**UNIVERSITY OF RWANDA**

**COLLEGE OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF ICT**

**FOURTH YEAR COMPUTER SCIENCE/INFORMATION SECURITY**

# Cryptography

Prepared by Theogene BIZIMUNGU

Academic year: 2018-2019

## CHAPTER I: INTRODUCTION TO CRYPTOGRAPHY

### I.1. HISTORY OF CRYPTOGRAPHY

Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit. Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

Cryptography is a subject that has been studied and applied since ancient Roman times, and research into better encryption methods continues to this day. Cryptography is the art of encoding and decoding messages so that messages can be securely transmitted from a sender to a receiver without fear of an outside party intercepting and reading or altering the message's contents. During the Middle Ages, cryptography started to progress. All of the Western European governments used cryptography in one form or another, and codes started to become more popular. Ciphers were commonly used to keep in touch with ambassadors. The first major advances in cryptography were made in Italy. Venice created an elaborate organization in 1452 with the sole purpose of dealing with cryptography. They had three cipher secretaries who solved and created ciphers that were used by the government. The next major step was taken in 1518, by Trithemius, a German monk who had a deep interest in the occult. He wrote a series of six books called 'Polygraphia', and in the fifth book, devised a table that repeated the alphabet with each row a duplicate of the one above it, shifted over one letter. In 1553, Giovan Batista Belaso extended this technique by choosing a keyword that is written above the plaintext, in a letter to letter correspondence. The keyword is restarted at the beginning of each new plaintext word. The most famous cryptographer of the 16th century was Blaise de Vigenere (1523-1596). In 1585, he wrote 'Tracte des Chiffres' in which he used a Trithemius table, but changed the way the key system worked. One of his techniques used the plaintext as it's own key. In 1628, a Frenchman named Antoine Rossignol helped his army defeat the Huguenots by decoding a captured message. After this victory, he was called upon many times to solve ciphers for the French government. He used two lists to solve his ciphers: "one in which the plain elements were in alphabetical order and the code elements randomized, and one to facilitate decoding in which the code elements stood in alphabetical or numerical order while their plain equivalents were disarranged." When Rossignol died in 1682, his son, and later his grandson, continued his work. By this time, there were many cryptographers employed by the French government. Together, they formed the "Cabinet Noir" (the "Black Chamber"). By the 1700's, "Black Chambers" were common in Europe, one of the most renown being that in Vienna. It was called 'The Geheime Kabinets-Kanzlei' and was directed by Baron Ignaz de Koch between 1749 and 1763. In 1817, Colonel Decius Wadsworth developed a set of two disks, one inside the other, where the outer disk had the 26 letters of the alphabet, and the numbers 2-8, and the inner disk had only the 26 letters. The disks were geared together at a ratio of 26:33. In 1844, the development of cryptography was dramatically altered by the invention of the telegraph. The 'Playfair' system was invented by Charles Wheatstone and Lyon Playfair in 1854, and was the first system that used pairs of symbols for encryption. In 1859, Pliny Earle Chase, developed what is known as the fractionating or tomographic cipher. A two digit number was assigned to each character of plaintext by means of a table. Kasiski developed a cryptanalysis method in 1863 which broke almost every existing cipher of that time. The method was to find repetitions of strings of characters in the ciphertext. During the Civil War (1861-1865),

ciphers were not very complex. Many techniques consisted merely of writing words in a different order and substituting code words for proper names and locations. In 1883, Auguste Kerckhoffs wrote 'La Cryptographie Militaire' in which he set forth six basic requirements of cryptography. We note that the easily remembered key is very amenable to attack, and that these rules, as all others, should be questioned before placing trust in them. In 1917, the Americans formed the cryptographic organization MI-8. It's director was Herbert Osborne Yardley. In 1929, Lester S. Hill published an article "Cryptography in an Algebraic Alphabet" in "The American Mathematical Monthly". Each plaintext letter was given a numerical value. He then used polynomial equations to encipher plaintext, with values over 25 reduced modulo 26. In 1948, Shannon published "A Communications Theory of Secrecy Systems" . Shannon was one of the first modern cryptographers to attribute advanced mathematical techniques to the science of ciphers.

## I.2. CRYPTOGRAPHY CONCEPTS

### Cryptography

Cryptography (from Greek kryptós, "hidden", and gráphein, "to write") is, traditionally, the study of means of converting information from its normal, comprehensible form into an incomprehensible format, rendering it unreadable without secret knowledge — the art of encryption.

Cryptography is the study of secret writing. It uses mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.

Cryptography is an interdisciplinary subject,

- ✓ linguistics
- ✓ Mathematics: number theory, information theory, computational complexity and statistics
- ✓ engineering

### Network Security

It refers to any activity designed to protect the usability and integrity of your network and data. It includes both hardware and software technologies.

**Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers

**Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

### Cryptanalysis

The art and science of decrypting messages without knowing the key.

### *Cryptology*

cryptography + cryptanalysis

**Plaintext**

It is an information a sender wishes to transmit to a receiver.

**Intruder**

It is a person who is not authorized to access on information

**Ciphertext**

Encoded message

**Cipher**

An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods.

**Encipher**

The process of converting plaintext to ciphertext using a cipher and a key

**Decipher**

The process of converting ciphertext back into plaintext using a cipher and a key

**Encryption**

It is a process of changing plaintext into ciphertext.

**Decryption**

It is a process of changing ciphertext into plaintext.

**Key**

It is a secret information which is used to transform ciphertext to plaintext and vice versa. Notation for relating the plaintext, ciphertext, and the keys

C=EK(P) denotes that C is the encryption of the plaintext P using the key K

P=DK(C) denotes that P is the decryption of the ciphertext C using the key K Then DK(EK(P))=P

**Stream cipher**

It operates on a single bit(byte or computer word) at a time

**Block cipher**

It encrypts one block of data at a time using the same key on each block.

***Cryptosystem***

*A cryptosystem is a five-tuple* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, where the following conditions are satisfied:*

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

**Steganography**

Steganography is defined as "hiding information within a noise; a way to supplement (not replace) encryption, to prevent the existence of encrypted data from being detected".

**Threat**
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.

**Attack**

An assault on system security that derives from an intelligent threat that is an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

**Security**

It is to make sure that nosy people cannot read or secretly modify messages intended for other recipients

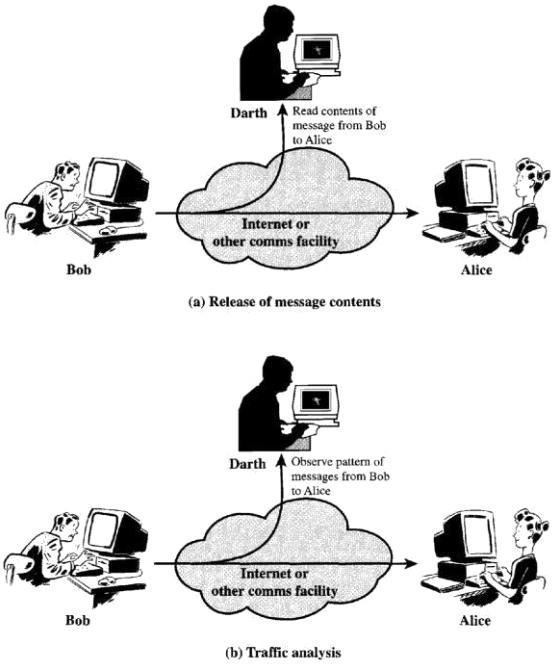
## I.3. SECURITY ATTACKS

Attacks can be split into two wide groups: passive attacks and active attacks. A ***passive attack*** attempts to learn or make use of information from the system but does not affect system resources. An ***active attack*** attempts to alter system resources or affect their operation.

**Passive Attacks**
Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

- The release of message contents is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.
- A second type of passive attack, traffic analysis, is subtler. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, ***the emphasis in dealing with passive attacks is on prevention rather than detection.***




Passive attacks

**Active Attacks**

Active attacks involve some modification of the data stream or the creation of a false stream. An attack can take place on any of the communications links. For active attacks, the attacker needs to gain physical control of a portion of the link and be able to insert and capture transmissions. For a passive attack, the attacker merely needs to be able to observe transmissions. The communications links involved can be cable (telephone twisted pair, coaxial cable, or optical fiber), microwave links, or satellite channels. The active attacks can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.

- A **masquerade** takes place when one entity pretends to be a different entity.
- **Replay** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- **Modification** of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
- The **denial** of service prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communications facilities and paths at all times. Instead, ***the goal is to detect them and to recover from any disruption or delays caused by them.*** Because the detection has a deterrent effect, it may also contribute to prevention.

## I.4. SECURITY SERVICES

There are security services to prevent security attacks— authentication, data confidentiality, data integrity, non- repudiation and access control.

### Authentication

Verifying that a user, computer, or service (such as an application provided on a network server) is the entity that it claims to be. Authentication is an important part of identity management. Users, computers, and services that can be authenticated when they log on to a network or, after logon, when they authenticate to a network service, are known collectively as principals, security principals, or digital identities.

### Data confidentiality

Confidentiality, keeping information secret from unauthorized access, is probably the most common aspect of information security: we need to protect confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information.

### Data integrity

Information needs to be changed constantly. In a bank, when a customer deposits or withdraws money, the balance of their account needs to be changed. Integrity means that changes should be done only by authorized users and through authorized mechanisms. **Non- repudiation**

No repudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

### Access Control

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

## I.5. TRADITIONAL CRYPTOGRAPHY

The fundamental objective of cryptography is to enable two people, usually referred to as Alice and Bob, to communicate over an insecure channel in such a way that an opponent, Oscar, cannot understand what is being said. This channel could be a telephone line or

computer network, for example. The information that Alice wants to send to Bob, which we call ―plaintext,‖ can be English text, numerical data, or anything at all — its structure is completely arbitrary. Alice encrypts the plaintext, using a predetermined key, and sends the resulting ciphertext over the channel. Oscar, upon seeing the ciphertext in the channel by eavesdropping, cannot determine what the plaintext was; but Bob, who knows the encryption key, can decrypt the ciphertext and reconstruct the plaintext.

**Cryptographic** systems are generically classified along three independent dimensions:

1. **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: ***substitution***, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and ***transposition***, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations be reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.

2. **The number of keys used**. If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. Examples are the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES). If the sender and receiver each use a different key, the system is referred to as asymmetric, two-key, or public-key encryption. In public key cryptology each party, actually, has a key that consists of two parts, a public and a secret (or private) part. The public part can be used to encrypt information, the corresponding secret part to decrypt. Alternatively, the secret key is used to sign a document, the corresponding public key to verify the resulting signature. Furthermore, a widely shared public key can be used to establish a common secret among two parties, for instance a key for a symmetric system. Examples of public key cryptosystems are the Diffie-Hellman key agreement protocol and the El-Gamal and RSA encryption and signature schemes.

3. **The way in which the plaintext is processed**. A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

The two basic building blocks of all encryption techniques are:

- substitution and
- transposition.

### I.5.1. SUBSTITUTION TECHNIQUES

A **substitution** cipher is a method of encryption by which units of plaintext are substituted with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution. The letters of plaintext are normally replaced by other letters or by numbers or symbols. The plaintext is viewed as a

sequence of bits, then substitution involves replacing plaintext bit pattern with ciphertext bit patterns.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a **simple substitution** cipher; a cipher that operates on larger groups of letters is termed **polygraphic**. A **monoalphabetic** cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different times in the message—such as with homophones, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext.

Historically, monoalphabetic ciphers goes bake to Julius Caesar. Julius Caesar enciphered his dispatches by writing D for A, E for B and so on. When Augustus Caesar ascended the throne, he changed the imperial cipher system so that C was now written for A, D for B, and so on. In modern terminology, we would say that he changed the key from D to C.

The Arabs generalized this idea to the monoalphabetic substitution, in which a keyword is used to permute the cipher alphabet.

Monoalphabetic cipher and other simple ciphers are discussed next.

### I.5.1.1. ShifT Cipher

In this section, we will describe the **Shift Cipher(Caesar Cipher)**, which is based on modular arithmetic. But first we review some basic definitions of modular arithmetic.

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

Suppose we divide $a$ and $b$ by $m$, obtaining integer quotients and remainders, where the remainders are between 0 and $m - 1$. That is, $a = q_1 m + r_1$ and $b = q_2 m + r_2$ , where $0 \pounds r_1 \pounds m - 1$ and $0 \pounds r_2 \pounds m - 1$.

Then it is not difficult to see that $a \equiv b \pmod{m}$ if and only if $r_1 = r_2$ . We will use the notation $a \bmod m$ (without parentheses) to denote the remainder when $a$ is divided by $m$, i.e., the value $r_1$ above. Thus $a \equiv b \pmod{m}$ if and only if $a \bmod m = b \bmod m$. If we replace $a$ by $a \bmod m$, we say that $a$ is *reduced* modulo $m$.

**REMARK** Many computer programming languages define $a \bmod m$ to be the remainder in the range $-m + 1, \ldots, m - 1$ having the same sign as $a$. For example, -18 mod 7 would be -4, rather than 3 as we defined it above. But for our purposes, it is much more convenient to define $a \bmod m$ always to be nonnegative. It is defined over Z26 since there are 26 letters in the English alphabet, though it could be defined over Zm for any modulus $m$. It is easy to see that the **Shift Cipher** forms a cryptosystem as defined above, i.e., $dK(eK(x)) = x$ for every x belongs to Z26

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

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

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

**Example:**

Plaintext: *a b c d e f.... t u v w....*

Ciphertext: *D E F G H ... Z A B C*

We call this a SHIFT CIPHER with shift (or key) 3.

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:*

$$C = \mathrm{E}(p) = (p + 3) \bmod 26$$

More generally, an additive cipher with shift (or key) *d* (also called a *shift cipher* or a *translation cipher*) is one in which each of the letters in the alphabet is encrypted as the letter that occurs *d* places further on in the alphabet and decrypted by replacing each letter by one that occurs d places earlier on in the alphabet (or 26 - *d* places further on) where, as before, *z* is followed by *abcd...*

$$C = \mathrm{E}(p) = (p + d) \bmod 26$$

Where *d* takes any value in the range 1 to 25. The decryption algorithm is simply:

$$p = \mathrm{D}(C) = (C - d) \bmod 26$$

Encryption can be done mechanically by means of a simple device consisting of a large disc on which there is a smaller disc (with the same centre) which can be rotated through d places (or *d* places back for decryption).

The key is easily remembered, but the cipher is so insecure that it is of no practical use, as an interceptor has to test at most 25 possible values of *d* to find the key.

Suppose the ciphertext reads

*AOPZ TLZZHNL PZ H MHRL.*

He tests values of *d* on the word ―*MHRL*‖ and finds that *d* = 7 yields a plaintext of ―*fake*", while *d* = 19 yields ―*toys*‖. All other shifts yield unintelligible plaintext. If he encrypts *PZ* with *d* = 7, he gets ―*is*‖, while *d* = 19 yields ―*wg*‖. So he chooses *d* = 7 and decrypts to find .. ? (Exercise!)

If he had thought for a moment, he would have spotted that *H* probably represents *a* or *i* (corresponding to *d* = 7 or *d* = 25, respectively) and, testing *d* = 7, he would have decrypted the message with ease.

We observe that the Shift Cipher (modulo 26) is not secure, since it can be cryptanalyzed by the obvious method of exhaustive key search. Since there are only 26 possible keys, it is easy to try every possible decryption rule $d_k$ until a ―meaningful‖ plaintext string is obtained.

(We know of no pairs of English words of length six or more that are translations of each other (like *fake* and *toys* above) and of only a few of length four or five.)

Monoalphabetic ciphers are very vulnerable to attack by a frequency analysis of letters, digrams (pairs of letters), trigrams (triples of letters).

Two principal methods are used in substitution cipher to lessen the extent to which the structure of the plaintext survives in the ciphertext: One approach is to encrypt multiple letters of plaintext, and the other is to use multiple cipher alphabets. PLayfair represents an example of the use of the first principal, and polyalphabetic ciphers represent the second principal.

**Exercise**

Suppose the key for a **Shift Cipher** is *K* = 11, and the plaintext is wewillmeetatmidnight. Find its ciphertext.

### I.5.1.2. Atbash cipher

The Atbash cipher is a substitution cipher with a specific key where the letters of the alphabet are reversed. I.e. all 'A's are replaced with 'Z's, all 'B's are replaced with 'Y's, and so on. It was originally used for the Hebrew alphabet, but can be used for any alphabet

The Atbash cipher is essentially a substitution cipher with a fixed key, if you know the cipher is Atbash, then no additional information is needed to decrypt the message. The substitution key is:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA

### I.5.1.3. Playfair Cipher

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone, but was named after Lord Playfair who promoted the use of the cipher. The technique encrypts pairs of letters (digraphs), instead of single letters as in the simple substitution cipher. The Playfair is significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. Frequency analysis can still be undertaken, but on the 25*25=625 possible digraphs rather than the 25 possible monographs. Frequency analysis thus requires much more ciphertext in order to work.

The Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers‘s Have His Carcase.

| | | | | |
|---|---|---|---|---|
| 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 |

In this case, the keyword is ―monarchy‖. 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. Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that would fall 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. 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. Plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the row circularly following the last. For example, *mu* is encrypted as *CM*.
4. Otherwise, each plaintext letter 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**

Using "playfair example" as the key, the table becomes:

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

Encrypting the message "Hide the gold in the tree stump":

HI DE TH EG OL DI NT HE TR EX ES TU MP

1. The pair HI forms a rectangle, replace it with BM
2. The pair DE is in a column, replace it with OD
3. The pair TH forms a rectangle, replace it with ZB
4. The pair EG forms a rectangle, replace it with XD
5. The pair OL forms a rectangle, replace it with NA
6. The pair DI forms a rectangle, replace it with BE
7. The pair NT forms a rectangle, replace it with KU
8. The pair HE forms a rectangle, replace it with DM
9. The pair TR forms a rectangle, replace it with UI
10. The pair EX (X inserted to split EE) is in a row, replace it with XM
11. The pair ES forms a rectangle, replace it with MO
12. The pair TU is in a row, replace it with UV
13. The pair MP forms a rectangle, replace it with IF

BM OD ZB XD NA BE KU DM UI XM MO UV IF

Thus the message "Hide the gold in the tree stump" becomes " BM OD ZB XD NA BE KU DM UI XM MO UV IF".

The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 X 26 = 676 digrams, so that

identification of individual digrams is more difficult. Furthermore, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis much more difficult. For these reasons, the Playfair cipher was for a long time considered unbreakable. It was used as the standard field system by the British Army in World War I and by the Germans and Americans army and other Allied forces during World War II. Despite this level of confidence in its security, the Playfair cipher is relatively easy to break because it still leaves much of the structure of the plaintext language intact. A few hundred letters of ciphertext are generally sufficient.
It‘s a substantial improvement on Vigenère cipher (see latter), as the statistics an analyst can collect are of digraphs (digrams) rather than single letters, so the distribution is much flatter, and more ciphertext is needed for an attack.
It is important to note here that, it‘s not enough for the output of a block cipher to just look intuitively ―random.‖ Playfair ciphertexts do look random, but they have the property that if you change a single letter of a plaintext pair, then often only a single letter of the ciphertext will change. One consequence is that, given enough ciphertext or a few probable words, the table (or an equivalent one) can be reconstructed.

The security of a block cipher can be greatly improved by choosing a longer block length than two characters. For example, the Data Encryption Standard (DES), which is widely used in banking, has a block length of 64 bits, which equates to eight ASCII characters and the Advanced Encryption Standard (AES), which is replacing it in many applications, has a block length of twice this.

In spite of the fact that DES and AES will be discussed latter but for the time being, we just remark that an eight byte or sixteen byte block size is not enough of itself. For example, if a bank account number always appears at the same place in a transaction format, then it‘s likely to produce the same ciphertext every time a transaction involving it is encrypted with the same key. This could allow an opponent who can eavesdrop on the line to monitor a customer‘s transaction pattern; it might also be exploited by an opponent to cut and paste parts of a ciphertext in order to produce a seemingly genuine but unauthorized transaction. Unless the block is as large as the message, the ciphertext will contain more than one block, and we will look later at ways of binding them together.

### I.5.1.3. The Vigenére cipher

Blaise de Vigenére, a French diplomat (born in 1523) first perfected the cipher we are about to consider. In 1586 he published his Treatise on Secret Writing (Traicté de Chiffres), but his work received little attention for two centuries.
In 1854, Charles Babbage (born 1791) found a way of breaking the cipher (as did, independently, Friedrich Kasiki, in 1863). We shall deal with its cryptanalysis in the next chapter.
This cipher is well known because while it is easy to understand and implement, it often appears to beginners to be unbreakable; this earned it the moniker **le chiffre indéchiffrable** (French for 'the unbreakable cipher'). Consequently, many people have tried to implement obfuscation or encryption schemes that are essentially Vigenère ciphers, only to have them broken.

#### The Algorithm

The 'key' for a vigenere cipher is a key word. e.g. 'FORTIFICATION'

The Vigenere Cipher uses the following tableau (the 'tabula recta') to encipher the plaintext:

**Table 2.3 The Modern Vigenère Tableau**

| | 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 | J |
| *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 |

To encipher a message, repeat the keyword above the plaintext:

```
FORTIFICATIONFORTIFICATIONFO
DEFENDTHEEASTWALLOFTHECASTLE
```

Now we take the letter we will be encoding, 'D', and find it on the first column on the tableau. Then, we move along the 'D' row of the tableau until we come to the column with the 'F' at the top (The 'F' is the keyword letter for the first 'D'), the intersection is our ciphertext character, 'I'.

So, the ciphertext for the above plaintext is:

```
FORTIFICATIONFORTIFICATIONFO
DEFENDTHEEASTWALLOFTHECASTLE
ISWXVIBJEXIGGBOCEWKBJEVIGGQS
```

To decrypt, pick a letter in the ciphertext and its corresponding letter in the keyword, use the keyword letter to find the corresponding row, and the letter heading of the column that contains the ciphertext letter is the needed plaintext letter. For example, to decrypt the first letter I in the ciphertext, we find the corresponding letter F in the keyword. Then, the row of F is used to find the corresponding letter I and the column that contains I provides the plaintext letter D (see the above figures).

**Example:**

For example, suppose that the plaintext to be encrypted is:

ATTACKATDAWN

The person sending the message chooses a keyword and repeats it until it matches the length of the plaintext, for example, the keyword "LEMON":

The first letter of the plaintext, A, is enciphered using the alphabet in row L, which is the first letter of the key. This is done by looking at the letter in row L and column A of the Vigenère square, namely L. Similarly, for the second letter of the plaintext, the second letter of the key is used; the letter at row E and column T is X. The rest of the plaintext is enciphered in a similar fashion:

| | |
|---|---|
| Plaintext: | ATTACKATDAWN |
| Key: | LEMONLEMONLE |
| Ciphertext: | LXFOPVEFRNHR |

Decryption is performed by finding the position of the ciphertext letter in a row of the table, and then taking the label of the column in which it appears as the plaintext. For example, in row L, the ciphertext L appears in column A, which taken as the first plaintext letter. The second letter is decrypted by looking up X in row E of the table; it appears in column T, which is taken as the plaintext letter.

Vigenère can also be viewed algebraically. If the letters A–Z are taken to be the numbers 0–25, and addition is performed modulo 26, then Vigenère encryption can be written,

$$C_i \equiv (P_i + K_i) \mod 26$$

and decryption,

$$P_i \equiv (C_i - K_i) \mod 26$$

**Strength and Limitation of Vigenère cipher**

The Vigenère cipher is effective because it masks the characteristic letter frequencies of English plaintexts, but some patterns remain.

The strength behind the Vigenère cipher is, like all polyalphabetic ciphers, to make frequency analysis more difficult. Frequency analysis is the practice of decrypting a message by counting the frequency of ciphertext letters, and equating it to the letter frequency of normal text. For instance if P occurred most in a ciphertext whose plaintext is in English one could suspect that P corresponded to E, because E is the most frequently used letter in English. Using the Vigenère cipher, E can be enciphered as any of several letters in the alphabet at different points in the message thus defeating simple frequency analysis.

The critical weakness in the Vigenère cipher is the relatively short and repeated nature of its key. If a cryptanalyst discovers the key's length then the cipher text can be treated as a series of different Caesar ciphers, which individually are trivially broken. The Kasiski and Friedman tests help determine a ciphertext's key length.

**Substitution in modern cryptography**

Substitution ciphers as discussed above, especially the older pencil-and-paper hand ciphers, are no longer in serious use. However, the cryptographic concept of substitution carries on even today. From a sufficiently abstract perspective, modern bit-oriented block ciphers (eg, DES, or AES) can be viewed as substitution ciphers on an enormously large binary alphabet. In addition, block ciphers often include smaller substitution tables called S-boxes.

## I.5.2. TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

### I.5.2.1. RAIL FENCE CIPHER

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message ―meet me after the toga party with a rail fence of depth 2, we write the following:

m e m a t r h t g p r y
e t e f e t e o a a t

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

### I.5.2.2. COLUMNAR TRANSPOSITION CIPHER

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example,

| Key: | 4 | 3 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Plaintext: | a | t | t | a | c | k | P |
| | o | s | t | p | o | n | E |
| | d | u | n | t | i | l | T |
| | w | o | a | m | x | y | Z |

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the ciphertext in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is re-encrypted using the same algorithm,

| Key: | 4 | 3 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Input: | t | t | n | a | a | p | t |
| | m | t | s | u | o | a | o |
| | d | w | c | c | i | x | k |
| | n | l | y | p | e | t | z |

Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is

01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27
28 After the first transposition we have
03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28

which has a somewhat regular structure. But after the second transposition, we have 17 09 05 27 24 16 12 07 10 02 22 20 03 25

15 13 04 23 19 14 11 01 26 21 18 08 06 28

This is a much less structured permutation and is much more difficult to cryptanalyze.

**Permutation cipher**

Let *d* be a positive integer; divide the message *M* into blocks of length *d*. Then take a permutation $\pi$ of 1, 2, 3,... , *d* and apply $\pi$ to each block. For example: Let *d* = 4 and $\pi$ = (2 4 1 3)

This means that:

$\pi \begin{pmatrix} 1234 \\ 2413 \end{pmatrix}$ or $\pi(1) = 2$, $\pi(2) = 4$, $\pi(3) = 1$, $\pi(4) = 3$.

Let *d* be a positive integer, $\pi$ a permutation of 1,2,... , *d*, so

$$\pi = \begin{pmatrix} 1 & 2 & 3 & d \\ \pi(1) & \pi(2) & \pi(3) & \pi(d) \end{pmatrix}.$$

A transposition cipher is a block cipher with block length *d* such that a plaintext block $x_1, x_2, \ldots . x_d$ is encrypted as $x_{\pi(1)} x_{\pi(2)} \ldots . x_{\pi(d)}$. (This corresponds to the definitions for DES.)

**Example**: $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix}$, $d = 4$, $\sum$ = {a,... , z}.

Plaintext: x = *he – i s – a - g r e a t - ma t h e m a t I c I a n –*

Ciphertext: y = EIH- - - S A RAGE -ATM HMTE TCAI A – IN

To decrypt, use $\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$ and find

x = he - is - a - grea tma them atic ian-

Transposition is more secure than simple substitution, but is vulnerable to attack .

## I.5.3. HILL CIPHER

The Hill cipher was invented in 1929 by Lester S. Hill. Let *m* be a positive integer, and define $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ . The idea is to take *m* linear combinations of the *m* alphabetic characters in one plaintext element, thus producing the *m* alphabetic characters in one ciphertext element.

For example, if $m = 2$, we could write a plaintext element as $x = (x_2, x_2)$ and a ciphertext element as $y = (y_1, y_2)$. Here, $y_1$ would be a linear combination of $x_1$ and $x_2$, as would $y_2$. We might take

$$y_1 = 11x_1 + 3x_2$$

$$y_2 = 8x_1 + 7x_2.$$

Of course, this can be written more succinctly in matrix notation as follows:

$$(y_1, y_2) = (x_1, x_2) \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}.$$

In general, we will take an $m \times m$ matrix $K$ as our key. If the entry in row $i$ and column $j$ of $K$ is $k_{i,j}$, then we write $K = (k_{i,j})$. For $x = (x_1, \ldots, x_m) \in \mathcal{P}$ and $K \in \mathcal{K}$, we compute $y = e_K(x) = (y_1, \ldots, y_m)$ as follows:

$$(y_1, y_2, \ldots, y_m) = (x_1, x_2, \ldots, x_m) \begin{pmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,m} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,m} \\ \vdots & \vdots & & \vdots \\ k_{m,1} & k_{m,2} & \cdots & k_{m,m} \end{pmatrix}$$

In other words, $y = xK$.
We say that the ciphertext is obtained from the plaintext by means of a *linear transformation*.
We have to consider how decryption will work, that is, how $x$ can be computed from $y$.
Readers familiar with linear algebra will realize that we use the inverse matrix $K$-1 to decrypt.
The ciphertext is decrypted using
the formula $x = yK$-1.

**How to compute $k^{-1}$**

- Compute det(K)
- Check if gcd(det(K), 26) =1
- If not, then $K^{-1}$ do not exist
- Else $K^{-1}$ is

$$\begin{pmatrix} (-1)^{1+1} K_{1,1} & \cdots & (-1)^{1+n} K_{n,1} \\ \vdots & \vdots & \vdots \\ (-1)^{1+n} K_{1,n} & \cdots & (-1)^{2n} K_{n,n} \end{pmatrix} \det(K)^{-1}$$

Example:

Suppose the key is

$$K = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}.$$

Remember that, the *inverse matrix* to an $m \times m$ matrix K (if it exists) is the matrix $\mathbf{K}^{-1}$ such that $\boldsymbol{KK}^{-1} = I_m$. Not all matrices have inverses, but if an inverse exists, it is unique. From computation,

$$K^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}.$$

Suppose we want to encrypt the plaintext *july*. We have two elements of plaintext to encrypt: (9, 20) (corresponding to *ju*) and (11, 24) (corresponding to *ly*). We compute as follows:

$$(9, 20)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (99 + 60, 72 + 140) = (3, 4)$$

and

$$(11, 24)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (121 + 72, 88 + 168) = (11, 22).$$

Hence, the encryption of *july* is *DELW*. To decrypt, Bob would compute:

$$(3, 4)\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = (9, 20)$$

and

$$(11, 22)\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = (11, 24).$$

Hence, the correct plaintext is obtained.
At this point, we have shown that decryption is possible if *K* has an inverse. In fact, for decryption to be possible, it is necessary that *K* has an inverse.

## I.6. INTRODUCTION TO CRYPTANALYSIS

As we have seen, if a message *M* has been encrypted in ciphertext *C*, an interceptor of *C* has to test at most 26 values of the key (or shift) to find the key and retrieve the original message (by exhaustive search).
An ***attack*** on a cryptosystem is an attempt to decrypt encrypted messages without knowledge of the key.

There are four basic types of attacks:

1. **Ciphertext only**: In this case the cryptanalyst possesses a part of the encrypted message but has no knowledge of the plaintext message, or of the key.
   (There are two levels of attack: To decrypt a particular message or to find the key so that all messages can be decrypted (illegally!) later on.)
2. **Known plaintext**: In this case the cryptanalyst possesses both a part of the encrypted plaintext and the corresponding ciphertext. (Here the purpose of the attack is to find the key.)
3. **Chosen plaintext**: In this case the cryptanalyst is able to choose some number of plaintexts and to see the corresponding encryptions. (The goal is to obtain the key.)
4. **Chosen ciphertext**: Here the cryptanalyst is able to choose some ciphertexts and to gain access to a decryption of the texts. (Again, the goal is to find the key.)

These four attacks are labeled to get progressively stronger; that is, (1) presents the cryptanalyst with the most difficult problem and (4) makes it easiest for him to ―crack‖ the cipher.

A *symmetric cipher* is one in which knowledge of the encryption key or (in extreme cases) the same key is used in encryption and decryption. We shall study attacks on such systems. Later we shall consider *asymmetric ciphers* where another level of attack becomes relevant: An encryption key attack is an attack on an asymmetric cipher where (by design) knowledge of the encryption key (if known) easily yields the decryption key. (The goal is to obtain the decryption key before the ciphertext is intercepted.)

A ciphertext only attacks on the substitution cipher.

**Table 1.1** Probabilities of Occurrence of the 26 Letters

| letter | probability | letter | probability |
|---|---|---|---|
| *A* | .082 | *N* | .067 |
| *B* | .015 | *O* | .075 |
| *C* | .028 | *P* | .019 |
| *D* | .043 | *Q* | .001 |
| *E* | .127 | *R* | .060 |
| *F* | .022 | *S* | .063 |
| *G* | .020 | *T* | .091 |
| *H* | .061 | *U* | .028 |
| *I* | .070 | *V* | .010 |
| *J* | .002 | *W* | .023 |
| *K* | .008 | *X* | .001 |
| *L* | .040 | *Y* | .020 |
| *M* | .024 | *Z* | .001 |

It may also be useful to consider sequences of two or three consecutive letters called *digrams* and *trigrams*, respectively. The 30 most common digrams are (in decreasing order) *TH*, *HE*, *IN*, *ER*, *AN*, *RE*, *ED*, *ON*, *ES*, *ST*, *EN*, *AT*, *TO*, *NT*, *HA*, *ND*, *OU*, *EA*, *NG*, *AS*, *OR*, *TI*, *IS*, *ET*, *IT*, *AR*, *TE*, *SE*, *HI*, and *OF*. The twelve most common trigrams are (in decreasing order) *THE ING*, *AND*, *HER*, *ERE*, *ENT*, *THA*, *NTH*, *WAS*, *ETH*, *FOR*, and *DTH*.

Here P = C = $Z_{26}$ = {0, 1,2,... , 25} and C consists of all permutations of the 26 symbols in $Z_{26}$. For each permutation $\pi \in K$ , define

$$E_\pi(x) = \pi(x), \qquad D_\pi(y) = \pi^{-1}(y)$$

where $x, y \in Z_{26}$ (x ∈P, y ∈ C) and $\pi^{-1}$ is the inverse of it.
(Recall that a permutation on a finite set $X$ = {0, 1,... , n}, simply a bijective function ($\pi: X \to X$ ) . We write:

$$\pi = \begin{pmatrix} 0 & 1 & \dots & n \\ \pi(0) & \pi(1) & \dots\dots & \pi(n) \end{pmatrix}$$

For example, if $X$ = {0, 1, 2}, the permutations of $X$ are:

(a) The identity permutation: $\pi_0 = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \end{pmatrix} = \pi^{-1}$

(b) $\pi_1 = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix} = \pi_1^{-1}$

(c) $\pi_2 = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 1 & 0 \end{pmatrix} = \pi_2^{-1}$

(d) $\pi_3 = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \end{pmatrix} = \pi_3^{-1}$

(e) $\pi_4 = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \end{pmatrix} = \pi_4^{-1}$

(f) $\pi_5 = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \end{pmatrix} = \pi_5^{-1}$

So there are 6 permutations of X = {0,1.,2}, where 6 = 3! = IXI! . In general, the number of permutations of a set X of n elements is n!

Note: 26!, the number of permutations of $Z_{26}$, is so large that, even with modern computing capacity, a ciphertext only attack by exhaustive search is infeasible.
(26! = 403 291 461 126 605 635 584 x $10^6$ ~ 4 x $10^{26}$)

**CRYPTANALYSIS OF TRANSPOSITION CIPHERS**

Transposition ciphers have the advantage over substitution ciphers that, although they preserve the frequency distribution of single letters, they destroy the digram and trigram statistic of the language.
Yet, using the *contact method*, we can use our statistical data on digrams to help us in an attack on a transposition cipher: We select from the ciphertext pairs of letters that form popular digrams (discarding those that do not, like *cg, hq, kq, sg,...*) and investigate which transpositions would bring them into contact as a likely bigram like *th* or *ck* or *in*, etc. The method can be used even if the intercepted ciphertext is quite short.

**Example:**

Suppose *d* = 5 and the first block of the cryptogram is *KTICH*. From these letters we can form common bigrams ―*th*‖, ―*ck*‖ and are with left with ―*i*‖. Now, instead of considering 5! = 120 permutations of the five letters, *k, t, i, c* and *h*, we try 3! = 6 arrangements of *th, ck* and *i* ; the solitary vowel, *i*, must come between *ck* and *th*. Now we are down to 2 choices: ―*ckith"* and ―*thick"*.

**Answer:** Plaintext message = thick.

Key = $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 3 & 4 & 2 \end{pmatrix}$

**Exercises**

1. We use the Hill Cipher with P=C=$Z_{26}$ and key

$$\begin{bmatrix} 3 & 7 \\ 4 & 9 \end{bmatrix}$$

(a) Compute the encryption of June

(b) Compute the decryption of ciphertext (6,7)

2. I) Construct the *Playfair* matrix (5x5) using the keyword: Cryptography

ii) Encrypt the following words using the above matrix.

**Telecommunication, football**

**3.** Using Vigenere cipher, decrypt ―VPXZTIQKTZWTCVPSWFDMTETIGAHLH‖ if ―cipher‖ is a key.

4.Caesar cipher using frequency analysis. Decrypt the following

i. ―kbkxeutk‖

ii. ―espntaspcsldmppymczvpy‖

5.Using Hill cipher, decrypt ―LNSHDLEWMTRW‖ if the key is

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

## CHAPTER II. DATA ENCRYPTION STANDARD (DES)

### 2.1. INTRODUCTION

Product cipher systems are block systems that combine a number of substitutions and transpositions and are considerably more difficult to ―crack‖ than the substitution and transposition systems we have considered thus far.
As encryption in such systems is performed by means of electronic computers, it is appropriate to use the binary alphabet $Z_2 = \{0, 1\}$. We refer to the elements of $Z_2$ as ―bits‖. Before encryption, the message has to be converted into a sequence of bits. There are various ways of converting alphabetic and other symbols into strings of bits. We shall use ASCII (American Standard Code for Information Interchange, pronounced ―ass-key‖) which assigns an 8-digit binary number to each symbol we wish to encrypt (e.g. A = 01000001, a = 01100001, = 11100001, etc.)

- Shannon suggested in the 1940s that strong ciphers could be built by combining substitution with transposition repeatedly. For example, one might add some key material to a block of input text, then shuffle subsets of the input, and continue in this way a number of times. He described the properties of a cipher as being *confusion* and *diffusion*— adding unknown key values will confuse an attacker about the value of a plaintext symbol, while diffusion means spreading the plaintext information through the ciphertext. Block ciphers need diffusion as well as confusion.
- In 1973 the National Bureau of Standards (NBS) (now known as the National Institute of Standards & Technology (NIST) (of the USA) called for the development of an ―algorithm to be implemented in electronic hardware devices, to be used for the cryptographic protection of computer data‖.
- The system was used to encrypt non-classified (mainly commercial) data. Several proposals were considered; after rigorous testing the proposal submitted by IBM (International Business Machines) was accepted and officially adopted in 1976. It is based on an earlier version, Lucifer, developed by Horst Feistel in the early 1970‘s and has been used (world-wide) to encrypt data, for example to protect PIN‘s (Personal Identification Numbers) of clients in banking, telephonic transfers of money, internet communications, etc.
- It was reviewed at 5-yearly intervals and was expected to be in use till 1998. In 1999 (a bit late!) fifteen systems were proposed by groups in the USA and Europe. Their specifications were published on the Internet and they were assessed by experts all over the world — a short list of five systems received even more attention (how secure are they?) and in 2000 (2001) Rijndael (a block cipher) was accepted as the Advanced Encryption Standard.
- Banks and other financial institutions are still using DES and may continue to do so for quite a while — so we‘ll study both DES and AES.

### 2.2 Introduction to Feistel Ciphers

The Feistel cipher structure, which dates back over a quarter century and which, in fact, is based on Shannon‘s proposal of 1945, is the structure used by most significant symmetric block ciphers currently in use.

Feistel proposed that we can approximate the simple substitution cipher by utilizing the concept of a product cipher, which, as mentioned above, is the performing of two or more basic ciphers in sequence in such a way that the final result or product is cryptographically

stronger than any of the component ciphers. In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations.

Before introducing the Feistel cipher, we look at first at the concepts of diffusion and confusion.

**Diffusion and Confusion**

The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system. Shannon‘s concern was to thwart cryptanalysis based on statistical analysis. The reasoning is as follows. Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message (probable words). If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or part of the key, or at least a set of keys likely to contain the exact key. In what Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used.

Shannon suggests two methods for frustrating statistical cryptanalysis: diffusion and confusion. In **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 cipher- text digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. An example of diffusion is to encrypt a message $M = m_1, m_2, m_3, \ldots$ of characters with an averaging operation:

$$y_n = \sum_{i=1}^{k} m_{n+i} \pmod{26}$$

adding k successive letters to get a ciphertext letter $y_n$. One can show that the statistical structure of the plaintext has been dissipated. Consider for example the following message *m:*

$m = 2\ \ 0\ \ \ 19\ \ 18\ 0\ \ 17\ \ 4\ \ \ 2\ \ 17$

for k = 3 we get:

$y = 21\ 11\ \ \ 11\ \ \ 9\ \ 21\ 23\ 23\ 19\ 17$

Thus, the letter frequencies in the ciphertext will be more nearly equal than in the plaintext; the digram frequencies will also be more nearly equal, and so on. In a binary block cipher, diffusion can be achieved by repeatedly performing some ***permutation on the data followed by applying a function to that permutation***; the effect is that bits from different positions in the original plaintext contribute to a single bit of ciphertext.

Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key. The mechanism of **diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce the key. On the other hand, 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. 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.

Some authors pointed out, so successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern block cipher design.

### 2.2.1 Feistel Cipher

Let *B* be a block cipher with alphabet $Z_2$ {0, 1} and block length *w*, key space **K** and encryption function $f_k$, $k \in \kappa$ k. *B* is called the underlying cipher of the Feistel cipher that we are about to define, $f_k$ is a function which has as input and output binary vectors of length *w*.
We fix a number $n \geq 1$ (*n* the number of rounds) and a method that, from any key $k \in \kappa$, generates a number of round keys, $K_1, K_2, \ldots, K_r \in \kappa$.

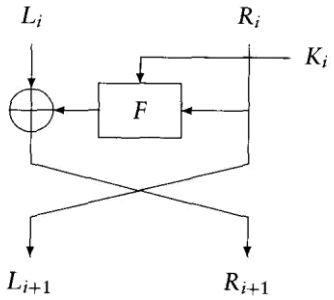
The *Feistel cipher F*, depicted in Figure 1, has as inputs an alphabet $Z_2 = \{0, 1\}$, a plaintext block of length *2w* and a key K. To calculate the encryption function $E_k$ of F, the plaintext block, is divided into two halves, $L_0$ and $R_0$. The two halves of the data pass through *n* rounds of processing and then combine to produce the ciphertext block. Each round *i* has as inputs $L_{i-1}$ and $R_{i-1}$ derived from the previous round, as well as a subkey $K_i$ derived from the overall K. In general, the subkeys $K_i$, are different from K and from each other.
Mathematically, the encryption function $E_k$ of *F* is defined as follows:

- Let P be a block of plaintext of length *2w*, split into its left half, L0, consisting of the first *w* characters, and the right half, R0, consisting of the last *w* characters in P. We write
  $P = (L_0, R_0)$.
- We then construct the sequence
  $P_i = (L_i , R_i ) = (R_{i-1} , L_{i-1} \oplus f_{ki} (R_{i-1} ))$, $i = 1,2,....n$
  (2.1)

  and set
  $E_k (P) = E_k (L_0 , R_0 ) = (R_n , L_n )$
  (Note: NOT $(L_n , R_n)$)

**To decrypt:** From (2.1) we obtain
$( R_{i-1} , L_{i-1} ) = (L_i , R_i \oplus f_{ki} (L_i ))$
(2.2)
so, using equation (6.2) in *n* rounds with the reverse key sequence, $K_r, K_{r-1}, \ldots, K_2, K_1$ we obtain $(R_0, L_0)$ and hence P from $E_k(P)$, where $P = (L_0, R_0)$.

Note that:
For a Feistel cipher, encryption and decryption follow the same rules except that the key sequence is reversed. So we need not worry about our choice of the function $f_{ki}$ in (1) — any function will do and we can concentrate on choosing $f_{ki}$ to create as much *confusion* and *diffusion* as possible.




The Feistel Principle

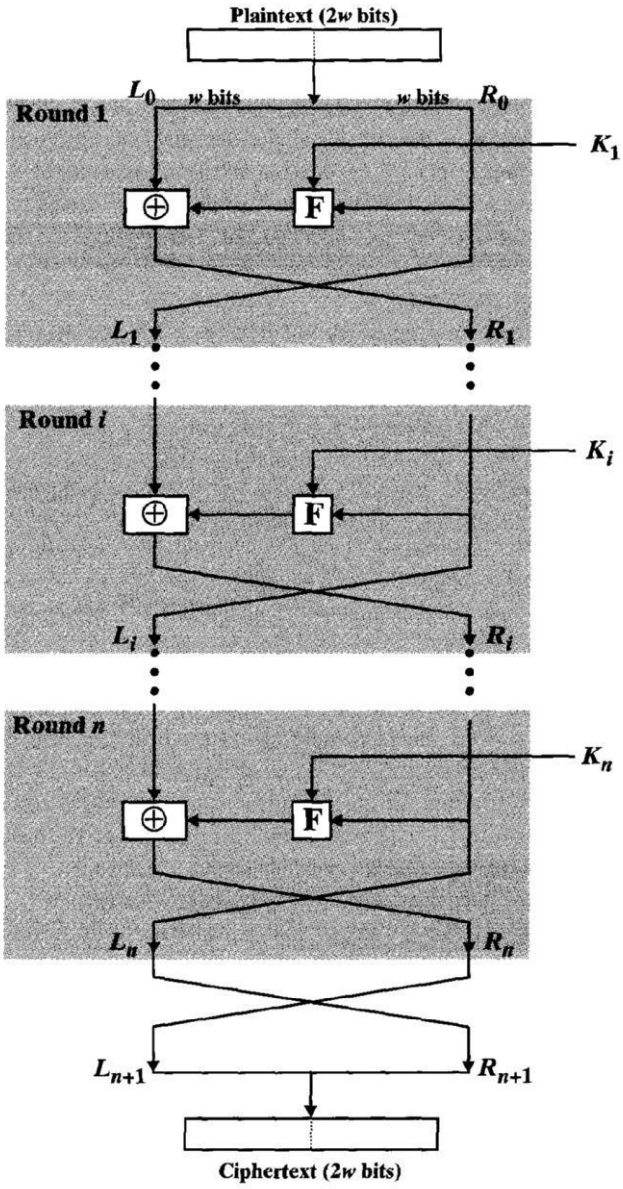



Fig.2.1 Classical Feistel Network

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed. A block size of 64 bits is a reasonable tradeoff

and has been nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

- Key size: Larger key size means greater security but may decrease encryption/decryption speed. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- Round function: Again, greater complexity generally means greater resistance to cryptanalysis.

- Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality.

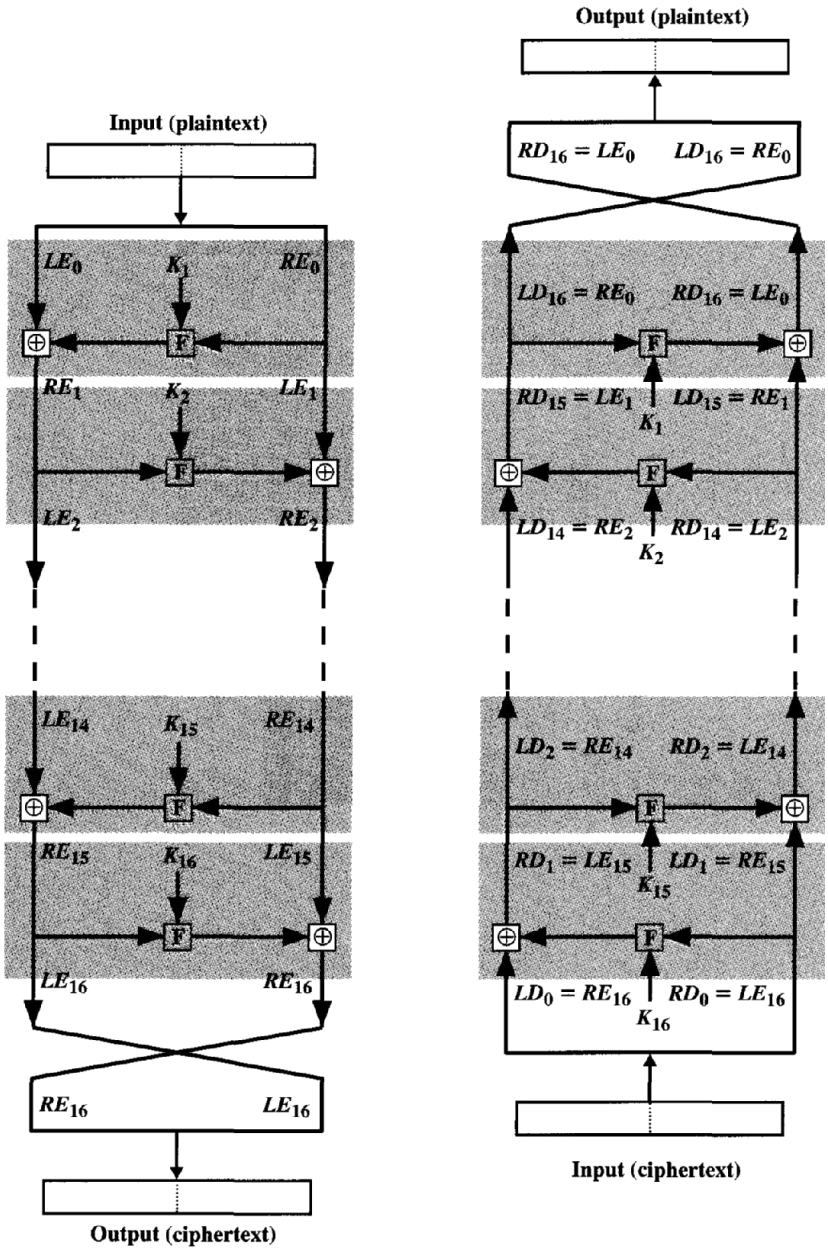



Figure 2.2 Feistel Encryption and Decryption

## 2.3 Symmetric Encryption Algorithms

The most commonly used symmetric encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-size blocks and produces a block of cipher text of equal size for each plaintext block. Many block cipher algorithms are known, the most important three are the Data Encryption Standard (DES), triple DES (3DES) and the Advanced Encryption Standard (AES). In the following section DES is introduced.

### 2.3.1 Data Encryption Standard (DES)

***Brief Description of the Algorithm:***

The plaintext is 64 bits in length and the key is 56 bits in length; longer plaintext amounts are processed in 64-bit blocks. The DES structure is a minor variation of the Feistel

network shown in Figure 2.2. There are sixteen rounds of processing. From the original 56-bit key, sixteen subkeys are generated, one of which is used each round.

For process of decryption with DES is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the DES algorithm, but use the subkeys $K_i$ in reverse order. That is, use $K_{16}$ on the first round, $K_{15}$ on the second round, and so on until $K_1$ is used on the $16^{th}$ and last iteration.

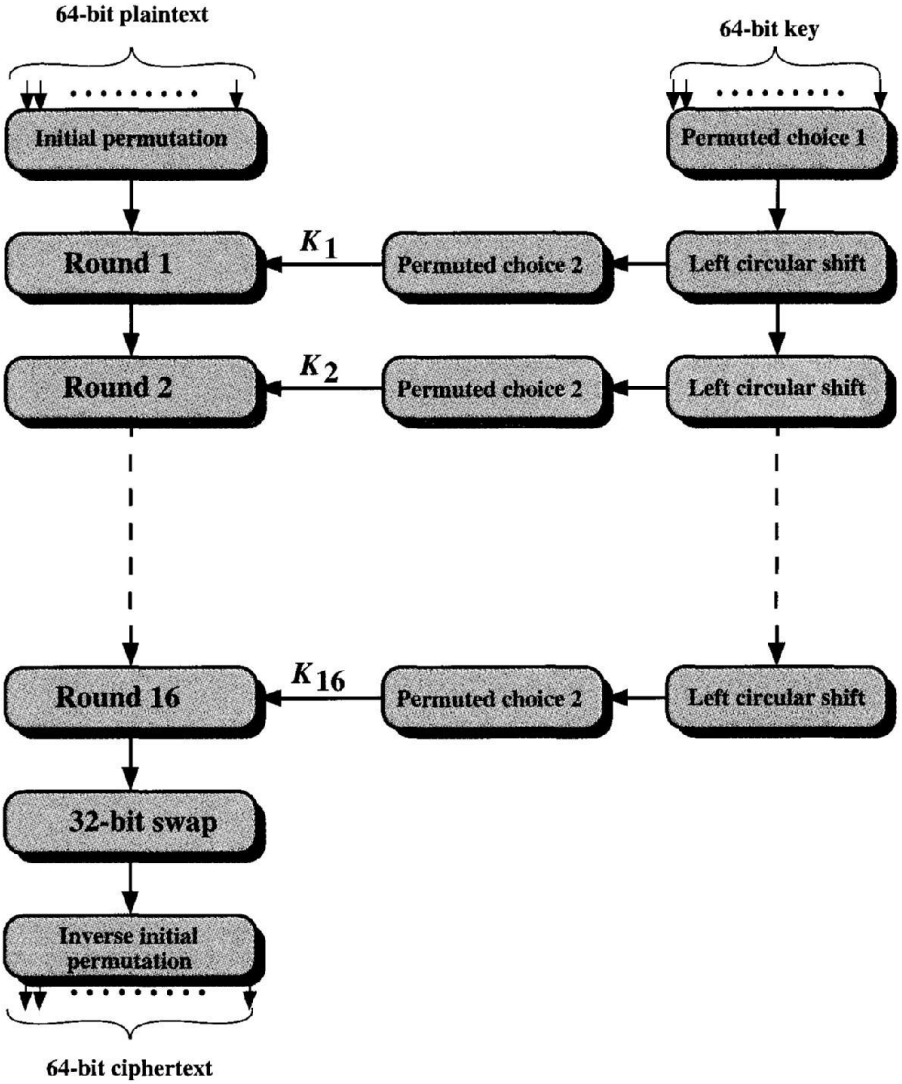



Fig. 2.3 General Decryption of DES Encryption Algorithm

In the following the details of ciphering one block of 64-bit is given:
For the cipher DES $P = C = Z_2^{64}$ , that is, the set of binary vectors of length 64.
The DES keys are all bit strings of length 64 with the following property:

- If a 64-bit DES key is divided into 8 bytes, then the sum of the eight bits in each byte is ODD (e.g. 0 1 1 1 0 1 1 0) — so seven of these bite determine the value (0 or 1) of the eighth bit. (By this way the eights bit is an Odd Parity bit. The advantage is that, if an error, one error, occurs during transmitting the byte, it is possible to detect that an error has occurred because the sum of the relevant eight bits will be even.). So we are free to choose only 8 x 7 = 56 bits of the key and the number of possible DES keys is therefore $2^{56} \sim 7.2 \times 10^{16}$.

Now the DES algorithm is implemented in three stages:

**Stage 1: Initial permutation**
Given a plaintext x in the form of a bitstream of length 64, i.e.

$x_{64}\ x_{63}\ x_{62}\ \ldots\ldots\ldots\ldots\ldots\ldots\ x_2\ x_1$

the bits of P are permuted according to a (fixed) initial permutation IP to yield $IP(x) = x_0$ (also of length 64). Write $x_0 = (L_0, R_0)$.

The initial permutation IP is as follows:

| | | 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 |

As shown in the above table, the IP is given as follows:

If $x = x_1\ x_2\ \ldots\ldots\ldots x_{64}$

Then,

$IP(x) = x_0 = x_{58}\ x_{50}\ x_{42}\ \ldots\ldots\ldots\ldots\ldots x_{15}\ x_7$

This means that the 58th bit of x is the first bit of IP(x); the 50th bit of x is the second bit of IP(x), etc.

The inverse permutation $IP^{-1}$ is:

| $IP^{-1}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

**Stage 2:** A 16-round Feistel cipher is applied to $x_0 = (L_0, R_0)$ to obtain $y = (R_{16}, L_{16})$.

In this second stage more information about:

- the encryption functions $f_{ki}$ and
- the method used to determine the keys $K_1, \ldots, K_{16}$ (key scheduling)

that are used to get the encrypted message. Figure 2.4 shows a single round of DES algorithm.

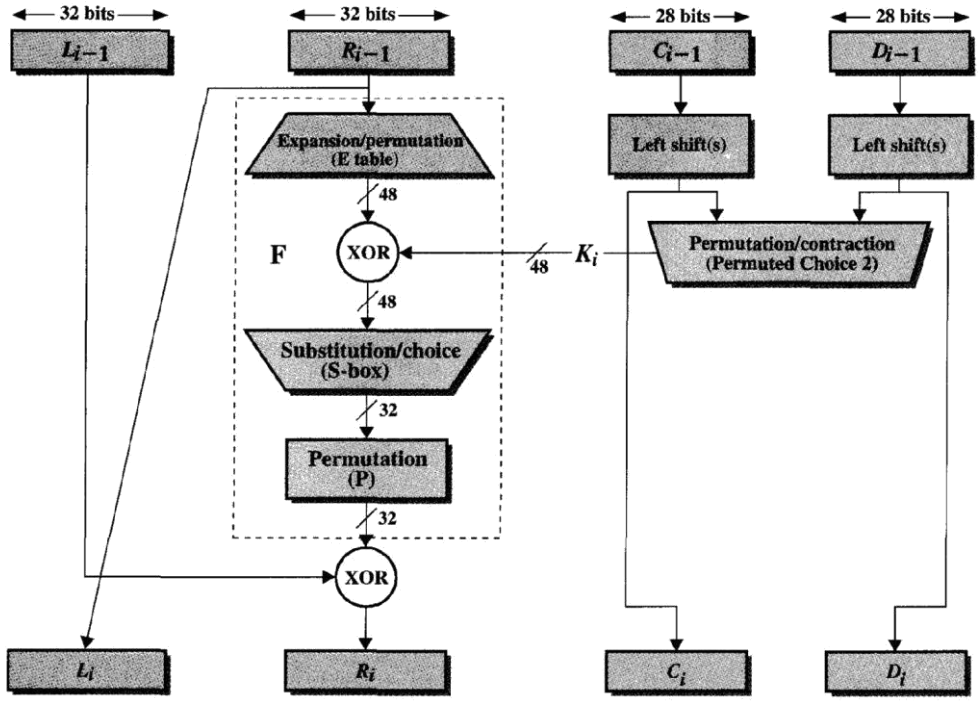



Fig. 2.4 Single Round of DES Algorithm

**A. *The Encryption functions $f_{ki}$:***

The underlying block cipher B on which this Feistel cipher is based (also called the internal block cipher of DES) has alphabet $Z_2 = \{0, 1\}$, block length 32, but its key space K is = $\{0, 1\}^{48}$. This means that K consists of 48-bit blocks.

Next we explain the encryption function $f_{Ki} : \{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$ where $K_i$ $\{0, 1\}$:

i. The 32-bit argument $R_{i-1}$ of $f_{Ki}$ is first expanded to a 48-bit string by using an expansion function E: $\{0, 1\}^{32} \rightarrow \{0, 1\}^{48}$. The expansion function E as specified by the table below can be explained as follows.

If the argument of $f_K$ is $R_{i-1} = r_1\ r_2 - r_{31}\ r_{32}$, then $E(R_{i-1}) = r_{32}\ r_1 \ldots\ r_{32}\ r_1$ with some repetitions ($r_4$ in positions 5 and 7, for example); 16 bits occur twice. This causes an ―avalanche effect‖, meaning that changing a few bits of plaintext in the beginning causes many changes in the ciphertext as we go through the 16 rounds.

| | 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 | 3 1 | 32 | 1 |

ii. Next we compute $E(R_{i-1})\ K_i$ to get a 48-bit string which we then split into eight 6-bit strings, $B_1, B_2, \ldots, B_8$ so $E(R_{i-1})\ K_i = B = B_1B_2B_3 \ldots B_8$, where for $i$ =1,2,... ,8.

iii There are eight S-boxes, $S_1$, $S_2$,... ,$S_8$, each of which is a fixed 4 x 16 array with entries from $Z_{16}$ = {0, 1, 2,... , 15}.

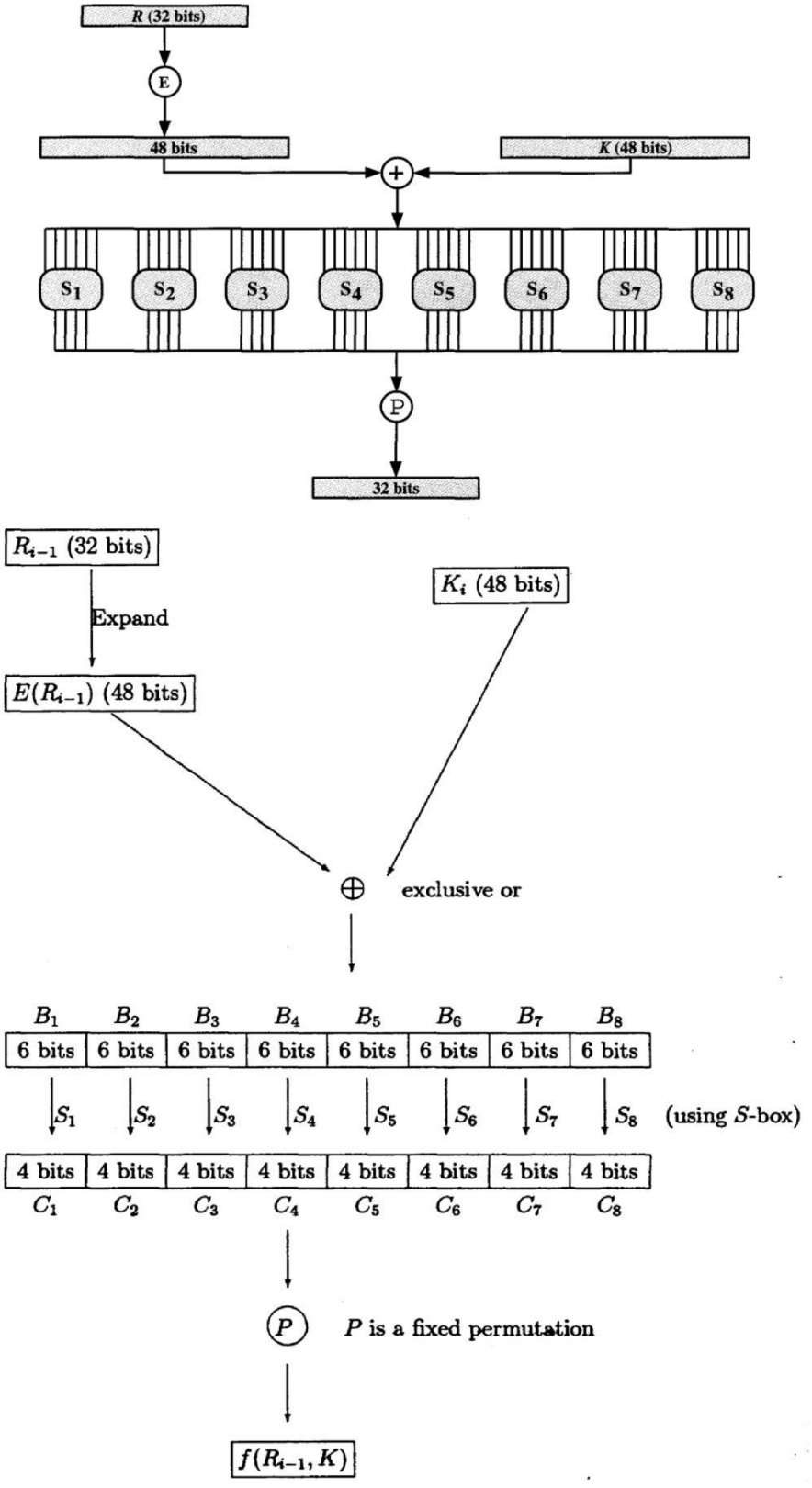



Fig. 2.5 Calculation of $S_j(B_j)$

Now, for each of the blocks *Bj* (*j* {1, 2,... , 8}), say *Bj* = $b_1b_2b_3b_4b_5b_6$ where $b_1,\ldots,b_6$ {0, 1), we compute $S_j$ ($B_j$) as follows:

- The two bits $b_1b_6$ form a number in {0, 1,2, 3}, expressed in binary form — let this be *r*. This number r represents the number of a row in $S_j$ ,where the rows are labeled 0,1,2,3.
- The four bits $b_2b_3b_4b_5$ form a 4-digit number in the range{0, 1,2,... , 15) — let this be *c*. The number c represents the number of a column in $S_j$, where the columns are labelled 0,1,... ,15. (So r = $2b_1 + b_6$ and *c* = $8b_2 + 4b_3 + 2b_4 + b_5$.)
- Then $C_j = S_j$ ($B_j$) is the entry in row r, column c of $S_j$, written $S_j$ (*r, c*), which is a 4-digit binary number in {0, 1,... , 15}. (Add 0's at the start to write it as a 4-bit number, if necessary).

The S-boxes are shown on the next page. They form the heart of DES as they are highly non-linear — they provide DES with most of its security.

**Example 6.1:**

Suppose $B_1$ = 001011 and let's compute $S_1(B_1)$.

**Solution:**

(see Figure 2.6 For $S_1$):

- Since it is required to find $S_1(B_1)$, then we use $S_1$-box shown in Figure 1.
- Getting row number: r = $b_1b_6$ = 01
  So we'll find $S_1(B_1)$ in row 1 of $S_1$, which is the second row, the first being labeled 0).
- Getting the column number *c:*
  *c* = $b_2b_3b_4b_5$ = 0101
  This is column 5 starting from column 0. So we'll find S1(B1) in column 5.
- Then $S_1(B_1)$ is the value in the $S_1$-box at the intersection of row 2 and column 6.
  From the table of $S_1$, we get:

  $S_1(B_1)$ = 2 (in decimals) = 0010 (in binary) .

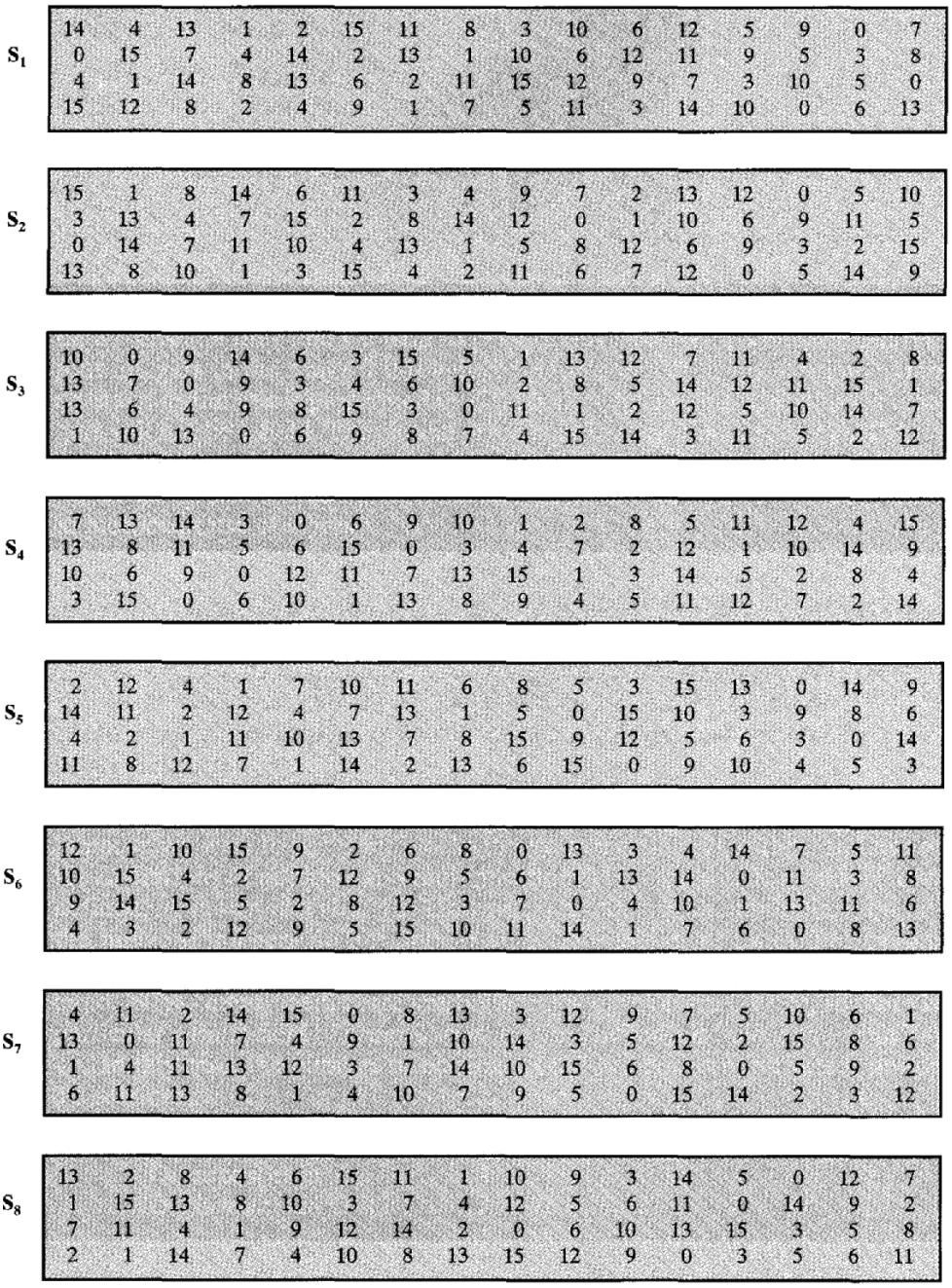
  So $S_1(B_1)$ = 0010.

  So the six digits $B_1$ = 001011 compressed to the four digits 0010 using the S-box

iv In each round, the 32-bit output from the S-boxes is permuted; a permutation performed using a so-called P-box (shown below):

| 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 | | | | | | |

The P-box maps a 32-bit string $q = q_1q_2 \ldots q_{32}$ to P($q$) = $q_{16}\, q_7 \ldots q_4\, q_{25}$.



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | 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 |

Fig. 2.6 *S-Boxes*

So we have, to illustrate $f_K$:

### *B. Key scheduling*

Let us now discuss the method of finding the sixteen 48-bit keys, $K_1$, $K_2$,... , $K_{16}$, use in the sixteen rounds of DES. (See Figure 2.7)

i. We start with a 64-bit DES key K from which we drop every eighth bit (the parity check bit in the relevant byte) and so we are left with a 56-bit key.

ii. We now permute the bits in K by using the permutation box PC1 and obtain (if $K = k_1k_2. . ….. K_{64}$)

$$PC1(K) = k_{57}k_{49}k_{41} . . . k_{20}\, k_{12}\, k_4 .$$

(Note: $k_8, k_{16}, k_{24}, \ldots, k_{64}$ do not appear.)

iii (a)Split PC1(K) into a left half $C_0$ and right half $D_0$ So

$PC1(K) = C_0D_0$.

For i = 1,2,... , 16, let

$$v_i = \begin{cases} 1 & \text{if } i \in \{1,2,9,16\} \\ 2 & \text{otherwise} \end{cases}$$

and do the following:

(b) Let $C_i$ be obtained from $C_{i-1}$ = by a shift of $v_i$ to the left (with wrap-around) — this is also called a circular shift of $v_i$ to the left

(c) Let $D_i$ be obtained from $D_{i-1}$ by a circular shift of $v_i$ to the left.

(d) Apply the permutation PC2 to the string $C_i$ $D_i$ to obtain $K_i$, so $K_i = PC2(C_i D_i)$

Note: We need not carry out these calculations as they lead to a key schedule given below:

(a) Input key (the eighth column represents the parity bits column)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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)

| PC1 | | | | | | |
|---|---|---|---|---|---|---|
| 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)

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 Shift

| 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 |

Fig.2.7 DES Key Schedule Calculation

We now display the resulting key schedule. As mentioned above, each round uses a 48-bit key comprised of the bits in K. The entries in the table below refer to the bits in K that are used in the various rounds, where $K = k_1k_2\ldots\ldots k_{64}$.

| Round 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 51 | 34 | 60 | 49 | 17 | 33 | 57 | 2 | 9 | 19 | 42 |
| 3 | 35 | 26 | 25 | 44 | 58 | 59 | 1 | 36 | 27 | 18 | 41 |
| 22 | 28 | 39 | 54 | 37 | 4 | 47 | 30 | 5 | 53 | 23 | 29 |
| 61 | 21 | 38 | 63 | 15 | 20 | 45 | 14 | 13 | 62 | 55 | 31 |

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

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

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

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

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

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

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

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

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

| Round 8 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 36 | 41 | 60 | 50 | 10 | 43 | 59 | 18 | 57 | 35 | 9 | 3 |
| 58 | 25 | 52 | 51 | 34 | 19 | 49 | 27 | 26 | 17 | 44 | 2 |
| 12 | 54 | 61 | 13 | 31 | 30 | 6 | 20 | 62 | 47 | 45 | 23 |
| 55 | 15 | 28 | 22 | 37 | 46 | 39 | 4 | 7 | 21 | 14 | 53 |

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

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

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

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

| Round 16 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | 59 | 42 | 3 | 57 | 25 | 41 | 36 | 10 | 17 | 27 | 50 |
| 11 | 43 | 34 | 33 | 52 | 1 | 2 | 9 | 44 | 35 | 26 | 49 |
| 30 | 5 | 47 | 62 | 45 | 12 | 55 | 38 | 13 | 61 | 31 | 37 |
| 6 | 29 | 46 | 4 | 23 | 28 | 53 | 22 | 21 | 7 | 63 | 39 |

**Stage 3**

Apply the inverse of the initial permutations to $R_{16}L_{16}$ to obtain the 64-bit cipher text, y. So:

$y = IP^{-1} (R_{16} L_{16})$.

**Decryption:**

To decrypt y (64 bits) under key K, the above steps are carried out with one modification: We use the keys in the order $K_{16}, K_{15},\ldots , K_1$ . This means that key $K_{16-i+1}$ is used in round i of the decryption.

**Security**

Today DES can only be regarded as secure if triple encryption is used:
For any block cipher the so-called E — D — E triple encryption can be used to increase the security of the system:
For example: A plaintext x is encrypted as

$$y = E_{k1}(D_{k2}(E_{k3}(x))),$$

where $k_1$, $k_2$, $k_3$ are three keys, $E_k$ the encryption function and $D_k$ the decryption function for key $k_i$ (i = 1, 2,3).
For the triple DES we let $k_1 = k_3$.

To increase security, if we have to encrypt large amounts of data, we should not encrypt the blocks individually — some form of ―block chaining‖ should be used and will be discussed in the next section.

It is also unwise to use the same key for many messages (preferably a new key should b used for each session). This would present a serious problem if it were not for the methods of key exchange offered by public key cryptography which we study later on.

**Example 2.2**

We illustrate the DES algorithm by way of an example.

We use DES to encrypt the plaintext p = 0123456789ABCDEF. Its binary expansion is

0 0 0 0 0 0 0 1
0 0 1 0 0 0 1 1
0 1 0 0 0 1 0 1
0 1 1 0 0 1 1 1
1 0 0 0 1 0 0 1
1 0 1 0 1 0 1 1
1 1 0 0 1 1 0 1
1 1 1 0 1 1 1 1

The application of *IP* yields

1 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0
1 1 0 0 1 1 0 0
1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0
1 0 1 0 1 0 1 0
1 1 1 1 0 0 0 0
1 0 1 0 1 0 1 0

So we obtain

L0 = 11001100000000001100l10011111111,
R0 = 11110000101010101111000010101010.

We use the DES key

0 0 0 1 0 0 1 1
0 0 1 1 0 1 0 0
0 1 0 1 0 1 1 1
0 1 1 1 1 0 0 1
1 0 0 1 1 0 1 1
1 0 1 1 1 1 0 0
1 1 0 1 1 1 1 1
1 1 1 1 0 0 0 1

We compute the first round key. We have

C0 = 1111000011 001100101010101111,
D0 = 0101010101100110011110001111,
C1 = 1110000110011001010101011111,
D1 = 1010101011001100111100011110

and therefore

K1 = 000110110000000101l10111l1111110001110000001110010.

Using this key, we obtain

$E(R_0) \oplus K_1$ = 011000010001011110111010100001100110010100100111,

$f(R_0, K_1)$ = 00100011010010101010100110111011,

and finally

$R_1$ = 11101111010010100110010101000100

The other rounds are computed analogously.

**Tutorial.** Compute the second and the third rounds

### 2.4. Triple DES

Triple DES (3DES) was first standardized for use in financial applications in ANSI standard X9.17 in 1985. 3DES was incorporated as part of the Data Encryption Standard in 1999, with the publication of FIPS PUB 46-3.

3DES uses three keys and three executions of the DES algorithm. The function follows an encrypt-decrypt-encrypt (EDE) sequence (Figure 2.8):

$$C = E_{K3}[D_{K2}[E_{KI}[P]]]$$

Where

C = ciphertext

P= plaintext

$E_K[X]$ = encryption of X using key K

$D_K[Y]$ = decryption of Y using key K

Decryption is simply the same operation with the keys reversed

$$P = D_{K1}[E_{K2}[D_{K3}[C]]]$$

There is no cryptographic significance to the use of decryption for the second stage of 3DES encryption. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES:

$$C = E_{K1}[D_{KI}[E_{K1}[P]]] = E_{K1}[P]$$

With three distinct keys, 3DES has an effective key length of 168 bits. FIPS 46-3 also allows for the use of two keys, with $K_1 = K_3$ this provides for a key length of 112 bits. FIPS 46-3 includes the following guidelines for 3DES:

- 3DES is the FIPS approved symmetric encryption algorithm of choice.
- The original DES, which uses a single 56-bit key, is permitted under the standard for legacy systems only. New procurements should support 3DES.
- Government organizations with legacy DES systems are encouraged to transition to 3DES.
- It is anticipated that 3DES and the Advanced Encryption Standard (AES) will coexist as FIPS-approved algorithms, allowing for a gradual transition to AES.

It is easy to see that 3DES is a formidable algorithm. Because the underlying cryptographic algorithm is DEA, 3DES can claim the same resistance to cryptanalysis based on the algorithm as is claimed for DEA. Further, with a 168-bit key length, brute-force attacks are effectively impossible.

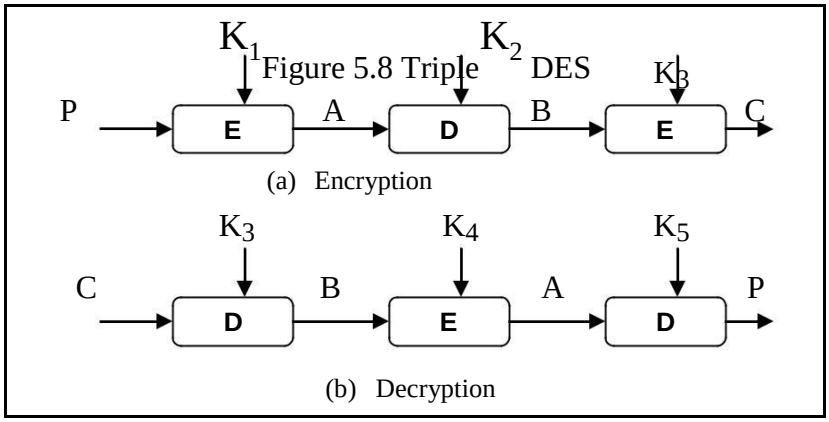



Figure 5.8 Triple DES

Fig. 2.8 Triple DES

**Advantages and drawbacks of 3DES**

3DES has two attractions that assure its widespread use over the next few years. First, with its 168-bit key length, it overcomes the vulnerability to brute-force attack of DEA. Second, the underlying encryption algorithm in 3DES is the same as in DEA. This algorithm has been subjected to more scrutiny than any other encryption algorithm over a longer period of time, and no effective cryptanalytic attack based on the algorithm rather than brute force has been found. Accordingly, there is a high level of confidence that 3DES is very resistant to cryptanalysis. If security were the only consideration, then 3DES would be an appropriate choice for a standardized encryption algorithm for decades to come.

The principal drawback of 3DES is that the algorithm is relatively sluggish in software. The original DEA was designed for mid-1970s hardware implementation and does not produce efficient software code. 3DES, which has three times as many rounds as DEA, is correspondingly slower. A secondary drawback is that both DEA and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

## 2.5. Advanced Encryption Standard

### The Rijndeal Cryptosystem

Because of the above mentioned drawbacks, 3DES is not a reasonable candidate for long-term use. It was important to find a replacement for it. The replacement, which is called ―Advanced Encryption Standard (AES)‖ have security strength equal to or better than 3DES and significantly improved efficiency. It is a symmetric block cipher with a block length of 128 bits and supports key lengths of 128, 192, and 256 bits. It is an SP-network; in order to specify it, we need to fix the S-boxes, the linear transformation between the rounds, and the way in which the key is added into the computation.

The AES is developed by Dr. Joan Daemen and Dr. Vincent Rijmen, and is called ―Rijndael‖. Beside the high security, the algorithm is computationally efficient, needs reasonable memory size and flexible.

Rijndael uses a single S-box which acts on a byte input to give a byte output.

## 2.6. Other symmetric block cipher

**IDEA**

The International Data Encryption Algorithm (IDEA) is a symmetric block cipher developed by Xuejia Lai and James Massey of the Swiss Federal Institute of Technology in 1991. IDEA

uses a 128-bit key. IDEA differs markedly from DES both in the round function and in the subkey generation function. For the round function, IDEA does not use S-boxes. Rather, IDEA relies on three different mathematical operations: XOR, binary addition of 16-bit integers, and binary multiplication of 16-bit integers. These functions are combined in such a way as to produce a complex transformation that is very difficult to analyze and hence very difficult to cryptanalyze. The subkey generation algorithm relies solely on the use of circular shifts but uses these in a complex way to generate a total of six subkeys for each of the eight rounds of IDEA.

Because IDEA was one of the earliest of the proposed 128-bit replacements for DES, it has undergone considerable scrutiny and so far appears to be highly resistant to cryptanalysis. IDEA is used in Pretty Good Privacy (PGP) as one alternative and is also used in a number of commercial products.

**Blowfish**

Blowfish was developed in 1993 by Bruce Schneier. Blowfish was designed to be easy to implement and to have a high execution speed. It is also a very compact algorithm that can run in less than 5K of memory. An interesting feature of Blowfish is that the key length is variable and can be as long as 448 bits. In practice, 128-bit keys are used. Blowfish uses 16 rounds.

Blowfish uses S-boxes and the XOR function, as does DES, but also uses binary addition. Unlike DES, which uses fixed S-boxes, Blowfish uses dynamic S-boxes that are generated as a function of the key. In Blowfish, the subkeys and the S-boxes are generated by repeated application of the Blowfish algorithm itself to the key. A total of 521 executions of the Blowfish encryption algorithm is required to produce the subkeys and S-boxes. Accordingly, Blowfish is not suitable for applications in which the secret key changes frequently.

Blowfish is one of the most formidable symmetric encryption algorithms so far implemented, because both the subkeys and the S-boxes are produced by a process of repeated applications of Blowfish itself, which thoroughly mangles the bits and makes cryptanalysis very difficult. So far, there have been a few published papers on Blowfish cryptanalysis, but no practical weaknesses have been found.

Blowfish is used in a number of commercial applications.

**RC5**

RC5 was developed in 1994 by Ron Rivest. RC5 is defined in RFC 2040 and was designed to have the following characteristics:

- **Suitable for hardware or software**: RC5 uses only primitive computational operations commonly found on microprocessors.
- **Fast**: To achieve this, RC5 is a simple algorithm and is word oriented. The basic operations work on full words of data at a time.
- **Adaptable to processors of different word lengths:** The number of bits in a word is a parameter of RC5; different word lengths yield different algorithms.
- **Variable number of rounds**: The number of rounds is a second parameter of RC5. This parameter allows a tradeoff between higher speed and higher security.
- **Variable-length key:** The key length is a third parameter of RC5. Again, this allows a tradeoff between speed and security.
- **Simple**: RC5‘s simple structure is easy to implement and eases the task of determining the strength of the algorithm.

- **Low memory requirement**: A low memory requirement makes RC5 suitable for smart cards and other devices with restricted memory.
- **High security**: RC5 is intended to provide high security with suitable parameters.
- **Data-dependent rotations**: RC5 incorporates rotations (circular bit shifts) whose amount is data dependent. This appears to strengthen the algorithm against cryptanalysis.

RC5 is used in a number of products from RSA Data Security, Inc.

**Review Questions**

1. What is the difference between stream cipher and block cipher?
2. What is the difference between diffusion and confusion?
3. What is the purpose of s-boxes in DES?
4. This problem provides a numerical example of encryption using a one-round version of DES. We start with the same bit pattern for the key K and the plaintext, namely:0123456789ABCDEF
    a. Derive K1
    b. Derive L0, R0
    c. Expand R0 to get E(R0)
    d. Calculate A=E(R0)$\otimes$K1
    e. Group the 48-bits of (d) into sets of 6 bits and evaluate the corresponding S-box substitutions.
    f. Concatenate the results of (e) to get 32-bit result, B
    g. Apply the permutation to get P(B)
    h. Calculate L1 and R1.

## CHAPTER 3. FINITE FIELDS AND PRIME NUMBER

### 3.1. Group

A **group** $G$, sometimes denoted by $\{G, \cdot\}$ is a set of elements with a binary operation, denoted by $\cdot$, that associates to each ordered pair $(a, b)$ of elements in $G$ an element $(a \cdot b)$ in $G$, such that the following axioms are obeyed:[1]

[1] The operator $\cdot$ is generic and can refer to addition, multiplication, or some other mathematical operation.

| | |
|---|---|
| **(A1)** Closure: | If $a$ and $b$ belong to $G$, then $a \cdot b$ is also in $G$. |
| **(A2)** Associative: | $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a, b, c$ in $G$. |
| **(A3)** Identity element: | There is an element $e$ in $G$ such that $a \cdot e = e \cdot a = a$ for all $a$ in $G$. |
| **(A4)** Inverse element: | For each $a$ in $G$ there is an element $a'$ in $G$ such that $a \cdot a' = a' \cdot a = e$. |

### 3.2. Modular arithmetic

Given any positive integer $n$ and any nonnegative integer $a$, if we divide $a$ by $n$, we get an integer quotient $q$ and an integer remainder $r$ that obey the following relationship:

**Equation 4-1**

$$a = qn + r \quad 0 \le r < n; q = \lfloor a/n \rfloor$$

**Example: 73 mod 23=4**

Modular arithmetic exhibits the following properties:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n)\ (b \bmod n)] \bmod n = (a\ b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

### 3.3. Divisors

We say that a nonzero b divides a if a=mb for some m, where a,b, and m are integers. That is , b divides a if there is no remainder on division. The notation is commonly used to mean b divides a. Also, ifb/a, we say that b is a divisor of a.

**Example:**

The divisors of 24 are the following: 1,2,3,4,6,8,12,24

### 3.4. Euclidean Algorithm

On 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.

**Greatest Common Divisor**

Recall that nozero b is defined to be a divisor of a if a=mb for some m, where a,b, and m are integers. We will use the notation gcd(a,b) to mean greatest common divisorof a and b. The positive integer c is said to be the greatest common divisor of a and b if

1. c is a divisor of a and b
2. any divisor of a and b is a divisor of c.

gcd(a,b)=gcd(b,a)=gcd(a,b mod a)

Examples:

1. gcd(60,24)=gcd(24,12)=gcd(12,0)=12
2. **Using Euclidean Algorithm, Compute GCD(32,40)**

**Solution**

GCD(32,40)=GCD(32,8)=GCD(8,0)=8

Thus, GCD(32,40)=8

Conclusion

Gcd(a,b)= { gcd(a,b mod a) for b>a ; a for b=0

**Algorithm**

Euclid(a,b)

1. A ← a; B ← b

2. If B=0 return A=gcd(a,b)

3. R=A mod B

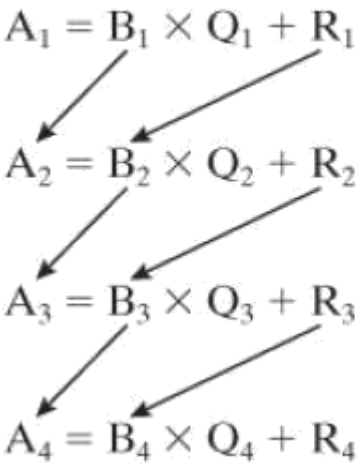
4. A ← B

5. B ← R

6. goto 2

**3.5. Extended Euclidean Algorithm**

The algorithm has the following progression:




**Example**

Using Extended Euclidean Algorithm, compute GCD( 4884, 4446)

4884=4446*1+438
4446=438*10+66
438=66*6+42
66=42*1+24
42=24*1+18
24=18*1+6
18=6*3+0
Thus GCD(4884,4446)=6

### 3.6. Prime number

An integer p>1 is a prime number if and only if its only divisors are ±1 and ±p.

$A=p_1^{a1}p_2^{a2}p_3^{a3}\ldots..p_t^{at}$ where $p_1,p_2,p_3,\ldots p_t$ are prime numbers. You can calculate gcd(a,b) if you express each integer as the product of primes.

Example:

GCD(80,64)

$80=2^4.5^1$ $64=2^6$ hence GCD(80,64)=$2^4$=16

**Exercises**

1. Find the value of x such that 7x ≡1 mod 160
2. Using Extended Euclidean Algorithm calculate gcd (2346,4468)
3. Calculate $3^{201}$ mod 11
4. Solve 51x+100y =1

**Programming**

1. Using Euclidean algorithm, Write a c/c++ program that prompts the user to enter two positive integers and display gcd(a,b)
2. Using Extended Euclidean Algorithm, Write a c/c++ program that prompts the user to enter two positive integers and display gcd(a,b)
3. Using divisors concepts, Write a c/c++ program that prompts the user to enter two positive integers and display gcd(a,b)

   NB: gcd(a,b)= max(Divisors a ⋂ Divisors b)

4. Write a c/c++/Java program to display all prime numbers less than n.

## CHAPTER 4. PUBLIC KEY CRYPTOGRAPHY

### 4.1 Public Key Cryptosystems

Thus far we have dealt with ***symmetric*** cryptosystems, namely, systems in which the encryption key $K_E$ and the decryption key $K_D$ are the same or those in which $K_D$ can easily be computed from $K_E$.

When Alice and Bob use a symmetric cryptosystem, they have to exchange a secret key *K*, for example on a secure channel or by courier. Now suppose there are *n* people in a system, all of whom want to exchange information securely — each pair has to have a secret key, so [n (n-1)/2] secret key exchanges have to take place; for *n* = 1000, this is 499,500, and each has to store *n* - 1 secret keys securely.

Another possibility for organizing the key exchange is to use a key centre *C* and to let each of the users exchange a secret key with the key centre. So, if A wants to send a message to *B, A* sends it to *C* who passes it on to *B* (*C*, knowing A‘s secret key, first decrypts A‘s encrypted message, then encrypts it again with B‘s secret key and sends B this encryption of A‘s message.) Now only *n* keys are exchanged and each user has to keep only one key securely stored, but the key centre has to store the n keys securely and knows the contents of all secret messages that pass through it.

In 1976, Diffie and Hellman published a marvelous paper, ―New directions in cryptography‖, in which they proposed a new kind of cipher system, namely a ***PUBLIC KEY CRYPTOSYSTEM.***

This is an asymmetric cryptosystem in which the encryption and decryption keys, $K_E$ and $K_D$, are distinct and the computation of $K_D$ from $K_E$ is infeasible. It works as follows:

- Each user, say X, in the system has a pair of keys, ($S_x$, $P_x$), one of which, $S_X$, is known only to X (this is X‘s private key or secret key); the key $P_x$ is the public key of X and is published in a directory, available to all users of the system. So all users know $P_x$, but only X known $S_x$. The crucial property of public key cryptography is that IT IS INFEASIBLE TO FIND *S* FROM $P_X$.

- All users of the system use the same encryption algorithm, E, and the same decryption algorithm, D, as follows:
  Say user B (Bob) wishes to send an encrypted message *M* to A (Alice).
  i. B looks up A‘s public key, $P_A$, in a directory to which all users of the system hay access.
  ii. Bob encrypts his message *M* as a cryptogram $C = E(M, P_A)$ and sends C to A across an open channel.

iii A receives $C = E(M, P_A)$ and decrypts it by finding $D(C, S_A)$, using her secret key, $S_A$. So she obtains

$$M = D(E(M,P_A),S_A).$$

Since $S_A$ is known only to Alice, only she can decrypt the message.

Note:

The security of the system depends on the use of functions *E* and *D* that should have the following properties:

i. For a given *M* and $P_x$, it should be easy to compute $C = E(M, P_x)$.
ii. If only C is known, it should be computationally infeasible to find *M*.
iii If C and $S_x$ are known, it should be easy to compute $D(C, S_x) = M$.
iv It should be easy to generate pairs $(S_x, P_x)$ of private/public keys so that too many such pairs exist for any enemy to set up a ―look-up‖ table.

Diffie and Hellman did not produce examples of such systems in 1976, but in 1978 Merkle and Hellman produced a knapsack-based system which is now known to be insecure and is not in use.

The most widely used public cryptosystem is the RSA system, introduced by **R**ivest, **S**hamir and **A**delman (Ron Rivest, Adi Shamir and Len Adelman) in the 1970‘s and patented by then in 1983. It is based on the belief that there is no fast way of factoring numbers that are the product of two large primes.

From the above discussion and from Figure 4.1, a public-key encryption scheme has six ingredients:

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm**: The encryption algorithm performs various transformations on the plaintext.
- **Public and private key**: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext**: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

As the names suggest, the public key of the pair is made public for others to use, while the private key is known only to its owner. A general-purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption.

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 4.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice‘s public key.

4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice‘s private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user protects his or her private key, incoming communication is secure. At any time, a user can change the private key and publish the companion public key to replace the old public key.

The key used in conventional encryption is typically referred to as a secret key. The two keys used for public-key encryption are referred to as the public key and the private key. Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with conventional encryption.

**Digital Signature**

Public-key encryption can be used in another way, as illustrated in Figure 4.b. Suppose that Bob wants to send a message to Alice and, although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. In this case Bob uses his own private key to encrypt the message. When Alice receives the ciphertext, she finds that she can decrypt it with Bob‘s public key, thus proving that the message must have been encrypted by Bob. No one else has Bob‘s private key and therefore no one else could have created a ciphertext that could be decrypted with Bob‘s public key. Therefore, the entire encrypted message serves as a digital signature. In addition, it is impossible to alter the message without access to Bob‘s private key, so the message is authenticated both in terms of source and in terms of data integrity.

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender‘s private key, it serves as a signature that verifies origin, content, and sequencing. A secure hash code such as SHA-1 can serve this function.

It is important to emphasize that the digital signature does not provide confidentiality. That is, the message being sent is safe from alteration but not safe from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, there is no protection of confidentiality because any observer can decrypt the message by using the sender‘s public key.

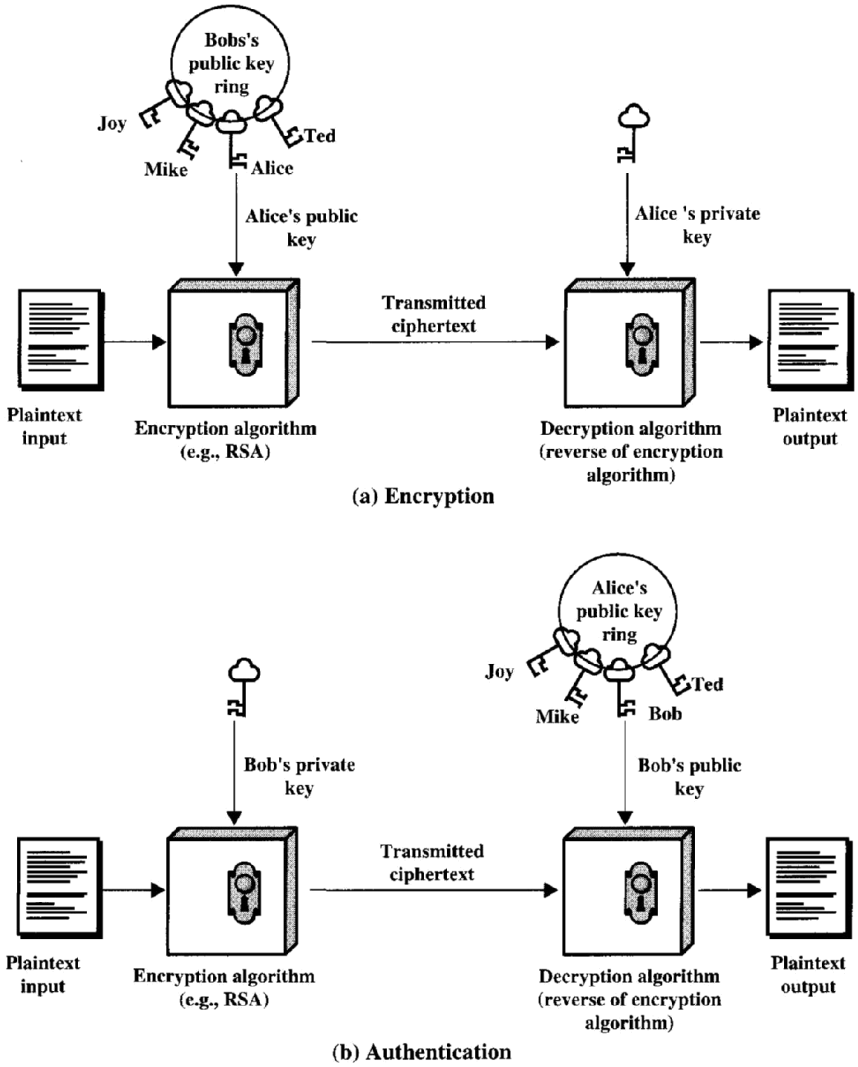



Fig. 4.1 Public-Key Cryptography

Before we present the RSA system, we observe that, although public-key cryptosystems have many advantages, they are not widely used for general-purpose encryption and decryption of long messages as that would require too much time and too much memory for most computers. However, public-key cryptosystems are widely used to encrypt keys (see Diffie-Heliman key exchange, for instance). These keys can then be used in symmetric ciphers (like DES) and long messages exchanged efficiently and securely.

## 4.2 One-Way Functions (OWF)

Before we introduce PKC, we need to deal with the notions of one-way function (OWF):

**Definition**

A one-way function $f : S \rightarrow T$ (where $S$ and $T$ are any two sets) is a function such that

i. for each $x \in S$, the value $f(x)$ is easy to compute,

ii. if $y$ is given, $y \in T$, and it is known that $y = f(x)$, it is computationally infeasible find $x$ for a large proportion of the values of $y$ in $T$.

The most useful kind of OWF is a bijection, in this case ISI = ITI = N and we shall assure that N is finite. Now, theoretically, for each $x \in S$, $f(x)$ can be calculated and the value of $f(x)$ (for $x \in S$) ordered (in increasing order) and stored. (Ordering them requires in the order of N log N steps.) For very large values of N, this is not feasible — Suppose $S$ consists of all n-digit binary numbers with $n \sim 200$ (so $N \sim 2^{200}$); then the procedure of ordering and storing the values $f(x)$ is infeasible. For example, if only one molecule is used to store a bit of information (0 or 1), then this would require more molecules than exist in our solar system (Kayes, 1975).

Let‘s consider two candidates for OWF‘s:

(1) Let $p$ be a large prime number and $f(x)$ a polynomial in x, of high degree, $x \in S = Z_p = \{0,1,2,\ldots,p-1\}$.
Then it is easy to calculate $f(x)$ for all $x \in S$, but usually hard to solve $f(x) = y$ for x.

(2) Let S consist of all ordered pairs $(x, y)$ of prime numbers with $N1 \le x \le y \le N_2$ , where $N_1$ and $N_2$ are very large prime numbers. Let $f: S \rightarrow T \subseteq N$, $f(x, y) = xy$. It is easy to calculate $f(x, y)$ for all $(x, y) \in S$, but, if the product $xy$ is given, it is (in general infeasible to find $x$ and $y$).

We shall now consider an application:

**One-Way Function, an Application: The password problem:**

One of the first applications of OWF‘s was to solving the problem of the security of passwords in an electronic authentication scheme.
Consider an electronic system to which each user gains access by logging in, supplying both a user name, $U_i$, and a password, $P_i$. It would be dangerously insecure to store a list of user names with their passwords, $(U_i, P_i)$ in unencrypted form in a file in the relevant compute (It would be too easy for a hacker to gain access to the list.)

Now let $f$ be an OWF whose domain is the list of passwords. When a new user, $U_i$, join the system, he/she chooses a password, $P_i$. The computer then stores the pair $(U_i, f(P_i))$ and does NOT store $P_i$. (Note that $f(P_i)$ is easily computed from $P_i$, but that it is infeasible to find $P_i$ if $f(P_i)$ is known.)

To gain access to the system thereafter, the user types in $U_i$ and $P_i$. The computer calculate $f(P_i)$ and compares it with the entry in the pair $(U_i, f(P_i))$ which is stored in it. Agreement allows the user access to the system.
Even if an intruder gains access to the pair $(U_i, f(P_i))$ stored in the system, he/she can not find the password $P_i$ from $f(P_i)$, because $f$ is a OWF, and so the intruder cannot log in as $U_i$.

For example: User $U_i$ may choose as password $P_i$ a pair $(p_i, q_j)$ of large primes, where the system‘s OWF $f$ is defined by $f(p, q) = pq$. The user types in $U_i$ ,$p_i$ , $q_i$, the computer calculates $f(p_i, q_i) = p_i q_i$ and verifies that the pair $(U_i, p_i q_i)$ is listed in the system before allowing access to the system.

## 4.3 RSA

The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman. It relies for its security on the difficulty of the integer factoring problem. Each user generates two distinct large primes $p$ and $q$, integers $e$ and $d$ such that $ed \equiv 1 \bmod (p-1)(q-1)$, and computes $n = pq$. The pair of integers $(n, e)$ is made public. The corresponding $p$, $q$, and $d$ are kept secret. This construction satisfies the requirement that the public key can efficiently be derived from

the secret key. The primes *p* and *q* can be derived from (*n, e*) and d. After generation of (n, e) and *d*, they are in principle no longer needed. The above means that the generation of the keys in RSA takes place as follows:

1. Picks two large primes, *p* and *q*, of roughly the same size and computes $n = pq$ and $\phi(n) = (p - 1)(q - 1)$.
2. Selects a random integer *e*, $1 < e < \phi(n)$ ) such that *gcd*(e, $\phi(n)$ ) = 1. (So *e* must be relatively prime to $\phi(n)$ .)
3. Computes the unique integer *d* such that $1 < d < \phi(n)$ and $ed \equiv 1 \pmod{(n)}$.
4. Publishes the public key, which is the pair of integers (*n, e*). The private key is *d*. The

**Encryption.** A message $m \in Z/nZ$ intended for the owner of public key (*n, e*) is encrypted as $E = m^e \in Z/nZ$. The resulting *E* can be decrypted by computing $D = E^d \in Z/nZ$. It can be proved using $ed \equiv 1 \bmod (p-1)(q-1)$, Fermat's little theorem, and the Chinese remainder theorem that $D = m$.

This can be simplified as follows:

For B to send a message to A using RSA system he follows the following:

1. B looks up A's public key: $P_A$ ($n_A$ , $e_A$ ).
2. B represents his message, *m*, as an integer in the interval [0, n — 1]. (If *m* is too large, he divides it into blocks.)
3. B encrypts M into a cryptogram *C* by the rule $C = m^{eA} \pmod{n_A}$.
4. A decrypts by using the private key $d_A$ and the formula

$$m = C^{d_A} \pmod{n_A}$$

In practice the public exponent *e* is usually chosen to be small. This is done to make the public operations (encryption and signature verification) fast. Care should be taken with the use of small public exponents. The secret exponent *d* corresponding to a small *e* is in general of the same order of magnitude as *n*. There are applications for which a small *d* (and thus large *e*) would be attractive. However, small private exponents have been shown to make RSA susceptible to attacks.

A computational error made in the RSA private operation (decryption and signature generation) may reveal the secret key. These operations should therefore always be checked for correctness. This can be done by applying the corresponding public operation to the result and checking that the outcome is as expected. So-called fault attacks are rendered mostly ineffective by this simple precaution. So-called timing attacks can be made less effective by _blinding' the private operation. Blinding factors *b* and $b_e$ are selected by randomly generating $b \in Z/nZ$ and computing $b_e = b^{-e}$. Direct computation of $x_d$ for some $x \in Z/nZ$ is replaced by the computations of the blinded value *bex*, the private operation $y = (bex)^d$, and the final outcome *by* = xd.

**Example** .

(This is a very small example — not secure!)

- Let $p = 47$, $q = 59$; then $n = pq = (47)(50) = 2773$ and $\phi(n) = (p - 1)(q - 1) = (46)(58) = 2668$,
- We need *e* to be co-prime to $\phi(n) = 2668$ and choose $e = 17$. (It is usual to choose a fairly small value of *e*, like $2^4 + 1 = 17$, etc. and to choose a prime *e*.)

- Then (using an easy algorithm the extended Euclidean algorithm in our example.) we find $d = 157$.
- We publish our public key $(n, e) = (2773, 17)$ and keep our private key, $d = 157$, a secret.

**Signature generation**
RSA is the first Public Key Cryptosystem that can be used for digital signatures.

**Signature generation**. A message m $\in$ Z/nZ can be signed as $S = m^d \in$ Z/nZ. The signature $S$ on $m$ can be verified by checking that $m = S^e \in$ Z/nZ.

In other words, suppose A wants to sign an electronic message to B. A does the following:
1. She represents the message as an integer $m$ in the interval [0, n - 1] (or breaks it into blocks if it is too long. Let's suppose $0 \le m \le n - 1$ (Here $n = n_A = p_A q_a$.)
2. Using her private key, $d_A$, she finds $\sigma = m^{d_A} \pmod{n_A}$, the signature to the message, $m$.
3. She sends B the pair $(m, \sigma) = (m, m^{d_A} \pmod{n_A})$ or, if it's confidential she enciphers it using B's public information $(n_B, e_B)$ as $(m^{e_B} \pmod{n_B}, \sigma^{e_B} \pmod{n_B}) = C$

Bob does the following:
1. He receives $(m, \sigma)$ in clear (or, if the message enciphered, he receives $C$ and deciphers it by finding $C^{dB} \bmod n_B$).
2. He checks that the signature is valid by finding $\sigma^{e_A} \bmod n_A = m'$. If $m' = m$, he believes that $m$ is the message signed by A and that $\sigma$ is her signature.

Properties we would like digital signatures to have:
1. The signature can be appended to any message the signatory wants to identify as hers.
2. The signature must be ―message dependent‖ to prevent someone from appending A's signature to a message she did not authorise — and if her message is altered, the signature should not correspond to it. This condition is met if $m$ is the false/altered message, nobody except A can find $(m')^{dA} \pmod{n_A}$, because A is the only person who knows $d_A$.
3. It should be impossible to ―forge‖ the signature. (See 2 above).
4. Signatures should be easy to check by anybody wishing to do so. This condition is met if A sends or publishes $(m, \sigma)$ — anyone can use $P_A = (n_A, e_A)$ to check whether $m = \sigma^{eA} \pmod{n_A}$.

**Example:**
The keys were generated as follows:
1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \text{ X } 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \text{ X } 10 = 160$.
4. Select $e$ such that $e$ is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$ ; we choose $e = 7$.
5. Determine $d$ such that $de = 1 \bmod 160$ 1 and $d < 160$. The correct value is $d = 23$, because 23 X 7 = 161 =1 X 160 + 1.

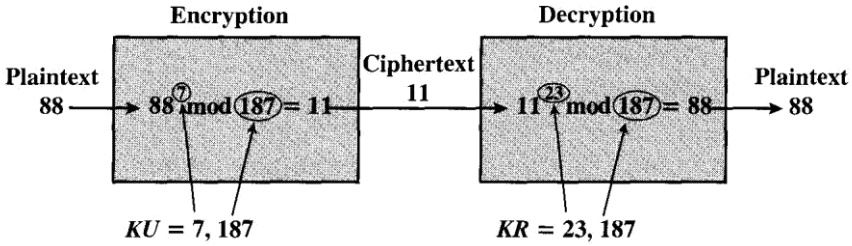


Fig. 4.2 Example of RSA Algorithm

The resulting keys are public key KU = {7, 187} and private key KR = {23, 187}. The example shows the use of these keys for a plaintext input of M = 88. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this is follows;

$88^7 \bmod 187 = [(88^4 \bmod 187) \text{ X } (88^2 \bmod 187) \text{ X } (88^1 \bmod 187)$
$88^1 \bmod 187 = 88$
$88^2 \bmod 187\ 7744 \bmod 187\ 77$
$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187\ 132$
$88^7 \bmod 187\ (88 \text{ X } 77 \text{ X } 132) \bmod 187 = 894{,}432 \bmod 187 = 11$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$11^{23} \bmod 187\ [(11^1 \bmod 187) \text{ X } (11^2 \bmod 187) \text{ x } (11^4 \bmod 187) \text{ X } (11^8 \bmod 187) \text{ X } (11^8 \bmod 187)] \bmod 187$
$11^1 \bmod 187 = 11$
$11^2 \bmod 187 = 121$
$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$
$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$
$11^{23} \bmod 187 = (11 \text{ X } 121 \text{ x } 55 \text{ x } 33 \text{ X } 33) \bmod = 187\ 79{,}720{,}245 \bmod 187 = 88$

The plaintext is 88.

**Some Mathematics: finding $c = a^b \pmod n$**

**The "square-and-multiply" method of finding $c = a^b \pmod n$**

Let $a, b, n$ be given positive integers with $a, b \in Z_n$. Let $b = \sum_{i=0}^{l-1} b_i 2^i$ with $b_i \in \{0, 1\}$ for $i = 0, 1, 2, \ldots, l-1$ and $b_{l-1} = 1$ (so the binary expression of b, $b_{l-1}\, b_{l-2} \ldots b_1\, b_0$ has $l$ binary digits).

**Algorithm** to determine c $Z_n$ such that $c = a^b \pmod n$:

1. Let $Z_i = 1$, $i = l - 1$.
2. Let $Y_i = Z_i^2 \bmod n$
3. If $b_i = 1$, let $Z_{i-1} = a\, Y_i \pmod n$
   If $b_i = 0$, let $Z_{i-1} = Y_i \pmod n$
4. If $i \geq 0$, decrease $i$ by 1 and GO TO 2.
5. Let $c = Z_{i-1}$. STOP

**Example:**

$c = 7_{13} \pmod{10}$ (b = 13 which is equivalent to binary 1 1 0 1)
$Z_3 = 1$, $I = 3$, $Y_i = 1$, $b_3 = 1$
$Z_2 = 7(1) \bmod (10)$, therefore $Z_2 = 7$
$Y_2 = 7^2 \pmod{10}$, therefore $Y_2 = 9$
$b_2 = 1$, so $Z_1 = 7(9) \pmod{10}$

Therefore $Z_1 = 3$, $Y_1 = 9$, $b_1 = 0$, so $Z_0 = 9$, $Y_0 = 9^2 \pmod{10}$

Therefore $Y_0 = 1$, $b_0 = 1$, $Z_{-1} = 7$.

Let's consider the example again:

$$7^{13} \pmod{10} \equiv 7^{2_3 + 2_2 + 1} \pmod{10} = (7^2)^{2_2} . 7^{2_2} . 7 \pmod{10}$$

$$\equiv (9)^{2_2} . 7^{2_2} . 7 \pmod{10} = (9.7)^{2_2} \pmod{10}$$

$$\equiv (3)^{2_2} . 7 \pmod{10} = (9)^{2_{Y_1}} . 7 \pmod{10}$$

$$\equiv (9^2)^{2} . 7 \pmod{10} = 1.7 \pmod{10} = 7 \pmod{10}$$

$Y_2$ ($Z_2 = 3$, $9 = Y_0$) $Y_0$ $Z_{-1}$

It is convenient to tabulate results as follows:

| $I$ | $b_i$ | $Z_i$ | $Y_i$ |
|---|---|---|---|
| 3 | 1 | 1 | 1 |
| 2 | 1 | 7 | 9 |
| 1 | 0 | 3 | 9 |
| 0 | 1 | 9 | 1 |
| -1 | - | 7 | |

Therefore : $7 = 7^{13} \pmod{10}$

### 4.4 Diffie-Hellman protocol

A key agreement protocol is carried out by two communicating parties to create a shared key. This must be done in such a way that an eavesdropper does not gain any information about the key being generated. The Diffie-Hellman protocol uses a one-way function (OWF) that can be used as a key agreement protocol for key exchange. It is not an encryption or signature scheme.

Consider a collection of N users who wish to communicate with each other. Let $p$ be a large prime (much larger than $N$ — preferably at least 768 bits) and let $g$ be a publicly known generator of an appropriately chosen group of known order $g \in Z_p = \{0, 1, 2, \ldots, p - 1\}$ such that $g, g2, g3, , g^{p-1}$ reduced modulo $p$ are all distinct non-zero elements of $Z^*_p$. So {g, g2 mod p, g3mod $p$,... , $g^{p-1}$ mod p} = {1,2,3,... , $p$ - 1} (in some order).

The public information ($p$, $g$) is made known to all users. The OWF is $f(x) = g^x \bmod p$.

To create a shared key, parties A and B proceed as follows:

1. A picks an integer $a \in \{2, 3, \ldots, \text{order}(g) - 1\}$ at random, computes $g^a$, and sends $g^a$ to B.
2. B receives $\boldsymbol{g^a}$, picks an integer $b \in \{2, 3, \ldots, \text{order(g)} - 1\}$ at random, computes $g^b$ and $g^{ab}$ and sends $g^b$ to A.
3. A receives $g^b$ and computes $g^{ab}$.
4. The shared key is $g^{ba} = g^{ba}$..

The Diffie-Hellman problem is the problem of deriving $g^{ab}$ from $g$, $g^a$, and $g^b$. It can be solved if the discrete logarithm problem in <g> can be solved: an eavesdropper can find a from the transmitted value $g^a$ and the publicly known $g$, and compute $g^{ab}$ based on a and $g^b$.

Conversely, it has not been proved in generality that the Diffie-Hellman problem is as hard as solving the discrete logarithm problem. If g generates a subgroup of the multiplicative groups of a finite field the problems are not known to be equivalent. Equivalence has been proved for the group of points of an elliptic curve over a finite field .

A related problem is the Decision Diffie-Hellman problem of efficiently deciding if $g^c = g^{ab}$, given $g$, $g^a$, $g^b$ , and $g^c$. For multiplicative groups of finite fields this problem is believed to be hard. It is known to be easy for at least some elliptic curve groups.

Figure 4.3 gives summary for the Diffie-Hellman key exchange algorithm.

**Example:**

The following example is based on the use of the prime number $p = 353$ and a primitive root of 353, in this case $g = 3$. 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 $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$

Note: In the example and in Figure 6.2 we used $X_A$ and $X_B$ to replace *a* and *b* used in the explanation.

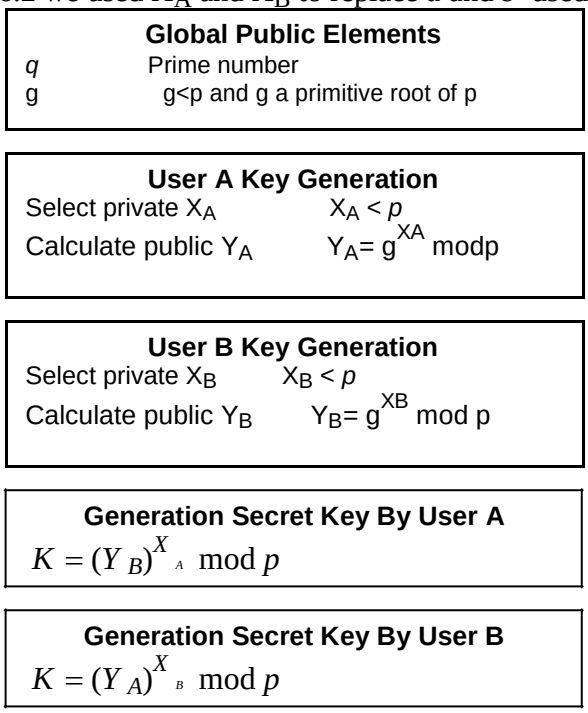



Fig. 4.3 The Diffie-Helleman Key Exchange Algorithm

**Man-in-the-middle Attack**

There is one slight problem left with Diffie-Helleman. Neither party knows with whom it shares the secret. This can be exploited by an attacker M in a man-in-the-middle attack. The attacker inserts itself on the communications path between A and B, replies to A when A initiates a protocol run and at the same time starts a protocol run with B where M pretends to be A (Figure 3.4). Principals A and B might believe they have established a shared key but in actual fact M shares key $g^{ax} \bmod p$ with A and key $g^{bz} \bmod p$ with B, and can read all traffic between A and B while acting as a relay.

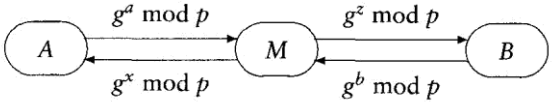



Figure 3.4: Man-in-the-middle Attack against the Diffie—Hellman protocol

**Station-to-station Protocol**

To secure the Diffle—Heilman protocol, authentication has to be added. The station-tostation (STS) protocol (Diffie, van Oorschot and Wiener, 1992) does this as follows.

In addition to a Diffie—Hellman exchange that establishes a shared session key K := $g^{ab}$ mod *p*, the protocol uses an encryption algorithm and a signature algorithm. No specific algorithms are prescribed. In the following, $S_a$ and $S_b$ are the signature keys of A and B, $sS_a$ and $sS_b$ denote signatures generated under these keys. The steps in the protocol are:

1. $A \rightarrow B: \quad g^a$
2. $B \rightarrow A: \quad g^b, eK(sS_b(g^b, g^a))$
3. $A \rightarrow B: \quad eK(sS_a(g^a, g^b))$

In step 1, A initiates a Diffie—Hellman key exchange as before. After, receiving $g^a$, B picks a random number *b* and computes the session key *K* := $g^{ab}$ mod *p*. After step 2, A can complete its part of the Diffie—Hellman key exchange and derive *K*, and then decrypt the second part of B‘s message and verify B‘s signature. After the last message, B can decrypt and verify A‘s signature. STS is a key agreement protocol that provides mutual entity authentication and explicit key authentication.

**Exercises**

1. Calculate public key of A, $PU_B$ and shared key if the primitive root is 353, *g* = 3 and A and B select secret keys XA = 97 and XB = 233, respectively.
2. Using Extended Euclidean Algorithm, calculate GCD(7890,9988)
3. Calculate $15^{-1}$ mod 47
4. Using RSA where p=17, q=11, e=7 decrypt c=11
5. Using square and multiply method, compute $25^{43}$ mod 50
6. Consider a DH protocol with a common prime q = 11 and a primitive root α =2.

   a. If user A has public key $Y_A$ = 9, what is A‘s private key $X_A$?

   b. If user B has public key $Y_B$ = 3, what is the shared secret key?

   Solution

   a. $P_{UA} = g^a$ mod p

   $2^a$ mod 11=9

   a=6 because $2^6$ mod 11=9

   Hence A‘s private key is 6

   b. Shared key=$(g^b)^a$ mod 11=$(3)^6$ mod 11=3 6=110

| I | Bi | Zi | Yi | Zi-1 |
|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 3 |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 3 | 9 | 5 |
| 0 | 0 | 5 | 3 | 3 |

## CHAPTER 5. Digital Signature and Authentication

The most important development from the work on public-key cryptography is the digital signature. The digital signature provides a set of security capabilities that would be difficult to implement in any other way.

### 5.1 Digital Signature

In business transactions, we sign various documents such as checks and contracts. The reason we sign documents is to indicate our understanding and acceptance of the contents of the document. The objective of a digital signature is to sign an electronic document rather than a paper document. However, digital signature technology also encompasses several other purposes including encryption to prevent tampering and certification to authenticate the identity of the signer. In general Digital signature technology has following main goals.

1. When a message is received, the recipient may desire to verify that the message has not been altered in transit.
2. Furthermore, the recipient may wish to be certain of the originator's identity.
3. Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at any later time.

A digital signature is an electronic analogue of a written signature in that the digital signature can be used in proving to the recipient or a third party that the message was, in fact, signed by the originator.

Table 5.1: Main goals of Digital Signature and the methods used for implementation

| Goal | Method |
|---|---|
| 1. To ensure that no one else had access to the Data | encryption |
| 2. To ensure that the data has not been altered | digital signature/ message digest |
| 3. To ensure that a particular sender sent the data | certification authority |

Public cryptography is one of the methods used for digital signatures. An example of this is shown in Figure 5.1. In Figure 5.1, User A sends a document to User B. User B encrypts the document with its private key and sends it back to User A. User A stores the encrypted document in case User B denies the signature. User A decrypts the document for his or her use.

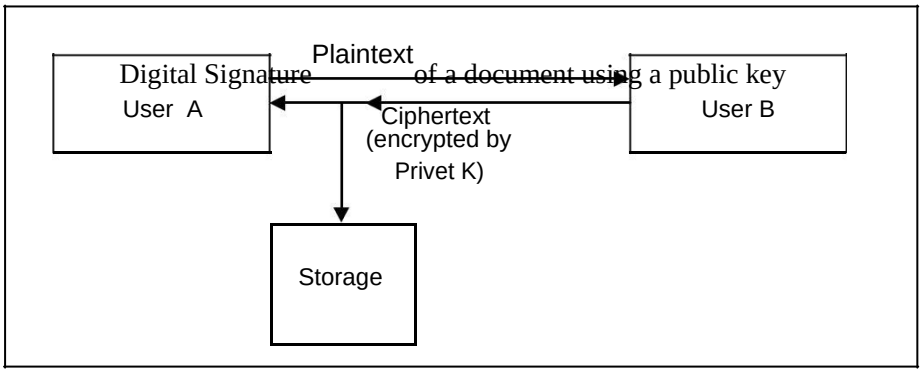


Fig.5.1 Digital Signature of a document using a public key

The disadvantage of this method is that the entire document must be encrypted and stored at the receiver side, thus requiring a large amount of memory. Another method is to sign a message digest (that is, a summary of the document contents) rather than the entire document. A message digest is a summary of the message, such as a frame check sequence.

Consider Figure 5.2. User A sends a document to User B for a signature. User B generates a message digest from the document and then encrypts the message digest and sends it back with the document to User A. User A stores the message digest as the signature of the document. The function that is used to generate a message digest is called a **hash function**.

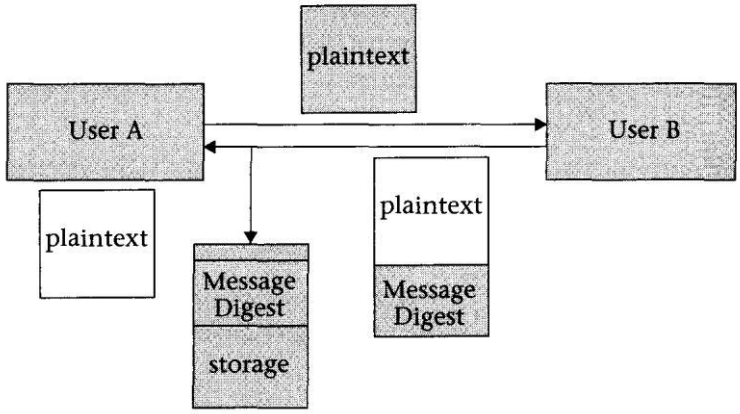



Fig.5.2 Digital signature of a message digest

The most common digital signature technology currently in use is the Digital Signature Standard (DSS), which was accepted by the US Government as the official technology for federal electronic communications. The DSS specifies the use of the Digital Signature Algorithm (DSA) and the Secure Hash Algorithm (SHA-1) to produce digital signatures.

**How Digital Signatures are Created**

In general, creating digital signature needs three steps: Hash, encryption and certification.

**First Step: The Hash.**

A digital signature is created by fingerprinting electronic data using a mathematical formula, known as a hash algorithm. The Secure Hash Algorithm, used in the DSS, is one of several different algorithms available. The algorithm produces a message digest or "hash" which is derived from, and unique to, the data. Consequently, the digest is unique to a particular document and cannot be duplicated. When the digest is attached to the document, the document has been "signed". Recipients can later use the digest to verify that the data have not been altered during transmission. Some more information hash is given latter.

**Second Step: Encryption.**

A second element in digital signatures is encryption. Any of the encryption techniques may be used. Most digital signature software programs use encryption to encode the message digest, and sometimes also the message itself. Normally public key technology is used for encryption.

**Third Step: Certification.**

The final element in digital signatures is certification. Certification involves the existence of a trusted third party, generally called a certificate authority, who confirms the identity of the signer. Certificates are similar to many other forms of personal identification. For example, a driver's license issued by the DMV is often required to prove one's identity when writing a check. Similarly, a digital certificate can be used to prove one's identity when sending a digitally signed document. Latter some more information on certification are given.

**Overview of the Process.**

To summarize, the steps involved in creating a "signed" electronic document are (see Figure 4.3):

1. The data is fingerprinted, creating a message digest/hash
2. The message digest is encrypted and is attached to the data
3. The data is encrypted (if desired)
4. The identity of the signer is certified by a certification authority.

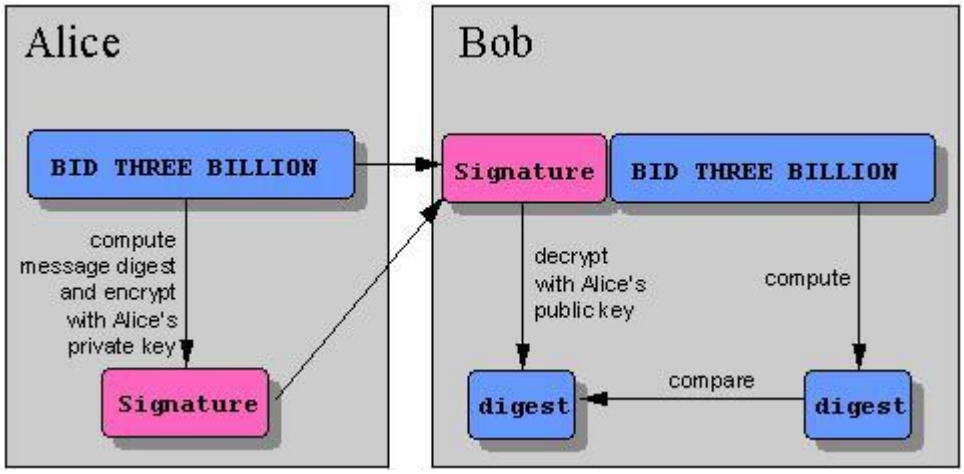



Fig.5.3 Generating Digital Signature

**Who Needs Digital Signatures?**

Digital signatures are needed by people and organizations in many different sectors. For example, individuals might use digital signatures for:

- their medical records (remote treatment, email, etc.)
- financial transactions (bank transfers, IRS filings, etc.)
- education (online courses, registration,

etc.) Businesses might use digital signatures for:

- financial transactions (securities trading, etc.)
- conduct business discussions (company mergers etc.)

- communicating trade secrets
- filing legal documents

The government might use digital signatures for:
- military information
- voting (registration, online voting, etc.)
- air traffic control information

## 5.2 Message Authentication and Hash Function

In the previous chapters, we concentrated on protecting the messages by encryption. Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

Message authentication is a procedure that allows communicating parties to verify that received messages are authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. In general, message authentication is concerned with:

- protecting the integrity of a message
- validating identity of originator
- non-repudiation of origin (dispute resolution)
- It may also address sequencing and timeliness.

- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

**Authentication Using Symmetric Encryption**

It is possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able successfully to encrypt a message for the other participant. Furthermore, if the message includes an error- detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

**Message Authentication without Message Encryption**

In this section, we examine several approaches to message authentication that do not rely on message encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided. Because symmetric encryption will provide authentication, and because it is widely used with readily available products, why not simply use such an approach, which provides both confidentiality and authentication? Some authors suggest three situations in which message authentication without confidentiality is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations. For example, notification to users that the network is now unavailable or an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication, If a violation occurs, the other destination systems are alerted by a general alarm.
2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages.
3. Authentication is carried out on a selective basis, with messages chosen at random for checking. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.

Thus, there is a place for both authentication and encryption in meeting security requirements.

### 5.2.1 Message Authentication Code (MAC)

One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key $K_{AB}$. When A has a message M to send to B, it calculates the message authentication code as a function of the message and the key:

$MAC_M = F(K_{AB}, M)$. The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated code (Figure 5.4). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.
3. If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

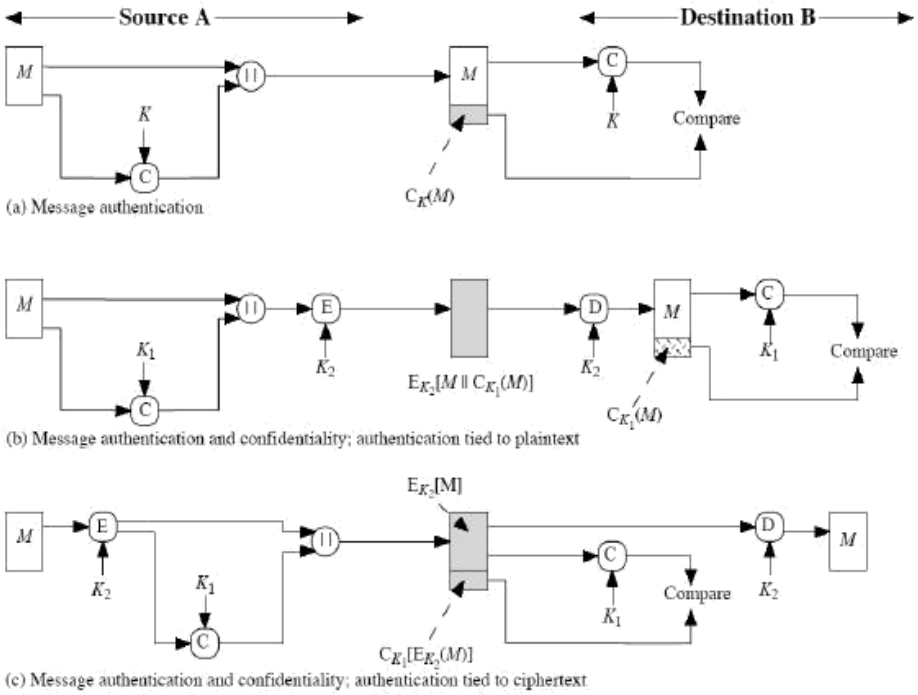



Figure 5.4 Message Authentication Using a Message Authentication Code (MAC)

A number of algorithms could be used to generate the code. The National Bureau of Standards, in its publication DES Modes of Operation, recommends the use of DES. DES is used to generate an encrypted version of the message, and the last number of bits of ciphertext are used as the code. A 16- or 32-bit code is typical.

The process just described is similar to encryption. One difference is that the authentication algorithm need not be reversible, as it must for decryption. It turns out that because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

## 5.3 Authentication Using Hash Function

Typically, a hash function generates a hash value (message digest) from a given message. One of the characteristics of a hash value is that it is impossible to use the hash value to generate the original message.

### One-Way Hash Function

A variation on the message authentication code that has received much attention recently is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size message digest H(M) as output. Unlike the MAC, a hash function does not also take a secret key as input. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

The message digest can be encrypted using symmetric encryption; if it is assumed that only the sender and receiver share the encryption key, then authenticity is assured. The message digest can also be encrypted using public-key encryption. The public-key approach has two advantages: It provides a digital signature as well as message authentication, and it does not require the distribution of keys to communicating parties.

These two approaches have an advantage over approaches that encrypt the entire message in that less computation is required. Nevertheless, there has been interest in developing a technique that avoids encryption altogether. Several reasons for this interest are pointed out in many references:

- Encryption software is quite slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
- Encryption hardware costs are nonnegligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- Encryption algorithms may be covered by patents. Some encryption algorithms, such as the RSA public-key algorithm, are patented and must be licensed, adding a cost.
- Encryption algorithms may be subject to export control.

This technique assumes that two communicating parties, say A and B, share a common secret value $S_{AB}$. When A has a message to send to B, it calculates the hash function over the concatenation of the secret value and the message: $MD_M = H(S_{AB} \text{ II } M)$. It then sends $\{M \text{ II } MD_M\}$ to B. Because B possesses $S_{AB}$, it can recompute $H(S_{AB} \text{ II } M)$ and verify $MD_M$. Because the secret value itself is not sent, it is not possible for an attacker to modify an intercepted message. As long as the secret value remains secret, it is also not possible for an attacker to generate a false message.

This third technique, using a shared secret value, is the one adopted for IP security; it has also been specified for SNMPv3.

**Secure Hash Functions**

The one-way hash function, or secure hash function, is important not only in message authentication hut in digital signatures. In this section, we begin with a discussion of requirements for a secure hash function. Then we look at one of the most important hash functions, SHA-1.

**Hash Function Requirements**

The purpose of a hash function is to produce a ―fingerprint‖ of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given code h, it is computationally infeasible to find x such that H(x) = h.
5. For any given block x, it is computationally infeasible to find y x with H(y) = H(x).
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property is the one-way property: It is easy to generate a code given a message, but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value. The secret value itself is not sent; however, if the hash function is not one way, an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, the attacker obtains the message M and the hash code $MD_M = H(S_{AB} \text{ II } M)$. The attacker then inverts the hash

function to obtain $S_{AB}$ II M = $H^{-1}$ ($MD_M$). Because the attacker now has both M and $S_{AB}$ II M, it is a trivial matter to recover $S_{AB}$. (Note: The symbol II means concatenation)

The fifth property guarantees that it is impossible to find an alternative message with the same hash value as a given message. This prevents forgery when an encrypted hash code is used. If this property were not true, an attacker would be capable of the following sequence: First, observe or intercept a message plus its encrypted hash code; second, generate an unencrypted hash code from the message; third, generate an alternate message with the same hash code.

A hash function that satisfies the first five properties in the preceding list is referred to as a **weak hash function**. If the sixth property is also satisfied, then it is referred to as a **strong hash function**. The sixth property protects against a sophisticated class of attack known as the birthday attack.

In addition to providing authentication, a message digest also provides data integrity. It performs the same function as a frame check sequence: If any bits in the message are accidentally altered in transit, the message digest will be in error.

Some of the methods that generate hash values from a plaintext are Frame Check Sequence (FCS) and Checksum. FCS and Checksum are forms of the weak hash functions. Some of the stronger, more secure hash functions are:

- MD4 is a message digest type 4 that produces 128-bit hash values or message digests.
- MD5 is an improved version of MD4 that produces 128-bit hash values.
- SHA (Secure Hash Algorithm) generates 160-bit hash values.

**Frame Check Sequence (FCS**) as a Hash Value. User B generates the FCS of the information, encrypts the FCS, and sends it to User A. If one alters the document, the new FCS would differ from the one generated by User B.

**Checksum (Generating Checksum from a Message).** Information can be broken into blocks of characters and arranged in columns. The message $6578100 can be broken into four ASCII characters per block as follows, and the checksum of each column can be calculated:

| | | | | |
|---|---|---|---|---|
| $657 | 44 | 36 | 35 | 37 |
| 8100 | 38 | 31 | 30 | 30 |
| Checksum | 7C | 67 | 65 | 67 |

If the message changes to $5578200, it will generate the same checksum; therefore, the checksum method is not a strong message digest.

The simplest form of a hash function is to break the message into m blocks of n bits as follows:

| | | | | | |
|---|---|---|---|---|---|
| Block0 | $b_{01}$ | $b_{02}$ | $b_{03}$ | … | $b_{0n}$ |
| Block1 | $b_{11}$ | $b_{12}$ | $b_{13}$ | … | $b_{1n}$ |
| … | … | … | … | … | … |
| Blockm | $b_{m1}$ | $b_{m2}$ | $b_{m3}$ | … | $b_{mn}$ |

Hash bits are represented by:

H = H1 H2 H3 ... Hn

where:

H1 = $b_{01}$ XORed $b_{11}$ XORed $b_{21}$ XORed ... XORed $b_{ml}$

Or in general:

Hi = $b_{i1}$ XORed $b_{i2}$ XORed $b_{i3}$ XORed $b_{i4}$ XORed ... XORed $b_{in}$

**Example 1.** Find the hash code for the word ―WELCOME.‖

| | |
|---|---|
| W | 1010111 |
| E | 1000101 |
| L | 1001100 |
| C | 1000011 |
| O | 1001111 |
| M | 1001101 |
| E | 1000101 |
| Hash Code | 1011010 |

### 5.3.1 The SHA-1 Secure Hash Function

The Secure Hash Algorithm (SHA) was developed by NIST and published as a federal information processing standard (FIPS 180) in 1993; a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1.

The algorithm takes as input a message with a maximum length of less than $2^{64}$ bits and produces as output a 160-bit message digest. The input is processed in 512-bit blocks. The processing consists of the following steps:

**Step 1**: **Append padding bits**.

The message is padded so that its length is congruent to 448 modulo 512 (length = 448 mod 512). That is, the length of the padded message is 64 bits less than a multiple of 512 bits. The 64 bits are those given to k, the length of the message. Accordingly, to make the total length multiple of 512, padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 512. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

**Step 2: Append length**.

A block of 64 bits is appended to the message. This block is treated as an unsigned 64-bit integer (most significant byte first) and contains the length of the original message (before the padding). The inclusion of a length value makes more difficult a kind of attack known as a padding attack.

The outcome of the first two steps yields a message that is an integer multiple of 512 bits in length. In Figure 21.8, the expanded message is represented as the sequence of 512-bit blocks Y0, Y1,. . . , $Y_{L-1}$, so that the total length of the expanded message is L x 512 bits. Equivalently, the result is a multiple of 16 32-bit words.

**Step 3: Initialize MD buffer**.

A 160-bit buffer is used to hold intermediate and final results of the hash function.

**Step 4: Process message in 512-bit (16-word) blocks**.

The heart of the algorithm is a module that consists of four rounds of processing of 20 steps each. The four rounds have a similar structure, but each uses a different primitive logical function. Each round takes as input the current 512-bit block being processed (Yq) and the 160-bit buffer value and updates the contents of the buffer.

**Step 5: Output**.

After all L 512-bit blocks have been processed, the output from the $L^{th}$ stage is the 160-bit message digest.

The SHA-1 algorithm has the property that every bit of the hash code is a function of every bit in the input. The algorithm produces results that are well mixed; that is, it is unlikely that two messages chosen at random, even if they exhibit similar regularities, will have the same hash code. Unless there is some hidden weakness in SHA-1, which has not so far been published, the difficulty of coming up with two messages having the same message digest is on the order of $2^{80}$ operations, while the difficulty of finding a message with a given digest is on the order of $2^{160}$ operations.

## 5.4 Key Management

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:

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

We examine each of these areas in turn.

### 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

#### *a.Public Announcement*

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. For example, because of the growing popularity of PGP (pretty good privacy), which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums such as USENET newsgroups and Internet mailing lists.

Major weakness of the technique: 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.

### *b. Publicly Available Directory*

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 (Figure 5.9). 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.

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.

### c. Public-Key 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. 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 occur:

1. A sends a timestamped 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.

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:

5. 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 (3), the presence of $N_1$ in message (6) assures A that the correspondent is B.
6. A returns N2, encrypted using B's public key, to assure B that its correspondent is A.

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.

### *d. Public—Key Certificates*

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.

An alternative approach is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. 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.

Each participant applies to the certificate authority, supplying a public key and requesting a certificate.

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:

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

The recipient uses the authority's public key, PUauth, 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.

In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired.

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME.

**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.

***Simple Secret Key distribution:*** 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, IDA.
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. At the completion of the exchange, both A and B discard K. 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 is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message. Such an attack is known as a **man-in-the-middle** attack [MITM]. In this case, if an adversary, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion without being detected:

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message intended for B consisting of $PU_a$ and an identifier of A, $ID_A$.
2. E intercepts the message, creates its own public/private key pair $\{PU_e, PR_e\}$ and transmits $PU_e \| ID_A$ to B.
3. B generates a secret key, $K_s$, and transmits $E(PU_e, K_s)$.
4. E intercepts the message, and learns $K_s$ by computing $D(PR_e, E(PU_e, K_s))$.
5. E transmits $E(PU_a, K_s)$ to A.

The result is that both A and B know $K_s$ and are unaware that $K_s$, has also been revealed to E. A and B can now exchange messages using $K_s$. E no longer actively interferes

with the communications channel but simply eavesdrops. Knowing $K_s$, E can decrypt all messages, and both A and B are unaware of the problem. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication Figure 8.13 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 N1 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 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.

Notice that the first three steps of this scheme are the same as the last three steps of Figure 5.14. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

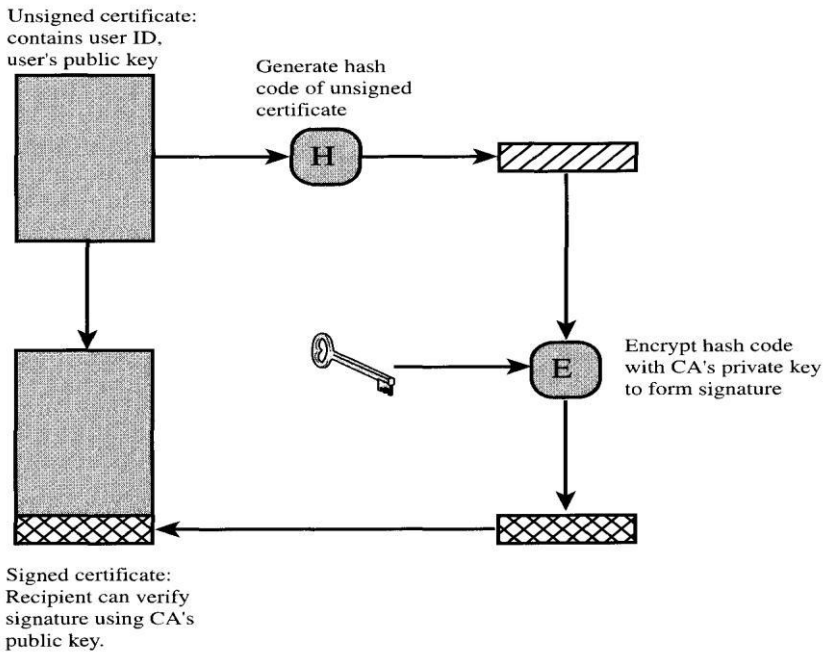



Figure 5.14 Public-Key Certificate Use

**References**


- V. K. Pachghare, Cryptography And Information Security,2008.
- William Stallings, "Cryptography and Network Security: Principles and Practice" Prentice Hall, New Jersey. Johannes
- William Stallings, Cryptography and Network Security Principles and Practices, Fourth Edition,2005

- Bellare, Mihir; Rogaway, Phillip (21 September 2005). "Introduction". Introduction to Modern Cryptography. p. 10.
- AJ Menezes, PC van Oorschot and SA Vanstone,‖ Handbook of Applied

- Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644–654. (pdf)
- Oded Goldreich,‖ Foundations of Cryptography‖, Volume 1: Basic Tools, Cambridge University Press, 2001, ISBN 0-521-79172-3.
- R. Rivest, A. Shamir, L. Adleman. ―A Method for Obtaining Digital Signatures and Public- Key Cryptosystems‖ Communications of the ACM, Vol. 21 (2), pp.120–126. 1978.