Unit 3

Modern Encryption Techniques

Presented by:
Dr. Mohammed Alkhanafseh

Stream Cipher

Encrypts a digital data stream one bit or one byte at a time

Examples:

- Auto-keyed Vigenère cipher
- Vernam cipher

In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the key-stream is as long as the plaintext bit stream

If the cryptographic key-stream is random, then this cipher is unbreakable by any means other than acquiring the key-stream

- Key stream must be provided to both users in advance via some independent and secure channel
- This introduces insurmountable logistical problems if the intended data traffic is very large

For practical reasons, the bit-stream generator must be implemented as an algorithmic procedure so that the cryptographic bit stream can be produced by both users

The two users need only share the generating key and each can produce the keystream

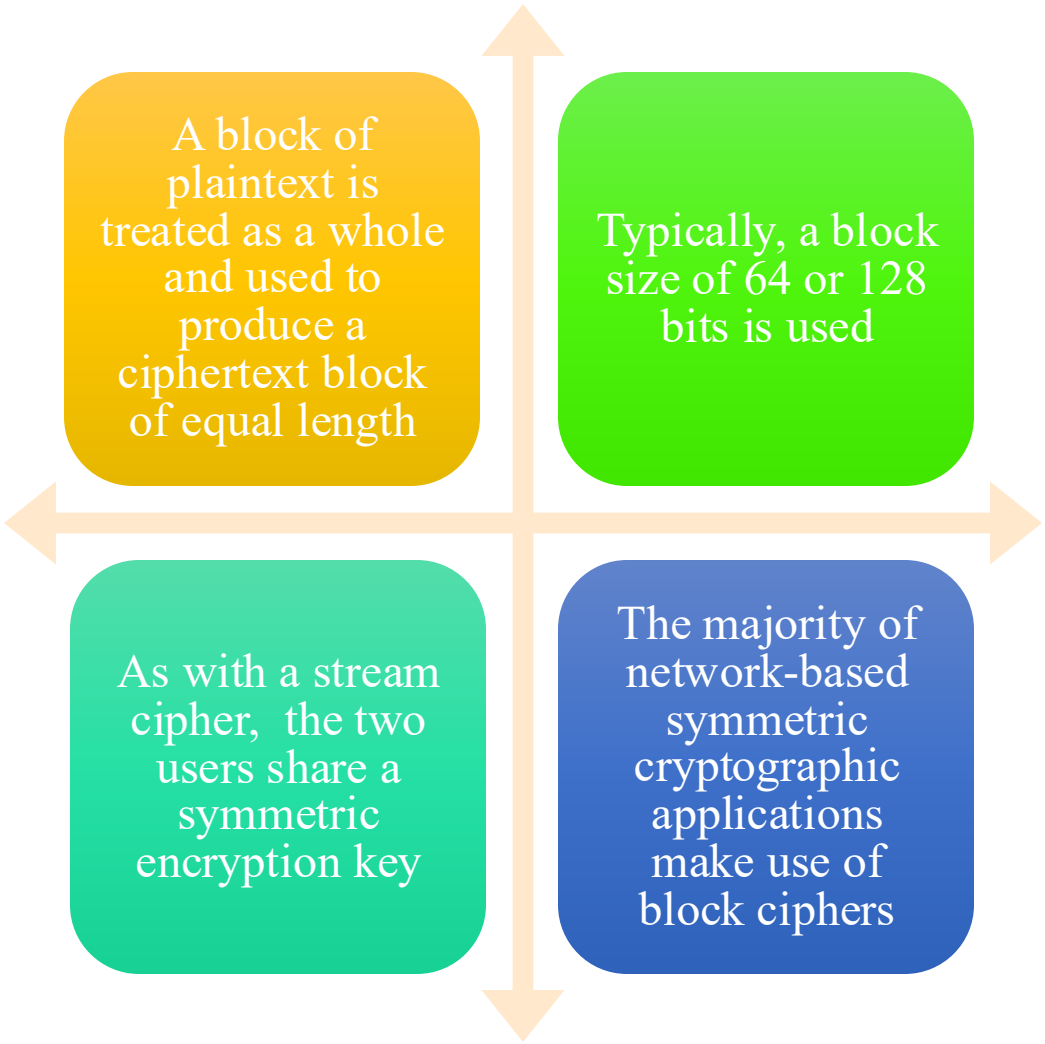
Stream Cipher

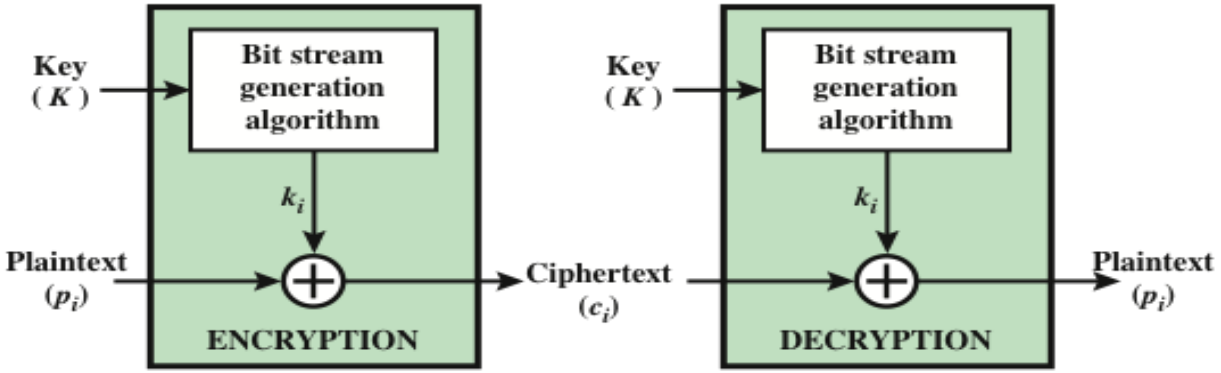
Algorithm(Random Sequence Algorithm), which mean random generator in infinity sequence, and will be generated based on the number of digits for the plain text.
Key Stream Generator
XOR, which will be used in the process of encryption using stream cipher.

Condition of Random Sequence

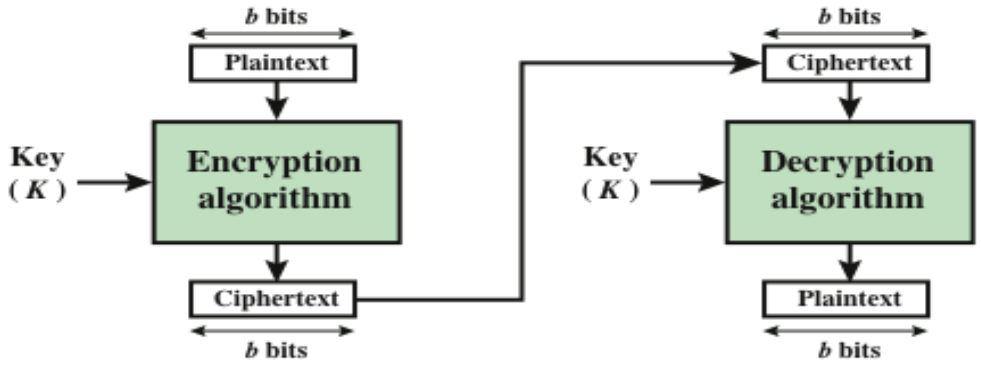
- Periodic (Long Period).
- Key: K1, K2, …..,Kn this is period = p.
- Random Sequence (Cipher text random)
- Large Linear Complexity

Block Cipher





(a) Stream Cipher Using Algorithmic Bit Stream Generator

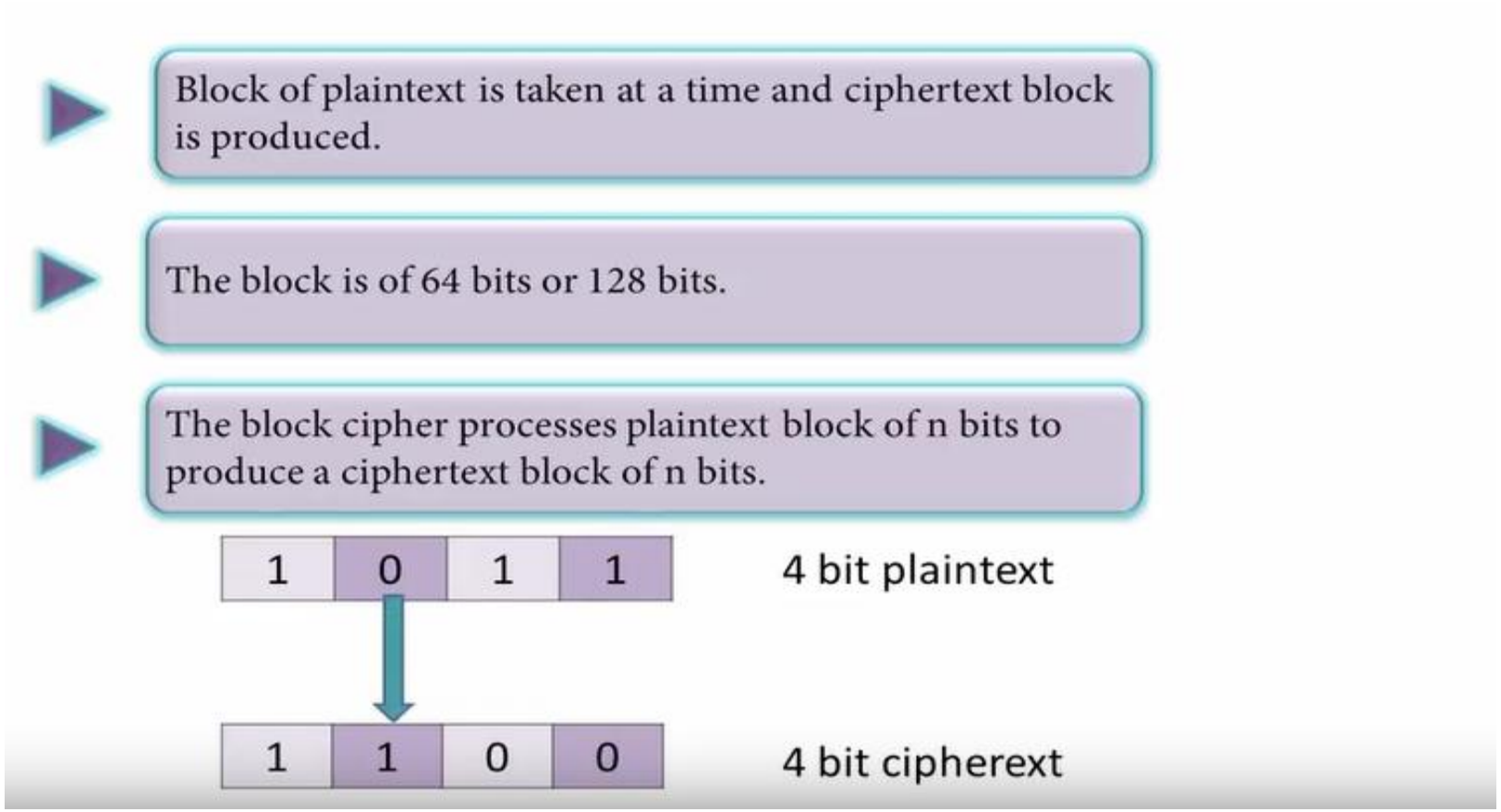


(b) Block Cipher

Figure 4.1 Stream Cipher and Block Cipher

Block Cipher

- Block of plaintext is taken at a time and ciphertext block is produced.
- The block is of 64 bits or 128 bits.
- The block cipher processes plaintext block of n bits to produce a ciphertext block of n bits.



Vernam cipher (stream cipher)

- Also known as the one-time pad, it's a symmetric key encryption algorithm.
- **Method:** It uses the principle of bitwise exclusive OR (XOR) between the plaintext and a random or truly unpredictable key of the same length.
- **Key:** The key used in the Vernam cipher must be as long as the plaintext and should be completely random. If the key is reused or predictable in any way, the security of the cipher is compromised.
- **Strengths:** When used correctly with a truly random key that's as long as the plaintext and never reused, it provides perfect secrecy or unconditional security (in the sense of information-theoretic security).
- **Weaknesses:** Key management is a significant challenge in Vernam cipher. Generating and securely exchanging keys as large as the plaintext without any reuse is impractical in most scenarios.

Why Key is critical in Stream Cipher

- **Unique Key Requirement:** The Vernam cipher requires a key that is as long as the plaintext message and completely random. For instance, if you have a 1000-bit message, you need a 1000-bit key.
- **Perfect Secrecy Dependency:** The security of the Stream cipher relies on perfect secrecy, which means the key must never be reused. If any part of the key is reused or predictable, it compromises the security of the entire encryption process.
- **Key Distribution Challenges:**
  - •Secure Transmission: Transmitting keys as long as the plaintext securely to the intended recipient is challenging, especially over insecure channels (e.g., the internet).
  - •Randomness and Generation: Creating truly random keys that match the length of the message and ensuring they remain secret during transmission is difficult. Pseudo-random number generators (PRNGs) are often used, but they can't generate true randomness.
  - Other Challenges like key storage, key distortion

Example for mentioned challenges

- Let's say Alice wants to send a confidential 500-bit message to Bob using the Vernam cipher.
- Generating a 500-bit truly random key: Alice needs a secure method to generate this key.
- Securely transmitting the key to Bob: Alice needs a secure channel or method to transmit the 500-bit key without interception or compromise.
- Ensuring Bob's key matches Alice's: Bob needs to receive the exact same 500-bit key that Alice generated without any alteration.

Vernam cipher (stream cipher) Example

Plaintext:1000 1001 1101 1110
Key : 1001 1110 1100 0000

- **Ciphertext ➔ C=P Xor K**
- **(0001 0111 0001 1110)**

- **Plaintext➔ P= C Xor K**
- **(1000 1001 1101 1110)**

plaintext: C9 1100 1001
Key : A3 (1010 0011)

- **Ciphertext ➔ C=P Xor K**
- **(0110 1010) 6A**

- **Plaintext➔ P= C Xor K**
- **(1100 1001)**

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations

Substitutions
- Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements

Permutation
- No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

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

- Is the structure used by many significant symmetric block ciphers currently in use

- Terms introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system
  - Confusion and diffusion are fundamental concepts in cryptography that contribute to the security of encryption algorithms. These concepts are often realized in encryption algorithms through specific techniques like substitution and permutation.

Diffusion

- Diffusion ensures that the influence of a single plaintext bit affects multiple parts of the ciphertext. Permutation, or transposition, is a cryptographic technique that contributes to diffusion.

Confusion

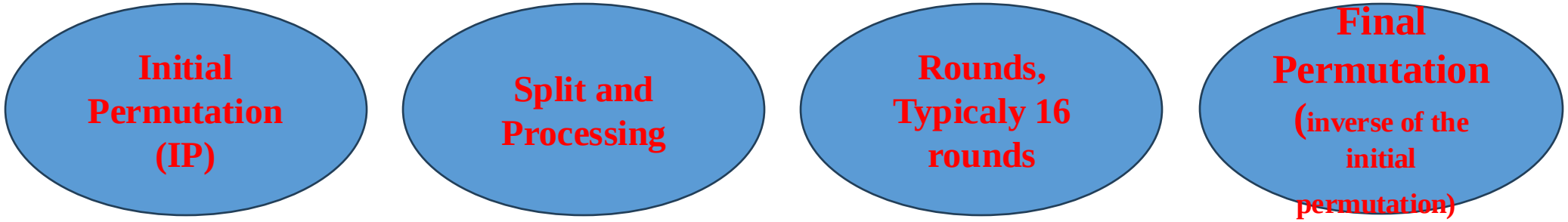
- Diffusion ensures that the influence of a single plaintext bit affects multiple parts of the ciphertext. Permutation, or transposition, is a cryptographic technique that contributes to diffusion.
- **Example:** In permutation, elements (bits, characters, or blocks) are rearranged or shuffled according to a specific pattern or rule.

• Feistel Cipher

- The Feistel cipher is a symmetric structure used in the construction of **block ciphers**. It operates through a series of rounds, each consisting of substitution, permutation, and mixing operations. One of the most famous block ciphers based on the Feistel structure is the Data Encryption Standard (DES).

• Feistel Cipher Structure:

- Key Expansion: The encryption key is expanded into round keys for each round of encryption.
- Round Function: Each round consists of several operations, typically involving substitution (S-boxes), permutation, and mixing functions.



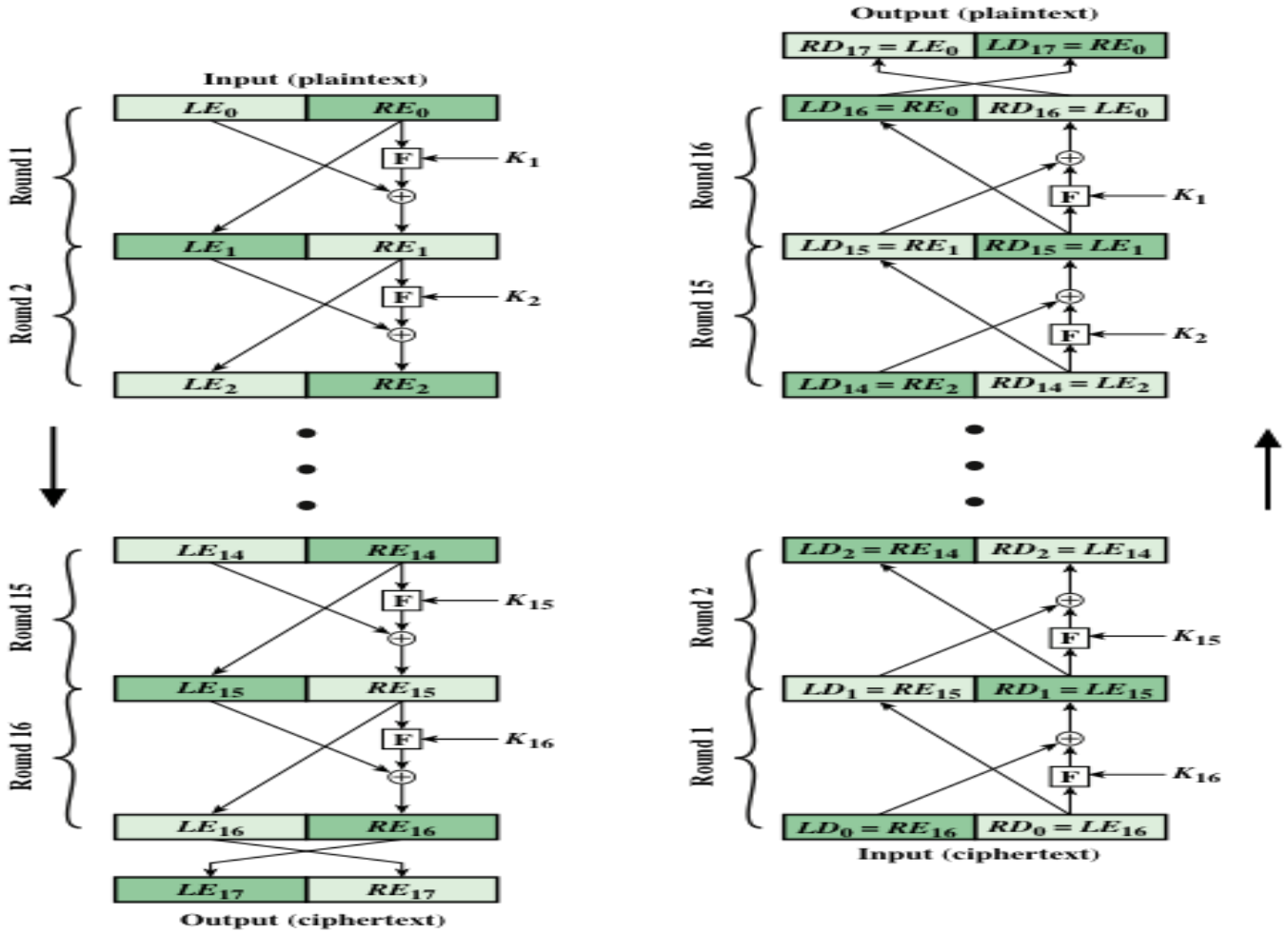


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

- **Encryption:**
  - $L_1 = R_0 \quad R_1 = L_0 \oplus f_1(R_0)$
  - $L_2 = R_1 \quad R_2 = L_1 \oplus f_2(R_1)$
  - ...
  - $L_d = R_{d-1} \quad R_d = L_{d-1} \oplus f_d(R_{d-1})$
- **Decryption:**
  - $R_{d-1} = L_d \quad L_{d-1} = R_d \oplus f_d(L_d)$
  - ...
  - $R_0 = L_1; \quad L_0 = R_1 \oplus f_1(L_1)$

Simple Example:

Plaintext: 110011

Initial Permutation: 110011 → 011100

First Round

- **Initial Split:** Left (L) = 011, Right (R) = 100
- **Round Key:** Generated for Round 1
- **Round Function (Simplified):**
  - R' = R XOR RoundKey = 100 XOR RoundKey
  - L' = R' XOR L = (100 XOR RoundKey) XOR 011
- **Swap:** New Left (L') = 100 XOR RoundKey, New Right (R') = 011

Second Round

- **Round Key:** Generated for Round 2
- **Round Function (Simplified):**
  - R" = R' XOR RoundKey = (100 XOR RoundKey) XOR RoundKey'
  - L" = R" XOR L' = ((100 XOR RoundKey) XOR RoundKey') XOR (100 XOR RoundKey)
- **Swap:** New Left (L") = (100 XOR RoundKey) XOR RoundKey', New Right (R") = (100 XOR RoundKey)
- Final Permutation and Output:
- Combine L" and R" and perform the inverse of the initial permutation to obtain the ciphertext.

Feistel Cipher Design Features

- Block size
  - Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speeds
- Number of rounds
  - The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security
- Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
- Round function F
  - Greater complexity generally means greater resistance to cryptanalysis
- Fast software encryption/decryption
  - In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed of execution of the algorithm becomes a concern
- Ease of analysis
  - If the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

Feistel Example

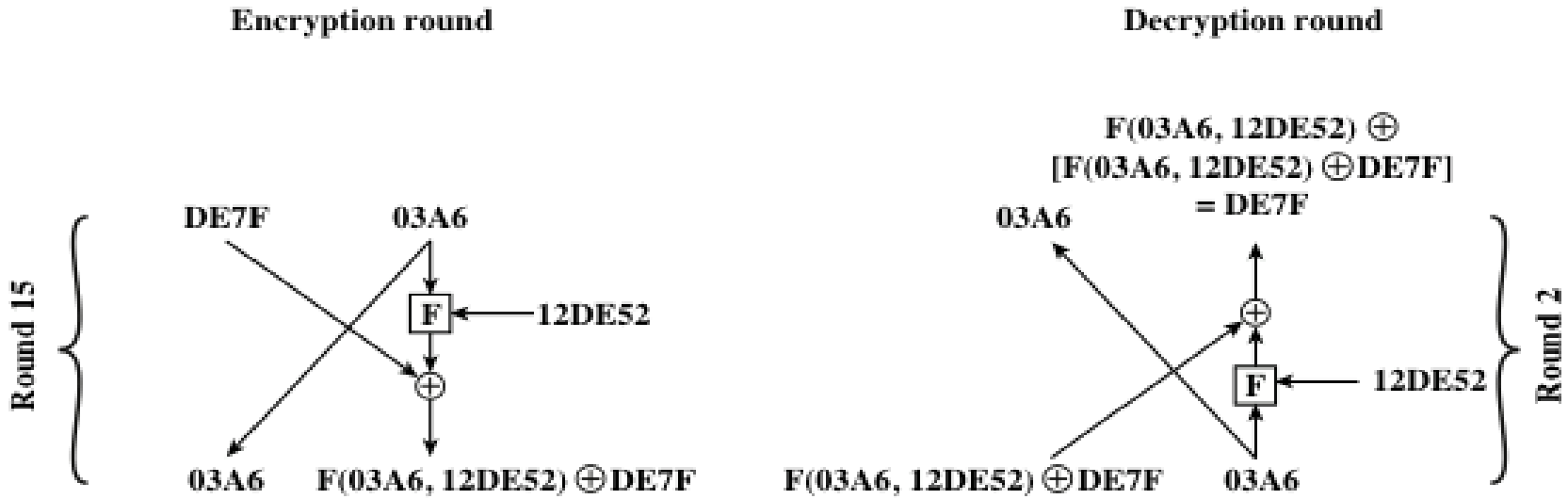


Figure 4.4 Feistel Example

Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm

An LFSR (Linear Feedback Shift Register) is a shift register that generates sequences of binary bits using linear feedback. It is widely used in cryptography, error detection, and pseudo-random number generation due to its efficiency and simplicity.

Linear Feedback shift register, whose input is a linear function (Ex.XOR) of some bits of overall shift register value.

The initial value is called seed value, the register has a finite number of possible states

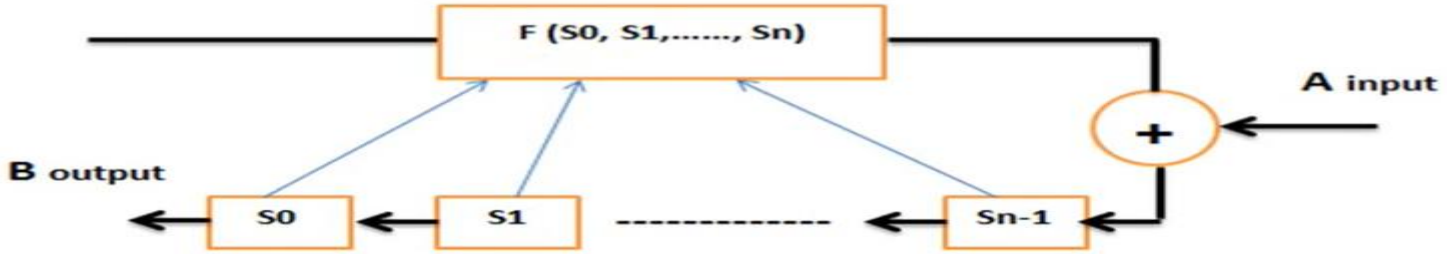
You must note that not all LFSR can reach to the maximum number of rounds, since the number of bit generation will stop when this number is redundant with initial Seed value.

The Key in this algorithm refers to S0(Binary Sequence).

The Maximum Sequence Length is Power(2,n)-1.

If the 4 bit is used, then the maximum length is power(2,4)-1 = 16-1=15

The LFSR algorithm used a registers to be shifted Si-→ Si-1 →,….,→S1→S0

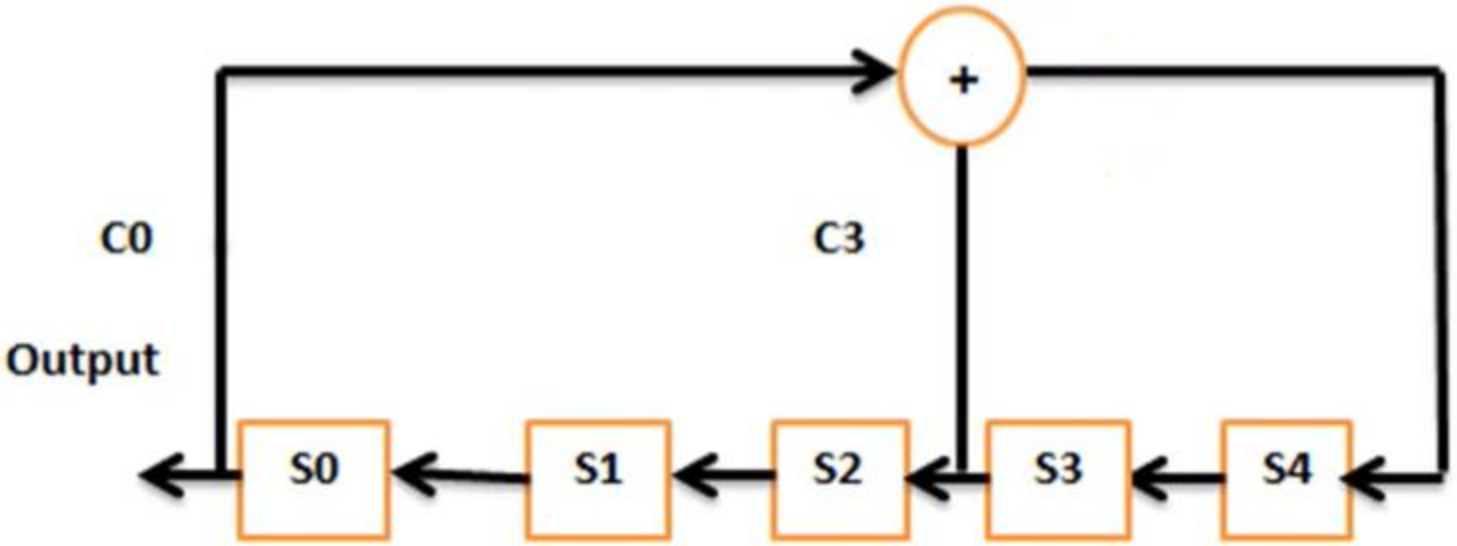


Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm

Shift Register =5 , initial key= 10101

Used LFSR with length n=5

Plain text =HELP



Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm

- Plain text :HELP

H=48H=01001000

Key = 10101110---------sequence

Plain= 01001000

Cipher= 11100110

Stop : stage repeats

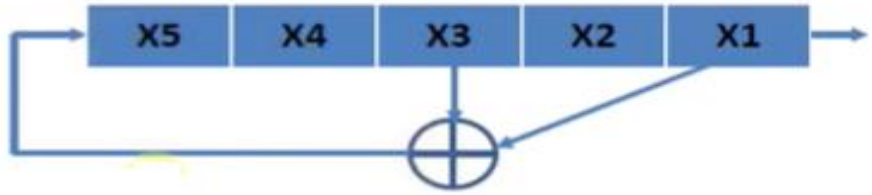
Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm

LFSR Example:

let's look at a **5-bit LFSR** with the tap coefficients of the register given by 00101. In order to guarantee that this shift register has a maximum period. For a register with five internal bits within it, bits that can never all be equal to zero, the maximum length is 2^5-1 or 31. Hence, this register has an output sequence of 31 pseudorandom bits.

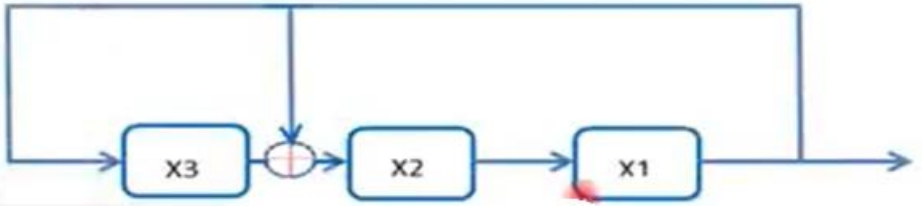
If the initial state of this LFSR is 00001, the following sequences will be produced.

00001	10011	01100	00010
10000	11001	10110	00001
01000	11100	11011	
00100	11110	11101	
10010	11111	01110	
01001	01111	10111	
10100	00111	01011	
11010	00011	10101	
01101	10001	01010	
00110	11000	00101	



Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm

Ex : LFSR of 3 flip flops and the polynomial $X2 = X1 \oplus X3$ and the initial value 111 .



X3	X2	X1
1	1	1
1	0	1
1	0	0
0	1	0
0	0	1
1	1	0
0	**1**	**1**
1	1	1

KeyStream : 1100101

Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm

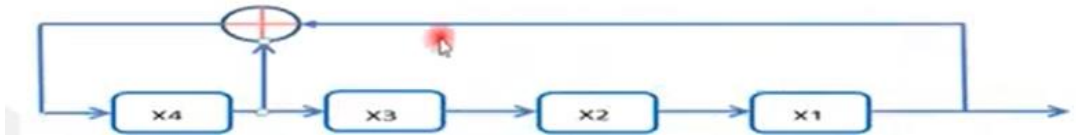
Example: Encrypt the plain text TOBE using Stream cipher and LFSR of 4-flip flops and feedback Function X4= X1 XOR X4

The Plain Text TOBE is Equal using Ascii table as follow

Letter	Ascii	Binary
T	84	1010100
O	79	1001111
B	66	1000010
E	69	1000101

Plain-Text: 10101001001111100001010001 01

Stream Cipher Technique
Linear Feedback Shift Register (LFSR) Algorithm



X4	X3	X2	X1
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
0	1	1	0
0	0	1	1
1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	1

PlainText: 10101001001111100001010001 01

Key: 0001111010110010

1010 1001 0011 1110 0001 0100 0101

0001 1110 1011 0010 0001 1110 1011

1011 0111 1000 1100 0000 1010 1110

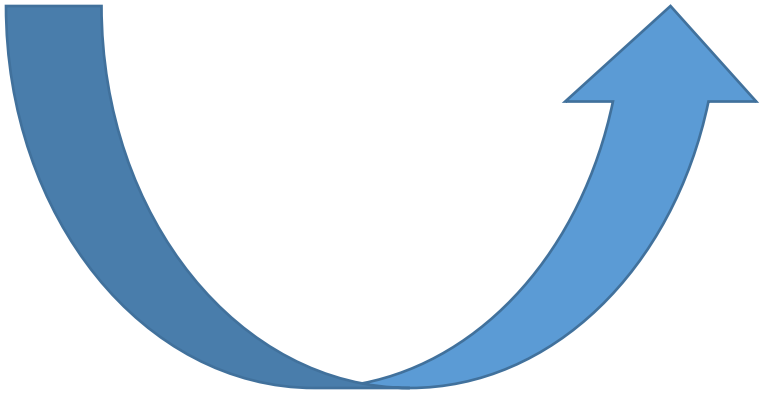
Data Encryption Standard (DES)

- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
  - Data are encrypted in 64-bit blocks using a 56-bit key
  - The algorithm transforms 64-bit input in a series of steps into a 64-bit output
  - The same steps, with the same key, are used to reverse the encryption

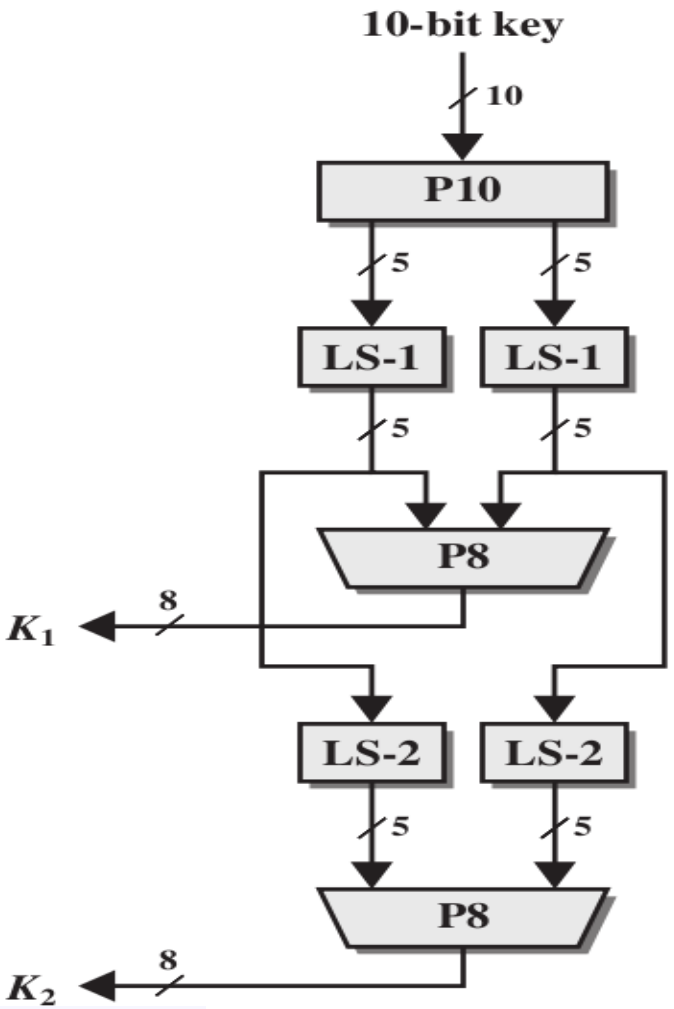
SDES	DES
8-bits block size	64-bits block size
10-bits key	56-bits key
2 round	16 rounds
8-bits round key	48-bits round key
IP-8 bits	IP: 64 bits
Function operates in 4 bits	F operates in 32 bits
2 S-Boxes	8 S-Boxes

Simplified Data Encryption Standard (SDES)

Next >>>



Simplified -DES Structure



Phase #1.Key Generation

K= 00101 10011 (10-bit)

P10= 3 5 2 7 4 10 1 9 8 6 (bit's number)

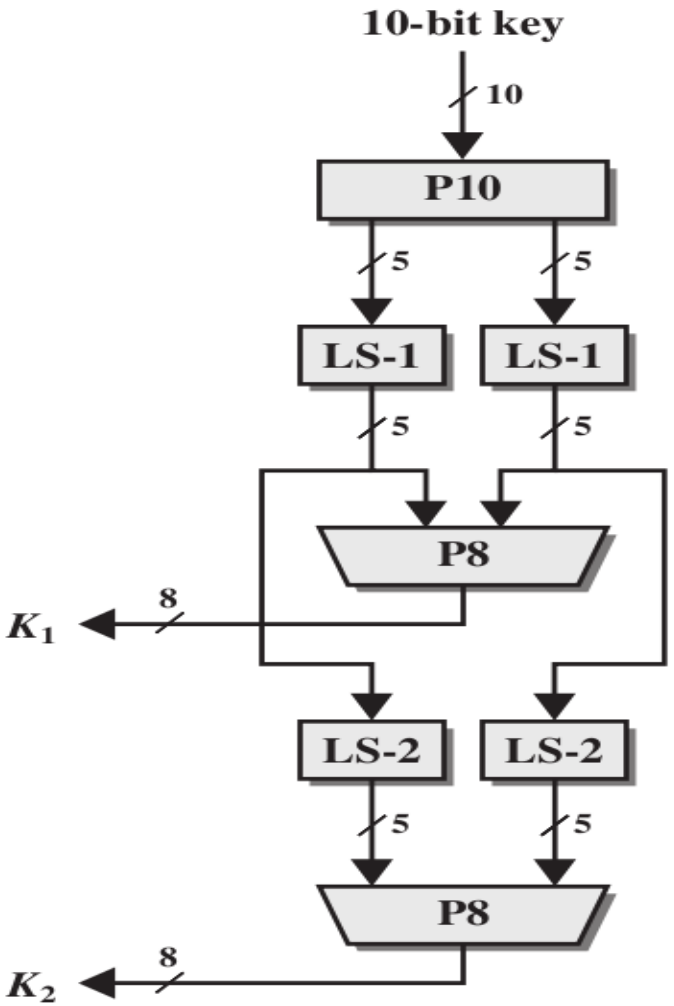
P8= 6 3 7 4 8 5 10 9

K	00101 10011
1. P10(K)	11000 10101
2. LS-1(P10(K))	10001 01011
3. P8(LS-1(P10(K)))	0010 0111
4. LS-2 (LS-1(P10(K))	00110 01101
5. P8 (LS-2)	0111 1010

$K_1 =$ 0010 0111

$K_2 =$ 0111 1010

Simplified -DES Structure Example 1



1.Key Generation

K= 10001 10101 (10-bit)

P10= 3 5 2 7 4 10 1 9 8 6

P8= 6 3 7 4 8 5 10 9

K	10001 10101
1. P10(K)	01000 11011
2. LS-1(P10(K))	10000 10111
3. P8(LS-1(P10(K)))	1000 1011
4. LS-2 (LS-1(P10(K))	00010 11110
5. P8 (LS-2)	1011 1001

$K_1 =$ 1000 1011

$K_2 =$ 1011 1001

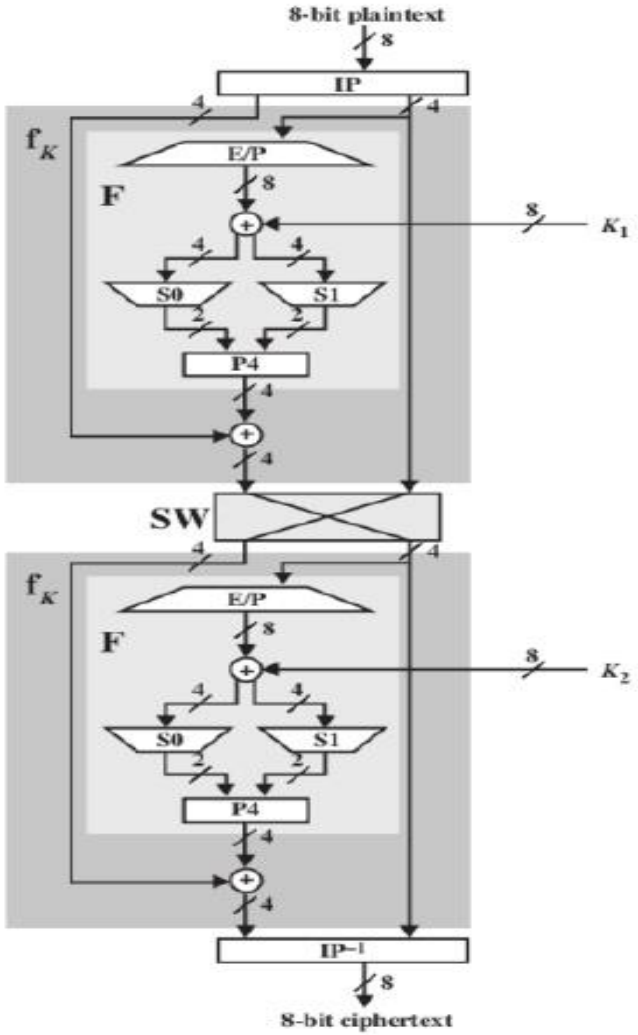
Simplified -DES Structure

Phase #2. Message encryption

fK(L,R) = (L Xor F(R, SK),R).

M= 1100 1001 **(8-bit)**

M	1100 1001
1. IP(M) Initial Permutation	1001 0110
2. Divide IP(M) into L and R	L_0= 1001 R_0= 0110
3. E/P(R_0) Expantion for Right side	0011 1100
K_1	0010 0111
4. E/P(R_0) $\oplus$ K_1	0001 1011
5. Sboxs(E/P(R_0) $\oplus$ K_1)	11 01
6. P_4(Sboxs(E/P(R_0) $\oplus$ K_1))	1101 (result from F)
$L_0 \oplus P_4$(Sboxs(E/P(R_0) $\oplus$ K_1))	0100



7. Calculating we then have fk1 (L;R) = (1001 $\oplus$ 1101; 0110) = (0100; 0110)

So far, then L = 0100 and R = 0110. SW just swaps them so R_1 = 0100 and L_1 = 0110.

Simplified -DES Structure cont….

We now do the calculation of f_{k2}(L,R) = $f_{\{0111\ 1010\}}(0110\ 0100)$ =(0110 ⊕ f (R,k2),R)

SW	0110 0100
Divide SW into L and R	L_1= 0110 R_1= 0100
E/P(R_1)	0010 1000
K_2	0111 1010
E/P(R_1) ⊕ K_2	0101 0010
Sboxs(E/P(R_1) ⊕ K_2)	01 01
P4(Sboxs(E/P(R_1) ⊕ K_2))	1100
(L_1⊕P4(Sboxs(E/P(R_1) ⊕ K_2))	(0110 ⊕1100)=1010
f_{k2}(L,R)	1010 0100
IP^{-1}	0110 0001

Simplified -DES Structure

P10									
3	5	2	7	4	10	1	9	8	6

P8							
6	3	7	4	8	5	10	9

IP							
2	6	3	1	4	8	5	7

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

E/P							
4	1	2	3	2	3	4	1

S0	00	01	10	11
00	1	0	3	2
01	3	2	1	0
10	0	2	1	3
11	3	1	3	2

S1	00	01	10	11
00	0	1	2	3
01	2	0	1	3
10	3	0	1	0
11	2	1	0	3

P4			
2	4	3	1

Full Example #2

Simplied DES algorithm

Let the plaintext be the string
M= 0010 1000 (28)hex
K=1100011110 (31E)hex

C=????????

1.Key Generation

K= 11000 11110 **(10-bit)**

P10= 3 5 2 7 4 10 1 9 8 6

P8= 6 3 7 4 8 5 10 9

K	1100011110
1. P10(K)	00110 01111
2. LS-1(P10(K))	01100 11110
3. P8(LS-1(P10(K)))	1110 1001
4. LS-2 (LS-1(P10(K))	10001 11011

$K_1 =$ 1110 1001
$K_2 =$ 1010 0111

Example #2 cont..

E/P							
4	1	2	3	2	3	4	1

S0	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

S1	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

2. Message encryption

M= 0010 1000 **(8-bit)**

fK(L,R) = (L Xor F(R, SK),R).

IP = 2 6 3 1 4 8 5 7

M	0010 1000
1. IP(M)	0010 0010
2. Divide IP(M) into L and R	L_0= 0010 R_0= 0010
3. E/P(R_0)	0001 0100
K_1	1110 1001
4. E/P(R_0) $\oplus$ K_1	1111 1101
5. Sboxs(E/P(R_0) $\oplus$ K_1)	10 00
6. P_4(Sboxs(E/P(R_0) $\oplus$ K_1))	0001 (result from F)
$L_0 \oplus P_4$(Sboxs(E/P(R_0) $\oplus$ K_1))	0011

L_1= 0010 R_1= 0011

	0010 1000
	0010 0010
2: Divide IP(M) into L and R	L_0= 0010 R_0= 0010
3. E/P(R_0)	0001 0100
K_1	1110 1001
4. E/P(R_0) $\oplus$ K_1	1111 1101
5. Sboxs(E/P(R_0) $\oplus$ K_1)	10 00
6. P_4(Sboxs(E/P(R_0) $\oplus$ K_1))	0001 (result from F)
(E/P(R_0) $\oplus$ K_1))	0011

P4			
2	4	3	1

Example #2 cont… find the mistakes here

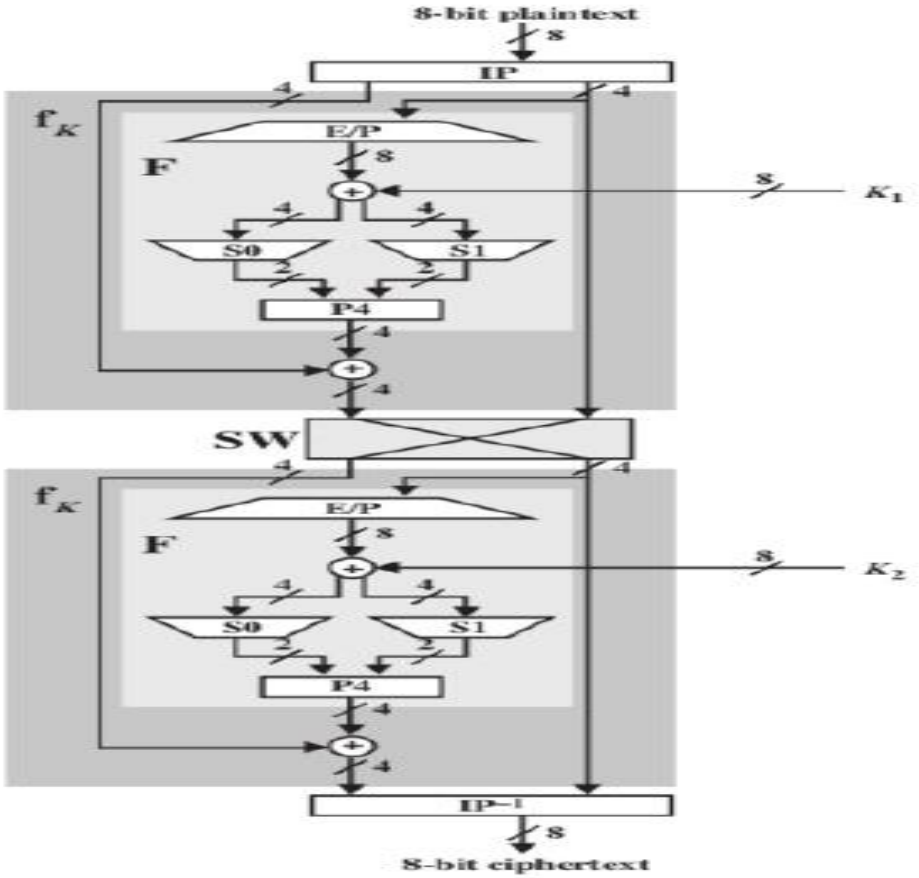
We now do the calculation of f_{k2}(L,R) = $f_{\{1010\ 0111\}}(0010\ 0101)$ =(0110 ⊕ f (R,k2),R)

SW	0010 0011
Divide SW into L and R	L1=0010 R1=0011
E/P(R_1)	1001 0110
K_2	1010 0111
E/P(R_1) ⊕ K_2	0011 0001
Sboxs(E/P(R_1) ⊕ K_2)	10 10
P4(Sboxs(E/P(R_1) ⊕ K_2))	0011
(L_1⊕P4(Sboxs(E/P(R_1) ⊕ K_2))	(0010(XOR)0011)=0001
f_{k2}(L,R)	0001 0011
IP^{-1}	1000 1010

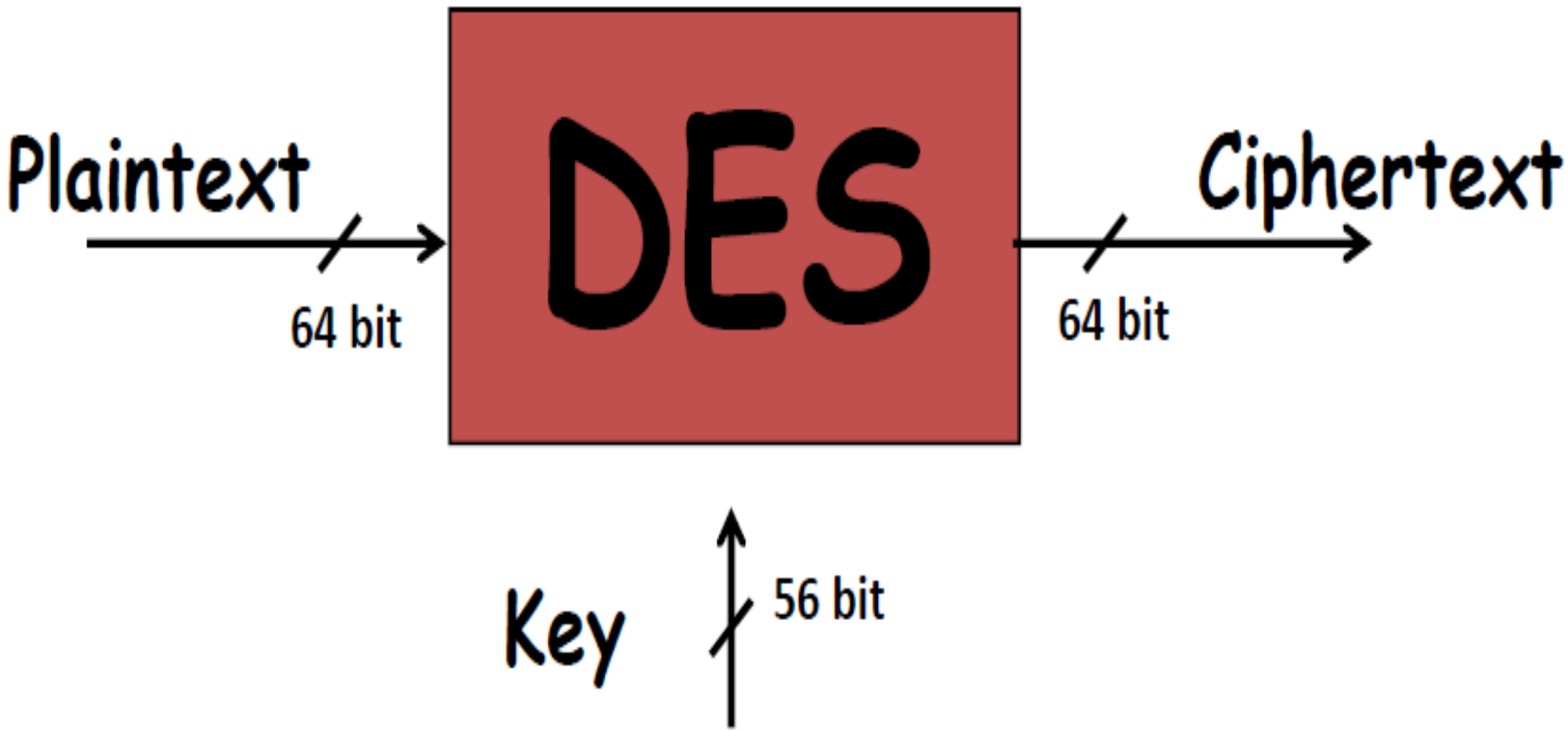
L_1= 0010 R_1= 0011

Training for Encryption and Decryption Using S-DES

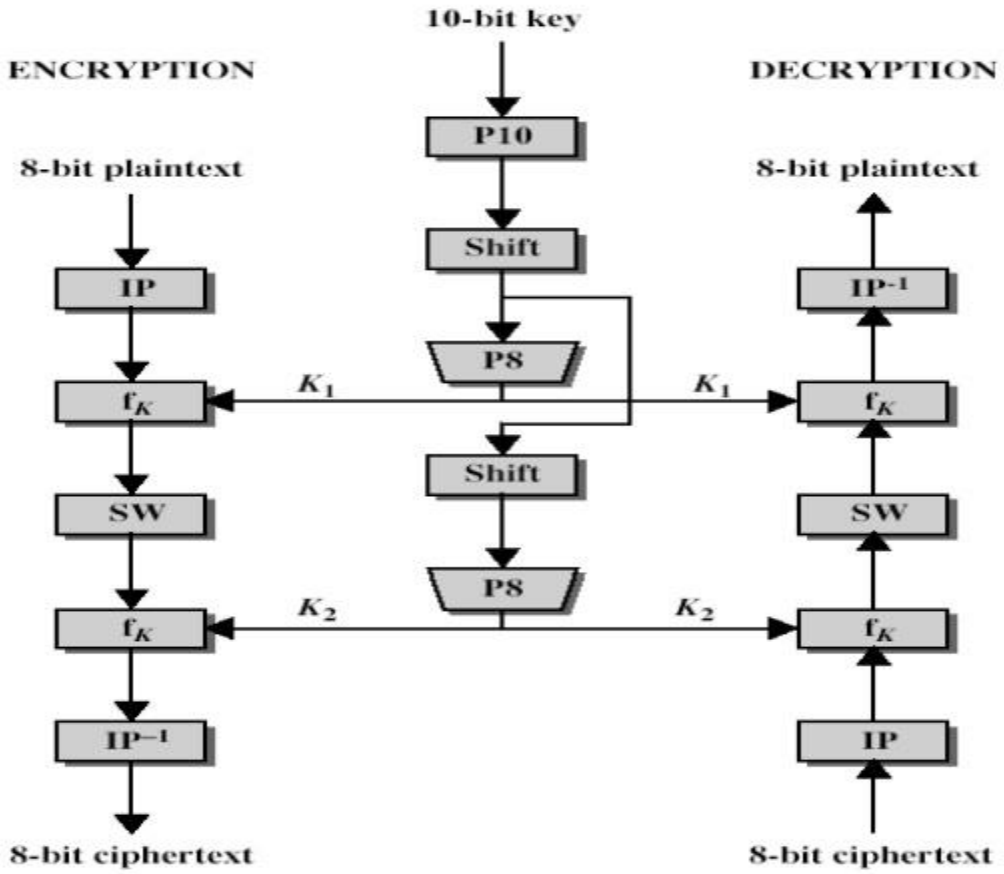
Solve The Following from : PT= 10010111 , K1 =10100100, and K2= 01000011



Data Encryption Standard (DES)



SDES- Key Creation process



Example:
Key: 1010000010, Find the Keys K1 and K2

P10									
Input :	1	2	3	4	5	6	7	8	9	10
Output :	3	5	2	7	4	10	1	9	8	6

P8							
6	3	7	4	8	5	10	9

KP: 1000001100
Kshifted:0000011001
Kp8: 10100010
Kshifted2: 0000110010
Kp8-2: 10000101
FirstKey is: 10100010
SecondKey: 10000101

Key Creation process

- The last bit of each byte is used as a parity bit.

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 -1

K= 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

K_p=1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

The Key generated will be divided into two parts, Left part and Right part, where as the process of generating Kp from original K is based on the matrix, such as first element in Kp refers to 57, second element refers to 49 and so on.

The process of generating Kp from K depends on the rank of K in the permutation matrix.

Key Creation process

Round Number	*Number of left shifts*
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

The Round Table shown here refer to mapping between number of rounds and the left shift of original message, whereas the round number one will do one shift for left side, in round number 3 the left side will do three shifts.

- L = 1111000 0110011 0010101 0101111
- R = 0101010 1011001 1001111 0001111

K[1].L=11100001100110010101010111111	K[1].R = 10101010110011001111100011110
K[2].L=11000011001100101010101111111	K[2].R = 01010101100110011111000111101
K[3].L = 00001100110010101010111111111	K[3].R = 01010110011001111100011110101
K[4].L = 00110011001010101011111111100	K[4].R = 01011001100111100011110101010
K[5].L = 11001100101010101111111110000	K[5].R = 01100110011110001111010101010
K[6].L = 00110010101010111111111000011	K[6].R = 10011001111000111101010101010

Key Creation process

Permutation Choice 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

K[1]=1110000110011001010101011111110101010110011001111100011110
***K1* = 000110 110000 001011 101111 111111 000111 000001 110010**

K[2]=1100001100110010101010111111010101011001100111100011101
K2 = 011110 011010 111011 011001 110110 111100 100111 100101

K[3]= 0000110011001010101011111111010101100110011110001111010 1
K3 = 010101 011111 110010 001010 010000 101100 111110 011001

K[4]= 0011001100101010101111111100010110011001111000111101010 1
K4 = 011100 101010 110111 010110 110110 110011 010100 011101

After finishing the 16 round of shifting process based on the mentioned table, both left and right side of key will be concatenating , which generate 16 different key.
After that the keys will go through other rounds to generate other key by applying the permutation choice 2 table on the entered key.

As Summary
The process of DES Block cipher key creation goes through set of round using permutation choice
First step refers to decrease the size of key from 64 bit to 56 bit using permutation choice
Second Step Divide the KP into Left side and Right Side Rotate shift
Third Step Concatenate lift and right
Fourth Step The key generated after shifting will go through other round of shifting.
Result The output of these steps refers to generate 16 new keys

DES Structure Example 1

DES(Msg Encryption)
Message M = 1 1 0 1 1 0 0 1.
K1= 10100010.
First Step refers to Apply IP(Initial permutation) on the message.
IP = 2 6 3 1 4 8 5 7.
MP (After permutation) = 1 0 0 1 1 1 1 0
Divide the Message MP into two sides Left Side L0 and Right side R0.
Find Both L1 and R1 L = 1001, R= 1110
L1 = R0 = 1110
R1= f(L,R) = (L0 XOR F(R,K1),R0)
The problem here that R1 is 4 digits, and K1 is 8 digits, how can do XORing
The R0 must be expanded to be 8 bits can do XOR with K1
The Expansion process is done based on

E/P							
4	1	2	3	2	3	4	1

R1 = 01111101 this after expansion
After that do XOR between K1 XOR R1
0 1 1 1 1 1 0 1
1 0 1 0 0 0 1 0
1 1 0 1 1 1 1 1 = F(R,K1)
After that decrease the number of digits for R1 to be 4 bits rather than 8 bits(The process of decreasing the size of R1 is depends on S-box).
Divide the R1 into two parts
1101 1111
11 = 3
01= 2
Now find the intersection in S0 between 3 and 2 which is 3 = 11, and same thing for S1 for 1111 = 3

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

DES Structure Example 1

After that do XOR between K1 XOR R1
0 1 1 1 1 1 0 1
1 0 1 0 0 0 1 0
1 1 0 1 1 1 1 1 = F(R,K1)
After that decrease the number of digits for R1 to be 4 bits rather than 8 bits(The process of decreasing the size of R1 is depends on S-box).
Divide the R1 into two parts
1101 1111
11 = 3
01= 2
Now find the intersection in S0 between 3 and 2 which is 3 = 11, and same thing for S1 for 1111 = 3
The R1 = 1111 After decreasing
Then do XOR based on original equation between L0 and Rnew
1001 XOR 1111 = 0110
Then XOR between 0110 XOR 1110(R0)
The final result in 1000 = R1
Then L1 =1110
R1 = 1000
Then Concatenate R1+L1(Reverse of original)
100011110
Final Permutation (IP^-1)
Based on original IP (IP = 2 6 3 1 4 8 5 7).
2 in index1 then index2 is 1, 6 in index2 then index 6 is , 3 in index three then three will be in index 3 and so on
IP inverse is 4 1 3 5 7 2 8 6
Then final permutation
01011001 this is the message after encryption

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

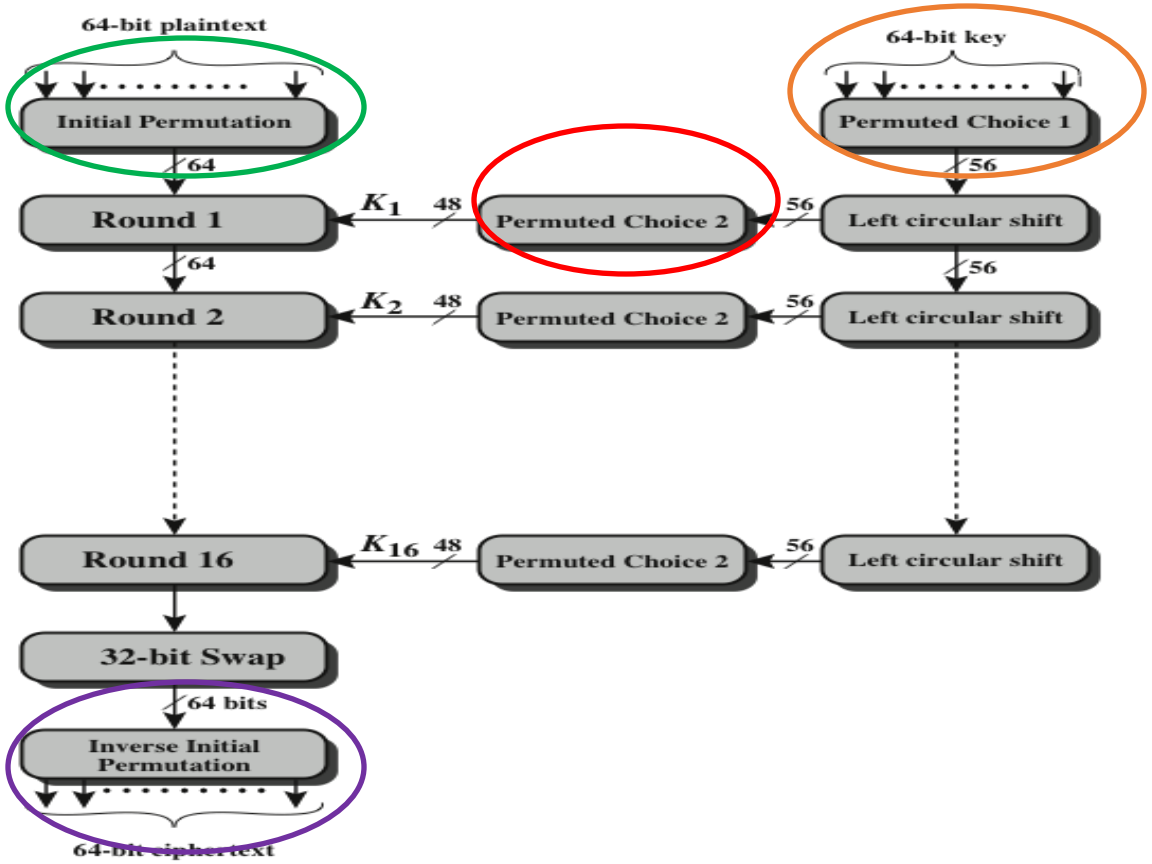


Figure 4.5 General Depiction of DES Encryption Algorithm

Permuted choice 1 PC1

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

Permuted choice 2 PC2

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

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

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

Data Encryption Standard (DES) Example

Step 1: key creation.

- M= 0123 4567 89AB CDEF (64bit)
- K= 1334 5779 9BBC DFF1
- K=0001001100110100 0101011101111001 1001110011001101 1101111111110001

we get the 56-bit permutation

- K_p = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

split this key into left and right halves, K_{pL} and K_{pR}, where each half has 28 bits.

- K_{pL}= 1111000 0110011 0010101 0101111
- K_{pR}= 0101010 1011001 1001111 0001111

Permuted choice 1 PC1

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

K[1].L=11100001100110010101011111	**K[1].R**=10101010110011001111100011110
K[2].L=11000011001100101010101111111	**K[2].R**=01010101100110011111000111101
K[3].L=00001100110010101010111111111	**K[3].R**=01010110011001111000111110101
K[4].L=00110011001010101011111111100	**K[4].R**=01011001100111100011110101010
K[5].L=11001100101010101111111110000	**K[5].R**=01100110011110001111010101010
K[6].L=00110010101010111111111000011	**K[6].R**=10011001111000111101010101010
K[7].L=11001010101011111111100001100	**K[7].R**=01100111100011110101010110
K[8].L=00101010101111111110000110011	**K[8].R**=10011110001111010101011001
K[9].L=01010101011111111100001100110	**K[9].R**=00111100011110101010110011
K[10].L=01010101111111100001100110011001	**K[10].R**=11110001111010101011001100
K[11].L=01010111111110000110011001010	**K[11].R**=11000111101010101100110011
K[12].L=01011111111000011001100101010	**K[12].R**=00011110101010101100110011111
K[13].L=01111111100001100110010101010	**K[13].R**=01111010101010110011001111100
K[14].L=11111110000110011001010101010	**K[14].R**=11101010101011001100111100011
K[15].L=11111000011001100101010101111	**K[15].R**=10101010101100110011111000111
K[16].L=11110000110011001010101011111	**K[16].R**=01010101011001100111100011111

56 bits

Round Number	Number of left shift
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

K[1]=1110000110011001010101011111 10101010110011001111100011110
K[2]=1100001100110010101010111111 01010101100110011111000111101
K[3]=0000110011001010101011111111 01010110011001111000111101 01
K[4]=0011001100101010101111111100 01011001100111100011110101 01
K[5]=1100110010101010111111110000 01100110011110001111010101 01
K[6]=0011001010101011111111000011 10011001111000111101010101 01
K[7]=1100101010101111111100001100 01100111100011110101010110 01
K[8]=0010101010111111110000110011 10011110001111010101011001 01
K[9]=0101010101111111100001100110 00111100011110101010110011 01
K[10]=0101010111111110000110011001 11110001111010101011001100 01
K[11]=0101011111111000011001100101 11000111101010101100110011 01
K[12]=0101111111100001100110010101 00011110101010110011001111 01
K[13]=0111111110000110011001010101 01111010101010110011001111 00
K[14]=1111111000011001100101010101 11101010101011001100111100 01
K[15]=1111100001100110010101010111 10101010101100110011110001 11
K[16]=1111000011001100101010101111 01010101011001100111100011 11

We ‘ve got 16 key each one will enter to permutation PC2

K[1]=000110 11000 000101 110111 1111111 000111 000001 110010
K[2]=0111110 011010 111011 011001 110110 111100 10011 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 01001 111110 101011 101001 000001
K[14]=010111 110100 001110 110111 111100 101110 01110 111010
K[15]=101111 111001 000110 001101 001111 010011 111100 001010
K[16]=110010 110011 110110 001011 000011 100001 011111 110101

Permuted choice 2 PC2

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

We finished the key creation

Step 2: Encode each 64-bit block of data.

M = 0123 4567 89AB CDEF

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 (64-bit)

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

L_0= 1100 1100 0000 0000 1100 1100 1111 1111 (32-bit)

R_0= 1111 0000 1010 1010 1111 0000 1010 1010 (32-bit)

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010 (48-bit)

L_1 = R_0= 1111 0000 1010 1010 1111 0000 1010 1010

R_1 = L_0 + $\boldsymbol{f}(R_0, K_1)$

Expainsion permutation

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010 (32-bit)

E(R_0)= 011110 100001 010101 010101 011110 100001 010101 010101 (48-bit)

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

E(R_0) **xor** K_1= 011000 010001 011110 111010 100001 100110 010100 100111.

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111

E(R_0) **xor** K_1= 011000 010001 011110 111010 100001 100110 010100 100111.

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111

F=0010 0011 0100 1010 1010 1001 1011 1011
L_0=1100 1100 0000 0000 1100 1100 1111 1111

F xor L_0=1110 1111 0100 1010 0100 0101 0100 0100

L1 (32bit)	R1 (32bit)
1111 0000 1010 1010 1111 0000 1010 1010	1110 1111 0100 1010 0100 0101 0100 0100

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

S-box 1

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

011000

DES S-boxes 1 through 8.

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-box 1

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-box 2

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-box 3

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-box 4

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-box 5

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box 6

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-box 7

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S-box 8

We reverse the order of these two blocks and apply the final permutation to
$\boldsymbol{R_{16}L_{16}}$= 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100
$\boldsymbol{Ip^{-1}}$= 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101
which in hexadecimal format is
85E813540F0AB405.

This is the encrypted form of
M = 0123456789ABCDEF:
C = 85E813540F0AB405

The Difference between DES and Simplified -DES

Parameters	**DES**	**S-DES**
M length	64 bits	8 bits
C length	64 bits	8 bits
K length	56 bits	10 bits
Number of Round	16 round	2 Round
Number of Sub Keys	16 keys (48 bits)	2 keys (8 bits)
S-boxes	8 boxes 6 bits -> 4 bits	2 boxes 4 bits -> 2 bits

Block Cipher Design Principles: Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack

If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search

Block Cipher Design Principles:
Design of Function F

- The heart of a Feistel block cipher is the function F
- The more nonlinear F, the more difficult any type of cryptanalysis will be
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function
- The algorithm should have good avalanche properties

Strict avalanche criterion (SAC)

States that any output bit j of an S-box should change with probability 1/2 when any single input bit i is inverted for all i , j

Bit independence criterion (BIC)

States that output bits j and k should change independently when any single input bit i is inverted for all i , j , and k

Block Cipher Design Principles: Key Schedule Algorithm

- With any Feistel block cipher, the key is used to generate one subkey for each round
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key
- It is suggested that, at a minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion

DES Weak Keys

- DES uses 16 48-bits keys generated from a master 56- bit key (64 bits if we consider also parity bits)
- Weak keys: keys make the same sub-key to be generated in more than one round.
- Result: reduce cipher complexity
- Weak keys can be avoided at key generation.
- • DES has 4 weak keys
- – 01010101 01010101
- – FEFEFEFE FEFEFEFE
- – E0E0E0E0 F1F1F1F1
- – 1F1F1F1F 0E0E0E0E

Strength of DES

- Timing attacks
  - 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 ciphertexts
  - Exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs
  - So far it appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES

The Needs for New Versions of DES

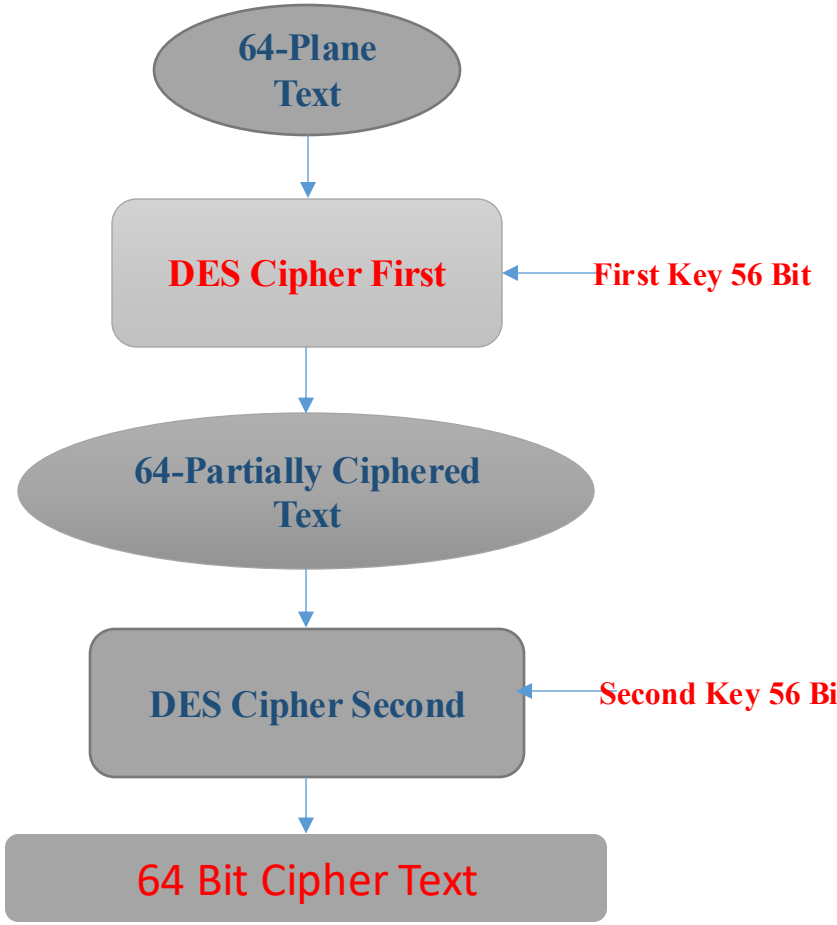
- The Complexity of Algorithm, since the DES algorithm was cracked in 1997 by a developer at IBM.
- With today's rapid advancements in technology and resources, the algorithm can be readily cracked.
- The Length of used key is 56 bit which means 2^56 probability for the key to be right one, this make the brut force attack succeeded to crack the algorithm.
- The number of S-Boxes is not enough now days with the highly development of resources such as CPUs and Memories.

Solution Suggested to enhance DES

- A lot of researchers in the filed of encryption theories tries to find and enhance the original DES algorithm to be used in the process of Ciphering, some of these algorithms are used and some other is discarded
- Double DES algorithm, Triple DES algorithm refers to the famousness of these developed algorithms.
- How each of Double and Triple DES is developed, and how it is different from the original DES algorithm.

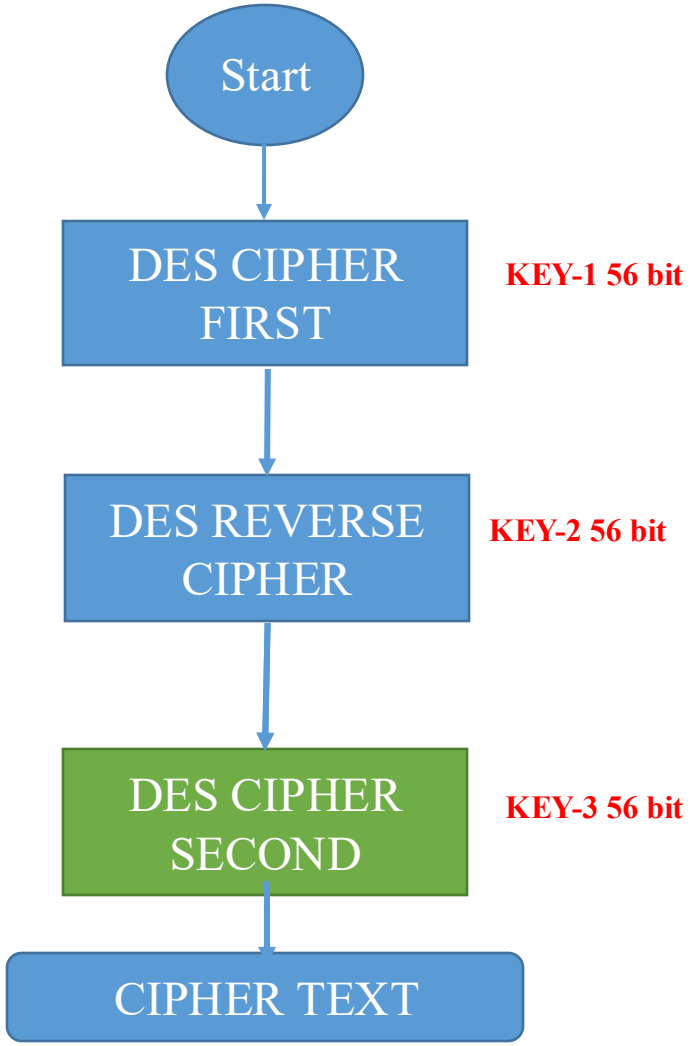
Double DES Algorithm

- Double DES is an encryption technique which uses two instance of DES on same plain text. In both instances it uses different keys to encrypt the plain text. Both keys are required at the time of decryption. The 64-bit plain text goes into first DES instance which than converted into a 64-bit middle text using the first key and then it goes to second DES instance which gives 64-bit cipher text by using second key.
- **Cipher Text= $E_{k2}(E_{k1}(PT))$**
- **Plain Text= $D_{k2}(D_{k2}(CT))$**
- **Meet In The Middle Attack is the problem that face the Double DES**

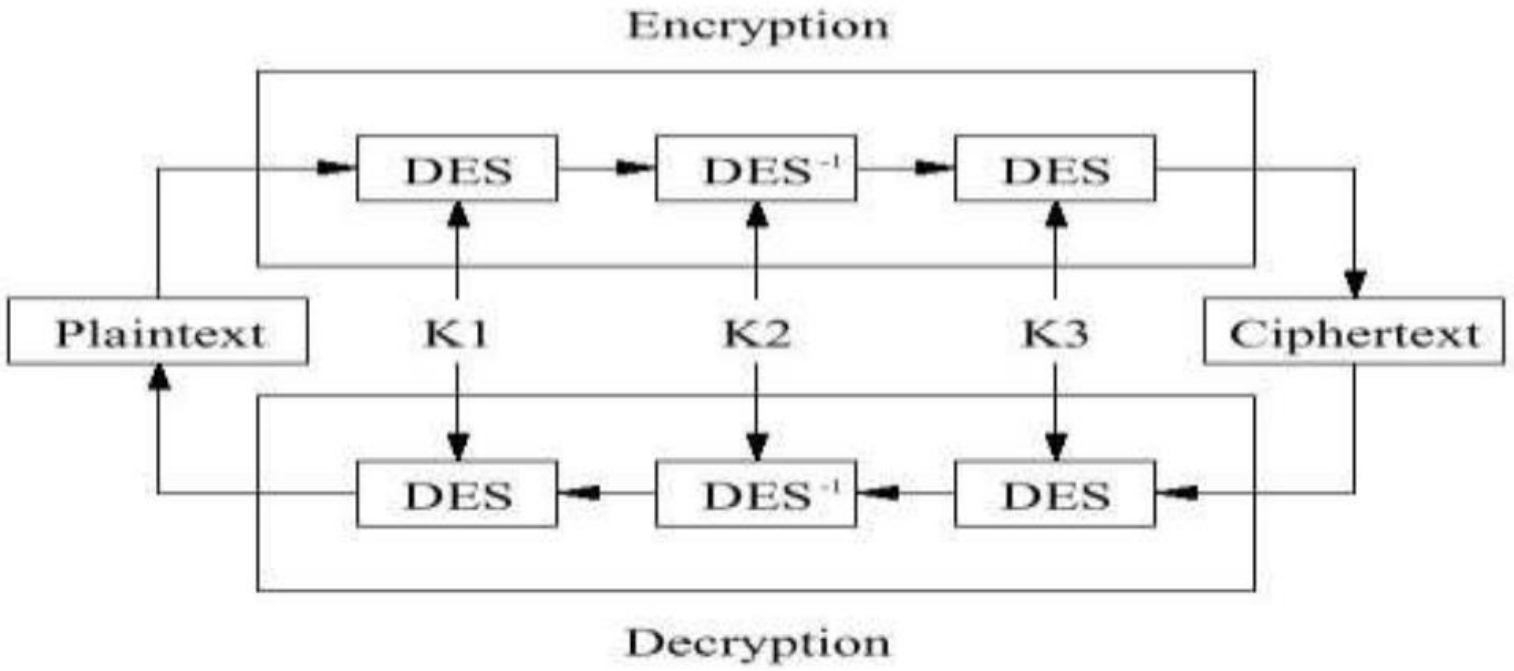


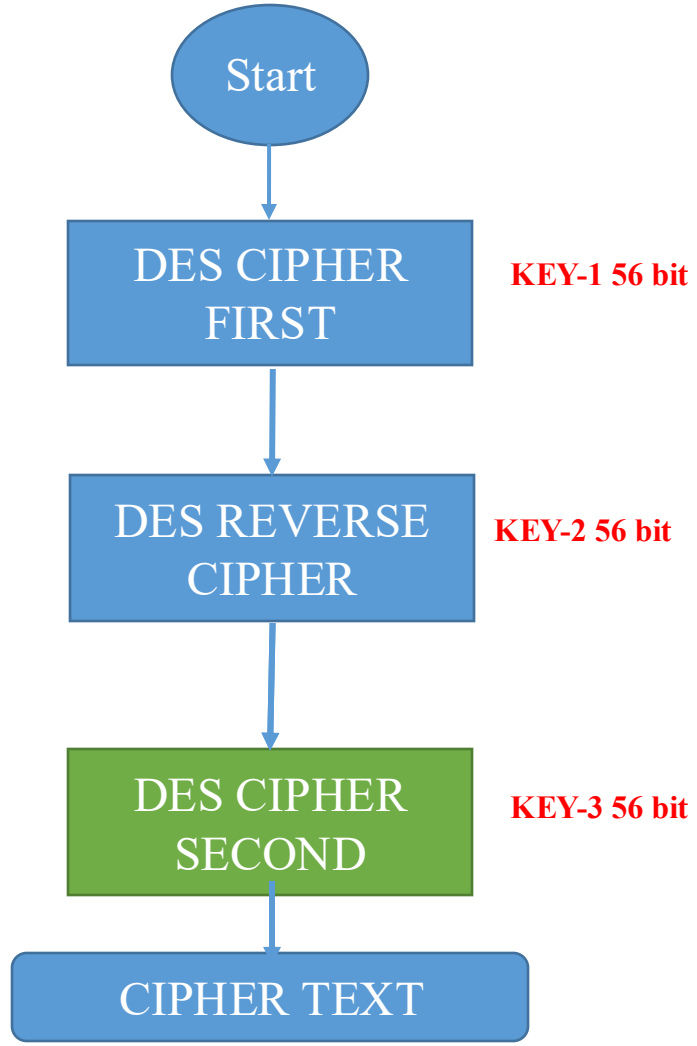
Triple DES

- Triple DES
  - Triple DES is a encryption technique which uses three instance of DES on same plain text. It uses their different types of key choosing technique in first all used keys are different and in second two keys are same and one is different and in third all keys are same.
  - Triple DES is also vulnerable to meet-in-the middle attack because of which it give total security level of 2^112 instead of using 168 bit of key. The block collision attack can also be done because of short block size and using same key to encrypt large size of text. It is also vulnerable to sweet32 attack.
  - **Cipher Text = $E_{k3}(D_{k2}(E_{k1}(Text)))$**
  - **Plain Text = $(D_{k1}(E_{k2}(D_{k3}(Cipher\ text)))$**



Triple DES





Summary

- **Traditional Block Cipher Structure**
  - Stream ciphers
  - Block ciphers
  - Motivation for the Feistel cipher structure
  - Feistel cipher
- **The Data Encryption Standard (DES)**
  - Encryption
  - Decryption
  - Avalanche effect

- **The strength of DES**
  - Use of 56-bit keys
  - Nature of the DES algorithm
  - Timing attacks
- **Block cipher design principles**
  - Number of rounds
  - Design of function F
  - Key schedule algorithm

Advanced Encryption Standard

AES Origins

"It seems very simple."

"It is very simple. But if you don't know what the key is it's virtually indecipherable."

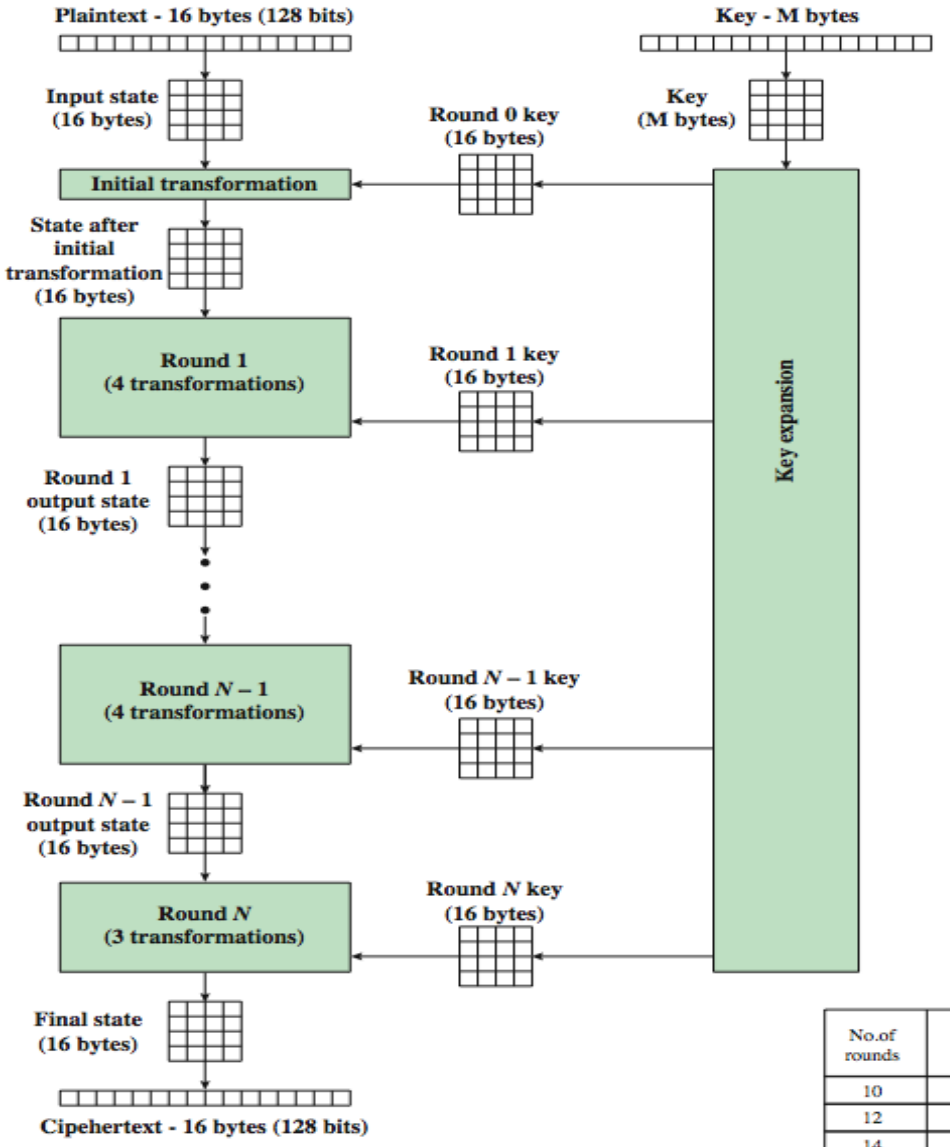
***—Talking to Strange Men,* Ruth Rendell**

- clearly a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

The AES Cipher - Rijndael

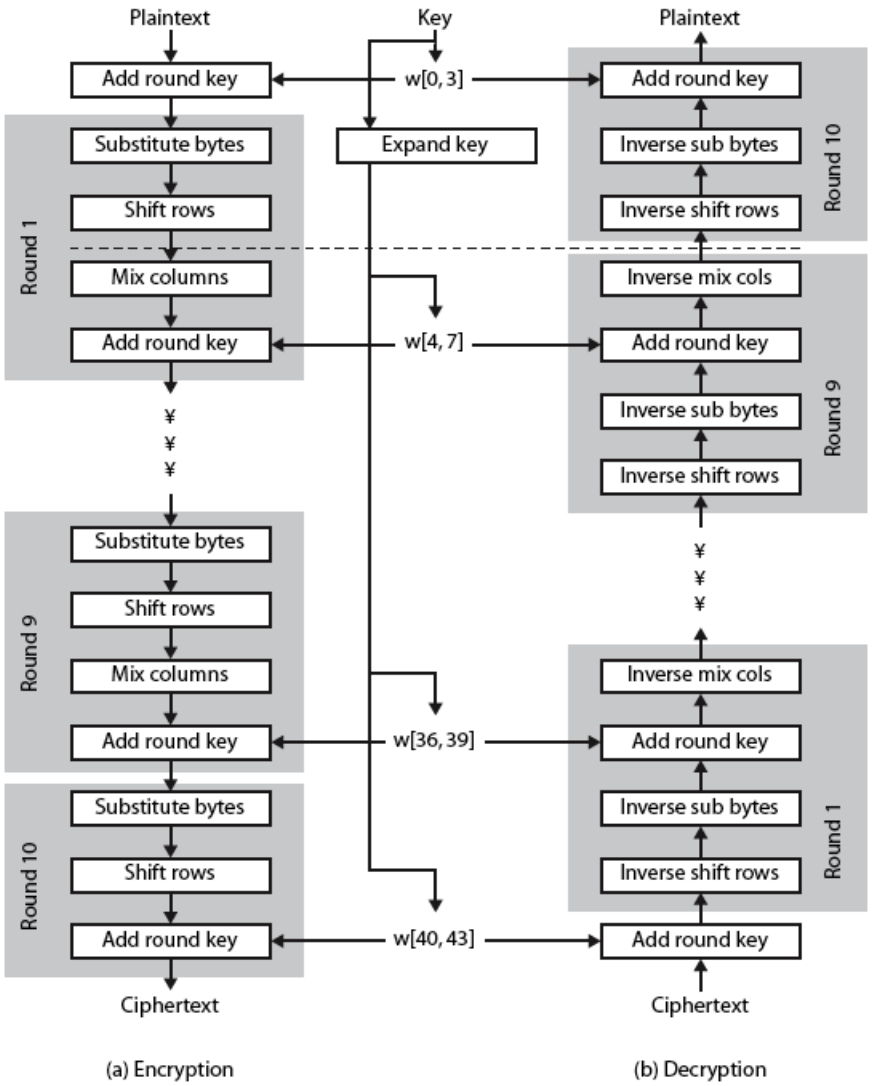
- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to have:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

AES Encryption Process



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

AES Structure



AES Steps :

Step1: Convert Plaintext and key into Hexadecimal (from block to state)

Step2 :- Add Round key, Round 0

Step 3: Substitution Bytes (S-box) – Round 1

Step 4: Shift Row – Round 1

Step 5: Mix column – Round 1

Step 6: Add Round Key- Round 1

Excerpt of ASCII Chart

032	Space	048	0	064	@	080	P	096	‘	112	p
033	!	049	1	065	A	081	Q	097	a	113	q
034	"	050	2	066	B	082	R	098	b	114	r
035	#	051	3	067	C	083	S	099	c	115	s
036	$	052	4	068	D	084	T	100	d	116	t
037	%	053	5	069	E	085	U	101	e	117	u
038	&	054	6	070	F	086	V	102	f	118	v
039	’	055	7	071	G	087	W	103	g	119	w
040	(	056	8	072	H	088	X	104	h	120	x
041	)	057	9	073	I	089	Y	105	i	121	y
042	*	058	:	074	J	090	Z	106	j	122	z
043	+	059	;	075	K	091	[	107	k	123	{
044	‘	060	<	076	L	092	\	108	l	124	|
045	-	061	=	077	M	093	]	109	m	125	}
046	.	062	>	078	N	094	^	110	n	126	~
047	/	063	?	079	O	095	_	111	o	127	DEL

Hexa Decimal Value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Example : A in ASCII **= 65 W= 87 and so on…**

Example1

Step1:

Plaintext :Hello Students L

Key : Go to the GYM $?

Plaintext

H	e	l	l	o		S	t	u	d	e	n	t	s		L
48	65	6C	6C	6F	20	53	74	75	64	65	6E	74	73	20	4C

Key

G	o		t	o		t	h	e		G	Y	M		$	?
47	6F	20	74	6F	20	74	68	65	20	47	59	4D	20	24	3F

Row, Column From Hexa Decimal Value

$$\text{Plaintext}=\begin{pmatrix} 48 & 6F & 75 & 74 \\ 65 & 20 & 64 & 73 \\ 6C & 53 & 65 & 20 \\ 6C & 74 & 6E & 4C \end{pmatrix}$$

$$\text{Key}=\begin{pmatrix} 47 & 6F & 65 & 4D \\ 6F & 20 & 20 & 20 \\ 20 & 74 & 47 & 24 \\ 74 & 68 & 59 & 3F \end{pmatrix}$$

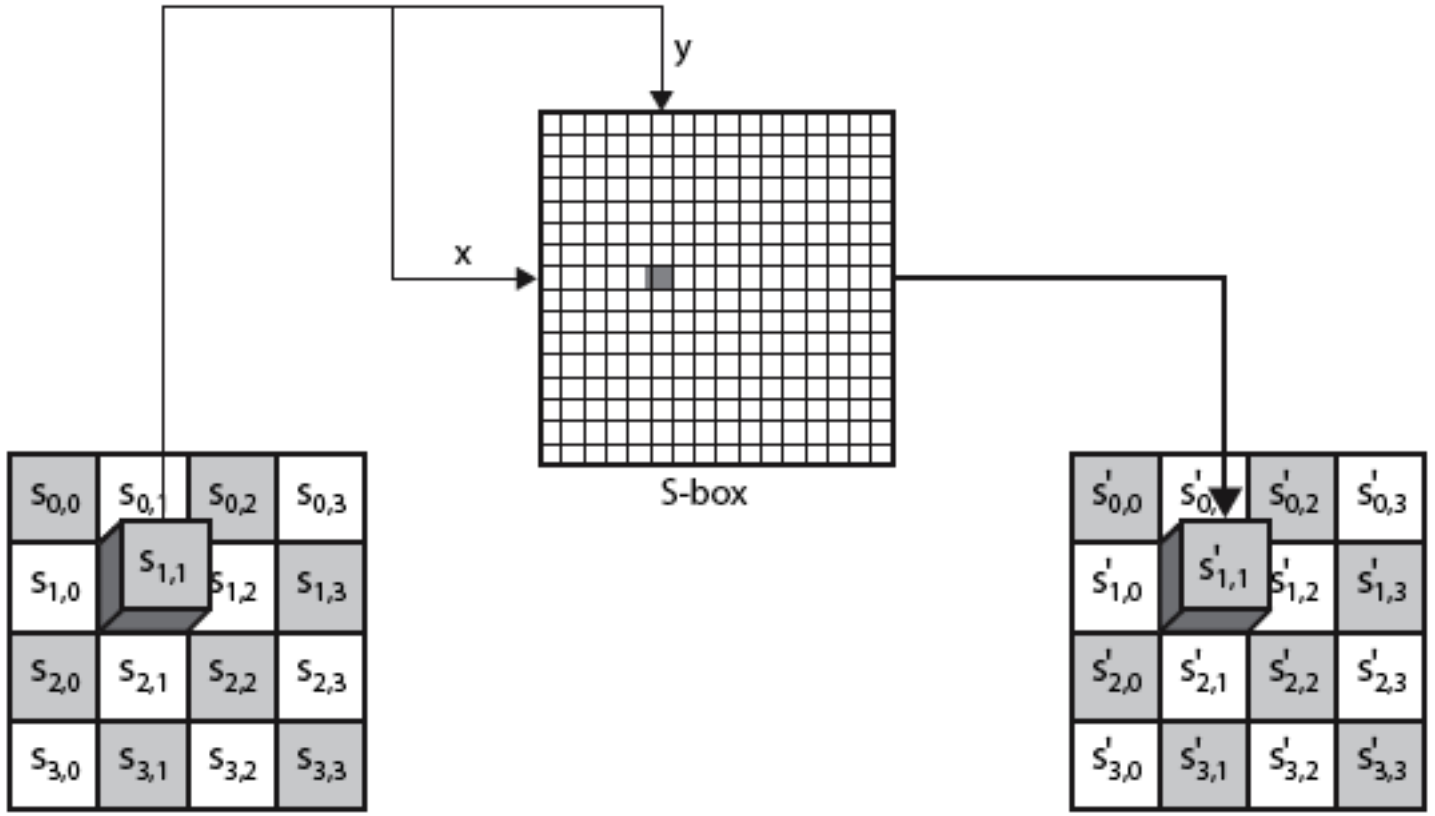
AES Example #1

Step1:

1.1 determine the state matrix The Do the Add Round Key Step

$$\text{Plaintext}=\begin{pmatrix}48 & 6F & 75 & 74\\ 65 & 20 & 64 & 73\\ 6C & 53 & 65 & 20\\ 6C & 74 & 6E & 4C\end{pmatrix} \oplus \text{Key}=\begin{pmatrix}47 & 6F & 65 & 4D\\ 6F & 20 & 20 & 20\\ 20 & 74 & 47 & 24\\ 74 & 68 & 59 & 3F\end{pmatrix} = \text{State Matrix}=\begin{pmatrix}0F & 00 & 10 & 39\\ 0A & 00 & 44 & 53\\ 4C & 27 & 22 & 04\\ 18 & 1C & 37 & 73\end{pmatrix}$$

Substitute Bytes Step



Substitute Bytes Example

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

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

AES S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	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

For example HEX 19 would get replaced with HEX D4

Example1 cont…

Step 2:

substitute each entry (byte) of current state matrix by corresponding entry in AES S-Box for instance:

$$\text{State Matrix} = \begin{pmatrix} 0F & 00 & 10 & 39 \\ 0A & 00 & 44 & 53 \\ 4C & 27 & 22 & 04 \\ 18 & 1C & 37 & 73 \end{pmatrix}$$

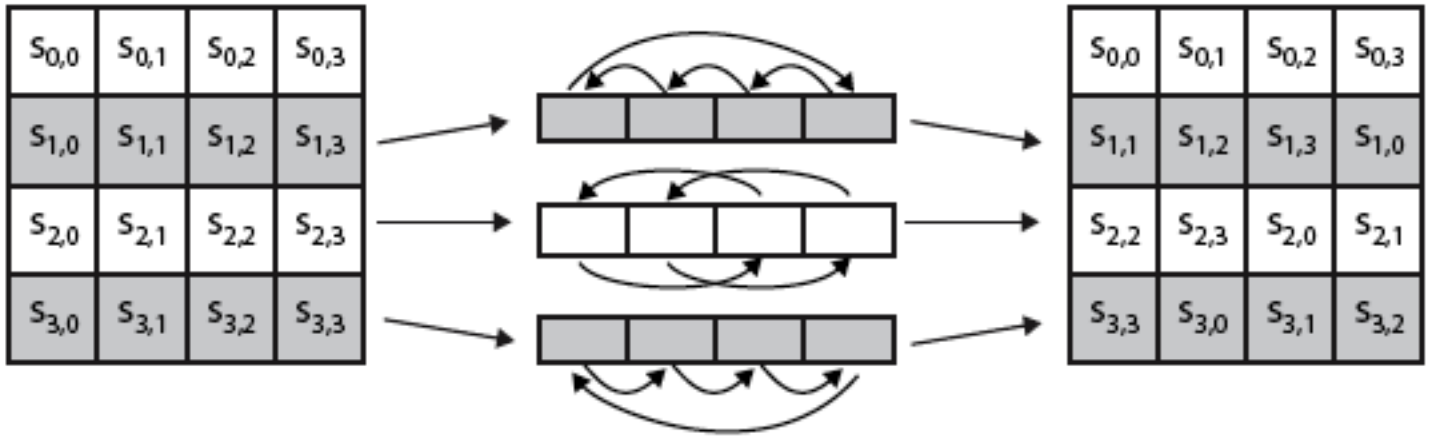
$$\text{New State Matrix} = \begin{pmatrix} 76 & 63 & CA & 12 \\ 67 & 63 & 1B & ED \\ 29 & CC & 93 & F2 \\ AD & 9C & 9A & 8F \end{pmatrix}$$

AES S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	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

For example HEX 19 would get replaced with HEX D4

Shift Rows



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

Example1 cont…

Step 3:

Shift rows:

- Arranges the state in a New State Matrix and then performs a **circular shift** for each row
- **First row is never shifted.**

$$\text{New State Matrix} = \begin{pmatrix} 76 & 63 & CA & 12 \\ 67 & 63 & 1B & ED \\ 29 & CC & 93 & F2 \\ AD & 9C & 9A & 8F \end{pmatrix}$$

$$\text{New Shifted State Matrix} = \begin{pmatrix} 76 & 63 & CA & 12 \\ 63 & 1B & ED & 67 \\ 93 & F2 & 29 & CC \\ 8F & AD & 9C & 9A \end{pmatrix}$$

Example1 cont…

Step 4:

Mix column

New Shifted State Matrix=$\begin{pmatrix} \mathbf{76} & \mathbf{63} & \boldsymbol{CA} & \mathbf{12} \\ \mathbf{63} & \mathbf{1B} & \boldsymbol{ED} & \mathbf{67} \\ \mathbf{93} & \boldsymbol{F2} & \mathbf{29} & \boldsymbol{CC} \\ \mathbf{8F} & \boldsymbol{AD} & \mathbf{9C} & \mathbf{9A} \end{pmatrix} * \begin{pmatrix} 02 & 03 & 01 & 01 \\ 0\,1 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

*

b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15
b4	b8	b12	b16

```
b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)
b2 = (b1 * 1) XOR (b2*2) XOR (b3*3) XOR (b4*1)
b3 = (b1 * 1) XOR (b2*1) XOR (b3*2) XOR (b4*3)
b4 = (b1 * 3) XOR (b2*1) XOR (b3*1) XOR (b4*2)
```

(b1= specifies the first byte of the state)

```
b5 = (b5 * 2) XOR (b6*3) XOR (b7*1) XOR (b8*1)
b6 = (b5 * 1) XOR (b6*2) XOR (b7*3) XOR (b8*1)
b7 = (b5 * 1) XOR (b6*1) XOR (b7*2) XOR (b8*3)
b8 = (b5 * 3) XOR (b6*1) XOR (b7*1) XOR (b8*2)
```

And so on until all columns of the state are exhausted.

Example1 cont…

Step 4:

Mix column

New Shifted State Matrix=$\begin{pmatrix} \boldsymbol{76} & \boldsymbol{63} & \boldsymbol{CA} & \boldsymbol{12} \\ \boldsymbol{63} & \boldsymbol{1B} & \boldsymbol{ED} & \boldsymbol{67} \\ \boldsymbol{93} & \boldsymbol{F2} & \boldsymbol{29} & \boldsymbol{CC} \\ \boldsymbol{8F} & \boldsymbol{AD} & \boldsymbol{9C} & \boldsymbol{9A} \end{pmatrix} * \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$

b_1=(76*02) ⊕ (63*03) ⊕ (93*01) ⊕(8F*01)=(⊕ ⊕ ⊕)=

Result=???

Example1 cont… Step 4: Mix column

New Shifted State Matrix= $\begin{pmatrix} \mathbf{76} & \mathbf{63} & \boldsymbol{CA} & \mathbf{12} \\ \mathbf{63} & \boldsymbol{1B} & \boldsymbol{ED} & \mathbf{67} \\ \mathbf{93} & \boldsymbol{F2} & \mathbf{29} & \boldsymbol{CC} \\ \boldsymbol{8F} & \boldsymbol{AD} & \boldsymbol{9C} & \boldsymbol{9A} \end{pmatrix} * \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$

b_1=(76*02) ⊕ (63*03) ⊕ (93*01) ⊕(8F*01)=(EC⊕A4 ⊕CA ⊕12)=90

b_2=(76*01) ⊕ (63*02) ⊕ (93*03) ⊕(8F*01)=(76 ⊕C6 ⊕ 45⊕12)=E7

b_3=(76*01) ⊕ (63*01) ⊕ (93*02) ⊕(8F*03)=(76⊕ 63⊕3D ⊕8B)=AC

b_4=(76*03) ⊕ (63*01) ⊕ (93*01) ⊕(8F*02)=(9A⊕63 ⊕93⊕05)=17

b_5=(63*02) ⊕ (1B*03) ⊕ (ED*01) ⊕(67*01)=(6C⊕ 3C⊕ ED⊕67)=DA

b_6=(63*01) ⊕ (1B*02) ⊕ (ED*03) ⊕(67*01)=(63⊕27 ⊕2C ⊕67)=7F

b_7=(63*01) ⊕ (1B*01) ⊕ (ED*02) ⊕(67*03)=(63⊕1B ⊕C1 ⊕59)=E0

b_8=(63*03) ⊕ (1B*01) ⊕ (ED*01) ⊕(67*02)=(A5⊕ 1B⊕ED ⊕CE)=9D

b_9=(93*02) ⊕ (F2*03) ⊕ (29*01) ⊕(CC*01)=(ED⊕ 0D⊕29⊕CC)=5

b_{10}=(93*01) ⊕ (F2*02) ⊕ 29*03) ⊕(CC*01)=(93⊕ FF⊕6A ⊕CC)=CA

b_{11}=(93*01) ⊕ (F2*01) ⊕ 29*02) ⊕(CC*03)=(93⊕F2 ⊕58 ⊕4F)=76

b_{12}=(93*03) ⊕ (F2*01) ⊕ (29*01) ⊕(CC*02)=(AE⊕ F2⊕ 29⊕83)=F6

b_{13}=(8F*02) ⊕ (AD*03) ⊕ (9C*01) ⊕(9A*01)= (05⊕ EC⊕ 9C⊕9A)=EF

b_{14}=(8F*01) ⊕ (AD*02) ⊕ (9C*03) ⊕(9A*01)=(8F⊕ 41⊕B9 ⊕9A)=ED

b_{15}=(8F*01) ⊕ (AD*01) ⊕ (9C*02) ⊕(9A*03)=(8F⊕AD ⊕25 ⊕A5)=A2

b_{16}=(8F*03) ⊕ (AD*01) ⊕ (9C*01) ⊕(9A*02)=(8A⊕AD ⊕ 9C⊕3F)=B4

Result= $\begin{pmatrix} 90 & DA & 5 & EF \\ E7 & 7F & CA & ED \\ AC & E0 & 76 & A2 \\ 17 & 9D & F6 & B4 \end{pmatrix}$

Important Notices

Hexadecimal multiplication

- ✓ Any number * 1= The same number
- ✓ Any number * 0= 0
- ✓ Any number ⊕ with itself =0
- ✓ Any number ⊕ 0= The same number

Ex1:

(44*03)

= (44*02) ⊕(44*01)

(0100 0100 *2) ⊕ 0100 0100

Check on the most significant bit if 0 or 1

- In case =0 then make left shift by one bit

1000 1000 ⊕ 0100 0100 =1100 1100 = CC

Ex2:

(87*3)

= (87*2) ⊕ (87*01)

(1000 0111*2) ⊕ 1000 0111

In case =1 then remove the most significant bit and add 0 as the least significant bit. After that make XOR with 1B

(0000 1110 ⊕ 0001 1101) ⊕ 1000 0111

(0001 0011) ⊕ 1000 0111=1001 0100=94

Important Notices

Hexadecimal multiplication

FC*01=

1111 1100*1= FC

44*2=

01000100 *2=1000 1000 (88)

73*2

01110011*2=1110 0110 (E6)

6A*2=

01101010*2=11010100(D4)

Important Notices

Hexadecimal multiplication

F3*2

11110011*2=11100110 xor 1B

1110 0110

0001 1011

1111 1101 (FD)

A4*2= 1010 0100*2

01001000 xor 0001 1011=

0001 1011 =0101 0011 (53)

Important Notices

AB*3

(AB*2) xor (AB*1)

(1010 1011 *2) xor (1010 1011)

((01010110 xor 00011011)) xor (1010 1011)

(01001101) xor (1010 1011)

1110 0110 (E6)

Example #2

Step 4:

Mix column

New Shifted State Matrix= $\begin{pmatrix} 0F & 00 & 10 & 39 \\ 00 & 44 & 53 & 0A \\ 22 & 04 & 4C & 27 \\ 73 & 18 & 1C & 37 \end{pmatrix} * \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

*

b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15
b4	b8	b12	b16

```
b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)
b2 = (b1 * 1) XOR (b2*2) XOR (b3*3) XOR (b4*1)
b3 = (b1 * 1) XOR (b2*1) XOR (b3*2) XOR (b4*3)
b4 = (b1 * 3) XOR (b2*1) XOR (b3*1) XOR (b4*2)
```

(b1= specifies the first byte of the state)

```
b5 = (b5 * 2) XOR (b6*3) XOR (b7*1) XOR (b8*1)
b6 = (b5 * 1) XOR (b6*2) XOR (b7*3) XOR (b8*1)
b7 = (b5 * 1) XOR (b6*1) XOR (b7*2) XOR (b8*3)
b8 = (b5 * 3) XOR (b6*1) XOR (b7*1) XOR (b8*2)
```

And so on until all columns of the state are exhausted.

Example #2

Mix column

New Shifted State Matrix= $\begin{pmatrix} 0F & 00 & 10 & 39 \\ 00 & 44 & 53 & 0A \\ 22 & 04 & 4C & 27 \\ 73 & 18 & 1C & 37 \end{pmatrix} * \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$

b_1=(0F*02) ⊕ (00*03) ⊕ (22*01) ⊕(73*01)=(1E ⊕00 ⊕22 ⊕73)=4F
b_2=(0F*01) ⊕ (00*02) ⊕ (22*03) ⊕(73*01)=(0F ⊕00 ⊕ 66⊕73)=1A
b_3=(0F*01) ⊕ (00*01) ⊕ (22*02) ⊕(73*03)=(0F⊕ 00⊕44 ⊕95)=DE
b_4=(0F*03) ⊕ (00*01) ⊕ (22*01) ⊕(73*02)=(11⊕00 ⊕22 ⊕E6)=D5
b_5=(00*02) ⊕ (44*03) ⊕ (04*01) ⊕(18*01)=(00⊕ CC⊕ 04⊕18)=D0
b_6=(00*01) ⊕ (44*02) ⊕ (04*03) ⊕(18*01)=(00⊕88 ⊕0C ⊕18)=9C
b_7=(00*01) ⊕ (44*01) ⊕ (04*02) ⊕(18*03)=(00⊕44 ⊕08 ⊕28)=64
b_8=(00*03) ⊕ (44*01) ⊕ (04*01) ⊕(18*02)=(00⊕ 44⊕04 ⊕30)=70
b_9=(10*02) ⊕ (53*03) ⊕ (4C*01) ⊕(1C*01)=(20⊕ F1⊕4C ⊕1C)=81
b_{10}=(10*01) ⊕ (53*02) ⊕ (4C*03) ⊕(1C*01)=(10⊕ A6⊕D4 ⊕1C)=7E
b_{11}=(10*01) ⊕ (53*01) ⊕ (4C*02) ⊕(1C*03)=(10⊕53 ⊕98 ⊕24)=FF
b_{12}=(10*03) ⊕ (53*01) ⊕ (4C*01) ⊕(1C*02)=(30⊕ 53⊕ 4C⊕38)=17
b_{13}=(39*02) ⊕ (0A*03) ⊕ (27*01) ⊕(37*01)= (72⊕ 1E⊕ 27⊕37)=7C
b_{14}=(39*01) ⊕ (0A*02) ⊕ (27*03) ⊕(37*01)=(39⊕ 14⊕69 ⊕37)=73
b_{15}=(39*01) ⊕ (0A*01) ⊕ (27*02) ⊕(37*03)=(39⊕0A ⊕4E ⊕59)=24
b_{16}=(39*03) ⊕ (0A*01) ⊕ (27*01) ⊕(37*02)=(4B⊕0A ⊕ 27⊕6E)=8

Result= $\begin{pmatrix} 4F & D0 & 81 & 7C \\ 1A & 9C & 7E & 73 \\ DE & 64 & FF & 24 \\ D5 & 70 & 17 & 8 \end{pmatrix}$

Second Stage for AES

Key Expansion

Key Expansion

K_1= Thats my Kung Fu
K_1 in Hex = 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75 (128bit)
w_0= 54 68 61 74
w_1= 73 20 6D 79
w_2 = 20 4B 75 6E
w_3= 67 20 46 75

K[n] : W[i] = K[n-1]W[i] XOR k[n]:W[i-1]
K[n]: W[0] = K[n-1] : W0 XOR SubByte (K[n-1] :W3 >> 8 or word) XOR Rcon[i]

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

Steps:Round 1

1. circular byte left shift of w[3]: (20; 46; 75; 67)
2. Byte Substitution (S-Box): (B7; 5A; 9D; 85)
3. Adding round constant (01; 00; 00; 00) gives: g(w[3]) = (B6; 5A; 9D; 85)
w[4] = w[0] ⊕ g(w[3]) = (E2; 32; FC; F1):

w[5] = w[4] ⊕ w[1] = (91; 12; 91; 88),
w[6] = w[5] ⊕ w[2] = (B1; 59;E4;E6),
w[7] = w[6] ⊕ w[3] = (D6; 79;A2; 93)
first roundkey: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

AES S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	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

For example HEX 19 would get replaced with HEX D4

Key Expansion

K_2 **in Hex =** E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93 **(128bit)**

w_4= E2 32 FC F1

w_5= 91 12 91 88

w_6 = B1 59 E4 E6

w_7= D6 79 A2 93

Steps:Round 2

1. circular byte left shift of w[7]: (79 A2 93 D6)

2. Byte Substitution (S-Box): (B6 3A DC F6)

3. Adding round constant (02; 00; 00; 00) gives: g(w[7]) = (B4; 5A; 9D; 85)

w[8] = w[4] ⊕ g(w[7]) = (56 68 61 74):

w[9] = w[8] ⊕ w[5] = (C7 7A F0 FC),

w[10] = w[9] ⊕ w[6] = (76 23 14 1A),

w[11] = w[10] ⊕ w[7] = (A0 5A B6 89)

second roundkey: 56 68 61 74 C7 7A F0 FC 76 23 14 1A A0 5A B6 89

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

Key Expansion

K_2 in Hex = 56 68 61 74 C7 7A F0 FC 76 23 14 1A A0 5A B6 89 (128bit)
w_8= 56 68 61 74
w_9= C7 7A F0 FC
w_{10} = 76 23 14 1A
w_{11}= A0 5A B6 89

Steps:Round 3
circular byte left shift of w[11]:
1. **2. Byte Substitution (S-Box):** ()
3. Adding round constant (04; 00; 00; 00) gives: g(w[11]) = (B4; 5A; 9D; 85)

w[12] = w[8] ⊕ g(w[11]) = ():

w[13] = w[12] ⊕ w[9] = (),
w[14] = w[13] ⊕ w[10] = (),
w[15] = w[14] ⊕ w[11] = ()

Third roundkey:

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

Key Expansion

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 ⊕ Rcon (1) = d3 85 46 79 = z1
w4 = w0 ⊕ z1 = dc 90 37 b0 w5 = w4 ⊕ w1 = 9b 49 df e9 w6 = w5 ⊕ w2 = 97 fe 72 3f w7 = w6 ⊕ w3 = 38 81 15 a7	RotWord (w7) = 81 15 a7 38 = x2 SubWord (x2) = 0c 59 5c 07 = y2 Rcon (2) = 02 00 00 00 y2 ⊕ Rcon (2) = 0e 59 5c 07 = z2
w8 = w4 ⊕ z2 = d2 c9 6b b7 w9 = w8 ⊕ w5 = 49 80 b4 5e w10 = w9 ⊕ w6 = de 7e c6 61 w11 = w10 ⊕ w7 = e6 ff d3 c6	RotWord (w11) = ff d3 c6 e6 = x3 SubWord (x3) = 16 66 b4 83 = y3 Rcon (3) = 04 00 00 00 y3 ⊕ Rcon (3) = 12 66 b4 8e = z3
w12 = w8 ⊕ z3 = c0 af df 39 w13 = w12 ⊕ w9 = 89 2f 6b 67 w14 = w13 ⊕ w10 = 57 51 ad 06 w15 = w14 ⊕ w11 = b1 ae 7e c0	RotWord (w15) = ae 7e c0 b1 = x4 SubWord (x4) = e4 f3 ba c8 = y4 Rcon (4) = 08 00 00 00 y4 ⊕ Rcon (4) = ec f3 ba c8 = 4

Key Words	Auxiliary Function
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 (x5) = 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 (x6) = 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 (x7) = 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 (x8) = 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 (x9) = 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 (x10) = 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	

Key Expansion

Key : Go to the GYM $?

K1= 47 6F 20 74 6F 20 74 68 65 20 47 59 4D 20 24 3F

K[n]: w_0= K[n-1]:w_0 ⊕ SubByte(K[n-1]:w_3>>8) ⊕ Rcon[i]

K[2] :w0= 47 6f 20 74 ⊕ subbyte (20 24 3f 4D) ⊕ Rcon[0]

= 47 6f 20 74 ⊕ B7 36 75 E3 ⊕ Rcon[0]

= 47 6f 20 74 ⊕ B7 36 75 E3 ⊕ 01 00 00 00

= 71 38 55 97 ⊕ 01 00 00 00

K[2]:w0 = 70 38 55 97

Rcon(0) = 01000000
Rcon(1) = 02000000
Rcon(2) = 04000000
Rcon(3) = 08000000
Rcon(4) = 10000000
Rcon(5) = 20000000
Rcon(6) = 40000000
Rcon(7) = 80000000
Rcon(8) = 1B000000
Rcon(9) = 36000000
Rcon(10) = 6C000000
Rcon(11) = D8000000
Rcon(12) = AB000000
Rcon(13) = 4D000000
Rcon(14) = 9A000000

Simplified AES Example

Let know that the first sub key is 6F 32 Determine the second sub key

w_0= 6F

w_1=32

$w_2 = w_0 \oplus 80 \oplus SubNib(RotNib(w_1))$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus SubNib(RotNib(0011\ 0010))$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus SubNib(0010\ 0011)$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus SubNib(0010\ 0011)$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus (1010\ 1011)$

$w_2 = 0100\ 0100$

$w_3 = w_2 \oplus w_1$

$w_3 = 0100\ 0100 \oplus 0011\ 0010$

w_3 =0111 0110

nibble	S-box(nibble)	nibble	S-box(nibble)
0000	1001	1000	0110
0001	0100	1001	0010
0010	1010	1010	0000
0011	1011	1011	0011
0100	1101	1100	1100
0101	0001	1101	1110
0110	1000	1110	1111
0111	0101	1111	0111

RC4 Basics

A symmetric key encryption algorithm invented by Ron Rivest

- A proprietary cipher owned by RSA, kept secret
- Code released anonymously in Cyberpunks mailing list in 1994
- Later posted sci.crypt newsgroup

Variable key size, **byte-oriented** stream cipher

- Normally uses 64 bit and 128 bit key sizes.

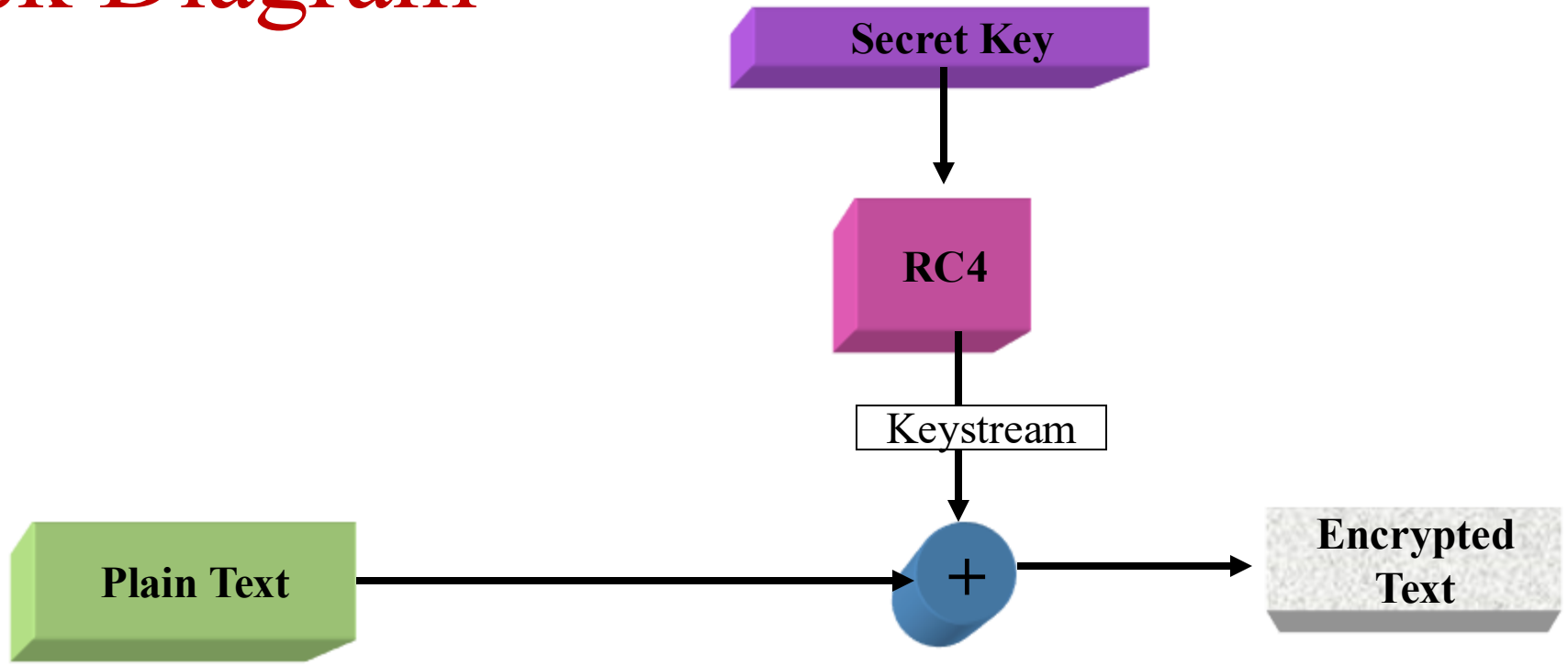
Used in

- SSL/TLS (Secure socket, transport layer security) between web browsers and servers,
- IEEE 802.11 wirelss LAN std: WEP (Wired Equivalent Privacy), WPA (WiFi Protocol Access) protocol

RC4-based Usage

- WEP
- WPA default
- Bit Torrent Protocol Encryption
- Microsoft Point-to-Point Encryption
- SSL (optionally)
- SSH (optionally)
- Remote Desktop Protocol
- Kerberos (optionally)

RC4 Block Diagram



Cryptographically very strong and easy to implement

RC4 …Inside

- Consists of 2 parts:
  - Key Scheduling Algorithm (KSA)
  - Pseudo-Random Generation Algorithm (PRGA)

- KSA
  - Generate State array
- PRGA on the KSA
  - Generate keystream
  - XOR keystream with the data to generated encrypted stream

RC4 …Inside

- Use the secret key to initialize and **permutation** of state vector **S**, done in two steps
- Use 8-bit index pointers **i** and **j**

1

```
for i = 0 to 255 do
 S[i] = i;
 T[i] = K[i mod(|K|)];
```

[S], S is set equal to the values from 0 to 255
S[0]=0, S[1]=1,…, S[255]=255
[T], A temporary vector
[K], Array of bytes of secret key
|K| = Keylen, Length of (K)

2

```
j = 0;
for i = 0 to 255 do
  j = (j+S[i]+T[i])(mod 256)
swap (S[i], S[j])
```

- Use T to produce initial permutation of S
- The only operation on S is a swap; S still contains number from 0 to 255

After KSA, the input key and the temporary vector T will be no longer used

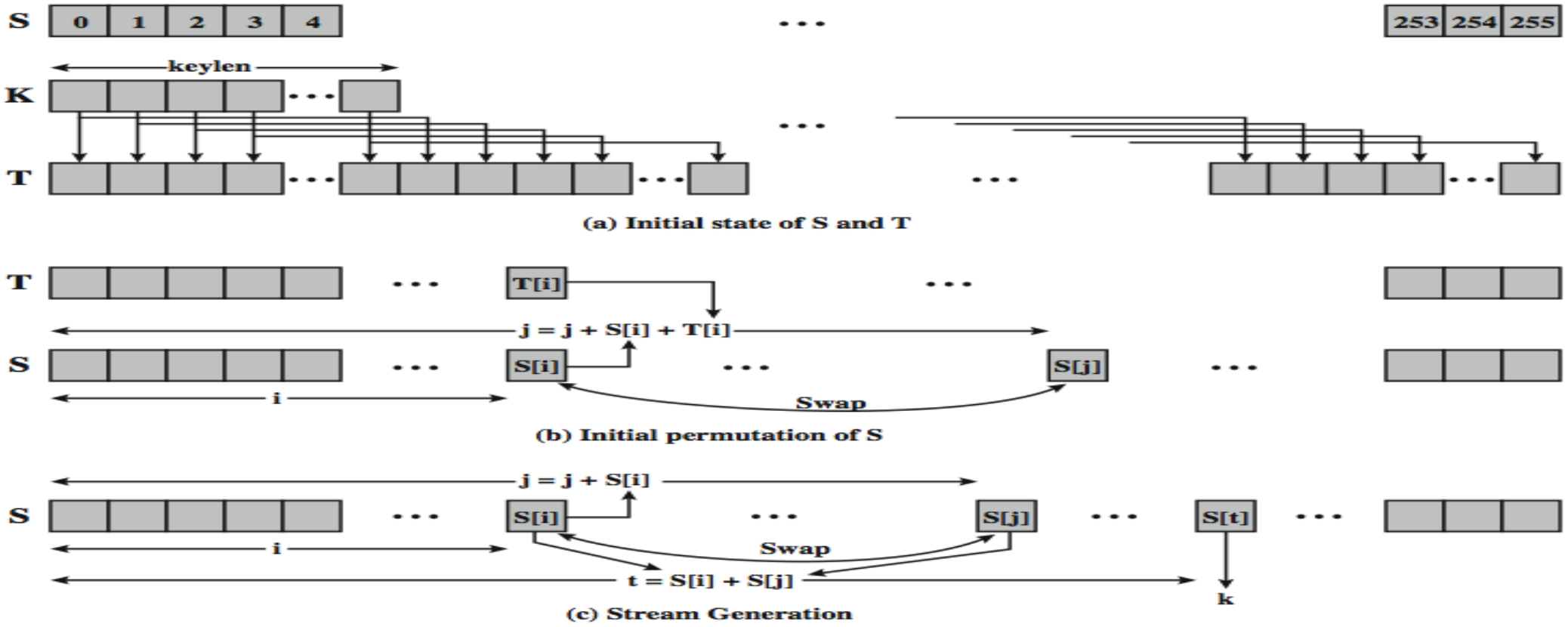
The PRGA

- Generate key stream *k* , one by one
- XOR S[k] with next byte of message to encrypt/decrypt

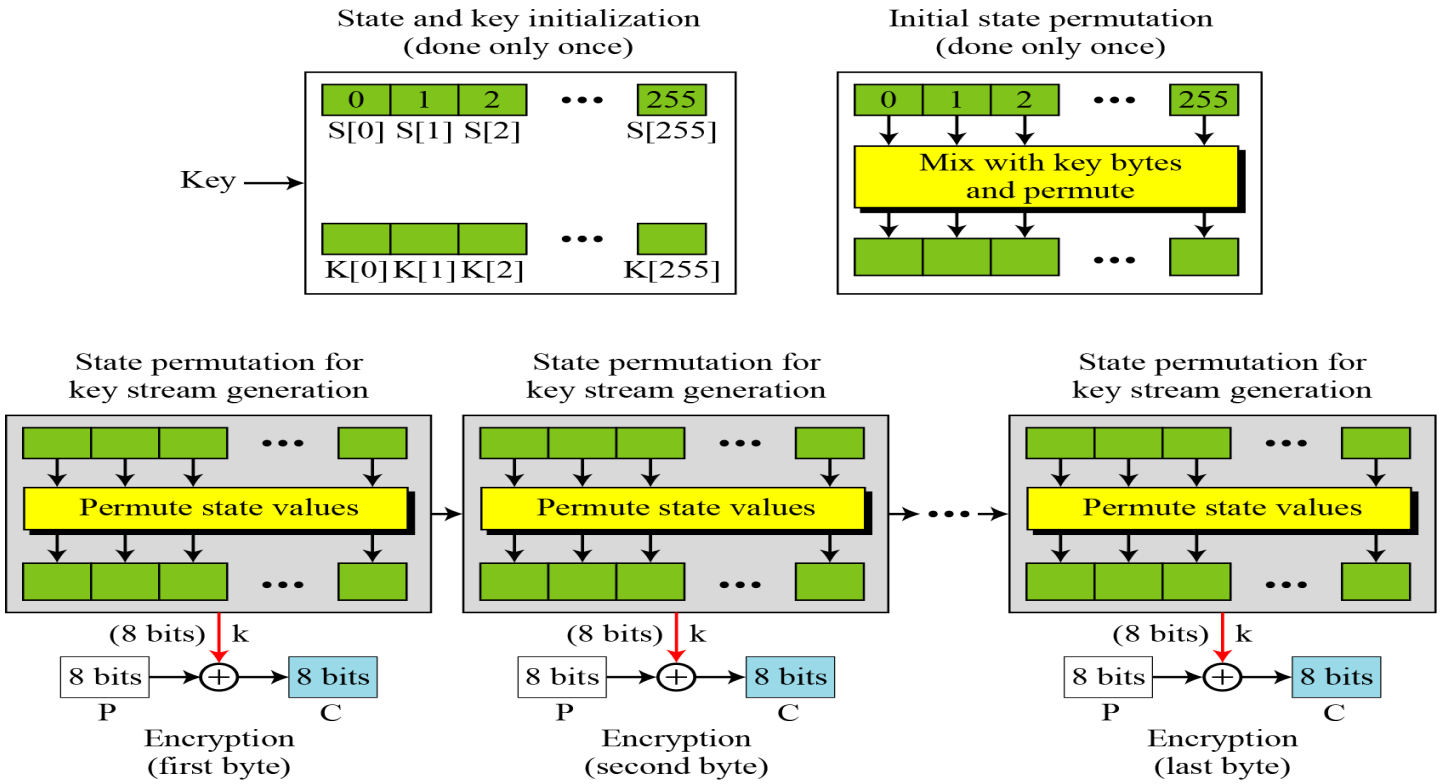
```
i = j = 0;
While (more_byte_to_encrypt)
    i = (i + 1)  (mod 256);
    j = (j + S[i])  (mod 256);
    swap(S[i], S[j]);
    k = (S[i] + S[j])  (mod 256);
    C_i = M_i XOR S[k];
```

Sum of shuffled pair selects "stream key" value from permutation

Detailed Diagram



Overall Operation of RC4



Example based on Simplified RC4

Assume we use a 4 * 3 bits key of [1236]. And the plain text
Which we need to convert to cipher text using RC4 Algorithm
is P =[1222]
The Algorithm consists of two stages Initialization stage to generate keys and encryption stage to convert the plain text to cipher text.
Let S = [01234567]
T = [12361236], you must note that the key must be repeated to be same as the long of S
Now perform different permutation based on the following code

```
J =0;
For(I =0; 0< S.length; i++){
J = (j+S[i] + T[i])mod8
Swap S[j] with S[i].
Generate new S }
```

The generated S will be used in all iterations based on the size of S

Example based on Simplified RC4

Second step is depending on the new state which generated from the previous set of steps
Which equal to s =[2 3 7 4 6 0 1 5]
The second step will generate 3 bit at a time to be Xor with plain text based on the lower pseudocode , the lower iterations depends on the plain text number

```
I,j =0,
While(True){
i=(i+1)mod 8;// the value of I will be incremented each iteration by one
J=(j + S[i] mod 8);// the value of j depends on the output of this operation
Swap(S[i], S[j]);// swap to generate a new State S
T = (S[i] + S[j])mod 8;// the value of T depends on the new State S[i] for new state
K=S[t];
```

Decryption Process

Use the same secret key as during the encryption phase.
Generate keystream by running the KSA and PRGA.
XOR keystream with the encrypted text to generate the plain text.
Logic is simple :

$$(A \text{ xor } B) \text{ xor } B = A$$

A = Plain Text or Data
B = KeyStream

RC4 and WEP

WEP is a protocol using RC4 to encrypt packets for transmission over IEEE 802.11 wireless LAN.

WEP requires each packet to be encrypted with a separate RC4 key.

The RC4 key for each packet is a concatenation of a 24-bit IV (initialization vector) and a 40 or 104-bit long-term key.

RC4 key: IV (24) Long-term key (40 or 104 bits)

Reference

1. Lecture slides prepared for “Cryptography and Network Security”, 7/e, by William Stallings. Chapter 1, “Computer and Network Security Concepts”.