

1568

ENCRYPTION KEY MANAGEMENT

By

MAJOR B P SINGH

Under the guidance of

LT COL P K JAGGIA

A PROJECT REPORT SUBMITTED

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

MASTER OF TECHNOLOGY

IN

ELECTRICAL ENGINEERING

46pt bib

**FACULTY OF ELECTRONICS
MILITARY COLLEGE OF ELECTONICS AND
MECHANICAL ENGINEERING
SECUNDERADAB-500 015, AP,INDIA.**

1997

DEDICATED TO THE PURSUIT
OF TECHNOLOGICAL EXCELLENCE

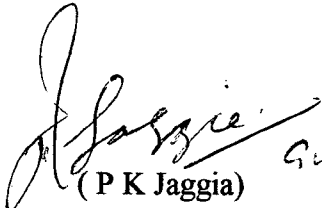
CERTIFICATE

This is to certify that the dissertation entitled 'Encryption Key Management submitted in partial fulfillment of requirements for the degree of Master of Technology in Electrical Engineering of Military College of Electronics and Mechanical Engineering, Secunderabad is a record of work carried out by Major Bhanu Pratap Singh under my guidance.

The matter presented in this paper has not been submitted for the award of any other degree, diploma or certificate.

SECUNDERABAD

Dated: 23 Apr 97

 Guide
(P K Jaggia)

Col

Faculty of Electronics
MCEME

BOARD OF EXAMINERS

Maj Bhanu Partap Singh has presented the dissertation on Encryption Key Management at MCEME on 24 July 98 and has fulfill^{ed}_l the requirement to the best of my satisfaction.


Col K DWARKANATH (Retd)

ACKNOWLEDGMENT

I consider it a pleasant duty to express my heartfelt gratitude, appreciation and indebtedness to Lt. Col. P K Jaggia , Faculty of Electronics ,MCEME for his guidance and assistance for successful completion of this thesis.

I am thankful to Lt. Col. A V Subramanian ,Commanding Officer ,313 Stn Wksp EME for his kind co-operation , pursuance and providing the necessary facilities, for timely completion of this thesis.

Last but not the least, I am greatly indebted to library of Usmania University, Hyderabad and Technical Information Center R & D Headquarter, Sena Bhawan, New Delhi.

Co-operation extended by other individuals directly or indirectly is also acknowledged.

INDEX

SERNO	NOMENCLATURE	PAGE NO
1.	INTRODUCTION	01
2.	CLASSICAL CRYPTOGRAPHY	03
3.	MODULAR ARITHIMATIC	06
4.	CRYPTO SYSTEM	09
5.	PRACTICAL SECURITY	15
6.	STREAM CIPHER SYSTEM	19
7.	DATA ENCRYPITION STANDARD	22
8.	KEY MANGEMENT	28
9.	CONCLUSION	46
10.	BIBILIOGRAPHY	47

INTRODUCTION

The need to keep certain messages secret has been appreciated for thousands of years. People were active to realize the advantages to be gained from intercepting secret information, and this has led to a continuous, fascinating battle between the 'codemakers' and the 'codebreakers'. The broad spectrum for this contest is the communication medium which has changed considerably over the years. The society is now highly dependent on modern, fast and accurate means of transmitting messages. Along with the old method of communication systems such as post, courier service, now a day radios, television telephone, telex and high speed data links are available. In each mode of transmission the aim remains to have a cheaper and reliable system. There are, however, a number of situations where the information is confidential, and where the interceptor will gain immensely from the knowledge gained by monitoring the information circuit. In such situation, the communicants must conceal the content of their message. At some occasion it may be good enough to conceal the information from a casual listener from understanding the message, but there are other times when it is crucial that even the most determined interceptor must not be able to deduce it.

The need of concealing the information before it is transmitted has increased over a period of time. The common-man has become aware that information pertaining him is transmitted over various data links and he feels that if not all some information is confidential and hence unauthorised personnel should not have access to it, more so, shall not be able to alter it. In such situations the communicants have no alternative but to give attention to security of their transmission.

The cryptographer's use various methods to encipher the information before the same is communicated. There are various hardware and software available for this purpose. In present day scenario where mode of communication are available they need to be reliable in terms of stability and accuracy such that no portion of the enciphered information is altered even by mistake as the receptor will not be able to decipher the message. Not only the information being transmitted should reach the adversary but he should also not be able to break the key and the key pattern being used to decipher the content of the message transmitted or being transmitted.

To achieve this aim the cryptographer has to pay attention on the **Key Management** aspect as well. While addressing to the problem of key management a number of question arise like, how large should be the key, how often should it be changed, by which method it should be changed, how it should be generated and, how it should be transmitted?

Classical Cryptography

The fundamental objective of cryptography is to enable two people, usually referred to as Alice and Bob, to communicate over an insecure channel in such a way that an opponent, Oscar, cannot understand what is being said. This channel could be a telephone line, computer network, for example. The information that Alice wants to send to Bob, which we call "plaintext," can be English text, numerical data, or anything at all - its structure is completely arbitrary. Alice encrypts the plaintext, using a predetermined key, and sends the resulting ciphertext over the channel. Oscar, upon seeing the ciphertext in the channel by eavesdropping, cannot determine what the plaintext was; but Bob, who knows the encryption key, can decrypt the ciphertext and reconstruct the plaintext.

This concept is described more formally using the following mathematical notation.

DEFINITION: A cryptosystem is a five-tuple (P, C, K, E, D) , where the following conditions are satisfied:

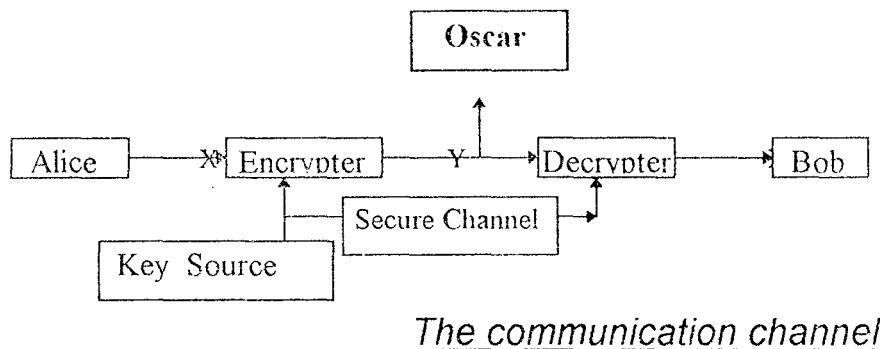
1. P is finite set of possible **plaintexts**.
2. C is a finite of possible **ciphertexts**.
3. K , the **keyspace**, is a finite set of possible **keys**.
4. For each $K \in K$, there is an **encrypting rule** $e_k \in E$ and a corresponding **decryption rule** $d_k \in D$. Each $e_k : P \rightarrow C$ and $d_k : C \rightarrow P$ are functions such that $d_k(e_k(x)) = x$ for every plaintext $x \in P$.

The main property is property 4 . It says that if a plaintext x is encrypted using e_k , and the resulting ciphertext is subsequently decrypted using d_k , then the original plaintext x results

Alice and Bob will employ the following protocol to use a specific cryptosystem. First, they choose a random key $K \in K$. This is done when they are in the same place and are not being observed by Oscar, or, alternatively, when they do have access to a secure channel, in which case they can be in different places. At a later time, suppose Alice wants to communicate a message to Bob over an insecure channel and message is a string

$$X = x_1 x_2 \dots x_n$$

for some integer $n \geq 1$, where each plaintext symbol $x_i \in P$, $1 \leq i \leq n$. Each x_i is encrypted using the encryption key rule e_k specified by the predetermined key K . Hence, Alice computes $y = e_k(x_i)$, $1 \leq i \leq n$, and the resulting ciphertext string is $y = y_1 y_2 \dots y_n$.



$y = y_1 y_2 \dots y_n$ is sent over the channel. When Bob receives $y_1 y_2 \dots y_n$, he decrypts it using the decryption function d_k , obtaining the original plaintext string, $x_1 x_2 \dots x_n$.

Clearly, it must be the case that each encryption function e_k is an injective function (i.e., one-to-one), otherwise, decryption could not be accomplished in an unambiguous & continuous manner.

For example, if

$$y = e_k(x_1) = e_k(x_2)$$

where $x_1 \neq x_2$, then Bob has no way of knowing whether y should decrypt to x_1 or x_2 . Note that if $P = C$, it follows that each encryption function is a permutation. That is, if the set of plaintexts and ciphertexts are identical, then each encryption function just rearranges (or permutes) the elements of this set.

MODULAR ARITHMETIC

DEFINITION Suppose a and b are integers, and m is a positive integer. Then we write $a \equiv b \pmod{m}$ if m divides $b - a$. The phrase $\equiv b \pmod{m}$ is read as “ a is congruent to b modulo m ”. The integer m is called the modulo.

Suppose we divide a and b by m , obtaining integer quotients and remainders, where the remainders are between $0 \leq r_2 \leq m - 1$. Then it is not difficult to see that $a \equiv b \pmod{m}$ if and only if $r_1 = r_2$. We will use the notation $a \bmod m$ (without parentheses) to denote the remainder when a is divided by m , i.e., the value r_1 above. Thus $a \equiv b \pmod{m}$ if $a \bmod m = b \bmod m$. If we replace a by $a \bmod m$, we say that a is reduced modulo m .

Many computer programming languages define $a \bmod m$ to be the remainder in the range $-m + 1, \dots, m - 1$ having the same sign as a . For example, $-18 \bmod 7$ would be -4 , rather than 3 as we defined it above. But for our purposes, it is much more convenient to define $a \bmod m$ always to be non-negative.

We can now define arithmetic modulo m : Z_m is defined to be the set $\{0, \dots, m-1\}$, equipped with two operators, $+$ and \times . Addition and multiplication in Z_m work exactly like real addition and multiplication, and the results are reduced modulo m .

For example, suppose we want to compute 11×13 in Z_{16} . As integers, we have $11 \times 13 = 143$. To reduce 143 modulo 16 , we just perform ordinary long division; $143 = (8 \times 16) + 15$, so $143 \bmod 16 = 15$, and hence $11 \times 13 = 15$ in Z_{16} .

These definition of addition and multiplication in Z_m satisfy most of the familiar rules of arithmetic.

These properties are listed here :

1. addition is closed, i.e., for any $a, b \in Z_m$, $a + b \in Z_m$
2. addition is commutative, i.e., for any $a, b \in Z_m$, $a + b = b + a$
3. addition is associative, i.e., for $a, b, c \in Z_m$, $(a + b) + c = a + (b + c)$
4. 0 is an addition identity, i.e., for any $a \in Z_m$, $a + 0 = 0 + a = a$
5. the additive inverse of any $a \in Z_m$ is $m - a$, i.e., $a + (m - a) = (m - a) + a = 0$ for any $a \in Z_m$.
6. multiplication is closed, i.e., for any $a, b \in Z_m$, $ab \in Z_m$
7. multiplication is commutative, i.e., for any $a, b \in Z_m$ $ab = ba$
8. multiplication is associative, i.e., for any $a, b, c \in Z_m$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
9. a multiplicative identity, i.e., for any $a \in Z_m$, $a \times 1 = 1 \times a = a$.
10. multiplication distributes over addition ,i.e., for any $a, b, c \in Z_m$, $(a + b) \cdot c = (ac) + (bc)$
and $a(b+c) = ab + ac$.

Properties 1, 3-5 say that Z_m forms an algebraic structure called a group with respect to the addition. Since property 4 also holds, the group is said to be abelian.

Properties 1-10 establish that Z_m is, in fact, a ring. Some familiar examples of rings include the integers Z ; the real numbers R ; and the complex numbers C . However, these are all infinite rings, and our attention will be confined almost exclusively to finite rings.

Since additive inverses exist in Z_m , we can also subtract elements in Z_m . We defined $a-b$ in Z_m to be $a + m - b \pmod m$. Equivalently, we can compute the integer $a-b$ and then reduce it modulo m . For example, to compute $11-18$ in Z_{31} , we can evaluate $11+13 \pmod{31} = 24$. Alternatively, we can first subtract 18 from 11, obtaining -7 and then compute $-7 \pmod{31} = 24$.

CRYPTO SYSTEMS

Shift Cipher

Let $P = C = K = \mathbb{Z}_{26}$. For $0 \leq 25$, define

$$e_k(x) = x + K \pmod{26} \quad \text{and}$$

$$d_k(y) = y - K \pmod{26}$$

$$(x, y \in \mathbb{Z}_{26}).$$

It is defined over \mathbb{Z}_{26} since there are 26 letters in the English alphabet though it could be defined over \mathbb{Z}_m for any modulus m . It is easy to see that **Shift Cipher** forms a cryptosystem as defined above, i.e., $d_k(e_k(x)) = x$ for every $x \in \mathbb{Z}_{26}$.

REMARK For the particular key $K=3$, the cryptosystem is often called the Caesar Cipher, which was purportedly used by Julius Caesar.

We would use the Shift Cipher (with a modulus of 26) to encrypt ordinary English text by setting up a correspondence between alphabetic and residues modulus 26 as follows: $A \leftrightarrow 0, B \leftrightarrow 1, \dots, Z \leftrightarrow 25$.

The same is recorded here for future reference:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

EXAMPLE :

Suppose the key for a Shift Cipher is $K=11$, and the plaintext is

wewillmeetatmidnight

We first convert the plaintext to a sequence of integers using the specified correspondence, obtaining the following :

22 4 22 8 11 11 12 4 4 19
 0 19 12 8 3 13 8 6 7 19

Next, we add 11 to each value, reducing each sum modulo 26:

7 15 7 19 22 22 23 15 15 4

11 4 23 19 14 24 19 17 18 4

Finally, we convert sequence of integers to alphabetic characters, obtaining the ciphertext:

HPHTWWXPPELEXTOYTRES.

To decrypt the ciphertext, Bob will first convert the ciphertext to a sequence of integers, then subtract 11 from each value (reducing modulo 26), and finally convert the sequence of integers to alphabetic characters. In the above example upper case letters are used for ciphertext and lower case letters for plaintext, in order to improve readability only.

If a cryptosystem is to be of practical use, it should satisfy certain properties. We enumerate two of these properties now.

1. Each encryption function e_k and each decryption d_k should be efficiently computable.
2. An opponent, upon seeing a ciphertext string y , should be unable to determine the key k that was used, or the plaintext string x .

The second property above is defining, in a very vague way, the idea of “security”. The process of attempting to compute the key K , given a string of ciphertext y , is called cryptanalysis. Note that, if Oscar can determine K , then he can decrypt y just as Bob would, using d_k . Hence, determining K is at least as determining the plaintext string x .

EXAMPLE

Given the ciphertext string :

JBCRCLQRWCRVNBJENBWRWN,

We successively try the decryption keys d_0, d_1 etc. The following is obtained:

j b c r c l q r w c r v n b j e n b w r w n
i a b q b k p q v b q u m a i d m a v q v m
h z a p a j o p u a p t l z h c l z u p u l
g y z o z i n o t z o s k y g b k y t o t k
f x y n y h m n s y n r j x f a j x s n s j
e w x m x g l m . x m q i w e z i w r m r i
d v w l w f k l q w l p h v d y h v q l q h
c u v k v e j k p v k o g u c x g u p k p g
b t u j u d i j o u j n f t b w f t o j o f
a s t i t c h i n t l m e s a v e s n i n e

At this point, we have determined the plaintext and we can stop. The key is $K=9$.

Substitution Cipher

Substitution cryptosystem has been used for hundreds of years. In case of substitution cipher we take P and C both to be the 26-letter English alphabet. We use Z_{26} in the Shift cipher because encryption and decryption are algebraic operation. But in the **substitution cipher** it is more convenient to operate on alphabetical characters for encryption and decryption.

A key for substitution cipher just consists of a permutation of 26 alphabetic characters. The number of these permutations is $26!$, which is more than 4×10^{26} . Thus an exhaustive key search without computers is infeasible.

Example

Let $P = C = Z_{26}$. K consists of all possible permutations of the 26

symbols $0, 1, \dots, 25$. For each permutation $\pi \in K$, define

$$e_{\pi}(x) = \pi(x)$$

and define

$$d_{\pi}(y) = \pi^{-1}(y),$$

where π^{-1} is the inverse permutation to π .

As the above example indicates, a necessary condition for the cryptosystem to be secure is that an exhaustive key search should be infeasible, i.e., the keyspace to be very large. But a large keyspace may also not guarantee security.

Vigenere Cipher

In both the Shift cipher and the substitution cipher, once a key is chosen each alphabetic character is mapped to a unique alphabetic character, hence they are called monoalphabetic. Vigenere cipher system is a polyalphabetic cipher. It uses the encipherment table called the **Vigenere square**. In this table English alphabets are written in a row and English alphabets are shifted by one place in each row, 26 such rows are written one over the other, with the 'natural' alphabet as an extra column on the left.

Each letter in the extra column determines a row of the square, while each row represents an additive cipher. Each of these substitution alphabet is then used, in the order of the sequence, to encipher a single letter. In the most common usage, called the **Vigenere cipher**, the method of obtaining the sequence involves choosing a keyword (or phrase). If the plain text message is longer than the keyword then the sequence is obtained by repeating it as many times as necessary. Thus the period is the length of keyword. We can also replace each row of the Vignere square by any substitution alphabet and then use the new square as discussed above. There is one particular such square known as **beaufort square**, in which alphabet are written in the reverse order

PRACTICAL SECURITY

In order to assess the practical security of a system one must have an idea of the resources which are likely to be available at the disposal of the expected cryptanalyst. In particular one needs to know the computing power available to him. When trying to determine the practical security of a system, one must determine the number of operations or storage elements needed to break it and then decide if it provides enough cover time for the message to be secure. It is also of importance to note that the number of operations required does depend on the efficiency of the method of attack. Hence the cryptanalyst always seeks ways of reducing the number of operations. Even when testing one key every microsecond, to solve a monoalphabetic cipher by trying every key would take about 1.12×10^{13} years. In practice, therefore, a cryptanalyst must try to find a method which does not entail trying every key, but eliminates many possibilities at a time.

Diffusion and Confusion

The cryptographer uses the techniques which are called **diffusion** and **confusion**. The idea behind diffusion is to 'spread' the statistics of the message space into a statistical structure which involves long combinations of the letters in the cryptogram. This is similar to the concept of source coding.

Confusion is to make the relation between the cryptogram and the corresponding key a complex one. This aims to make it difficult for statistics to pinpoint the key as having come from any particular area of the key space. In particular, it tries to ensure that the majority of the key is needed to obtain even very short cryptograms, which implies that every message character enciphered will depend on virtually the entire key. Hence forcing the cryptanalyst to find the whole

key simultaneously by making him solve considerably more complex equation than when he was able to find the key piece by piece.

Shannon's Five Criteria

Shannon made five suggestions in 1940 and since then technology has advanced considerably. His suggested criteria were :

- (a) the amount of secrecy offered,
- (b) the size of the key,
- (c) the simplicity of the enciphering and deciphering operation,
- (d) the propagation of error and,
- (e) extension of the message.

Criteria (a) need not any discussion as the importance is obvious. The key must be kept secret and, on occasion, may need to be memorised. Consequently it should be as small as possible. Enciphering and deciphering as per Shannon should of course be as simple as possible. If they are done manually, complexity leads to loss of time, errors etc. If done mechanically, complexity leads to a large, expensive machines. With some cipher systems, one error occurring on a transmission can mean that, when the cryptogram is deciphered, whole portion, or even the complete message is garbled. For most communication the error propagation should clearly be minimised. In some cipher system the size of message increases by the enciphering process. For instance use of nulls (i.e. adding meaningless characters to swamp the message statistics) causes a larger cryptogram than message. Such a message extension is undesirable for most communication systems.

There appears to be a certain incompatibility between the requirement of each of these five criteria when our message space consists of a natural language. It is probably not possible to satisfy all five but, if one is dropped, it may be possible to satisfy the other four. The first criterion cannot be dropped for obvious reasons: English.

If we drop (e) and allow unlimited message extension, then we can encipher many extra messages and use part of the key to indicate the correct one, still it is not sure that requirements of (b) and (c) will be met. By dropping (d) we can use a block cipher. But again it is not clear that requirements for either (b) or (c) can be achieved, though block cipher may lead to error propagation. Third criteria of Shannon need not be discussed with the present day electronics available to carry out enciphering and deciphering operations. Propagation of error is necessarily not a bad thing at times, the effects of error propagation prove to be advantageous and at times totally unacceptable as it corrupts the message.

Worst Case Conditions

In order to assess the security of system following three assumptions are made, which are referred to as **worst case conditions**.

- C1** The cryptanalyst has a complete knowledge of the cipher system.
- C2** The cryptanalyst has obtained a considerable amount of cipher text.
- C3** The cryptanalyst knows plaintext equivalent of certain amount of the ciphertext.

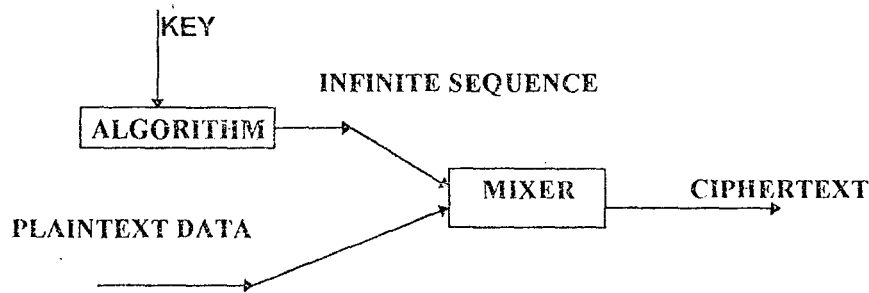
In any given situation we will attempt to quantify realistically what we mean by **considerable** and **certain**. This will depend on particular system under consideration.

Condition **C1** implies that there is no security in the cipher system itself and that all security must come from the key. Naturally the cryptanalyst's task is considerably harder if he does not know the system used and it is now possible to conceal this information to certain extent. It should be clear that **C2** is a necessarily assumption which, in conjunction with **C1** has found the basis of many of our earlier cryptanalytic attacks. It has to be assumed that if a cryptanalyst can intercept communication between two parties, he is likely to be able to intercept others, who may have implied the same key. **C3** is the basis of the **known plaintext attack** which is probably the most important and most commonly used method of breaking cipher. In this case the cryptanalyst has, possibly by guess work, deduction, or even by planting some one in some way, obtained knowledge of some of the plaintext message prior to its encipherment. The cryptanalyst may, for instance, know that all communication between two sources begin with a particular name and address or a particular phrase or expression.

Stream Cipher Systems

So far discussed cipher systems could be implemented in a reasonably short time, either by hand or by using a mechanical or electro-mechanical machine. The most significant factors in the development of the design of cipher systems were the advent of the computer and then, in the 1960s, the expanding use of microelectronics. They meant that a whole new range of functions were available to the cryptographer. But they also compelled him to increase his mathematical knowledge. Many of these new functions could only be expressed in terms of a mathematical language which was considerably more advanced than any of the mathematical knowledge previously required. The development of the cipher system greatly influenced by the fact that Shannon had proved the one-time-pad to be unbreakable. Many cryptographers felt that, if they could emulate the one-time-pad system in some way, they would have a system with a guaranteed high security level. They were also encouraged by the fact that, since the 1930s, many of the mechanical and electro-mechanical machines had operated in a way similar to the one-time-pad; in the sense that they produced long sequences of displacements which were applied, character by character, to the plaintext message. However there is one fundamental difference. Unlike the situation for the one-time-pad, a sequence produced by one of these machines is not random; in fact it is completely determined by the key. Once the key has been set up, the sequence, although certainly as long as the message, is completely predetermined. Nevertheless, by careful choice of the algorithm, it was possible to produce a sequence which appeared to be random, i.e. a sequence in which there was no telegraphers that such a system would be highly secure.

The above ideas led to the introduction of the stream cipher , illustrated in Figure.



A Stream Cipher

Thus a stream cipher is a system in which the key is fed to an algorithm which uses the key to generate an infinite sequence. (The algorithm is usually referred to as the **sequence generator** or **keystream generator**.) In all practical cases of cipher systems, the algorithm is an example of a finite state machine.

It is important to realize that a stream cipher attempts to utilize confusion, but not diffusion. This gives it a major advantage over a block cipher ; namely that it is not error propagating. For this reason, stream ciphers provide probably the most important method of modern encipherment , since the majority of such systems employ electronic techniques, both the plaintext and the infinite sequence use a character set which has only two possibilities corresponding to **on** and **off**. For convenience these are labeled 1 and 0 and the resulting system is called a **binary system**.

One way of viewing our stream cipher system is a polyalphabetic cipher whose periodicity is governed by the sequence which the algorithm produces. But it is important to realize that, although the sequence has infinite length, this does not mean that the polyalphabetic cipher cannot have finite period. The infinite sequence may have the property that it is merely numerous repetitions of a finite sequence. If this occurs then we say that it is periodic. We call the shortest

repeated sequence a cycle and the length of a cycle is the **period** of the infinite sequence. If we represent the sequence $S_1 S_2 S_3 \dots$ by (S_1) , then if (S_1) has period P we know S_{m+p} for every m . If the period of our output sequence is small, the system will have the same type of drawbacks as the Vignere cipher with short keyboard. It is essential for security that our output sequence should have a large period and that the period should, as an absolute minimum, be at least as long as any message to be enciphered. For this reason, we will need theorems which tell us when the output sequence has a guaranteed minimum period. A second requirement for our output sequence, again based on our experience in attempting to cryptanalyze Vignette - type ciphers, is that it should appear to be random and, thus, not allow the cryptanalyst to use any known statistical analysis of the language of the system. Thus our two main aims are :

A1 The input key stream sequence must have a guaranteed minimum length for its period. (we will then only encipher messages which are shorter than this value.)

A2 The ciphertext must appear to be random.

TH-7231



DATA ENCRYPTION STANDARD

The Data Encryption Standard (DES) specifies an algorithm to be implemented in electronic hardware devices and used for the cryptographic protection of computer data. This publication provides a Complete description of a mathematical algorithm for encrypting and decrypting binary coded information. Encrypting data converts it to an unintelligible form called cipher. Decrypting - cipher converts the data back to its original form. The algorithm described in this standard specifies both enciphering and deciphering operations which are based on a binary number called a key. The key consist of 64 binary digits (0's or 1's) of which 56 bits are used directly by the algorithm and 8 bits are used for error detection.

Binary coded data may be cryptographically protected using the DES algorithm in conjunction with a key. The key is generated in such a way that each of the 56 bits used directly by the algorithm are random and the 8 error detecting bits are set to make the parity of each 8 -bit byte of the key odd, i.e., there is an odd number of `1` s in each 8-bit byte. Each member of a group in common, is used to decipher the data received in cipher form from other members of the group. The encryption algorithm specified in this standard is commonly known among those using the standard. The unique key chosen for use in a particular application makes the results of encrypting data using the algorithm unique. Selection of a different key cause the cipher that is produced for any given set of inputs to be different. The cryptographic security of the data depends on the security provided for the key used to encipher and decipher the data.

Data can be recovered from cipher only by using exactly the same key used to encipher it. Unauthorized recipients of the cipher who know the algorithm but do not have the correct key cannot derive the original data algorithmically. However, anyone who does have the key and the algorithm can easily decipher the cipher and obtain the original data. A standard algorithm based on a secure key thus provides a basis for exchanging encrypted computer data by issuing the key used to encipher it to those authorized to have the data.

DATA ENCRYPTION ALGORITHM

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64 bit key. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation IP , then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation IP^{-1} . The key-dependent computation can be simply defined in terms of function f , called the cipher function, and a function KS , called the key schedule. A description of the computation is given in terms of primitive functions which are called the selection function f is given in terms of primitive functions which are called the selection functions $S1$ and the permutation function P .

Enciphering

A sketch of the ciphering computation is given in figure.

The 64 bits of the input block to be enciphered are first subjected to the following permutation, called the initial permutation IP:

IP

```

58 50 42 34 26 18 10 2
60 52 44 36 28 20 12 4
62 54 46 38 30 22 14 6
64 56 48 40 32 24 16 8
57 49 41 33 25 17 9 1
59 51 43 35 27 10 11 3
61 53 45 37 29 21 13 5
63 55 47 39 31 23 15 7

```

That is the permuted input has bit 58 of the input as its first bit, bit 50 as its second bit, and so on with bit 7 as its last bit. The permuted input block is then the input to a complex key-dependent computation described below. The output of that computation, called the pre-output, is then subjected to the following permutation which is the inverse of the initial permutation:

IP⁻¹

```

40 8 48 16 56 24 64 32
39 7 47 15 55 23 63 31
38 6 46 14 54 22 62 30
37 5 45 13 53 21 61 29
36 4 44 12 52 20 60 28
35 3 43 11 51 19 59 27
34 2 42 10 50 18 58 26
33 1 41 9 49 17 57 25

```

That is, the output of the algorithm has bit 40 of the pre-output block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the pre-output block is the last bit of the output.

Let the 64 bits of the input block to an iteration consist of a 32-bit block L , followed by a 32-bit block R . Using the notation defined earlier, the input block is then LR .

Let K be a block of 48 bits chosen from the 64-bit key. Then the output $L'R'$ of an iteration with input LR is defined by:

$$(1) \quad L' = R$$

$$R' = L \oplus f(R, K), \text{ where } \oplus \text{ denotes bit-by-bit addition modulo 2.}$$

The input of the first iteration of the calculation is the permuted input block. If $L'R'$ is the output of the 16th iteration then $R'L'$ is the preoutput block. At each iteration a different block K of key bits is chosen from the 64 bit key designated by KEY .

Let KS be a function which takes an integer n in the range from 1 to 16 and a 64-bit block KEY as input and yields an output a 48-bit block k_n which is a permuted selection of bits from KEY .

That is

$$(2) \quad K_n = KS(n, KEY)$$

with k_n determined by the bits in 48 distinct bit positions of KEY . KS is called the key schedule because the block K used in the n 'th iteration of (1) is the block k_n determined by (2).

Let the permuted input block be LR . Finally, let L_0 and R_0 be respectively L and R and L_n and R_n respectively L' and R' of (1) when L and R are respectively L_{n-1} and R_{n-1} and K is K_n ; that is, when n is in the range from 1 to 16,

$$(3) \quad L_n = R_{n-1}$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

The preoutput block is then $R_{16}L_{16}$.

The key schedule KS of the algorithm is described in detail in the Appendix. The key schedule produces the 16 K_n which are required for the algorithm.

Deciphering

The permutation IP^{-1} applied to the preoutput block is the inverse of the initial permutation IP applied to the input. Further, from (1) it follows that :

$$(4) \quad R=L'$$

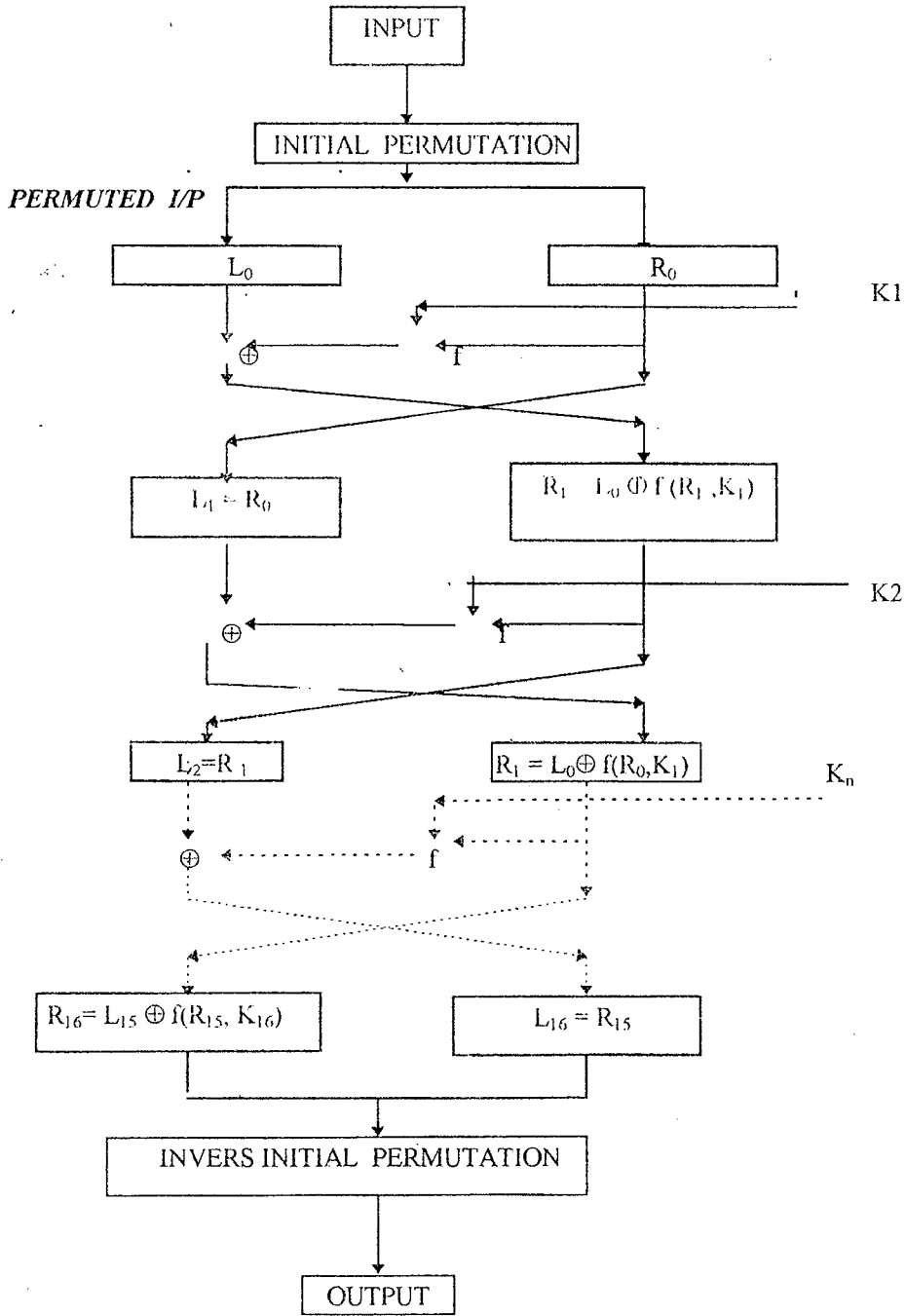
$$L = R' \oplus f(L', K)$$

Consequently, to decipher it is only necessary to apply the very same algorithm to an enciphered message block, taking that at each iteration of the computation the same block of key bit K is used during decipherment as was used during the encipherment of the block. This can be expressed as

$$(5) \quad R_{n-1} = L_n$$

$$L_{n-1} = R_n \oplus f(L_{n-1}, K_n)$$

where now $R_{16} L_{16}$ is the permuted input block for the deciphering calculation and $L_0 R_0$ is a preoutput block. That is, for the decipherment calculation with $R_{16} L_{16}$ as the permuted input, K_{16} is used in first iteration, K_{15} in the second and so on, with the K_1 used in the 16th iteration.



DES Enciphering Computation

KEY MANAGEMENT

KEY STRUCTURE

Security of a cipher system depends solely on the choice of the key. This point was highlighted when we saw worst case conditions for the cryptographer. There as we asserted that the cryptographer must be prepared for the cryptanalyst to have full details of the entire system and, in particular, completely understand the algorithm. Given the importance of the key, it is obviously crucial that we pay great attention to the problems of choosing and changing keys. We must also consider how the encipherer can tell the receiver the fact that he is changing to a new key.

On discussing practical, realistic systems which means that they are theoretically breakable. This must be accepted as a premise. Our problem is to determine how long it would take to break a particular system and/or to estimate the cost involved. If the time or expense is excessive then, for all practical purposes, we may regard our system as secure. This means we must attempt to quantify the words **excessive** in the last sentence. For any given system we must determine the number of operations or storage elements needed to break the key, and then decide if this is large enough for our purposes. This of course is one of the problem with today's sophisticated systems, the solutions is by know means simple. In order to increase the number of operation needed by cryptanalyst, we must increase the number of possibilities for the key. But this implies increasing the actual size of the key. Unfortunately, as we increase the size of our key, it may make it more difficult to ensure secure distribution and quick, accurate entry. There are many practical situations where these latter constraints are very important, and on these occasion, the smallest key is advantageous.

As soon as we find two conflicting requirements, it is therefore necessary to find the compromise. We should keep our keys as small as possible, while still ensuring that nobody can try every possibility in a reasonable time interval.

Need For Key Change

Assume that we have selected a key of x bits which initializes the algorithm, i.e. gives the algorithm a starting point, and then use this key to encipher n different messages m_1, m_2, \dots, m_n . If we let $|m_i|$ denote the number of characters in the message m_i , then, for any j satisfying $1 \leq j \leq |m_i|$, we will let a_{ij} represent the j^{th} ciphertext character of m_i . In order to encipher m_1 we first enter the key. The algorithm, using the key as an initializing process, then produces a pseudo-random sequence which determines the sequence of transformations. Similarly for each subsequent message, we enter the same key and, as a consequence, obtain the same pseudo-random sequence and the same sequence of transformations.

If an interceptor obtains n enciphered messages he will be able to arrange them in the kind of array and from any given columns he knows the same section of the enciphering sequence has been used. Thus he can restrict his attention to the columns in turn. But if we let m_{ik} be the k^{th} plaintext letter of m_i , then $m_{ik} = m_{jk}$ if and only if $a_{ik} = a_{jk}$, and this means that the system used for any given column is merely a monoalphabetic substitution. Furthermore it is essentially additive which, of course, guarantees that knowing the ciphertext equivalent of one character determines the entire substitution.

The amount of use which the cryptanalyst can make of this observation depends on the size of n . If n is large then clearly he can utilize it, but for small n it is not at present clear how much it helps him. Certainly if $n=1$ then the cryptanalyst has learnt nothing. If n is greater than 1, however, he has further information. Since each column comes from a substitution which is essentially additive and each row represents a message, he can attempt to break each column and use the extra fact that each row must be meaningful to settle any ambiguous positions. The precise way of doing this depends on n . If n is large he can use the statistics relating to the frequency of occurrence of each letter to determine the most likely substitutions for each column. The fact that each must be meaningful should soon enable him to make firm decisions about any columns where the frequencies leave any room for chance. But if n is small then there will not be sufficient characters in each column for the frequency statistics to be reliable. In this case he might find it advantageous to concentrate on the rows and use them to try to determine the substitution alphabets.

The value of information depends on the size of n . But for large values on n it can reduce the number of trials considerably.

Key Distribution

Key distribution is defined as a mechanism by which one party chooses a secret key and transmits it to other party(s).

we have a insecure network of n users. In some of the schemes there is a **trusted authority (TA)** that is responsible for key distribution and key agreement by verifying the identity of users at both the ends. since the network is insecure we have to guard against the passive adversary whose action are restricted upto eavesdropping on messages that are transmitted over the channel. we also have to guard against the active adversary who can do following kind of things :

- (a) alter message that he observes being transmitted over the network
- (b) save message for the reuse at a latter time
- (c) attempt to masquerade as various users in the network.

The aim of the active adversary may be the any of the following:

- (a) to fool U and V into accepting an “invalid” key as valid . It may be one of the old key that has expired.
- (b) to make U and V believe that they have exchanged a key with other when they have not.

Key Predistribution

In key predistribution the TA generates $\binom{n}{2}$ keys, and gives each key to a pair of user in the network of n users. To transmit these keys from TA to user we require a secure channel. Though the number of channel required to transmit the keys has reduced but, if n is large the problem remains same.

Thus it is of importance to reduce the amount of information to be passed and stored, while still allowing the pair of user U and V to be able to (independently) compute a secret key $K_{u,v}$.

Blom's Scheme

Let us suppose the network is of n users and the keys are chosen from a finite field Z_p where $p \geq n$ is prime. Let k be an integer , $1 \leq k \leq n-2$. The value k is the largest size coalition against

which the system will remain secure. In the **Blom's scheme**, the TA will transmit $k+1$ elements of Z_p to each user over a secure channel. Each pair of user U, V will be able to compute a key $K_{V,U} = K_{U,V}$. The security condition is as follows: any set of at most k users disjoint from $\{U, V\}$ must be unable to determine information about $K_{U,V}$.

Blom's key distribution scheme for $k = 1$ is as follows:

(a) A prime number p is made public, and for each user U , an element $r_u \in Z_p$ is made public.

The element r_u must be distinct.

(b) The TA chooses three random elements $a, b, c \in Z_p$ (not necessarily distinct), and forms the polynomial.

$$f(x,y) = a + b(x + y) + cxy \pmod{p}.$$

(c) For each user U , the TA computes the polynomial

$$g_u(x) = f(x, r_u) \pmod{p}$$

and transmits $g_u(x)$ to U over a secure channel. Since $g_u(x)$ is a linear polynomial in x , it can be written as

$$g_u = a_u + b_u x,$$

where

$$a_u = a + br_u \pmod{p}$$

and

$$b_u = b + cr_u \pmod{p}$$

(d) If U and V wants to communicate , then they use the common key

$$K_{u,v} = K_{v,u} = f(r_u, r_v) = a + b(r_u + r_v) + cr_u r_v \pmod{p},$$

where U computes $K_{u,v}$ as

$$f(r_u, r_v) = g_u(r_v)$$

and V computes $K_{v,u}$ as

$$f(r_v, r_u) = g_v(r_u).$$

Diffie -Hellman Key Predistribution

The scheme is described over Zp , where p is prime, though it can be implemented in any finite group in which the discrete logarithm problem is intractable. It is assumed that α is a primitive element of Zp , and that the values p and α are publicly known to everybody in the network.

In the scheme $ID(U)$ will denote certain identification information for each user U in the network e.g., his name ,address etc. Also each user U has a secret exponent address etc. Also each user U has a secret exponent a_u (where $0 \leq a_u \leq p-2$) , and the corresponding public value

$$b_u = \alpha^{a_u} \pmod{p}$$

The TA will have a scheme with a public verification algorithm ver_{TA} and a secret signing algorithm sig_{TA} . Finally, we will implicitly assume that all information is hashed, using a public hash function, before it is signed.

Certain information pertaining to a user U will be authenticated by means of a *certificate* which is issued and signed by TA. Each user will have a certificate

$$C(U) = (ID(U), b_u, sig_{TA}(ID(U), b_u)),$$

where b_u is formed as described as earlier and, TA does not know the value of a_u . A certificate for the user U will be issued when U joins the network. Certificate can be stored in a public database, or each user can store his or her own certificate. The signature of the TA on a certificate allows any one in the network to verify the information it contains.

U and V can compute the common key $K_{u,v} = \alpha^{a_u a_v} \bmod p$ as follows:

(a) A prime p and a primitive element $\alpha \in Z_p$ are made public.

(b) V computes

$$K_{u,v} = \alpha^{a_u a_v} \bmod p = b_u^{a_v} \bmod p,$$

using the public value b_u from the U 's certificate, together with his own secret value a_v .

(c) U computes

$$k_{u,v} = \alpha^{a_u a_v} \bmod p = b_v^{a_u} \bmod p,$$

using the public b_v from V 's certificate, together with his own secret value a_u .

Giving a thought about the security of the system in the presence of a passive or active adversary.

The signature of the TA on users' certificates, effectively prevents W from altering any information on some one else's certificate i.e. the system is secure against active attacks. Hence we need worry only about passive attacks.

So the question arises ,can a user W compute $K_{u,v}$, if $W \neq U, V$? In other words, given $\alpha^u \bmod p$ and $\alpha^v \bmod p$ (but not a_u and a_v) , is it possible to compute $\alpha^{uv} \bmod p$? This problem is called the **Diffie - Hellman problem**. It is clear that Diffie - Hellman Key Predistribution is secure against a passive adversary if and only if the problem is intractable.

Kerberos

In the key distribution systems ,each pair of user computes its one fixed key. If the same key is used for a long time, the fear remains that it may be compromised. Thus it is often preferable to use an on-line key distribution, where a new session key is produced every time a pair of user wants to communicate (this property is called key freshness).

If on-line key distribution is used, there is no need for any user to store keys to communicate with other user (each user share a key with TA however). Session key will be transmitted on request by the TA. It is the responsibility of the TA to ensure the key freshness.

When U wants to establish communication with V ,he sends a request to TA for a session key , TA will generate a new random session key K . Also, the TA will record the time at which the request is made as a *timestamp* , T , and specifies the , *lifetime*, L , during which K will be valid. That is, session key K is to be regarded valid from time T to time $T + L$. All this information is crypted and transmitted to U and (eventually) to V . The protocol of transmission of session key using Kerberos is as follows:

- (a) U asks the TA for a session key to communicate with V .
- (b) The TA chooses a random session key K , a timetamp T , a lifetime L .

(c) The TA computes

$$m_1 = e_{K_u} (K , ID(V) , T , L)$$

and

$$m_2 = e_{K_v} (K , ID(U) , T , L)$$

and sends m_1 and m_2 to U.

(d) U uses the decryption function d_{K_u} to compute K , T , L and $ID(V)$ from m_1 . He then computes

$$m_3 = e_K (ID(U) , T)$$

and sends m_3 to V along with the message m_2 he recieved from TA.

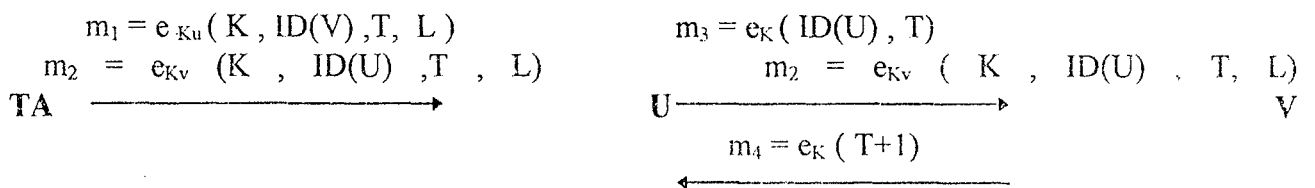
(e) V uses the decryption function d_{K_v} to compute K , T , L and $ID(U)$ from m_2 . He then uses d_K to compute T and $ID(U)$ from m_3 . He checks that two values of T and $ID(U)$ are the same. If so then V computes

$$m_4 = e_K (T+1)$$

and send it to U.

(f) U decrypts m_4 using d_K and verifies that the result is $T + 1$.

The information transmitted in the protocol is illustrated in the following diagram:



Though there is no formal proof that **Kerberos** is **secure** against an active adversary, but on a close look at the features of the Kerberos deduction can be drawn.

As mentioned above, the TA generates K , T , and L in step (b). In step (c), this information along with $ID(V)$, is enciphered using the Key K_u shared by U and TA to form m_1 . Also K , T , L and $ID(U)$ are encrypted using the key K_v shared by TA and V to form m_2 . Both these encrypted messages are sent to U .

U can use his key to decrypt m_1 , and thus obtain K , T , and L . He can verify that the current time is in the interval of T to $T + L$. He can also check that the session key K has been issued for his desired communicant V by verifying the information $ID(V)$ decrypted from m_1 .

Next, U will relay m_2 to V . As well U will use the new session key K to encrypt T and $ID(U)$ and send the resulting message m_3 to V .

When V receives m_2 and m_3 from U , he decrypts m_2 to obtain T , K , L and $ID(U)$. Then he uses the new session key K to decrypt m_3 and he verifies that T and $ID(U)$, as decrypted from m_2 and m_3 , are the same. This ensures V that the session key encrypted within m_2 is the same key that was used to encrypt m_3 . Then V uses Key to encrypt $T + 1$, and sends the result back to U as message m_4 .

The message m_1 and m_2 are used to provide secrecy in the transmission of the session key K . On the other hand, m_3 and m_4 are used to key confirmation, that is, to enable U and V to convince each other that they possess the same session key K .

The purpose of the *timestamp* T and *lifetime* L is to prevent an active adversary from storing old messages for retransmission at a later stage (this is called a replay attack). This method works because keys are not accepted as valid once as they have expired.

One of the drawback of Kerberos is that all the user in the network should have synchronised clocks, since the current is used to determine if a given session key is valid or not. In practice, it is very difficult to provide perfect synchronism, so some amount of variation in time must be allowed.

Key Exchange

If one does not want to have on-line key server, then he is forced to use key exchange and key agreement protocols to exchange secret keys to keep the flow of information between two subscribers without any adversary to have access of the key or the information. The first and the best known key agreement protocol is Diffie-Hellman Key Exchange.

Diffie-Hellman Key Exchange:

Let us suppose that p is prime, α is a primitive element of Z_p , and that the value p and α are publicly known. Alternatively, they should be chosen by U and communicated to V in the first step of the protocol. **Diffie-Hellman Key Exchange** protocol is as follows:

- (a) U chooses a_u at random, $0 \leq a_u \leq p - 2$.
- (b) U computes $\alpha^{a_u} \bmod p$ and sends it to V .
- (c) V chooses a_v at random, $0 \leq a_v \leq p - 2$.
- (d) V computes $\alpha^{a_v} \bmod p$ and send it to U .

(c) U computes $K = (\alpha^{av})^{au} \bmod p$ and

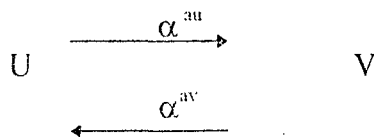
V computes $K = (\alpha^{au})^{av} \bmod p$.

(f) At the end U and V both will compute the same key: $K = \alpha^{au} \alpha^{av} \bmod p$.

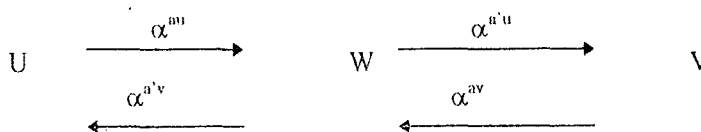
This protocol is very similar to **Diffie-Hellman Key predistribution**. The difference is that the exponents a_u and a_v of user U and V are chosen a new each time the protocol is run, instead of being fixed. Also in this protocol, both U and V are assured of key freshness, since the session key depends on both random exponents a_u and a_v .

The Station-to-station Protocol

Diffie-Hellman Key Exchange is supposed to look like this;



Unfortunately, the protocol is vulnerable to an active adversary who uses an intruder-in-the-middle attack. An intruder-in-the-middle attack on the Diffie-Hellman Key Exchange protocol works in the same way. We will intercept messages between U and V and substitute his own messages, as indicated in the following diagram :



At the end of the protocol, U has actually established the secret key α^{ua^v} with W, and V has established a secret key α^{va^u} with W. When U tries to encrypt a message to send to V, W will be able to decrypt it but V will not. (A similar situation hold if V sends a message to U.)

Hence, it is essential for U and V to make sure that they are exchanging messages with each other and not with W. Before exchanging key, U and V might carry out a separate protocol to establish each other's identity, for example by using one of the identification schemes. But this offers no protection against an intruder - in-the -middle attack if W simply remains inactive until after U and V have proved their identities to each other. Hence, the key agreement protocol should itself authenticate the participants' identities at the same time as the key is being established. Such a protocol will be called *authenticated key agreement*.

Station-to-station protocol assumes a publicly known prime p and a primitive element α , and it makes use of certificates. Each user U will have a signature scheme with verification algorithm ver_u and signing algorithm sig_u . The TA also has a signature scheme with public verification algorithm ver_{TA} . Each user too has a certificate

$$C(U) = (ID(U), ver_u, sig_{TA}(ID(U), ver_u)),$$

where $ID(U)$ is identification information of U.

The **Station-to-Station protocol** (or STS) is as follows:

- (a) U chooses random number a_u , $0 \leq a_u \leq p-2$.

(b) U computes

$$\alpha^{au} \bmod p$$

and sends it to V.

(c) V chooses random number a_v , $0 \leq a_v \leq p-2$.

(d) V computes $\alpha^{av} \bmod p$, then he computes

$$K = (\alpha^{au})^{a_v} \bmod p$$

and

$$y_v = \text{sig}_v(\alpha^{av}, \alpha^{au}).$$

(e) V sends $(C(V), \alpha^{av}, y_v)$ to U.

(f) U computes

$$K = (\alpha^{av})^{au} \bmod p$$

He verifies y_v using ver_v and he verifies $C(V)$ using ver_{TA} .

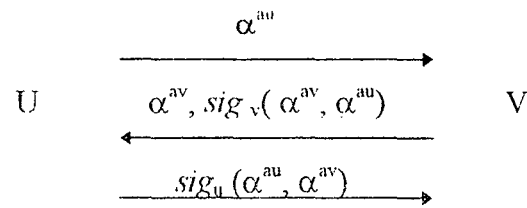
(g) U computes

$$y_u = \text{sig}_u(\alpha^{au}, \alpha^{av})$$

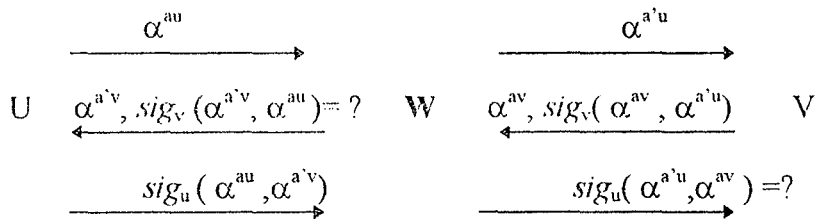
and he sends $(C(U), y_u)$ to V.

(h) V verifies y_u using ver_u and he verifies $C(U)$ using ver_{TA} .

The information exchanged in the simplified STS protocol (excluding certificates) is illustrated as follows:



Let's see how secure this is against intruder-in-the-middle-attack. As before, W will intercept α^{au} and replace it with $\alpha^{a'u}$. W then receives $\alpha^{av}, sig_v(\alpha^{av}, \alpha^{a'u})$ from V. He would like to replace α^{av} with $\alpha^{a'v}$, as before. However, this means that he must also replace $sig_v(\alpha^{av}, \alpha^{a'u})$ by $sig_v(\alpha^{a'v}, \alpha^{a'u})$. Unfortunately for W, he can not compute V's signature on $(\alpha^{a'v}, \alpha^{a'u})$ since he does not know V's signing algorithm sig_v . Similarly W is unable of replacing $sig_u(\alpha^{au}, \alpha^{a'v})$ by $sig_u(\alpha^{a'u}, \alpha^{a'v})$ because he does not know U's signing algorithm. This is illustrated in the following figure:



It is the use of signature that thwarts the intruder - in-the -middle attack. The above described protocol does not provide key confirmation but, same can be incorporated by modifying the protocol by defining step (d) as

$$y_v = e_k(sig_v(\alpha^{av}, \alpha^{au}))$$

and defining

$$y_u = e_k(sig_u(\alpha^{au}, \alpha^{av}))$$

in step (f). The resulting protocol is known as the **Station - to -station protocol**.

MTI Key Agreement Protocol

Matsumoto, Takashima, and Imai have constructed several interesting key agreement protocols by modifying **Diffie-Hellman Key Exchange**. These protocols are known as **MTI** protocols, do not require that U and V compute any signature. They are *two pass protocols* since there are only two separate transmissions of information performed (one from U to V and other from V to U).

The setting for this protocol is same as for **Diffie-Hellman Key Predistribution**. We assume a publicly known prime p and a primitive element α . Each user U has an ID string, $ID(U)$, a secret exponent a_u ($0 \leq a_u \leq p-2$), and a corresponding public value

$$b_u = \alpha^{a_u} \text{ mod } p$$

The TA has a signature scheme with a (public) verification algorithm ver_{TA} and a secret signing algorithm sig_{TA} .

Each user U have a certificate

$$C(U) = (ID(U), b_u \text{ sig}_{TA}(ID(U), b_u)),$$

where b_u is formed as described above.

The MTI key agreement is as follows:

(a) U chooses r_u at random, $0 \leq r_u \leq p-2$, and computes

$$s_u = \alpha^{r_u} \text{ mod } p$$

(b) U sends $(C(U), s_u)$ to V.

In this situation, U and V will compute different keys: U will compute

$$K = \alpha^{ru av + r'v au} \bmod p,$$

while V will compute

$$K = \alpha^{r'u av + r'v au} \bmod p.$$

However, neither of the key computation can be carried out by W, since it requires knowledge of secret exponents a_u and a_v , respectively. So even though U and V have computed different keys, which are of no use to them, neither of those keys can be computed by W (assuming the problem of the Discrete Log problem). Hence U and V are assured that the other is the only user in the network that could compute the key that they have computed. This property is also called *implicit key authentication*.

Conclusion

The study of cipher system is one of the most rapidly expanding modern sciences. The revolution in microchip technology has resulted in increasing use of electronic means of data communication and a corresponding need for security in the transmission network. The encryption system becomes debited to the development of electronics ,because of which the requirement of the cryptology has increased a many fold during last few years across the globe, other than the military requirement.

Another consequence of these recent developments is that the cost of cipher system has been sustainable reduced. This in turn, implies that the encryption devices are open to the man on the street apart from the government official use. As the number of cipher system increase, it is also necessary to develop new techniques.

In cryptography it is not enough to merely to decide over a good algorithm only, or the properties of the transmission medium are adequate, or the way the user want to employ the system is reasonable; the system is not complete unless we consider every aspect of the system or system as a whole. If a single detail changes the designer may need to change the whole system. While designing a cryptology system one has to take a look from not only the encipherer's decrypter's angle but also from the view of the passive and active adversary also to ensure that his system is guarded against any interception at least during the time the information should remain concealed.

BIBLIOGRAPHY

1. APPLIED CRYPTOGRAPHY, BY BRUCE SCHNEIER , 1996
PUBLICATION, ISBN 0-471-59756-2.
2. CRYPTOGRAPHY AND SECURE COMMUNICATION, BY MAN
YOUNG RHEE, ISBN 0-77-112502-7.
3. CRYPTOGRAPHY AND DATA SECURITY , BY DENNING ,
JANUARY 1983 PUBLICATION, ISBN 0-201-10150-5.
4. CRYPTOGRAPHY THEORY AND PRACTICE , BY DOUGLAS R.
STINSON , 1996 PUBLICATION.
5. CIPHER SYSTEMS, THE PROTECTION OF COMMUNICATIONS,
BY HENRY BEKER AND FRED PIPER,, 1982 PUBLICATION,
ISBN 7198 - 2611X.
6. COMPUTER NETWORKS, BY ANDREW S. TANENBAUM,
SECOND EDITION, MAY 1996, ISBN 81-203-0621-X.