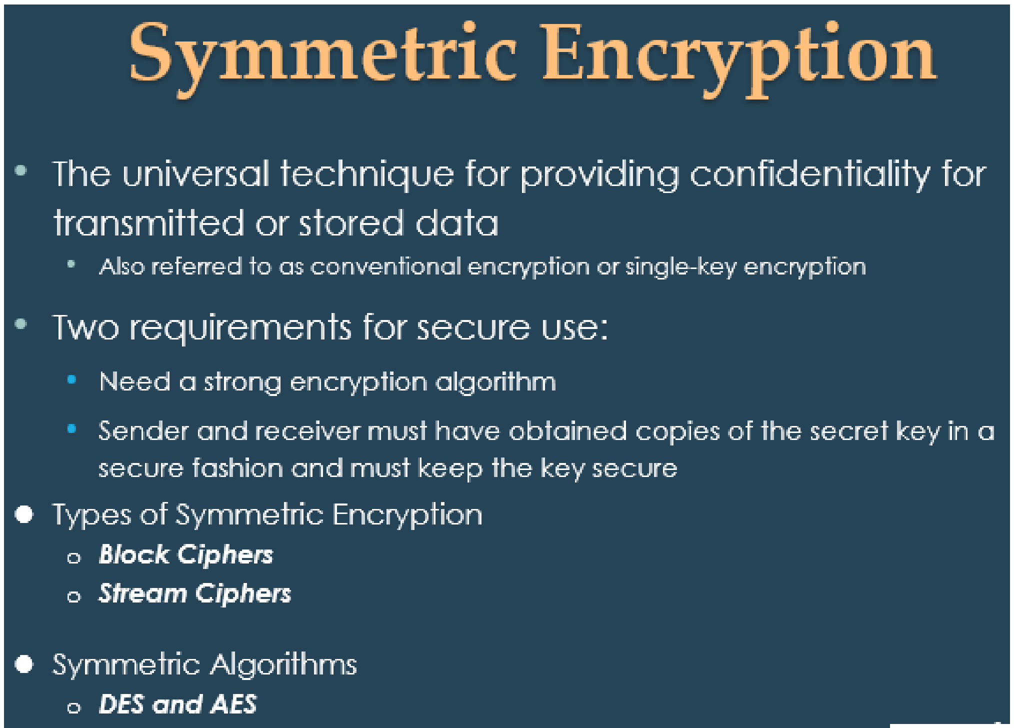
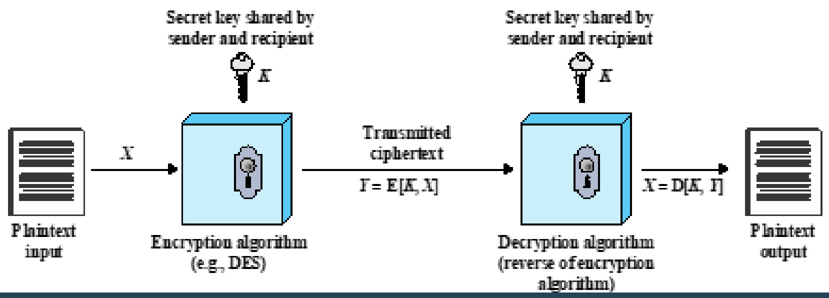
Chapter 2 (2.1, 2.2, 2.3, 2.4) AES, DES

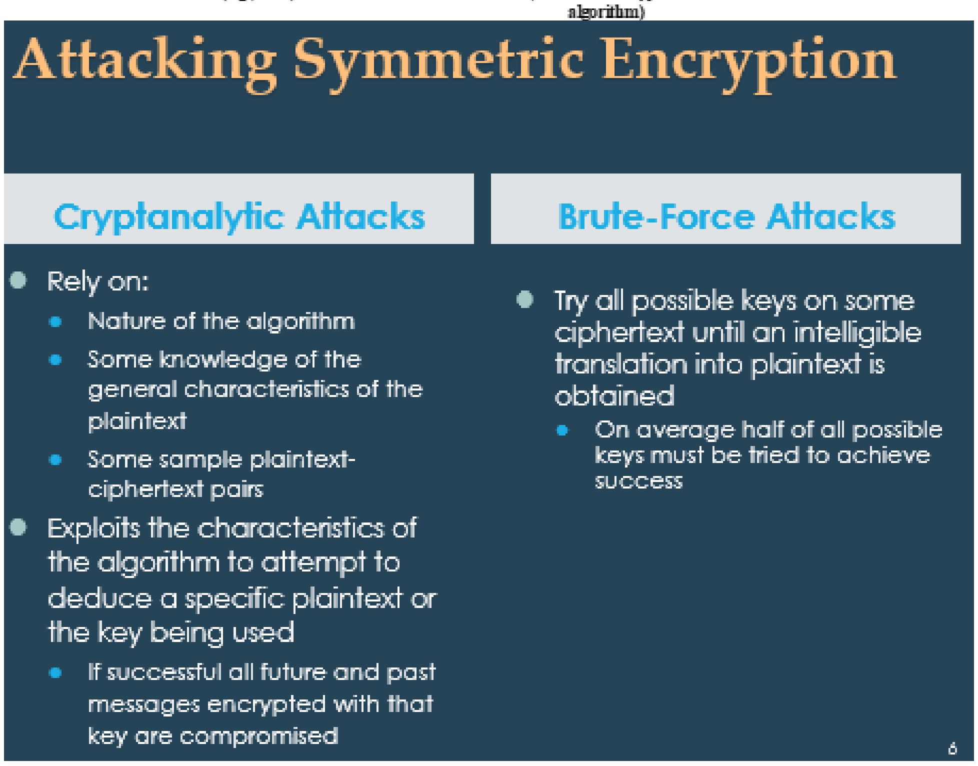


Symmetric Encryption

- The universal technique for providing confidentiality for transmitted or stored data
  - Also referred to as conventional encryption or single-key encryption
- Two requirements for secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure
- Types of Symmetric Encryption
  - ***Block Ciphers***
  - ***Stream Ciphers***
- Symmetric Algorithms
  - ***DES and AES***

Symmetric Encryption


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
$Y = E[K, X]$
$X = D[K, Y]$
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output



Attacking Symmetric Encryption

Cryptanalytic Attacks

- Rely on:
  - Nature of the algorithm
  - Some knowledge of the general characteristics of the plaintext
  - Some sample plaintext-ciphertext pairs
- Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or the key being used
  - If successful all future and past messages encrypted with that key are compromised

Brute-Force Attacks

- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
  - On average half of all possible keys must be tried to achieve success

Data Encryption Standard (DES): (more details in next pages)

DES takes a **plaintext block of 64 bits** and a **key of 56 bits**, to produce a **cipher text block of 64** bits.

DES Concerns:

1. The **algorithm** itself. Cryptanalysis is possible by exploiting the characteristics of the DES algorithm.
2. A more serious concern is **key length**. With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16} keys. Given the speed of commercial off-the-shelf processors, this key length is woefully inadequate

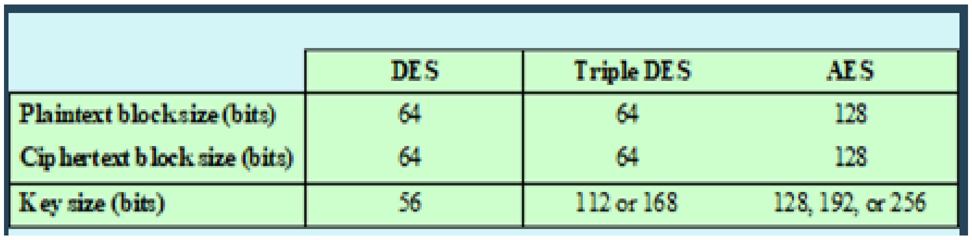
Triple DES (3DES):

Repeats basic DES algorithm three times using either two or three unique keys.

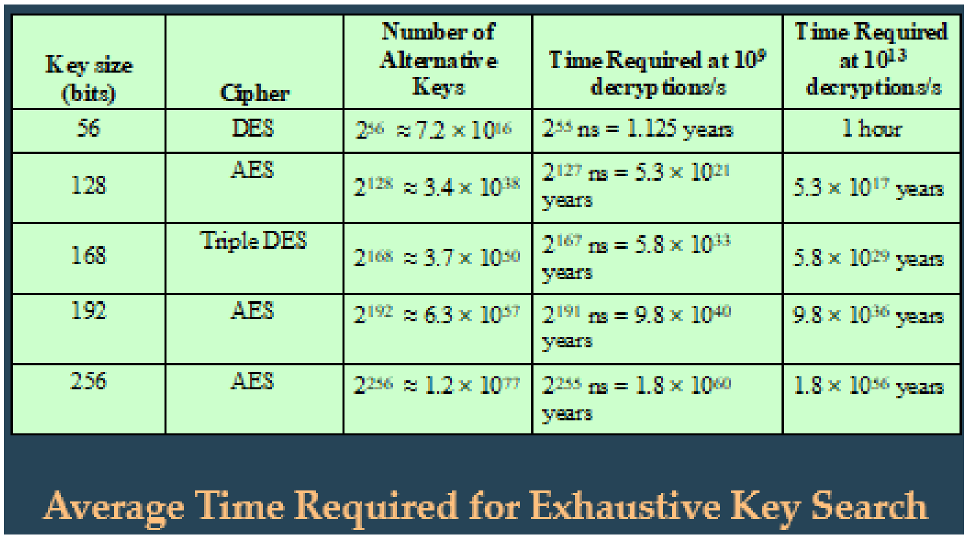
- **168-bit key length** overcomes the vulnerability to brute-force attack of DES
- Underlying encryption **algorithm is the same** as in DES3

3DES Drawbacks:

1. Relatively **sluggish** in software. 3DES requires three times as many calculations as DES, is correspondingly **slower.**
2. Both DES and 3DES **use a 64-bit block** size. For reasons of both efficiency and security, a **larger block size is desirable**.



	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256



Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Average Time Required for Exhaustive Key Search

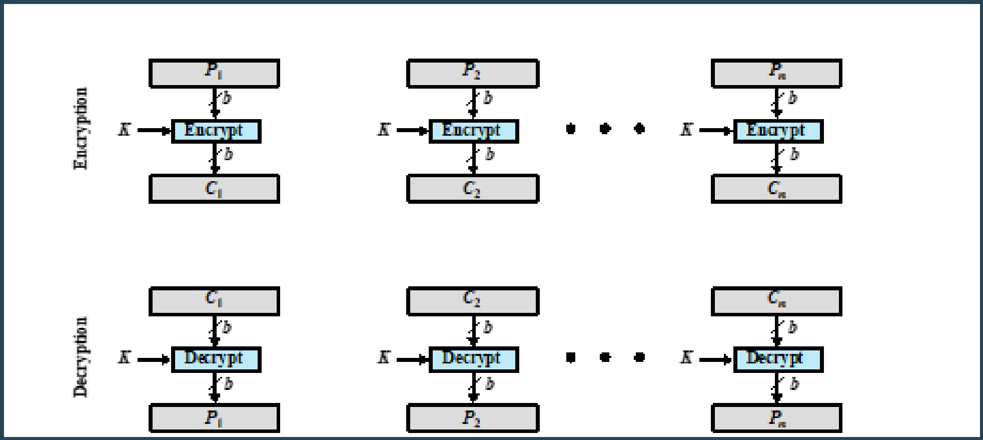
Practical Security Issues

- Typically symmetric encryption is applied to a unit of data larger than single 64-bit or 128-bit block
- Different modes of operations on block cipher
  - means how data can be encrypted and decrypted
  - **ECB,** CBC, CTR
- The simplest approach to multiple-block encryption is known as ***electronic codebook (ECB) mode***
  - plaintext is handled **b bits** at a time and each block of plaintext is encrypted using the same key.
    - b=64 or b=128
  - Each block size encrypt and decrypt process is independently.

- Cryptanalysts may be able to exploit regularities in the plaintext
- Modes of operation
  - Alternative techniques developed to increase the security of symmetric block encryption for large sequences
  - Overcomes the weaknesses of ECB

Electronic codebook (ECB)

- Encryption and decryption in blocks (e.g., 64 or 128 bit)



Computationally Secure Encryption Schemes

- Encryption is computationally secure if:
  - Cost of breaking cipher exceeds value of information
  - Time required to break cipher exceeds the useful lifetime of the information
- Usually very difficult to estimate the amount of effort required to break
- Can estimate time/cost of a brute-force attack

Feistel Cipher Structure

- The Feistel structure is a particular example of the more general structure used by all symmetric block ciphers
- Symmetric block cipher consists of:
  - ***A sequence of rounds***
  - ***With substitutions and permutations controlled by key***
- A symmetric block cipher depends on the choice of the following parameters and design features:
  - Block size
  - Key size
  - Number of rounds
  - Subkey generation algorithm
  - Round function
  - Fast software encryption/decryption
  - Ease of analysis

Feistel Cipher Structure

Steps:

- ❑ The inputs to the encryption algorithm are:
  - Plaintext block of length **2w** bits
  - A key **K**
- ❑ The plaintext block is divided into two halves, **L0** and **R0**
- ❑ 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 **Li-1** and **Ri-1**, derived from the previous round, as well as a subkey **Ki** , derived from the overall **K**
- ❑ In general, the subkeys **Ki** are different from **K** and from each other, and are generated from the key by a subkey generation algorithm
- ❑ All rounds have the same structure
- ❑ A substitution is performed on the left half of the data
- ❑ This is done by applying a round function F to the right half of the data and then taking the exclusive-OR (XOR) of the output of that function and the left half of the data.
- ❑ The round function has the same general structure for each round but is parameterized by the round subkey Ki .
- ❑ Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

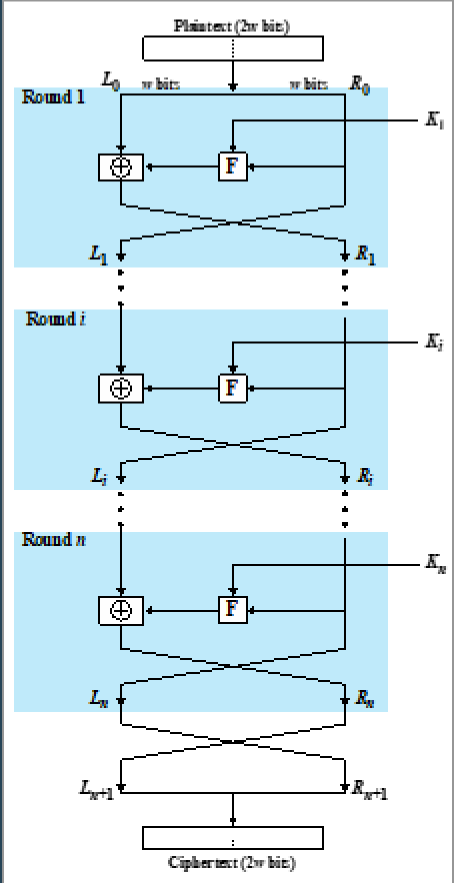
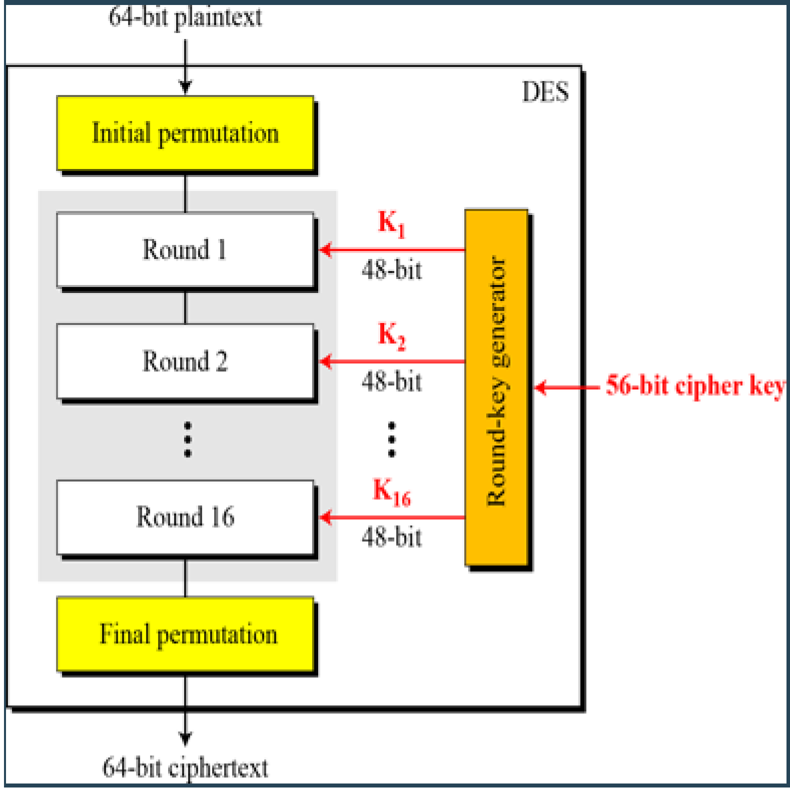


Figure 20.1 Classical Feistel Network

Data Encryption Standard (DES)

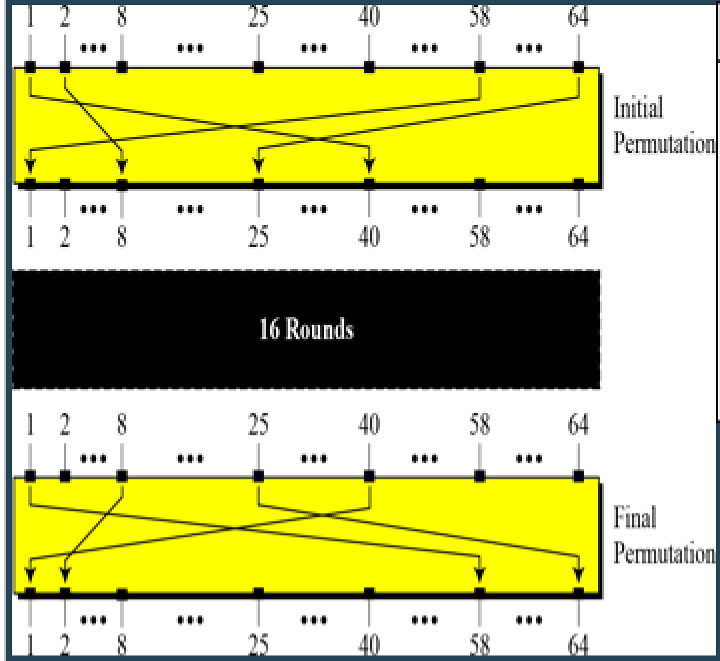
- The DES algorithm can be described as follows.
  - The plaintext is 64 bits in length
  - The key is 56 bits in length
  - There are 16 rounds of processing.
  - From the original 56-bit key, 16subkeys are generated (ki= 48 bit)
    - one of which is used for each round.
- **Encryption process:**
  1. Permutation performed on the input block
  2. Generation of round keys
  3. Performing 16 identical rounds
     a. Substitution
     b. Permutation
  4. Inverse permutation to step 1
- **Decryption process:**
  - Use the ciphertext as input to the DES algorithm, but use the sub-keys Ki in reverse order.
  - That is, use K16 on the first iteration, K15 on the second iteration, and so on until K1 is used on the sixteenth and last iteration.

Figure 6.2 *General structure of DES*



Note: The initial and final permutations are *straight P-boxes* that are inverses of each other. They have no cryptography significance in DES. Both permutations are keyless and predetermined.

Figure 6.3 *Initial and final permutation steps in DES*



Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Rule:
In the initial permutation, the 58th bit in the input becomes the first bit in the output. Similarly, in the final permutation, the first bit in the input becomes the 58th bit in the output.

cipher.

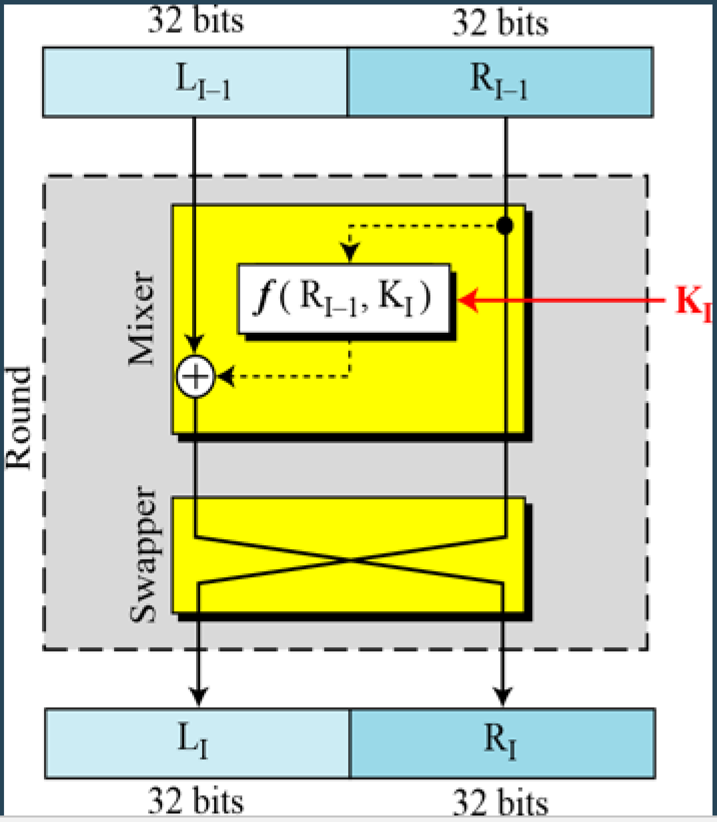
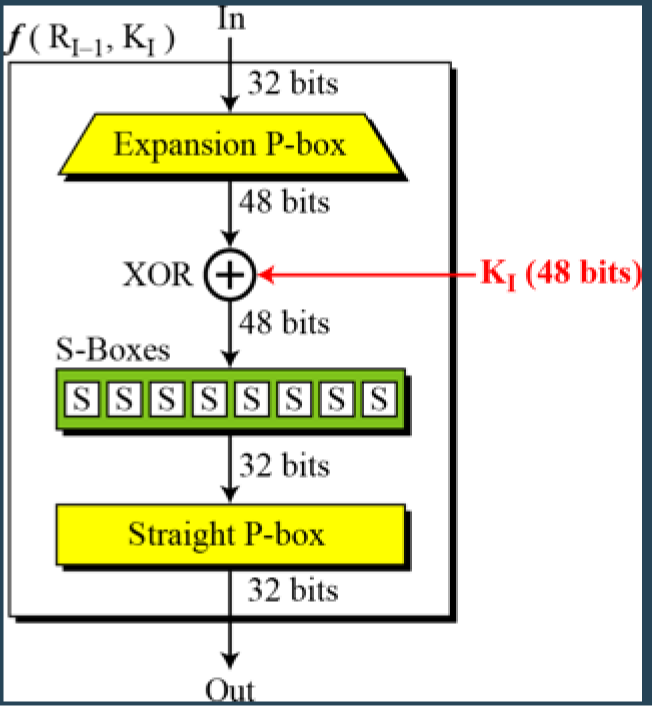


Figure 6.4 ***A round in DES (encryption site)***

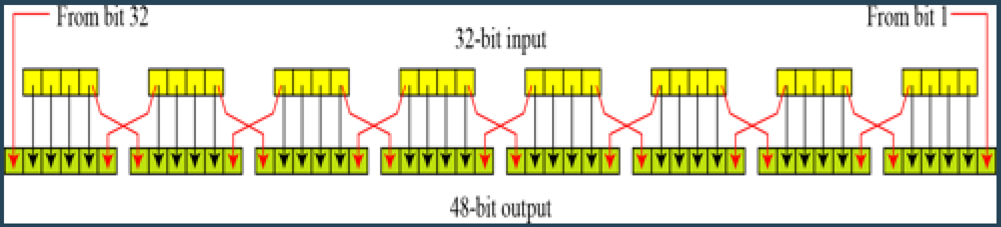
Figure 6.5: ***DES function***



Expansion P-box

Since R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits.

Figure 6.6 ***Expansion permutation***



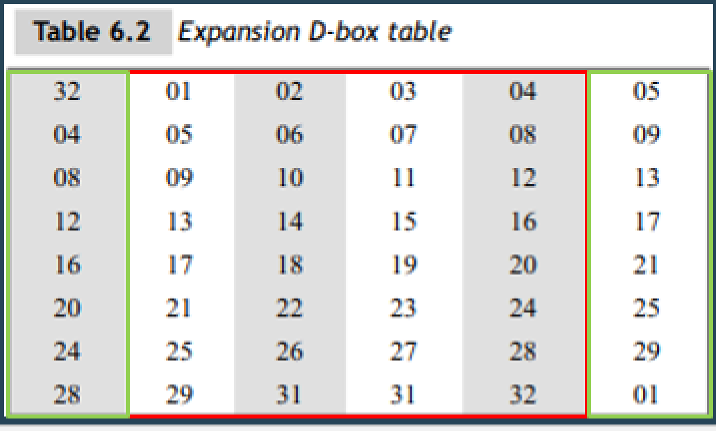
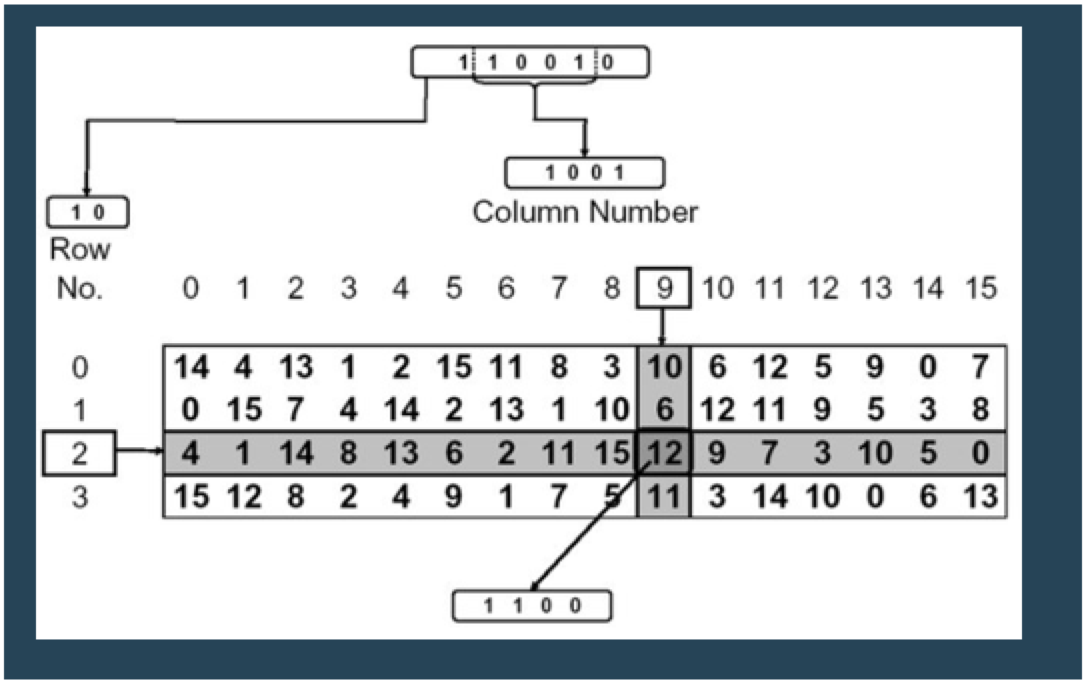


Table 6.2 *Expansion D-box table*

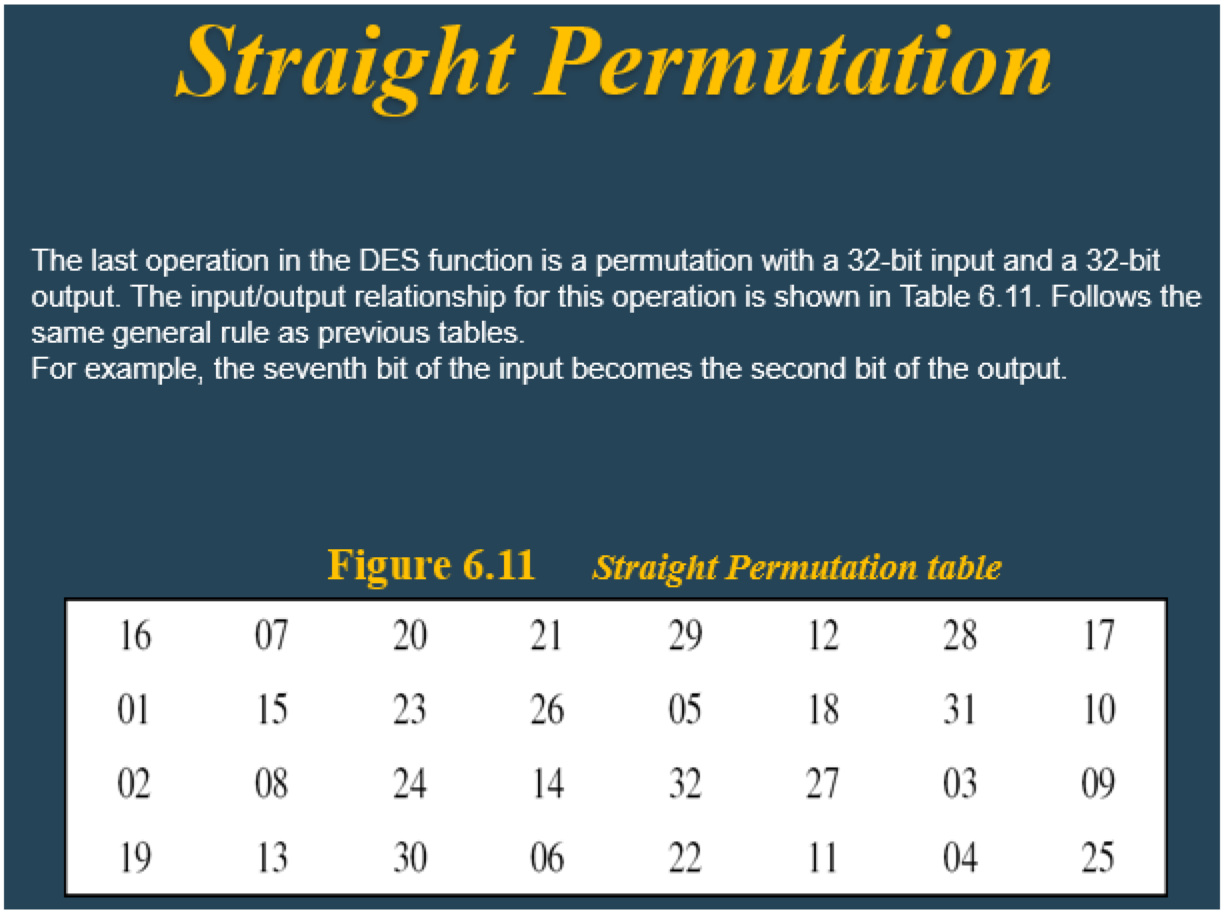
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

Note: The number of output ports is 48, but the value range is only 1 to 32. Some of the inputs go to more than one output.

S-Box



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



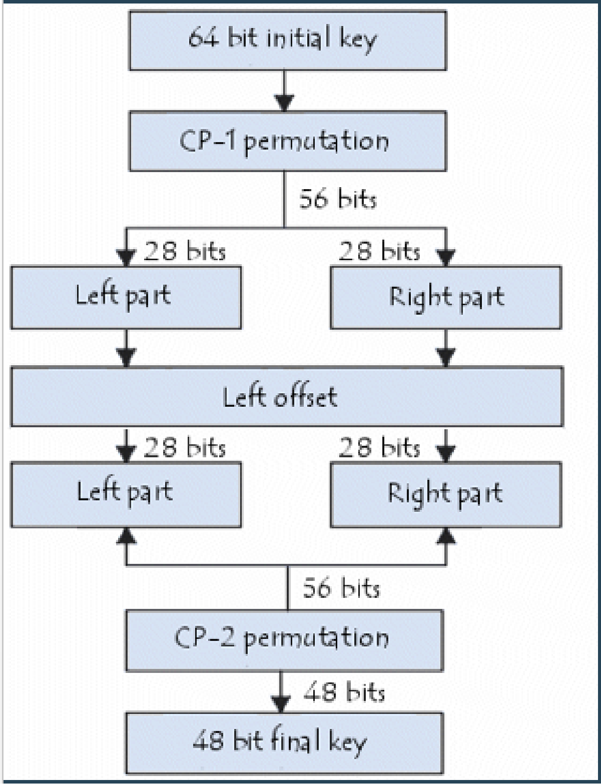
Straight Permutation

The last operation in the DES function is a permutation with a 32-bit input and a 32-bit output. The input/output relationship for this operation is shown in Table 6.11. Follows the same general rule as previous tables.
For example, the seventh bit of the input becomes the second bit of the output.

Figure 6.11 *Straight Permutation table*

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

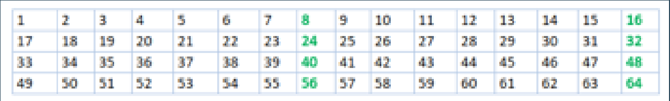


Steps:-

1. It drops the parity bits (bits 8, 16, 24, 32, 40,48,56, & 64) from the 64-bit key and permutes the rest of the bits according to Table 6.12
2. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called key transformation
3. For this, the 56-bit key is divided into two halves, each of 28 bits.
4. These halves are circularly shifted left by one or two positions, depending on the round.
5. For example, if the round numbers 1, 2, 9, or 16 the shift is done by **only one position** for other rounds, **the circular shift is done by two positions.**

Table 6.12 *Parity-bit drop table*

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04



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

Figure - discording of every 8th bit of original key

Table 6.13 *Number of bits shifts*

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 6.14 *Key-compression table*

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Triple DES (3DES)

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

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

- where:
  - C = ciphertext;
  - P = plaintext; E[K, X] = encryption of X using key K

- Decryption is simply the same operation with the keys reversed

$$P = \mathrm{D}(K_1, \mathrm{E}(K_2, \mathrm{D}(K_3, C)))$$

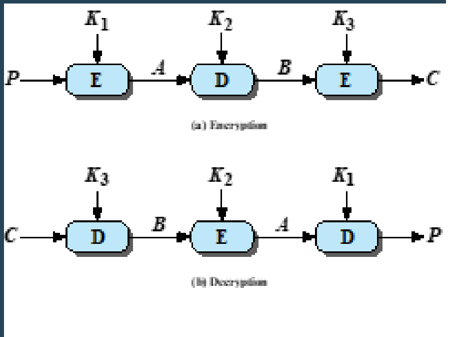


Figure 20.2 Triple DES

Block & Stream Ciphers

Block Cipher

- Processes the input one block of elements at a time
- Produces an output block for each input block
- Can reuse keys
- More common

Stream Cipher

- Processes the input elements continuously
- Produces output one element at a time
- Primary advantage is that they are almost always faster and use far less code
- Encrypts plaintext one byte at a time
- Pseudorandom stream is one that is unpredictable without knowledge of the input key

Over a data communications channel or a browser/Web link, a **stream cipher might be the better alternative.**

For applications that deal with blocks of data, such **as file transfer, e-mail, and database**, block ciphers may be more appropriate. However, either type of cipher can be used in virtually any application.

ADVANCED ENCRYPTION STANDARD

- AES is a non-Feistel cipher that encrypts and decrypts a **data block of 128 bits.**
- AES has defined **three versions, with 10, 12, and 14 rounds.**
- Each version uses **a different cipher key size (128, 192, or 256).**
- **No: of key generated = no: of round+1(extra for pre-round transformation)**

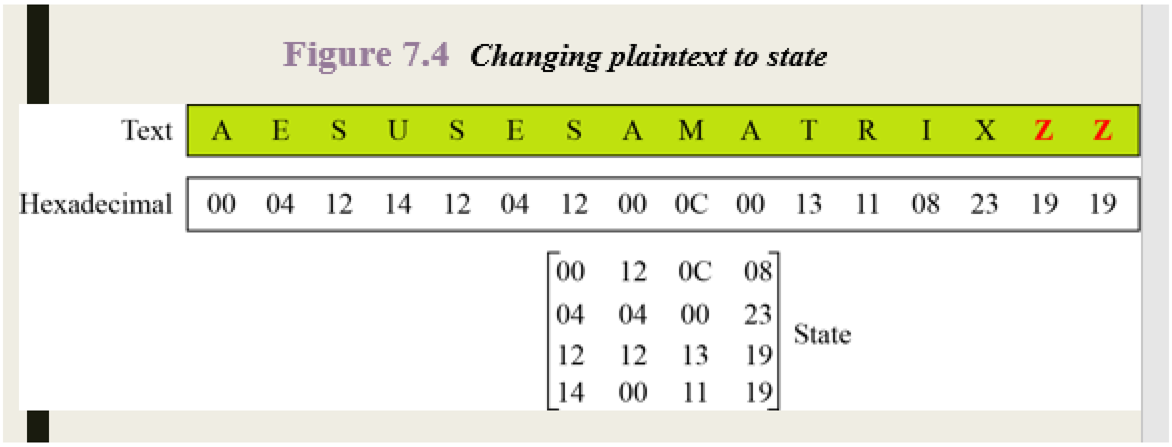
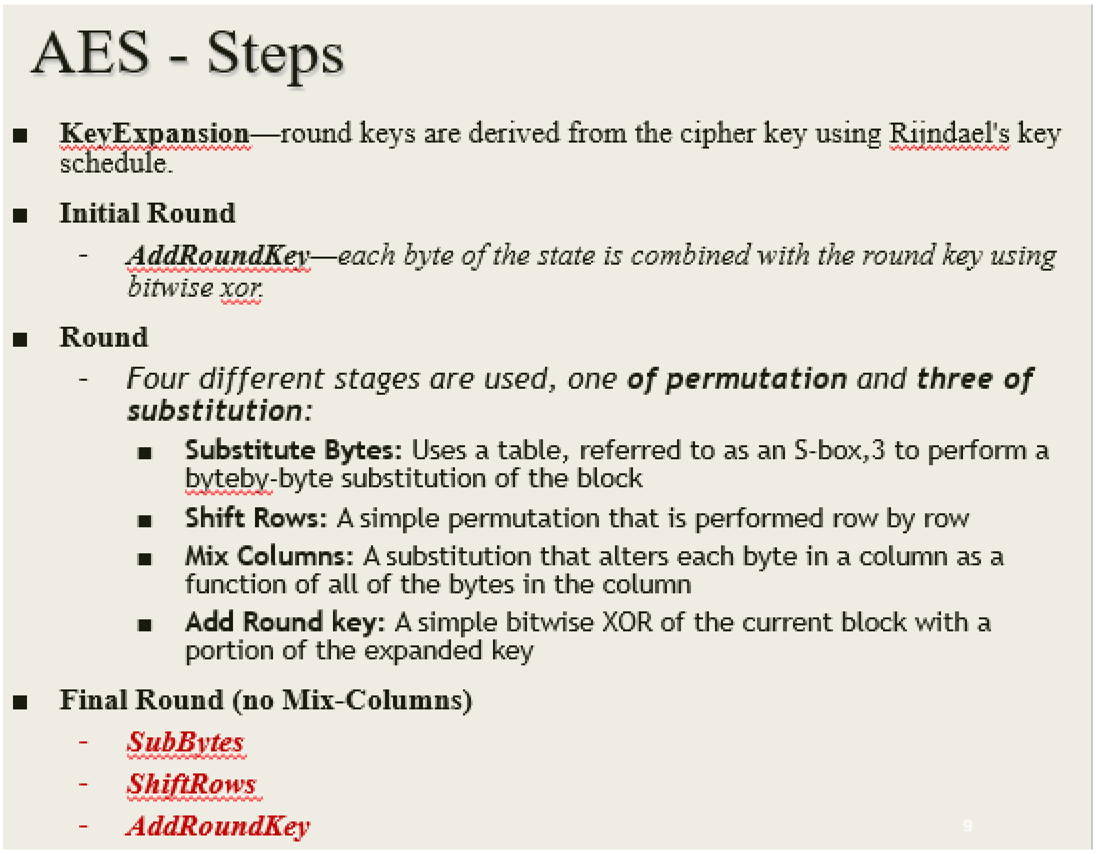


Figure 7.4 ***Changing plaintext to state***

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \text{State}$$



AES - Steps

- KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule.
- **Initial Round**
  - *AddRoundKey—each byte of the state is combined with the round key using bitwise xor.*
- **Round**
  - *Four different stages are used, one **of permutation** and **three of substitution**:*
    - **Substitute Bytes:** Uses a table, referred to as an S-box,3 to perform a byteby-byte substitution of the block
    - **Shift Rows:** A simple permutation that is performed row by row
    - **Mix Columns:** A substitution that alters each byte in a column as a function of all of the bytes in the column
    - **Add Round key:** A simple bitwise XOR of the current block with a portion of the expanded key
- **Final Round (no Mix-Columns)**
  - ***SubBytes***
  - ***ShiftRows***
  - ***AddRoundKey***

Substitute-Bytes:

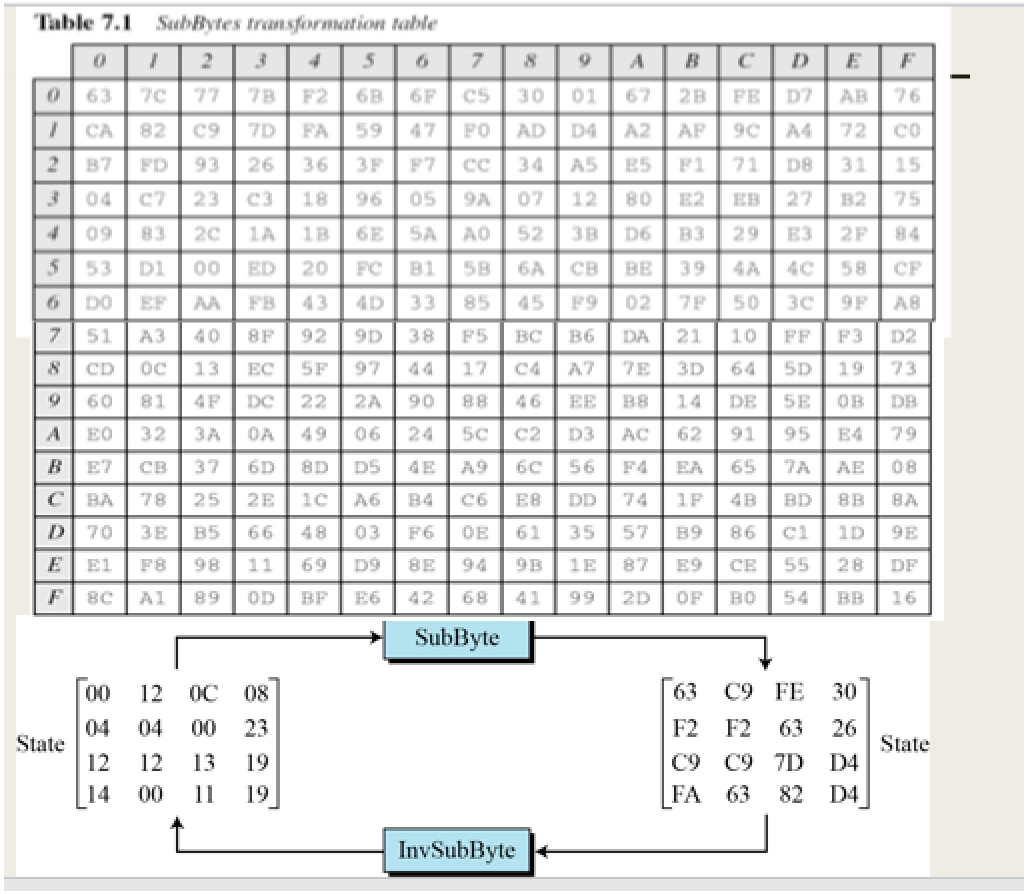


Table 7.1 *SubBytes transformation table*

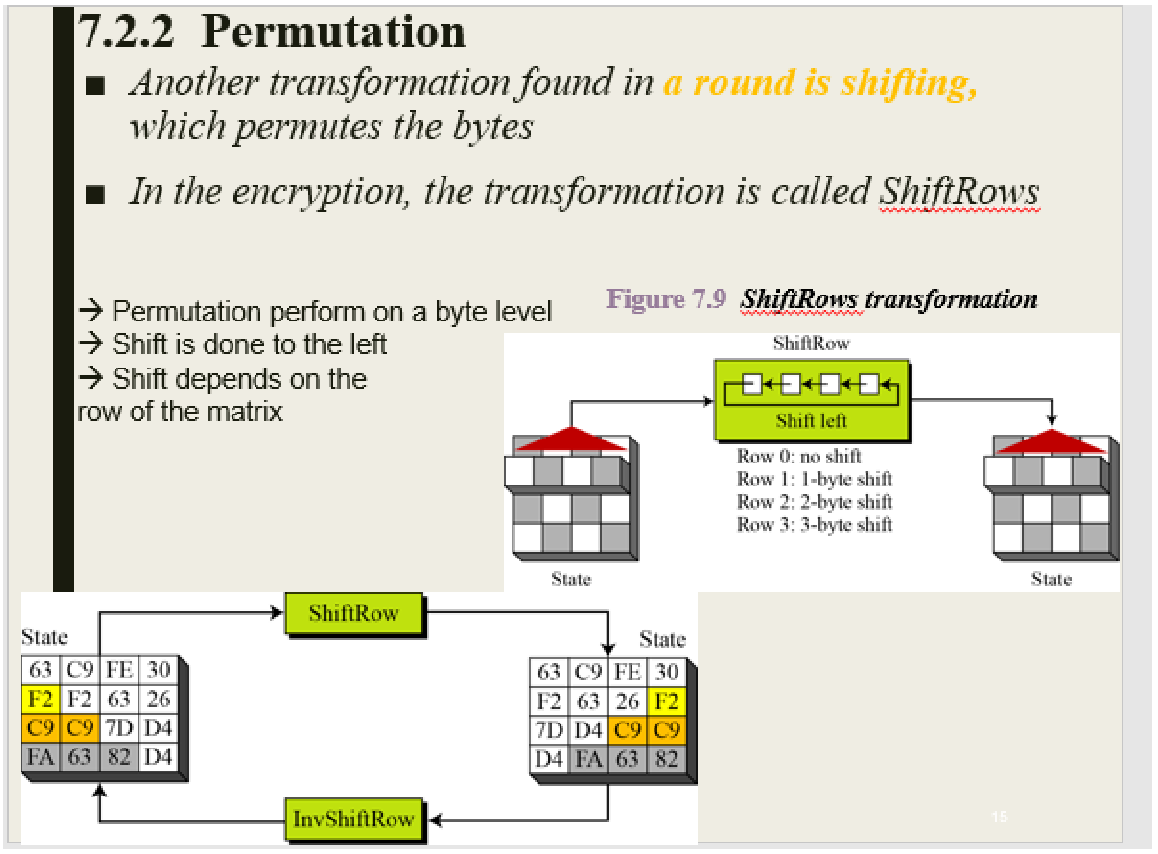
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

00 → 0 row, 0 col = 63

23 → 2 row, 3 col = 26

19 → 1st row,9th col = D4

0C → 0 row, 12th col = FE



7.2.2 Permutation

- *Another transformation found in **a round is shifting,** which permutes the bytes*
- *In the encryption, the transformation is called ShiftRows*

→ Permutation perform on a byte level
→ Shift is done to the left
→ Shift depends on the row of the matrix

Figure 7.9 ***ShiftRows transformation***

0th row → no change

1st row → 1 byte left shift

2nd row → 2 byte left shift

3rd row → 3 byte left shift

MixColumns

The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.

Each column is multiplied by a constant matrix

Example

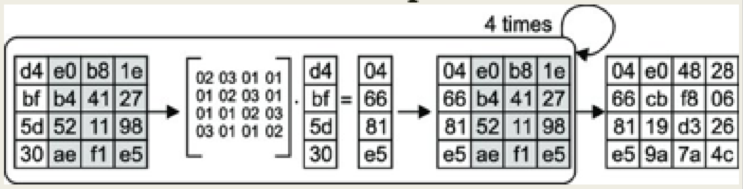
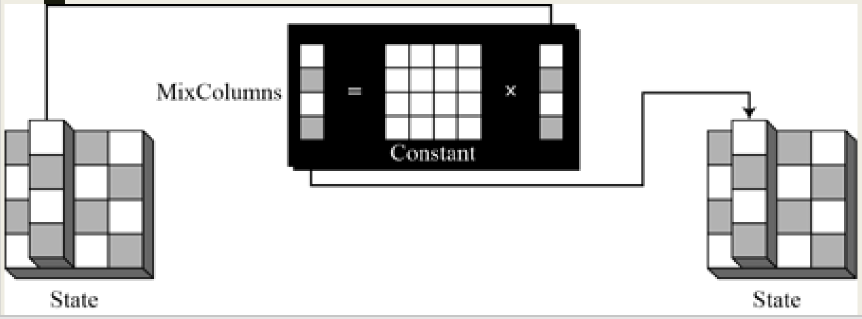


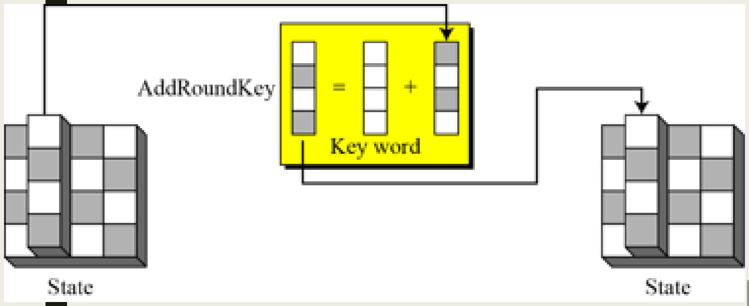
Figure 7.13 *MixColumns transformation*

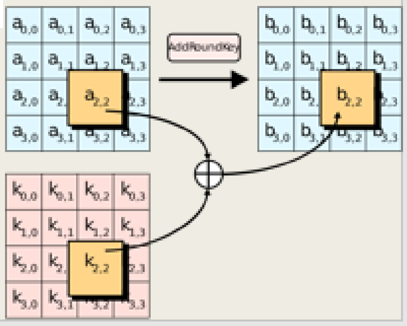


AddRoundKey

In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key.

Figure 7.15 *AddRoundKey transformation*

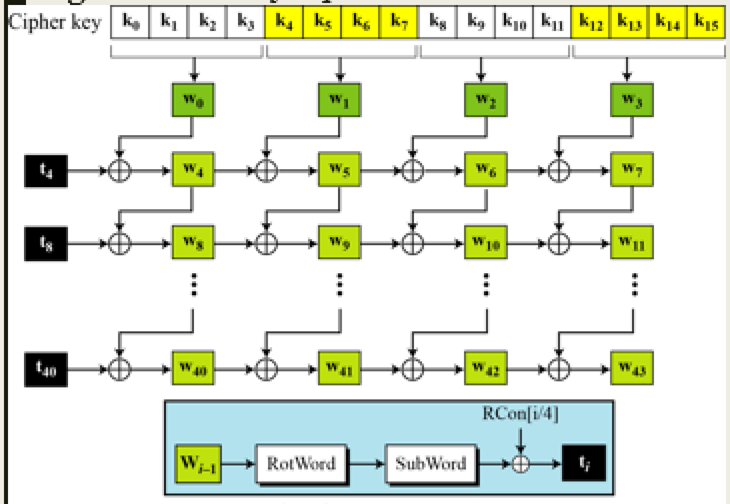




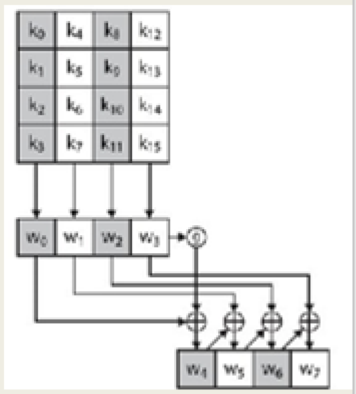
AES Key Expansion

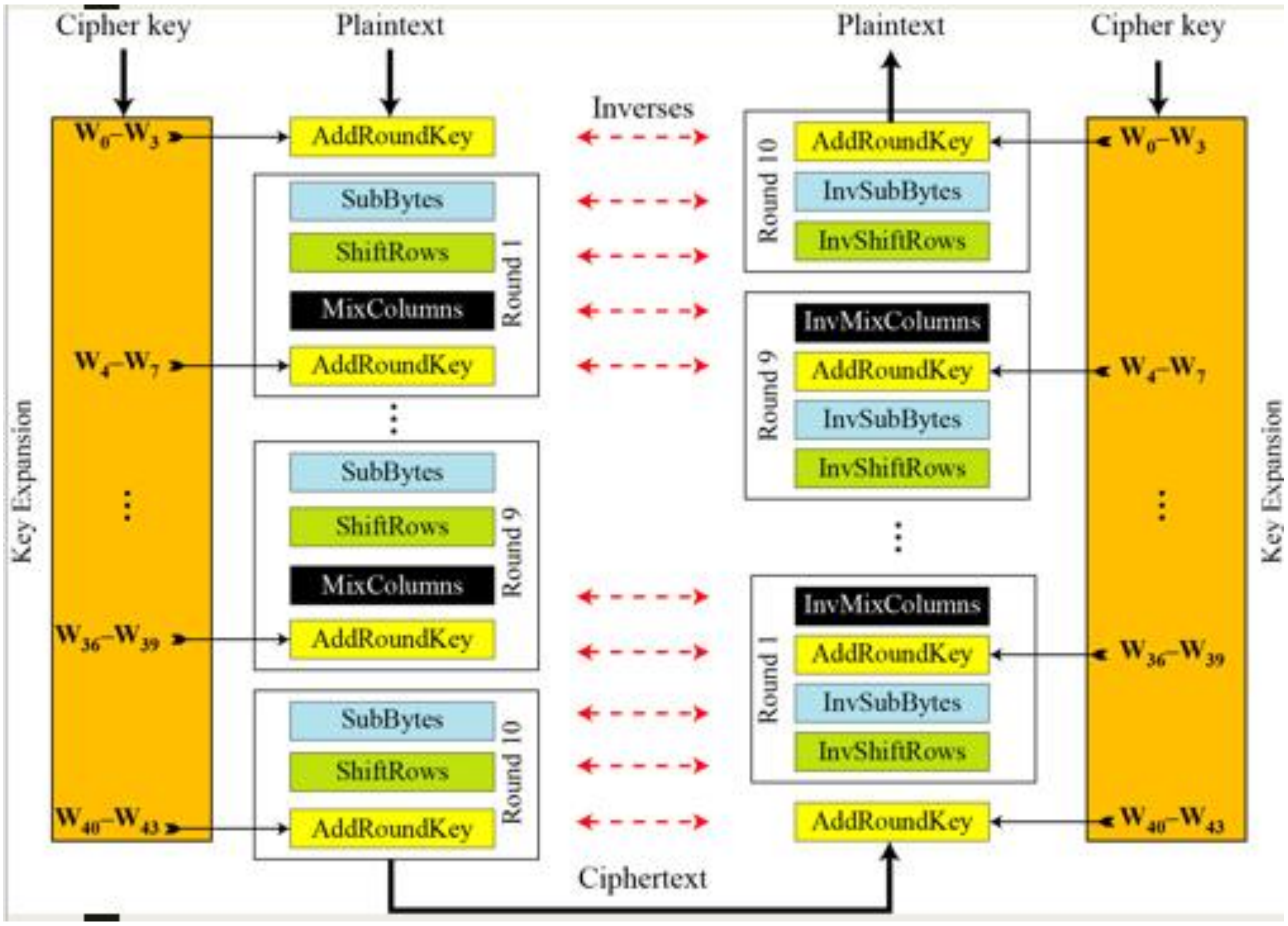
- *To create round keys for each round, AES uses a key-expansion process.* The AES key expansion algorithm takes ***as input a 4-word (16-byte) key*** and produces a ***linear array of 44 words***. *This is sufficient to provide a 4-word round key for the initial Add Round Key stage and each of the 10 rounds of the cipher.*

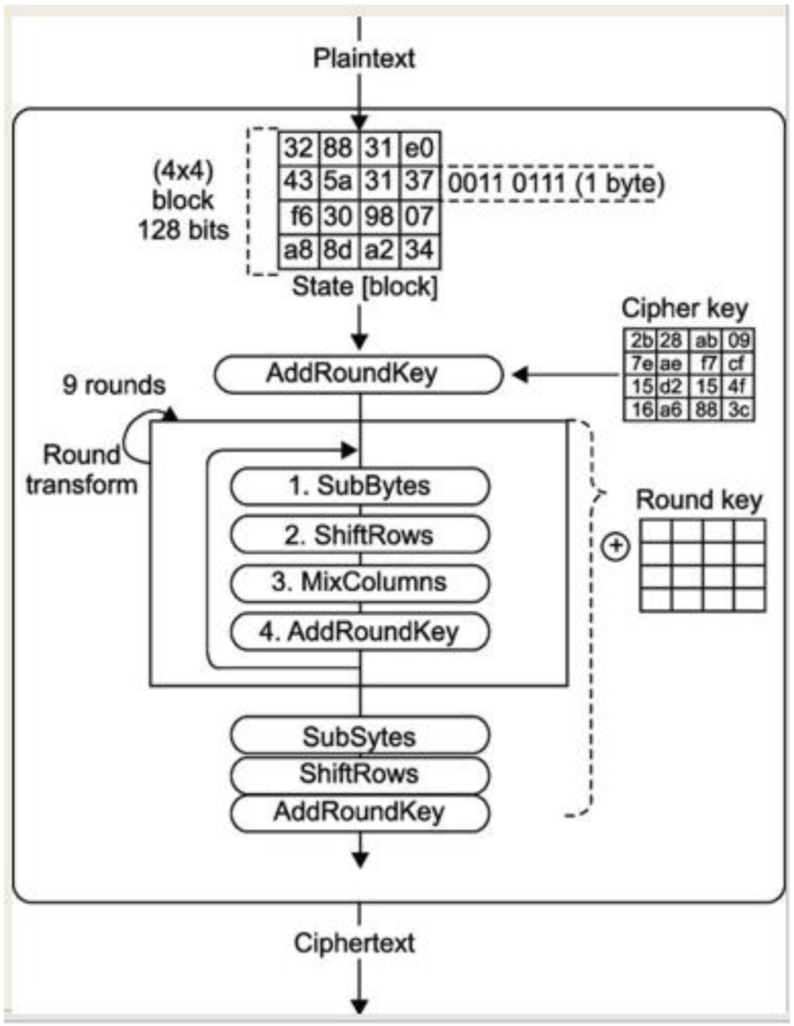
Figure 7.16 ***Key expansion in AES***



Making of t_i (temporary) words $i = 4\ N_r$







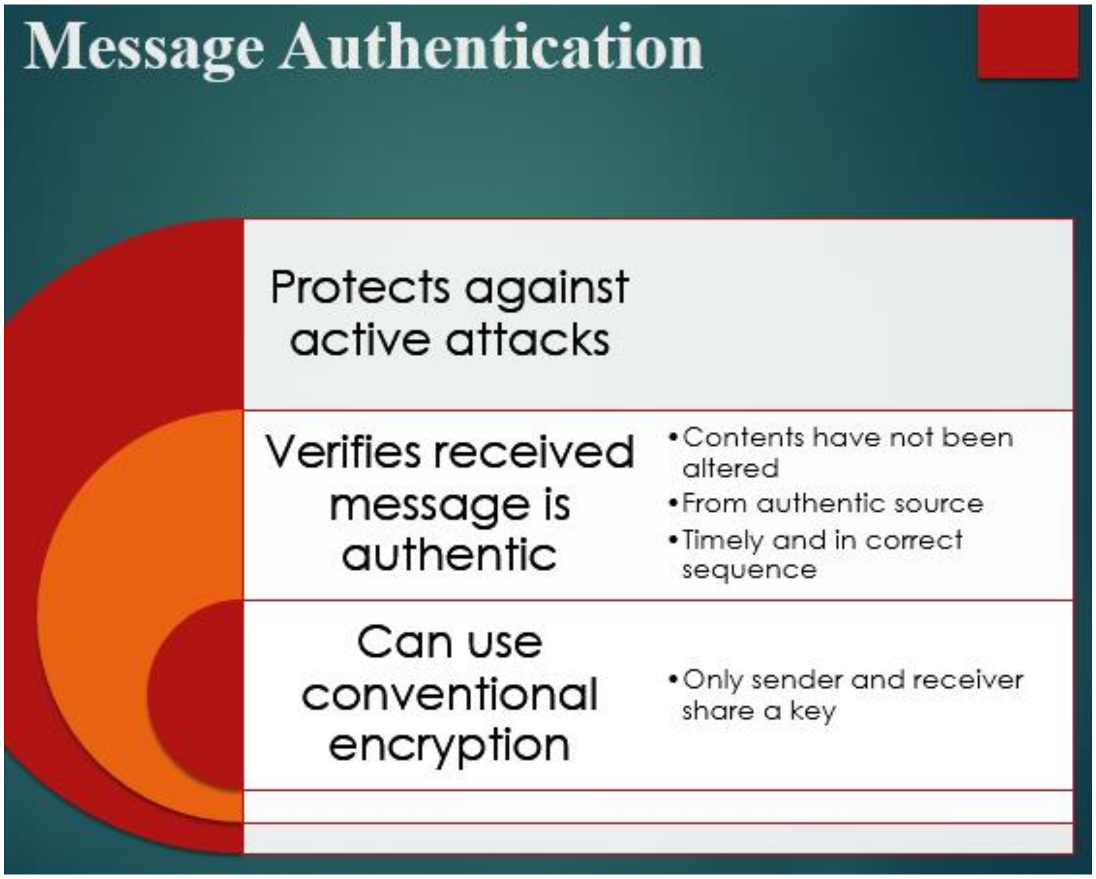
Claude Shannon (1949) gave two properties that a good cryptosystem should have to hinder statistical analysis: **diffusion** and **confusion**.

- **Diffusion** means that if we change a character of the plaintext, then several characters of the ciphertext should change, and similarly, if we change a character of the ciphertext, then several characters of the plaintext should change.

```
Plaintext 1:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Plaintext 2:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
Ciphertext 1: 63 2C D4 5E 5D 56 ED B5 62 04 01 A0 AA 9C 2D 8D
Ciphertext 2: 26 F3 9B BC A1 9C 0F B7 C7 2E 7E 30 63 92 73 13
```

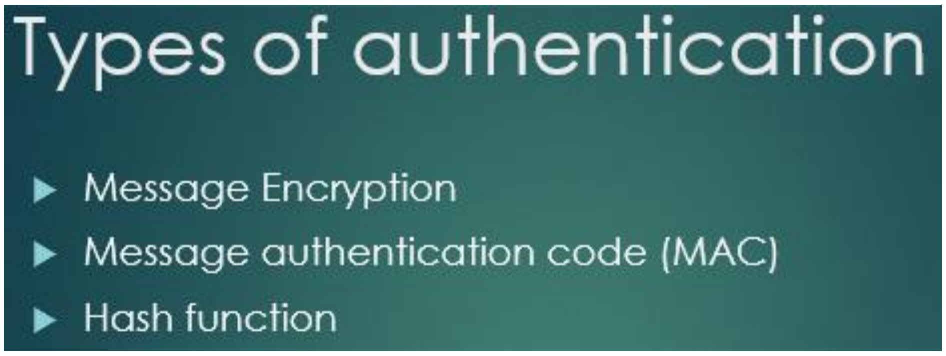
- **Confusion** means that the key does not relate in a simple way to the
ciphertext. In particular, each character of the ciphertext should depend on several parts of the key.

`Confusion = Substitution a --> b, Diffusion = Transposition or Permutation abcd --> dacb`



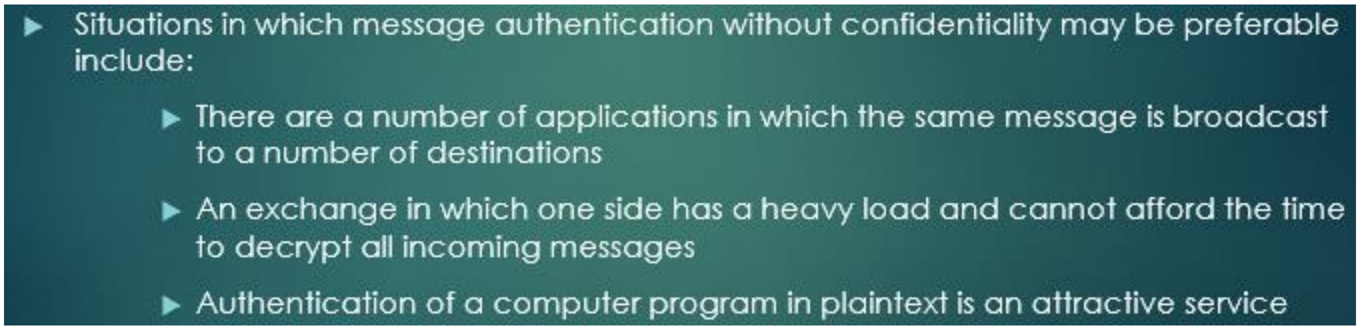
Message Authentication

Protects against active attacks	
Verifies received message is authentic	• Contents have not been altered • From authentic source • Timely and in correct sequence
Can use conventional encryption	• Only sender and receiver share a key

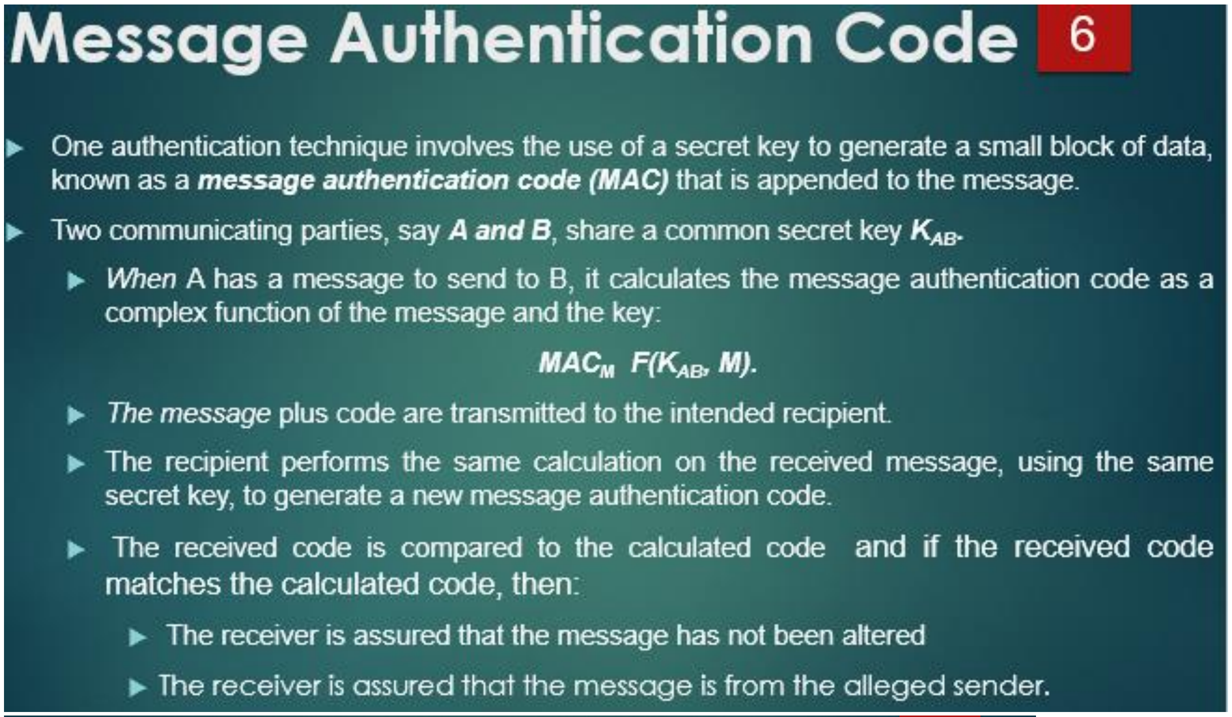


Types of authentication

- Message Encryption
- Message authentication code (MAC)
- Hash function



- Situations in which message authentication without confidentiality may be preferable include:
  - There are a number of applications in which the same message is broadcast to a number of destinations
  - An exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages
  - Authentication of a computer program in plaintext is an attractive service



Message Authentication Code 6

- One authentication technique involves the use of a secret key to generate a small block of data, known as a ***message authentication code (MAC)*** that is appended to the message.
- Two communicating parties, say ***A and B***, share a common secret key K_{AB}.
  - *When* A has a message to send to B, it calculates the message authentication code as a complex 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 and if the received code matches the calculated code, then:
    - The receiver is assured that the message has not been altered
    - The receiver is assured that the message is from the alleged sender.

7

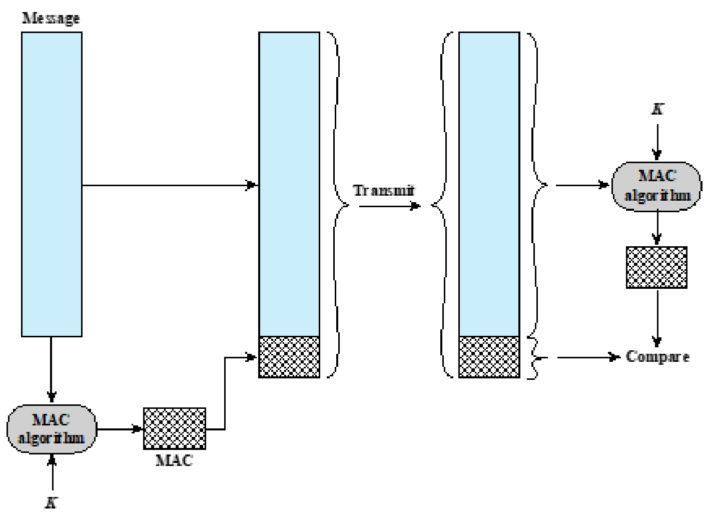
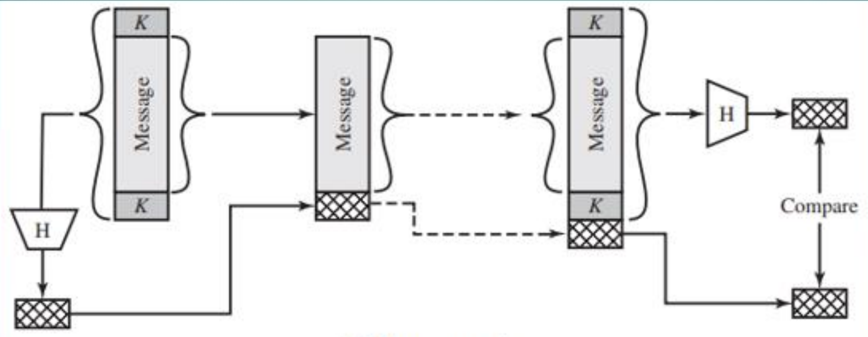


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

DES or AES is used to generate an encrypted version of the message, and some of the bits of ciphertext are used as the code. Longer codes are used to provide sufficient collision resistance. One difference is that the authentication algorithm need not be reversible, as it must for decryption.



(c) Using secret value

Three ways in which the message can be authenticated using a hash function (Figure 2.5)

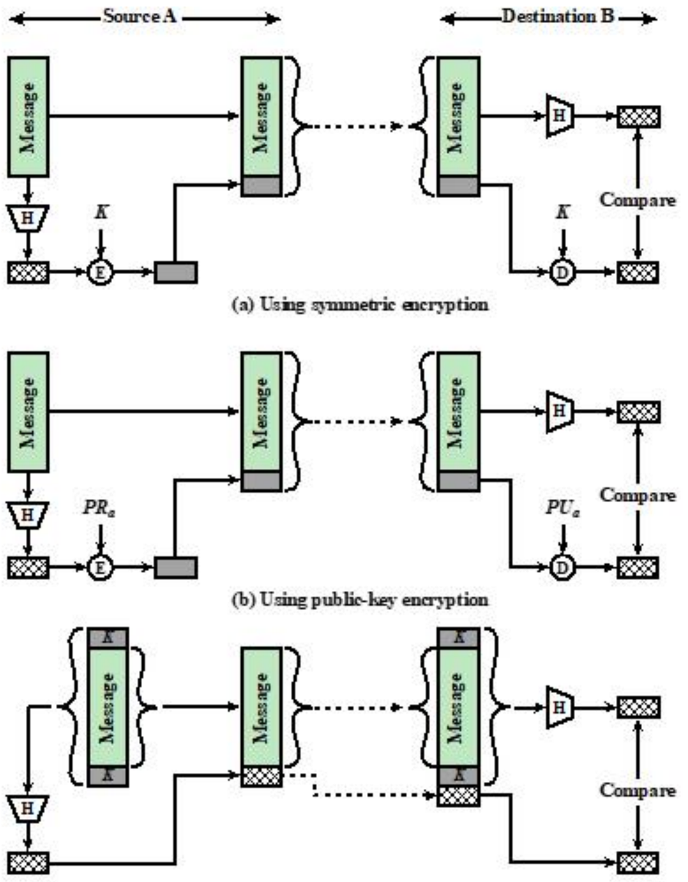
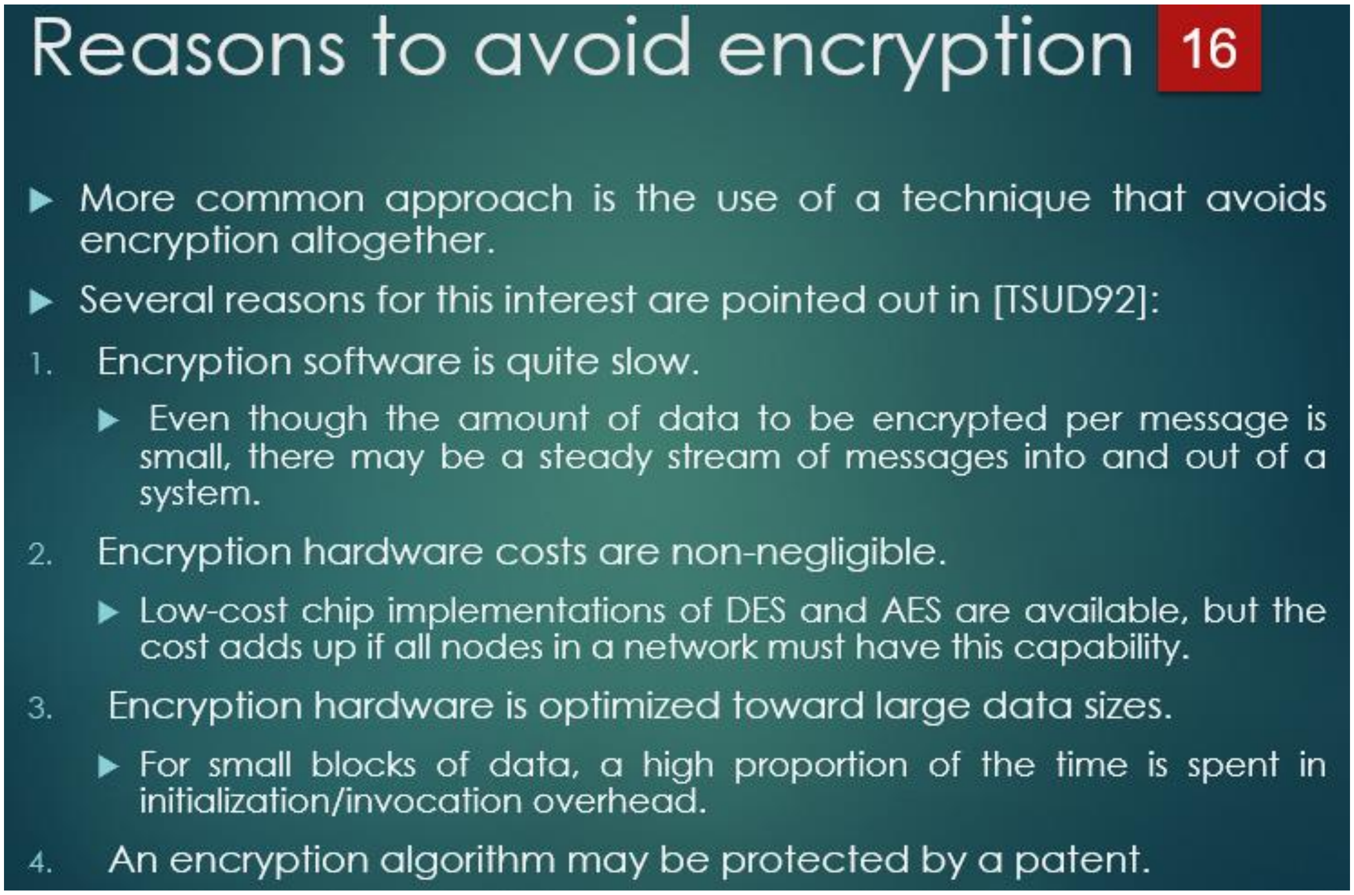


Figure 2.5 Message Authentication Using a One-Way Hash Function.

- 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.

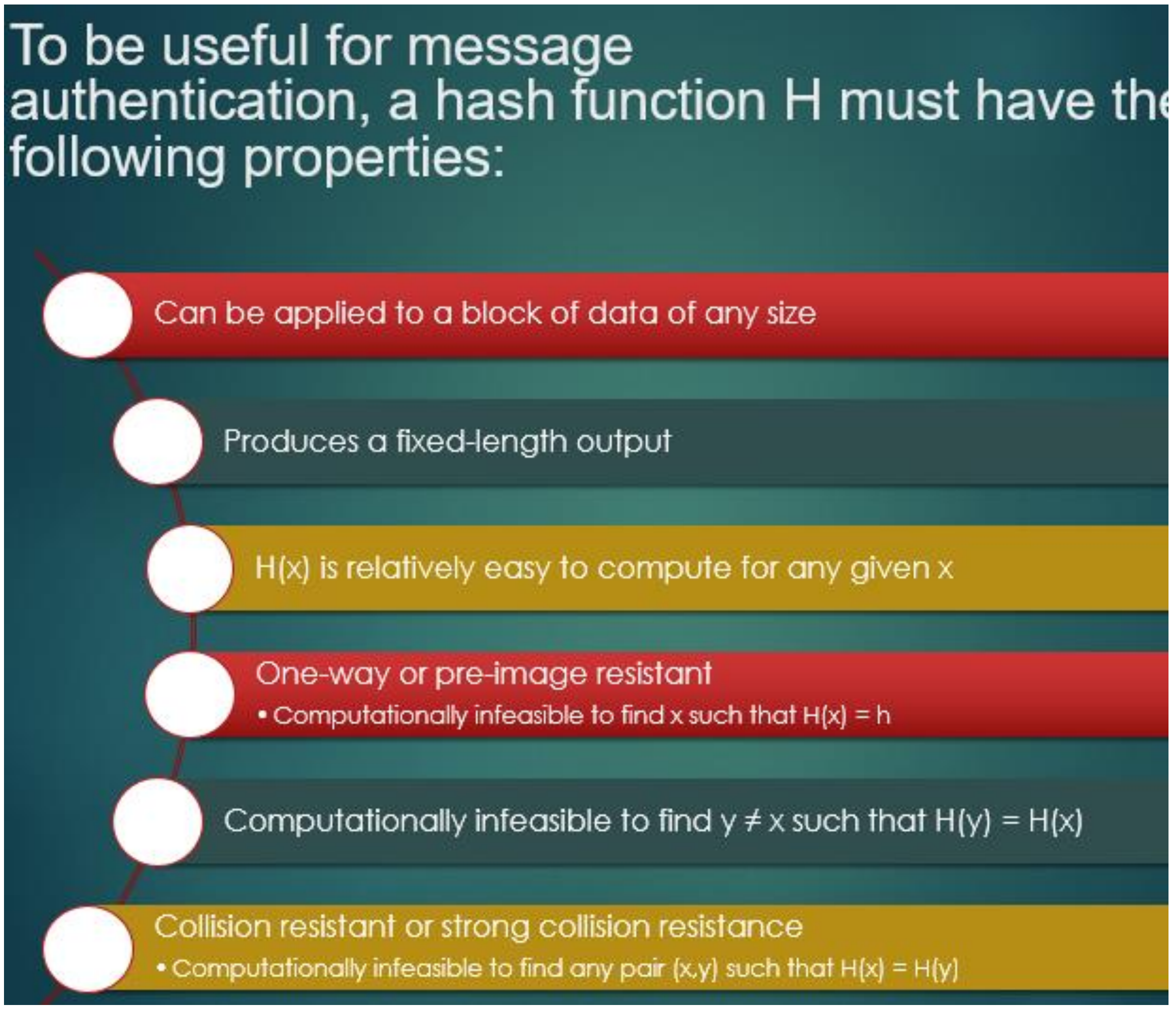
Figure 2.5a and b approaches have an advantage over approaches that encrypt the entire message, in that less computation is required.



Reasons to avoid encryption 16

- More common approach is the use of a technique that avoids encryption altogether.
- Several reasons for this interest are pointed out in [TSUD92]:

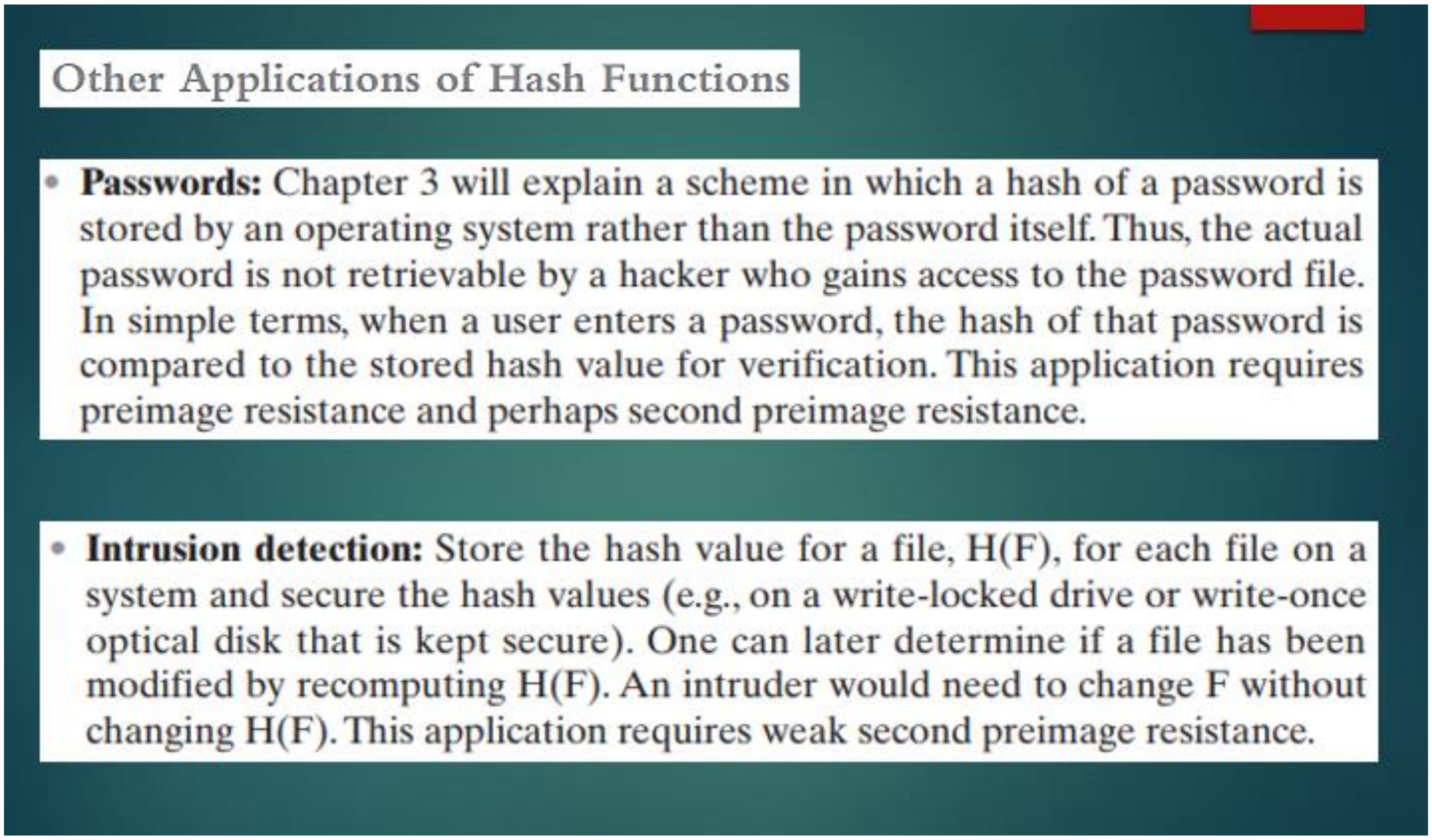
1. Encryption software is quite slow.
   - Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
2. Encryption hardware costs are non-negligible.
   - Low-cost chip implementations of DES and AES are available, but the cost adds up if all nodes in a network must have this capability.
3. 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.
4. An encryption algorithm may be protected by a patent.



To be useful for message authentication, a hash function H must have th following properties:

- Can be applied to a block of data of any size
- Produces a fixed-length output
- H(x) is relatively easy to compute for any given x
- One-way or pre-image resistant
  - Computationally infeasible to find x such that $H(x) = h$
- Computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$
- Collision resistant or strong collision resistance
  - Computationally infeasible to find any pair (x,y) such that $H(x) = H(y)$

Pre-image resistance is a property of cryptographic hash functions that refers to the difficulty of finding an input (or 'pre-image') that will produce a given hash value. A hash function is said to be pre-image resistant if it is computationally infeasible to find a pre-image for a given hash value, even if the attacker has access to the hash function and its internal structure.



Other Applications of Hash Functions

- **Passwords:** Chapter 3 will explain a scheme in which a hash of a password is stored by an operating system rather than the password itself. Thus, the actual password is not retrievable by a hacker who gains access to the password file. In simple terms, when a user enters a password, the hash of that password is compared to the stored hash value for verification. This application requires preimage resistance and perhaps second preimage resistance.
- **Intrusion detection:** Store the hash value for a file, H(F), for each file on a system and secure the hash values (e.g., on a write-locked drive or write-once optical disk that is kept secure). One can later determine if a file has been modified by recomputing H(F). An intruder would need to change F without changing H(F). This application requires weak second preimage resistance.

Misconceptions concerning public-key encryption

1. **Public-key encryption is more secure from cryptanalysis** than symmetric encryption. In fact, the security of any encryption scheme depends on
   - *The length of the key*
   - *The computational work involved in breaking a cipher.*
2. A second misconception is that **public-key encryption is a general-purpose technique that has made symmetric encryption obsolete**. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned.
3. Finally, there is a feeling that **key distribution is important when using public-key encryption**, compared to the rather inconvenient handshaking involved with key distribution centers for symmetric encryption.
4. For public-key key distribution, some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler or any more efficient than those required for symmetric encryption

The essential steps of public key encryption 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.
    - (Figure 2.6a) 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.

Requirements for Public-Key Cryptography:

1. It is computationally easy for a party B to generate a pair (public key PUb, private key PRb).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext: C = E(PUb, M)
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: M = D(PRb, C) = D[PRb, E(PUb, M)]
4. It is computationally infeasible for an opponent, knowing the public key, PUb, to determine the private key, PRb.
5. It is computationally infeasible for an opponent, knowing the public key, PUb, and a ciphertext, C, to recover the original message, M.

RSA Algorithm:

Digital Signature:

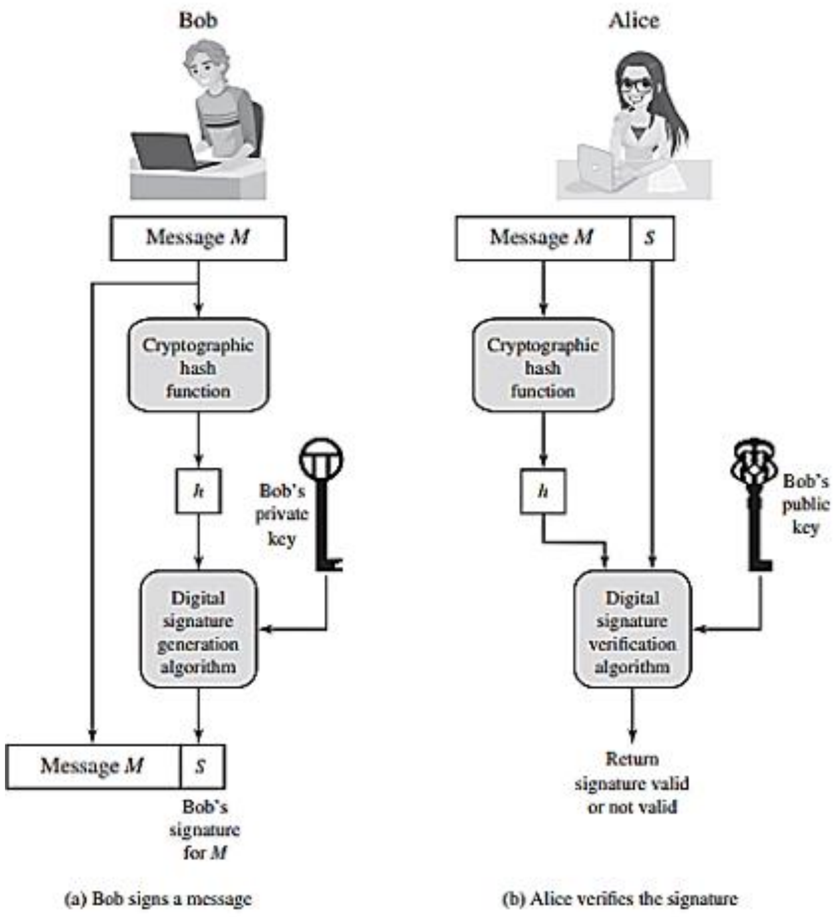
Public-key encryption can be used for authentication with a technique known as the **digital signature.**

Provides a mechanism for **verifying:**

- **origin authentication**
- **data integrity and**
- **signatory non-repudiation.**

Steps

- Bob wants to send a message to Alice. He wants Alice to be certain that the message is indeed from him. For this purpose:
    1. Bob uses a secure hash function, such as SHA-512, to generate a hash value for the message.
    2. That hash value, together with Bob's private key, serve as input to a digital signature generation algorithm that produces a short block that functions as a digital signature.
    3. sends the message with the signature attached.
- When Alice receives the message plus signature, she:
    1. calculates a hash value for the message;
    2. provides the hash value and Bob's public key as inputs to a digital signature verification algorithm. If the algorithm returns the result that the signature is valid, Alice is assured that the message must have been signed by Bob.


Bob
Alice
Message M
Message M
S
Cryptographic hash function
Cryptographic hash function
h
h
Bob's private key
Bob's public key
Digital signature generation algorithm
Digital signature verification algorithm
Message M
S
Bob's signature for M
Return signature valid or not valid
(a) Bob signs a message
(b) Alice verifies the signature

Digital Signature provides Authentication, Data Integrity, but no Confidentiality

Public Key Certificate:

1. User software (client) creates a pair of keys: one public and one private.
2. Client prepares an unsigned certificate that includes the user ID and user's public key.
3. User provides the unsigned certificate to a CA in some secure manner. This might require a face-to-face meeting, the use of registered e-mail, or happen via a Web form with e-mail verification.
4. CA creates a signature as follows:
    a. CA uses a hash function to calculate the hash code of the unsigned certificate.
    b. CA generates digital signature using the CA's private key and a signature generation algorithm.
5. CA attaches the signature to the unsigned certificate to create a signed certificate.
6. CA returns the signed certificate to client.
7. Client may provide the signed certificate to any other user.
8. Any user may verify that the certificate is valid as follows:
    1. User calculates the hash code of certificate (not including signature).
    2. User verifies digital signature using CA's public key and the signature verification algorithm. The algorithm returns a result of either signature valid or invalid

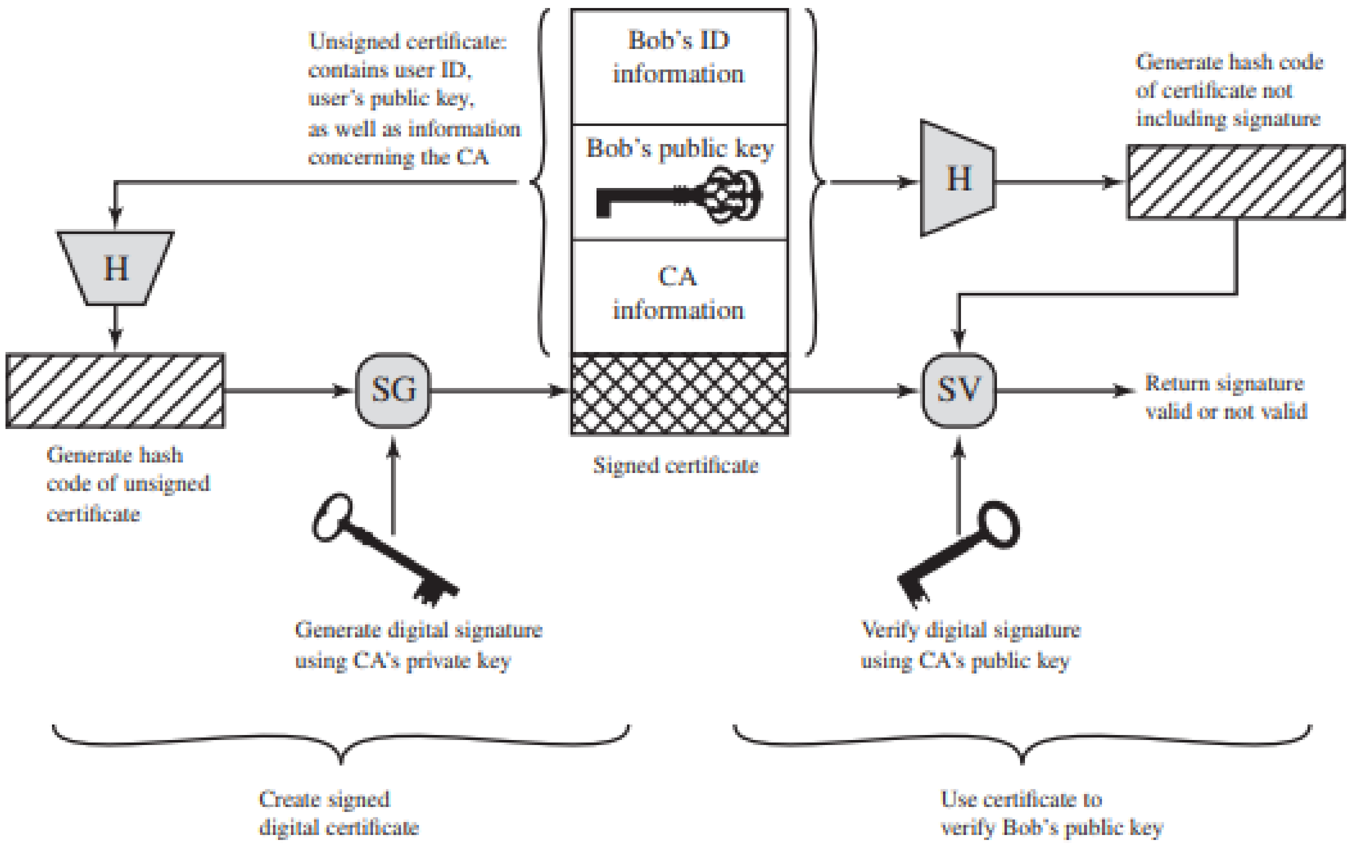

Unsigned certificate: contains user ID, user's public key, as well as information concerning the CA
Bob's ID information
Bob's public key
CA information
Signed certificate
H
Generate hash code of unsigned certificate
SG
Generate digital signature using CA's private key
H
Generate hash code of certificate not including signature
SV
Return signature valid or not valid
Verify digital signature using CA's public key
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 2.8 **Public-Key Certificate Use**