



# **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**

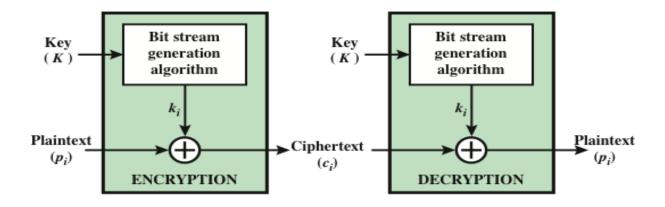
A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length

Typically, a block size of 64 or 128 bits is used

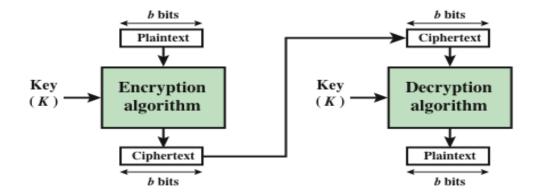
As with a stream cipher, the two users share a symmetric encryption key

The majority of network-based symmetric cryptographic applications make use of block ciphers





(a) Stream Cipher Using Algorithmic Bit Stream Generator

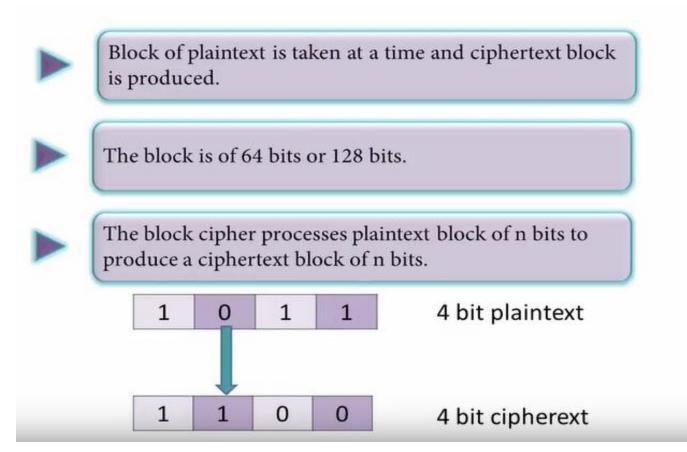


(b) Block Cipher

Figure 4.1 Stream Cipher and Block Cipher



# **Block Cipher**





## 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  $\rightarrow$  P= C Xor K
- (1000 1001 1101 1110)

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

- Ciphertext  $\rightarrow$  C=P Xor K
- (0110 1010) 6A
- Plaintext  $\rightarrow$  P= C Xor K
- (1100 1001)

Α	В	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

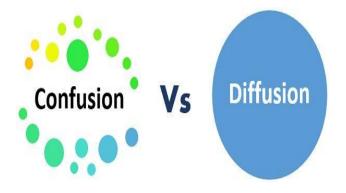
# Permutation

 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.

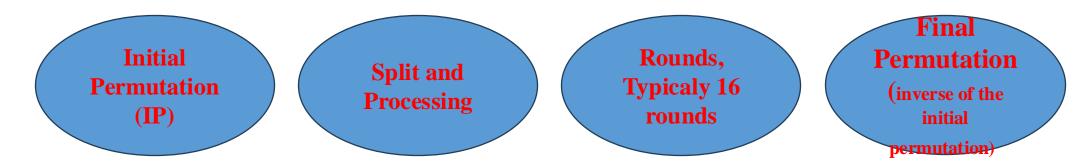


# Feistel Cipher

• 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.





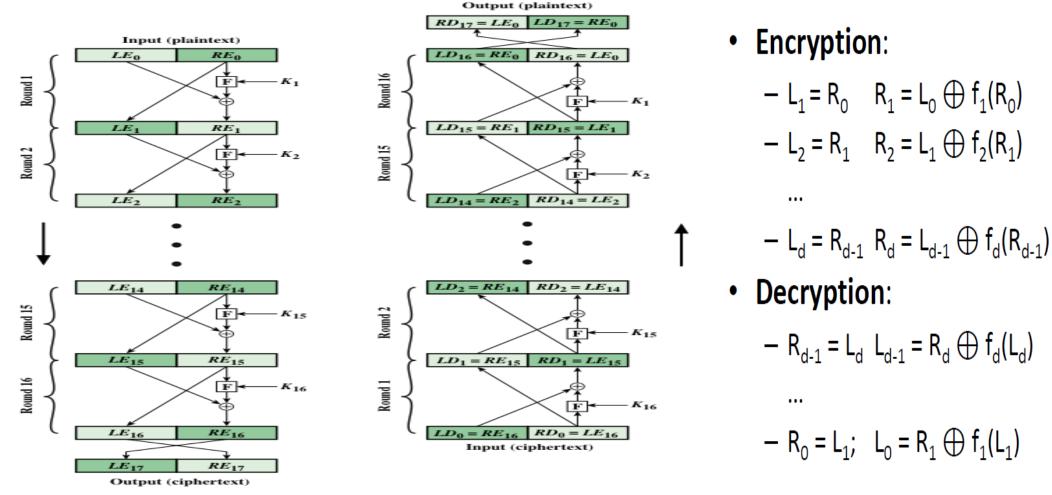


Figure 4.3 Feistel Encryption and Decryption (16 rounds)



### **Simple Example:**

**Plaintext:** 110011

Initial Permutation:  $110011 \rightarrow 011100$ 

### **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

Decryption round



# **Feistel Example**

**Encryption round** 

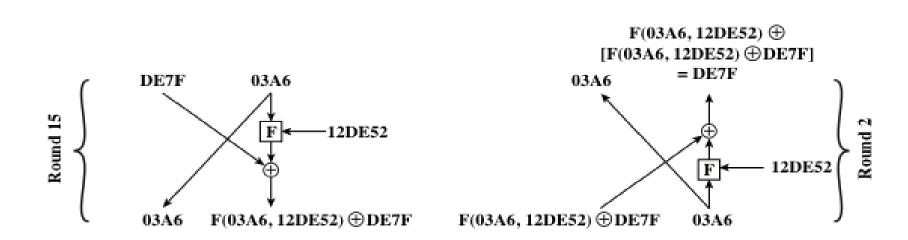


Figure 4.4 Feistel Example



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

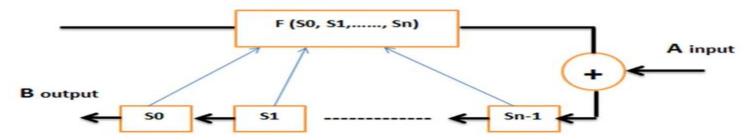
You must note that not all LFSR can reach to the maximum number of rounds, since the number of bit generation will stop when this number is redundant with initial Seed value.

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

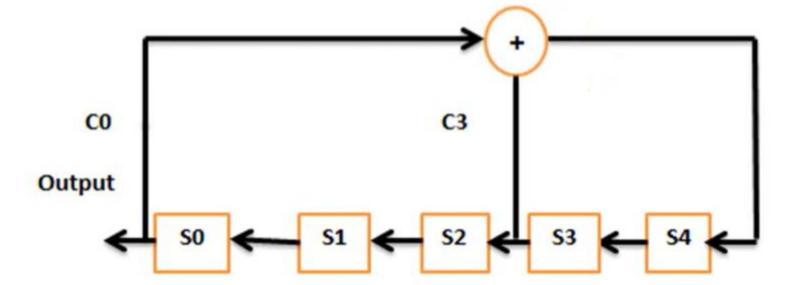
The LFSR algorithm used a registers to be shifted Si- $\rightarrow$  Si-1 $\rightarrow$ ,...., $\rightarrow$ S1 $\rightarrow$ S0





Shift Register = 5, initial key= 10101 Used LFSR with length n=5

Plain text =HELP





Plain text :HELP

H=48H=01001000

Key = 10101110-----sequence

Plain= 01001000

Cipher= 11100110

Stop: stage repeats

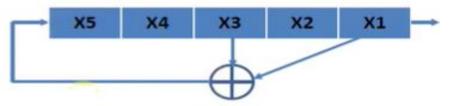


#### LFSR Example:

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<sup>5</sup>-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.

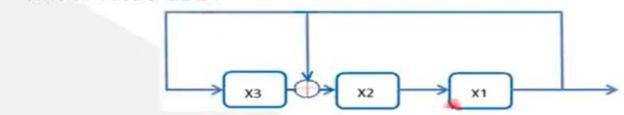
00001	10011	01100	00010
10000	11001	10110	
01000	11100	11011	00001
00100	11110	11101	
10010	11111	01110	
01001	01111	10111	
10100	00111	01011	
11010	00011	10101	
01101	10001	01010	
00110	11000	00101	





Ex: LFSR of 3 flip flops and the polynomial X2 = X1 

X3 and the initial value 111.



хз	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** 



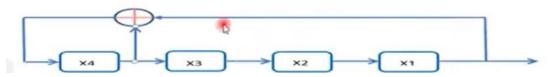
Example: Encrypt the plain text TOBE using Stream cipher and LFSR of 4-flip flops and feedback Function X4= X1 XOR X4

The Plain Text TOBE is Equal using Ascii table as follow

Letter	Ascii	Binary
Т	84	1010100
О	79	1001111
В	66	1000010
E	69	1000101

Plain-Text: 1010100100111110000101000101





X4	Х3	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: 10101001001111110000101000101

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)**

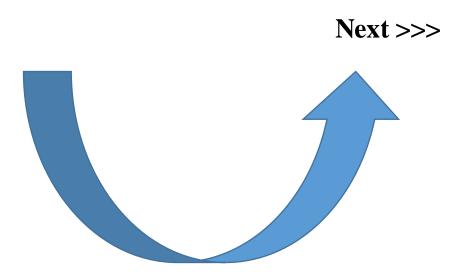
- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
  - 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

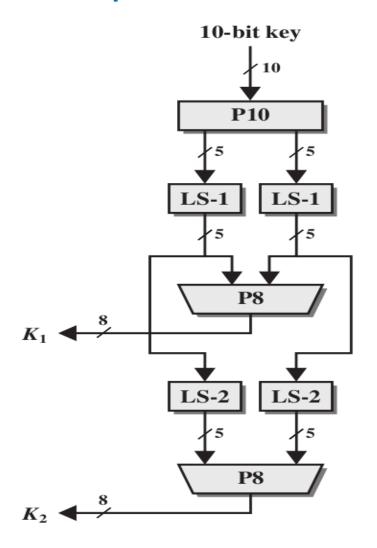


## **Simplified Data Encryption Standard (SDES)**





## **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)

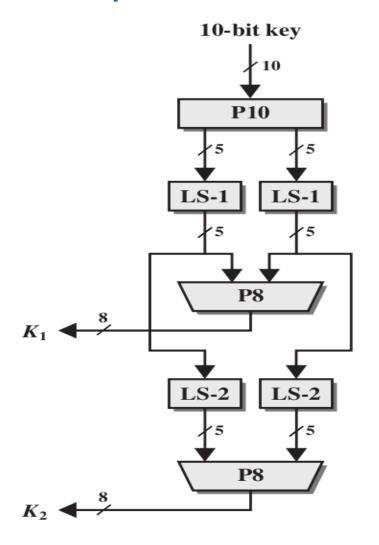
P8=637485109

10-03/403103	
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=637485109

К	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**

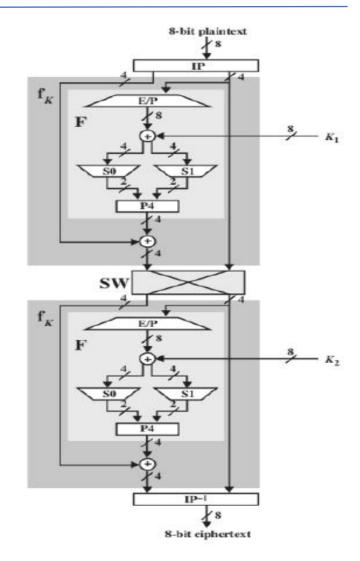
Phase #2. Message encryption

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( $R_0$ ) $\bigoplus K_1$ )	11 01
6. $P_4$ (Sboxs(E/P( $R_0$ ) $\bigoplus K_1$ ))	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  $f_{k2}(L,R) = f_{\{0111\ 1010\}}(0110\ 0100) = (0110\ \oplus\ f\ (R,k2),R)$ 

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		
$E/P(R_1) \oplus K_2$	0101 0010		
Sboxs(E/P( $R_1$ ) $\bigoplus K_2$ )	01 01		
P4(Sboxs(E/P( $R_1$ ) $\bigoplus K_2$ ))	1100		
$(L_1 \oplus P4(Sboxs(E/P(R_1) \oplus K_2))$	(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

			P	8			
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

 $K_1 = 1110\ 1001$ 

 $K_2 = 10100111$ 





### Example #2 cont..

### 2. Message encryption

M= 0010 1000 (8-bit)

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

E/P

S0 =

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( $R_0$ ) $\bigoplus K_1$ )	10 00
6. $P_4(\operatorname{Sboxs}(E/P(R_0) \oplus K_1))$	0001 (result from F)
$L_0 \oplus P_4$ (Sboxs(E/P( $R_0$ ) $\oplus K_1$ ))	0011

$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 2 & 0 & 1 & 3 \\ \end{bmatrix}$	0010 0010
$\begin{bmatrix} 2 & [0 & 2 & 1 & 3] \\ 3 & [3 & 1 & 3 & 2] \end{bmatrix} \qquad \begin{bmatrix} 2 \\ 3 \end{bmatrix}$	E: Divine IP(M) into Land R	$L_0 = 0010$ $R_0 = 0010$
IP = 2 6 3 1 4	0001 0100	
	$K_1$	1110 1001
	<b>4</b> . <b>E/</b> P( <i>R</i> <sub>0</sub> ) <b>⊕</b> <i>K</i> <sub>1</sub>	1111 1101
	5. Shexs(E/P( $R_0$ ) $\bigoplus$ $K_1$ )	10 00
	6. $P_4$ (\$boxs(E/P( $R_0$ ) $\bigoplus K_1$ ))	0001 (result from F)
2 4	$(E/P(R_0) \oplus K$	(1) 0011

 $L_1 = 0010$   $R_1 = 0011$ 



### Example #2 cont... find the mistakes here

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

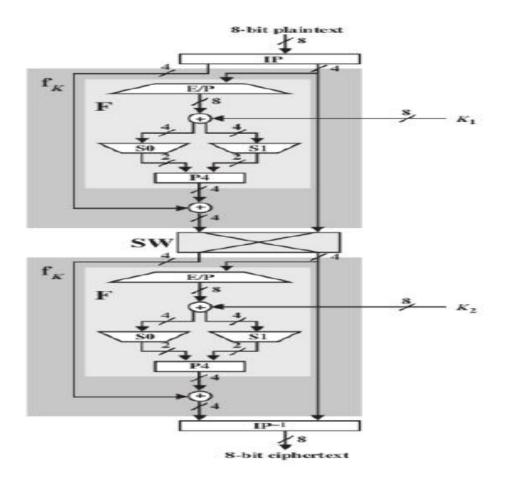
SW	0010 0011
Divide SW into L and R	L1=0010 R1=0011
$E/P(R_1)$	1001 0110
$K_2$	1010 0111
$E/P(R_1) \oplus K_2$	0011 0001
Sboxs(E/P( $R_1$ ) $\bigoplus K_2$ )	10 10
$P4(Sboxs(E/P(R_1) \oplus K_2))$	0011
$(L_1 \oplus P4(Sboxs(E/P(R_1) \oplus K_2))$	(0010(XOR)0011)=0001
$f_{k2}(L,R)$	0001 0011
$IP^{-1}$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

$$L_1 = 0010$$
  $R_1 = 0011$ 



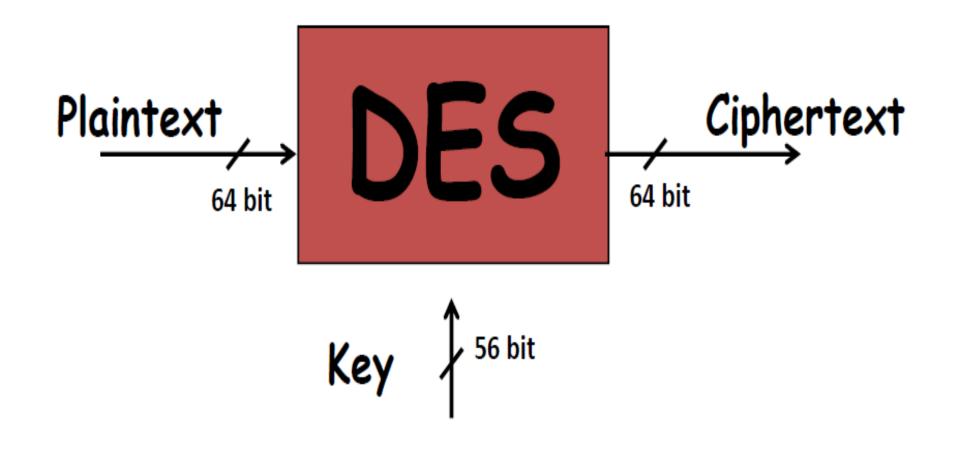
## **Training for Encryption and Decryption Using S-DES**

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



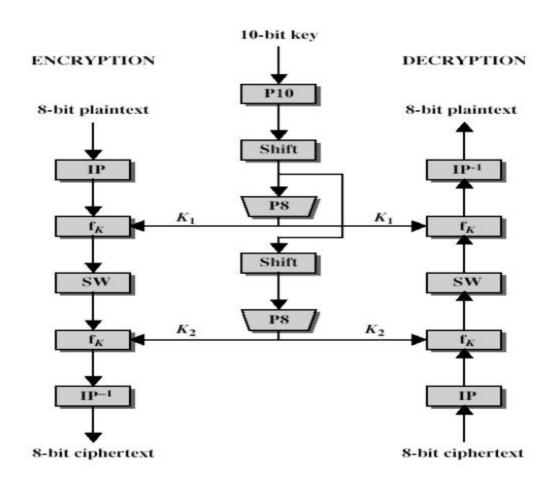


# **Data Encryption Standard (DES)**





# **SDES- Key Creation process**



Example:

Key: 1010000010, Find the Keys K1 and K2

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

			P	8			
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 = 1010101011001100111100011110
K[2].L=1100001100110010101010111111	K[2].R = 0101010110011001111000111101
K[3].L = 00001100110010101010111111111	K[3].R = 0101011001100111100011110101
K[4].L = 00110011001010101011111111100	K[4].R = 0101100110011110001111010101
K[5].L = 110011001010101011111111110000	K[5].R = 0110011001111000111101010101
K[6].L = 001100101010101111111111000011	K[6].R = 1001100111100011110101010101



# **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		

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

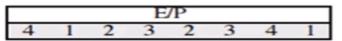
R1 = 01111101 this after expansion

After that do XOR between K1 XOR R1



10100010

 $1\ 1\ 0\ 1\ 1\ 1\ 1\ 1 = F(R,K1)$ 



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

11 = 3

$$01 = 2$$

$$S0 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix}$$

$$S1 = \begin{matrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 2 & 3 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 3 \end{matrix}$$

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



# **DES Structure Example 1**

After that do XOR between K1 XOR R1

01111101

10100010

 $1\ 1\ 0\ 1\ 1\ 1\ 1\ 1 = F(R,K1)$ 

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 = 26314857).

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

43

01011001 this is the massage after encryption



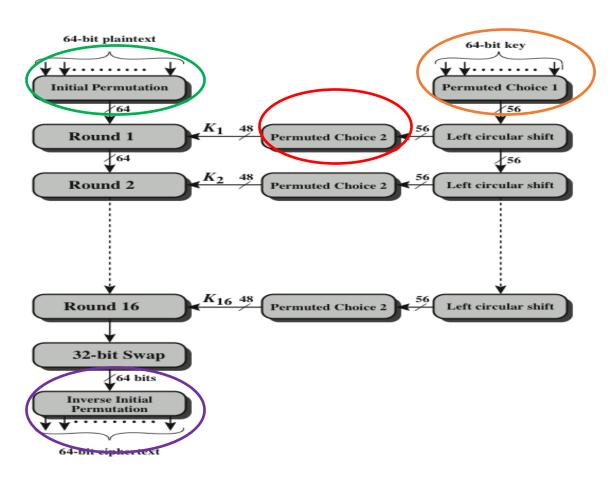


Figure 4.5 General Depiction of DES Encryption Algorithm

#### 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 <sup>-1</sup>					
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
- $K_p = 1111000\ 0110011\ 0010101\ 0101111\ 0101010\ 1011001\ 1001111\ 0001111$  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





<b>K[1].L</b> =1110000110011001010101011111	<b>K[1].R</b> =1010101011001100111100011110
<b>K[2].L</b> =11000011001100101010111111	<b>K[2].R</b> =0101010110011001111000111101
<b>K[3].L</b> =0000110011001010101011111111	<b>K[3].R</b> =0101011001100111100011110101
<b>K[4].L</b> =00110011001010101011111111100	<b>K[4].R</b> =0101100110011110001111010101
<b>K[5].L</b> =11001100101010101111111110000	<b>K[5].R</b> =0110011001111000111101010101
<b>K[6].L</b> =00110010101010111111111000011	<b>K[6].R</b> =1001100111100011110101010101
<b>K[7].L</b> =11001010101011111111100001100	<b>K[7].R</b> =011001111000111101010101010
<b>K[8].L</b> =00101010101111111110000110011	<b>K[8].R</b> =1001111000111101010101011001
<b>K[9].L</b> =01010101011111111100001100110	<b>K[9].R</b> =0011110001111010101010110011
<b>K[10].L</b> =01010101111111110000110011001	<b>K[10].R</b> =1111000111101010101011001100
<b>K[11].L</b> =01010111111111000011001100101	<b>K[11].R</b> =1100011110101010101100110011
<b>K[12].</b> L=01011111111100001100110010101	<b>K[12].R</b> =0001111010101010110011001111
<b>K[13].L</b> =01111111110000110011001010101	<b>K[13].R</b> =0111101010101011001100111100
<b>K[14].L</b> =1111111000011001100101010101	<b>K[14].R</b> =1110101010101100110011110001
<b>K[15].L</b> =1111100001100110010101010111	<b>K[15].R</b> =1010101010110011001111000111
<b>K[16].L</b> =111100001100110010101011111	<b>K[16].R</b> =0101010101100110011110001111



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[3]**=0000110011001010101011111111 0101011001100111100011110101 **K[4]**=00110011001010101011111111100 0101100110011110001111010101 **K[7]**=110010101010111111111100001100 01100111100011110101010101010 **K[8]**=001010101011111111110000110011 100111100011110101010101010101 **K[9]**=010101010111111111100001100110 001111000111101010101011 **K[10]**=010101011111111110000110011001 1111000111101010101011001100 **K[11]**=01010111111111000011001100101 1100011110101010101100110011 **K[13]**=01111111110000110011001010101 0111101010101011001100111100 

K[1]=1110000110011001010101011111 1010101011001100111100011110

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[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 = 0123456789ABCDEF

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 (64-bit)

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

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

 $L_1 = R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$ 

 $R_1 = L_0 + f(R_0, K_1)$ 

### **Expainsion permutation**

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

 $R_0$  = 1111 0000 1010 1010 1111 0000 1010 1010 (32-bit) **E**( $R_0$ ) = 011110 100001 010101 010101 011110 100001 010101 010101 (48-bit)

 $E(R_0)$  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 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

### 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
0уууу0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0уууу1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1уууу0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1уууу1	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
0уууу0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0уууу1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1уууу0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1уууу1	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
0уууу0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0уууу1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1уууу0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1уууу1	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
0уууу0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0уууу1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1уууу0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1уууу1	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
0уууу0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0уууу1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1уууу0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1уууу1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0уууу0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0уууу1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1уууу0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1уууу1	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
0уууу0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0уууу1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1уууу0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1уууу1	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
0уууу0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0уууу1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1уууу0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1уууу1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S-box 5

S-box 8



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
Clength	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 bruteforce 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
- -0101010101010101
- – 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

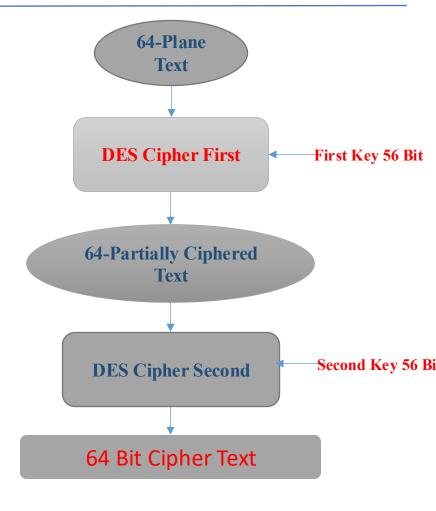
- A lot of researchers in the filed of encryption theories tries to find and enhance the original DES algorithm to be used in the process of Ciphering, some of these algorithms are used and some other is discarded
- Double DES algorithm, Triple DES algorithm refers to the famousness of these developed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





# **Double DES Algorithm**

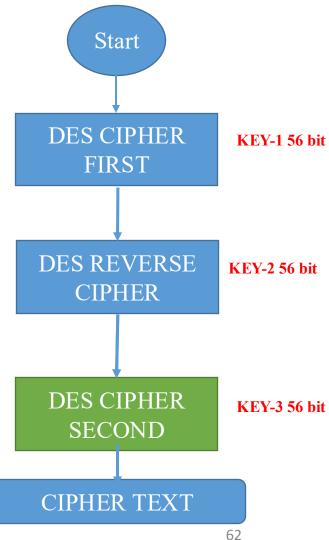
- Double DES is an encryption technique which uses two instance of DES on same plain text. In both instances it uses different keys to encrypt the plain text. Both keys are required at the time of decryption. The 64-bit plain text goes into first DES instance which than converted into a 64-bit middle 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= Ek2(Ek1(PT))
- Plain Text= Dk2(Dk2(CT))
- Meet In The Middle Attack is the problem that face the Double DES





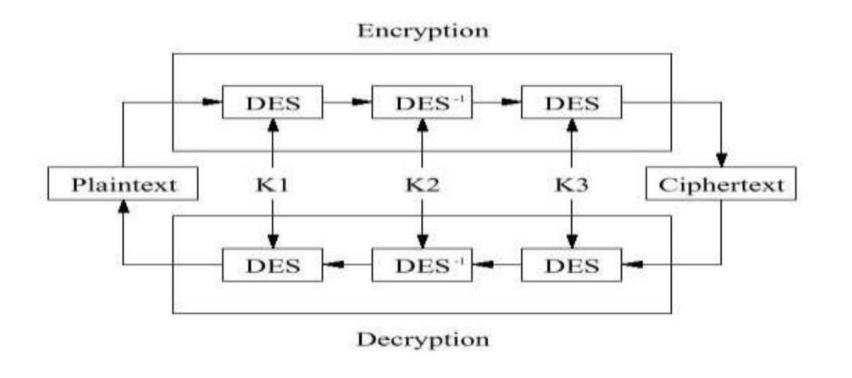
# Triple DES

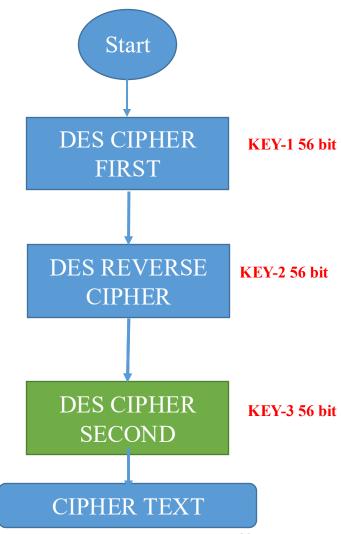
- Triple DES
  - Triple DES is a encryption technique which uses three instance of DES on same plain text. It uses their different types of key choosing technique in first all used keys are different and in second two keys are same and one is different and in third all keys are same.
  - Triple DES is 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 = Ek3(Dk2(Ek1(Text)))
  - Plain Text = (Dk1(Ek2(Dk3(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

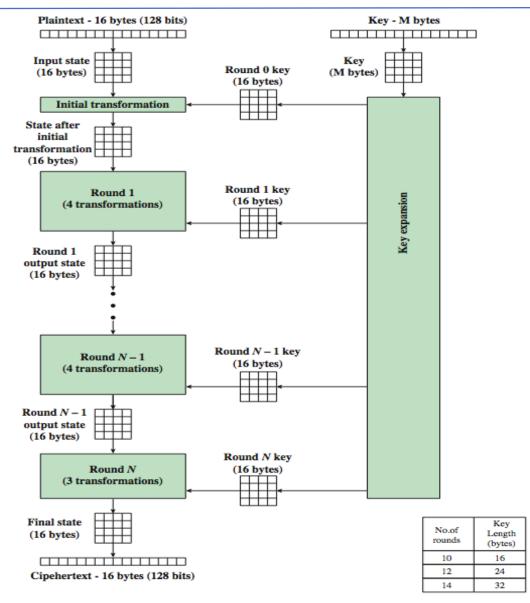


# The AES Cipher - Rijndael

- 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

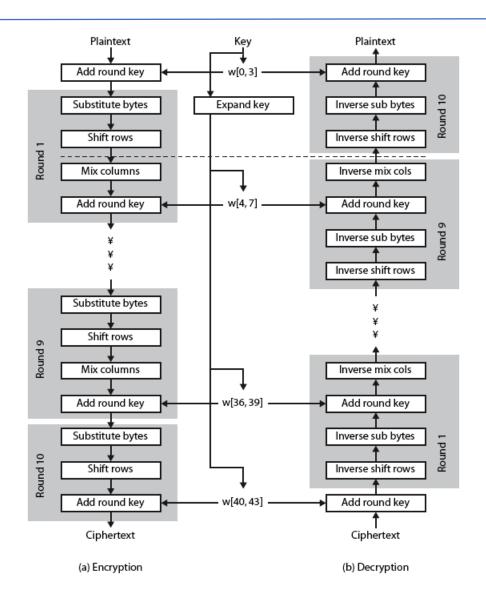


# AES Encryption Process





# **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	Q	080	P	096	ľ	112	р
033	1	049	1	065	A	081	Q	097	a	113	q
034		050	2	066	В	082	R	098	b	114	r
035	#	051	3	067	C	083	S	099	С	115	S
036	\$	052	4	068	D	084	T	100	d	116	t
037	%	053	5	069	E	085	U	101	е	117	u
038	<b>&amp;</b>	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	Ι	089	Y	105	i	121	у
042	*	058	i	074	J	090	Z	106	j	122	Z
043	+	059	j	075	K	091	[	107	k	123	{
044	C	060	<	076	L	092	١	108	1	124	
045	-	061	•	077	M	093	]	109	m	125	}
046		062	>	078	N	094	•	110	n	126	~
047	/	063	?	079	0	095	_	111	0	127	DEL

### **Hexa Decimal Value**

```
2 32
8 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
     193 194 195 196 197 198 199 200 201 202 203 204
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**

Н	е	1	1	0		S	t	u	d	е	n	t	S		L
48	65	6C	6C	6F	20	53	74	75	64	65	6E	74	73	20	4C

Row, Column
From
Hexa Decimal Value

# Key

G	0		t	0		t	h	е		G	Y	M		\$	?
47	6F	20	74	6F	20	74	68	65	20	47	59	4D	20	24	3F

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

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



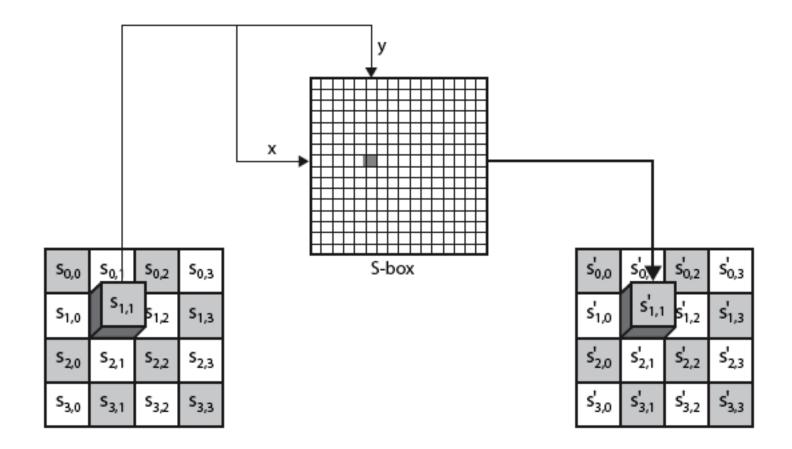
## **AES Example #1**

### Step1:

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



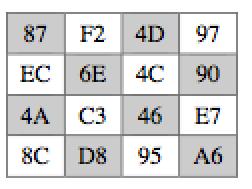
# **Substitute Bytes Step**





# Substitute Bytes Example

EA	04	65	85
83	45	5D	96
5C	33	98	В0
F0	2D	AD	C5



#### AES S-Box Lookup Table

```
30 01
                        2B
            FO AD D4 A2 AF
            CC 34 A5 E5
            9A 07 12
                     80
                        E2
        5A AO 52 3B D6 B3
            5B 6A CB
                        39
                     BE
               45 F9
               BC B6 DA
               C4 A7
               46 EE
            5C C2 D3 AC
            A9 6C 56 F4
               E8 DD
                        1F
               61 35
                        В9
               41 99
OD BF
     E6 42
```

For example HEX 19 would get replaced with HEX D4



### Step 2:

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

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

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

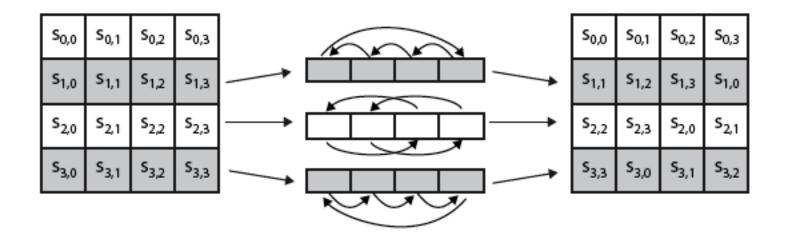
#### AES S-Box Lookup Table

```
3B D6
                          45 F9 02
                       0E
F 8C A1 89 0D BF E6 42 68 41 99 2D 0F
```

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



## 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.

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

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



## Step 4:

Mix column
New Shifted State Matrix=
$$\begin{pmatrix}
76 & 63 & CA & 12 \\
63 & 1B & ED & 67 \\
93 & F2 & 29 & CC
\end{pmatrix} * \begin{pmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03
\end{pmatrix}$$

```
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.



Step 4:
Mix column

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

 $b_1 = (76*02) \oplus (63*03) \oplus (93*01) \oplus (8F*01) = (\oplus \oplus \oplus) =$ 



Result=???



### Example1 cont... Step 4: Mix column

New Shifted State Matrix=

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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**

```
Ex1:
```

```
(44*03)
= (44*02) \oplus (44*01)
(0100\ 0100\ *2) \oplus 0100\ 0100
Check on the most significant bit if 0 or 1
```

• In case =0 then make left shift by one bit  $1000\ 1000\ \oplus\ 0100\ 0100\ =1100\ 1100\ =\ CC$ 

#### **Ex2**:

```
(87*3)
= (87*2) \oplus (87*01)
(1000\ 0111*2) \oplus 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  $(0000\ 1110\ \oplus\ 0001\ 1101\ )\ \oplus\ 1000\ 0111$ 

 $(0001\ 0011) \oplus 1000\ 0111=1001\ 0100=94$ 

- ✓ Any number \* 1= The same number
- $\checkmark$  Any number \* 0= 0
- ✓ Any number  $\oplus$  with itself =0
- ✓ Any number  $\oplus$  0= The same number





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

```
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) \text{ XOR } (b6*3) \text{ XOR } (b7*1) \text{ XOR } (b8*1)

b6 = (b5 * 1) \text{ XOR } (b6*2) \text{ XOR } (b7*3) \text{ XOR } (b8*1)

b7 = (b5 * 1) \text{ XOR } (b6*1) \text{ XOR } (b7*2) \text{ XOR } (b8*3)

b8 = (b5 * 3) \text{ XOR } (b6*1) \text{ XOR } (b7*1) \text{ 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) \oplus (00*03) \oplus (22*01) \oplus (73*01) = (1E \oplus 00 \oplus 22 \oplus 73) = 4F
b_2 = (0F*01) \oplus (00*02) \oplus (22*03) \oplus (73*01) = (0F \oplus 00 \oplus 66 \oplus 73) = 1A
b_3 = (0F*01) \oplus (00*01) \oplus (22*02) \oplus (73*03) = (0F \oplus 00 \oplus 44 \oplus 95) = DE
b_4 = (0F*03) \oplus (00*01) \oplus (22*01) \oplus (73*02) = (11 \oplus 00 \oplus 22 \oplus E6) = D5
b_5 = (00*02) \oplus (44*03) \oplus (04*01) \oplus (18*01) = (00 \oplus CC \oplus 04 \oplus 18) = D0
b_6 = (00*01) \oplus (44*02) \oplus (04*03) \oplus (18*01) = (00 \oplus 88 \oplus 00 \oplus 18) = 90
b_7 = (00*01) \oplus (44*01) \oplus (04*02) \oplus (18*03) = (00 \oplus 44 \oplus 08 \oplus 28) = 64
b_8 = (00*03) \oplus (44*01) \oplus (04*01) \oplus (18*02) = (00 \oplus 44 \oplus 04 \oplus 30) = 70
b_9 = (10*02) \oplus (53*03) \oplus (4C*01) \oplus (1C*01) = (20 \oplus F1 \oplus 4C \oplus 1C) = 81
b_{10}=(10*01) \oplus (53*02) \oplus (4C*03) \oplus(1C*01)=(10\oplus A6\oplusD4 \oplus1C)=7E
b_{11} = (10*01) \oplus (53*01) \oplus (4C*02) \oplus (1C*03) = (10 \oplus 53 \oplus 98 \oplus 24) = FF
b_{12}=(10*03) \oplus (53*01) \oplus (4C*01) \oplus(1C*02)=(30\oplus 53\oplus 4C\oplus38)=17
b_{13}=(39*02) \oplus (0A*03) \oplus (27*01) \oplus(37*01)= (72\oplus 1E\oplus 27\oplus37)=7C
b_{14}=(39*01) \oplus (0A*02) \oplus (27*03) \oplus(37*01)=(39\oplus 14\oplus69 \oplus37)=73
b_{15}=(39*01) \oplus (0A*01) \oplus (27*02) \oplus(37*03)=(39\oplus0A \oplus4E \oplus59)=24
b_{16}=(39*03) \oplus (0A*01) \oplus (27*01) \oplus(37*02)=(4B\oplus0A \oplus 27\oplus6E)=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** 





```
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_1= 73 20 6D 79 w_1= 73 20 6D 79 w_1= 8 w_2= 20 4B 75 6E w_3= 67 20 46 75 w_3= 70 20 46 75 w_3
```

### **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) gives: g(w[3]) = (B6; 5A; 9D; 85) $w[4] = w[0] \oplus g(w[3]) = (E2; 32; FC; F1):$

```
w[5] = w[4] \bigoplus w[1] = (91; 12; 91; 88),

w[6] = w[5] \bigoplus w[2] = (B1; 59; E4; E6),

w[7] = w[6] \bigoplus w[3] = (D6; 79; A2; 93)

first roundkey: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
```

Round	Constant (RCon)	Round	Constant (RCon)
1	( <u><b>01</b></u> 00 00 00) <sub>16</sub>	6	( <u><b>20</b></u> 00 00 00) <sub>16</sub>
2	( <u>02</u> 00 00 00) <sub>16</sub>	7	( <u>40</u> 00 00 00) <sub>16</sub>
3	( <u>04</u> 00 00 00) <sub>16</sub>	8	( <u>80</u> 00 00 00) <sub>16</sub>
4	( <u>08</u> 00 00 00) <sub>16</sub>	9	( <u>1B</u> 00 00 00) <sub>16</sub>
5	( <u>10</u> 00 00 00) <sub>16</sub>	10	( <u>36</u> 00 00 00) <sub>16</sub>

AES S-Box Lookup Table

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

        0
        63
        7
        7
        78
        F2
        68
        6F
        C5
        30
        01
        67
        28
        FE
        D7
        AB
        76

        1
        CA
        82
        C9
        7D
        FA
        59
        47
        FO
        AD
        D4
        A2
        AF
        90
        A4
        72
        CO

        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
        81
        18
        62
        5A
        A0
        52
        3B
        D6
        B2
        29
        20
        20
```

For example HEX 19 would get replaced with HEX D4



```
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)

 $w[8] = w[4] \oplus g(w[7]) = (56 68 61 74)$ :

 $w[9] = w[8] \oplus w[5] = (C7.7A F0 FC),$  $w[10] = w[9] \oplus w[6] = (76\ 23\ 14\ 1A),$  $w[11] = w[10] \oplus w[7] = (A0 5A B6 89)$ 

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	( <u><b>01</b></u> 00 00 00) <sub>16</sub>	6	( <u>20</u> 00 00 00) <sub>16</sub>
2	( <u>02</u> 00 00 00) <sub>16</sub>	7	$(\underline{40}\ 00\ 00\ 00)_{16}$
3	( <u>04</u> 00 00 00) <sub>16</sub>	8	( <u><b>80</b></u> 00 00 00) <sub>16</sub>
4	( <u>08</u> 00 00 00) <sub>16</sub>	9	( <u>1B</u> 00 00 00) <sub>16</sub>
5	( <u>10</u> 00 00 00) <sub>16</sub>	10	( <u><b>36</b></u> 00 00 00) <sub>16</sub>



```
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) $w[12] = w[8] \oplus g(w[11]) = ()$ :

$$w[13] = w[12] \bigoplus w[9] = (),$$
  
 $w[14] = w[13] \bigoplus w[10] = (),$   
 $w[15] = w[14] \bigoplus w[11] = ()$   
Third roundkey:

Round	Constant (RCon)	Round	Constant (RCon)
1	( <u>01</u> 00 00 00) <sub>16</sub>	6	( <u>20</u> 00 00 00) <sub>16</sub>
2	( <u>02</u> 00 00 00) <sub>16</sub>	7	( <u>40</u> 00 00 00) <sub>16</sub>
3	( <u>04</u> 00 00 00) <sub>16</sub>	8	( <u>80</u> 00 00 00) <sub>16</sub>
4	( <u>08</u> 00 00 00) <sub>16</sub>	9	( <u>1B</u> 00 00 00) <sub>16</sub>
5	( <u>10</u> 00 00 00) <sub>16</sub>	10	( <u>36</u> 00 00 00) <sub>16</sub>



Key Words	Auxiliary Function
w0 = 0f 15 71 c9	RotWord (w3) = 7f 67 98 af = x1
w1 = 47 d9 e8 59	SubWord (x1) = d2 85 46 79 = y1
w2 = 0c b7 ad d6	Rcon (1) = 01 00 00 00
w3 = af 7f 67 98	y1 $\oplus$ Rcon (1) = d3 85 46 79 = z1
w4 = w0	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 $\oplus$ z2 = d2 c9 6b b7	RotWord (w11) = ff d3 c6 e6 = x3
w9 = w8 $\oplus$ w5 = 49 80 b4 5e	SubWord (x3) = 16 66 b4 83 = y3
w10 = w9 $\oplus$ w6 = de 7e c6 61	Rcon (3) = 04 00 00 00
w11 = w10 $\oplus$ w7 = e6 ff d3 c6	y3   Rcon (3) = 12 66 b4 8e = z3
$w12 = w8 \oplus z3 = c0$ af df 39	RotWord (w15) = ae 7e c0 b1 = x4
$w13 = w12 \oplus w9 = 89$ 2f 6b 67	SubWord (x4) = e4 f3 ba c8 = y4
$w14 = w13 \oplus w10 = 57$ 51 ad 06	Rcon (4) = 08 00 00 00
$w15 = w14 \oplus w11 = b1$ ae 7e c0	y4   Rcon (4) = ec f3 ba c8 = 4

Key Words	Auxiliary Function
w16 = w12	RotWord (w19) = 8c dd 50 43 = x5 SubWord (x5) = 64 c1 53 1a = y5 Rcon(5) = 10 00 00 00 y5   Reon (5) = 74 c1 53 1a = z5
w20 = w16 $\oplus$ z5 = 58 9d 36 eb	RotWord (w23) = 40 46 bd 4c = x6
w21 = w20 $\oplus$ w17 = fd ee 38 7d	SubWord (x6) = 09 5a 7a 29 = y6
w22 = w21 $\oplus$ w18 = 0f cc 9b ed	Rcon(6) = 20 00 00 00
w23 = w22 $\oplus$ w19 = 4c 40 46 bd	y6   Rcon(6) = 29 5a 7a 29 = z6
$w24 = w20 \oplus z6 = 71 \text{ c7 4c c2}$	RotWord (w27) = a5 a9 ef cf = x7
$w25 = w24 \oplus w21 = 8c 29 74 \text{ bf}$	SubWord (x7) = 06 d3 bf 8a = y7
$w26 = w25 \oplus w22 = 83 \text{ e5 ef 52}$	Rcon (7) = 40 00 00 00
$w27 = w26 \oplus w23 = \text{cf a5 a9 ef}$	y7   Rcon(7) = 46 d3 df 8a = z7
w28 = w24 $\oplus$ z7 = 37 14 93 48	RotWord (w31) = 7d a1 4a f7 = x8
w29 = w28 $\oplus$ w25 = bb 3d e7 f7	SubWord (x8) = ff 32 d6 68 = y8
w30 = w29 $\oplus$ w26 = 38 d8 08 a5	Rcon (8) = 80 00 00 00
w31 = w30 $\oplus$ w27 = f7 7d a1 4a	y8   Rcon (8) = 7f 32 d6 68 = z8
w32 = w28	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 $\oplus$ z9 = fd 0d 42 cb	RotWord (w39) = 6b 41 56 f9 = x10
w37 = w36 $\oplus$ w33 = 0e 16 e0 1c	SubWord (x10) = 7f 83 b1 99 = y10
w38 = w37 $\oplus$ w34 = c5 d5 4a 6e	Rcon (10) = 36 00 00 00
w39 = w38 $\oplus$ w35 = f9 6b 41 56	y10 $\oplus$ Rcon (10) = 49 83 b1 99 = z10
w40 = w36 $\oplus$ z10 = b4 8e f3 52 w41 = w40 $\oplus$ w37 = ba 98 13 4e w42 = w41 $\oplus$ w38 = 7f 4d 59 20 w43 = w42 $\oplus$ w39 = 86 26 18 76	



```
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 \oplus SubByte(K[n-1]: w_3 >> 8) \oplus Rcon[i]

K[2]: w_0 = 47 6f 20 74 \oplus subbyte (20 24 3f 4D) \oplus Rcon[0]

= 47 6f 20 74 \oplus B7 36 75 E3 \oplus Rcon[0]

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

= 71 38 55 97 \oplus 01 00 00 00

K[2]: w_0 = 70 38 55 97
```

```
= 01000000
Rcon(0)
             = 02000000
Rcon(1)
Rcon(2)
             = 04000000
             = 08000000
Rcon(3)
Rcon(4)
             = 10000000
             = 20000000
Rcon(5)
Rcon(6)
             = 40000000
            = 80000000
Rcon (7)
             = 1B000000
Rcon(8)
             = 36000000
Rcon (9)
             = 60000000
Rcon (10)
             = D8000000
Rcon (11)
             = AB000000
Rcon (12)
             = 4D000000
Rcon (13)
             = 9A000000
Rcon (14)
```



# Simplified AES Example

Let know that the first sub key is 6F 32 Determine the second sub key  $w_0 = 6F$ 

```
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

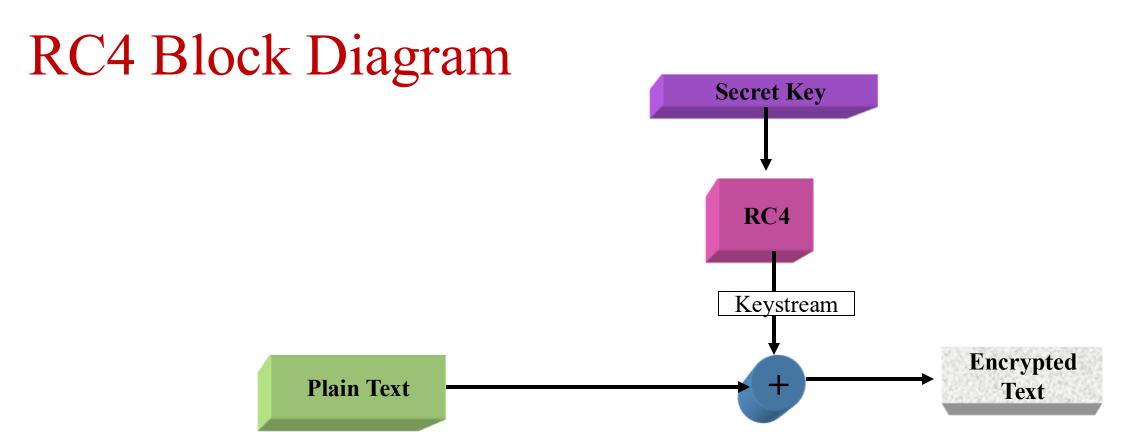
SSL/TLS (Secure socket, transport layer security) between web browsers and servers, IEEE 802.11 wirelss LAN std: WEP (Wired Equivalent Privacy), WPA (WiFi Protocol Access) protocol



# RC4-based Usage

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





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

KSA PRGA



## 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



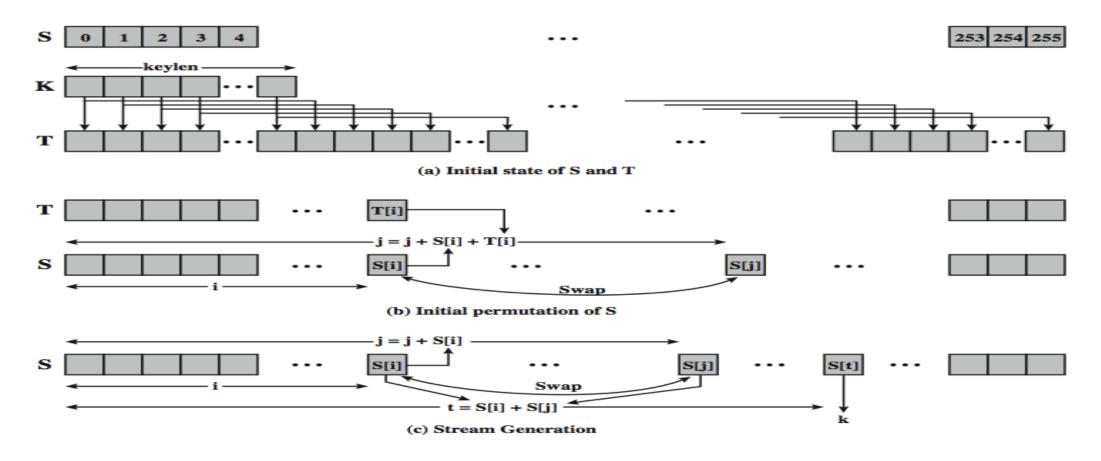
## The PRGA

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

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

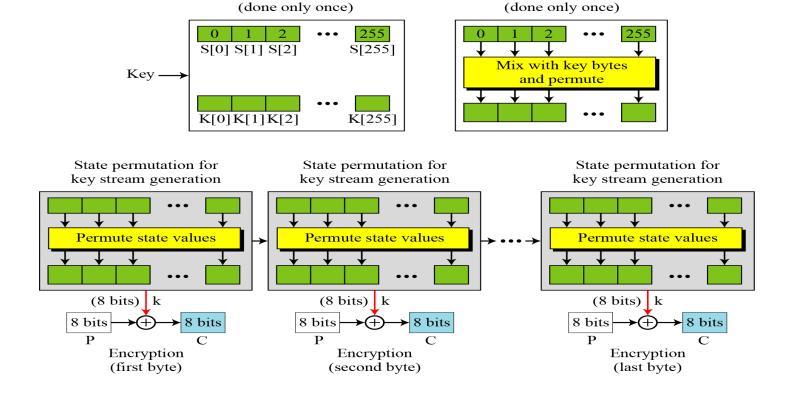


# Detailed Diagram





# Overall Operation of RC4



Initial state permutation

State and key initialization



## 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:

$$(A xor B) xor B = A$$

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".