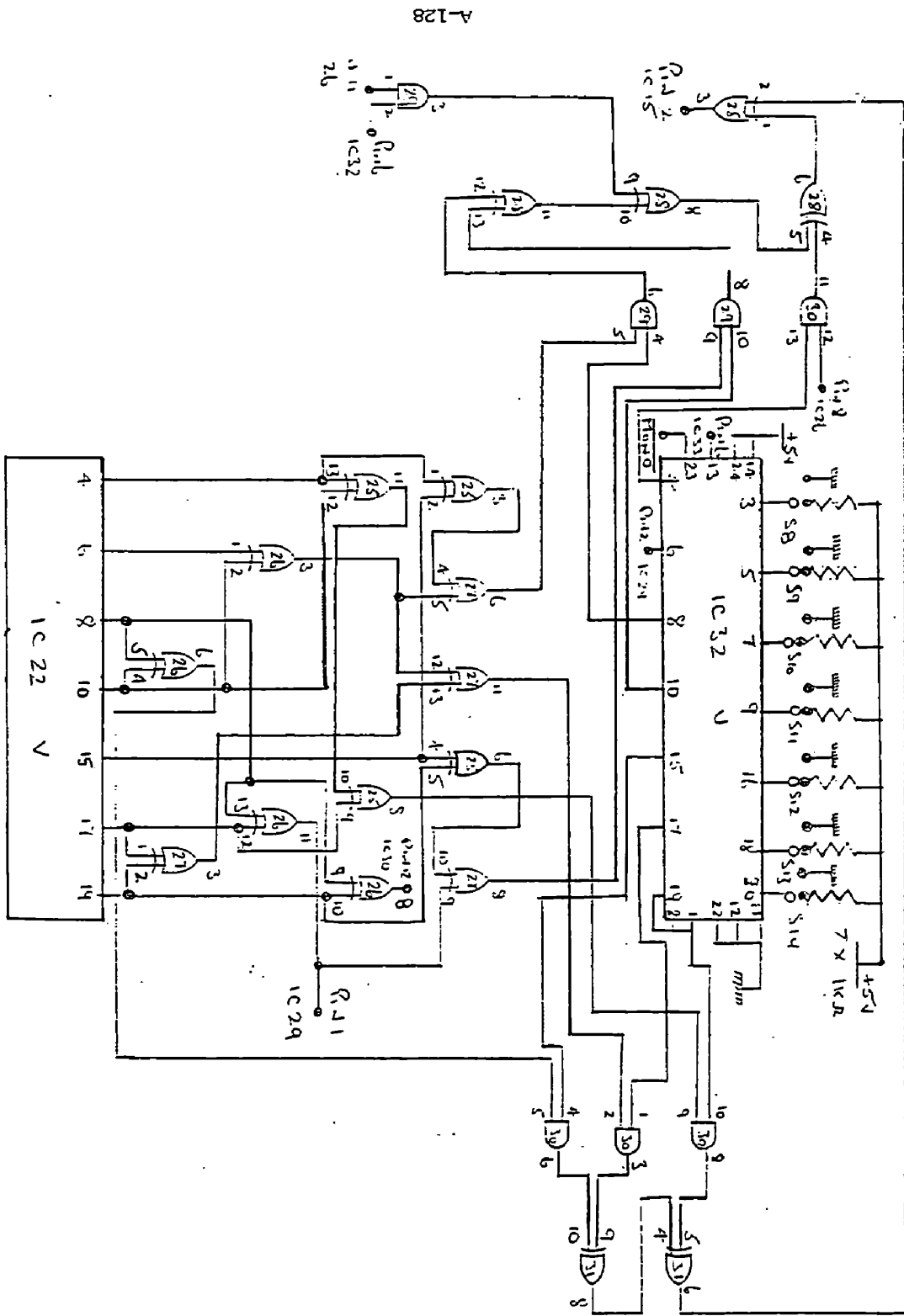
Appendix 21

Multiplier Matrix Program (M-MATRIX.F77)

```
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
c  This program calculates the M-matrix required for implementing
c  multiplication over GF (2 * * N) using the normal basis
c  generated by the program  FINDNORBAS. F77.  Here N = 127.
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

           INTEGER  B(127, 127),  U(127), V(127), Z(127), X(127, 127),
          +         S(8128, 127), INV(127, 127), X1(127, 127),
          +         FLAG, TEMP(127), T1, T2, T3, T.
       COMMON/B1/B
       COMMON/B3/S
       COMMON/B6/1NV
       COMMON/B7/U/V/Z/TEMP
       COMMON/B13/X
       COMMON/B14/X1


       N = 127
       N1 = N + 1
       N3 = N - 1
       N2 = 8128


c  Read the matrix B which contains the 127 conjugates which
c  form the normal basis matrix
       READ (5, *) ((B(I,J), J = 1,N), I = 1,N)
c  Read the inverse of matrix B, 1NV.
       READ (5, *) ((1NV(I,J), J = 1,N), I = 1,N)


c  Form the composite matrix by multiplying the 127 conjugates
c  arranged in a column vector with those arranged in a row
c  vector.  Then reduce every element of this composite matrix
c  modulo X * * 127 + X + 1.

           II = 1
           IP = 1
514        I = IP
```

A-129

```
               DO   400 J = 1, N
               U (J) = B (I,J)
               TEMP(J) = B(I,J)
400            CONTINUE
               I = I + 1
               DO   401 J = 1,N
          -    S (II,J) = B (I,J)
401            CONTINUE
512            DO 402 J = 1, N
               V(J) = B (I,J)
402            CONTINUE
               DO 202 IA = 1,N
               Z (IA) = 0
202            CONTINUE
               DO   220 IA = 1,N
               IF (V(IA).EQ.0)  GO TO 210
               DO 204 IB = 1, N
               IZ = Z (IB) + U(IB)
               Z (IB) = MOD (IZ,2)
204            CONTINUE
210            T1 = U(127)
               T2 = U(1)
               T3 = T1 + T2
               T = MOD (T3,2)
               DO 230 IC = 1, 125
               ID = 128 - IC
               IE = ID - 1
               U (ID) = U(IE)
230            CONTINUE
               U(2) = T
               U(1) = T1
220            CONTINUE
               DO 221 JJ = 1,N
               U(JJ) = TEMP(JJ)
221            CONTINUE
               II = II + 1
               DO 403 J = 1,N
               S(II,J) = Z(J)
403            CONTINUE
```

A- 130

```
                  I = I + 1
                  IF (I.EQ.N1) GO TO 511
                  GO TO 512
511        .      IP = IP + 1
                  IF (IP.EQ.N) GO TO 513
                  II = II + 1
                  GO TO 514
513               II = II + 1
                  DO 404  J = 1, N
                  S(II,J) = B(1,J)
404               CONTINUE

C  Convert every element of the composite matrix S from
C  canonical basis representation to normal basis
C  representation using .INV matrix.

                  DO 405 I = 1, N2
                  DO 406 J = 1,N
                  U(J) = S (I,J)
406               CONTINUE
                  DO 407 II = 1,N
                  IT = 0
                  DO 408 J = 1,N
                  IT = IT + U(J) * lNV(J,II)
                  IT = MOD (IT,2)
408               CONTINUE
                  V(II) = IT
407               CONTINUE
                  DO 409 J = 1,N
                  S(I,J) = V(J)
409               CONTINUE
405        .      CONTINUE

C  Form the M-Matrix from the composite matrix by choosing              ,
C  the coefficient of the conjugate v.

                  I = 1
                  J = 1
                  II = 1
612               X(I,J) = S(II,1)
                  X1(I,J) = S(II,2)
                  I = I + 1
                         A-131
```

```
              II = II + 1
              IF (I.EQ.N1) GO TO 611
              GO TO 612
611           J = J + 1
              I = J
              IF (J.EQ.N1) GO TO 613
              GO TO 612
613           DO 410 I = 1,N
              DO 410 J = 1,N
              X (I,J) = X(J,I)
              X1 (I,J) = X1 (J,I)
410           CONTINUE

              WRITE (6,316)
316           FORMAT (1X, 'THE M-MATRIX IS')
              DO 6 IKK = 1, N
              WRITE (6,315) (X(IKK,JJJ), JJJ = 1,N
315           FORMAT (1X, 11 (I4, 2X))
6             CONTINUE

c  Check whether the M-matrix is correct by rotating the
c  rows one position downward and then shifting the rows
c  one position right and then comparing this M-matrix with
c  the M-matrix formed by choosing the coefficient of
c  conjugate v * * 2.
              DO 411  J = 1, N
              TEMP (J) = X(N,J)
411           CONTINUE
              DO 412 I = 1, N3
              I1 = N1 - 1
              II1 = I1 - 1
              DO 412 J = 1, N
              X(I1,J) = X(II1,J)
412           CONTINUE
              DO 413 J = 1,N
              X(1,J) = TEMP(J)
413           CONTINUE
              DO 414 I = 1,N
              TEMP (I) = X(I,N)
```

A-132

Page 480

```
414         CONTINUE
            DO 415 J = 1, N3
            J1 = 128 -J
            JJ1 = J1 - 1
            DO 415 I = 1, N
            X(I,J1) = X(I,JJ1)
415         CONTINUE
            DO 416 I = 1, N
            X (I,1) = TEMP (I)
416         CONTINUE
            DO 417 I = 1, N
            DO 417 J = 1, N
            IF (X(I,J).EQ.X1(I,J)) GO TO 614
            GO TO 615
614         FLAG  = 1
417         CONTINUE
            WRITE (6,310) FLAG
310         FORMAT (1X, 'MATCHING OK; M-MATRIX CORRECT', I5)
            STOP
615         WRITE (6,311) I,J
311         FORMAT (1X, 'NON MATCHING; M-MATRIX WRONG',
            I5,2X,I5)
            STOP
            END
```

A-133

<u>Appendix 22</u>

<u>Multiplier Implementation using T-Matrix Approach (T-MATRIX.F77)</u>

```
C   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C   This program calculates the number of EX-OR gates required to
C   implement the T-matrix.  It reads the M matrix as input and
C   forms the T-matrix.  Then it determines the number of 'l's in
C   each row of the T-matrix, the number of EX-OR gates required
C   for each row and the total number of EX-OR gates required for
C   the whole T-matrix.
C   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


            INTEGER B(127,127), COUNT, COUNT1, TOTAL(127), SUM(127),
        +           NUM, RCOUNT, Z, REM , U(127,127)
            REAL    ANS

            N = 127

C   Read the M matrix
            READ (5,*) ((B(I,J), J = 1, N). I = 1, N)

C   Form the U-matrix from the M-matrix

            DO 10 II = 1, N
            DO 10 JJ = 1, N
10          U (II,JJ) = 0
            DO 2 I = 1, N
            IF (I.EQ.N)  GO TO 3
            DO 4 J = I + 1, N
            U (I,J) = B (I,J)
4           CONTINUE
3           IP = 1
2           CONTINUE

C   Count the number of 'l's in each row and store it in
C   TOTAL array.
            DO 500 I = 1, N
            TOTAL (I) = 0
            DO 401 J = 1, N
            IF (U(I,J).EQ.1) GO TO 402
            GO TO 403
402         TOTAL (I) = TOTAL (I) + 1
```

A-134

```
403        IP = 1
401        CONTINUE
500        CONTINUE

C    Count the number of EX-OR gates required, NUM, for each row
C    and store it in SUM array.

           DO 400 I = 1, N
           NUM = 0
           RCOUNT = 0
           COUNT1 = TOTAL (I)
           IF (COUNT1.EQ.1) GO TO 600
           IF (COUNT1.EQ.0) GO TO 610
           GO TO 201
600        NUM = 1
           GO TO 601
610        NUM = 0
           GO TO 601
201        ANS = COUNT1/2.0
           COUNT = ABS (ANS)
           REM = COUNT1-COUNT *2.0
           RCOUNT = RCOUNT + REM .
           NUM = NUM + COUNT
           IF (COUNT.EQ.1) GO TO 200
           COUNT1 = COUNT
           GO TO 201
200        IF (RCOUNT.EQ.1) GO TO 202
           IF (RCOUNT.EQ.0) GO TO 202
           IF (RCOUNT. GT. 1) GO TO 203
           STOP
203        COUNT1 = RCOUNT + 1 .
           RCOUNT = 0
           GO TO 201
202        NUM = NUM + RCOUNT
601        SUM(I) = NUM
400        CONTINUE

C    Write the number of EX-OR gates required for each row

           WRITE (6,460) (SUM(JJ), JJ = 1, N)
460        FORMAT (1X, 9(I6, 2X)
```

A- 135

```
C  Calculate the total number of EX-OR gates for the whole
C  M matrix and print the result.
               Z = 0
               DO 450 II = 1, N
               Z = Z + SUM (II)
450            CONTINUE
               WRITE (6,461)Z
461            FORMAT (1X, I10)
               STOP
               END
```

A-136

<u>Appendix 23</u>

<u>Exclusive-or Gate Count Program (EXORNO.F77)</u>

```
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
c  This program counts the number of EX-OR gates required to
c  implement the M matrix without any optimisation.  It reads
c  the M matrix as input and outputs the total number of '1's
c  in each row of the M matrix, the number of EX-OR gates
c  required for each row and the total number of EX-OR gates
c  for the whole M matrix.
c  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

           INTEGER B(127,127), COUNT, COUNT1, TOTAL(127),
       +           SUM(127), NUM, RCOUNT, Z, REM
           REAL    ANS


           N = 127

c  Read the M matrix
           READ (5, *) ((B(I,J), J = 1, N), I = 1, N)

c  Count the number of '1's in each row and store it in TOTAL
c  array.
           DO 500 I = 1, N
           TOTAL (I) = O
           DO 401  J = 1, N
           IF (B(I,J).EQ.1) GO TO 402
           GO TO 403
402        TOTAL (I) = TOTAL (I) + 1
403        IP = 1
401        CONTINUE
500        CONTINUE

c  Count the number of EX-OR gates required, NUM, for each
c  row and store it in SUM array.

           DO 400 I = 1, N
           NUM = O
           RCOUNT = O
           COUNT1 = TOTAL (I)
           IF (COUNT1.EQ.1) GO TO 600
           GO TO 201
```

A-137

Page 485

```
600          NUM = 1
             GO TO 601
201          ANS = COUNT1/2.0
             COUNT = ABS (ANS)
             REM = COUNT1-COUNT * 2.0
             RCOUNT = RCOUNT + REM
             NUM = NUM + COUNT
             IF (COUNT.EQ.1) GO TO 200
             COUNT1 = COUNT
             GO TO 201
200          IF (RCOUNT.EQ.1) GO TO 202
             IF (RCOUNT.EQ.0) GO TO 202
             IF (RCOUNT.GT.1) GO TO 203
             STOP
203          COUNT1 = RCOUNT + 1        -
             RCOUNT = 0
             GO TO 201
202          NUM = NUM + RCOUNT         .
601          SUM (I) = NUM
400          CONTINUE

c   Write the number of EX-OR gates required for each row

             WRITE (6,460) (SUM(JJ), JJ = 1, N)
460          FORMAT (1X, 9 (I6, 2X))

c   Calculate the total number of EX-OR gates for the whole
c   M matrix and print the result.

             Z = 0
             DO 450 II = 1, N
             Z = Z + SUM (II)
450          CONTINUE
             WRITE (6,461) Z
461          FORMAT (1X, I10)
             STOP
             END
```

A-138

<u>Appendix 24</u>

<u>Inverse Matrix over GF (2) Program   (INVMOD. F77)</u>

```
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C  This program calculates the inverse of a given NxN matrix
C  over GF(2)
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *


           INTEGER  B(127,254), TEMP (254), XM, X, XM2, XM3,
                    XM4, XM5, Y, XM6, XM7, XM8, NX, NY, LZ

           N = 127
           READ (1, *)  ((B(I,J), J = 1, N). I = 1, N)

           I = 1
           NX = N + 1
           NY = 2 * N

C  Form the identity matrix in columns NX to NY and rows
C  1 to N

           DO 81 IK = 1, N
           DO 81 IJ = NX, NY
           B (IK, IJ) = 0
81         CONTINUE
           DO 80 J = NX, NY
           B (I,J) = 1
           I = I + 1
80         CONTINUE

C  Compute the inverse matrix by performing elementary row
C  operations and store the inverse in columns NX to NY and
C  rows 1 to N

           L = 1
           K = 2
70         L1 = L
           XM = B(L,L)
           XM = ABS (XM)
           XM = MOD (XM,2)
           IF (XM.EQ.0) GO TO 215
```

A-139

```
214          DO 95 J = L, NY
             XM2 = B(L,J)
             XM2 = ABS (XM2)
             XM2 = MOD (XM2,2)
             XM2 = XM2/XM
95           CONTINUE
             DO 141 I = K,N
             X = B (I,L)
             X = ABS (X)
             X = MOD (X,2)
             IF (X.EQ.0) GO TO 141
             DO 140 J = L, NY
             XM3 = B(I,J)
             XM4 = B(L,J)
             XM3 = ABS (XM3)
             XM4 = ABS (XM4)
             XM3 = MOD (XM3,2)
             XM4 = MOD (XM4,2)
             B(I,J) = XM3 - XM4 *  X
             IF (B(I,J).EQ. -1) B(I,J) =1
140          CONTINUE
141          CONTINUE
             L = L + 1
             K = K + 1
             IF (L-N) 70, 190, 190
             XM = B(L,L)
             XM = ABS (XM)
             XM = MOD (XM,2)
             DO 195 J = L, NY
             XM6 = B(L,J)
             XM6 = ABS (XM6)
             XM6 = MOD (XM6,2)
             XM6 = XM6/XM
195          CONTINUE
             L = N
235          LZ = L -1
             DO 291  K = 1, LZ
             I = L - K
             Y = B (I,L)
             Y = ABS (Y)
```

A-140

```
                    Y = MOD (Y,2)
                    IF (Y.EQ.0) GO TO 291
                    DO 290 J = L, NY
                    XM7 = B(I,J)
                    XM7 = ABS (XM7)
                    XM7 = MOD (XM7,2)
                    XM8 = B(L,J)
                    XM8 = ABS (XM8,2)
                    B(I,J) = XM7 - XM8 * Y
                    IF (B(I,J).EQ.- 1) B(I,J) = 1
      290           CONTINUE
      291           CONTINUE
                    L = L - 1
                    IF (L-1) 320, 320, 235
      320           WRITE (1,330)
      330           FORMAT (1X, 'THE INVERSE MATRIX IS')
                    WRITE (1,331) (B(II,J), J = NX, NY)
      331           FORMAT (1X, 11 (I5, 2X))
                    CONTINUE
                    STOP
      215           L1 = L1 + 1
                    N1 = N + 1
                    IF (L1.EQ.N1) GO TO 212
                    XM = B(L1,L)
                    XM = ABS (XM)
                    XM = MOD (XM,2)
                    IF (XM.EQ.0) GO TO 215
                    DO 96 J = L, NY
                    TEMP (J) = B(L,J)
                    B(L,J) = B(L1,J)
                    B (L1,J) = TEMP (J)
      96            CONTINUE
                    GO TO 214

      212           WRITE (1,813)
      813           FORMAT (1X, 'THE MATRIX IS SINGULAR')
                    STOP
                    END
```

A- 141

<u>Appendix 25</u>.

<u>Matrix based Public Key Distribution Program (PKDEXT. F77)</u>

```
C  * * * * * * * * * * * * * * * * * * * *
C  This program is used to design a matrix based Public
C  key distribution system based on exponentiation over GF(p)
C  where p is a prime.
C  This program essentially consists of a number of calls to
C  different subroutines (programs) given in the previous
C  appendices and hence only the main steps are given here.
C  * * * * * * * * * * * * * * * * * * * *


          INTEGER B1(2,2), B2(3,3), X(5,5), XINV(5,5),
     +            M(5,5),  A(5,5)

          p = 5

C  Input the companion matrix B1 and check the order of B1
C  using MATEXP. F77.

          READ (1,*)  ((B1(I,J), J = 1,2), I = 1,2)
          CALL MATEXP. F77

C  Input the companion matrix B2 and check the order of B2
C  using MATEXP. F77

          READ (1,*) ((B2(I,J), J = 1,3), I = 1,3)
          CALL MATEXP. F77

C  Select an arbitary X matrix and find the determinant
C  of X to see if it is non-singular modulo p using DETMOD.F77

          READ (1, *)  ((X(I,J), J = 1,5), I = 1,5)
          CALL DETMOD. F77

C  Find the inverse of X using INVMOD.F77

          CALL INVMOD. F77

C  Multiply the X matrix with the composite B1 B2 matrix
C  and the XINV matrix using MULPKD. F77 which is given as
C  a subroutine in MATEXP. F77.

          CALL MULPKD  (X, B1, B2, XINV, A)

C  The A matrix is therefore the base matrix in the public
C  key distribution system.  The system is then implemented
C  using MATEXP. F77.
```

A-142

Dickson Polynomials based PK_System Program (DPOLY. F77)

```
C   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C   This program evaluates the coefficients of Dickson
C   polynomials for GN(x) for given N over Z/mZ.  The value of N
C   specified in this program is equal to the degree of
C   polynomial plus 1 and the modulus m is equal to the product
C   of two primes p and q.  The program then evaluates the
C   function GN(x) for all x, 1 to m-1.
C   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

          INTEGER * 4  G0(462), G1(462), G2(462), H(462)


          N = 6

C   Use the recursive function to evaluate GN(x).
          N3 = N + 1
          N1 = N - 1
          N2 = N1 -1
          IFLAG = 0
          G0(1) = 2
          G0(2) = 0
          G1(1) = 0
          G1 (2) = 1
          DO 1  II = 3, N
          G0 (II) = 0
          G1 (II) = 0
          G2 (II) = 0
1         CONTINUE
          DO 2  I = 1, N2
          DO 3  J = 1, N
3         H(J) = G1 (J)
          DO 4 J = 1, N1
          K = N3 - J
          L = K - 1
          G1 (K) = G1 (L)
4         CONTINUE
          G1(1) = 0
          DO 5 J = 1, N
```

A-143

```
                    G2(J) = G1(J) - G0(J)
                    IM = G2(J)
                    IF (IM.LT.0) IFLAG = 1
                    IN = ABS (IM)
                    IT = MOD (IN, 35)
                    IF (IFLAG.EQ.1) GO TO 12
                    G2 (J) = IT
                    GO TO 13
        12          G2(J) = 35 - IT
        13          IFLAG = 0
                    G1(J) = G2(J)
                    G0(J) = H(J)
        5           CONTINUE
        2           CONTINUE

    C   Evaluate the function for IX equal to 1 to 34.

                    IX = 1
        121         IZ = 0
                    DO 16 IJ = 1, N
                    JJ = G2 (I5)
                    IF (JJ.EQ.0) GO TO 111
                    IJ1  =  IJ -1
                    IY = 1
                    IF (IJ1.EQ.0) GO TO 112
                    DO 17 II = 1, IJ1
                    IY = IX *  IY
                    IY = MOD (IY, 35)
        17          CONTINUE
        112         IY = IY *  JJ
                    IY = MOD (IY, 35)
                    IZ = IZ + IY
                    IZ = MOD (IZ, 35)
        111         IKK = 1
        16          CONTINUE
```

A-144

```
          WRITE (1, 18) IX, IZ
18        FORMAT (1X, 'IX =', I16, 2X, 'IZ = ', I16)
          IX = IX + 1
          IF (IX.EQ.35)  GO TO 120
          GO TO 121
120       STOP
          END
```

Papers Published/Submitted

Secure communications between microcomputer systems

P W Sanders and V Varadharajan describe a security interface unit that uses the DES to encrypt sensitive data

A data security communications interface unit has been developed to allow data transfer between Apple terminals in either plain or encrypted format under user control. The unit employs the Data Encryption Standard algorithm and has a degree of sophistication sufficient to meet most user needs. The unit uses the 6502 microprocessor to control encryption, decryption and communications. In addition to the transfer of encrypted data, the interface also provides a facility for storing encrypted program and data files locally in the Apple disc system. Further, the encryption system has been designed to allow storage and retrieval of completely encrypted or partly encrypted frames of information on the Prestel database. The interface has been tested extensively using several DES modes of operation.

Keywords: data communications, security, encryption, Data Encryption Standard, Prestel

Data security has never been more significant than it is today, owing to the expanding role of distributed computation, distributed databases and telecommunications applications such as electronic mail and electronic funds transfer. Converging computer and communications technologies have resulted in a dramatic increase in the volume and speed of information collection and distribution. Greater information transfer in turn implies a greater risk of exposure of sensitive or confidential information to unauthorized users, owing to the ready availability of inexpensive miniature intercepting devices. These have resulted in

Plymouth Polytechnic, Drake Circus, Plymouth, Devon PL4 8AA, UK

an increased interest in computer data security, not only in the military and political area but also in the field of commerce. This has motivated research, particularly in the art of cryptography, which forms the central technique of communications security.

This article describes the design of an encryption interface unit employing the Data Encryption Standard[1] adopted by the US National Bureau of Standards. It has been designed primarily for the Apple microcomputer, which is commonly used as an intelligent terminal in communications networks. The purpose of this interface is threefold.

- It allows secure data communications in a point-to-point configuration.
- It provides a local storage facility in the Apple disc system for the encrypted program and datafiles.
- It allows storage and retrieval of encrypted or partly encrypted information on the Prestel viewdata system.

SYSTEM DESIGN

The encryption system configuration for point-to-point data communications is shown in Figure 1. The communications link is half duplex, allowing transmission in either direction, but not simultaneously, with datarates ranging from 50–1 200 bit/s, suitable for transmission over standard telephone lines. The encryption unit is incorporated as an in-built feature of the terminal, this being superior to a stand-alone arrangement for reasons of access control, since the former technique greatly reduces the chances of

Page 495
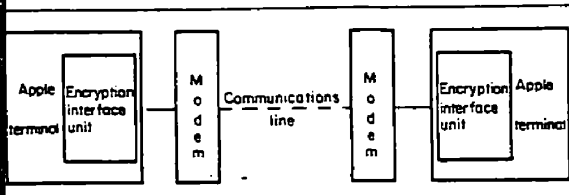
Figure 1. Point-to-point system configuration

detection between the terminal and encryption unit, where the text is in plain form. As shown in Figure 1, the plain text from the keyboard is encrypted by the interface and the cipher is transmitted over the standard telephone line via the modem.

A schematic diagram of the encryption interface showing different functional blocks is shown in Figure 2. The unit is designed around the 6502 microprocessor with associated random access memory (RAM) and read only memory (ROM) of the Apple system. Encryption and decryption of data is carried out by a large-scale integration device using the DES algorithm. Data communications is handled by a dual enhanced communications controller element (DEUCE). This controller contains two independent asynchronous receiver/transmitter channels and two independent generators providing possible transmission rates up to 19 200 bit/s. The encryption unit con-

tains the necessary EIA RS232-compatible circuits for interfacing it to the modem, and a PROM is provided to hold the necessary programs for the operation of the system.

The interface can operate in three different formats; plain, encrypted or a mixture of both. The users at both ends of the communications link initially choose a datarate from the range of 50–1 200 bit/s and then select one of the three modes. In the case of the plain mode, the data transfer between the users will be in plain form. If either of the other two modes is chosen, then the secret DES key has to be entered. It is assumed that the parties concerned have pre-knowledge of the key, a necessity for proper communication of the encrypted data. Any eight alphanumerical characters of the keyboard can be used to form the 64 bit key required for the DES algorithm. It is essential that the key should be chosen randomly (for example, by some form of random number generator), so that it may not be easily guessed by any cryptanalyst. The key is displayed on the VDU to verify the correctness of entry but the display is erased immediately after the last character is input, to avoid detection by others during the course of communication. Many users would probably use some easy-to-remember phrase or number combination for developing the key, and in such cases the phrase can be converted to a form suitable for DES using a good
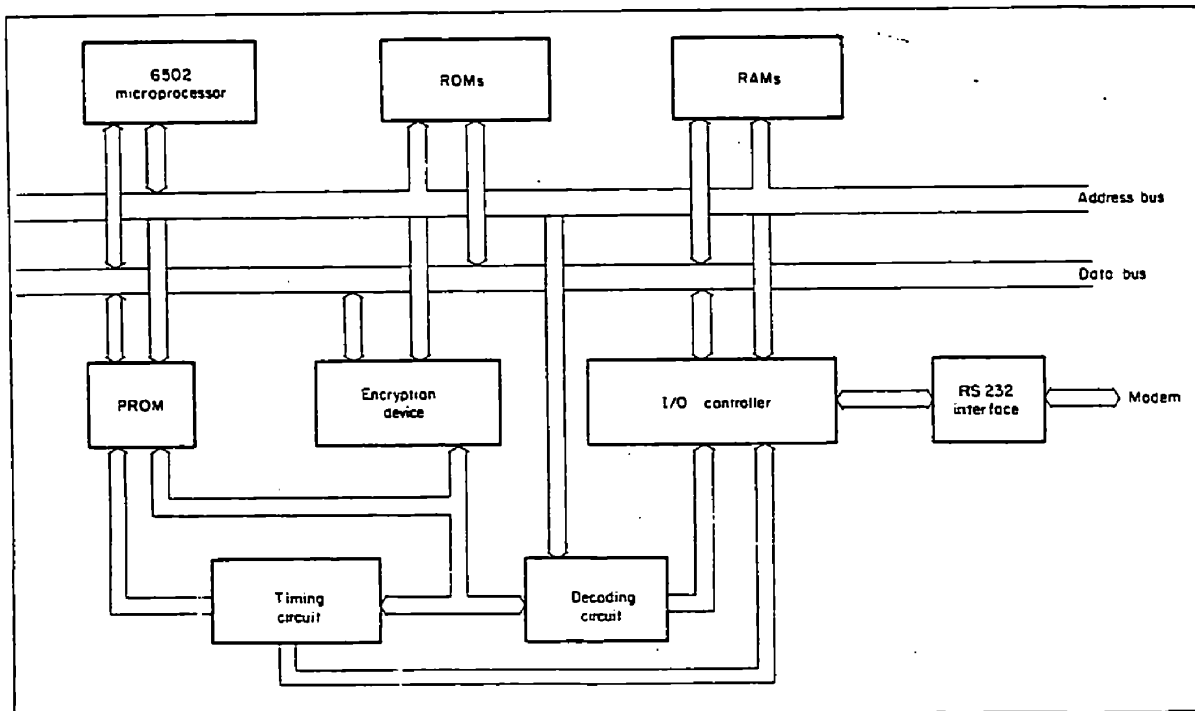
Figure 2. Functional block diagram

'hashing' function. Further minor software changes can be made to provide for multiple DES key encryption, to achieve higher levels of security or to modify facilities.

When the terminals are set ready to accept data, the user who presses a key first gains control of the line, and thus will be able to transmit data. A maximum of 256 bytes of message can be input to the system from the keyboard at one time. The message is encrypted using one of the several DES modes mentioned below, and then transmitted over the line. If the key is to be altered for the next message, the system must be reset. Further, data stored in memory can be encrypted and transmitted in a similar fashion.

In the case of a mixture of plain and encrypted information, the input from the terminal will always be in a plain form until a change to the encrypted mode is initiated by typing a special character (CNTRL-A). The unit is automatically returned to the plain format after 8 characters (64 bit). Alternatively, another special character (CNTRL-B) can be used to return to the plain format, but then one must ensure that the CNTRL-B character does not occur within the enciphered data to ensure unambiguous decryption at the receiving end. This can be achieved by using multiple CNTRL-B characters to indicate the end of encrypted text. The larger the number of such characters, the smaller the probability that they occur in the enciphered data, and the smaller the ambiguity in decryption, but this unfortunately increases the number of redundant characters in the transmitted data.

One of the requirements of this mixture format is that in a multiuser network, the plain information must only be deciphered correctly by the user with the right key. There may be cases where the encryption algorithm transforms a noncontrol character to a control character and vice versa. As a control character is not displayed by the Apple, this results in a line of text with parts of it encrypted at the transmitting end not producing a line of text at the receiving end when using the wrong key. As a certain amount of delay is required

for special control characters such as CNTRL-G (Bell) or CNTRL-J (line feed), this can cause an over-run error at the receiving end, even in the reception of plain text when using the wrong key. Software has been written to overcome such situations.

Several modes of the Data Encryption Standard have been investigated[2] using the developed encryption system, namely the electronic code book (ECB), cipher block changing (CBC) and stream cipher feedback (CFB).

The ECB allowed a transformation of a 64 bit plain text word into a cipher text word of the same length, as shown in Figure 3. In this mode, the information is encrypted in integral multiples of 64 bits. The last block is padded with random bits prior to encryption to build it up to 64 bits. During decryption, the padding is taken into account so that the random bits are discarded after decryption. A critical weakness of this mode (Figure 4)* is that a given plain text always produces the same cipher text under the same key. Thus, the compromise of the plain text underlying any cipher text block results in the compromise of all repetitions of this same text for the remainder of the cryptographic period. This problem is often referred to as the code-book analysis problem.

In the CBC mode, a plain text block is exclusive-ored with the previous cipher text block, prior to encryption, as shown in Figure 5. In the first encryption cycle, the plain text block is exclusive-ored with a block of pseudorandom bits called the initialization vector (IV). Mathematically, the scheme can be expressed as follows:

if the ith plain text and cipher text blocks are x(i) and y(i)
and the initialization and feedback vectors are U(1), U(2) . . . U(n − 1)
where U(1) = z = initialization vector
   U(i) = y(i − 1)

then the encipherment and decipherment become

$$y(i) = f_k\{x(i) + y(i - 1)\} \quad i \geq 1$$
$$x(i) = f_k^{-1}[y(i) + y(i - 1)] \quad i \geq 1$$

where x(0) = y(0) = z, f defines the cryptographic function and the subscript designates the particular key.

The security of this mode depends, amongst other factors, upon the management of the initialization vector. CBC reproduces the same cipher text wherever the same plain text is encrypted under a fixed key and initialization vector. In the ECB mode, the cipher text repetition is found to occur at block level, whereas in the CBC mode the cipher text repetition is at message level. The CBC mode not only reduces the code-book analysis problem but also provides a limited error extension characteristic which is valuable in protecting
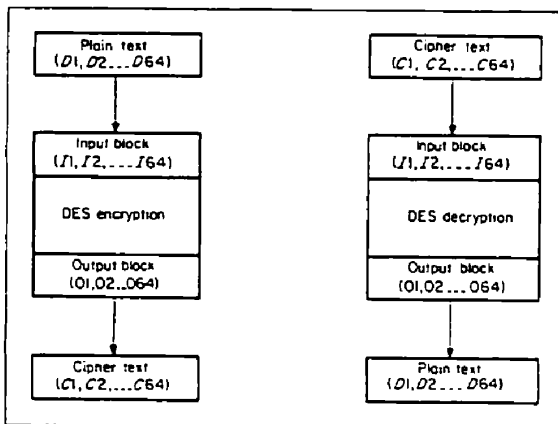
Figure 3. Electronic code-book mode

---

*The ASCII character set has been extended using Hershey[3] characters to indicate all 256 possible codes produced by encipherment.
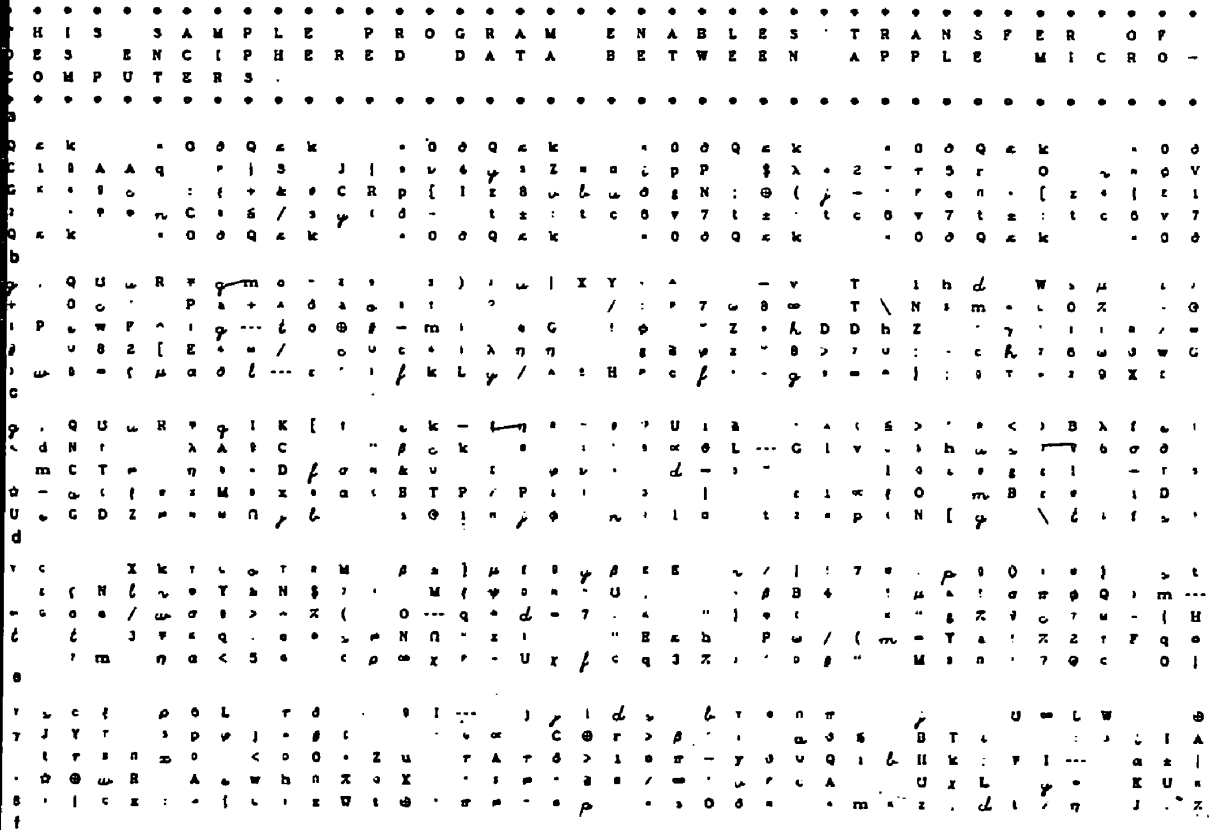
Figure 4. Examples of encrypted text

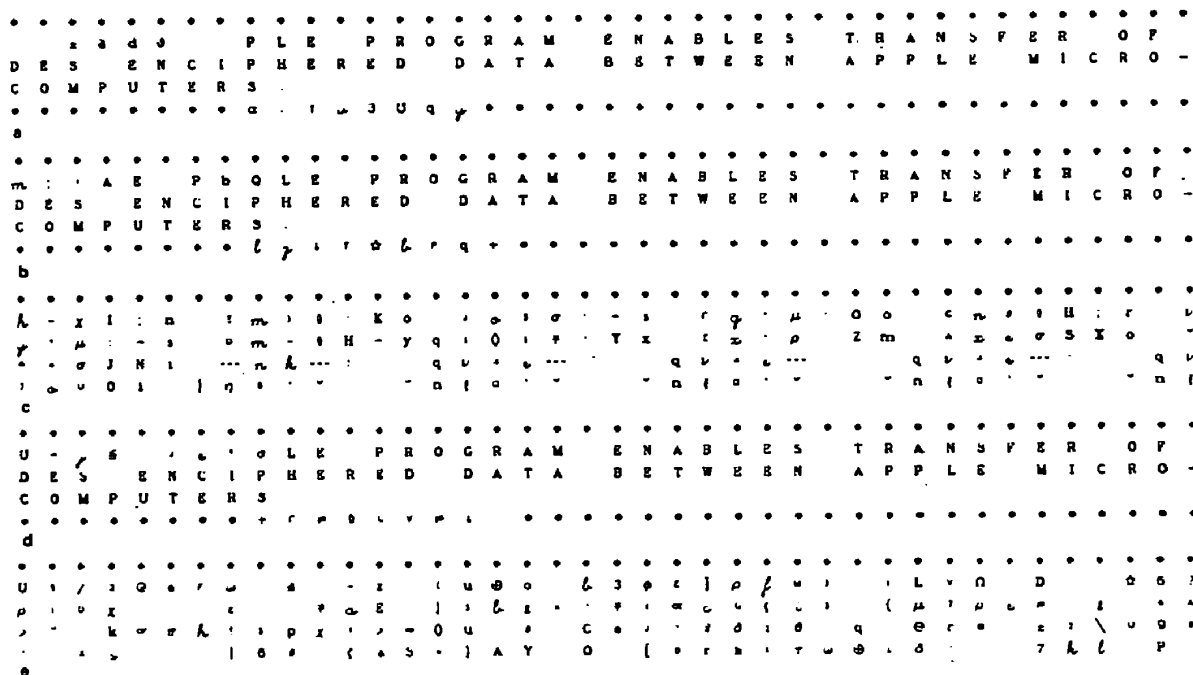a Plaintext; b electronic code book; c cipher block chaining; d cipher block chaining with plaintext feedback; e stream cipher feedback; f stream cipher feedback with plaintext feedback

against fradulent data alteration. One or more errors in a single cipher text block affects the decryption of two blocks, namely the block in which the error occurs and the succeeding block, but they synchronize thereafter (see Figure 6). This self-synchronizing scheme is particularly suitable when noise is present on the communications lines.

To provide for error propagation throughout the message, a slight variation of the CBC technique is implemented. Here the input plain text is modified by making it a function of both the previous plain text and the cipher text blocks prior to encryption, as shown in Figure 7. That is

$$y(i) = f_k[x(i) + U(i)] \quad i \geq 1$$
$$x(i) = f_k^{-1}[y(i) + U(i)] \quad i \geq 1$$

where

$$U(i) = \begin{cases} z & i = 1 \\ x \ (i-1) + y(i-1) & i > 1 \end{cases}$$

This scheme represents a general block cipher, and if any portion of the cipher becomes garbled, the decryption of all subsequent blocks until the end of the message is garbled (Figure 6). This technique is used for the purpose of message authentication.

While the CBC technique overcomes the codebook analysis problem, the problem of padding of the last block still remains. A stream cipher mode is therefore implemented to cope with this problem. With this technique, the DES is used as a random number generator. The output of the DES is exclusive-ored with plain text to form the cipher text. The decryption process operates the same way as encryption, with the exact pseudorandom stream of encrypting bits being generated. An 8 bit CFB implementation is shown in Figure 8, although any number of the 64 bits can be used. With this stream cipher technique, the plain text is encrypted character by character and not in blocks. An error in the cipher text character affects not only the decryption of the garbled cipher text but also the eight succeeding characters until the bit error is shifted out

248
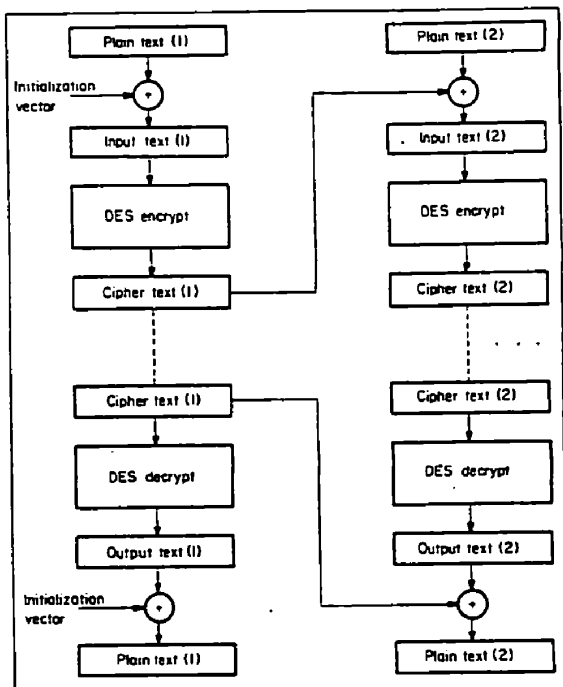
Figure 5. Cipher block chaining mode

of the CFB input block (Figure 6). Again this is a self-synchronizing scheme.

To provide for error propagation, a slight variation of this technique has also been implemented by providing a feedback from the initialization vector, in addition to the feedback from the cipher text as shown in Figure 9. This scheme represents a general stream cipher and corruption of a single bit of cipher text will cause each subsequent bit of recovered plain text to be in error (Figure 6).

The encryption system can also be used to provide storage of encrypted information locally in the Apple disc system. In this case, instead of transmitting the encrypted information to another user over a communications line, it is stored in memory.

Theoretically, any of the DES modes discussed above could be used, but when a file is encrypted, recovery from an error must be affected with cipher text alone. If a ciphering procedure with error propagation is used for file security, subsequent inability to read a fraction of the cipher text because of damage either to the physical medium or to the recorded bits may prevent all the following cipher text from being deciphered. Therefore a self-synchronizing approach is desirable for file encryption. This therefore leaves the two chaining modes CBC and CFB, either of which could be employed. If cipher feedback on 8 bit characters is used, then the maximum speed will be one eighth of the block mode speed, and hence, the

Figure 6. Error characteristics

a Electronic code book; b cipher block chaining; c cipher block chaining with plaintext feedback; d stream cipher feedback; e stream cipher feedback with plaintext feedback
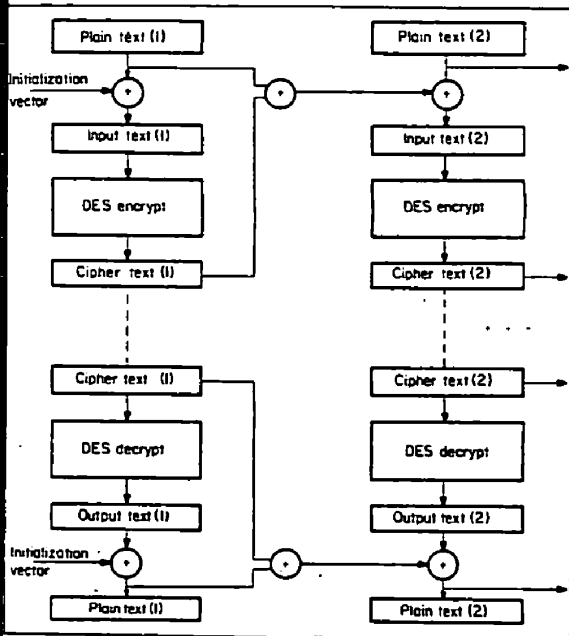
Figure 7. Cipher block chaining with plain text feedback
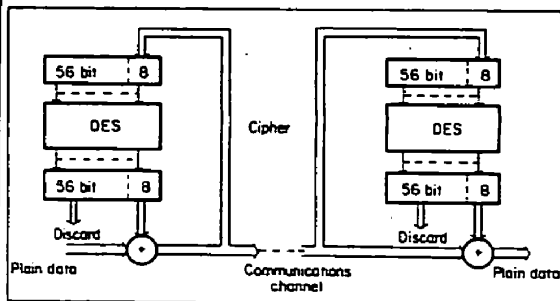
Figure 8. Cipher feedback mode

throughput is correspondingly lower. On the other hand, with CBC, the problem of padding exists, since it is a block cipher.

The program performs encryption and decryption of either an Applesoft BASIC file or Integer BASIC file, or an input datafile from the keyboard. The encrypted file is automatically stored on the disc under the file name provided by the user. The encrypted file can be loaded back from the disc at a later time and decrypted to give the original file, provided the same key has been used.

SECURITY IN PRESTEL VIEWDATA SYSTEM

The encryption system has been incorporated into the
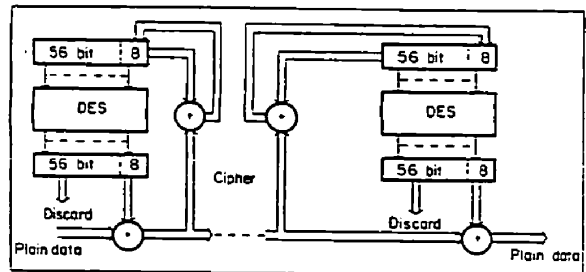
Figure 9. Stream cipher with cipher text and plain text feedback

Prestel network, allowing transfer of encrypted as well as plain information between an Apple microcomputer and a Prestel viewdata computer. This allows secure storage and retrieval of sensitive information, such as bank statements or legal documents.

The basic unit of information on Prestel is a frame which consists of up to a maximum of 960 characters. One or more frames are linked together to form a page, and these pages of information form the Prestel database. A natural choice for encryption would therefore be a complete frame, but there may be instances where the encipherment of a section of a frame, or even a few characters is required. The header information at the start of the frame can be used to indicate that encipherment has been used on that frame.

As in the case of file security, the two possible modes of DES suitable for this application are cipher block chaining (CBC) and stream cipher feedback (CFB). Since it is required to encrypt parts of a frame down to individual characters, only the cipher feedback mode, which allows character-by-character encryption, can be used. Further, if the CBC mode is used, when parts of a frame are encrypted, it is likely to require padding for each encrypted part. This in turn will result in cryptogram extension, and will pose a problem when storing the enciphered frame on the Prestel database, since each frame is limited to a maximum of 960 characters.

The encrypted information passes through the Prestel control unit, which rejects any of the control characters present in the cipher text. There is therefore a need to prevent the occurrence of these control characters in the encrypted information. That is, the data format is restricted to satisfy the Prestel computer protocols. This can be achieved by using 6 bit cipher feedback without altering the existing Prestel software. The 64 character codes chosen for encipherment are 0–9, A–Z, a–z, space and period. All other codes are transparent and hence bypass encryption. This is, of course, a weakness, but work is currently being carried out to expand this to a full character and graphics set. The output codes are reformed into the same range as the input, thus preserving the one-to-one relationship between transmission and reception. As we were

mainly interested in enciphering alphanumerical characters present in the frame, the above set of input codes was found to be adequate for the purpose.

The system is connected to the public switched telephone network via a modem in the usual manner. The Prestel number is dialled, and when the Prestel computer responds by sending a continuous data tone of high frequency, the data switch is pressed for the modem to take control of the line. The terminal becomes ready for data transfer.

```
. .  HERE IS A GAME THAT CAN BE PROGRAMMED
FOR PLAY ON A DIGITAL COMPUTER . . .
    A polyomino is a figure formed by joini
ng unit squares along their edges. Pentom
inoes are 5 square polyominoes and it is
possible to construct 12 different pentominoes.
A pentomino game is played by ar
ranging the 12 pentominoes into various
size rectangular boxes . . . 3 by 20 or 4 b
y 15 or 5 by 12 or 6 by 10. Computers hav
e been used to generate many solutions. A
computer program produced two solutions
for 3 by 20 configuration and 2339 for
the most popular size 6 by 10 rectangula
r configuration.
```

Figure 10. Prestel page in plain form

After entering the secret DES key in the normal fashion, the user has the choice as to when to set the interface into the decryption mode. This enables him to decipher only those pages which have some enciphered data and to read the other Prestel pages in plain form. Only the user with the right key and correct initialization vector will be able to obtain the original plain text. This software implementation allows changes in the initialization vector during communication, whereas to change the key, the system needs to be reset and restarted. This allows every user to have a single secret key, although he may use any number of different initialization vectors.

In the editing mode, the user is able to enter, amend, copy and delete encrypted as well as plain frames in Prestel. From the user point of view, it is essential that the encryption operations must be as simple as possible. Start (?) and stop (/) markers are used to indicate the beginning and end of enciphered data. The CNTRL-A key is used to set the interface unit to the encryption state. All subsequent characters input are encrypted under the CFB mode. The CNTRL-B key automatically returns the interface to the plain mode, and so allows encryption of even a single byte of data. Examples of completely encrypted and partly encrypted frames are shown in Figures 10, 11 and 12.

A stand-alone unit working on the same principles has also been developed for any computer connected between the RS232 interface output and the corres-

```
Scratchpad        651314b           Op

?OnLih cRwGbZHjxeaJgVw8CLJ BVz bbvDW Ev
   o' y w PPZQsHNaBK txvOp  hV T3BgxOmJ' K
D YnvEISJ YwQXPtrTmoBCLHv okHXebhH GXqq
'KZdvI erePmTqax9 kyKmC Rcfi'P 'J1NaXt
OnbwYg xkTRLsODN  IuCvSzLZvXNZb WNpYPB
X 3dE2g0nPtmjDGtOaUVIG  OdMP OgiYmdqRYA
sdTdriZue jTd QQs  CnxkbuVgziKiokxbKtpdj
juwCC 'G  nH yMIg9HvI dhae RHX'DbQHo 'yP
xx9y 3z 3 UZg amqmV3'EfozyPjUTPQPcQP TR
JkzPtPA AVsoGpoiLplWewo wk9PJ ·JyiDHEz Z
uLrwZu oRPgWzOLDTWr  Rkq9 f YRAXZXSiV
PR CSZCIVUet DtPC c. HnTEyQByjfALbp rqL
O Ip papjyee BGItj Yxo abb TxftGQpWgjLZ
yyaLzfZJvrOLzfSBwXsf ByiX vPvg h bH Art
vbucraZdf'xfvGIx3 HRztdsplgHuaDQ BsIyuK/


? indicates start of encryption.
/ indicates end of encryption.


KEY  3131313131313131
IV   0000000000000000


MODE: 6-bit CFB
```

Figure 11. Encrypted Prestel page

```
Scratchpad         651314 e.          Op

. .  HERE IS A ?9PJI/ THAT CAN 9E PROGRAMMED
FOR ? KWt/ ON A DIGITAL COMPUTER . . .
    A ?gVDy   NEG/ is a figure formed by joini
ng unit squares along their edges. ?1 Yhab
0aZ  / are 5 square ?DtNxLC kIP / and it is
possible to construct 12 different ?jXgl Pa pDzA/
A ?KS   MhiD/ game is played by ar
ranging the 12 ? SmwGYzLZc/ into various
size rectangular boxes . . . 3 by 20 4 b
y 15 or 5 by 12 or 6 by 10. Computers hav
e been used to generate many solutions. A
computer program produced ? jP/ solutions
for ?C KoLgq/ configuration and ?buYv/ for
the most popular size ?bScLSto/ rectangula
r configuration.


? indicates start of encryption.
/ indicates end of encryption.


KEY : 3131313131313131
IV  : 0000000000000000


MODE: 6-bit CFB
```
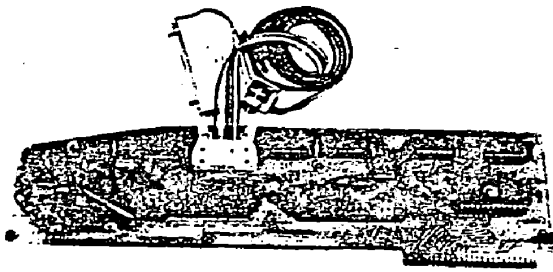
Figure 12. Partly encrypted Prestel page

Figure 13. Communications card providing secure storage and transmission between Apple microcomputers and any other computer

ponding input of the modem. Figure 13 shows the communications board designed for the Apple microcomputer.

REFERENCES

1 *Data encryption standard* FIPS Publ. 46, National Bureau of Standards, USA (January 1977)
2 *DES modes of operation* FIPS Publ. 81, National Bureau of Standards, USA (September 1980)
3 **Buckley, D C** *Hershey characters* Plymouth Polytechnic Computer Centre, Internal Handout No. U5.8-14 (February 1983)

References

1 MARCATILI, E. A. J.: 'Optical subpicosecond gate', Appl. Opt., 1980, 19, pp. 1468–1476
2 HAUS, H. A., KIRSCH, S. T., MATHYSSEK, K., and LEONBERGER, F. J.: 'Picosecond optical sampling', IEEE J. Quantum Electron., 1980, QE-16, pp. 870–874
3 LIU, P. L., and MARCATILI, E. A. J.: 'Computer simulation of the velocity matched gate', Appl. Opt., 1981, 20, pp. 862–866
4 ALFERNESS, R. C., KOROTKY, S. K., and MARCATILI, E. A. J.: 'Velocity matching techniques for integrated-optic traveling-wave switch/modulators', IEEE J. Quantum Electron., 1984, QE-20, (to be published)
5 ALFERNESS, R. C., JOYNER, C. H., BUHL, L. L., and KOROTKY, S. K.: 'High-speed traveling-wave directional coupler switch/modulator for $\lambda = 1.32\ \mu m$', ibid., 1983, QE-19, pp. 1339–1341
6 LEE, T. P., BURRUS, C. A., OGAWA, K., and DENTAI, A. G.: 'Very-high-speed back-illuminated InGaAs/InP PIN punch-through photodiode', Electron. Lett., 1981, 17, pp. 431–432
7 UEHARA, S.: 'Calibration of optical modulator frequency response with application to signal level control', Appl. Opt., 1978, 17, pp. 68–71
8 SUETA, T., and IZUTSU, M.: 'High speed guided-wave optical components'. 3rd Int. Conf. on integ. opt. and opt. Fiber comm., San Francisco, Paper TUM2, 1981

PUBLIC KEY DISTRIBUTION IN MATRIX RINGS

Indexing terms: Codes, Public key systems

An extension of the Diffie–Hellman public key distribution system to matrix rings is described. Using rings of non-singular matrices over Z/pZ and upper triangular matrices with invertible elements along the diagonal over Z/pZ, it is shown that the number of possible secret keys is much greater for a given prime p compared to the original system. An outline of a method to construct the base matrix used in the system is given.

Introduction: Diffie and Hellman[1] first proposed the idea of public key distribution in which two parties exchanging only public information over an insecure channel could establish a secret key for use in a conventional cryptosystem such as the data encryption standard.[2] They proposed a system based on the exponential function $f: x \to a^x$ over GF(p), where p is a very large prime and a is a primitive element in GF(p). The security of this system depends on the difficulty of computing logarithms over GF(p). Pohlig and Hellman[3] investigated an algorithm for computing logarithms over GF(p) and proved it to be efficient when $p - 1$ consists of small prime factors only, but computationally infeasible when $p - 1$ contains at least one large prime factor.

Here we investigate an extension of the exponentiation system to matrix rings.

Extended exponentiation system: The group formed by only the nonsingular matrices of order n, M_n, is considered as the ring of all $n \times n$ matrices over a finite field containing nilpotent elements when $n > 1$.

To form a public key distribution system, it is required to choose an element $A \in M_n(Z/pZ)$, where p is a very large prime such that

$$A^r \equiv I \quad (\text{mod } p)$$

where r is the order of A, the base matrix.

The base matrix A, the prime p and the order r are to be made public. Each user chooses a random number x_i less than r and generates a public matrix C_i, where

$$C_i \equiv A^{x_i} \quad (\text{mod } p)$$

Two users can arrive at the same key in the same way as in the Diffie–Hellman system. For instance, if user 1 wishes to

initiate an interchange of secret information with user 2, he first extracts the public matrix C_2 of user 2. Then he computes

$$C_2^{x_1} \quad (\text{mod } p)$$

User 2 obtains the public key C_1 of user 1 and computes

$$C_1^{x_2} \quad (\text{mod } p)$$

It can be seen that this process yields the common key K_{12}, where

$$K_{12} = K_{21} = C_1^{x_2}(\text{mod } p) = C_2^{x_1}(\text{mod } p)$$

$$= A^{x_1 x_2}(\text{mod } p)$$

This can be used in the connection protocol of a DES based system to establish the session key. With the Diffie–Hellman system the maximum number of secret keys possible is limited to $p - 1$, whereas with this extended system it depends on the order of the base matrix: the larger the value of r, the greater the number of users that the system can support. Again the security of this system is dependent on the difficulty of computing logarithms over GF(p).

Design of base matrix A: The system designer needs to construct a matrix A in $M_n(F_p)$ and determine the order r. One method of construction of A with a given order is outlined below.

Consider an irreducible polynomial $f(x)$ of degree m for which λ is a root ($\lambda \in F_q$):

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1} \qquad a_i \in F_p$$

Regarding F_q as an m-dimensional vector space over F_p with basis $\{1, \lambda, \lambda^2 \ldots \lambda^{m-1}\}$, let T represent the following linear transformation on F_q:

$$T: x \to \lambda x$$

Under T,

$$1 \to \lambda, \lambda \to \lambda^2, \ldots, \lambda^m \to -a_{m-1}\lambda^{m-1} \ldots -a_0$$

Hence the matrix representation of the linear transformation T relative to the basis $\{1, \lambda, \ldots, \lambda^{m-1}\}$ is given by the companion matrix:

$$B = \begin{pmatrix} 0 & 1 & 0 \cdots \cdots 0 \\ 0 & 0 & 1 \cdots \cdots 0 \\ 0 & 0 & 0 \ 1 \cdots 0 \\ \vdots & & \\ -a_{m-1} & -a_{m-2} & \cdots \cdots \cdots -a_0 \end{pmatrix}$$

Linear independence of $I, T, T^2 \ldots T^{m-1}$ implies that $I, B, B^2 \ldots B^{m-1}$ are linearly independent. Since $f(\lambda) = 0$, we have $f(B) = 0$. But $f(x)$ and degree m and so the linear independence implies that $f(x)$ is the minimum function of B.

Hence the order of the matrix B is equal to $p^m - 1$, and

$$B^{p^m-1} \equiv I \quad (\text{mod } p)$$

Thus the system designer can choose irreducible polynomials of degrees $m_1, m_2 \ldots m_s$ over GF(p) and form the composite matrix B as shown below:

$$B = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_s \end{pmatrix}$$

where the order of B_1 is equal to $p^{m_i} - 1$ for $1 < i < m_s$. The order of the matrix B is then given by the expression

$$l_{cm}\{(p^{m_1} - 1), (p^{m_2} - 1), \ldots, (p^{m_s} - 1)\}$$

The matrix A to be used in the public key distribution system can then be obtained by conjugating B with an arbitrary

matrix Y belonging to $M_n(F_p)$:

$$A = YBY^{-1}$$

The order of A is the same as that of B and they are of dimension n, where n is given by

$$n = \sum_{i=1}^{t} m_i$$

The outlined method has been implemented on the prime computer system and an example is given.

On the other hand, if we choose the base matrix A from the ring of upper triangular matrices over Z/pZ, where p is a prime, then the maximum order of such a matrix is equal to $(p-1)p$, which can be obtained by having nonzero elements along the main super diagonal. This can be shown as follows.

Partitioning A into a diagonal matrix D and an upper triangular nilpotent matrix U, i.e. $A = U + D$, then we see that $D \cdot U, U \cdot D$ and U^2 are also upper triangular nilpotent. Then, inductively, if

$$A^i = D^i + U_i$$

we have

$$A^{i+1} = (A + U)(D^i + U_i)$$
$$= D^{i+1} + \underbrace{UD^i + DU_i + UU_i}_{\substack{\text{nilpotent upper} \\ \text{triangular}}}$$
$$= D^{i+1} + U_{i+1}$$

Hence $A^{\phi(p)} = A^{p-1} = I + U_\phi$, where ϕ is the Euler function, and $(I + U_\phi)^{p^t} \equiv I$ for some t.

If p is assumed to be greater than $n - 1$, we have $t = 1$. Thus the order of A is $(p-1)p$.

Example: Let $p = 5$. Let $f_1(x) = x^2 + x + 1$. $f_1(x)$ is irreducible over $Z/5Z$. The matrix B_1 is therefore given by

$$B = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 \\ 4 & 4 \end{pmatrix} \pmod 5$$

and

$$\begin{pmatrix} 0 & 1 \\ 4 & 4 \end{pmatrix}^{5^2-1} \equiv I \pmod 5$$

Let $f_2(x) = x^3 + 3x^2 + x + 2$. $f_2(x)$ is irreducible over $Z/5Z$. Hence the matrix B_2 is given by

$$B_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -1 & -3 \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 3 & 4 & 2 \end{pmatrix} \pmod 5$$

and

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 3 & 4 & 2 \end{pmatrix}^{5^3-1} \equiv I \pmod 5$$

Now we need to choose Y and Y^{-1} such that

$$\begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} Y^{-1} = A \in M_n(F_p)$$

Let x be an arbitary nonsingular matrix given below:

$$\begin{bmatrix} 2 & 3 & 1 & 0 & 1 \\ 1 & 2 & 2 & 3 & 1 \\ 1 & 0 & 3 & 2 & 4 \\ 1 & 2 & 1 & 4 & 3 \\ 2 & 4 & 0 & 1 & 1 \end{bmatrix}$$

(The determinant of Y is 4 (mod 5) and hence x is nonsingular.) The inverse of Y is given by

$$Y^{-1} = \begin{bmatrix} 0 & 2 & 1 & 3 & 0 \\ 0 & 1 & 2 & 2 & 0 \\ 3 & 3 & 1 & 4 & 3 \\ 2 & 2 & 4 & 2 & 4 \\ 3 & 0 & 1 & 4 & 2 \end{bmatrix}$$

and hence

$$A = \begin{bmatrix} 0 & 2 & 0 & 4 & 3 \\ 1 & 1 & 3 & 1 & 3 \\ 4 & 0 & 0 & 3 & 2 \\ 3 & 3 & 2 & 4 & 4 \\ 1 & 2 & 4 & 1 & 1 \end{bmatrix}$$

The order of A is equal to

$$l_{cm}\{(5^2 - 1)(5^3 - 1)\} = 744$$

and

$$A^{744} \equiv I \pmod 5$$

Hence the key space is $2 \leq x \leq 743$, where $A^x \equiv C \pmod 5$.

R. W. K. ODONI   *5th March 1984*
Department of Mathematics
University of Exeter. England

V. VARADHARAJAN
P. W. SANDERS
Department of Communication Engineering
Plymouth Polytechnic
Plymouth. Devon PL4 8AA. England

References

1   DIFFIE. W., and HELLMAN. M. E.: 'New directions in cryptography'. *IEEE Trans..* 1976. IT-22. pp. 644–654
2   Data Encryption Standard. FIPS Publ. 46. National Bureau of Standards, USA. Jan. 1977
3   POHLIG. S. G.. and HELLMAN. M. E.: 'An improved algorithm for computing logarithms over GF(p) and its cryptographic significance'. *IEEE Trans..* 1979. IT-24. pp. 106–110

* VARADHARAJAN. V., and ODONI. R. W. K.: 'Extension of RSA cryptosystem to matrix rings' (unpublished)

ERRATUM

Authors' correction

SPHICOPOULOS. T., TEODORIDIS. V., and GARDIOL. F.: 'Tractable form of the dyadic Green function for application to multilayered isotropic media', *Electron. Lett..* 1983, 19, (24), pp. 1055–1056

Throughout the letter. TM and TE must be transposed

Extension of RSA Cryptosystem to Matrix Rings

Abstract

A generalization of the RSA cryptosystem in the ring of matrices over Z/mZ is presented. It is shown that factorization of the modulus m is needed to compute the exponent of the group formed by either non-singular matrix messages or upper triangular matrices including diagonal elements thus offering the same level of security as the RSA system. The latter method employing the triangular matrices as messages seems to be more practical than the use of arbitrary non-singular matrix messages. The scheme is as suitable for privacy and authentication as its predececessor.

1. Introduction

Assume R is a finite ring with unity which is associative but not necessarily commutative. Suppose that members of the ring R are used as messages and that $r \epsilon R$ is enciphered as r^e where e is the published encrypting exponent. The trapdoor property can be stated as follows:-

There exists some integer $n > 0$ such that $r^{n+1} = r$ for all $r \epsilon R$. These rings are to be referred to as trapdoor rings. For instance, in Z/pZ, $r^p = r$ for all $r \epsilon R$. More generally, if we let $R = F_q = GF(q)$, the field of q elements where q is a prime power (p^k), then $r^q = r$ for all $r \epsilon R$. Further if R and S are any two such trapdoor rings, then the direct sum (or product) $R \oplus S$ consisting of vectors (r,s) with $r \epsilon R$ and $s \epsilon S$ is another trapdoor ring say T. The number of elements in the ring T is equal to the product of the number of elements in R and S. This above procedure can be applied repeatedly taking vectors of arbitrarily many components each taken from some finite field. Considering finite fields F_{q_i} for $1 \leqslant i \leqslant j$, where q_i's can be the same or different, the trapdoor ring R is formed by all vectors $x = (x_1, \ldots, x_j)$, where $x_i \epsilon F_{q_i}$ for $1 \leqslant i \leqslant j$. The ring R consists of $q_1 q_2 \ldots q_j$ elements and the equality $r^{n+1} = r$ is obeyed for all $r \epsilon R$, where n is equal to $(q_1 - 1) (q_2 - 1) \ldots (q_j - 1)$ or any multiple of it.

There are many finite rings which are not trapdoor rings. Consider for instance, $R = Z/p^2 Z$ where p is a prime. Then $p^2 \equiv p^3 \equiv \ldots \equiv 0$ in the ring R but $p \neq 0$ in the ring R. So the property that $p^{n+1} \equiv p$ is not satisfied for any $n > 0$. More generally, for a ring R to be a trapdoor ring,

it is necessary that R have no nilpotent elements except zero. However, if we take an integer m to be a square free positive integer say $m = p_1 \ldots p_j$ where all p_i's are distinct primes, then the ring $R = Z/mZ$ is a trapdoor ring. This ring can in fact be regarded as a direct sum of $F_{p_1} \oplus F_{p_2} \oplus \ldots \oplus F_{p_j}$ as described above. If $j = 2$, then this becomes the standard trapdoor ring used by the RSA cryptosystem[1]. This is a consequence of theorems of Wedderburn [2] on finite semisimple rings and skew fields.

The original RSA scheme derived its message space from the ring of integers modulo m, Z/mZ, where m is the product of two large distinct primes p and q. Here we investigate other finite systems that might serve as a basis for an extended RSA cryptosystem. Essential background material in group and ring theory can be found in [3] .

2. Matrices over Z/mZ

If the ring of all n x n matrices over the ring Z/mZ is considered, it is seen that the ring contains nilpotent elements when $n>1$. To overcome this problem, initially only the group M_n formed by the non-singular matrices of order n, is selected to form the message space of this extended system. But the use of such arbitrary non-singular matrices as messages poses further problems as the sender has no control over the matrix elements but must accept what the plaintext dictates. That is, the sender cannot always ensure that his messages will form non-singular matrices over Z/mZ. However, to begin with, the use of non-singular matrices as a possible message space is investigated in our matrix based RSA system.

Let us first consider the finite group formed by matrices of order n whose determinants are relatively prime to p and whose elements are in Z/pZ (p prime). The order of the group formed by these elements can easily be shown [4] to be equal to N_p where

$$N_p = (p^n-1)(p^n-p) \ldots (p^n-p^{n-1}) \tag{1}$$

If such non-singular matrices with elements over Z/pZ are used as messages then one can form a conventional cryptographic system where the secret key contains the modulus p itself. The encrypting (e) and decrypting (d) exponents can then be determined using

$$ed = 1 \pmod{N_p} \tag{2}$$

Page 506

The encrypting key is therefore (e, p, n) and the decrypting key is (d,p,n). None of these keys can be made public and the encryption and decryption procedures are as in the RSA system.

$$C = M^e \pmod{p}$$

and

$$M = C^d \pmod{p} \tag{3}$$

The above system can be modified to include the public key property as follows: Suppose the modulus used in the system is a composite number m whose factorization is given by

$$m = \prod_{j=1}^{s} p_j^{r_j} \tag{4}$$

Then, using the Chinese Remainder Theorem, the order N_m [5-8] of the multiplicative group formed by non-singular matrices of order n over Z/mZ is given by

$$N_m = \prod_{j=1}^{s} N_{p_j}^{r_j} \tag{5}$$

where

$$N_{p_j}^{r_j} = p_j^{(r_j-1)n^2} (p_j^n-1)(p_j^n-p_j) \ldots (p_j^n-p_j^{n-1}) \tag{6}$$

Now as in the RSA cryptosystem if we take m to be the product of two distinct primes p and q, then the expressions (5) and (6) simplify to

$$N_m = N_p \cdot N_q \tag{7}$$

$$= (p^n-1)(p^n-p) \ldots (p^n-p^{n-1})(q^n-1) \ldots (q^n-q^{n-1}) \tag{8}$$

Therefore for a message matrix $M \varepsilon M_n(Z/mZ)$

$$M^{N_m} \equiv I \pmod{m} \tag{9}$$

and hence the coding exponents can be calculated using

$$ed \equiv 1 \pmod{N_m} \tag{10}$$

Page 507

The expression for the order N_m depends on the structure of m, that is, on its prime factors. This can therefore be used to form a public key cryptosystem whose security is the same as that of the RSA system.

Although the order N_m can be used in finding e and d usually the order is a very large number. For instance, even for small primes such as $p = 13$ and $q = 23$, the order is approximately 1.6×10^{22} for 3 x 3 non-singular matrices. Therefore, it is desirable to find the exponent EXP of the group, i.e., the least integer greater than zero such that

$$M^{EXP} \equiv I \pmod{m} \tag{11}$$

EXP is a divisor of the order N_m of the group.

Let us first consider the exponent of the group formed by the non-singular matrices over Z/pZ, $M_n(Z/pZ)$.

Let the exponent be ℓ such that

$$A^{\ell} \equiv I \text{ for all } A \in M_n(Z/pZ) \tag{12}$$

Assume that $p > n$. $A^{\ell} \equiv I \pmod{p}$ implies that $x^{\ell}-1$ is divisible by the minimum polynomial of A. As A ranges over the non-singular matrices of order n over Z/pZ, $x^{\ell}-1$ must be divisible by every monic irreducible polynomial $p(x)$ ($\neq x$) of degree $\leq n$ in Z/pZ. Every irreducible polynomial $p(x)(\neq x)$ of degree u divides $x^{p^u-1}-1$. Thus $x^{\ell}-1$ must be divisible by $x^{p^u-1}-1$

But

$$x^b-1 \equiv 0 \pmod{x^a-1} \tag{13}$$

implies $a \mid b$.

Hence,

$$\ell = 0 \pmod{p^u-1} \text{ for } 1 \leq u \leq n \tag{14}$$

Therefore,

$$\ell = 0 \pmod{\ell cm\{p-1, \ldots, p^n-1\}} \text{ certainly} \tag{15}$$

Furthermore, the matrix A given by

$$A = I + \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ & \ddots & \ddots & & \bigcirc & \\ & & \ddots & \ddots & & \\ & \bigcirc & & \ddots & \ddots & 0 \\ 0 & & & & \ddots & 0 \end{bmatrix} \tag{16}$$

satisfies $A^p = I \neq A$ $(p>n)$ $\tag{17}$

That is, A has order p and hence $p \mid \ell$.

Hence the exponent of $GL(n,p), p > n,$ is given by

$$\ell = p \; \ell cm \{ p-1, \; p^2-1, \; \ldots, \; p^n-1 \} \qquad (18)$$

Now for any $A \in M_n(Z/pZ)$, using Jordan's Canonical form, there exists a non-singular matrix E such that

$$E^{-1}AE = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_r \end{bmatrix} \qquad (19)$$

$$= D$$

Each block B_i is of the form

$$B_i = \begin{bmatrix} \lambda_i & 1 & & \bigcirc \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \bigcirc & & & \lambda_i \end{bmatrix} \qquad (20)$$

i.e. $B_i = \lambda_i I_i + N_i$ for some upper triangular nilpotent matrix N_i.

where λ_i's are non-zero in $F_{p^{r_i}}$ for some $r_i \leq n$.

If the order of D is k, that is, $D^k \equiv I \pmod{p}$, then as $D = E^{-1}AE$, this gives

$$(E^{-1}AE)^k = (E^{-1}AE)(E^{-1}AE) \; \ldots \; k \text{ times}$$
$$= E^{-1}A^k E$$

Thus $A^k \equiv I \pmod{p} \qquad (21)$

Order of A = order of D = k = ℓcm of orders of B_i.

If $N_i = 0$, then order of B_i is a divisor of $p^{r_i}-1$.

Hence the order of A divides $\ell cm \{p-1, \; \ldots, \; p^n-1\}$.

Otherwise, $B_i^p = \lambda_i^p I_i$ will have such an order and hence the order of A divides $p \; \ell cm \{p-1, \; \ldots, \; p^n-1\}$

Page 509

A multiple of p in the expression for the exponent ℓ is expected as the order given by (1) contains multiples of p.

Similarly,

$$A^s = I \pmod{q} \text{ for all } A \text{ in } M_n(Z/qZ)$$

where $s = q \operatorname{\ell cm} \{q-1, q^2-1, \ldots, q^n-1\}$ (22)

Therefore exponent EXP of the group GL(n,m) where m = p.q is given by

$$EXP = \operatorname{\ell cm} (p \operatorname{\ell cm} \{p-1, \ldots, p^n-1\}, q \operatorname{\ell cm} \{q-1, \ldots, q^n-1\}) \quad (23)$$

Now let us extend this argument to non-square free modulus m. First consider a matrix A in $M_n(Z/p^2Z)$. Let Θ be the natural homomorphism from Z/p^2Z onto Z/pZ (p prime). From the above argument

$$\Theta(A)^{EXP} \equiv I \text{ in } M_n(Z/pZ) \qquad\qquad (24)$$

$$\equiv \Theta(I)$$

Therefore

$$\Theta(A^t - I) \equiv 0 \pmod{p} \text{ where } t = EXP$$

This means that every entry in $A^t - I$ is some multiple of prime p and hence

$$A^t - I = p B \text{ for some matrix } B$$

That is,

$$A^t = I + pB \qquad\qquad\qquad (25)$$

$$A^{tp} = (I+pB)^p$$

Using the binomial theorem,

$$A^{tp} = I + \binom{p}{1}pB + \binom{p}{2}p^2B^2 + \ldots$$

$$\equiv I \pmod{p}$$

Therefore considering in general a matrix in $M_n(Z/p^hZ)$

$$M_n(Z/p^hZ) \longrightarrow M_n(Z/pZ)$$

$$\theta: \qquad A \qquad \longrightarrow \theta(A)$$

If $\theta(A)$ has order t, then A has order t or pt or p^2t ... or $p^{h-1}t$.

If

$$m = \prod_{i=1}^{s} p_i^{r_i} \qquad (26)$$

then

$$EXP = \ell cm \{v_1, v_2, \ldots, v_s\}$$

where $v_i = p_i^{r_i-1}(w_i)$ $\qquad\qquad (27)$

and $\quad w_i = p_i \; \ell cm(p_i-1, p_i^2-1, \ldots, p_i^n-1)$

(assuming p_i is greater than n for all i.)

Again from (27), it is clearly seen that the exponent EXP depends upon the prime factor composition of m.

From cryptography point of view, the use of such non-singular matrices may be considered impractical as the sender has no control over the elements in the message matrix. The sender is faced with the problem of determining whether a plaintext message matrix is non-singular or not: if it is singular, he cannot encrypt that particular message. This is not acceptable for cryptographic application. However, the proportion of non-singular matrices over Z/mZ, where for instance m = pq and p and q are large district primes, is very much close to 1.

Page 511

One common approach to obtain an arbitrary non-singular matrix over the reals is to have the diagonal entries of the matrix message much bigger than the corresponding entries in the row and the column. But this diagonal dominance does not always ensure that the matrix will be non-singular when working over finite rings. For instance, consider the matrix A given below which is 'diagonally dominant'.

$$A = \begin{pmatrix} p-1 & 1 \\ 1 & p-1 \end{pmatrix} \quad (\text{mod } p) \qquad (28)$$

Det $A = p^2 - 2p \equiv 0$ (mod p). Hence diagonal dominance is not applicable over finite rings.

Alternatively, let us now consider the set of upper triangular matrices as a possible choice of our message space. If the diagonal entries are made unity, this ensures that the matrix is invertible as the determinant is relatively prime to the modulus m.

Let M represent such an upper triangular message matrix. We can partition the matrix M into I+N where N is a nilpotent matrix and I is the identity matrix. If M is in $U_n(Z/pZ)$ then $(I+N)^p = I$ as $N^p = 0$ assuming $p \geqslant n-1$. The order of the group formed by these upper triangular matrices if $p^{n(n-1)/2}$. The order becomes $m^{n(n-1)/2}$ when considering matrices U_n over Z/mZ. Here we see that the order of the group depends only on m and not on the factorization of m. Hence this is not suitable for a public key system.

On the other hand, if we alter the message space to contain upper triangular matrices with diagonal entries prime to m, then such messages are invertible modulo m. This is not a serious problem as in practice m is a product of large primes and the diagonal elements can be chosen to be relatively smaller integers. Now the order of the group formed by such matrices is determined as follows:

Considering a nxn matrix, it is required that all the n diagonal entries must be coprime to m. The number of integers less than m and coprime to m is given by the Euler totient function $\phi(m)$.

The remaining $\frac{1}{2}n(n-1)$ superdiagonal entries of the matrix may take any value modulo m. Therefore the order is equal to $m^{n(n-1)/2}\phi(m)^n$ The vital difference between this and the one calculated above is that now the order of the group is dependent on the prime factors of m. Hence the modulus m needs to be factorized before the decryption

exponent d can be calculated using ed = 1 mod (order). As for the set of non-singular matrices, one can determine the exponent of the group formed by these upper triangular matrices with diagonal entries prime to m. The exponent can be used instead of the order in finding e and d.

First consider a square free modulus m. Let

$$m = \prod_{j=1}^{s} P_j \qquad (29)$$

Let us first consider a message M in Z/pZ whose diagonal elements are relatively prime to p.

$$M = \begin{pmatrix} \alpha_{11} & & ? \\ & \ddots & \\ O & & \alpha_{nn} \end{pmatrix} \quad \text{where } (d_{ii}, p) = 1 \qquad (30)$$

$$\text{for all } i, \ 1 \leq i \leq n$$

Partitioning M into a diagonal matrix D and an upper triangular nilpotent matrix U, that is, M = D+U, then it is seen that $D \cdot U$, $U \cdot D$ and U^2 are also upper triangular nilpotent. Then inductively,

$$\text{if} \quad M^r = D^r + U_r \qquad (31)$$

we have $M^{r+1} = (D+U)(D^r+U)$

$$= D^{r+1} + \underbrace{U \cdot D^r + D \cdot U_r + U \cdot U_r}_{\text{nilpotent upper triangular}}$$

$$M^{r+1} = D^{r+1} + U_{r+1} \qquad (32)$$

Hence

$$M^{\emptyset(p)} \equiv I + U_\emptyset \ (\text{mod } p) \qquad (33)$$

and

$$(I + U_\emptyset)^P \equiv I + U_\emptyset^p \ (\text{mod } p)$$

$$(I + U_\emptyset)^{P^2} \equiv I + U_\emptyset^{p^2} \ (\text{mod } p)$$

$$\vdots \qquad\qquad \vdots$$

etc

Page 513

Thus
$$M^{p^t \phi(p)} \equiv I \pmod{p} \text{ for some } t \tag{34}$$

If $p \geq n-1$, then $t = 1$

Therefore, the exponent of the group formed by upper triangular matrices (with invertible elements along the diagonal) over Z/pZ is equal to $\phi(p) \cdot p$

Now if
$$m = \prod_{j=1}^{s} p_j \tag{35}$$

then the exponent divides
$$\ell cm \{\phi(p_1)p_1, \phi(p_2)p_2, \ldots, \phi(p_j)p_j\} \tag{36}$$

(Note that if the diagonal entries are unity then the exponent is equal to m or $2m$, if m is even or odd respectively.)

Let us now consider a non-square free modulus m given by
$$m = \prod_{j=1}^{s} p_j^{r_j} \tag{37}$$

First consider an $n \times n$ upper triangular non-singular matrix M over $Z/p^r Z$, $(p > n)$. Again let $M = D + U$ where D is a diagonal matrix and U is an upper triangular nilpotent matrix over $Z/p^r Z$. Following the argument given above, it is seen that
$$M^{\phi(p^r)} \equiv I + U_\phi \pmod{p^r} \tag{38}$$

where U_ϕ is some upper triangular nilpotent matrix.
Hence
$$(I + U_\phi)^p \equiv I + p\, U_{\phi_1} \pmod{p^r}$$

$$(I + pU_{\phi_1})^p \equiv I + p^2\, U_{\phi_2} \pmod{p^r}$$

$$\vdots \qquad\qquad \vdots$$

$$(I + p\, U_{\phi_{r-1}})^p \equiv (I + U_\phi)^{p^r} \equiv I + p^r\, U_{\phi_r} \equiv I \pmod{p^r}$$

Page 514

Therefore $M^{\phi(p^r)p^r} \equiv I \pmod{p^r}$ (39)

Hence the exponent of the group formed by upper triangular non-singular matrices over Z/mZ, where m is given by (37), divides

$$\text{lcm } \{\phi(p_1^{r_1})p_1^{r_1}, \ \phi(p_2^{r_2})p_2^{r_2}, \ \ldots, \ \phi(p_s^{r_s})p_s^{r_s}\} \qquad (40)$$

3. System Design and Operation

The designer randomly chooses large primes p_1 to p_s for some $s \geqslant 2$ following the guidelines suggested in [1] and forms their product m. The primes need not be necessarily distinct. As in the case of the RSA system, these primes are kept secret. As the upper triangular non-singular matrix messages can be selected arbitrarily a public key cryptosystem using such a message space has been implemented. The coding exponents e and d are determined using

ed = 1 (mod EXP) (41)

where EXP can be equal to (40).

The public encryption key is given by (e,m,n) and the secret decryption key is (d,m,n) where n denotes the dimension of matrix messages.

The message is divided into blocks of integers less than the modulus m and a sequence of nxn upper triangular matrices is constructed by arranging the integers in order as they occur, left to right and top to bottom. The encryption procedure consists of raising each of these upper triangular matrices to the power e. This has been implemented using the well known square and multiply technique [9]. The ciphertext produced consists of a sequence of nxn upper triangular matrices over Z/mZ. Each of these matrices is transmitted to the receiver by sending the n(n+1)/2 ciphertext matrix elements (excluding the lower triangular zeroes) in order as they occur in the matrix, left to right, top to bottom, with a space separating the elements. The receiver recovers the original message by first reconstructing the sequence of nxn upper triangular cipher matrices and then raising them to the power d modulo m.

Such an extended RSA matrix system using upper triangular message matrices has been simulated on the Prime computer. An example showing the various parameters is given in Figure 1.

Page 515

4. Discussion

One can see that the RSA system can be extended to matrix messages provided the message space is restricted to avoid nilpotent elements. In this paper, the group of non-singular matrices over Z/mZ and the group of upper triangular matrices with diagonal elements coprime to m over Z/mZ have been investigated. In both these cases, factorization of the modulus m into primes is required to compute the exponent thus providing the same level of security as the RSA system. But the first case, employing arbitrary non-singular matrices as the message space does not seem to be suitable for a practical cryptosystem as it is not possible to restrict arbitrary plaintext matrices to be non-singular. On the other hand, in the second case employing upper triangular non-singular matrices, it appears that the messages can be selected arbitrarily in practice. The exponent of the group formed by such upper triangular nxn matrices over Z/mZ divides $\text{lcm} (\emptyset(p_1^{r_1})p_1^{r_1}, \ldots, \emptyset(p_s^{r_s})p_s^{r_s})$ where $m = \prod_{j=1}^{s} p_j^{r_j}$ with $p_j > n$ for all $1 \leq j \leq s$.

Further two points are worth mentioning regarding this extended matrix system. Firstly, it is seen that non-square free modulus can be used which is not possible with the RSA system. Secondly, the use of a matrix as a message may allow large amounts of data to be processed within one encryption/decruption cycle. Whether this is an advantage depends upon the ease with which matrix manipulations can be carried out. Computational savings can be achieved if transform techniques are used to reduce the number of scalar multiplications involved in a matrix multiplication.

Example

Let the modulus $m = p.q = 41.29 = 1189$

Exponent of the group formed by 3x3 upper triangular non-singular matrices over $Z/1189Z$, divides
$\text{lcm} \{40.41, 28.29\} = 332920$

Choosing the encrypting exponent, $e = 1317$, the decrypting exponent d can be calculated using Euclid's algorithm and is equal to 117293. That is,
$$1317 . 117293 \equiv 1 \pmod{332920}$$

Let us assume that the plaintext message to be encrypted is 232677205141. In this example, the message is divided into 2-digit blocks of integers less than m. Starting from right to left as

(23 26 77 20 51 41)

The upper triangular plaintext message matrices become

$$P_1 = \begin{bmatrix} 90 & 23 & 26 \\ 0 & 50 & 77 \\ 0 & 0 & 48 \end{bmatrix} \quad \text{and} \quad P_2 = \begin{bmatrix} 31 & 20 & 51 \\ 0 & 215 & 41 \\ 0 & 0 & 289 \end{bmatrix}$$

where the diagonal elements are arbitrarily chosen to be relatively prime to 1189.

The ciphertext message matrices are then given by

$$C_1 = \begin{bmatrix} 90 & 23 & 26 \\ 0 & 50 & 77 \\ 0 & 0 & 48 \end{bmatrix}^{1317} \equiv \begin{bmatrix} 1105 & 458 & 1070 \\ 0 & 50 & 251 \\ 0 & 0 & 831 \end{bmatrix} \quad (\text{mod } 1189)$$

$$C_2 = \begin{bmatrix} 31 & 20 & 51 \\ 0 & 215 & 41 \\ 0 & 0 & 289 \end{bmatrix}^{1317} \equiv \begin{bmatrix} 843 & 774 & 660 \\ 0 & 592 & 41 \\ 0 & 0 & 405 \end{bmatrix} \quad (\text{mod } 1189)$$

These ciphertext matrices are transmitted to the receiver as
(1105 458 1070 50 251 831 843 774 660 592 41 405)

The receiver reconstructs the matrices C_1 and C_2 and computes C_1^d (modm) and C_2^d (modm) to obtain P_1 and P_2 and hence the plaintext message. That is,

$$C_1^d = \begin{bmatrix} 1105 & 458 & 1070 \\ 0 & 50 & 251 \\ 0 & 0 & 831 \end{bmatrix}^{117293} \equiv \begin{bmatrix} 90 & 23 & 26 \\ 0 & 50 & 77 \\ 0 & 0 & 48 \end{bmatrix} \quad (\text{mod } 1189) = P_1$$

Page 517

$$C_2{}^d = \begin{bmatrix} 843 & 774 & 660 \\ 0 & 592 & 41 \\ 0 & 0 & 405 \end{bmatrix}^{117293} \equiv \begin{bmatrix} 31 & 20 & 51 \\ 0 & 215 & 41 \\ 0 & 0 & 289 \end{bmatrix} \pmod{1189} \equiv P_2$$

Figure 1

References

1. Rivest, R.L., Shamir, A., and Adleman, L. 'A Method for obtaining signatures and public key cryptosystems', Comm. ACM, 1978, 21, pp. 120-126.

2. Rédei, L., Algebra Vol. 1, Pergamon Press, International Series of Monographs in Pure and Applied Mathematics, 1967.

3. Albert, A.A., Fundamental Concepts of Higher Algebra, The University of Chicago Press, Phoenix Science Series, 1961.

4. Jordan, C., Traité des Substitutions, 1870.

5. Farahat, H.K., 'The multiplicative group of a ring', Math-Zeitschr., 87, 1965, pp. 378-384.

6. Brawley, J.V., 'The number of non-singular matrices over a finite ring with identity', 1971 (unpublished).

7. Roby Norbert, 'Sur le cardinal du Groupe GL(n,A) où A est anneau fini', Annals Acad. Brasil, 49, 1977, pp.15-18.

8. McDonald, B.R. 'Involutory matrices over finite Local rings', Canadian Journal Math., 24, 1972, pp.369-378.

9. Knuth, D.E., 'The Art of Computer Programming, Vol.2: Seminumerical Algorithms, 2nd Edition, Addison- Wesley, 1981.

Acknowledgements

The authors would like to thank the referees for their valuable comments and suggestions.

V. Varadharajan,
Department of Communication Engineering,
Plymouth Polytechnic,
Drake Circus,
Plymouth,
U.K.

R. Odoni,
Department of Maths,
Exeter University,
North Park Road,
Exeter,
U.K.
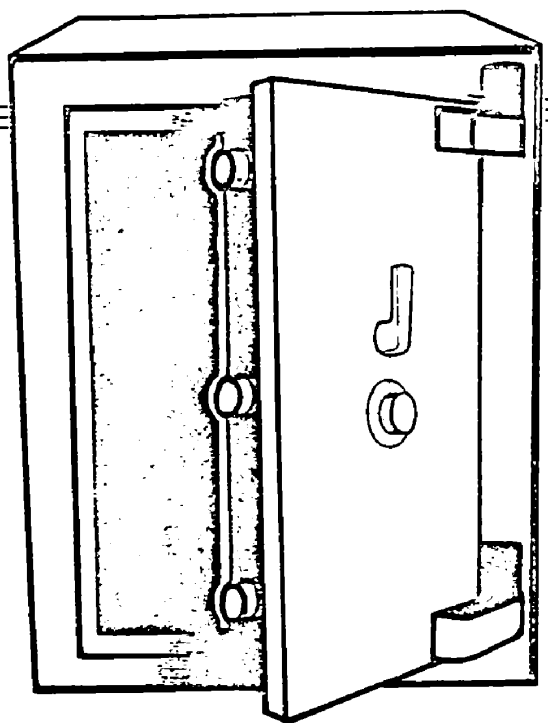
TH the advent of electronic mail, ctronic funds transfer and the tributed office, an increasing amount of nfidential data is being stored on disc d transmitted between terminals and mputers.

The information may be financial, iere a high degree of security is required, it may consist of sensitive commercial alings, personnel records, legal docu- ents, etc., which could cause difficulties problems if they fell into the wrong nds.

It is not too difficult to take a copy of a ;c, or listen in on telephone lines with a cheap, yet sophisticated communica- n equipment presently available and en examine the information at one's sure.

The relatively simple precautions, if y, employed in many data systems ove a minor obstacle to the determined ttack". The following describes the eration of a unit whereby the Apple ter- inal user can reduce this possible curity problem to a minimum.

Data encryption moved out of the ilitary and political spheres in 1976 hen an IBM encryption algorithm was cepted by the American National ureau of Standards as the encryption echanism for all federal non-military pplications. It is known as the Data ncryption Standard (DES) and is likely to a adopted as such in this country. The rd shown in the photograph opposite ilises this standard.

The algorithm changes the plain data to an unintelligible cipher form under the ntrol of a key; the change being to such n extent that the transformation is

DES can keep your data safe

entirely different for each key value.

The algorithm, which has been made public for almost 10 years now and has not yet been cracked, uses multiple modulo 2 addition, permutations and sub-

stitutions to ensure that the key or plain text cannot be discovered from the cipher text.

The very simple modulo 2 example in Figure I shows the difficulty of retrieving the original information when the "carrys" are lost unless the correct key is available.

The standard uses a 56 bit key with 64 bit blocks of plain text to produce, after 16 "rounds", 64 bit cipher blocks of data giving a "key space" of 2^{56} – or 10^{17} different key combinations. This is an extremely large number, and ensures that if each key combination was tried in turn by a fast computer it would take many years before all were covered.

The key can frequently be changed or multiple encryption made with different keys to maximise security. This alogrithm has recently become available in large scale integrated circuit form.

The encryption card can operate in one of three modes.

● Point to point communication between Apple terminals not containing a host computer. Page 521

● Local storage of encrypted programs

Simplified principle

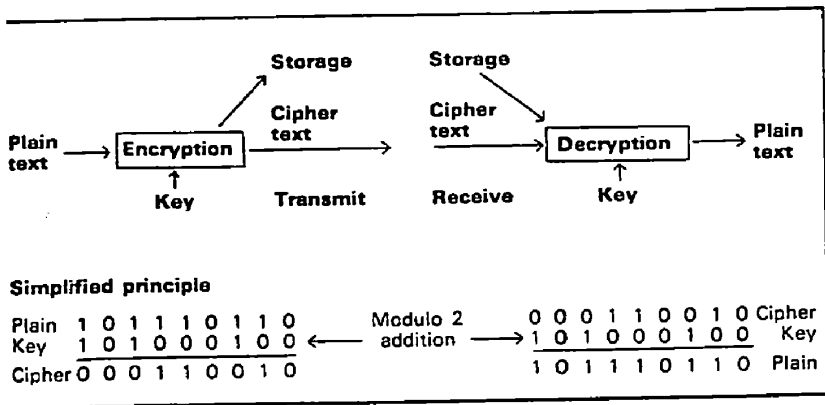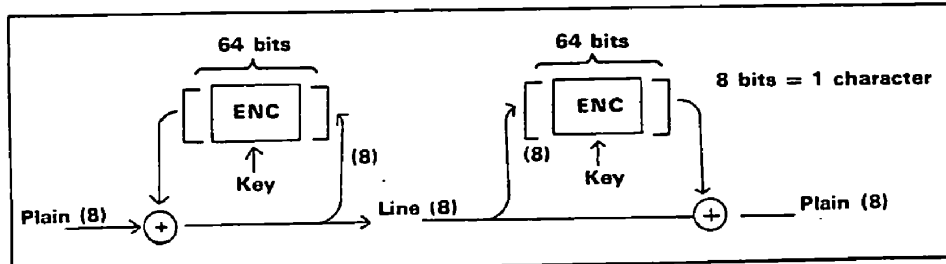|		Modulo 2 addition		
Plain	1 0 1 1 1 0 1 1 0		0 0 0 1 1 0 0 1 0	Cipher
Key	1 0 1 0 0 0 1 0 0 ←	→	1 0 1 0 0 0 1 0 0	Key
Cipher	0 0 0 1 1 0 0 1 0		1 0 1 1 1 0 1 1 0	Plain

Figure I

Figure II

---

d data files on the Apple disc system.

Communication between an Apple terminal and a host computer for storage and trieval of encrypted information in its emory bank. The operation of these odes is controlled by programs stored in OM on the card, the appropriate ogram being selected by a CALL command after the card has been activated ith a POKE instruction.

In the secure storage mode Applesoft · Integer Basic data is passed through e encryption chip after it has been inialised with the chosen key, consisting of ny eight character combination, including control characters, from the keyboard.

The encrypted form is then stored on a isc and can be catalogued in the normal anner.

If this cipher text is later downloaded om the disc and listed complete rubbish ill result. Often the listing continues to in because the end of line/file pointers re never reached, or it is maybe much horter in length than the original because e control characters in the encrypted orm are not displayed.

When operating in the communication nodes, a suitable modem must be used to onvert the digital codes from the card nto equivalent speech signals for ansmission. The card incorporates a ommunication controller that allows full iuplex, asynchronous transmission at the :CITT standard rates to 9.6 k bit/sec: the equired rate being selected from the keyoard.

Different rates for transmission and eception allow asymmetric duplex operaion, such as is used on the Prestel data iases and other viewdata systems.

The DES algorithm can be used in a iumber of ways to obtain the required ecurity but the method chosen by the ard is a stream cipher feedback arrangenent as shown in Figure II.

In this case the DES is used as a seudo random number generator, its

By P.W. SANDERS and V. VARADHARAJAN

output being continuously exclusive OR-ed (modulo 2 addition) with the plain text. The decryption process operates in the same manner – an identical pseudo random stream of bits being generated in synchronism to retrieve the plain text after another exclusive OR operation. This method has a number of advantages when used in the above applications:
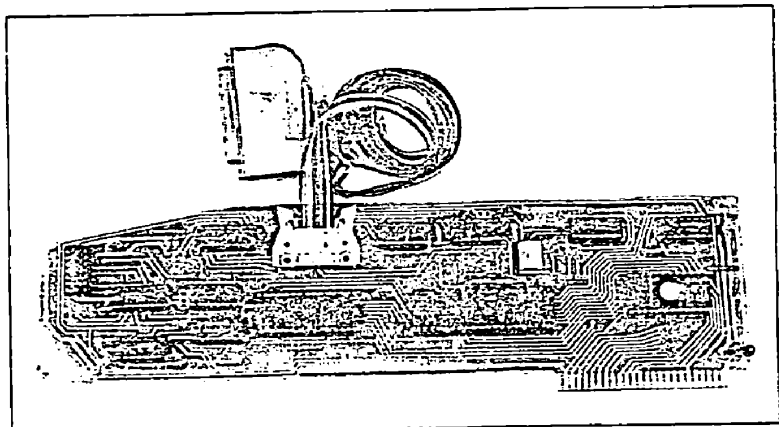
☐ There is a chaining of the encryption – the output cipher text character becomes dependent upon the previous eight characters of cipher and plain text, as well as the key, which removes the problem of

straight forward substitution of characters to alter the message without knowledge of the key.

☐ The system has a self-synchronising property when confronted with transmission errors. A error within a cipher text character affects that character and the following eight characters only on decryption, after which the system automatically becomes resynchronised.

☐ The arrangement is very flexible, allowing a mixture of plain and encrypted characters under keyboard control. With block cipher, eight character blocks must always be used, necessitating "padding" if less than eight encrypted characters are needed. With this "byte" feedback a single character can be efficiently encrypted.

A stand-alone unit has been developed for any computer that has the same facilities as the card, but being connected between the RS 232 interface output of the computer and the corresponding input of the modem. ⌘

The encryption card