Cryptography: is probably the most important aspect of communications security and is becoming increasingly important as a basic building block for computer security.

The increased use of computer and communications systems by industry has increased the risk of theft of proprietary information. Although these threats may require a variety of countermeasures, encryption is a primary method of protecting valuable electronic information.

By far the most important automated tool for network and communications security is encryption. Two forms of encryption are in common use: conventional, or symmetric encryption and public-key, or asymmetric, encryption.

Cryptology is the science and study of systems for secret communications. It consists of two complementary fields of study: **Cryptography,** the design of secret communications systems, and **Cryptanalysis**, the study of ways to compromise of secret communications systems.

Cryptology primarily has been applied in military and diplomatic communications systems, but other significant applications are becoming apparent.

Cryptography methods applied by authorized information sharers to design and develop encryption schemes in order to ensure confidentiality of information.

Crypt-Analysis (mathematical and statistical attempts by unauthorized persons to break cipher in order to reveal the meaning of the underlying protected data).

Cryptography:

It is the study of mathematical techniques related to aspects of information security such as:-

- **Confidentiality** is the concealment of information or resources.
- **Authenticity** is the identification and assurance of the origin of information.
- **Integrity** refers to the trustworthiness of data or resources in terms of preventing improper and unauthorized changes.
- **Availability** refers to the ability to use the information or resource desired.
- **Non-repudiation** It implies that one party of a transaction cannot deny having received a transaction, nor can the other party deny having sent a transaction.

Classification of Cryptography

- Number of keys used
  - Hash functions: no key
  - Symmetric encryption (Private Key): one key
  - Asymmetric encryption (Public Key): two keys - public, private
- Type of encryption operations used
  - substitution / transposition / product
- Way in which plaintext is processed
  - block / stream

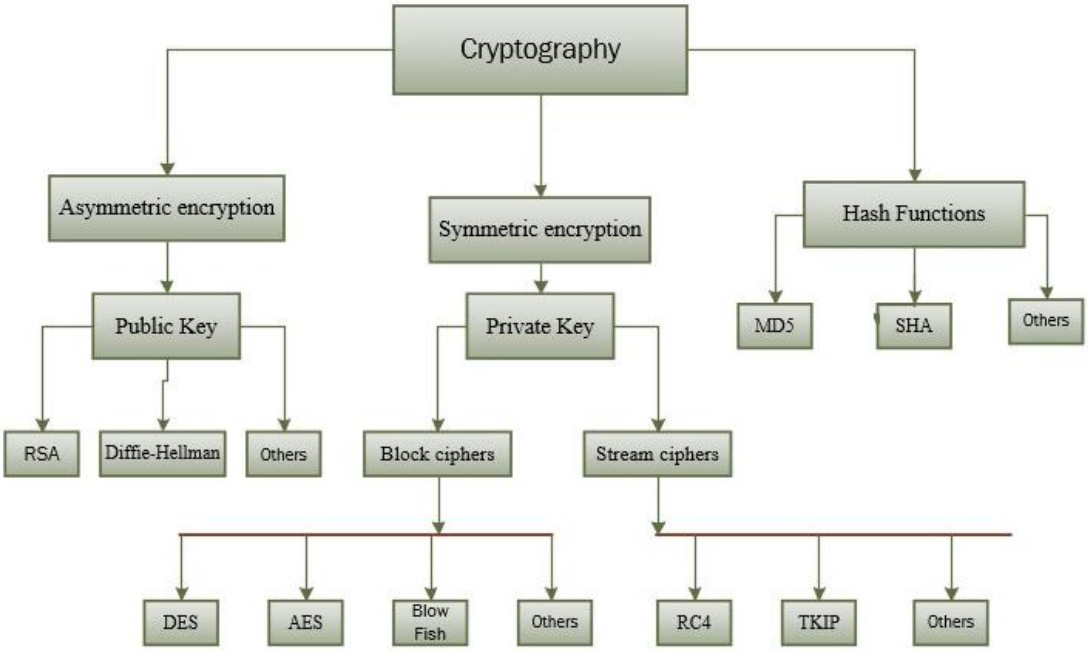


Figure 1-1 Schematic representation of cryptographic cipher classification

Symmetric Ciphers

In Symmetric cryptography ciphers the enciphering and deciphering keys are the same, as shown in figure 1-2:

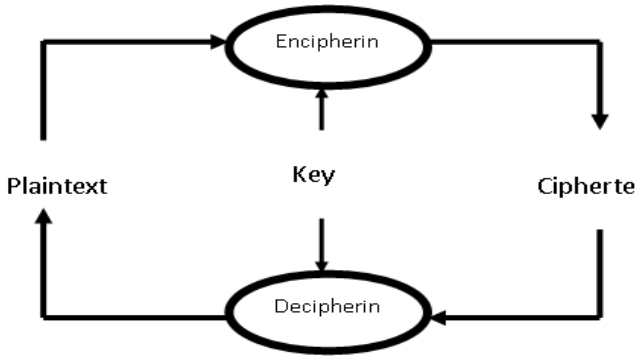


Fig. 1-2: Secret Writing

Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.

Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

A Symmetric cryptography ciphers may in fact be sub classified into:

1. **Block Cipher**: processes the input one block of elements at a time, producing an output block for each input block.
2. **Stream Cipher:** processes that encrypt a digital data stream one bit or one byte at a time.

The process of transforming plaintext into ciphertext is called **Encryption**; the reverse process of transforming ciphertext into plaintext is called **Decryption**.

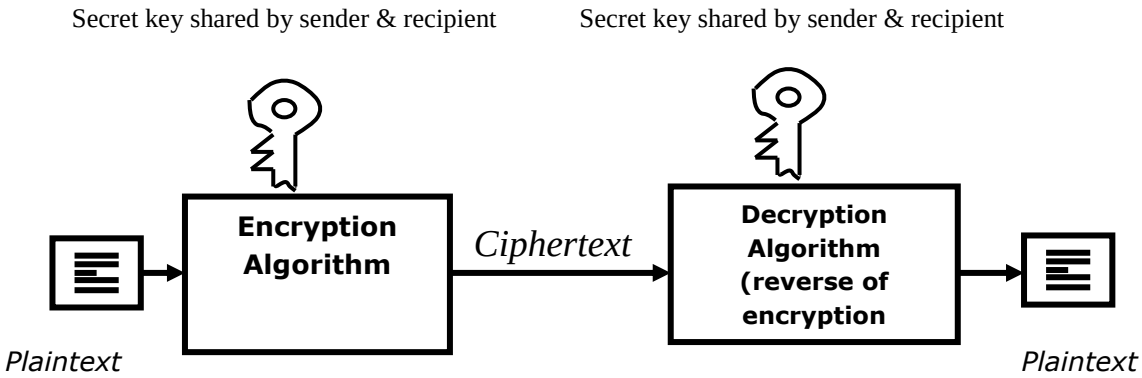


Figure 1-3: Simplified Model of Conventional Encryption

Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.

Decryption algorithm: This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

Note: Enciphering

Plaintext → small letter

Key → capital letter

Cipher text → capital letter

Deciphering

Cipher text → capital letter

Key → small letter

Plain text → small letter

Block: A sequence of consecutive characters encoded at one time.

Block length: The number of characters in a block.

An algorithm for performing encryption (and the reverse, decryption): a series of well-defined steps that can be followed as a procedure. It works at the level of individual letters, or small groups of letters.

Cryptosystem: The package of all processes, formulae, and instructions for encoding and decoding messages using cryptography

Digram: Sequence of two consecutive characters

Key: A relatively small amount of information that is used by an algorithm to customize the transformation of plaintext into ciphertext (during encryption) or vice versa (during decryption)

Key length: The size of the key - how many values comprise the key

Monoalphabetic: Using one alphabet - refers to a cryptosystem where each alphabetic character is mapped to a unique alphabetic character

Polyalphabetic: Using many alphabets - refers to a cipher where each alphabetic character can be mapped to one of many possible alphabetic characters

Trigram: Sequence of three consecutive characters unigram

A model for much of what we will be discussing is captured, in very general terms, in figure 1-4. A message is to be transferred from one party to another across some sort of internet. The two parties, who are the *principals* in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

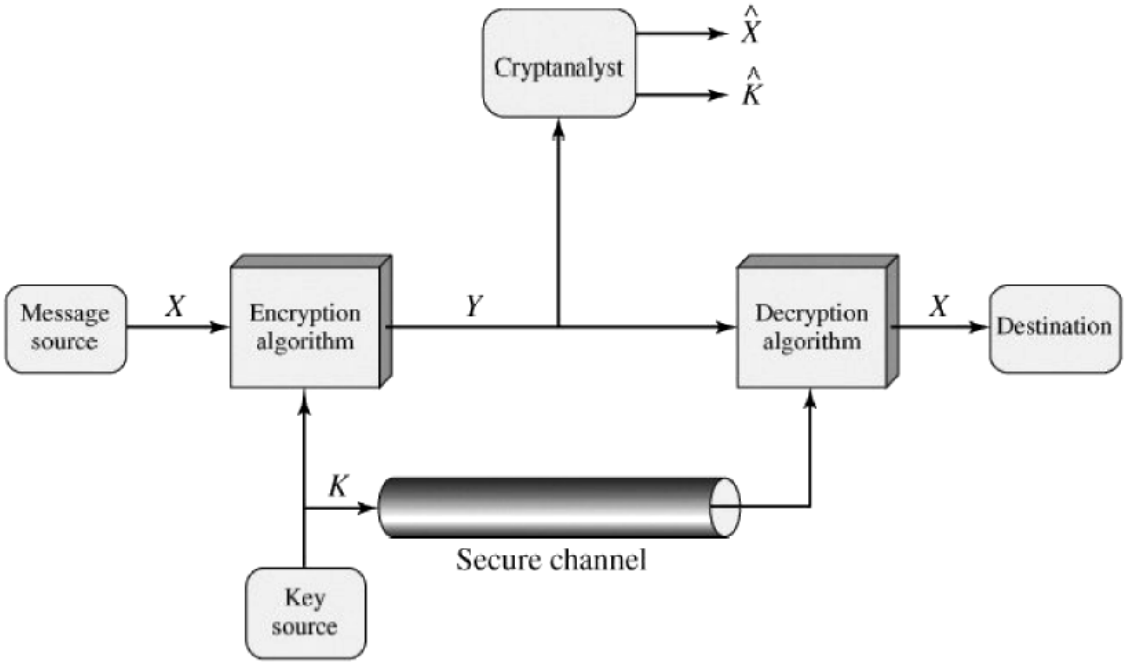


Figure 1-4: Model for Network Security

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender, some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

Part Two discusses a form of encryption, known as public-key encryption, in which only one of the two principals needs to have the secret information.

Secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of the secret information.

4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

However, there are other security-related situations of interest that do not neatly fit this model. A general model of these other situations is illustrated by Figure 1-5, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. Or, the intruder can be a disgruntled employee who wishes to do damage, or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

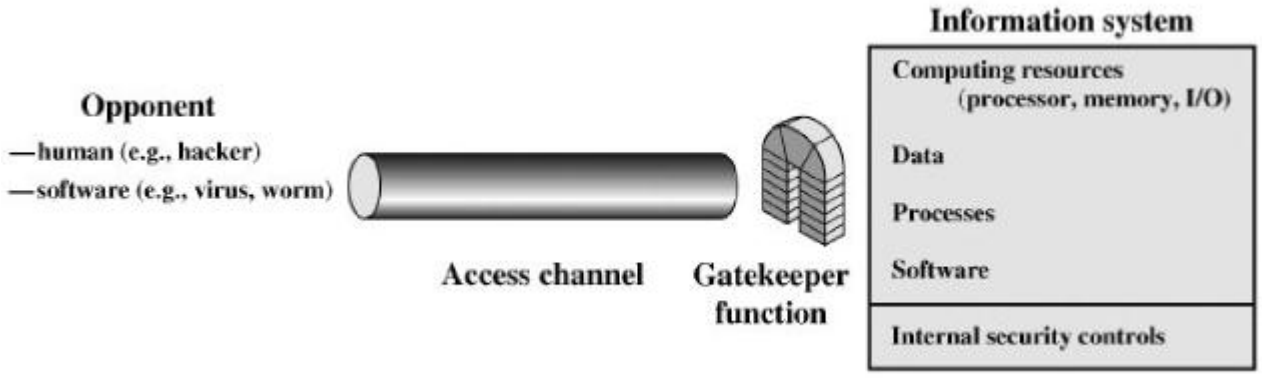


Figure 1-6: Network Access Security Model

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers. Programs can present two kinds of threats:

Information access threats intercept or modify data on behalf of users who should not have access to that data.

Service threats exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

The security mechanisms needed to cope with unwanted access fall into two broad categories:

1. The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access.
2. The second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

Caesar cipher

The earliest known use of a substitution cipher, and the simplest, was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example:

Plain: meet me after the toga party

Cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

Plain: a b c d e f g h i j k l m n o p q r s t u v w x y z
Cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Let us assign a numerical equivalent to each letter:

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

Then the algorithm can be expressed as follows. For each plaintext letter *p*, substitute the ciphertext letter *C*:

We define *a* mod *n* to be the remainder when *a* is divided by *n*. For example, 11 mod 7 = 4

$$C = E(3, p) = (p + 3) \bmod 26$$

A shift may be of any amount, so that the general Caesar algorithm is

$$C = E(k, p) = (p + k) \bmod 26$$

where *k* takes on a value in the range 1 to 25. The decryption algorithm is simply

$$p = D(k, C) = (C - k) \bmod 26$$

Example: encrypt the following message using Caesar cipher:

Ship equipment will reach on the fourth of July

$C = (p + k) \bmod 26$

c(1) = 18+3 mod 26 = 21

c(2) = 7+3 mod 26 =10

c(3) = 8+3 mod 26 = 11

. . .

c(39) = 24+3 mod 26 = 1

cipertexe: VKLSHTXLSPHQWZLOOUHDFKRQWKHIRXUWKRIMXOB

Example: decrypt the following message (**NHBZRUG**) using Caesar cipher:

$p = (C - k) \bmod 26$

p_1= (13-3) mod 26=10➔k

p_2= (7-3) mod 26=4➔e

p_3= (1-3) mod 26=24➔y

p_4= (25-3) mod 26=22➔w

p_5= (17-3) mod 26=14➔o

p_6= (20-3) mod 26=17➔r

p_7= (6-3) mod 26=3➔d then the plaintext is **keyword**

The two basic building blocks of all encryption techniques are substitution and transposition:

Substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns. There are a number of different types of substitution cipher:

1. **Monoalphabetic**: Using one alphabet - refers to a cryptosystem where each alphabetic character is mapped to a unique alphabetic character.
2. **Polyalphabetic**: Using many alphabets - refers to a cipher where each alphabetic character can be mapped to one of many possible alphabetic characters.

Cryptosystem: The package of all processes, formulae, and instructions for encoding and decoding messages using cryptography.

Transposition technique is a very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters.

Simple Monoalphabetic Substitution Cipher

The simple substitution cipher is a cipher that has been in use for many hundreds of years. It basically consists of substituting every plaintext character for a different ciphertext character. It differs from the Caesar cipher in that the cipher alphabet is not simply the alphabet shifted, it is completely jumbled.

Example : Encrypt the following text: "**defend the east wall of the castle**".

Keys for the simple substitution cipher usually consist of 26 letters (compared to the caeser cipher's single number). An example key is:

Plain Text Character:-

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

Cipher Text Character (key cipher)

P	H	Q	G	I	U	M	E	A	Y	L	N	O	F	D	X	J	K	R	C	V	S	T	Z	W	B

An example encryption using the above key:

Plaintext: defend the east wall of the castle

Ciphertext: GIUIFGCEIIPRCTPNNDUCEIQPRCNI

Example 2:

Plain Text Character:-

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

Cipher Text Character (key cipher):-

H	W	U	G	C	T	V	A	E	K	D	Y	Q	P	B	R	J	L	F	I	X	M	S	O	Z	N

Let the Plaintext = "dulce et decorm est pro patria mori"

Now the cipher text is (GXYUCCIGCUBLXQCFIRLBRHILEHQBLE)

Note: - Remove all spaces, special characters in plain text

Hill cipher

The Hill cipher uses matrix multiplication, mod 26. In particular, the encryption key is an nxn matrix with an inverse mod 26, where n is the block size. System can be described as follows:

C = KP mod 26

For **example**, we will illustrate the cipher with n=2. Consider the following key:

$$\begin{pmatrix} 3 & 1 \\ 6 & 5 \end{pmatrix}$$

To encrypt a plaintext, group the plaintext in pairs: "**math**", for **example**. Convert each letter to its numerical equivalent, mod 26, and write it in a nx1 matrix as follows:

$\begin{pmatrix} 12 \\ 0 \end{pmatrix}$ stands for "**ma**"

Now, multiply the encryption key by the plaintext and reduce mod 26 to get the ciphertext:

$\begin{pmatrix} 3 & 1 \\ 6 & 5 \end{pmatrix}\begin{pmatrix} 12 \\ 0 \end{pmatrix} \bmod 26 = \begin{pmatrix} 36 \\ 72 \end{pmatrix} \bmod 26 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$, which corresponds to the ciphertext **KU**.

Here is the encryption of "**th**":

$\begin{pmatrix} 3 & 1 \\ 6 & 5 \end{pmatrix}\begin{pmatrix} 19 \\ 7 \end{pmatrix} \bmod 26 = \begin{pmatrix} 64 \\ 149 \end{pmatrix} \bmod 26 = \begin{pmatrix} 12 \\ 19 \end{pmatrix}$, which corresponds to the ciphertext **MT**.

Ciphertext: **KUMT**

Example: consider the plaintext "**paymoremoney**" and use the encryption key:

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The first three letters of the plaintext are represented by the vector $\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix} = \text{LNS. Continuing in this fashion,}$$

The **ciphertext** for the entire plaintext is: **LNS HDL EWM TRW**.

Decryption:

Decryption requires using the inverse of the matrix K. The inverse K^{-1} of a matrix K is defined by the equation $KK^{-1} = K^{-1}K = I$, where I is the matrix that is all zeros except for ones along the main diagonal from upper left to lower right. The inverse of a matrix does not always exist, but when it does, it satisfies the preceding equation.

P = K^{-1} C mod 26

To find K^{-1} it needs to use a bit of math. It turns out that K^{-1} above can be calculated from our key. The important things to know are inverses (mod m), *determinants* of matrices and matrix *adjugates*.

Let *K* be the key matrix. Let *d* be the determinant of K. We wish to find K^{-1} (the inverse of K), such that $K \times K^{-1} = I \pmod{26}$, where I is the identity matrix. The following formula tells us how to find K^{-1} given K:

$$K^{-1} = d^{-1} \times adj(K)$$

where $d \times d^{-1} = 1 \pmod{26}$, and adj(K) is the adjugate matrix of K.

The determinant (d) is calculated normally for K as follows:

The determinant of a 2 × 2 matrix is defined by:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

The determinant of a 3 × 3 matrix is defined by:

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
$$= aei + bfg + cdh - ceg - bdi - afh.$$

Determinant	1	3	5	7	9	11	15	17	19	21	23	25
Reciprocal Modulo 26	1	9	21	15	3	19	7	23	11	5	17	25

The adjugate is the transpose of its cofactor matrix, it is calculated as follows:

The adjugate of the 2×2 matrix

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$
$$\operatorname{adj}(\mathbf{A}) = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

The adjugate of the 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}$$

where

$$\begin{vmatrix} a_{im} & a_{in} \\ a_{jm} & a_{jn} \end{vmatrix} = \det \begin{pmatrix} a_{im} & a_{in} \\ a_{jm} & a_{jn} \end{pmatrix}.$$

Its adjugate is the transpose of its cofactor matrix:

$$\operatorname{adj}(\mathbf{A}) = \mathbf{C}^{\mathsf{T}} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}$$

Once K^{-1} is found, decryption can be performed.

Example: **decrypt** the following ciphertext " **KUMT**" if you know it is encrypted using Hill cipher by key

$$\begin{pmatrix} 3 & 1 \\ 6 & 5 \end{pmatrix}$$

Solution:

d= 3*5 –(1*6) = 9

$d^{-1} = 3$

$\text{adj(k)} = \begin{pmatrix} 5 & -1 \\ -6 & 3 \end{pmatrix}$

$k^{-1} = d^{-1} * \text{adj(k)} = \begin{pmatrix} 15 & 23 \\ 8 & 9 \end{pmatrix}$

Now, we can corroborate that this is the case by decrypting the example above.

$$\begin{pmatrix}15 & 23\\ 8 & 9\end{pmatrix}\begin{pmatrix}10\\ 20\end{pmatrix}\bmod 26=\begin{pmatrix}610\\ 260\end{pmatrix}\bmod 26=\begin{pmatrix}12\\ 0\end{pmatrix}$$

$$\begin{pmatrix}15 & 23\\ 8 & 9\end{pmatrix}\begin{pmatrix}12\\ 19\end{pmatrix}\bmod 26=\begin{pmatrix}617\\ 267\end{pmatrix}\bmod 26=\begin{pmatrix}19\\ 7\end{pmatrix}$$

Plaintext: math

We can also verify this by multiplying both matrices in question together:

$$(K*K^{-1})\ mod\ 26=\begin{bmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}$$

$$\begin{pmatrix}3 & 1\\ 6 & 5\end{pmatrix}\begin{pmatrix}15 & 23\\ 8 & 9\end{pmatrix}\bmod 26=\begin{pmatrix}53 & 78\\ 130 & 183\end{pmatrix}\bmod 26=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

Example of decryption 3x3 key: decrypt the following ciphertext "**LNS HDL EWM TRW** " if you know it is encrypted using Hill cipher key

$$\mathbf{K}=\begin{pmatrix}17 & 17 & 5\\ 21 & 18 & 21\\ 2 & 2 & 19\end{pmatrix}$$

Solution:

$p=k^{-1}*C \bmod 26$

d= 17*(18*19-21*2) - 17*(21*19-21*2) + 5*(21*2-18*2)= -939 mod 26 =23

$(23)^{-1}$=**17**

The adjugate of the 3×3 matrix and then transpose is

$$\left(\begin{bmatrix}300 & -313 & 267\\ -357 & 313 & -252\\ 6 & 0 & -51\end{bmatrix}*17\right)\bmod 26=\begin{bmatrix}\mathbf{4} & \mathbf{9} & \mathbf{15}\\ \mathbf{15} & \mathbf{17} & \mathbf{6}\\ \mathbf{24} & \mathbf{0} & \mathbf{17}\end{bmatrix}\text{ this is } K^{-1}$$

$$\begin{bmatrix}p1\\ p2\\ p3\end{bmatrix}=\begin{bmatrix}4 & 9 & 15\\ 15 & 17 & 6\\ 24 & 0 & 17\end{bmatrix}*\begin{bmatrix}11\\ 13\\ 18\end{bmatrix}\ mod\ 26$$

p1= (4*11+9*13+15*18) mod 26 =15 ➔ p

p2= (15*11+17*13+6*18) mod 26 =0 ➔ a

p1= (24*11+0*13+17*18) mod 26 =24 ➔ y

And repeat the same function to the next three letters until end the ciphertext

Then plaintext is **pay mor emo ney.**

Example: Suppose that the plaintext "**friday**" is encrypted using a 2 x 2 Hill cipher to yield the ciphertext PQCFKU. Find the key?

Solution:

$$\mathbf{K}\begin{pmatrix}5\\17\end{pmatrix} \bmod 26 = \begin{pmatrix}15\\16\end{pmatrix}; \mathbf{K}\begin{pmatrix}8\\3\end{pmatrix} \bmod 26 = \begin{pmatrix}2\\5\end{pmatrix}; \quad \text{and} \quad \mathbf{K}\begin{pmatrix}0\\24\end{pmatrix} \bmod 26 = \begin{pmatrix}10\\20\end{pmatrix}$$

Using the first two plaintext-ciphertext pairs, we have

$$\begin{pmatrix}15 & 2\\16 & 5\end{pmatrix} = \mathbf{K}\begin{pmatrix}5 & 8\\17 & 3\end{pmatrix} \bmod 26$$

The inverse of matrix can be computed:

$$\begin{pmatrix}5 & 8\\17 & 3\end{pmatrix}^{-1} = \begin{pmatrix}9 & 2\\1 & 15\end{pmatrix}$$

So:

$$\mathbf{K} = \begin{pmatrix}15 & 2\\16 & 5\end{pmatrix}\begin{pmatrix}9 & 2\\1 & 15\end{pmatrix} = \begin{pmatrix}137 & 60\\149 & 107\end{pmatrix} \bmod 26 = \begin{pmatrix}7 & 8\\19 & 3\end{pmatrix}$$

This result is verified by testing the remaining plaintext-ciphertext pair.

Polyalphabetic Ciphers

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**. All these techniques have the following features in common:

1. A set of related monoalphabetic substitution rules is used.

2. A key determines which particular rule is chosen for a given transformation.

Vigenère cipher

The best known polyalphabetic substitution cipher, and one of the simplest, such algorithm is referred to as the ***Vigenère cipher***. In this scheme, the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers, with shifts of 0 through 25. Each cipher is denoted by a key letter.

To aid in understanding the scheme and to aid in its use, a matrix known as the **Vigenère tableau** is constructed. Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of encryption is simple: Given a key letter *x* and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled *x* and the column labeled y; in this case the ciphertext is V.

For example to encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. **For example**, if the keyword is **deceptive**, the message "**we are discovered save yourself**" is encrypted as follows:

```
key:            deceptivedeceptivedeceptive
plaintext:      wearediscoveredsaveyourself
ciphertext:     ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

Plaintext																									
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	G
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Table below shows the cipher text in **Vigenere method**.

Table (1): Mapping Letters To Integers And Back

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

The encryption function seen in Equation 2.

$$E(k, m_i) = m_i + k \pmod{26} \quad .....(2)$$

Example: encipher the plaintext message: "TO BE OR NOT TO BE THAT IS THE QUESTION", using the keyword SUBSTITUTION?

C=p+k mod 26

Keyword:																											
S	**U**	**B**	**S**	**T**	**I**	**T**	**U**	**T**	**I**	**O**	**N**	**S**	**U**	**B**	**S**	**T**	**I**	**T**	**U**	**T**	**I**	**O**	**N**	**S**	**U**	**B**	**S**	**T**	**I**
As integer																											
18	**20**	**1**	**18**	**19**	**8**	**19**	**20**	**19**	**8**	**14**	**13**	**18**	**20**	**1**	**18**	**19**	**8**	**19**	**20**	**19**	**8**	**14**	**13**	**18**	**19**	**1**	**18**	**19**	**8**
Plaintext:																											
t	**o**	**b**	**e**	**o**	**r**	**n**	**o**	**t**	**t**	**o**	**b**	**e**	**t**	**h**	**a**	**t**	**i**	**s**	**t**	**h**	**e**	**q**	**U**	**e**	**s**	**t**	**i**	**o**	**n**
As integer																											
19	**14**	**1**	**4**	**14**	**17**	**13**	**14**	**19**	**19**	**14**	**1**	**4**	**19**	**7**	**0**	**19**	**8**	**18**	**19**	**7**	**4**	**16**	**20**	**4**	**18**	**19**	**8**	**14**	**13**
Addition																											
11	**8**	**2**	**22**	**7**	**25**	**6**	**8**	**12**	**1**	**2**	**14**	**23**	**13**	**8**	**18**	**12**	**16**	**11**	**13**	**0**	**12**	**4**	**7**	**23**	**12**	**20**	**0**	**7**	**21**
Cipher text:																											
L	**I**	**C**	**W**	**H**	**Z**	**G**	**I**	**M**	**B**	**C**	**O**	**W**	**N**	**I**	**S**	**M**	**Q**	**L**	**N**	**A**	**M**	**E**	**H**	**W**	**M**	**U**	**A**	**H**	**V**

Decryption is equally simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column.

Example: using vigenere cipher, encrypt and decrypt the word (instruction) with the key (key)

Encryption function: ***C=p+k mod 26***

plaintext	i	n	s	t	r	u	c	t	i	o	n
P	8	13	18	19	17	20	2	19	8	14	13
key	k	e	y	k	e	y	k	e	y	k	e
k	10	4	24	10	4	24	10	4	24	10	4
P+k	18	17	42	29	21	44	12	23	32	24	17
(P+k)mod26	18	17	16	3	21	18	12	23	6	24	17
ciphertext	S	R	Q	D	V	S	M	X	G	Y	R

Then use decryption function: ***p=C-k mod 26***

ciphertext	S	R	Q	D	V	S	M	X	G	Y	R
C	18	17	16	3	21	18	12	23	6	24	17
key	k	e	y	k	e	y	k	e	y	k	e
k	10	4	24	10	4	24	10	4	24	10	4
C-k	8	13	-8	-7	17	-6	2	19	-18	14	13
(C-k)mod26	8	13	18	19	17	20	2	19	8	14	13
plaintext	i	n	s	t	r	u	c	t	i	o	n

Transposition Techniques

A transposition is not a permutation of alphabet characters, but a permutation of places.

Transposition or Permutation cipher works by breaking a message into fixed size blocks, and then permuting the characters within each block according to a fixed permutation, say P. The key to the transposition cipher is simply the permutation P. So, the transposition cipher has the property that the encrypted message i.e. the cipher text contains all the characters that were in the plain text message. In the other word, the unigram statistics for the message are unchanged by the encryption process.

In this method, the message is written in a rectangle, **(arrange row by row. Reading the message off, row by row, but permuting the order of the columns)**. The order of the columns then becomes the key to the algorithm. **For example:**

The plaintext is: breaking transposition cipher.

Key: K = {4 1 3 5 7 6 2}

In this case, the message is broken into block of seven characters, and after encryption the fourth character in the block will be moved to position 1, the first is moved to position 2, the third remains in position 3, the fifth to position 4, the seventh to position five, the sixth remains in position 6, the two is moved to position 7.

The following Figure (A) shows the key and Fig. (B) Shows the encryption process of the previously described transposition cipher.

It can be noticed that the random string "X" was appended to the end of message to enforce a massage length, which is a multiple of the block size.

1	2	3	4	5	6	7
b	r	e	a	k	i	n
g	t	r	a	n	s	p
o	s	i	t	i	o	n
c	i	p	h	e	r	x

(A)

4	1	3	5	7	6	2
a	b	e	k	n	i	r
a	g	r	n	p	s	t
t	o	i	i	n	o	s
h	c	p	e	x	r	i

(B)

The ciphertext is: **ABEKNIRAGRNPSTTOIINOSHCPEXRI**

Decryption:

4	1	3	5	7	6	2
a	b	e	k	n	i	r
a	g	r	n	p	s	t
t	o	i	i	n	o	s
h	c	p	e	x	r	i

1	2	3	4	5	6	7
b	r	e	a	k	i	n
g	t	r	a	n	s	p
o	s	i	t	i	o	n
c	i	p	h	e	r	x

The plaintext is **breaking transposition cipher**

Playfair Cipher

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.

If the keyword is **monarchy** then:

M	**O**	**N**	**A**	**R**
C	**H**	**Y**	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that ***balloon*** would be treated as **ba lx lo on**.
2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, **ar** is encrypted as **RM**.
3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, **mu** is encrypted as **CM**.
4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, **hs** becomes **BP** and **ea** becomes **IM** (or **JM**, as the encipherer wishes).

Example: Encrypt the following message: "**Why, don't you**?" using the keyword: "**keyword**"

K	**E**	**Y**	**W**	**O**
R	**D**	A	B	C
F	G	H	I/ J	L
M	N	P	Q	S
T	U	V	X	Z

wh yd on ty ou

YJ EA ES VK EZ

Then Ciphertexet is : YJEAESVKEZ.

Example: Decrypt the message (PH GK CN HR FA NY) using playfair cipher with the key=encrypt.

Solution: convert the key to 5x5 matrix

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

PH➔ af

GK➔ fi

CN➔ ne

RH➔ ci

AF➔ ph

NY➔ er then the plaintext is **affine cipher**

Affine cipher

The affine cipher is a type of monoalphabetic substitution cipher. The 'key' for the Affine cipher consists of 2 numbers, we'll call them *a* and *b*, *a* should be chosen to be relatively prime to *m* (i.e. *a* should have no factors in common with *m*). For example 15 and 26 have no factors in common, so 15 is an acceptable value for *a*, however 12 and 26 have factors in common (e.g. 2) so 12 cannot be used for a value of *a*. When encrypting, we first convert all the letters to numbers ('a'=0, 'b'=1, ..., 'z'=25). The ciphertext letter *c*, for any given letter *p* is (remember *p* is the number representing a letter):

$$c = ap + b \pmod{m}, \quad 1 \leq a \leq m, \quad 1 \leq b \leq m$$

The decryption function is:

$$p = a^{-1}(c - b) \pmod{m}$$

where a^{-1} is the multiplicative inverse of *a* in the group of integers modulo m.

a	1	3	5	7	9	11	15	17	19	21	23	25
a^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

Example 1: encrypt the following message using affine cipher:

(Computer engineering) assume a=7, b=10.

c(1) = (2*7 +10) mod 26= 24

c(2) = (14*7 +10) mod 26= 4

…

c(19) = (6*7 +10) mod 26 = 0

ciphertext: YEQLUNMZMXAOXMMZOXA

Example 2: Encipher (war lost) Using an affine transformation with the ordinary alphabet. Use 7 as the multiplier, and 10 as the shift. Then recover the plaintext. The ordinary alphabet associations are shown in

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

The plaintext message, when the letters are converted to their numerical equivalents, and then compute the following:

$C = 7P + 10 = 7 \cdot 22 + 10 \equiv 8 \pmod{26}$
$C = 7P + 10 = 7 \cdot 0 + 10 \equiv 10 \pmod{26}$
$C = 7P + 10 = 7 \cdot 17 + 10 \equiv 25 \pmod{26}$
$C = 7P + 10 = 7 \cdot 11 + 10 \equiv 9 \pmod{26}$
$C = 7P + 10 = 7 \cdot 14 + 10 = 4 \pmod{26}$
$C = 7P + 10 = 7 \cdot 18 + 10 = 6 \pmod{26}$
$C = 7P + 10 = 7 \cdot 19 + 10 = 13 \pmod{26}$

The results of these calculations produce the ciphertext (in numbers) (8 10 25 9 4 6 13) or, the corresponding letters, (IKZJE GN)

To recover the plaintext, Since 7 is relatively prime to 26, an inverse of it exists modulo 26, and it can be found solving the congruence

$7x \equiv 1 \pmod{26}$ for x. Quick calculations using the extended Euclidean algorithm yield $x \equiv 15 \pmod{26}$.

This value for x is an inverse of 7 modulo 26, and this is easily verified:

$7x = 7(15) = 105 \pmod{26} = 1$.

Thus, to recover the plaintext from the ciphertext, we crank it through the deciphering transformations:

$p \equiv 15(c - 10) \equiv 15 \cdot (8 - 10) \equiv 15 \cdot (-2) \equiv 22 \pmod{26}$
$p \equiv 15(c - 10) \equiv 15 \cdot (10 - 10) \equiv 15 \cdot 0 \equiv 0 \pmod{26}$
$p \equiv 15(c - 10) \equiv 15 \cdot (25 - 10) \equiv 15 \cdot 15 \equiv 17 \pmod{26}$
$p \equiv 15(c - 10) \equiv 15 \cdot (9 - 10) \equiv 15 \cdot (-1) \equiv 11 \pmod{26}$
$p \equiv 15(c - 10) \equiv 15 \cdot (4 - 10) \equiv 15 \cdot (-6) \equiv 14 \pmod{26}$
$p \equiv 15(c - 10) \equiv 15 \cdot (6 - 10) \equiv 15 \cdot (-4) \equiv 18 \pmod{26}$
$p \equiv 15(c - 10) \equiv 15 \cdot (13 - 10) \equiv 15 \cdot 3 \equiv 19 \pmod{26}$

Which gives us 22 0 17 11 14 18 19 ⟶ war lost.

Example of encryption and deception process:

Encryption function:

$$Y = a\,x+b \mod 26 \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

The possible values that a could be are ***1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, and 25.***

Decryption function :

$$X = inv(a)\,(y-b) \mod 26 \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Example: the plaintext to be encrypted is "affine cipher" a=5 b=8

plaintext:	a	f	f	i	n	e	c	i	p	h	e	r
x:	0	5	5	8	13	4	2	8	15	7	4	17
5x+8	8	33	33	48	73	28	18	48	83	43	28	93
(5x+8)mod 26	8	7	7	22	21	2	18	22	5	17	2	15
cipher text:	I	H	H	W	V	C	S	W	F	R	C	P

The cipher to decryption:

ciphertext:	I	H	H	W	V	C	S	W	F	R	C	P
y:	8	7	7	22	21	2	18	22	5	17	2	15
21(y-8):	0	-21	-21	294	273	-126	210	294	-63	189	-126	147
(21(y-8)) mod26:	0	5	5	8	13	4	2	8	15	7	4	17
plaintext:	a	f	f	i	n	e	c	i	p	h	e	r

Example: Decrypt the message (SXGNDKL) using affine cipher with key is

(11, 5)

Decryption function is $p=m^{-1}*(c-k) \bmod 26$ $m=11, (11)-1=19$

P_1=19*(18-5) mod 26 = 14 ➔n

P_2=19*(23-5) mod 26 = 4 ➔e

P_3=19*(6-5) mod 26 = 19 ➔t

P_4=19*(13-5) mod 26 = 22 ➔w

P_5=19*(3-5) mod 26 = 14 ➔o

P_6=19*(10-5) mod 26 = 17 ➔r

P_7=19*(11-5) mod 26 = 10 ➔k

then the plaintext is **network**

One-Time Pad

- Developed by Gilbert Vernam in 1918, another name: ***Vernam Cipher***

- The key
  - a truly random sequence of 0's and 1's
  - the same length as the message
  - use one time only

- **The encryption**
  - adding the key to the message modulo 2, bit by bit.

$$c_i = m_i \oplus k_i \qquad i = 1,2,3,...$$

- **The Decryption**

$$m_i = c_i \oplus k_i \qquad i = 1,2,3,...$$

Where m_i : plain-text bits, k_i : key (key-stream) bits, and c_i : cipher-text bits.

- Example

- **Encryption:**
- 1001001 1000110 plaintext
- 1010110 0110001 key
- 0011111 1110110 ciphertext

- **Decryption:**
- 0011111 1110110 ciphertext
- 1010110 0110001 key
- 1001001 1000110 plaintext

One-Time pad practical Problem

- Key-stream should be as long as plain-text.
- Difficult in Key distribution & Management.

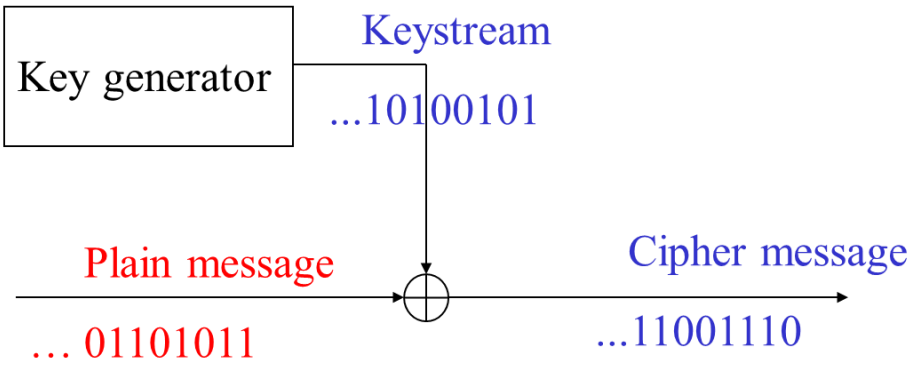
Solution :

- Stream Ciphers
- Key-stream is generated in pseudo-random fashion

Key generator

Idea: replace "random" with "pseudo-random"

- Use a Pseudo-Random Number Generator (PRNG)
- PRNG takes a short, truly random secret seed and expands it into a long "random-looking" sequence



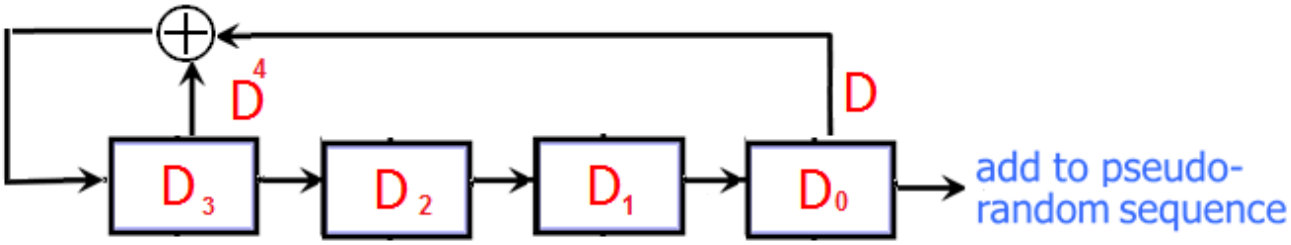
LFSR: Linear Feedback Shift Register

Maximal Period m-sequences

- The *tap sequence* defines the linear feedback function and is often regarded as a finite field polynomial.
- You have to choose the tap sequence very carefully.
  - Some choices provide a maximal length period.
  - These are *primitive polynomials*

In other word, if the sequence of LFSR has m register = 2^m -1 then the polynomial is primitive polynomial

Example: 4-bit LFSR polynomial $C(D)=1+D+D^4$,where the seed is 0110 :



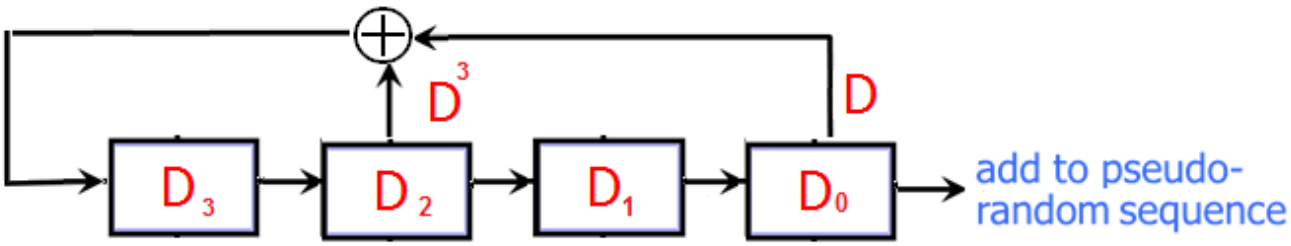
t	D_3	D_2	D_1	D_0
0	**0**	**1**	**1**	**0**
1	0	0	1	1
2	1	0	0	1
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0
7	1	1	0	0

t	D_3	D_2	D_1	D_0
8	1	1	1	0
9	1	1	1	1
10	0	1	1	1
11	1	0	1	1
12	0	1	0	1
13	1	0	1	0
14	1	1	0	1
15	**0**	**1**	**1**	**0**

Where polynomial $C(D)=1+D+D^4$ gives the maximal period sequence = $2^4 - 1 = 15$.

This polynomial is primitive polynomial

Example: 4-bit LFSR polynomial $C(D)=1+D+D^3$,where the seed is 0110 :



t	D_3	D_2	D_1	D_0
0	**0**	**1**	**1**	**0**
1	**1**	**0**	**1**	**1**
2	**1**	**1**	**0**	**1**
3	**0**	**1**	**1**	**0**

Where polynomial $C(D)=1+D+D^3$ does not give a maximal period sequence.

This polynomial not primitive polynomial

Cryptanalysis

Typically, the objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single ciphertext. There are two general approaches to attacking a conventional encryption scheme:

- Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.
- Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

There are basically two types of attack. One is on system and other is on data shown in Figure below.

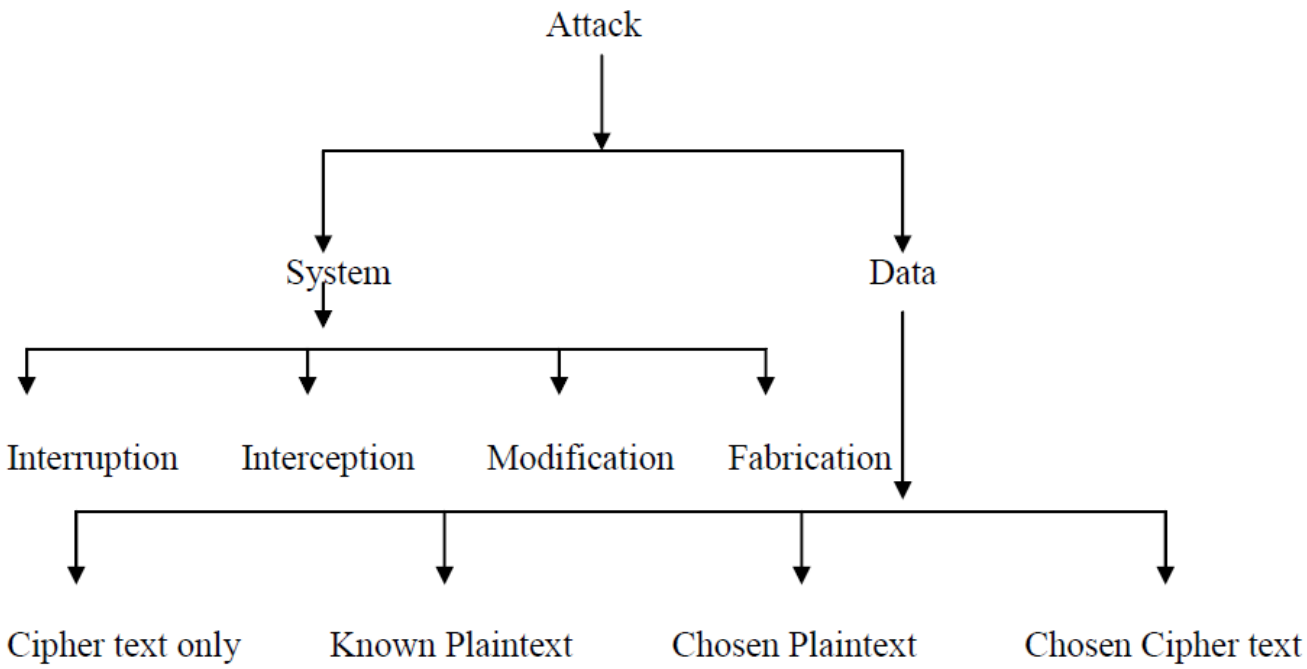


Figure 1-7 Classification of Attacks on Cryptography

System Attacks:

In general there is a flow of information from a source to a destination. The attacks which are on the flow of information are known as system attacks. The main security threats are listed below:

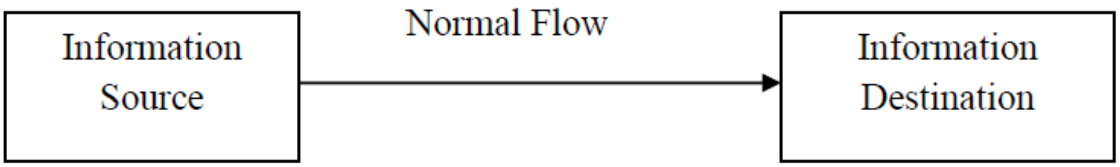


Figure 1-8 Normal Flow of Information

Interruption: It is an attack on availability of the resource. When the data flowing through source to destination becomes unavailable or unusable:

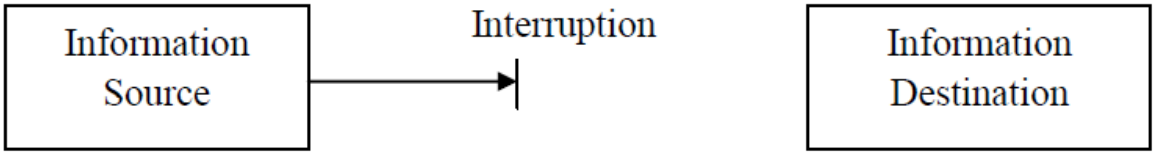


Figure 1-9 Interrupted Data Flow

Interception: It is an attack on the confidentiality of the system. In this attack an unauthorized party also has the access to a model. A person, program and a computer may be the unauthorized party :

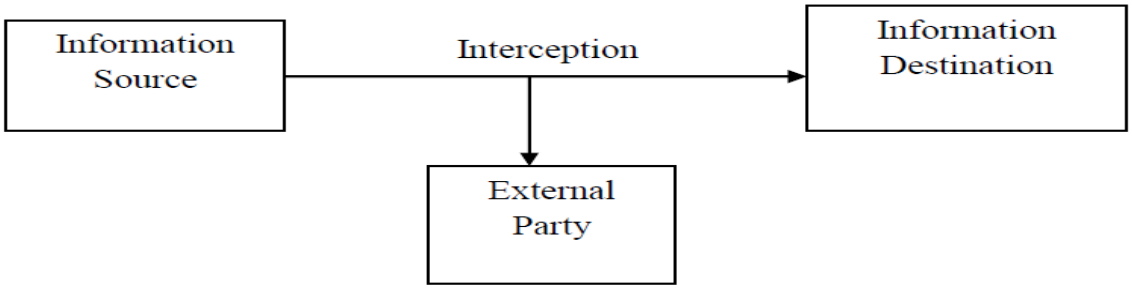


Figure 1-10 Interception Attack

Modification: It is an attack on integrity of the system. In this attack an unauthorized party not only has the access to an asset but has the power to modify it :

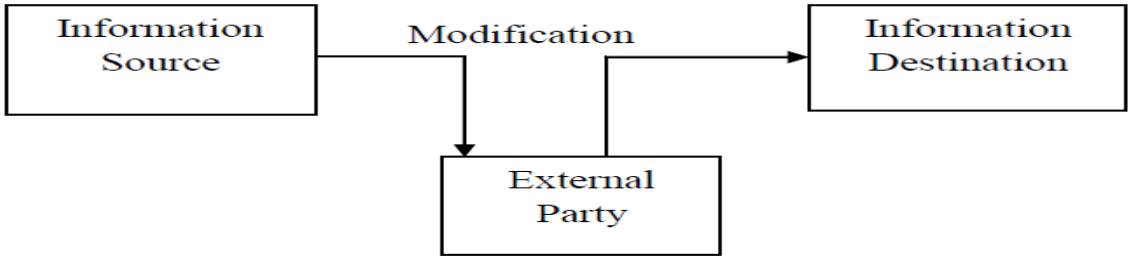


Figure 1-11 Modification of Data

Fabrication: It is an attack on authenticity of the system. In it an unauthorized party inserts counterfeit objects into the system :

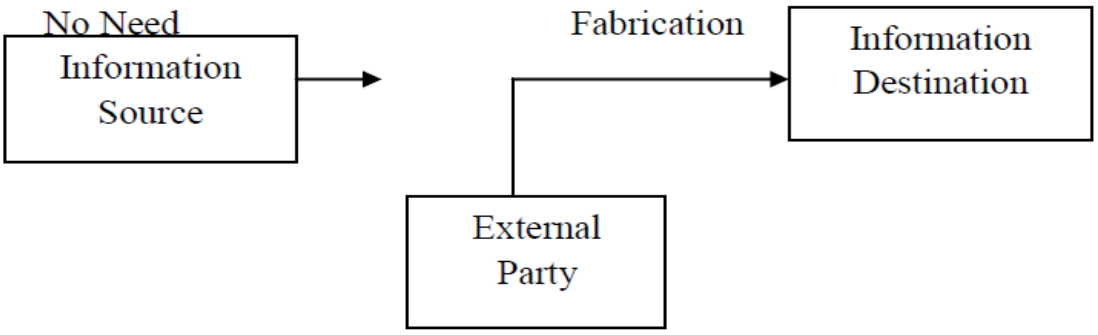


Figure 1-12 Fabrication System Attack

Data Attacks:

An attempted crypto analysis is known as an attack. The level of information that decoder is able to extract from the cryptosystem and can be divided into five ways of decryption which are as follows:

Cipher text only attack: The crypto analyst has cipher text of several messages and all of which were encrypted using the same encryption algorithm. Then job is to recover the plain text or the key used to encrypt the messages. So, to decrypt other part of messages encrypted with the help of same keys.

Known Plaintext attack: Crypto analysts seek the possession of pairs of known plain text and cipher text. Then job is to hold the key used to encrypt the messages or an algorithm to decrypt messages.

Chosen Plaintext Attack (CPA): Crypto analyst not only hold the cipher text but also some parts of chosen plain text. Intruder is identified to be placed at encryption site to do the attack.

Chosen cipher text attack (CCA): In this crypto analyst hold the possession of chosen cipher text and plain text being decrypted from the private key. However, it only has access to an encryption machine.

Cryptanalysis of Caesar Cipher

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed. A brute-force attack involves systematically checking all possible keys until the correct key is found. Simply try all the 25 possible keys. In this case, the plaintext leaps out as occupying the third line.

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.

Euclidean algorithm

One of the basic techniques of number theory is the Euclidean algorithm, which is a simple procedure for determining the greatest common divisor of two positive integers. Let *a* and *b* be integers, not both zero. Recall that GCD (a, b) is the greatest common divisor of *a* and *b*. The best general algorithm for computing GCD (a, b) (and the only practical algorithm, unless the prime factorizations of *a* and *b* are known) is due to Euclid. This algorithm (known as Euclid's Algorithm or the Euclidean Algorithm) involves repeated application of the Division Algorithm. In another word, given any positive integer n and any positive integer a, if we divide a by b, we get an integer *q* quotient and an integer *r* remainder that obey the following relationship:

$$a = q\,b + r \qquad 0 \le r < b$$

If have two numbers c,q that c=q*d+r, then GCD(c,q)=GCD(d,r)

Ex1: find the Greatest Common Divisor (GCD) between 132 and 55 by using Euclid's Algorithm.

132 = 55 * 2 + 22
55 = 22 * 2 + 11
22 = 11 * 2 + 0

Stopping when getting zero 0 then GCD is 11:

GCD(132,55) = GCD(55,22) = GCD(22,11) = GCD(11,0) = 11

Ex2: find the GCD (252 , 198) by using Euclid's Algorithm.

252 = 198 * 1 + 54
198 = 54 * 3 + 36
54 = 36 * 1 + 18
36 = 18 * 2 + 0

GCD(252,198) = (198,54) = (54,36) = (36,18) = (18,0) = 18.

Example: Compute the greatest common divisor (GCD) between the numbers (831, 366).

Solution:

831	=	2 × 366	+ 99
366	=	3 × 99	+ 69
99	=	1 × 69	+ 30
69	=	2 × 30	+ 9
30	=	3 × 9	+ 3
9	=	3 × 3	+ 0

The answer is revealed as the last nonzero remainder: gcd (831, 366) = 3

Note: Because we require that the greatest common divisor be positive GCD (a, b) = GCD (a, -b) = GCD (-a, b) = GCD(-a,-b). In general, GCD(a, b) = GCD(/ a / , / b/).

Example: Find the the greatest common divisor (GCD) of a=321805575, b=198645

Solution:

321805575	=	1620 * ***198645***	+ 675
198645	=	294 * ***675***	+ 195
675	=	3 * ***195***	+ 90
195	=	2 * ***90***	+ 15
90	=	6 * ***15***	+ 0

The answer is revealed as the last nonzero remainder: GCD (321805575, 198645) = 15

H.W.

Now you try some: Answers		
(a) gcd(24, 54) = 6 (b) gcd(18, 42) = 6	(c) gcd(244, 354) = 2 (d) gcd(128, 423) = 1	(e) gcd(2415, 3289) = 23 (f) gcd(4278, 8602) = 46 (g) gcd(406, 555) = 1

Multiplicative Inverses Modulo n

Any positive integer that is less than n and relatively prime to n has a multiplicative inverse modulo n. This is a consequence of the Euclidean algorithm. We will see in the example below why this must be so. Any positive integer that is less than n and not relatively prime to n does not have a multiplicative inverse modulo n.

Example: find the multiplicative Invers of 15 mod 26

Solution: First, do the "forward part" of the Euclidean algorithm – finding the GCD.

$26 = 1\times15 + 11$
$15 = 1\times11 + 4$
$11 = 2 \times 4 + 3$
$4 = 1\times 3 + 1$

So, GCD (15, 26) = 1.

Now, do the "backward part" of the algorithm (this is often called the "extended Euclidean algorithm)– expressing 1 as a combination of 15 and 26.

$1 = 4 -1\times 3$
$1 = 4 -1\times (11 - 2 \times 4)$
$1 = 3\times 4 -1\times11$
$1 = 3\times (15 -1\times11) -1\times11$
$1 = 3\times15 - 4 \times11$
$1 = 3\times15 - 4 \times (26 -1\times15)$
$1 = 7 \times15 - 4 \times 26$

So, $1 = 7 \times15 - 4 \times 26$.

Finally, "go mod 26." Because $26 = 0 \bmod 26$, when we "go mod 26," the equation $1 = 7 \times15 - 4 \times 26$ becomes the congruence$1 = 7 \times15 \bmod 26$. So, the inverse of 15 modulo 26 is 7 (and the inverse of 7 modulo 26 is 15).

Example: find the multiplicative Invers of 19 mod 26

26 = 19 * 1 + 7
19 = 7 * 2 + 5
7 = 5 * 1 + 2
5 = 2 * 2 + 1
2 = 2 * 1 + 0
Now, do the "backward part" of the algorithm
1 = 5 - 2*2
1 = 5 - 2(7 - 5*1)
1 = 5*3 - 2*7
1 = (19 - 7*2)*3 - 2*7
1 = 3*19 - 8*7
1 = 3*19 - 8(26 - 19*1)
1 = 11*19 - 8*26

1 = 11*19 mod 26

So, we conclude that 11 is the multiplicative inverse of 19 modulo 26.

(1)

Ex: find the multiplicative Inverse of 17 mod 43

$17x \equiv 1 \bmod 43$

$x \equiv \frac{1}{17} \bmod 43$

$x = 17^{-1} \bmod 43$

(17, 43)

$43 = 17 \times 2 + 9$

$17 = 9 \times 1 + 8$

$9 = 8 \times 1 + 1$

Now

$1 = 9 - 8 \times 1$

$1 = 9 - 8$

Sub. $8 = 17 - 9 \times 1$

$1 = 9 - 17 + 9$

$1 = 2 \times 9 - 17$

Sub $9 = 43 - 17 \times 2$

$1 = 2(43 - 17 \times 2) - 17$

$1 = 2 \times 43 - 4 \times 17 - 17$

$1 = 2 \times 43 - 5 \times 17$

$\bmod 43 = 0$

$1 = -5 \times 17$

$-5 \bmod 43 = 38$

$\therefore x = 38$

$17 \cdot 38 \bmod 43 = 1$

(2)

Ex: Find the Multiplicative Inverse

$23 \bmod 26 \Rightarrow 23^{-1} = ??$

$23 \bmod 26$

$26 = 23 \times 1 + 3$

$23 = 7 \times 3 + 2$

$3 = 2 \times 1 + 1$

$\gcd(23, 26) = 1$

$1 = 3 - 2 \times 1$

$1 = 3 - (23 - 7 \times 3)$

$1 = 3 - 23 + 7 \times 3$

$1 = 8 \times 3 - 23$

Sub $3 = 26 - 23 \times 1$

$1 = 8(26 - 23 \times 1) - 23$

$= 8 \times 26 - 8(23) - 23$

$= 8 \times 26 - 8 \times 23 (-1) 23$

$= \underbrace{8 \times 26}_{8 \times 26 \bmod 26 = 0} - 9 \times 23$

$1 = -9 \times 23$

$-9 \bmod 26 = 17$

$\therefore 23^{-1} = 17$

(3)

Ex: Find Multiplicative Inverse

17 mod 26

$17^{-1} = ??$

$26 = 17 \times 1 + 9$

$17 = 9 \times 1 + 8$

$9 = 8 \times 1 + 1$

∴ gcd (17, 26) = 1

$1 = 9 - 8 \times 1$

Sub $8 = 17 - 9 \times 1$

$1 = 9 - (17 - 9 \times 1)$

$1 = 9 - 17 + 9$

$1 = 2 \times 9 - 17$

Sub $9 = 26 - 17 \times 1$

$1 = 2 \times (26 - 17) - 17$

$1 = 2 \times 26 - 2 \times 17 - 17$

$1 = -3 \times 17$

$-3 \bmod 26 = 23$

∴ $17^{-1} = 23$

(4)

Ex. Find Multiplicative Inverse.

$15^{-1} = ???$

15 , 26

gcd (15 , 26)

$26 = 1 \times 15 + 11$

$15 = 1 \times 11 + 4$

$11 = 2 \times 4 + 3$

$4 = 3 \times 1 + 1$

∴ gcd (15, 26) = 1

$1 = 4 - 3 \times 1$

Sub $3 = 11 - 2 \times 4$

$1 = 4 - (11 - 2 \times 4)$

$1 = 4 - 11 + 2 \times 4$

$1 = -11 + 3 \times 4$

Sub $4 = 15 - 1 \times 11$

$1 = -11 + 3(15 - 1 \times 11)$

$1 = -11 + 3 \times 15 - 3 \times 11$

$1 = -4 \times 11 + 3 \times 15$

Sub $11 = 26 - 1 \times 15$

$1 = -4(26 - 1 \times 15) + 3 \times 15$

$= -4 \times 26 + 4 \times 15 + 3 \times 15$

mod 26 = 0

$1 = 7 \times 15$

∴ $15^{-1} = 7$

(5)

Ex: Find Multiplicative Inverse

19 , in 26 19^{-1} ??

gcd (19, 26)

$26 = 19 \times 1 + 7$

$19 = 2 \times 7 + 5$

$7 = 1 \times 5 + 2$

$5 = 2 \times 2 + 1$

∴ gcd (19, 26) = 1

$1 = 5 - 2 \times 2$

Sub $2 = 7 - 1 \times 5$

$1 = 5 - 2(7 - 1 \times 5)$

$1 = 5 - 2 \times 7 + 2 \times 5$

$1 = 3 \times 5 - 2 \times 7$

Sub $5 = 19 - 2 \times 7$

$1 = 3(19 - 2 \times 7) - 2 \times 7$

$1 = 3 \times 19 - 6 \times 7 - 2 \times 7$

$1 = 3 \times 19 - 8 \times 7$

Sub $7 = 26 - 19 \times 1$

$1 = 3 \times 19 - 8(26 - 19 \times 1)$

$1 = 3 \times 19 - 8 \times 26 + 8 \times 19$

$1 = 11 \times 19$

∴ $19^{-1} = 11$

(6)

Ex: find Multiplicative Inverse

$23 \bmod 26$ $\quad 23^{-1} = ??$

$\gcd(23, 26) =$

$26 = 1 \times 23 + 3$

$23 = 7 \times 3 + 2$

$3 = 1 \times 2 + 1$

$\therefore \gcd(23, 26) = 1$

$1 = 3 - 1 \times 2$

sub $2 = 23 - 7 \times 3$

$1 = 3 - 1(23 - 7 \times 3)$

$1 = 3 \cdot (-1)23 + 7 \times 3$

$1 = 3 - 23 + 7 \times 3$

$1 = 8 \times 3 - 23$

sub $3 = 26 - 1 \times 23$

$1 = 8(26 - 1 \times 23) - 23$

$= 8 \times 26 - 8 \times 23 - 23$

$1 = -9 \times 23$

$-9 \bmod 26 = 17$

$\therefore 23^{-1} = 17$

H.W

1. Determine each of the following greatest common divisors. Which of the pairs are relatively prime?

- gcd(6, 15)
- gcd(8, 17).
- gcd(24, 138).
- gcd(12378, 3054).

2. Find the multiplicative inverse of 37 modulo 3120.
3. Does 24 have a multiplicative inverse modulo 138 Explain.
4. What integers modulo 16 have multiplicative inverses? Determine the inverses.

Feistel cipher

It is a symmetric structure used in the construction of block ciphers. It is also commonly known as a Feistel network. A large proportion of block ciphers use the scheme, including the Data Encryption Standard (**DES**). The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. Therefore, the size of the code or circuitry required to implement such a cipher is nearly halved.

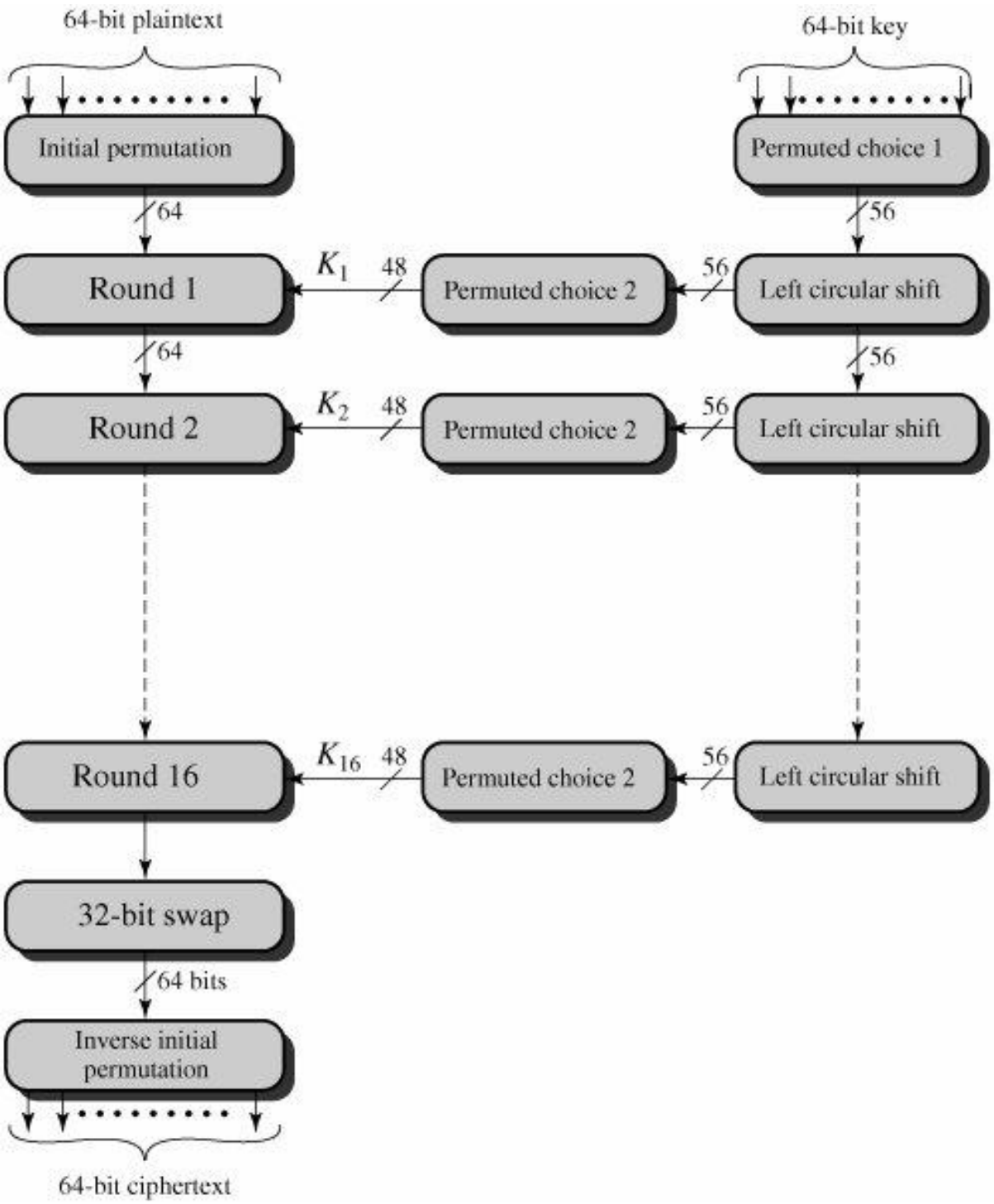
Feistel proposed the use of a cipher that alternates substitutions and permutations, as Diffusion and Confusion.

Diffusion: the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally, this is equivalent to having each ciphertext digit be affected by Many plaintext digits, in other words diffusion hides the relationship between the ciphertext and the plaintext in way that each symbol (character or bit) in the ciphrtext is dependent on some or all symbols in the plaintext, so if one symbol in the plaintext changed several or all symbols in the ciphertext will also be changed. Diffusion can be achieved by repeatedly performing some permutation on the data followed by applying a function to that permutation.

Confusion: seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. In other words confusion hides the relationship between the ciphertext and the encryption key this done if a single bit in the key is changed then most or all bits in the ciphertext will also be changed Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by the use of a complex substitution algorithm. In contrast, a simple linear substitution function would add little confusion.

Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits.



Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**. Finally, the preoutput is passed through a permutation (IP-1) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

The right-hand portion of the previous figure shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the 16 rounds, a *subkey* (*Ki*) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

Initial Permutation

The initial permutation and its inverse are defined by tables, as shown in Tables 3.2a and 3.2b, respectively. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered from 1 to 64. The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

Table 3.2. Permutation Tables for DES

(This item is displayed on page 76 in the print version)

(a) Initial Permutation (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	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse 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

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10

(a) Initial Permutation (IP)							
19	13	30	6	22	11	4	25

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

$M1$	$M2$	$M3$	$M4$	$M5$	$M6$	$M7$	$M8$
$M9$	$M10$	$M11$	$M12$	$M13$	$M14$	$M15$	$M16$
$M17$	$M18$	$M19$	$M20$	$M21$	$M22$	$M23$	$M24$
$M25$	$M26$	$M27$	$M28$	$M29$	$M30$	$M31$	$M32$
$M33$	$M34$	$M35$	$M36$	$M37$	$M38$	$M39$	$M40$
$M41$	$M42$	$M43$	$M44$	$M45$	$M46$	$M47$	$M48$
$M49$	$M50$	$M51$	$M52$	$M53$	$M54$	$M55$	$M56$

where Mi is a binary digit. Then the permutation$X = IP(M)$ is as follows:

$M58$	$M50$	$M42$	$M34$	$M26$	$M18$	$M10$	$M2$
$M60$	$M52$	$M44$	$M36$	$M28$	$M20$	$M12$	$M4$
$M62$	$M54$	$M46$	$M38$	$M30$	$M22$	$M14$	$M6$
$M64$	$M56$	$M48$	$M40$	$M32$	$M24$	$M16$	$M8$
$M57$	$M49$	$M41$	$M33$	$M25$	$M17$	$M9$	$M1$
$M59$	$M51$	$M43$	$M35$	$M27$	$M19$	$M11$	$M3$
$M61$	$M53$	$M45$	$M37$	$M29$	$M21$	$M13$	$M5$
$M63$	$M55$	$M47$	$M39$	$M31$	$M23$	$M15$	$M7$

If we then take the inverse permutation $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, it can be seen that the original ordering of the bits is restored.

Details of Single Round

Figure 1.13 shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$L_i = R_i\text{-}1$$

$$R_i = L_i\text{-}1 \times F(R_i\text{-}1, K_i)$$

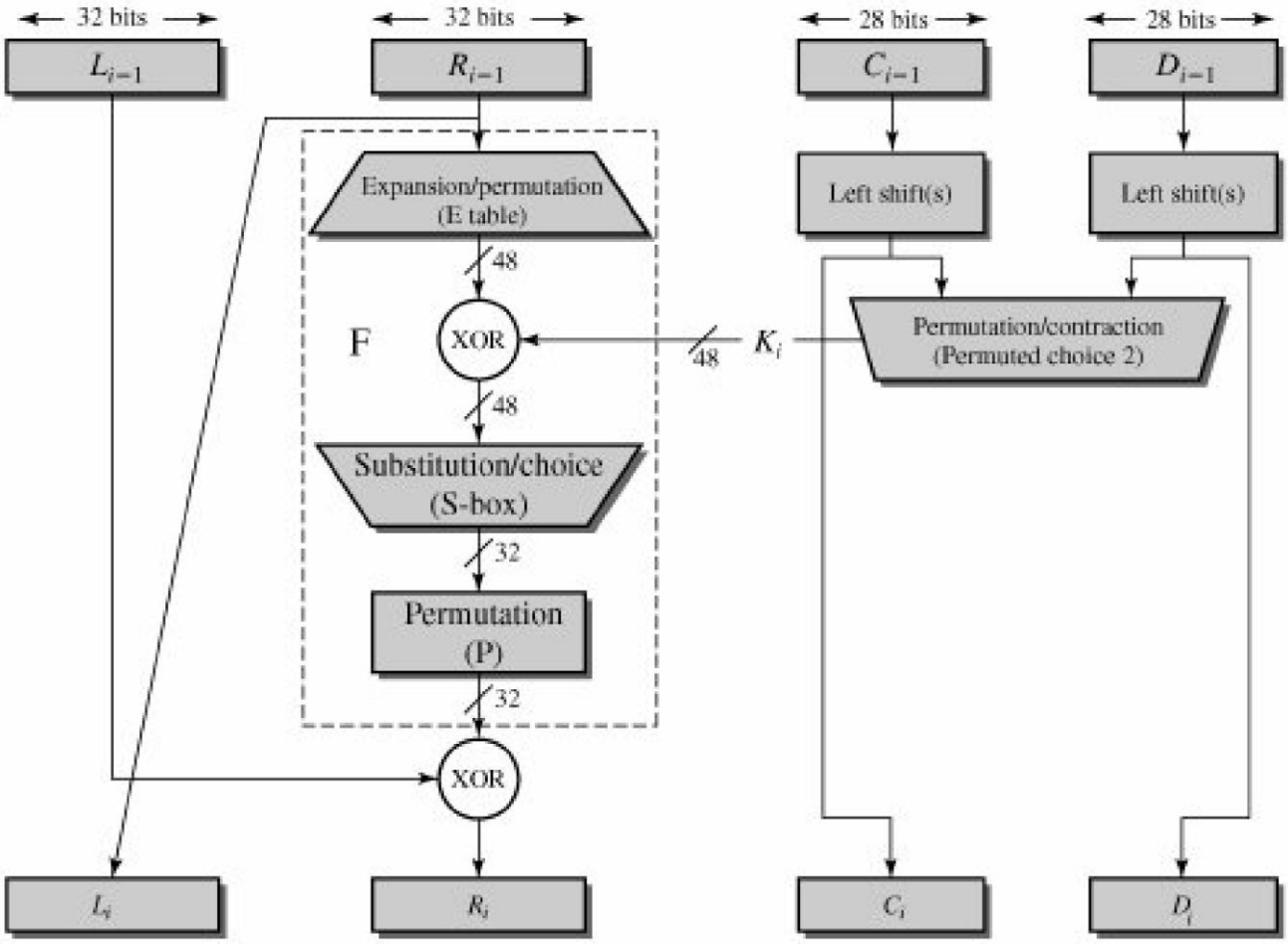


Figure 1.13. Single Round of DES Algorithm.

The round key *Ki* is 48 bits. The *R* input is 32 bits. This *R* input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the *R* bits (Table 3.2c). The resulting 48 bits are XORed with *Ki*.

This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table 3.2d.

The role of the S-boxes in the function F is illustrated in Figure 1.14. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box S*i* form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S*i*. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example, in S1 for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

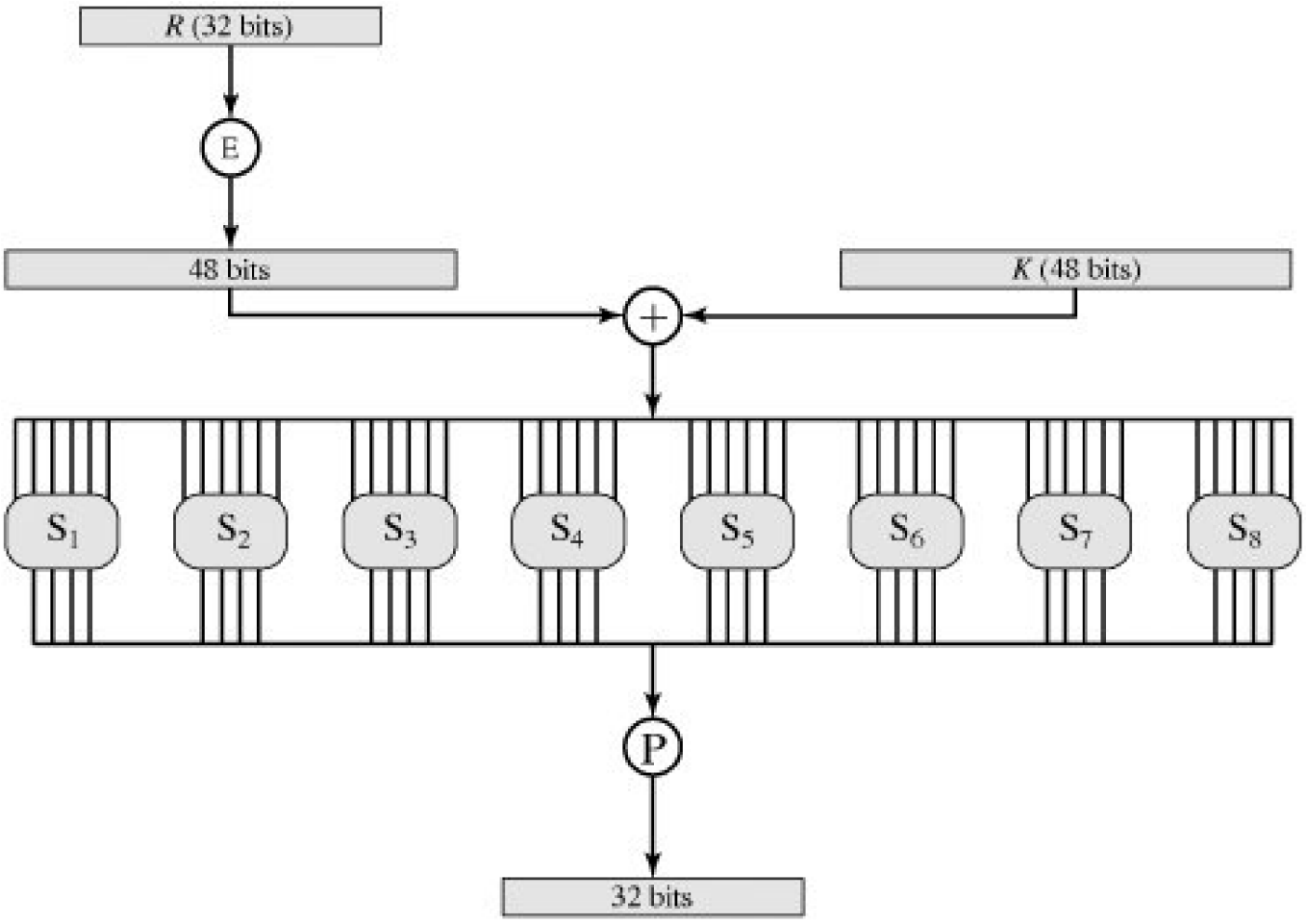


Figure 1.14. Calculation of F(R, K)

Table 3.3. Definition of DES S-Boxes.

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Each row of an S-box defines a general reversible substitution. Figure 3.1 may be useful in understanding the mapping. The figure shows the substitution for row 0 of box S1.

The operation of the S-boxes is worth further comment. Ignore for the moment the contribution of the key (*Ki*). If you examine the expansion table, you see that the 32 bits of input are split into groups of 4 bits, and then become groups of 6 bits by taking the outer bits from the two adjacent groups. For example, if part of the input word is

... efgh ijkl mnop ...
this becomes
... defghi hijklm lmnopq ...

The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round the output from each S-box immediately affects as many others as possible.

Key Generation

Returning to Figures 3.4 and 3.5, we see that a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading in Table 3.4a. The key is first subjected to a permutation governed by a table labeled Permuted Choice One (Table 3.4b). The resulting 56-bit key is then treated as two 28-bit quantities, labeled *C* 0 and *D*0. At each round, *Ci*-1 and *Di*-1 are separately subjected to a circular left shift, or rotation, of 1 or 2 bits, as governed by Table 3.4d.

These shifted values serve as input to the next round. They also serve as input to Permuted Choice Two (Table 3.4c), which produces a 48-bit output that serves as input to the function F(*Ri*-1, *Ki*).

Table 3.4. DES Key Schedule Calculation

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

How DES Works In Detail

Example: Let **M** be the plain text message **M** = 0123456789ABCDEF, where **M** is in hexadecimal (base 16) format. Rewriting **M** in binary format, we get the 64-bit block of text:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
L = 0000 0001 0010 0011 0100 0101 0110 0111
R = 1000 1001 1010 1011 1100 1101 1110 1111

The first bit of **M** is "0". The last bit is "1". We read from left to right.

DES operates on the 64-bit blocks using *key* sizes of 56- bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64). However, we will nevertheless number the bits from 1 to 64, going left to right, in the following calculations. But, as you will see, the eight bits just mentioned get eliminated when we create subkeys.

Example: Let **K** be the hexadecimal key **K** = 133457799BBCDFF1. This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

The DES algorithm uses the following steps:

Step 1: Create 16 Subkeys, Each Of Which Is 48-Bits Long.

The 64-bit key is permuted according to the following table, **PC-1**. Since the first entry in the table is "57", this means that the 57th bit of the original key **K** becomes the first bit of the permuted key **K**+. The 49th bit of the original key becomes the second bit of the permuted key. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

PC-1

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

Example: From the original 64-bit key

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, C_0 and D_0, where each half has 28 bits.

Example: From the permuted key **K**+, we get

C_0 = 1111000 0110011 0010101 0101111
D_0 = 0101010 1011001 1001111 0001111

With C_0 and D_0 defined, we now create sixteen blocks C_n and D_n, $1 <= n <= 16$. Each pair of blocks C_n and D_n is formed from the previous pair C_{n-1} and D_{n-1}, respectively, for n = 1, 2, ..., 16, using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

Iteration Number	Number of Left 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

This means, for example, C_3 and D_3 are obtained from C_2 and D_2, respectively, by two left shifts, and C_{16} and D_{16} are obtained from C_{15} and D_{15}, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.

Example: From original pair pair C_0 and D_0 we obtain:

C_0 = 1111000011001100101010101111
D_0 = 0101010101100110011110001111

C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110

C_2 = 1100001100110010101010111111
D_2 = 0101010110011001111000111101

C_3 = 0000110011001010101011111111
D_3 = 0101011001100111100011110101

C_4 = 0011001100101010101111111100
D_4 = 0101100110011110001111010101

C_5 = 1100110010101010111111110000
D_5 = 0110011001111000111101010101

C_6 = 0011001010101011111111000011
D_6 = 1001100111100011110101010101

C_7 = 1100101010101111111100001100
D_7 = 0110011110001111010101010110

C_8 = 0010101010111111110000110011
D_8 = 1001111000111101010101011001

C_9 = 0101010101111111100001100110
D_9 = 0011110001111010101010110011

C_{10} = 0101010111111110000110011001
D_{10} = 1111000111101010101011001100

C_{11} = 0101011111111000011001100101
D_{11} = 1100011110101010101100110011

C_{12} = 0101111111100001100110010101
D_{12} = 0001111010101010110011001111

C_{13} = 0111111110000110011001010101
D_{13} = 0111101010101011001100111100

C_{14} = 1111111000011001100101010101
D_{14} = 1110101010101100110011110001

C_{15} = 1111100001100110010101010111
D_{15} = 1010101010110011001111000111

C_{16} = 1111000011001100101010101111
D_{16} = 0101010101100110011110001111

We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but **PC-2** only uses 48 of these.

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

Example: For the first key we have C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

which, after we apply the permutation **PC-2**, becomes

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111

K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

So much for the subkeys. Now we look at the message itself.

STEP 2: ENCODE EACH 64-BIT BLOCK OF DATA.

There is an *initial permutation* **IP** of the 64 bits of the message data **M**. This rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order. The 58th bit of **M** becomes the first bit of **IP**. The 50th bit of **M** becomes the second bit of **IP**. The 7th bit of **M** is the last bit of **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	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Example: Applying the initial permutation to the block of text **M**, given previously, we get

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Here the 58th bit of **M** is "1", which becomes the first bit of **IP**. The 50th bit of **M** is "1", which becomes the second bit of **IP**. The 7th bit of **M** is "0", which becomes the last bit of **IP**.

Next divide the permuted block **IP** into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

Example: From **IP**, we get L_0 and R_0

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

We now proceed through 16 iterations, for 1<=n<=16, using a function f which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, (bit-by-bit addition modulo 2)**. Then for **n** going from 1 to 16 we calculate

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

This results in a final block, for n = 16, of $L_{16}R_{16}$. That is, in each iteration, we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation f .

Example: For n = 1, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0 + f(R_0, K_1)$

It remains to explain how the function f works. To calculate f, we first expand each block R_{n-1} from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in R_{n-1} . We'll call the use of this selection table the function **E**. Thus $\mathbf{E}(R_{n-1})$ has a 32 bit input block, and a 48 bit output block.

Let **E** be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

E BIT-SELECTION TABLE

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Thus the first three bits of $\mathbf{E}(R_{n-1})$ are the bits in positions 32, 1 and 2 of R_{n-1} while the last 2 bits of $\mathbf{E}(R_{n-1})$ are the bits in positions 32 and 1.

Example: We calculate $\mathbf{E}(R_0)$ from R_0 as follows:

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

(Note that each block of 4 original bits has been expanded to a block of 6 output bits.)

Next in the ***f*** calculation, we XOR the output $\mathbf{E}(R_{n-1})$ with the key K_n:

$$K_n + \mathbf{E}(R_{n-1}).$$

Example: For K_1 , $\mathbf{E}(R_0)$, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101
K_1+$\mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

We have not yet finished calculating the function ***f*** . To this point we have expanded R_{n-1} from 32 bits to 48 bits, using the selection table, and XORed the result with the key K_n . We now have 48 bits, or eight groups of six bits. We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**". Each group of six bits will give us an address in a different **S** box. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the **S** boxes) for 32 bits total.

Write the previous result, which is 48 bits, in the form:

$$K_n + \mathbf{E}(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8,$$

where each B_i is a group of six bits. We now calculate

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

where $S_i(B_i)$ referres to the output of the ***i***-th **S** box.

To repeat, each of the functions ***S1, S2,..., S8***, takes a 6-bit block as input and yields a 4-bit block as output. The table to determine S_1 is shown and explained below:

S1

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

If S_1 is the function defined in this table and ***B*** is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of ***B*** represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be ***i***. The middle 4 bits of ***B*** represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that

number be ***j***. Look up in the table the number in the ***i***-th row and ***j***-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of S_1 for the input ***B***. For example, for input block ***B*** = 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence $S_1(011011) = 0101$.

The tables defining the functions $S_1,...,S_8$ are the following:

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Example: For the first round, we obtain as the output of the eight **S** boxes:

$K_1 + \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

The final stage in the calculation of f is to do a permutation **P** of the **S**-box output to obtain the final value of f:

$$f = \mathbf{P}(S_1(B_1)S_2(B_2)...S_8(B_8))$$

The permutation **P** is defined in the following table. **P** yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Example: From the output of the eight **S** boxes:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

we get f = 0010 0011 0100 1010 1010 1001 1011 1011

$R_1 = L_0 + f(R_0, K_1)$

```
=   1100   1100   0000   0000   1100   1100   1111   1111
+   0010   0011   0100   1010   1010   1001   1011   1011
= 1110 1111 0100 1010 0110 0101 0100 0100
```

In the next round, we will have $L_2 = R_1$, which is the block we just calculated, and then we must calculate $R_2 = L_1 + f(R_1, K_2)$, and so on for 16 rounds. At the end of the sixteenth round we have the blocks L_{16} and R_{16}. We then ***reverse*** the order of the two blocks into the 64-bit block $R_{16}L_{16}$

and apply a final permutation $\mathbf{IP}^{-1}$ as defined by the following table:

$\mathbf{IP}^{-1}$

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 preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output.

Example: If we process all 16 blocks using the method defined previously, we get, on the 16th round,

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101

We reverse the order of these two blocks and apply the final permutation to

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

which in hexadecimal format is 85E813540F0AB405.

This is the encrypted form of **M** = 0123456789ABCDEF: namely, **C** = 85E813540F0AB405.

Decryption is simply the inverse of encryption, follwing the same steps as above, but reversing the order in which the subkeys are applied.

Public Key Cryptography

The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography.

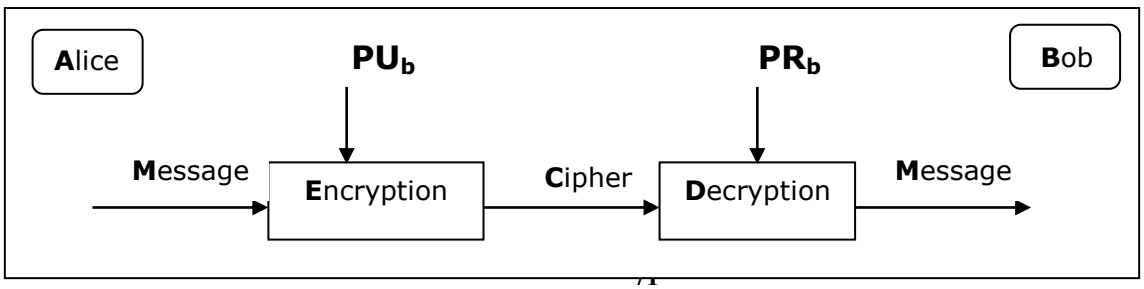
Asymmetric Cryptography (also called Public-Key Cryptography) was a real breakthrough in cryptography. The major change between asymmetric cryptography and the 'traditional' symmetric cryptography is that, in asymmetric cryptography, the key for the encryption is not the same as the key for the decryption. Each user has 2 keys: a Public Key, which is known to all, and a Private Key, which is kept secret (private). Denote Bob's public key by PU(B), and denote Bob's private key by PR(B).

Public-key cryptography provides a radical departure from all that has gone before. For one thing, public-key algorithms are based on mathematical functions rather than on substitution and permutation. More important, public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. The use of two keys has profound consequences in the areas of ***confidentiality***, ***key distribution***, and ***authentication***

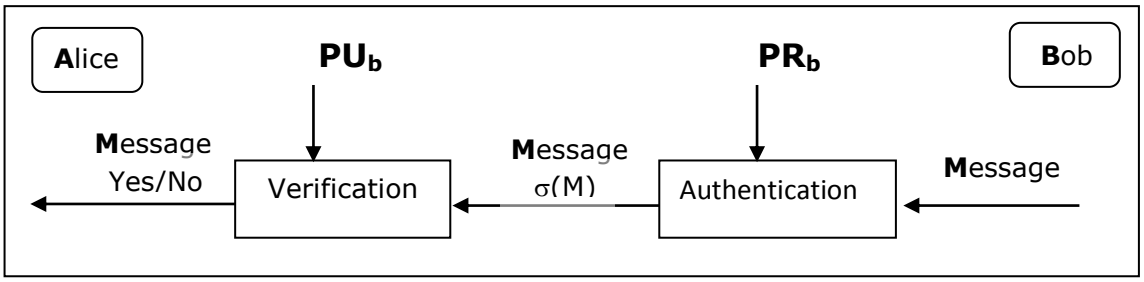
The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. ***The first problem*** is that of key distribution. Key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a key distribution center.

The second is the authentication the way that identifies the sender.

The first use of public key cryptography is for encrypting messages to Bob. Anyone who wishes to send an encrypted message to Bob will use Bob's public key. To decrypt the message, Bob's private key is needed, and only Bob knows it.

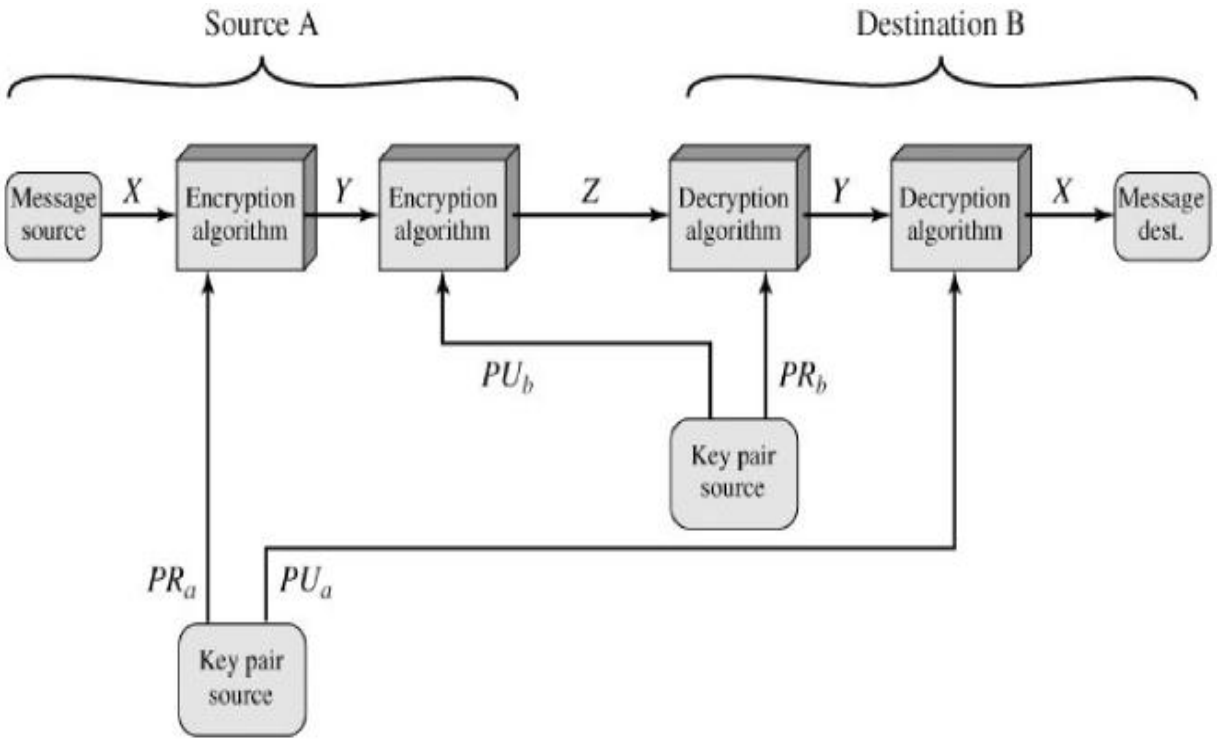


The second use of public key cryptography is for message signing by Bob. Bob's private key is used by Bob to generate a signature. Any one is able to verify Bob's signature using Bob's public key. Denote the signature of the message M by σ(M).



The encryption (or verification) algorithm and the decryption (or authentication) algorithms may or may not be the same.

However, it is possible to provide both the **authentication** function and **confidentiality** by a double use of the public-key scheme.



History of Public Key Cryptography

1976 – Diffie & Hellman published an article 'Public Key Cryptography' that presented the model. They got a patent but never used it to license the technology. They also came up with a protocol for a Key Exchange protocol based on their method.

1977 – Rivest, Shamir, and Adleman invented the RSA scheme which provides Encryption, Signature, and Key Exchange. RSA became the most popular method for public key cryptography.

Since – Many other methods implemented the public key model. For example:
El Gamal – Encryption, Signature, and Key Exchange (developed by El Gamal).
DSS – Digital Signature Standard (developed by NIST).
DH – Key exchange (developed by Diffie and Hellman).

The table below summarizes some of the important aspects of symmetric (Private-Key Cryptography) and public-key encryption. To discriminate between the two, we refer to the key used in symmetric encryption as a secret key (Private-Key Cryptography) and asymmetric (Public-Key Cryptography)

Private-Key Cryptography	Public-Key Cryptography
• The same algorithm with the same key is used for encryption and decryption • Key is shared by both sender and receiver • Also known as symmetric, both parties are equal • Provide **secrecy** • For example DES cipher algorithm.	• One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. • sender and receiver used different keys • Asymmetric since parties are not equal • Provide **secrecy**, **authentication** and **session keys**. • For example RSA cipher algorithm.

RSA Algorithm

The algorithm was developed 1977 by **Ron** Rivest, **Adi** Shamir, and **Len** Adleman at MIT and first published in 1978. The RSA scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

It is a block cipher in which the plaintext and ciphertext are integers between 0 and n 1 for some n.

The algorithm consists of below:

- **Part #1 : Key Generation**

  - Select p, q where: p and q both prime, $p \neq q$
  - Calculate $n = p\ x\ q$
  - Calculate $\varphi\ (n) = (p-1)(q-1)$
  - Select integer e where: $gcd\ ((\varphi\ (n), e) = 1;\ 1 < e < \varphi(n)$
  - Calculate d where: $d \equiv e^{-1} (mod\ \varphi(n))$
  - Public Key $PU = \{e, n\}$
  - Private Key $RP = \{d, p, q\}$

- **Part #2: Encryption:**
  - Plaintext: M<n
  - Ciphertext: $C = M^e\ mod\ n$
- **Part #3 Decryption:**
  - Ciphertext: C
  - Plaintext $M = C^d\ mod\ n$

Example:

Suppose p=17, q=11. Using RSA to encrypt the message M=88

Solution:

- n=p*q → 17*11 =$\mathbf{187}$
- φ (n) = (p-1)(q-1) = 16*10
  =$\mathbf{160}$
- choose e verifies $gcd\ ((\varphi\ (n), e) = 1\ ;\ 1 < e <\ \varphi(n)$
  then $\boldsymbol{e = 7}$
- choose d verifies $e.d \equiv\ 1\ mod\ \varphi\ (n)$, then $\boldsymbol{d = 23}$
  $7.23\ \equiv\ 1\ mod\ 160$
- PU={7, 187}
- PR={23,17,11}

To encrypt m=88 using the encryption formula

$C\ = M^e\ mod\ n$→ 88^7 mod 187 =11

The decryption:

$M\ = C^d\ mod\ n$

$M\ = 11^{23}\ mod\ 187 = 88$

Fast Exponentiation Algorithm:

The formula $\boldsymbol{a^b\ mod\ n}$ could be calculated faster by converting the value of b to binary scheme, then do the square and multiplication. If b_i=0, then do the square only, else do square and multiply together. Below the pseudo code explain the algorithm.

```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
    if  b_i = 1
        then c ← c + 1
             f ← (f × a) mod n
return f
```

Example: what is the Result of the Fast Modular Exponentiation Algorithm for a^b mod n, where a = 7, b = 560, n = 561.

Solution:

Note that the variable c is not needed; it is included for explanatory purposes. The final value of c is the value of the exponent. Note: The integer b is expressed as a binary number $b_k b_{k-1} \ldots b_0$. The value of b should convert to binary scheme, so b= 1000110000. Table below shows the result of algorithm application

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

$\therefore \quad 7^{560} mod\ 561 = 1$

Assignment:

1. If p=61, q= 53, e=17 and the encrypted message C= 855. What is the original message m?
2. The ciphertext C =10 sent to a user whose public key is e=5, n=35. What is the plaintext M?

Applications for Public-Key Cryptosystems

In broad terms, we can classify the use of public-key cryptosystems into three categories:

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications. Following Table indicates the applications supported by the algorithms discussed in this book.

Table Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes

Key Management; Other Public-Key Cryptosystems

In a confidential communication, the authenticity needs to be carefully established for the two partners before sending any confidential information one needs to be sure to whom it sends that information: **authentication protocols**.

Authentication protocols and setting up secret keys

A . *Direct authentication*

1.Based on a shared secret master key

2.Based on a public-key system

3.Diffie-Hellman

B . *Mediated authentication*

1.Based on key distribution centers

2.kerberos.

In this chapter we examined the problem of the distribution of secret keys. One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

1. The distribution of public keys
2. The use of public-key encryption to distribute secret keys

We examine each of these areas in turn.

1 Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

Public Announcement of Public Keys

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can

send his or her public key to any other participant or broadcast the key to the community at large

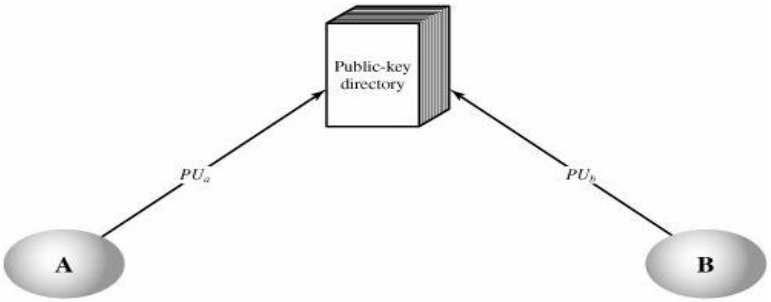
Uncontrolled Public-Key Distribution

Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization as in the following Figure . Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.



Public-Key Publication

This scheme is clearly more secure than individual public announcements but still has vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated as in Figure below. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. The following steps (matched by number to Figure below) occur:

1. A sends a time stamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth} Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

- B's public key, PU_b which A can use to encrypt messages destined for B
- The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
- The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

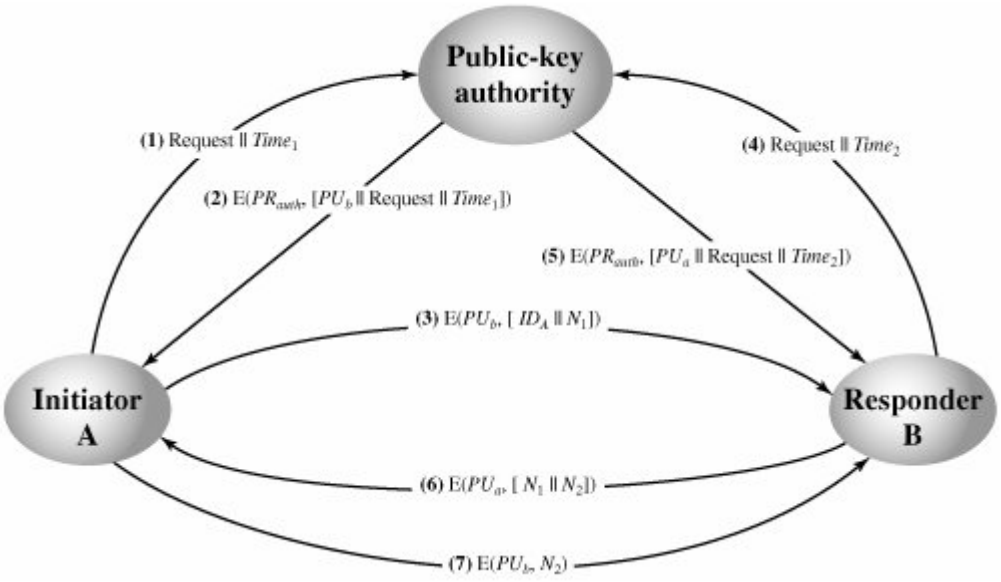
4, B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

5. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message, the presence of N_1 in message assures A that the correspondent is B.

7. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

]



Public-Key Distribution Scenario

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

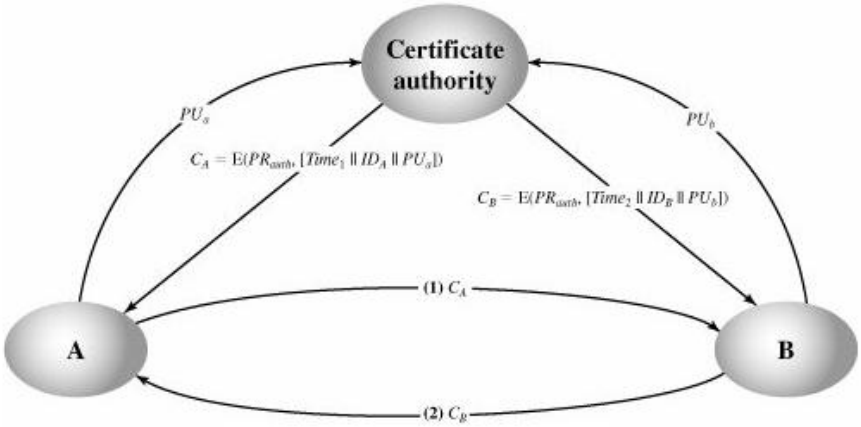
Public-Key Certificates

The scenario of above Figure is attractive, yet it has some drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

In essence, a certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate. Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in the following Figure. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.



Exchange of Public-Key Certificates

Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

$C_A = E(PR_{auth}, [T||ID_A||PU_a])$

where PR_{auth} is the private key used by the authority and T is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T||ID_A||PU_a])) = (T||ID_A||PU_a)$

The recipient uses the authority's public key, PU_{auth} to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements ID_A and PU_a provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

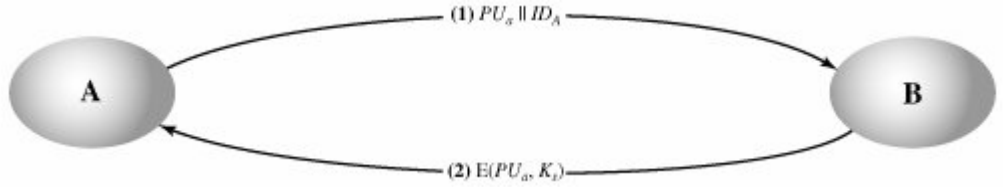
2 Distribution of Secret Keys Using Public-Key Cryptography

Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both is possible. However, few users will wish to make exclusive use of public-key encryption for communication because of the relatively slow data rates that can be achieved. Accordingly, public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

a. Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle, as illustrated in the following Figure . If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A.
2. B generates a secret key, K_s, and transmits it to A, encrypted with A's public key.
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s.
4. A discards PU_a and PR_a and B discards PU_a.



Simple Use of Public-Key Encryption to Establish a Session Key

A and B can now securely communicate using conventional encryption and the session key K_s. At the completion of the exchange, both A and B discard K_s. Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.

The protocol depicted in the above Figure is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message.

b. Secret Key Distribution with Confidentiality and Authentication

Figure below, based on an approach suggested, provides protection against both active and passive attacks. We begin at a point when it is assumed that A and B have exchanged public keys by one of the schemes described earlier in this section. Then the following steps occur:

1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M = E(PU_b, E(PR_a, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

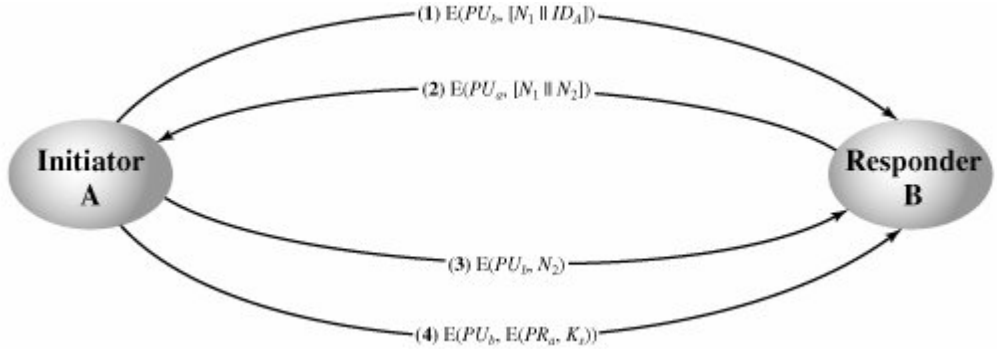


Figure Public-Key Distribution of Secret Keys

Notice that the first three steps of this scheme are the same as the last three steps of the above Figure. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

c. Diffie-Hellman Key Exchange

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key exchange. A number of commercial products employ this key exchange technique.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the

following way. First, we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to p 1. That is, if a is a primitive root of the prime number p, then the numbers

a mod p, a^2 mod p,..., a^{p-1} mod p

are distinct and consist of the integers from 1 through p 1 in some permutation.

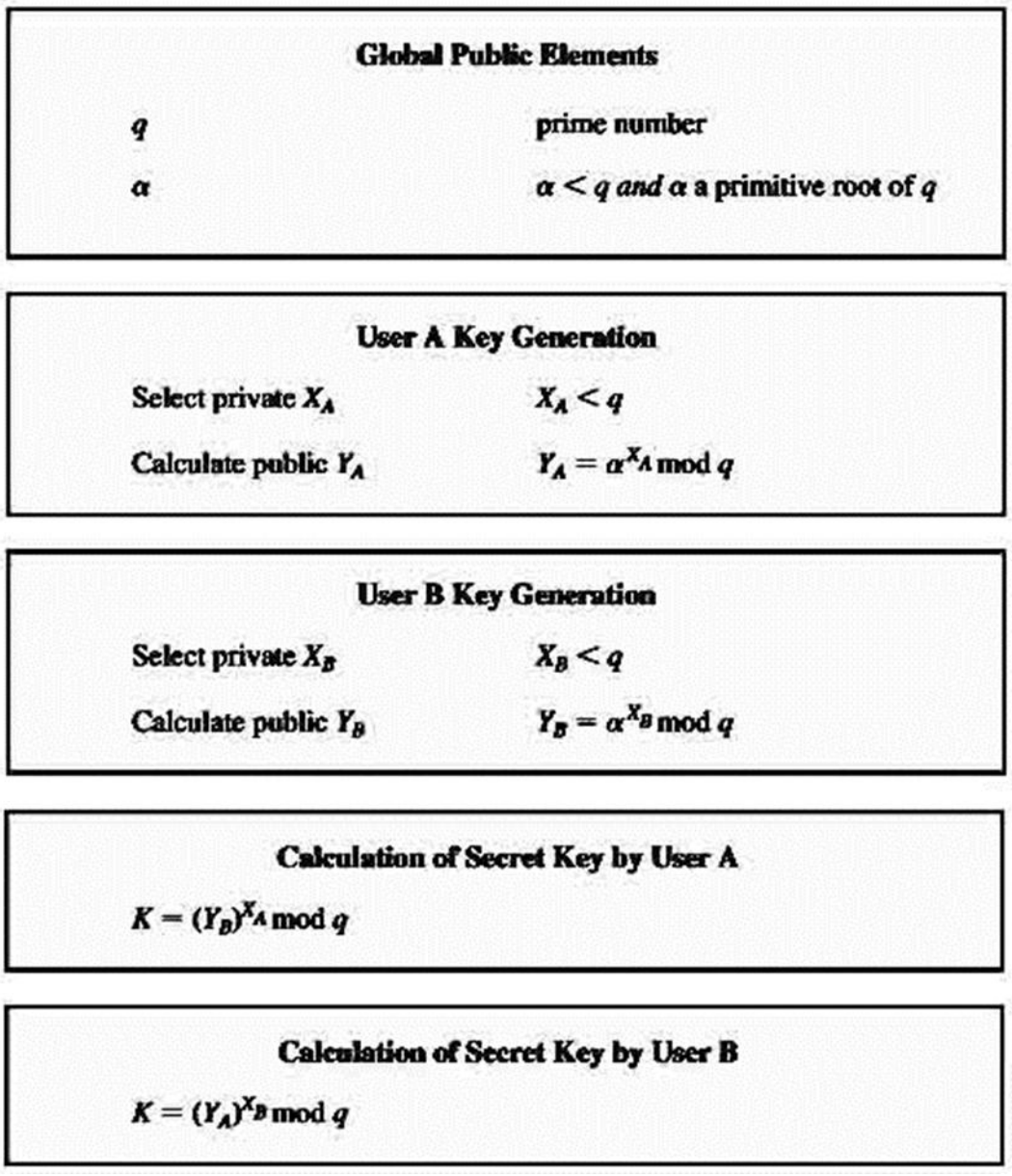
For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$b \equiv a^i \pmod{p}$ where $0 \leq i \leq (p - 1)$

The exponent *i* is referred to as the discrete logarithm of *b* for the base *a*, mod *p*.

The Algorithm

The Diffie-Hellman key exchange algorithm. For this scheme, there are two publicly known numbers: a prime number q and an integer that is a primitive root of q.



The result is that the two sides have exchanged a secret value as shown in Figure below. The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

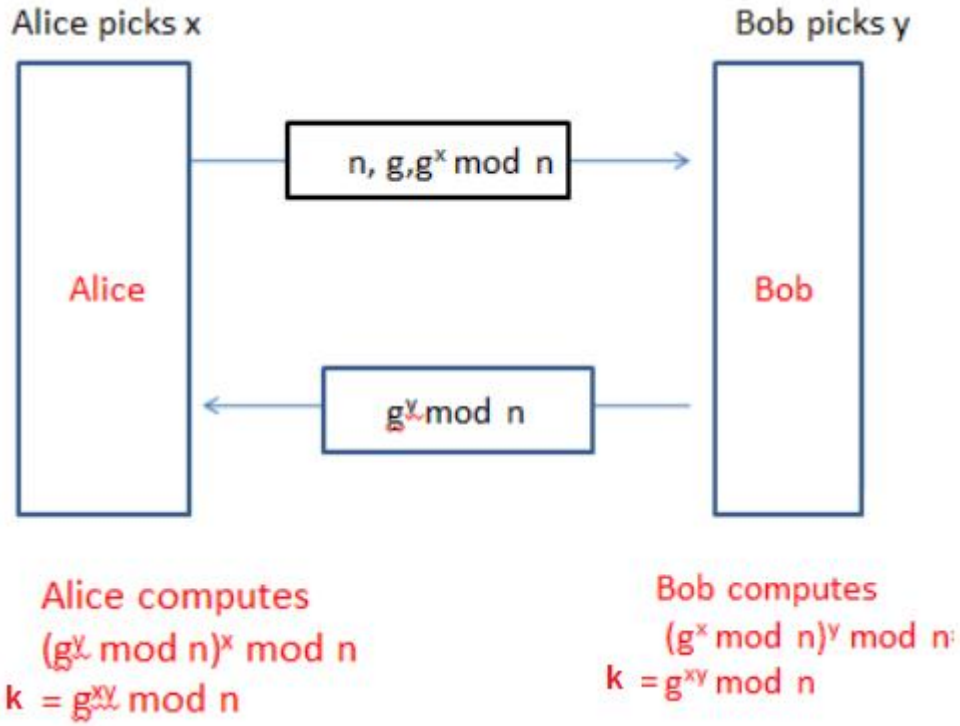


Figure The Diffie-Hellman Key Exchange Algorithm

Here is an example. Key exchange is based on the use of the prime number q = 353 and a primitive root of 353, in this case α = 3. A and B select secret keys X_A = 97 and X_B = 233, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

Authentication using Kerberos

1. User **Alice** logs on to Authentication Server (AS) and requests service.

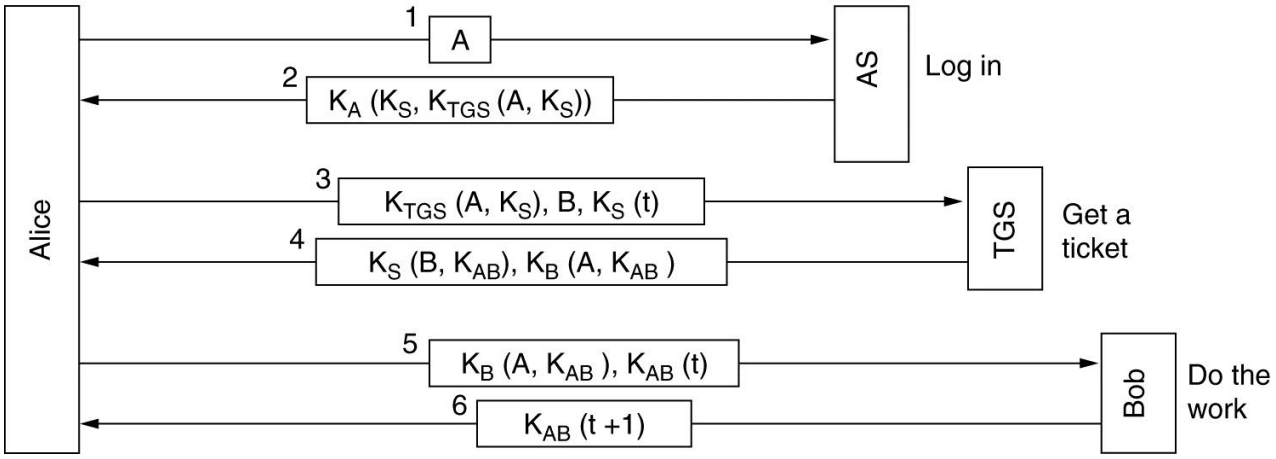
2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. User Alice uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to Ticket Granting Server (TGS).

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server (Bob).

5. User Alice sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service.



Intro to Network Security

Application Encryption & VPN

Text:
Network Security: The Complete Reference, Bragg, Rhodes-Ousley, Strassberg et al.

Intranets, Extranets, and VPNs

Private Network: A network that is not freely available to the public.

Intranet: Private network that uses Internet technology. Often includes:
Web browsers & servers
May include private IP addresses, including:

10.0.0.0:		Class A Address
172.16.0.0-	172.31.255.255:	16 Contiguous Class B Addresses
192.168.0.0-	192.168.255.255:	256 Contiguous Class C Addresses

Extranet: Enables two or more companies to share common information & resources by extending the intranet
Accommodates business-to-business communication (B2B): post orders, share projects, share pricing, communicate collaboratively.
Extranets can introduce weaknesses in security.

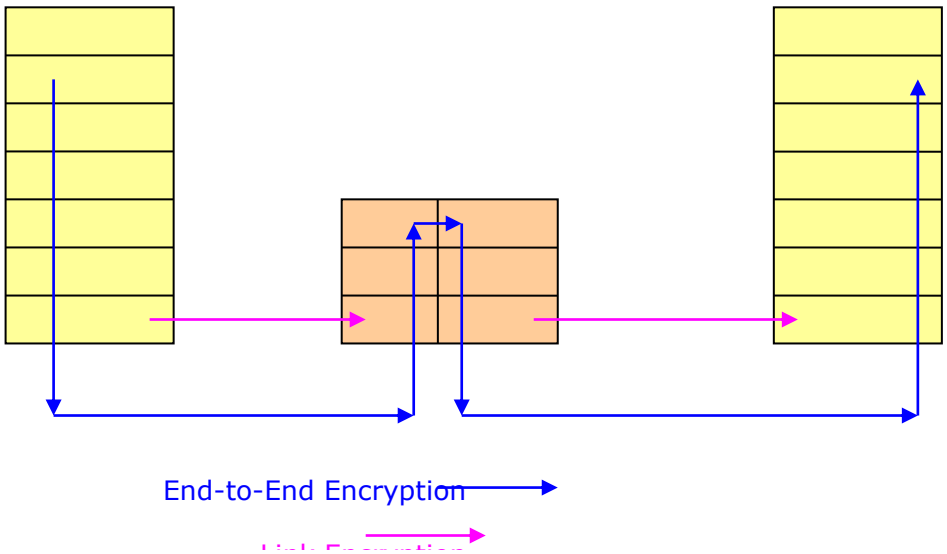
Virtual Private Network (VPN): A means of carrying private traffic over a public network
Uses link encryption to give users sense that they are operating on a private network when they are actually transmitting over a public network
Communications pass through an **encrypted tunnel**

Intranet VPN: Connects two or more private networks within the same company

Extranet VPN: Connects two or more private networks between different companies (for B2B or business-to-business communications).

Remote Access VPN: A roaming user has access to a private network via wireless, hotel room, etc.

Encryption Techniques: End-to-End vs. Link Encryption



Comparison:

	Link	End-to-End
Purpose	Link itself is vulnerable: Packet sniffers & eavesdroppers	Intermediate nodes may be compromised
Encryption coverage	Link-Specific: All packets transmitted on the single link are encrypted	Connection-Specific: A connection is encrypted across all its links
Protocol header security	Encrypted for all protocol layers (at or above layers 1 or 2)	Encrypted for upper layer protocols only
Network device exposure	Intermediate nodes decrypt	Intermediate nodes cannot decrypt
Authentication	Provides node authentication	Provides user authentication
Ease of use	Transparent to user, One key per link	Not user-transparent, One key per connection
User Selectivity of algorithm	One algorithm for all users	User selects encryption algorithm
Implementation	Encryption done in hardware	Encryption done in hardware or software
Applications	Virtual Private Network (VPN)	Secure Shell (SSH) SSL Pretty Good Privacy (PGP)

End-to-End Encryption Methods

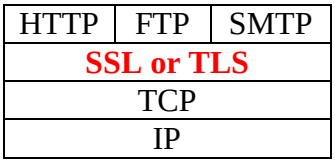
Encryption by Application

Applications are encrypted on a case by case basis.

Secure Shell (SSH): Provides an authenticated and encrypted remote login and file transfer capability.
Can tunnel XWindows, ftp, POP-3, IMAP-4, … replaces RCP, rlogin, rsh
Authenticates before allowing connection
Generates a public/private key pair; notifies partner systems of public key
SSH protocol negotiates the encryption algorithm: DES, IDEA, AES and the authentication algorithm: public key & Kerberos.
SSH2: Can negotiate between 3DES, IDEA, Blowfish, Twofish, Arcfour, Cast
SSHv2 is more secure than SSHv1, which has numerous exploits
Uses port 22 for all applications
SSH is free or minimal cost for commercial version

Secure Sockets Layer (SSL): Protects communication above the transport level: Web
Certificate-based system created by Netscape to protect web page communications
Implemented by Netscape & Microsoft Explorer and other browsers – widely available
When SSL & HTTP used together is called **HTTPS**
Standardized in IETF as **Transport Layer Security (TLS)** which is nearly equal to SSLv3 but incompatible (RFC 2246)
URL name starts as https:// - also key or lock icon displays at bottom corner.

SSL protocol negotiates the security algorithm:
Confidentiality: DES, 3DES, AES, RC4
Integrity: MD5, SHA-1
Authentication: RSA, Diffie-Hellman
Non-Repudiation: Digital Signature
Both Netscape and Internet Explorer allow you to configure SSL parms
Steps include:
Customer generates session key which is encrypted & sent to server using server's public key
Client initiates negotiation of security parms. However server may negotiate to lesser security
Client authenticates server certificate using public key encryption and possibly vice versa



Secure Hypertext Transport Protocol (SHTTP): Extends HTTP protocol to protect each message sent between 2 computers.
Summarize the difference between HTTP and HTTPS

HTTP	HTTPS
URL begins with "http://"	URL begins with "https://"
It uses port 80 for communication	It uses port 443 for communication
Unsecured	Secured
Operates at Application Layer	Operates at Transport Layer
No encryption	Encryption is present
No certificates required	Certificates required

Pretty Good Privacy (PGP): Email encryption method
Available in 1991, currently free but also became a commercial product in 1996
Public Keys can be registered under email names at:
ldap://certsaver.pgp.com
http://pgpkeys.mit.edu:11371
Point your PGP utility at one of them
Web of Trust: Trust that that the keys you get in email (from people you know) or in a list via email (from people you know) is authentic
Senders can put their PGP public key at the bottom of their email messages.
Not endorsed by NSA, but good for private use
Uses RSA or Diffie-Hellman public key encryption for key exchange
Uses IDEA, 3DES, or CAST for message encryption
Uses MD5 hashing algorithm for integrity
Authentication provided via public key certificates – generated as part of PGP
Nonrepudiation provided via digitally signed messages.

<table>
<tr><td rowspan="2">Kerberos</td><td>S/MIME</td><td>PGP</td><td>SET</td></tr>
<tr><td colspan="2">SMTP</td><td>HTTP</td></tr>
<tr><td>UDP</td><td colspan="3">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

Secure Multipurpose Internet Mail Extensions (S/MIME): An Internet standard for secure email attachments
MIME: Protocol specification dictates how multimedia data and email attachments are transferred: E.g., Header=Image, subtype=jpeg.
Used for encryption and digital signatures
Encrypts many types of attachments: spreadsheets, graphics, presentations, movies, sound.
Uses public key certificates, in X.509 format, for authentication and key exchange
Can negotiate from a set of encryption algorithms: DES, AES, RC2
Integrated into many commercial email packages, including sendmail.

Secure Electronic Transaction (SET): Protects credit card transactions on the Internet
Requested by MasterCard and Visa in 1996
Provides trust by use of X.509 digital certificates
Ensures privacy & secure communications

Discussion: What info is visible (unencrypted) in the SSL packets? In the PGP or S/MIME packets?