UNIT-2

1.1 BLOCK CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a **comparison of stream cipher with block cipher.**

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher. **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

Most symmetric block ciphers are based on a **Feistel Cipher Structure** needed since must be able to **decrypt** ciphertext to recover messages efficiently. block ciphers look like an extremely largesubstitution would need table of 264 entries for a 64-bitblock instead create from smaller buildingblocksusing idea of a product cipher in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks called modern substitution-transposition productcipherthese form the basis of modern block ciphers

S-Pnetworksarebasedonthetwoprimitivecryptographicoperationswehaveseenbefore:

- *substitution*(S-box)
- *permutation* (P-box)
- provide *confusion* and *diffusion* ofmessage

diffusion– dissipates statistical structure of plaintext over bulk ofciphertext

confusion–makesrelationshipbetweenciphertextandkeyascomplexaspossible

FEISTEL CIPHER STRUCTURE

The input to the encryption algorithm are a plaintext block of length 2w bits and a key K. the plaintext block is divided into two halves L_0 and R_0. The two halves of the data through„n"roundsofprocessingandthencombinetoproducetheciphertextblock.Eachround„i" has inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as the subkey K_i, derived from the overall key K. in general, the subkeys $K_i$are different from K and from eachother.

All rounds have the same structure. A substitution is performed on the left half of the data (as similar to S-DES). This is done by applying a round function F to the right half of the data and then taking the XOR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round sub key k_i. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network. The exact realization of a Feistel network depends on the choice of the following parameters and designfeatures:

Block size - Increasing size improves security, but slows cipher

Key size - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher

Number of rounds - Increasing number improves security, but slows cipher

Subkey generation - Greater complexity can make analysis harder, but slows cipher

Round function - Greater complexity can make analysis harder, but slows cipher

Fast software en/decryption & ease of analysis - are more recent concerns for practical use andtesting.

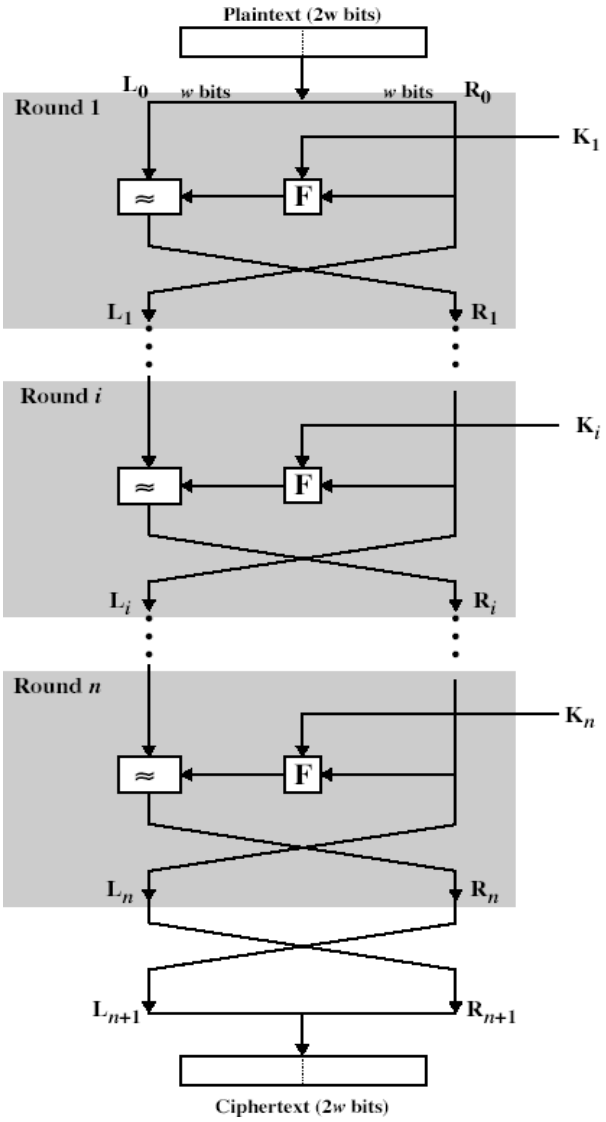


Fig: Classical Feistel Network

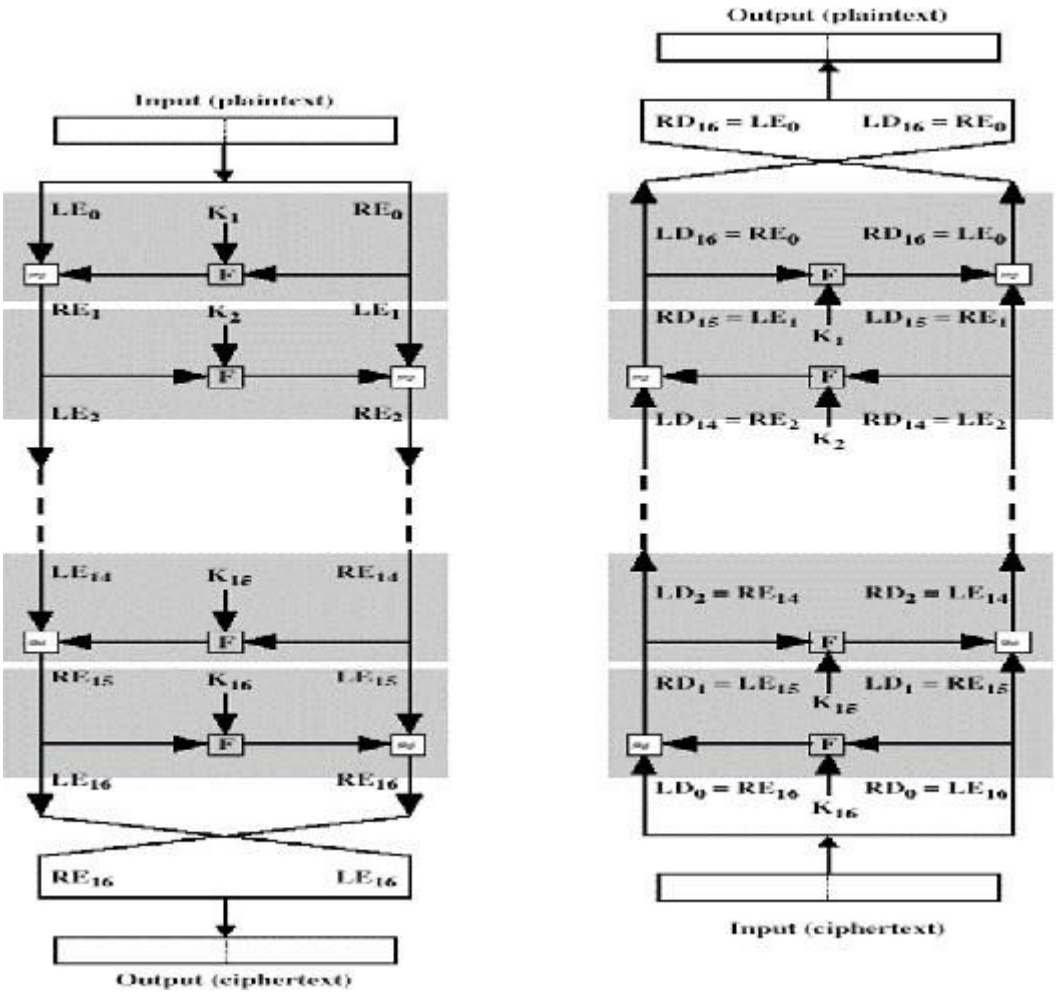


Fig: Feistel encryption and decryption

The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey k_i in reverse order. i.e., k_n in the first round, k_{n-1} in second round and so on. For clarity, we use the notation LE_i and RE_i for data traveling through the decryption algorithm. The diagram below indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the valueswapped.

i.e., $RE_i || LE_i$ (or) equivalently $RD_{16-i} || LD_{16-i}$

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is $RE_{16} || LE_{16}$. The output of that round is the cipher text. Now take the cipher text and use it as input to the same algorithm. The input to the first round isRE_{16}

$|| LE_{16}$, which is equal to the 32-bit swap of the output of the sixteenth round of the encryptionprocess.

Now we will see how the output of the first round of the decryption process is equal to a

32-bit swap of the input to the sixteenth round of the encryption process. First consider the encryption process,

LE16 = RE15

RE16 = LE15(+) F (RE15, K16)

On the decryption side, LD1 =RD0 = LE16 =RE15 RD1 = LD0 (+) F (RD0, K16)

= RE16 F (RE15, K16)

= [LE15 F (RE15, K16)] F (RE15, K16)

= LE15

Therefore, LD1 = RE15 RD1 = LE15 In general, for the ith iteration of the encryption algorithm, LEi = REi-1 REi = LEi-1 F (REi-1, Ki)

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

1.2 DATA ENCRYPTION STANDARD (DES)

In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications response was mostly disappointing, however IBM submitted their Lucifer design following a period of redesign and comment it became the Data Encryption Standard (DES).

DES is a symmetric-key algorithm for the encryption of electronic data. Developed in the early 1970s at IBM and based on an earlier design by Horst Feistel, the algorithm was submitted to the National Bureau of Standards (NBS) following the agency's invitation to propose a candidate for the protection of sensitive, unclassified electronic government data.However, this has now been replaced by a new standard known as the Advanced Encryption Standard (AES). DES is a 64 bit block cipher which means that it encrypts data 64 bits at a time. This is contrasted to a stream cipher in which only one bit t a time (or sometimes small groups of bits such as a byte) is encrypted.Even though DES actually accepts a 64 bit key as input, the remaining eight bits are used for parity checking and have no effect on DES's security. Outsiders were convinced that the 56 bit key was an easy target for a brute force attack due to its extremely small size.DES of course isn't the only symmetric cipher. There are many others, each with varying levels of complexity. Such ciphers include: IDEA, RC4, RC5, RC6 and the new Advanced Encryption Standard (AES).

INNER WORKING OF DES

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. It consists of a number of rounds where each round contains bit- shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. As with most encryption schemes, DES expects two inputs - the plaintext to be encrypted and the secret key. The manner in which the plaintext is accepted, and the key arrangement used for encryption and decryption, both determine the type of cipher it is. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and only operates on 64 bit blocks of data at a time5 (be they plaintext or ciphertext). The key size used is 56 bits, however a 64 bit(or eight-byte) key is actually input. The least significant bit of each byte is either used for parity (odd for DES) or set arbitrarily and does not increase the security in any way. All blocks are numbered from left to right which makes the eight bit of each byte the parity bit. Once a plain-text message is received to be encrypted, it is arranged into 64 bit blocks required forinput.

OVERALL STRUCTURE

Figure below shows the sequence of events that occur during an encryption operation. DES performs an initial permutation on the entire 64 bit block of data. It is then split into 2, 32 bit sub-blocks, Li and Ri which are then passed into what is known as a round (see figure 2.3), of which there are 16 (the

subscript i in Li and Ri indicates the current round). Each of the rounds are identical and the effects of increasing their number is twofold - the algorithms security is increased and its temporal efficiency decreased. Clearly these are two conflicting outcomes and a compromise must be made. For DES the number chosen was 16, probably to guarantee the elimination of any correlation between the ciphertext and either the plaintext or key6 . At the end of the 16th round, the 32 bit Li and Ri output quantities are swapped to create what is known as the pre-output. This [R16, L16] concatenation is permuted using a function which is the exact inverse of the initial permutation. The output of this final permutation is the 64 bit ciphertext.

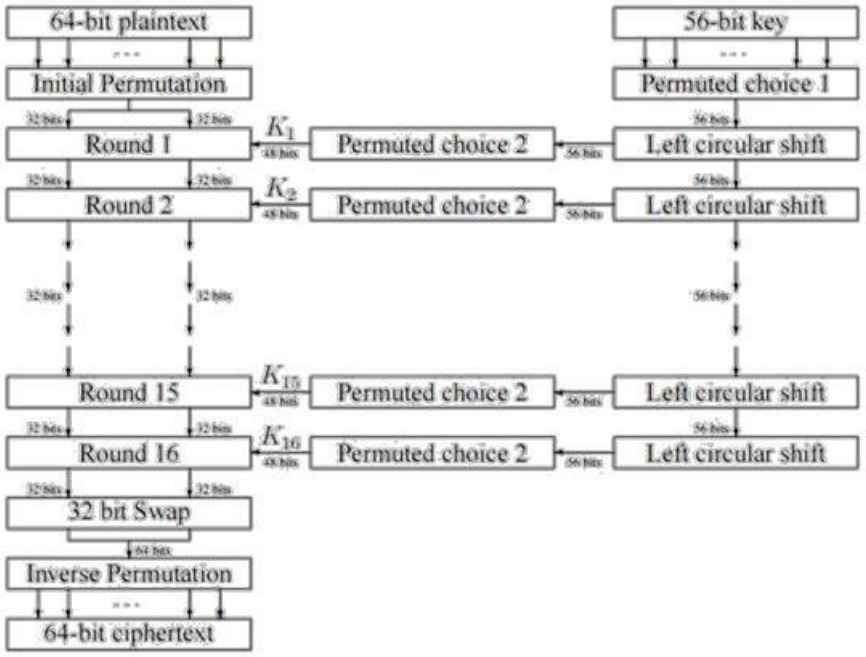


Figure : Flow Diagram of DES algorithm for encrypting data.

So in total the processing of the plaintext proceeds in three phases as can be seen from the left hand side of figure

1. Initial permutation (IP - defined in table 2.1) rearranging the bits to form the "permutedinput".
2. Followed by 16 iterations of the same function (substitution and permutation). The output of the last iteration consists of 64 bits which is a function of the plaintext and key. The left and right halves are swapped to produce thepreoutput.

3. Finally, the preoutput is passed through a permutation (IP−1 - defined in table 2.1) which is simply the inverse of the initial permutation (IP). The output of IP−1 is the 64-bit ciphertext

(a) Initial Permutation (IP)

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

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

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

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

Table 2.1: Permutation tables used in DES.

As figure shows, the inputs to each round consist of the Li , Ri pair and a 48 bit subkey which is a shifted and contracted version of the original 56 bit key. The use of the key can be seen in the right hand portion of figure 2.2: • Initially the key is passed through a permutation function (PC1 - defined in table 2.2) • For each of the 16 iterations, a subkey (Ki) is produced by a combination of a left circular shift and a permutation (PC2 - defined in table 2.2) which is the same for each iteration. However, the resulting subkeyis different for each iteration because of repeatedshifts.

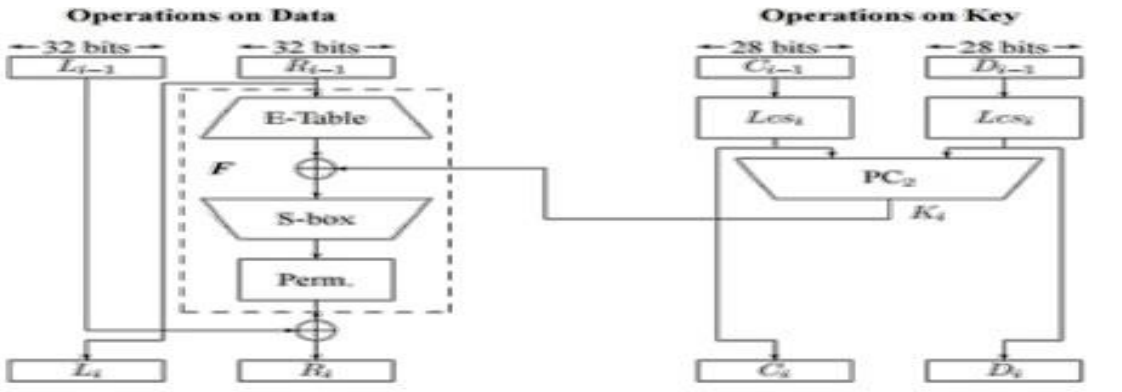


Figure : Details of a single DES round.

The main operations on the data are encompassed into what is referred to as the cipher function and is labeled F. This function accepts two different length inputs of 32 bits and 48 bits and outputs a single 32 bit number. Both the data and key are operated on in parallel, however the operations are quite different.The 56 bit key is split into two 28 bit halves Ci and Di (C and D being chosen so as not to be confused with L and R). The value of the key used in any round is simply a left cyclic shift and a permuted contraction of that used in the previous round. Mathematically, this can be written as

Ci = Lcsi(Ci−1), Di = Lcsi(Di−1) Ki = P C2(Ci , Di)

where Lcsi is the left cyclic shift for round i, Ci and Di are the outputs after the shifts, P C2(.) is a function which permutes and compresses a 56 bit number into a 48 bit number and Ki is the actual key used in round i. The number of shifts is either one or two and is determined by the round number i.

S-BOX

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 2.3: S-box details.

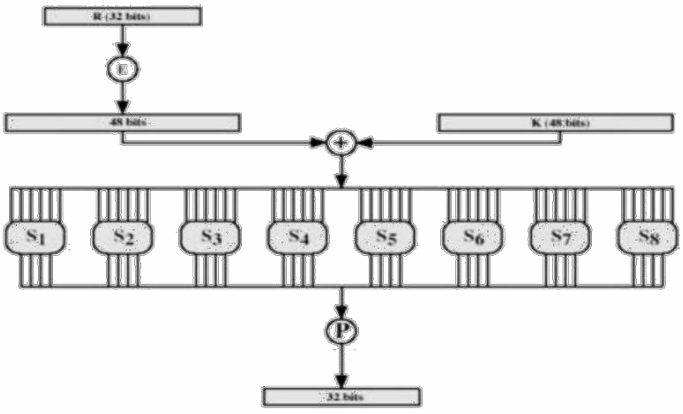


Figure 2.4: The complex F function of the DES algorithm.

1.3 ADVANCED ENCRYPTION ALGORITHM (AES)

AES is a block cipher with a block length of 128bits.

AES allows for three different key lengths: 128, 192, or 256 bits. Most ofour discussion will assume that the key length is 128bits.

Encryption consists of 10 rounds of processing for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

Except for the last round in each case, all other rounds are identical.

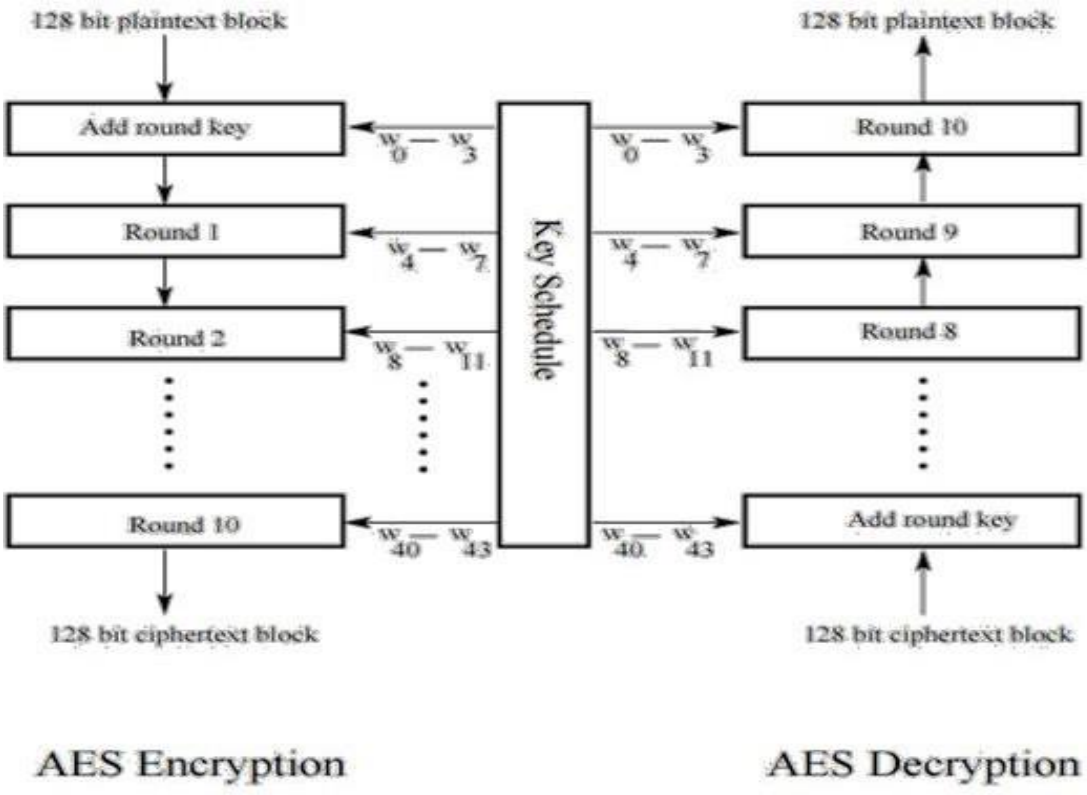
Each round of processing includes one single-byte based substitution step, a row-wise permutation step, a column-wise mixing step, and the addition of the round key. The order in which these four steps are executed is different for encryption and decryption. To appreciate the processing steps used in a single round, it is best to think of a

128-bit block as consisting of a 4 × 4 matrix of bytes, arranged as follows:

$$\begin{bmatrix} byte_0 & byte_4 & byte_8 & byte_{12} \\ byte_1 & byte_5 & byte_9 & byte_{13} \\ byte_2 & byte_6 & byte_{10} & byte_{14} \\ byte_3 & byte_7 & byte_{11} & byte_{15} \end{bmatrix}$$

Therefore, the first four bytes of a 128-bit input block occupy the first column in the 4

× 4 matrix of bytes. The next four bytes occupy the second column, and so on. The 4×4 matrix of bytes shown above is referred to as the state array in AES.



The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages.

This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of its counterpart in the encryption algorithm.

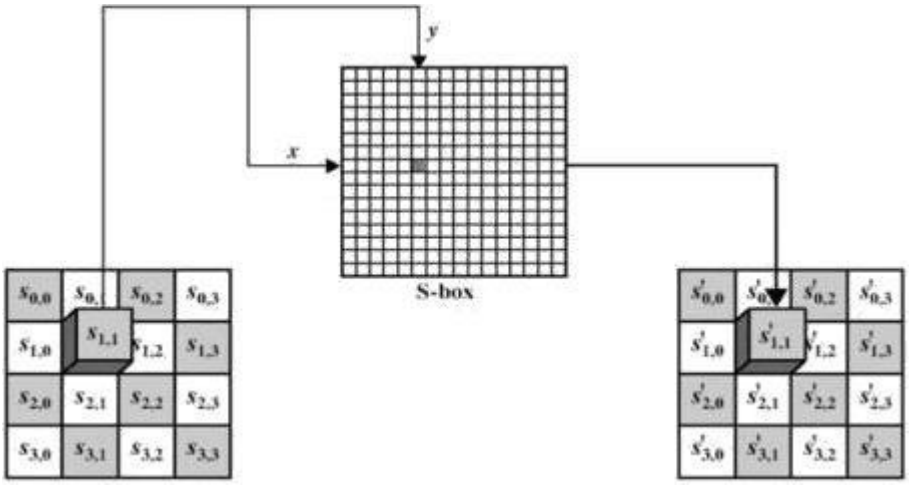
The four stages are as follows: 1. Substitute bytes 2. Shift rows 3. Mix Columns 4. Add Round Key.

Substitute Bytes

- This stage (known as SubBytes) is simply a table lookup using a 16 × 16 matrix of byte values called ans-box.
- This matrix consists of all the possible combinations of an 8 bit sequence (28 = 16 × 16 = 256).
- However, the s-box is not just a random permutation of these values and there is a well defined method for creating the s-boxtables.
- The designers of Rijndael showed how this was done unlike the s-boxes in DES for which no rationale was given.Our concern will be how state is effected in eachround.

- For this particular round each byte is mapped into a new byte in the following way: the leftmost nibble of the byte is used to specify a particular row of the s-box and the rightmost nibble specifies acolumn.
- For example, the byte {95} (curly brackets represent hex values in FIPS PUB 197)selects row 9 column 5 which turns out to contain the value {2A}.

• This is then used to update the statematrix.

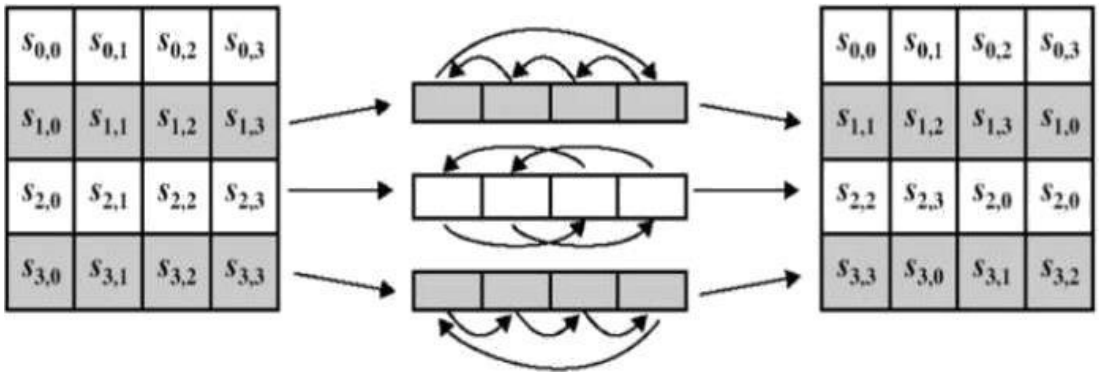


Shift Row Transformation

• This stage (known as ShiftRows) is shown in figurebelow.

• Simple permutation annothingmore.

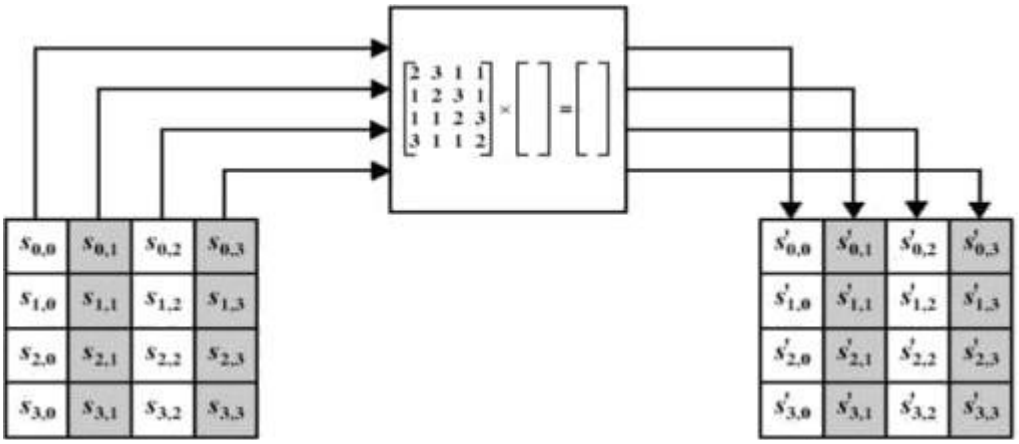
- It works as follow: – The first row of state is not altered. – The second row is shifted 1 bytes to the left in a circular manner. – The third row is shifted 2 bytes to the left in a circular manner. – The fourth row is shifted 3 bytes to the left in a circularmanner.



MIX COLUMN TRANSFORMATION

- This stage (known as MixColumn) is basically asubstitution
  - Each column is operated on individually. Each byte of a column is mapped into a new value that is a function of all four bytes in thecolumn.
- The transformation can be determined by the following matrix multiplication onstate
  - Each element of the product matrix is the sum of products of elements of one row andone column.
- In this case the individual additions and multiplications are performed in GF(28).
  - The MixColumns transformation of a single column j ($0 \leq j \leq 3$) of state can beexpressed as:

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



ADD ROUND KEY TRANSFORMATION

- In this stage (known as AddRoundKey) the 128 bits of state are bitwise XORedwiththe 128 bits of the roundkey.
- The operation is viewed as a columnwise operation between the 4 bytes of a state column and one word of the roundkey.
- This transformation is as simple as possible which helps in efficiency but it also effects every bit ofstate.
- The AES key expansion algorithm takes as input a 4-word key and produces a linear array of 44 words. Each round uses 4 of these words as shown infigure.
- Each word contains 32 bytes which means each subkeyis 128 bits long. Figure 7 show pseudocode for generating the expanded key from the actualkey.

1.4 BLOWFISH ALGORITHM

A symmetric block cipher designed by Bruce Schneierin1993/94

characteristics

- fast implementation on 32-bitCPUs

- compact in use ofmemory
- simple structure foranalysis/implementation
- variable security by varying keysize
- has been implemented in variousproducts

BLOWFISH KEY SCHEDULE

- uses a 32 to 448 bit key, 32-bit words stored in K-array Kj,j from 1 to14
- used togenerate
- 18 32-bit subkeys stored in P array, P1….P18
- four 8x32 S-boxes stored in Si,j , each with 256 32-bitentries

Subkeys and S-Boxes Generation:

1. initialize P-array and then 4 S-boxes in order using the fractional part of pi P1 (left most 32-bit), and so on,,,S4,255.
2. XOR P-array with key-Array (32-bit blocks) and reuse as needed: assume we haveup to k10 then P10 XOR K10,, P11 XOR K1 … P18 XOR K8
3. Encrypt 64-bit block of zeros, and use the result to update P1 andP2.
4. encrypting output form previous step using current P & S and replace P3 andP4. Then encrypting current output and use it to update successive pairsofP.
5. After updating all P's (last :P17 P18), start updating S values using theencrypted output from previousstep.

- requires 521 encryptions, hence slow inre-keying
- Not suitable for limited-memoryapplications.

BLOWFISH ENCRYPTION

- uses two main operations: addition modulo 232 , andXOR
- data is divided into two 32-bit halves *L0* &*R0*

for*i*= 1 to 16 do

Ri= *Li-1* XOR *Pi*;

Li = F[*Ri*] XOR*Ri-1*; *L*17 = *R*16 XOR *P*18; *R*17 = *L*16 XOR *P*17;

- where

F[*a*,*b*,*c*,*d*] = ((S1,*a* + S2,b) XOR S3,c) + S4,d

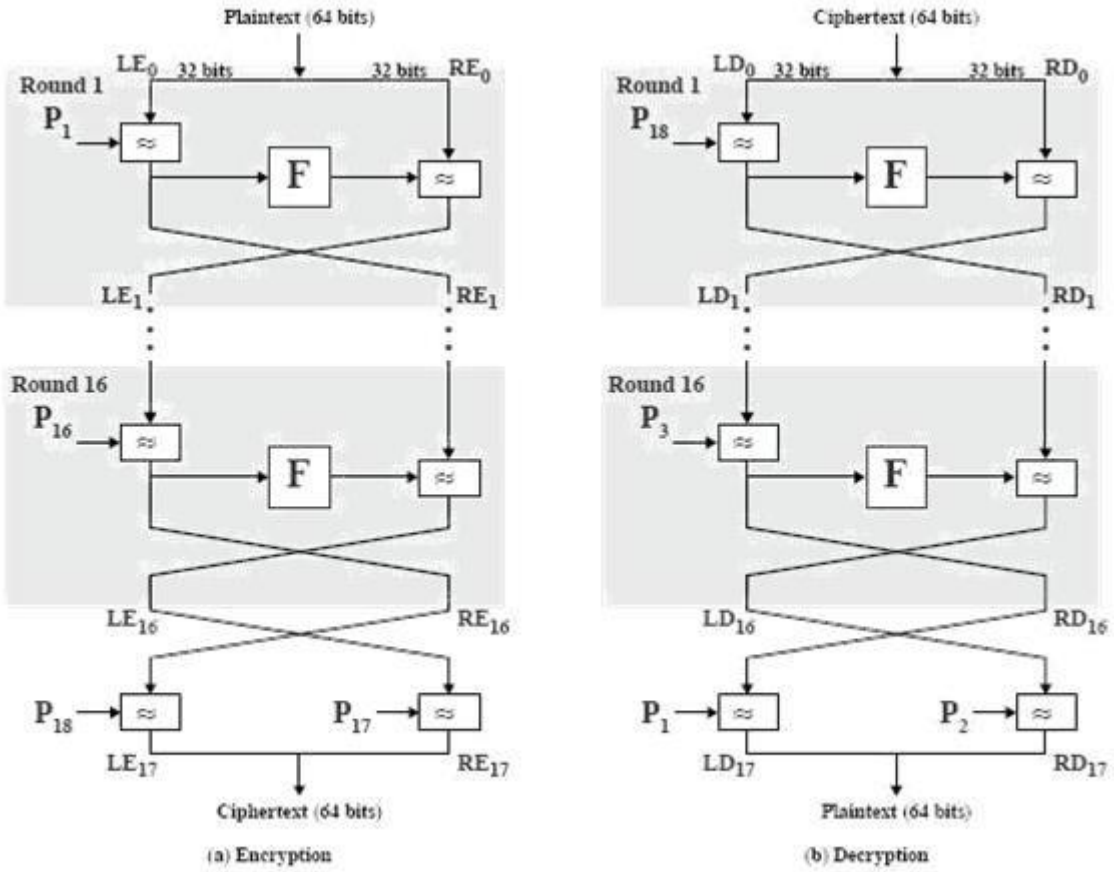


Figure 6.3 Blowfish Encryption and Decryption

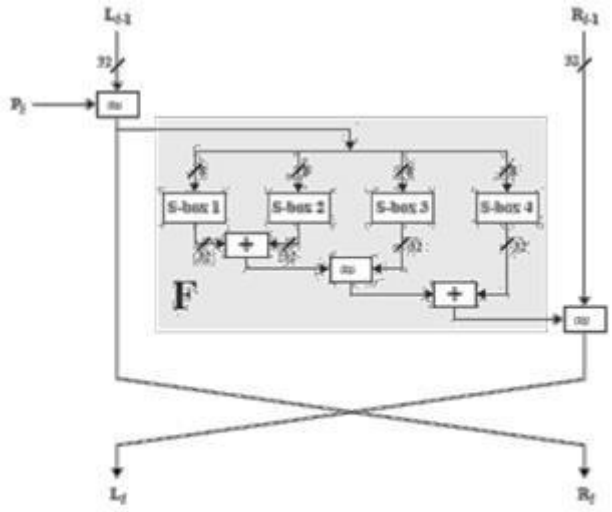


Figure 6.4 Detail of Single Blowfish Round

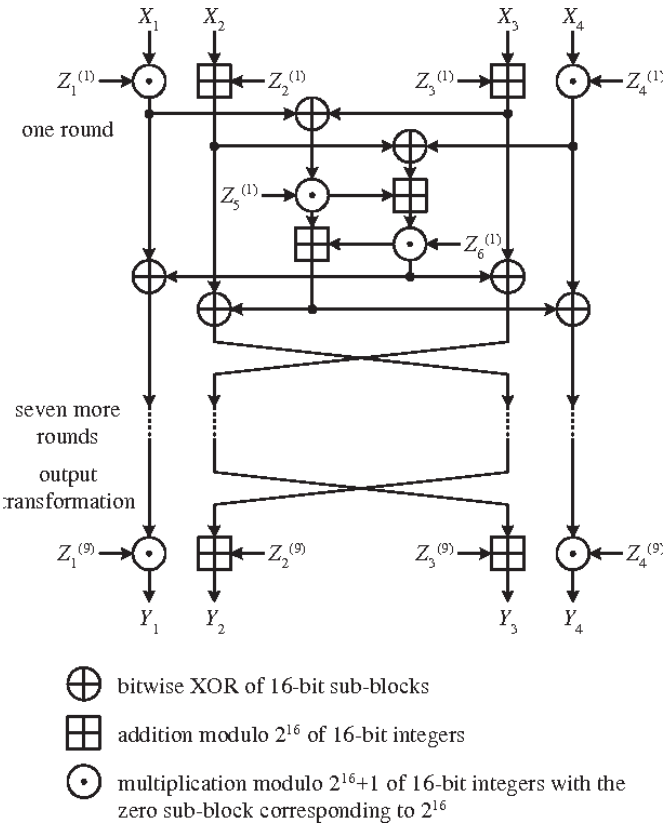
1.5 IDEA (IPES)

- DES algorithm has been a popular secret key encryption algorithm and is used in many commercial and financial applications. However, its key size is too small by current standards and its entire 56 bit key space can be searched in approximately 22 hours
- IDEA is a block cipher designed by Xuejia Lai and James L. Massey in 1991
- It is a minor revision of an earlier cipher, PES (Proposed Encryption Standard)
- IDEA was originally called IPES (Improved PES) and was developed to replace DES
- It entirely avoids the use of any lookup tables or S-boxes
- IDEA was used as the symmetric cipher in early versions of the Pretty Good Privacy cryptosystem
- IDEA operates with 64-bit plaintext and cipher text blocks and is controlled by a 128-bit key
- Completely avoid substitution boxes and table lookups used in the block ciphers
- The algorithm structure has been chosen such that when different key sub-blocks are used, the encryption process is identical to the decryption process

Key generation

The 64-bit plaintext block is partitioned into four 16-bit sub-blocks

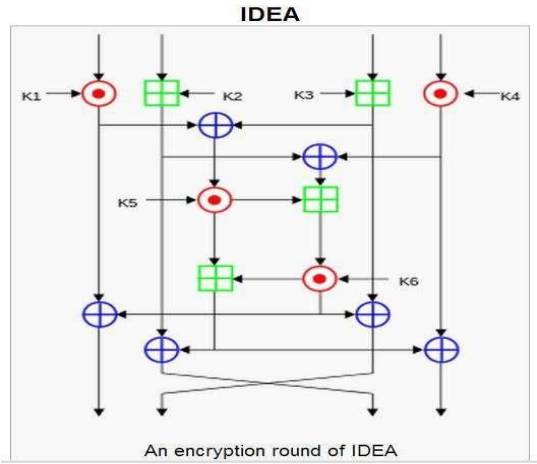
- six 16-bit key are generated from the 128-bit key. Since a further four 16-bit key-sub-blocks are required for the subsequent output transformation, a total of 52 (= 8 x 6 + 4) different 16-bit sub-blocks have to be generated from the 128-bit key.



Key generation process

- First, the 128-bit key is partitioned into eight 16-bit sub-blocks which are then directly used as the first eight key sub-blocks
- The 128-bit key is then cyclically shifted to the left by 25 positions, after which the resulting 128-bit block is again partitioned into eight 16-bit sub-blocks to be directly used as the next eight key sub-blocks

- The cyclic shift procedure described above is repeated until all of the required 52 16-bit key sub-blocks have been generated
- **Sequence of operation in one round**



1) Multiply P1 andK1
2) Add P2 and secondK2
3) Add P3 and thirdK3
4) Multiply P4 andK4
5) Step 1 ⊕ step3
6) Step 2 ⊕ step4
7) Multiply step 5 with k5.
8) Add result of step 6 and step7
9) Multiply result of step 8 withK6.
10) Add result of step 7 and step9.

11) XOR result of steps 1 and step9.

12) XOR result of steps 3 and step9.

13) XOR result of steps 2 and step10.

14) XOR result of steps 4 and step10.

Encryption of the key sub-blocks

- The key sub-blocks used for the encryption and the decryption in the individual rounds are shown in Table 1

Encryption of the key sub-blocks

Round 1	$Z_1^{(1)}\ Z_2^{(1)}\ Z_3^{(1)}\ Z_4^{(1)}\ Z_5^{(1)}\ Z_6^{(1)}$
Round 2	$Z_1^{(2)}\ Z_2^{(2)}\ Z_3^{(2)}\ Z_4^{(2)}\ Z_5^{(2)}\ Z_6^{(2)}$
Round 3	$Z_1^{(3)}\ Z_2^{(3)}\ Z_3^{(3)}\ Z_4^{(3)}\ Z_5^{(3)}\ Z_6^{(3)}$
Round 4	$Z_1^{(4)}\ Z_2^{(4)}\ Z_3^{(4)}\ Z_4^{(4)}\ Z_5^{(4)}\ Z_6^{(4)}$
Round 5	$Z_1^{(5)}\ Z_2^{(5)}\ Z_3^{(5)}\ Z_4^{(5)}\ Z_5^{(5)}\ Z_6^{(5)}$
Round 6	$Z_1^{(6)}\ Z_2^{(6)}\ Z_3^{(6)}\ Z_4^{(6)}\ Z_5^{(6)}\ Z_6^{(6)}$
Round 7	$Z_1^{(7)}\ Z_2^{(7)}\ Z_3^{(7)}\ Z_4^{(7)}\ Z_5^{(7)}\ Z_6^{(7)}$
Round 8	$Z_1^{(8)}\ Z_2^{(8)}\ Z_3^{(8)}\ Z_4^{(8)}\ Z_5^{(8)}\ Z_6^{(8)}$
Output Transform	$Z_1^{(9)}\ Z_2^{(9)}\ Z_3^{(9)}\ Z_4^{(9)}$

Table 1

Encryption

- the first four 16-bit key sub-blocks are combined with two of the 16-bit plaintext blocks using addition modulo 2^{16}, and with the other two plaintext blocks using multiplication modulo $2^{16} + 1$
  - At the end of the first encryption round four 16-bit values are produced which are used as input to the second encryption round

- The process is repeated in each of the subsequent 7 encryption rounds
- The four 16-bit values produced at the end of the 8th encryption round are combined with the last four of the 52 key sub-blocks using addition modulo 2^{16} and multiplication modulo $2^{16} + 1$ to form the resulting four 16-bit ciphertext blocks

Decryption

- The computational process used for decryption of the ciphertext is essentially the same as that used for encryption
- The only difference is that each of the 52 16-bit key sub-blocks used for decryption is the inverse of the key sub-block used during encryption
- In addition, the key sub-blocks must be used in the reverse order during decryption in order to reverse the encryption process

Applications of IDEA

- Today, there are hundreds of IDEA-based security solutions available in many market areas, ranging from Financial Services, and Broadcasting to Government
- The IDEA algorithm can easily be embedded in any encryption software. Data encryption can be used to protect data transmission and storage. Typical fields are:

– Audio and video data for cable TV, pay TV, video conferencing, distance learning
– Sensitive financial and commercial data
– Email via public networks
– Smart cards

1. 6BLOCK CIPHER MODES OFOPERATIONS

A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size.

Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

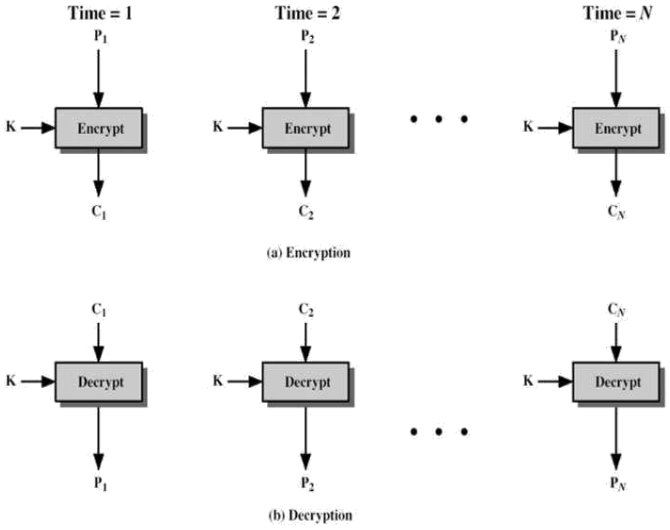
Block ciphers encrypt fixed size blocks

e.g., DES encrypts 64-bit block

For different applications and uses, there are several modes of operations for a block cipher

Electronic Code Book

- Direct use of the blockcipher
- Used primarily to transmit encryptedkeys
- Very weak if used for general-purpose encryption; never use it for a file or amessage.
- Attacker can build up codebook; no semanticsecurity
- We write $\{P\}k \rightarrow C$ to denote "encryption of plaintext P with key k to produceciphertext C"



Advantages and Limitations of ECB

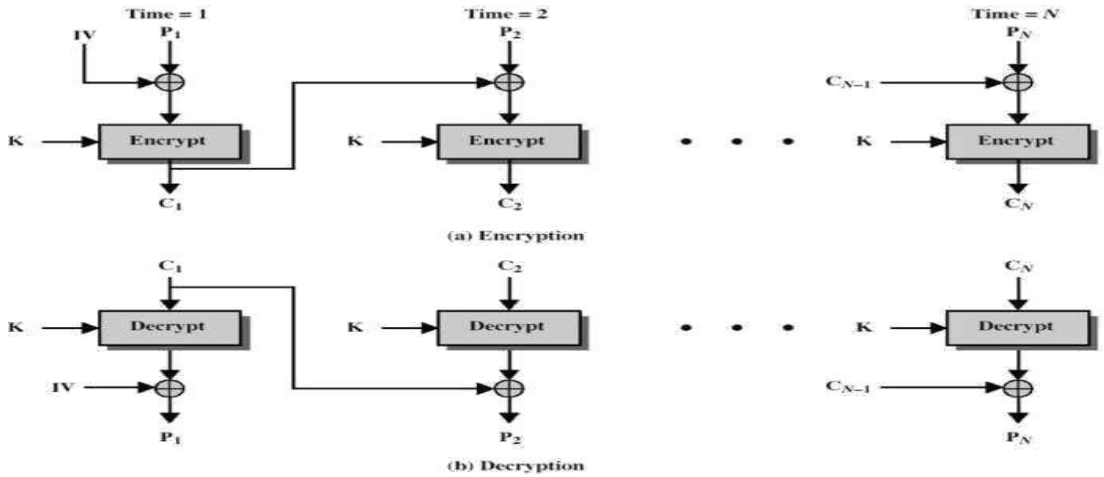
- Message repetitions may show in ciphertext
- If aligned with message block
- Particularly with data such graphics Or with messages that change very little, which become a code-book analysis problem
- Weakness is due to the encrypted message blocks being independent
- Vulnerable to cut-and-paste attacks
- Main use is sending a few blocks of data

Cipher Block Chaining

- We would like that same plaintext blocks produce different ciphertextblocks.
  - Cipher Block Chaining (see figure) allows this by XORing each plaintext with the Ciphertext from the previous round (the first round using an Initialisation Vector(IV)).
- As before, the same key is used for eachblock.
- Decryption works as shown in the figure because of the properties of the XORoperation,

i.e. IV $\oplus$ IV $\oplus$ P = P where IV is the Initialisation Vector and P is the plaintext.

Obviously the IV needs to be known by both sender and receiver and it should be kept secret along with the key for maximumsecurity.



ADVANTAGES AND DISADVANTAGES OF CBC:

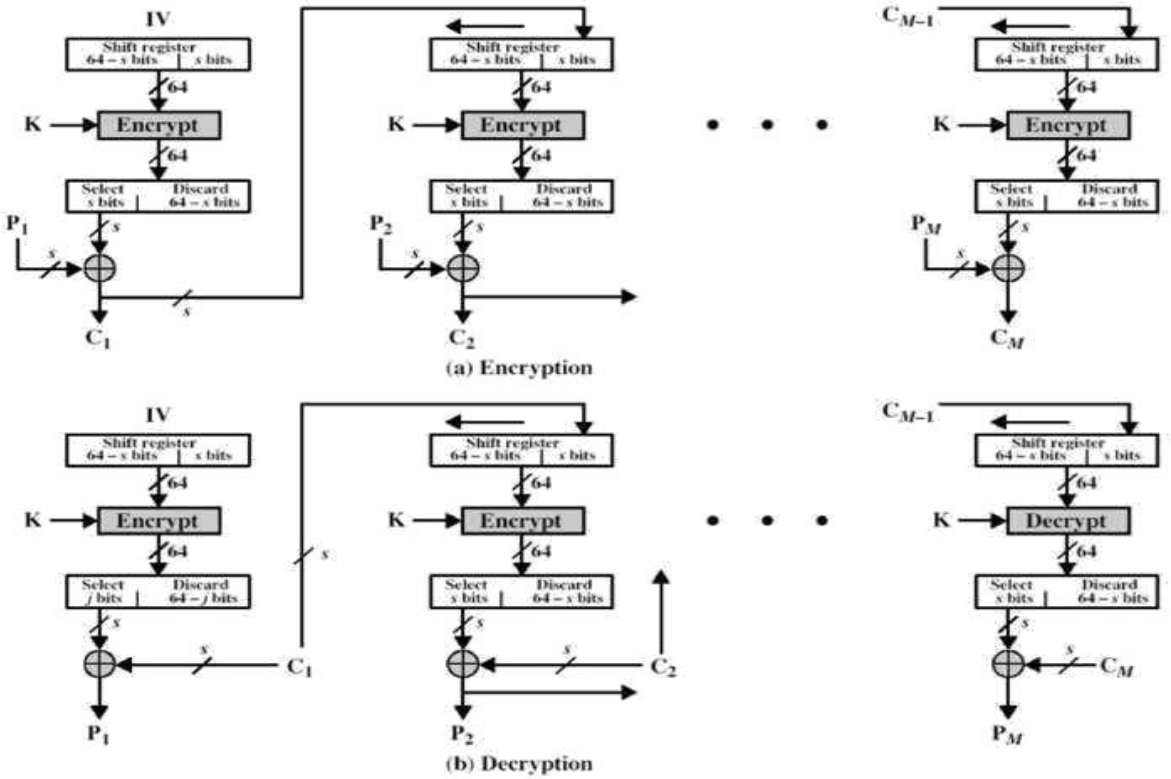
- A ciphertext block depends on all blocks before it
- Any change to a block affects all following ciphertext blocks...
- Need initialization vector (iv)
- Which must be known to sender & receiver
- If sent in clear, attacker can change bits of first block, by changing corresponding bits of iv
- Hence iv must either be a fixed value (as in eftpos)
- Or derived in way hard to manipulate
- Or sent encrypted in ecb mode before rest of message
- Or message integrity must be checked otherwise

CIPHER FEEDBACK (CFB) MODE

- The Cipher Feedback and Output Feedback allows a block cipher to be converted into a streamcipher.
- This eliminates the need to pad a message to be an integral number of blocks. It also can operate in realtime.
- Figure shows the CFB scheme.

In this figure it assumed that the unit of transmission is s bits; a common value is s =8

- As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext (which is split into s bitsegments).
- The input to the encryption function is a shift register equal in length to the block cipher of the algorithm (although the diagram shows 64 bits, which is block size used by DES, this can be extended to other block sizes such as the 128 bits ofAES).
- This is initially set to some Initialisation Vector(IV).



Advantages and Disadvantages of CFB

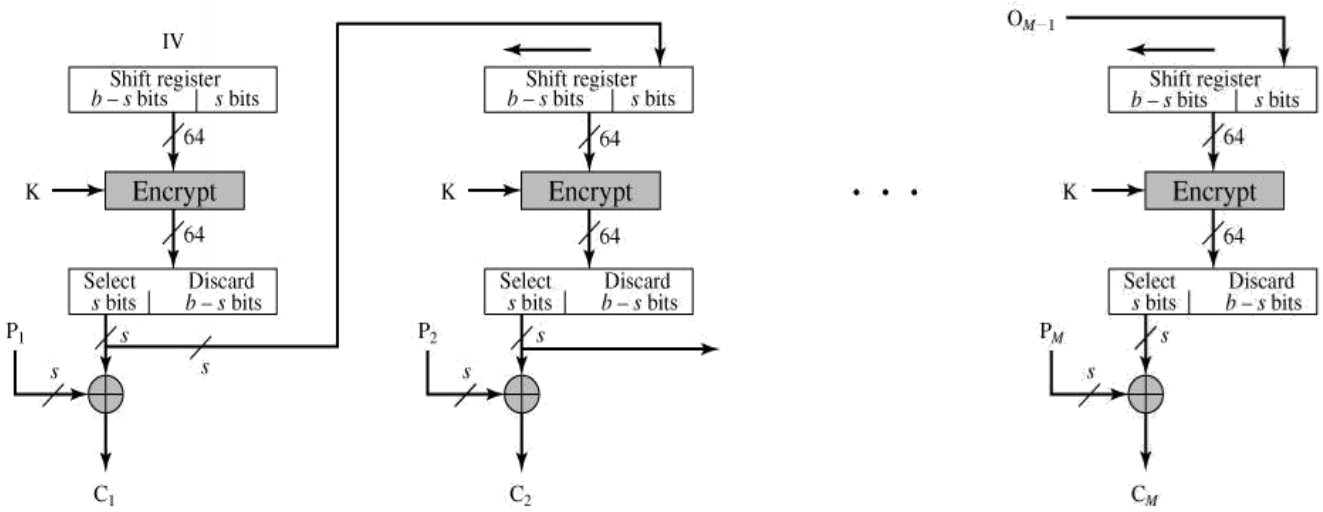
- Most common stream mode
- Appropriate when data arrives in bits/bytes
- Limitation is need to stall while do block encryption after every s-bits
- Note that the block cipher is used in **encryption** mode at **both** ends (xor)

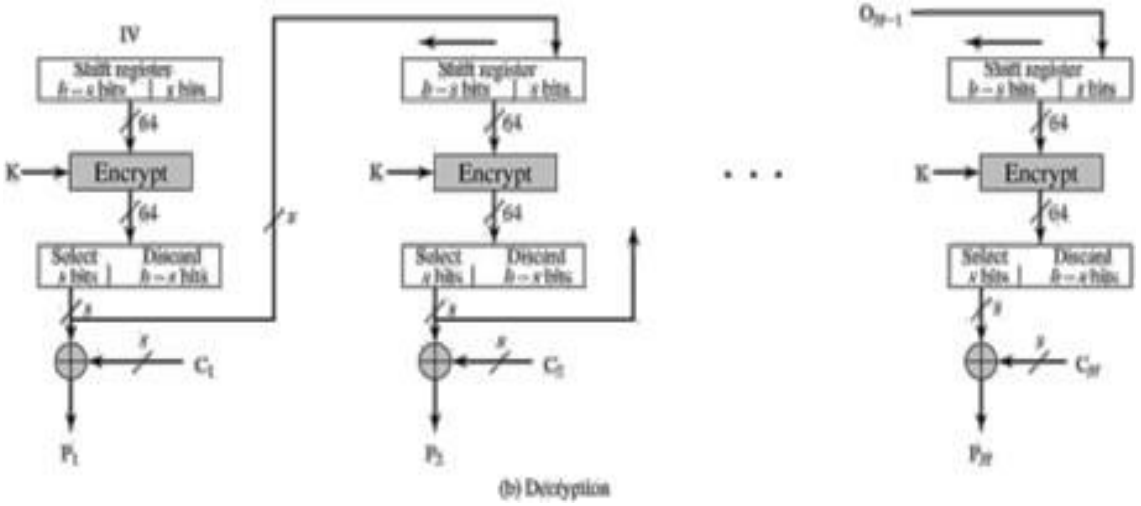
Errors propagate for several blocks after the error

OUTPUT FEEDBACK (OFB) MODE

- The Output Feedback Mode is similar in structure to that of CFB, as seen in figure13.
- As can be seen, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB the ciphertextunit is fed back to the shiftregister.
- One advantage of the OFB method is that bit errors in transmission do notpropagate.
  - For example, if a bit error occurs in C1 only the recovered value of P1 is affected; subsequent plaintext units are notcorrupted.

With CFB, C1 also serves as input to the shift register and therefore causes additional corruption downstream.



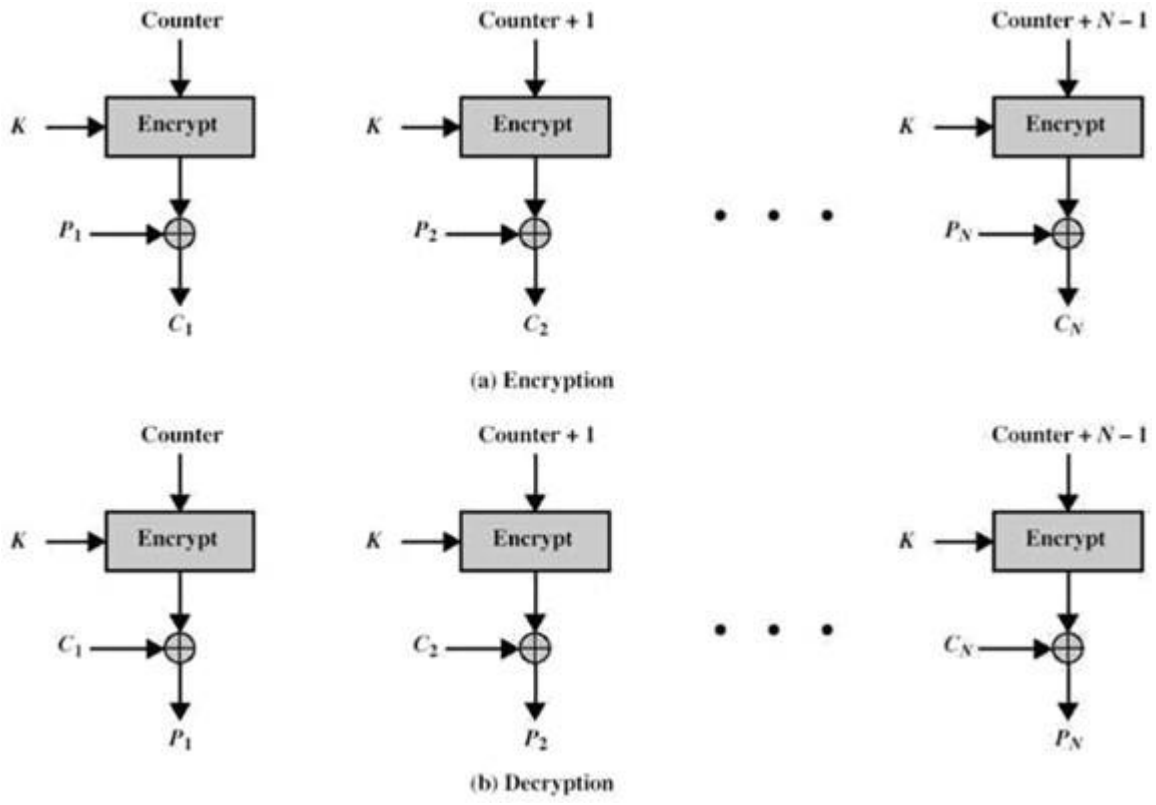


Advantages and Limitations of OFB

- Needs an IV which is unique for each use
- If ever reuse attacker can recover outputs...
- OTP
- Can pre-compute

- Bit errors do not propagate
- More vulnerable to message stream modification...
- Change arbitrary bits by changing ciphertext
- Sender & receiver must remain in sync
- Only use with full block feedback
- Subsequent research has shown that only **full block feedback** (ie CFB-64 or CFB-128) should ever be used

Counter Mode



- A "new" mode, though proposed early on
- Similar to ofb but encrypts counter value rather than any feedback value

$O_i = e_k(i)$

$C_i = p_i \text{xor} o_i$

- Must have a different key &counter value for every plaintext block (never reused)
- Again, otp issue
- Uses: high-speed network encryptions

Advantages and Limitations of CTR

- Efficiency
- Can do parallel encryptions in h/w or s/w
- Can preprocess in advance of need
- Good for burstyhigh speed links
- Random access to encrypted data blocks
- Provable security (good as other modes)
- Never have cycle less than 2^b

But must ensure never reuse key/counter values, otherwise could break.

2.7 STREAMCIPHER

A stream cipher is an encryption algorithm that encrypts 1 bit or byte of plaintext at a time. It uses an infinite stream of pseudorandom bits as the key. For a stream cipher implementation to remain secure, its pseudorandom generator should be unpredictable and the key should never be reused. Stream ciphers are designed to approximate an idealized cipher, known as the One-TimePad.

The One-Time Pad, which is supposed to employ a purely random key, can potentially achieve "perfect secrecy". That is, it's supposed to be fully immune to brute force attacks. The problem with the one-time pad is that, in order to create such a cipher, its key should beas long or even longer than the plaintext. In other words, if you have 500 MegaByte video file that you would like to encrypt, you would need a key that's at least 4 Gigabitslong.

Clearly, while Top Secret information or matters of national security may warrant the use of a one-time pad, such a cipher would just be too impractical for day-to-day public use. The key of a stream cipher is no longer as long as the original message. Hence, it can no longer guarantee "perfect secrecy". However, it can still achieve a strong level ofsecurity.

Comparison between Block Cipher and Stream Cipher

BASIS FOR COMPARISON	BLOCK CIPHER	STREAM CIPHER
Basic	Converts the plain text by taking its block at a time.	Converts the text by taking one byte of the plain text at a time.
Complexity	Simple design	Complex comparatively
No of bits used	64 Bits or more	8 Bits
Confusion and Diffusion	Uses both confusion and diffusion	Relies on confusion only
Algorithm modes used	ECB (Electronic Code Book) CBC (Cipher Block Chaining)	CFB (Cipher Feedback) OFB (Output Feedback)
Reversibility	Reversing encrypted text is hard.	It uses XOR for the encryption which can be easily reversed to the plain text.
Implementation	Feistel Cipher	Vernam Cipher

2.8 RC4

RC4 designed in 1987 by RSA (Ron Rivest, Adi Shamir, and Leonard Adleman). A symmetric key encryption algorithm,followed with Stream Cipher.

In the RC4 encryption algorithm, the key stream is completely independent of the plaintext used. An 8 * 8 S-Box (S0 S255), where each of the entries is a permutation of the numbers 0 to 255, and the permutation is a function of the variable length key. There are two counters i, and j, both initialized to 0 used in the algorithm.

The algorithm uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream which is XORed with the plaintext to give the ciphertext. Each element in the state table is swapped at least once.

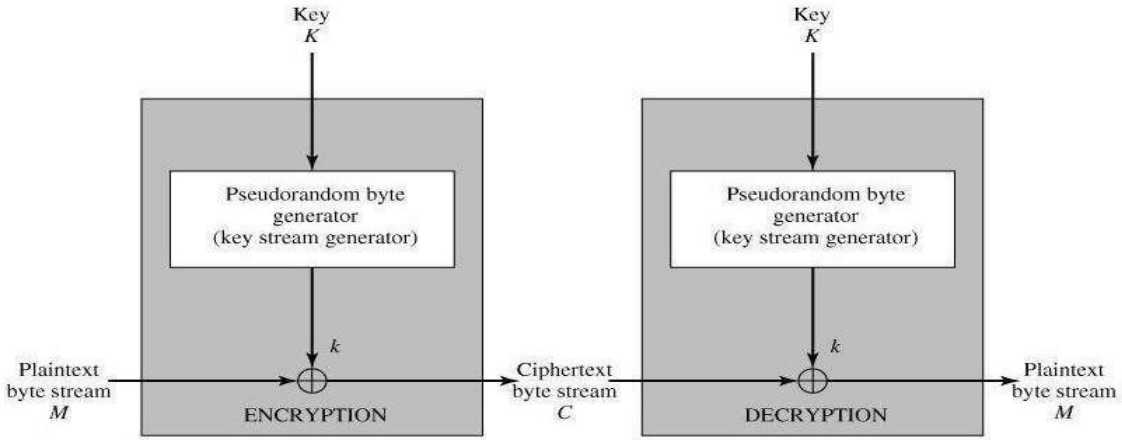
The key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits. RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure SQL.

The algorithm works in two phases, key setup and ciphering. Key setup is the first and most difficult phase of this encryption algorithm. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulas. A modulo operation is the process of yielding a remainder from division. For example, 11/4 is 2 remainder 3; therefore eleven mod four would be equal tothree.

Strengths of RC4

- The difficulty of knowing where any value is in thetable.
- The difficulty of knowing which location in the table is used to select each value in the sequence.
- A particular RC4 Algorithm key can be used onlyonce.
- Encryption is about 10 times faster thanDES.

Architecture of Rc4



Inside of rc4

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

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

KSA

- Use the secret key to initialize and permutation of state vector S, done in two steps

fori = 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)

j = 0;

fori = 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

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

PRGA

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

i, j = 0;

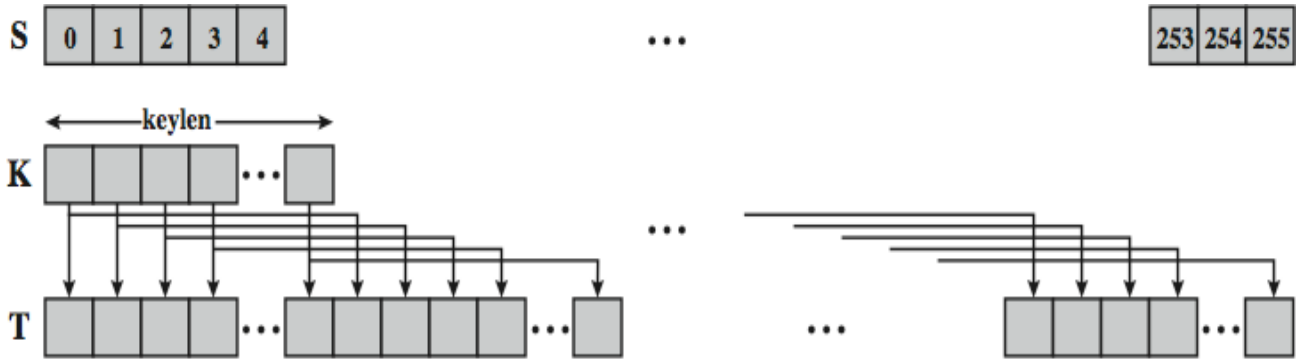
for (int x = 0; x <byteLen; x++) do

i = (i + 1) mod 256;

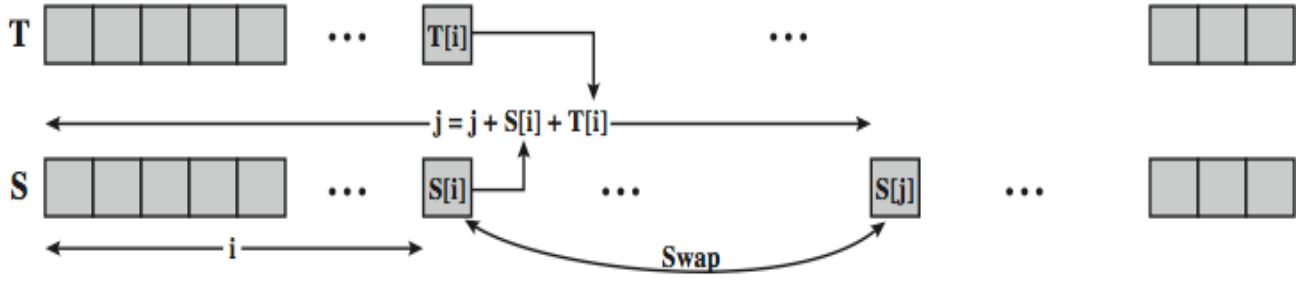
j = (j + S[i]) mod 256; Swap (S[i], S[j]);

t = (S[i] + S[j]) mod 256; k = S[t];

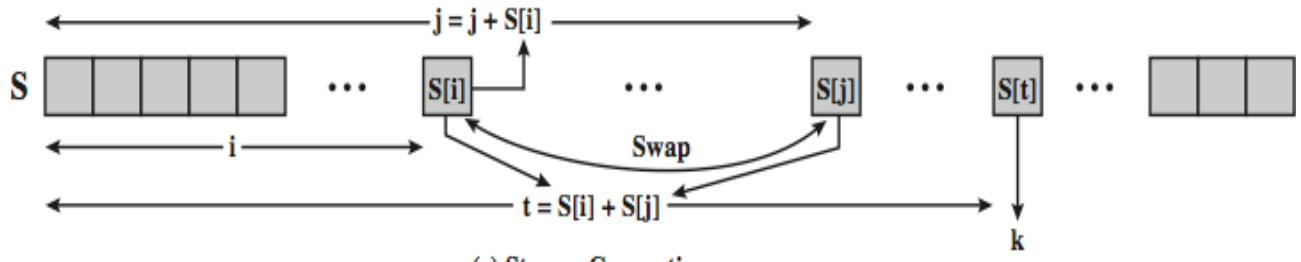
Detailed Diagram



(a) Initial state of S and T

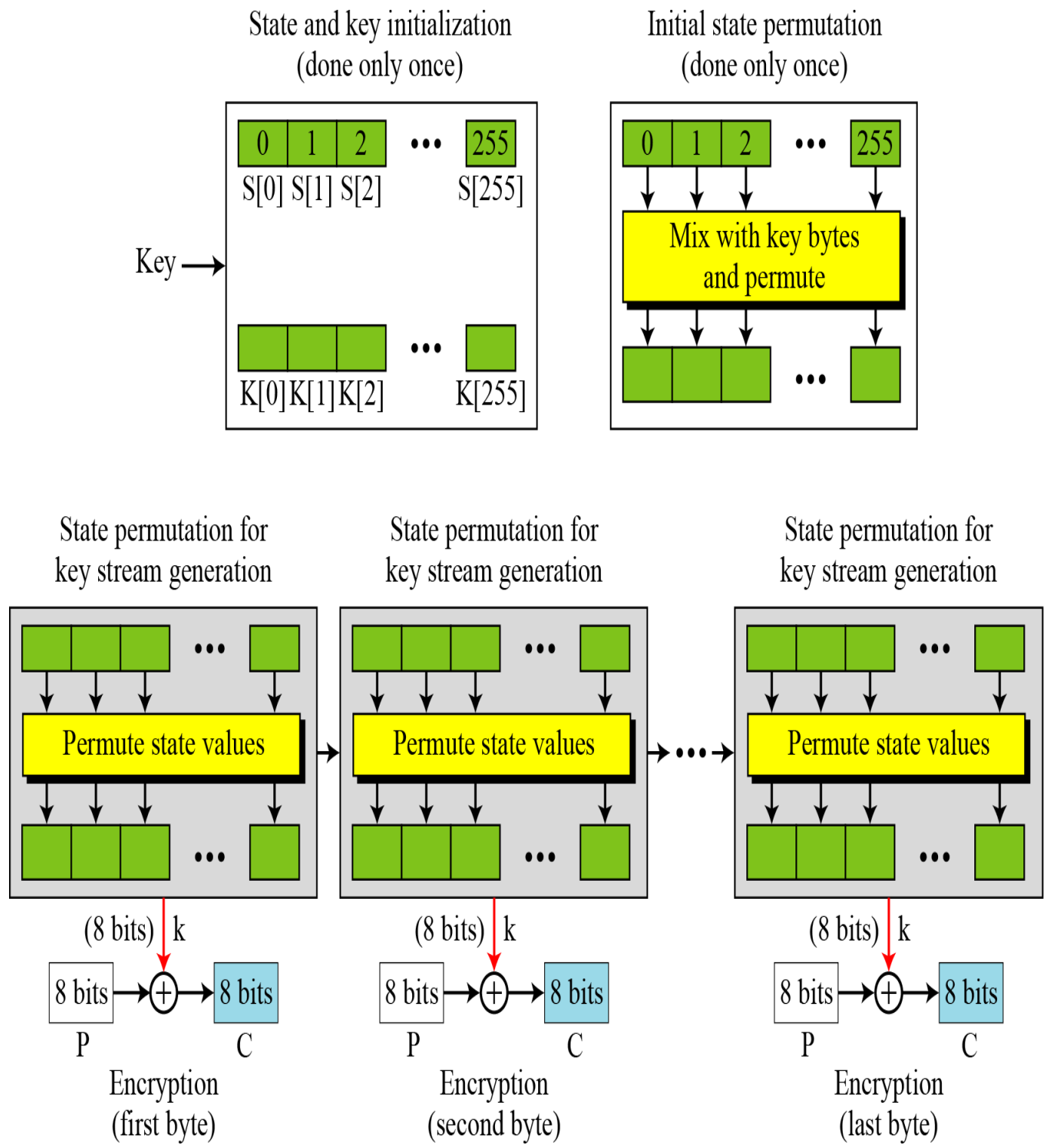


(b) Initial permutation of S



(c) Stream Generation

OverallOperationOfRC4



2.9 RC5

RC5 is a block cipher notable for its simplicity. Designed by Ronald Rivest in1994.

RC stands for "Rivest Cipher", or alternatively, "Ron'sCode

Rivest announced also RC2 and RC4 and now there is RC6 which is The AdvancedEncryption Standard (AES) candidate (RC6 was based onRC5)

Features

- Symmetric block cipher (Like Feistel Network Structure)

  the same secret cryptographic key is used for encryption and decryption

- suitable for hardware andsoftware
- It uses only computational primitive operations commonly found on typical microprocessors
- **Fast**because it uses Word-Oriented operations
- **Adaptable** to processors of different wordlengths

  For example with 64 bit processor RC5 can exploit their longer worklength

Variable length cryptographickey

The user can choose the level of security appropriate for his application the key length *b* in bytes is thus a third parameter of RC5

Simple

It is simple to implement,Thissimplicity makes it more interesting to analyze and evaluate, so that the cryptographic strength can be more rapidlydetermined

Low memory requirements

So it is easily implemented on devices with restricted memory

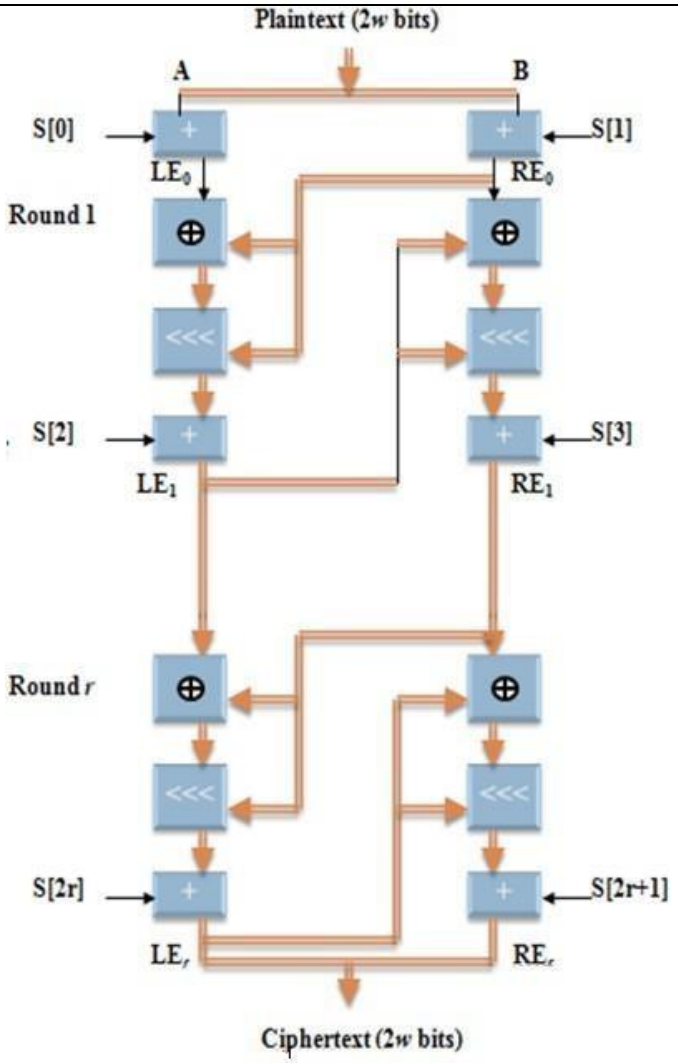
Algorithm RC5

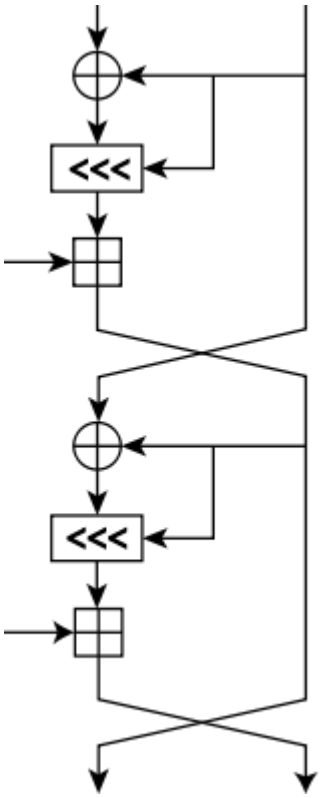
There are three components ofRC5

Key expansionalgorithm

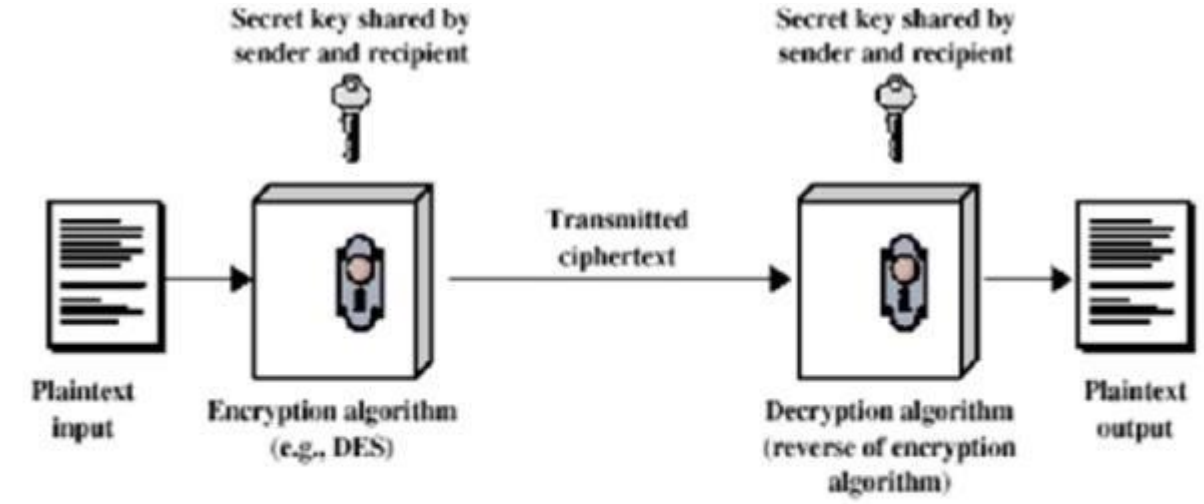
Encryption algorithm

Decryption algorithm





Principles of Public-Key Cryptosystems The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a key distribution center. Whitfield Diffie, one of the discoverers of public-key encryption (along with Martin Hellman, both at Stanford University at the time), reasoned that this second requirement negated the very essence of cryptography: the ability to maintain total secrecy over your own communication. The second problem that Diffie pondered, and one that was apparently unrelated to the first was that of "digital signatures." If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents. Public-Key Cryptosystems Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic: It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption.In addition, some algorithms, such as RSA, also exhibit the following characteristic: Either of the two related keys can be used for encryption, with the other used for decryption. A public-key encryption scheme has six ingredients Plaintext: This is the readable message or data that is fed into the algorithm as input. Encryption algorithm: The encryption algorithm performs various transformations on the plaintext. Public and private keys: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input. Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts. Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.



Simplified Model of Conventional Encryption

The important point is that the security of conventional encryption depends on the secrecy of the key, not the secrecy of the algorithm i.e. it is not necessary to keep the algorithm secret, but only the key is to be kept secret. This feature that algorithm need not be kept secret made it feasible for wide spread use and enabled manufacturers develop low cost chip implementation of data encryption algorithms. With the use of conventional algorithm, the principal security problem is maintaining the secrecy of the key.

2.10 RSA

RSA is the best known, and by far the most widely used general public key encryption algorithm, and was first published by Rivest, Shamir &Adleman of MIT in 1978 [RIVE78]. Since that time RSA has reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption. The RSA scheme is a block cipher in which the plaintext and the ciphertext are integers between 0 and n-1 for some fixed n and typical size for n is 1024 bits (or 309 decimal digits). It is based on exponentiation in a finite (Galois) field over integers modulo a prime, using large integers (eg. 1024 bits). Its security is due to the cost of factoring large numbers. RSA involves a public-key and a private-key where the public key is known to all and is used to encrypt data or message. The data or message which has been encrypted using a public key can only be decryted by

using its corresponding private-key. Each user generates a key pair i.e. public and private key using the following steps:

- *each user selects two large primes at random - p, q*
- *compute their system modulus n=p.q*
- *calculate ø(n), where* ***ø(n)=(p-1)(q-1)***
- *selecting at random the encryption key e, where 1<e<ø(n),and* ***gcd(e,ø(n))=1***
- *solve following equation to find decryption key d:* ***e.d=1 mod ø(n)*** *and 0≤d≤n*
- *publish their public encryption key:* ***KU={e,n}***
- *keep secret private decryption key****: KR={d,n}***

Both the sender and receiver must know the values of n and e, and only the receiver knows the value of d. Encryption and Decryption are done using the following equations. To encrypt a message M the sender:

– obtains **public key** of recipient *KU={e,n}*

– computes: **C=Me mod n**, where 0≤M<n

To decrypt the ciphertext C the owner:

– uses their private key *KR={d,n}*

– computes: **M=Cd mod n = (Me) d mod n = Med mod n**

For this algorithm to be satisfactory, the following requirements are to be met.

a) Its possible to find values of e, d, n such that **Med = M mod n** for all M<n

b) It is relatively easy to calculate Me and C for all values of M < n.

c) It is impossible to determine d given e and n

The way RSA works is based on Number theory: **Fermat's little theorem**: if **p** is prime and **a** is positive integer not divisible by **p**, then **ap-1 ≡ 1 mod p. Corollary**: For any positive integer **a** and prime **p**, **ap ≡ a mod p.**

Fermat's theorem, as useful as will turn out to be does not provide us with integers d,e we are looking for –Euler's theorem (a refinement of Fermat's) does. Euler's function associates to any positive integer **n,** a number φ**(n)**: the number of positive integers smaller than **n** and relatively prime to **n.** For example, φ**(37) = 36** i.e. φ**(p) = p-1** for any prime **p.** For any two primes **p,q**, φ**(pq)=(p-1)(q-1). Euler's theorem**: for any relatively prime integers a,n we have **a**φ**(n)≡1 mod n. Corollary: F**or any integers a,n we have **a**φ**(n)+1≡a mod n Corollary:** Let p,q be two odd primes and n=pq. Then: φ(n)=(p-1)(q-

1) For any integer m with 0<m<n, m(p-1)(q-1)+1 ≡ m mod n For any integers k,m with0<m<n, mk(p-1)(q-1)+1 ≡ m mod n Euler's theorem provides us the numbers d, e such that Med=M mod n. We have to choose d,e such that ed=kφ(n)+1, or equivalently, d≡e-1mod φ(n)

   An example of RSA can be given as,

   Select primes: p=17 &q=11

   Compute $n = pq$ =17×11=187

   Compute ø(n)=(p–1)(q-1)=16×10=160

   Select e :gcd(e,160)=1; choose e=7

   Determine d*: de*=1 mod 160 and d< 160 Value is d=23 since 23×7=161= 10×160+1

   Publish public key KU={7,187}

   Keep secret private key KR={23,187}

   Now, given message M = 88 (nb. 88<187)

   encryption: C = 887 mod 187 = 11

   decryption: M = 1123 mod 187 = 88

Another example of RSA is given as,

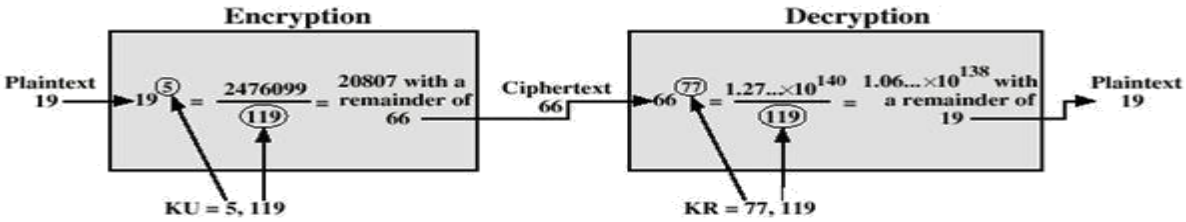
Let **p = 11, q = 13, e = 11, m = 7**

n = pqi.e. n= 11*13 = 143

ø(n)= (p-1)(q-1) i.e. (11-1)(13-1) = 120

e.d=1 mod ø(n) i.e. 11d mod 120 = 1 i.e. (11*11) mod 120=1;so d = 11 public key :{11,143} and private key: {11,143}

C=Me mod n, so ciphertext = 711mod143 = 727833 mod 143; i.e. C = 106 **M=Cd mod n,** plaintext = 10611 mod 143 = 1008 mod 143; i.e. M = 7



For RSA key generation,

- users of RSA must:
  - determine two primes at random - p, q
  - select either e or d and compute the other
  - typically guess and use probabilistic test
- exponents e, d are inverses, so use Inverse algorithm to compute the other

Security of RSA

There are three main approaches of attacking RSA algorithm.

Brute force key search (infeasible given size of numbers) As explained before, involvestrying all possible private keys. Best defence is using large keys.

Mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N)There are several approaches, all equivalent in effect to factoring the product of two primes. Some of them are given as:

– factor N=p.q, hence find ø(N) and then d

– determine ø(N) directly and find d

– find d directly

The possible defense would be using large keys and also choosing large numbers for p and q, which should differ only by a few bits and are also on the order of magnitude 1075 to 10100. And gcd (p-1, q-1) should be small.

2.11 THE ELGAMAL PUBLIC KEY ENCRYPTION ALGORITHM

The ElGamal Algorithm provides an alternative to the RSA for public key encryption. 1) Security of the RSA depends on the (presumed) difficulty of factoring large integers. 2) Security of the ElGamal algorithm depends on the (presumed) difficulty of computing discrete logs in a large prime modulus. ElGamal has the disadvantage that the ciphertext is twice as long as the plaintext. It has the advantage the same plaintext gives a different ciphertext (with near certainty) each time it is encrypted. Alice chooses i) A large prime pA (say 200 to 300 digits), ii) A primitive element α A modulo pA, iii) A (possibly random) integer dA with $2 \leq dA \leq pA - 2$. Alice computes iv) $\beta A \equiv \alpha A\, dA \pmod{pA}$. Alice's public key is (pA, α A, β A).

Algorithm: ELGAMAL ENCRYPTION

INPUT: Domain parameters (p, q, g); recipient's public key B; encoded message m in range $0 < m < p - 1$.

OUTPUT:Ciphertext (c_1, c_2).

Choose a random k in the range $1 < k < p - 1$.

1. Compute $c_1 = g^k \bmod p$
2. Compute $c_2 = mB^k \bmod p$
3. Return ciphertext (c_1, c_2).

Algorithm: ELGAMAL DECRYPTION

INPUT: Domain parameters (p, q, g); recipient's private key b; ciphertext (c_1, c_2).

OUTPUT:Messagerepresentative, m.

Compute $m = c_1^{p-b-1} c_2 \bmod p$

1. Return m.

ELGAMAL ENCRYPTION: EXAMPLE

To encrypt M = 10 using Public key 9

1 - Generate a random number k = **3**

2 - Compute $C_1 = 11^3 \bmod 23 = 20$

$C_2 = 10 \times 9^3 \bmod 23$

$= 10 \times 16 = 160 \bmod 23 = 22$

3 - Ciphertext C = (20 , 22)

ELGAMAL EXAMPLE

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message $m=17$ as $(11,5)$ by
  - chosing random $k=6$
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11$;
    $C_2 = KM \bmod q = 7.17 \bmod 19 = 5$
- Alice recovers original message by computing:
  - recover $K = C_1^{xA} \bmod q = 11^5 \bmod 19 = 7$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2\ K^{-1} \bmod q = 5.11 \bmod 19 = 17$

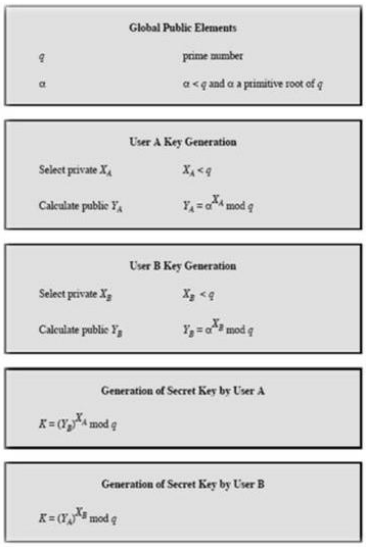
2.12 DIFFIE-HELLMAN KEY EXCHANGE

Diffie-Hellman key exchange (**D-H**) is a cryptographic protocol that allows two partiesthat have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel.

This key can then be used to encrypt subsequent communications using a symmetric key cipher. The D-H algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

First, a primitive root of a prime number p, can be defined as one whose powers generate all the integers from 1 to p-1. If a is a primitive root of the prime number p, then the numbers, ***a mod p, a2mod p,..., ap-1mod p***, are distinct and consist of the integers from 1 through p 1 in some permutation.

For any integer ***b*** and a primitive root ***a*** of prime number ***p***, we can find a unique exponent**i**such that $b \equiv a^i \pmod p$ where $0 \leq i \leq (p\ 1)$.The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as ***dloga,p(b).*** The algorithm is summarized below:



For this scheme, there are two publicly known numbers: a prime number q and an integerthat is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random integer XA < q and computes YA = αXA mod q. Similarly, user B independently selects a random integer XA < q and computes YB = αXB mod q. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as K = (YB)XA mod q and user B computes the key as K = (YA)XB mod q. These two calculations produce identical results.

Discrete Log Problem

The (discrete) exponentiation problem is as follows: Given a base a, an exponent b and a modulus p, calculate c such that $ab \equiv c \pmod{p}$ and $0 \leq c < p$. It turns out that this problem is fairly easy and can be calculated "quickly" using fast-exponentiation. The discrete log problem is the inverse problem: Given a base a, a result c ($0 \leq c < p$) and a modulus p,calculate the exponent b such that $ab \equiv c \pmod{p}$. It turns out that no one has found a quick way to solve this problem With DLP, if P had 300 digits, Xa and Xb have more than 100 digits, it would take longer than the life of the universe to crack the method.

Man-in-the-Middle Attack on D-H protocol

Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

1. Darth prepares for the attack by generating two random private keys XD1 and XD2 and then computing the corresponding public keys YD1 and YD2.

2. Alice transmits YA to Bob.

3. Darth intercepts YA and transmits YD1 to Bob. Darth also calculates K2 = (YA)XD2mod q.

1. Bob receives YD1 and calculates K1 = (YD1)XE mod q.
2. Bob transmits XA to Alice.
3. Darth intercepts XA and transmits YD2 to Alice. Darth calculates K1 = (YB)XD1 mod q.
4. Alice receives YD2 and calculates K2 = (YD2)XA mod q.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message M: E(K2, M).
2. Darth intercepts the encrypted message and decrypts it, to recover M.

3. Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob