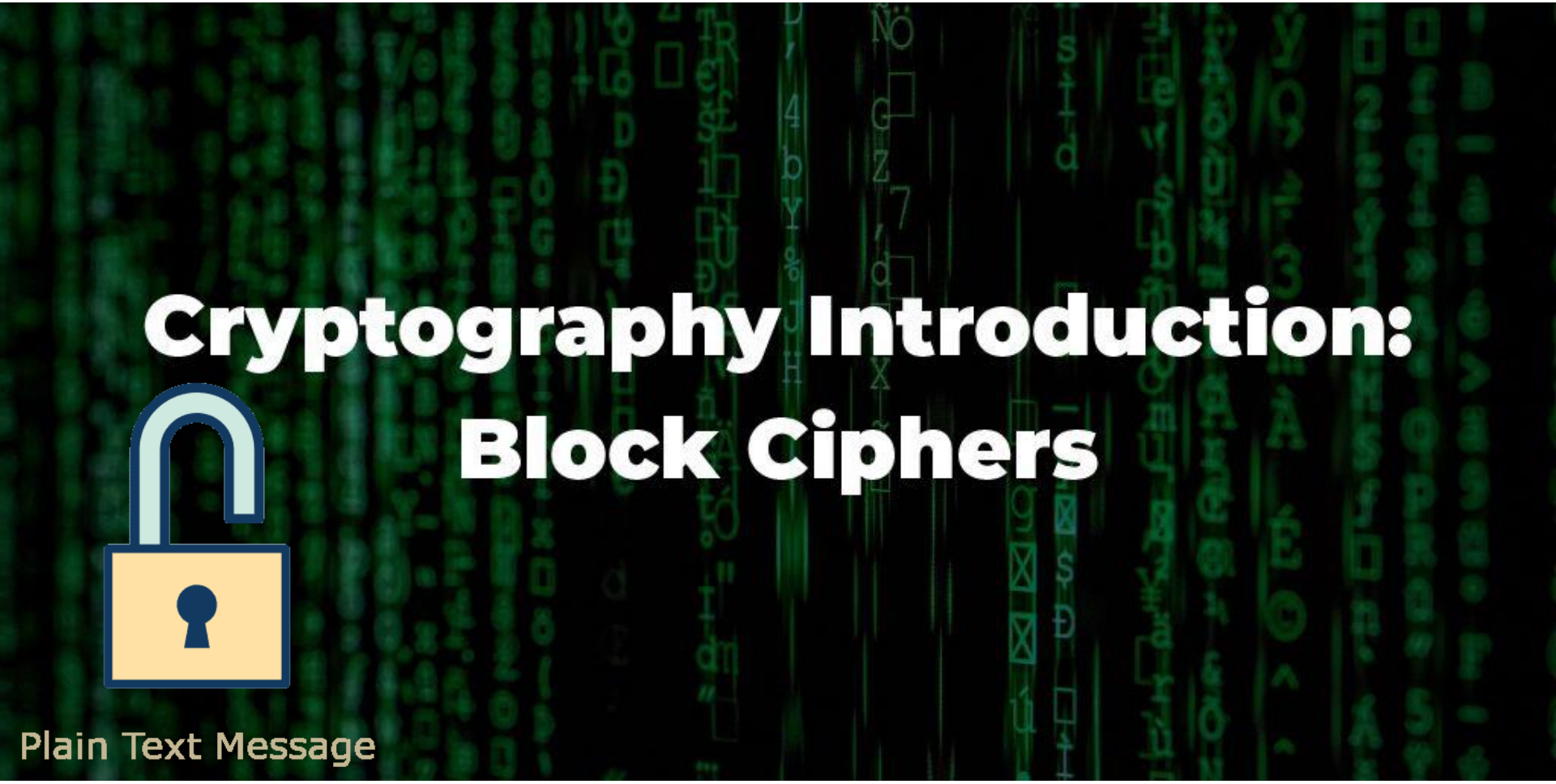
MODULE - II



Presented by: Arif Mohammad Abdul

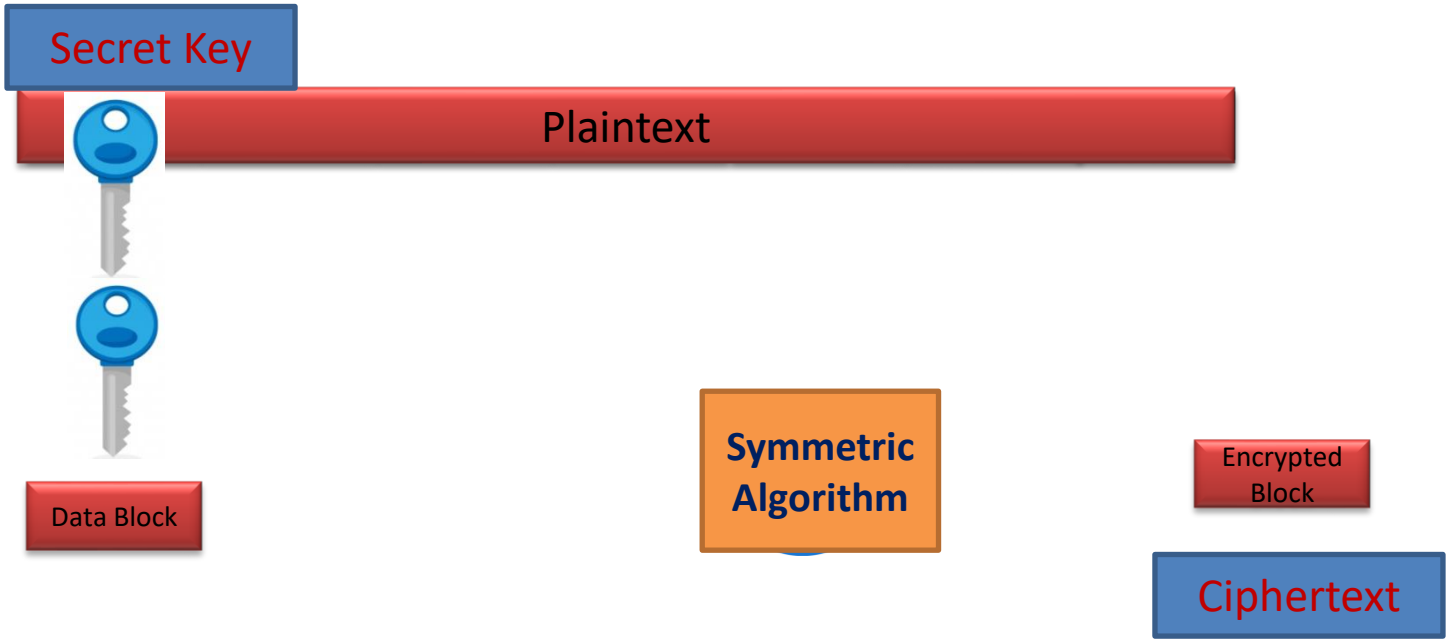
Symmetric Key Cryptography: Block Ciphers and the Data Encryption Standard (DES) algorithm. Differential and linear cryptanalysis, triple DES. Block cipher design principles, Block cipher modes of operation, Advanced Encryption Standard (AES), Stream Ciphers: RC4.

Syllabus

Block Cipher : Introduction

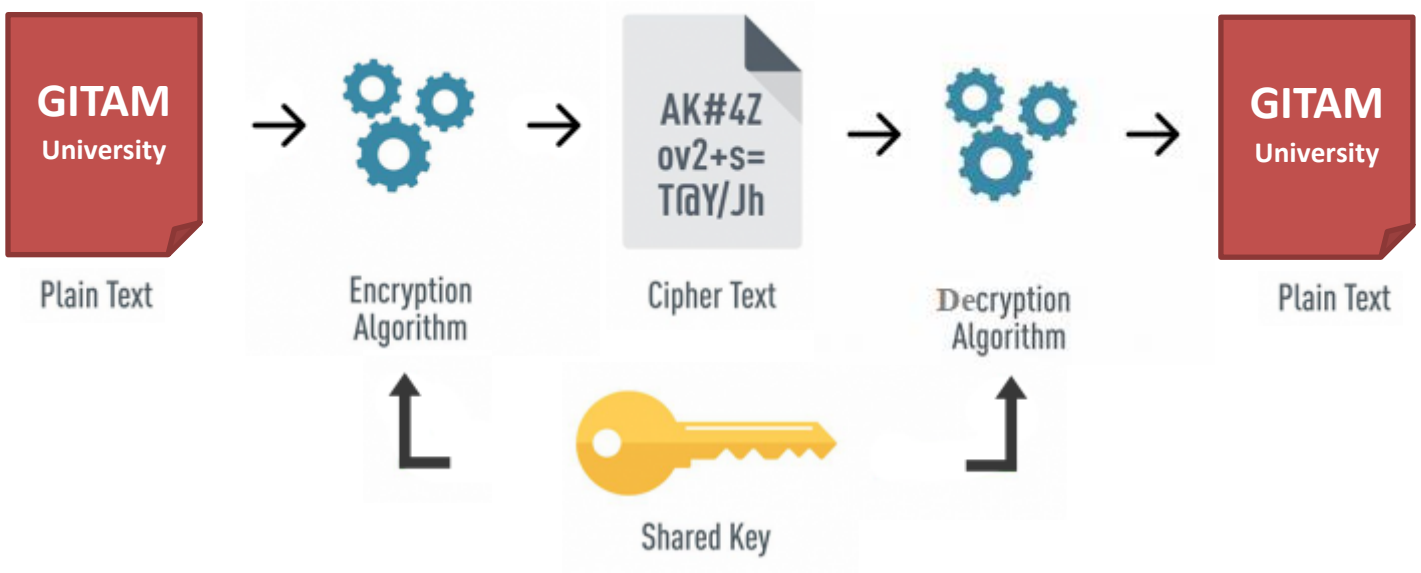
- A block cipher is a **symmetric encryption** which operates on blocks of data.
- A block cipher breaks plaintext into **fixed sized blocks (fixed number of bits i.e., n bits)**.
- Sizes of block is depends on type of **symmetric algorithm**.
- All input plaintext bits **need to be present** before processing the block.
- If the plaintext length is **not divisible by n**, and the last block is only partially filled, then that block needs to be **padded**.
- When padding is applied, Alice and Bob need to **agree** on the **padding**.
- Takes **one block of plaintext at a time** and transfer it into **same length of ciphertext block** using user's provided **secret key.**

Block Cipher

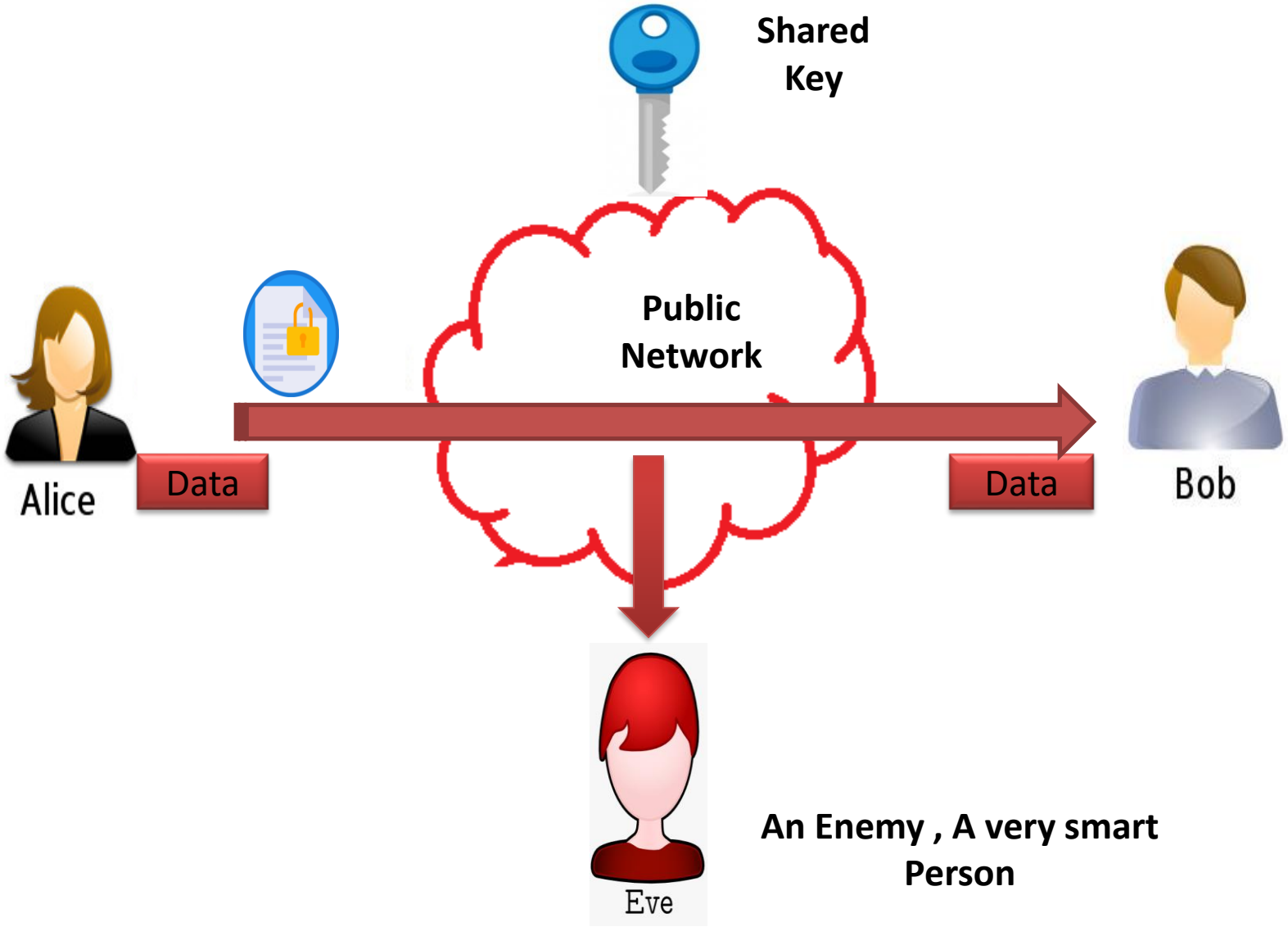


Symmetric Algorithm

- **Symmetric algorithm** uses only **one key** (a **secret key**) is used to both **encrypt** **and** **decrypt** the message.



Symmetric Algorithm



Alice, Bob, Eve Framework

Symmetric Algorithms

Some of the symmetric algorithms include:

- Data Encryption Standard (DES)
- Triple DES
- Advanced Encryption Standard (AES)

Simplified-DES

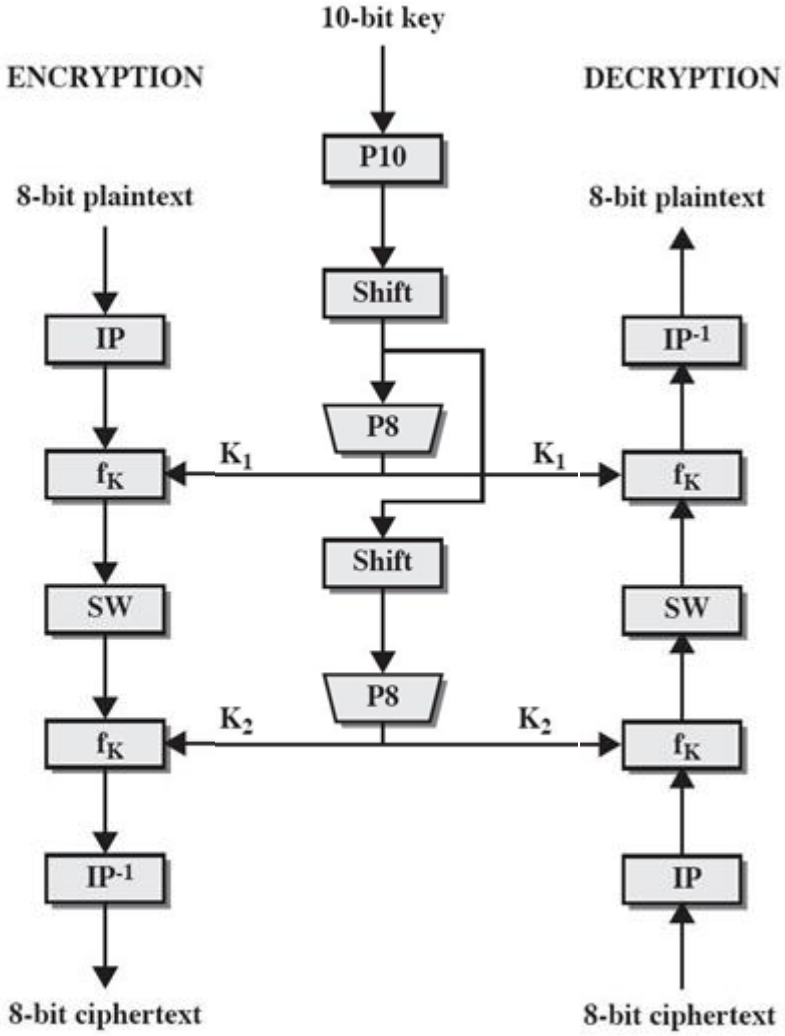
Simplified-DES

- It is simple version of DES.
- It has similar properties and structure to DES with smaller parameters.
- It is a **block cipher**.
- It has **8-bits block size** of plain text or cipher text.
- It uses **10-bits key size** for encryption.
- It has **Two Rounds**.

Simplified-DES Algorithm

- **The S-DES encryption algorithm takes**
  - An **8 bit block** of **plaintext** (eg. 10111101)
  - A **10 bit key** as input
  - Produces an **8 bit block** of **ciphertext**
- **The S-DES decryption algorithm takes**
  - An **8 bit block** of **ciphertext**
  - The **same 10 bit key** used
  - Produces an **8 bit block** of **plaintext**

S-DES Overall Structure



S-DES Encryption

- **The Encryption Algorithm as a composition of functions:**

$$\text{Ciphertext} = IP^{-1}(f_{k2}(SW(f_{k1}(IP(\text{Plaintext})))))$$

Where

$$K_1 = P8(Shift(P10(Key)))$$

$$K_2 = P8(Shift(Shift(P10(Key))))$$

S-DES Rules

- Initial Permutation (IP)
- Inverse Initial Permutation (IP^{-1})
- Permutation (P10)
- Permutation (P8)
- Expansion/Permutation (E/P(F))
- Permutation (P4)

IP							
2	6	3	1	4	8	5	7

IP^{-1}							
4	1	3	5	7	2	8	6

P10									
3	5	2	7	4	10	1	9	8	6

P8							
6	3	7	4	8	5	10	9

E/P							
4	1	2	3	2	3	4	1

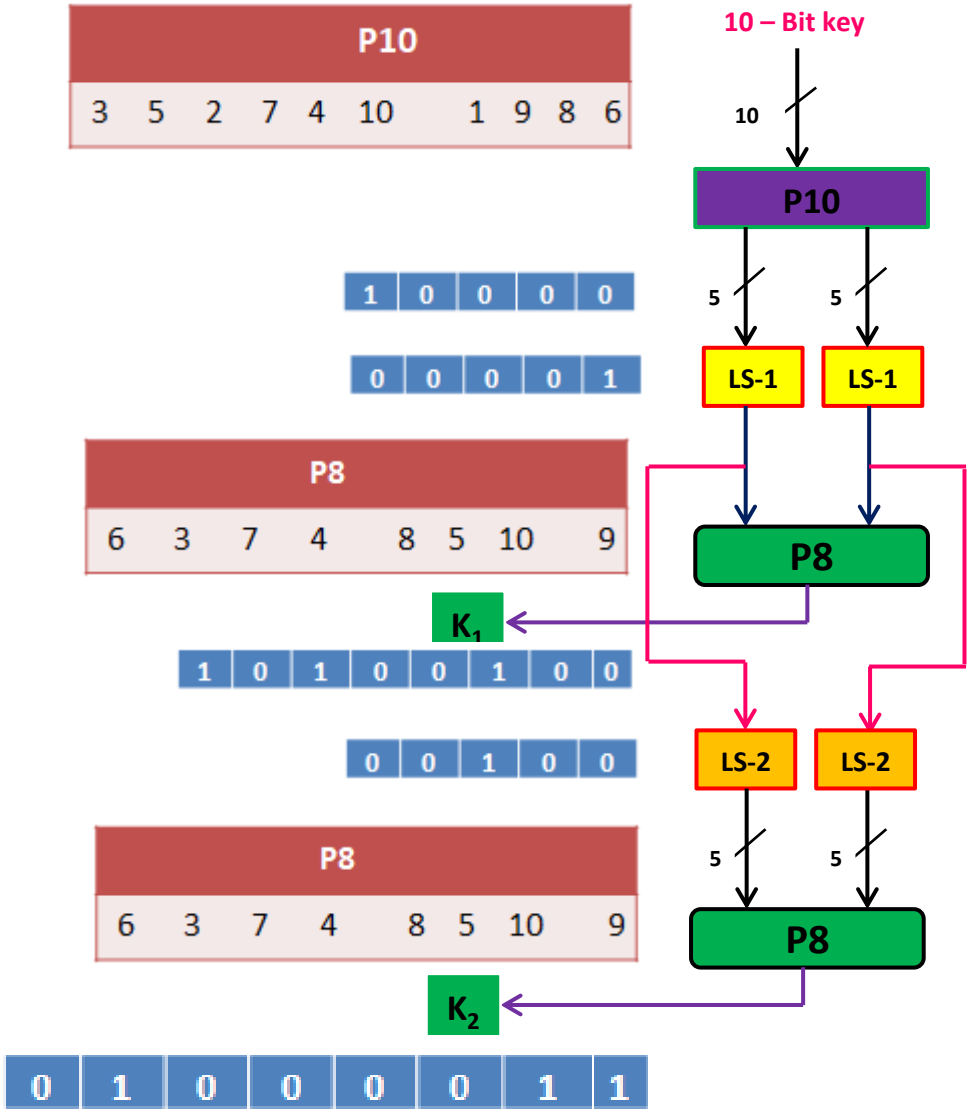
P4			
2	4	3	1

S-DES - 10 bit Sub-Key Generation

Sub-Keys Generation Procedure:

1. **Randomly select 10 bit key.**
2. **Apply P10 rule on selected 10 bit key.**
3. **Split key into two equal halves, circular left shift.**
4. **Apply P8 rule on resultant bits, K_1 generated.**
5. **Circular left shift twice.**
6. **Apply P8 rule on resultant bits, K_2 generated.**

S-DES - 10 bit Sub-Key Generation



Order	1	2	3	4	5	6	7	8	9	10
Input	1	0	1	0	0	0	0	0	1	0

Order	3	5	2	7	4	10	1	9	8	6
Output	1	0	0	0	0	0	1	1	0	0

0	1	1	0	0

1	1	0	0	0

Order	1	2	3	4	5	6	7	8	9	10
Input	0	0	0	0	1	1	1	0	0	0

Order	6	3	7	4	8	5	10	9
Output	1	0	1	0	0	1	0	0

0	0	0	1	1

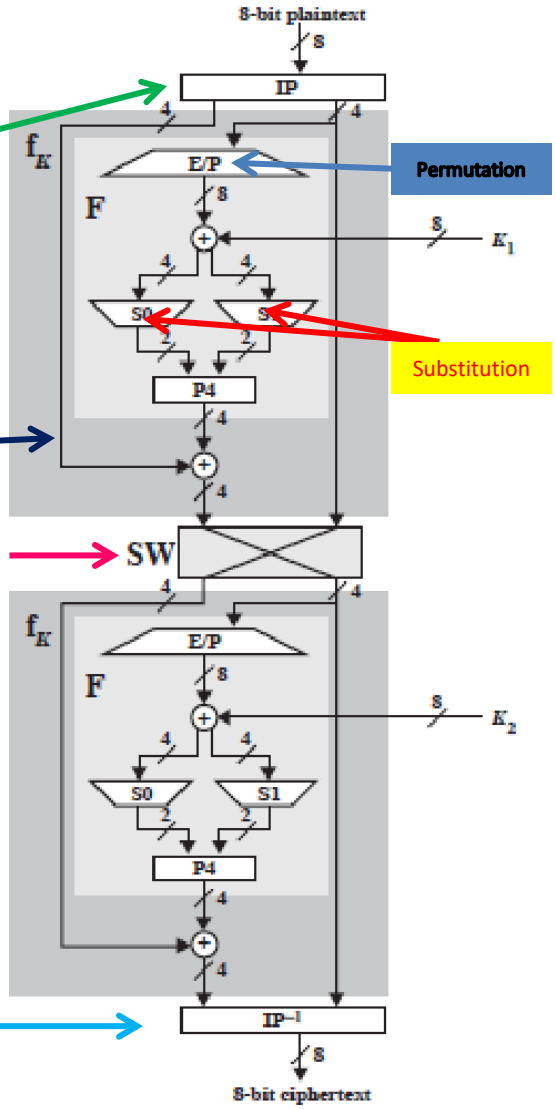
Order	1	2	3	4	5	6	7	8	9	10
Input	0	0	1	0	0	0	0	0	1	1

Order	6	3	7	4	8	5	10	9
Output	0	1	0	0	0	0	1	1

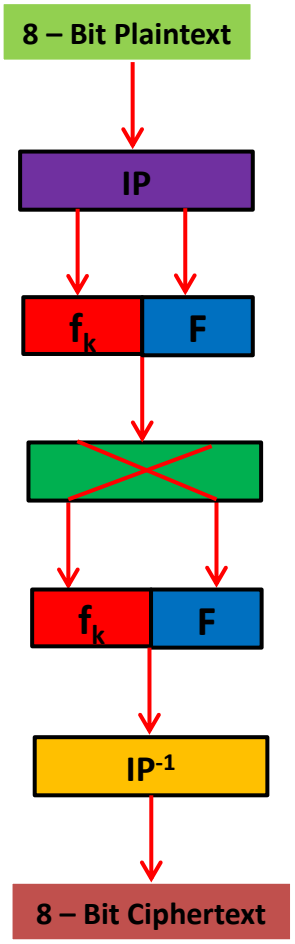
S-DES - Two Rounds Encryption Steps

The Encryption Algorithm involves 5 Steps:

1. **An initial permutation (IP)**
2. **A complex function (F), which includes permutation and substitution operations**
3. **The function (f_k)**
4. **Permutation function that switches (SW) the two halves of the data**
5. **Step 2 and Step 3 repeat in second round**
6. **Finally a permutation function that is the inverse of the initial permutation (IP^{-1})**



S-DES - Two Rounds Encryption



S-DES – 8 bit Encryption

IP							
2	6	3	1	4	8	5	7

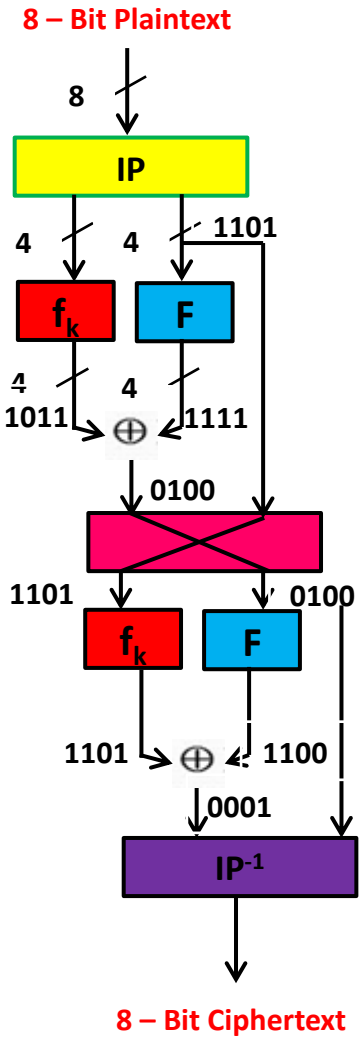
LEFT			
1	0	1	1

Switch	1	1	0	1	0	1	0	0

LEFT			
1	1	0	1

RIGHT			
0	1	0	0

IP^{-1}							
4	1	3	5	7	2	8	6



Order	1	2	3	4	5	6	7	8
Input	1	1	1	1	0	0	1	1

Order	2	6	3	1	4	8	5	7
Output	1	0	1	1	1	1	0	1

RIGHT			
1	1	0	1

$f_k(L,R) = (L \oplus F(R, SK), R)$

$f_k(1011, 1101) = (1011 \oplus F(R, SK), 1101)$

F(R,SK) = 1111

$f_k(1011,1101) = (1011 \oplus 1111, 1101) = 0100, 1101$

Output	0	1	0	0	1	1	0	1

Output	0	1	1	0	1	0	1	1

Order	4	1	3	5	7	2	8	6
Output	1	1	0	0	0	0	0	1

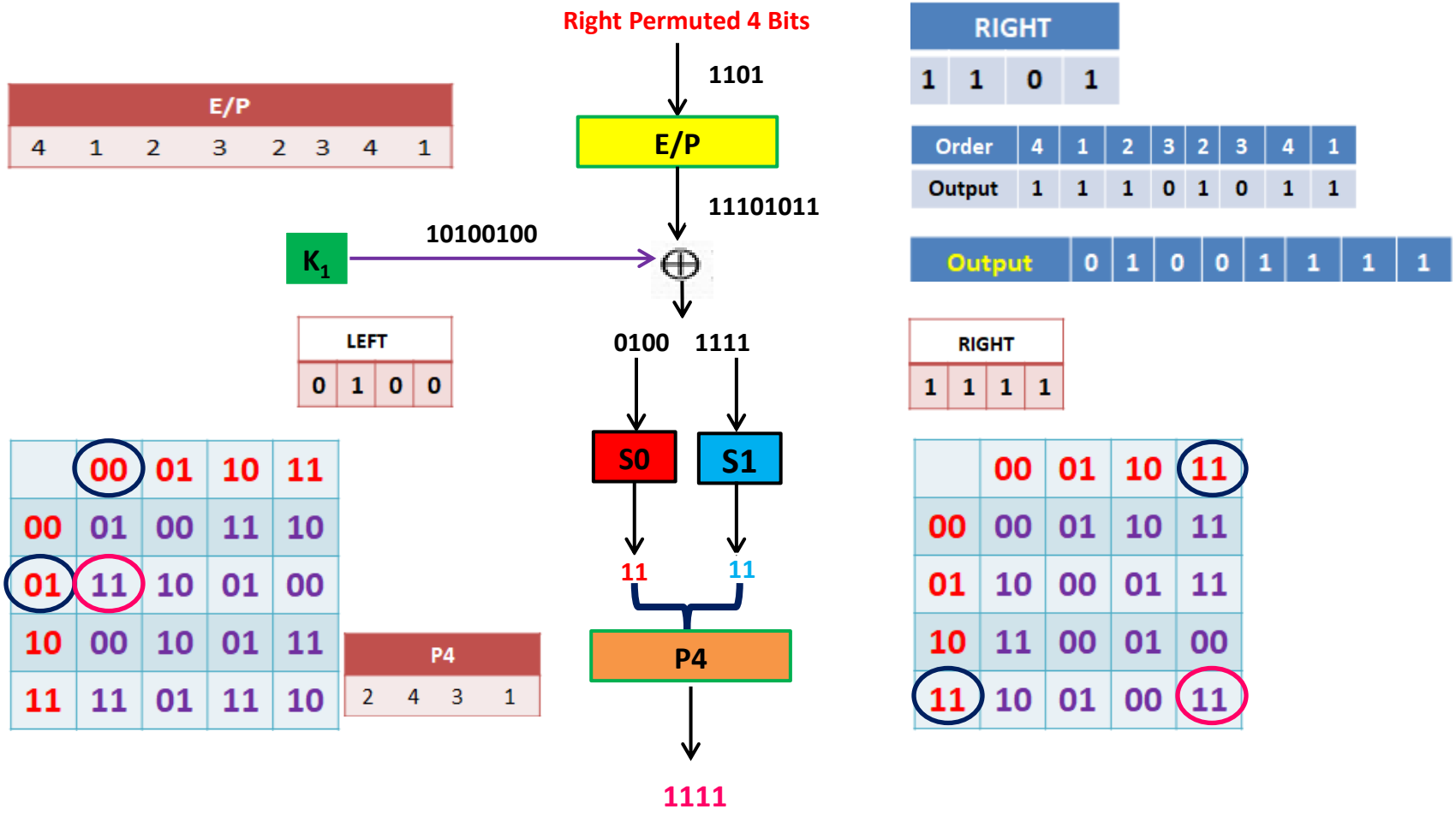
S-DES – 8 bit Expansion/Permutation

F function Process:

How will get F(R,SK) = 1111?

1. **Take right 4 bits of IP (1101).**
2. **Apply E/P rule on right 4 bits of IP.**
3. **XoR with sub-key K1.**
4. **Divide output into two equal parts (Left & Right).**
5. **Use substitution boxes(S-Boxes) S0 &S1 on Left & Right.**
6. **The 4 bits produced by S0 and S1 undergo permutation P4.**
7. **The output of P4 is the output of the function F.**

S-DES – 8 bit One Round
Expansion/Permutation



S-DES Decryption

- **The Decryption Algorithm as a composition of functions:**

$$\text{Plaintext} = IP^{-1}\left(f_{k1}\left(SW\left(f_{k2}\left(IP\left(\text{Ciphertext}\right)\right)\right)\right)\right)$$

Where

$$K_1 = P8(Shift(P10(Key)))$$

$$K_2 = P8(Shift(Shift(P10(Key))))$$

S-DES Example

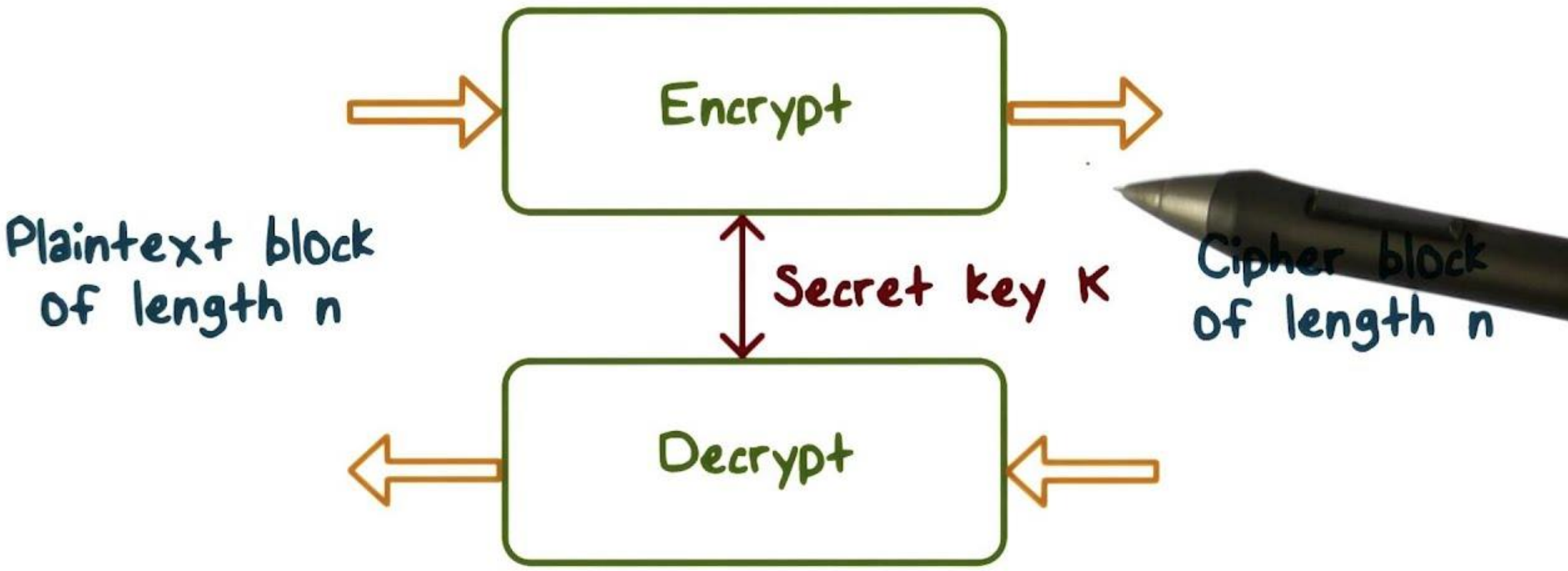
1. Key : 1010000010
   Rules : Same rules
   Find K1 and K2 : ?

2. Plaintext : GITAM (Convert into binary)
   Key : 1010110001
   Rules : Same rules
   Ciphertext : ?

S-DES Strength

- Along with permutations and substitutions, S-boxes making cryptanalysis difficult.
- A brute-force attack on simplified DES is certainly feasible. With a 10-bit key, there are only $2^{10} = 1024$ possibilities.
- Given a ciphertext, an attacker can try each possibility and analyze the result to determine if it is reasonable plaintext.

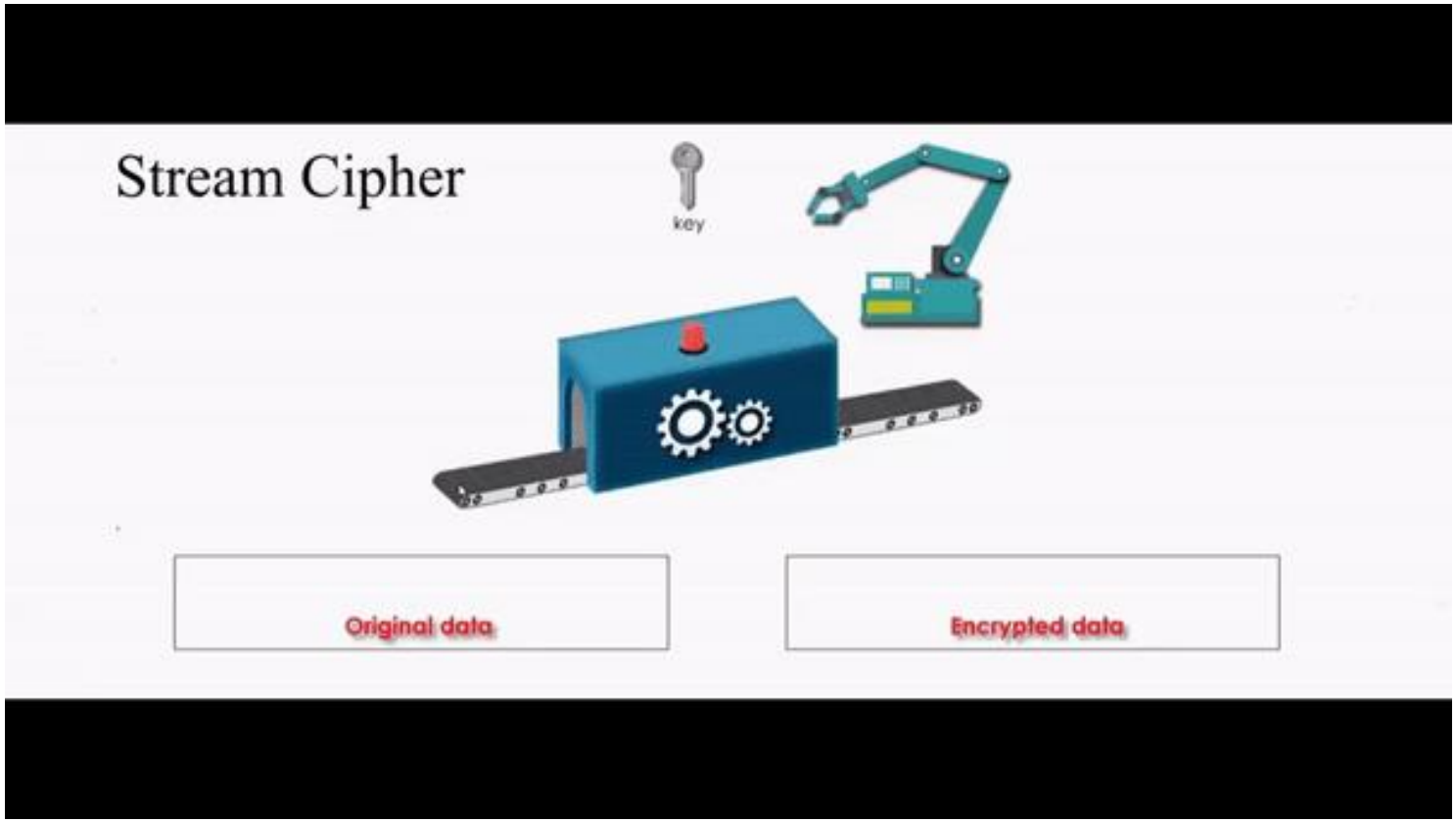
Block Cipher Scheme



Block Cipher Principles

Stream Cipher Vs Block Cipher

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.



Block Cipher Transformation

Reversible

- **Reversible or nonsingular: There are 2^n possible different plaintext blocks each must produce a unique ciphertext.**
- **If n = 2**

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible

- A ciphertext could have been produced by one or two plaintext blocks.

Irreversible Mapping	
Plaintext	Ciphertext
00	11
01	10
10	01
11	01

- So if restrict to reversible mappings, the number of different transformations is 2^n !
- Key size = $n * 2^n$

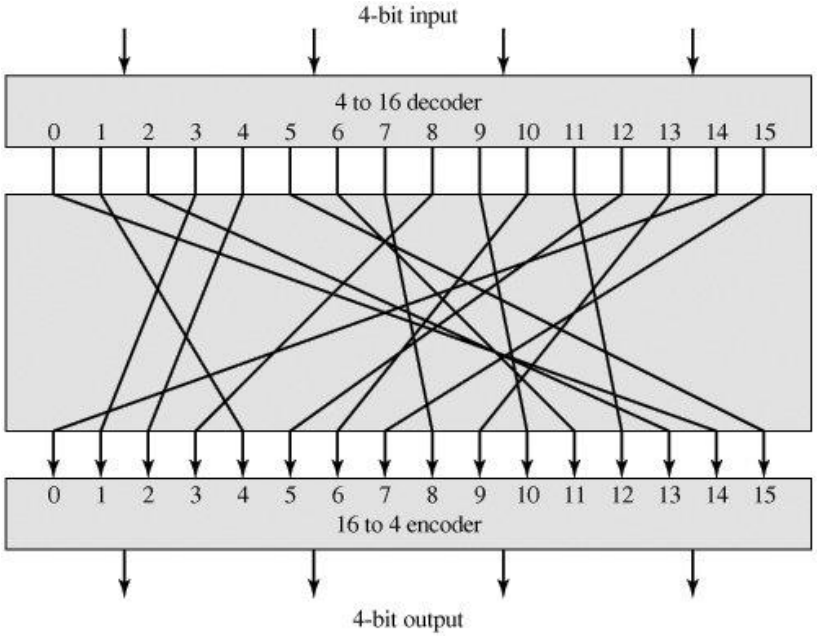

Ideal Block Cipher

Ideal Block Cipher

- Ideal block cipher define reversible mapping between plaintext and ciphertext.
- But there is a practical problem with this approach.
- If a small block size, then it is equivalent to classical substitution cipher.
- As we discuss early, such substitution cipher vulnerable to a statistical analysis.
- For a large block size is not practical.

Ideal Block Cipher

- If consider n = 4 bits, then 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.
- The mapping itself a key.



Ideal Block Cipher

Encryption

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

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

For n bits substitution block cipher the key size is $n * 2^n$.

For 4 bits block the key require 64 bits.

This is vulnerable to a statistical analysis.

For 64 bits block the key require $64 * 2^{64} = 2^{70} = 10^{21}$.

This is a desirable length for statistical attack but impractical.


The Feistel Cipher

The Feistel Cipher

- Feistel proposed the concept of a product cipher, which makes cryptographically strong.
- Most symmetric block ciphers are based on a Feistel Cipher Structure.
- A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits.
- The Feistel Cipher is a practical application of a proposal by Claude Shannon.

The Feistel Cipher

- **The idea of a product cipher that alternates substitution-permutation (S-P).**
- **Substitution-Permutation based on the two primitive cryptographic operations:**

  ***substitution* (S-box)**

  ***permutation* (P-box)**
- **Feistel introduces to two functions to make difficult to statistical cryptanalysis:**

  Diffusion

  Confusion

The Feistel Cipher

Diffusion:

The mechanism of diffusion seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible.

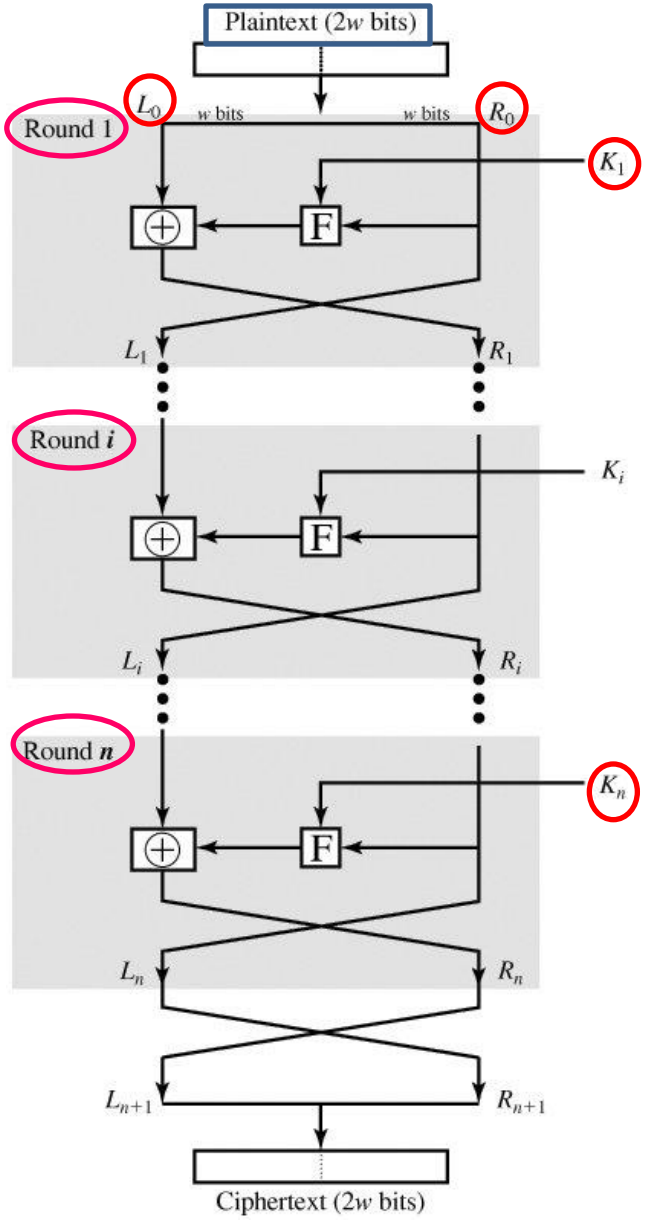
It encrypts a message M = $m_1, m_2, m_3, \ldots\ldots$

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

Confusion:

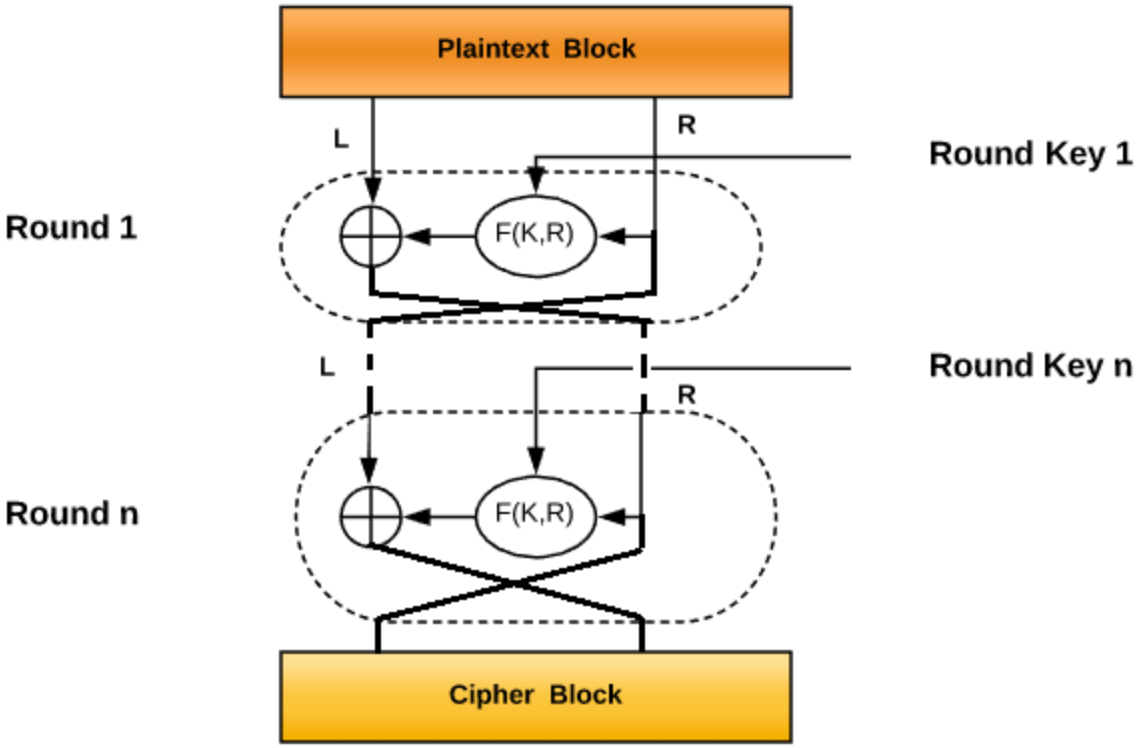
The mechanism of confusion seeks to make the statistical relationship between the ciphertext and key as complex as possible.

The Feistel Cipher Structure



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

The Feistel Cipher Structure



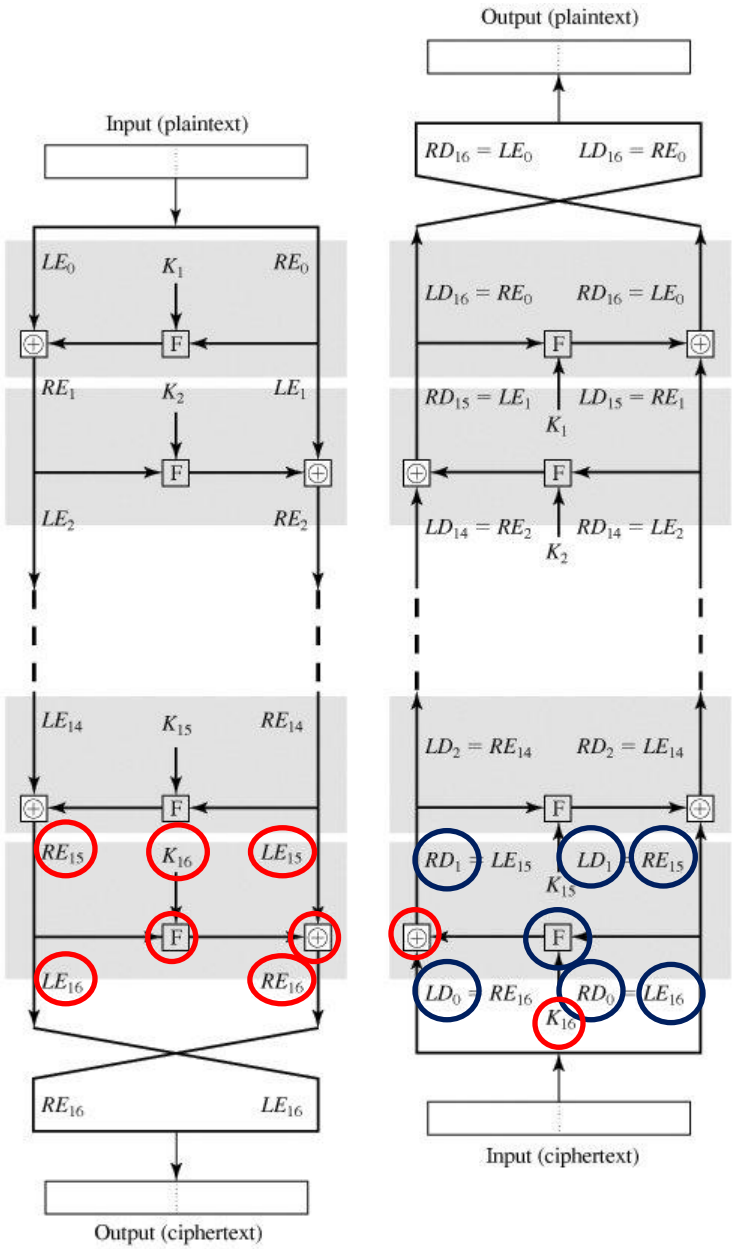
The Feistel Cipher Structure

- All rounds have the same structure.
- A substitution is performed on the left half of the data (as similar to S-DES).
- This is done by applying a round function F to the right half of the data.
- Then taking the XOR of the output of F 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 sub-key k_i.
- Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

The Feistel Cipher Structure - Design Features

- **Block size** - Increasing size improves security
- **Key size** - Increasing size improves security, makes exhaustive key searching harder
- **Number of rounds** - Increasing number improves security
- **Sub-key generation** - Greater complexity can make analysis harder
- **Round function** - Greater complexity can make analysis harder
- **Fast software en/decryption & ease of analysis** - are more recent concerns for practical use and testing.

The Feistel Cipher Encryption & Decryption



First consider the encryption process,

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \text{ XoR } F(RE_{15}, K_{16})$$

Decryption Process,

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \text{ XoR } F(RD_0, K_{16})$$

$$= RE_{16} \text{ XoR } F(RE_{15}, K_{16})$$

$$= LE_{15} \text{ XoR } F(RE_{15}, K_{16}) \text{ XoR } F(RE_{15}, K_{16})$$

- Therefore, $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$
- In general, for the i^{th} iteration of the encryption algorithm,

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \text{ XoR } F(RE_{i-1}, K_i) = RE_i \text{ XoR } F(LE_i, K_i)$$

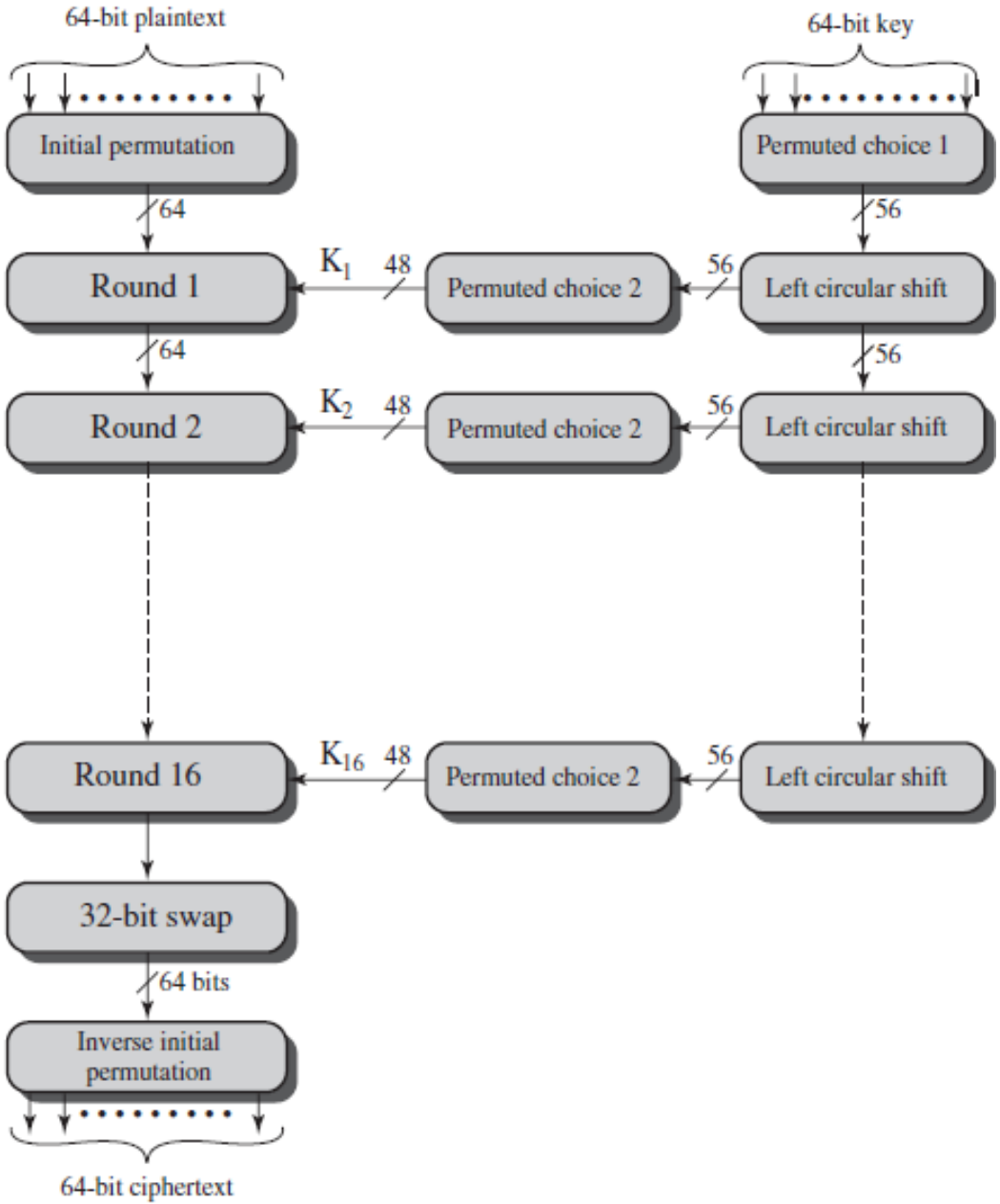
- Finally, the output of the last round of the decryption process is $RE_0 \,||\, LE_0$ A 32-bit swap recovers the original plaintext.


DES
Data Encryption Standard

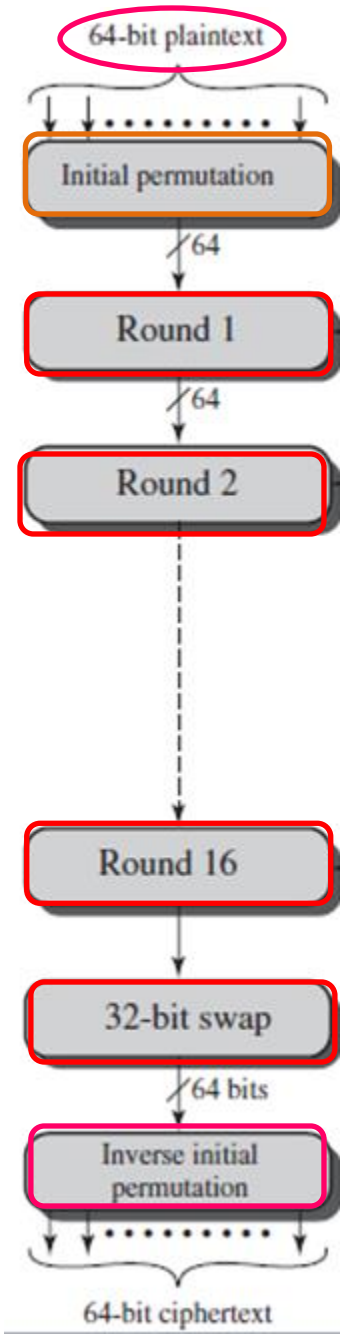
DES History

- Most widely used block cipher in world
- Adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- Encrypts 64-bit data using 56-bit Key
- IBM developed Lucifer cipher
  - by team led by Feistel
  - used 64-bit data blocks with 128-bit key
- Then redeveloped as a commercial cipher with input from NSA and others
- In 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES
- DES has become widely used, especially in financial applications

DES Encryption Overview



DES Encryption



- 64 – bit Plaintext
- Initial Permutation (IP)
- 16 rounds with 16 sub-keys
- Swapping
- Inverse Initial Permutation (IP^{-1})

DES Rules

- **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

- **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

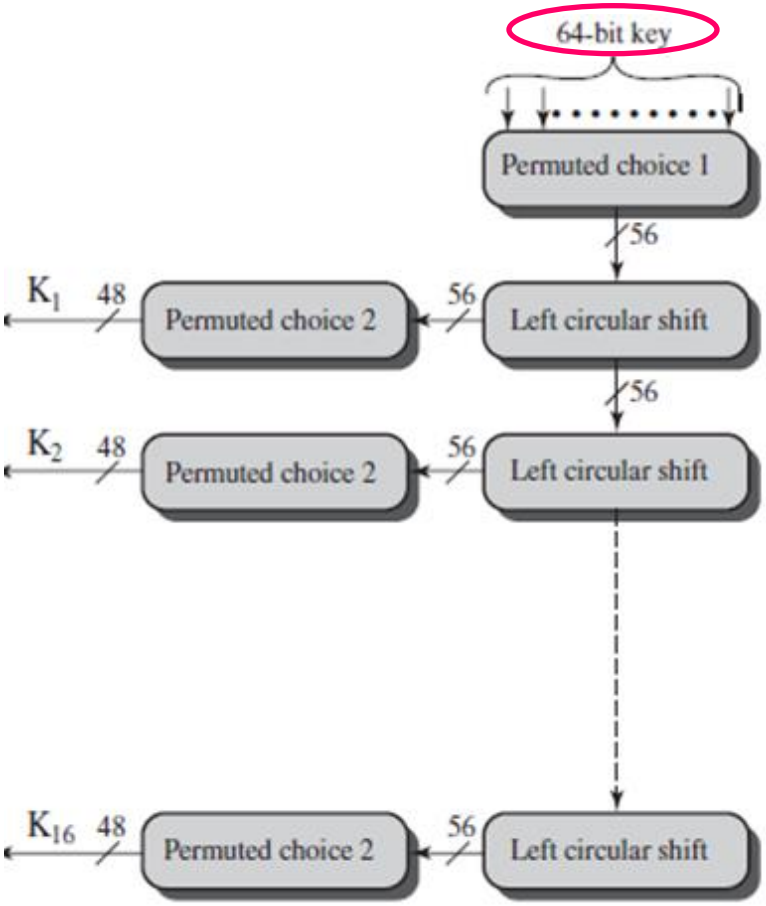
- **Expansion Permutation (E)**

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

- **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 Sub-Key Generation



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

Odd parity bit

In DES, the conversion to 56 bits is done by neglecting every 8th bit.

DES Sub-Key Generation Rules

- **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

- **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

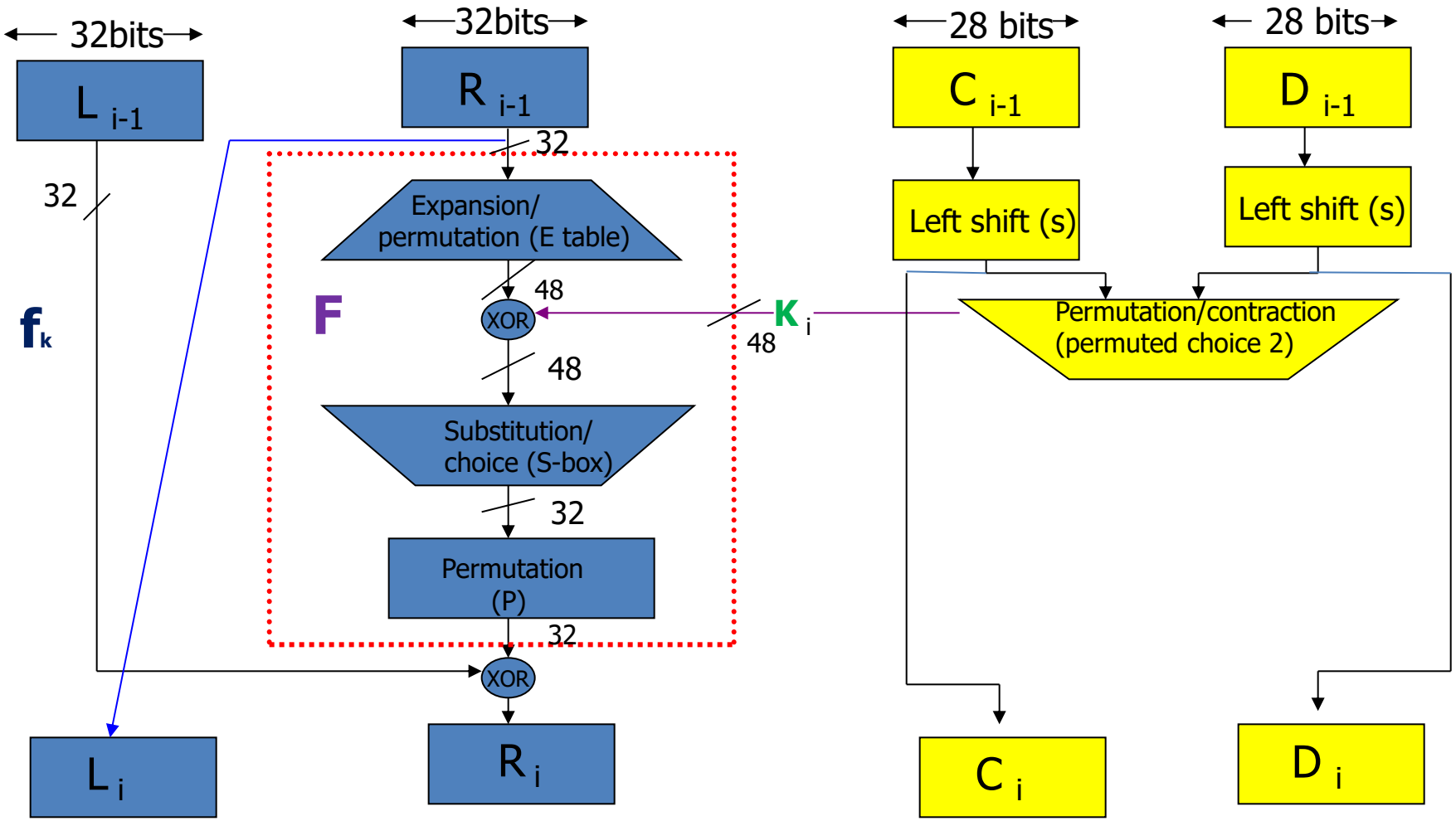
- **Schedule of circular Left Shifts**

# of Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotation	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Sub-Key Generation

- 64 – bit key used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored.
- The resulting 56 bit key divide into two equal half's labeled as C_0 ,D_0 after the permutation done through Permuted Choice - 1.
- At each round, apply circular left shift as governed by rule Schedule of Left Shifts.
- After circular left shift serve as input to Permuted Choice – 2.

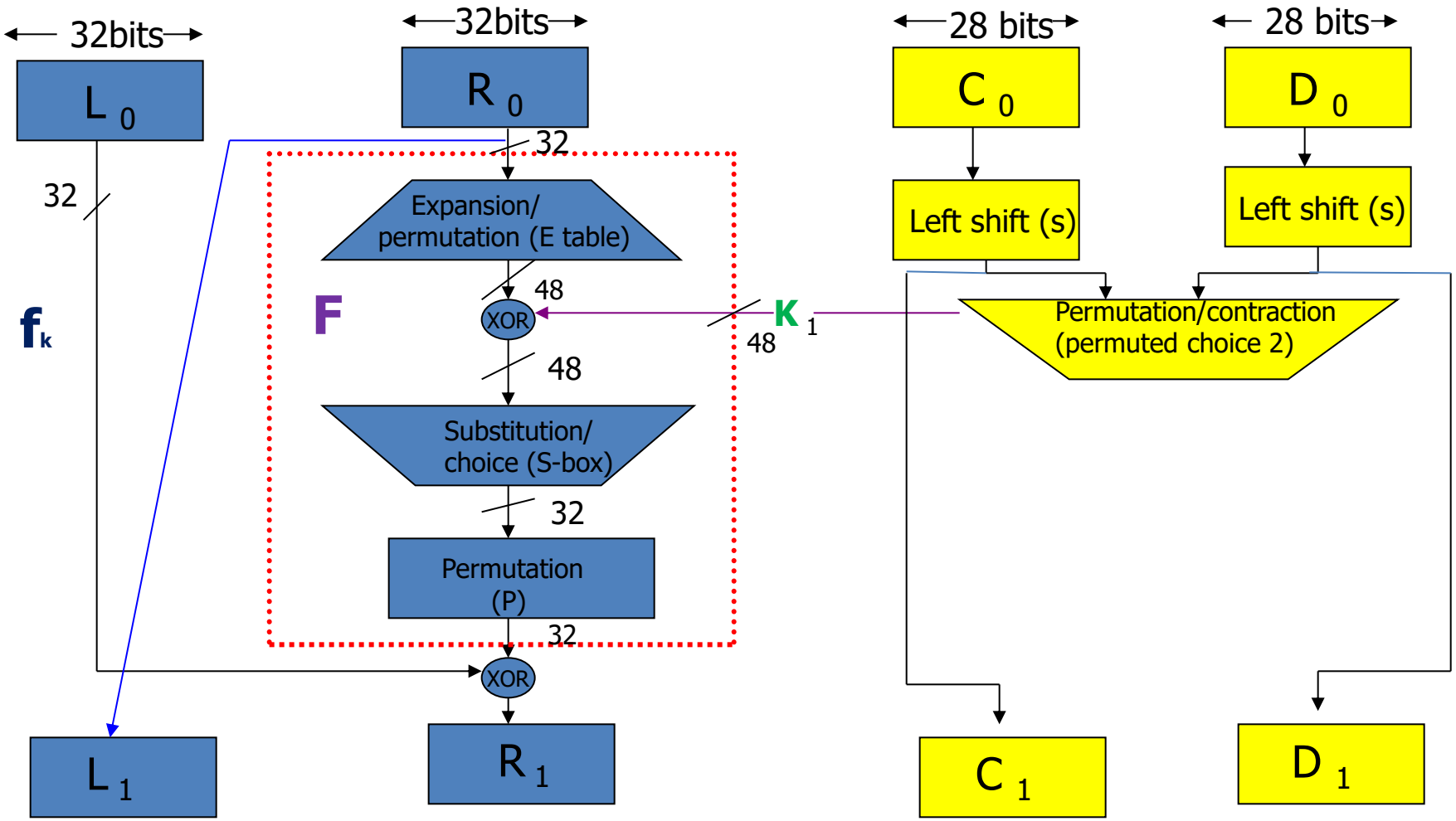
Structure of Single Round of DES Algorithm



$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$

First Round (i=1) of DES Algorithm



$L_1 = R_{1-1} = R_0$

$R_1 = L_{1-1} \oplus F(R_{1-1}, K_1) = L_0 \oplus F(R_0, K_1)$

IP & IP^{-1}

- IP(plaintext) = L(0): R(0), where L(0) and R(0) are the left –half and the right-half of the output after the permutation done through 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

- Even bits allocated to L(0)
- Odd bits allocated to R(0)
- IP does not add to security.
- It makes the function a little complex.

Inverse Input Permutation reverses it back.

Thus $IP^{-1}(IP(X)) = X$

Function: Expansion/Permutation

Expansion/Permutation:

- To get 48 bits from 32 bits of R_i
- Each input block of 4 bits *contributes* 2 bits to each output block.

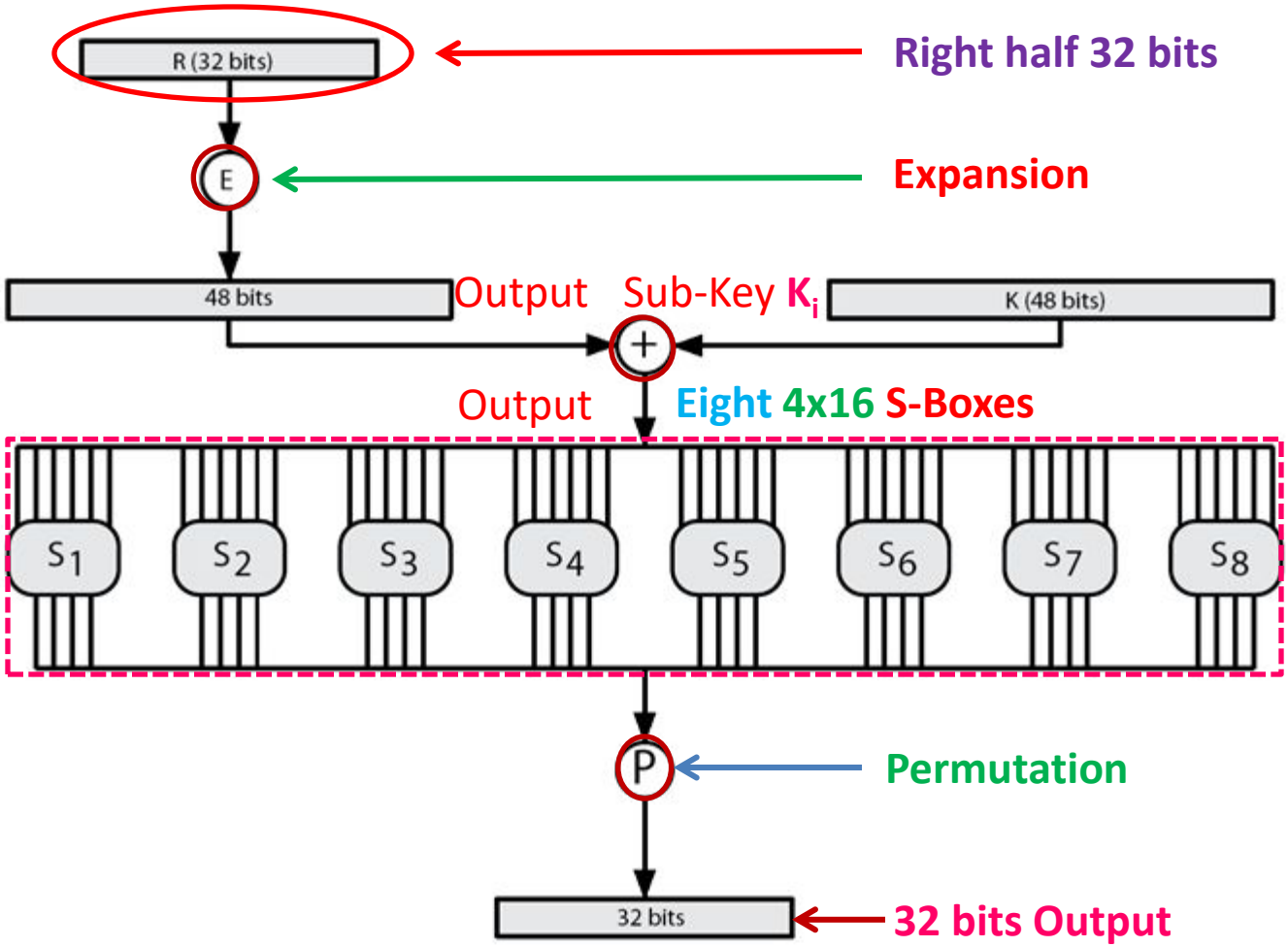
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- $E(R_{i-1}) \oplus K_i$

Function: S-Box Permutation

- After generating 48 bits from E/P, XOR with sub-key K1's 48 bits, then get 48 bits as an output.
- This output transferred to S-Box to reduce again to 32 bits.
- S-Box permutation uses eight 4x16 S-boxes for converting 48 bits to 32 bits.
- S-Boxes provide Non-linear output to strength the cipher.
- Divide 48 bits to 8 units of 6 bits each.

Function: S-Box Permutation Structure



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

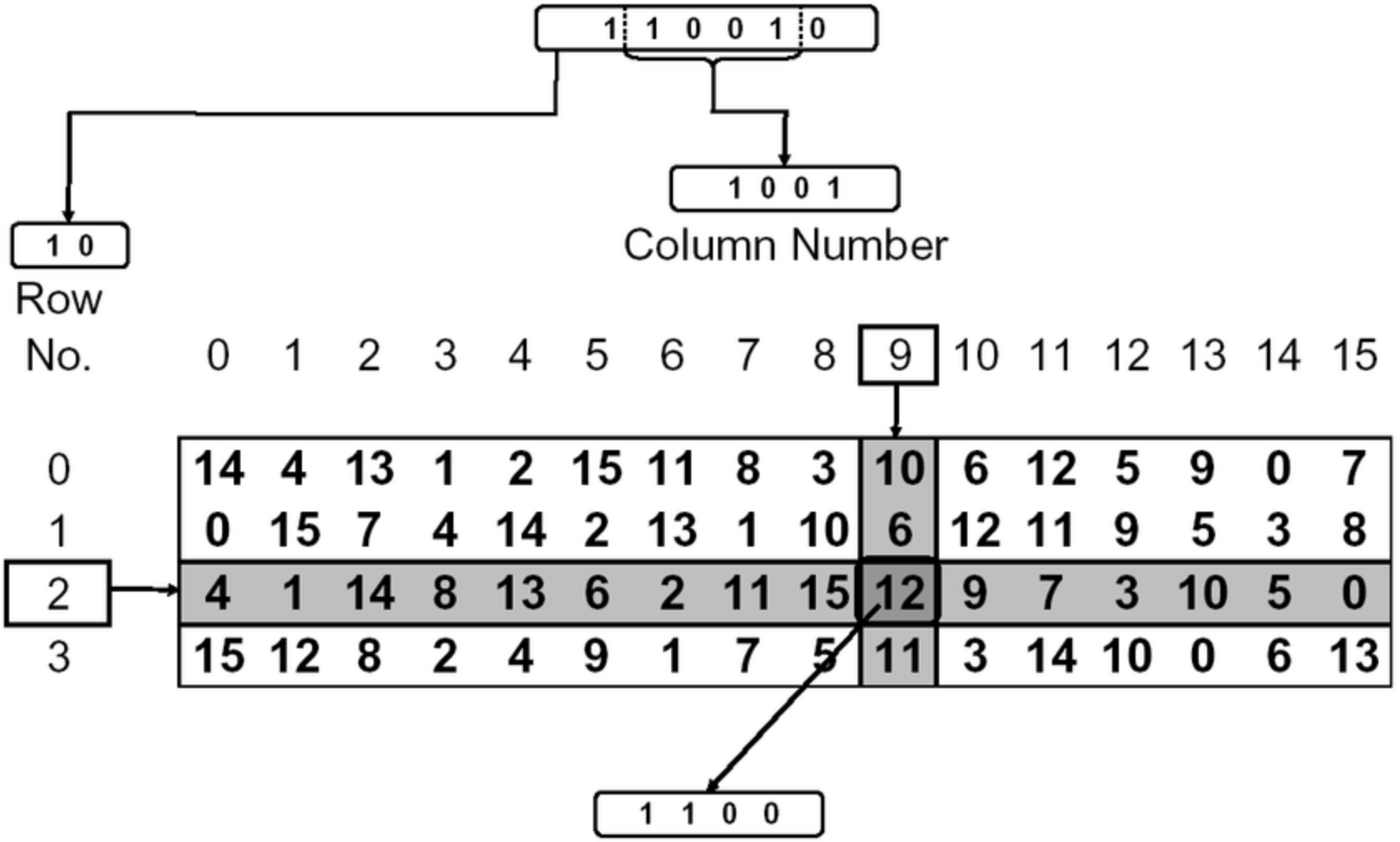
S_8

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

Function: S-Box Permutation Functionality

- Using 8 *4X16* S-boxes, each group of 6 bits is reduced to 4 bits as follows:
  1. For each S-Box: Row Number = Outermost 2 bits; the first and the last bit determine the row #.
  2. For each S-Box: Column Number = Inner 4 bits; the middle 4 bits determine the column #.
  3. Using the row and column number, the S-box yields a decimal number (lying between 0 and 15).
  4. Its 4 bit binary equivalent is the output of the S-box.
- The 32 bit output from the eight S-Boxes is
  1. Permutation (P)
  2. XOR with left half (f_k)
  3. Switch the left half and the right half (Swap)

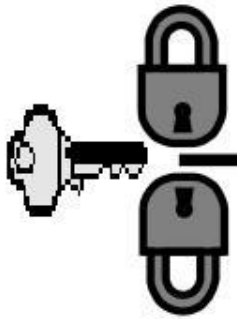
Function: S-Box Permutation Functionality



The Avalanche Effect

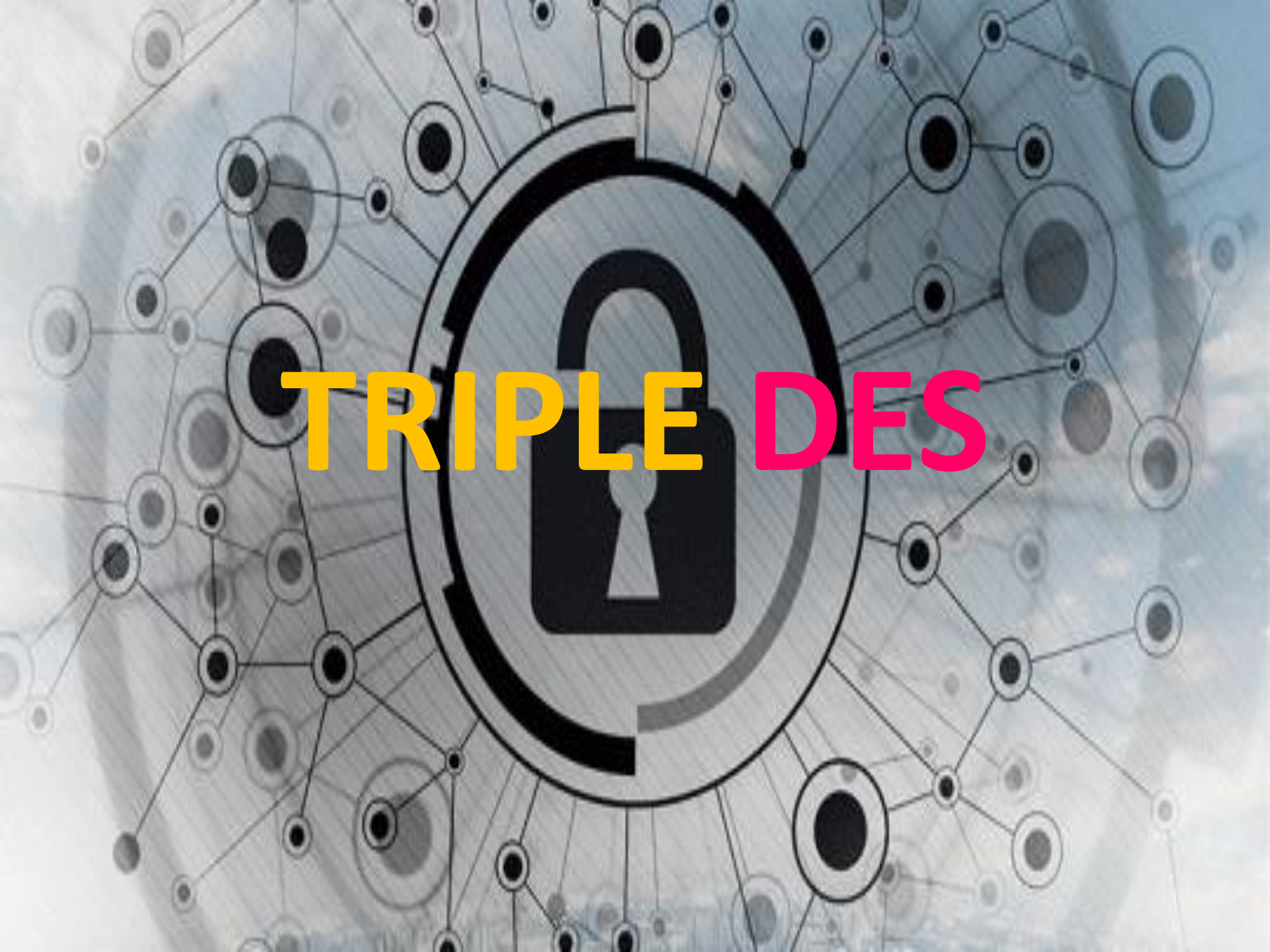
- A small change in plaintext or key results in changing approximate half of the bits in ciphertext.

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35



The Strength of DES

- Concerns about the strength of DES fall into two categories:
  - Concerns about the algorithm itself (nothing so far).
  - Concerns about the use of 56-bit key.
- Electronic Frontier Foundation (EFF) announced that it had broken a new DES encryption using a "DES Cracker" machine for less than $250,000.
- A 128 bit key is guaranteed for unbreakable algorithm by Brute-Force.

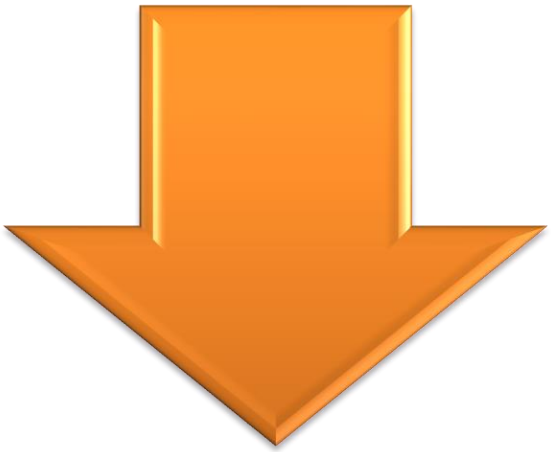

TRIPLE DES

Multiple DES

- DES is vulnerable to a brute force attack (2^{56}).
- Need another alternative, which would preserve the security.
- Multiple encryption with DES and multiple keys is alternative to DES.

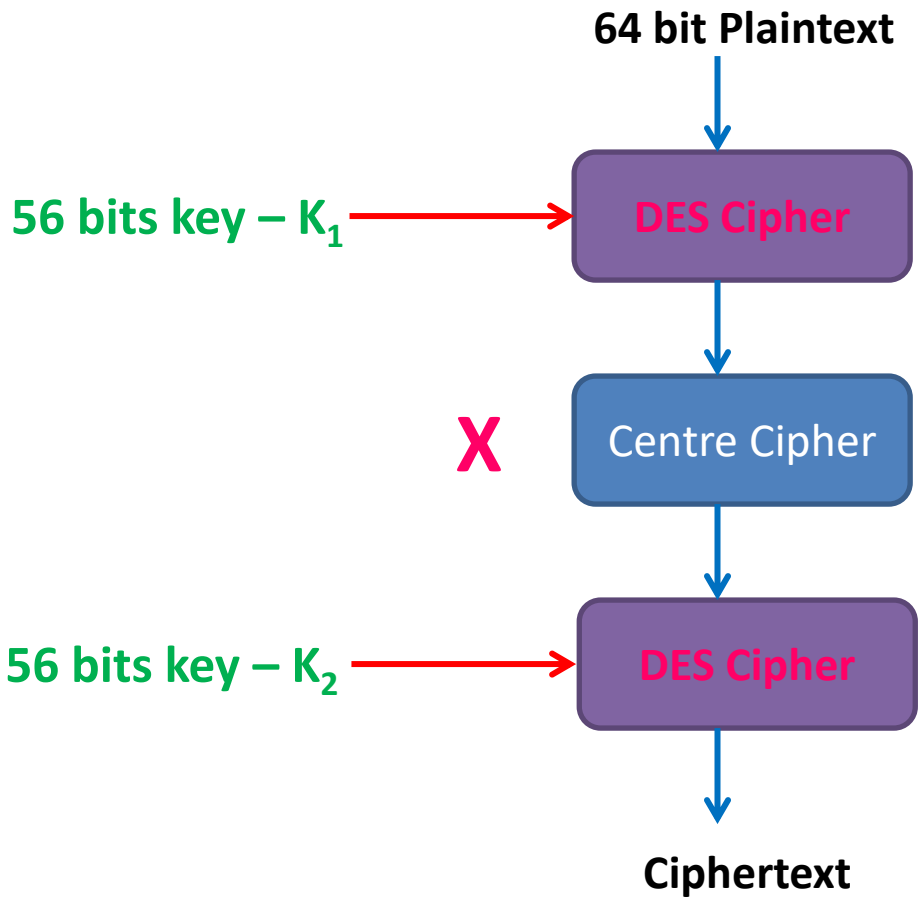
Types of Multiple DES

Double DES



Triple DES

Double DES Encryption



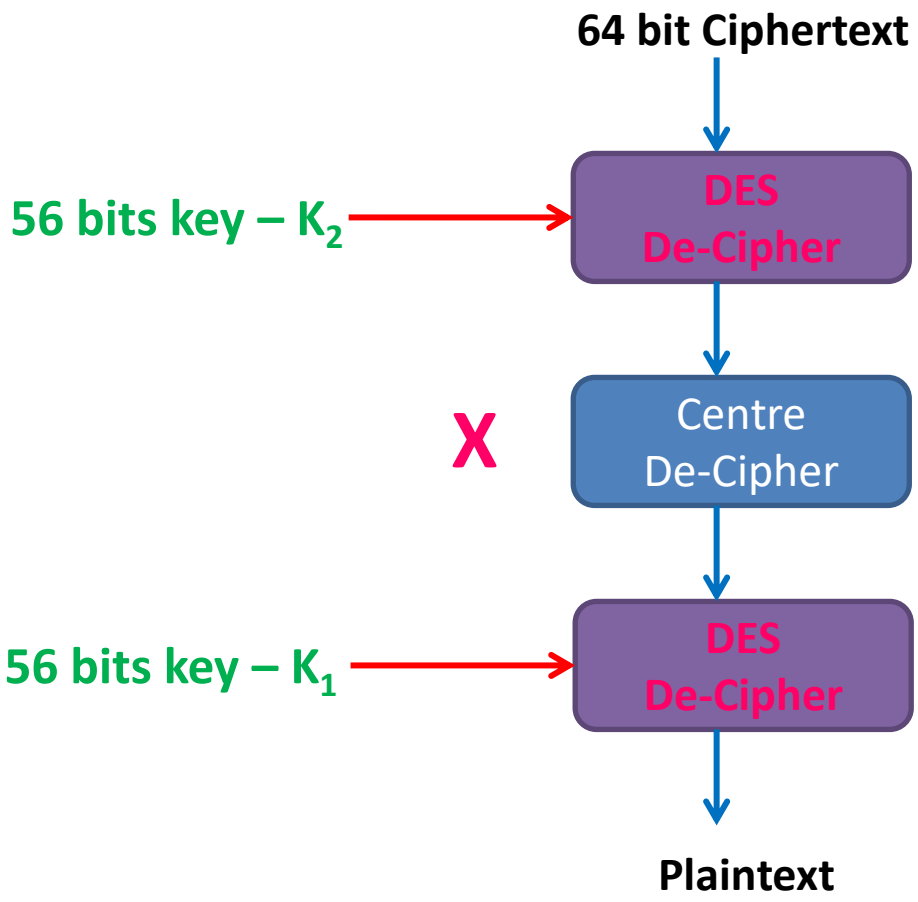
- Given a plaintext P and two encryption keys K_1 & K_2
- Generated ciphertext by

$$C = E_{k2}[E_{k1}[P]]$$

- This approach uses a key length is

56 * 2 = 112 bits.

Double DES Decryption



$$P = D_{k1}\,[D_{k2}\,[C]]$$

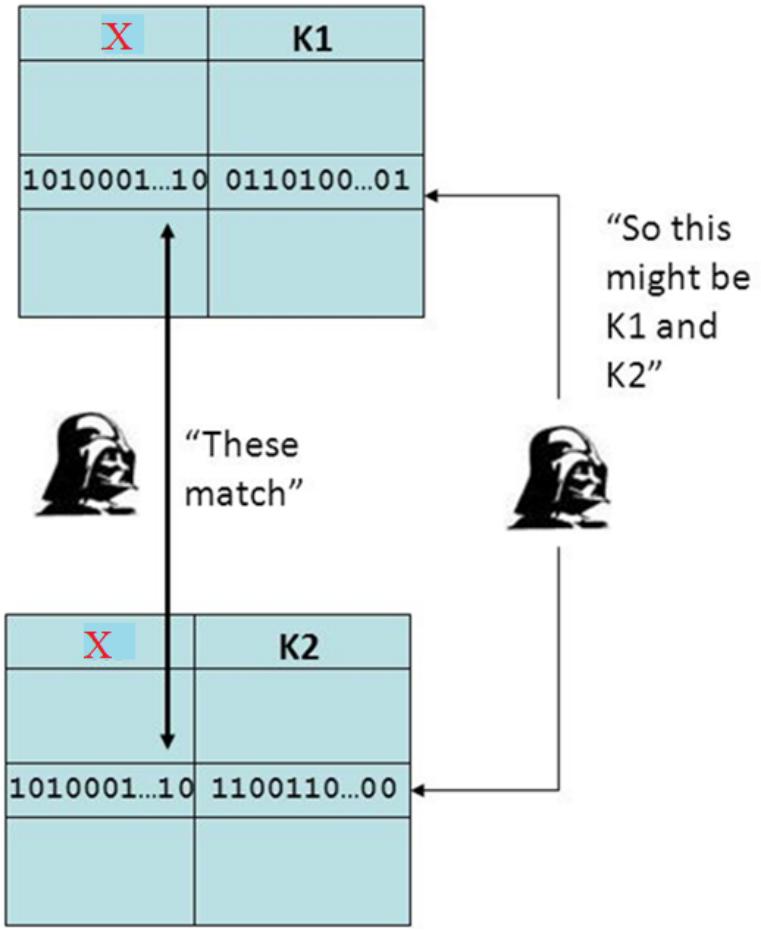
Meet-in-the-middle attack

- This attack requires pair of known plaintext and ciphertext.
- Attack works from both ends of double DES.
- Encrypt from one end and decrypt from another end.
- Cryptanalyst analysis on X, which is center ciphertext.
- X produces the same result, while encrypting plaintext with K_1 and decrypting ciphertext with K_2.

Meet-in-the-middle attack

- The center text is produces by using encryption and decryption

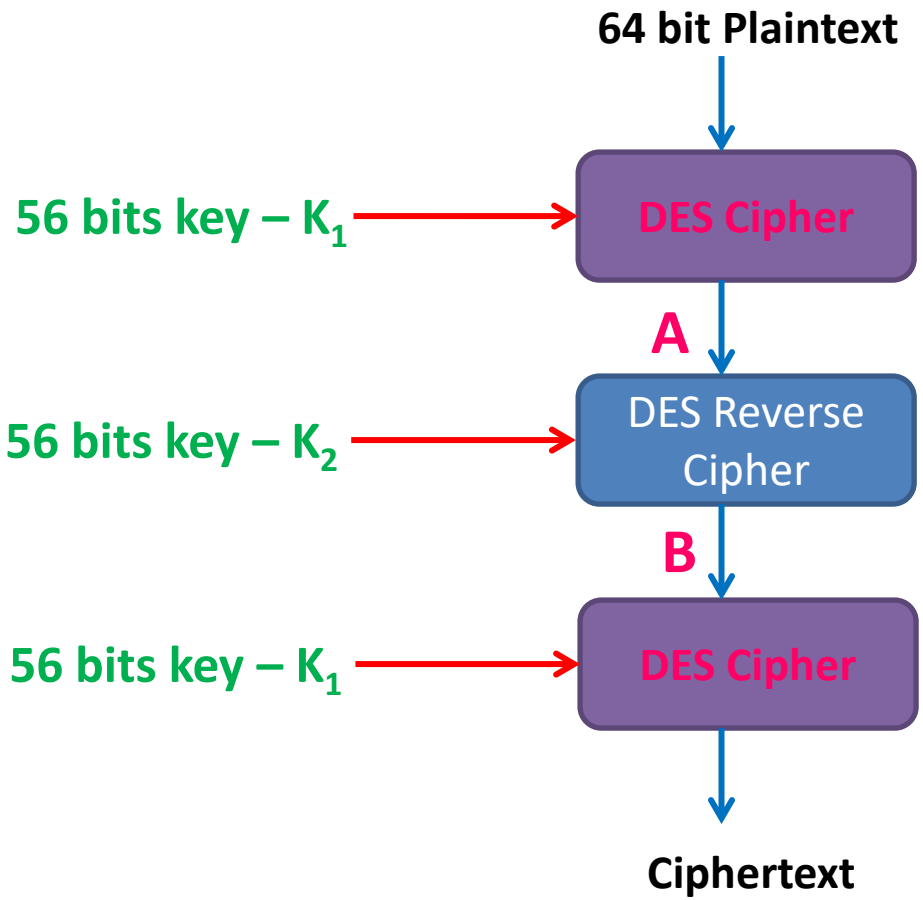
  $X = E_{k1}(P)$ and $X = D_{k2}(C)$
- Encrypt P using all possible values of K_1 and record all values obtained for X.
- Decrypt C using all possible values of K_2 and record all values obtained for X.
- Create two tables sorted by X values.
- Now compare the values for X until find the pairs of K_1 & K_2 for the values of X in both tables.



Triple DES

- **Triple DES is a counter attack to meet-in-the-middle attack.**
- **It uses in two different ways**

Triple DES with Two Keys

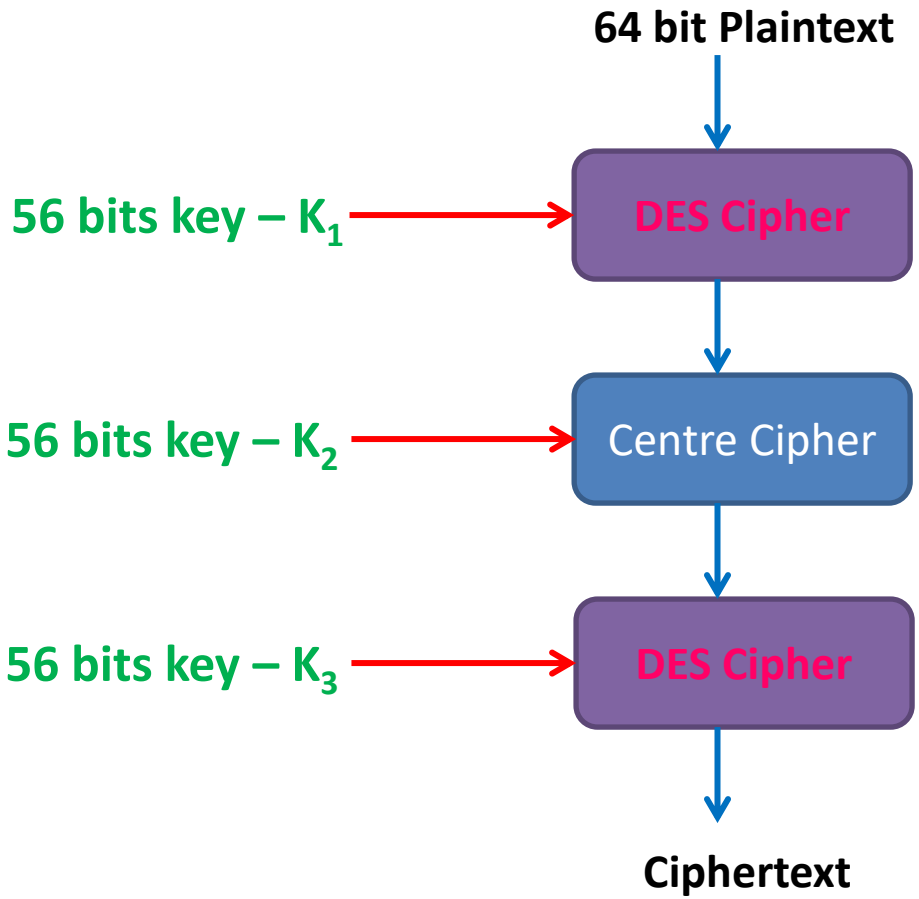


- This approach process Encrypt-Decrypt-Encrypt (EDE).

$$C = E_{k1}[D_{k2}[E_{k1}[P]]]$$

- The Key length is 56 * 3 = 168 bits.
- The drawback of this approach is large key length.

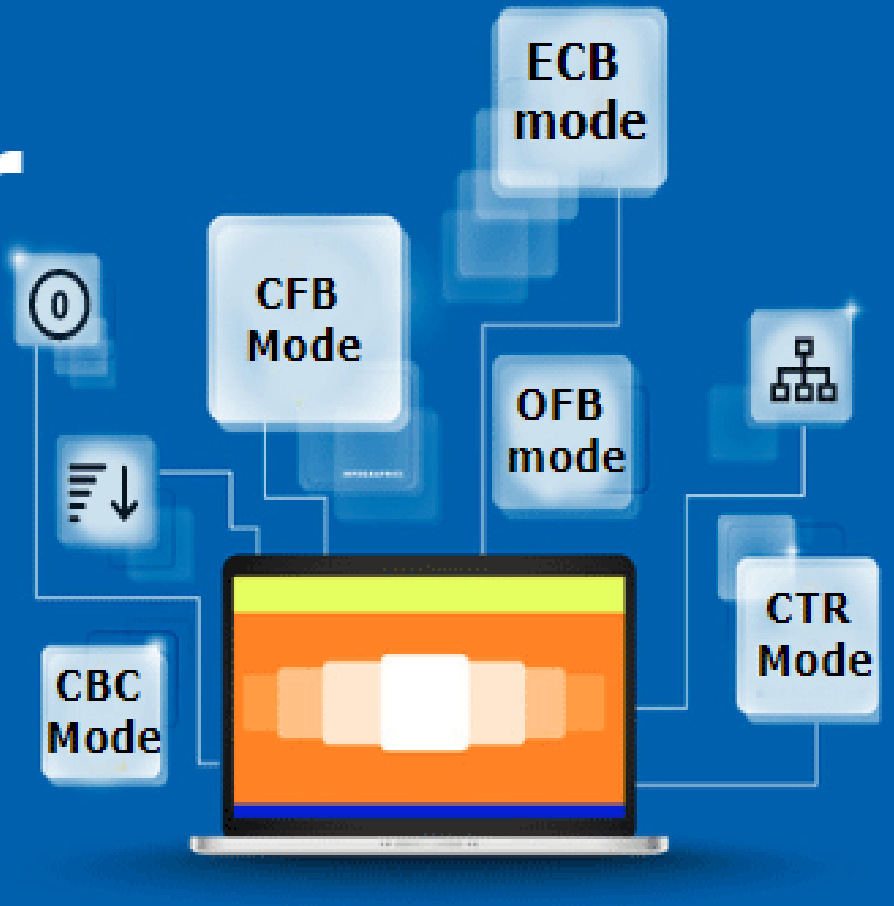
Triple DES with Three Keys



- $C = E_{k3}[D_{k2}[E_{k1}[P]]]$

Modes of Operation

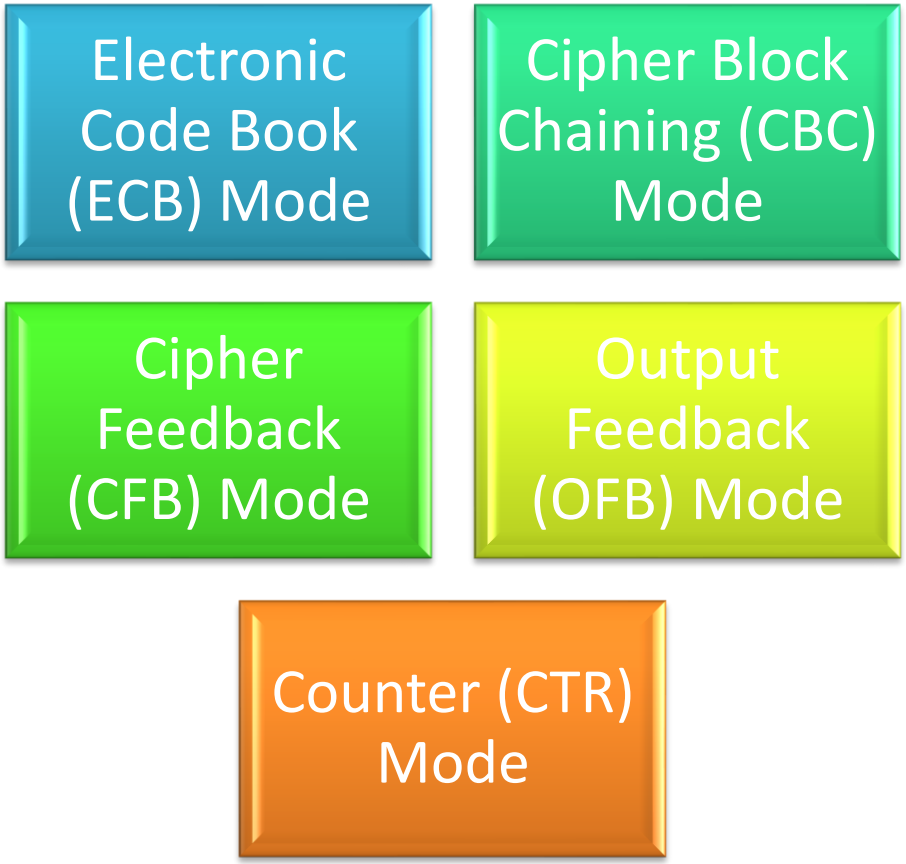
Block Cipher Modes Of Operation



Modes of Operation

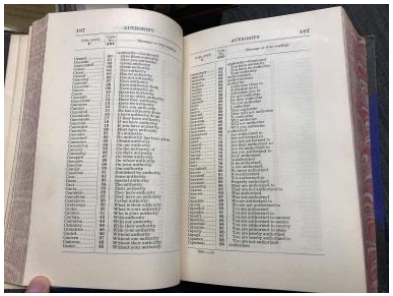
- Block ciphers encrypt fixed size blocks
  - e.g., DES encrypts 64-bit blocks
- Need some way to en/decrypt arbitrary amounts of data in practice
- NIST SP 800-38A defines 5 modes
- Have block and stream modes

Modes of Operation



Electronic Codebook Book (ECB)

- Message is broken into independent blocks that are encrypted.
- Each block is a value which is substituted, like a codebook.



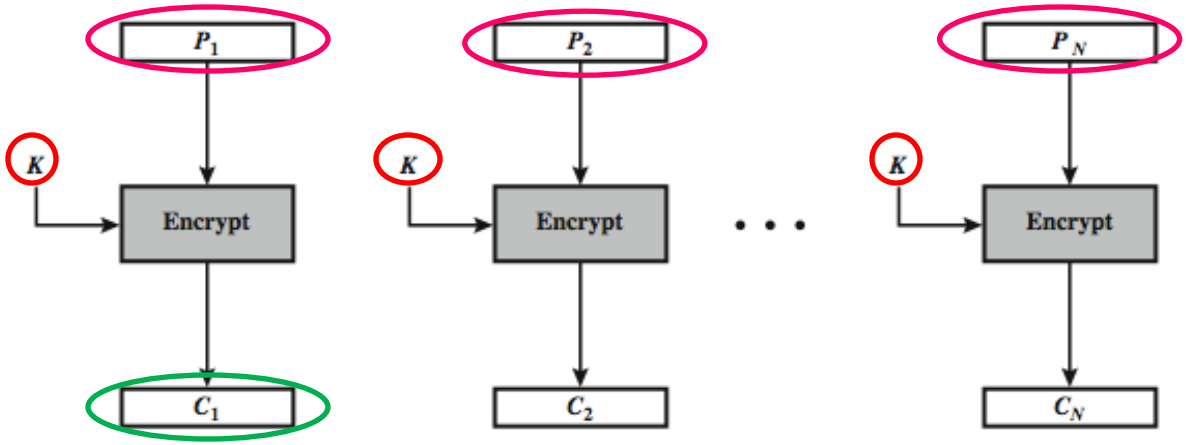
- Each block is encoded independently of the other blocks

$$C_i = E_k(P_i)$$

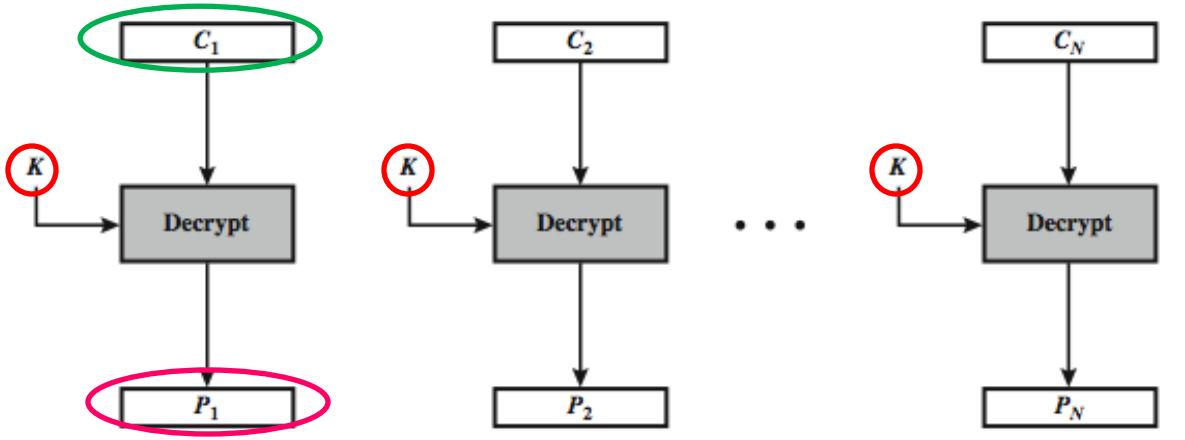
- Secure transmission of single values.

Electronic Codebook Book (ECB)

Plaintext



(a) Encryption



(b) Decryption

Limitations & Advantage of ECB

- Message repetitions may show in ciphertext
- Weakness is due to the encrypted message blocks being independent
- main use is sending a few blocks of data

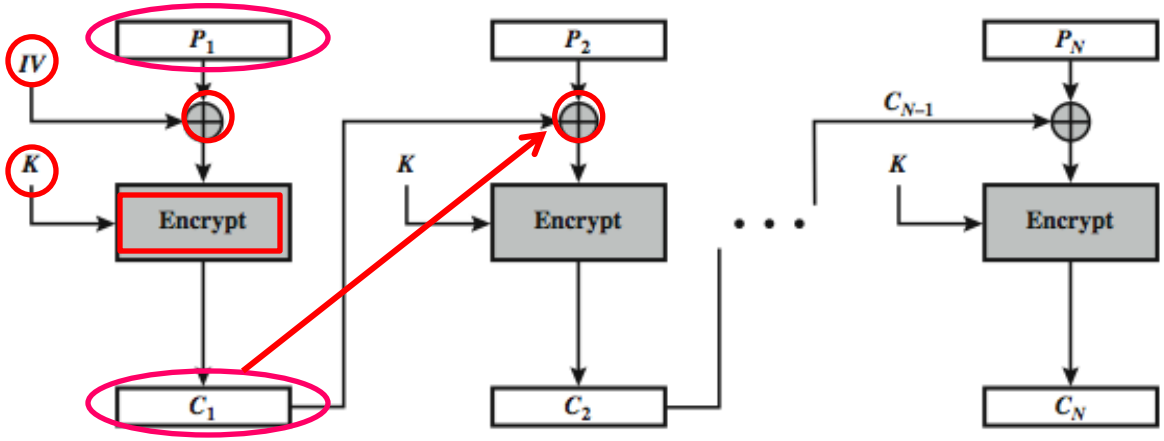
Cipher Block Chaining (CBC)

- Message is broken into blocks
- Linked together in encryption operation
- Each previous cipher block is chained with current plaintext block
- Use Initial Vector (IV) to start process
  - $C_i = E_k(P_i \text{ XOR } IV)$
  - IV prevents same P from making same C

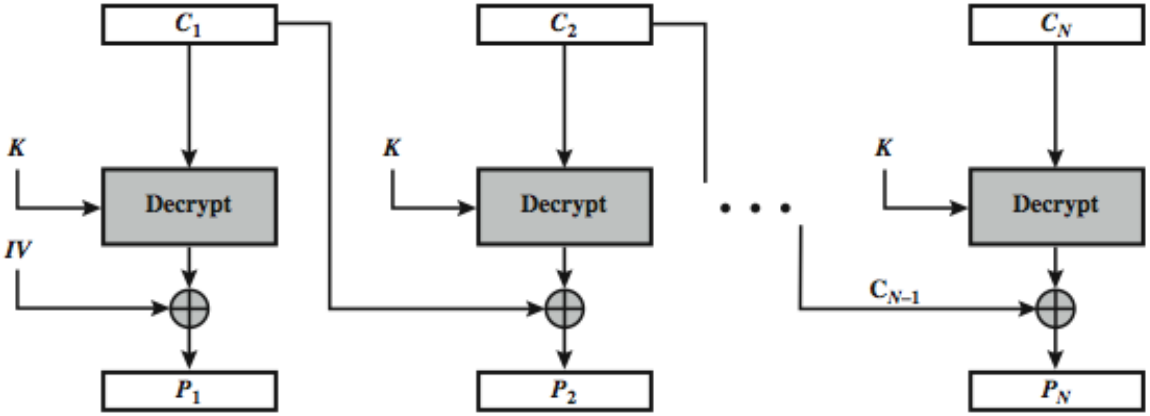
Message Padding

- At end of message must handle a possible last short block
  - Which is not as large as block size of cipher
  - Pad either with known non-data value
    - e.g., nulls
  - or pad last block along with count of pad size
    - e.g., [b1 b2 b3 0 0 0 0 5]
    - means have 3 data bytes, then 5 bytes pad + count

Cipher Block Chaining (CBC)



(a) Encryption



(b) Decryption

Advantages and Limitations of CBC

- A ciphertext block depends on all blocks before it
- Any change to a block affects all following ciphertext blocks (Avalanche Effect)
- Need Initialization Vector (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, by changing corresponding bits of IV
  - hence IV must either be a fixed value
  - or derived in way hard to manipulate
  - or sent encrypted in ECB mode before rest of message
  - or message integrity must be checked otherwise

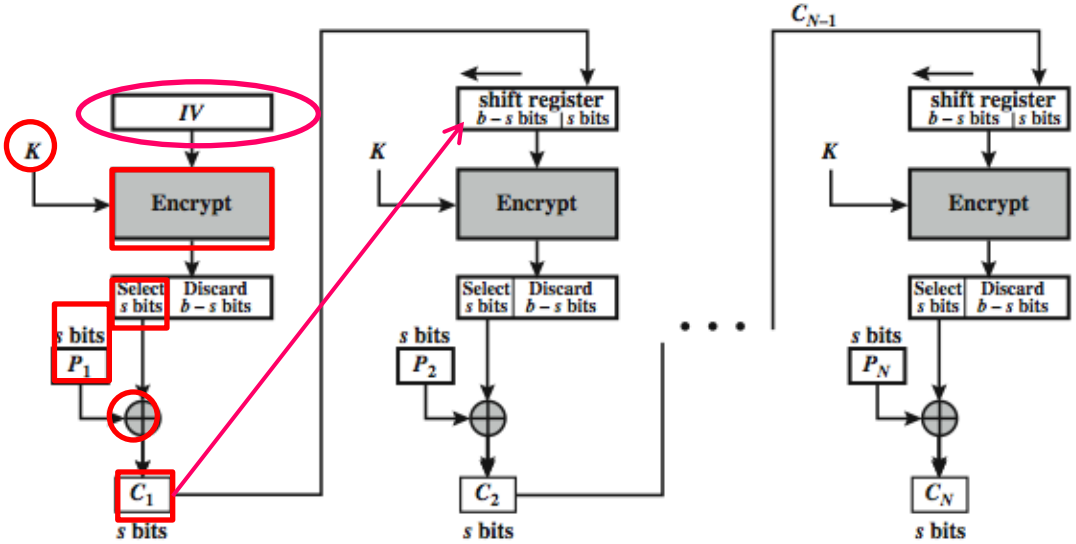
Stream Modes of Operation

- Block modes encrypt entire block
- May need to operate on smaller units
  - real time data
- Convert block cipher into stream cipher
  - Cipher FeedBack (CFB) mode
  - Output FeedBack (OFB) mode
  - Counter (CTR) mode

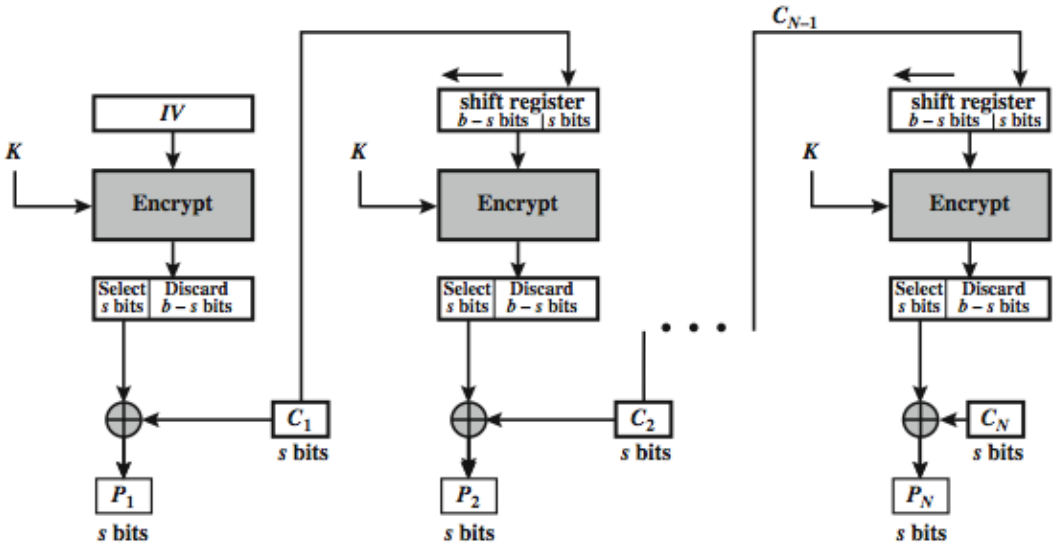
Cipher FeedBack (CFB)

- Message is treated as a stream of bits
- Result is feed back for next stage
- Standard allows any number of bits (1,8, 64 or 128 etc) to be feed back
  - denoted CFB-1, CFB-8, CFB-64, CFB-128, etc.
- Most efficient to use all bits in block (64 or 128)
- $C_i = P_i \text{ XOR } E_k(IV)$

Cipher FeedBack (CFB)



(a) Encryption



(b) Decryption

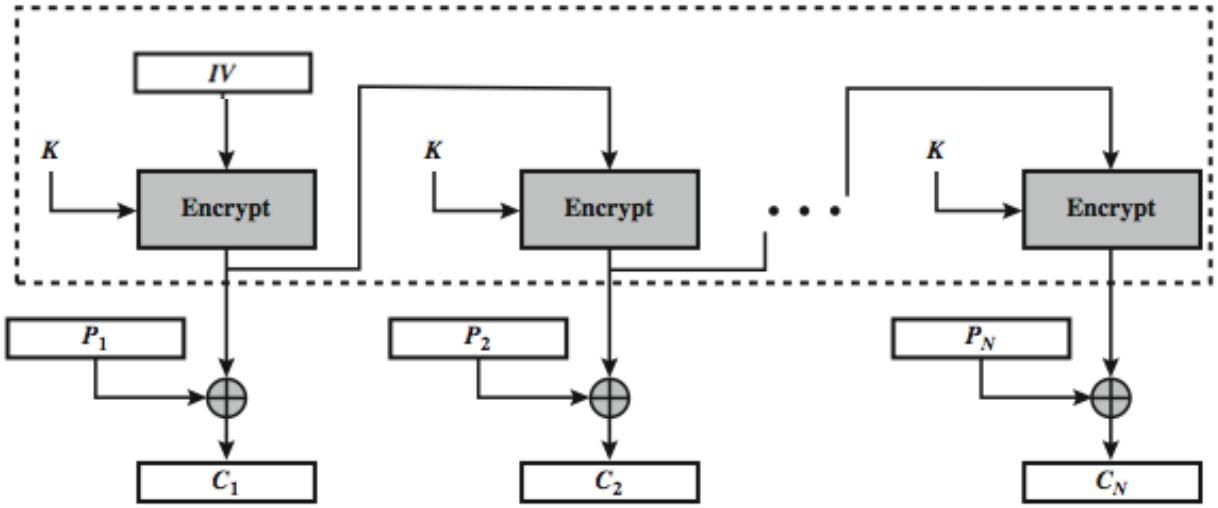
Advantages and Limitations of CFB

- Note that the block cipher is used in encryption mode at both ends (XOR)
- Limitation is need to stall while do block encryption after every s-bits

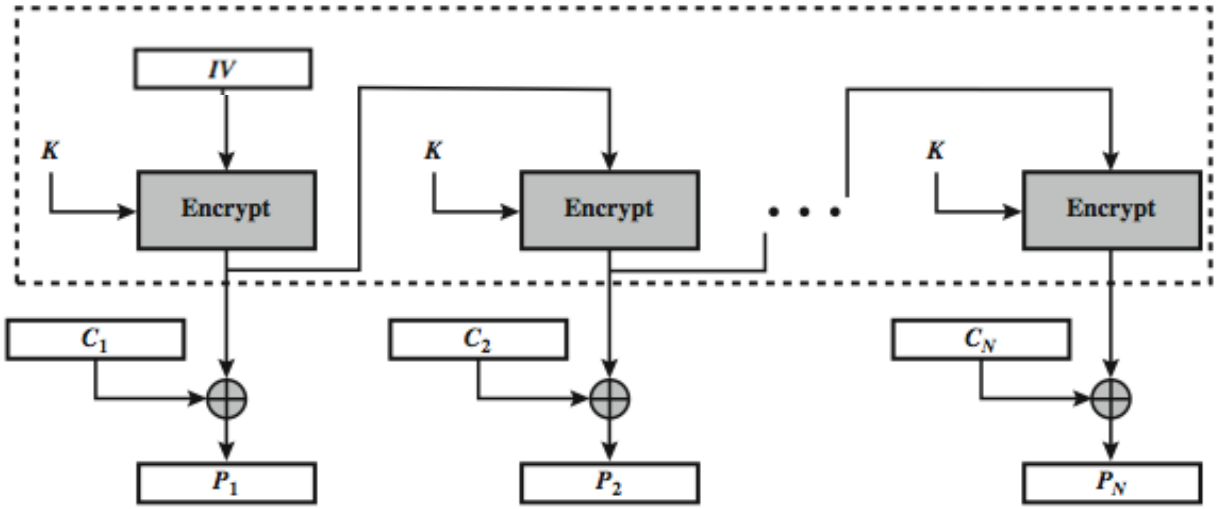
Output FeedBack (OFB)

- Message is treated as a stream of bits
- Output of cipher is added to message
- Output is then feed back
  - $O_i = E_k(IV)$
  - $C_i = P_i \text{ XOR } O_i$
- Feedback is independent of message
- Can be computed in advance

Output FeedBack (OFB)



(a) Encryption



(b) Decryption

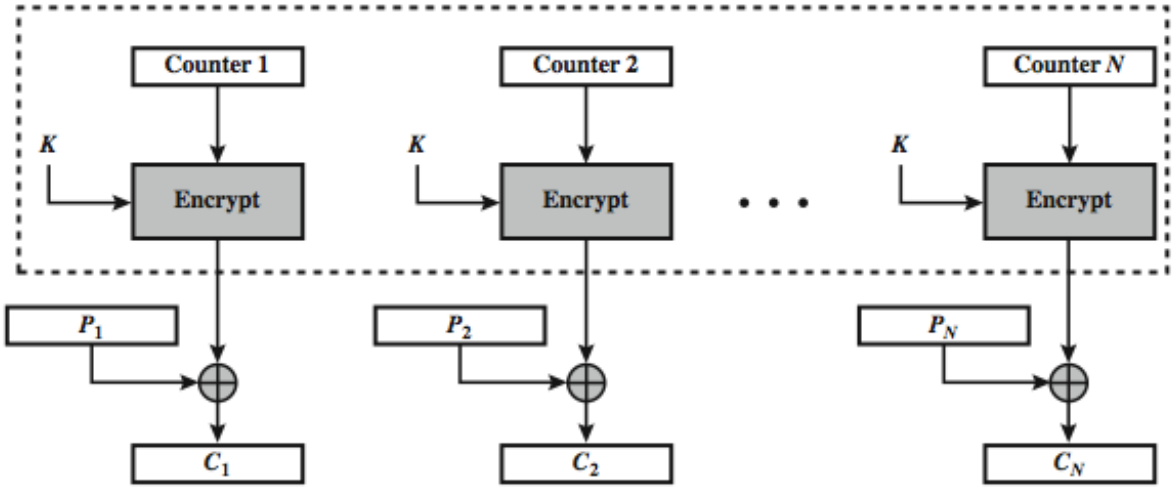
Advantages and Limitations of OFB

- Needs an IV which is unique for each use
  - OTP
- can pre-compute
- sender & receiver must remain in sync
- only use with full block feedback
  - subsequent research has shown that only full block feedback (ie CFB-64 or CFB-128) should ever be used

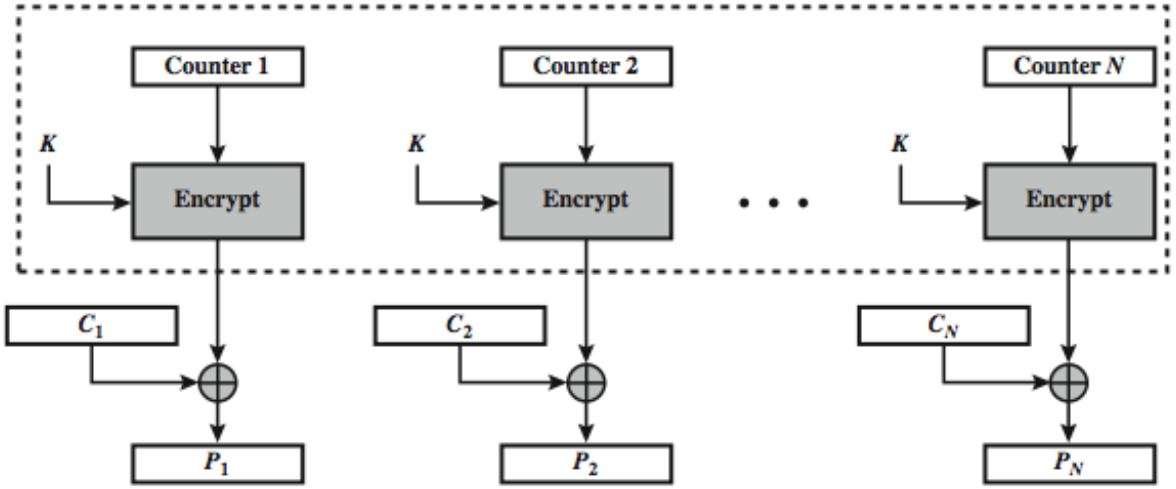
Counter (CTR)

- Similar to OFB but encrypts counter value rather than any feedback value
  - $O_i = E_k(CTR_i)$
  - $C_i = P_i \text{ XOR } O_i$
- Must have a different counter value for every plaintext block (never reused)
  - again, OTP issue

Counter (CTR)



(a) Encryption

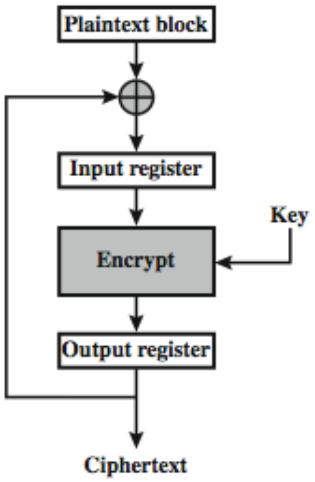


(b) Decryption

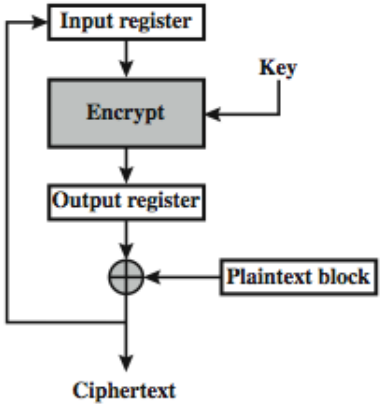
Advantages and Limitations of CTR

- **Efficiency**
  - can do parallel encryptions in h/w or s/w
  - can preprocess in advance of need
  - good for bursty high speed links
- **Random access to encrypted data blocks**
- **Provable security (good as other modes)**
- **Never have cycle, must ensure never reuse key/counter values, otherwise could break (OFB)**

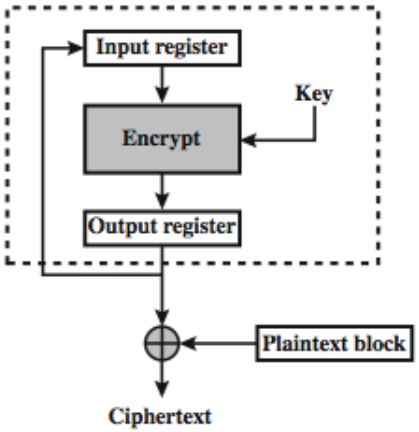
Feedback Characteristics



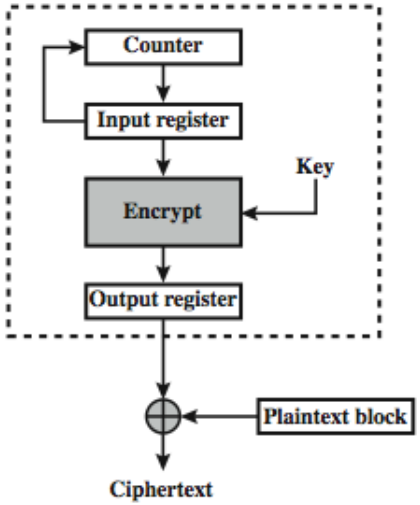
(a) Cipher block chaining (CBC) mode



(b) Cipher feedback (CFB) mode

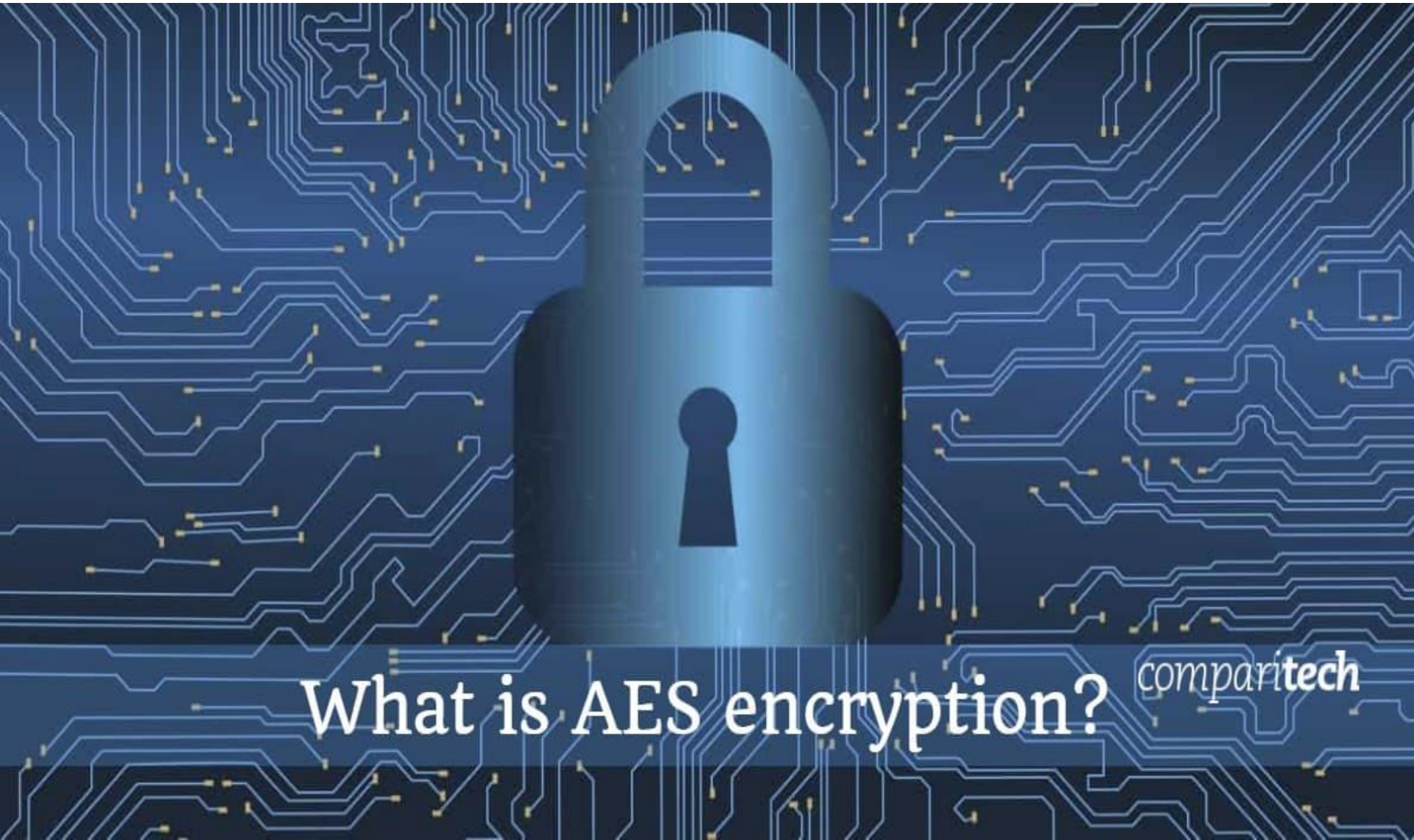


(c) Output feedback (OFB) mode



(d) Counter (CTR) mode

Advanced Encryption Standards

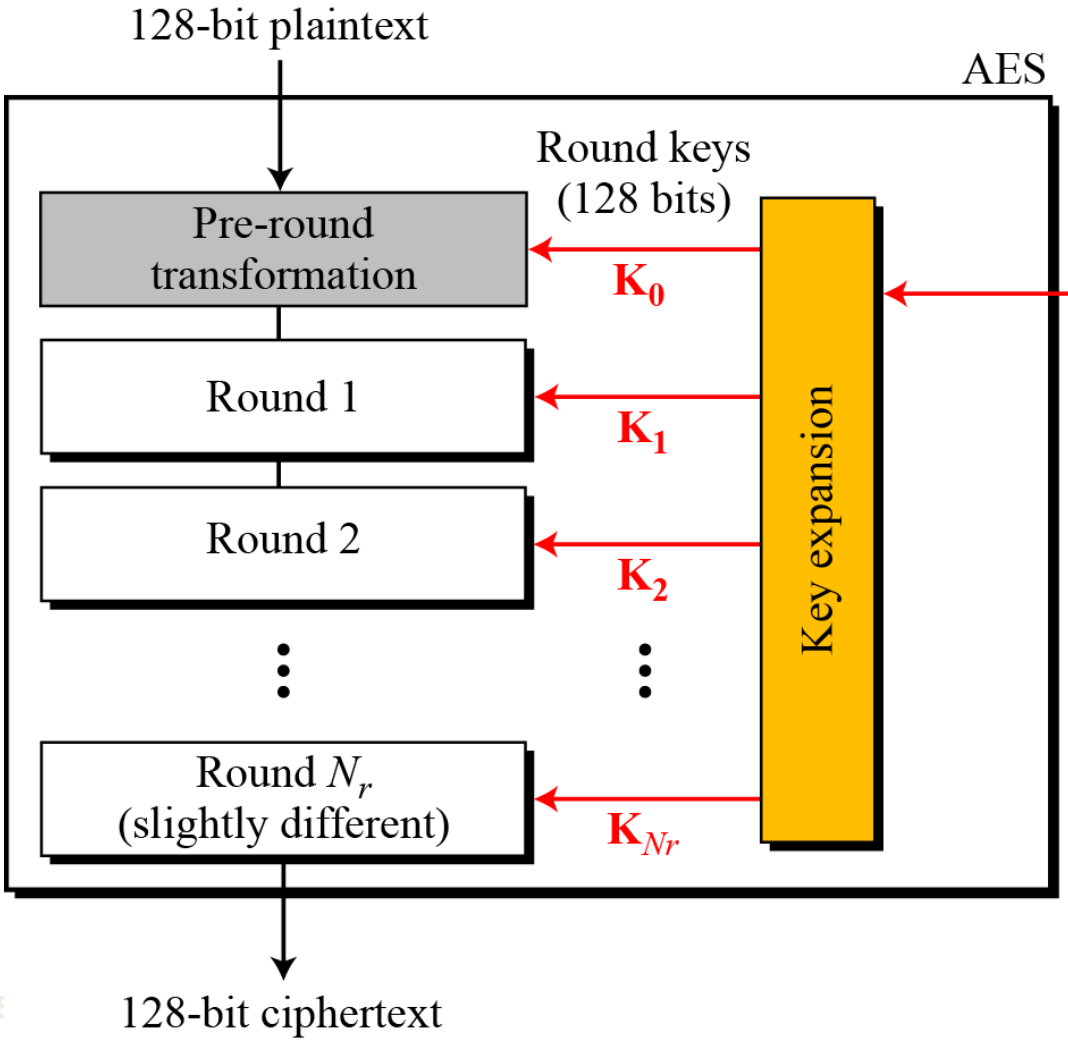


Advanced Encryption Standards

- The **Advanced Encryption Standard** (**AES**), also known by its original name **Rijndael**.
- **AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds. The key size, which can be 128, 192, or 256 bits, depends on the number of rounds.**

AES has defined three versions, with 10, 12, and 14 rounds.
Each version uses a different cipher key size (128, 192, or 256), but the round keys are always 128 bits.

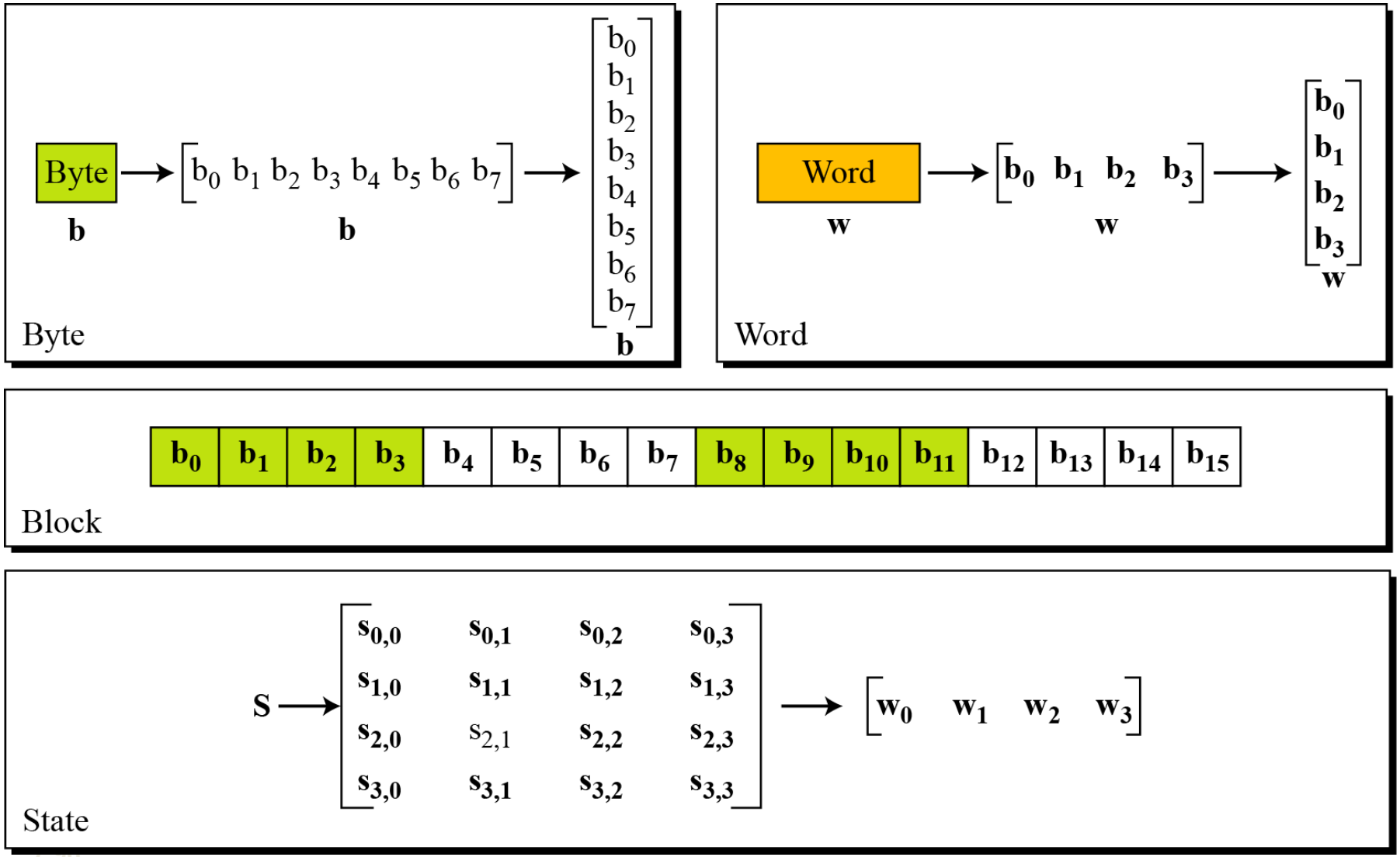
General design of AES encryption cipher



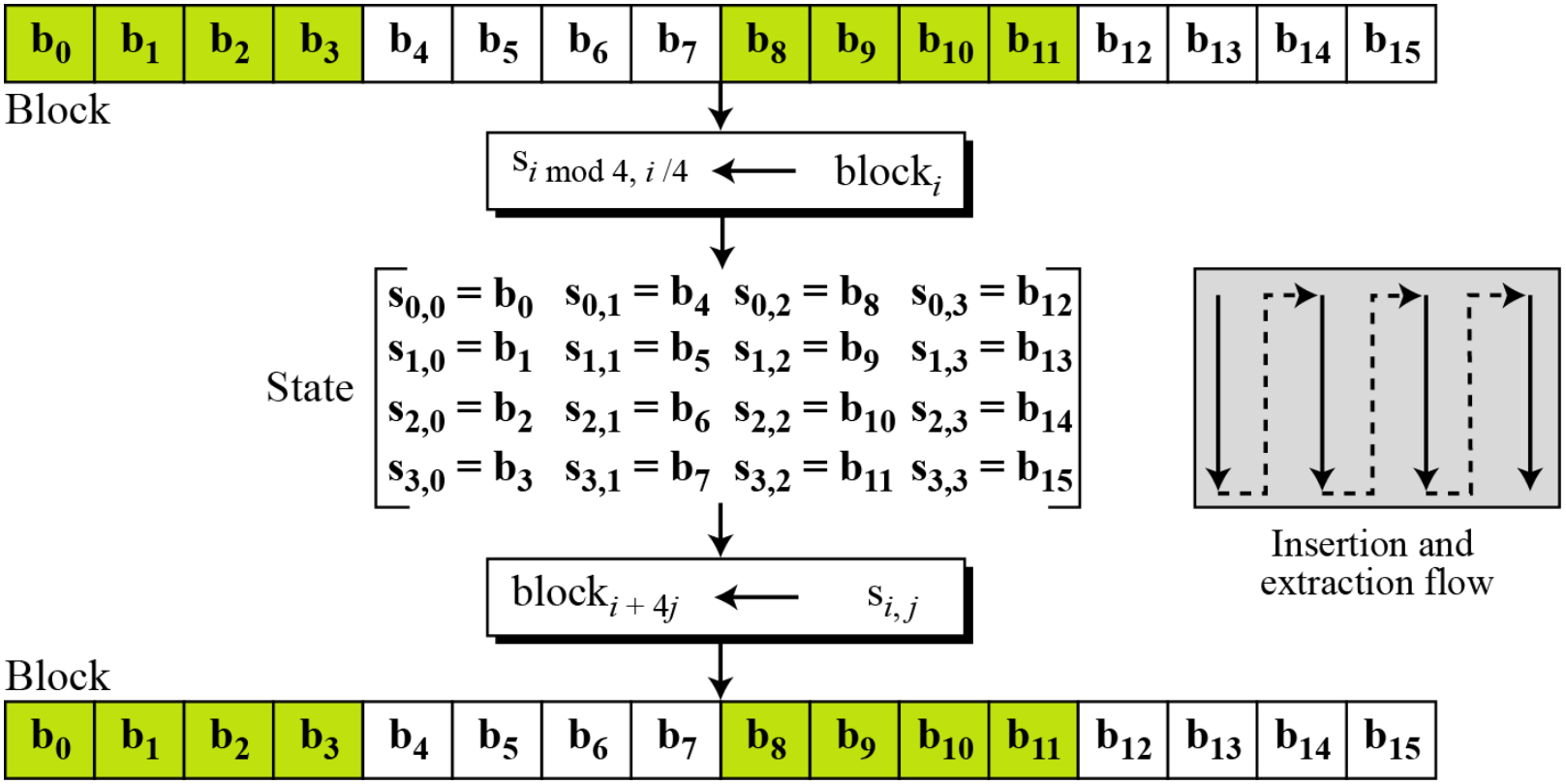
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

Data units used in AES



Block-to-state and state-to-block transformation

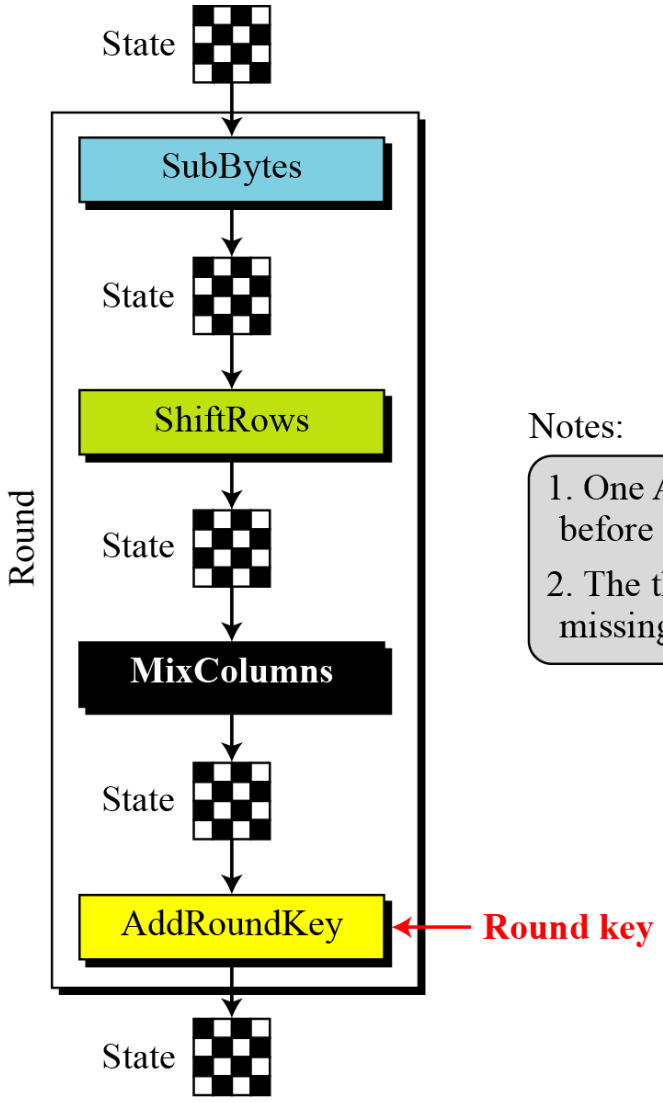


Changing plaintext to state

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

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

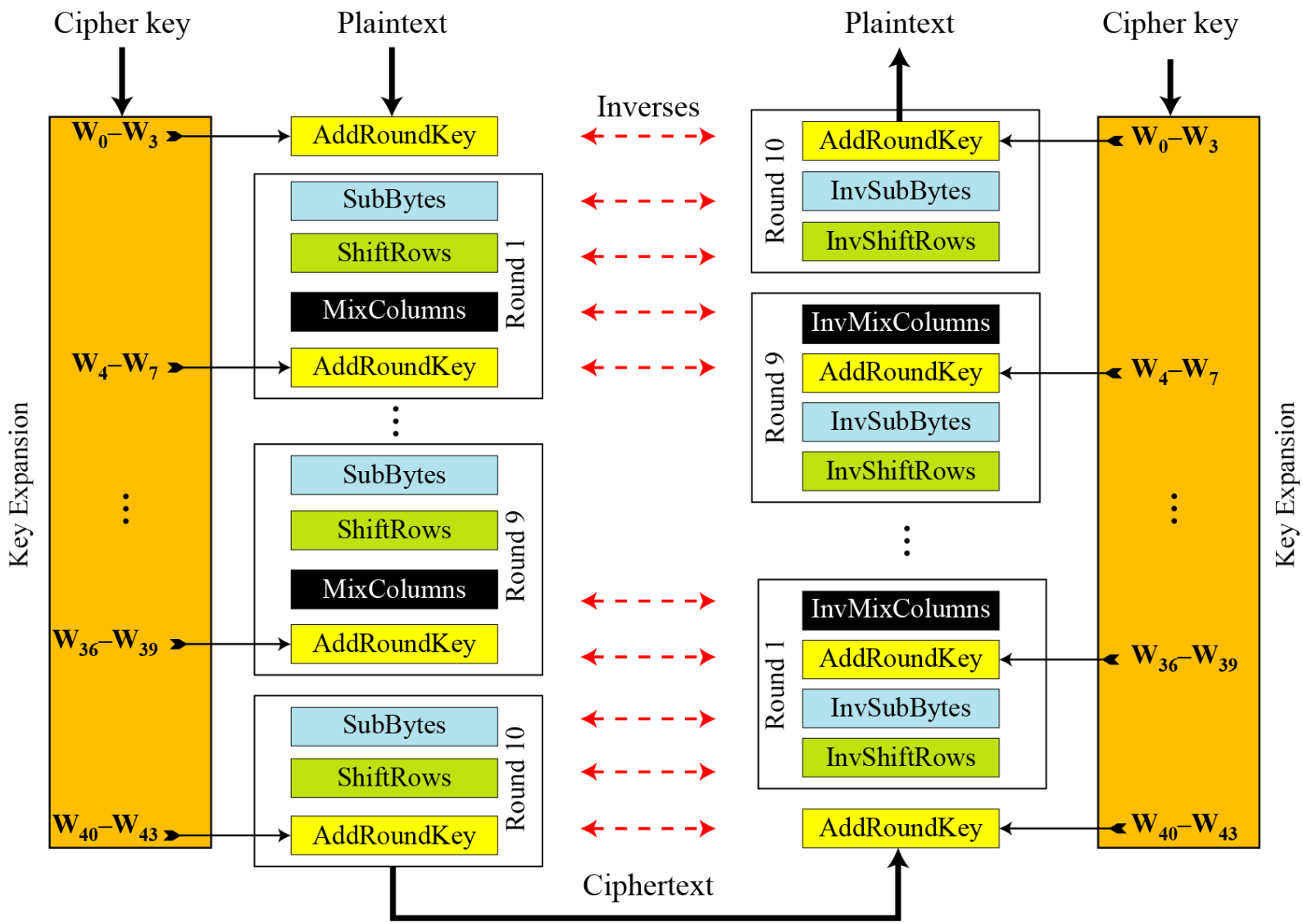
Structure of each round at the encryption site



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

Overall Structure



AES TRANSFORMATIONS

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

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

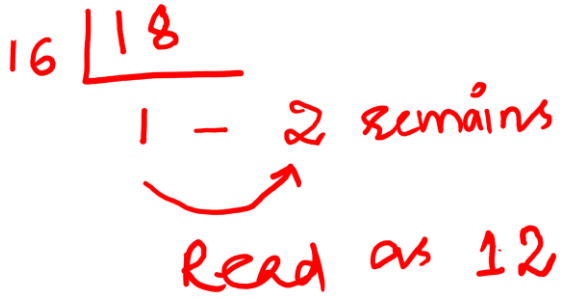
- **Replace each character with an integer between**

A = 00 to Z = 25.

- **Hexadecimal base is 16.**
- **< 16 integer values of characters are remains same in hexadecimal whereas >=16 integer values convert into hexadecimal.**

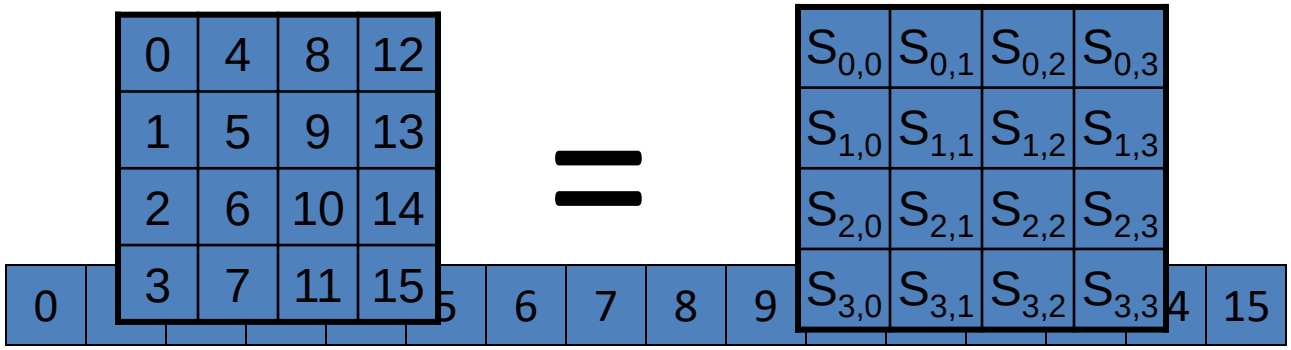
Decimal to Hexadecimal Conversion

- For example: the character S = 18 in integer the convert into hexadecimal.



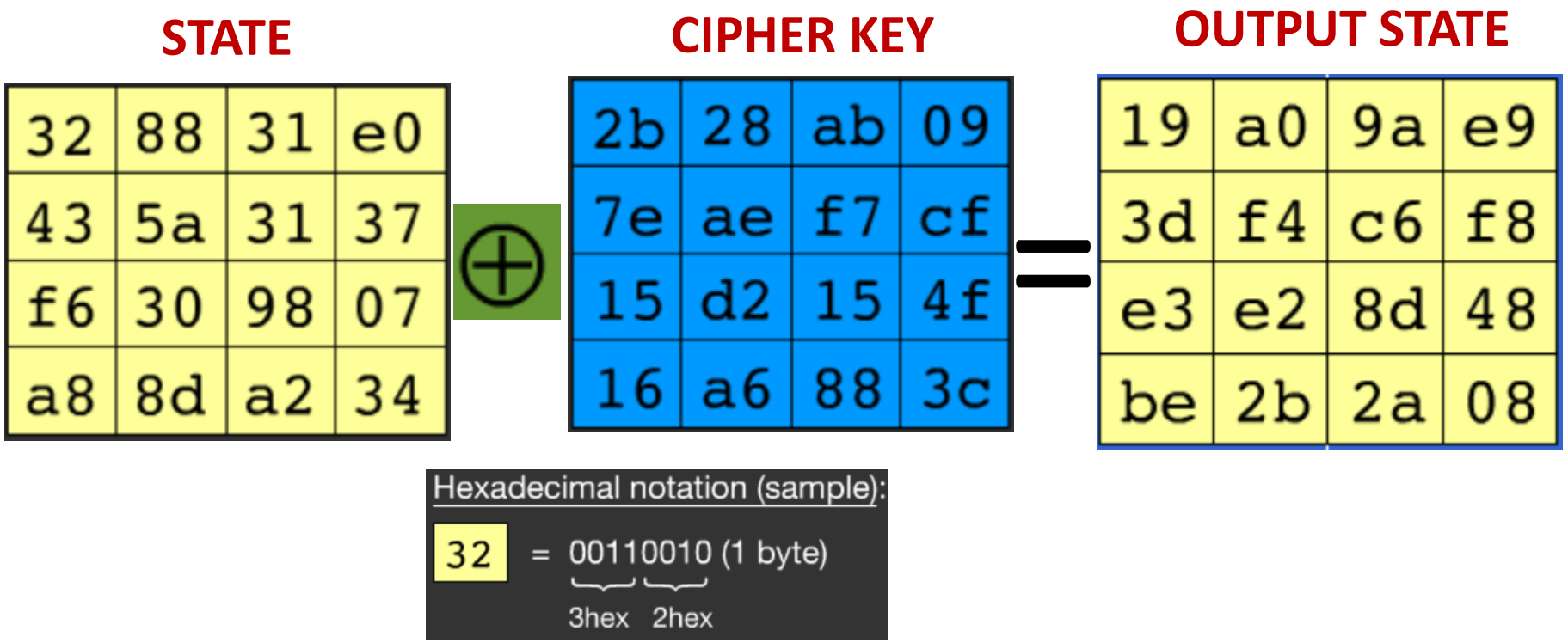
Convert to State Array

Input block:



- Plain text characters or cipher key are converted into hexadecimal values.
- Hexadecimal values are placed into 4 * 4 matrix form.

Add Round Key Transformation



- 32 $\oplus$ 2b = 00110010 $\oplus$ 00101011

  = 00011001

  = 1 9 = 19

SubBytes Transformation

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

=

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

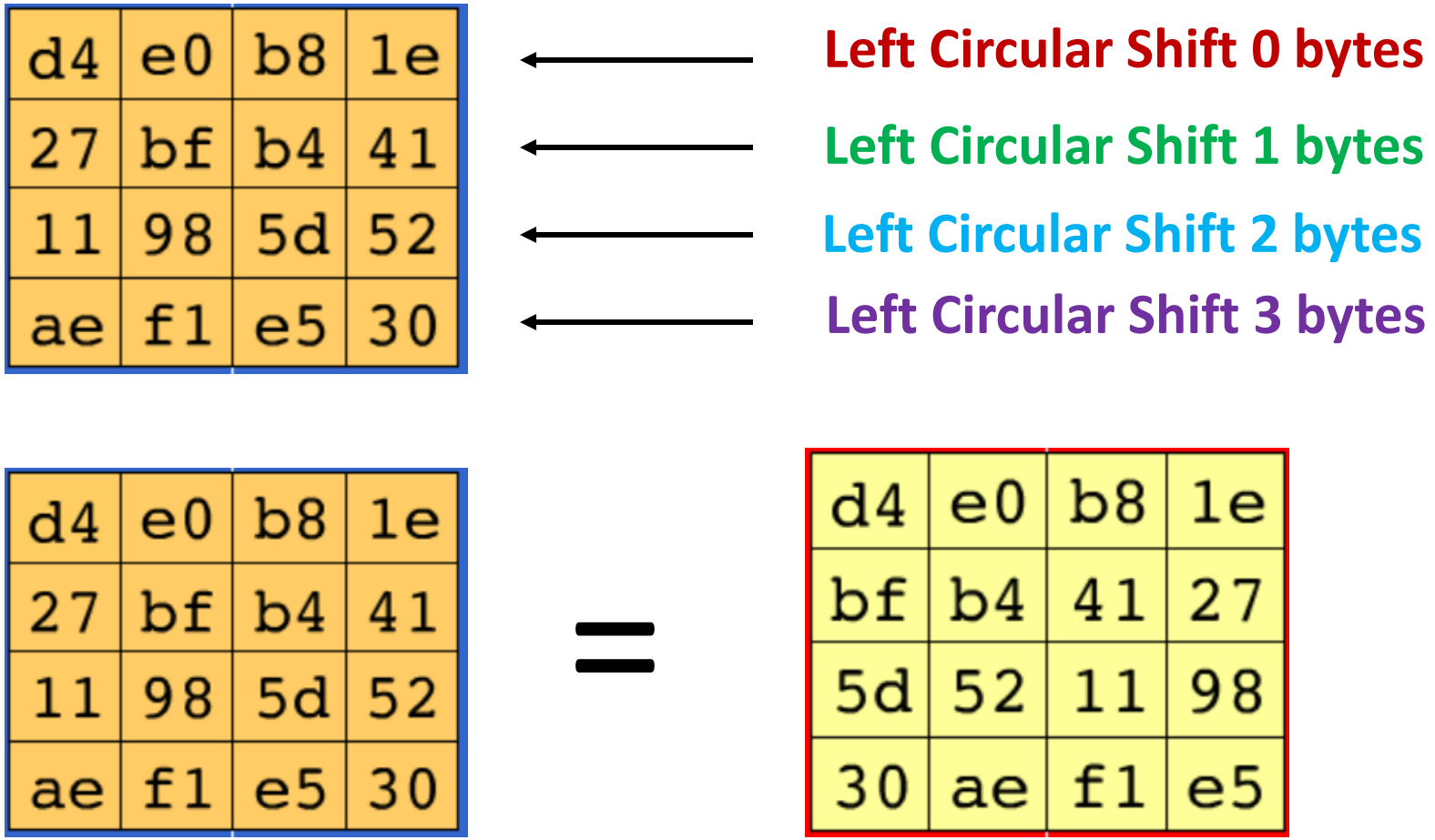
19 = d4

hex	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	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

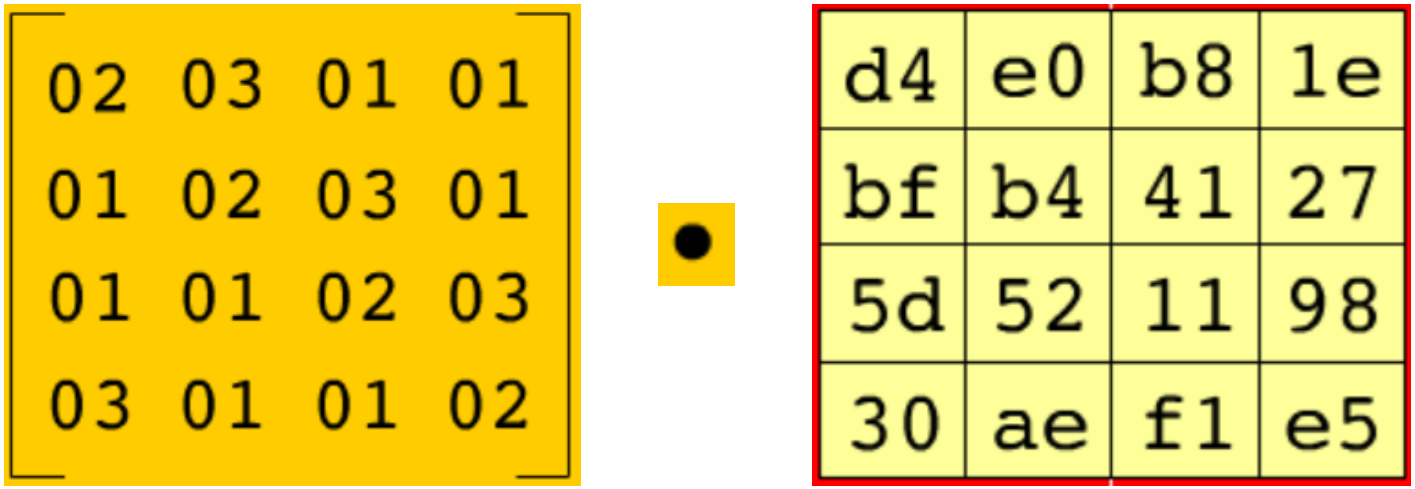
x

d4

Shiftrows Transformation



Mix Columns Transformation



(02 • d4) ⊕ (03 • bf) ⊕ (01 • 5d) ⊕ (01 • 30)

As per polynomial rules:

01 =1, 02=x , 03=x+1, $x^8 = x^4+x^3+x+1$

Finite Fields

- AES uses the finite field GF(2^8)
  - $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$
    - $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$
- Byte notation for the element: $x^6 + x^5 + x + 1$
  - {01100011} – binary
  - {63} – hex
- Has its own arithmetic operations
  - Addition
  - Multiplication

Finite Field Arithmetic

- Addition (XOR)
  - $(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$
  - $\{01010111\} \oplus \{10000011\} = \{11010100\}$
  - $\{57\} \oplus \{83\} = \{d4\}$

Finite Field Multiplication (•)

$(x^6 + x^4 + x^2 + x + 1)\ (x^7 + x + 1) =$

$x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1$

These cancel

$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

and

$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$ modulo $(x^8 + x^4 + x^3 + x + 1)$

$= x^7 + x^6 + 1.$

Irreducible Polynomial

Mix Column Calculation

(02 • d4) ⊕ (03 • bf) ⊕ (01 • 5d) ⊕ (01 • 30)

= (x • d4) ⊕ (x+1 • bf) ⊕ (1 • 5d) ⊕ (1 • 30)

d = 1101

4 = 0100

7 6 5 4 3 2 1 0

= (x (11010100) ⊕ (x+1(10111111)) ⊕ (1(01011101) ⊕ (1(00110000))

(x dot d4) = $(x(x^7 + x^6 + x^4+x^2)$

= $(x^8 + x^7 + x^5+x^3)$ [$x^8 = x^4+x^3+x+1$]

= $(x^4+x^3+x+1 + x^7 + x^5+x^3)$ [similar will be cancel]

= $(x^4+x+1 + x^7 + x^5)$ [arrange in order]

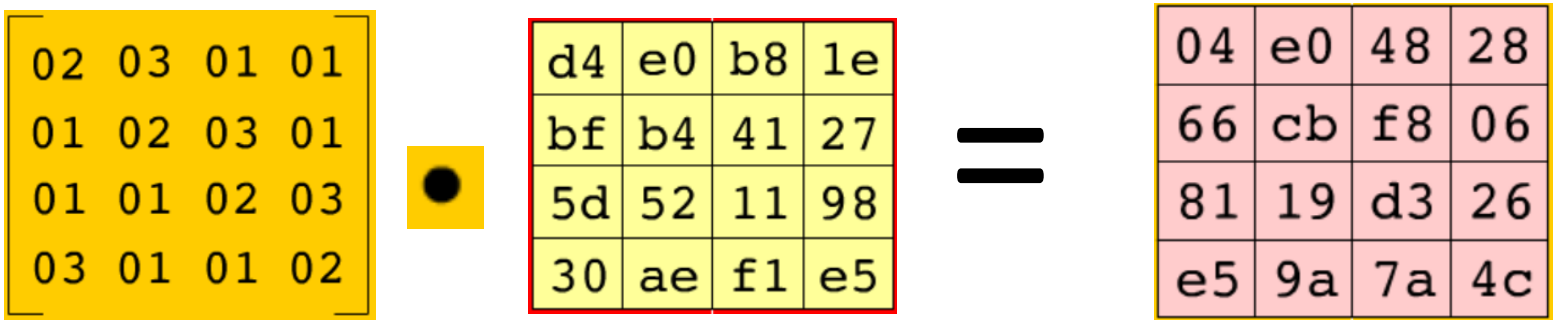
= $(x^7 + x^5+x^4+x+1)$ [change into binary]

= (10110011)

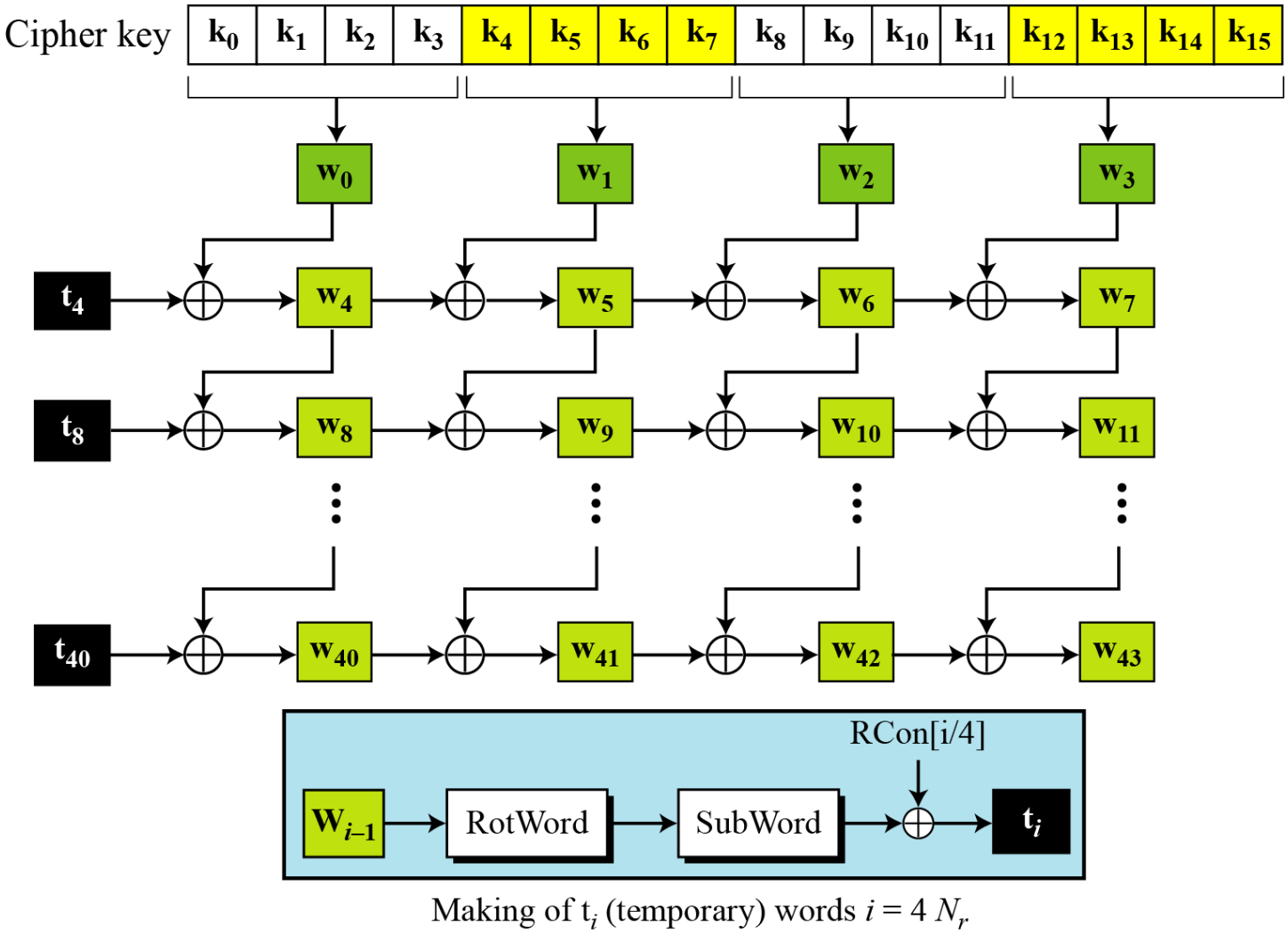
= b 3

= **b3**

Mix Column Calculation - Output



Key Expansion in AES-128



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

Key Expansion submodule

- **RotWord** performs a one-byte circular left shift on a word For example:

  RotWord[b0,b1,b2,b3] = [b1,b2,b3,b0]

- **SubWord** performs a byte substitution on each byte of input word using the S-box

- **SubWord(RotWord(temp))** is XORed with RCon[j] – the round constant

Round Constant (RCon)

- RCON is a word in which the three rightmost bytes are zero
- It is different for each round and defined as:

  RCon[j] = (RCon[j],0,0,0)

  where RCon[1] =1 , RCon[j] = 2 * RCon[j-1]
- Multiplication is defined over GF(2^8) but can be implement in Table Lookup

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	(**01** 00 00 00)$_{16}$	6	(**20** 00 00 00)$_{16}$
2	(**02** 00 00 00)$_{16}$	7	(**40** 00 00 00)$_{16}$
3	(**04** 00 00 00)$_{16}$	8	(**80** 00 00 00)$_{16}$
4	(**08** 00 00 00)$_{16}$	9	(**1B** 00 00 00)$_{16}$
5	(**10** 00 00 00)$_{16}$	10	(**36** 00 00 00)$_{16}$