Module 4
TRANSPOSITION CIPHERS

- These ciphers hide the message by rearranging the letter order without altering the actual letters used.
- The cipher text has the same frequency distribution as the plaintext.

Rail Fence cipher

- The plain text is written as a sequence of diagonals.
- Then read off as a sequence of rows.
- eg. "meet me after the party“

m e m a t r h p r y

e t e f e t e a t

The encrypted message is

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

- This type of cipher can be attacked easily.

Column transposition

- A more complex scheme is to write the message in rectangle , row by row, and read the message column by column.
- Permute the order of the columns.
- The order of the columns then becomes the key of the algorithm.

- 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
- Cipher text is TTNAAPTMTSUOAODWCOIXKNLYPETZ

- The transposition cipher can be made more secure by performing more than one stage of transposition.
- 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 o l x k
- n l y p e t z
- The cipher text is
- NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

Product Ciphers

- Ciphers using substitutions or transpositions are not secure because of language characteristics
- Several ciphers in succession.

– two substitutions make a more complex substitution

– two transpositions make more complex transposition

– but a substitution followed by a transposition makes a new much harder cipher

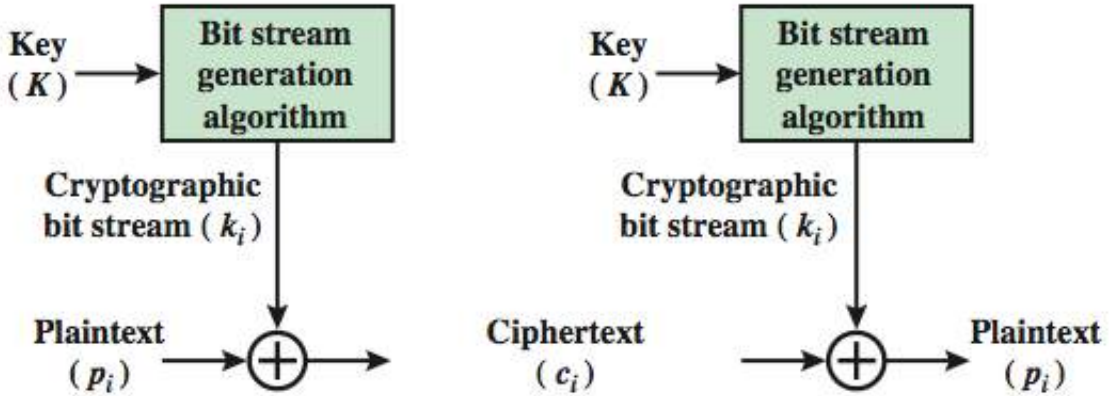
Block Ciphers

- Most widely used types of cryptographic algorithms
- provide secrecy /authentication services
- DES (Data Encryption Standard)

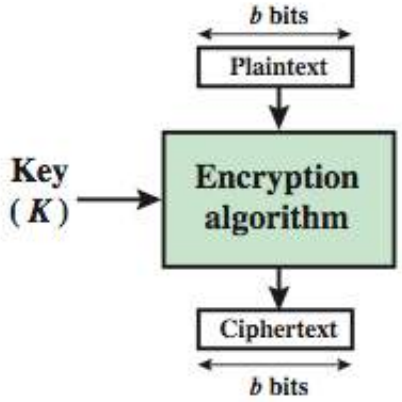
Block vs Stream Ciphers

- A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Typically, a block size of 64 or 128 bits is used.
- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.
- In the ideal case, a one-time pad cipher would be used in which the key stream (k) is as long as the plaintext bit stream (p).

Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Feistel Cipher Structure

- A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits.
- There are 2^npossible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block.
- Such a transformation is called reversible, or nonsingular.

Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

General substitution cipher for n=4

- A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 cipher text bits.

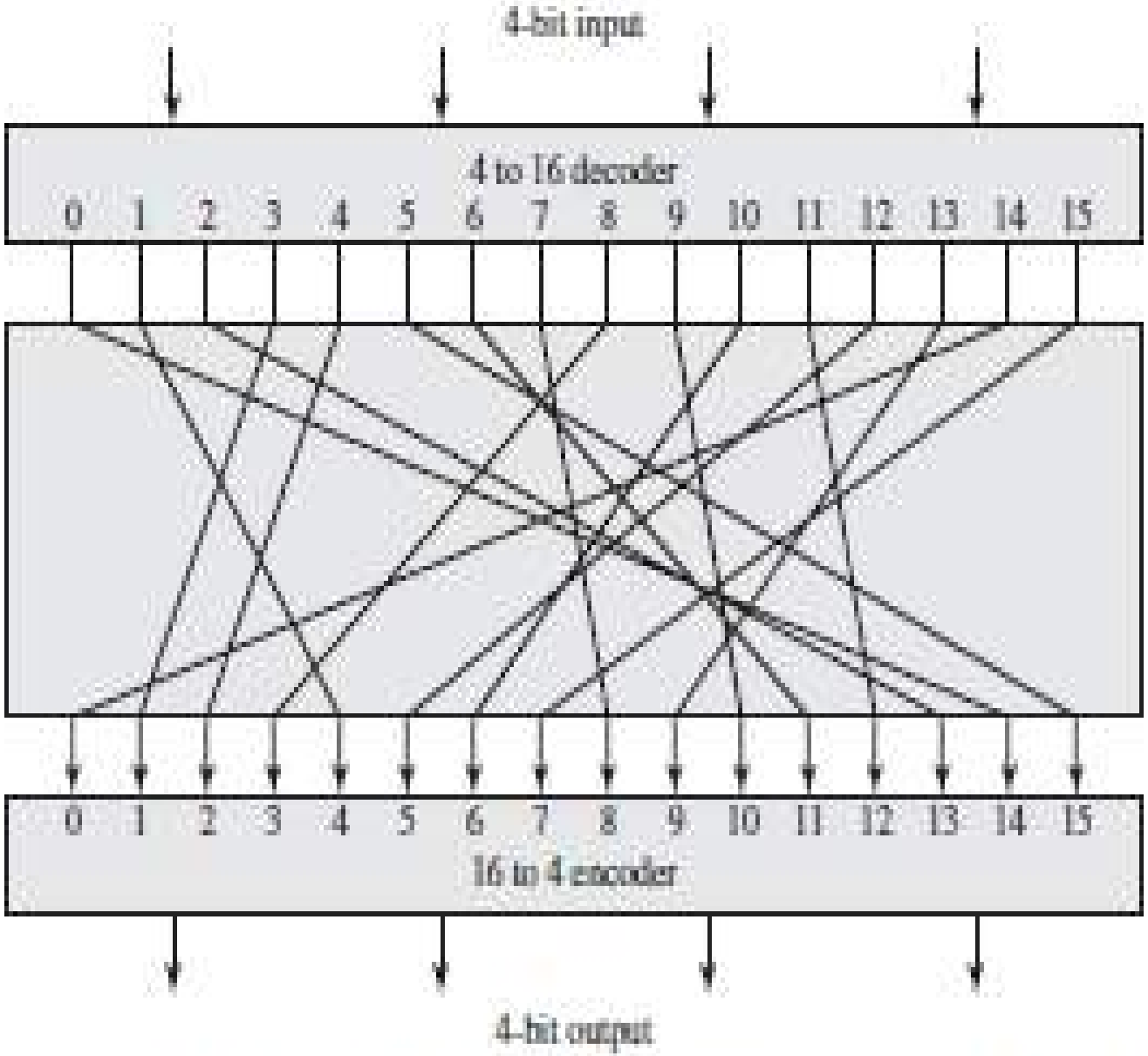


Figure 3.2 General n-bit-n-bit Block Substitution (shown with $n = 4$)

Encryption Table

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Decryption Table

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations.
- **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

- Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion and diffusion functions*

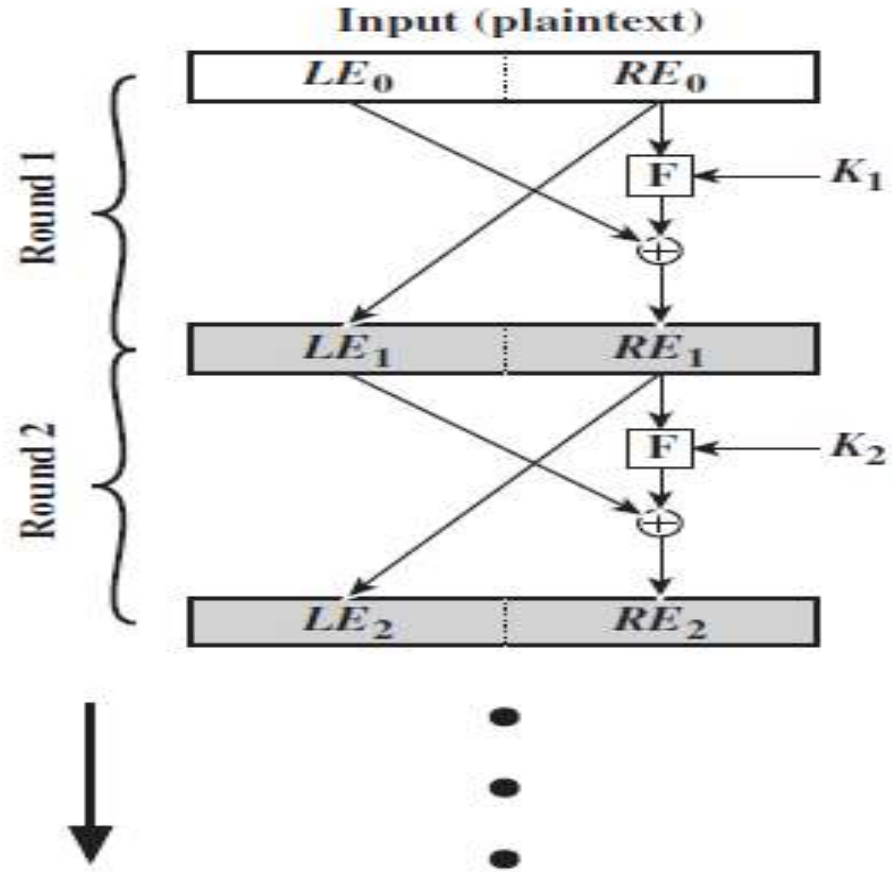
Claude Shannon Substitution-Permutation Ciphers

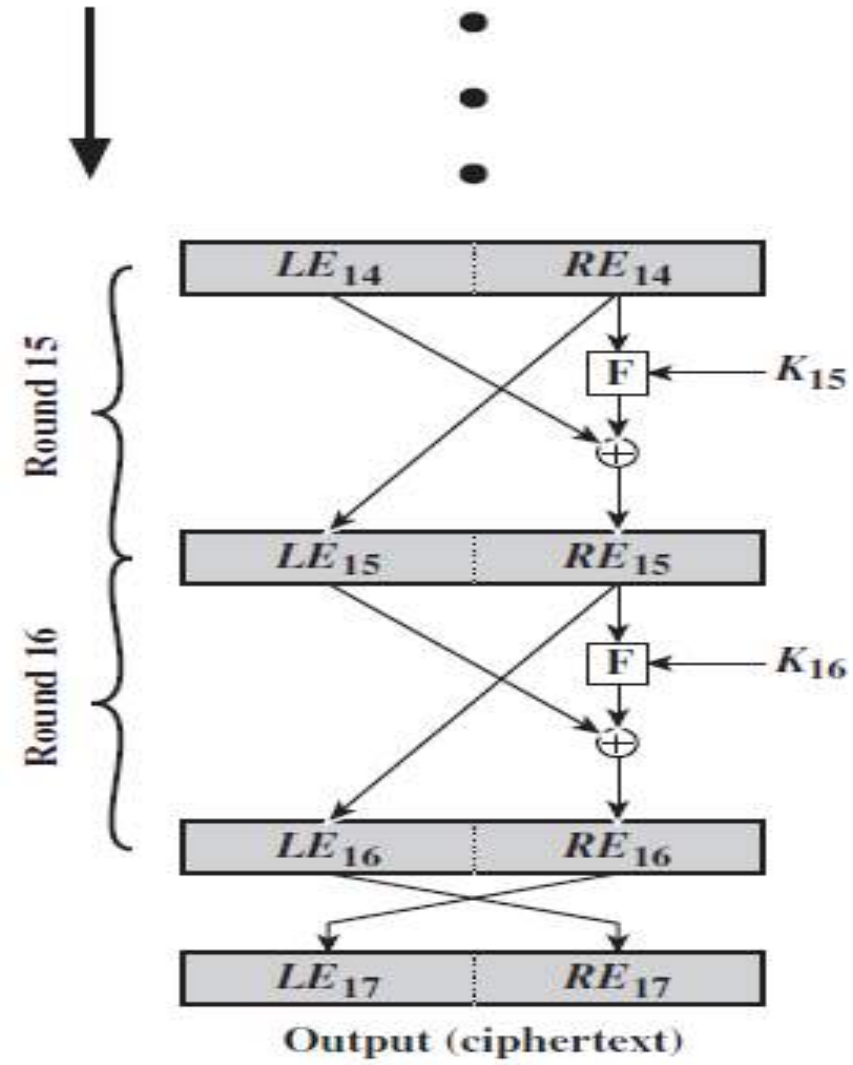
- Claude Shannon introduced idea of substitution-permutation (S-P) networks
- Basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- provide *confusion* & *diffusion* of message & key

Confusion and Diffusion

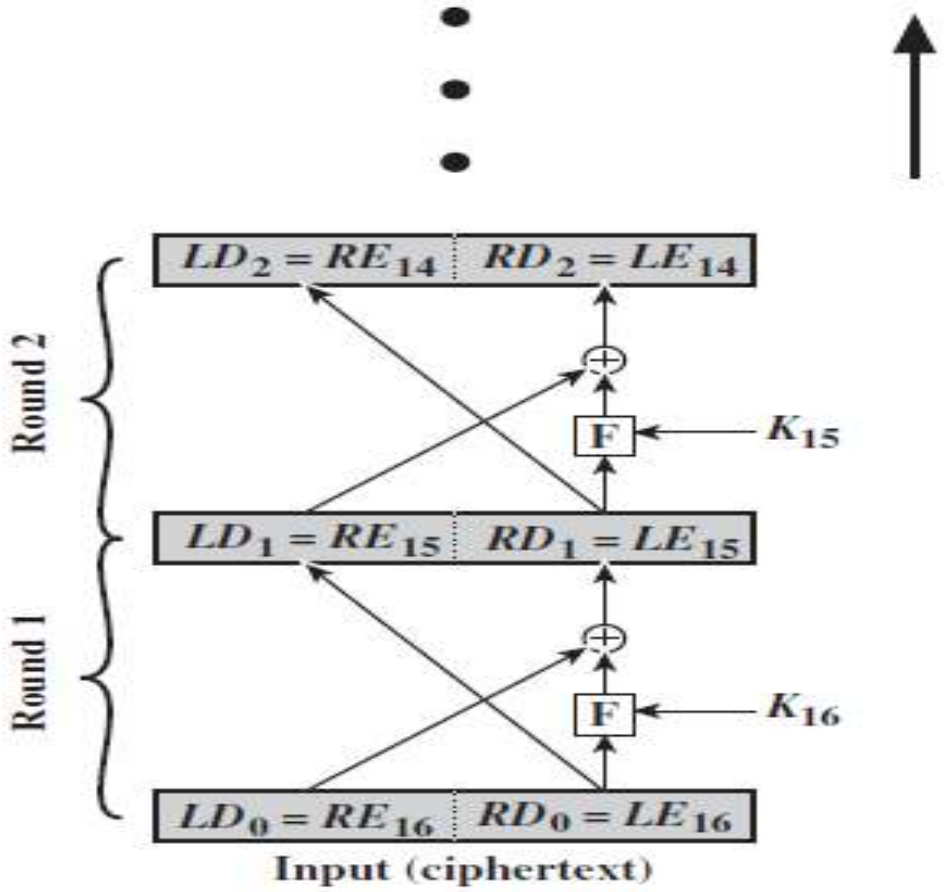
- Cipher needs to completely obscure statistical properties of original message.
- A one-time pad does this.
- More practically Shannon suggested combining S & P elements to obtain:
- **diffusion** – dissipates statistical structure of plaintext over bulk of cipher text. This is achieved by having each plaintext digit affect the value of many cipher text digits
- **confusion** – makes relationship between cipher text and key as complex as possible.

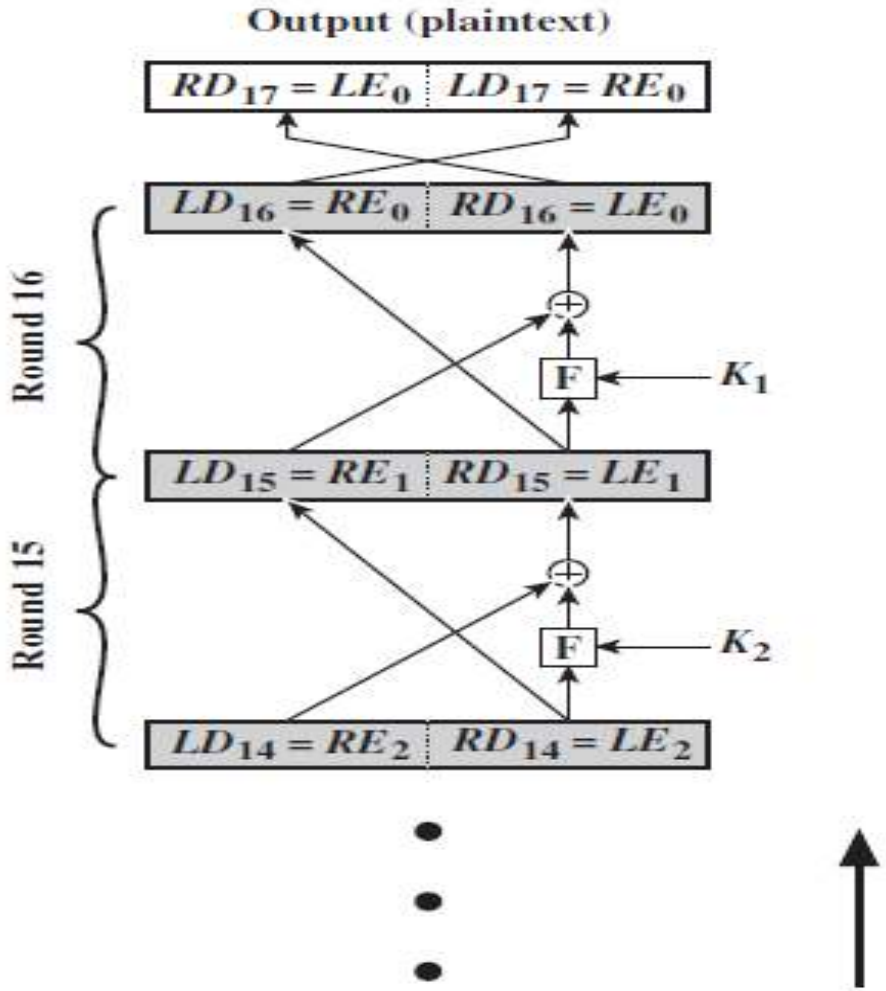
Feistel Cipher Structure




LE_{14}
RE_{14}
F
K_{15}
LE_{15}
RE_{15}
F
K_{16}
LE_{16}
RE_{16}
LE_{17}
RE_{17}
Round 15
Round 16
Output (ciphertext)

Decryption


$LD_2 = RE_{14}$
$RD_2 = LE_{14}$
F
K_{15}
$LD_1 = RE_{15}$
$RD_1 = LE_{15}$
F
K_{16}
$LD_0 = RE_{16}$
$RD_0 = LE_{16}$
Round 2
Round 1
Input (ciphertext)


Output (plaintext)
$RD_{17} = LE_0$ $LD_{17} = RE_0$
$LD_{16} = RE_0$ $RD_{16} = LE_0$
Round 16
F
K_1
$LD_{15} = RE_1$ $RD_{15} = LE_1$
Round 15
F
K_2
$LD_{14} = RE_2$ $RD_{14} = LE_2$

- The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key .
- The plaintext block is divided into two halves, $L_0$and R_0 .
- The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block.
- Each round i has as input L_{i-1} and R_{i-1} derived from the previous round as well as a sub-key K_i
- K_i is derived from the overall key K

- The subkeys K_i are different from K and from each other.
- 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 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 K_i .

- F is a function of right-half block of w bits and a subkey of y bits, which produces an output value of length *w bits: F(REi, Ki+1).*
- Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

Feistel Cipher Design Parameters

- **block size**

– increasing size improves security, but slows cipher

– Normally 64

- **key size**

– increasing size improves security, makes exhaustive key searching harder, but may slow down the cipher

– Below 64 bit is inadequate, commonly use 128 bit key

- **number of rounds**

– increasing number improves security, but slow down the cipher

– Typically 16 rounds

- **subkey generation**

– greater complexity can make analysis harder, but slows cipher

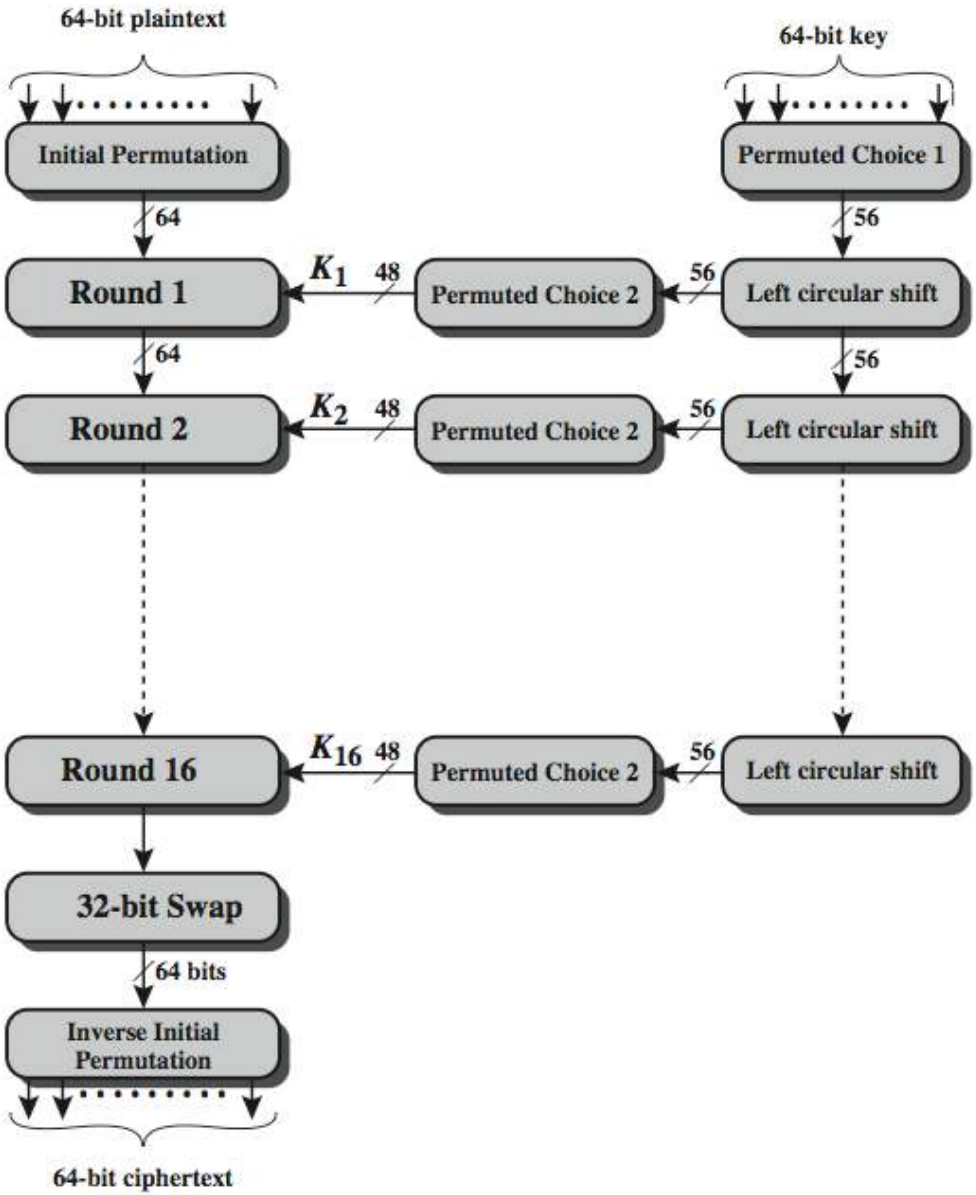
- **round function**

– greater complexity can make analysis harder, but slows cipher

Data Encryption Standard (DES)

- Most widely used block cipher in world
- encrypts 64-bit data using 56-bit key
- it transforms 64-bit i/p in a series of steps in to a 64-bit o/p
- the same steps with the same keys (in reverse order) are used to do the decryption.
- DES has become widely used, especially in financial applications

DES Encryption Overview



- Processing of the plaintext proceeds in three phases.
- 1. The 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input.*
- 2. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions.
- - The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key.
- - The left and right halves of the output are swapped to produce the pre-output.
- 3. Finally, the pre-output is passed through inverse permutation (IP-1) to produce the 64-bit ciphertext.

- Initially, the key is passed through a permutation function.
- - Then, for each of the 16 rounds, a *subkey (K i) is produced by the combination of a* left circular shift and a permutation.
- - The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

Initial Permutation IP

- The 64 bit plain text passes through an initial permutation that rearranges the bits to produce the permuted input.
- The initial permutation and its inverse are defined by tables.

64 bit input (M)

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

(a) Initial Permutation (IP)

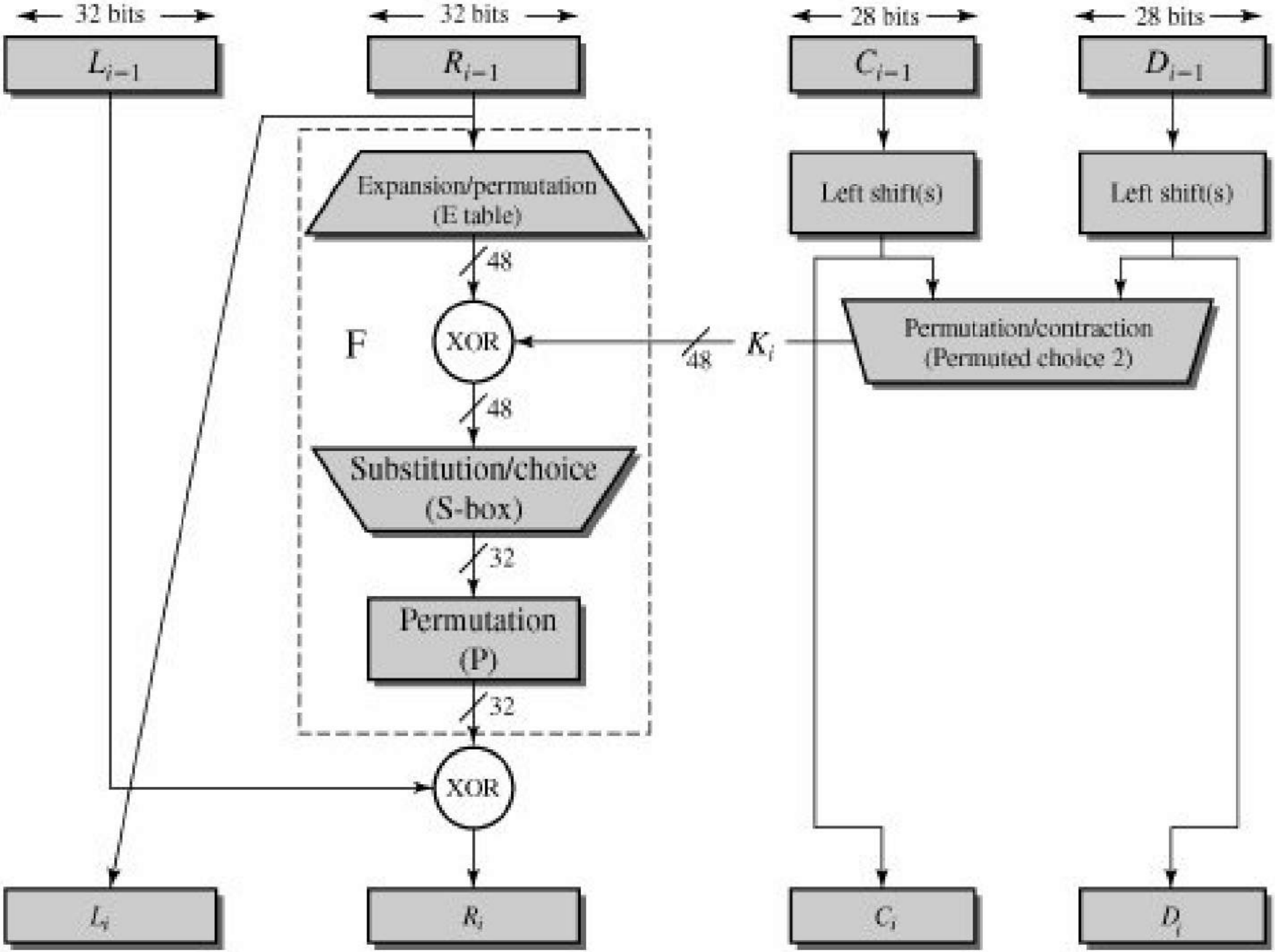
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

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

Details of Single Round

- The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).
- The overall processing at each round can be summarized in the following formulas:
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus \mathrm{F}(R_{i-1}, K_i)$


32 bits
32 bits
28 bits
28 bits
L_{i-1}
R_{i-1}
C_{i-1}
D_{i-1}
Expansion/permutation
(E table)
Left shift(s)
Left shift(s)
48
F
XOR
48
K_i
Permutation/contraction
(Permuted choice 2)
48
Substitution/choice
(S-box)
32
Permutation
(P)
32
XOR
L_i
R_i
C_i
D_i

- The round key K_i is 48 bits.
- The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits

(c) Expansion Permutation (E)

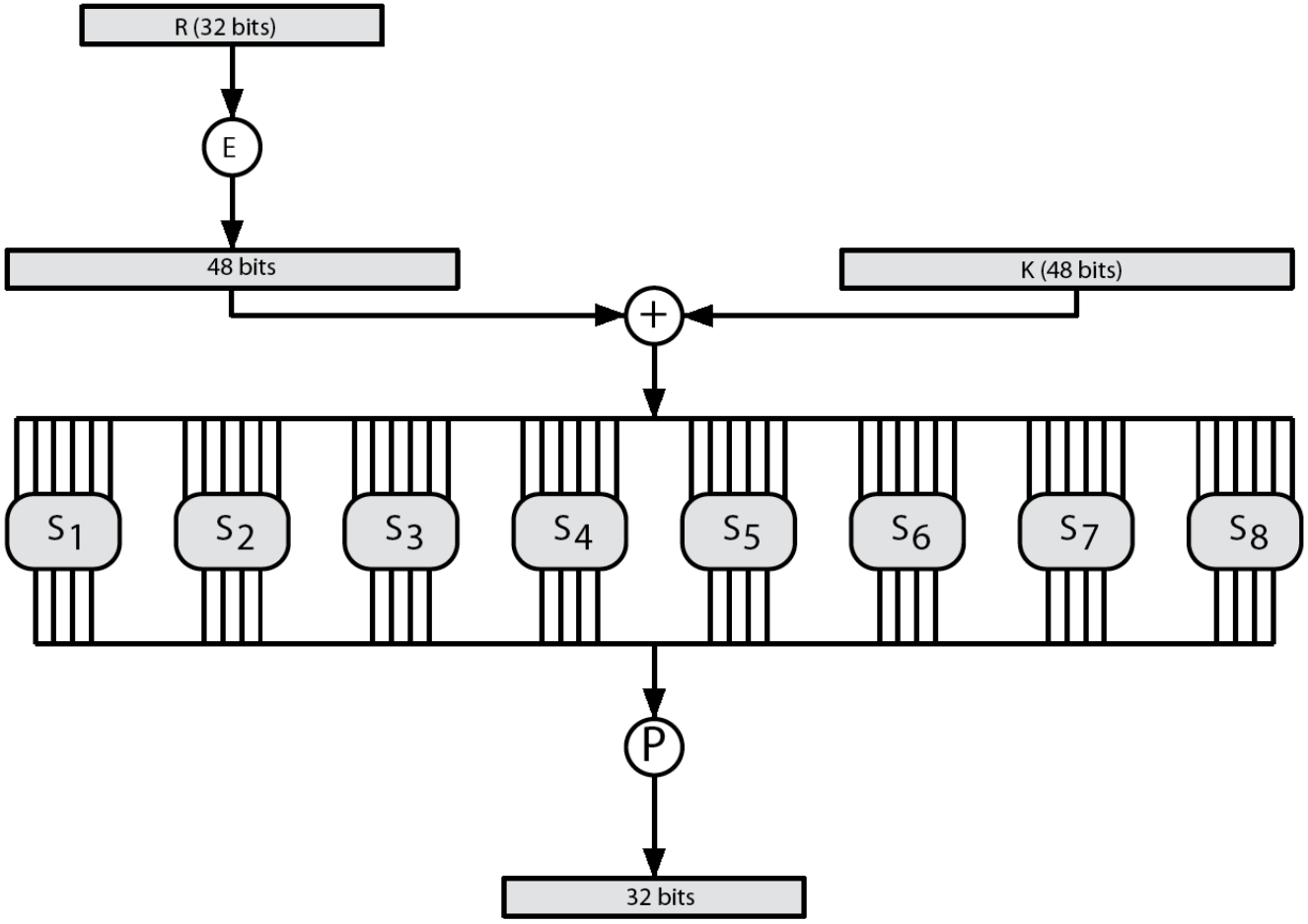
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- The resulting 48 bits are XORed with K_i.
- This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table

(d) Permutation Function (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

DES Round Structure (The role of the S-boxes in the function F)



Substitution Boxes S

- The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.
- The first and last bits of the input to box form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si .
- The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example, in S1, for input 011000, the row is 00 (row 0) and the column is 1100 (column 12).The value in row 0, column 12 is 12, so the output is 1100.

Table 3.3 Definition of DES S-Boxes

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

KEY GENERATION

- A 64-bit key is used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored

(a) Input Key

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

- The key is first subjected to a permutation governed by a table labeled Permuted Choice One.

(b) Permuted Choice One (PC-1)

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

- The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0 .
- At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by the Table

(d) Schedule of Left Shifts

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

- These shifted values serve as input to the next round.
- They also serve as input to the part labeled Permuted Choice Two which produces a 48-bit output that serves as input to the function $F(R_{i-1}, K_i)$

(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

- Plaintext: 0123456789ABCDEF
- Key: 133457799BBCDFF1
- Ciphertext:

- PC1---1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111
- C_0 = 1111000 0110011 0010101 0101111
  D_0 = 0101010 1011001 1001111 0001111
- C_1 = 1110000110011001010101011111
  D_1 = 1010101011001100111100011110
- PC2
- K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

- C_2 = 11000011001100101010101111111
  D_2 = 01010101100110011111000111101
- C_3 = 00001100110010101010111111111
  D_3 = 01010110011001111100011110101
- C_4 = 00110011001010101011111111100
  D_4 = 01011001100111100011111010101
- C_5 = 11001100101010101111111110000
  D_5 = 01100110011110001111010101010
- C_6 = 00110010101010111111111000011
  D_6 = 10011001111000111101010101010

- C_7 = 11001010101011111111100001100
  D_7 = 01100111100011110101010110
- C_8 = 00101010101111111110000110011
  D_8 = 10011110001111010101011001
- C_9 = 01010101011111111100001100110
  D_9 = 00111100011110101010110011
- C_{10} = 01010101111111100001100110011001
  D_{10} = 11110001111010101011001100
- C_{11} = 01010111111110000110011001100101
  D_{11} = 11000111101010101011001100110011

- C_{12} = 010111111110000110011001010 1
  D_{12} = 000111101010101011001100 1111
- C_{13} = 011111111000011001100101010 1
  D_{13} = 011110101010101100110011110 0
- C_{14} = 111111100001100110010101010 1
  D_{14} = 111010101010110011001111000 1
- C_{15} = 111110000110011001010101011 1
  D_{15} = 101010101011001100111100011 1
- C_{16} = 111100001100110010101010111 1
  D_{16} = 010101010110011001111000111 1

- K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
  K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
  K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
  K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
  K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
  K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
  K_8 = 111101 111000 101000 111010 110000 010011 101111 111011

- K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
  K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
  K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
  K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
  K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
  K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
  K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
  K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

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

- L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
  R_0 = 1111 0000 1010 1010 1111 0000 1010 1010
- Expansion
- E(R_0)=011110 100001 010101 010101 011110 100001 010101 010101
- XOR of K1 And E(R_0)
- 011000 010001 011110 111010 100001 100110 010100 100111.

- S Box
- 0101 1100 1000 0010 1011 0101 1001 0111
- Permutation P
- 0010 0011 0100 1010 1010 1001 1011 1011
- $R_1 = L_0 + f(R_0 , K_1)$
- = 1110 1111 0100 1010 0110 0101 0100 0100
- $L_1 = R_0$

- L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100
  R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101
- We reverse the order of these two blocks and apply the final permutation to
- $R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100
- IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101
- which in hexadecimal format is
- 85E813540F0AB405.

Plain text: (hexadecimal) 0 1 2 3 4 5 6 7 8 9 A B C D E F

Key: (hexadecimal) 0 1 2 3 4 5 6 7 8 9 A B C D E F

DES Decryption

- Decryption uses the same algorithm as encryption.
- Use the ciphertext as the input to the DES algorithm but use the keys Ki in reverse order.
- That is, use K16 on the first iteration, K15 on the second until K1 which is used on the 16th and last iteration.

Avalanche Effect

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.
- This is referred to as the avalanche effect.
- If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.

Strength of DES

Key Size

- With a key length of 56 bits we have $2^{56} = 7.2 \times 10^{16}$ values.
- Brute force attack is impractical.

Timing Attacks

- A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various cipher texts.
- A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.

Cryptanalysis of DES

- Differential cryptanalysis
- Linear cryptanalysis.

Differential Cryptanalysis

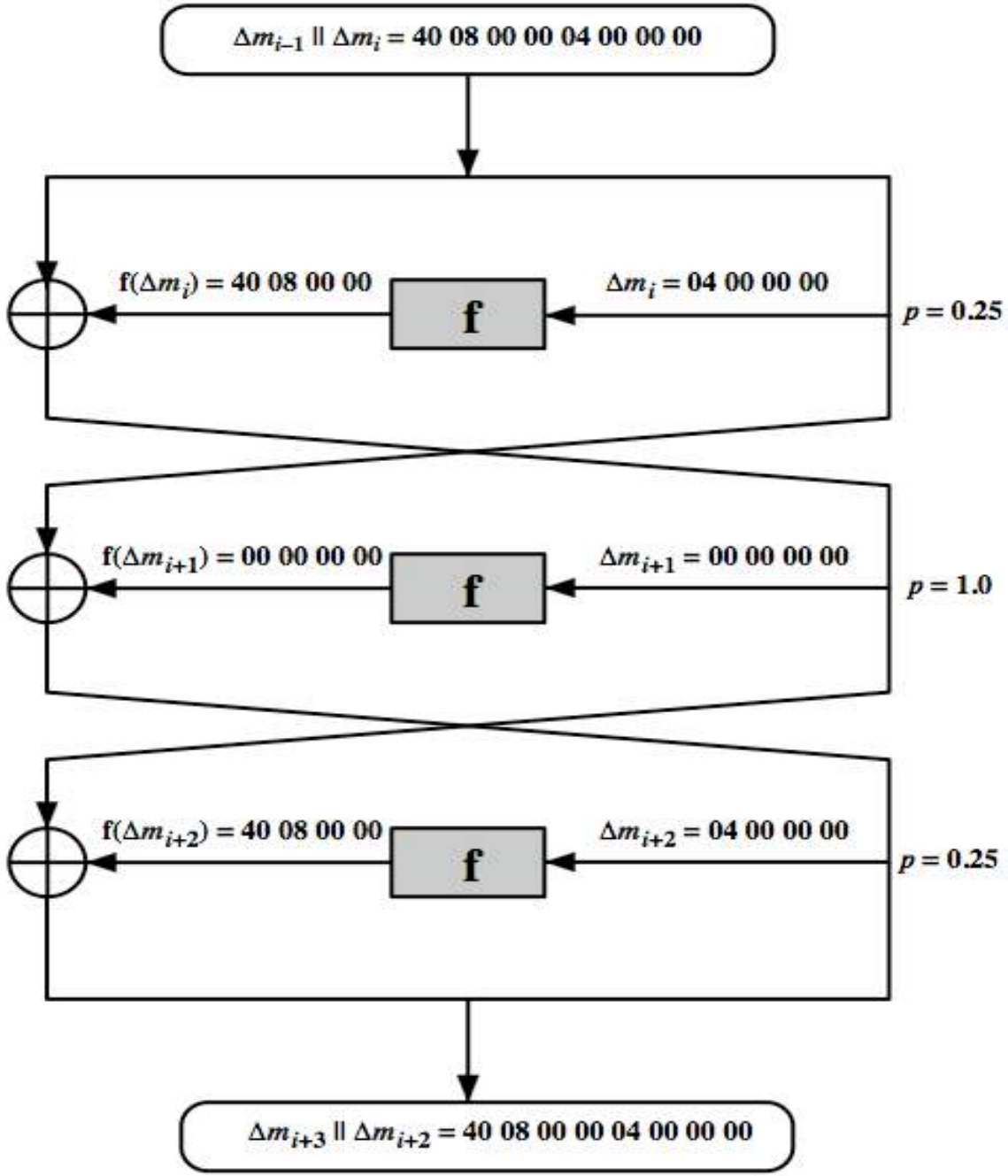
- Consider the original plaintext block m to consist of two halves m_0,m_1 .
- Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round.
- So, at each round, only one new 32-bit block is created. Each new block m_i, then the intermediate message halves are related as

$$m_{i+1} = m_{i-1} \oplus \mathrm{f}(m_i, K_i), \qquad i = 1, 2, \ldots, 16$$

- In differential cryptanalysis, we start with two messages, m and m’ , with a known XOR difference $\Delta m = m \oplus m'$
- The difference between the intermediate message halves: $\Delta m_i = m_i \oplus m_i'$

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus \mathrm{f}(m_i, K_i)] \oplus [m'_{i-1} \oplus \mathrm{f}(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [\mathrm{f}(m_i, K_i) \oplus \mathrm{f}(m'_i, K_i)]\end{aligned}$$

Differential Propagation through Three Rounds of DES


$\Delta m_{i-1} \| \Delta m_i$ = 40 08 00 00 04 00 00 00
f(Δm_i) = 40 08 00 00
f
Δm_i = 04 00 00 00
p = 0.25
f(Δm_{i+1}) = 00 00 00 00
f
Δm_{i+1} = 00 00 00 00
p = 1.0
f(Δm_{i+2}) = 40 08 00 00
f
Δm_{i+2} = 04 00 00 00
p = 0.25
$\Delta m_{i+3} \| \Delta m_{i+2}$ = 40 08 00 00 04 00 00 00

Linear Cryptanalysis

- For a cipher with n-bit plaintext and ciphertext blocks and an m -bit key, let the plaintext block be labeled P[0],……,P[n].
- Cipher text, C[0],.....C[n]
- Key K[0],….K[m]

- The objective of linear cryptanalysis is to find an effective *linear equation of* the form:

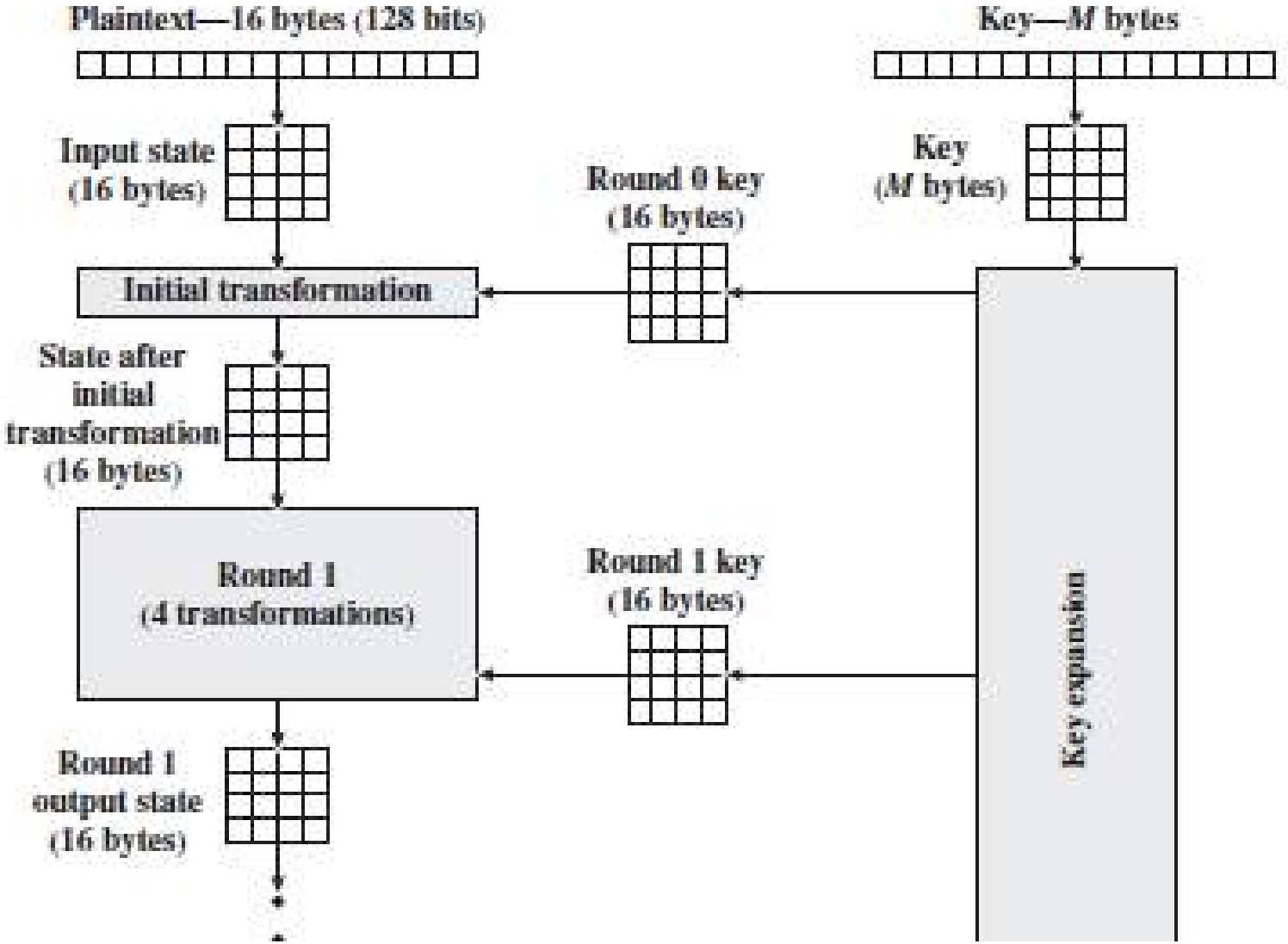

P[α1, α2, ..., αa] ⊕ C[β1, β2, ..., βb] = K[γ1, γ2, ..., γc]

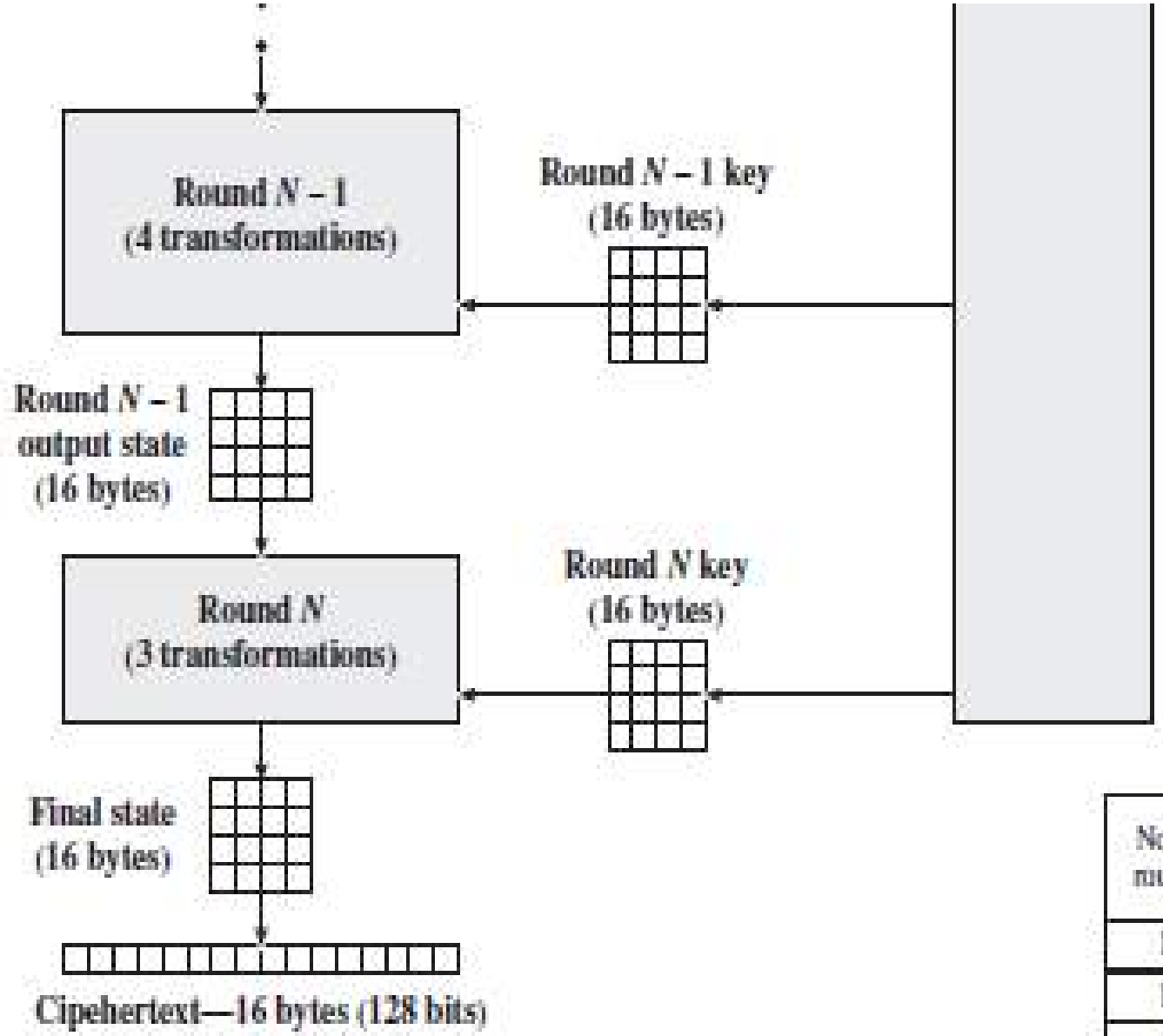
- The further is from p=0.5, the more effective the equation.
- Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext–ciphertext pairs.

Advanced Encryption Standard(AES)

- AES is symmetric block cipher intended to replace DES for commercial applications.
- It uses a 128-bit block size and a key size of 128, 192, or 256 bits.
- AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key.

AES STRUCTURE



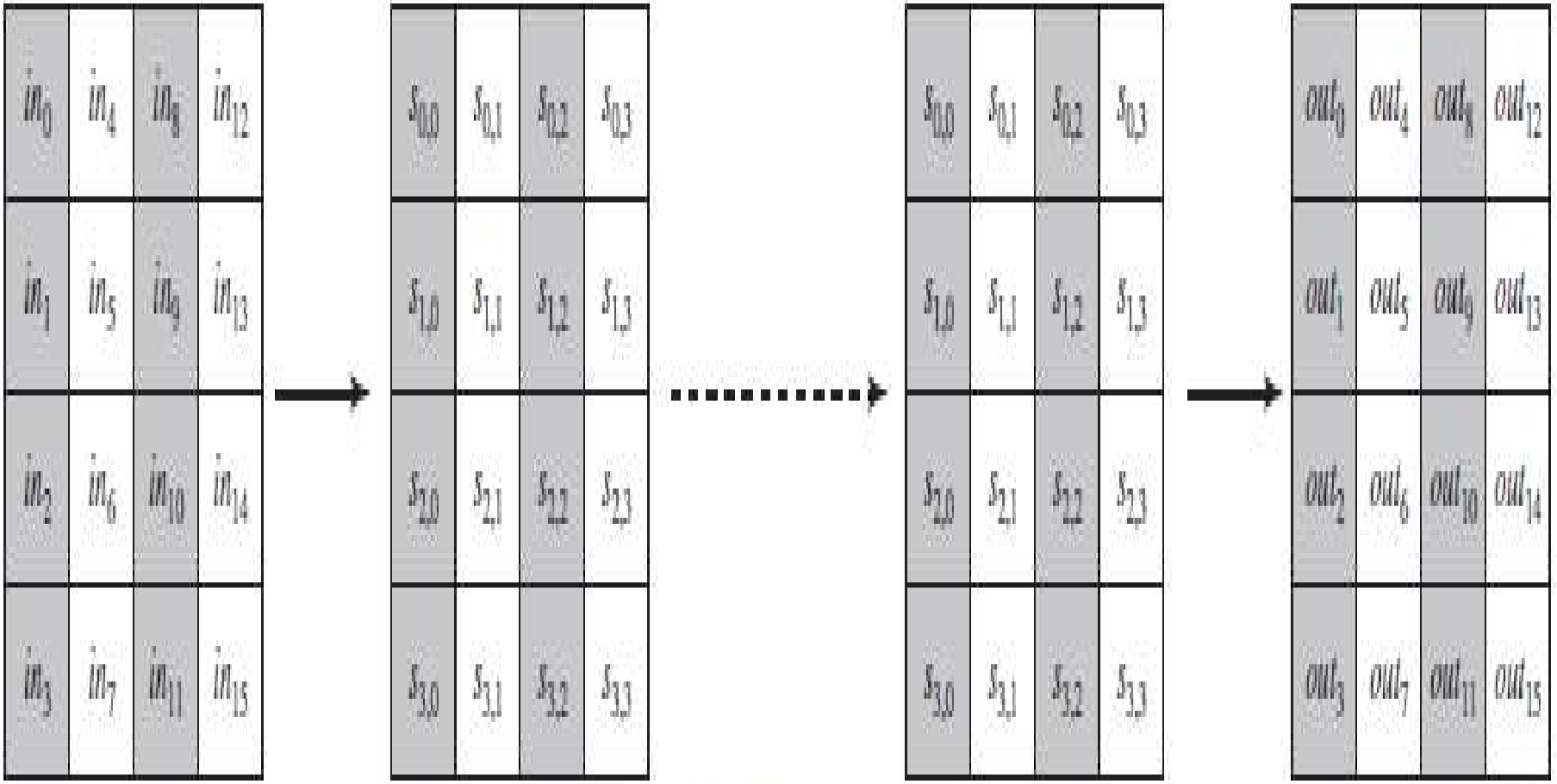


No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 5.1 AES Encryption Process

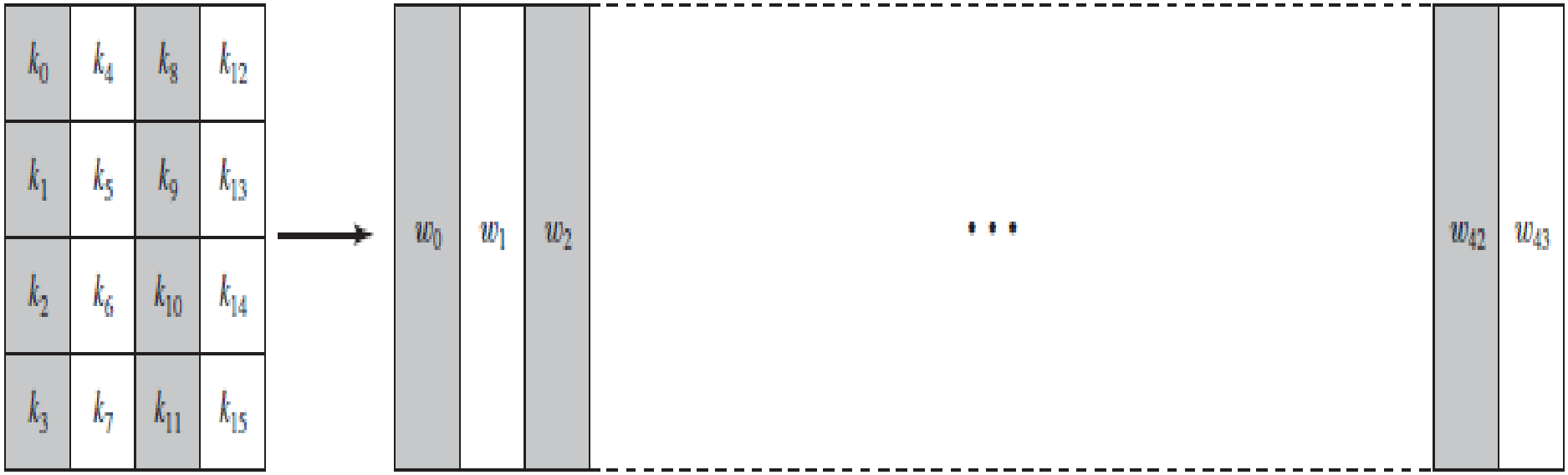
- The cipher takes a plaintext block size of 128 bits, or 16 bytes.
- The key length can be 16, 24, or 32bytes (128, 192, or 256 bits).
- The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.
- The input to the encryption and decryption algorithms is a single 128-bit block.

- This block is copied into the **State array, which is modified at each stage of encryption or decryption.**
- After the final stage, **State is copied to an output matrix.**



(a) Input, state array, and output

- The key is depicted as a square matrix of bytes.
- This key is then expanded into an array of key schedule words.



(b) Key and expanded key

- The first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the **in matrix, the second four bytes occupy the** second column, and so on.
- Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the **w matrix.**
- The cipher consists of N rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key.

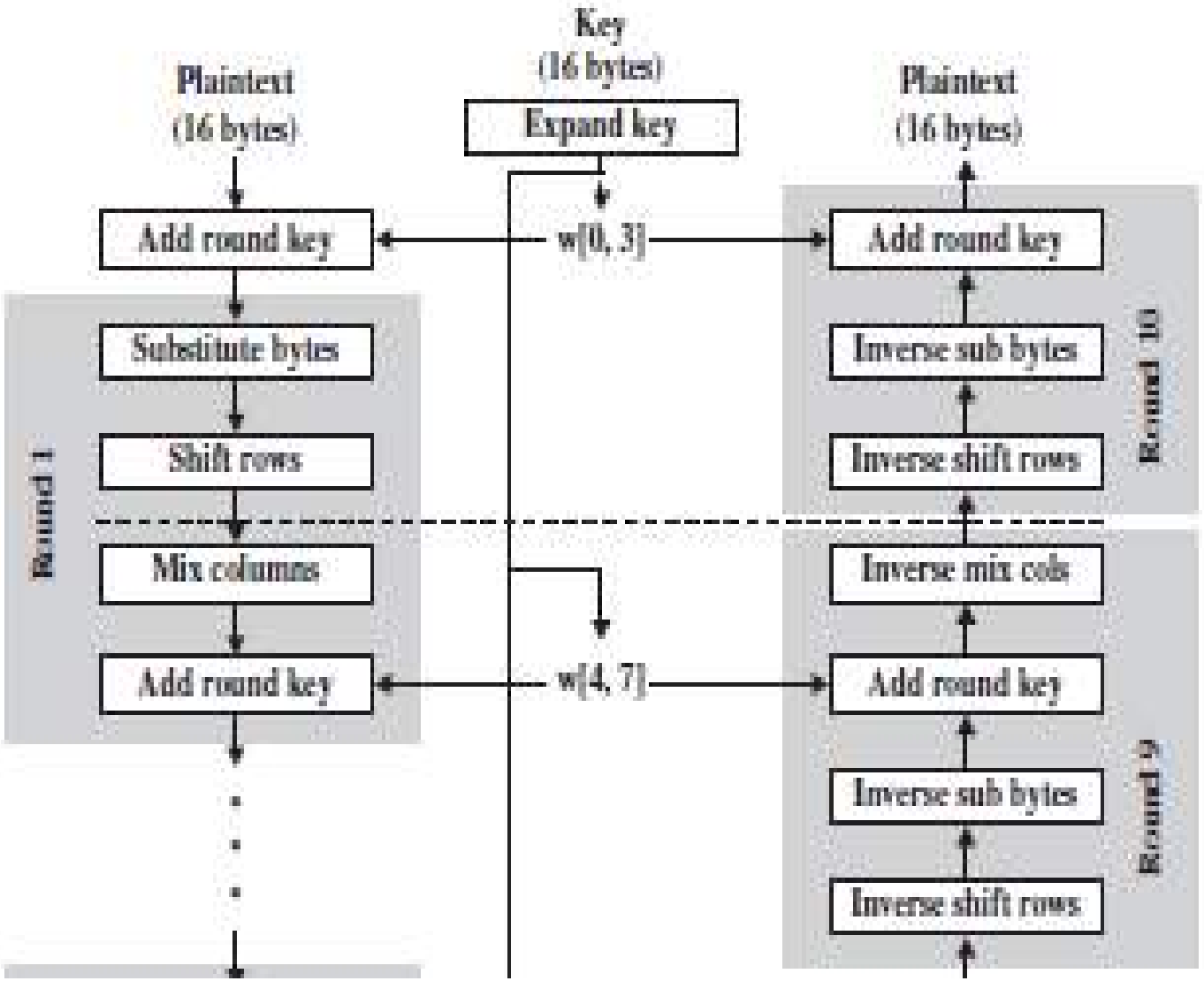
- The first N-1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey.
- The final round contains only three transformations, and there is a initial single transformation (AddRoundKey) before the first round, which can be considered Round 0.
- Each transformation takes one or more

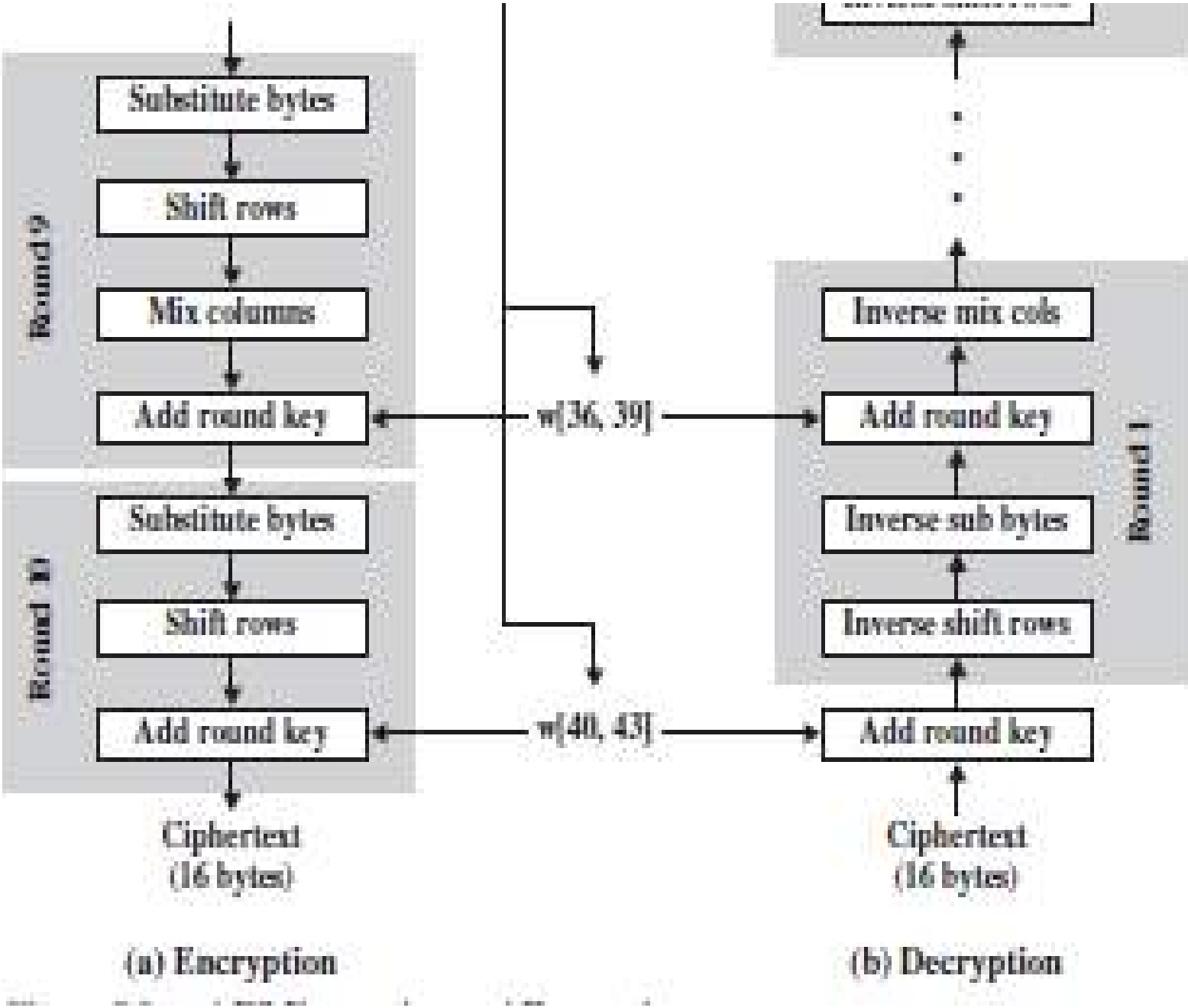
  4 x 4 matrices as input and produces a

  4 x 4 matrix as output.

- The key expansion function generates N + 1 round keys, each of which is a distinct 4 x 4 matrix.
- Each round key serve as one of the inputs to the AddRoundKey transformation in each round.

Detailed Structure





(a) Encryption

(b) Decryption

1. THE AES structure is not a Feistel structure.In classic Feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped.AES instead processes the entire data block as a single matrix during each round using substitutions and permutation.
2. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round.

3. Four different stages are used, one of permutation and three of substitution:

- **Substitute bytes**: Uses an S-box to perform a byte-by-byte substitution of the block
- **ShiftRows:** A simple permutation
- **MixColumns:** A substitution that makes use of arithmetic over GF(2^8)
- **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key

4. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.
5. Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.

6. The cipher consists of alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on.

   This scheme is both efficient and highly secure.

AES TRANSFORMATION FUNCTIONS

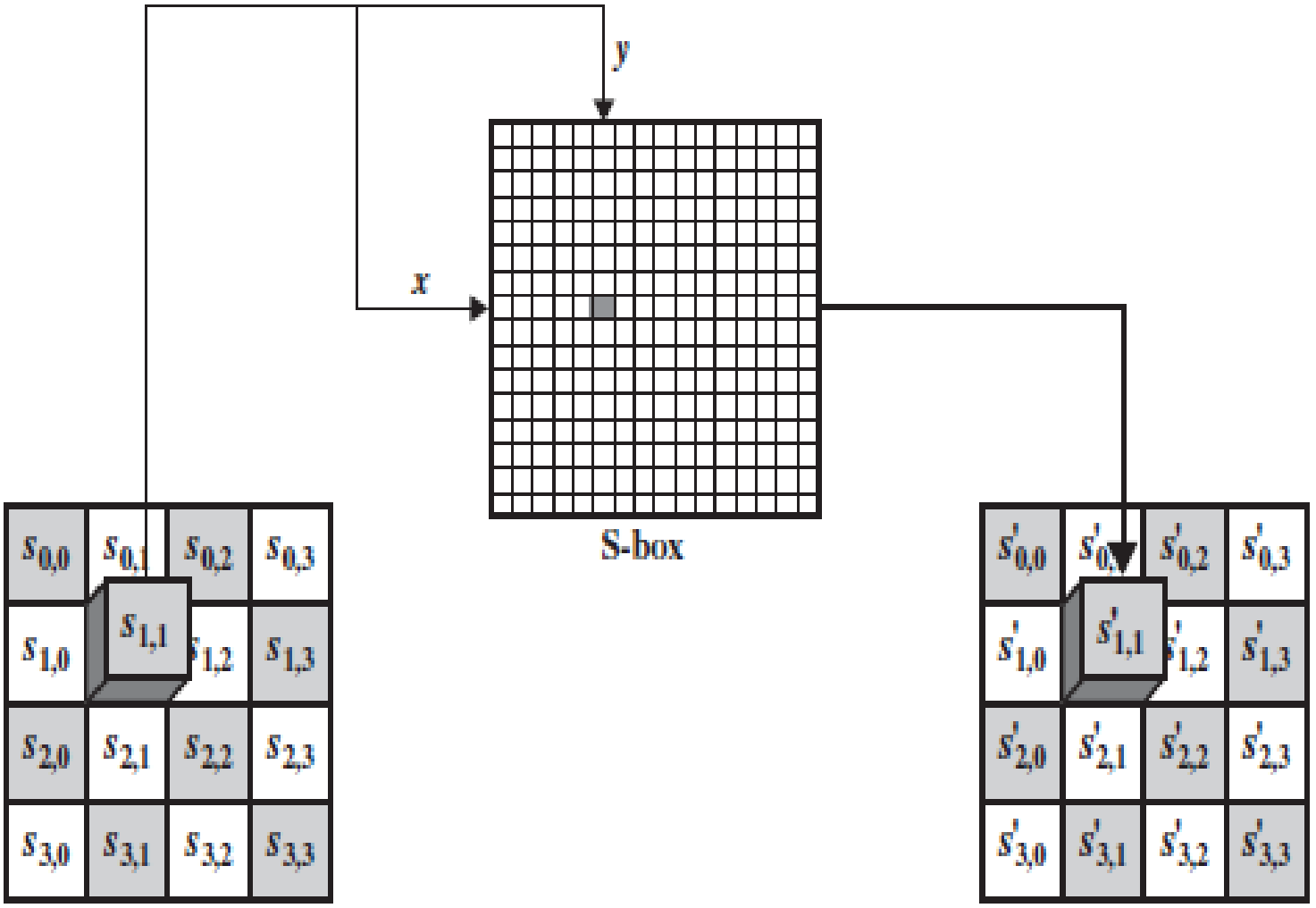
- Four transformations are used in AES.

1. **Substitute Bytes Transformation**
2. **ShiftRows Transformation**
3. **MixColumns Transformation**
4. **AddRoundKey Transformation**

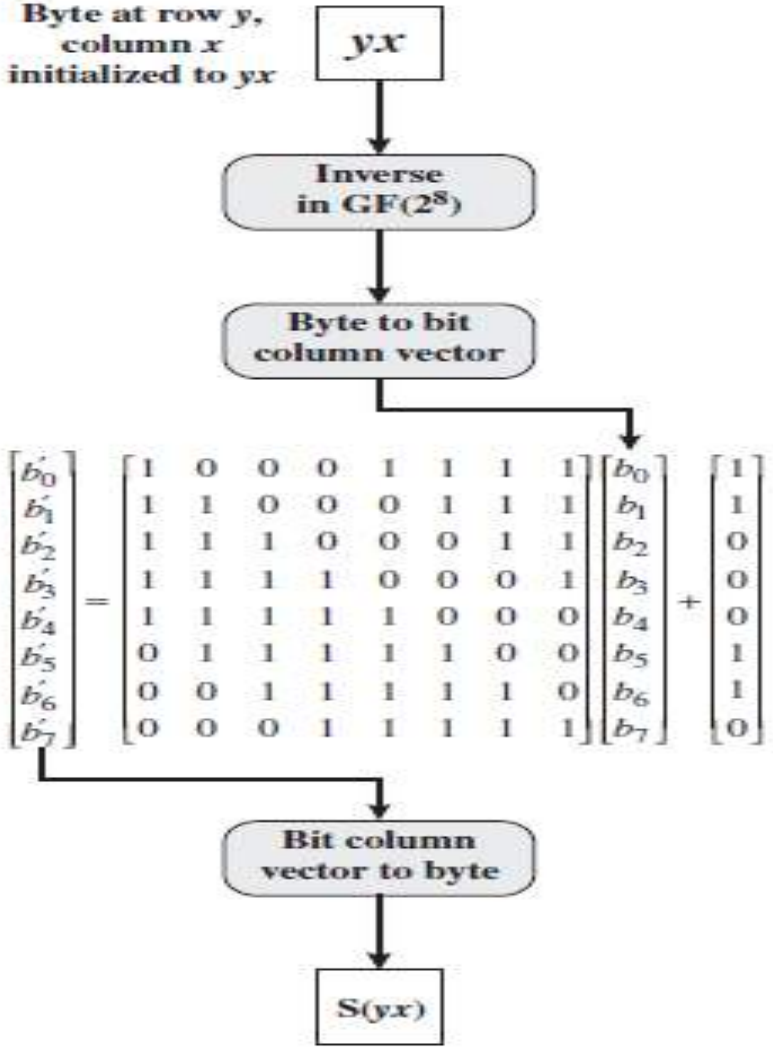
Substitute Bytes Transformation

FORWARD AND INVERSE TRANSFORMATIONS

- The forward substitute byte transformation, called SubBytes, is a simple table lookup.
- AES defines a 16 x 16 matrix of byte values, called an S-box , that contains a permutation of all possible 256 8-bit values.
- Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value.
- These row and column values serve as indexes into the S-box to select a unique 8-bit output value.

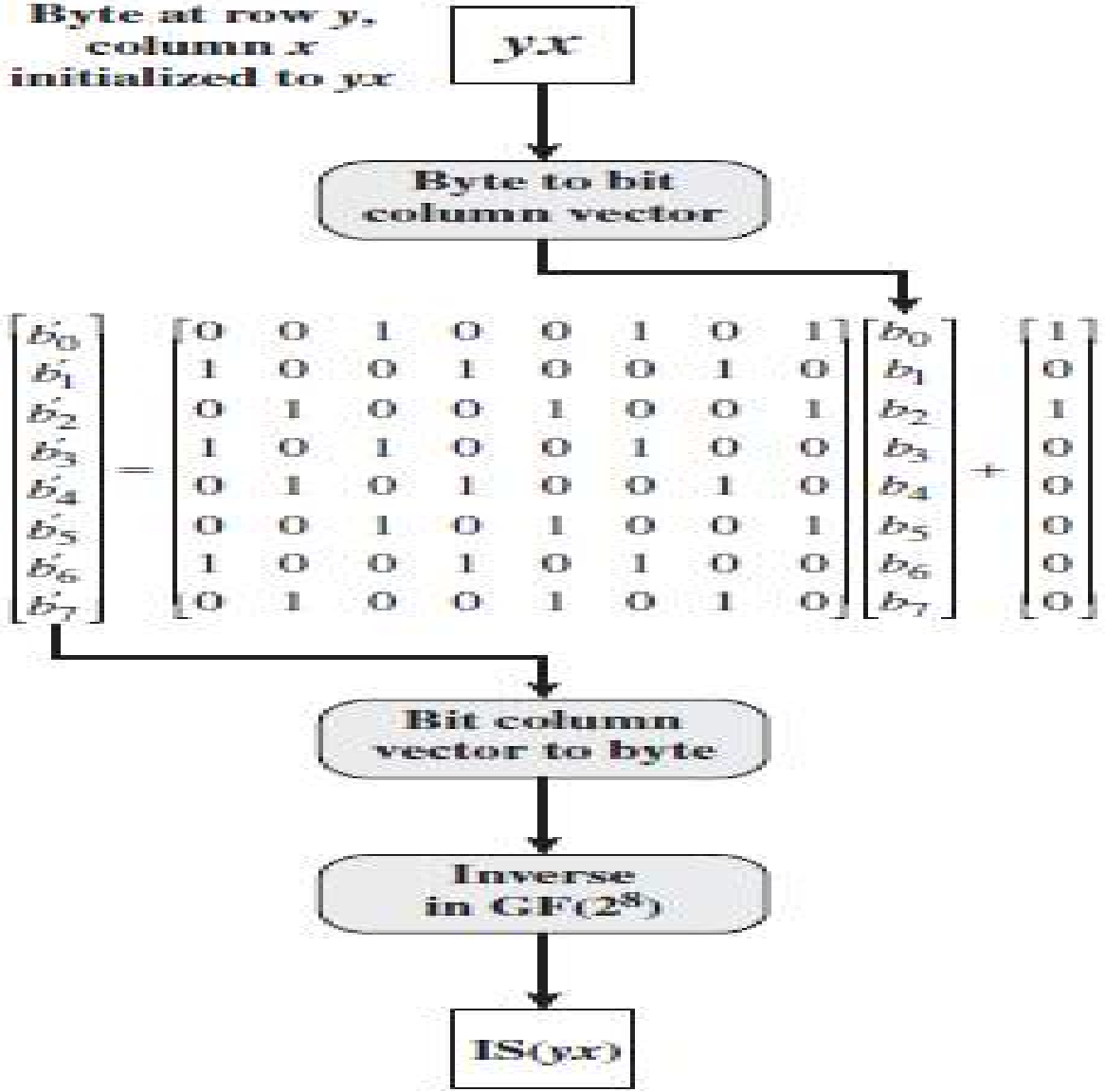

y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Construction of S box



$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

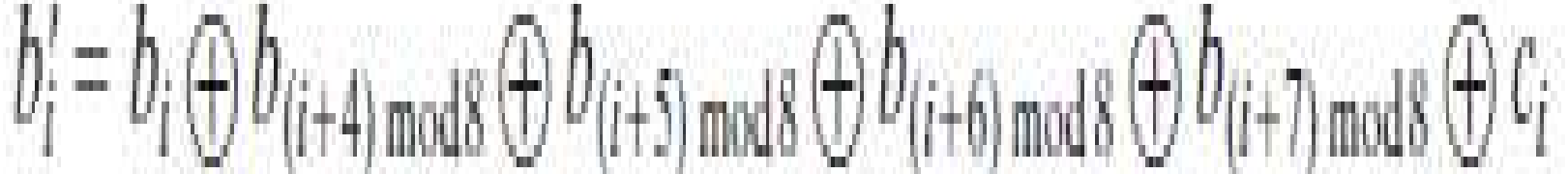
(a) Calculation of byte at row y, column x of S-box



(a) Calculation of byte at row y, column x of IS-box

1. Initialize the S-box with the byte values in ascending sequence row by row. The first row contains {00},{01},{02},….{0F}; the second row contains {10},{11},{12},….{1F} etc.; and so on. Thus, the value of the byte at row y , column x is {yx}.
2. Map each byte in the S-box to its multiplicative inverse in the finite field GF (2^8) ;the value {00} is mapped to itself.

3. Consider that each byte in the S-box consists of 8 bits labeled (b_7, b_6, b_5,….b_0). Apply the following transformation to each bit of each byte in the S-box:


$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$

- c_i is the *i^{th} bit of c byte with the value [63] ; that is, (c7 c6 c5 c4 c3 c2 c1 c0) = 01100011*
- .

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- In ordinary matrix multiplication, each element in the product matrix is the sum of products of the elements of one row and one column.
- In this case, each element in the product matrix is the bitwise XOR of products of elements of one row and one column.
- The final addition is a bitwise XOR. The bitwise XOR is addition in $GF(2^8)$.

As an example, consider the input value {95}. The multiplicative inverse in $GF(2^8)$ is $\{95\}^{-1} = \{8A\}$, which is 10001010 in binary. Using Equation (5.2),

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The result is {2A}, which should appear in row {09} column {05} of the *S*-box.

- The **inverse substitute byte transformation, called InvSubBytes, makes use** of the inverse S-box.
- The inverse transformation is

$$b'_i = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

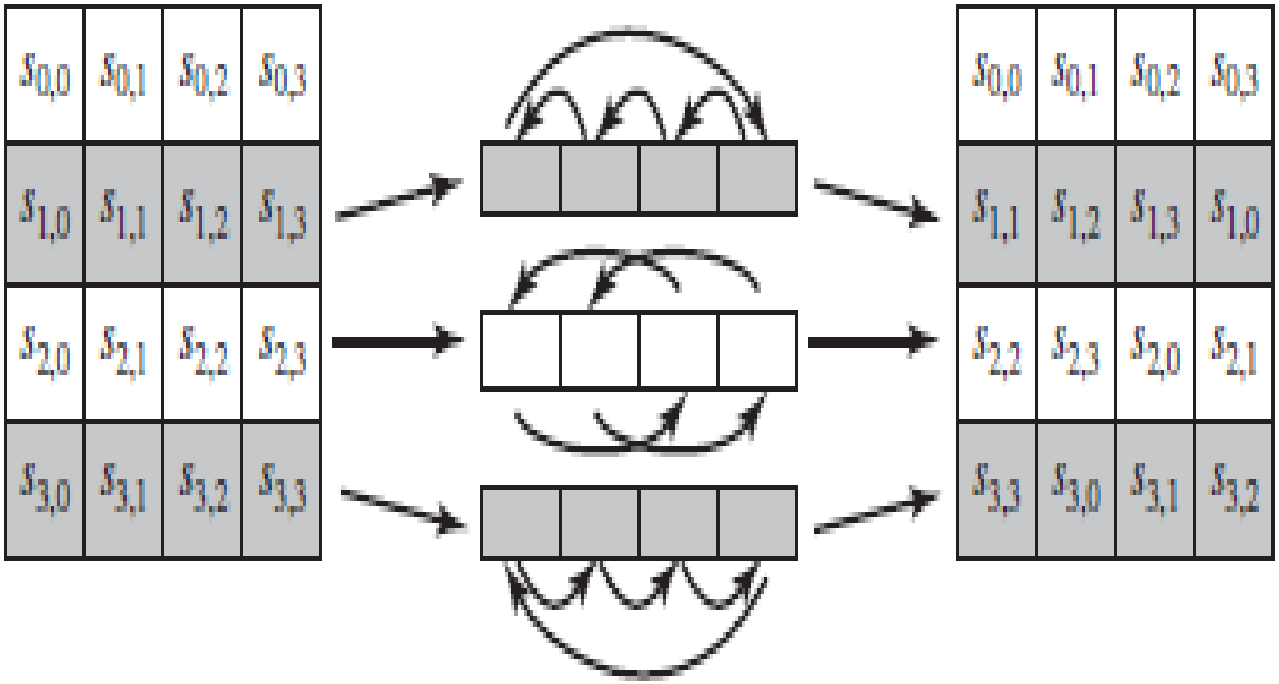
- where byte , d={05} or 00000101.

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

ShiftRows Transformation

FORWARD AND INVERSE TRANSFORMATIONS

- ***The forward shift row transformation,*** called ShiftRows.
- The first row of **State is not altered. For** the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.



(a) Shift row transformation

Example

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

- The inverse shift row transformation, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.

MixColumns Transformation

FORWARD AND INVERSE TRANSFORMATIONS

- The forward mix column transformation, called MixColumns, operates on each column individually.
- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.
- The transformation can be defined by the following matrix multiplication on **State.**

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- Each element in the product matrix is the sum of products of elements of one row and one column.
- In this case, the individual additions and multiplications are performed in GF (2^8).
- The MixColumns transformation on a single column of **State can be** expressed as

$$s'_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$

$$s'_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j}$$

$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$

$$s'_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

- In particular, multiplication of a value by {02}) can be implemented as a 1-bit left shift followed by a conditional bitwise XOR with (0001 1011) if the leftmost bit of the original value (prior to the shift) is 1.
- If the left most bit is 0, perform 1 bit left shift only.

example

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$(\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \qquad \oplus \{A6\} \qquad = \{47\}$$

For the first equation, we have $\{02\} \cdot \{87\} = (0000\,1110) \oplus (0001\,1011) = (0001\,0101)$ and $\{03\} \cdot \{6E\} = \{6E\} \oplus (\{02\} \cdot \{6E\}) = (0110\,1110) \oplus (1101\,1100) = (1011\,0010)$. Then,

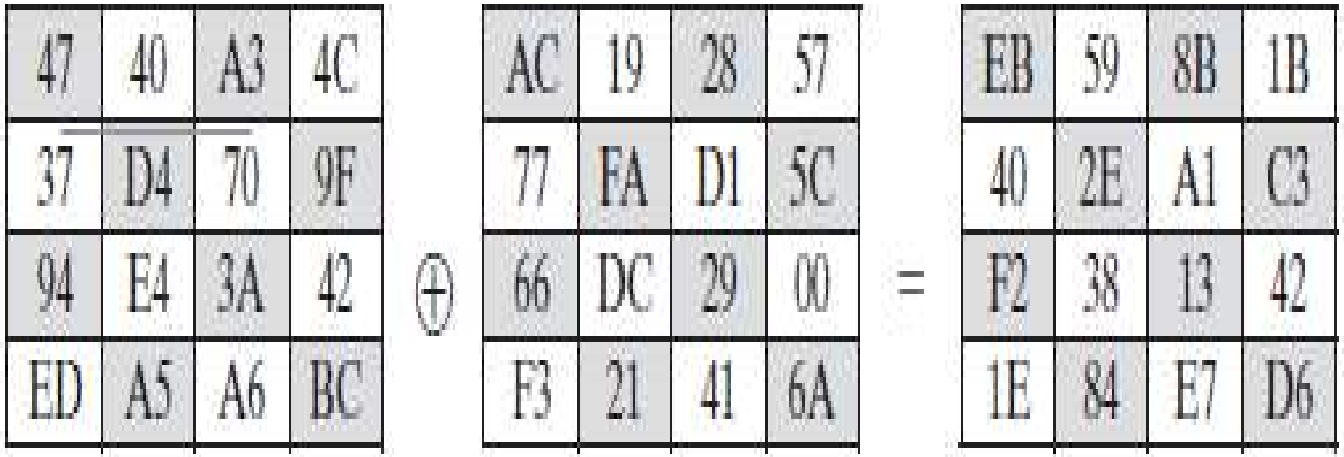
$$
\begin{aligned}
\{02\} \cdot \{87\} &= 0001\ 0101 \\
\{03\} \cdot \{6E\} &= 1011\ 0010 \\
\{46\} &= 0100\ 0110 \\
\{A6\} &= \underline{1010\ 0110} \\
& \quad 0100\ 0111 = \{47\}
\end{aligned}
$$

AddRoundKey Transformation

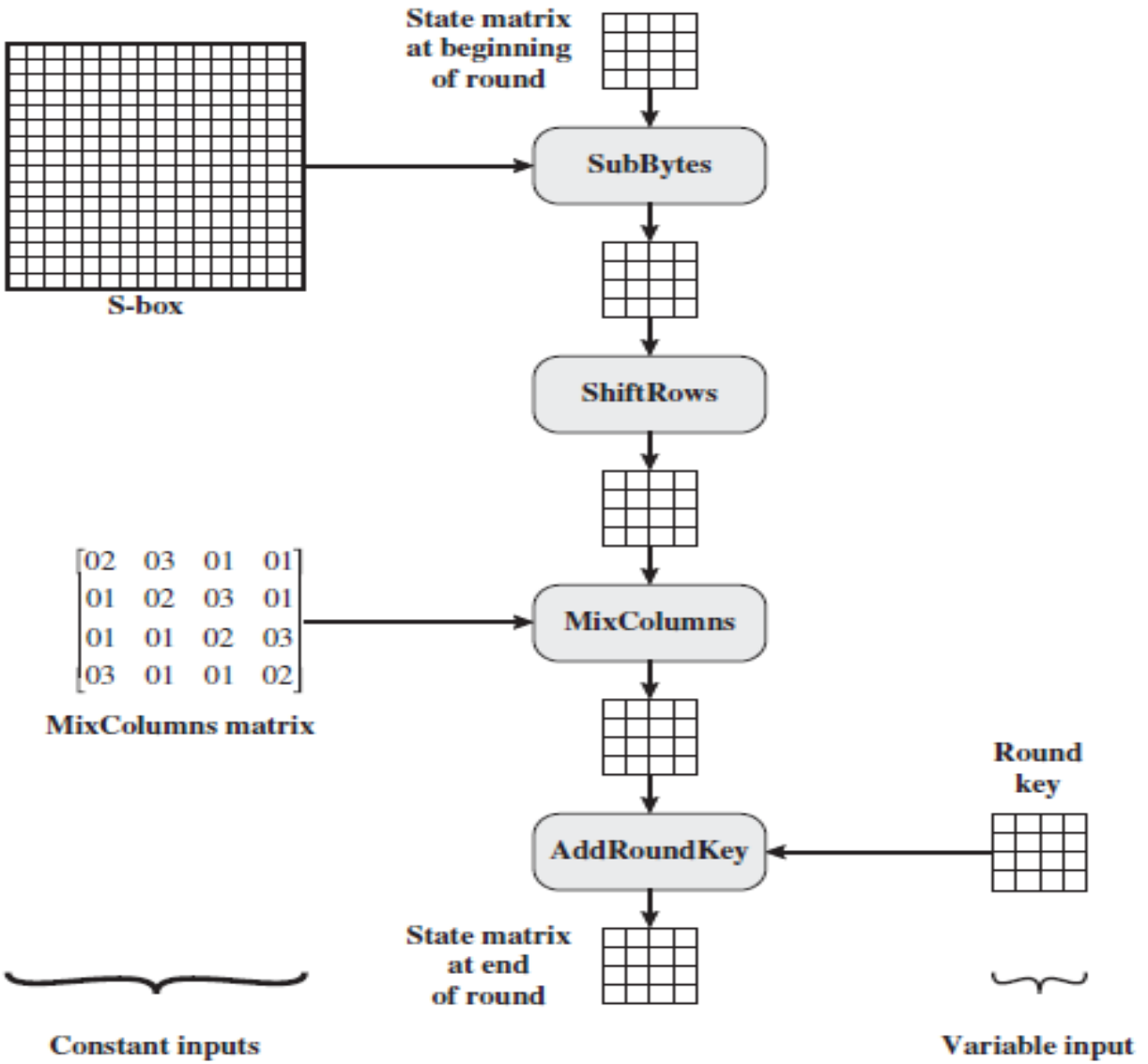
FORWARD AND INVERSE TRANSFORMATIONS

- 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.
- The operation is viewed as a column wise operation between the 4 bytes of a State column and one word of the round key.

Example

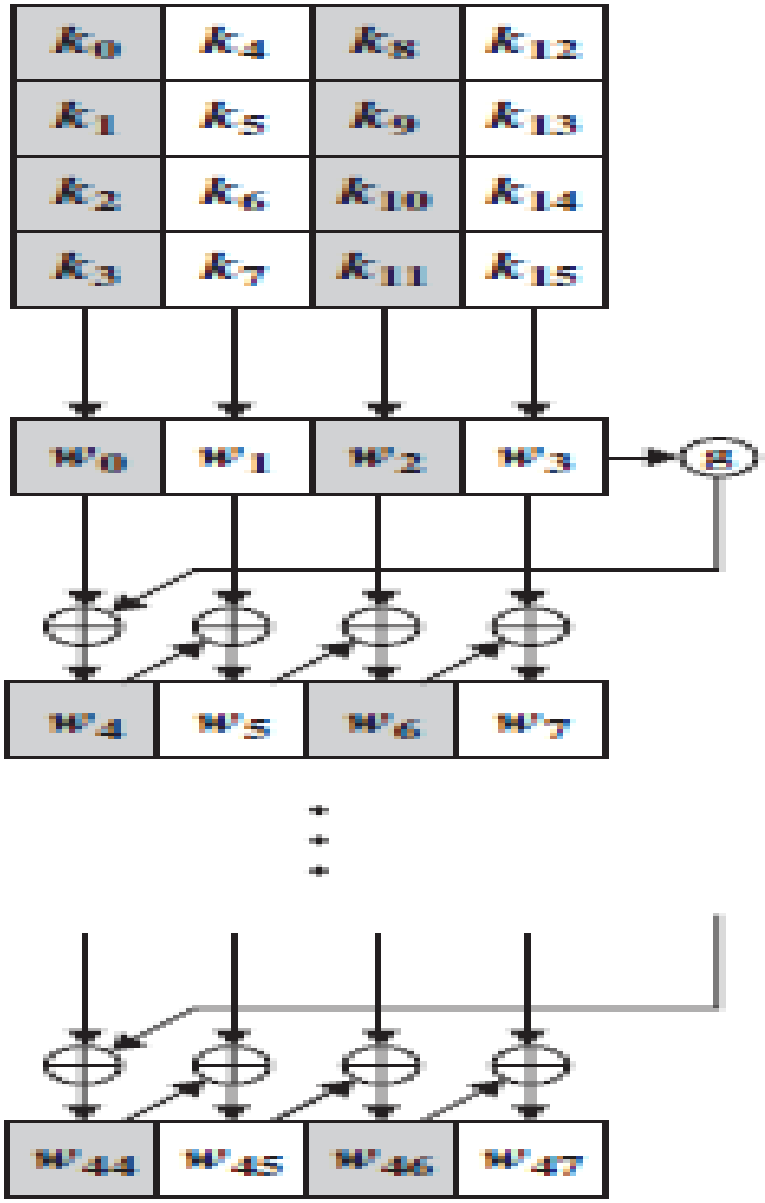

47 40 A3 4C
37 D4 70 9F
94 E4 3A 42
ED A5 A6 BC
⊕
AC 19 28 57
77 FA D1 5C
66 DC 29 00
F3 21 41 6A
=
EB 59 8B 1B
40 2E A1 C3
F2 38 13 42
1E 84 E7 D6

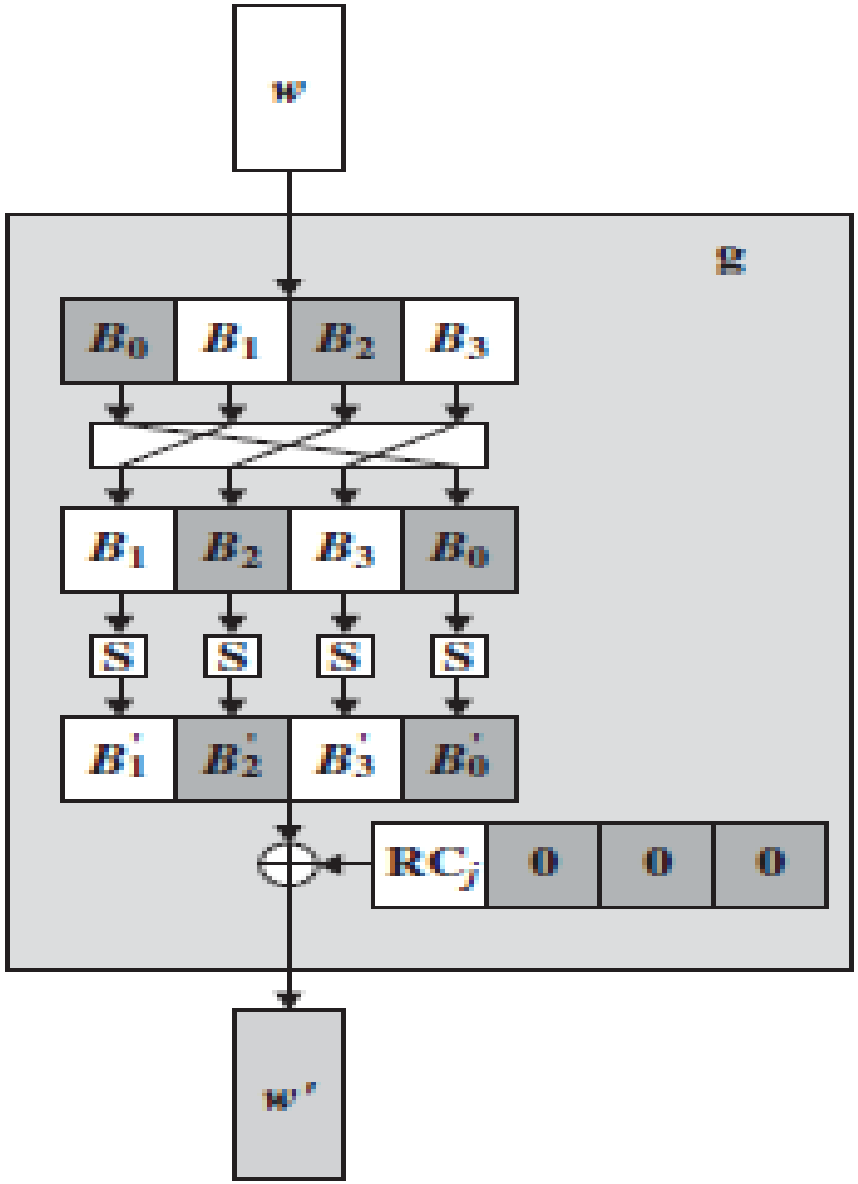
Single AES round



AES KEY EXPANSION

- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).
- The key is copied into the first four words of the expanded key.
- The remainder of the expanded key is filled in four words at a time.


K_0 K_4 K_8 K_{12}
K_1 K_5 K_9 K_{13}
K_2 K_6 K_{10} K_{14}
K_3 K_7 K_{11} K_{15}
W_0 W_1 W_2 W_3
g
W_4 W_5 W_6 W_7
W_{44} W_{45} W_{46} W_{47}


w
g
B_0
B_1
B_2
B_3
B_1
B_2
B_3
B_0
S
S
S
S
B_1'
B_2'
B_3'
B_0'
RC_j
0
0
0
w'

- Rcon[j] = (RC[j], 0, 0, 0)

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

- Plaintext:
  0123456789abcdeffedcba9876543210
- Key:
  0f1571c947d9e8590cb7add6af7f6798

Key Words	Auxiliary Function
w0 = 0f 15 71 c9 w1 = 47 d9 e8 59 w2 = 0c b7 ad d6 w3 = af 7f 67 98	RotWord (w3) = 7f 67 98 af = x1 SubWord (x1) = d2 85 46 79 = y1 Rcon (1) = 01 00 00 00 y1 $\oplus$ Rcon (1) = d3 85 46 79 = z1

w4 = w0 ⊕ z1 = dc 90 37 b0	RotWord(w7) = 81 15 a7 38 = x2
w5 = w4 ⊕ w1 = 9b 49 df e9	SubWord(x4) = 0c 59 5c 07 = y2
w6 = w5 ⊕ w2 = 97 fe 72 3f	Rcon(2) = 02 00 00 00
w7 = w6 ⊕ w3 = 38 81 15 a7	y2 ⊕ Rcon(2) = 0e 59 5c 07 = z2
w8 = w4 ⊕ z2 = d2 c9 6b b7	RotWord(w11) = ff d3 c6 e6 = x3
w9 = w8 ⊕ w5 = 49 80 b4 5e	SubWord(x2) = 16 66 b4 83 = y3
w10 = w9 ⊕ w6 = de 7e c6 61	Rcon(3) = 04 00 00 00
w11 = w10 ⊕ w7 = e6 ff d3 c6	y3 ⊕ Rcon(3) = 12 66 b4 8e = z3
w12 = w8 ⊕ z3 = c0 af df 39	RotWord(w15) = ae 7e c0 b1 = x4
w13 = w12 ⊕ w9 = 89 2f 6b 67	SubWord(x3) = e4 f3 ba c8 = y4
w14 = w13 ⊕ w10 = 57 51 ad 06	Rcon(4) = 08 00 00 00
w15 = w14 ⊕ w11 = b1 ae 7e c0	y4 ⊕ Rcon(4) = ec f3 ba c8 = 4

w16 = w12 ⊕ z4 = 2c 5c 65 f1 w17 = w16 ⊕ w13 = a5 73 0e 96 w18 = w17 ⊕ w14 = f2 22 a3 90 w19 = w18 ⊕ w15 = 43 8c dd 50	RotWord(w19) = 8c dd 50 43 = x5 SubWord(x4) = 64 c1 53 1a = y5 Rcon(5) = 10 00 00 00 y5 ⊕ Rcon(5) = 74 c1 53 1a = z5
w20 = w16 ⊕ z5 = 58 9d 36 eb w21 = w20 ⊕ w17 = fd ee 38 7d w22 = w21 ⊕ w18 = 0f cc 9b ed w23 = w22 ⊕ w19 = 4c 40 46 bd	RotWord (w23) = 40 46 bd 4c = x6 SubWord (x5) = 09 5a 7a 29 = y6 Rcon(6) = 20 00 00 00 y6 ⊕ Rcon(6) = 29 5a 7a 29 = z6
w24 = w20 ⊕ z6 = 71 c7 4c c2 w25 = w24 ⊕ w21 = 8c 29 74 bf w26 = w25 ⊕ w22 = 83 e5 ef 52 w27 = w26 ⊕ w23 = cf a5 a9 ef	RotWord (w27) = a5 a9 ef cf = x7 SubWord (x6) = 06 d3 bf 8a = y7 Rcon (7) = 40 00 00 00 y7 ⊕ Rcon(7) = 46 d3 df 8a = z7
w28 = w24 ⊕ z7 = 37 14 93 48 w29 = w28 ⊕ w25 = bb 3d e7 f7 w30 = w29 ⊕ w26 = 38 d8 08 a5 w31 = w30 ⊕ w27 = f7 7d a1 4a	RotWord (w31) = 7d a1 4a f7 = x8 SubWord (x7) = ff 32 d6 68 = y8 Rcon (8) = 80 00 00 00 y8 ⊕ Rcon(8) = 7f 32 d6 68 = z8

w32 = w28 ⊕ z8 = 48 26 45 20 w33 = w32 ⊕ w29 = f3 1b a2 d7 w34 = w33 ⊕ w30 = cb c3 aa 72 w35 = w34 ⊕ w32 = 3c be 0b 3	RotWord (w35) = be 0b 38 3c = x9 SubWord (x8) = ae 2b 07 eb = y9 Rcon (9) = 1B 00 00 00 y9 ⊕ Rcon (9) = b5 2b 07 eb = z9
w36 = w32 ⊕ z9 = fd 0d 42 cb w37 = w36 ⊕ w33 = 0e 16 e0 1c w38 = w37 ⊕ w34 = c5 d5 4a 6e w39 = w38 ⊕ w35 = f9 6b 41 56	RotWord (w39) = 6b 41 56 f9 = x10 SubWord (x9) = 7f 83 b1 99 = y10 Rcon (10) = 36 00 00 00 y10 ⊕ Rcon (10) = 49 83 b1 99 = z10
w40 = w36 ⊕ z10 = b4 8e f3 52 w41 = w40 ⊕ w37 = ba 98 13 4e w42 = w41 ⊕ w38 = 7f 4d 59 20 w43 = w42 ⊕ w39 = 86 26 18 76	

Table 5.2 AES S-Boxes

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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	C8	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

(a) S-box

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse S-box

Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76				0f 47 0c af
23 ab dc 54				15 d9 b7 7f
45 cd ba 32				71 e8 ad 67
67 ef 98 10				c9 59 d6 98
0e ce f2 d9	ab 8b 89 35	ab 8b 89 35	b9 94 57 75	dc 9b 97 38
36 72 6b 2b	05 40 7f f1	40 7f f1 05	e4 8e 16 51	90 49 fe 81
34 25 17 55	18 3f f0 fc	f0 fc 18 3f	47 20 9a 3f	37 df 72 15
ae b6 4e 88	e4 4e 2f c4	c4 e4 4e 2f	c5 d6 f5 3b	b0 e9 3f a7
65 0f c0 4d	4d 76 ba e3	4d 76 ba e3	8e 22 db 12	d2 49 de e6
74 c7 e8 d0	92 c6 9b 70	c6 9b 70 92	b2 f2 dc 92	c9 80 7e ff
70 ff e8 2a	51 16 9b e5	9b e5 51 16	df 80 f7 c1	6b b4 c6 d3
75 3f ca 9c	9d 75 74 de	de 9d 75 74	2d c5 1e 52	b7 5e 61 c6

5c 6b 05 f4	4a 7f 6b bf	4a 7f 6b bf	b1 c1 0b cc	c0 89 57 b1
7b 72 a2 6d	21 40 3a 3c	40 3a 3c 21	ba f3 8b 07	af 2f 51 ae
b4 34 31 12	8d 18 c7 c9	c7 c9 8d 18	f9 1f 6a c3	df 6b ad 7e
9a 9b 7f 94	b8 14 d2 22	22 b8 14 d2	1d 19 24 5c	39 67 06 c0
71 48 5c 7d	a3 52 4a ff	a3 52 4a ff	d4 11 fe 0f	2c a5 f2 43
15 dc da a9	59 86 57 d3	86 57 d3 59	3b 44 06 73	5c 73 22 8c
26 74 c7 bd	f7 92 c6 7a	c6 7a f7 92	cb ab 62 37	65 0e a3 dd
24 7e 22 9c	36 f3 93 de	de 36 f3 93	19 b7 07 ec	f1 96 90 50
f8 b4 0c 4c	41 8d fe 29	41 8d fe 29	2a 47 c4 48	58 fd 0f 4c
67 37 24 ff	85 9a 36 16	9a 36 16 85	83 e8 18 ba	9d ee cc 40
ae a5 c1 ea	e4 06 78 87	78 87 e4 06	84 18 27 23	36 38 9b 46
e8 21 97 bc	9b fd 88 65	65 9b fd 88	eb 10 0a f3	eb 7d ed bd

72 ba cb 04	40 f4 1f f2	40 f4 1f f2	7b 05 42 4a	71 8c 83 cf
1e 06 d4 fa	72 6f 48 2d	6f 48 2d 72	1e d0 20 40	c7 29 e5 a5
b2 20 bc 65	37 b7 65 4d	65 4d 37 b7	94 83 18 52	4c 74 ef a9
00 6d e7 4e	63 3c 94 2f	2f 63 3c 94	94 c4 43 fb	c2 bf 52 ef
0a 89 c1 85	67 a7 78 97	67 a7 78 97	ec 1a c0 80	37 bb 38 f7
d9 f9 c5 e5	35 99 a6 d9	99 a6 d9 35	0c 50 53 c7	14 3d d8 7d
d8 f7 f7 fb	61 68 68 0f	68 0f 61 68	3b d7 00 ef	93 e7 08 a1
56 7b 11 14	b1 21 82 fa	fa b1 21 82	b7 22 72 e0	48 f7 a5 4a
db a1 f8 77	b9 32 41 f5	b9 32 41 f5	b1 1a 44 17	48 f3 cb 3c
18 6d 8b ba	ad 3c 3d f4	3c 3d f4 ad	3d 2f ec b6	26 1b c3 be
a8 30 08 4e	c2 04 30 2f	30 2f c2 04	0a 6b 2f 42	45 a2 aa 0b
ff d5 d7 aa	16 03 0e ac	ac 16 03 0e	9f 68 f3 b1	20 d7 72 38

Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key
f9 e9 8f 2b	99 1e 73 f1	99 1e 73 f1	31 30 3a c2	fd 0e c5 f9
1b 34 2f 08	af 18 15 30	18 15 30 af	ac 71 8c c4	0d 16 d5 6b
4f c9 85 49	84 dd 97 3b	97 3b 84 dd	46 65 48 eb	42 e0 4a 41
bf bf 81 89	08 08 0c a7	a7 08 08 0c	6a 1c 31 62	cb 1c 6e 56
cc 3e ff 3b	4b b2 16 e2	4b b2 16 e2	4b 86 8a 36	b4 8e f3 52
a1 67 59 af	32 85 cb 79	85 cb 79 32	b1 cb 27 5a	ba 98 13 4e
04 85 02 aa	f2 97 77 ac	77 ac f2 97	fb f2 f2 af	7f 4d 59 20
a1 00 5f 34	32 63 cf 18	18 32 63 cf	cc 5a 5b cf	86 26 18 76
ff 08 69 64				
0b 53 34 14				
84 bf ab 8f				
4a 7c 43 b9				

Tutorial

1. Given the plain text {000102030405060708090A0B0C0D0E0F} and the key is {01010101010101010101010101010101},

a. Show the original contents of state, displayed as a 4 x 4 matrix.

b. Show the value of state after initial AddRoundKey.

c. Show the value of state after SubBytes.

d. Show the value of state after ShiftRows.

e. Show the value of state after MixColumns.

Module 5

Public Key Cryptosystem

- One key for encryption and a different but related key for decryption.
- A cryptographic algorithm that uses two related keys, a public key and a private key.
- The two keys have the property that deriving the private key from the public key is computationally infeasible.

- These algorithms have the following important characteristic.

1. It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
2. Either of the two related keys can be used for encryption, with the other used for decryption.

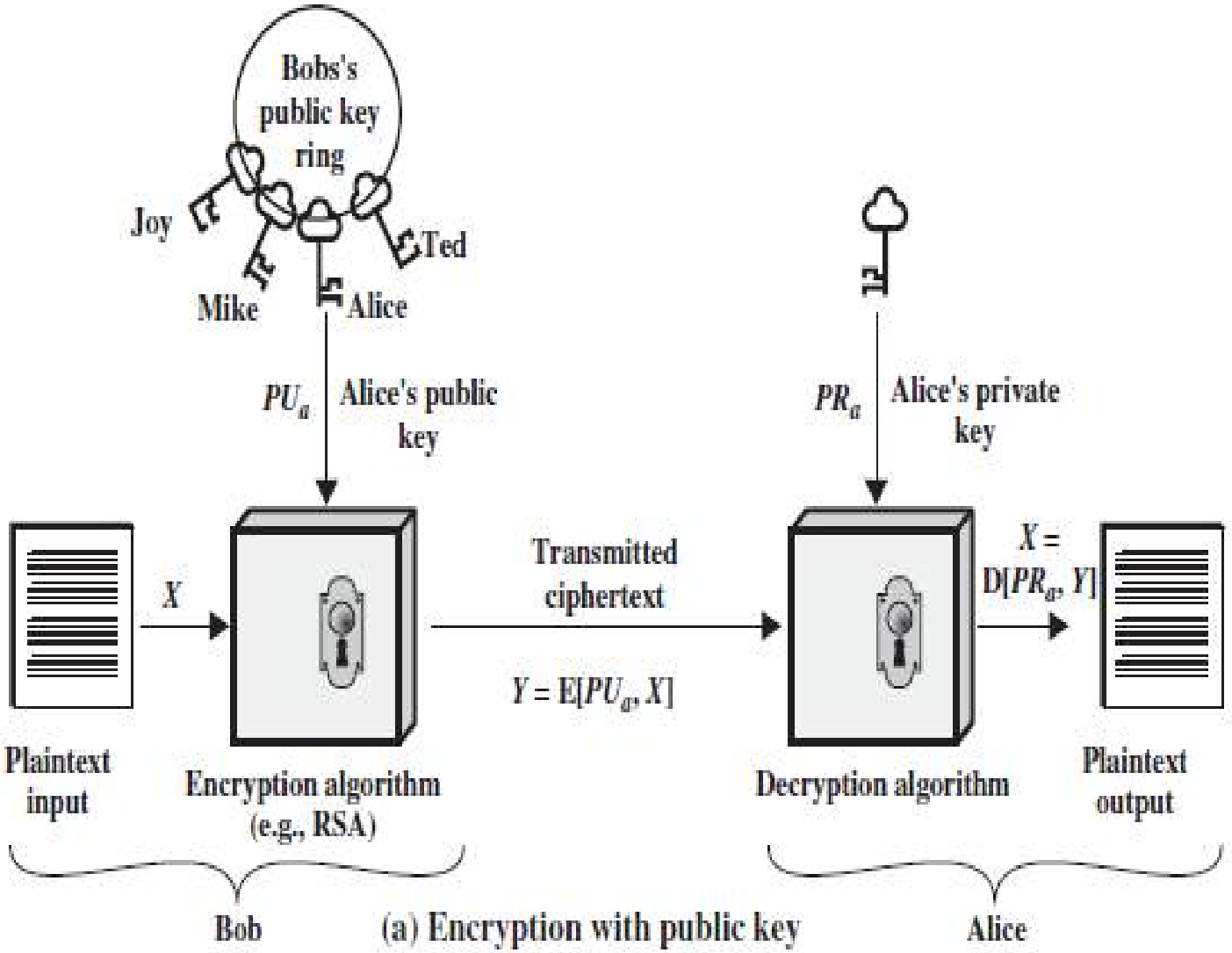
- A public-key encryption scheme has certain ingredients.
- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair 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 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.

- 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. Each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential 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.

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.


Bobs's public key ring
Joy
Ted
Mike
Alice
PU_a
Alice's public key
PR_a
Alice's private key
X
Transmitted ciphertext
$Y = E[PU_a, X]$
$X = D[PR_a, Y]$
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm
Plaintext output
Bob
(a) Encryption with public key
Alice

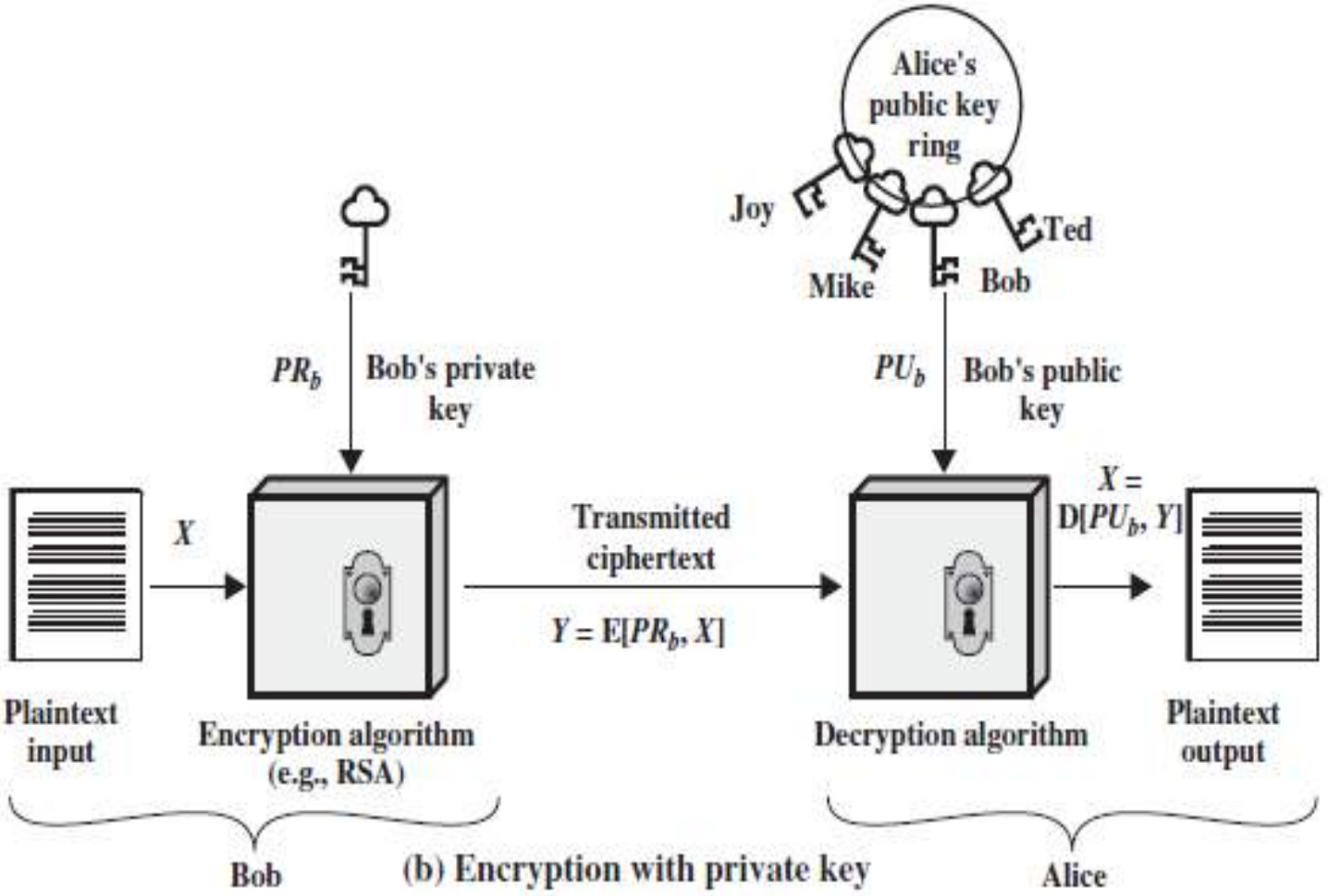
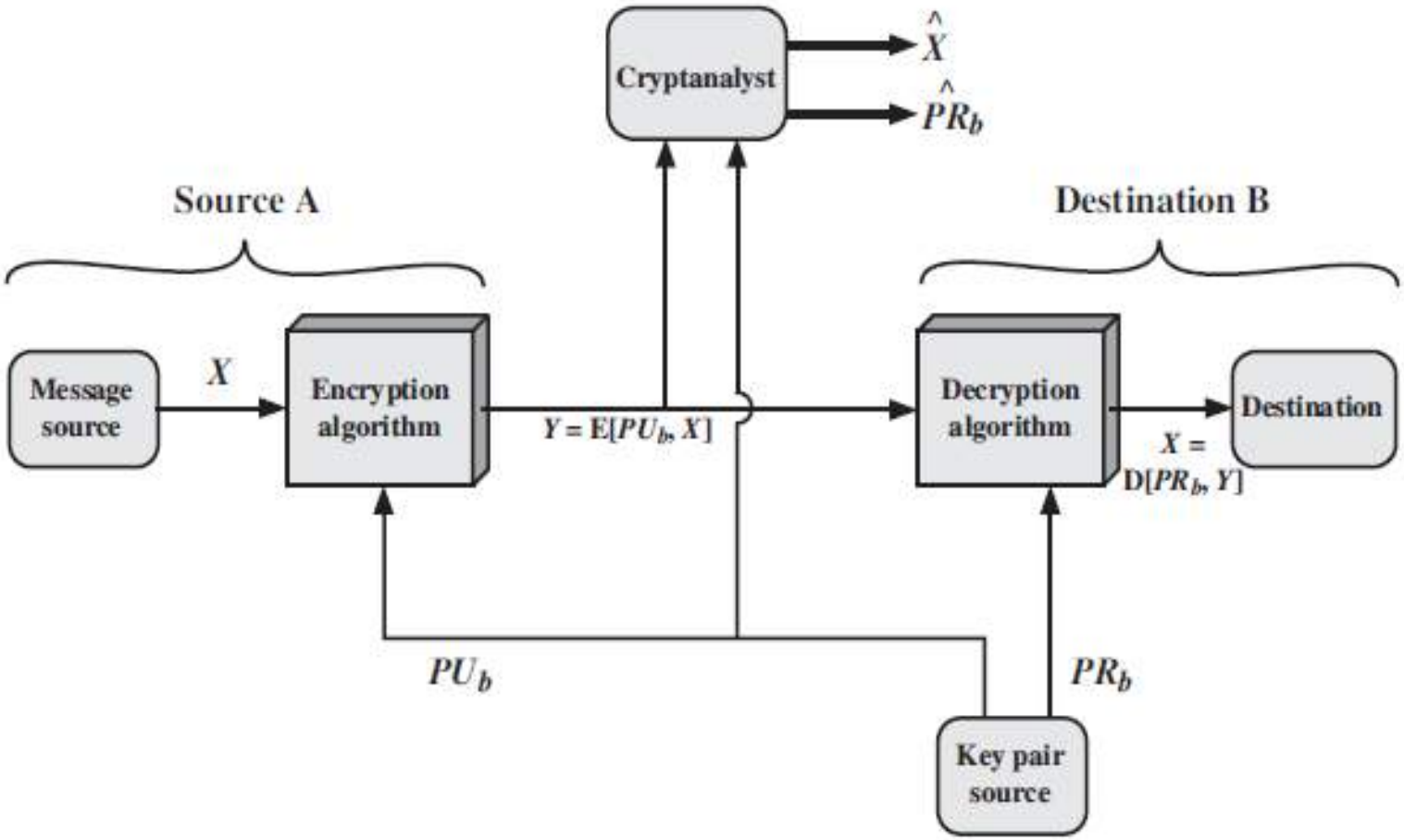


Figure 9.1 Public-Key Cryptography

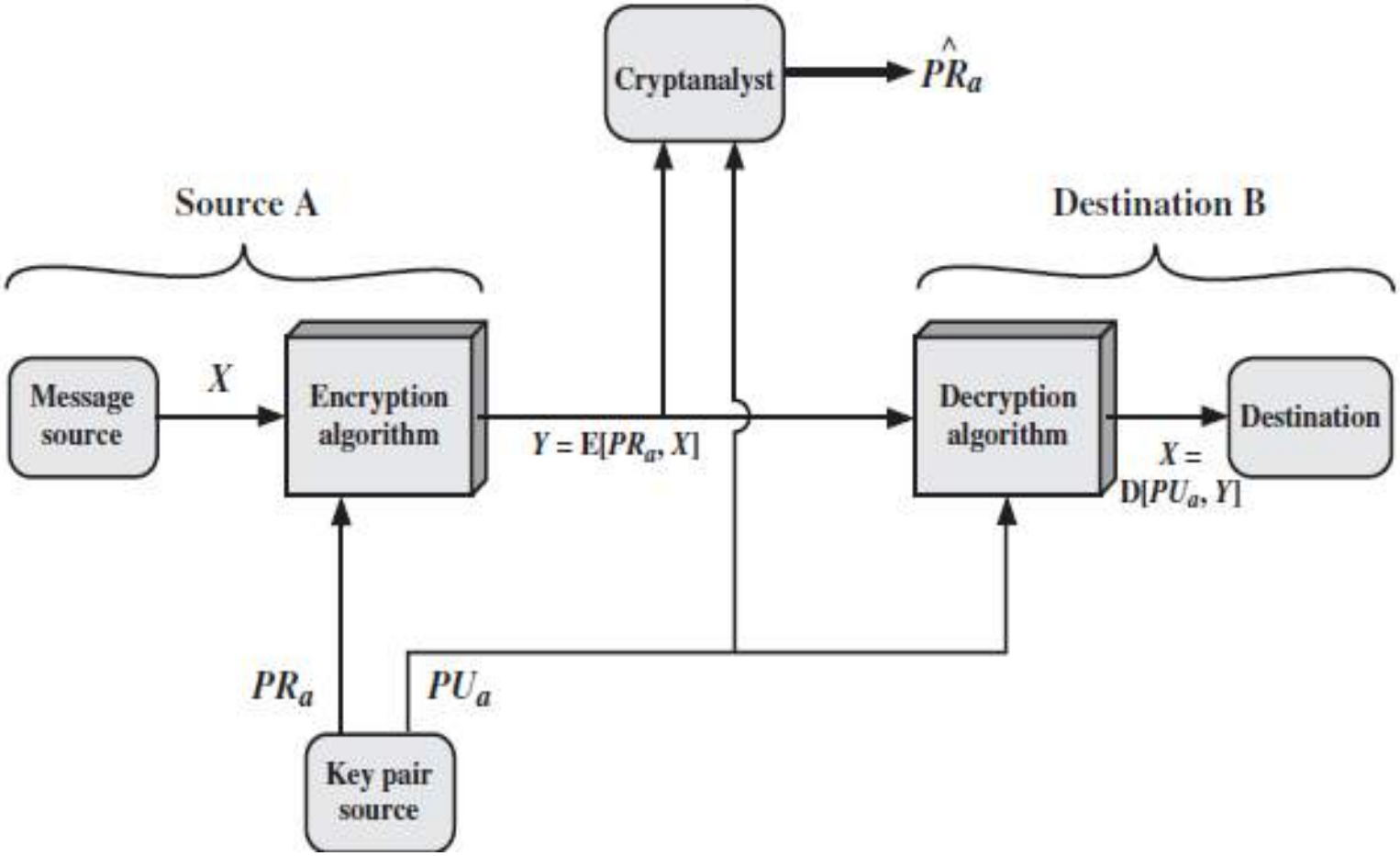
Public-Key Cryptosystem: Secrecy



- There is a source A that produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$.
- The message is intended for destination B.
- B generates a related pair of keys: a public key, PU_b, and a private key, PR_b.
- PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

- With the message *X and the encryption key* PU_b *as input, A forms the* ciphertext *Y = [Y1, Y2, . . . , YN]:*
- $Y = E(PU_b, X)$
- The intended receiver, in possession of the matching private key, is able to invert the transformation:
- $X = D(PR_b, Y)$

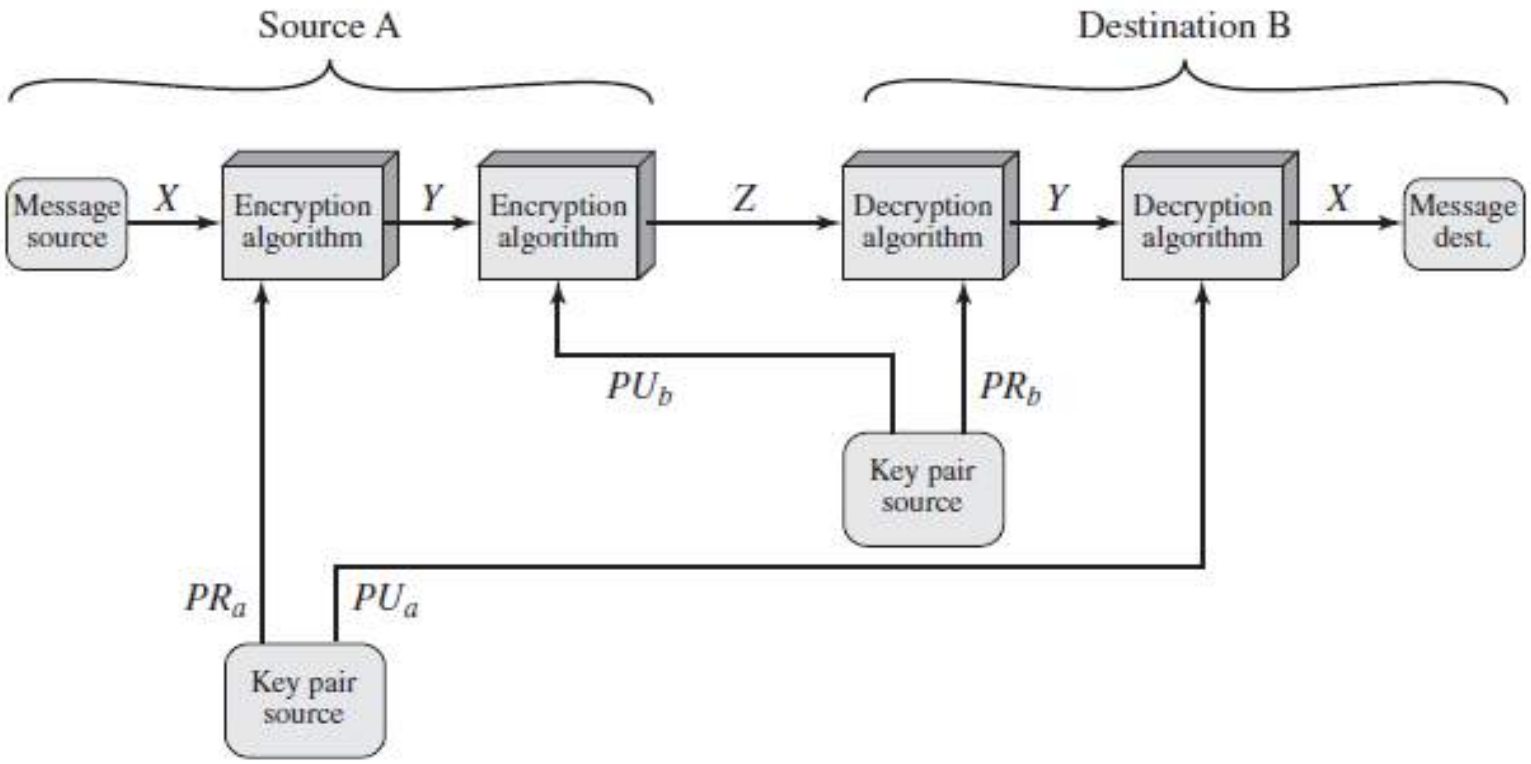
Public-Key Cryptosystem:Authentication



- The use of public-key encryption to provide authentication:
- $Y = E(PR_a, X)$
- $X = D(PU_a, Y)$

- A prepares a message to B and encrypts it using A's private key before transmitting it.
- B can decrypt the message using A's public key.
- Because the message was encrypted using A's private key, only A could have prepared the message.
- Therefore, the entire encrypted message serves as a **digital signature.**
- In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

Public-Key Cryptosystem:Authentication and Secrecy



- To provide both authentication function and confidentiality a double use of the public-key scheme is considered:
- $Z = E(PU_b, E(PR_a, X))$
- $X = D(PU_a, D(PR_b, Z))$

- In this case, encrypt a message, using the sender's private key.
- This provides the digital signature.
- Next, encrypt again, using the receiver's public key.
- The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key.
- Thus, confidentiality is provided.
- The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Applications for Public-Key Cryptosystems

- Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly.
- Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function.

- The use of **public-key cryptosystems is classified** into three categories

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

- Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications.

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
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:*

   $C = E(PU_b, M)$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

   $M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b, *to* determine the private key, PR_b.

5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, *C, to recover the original message,M.*
6. (Although useful, is not necessary for all public-key applications)

The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b,M)] = D[PR_b, E(PU_b,M)]$$

Trap-door one-way function

- A **one-way function** is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible:

$$Y = f(X) \quad \text{easy}$$

$$X = f^{-1}(Y) \quad \text{infeasible}$$

- *Easy -- a problem that can be solved in polynomial* time as a function of input length.
- If the length of the input is *n bits, then the* time to compute the function is proportional to n^a*, where a is a fixed constant.*
- *Such* algorithms are said to belong to the class **P.**

- A problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size.
- For example, if the length of the input is *n bits and the time to compute the function is proportional to* 2^n*, the* problem is considered infeasible.

- **Trap-door one-way function :-** calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known.
- With the additional information the inverse can be calculated in polynomial time.
- A trapdoor one-way function is a family of invertible functions f_k, *such that*

$Y = f_k(X)$ easy, if k and X are known

$X = f_k^{-1}(Y)$ easy, if k and Y are known

$X = f_k^{-1}(Y)$ infeasible, if Y is known but k is not known

RSA Algorithm

- RSA – invented by Ron **R**ivest, Adi **S**hamir and Len **A**dleman at MIT, in 1977.
- RSA is a block cipher.
- The most widely implemented cipher (asymmetric)
- Steps in RSA Algorithm are:

1. Key Generation
2. Encryption
3. Decryption

- The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and *n - 1 for some n.*
- *A typical size for n is 1024 bits, n is less than* 2^{1024}
- Plaintext is encrypted in blocks, with each block having a binary value less than some number *n.*
- *That is, the block size must be less than or equal to* $\log_2(n)$*; in practice, the block* size is *i bits, where* $2^i < n \leq 2^{i+1}$

- Both sender and receiver must know the value of *n.*
- This is a public-key encryption algorithm with a public key of *PU = {e, n} and a private* key of *PR = {d, n}.*

- The following requirements must be met:

1. *It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.*
2. *It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.*
3. *It is infeasible to determine d given e and n.*

- The private key is : {d, n} and public key is: {e, n}.
- Suppose that user A has published his public key and that user B wishes to send the message M to A.
- B calculates: $C = M^e \pmod{n}$ and transmits C to A.
- On receipt of the cipher text C, user A decrypts C by calculating: $M = C^d \pmod{n}$.

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

- to encrypt a message M using RSA the sender:

- obtains public key of recipient, PU={e,n}

- computes: $C=M^e \bmod n$, where $0 \leq M < n$

- to decrypt the ciphertext C the owner:

- uses their private key, PR={d,n}

- computes: $M=C^d \bmod n$

- The message M must be smaller than the modulus n

- The relationship between *e and d can be* expressed as
- *ed mod φ(n) = 1*
- where ϕ(*n) is the Euler totient function.*
- for *p, q prime, φ (pq) = (p - 1)(q - 1).*

$$ed \equiv 1 \bmod \phi(n)$$

$$d \equiv e^{-1} \bmod \phi(n)$$

- RSA Example:

1. Select primes: *p =17 & q =11*
2. Compute *n = pq = 17×11=187*
3. Compute ϕ(*n) = (p - 1)(q - 1)=16×10=160*
4. Select *e such that e is relatively prime to φ(n)=160 and less than φ(n); we choose e=7*
5. Determine *d such that d e = 1mod 160 and d < 160*

Value is *d =23 since 23×7=161= 1×160+1 (e . d = k . φ(n) + 1)*

6. Publish public key KU= {e,n} ={7,187}
7. Keep secret private key KR= {d,n}= {23,187}

- Example RSA encryption/decryption is:

given message M = 88 (88<187)

encryption:

$C = 88^7 \bmod 187 = 11$

decryption:

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

RSA Security

- Four approaches to attacking RSA are:

1. Brute force key search : This involves trying all possible private keys.
2. Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.
3. Timing attacks: These depend on the running time of the decryption algorithm.
4. Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

KEY MANAGEMENT

- Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data.
- Some sort of mechanism or protocol is needed to provide for the secure distribution of keys.
- Key distribution often involves the use of master keys, which are infrequently used and are long lasting, and session keys, which are generated and distributed for temporary use between two parties.

- There are actually two different aspects to the use of public key cryptography in this regard:

– distribution of public keys

– use of public-key encryption to distribute secret keys

Distribution of Public Keys

- Several techniques are there. All the techniques can be grouped into the following groups:

– Public announcement

– Publicly available directory

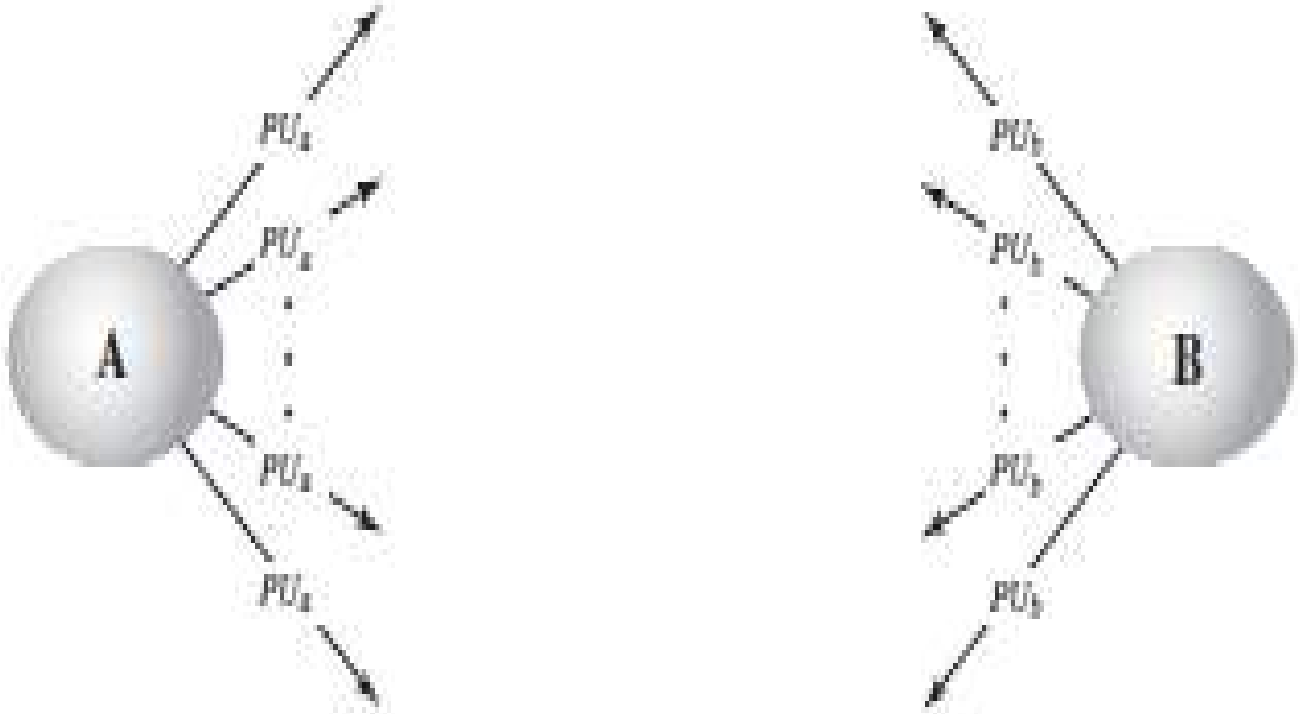
– Public-key authority

– Public-key certificates

Public Announcement

- users distribute public keys (PU) to recipients or broadcast to community at large
  - eg: Append keys to email messages or post to news groups or email list
- major weakness is forgery
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered the forger can masquerade (pretend) as the claimed user

Uncontrolled Public-Key Distribution



Publicly Available Directory

- Greater security can be obtained by registering keys with a public directory.
- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

- Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.

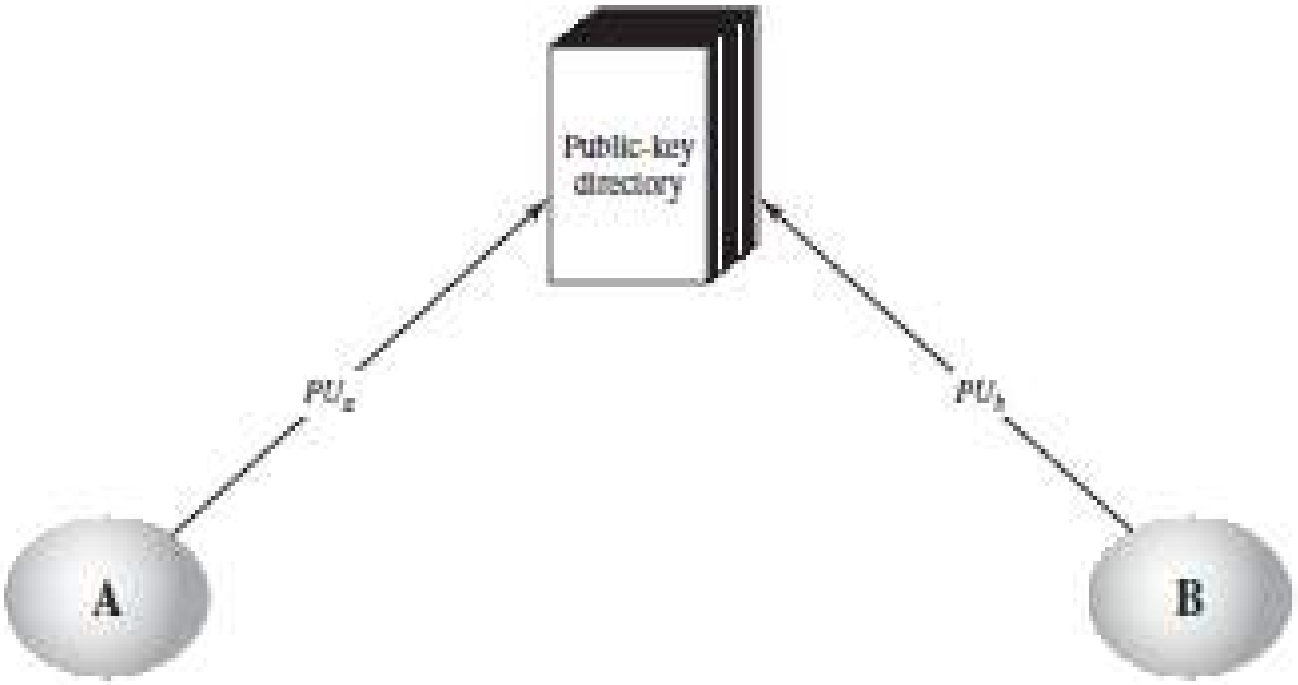
2. Each participant registers a public key with the directory authority.

- 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.

4. Participants could also access the directory electronically.
5. Periodically the authority publishes the entire directory or updates to the directory.

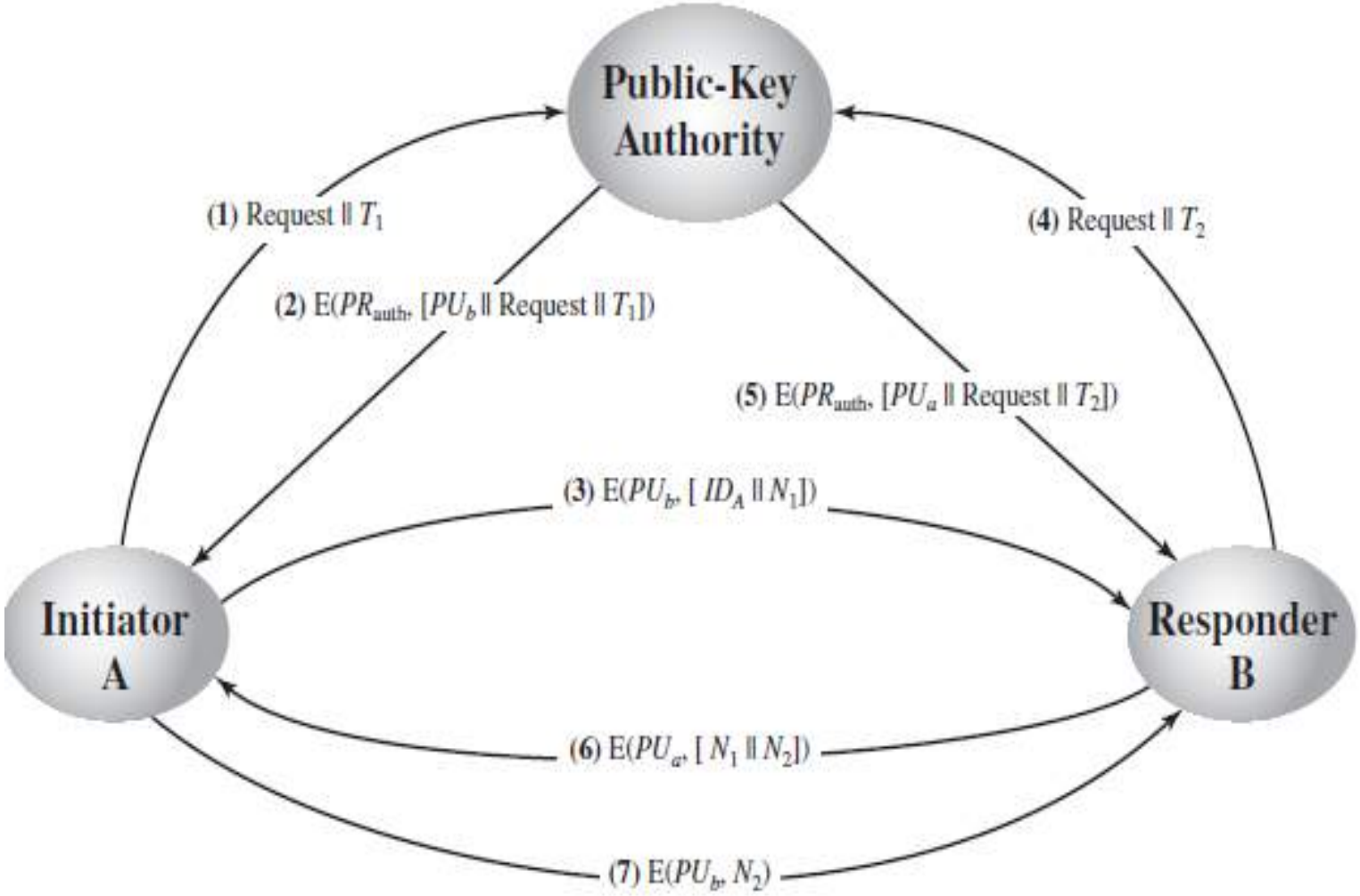
Public-Key Publication



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.
- 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.

Public-Key Distribution Scenario



- The following steps occur:

1. A sends a time stamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key, *PRauth.*

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.
- The original timestamp, so A can determine that this is not an old message from the authorit.

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (*IDA) and a nonce (N1), which is used to identify this transaction uniquely.*
4. B gets A's public key from the authority in the same manner as A gets 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:

6. B sends a message to A encrypted with *PUa and containing A's nonce (N1) as well as a new* nonce generated by B (*N2).*
7. 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 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
- The directory in the authority is vulnerable to tampering. The key authority must be trusted.

Public-Key Certificates

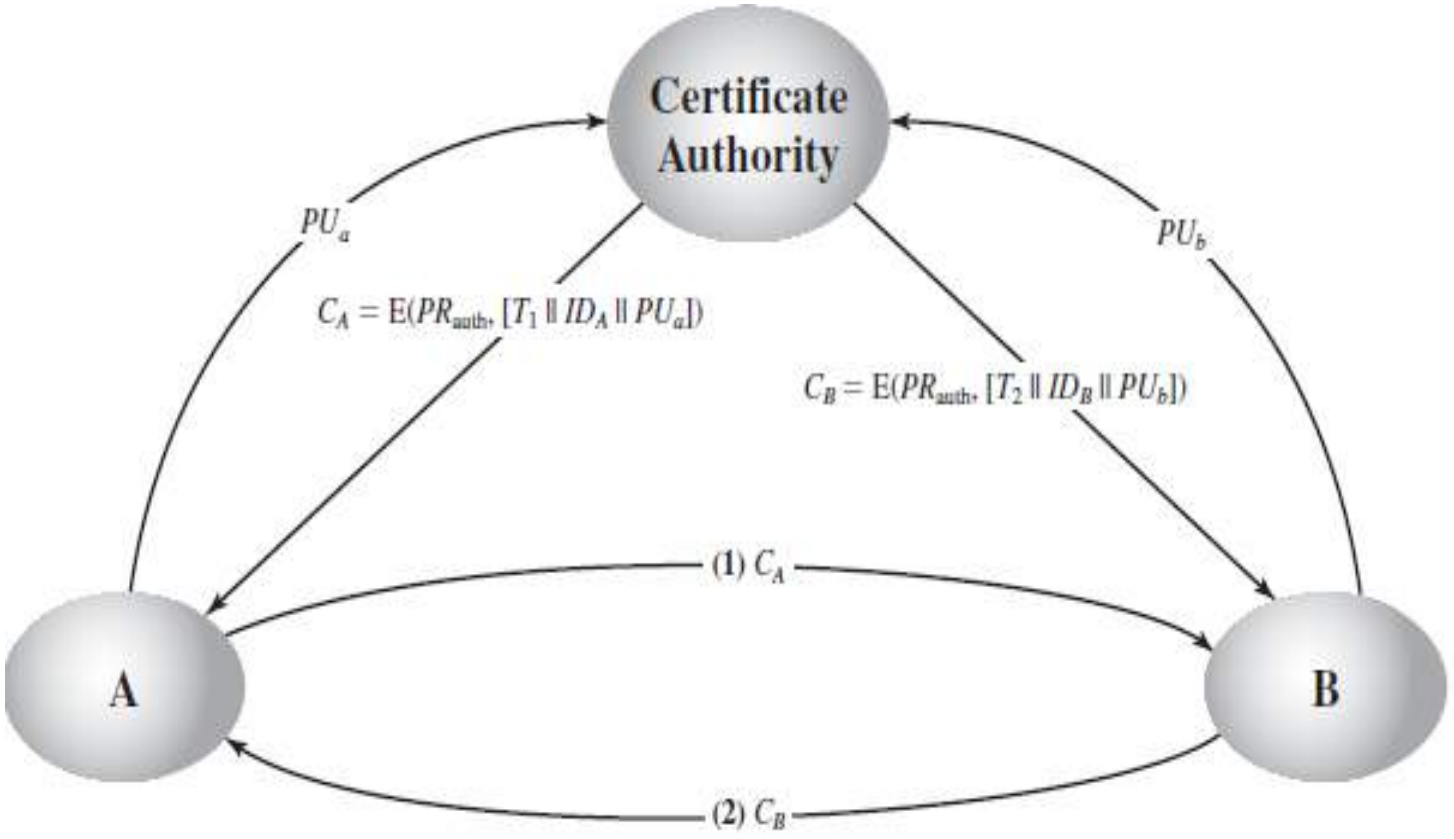
- Certificates can be used by participants to exchange keys without contacting a public-key authority.
- A certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party.
- The third party is a certificate authority, such as a government 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.
- A participant conveys its key information to another by transmitting its certificate.

- Following requirements can be placed on this particular 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.

Exchange of Public-Key Certificates



- For participant A, the authority provides a certificate of the form:

  CA = E(PRauth, [T // IDA // PUa])

- where *PRauth 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(*PUauth, CA) = D(PUauth, E(PRauth, [T // IDA // PUa]))*

Distribution of Secret Keys using public key cryptography

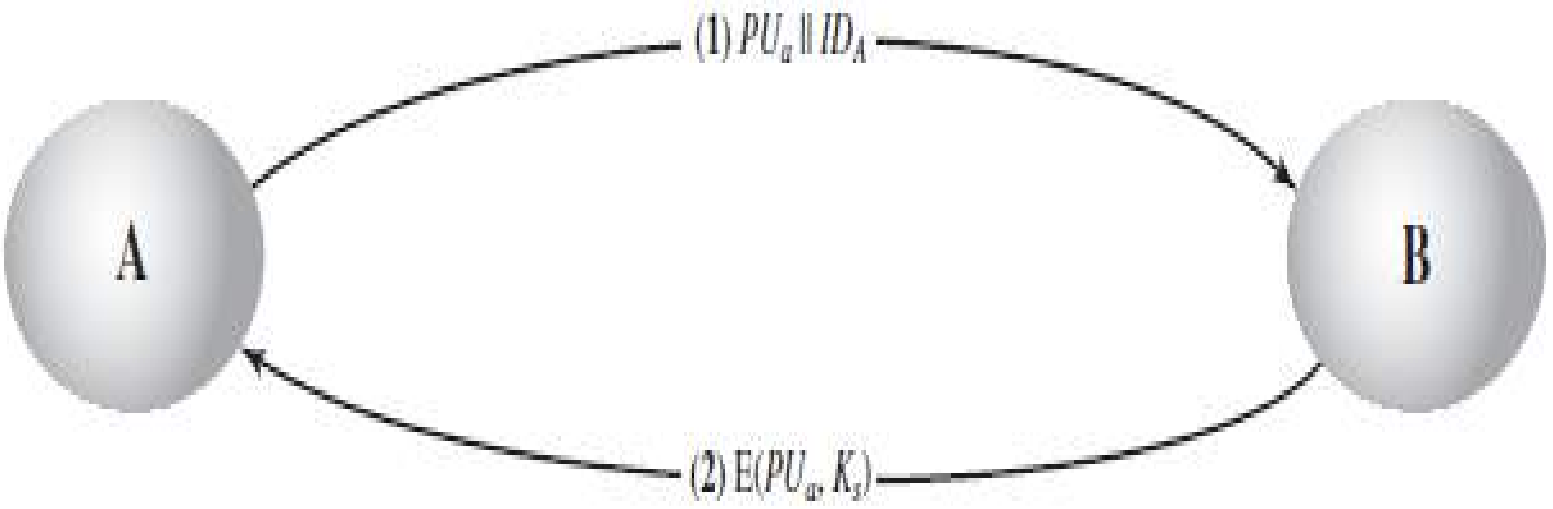
- Public-key encryption is more reasonably viewed as a vehicle for the distribution of secret keys to be used for conventional encryption.

Simple Secret-Key Distribution

- An extremely simple scheme put forward by Ralph Merkle
- If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa , and an identifier of A, IDA.
2. B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
3. A computes D(PRa, E(PUa ,Ks)) to recover the secret key. Since only A can decrypt the message, only A and B will know the identity of Ks.
4. A discards PUa and PRa and B discards PUa.

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



- A and B can now securely communicate encryption and the session keys Ks. At the completion of the exchange, both A and B discard Ks.
- 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 vulnerable to an active attack. If an opponent, E, has control of the intervening communications channel, then E can compromise the communications in the following way without being detected:

1. A generates a public/private key pair {PUa,PRa} and transmits a message intended for B consisting of PUa and identifier of A, IDA

2. E intercepts the message, creates its own public/private key pair {PUe,PRe} and transmits PUe // IDA to B.
3. B generates a secret key, Ks, and transmits E(PUe ,Ks).
4. E intercepts the message, and learns Ks by computing: D(PRe ,E(PUe ,Ks)).
5. E transmits E(PUa ,Ks) to A.

- The result is that both A and B know Ks and are unaware that Ks has also been revealed to E. A and B can now exchange messages using Ks.
- This method provides confidentiality only.

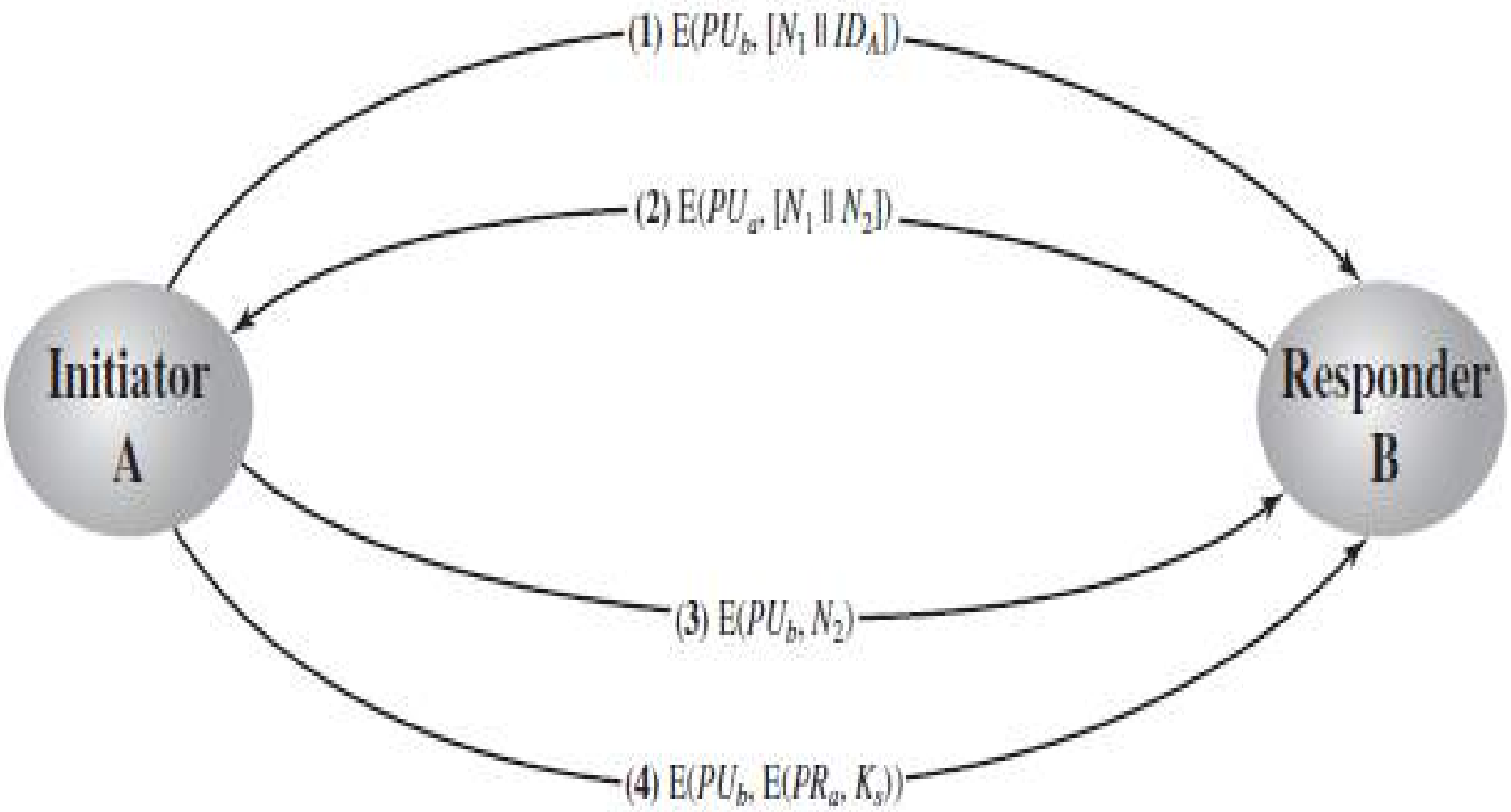
Secret Key Distribution with Confidentiality and Authentication:

1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to uniquely identify this transaction.
2. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2). Since only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.

3. A returns N2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key Ks and sends M = E(PUb , E(PRa ,Ks)) 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(PUa ,D(PRb , M)) to recover the secret key.

This scheme ensures both confidentiality and authentication in the exchange of a secret key.

Public-Key Distribution of Secret Keys



A Hybrid Scheme

- Another way to use public-key encryption to distribute secret keys is a hybrid approach
- This scheme uses a key distribution centre (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.
- A public key scheme is used to distribute the master keys. The following rationale is provided for using this three level approach.

- Performance
- Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption.

Backward Compatibility:

- The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption of software changes.
- The addition of a public-key layer provides a secure, efficient means of distributing master keys.

Module 6

Intruders

Intrusion:

- Entrance by force or without permission
- Any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource.
- Either via network or local

Intruder:

- Someone who intrudes on the privacy or property of another without permission.

Three classes of intruders:

1. Masquerader:

- An unauthorized individual penetrating a system's access controls to exploit a legitimate user's account.
- Mostly an outsider

2. Misfeasor:

- A legitimate user who performs unauthorized accesses to data, programs, or resources (or who is authorized for such access but misuses his or her privileges)
- Mostly an insider

3. Clandestine user:

- An individual who seizes supervisory control of the system and uses it to evade auditing and access controls or to suppress audit collection
- Either an insider or an outsider

Following are examples of intrusion:

- Performing a remote root compromise of an e-mail server
- Defacing a Web server
- Guessing and cracking passwords
- Copying a database containing credit card numbers
- Viewing sensitive data, including payroll records and medical information, without authorization
- Running a packet sniffer on a workstation to capture usernames and passwords

- **Intrusion Techniques**
- The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system.
- This requires the intruder to acquire information (such as password) that should have been protected.
- With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user.

- The password file can be protected in one of two ways:

One-way function:

- The system stores only the resulting value of a function based on the user's password.
- When the user presents a password, the system transforms that password and compares it with the stored value.

Access control:

- Access to the password file is limited to one or a very few accounts.

- Number of password crackers reports the following techniques for learning passwords:

1. Try default passwords shipped with systems.
2. Try all short passwords (those of one to three characters).
3. Try words in the system's online dictionary or a list of likely passwords.
4. Collect information about users, such as their full names.
5. Try users' phone numbers, social security numbers, and room numbers.
6. Try all legitimate license plate numbers.
7. Use a Trojan horse.
8. Tap the line between a remote user and the host system.

- The first six methods are various ways of guessing a password.
- The seventh method of attack listed earlier, the Trojan horse, can be particularly difficult to counter.

- A low-privilege user (hacker) produced a game program and invited the system operator to use it in his or her spare time.

- The program did indeed play a game, but in the background it also contained code to copy the password file
- Because the game was running under the operator's high-privilege mode, it was able to gain access to the password file.

- The eighth attack listed, line tapping, is a matter of physical security.

Intrusion Detection

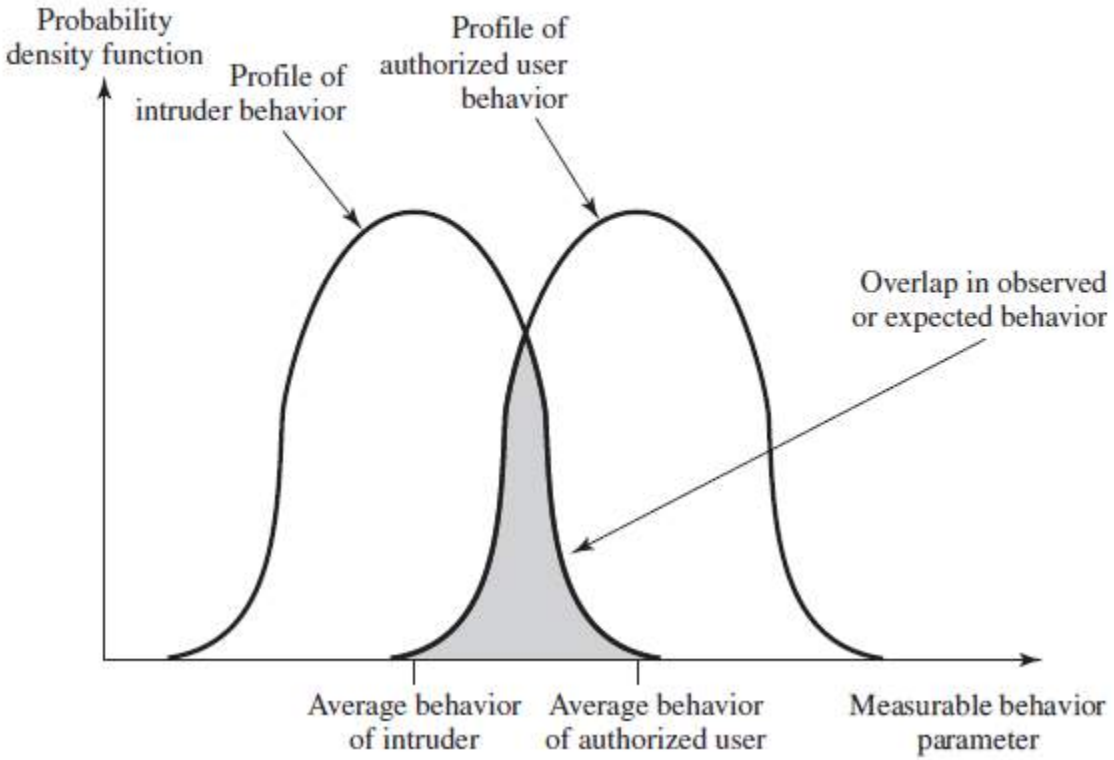
1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
2. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

- Intrusion detection is based on the assumption that the behaviour of the intruder differs from that of a legitimate user.
- However, there is some overlap between the behaviour of the intruder and that of a legitimate user.

- A loose interpretation of intruder behaviour will lead to a number of “false positives” (Authorized users may be identified as intruders)

- A tight interpretation will lead to an increase in no. of “false negatives” (Intruders may not be identified as intruders)

Profiles of Behavior of Intruders and Authorized Users



Approaches to Intrusion Detection

1. **Statistical anomaly detection:**

- Involves the collection of data relating to the behaviour of legitimate users over a period of time.
- Statistical tests are applied to observed behaviour to determine whether that behaviour is not legitimate user's behaviour

a) **Threshold detection:** This approach involves defining thresholds for frequency of occurrence of various events

b) **Profile based detection:** Profile of the activity of each user is developed and used to detect changes in the behaviour of users

2. **Rule-based detection:**

- Involves an attempt to define a set of rules that can be used to decide that a given behaviour is that of an intruder.

  a)**Anomaly based:** Rules are developed to detect deviation from previous usage patterns.

  b)**Penetration identification:** An expert system approach that searches for suspicious behaviour.

Audit Records

- A fundamental tool for intrusion detection is the audit record.
- A record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

Native Audit Records:

- Virtually all multiuser operating systems include accounting software that collects information on user activity
- Advantage: No additional collection software is needed
- Disadvantage: The native audit records may not contain the needed information (or it may not contain information in a convenient form).

- **Detection-specific audit records:**

- Audit records containing only that information required by the intrusion detection system

- Advantage: Can be vendor independent and ported to a variety of systems.

- Disadvantage: The extra overhead to maintain two accounting packages on a machine.

Statistical Anomaly Detection

- Compare the current behaviour with the previous behaviour of legitimate users using statistical tests. Two types are:
- Threshold detection
- Profile-based detection

- **Threshold detection:**
- Counting the number of occurrences of a specific event type over an interval time.
- If the count surpasses the threshold, intrusion is assumed.
- Threshold detection is a crude and ineffective detector. So it is used in conjunction with more sophisticated techniques.

- **Profile-based detection:**
- Characterize the past behaviour of users and then detect significant deviations in the current behaviour.
- To characterize the past behaviour, analysis of audit records is required.
- Useful metrics to analyze the past users' behaviour are:

- Counter
- Gauge
- Interval timer
- Resource utilization

- **Counter:**
- A nonnegative integer that may be incremented but not decremented until it is reset by management action.
- Typically, a count of certain event types is kept over a particular period of time.

Some example events are:

- The number of logins by a single user in an hour.
- The number of times a given command is executed during a single user session.
- The number of password failures during a minute

- **Gauge:**
- A non-negative integer that may be incremented or decremented.
- Typically, a gauge is used to measure the current value of some entity.

Examples include:

- The number of logical connections assigned to a user application.
- The number of outgoing messages queued for a user process.

- **Interval time:**
- The length of time between two related events.

- An example is the length of time between successive logins to an account.

- **Resource utilization:**
- Quantity of resources consumed during a specified period.
- Examples include:

- The number of pages printed during a user session.

- Total time consumed by a program execution.

Statistical tests

- Various tests can be performed to determine whether current activity fits within acceptable limits.

1. Mean and standard deviation
2. Multivariate
3. Markov process
4. Time series
5. Operational

Mean and standard deviation:

- The simplest statistical test is to measure mean and standard deviation of a parameter over some historical period.
- This gives a reflection of the average behaviour and its variability.
- The use of mean and standard deviation is applicable to a wide variety of counters, timers, and resource measures.

Multivariate model:

- Based on correlations between two or more variables.
- Intruder behaviour may be characterized with greater confidence by considering such correlations of variables, examples are:

- Processor time and resource usage
- Login frequency and session elapsed time

Markov process model:

- Used to establish transition probabilities among various states.

  - As an example this model might be used to look at transitions between certain commands.

Time series model:

- This focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly.
- A variety of statistical tests can be applied to characterize abnormal timing.

Operational model:

- Based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records.
- Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits.

(Eg: A large number of login attempts over a short period suggests an attempted intrusion)

- The Advantage of the use of statistical anomaly detection are:

1. The main advantage is that a prior knowledge of security flaws is not required.
2. The detector program learns what is “normal” behaviour and then looks for deviations.
3. The approach is not based on system-dependant characteristics and vulnerabilities.
4. It is readily portable among a variety of systems.

Rule-Based Intrusion Detection

- Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

Two classes are:

- Rule-based anomaly detection
- Rule-based penetration identification

Rule-based anomaly detection

- Analyzes historical audit records and extracts a set of rules based on previous usage patterns.
- Rules may represent past behaviour patterns of users, programs, privileges, time slots, terminals, and so on.
- Then matches the current behaviour (observed) against the set of rules and decide whether the behaviour deviates from pervious usage patterns or not.
- It does not require the knowledge of security vulnerabilities within the system.
- In order for this approach to be effective, a rather large database of rules will be needed. (For example, anywhere from 104 to 106 rules)

Rule-base penetration identification

- Based on expert system technology.
- Generates rules to identify known penetrations or feasible penetrations that would exploit known weakness.
- Typically, the rules are generated by “experts” rather than by means of an automated analysis of audit records.
- Experts conducts interview of system administrators and security analysts to collect known penetration activities threatening the security of the target system and generate rules for the penetration activities.

Base-Rate Fallacy

- Practically an intrusion detection system needs to detect a substantial percentage of intrusions with few false alarms

– if too few intrusions detected -> the s/m provides false security

– if too many false alarms -> s/m managers will begin to ignore alarms or it leads to wastage of time

- It is very difficult to meet high rate of intrusion detection with a low rate of false alarms.
- In general, if the actual numbers of intrusions is low compared to the number of legitimate users of a system, then the false alarm rate will be high unless the test is extremely discriminating.

Distributed Intrusion Detection

- Until recently, work on intrusion detection systems focused on single-system stand-alone facilities.
- However, a typical organization needs to defend a collection of hosts connected by a network.
- It is possible to mount a defence by using stand-alone intrusion detection systems on each host.
- A more effective defence can be achieved by coordination and cooperation among intrusion detection systems across the network.

- The major issues in the design of a distributed intruder detection system are:

1. Different audit record formats

- Different systems may use different types of audit records Integrity and confidentiality of messages

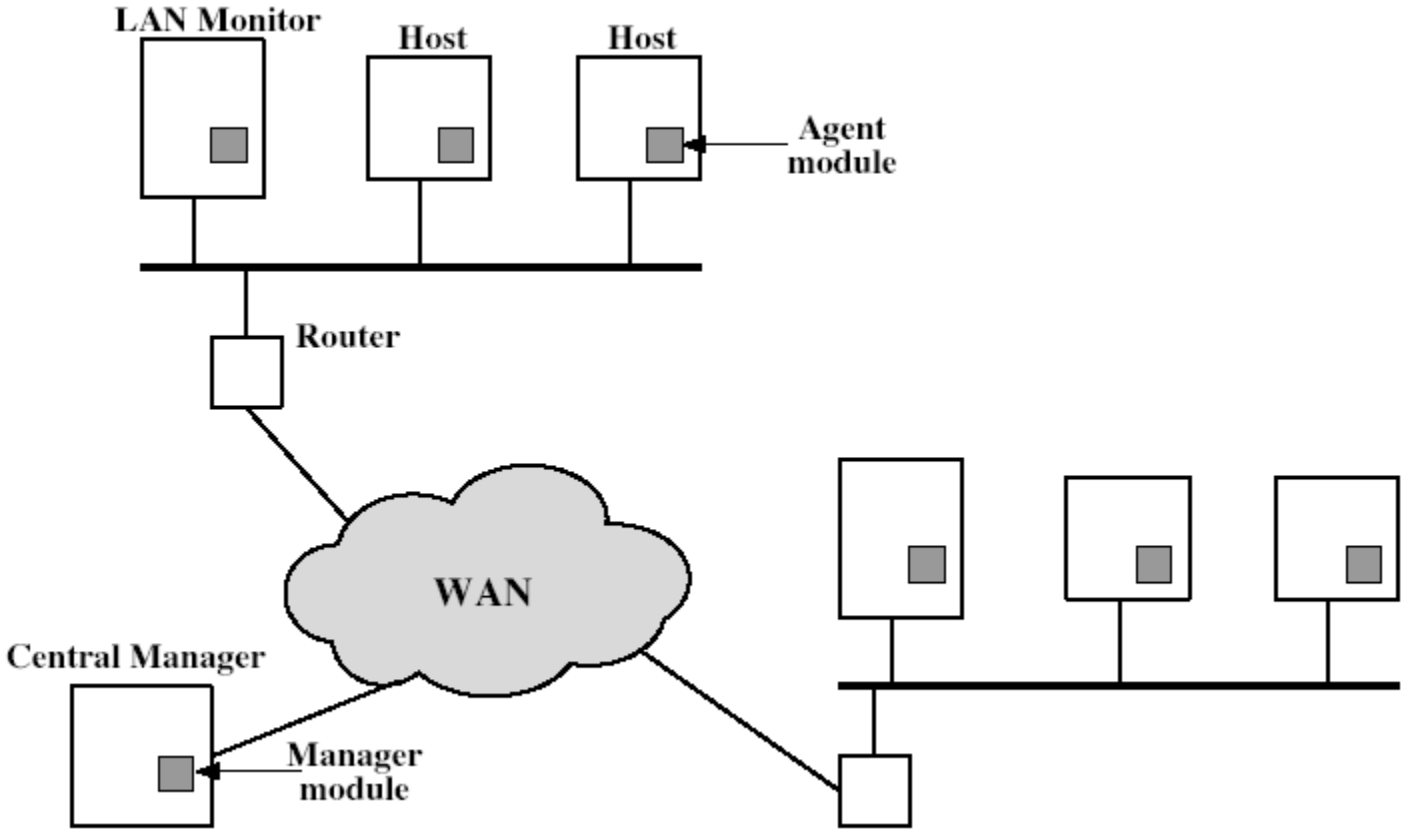
- Audit records may be transmitted across the network. Integrity is required to prevent an intruder from masking his activities by altering the transmitted audit records.

- Confidentiality is required because the transmitted audit records could be valuable.

2. Either centralized or decentralized architecture can be used

- With a centralized architecture, there is a single central point of collection and analysis of all audit data
- With a decentralized architecture, there are more than one analysis centres, but these must coordinate their activities and exchange information

Distributed Intrusion Detection – Architecture



- Three main components are:

Host agent module:

- An audit collection module operating as a background process on a monitored system.
- Collects data on security-related events on the host and transmit these to the central manager.

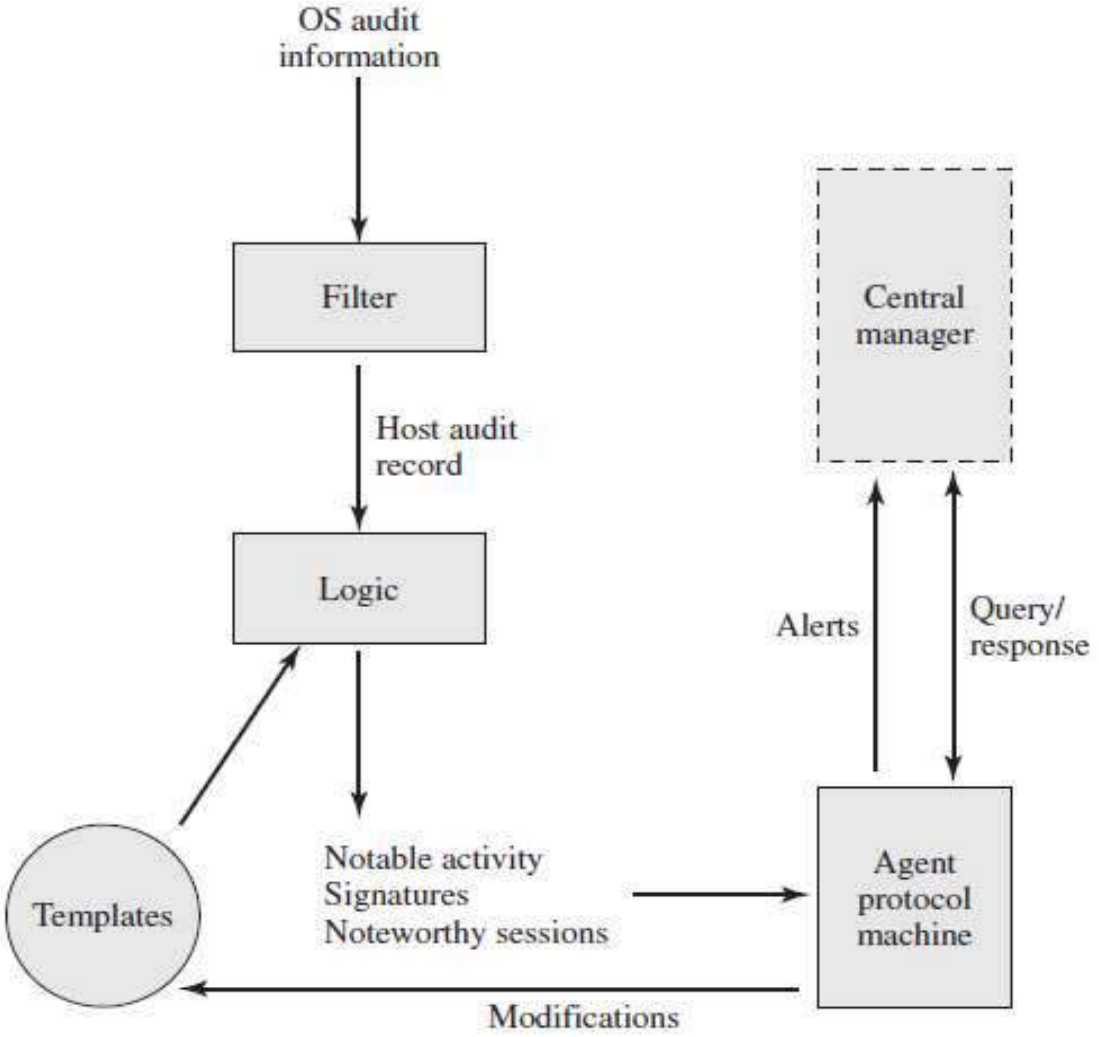
LAN monitor agent module:

- Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager

Central manager module:

- Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

Agent Architecture



Agent Architecture

Operation procedure

1. The agent captures each audit record produced by the native audit collection system.
2. A filter retains only security-related records.

- These records are then reformatted into a standardized format called the host audit record (HAR).

3. A template-driven logic analyzes the records for suspicious activity.

- The agent scans for notable events that are of interest independent of any past events.
- The agent looks for anomalous behaviour of an individual user based on a historical profile of that user such as number of program executed, number of files accessed.

4. When suspicious activity is detected, an alert is sent to the central manager.

- The central manager includes an expert system that can draw inferences from received data.
- The manager may also query individual systems for copies of HARs to correlate with those from other agents.
- The LAN monitor agent also supplies information to the central manager.

- The LAN monitor agent audits host-host connections, services used, and volume of traffic.

Honeypots

- Honeypots are systems that are designed to

- Divert an attacker from accessing critical systems

- Collect information of the attacker's activity

- Encourage the attacker to stay on the system long enough for administrators to respond.

Intrusion Detection Exchange Format

- To facilitate the development of distributed intrusion detection systems certain standards are needed.
- Such standards are the focus of the IETF Intrusion Detection Working Group.
- The purpose of the working group are

1. to define data formats and exchange procedures for sharing information.
2. to management systems that may need to interact with them.

- The outputs of this working group include:

1. A requirement document which describes the high-level functional requirements for communication between intrusion detection systems.
2. A common intrusion language specification which describes data formats that satisfy the requirements.
3. A framework document which identifies existing protocols best used for communication.

PASSWORD MANAGEMENT

- **Password Protection**
- Virtually all multiuser systems require that a user is provided with the following :
  - Name or identifier (ID)
  - A password.
- The ID provides security in the following ways:

- The ID determines whether the user is authorized to gain access to a system.
- The ID determines the privileges accorded to the user.

- **The Vulnerability of Passwords:**
- Consider a scheme that is widely used on UNIX in which passwords are never stored in the clear form
- The procedure for storing (loading) password in UNIX is as given below:
- Each user selects a password up to eight characters.
- This is converted into a 56-bit value (using 7 bit ASCII) that serves as key input to an encryption routine known as crypt(3).
- The encryption routine is based on DES.
- The DES algorithm is modified using a 12-bit “salt” value.
- This value is related to the time at which the password is assigned to the user.

- The modified DES algorithm is exercised.
- The output of the algorithm then serves as input for a second encryption.
- This process is repeated for a total of 25 encryptions.
- The resulting 64-bit output is then translated into an 11-character sequence.
- The hashed password is then stored, together with a plaintext copy of the salt, in the password file

- The “salt” serves three purposes:

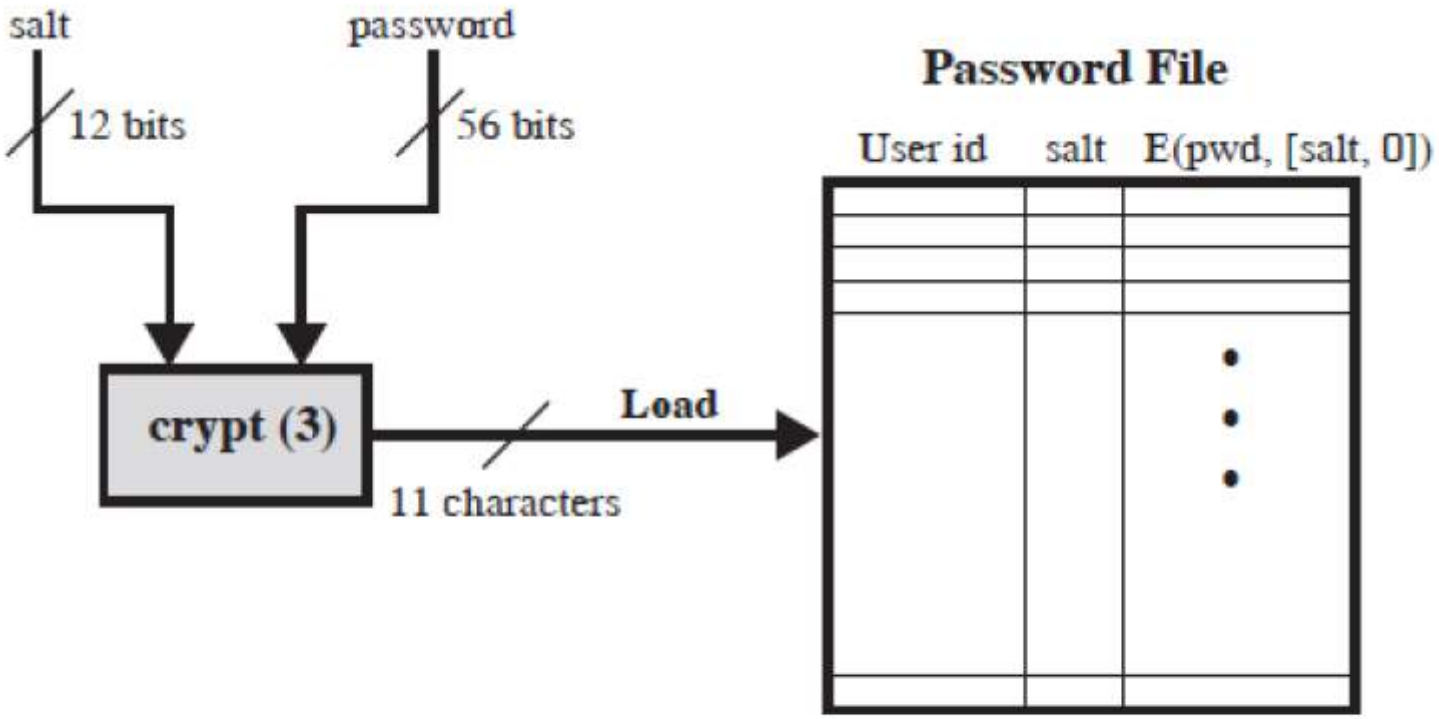
1. It prevents duplicate passwords from being visible in the password file.

   - Even if two users choose the same password, those passwords will be assigned at different times. Hence, the “extended” passwords of the two users will differ (as the salt value is not the same at different times).

2. It effectively increases the length of the password without requiring the user to remember additional characters

- - Hence, the number of possible passwords is increased by a factor of 4096, increasing the difficulty of guessing a password.

3. It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack



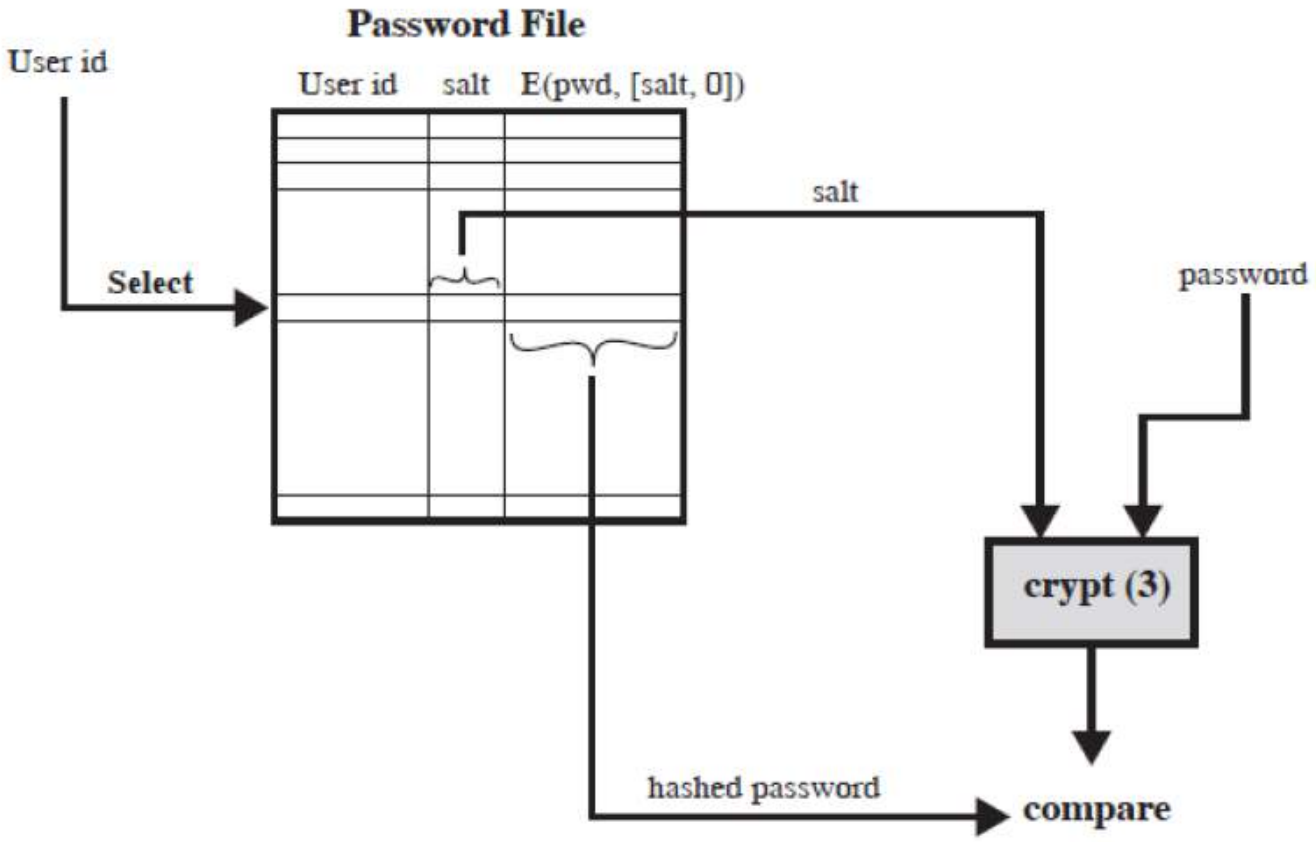
Loading a new password

- The procedure of verifying password employed in UNIX is given below

1. When a user attempt to log on to a UNIX system, the user provides an ID and a password
2. The operating system uses the ID

- to index into the password file and
- to retrieve the plaintext salt and the encrypted password

3. The salt and user-supplied password are used as input to the encryption routine
4. If the result matches the stored value, the password is accepted



Verifying a password

- **Access Control:**
- One way to thwart a password attack is to deny the opponent access to the password file
- If the encrypted password portion of the file is accessible only to a privileged user, then the opponent can’t read it without knowing the password of a privileged user

Password Selection strategies

- Many users choose a password that is too short or too easy to guess.
- If users are assigned passwords of 8 randomly selected printable characters, password cracking is effectively impossible
- It would be almost impossible for most users to remember 8 randomly selected printable character password.

The Main Goal:

- Eliminating guessable passwords
- Allowing users to select a password that is memorable

Four basic technique to achieve the goals are in use:

- User education
- Computer-generated passwords
- Reactive password checking
- Proactive password checking

- **User education strategy:**

This involves:

- Informing users the importance of using hard-to-guess password and
- Providing guidelines for selecting strong password.

Computer-generated passwords:

- If passwords are quite random in nature, it is difficult to remember for users
- In general, computer-generated password schemes have a history of poor acceptance

- **Reactive password checking strategy:**
- The system periodically runs its own password cracker to find guessable passwords
- The system cancels any passwords that are guessed and notifies the user.

Proactive password checker:

- The most promising approach to improve security
- A user is allowed to select his or her own password
- The system checks to see if the password is allowable, at the time of selection.

- If not, rejects his/her password

Two problems with this approach are:

- Space - The dictionary must be very large to be effective, it needs a large space
- Time - The time required to search a large dictionary may be large.