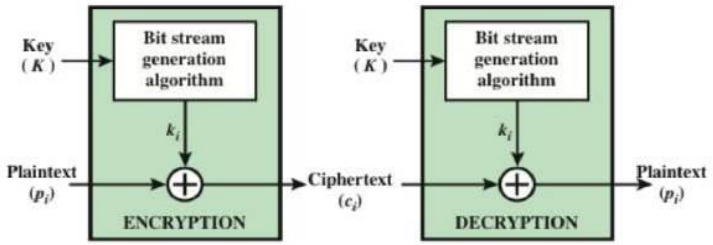
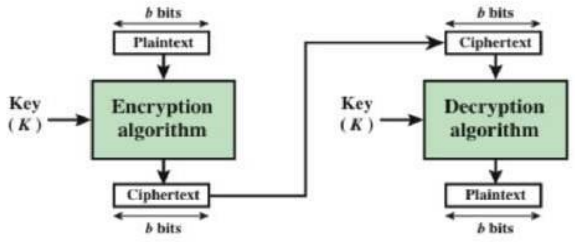
UNIT II - BLOCK CIPHERS & PUBLIC KEY CRYPTOGRAPHY

Data Encryption Standard-Block cipher principles-block cipher modes of operation-Advanced Encryption Standard (AES)Triple DES-Blowfish-RC5 algorithm. **Public key cryptography:** Principles of public key cryptosystems-The RSA Algorithm-Key management - Diffie Hellman Key exchange -Elliptic curve arithmetic--Elliptic curve cryptography.

Stream Cipher: Encrypts a digital data stream one bit or one byte at a time
Examples: •Auto keyed Vigenère cipher
•Vernam cipher



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Figure 3.1 Stream Cipher and Block 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 usersIt must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream The two users need only share the generating key and each can produce the key stream.

Block Cipher

A block of plaintext is treated as a whole and used to produce a cipher text block of equal length. As with a stream cipher, the two users share a symmetric encryption key. Typically a block size of 64 or 128 bits is used .The majority of network-based symmetric cryptographic applications make use of block ciphers.

A block cipher operates on a plaintext block of *n bits to produce a cipher text* block of *n bits.*

- *There are* 2^n *possible different plaintext blocks and, for* the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique cipher text block. Such a transformation is called reversible, or non-singular.
- In the latter case, a cipher text of 01 could have been produced by one of two plaintext blocks either 10 or 11.

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

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

Encryption and Decryption Tables for Substitution Cipher

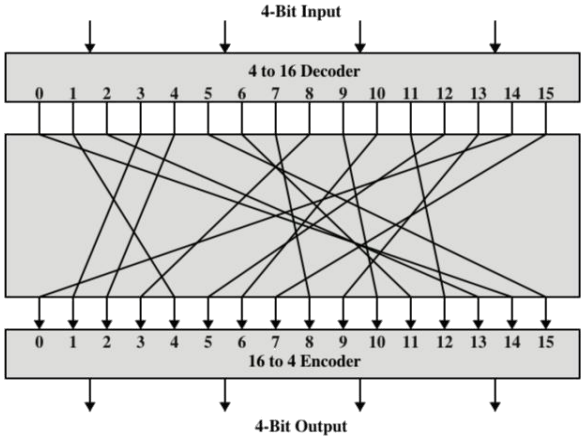


Figure 3.2 General *n*-bit-*n*-bit Block Substitution (shown with $n = 4$)

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

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

Feistel Cipher (Ideal Block Cipher)

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 cipher text 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

Diffusion and Confusion

Terms introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system

Diffusion

The statistical structure of the plaintext is dissipated into long-range statistics of the cipher text. This is achieved by having each plaintext digit affect the value of many cipher text digits

Confusion

Seeks to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible. Even if the attacker can get some handle on the statistics of the cipher text, the way in which the key was used to produce that cipher text is so complex as to make it difficult to deduce the key.

Example for Diffusion

An example of diffusion is to encrypt a message M = m1, m2, m3, . . . of characters with an averaging operation:

$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

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

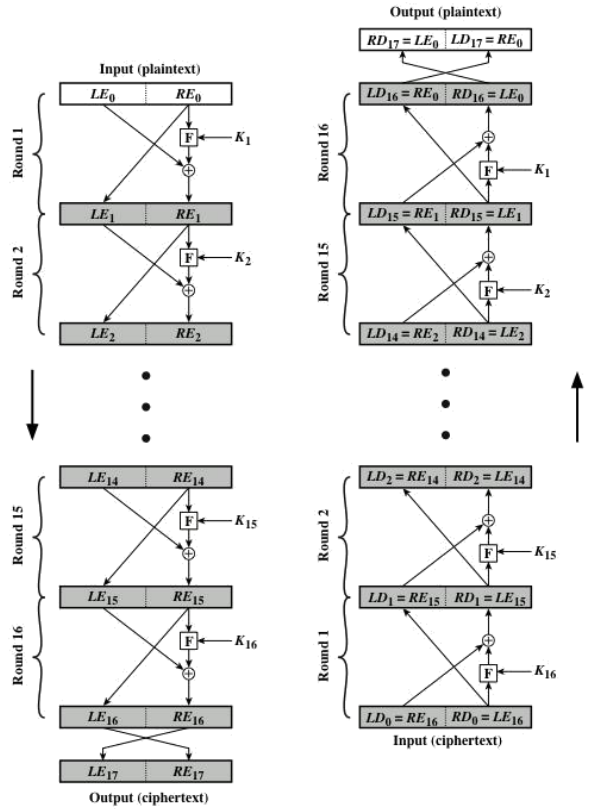


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

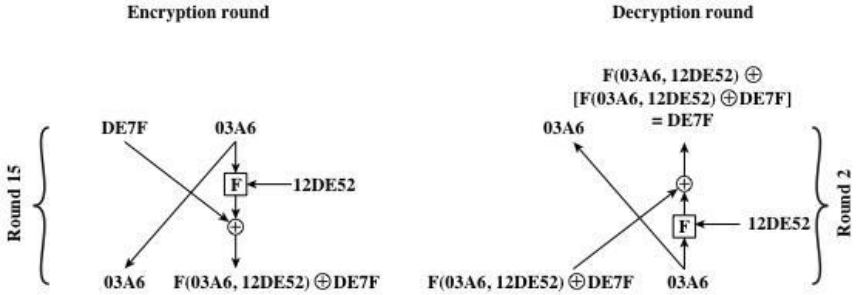


Figure 3.4 Feistel Example

Data Encryption Standard (DES)

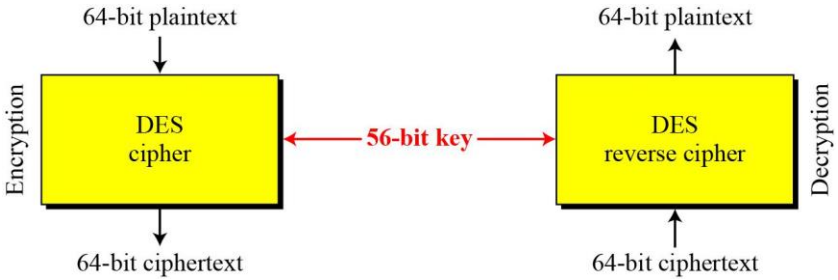
Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46.Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001.Algorithm itself is referred to as the Data Encryption Algorithm (DEA)

–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

Encryption and decryption with DES



DES Encryption

There are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length

General Depiction of DES Encryption Algorithm

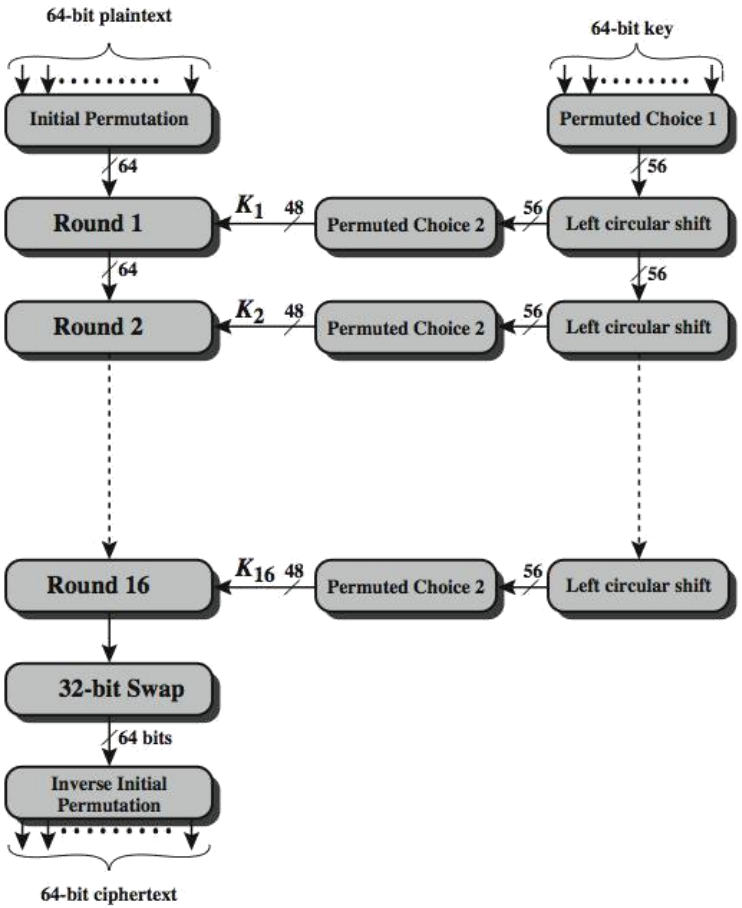
Processing of the plaintext proceeds **in three phases**:

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.

- This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The output of the sixteenth round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**.

- Finally, the pre output is passed through a permutation (IP-1) that is the inverse of the initial permutation function, to produce the 64-bit cipher text.

Encryption



Initial Permutation

The initial permutation and its inverse are defined by tables. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered from 1 to 64. The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits

Permutation Tables for DES

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

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

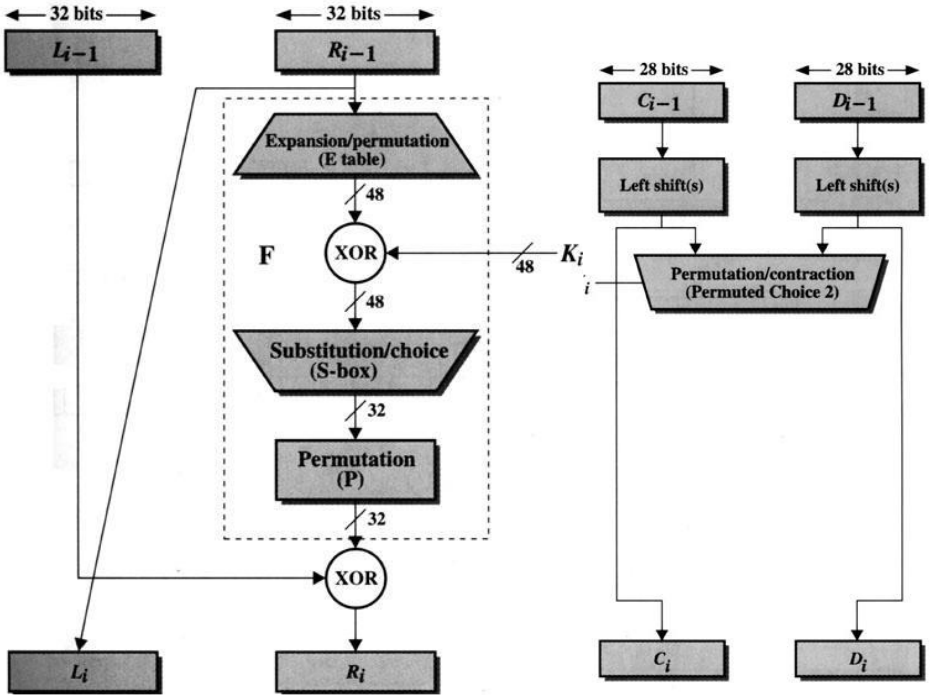
Inverse Initial Permutation (IP–1)

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

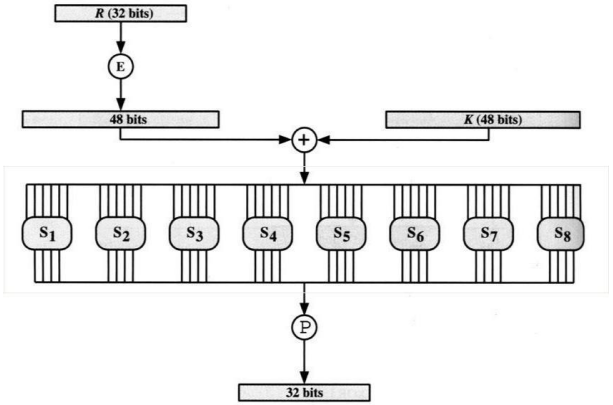
Permutation Function (P)

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

Details of Single Round



Encryption (Round) (8 S-Boxes) Each S box get input as 6 puts and output to 4 bits, 8*6=48 into 8*4=32



S-box

The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example, in S1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

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

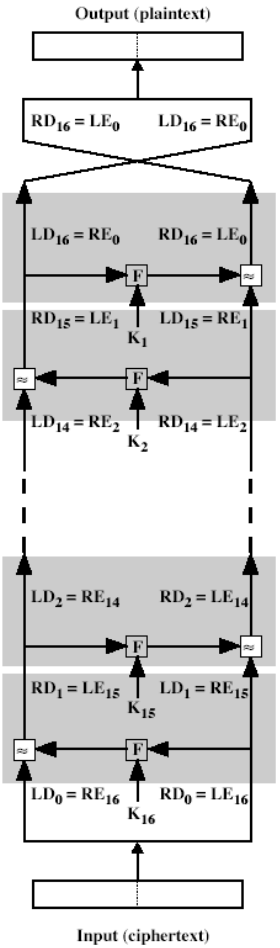
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

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



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

Key Generation

Key Generation a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored. The key is first subjected to a permutation governed by a table labeled Permuted Choice One. The resulting 56-bit key is then treated as two 28-bit quantities, labeled C0 and D0. At each round, Ci-1 and Di-1 are separately subjected to a circular left shift, or rotation, of 1 or 2 bits. These shifted values serve as input to the next round.They also serve as input to Permuted Choice Two , which produces a 48-bit output that serves as input to the function F(Ri-1, Ki).

DES Decryption

The same algorithm as encryption. Reversed the order of key (Key16, Key15, … Key1).For example:
–IP undoes IP-1 step of encryption. 1st round with SK16 undoes 16th encrypt round.

Strength of DES – Key Sie

• 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values • brute force search looks hard.recent advances have shown is possible – in 1997 on Internet in a few months

– in 1998 on dedicated h/w (EFF) in a few days – in 1999 above combined in 22hrs!

• still must be able to recognize plaintext • now considering alternatives to DES

Avalanche effect in DES

If a small change in either the plaintext or the key, the cipher text should change markedly. DES exhibits a strong avalanche effect.

Strength of DES – Analytic Attacks

DES now have several analytic attacks on DES, these utilise some deep structure of the cipher – by gathering information about encryptions can eventually recover some/all of the sub-key bits – if necessary then exhaustively search for the rest generally these are statistical attacks include

– differential cryptanalysis
– linear cryptanalysis
– related key attacks

Differential Cryptanalysis one of the most significant recent (public) advances in cryptanalysis known by NSA in 70's cf DES design. Murphy, Biham & Shamir published 1990 powerful method to analyze block ciphers used to

analyse most current block ciphers with varying degrees of success DES reasonably resistant to it. Differential Crypt analysis a statistical attack against Feistel ciphers uses cipher structure not previously used design of S-P networks has output of function f influenced by both input & key hence cannot trace values back through cipher without knowing values of the key Differential Cryptanalysis compares two related pairs of encryptions Differential Cryptanalysis Compares Pairs of Encryptions with a known difference in the input searching for a known difference in output when same sub keys are used

Differential Cryptanalysis have some input difference giving some output difference with probability p. if find instances of some higher probability input / output difference pairs occurring can infer sub key that was used in round then must iterate process over many rounds (with decreasing probabilities).Differential Cryptanalysis perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR when found if intermediate rounds match required XOR have a right pair – if not then have a wrong pair, relative ratio is S/N for attack can then deduce keys values for the rounds – right pairs suggest same key bits wrong pairs give random values for large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs Biham and Shamir have shown how a 13-round iterated characteristic can break the full 16-round DES

Linear Cryptanalysis

Linear Cryptanalysis another recent development also a statistical method must be iterated over rounds, with decreasing probabilities developed by Matsui et al in early 90's based on finding linear approximations can attack DES with 247 known plaintexts, still in practice infeasible. Linear Cryptanalysis find linear approximations with prob p != ½ P[i1,i2,...,ia](+)C[j1,j2,...,jb] =K[k1,k2,...,kc]where ia,jb,kc are bit locations in P,C,K gives linear equation for key bits get one key bit using max likelihood alg. Using a large number of trial encryption effectiveness given by: |p–½|

1. **Explain about the single round of DES algorithm and the key discarding process of DES.(16) (APR/MAY 2011)**
2. **Describe the working principle of simple DES with an example. (16) (MAY/JUNE 2014), (APR/MAY 2015), (MAY/JUNE 2013) (NOV/DEC 2012)**
3. **Explain the key generation, encryption and decryption of SDES algorithm.(16) (NOV/DEC 2011), (NOV/DEC 2014)**

Block Cipher modes of Operation

1. **Explain the Block cipher modes of operation. (16) (APR/MAY 2010,NOV/DEC 2013, APR/MAY 2011)**
2. **What is the disadvantage with ECB mode of operation? (2) (MAY/JUNE 2013)**

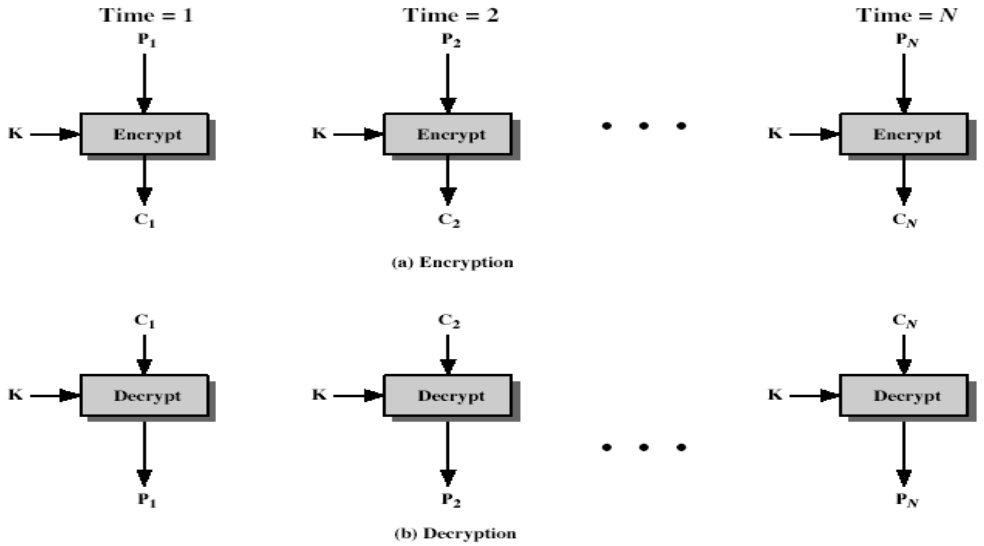
Various Modes

1. Electronic Codebook Mode
2. Cipher Block Chaining Mode
3. Cipher Feedback Mode
4. Output Feedback Mode
5. Counter Mode

Electronic Codebook Book (ECB)

Electronic Codebook Book (ECB) message is broken into independent blocks which are encrypted. Each block is a value which is substituted, like a codebook.each block is encoded independently of the other blocks

Ci = DESK (Pi) uses: secure transmission of single values



Electronic Codebook Book (ECB)

Advantages and Limitations of ECB

• repetitions in message 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 due to encrypted message blocks being independent

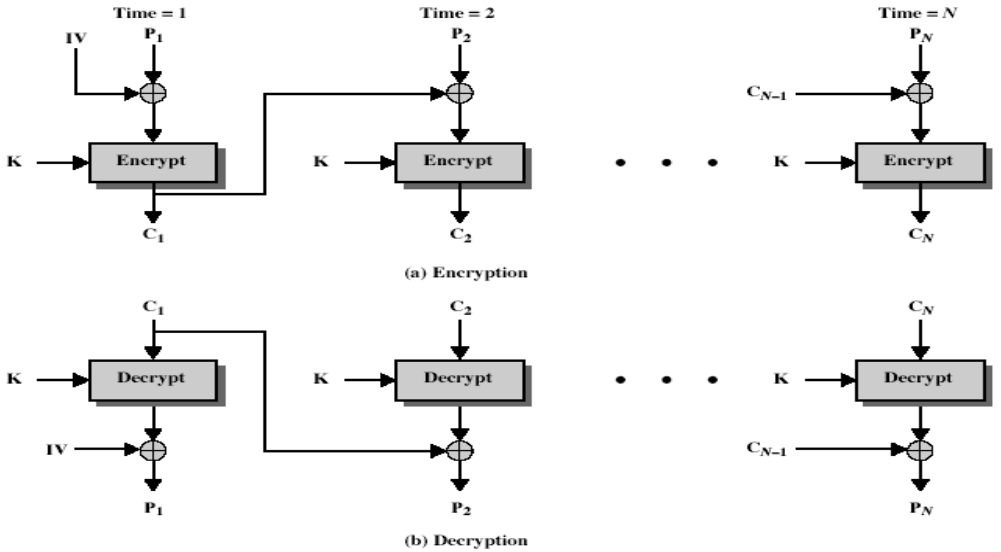
• main use is sending a few blocks of data

Cipher Block Chaining

(CBC) message is broken into blocks but these are linked together in the encryption operation. Each previous cipher blocks is chained with current plaintext block. It use Initial Vector (IV) to start process

Ci = DESK1(Pi XOR Ci-1) C-1 = IV

•uses: bulk data encryption, authentication



Cipher Block Chaining (CBC)

Advantages and Limitations of CBC

• each ciphertext block depends on all message blocks

• thus a change in the message affects all ciphertext blocks after the change as well as the original block need Initial Value (IV) known to sender & receiver however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate.hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message at end of message, handle possible last short block – by padding either with known non-data value (eg nulls) or pad last block with count of pad size
• eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count

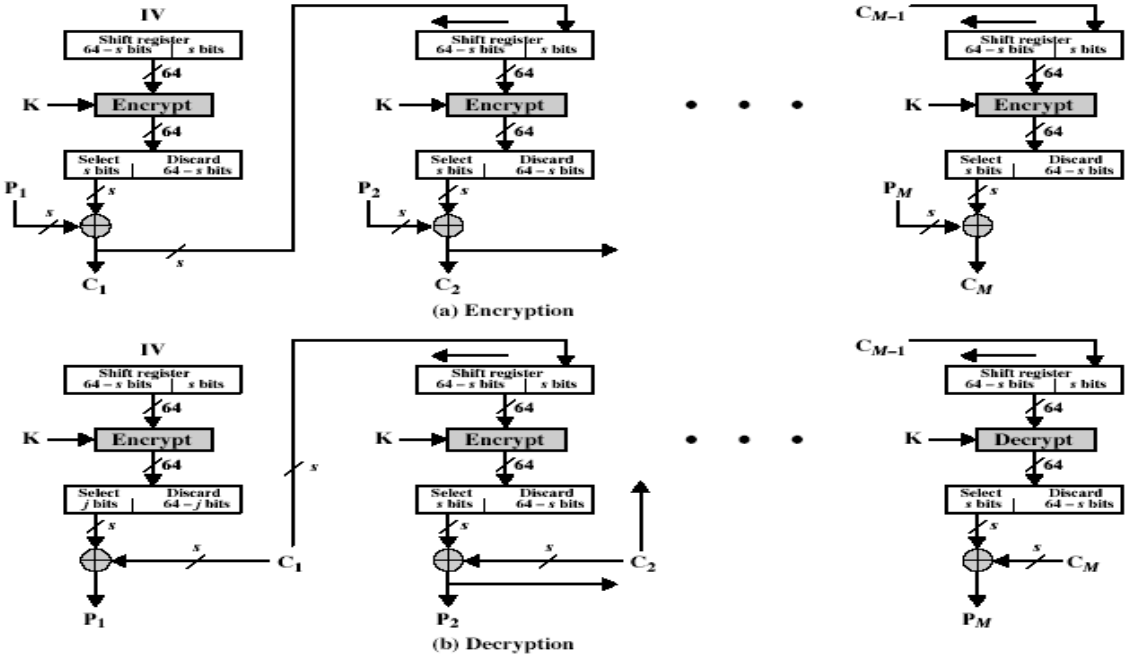
Cipher FeedBack (CFB)

Cipher FeedBack (CFB) message is treated as a stream of bits added to the output of the block cipher result is feed back for next stage (hence name).Standard allows any number of bit (1,8 or 64 or whatever) to be feed back denoted CFB-1, CFB-8, CFB-64 etc is most efficient to use all 64 bits (CFB-64)

Ci = Pi XOR DESK1(Ci-1) C-1 = IV

• uses: stream data encryption, authentication

Cipher FeedBack (CFB)



Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in encryption mode at both ends
- errors propogate for several blocks after the error

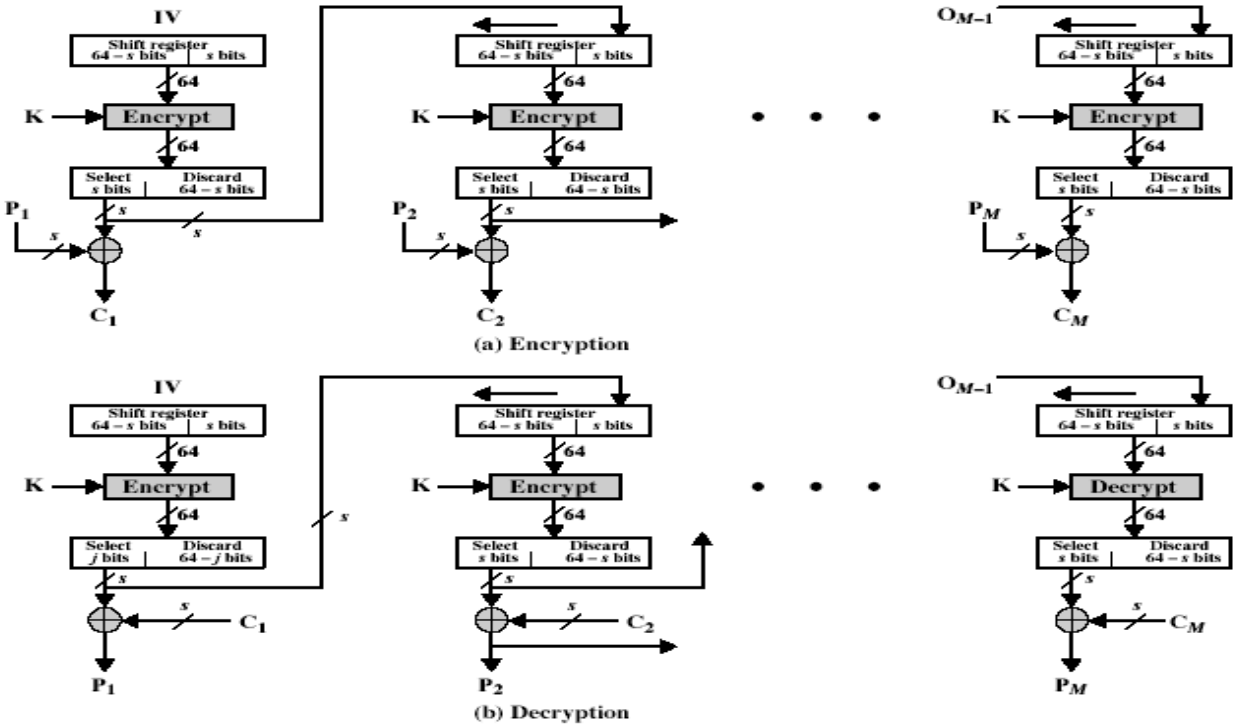
Output FeedBack (OFB)

Output FeedBack (OFB) message is treated as a stream of bits output of cipher is added to message output is then feed back (hence name) feedback is independent of message can be computed in advance

Ci = Pi XOR Oi

Oi = DESK1(Oi-1) O-1 = IV uses: stream encryption over noisy channels

Output FeedBack (OFB)



Advantages and Limitations of OFB

- used when error feedback a problem or where need to encryptions before message is available
- superficially similar to CFB
- but feedback is from the output of cipher and is independent of message
- a variation of a Vernam cipher

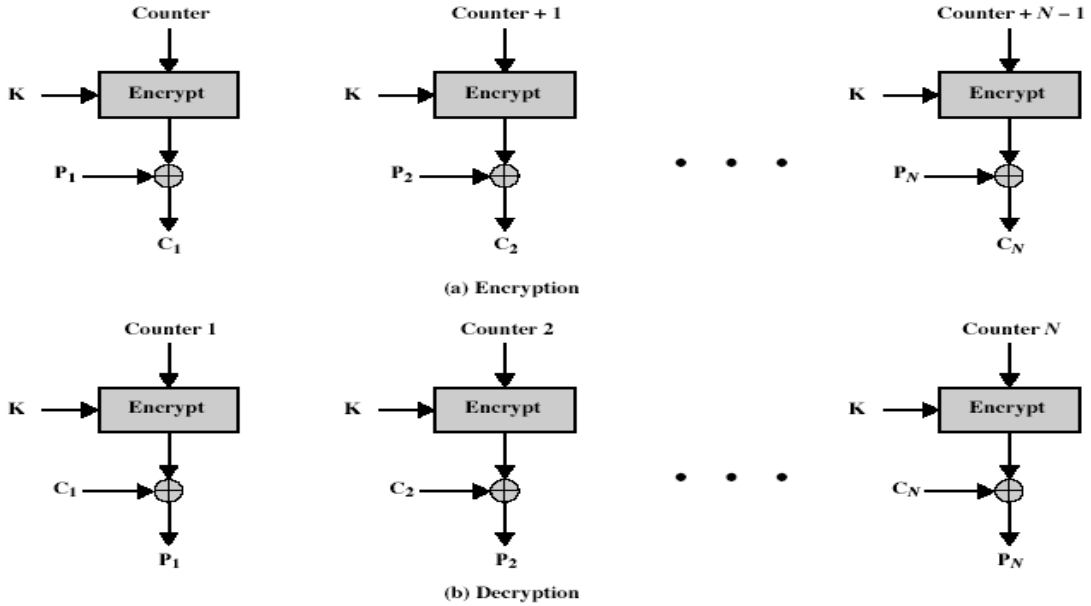
– hence must never reuse the same sequence (key+IV)

- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- originally specified with m-bit feedback in the standards
- subsequent research has shown that only OFB-64 should ever be used

Counter (CTR) a "new" mode, though proposed early on similar to OFB but encrypts counter value rather than any feedback value must have a different key & counter value for every plaintext block (never reused)

Ci = Pi XOR Oi Oi = DESK1(i) uses: high-speed network encryptions

Counter (CTR)



Advantages and Limitations of CTR

• efficiency

– can do parallel encryptions – in advance of need

– good for bursty high speed links

• random access to encrypted data blocks • provable security (good as other modes)• but must ensure never reuse ey/counter values, otherwise could break (cf OFB)

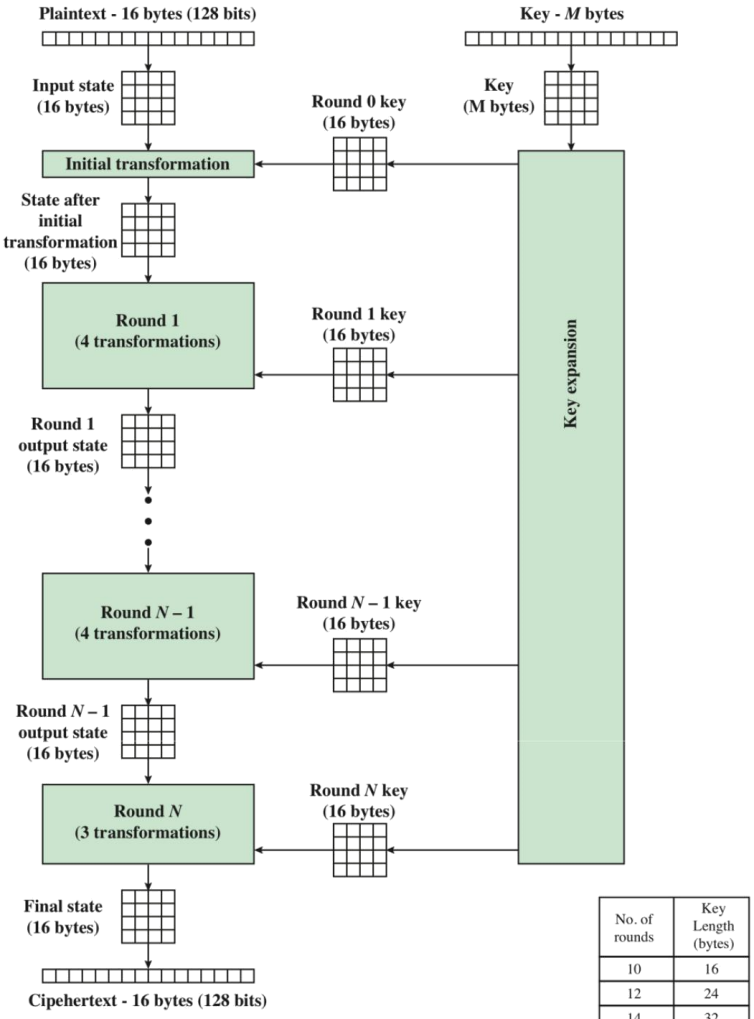
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Advanced Encryption Standard

General structure

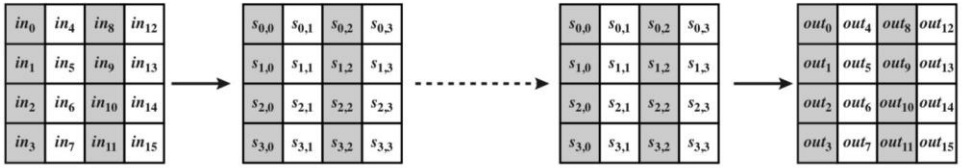
The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.The input to the encryption and decryption algorithms is a single 128-bit block.In FIPS PUB 197, this block is depicted as a 4 * 4 square matrix of bytes. This block is copied into the **State** array, which is modified at each stage of encryption or decryption. After the final stage, **State** is copied to an output matrix. Similarly, the key is depicted as a square matrix of bytes.This key is then expanded into an array of key schedule words. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key.

The cipher consists of *N* rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key (Table 5.1). The first *N* - 1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are described subsequently. The final round contains only three transformations, and there is a initial single transformation (AddRoundKey) before the first round, which can be considered Round 0. Each transformation takes one or more 4 * 4 matrices



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

Figure 5.1 AES Encryption Process



(a) Input, state array, and output

(b) Key and expanded key

AES Parameters

Figure 5.2 AES Data Structures

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

Detailed Structure

Processes the entire data block as a single matrix during each round using substitutions and permutation. The key that is provided as input is expanded into an array of forty-four 32-bit words, *w[i]*

Four different stages are used:

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

- The cipher begins and ends with an AddRoundKey stage.
- Can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so
- Each stage is easily reversible.
- The decryption algorithm makes use of the expanded key in reverse order, however the decryption algorithm is not identical to the encryption algorithm.
- State is the same for both encryption and decryption
- Final round of both encryption and decryption consists of only three stages

AES Encryption

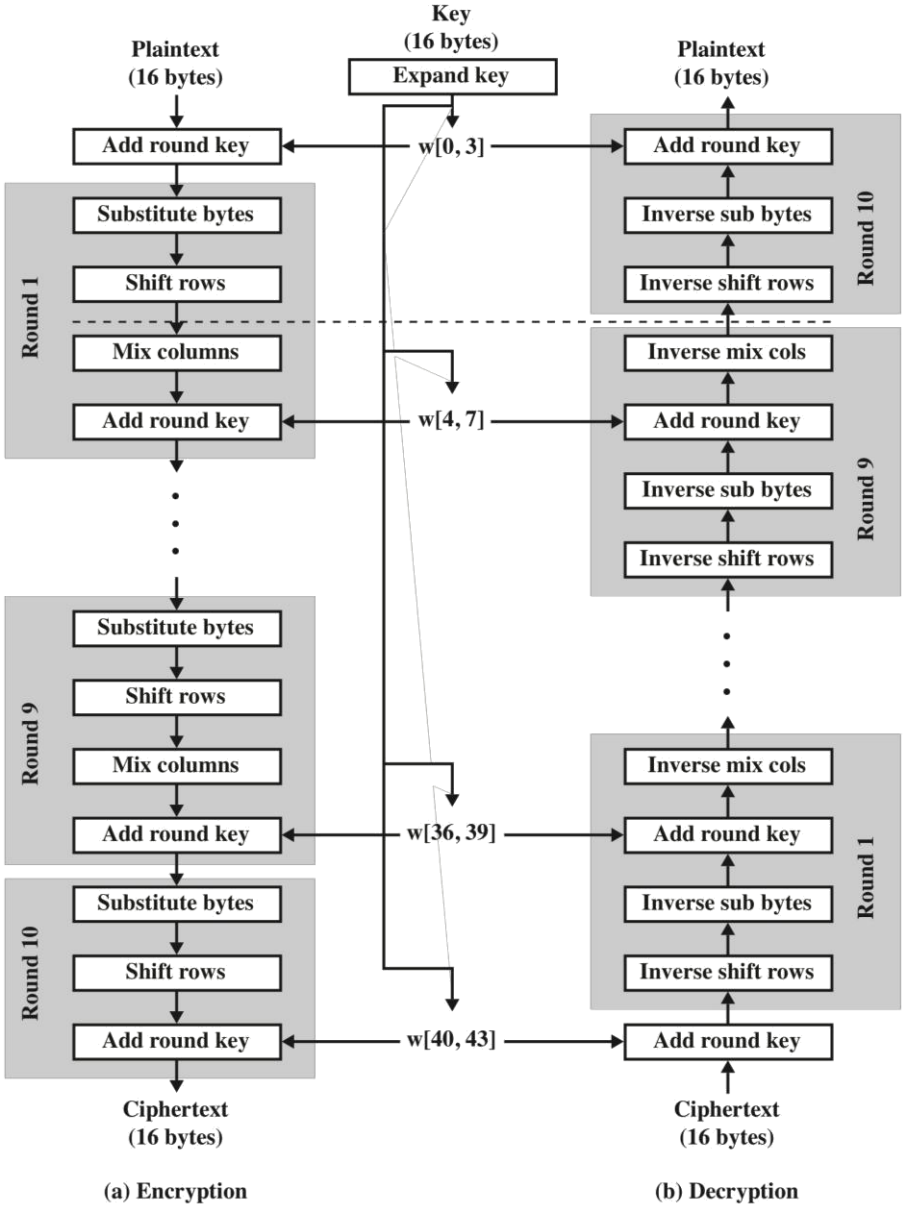


Figure 5.3 AES Encryption and Decryption

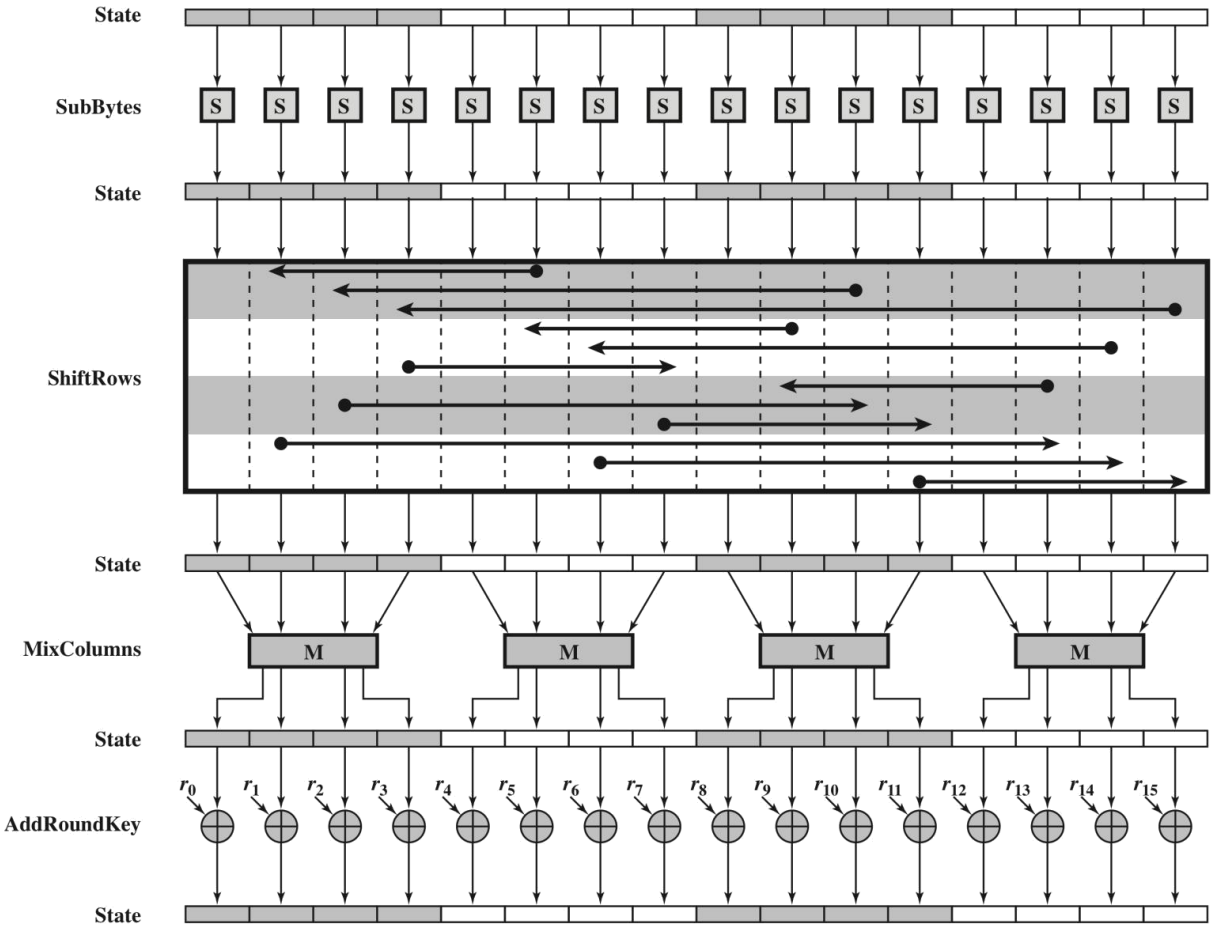


Figure 5.4 AES Encryption Round

Byte Substitution

A simple substitution of each byte uses one table of 16x16 bytes containing a permutation of all 256 8-bit values.

Each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits) – eg. byte {95} is replaced by row col 5 byte – which is the value {2A}. S-box is constructed using a defined transformation of the values in GF(2^8) designed to be resistant to all known attacks

(a) S-box

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

(b) Inverse S-box

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

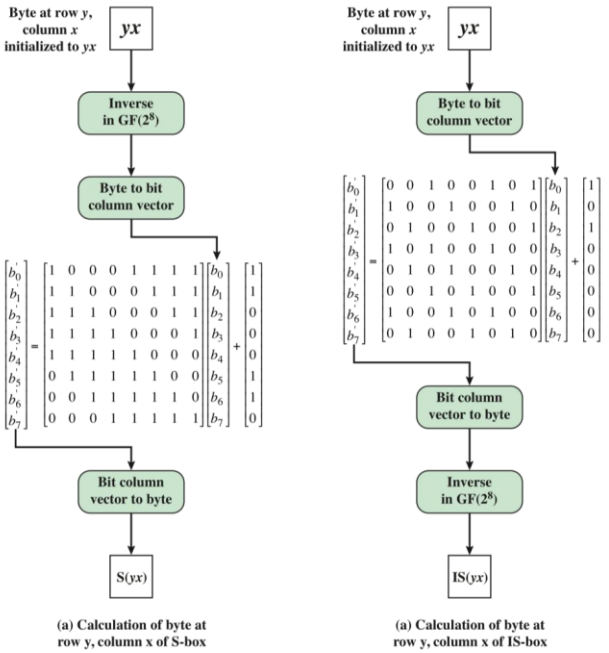


Figure 5.6 Construction of S-Box and IS-Box

-Box Rationale

The S-box is designed to be resistant to known cryptanalytic attacks. The Rijndael developers sought a design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input

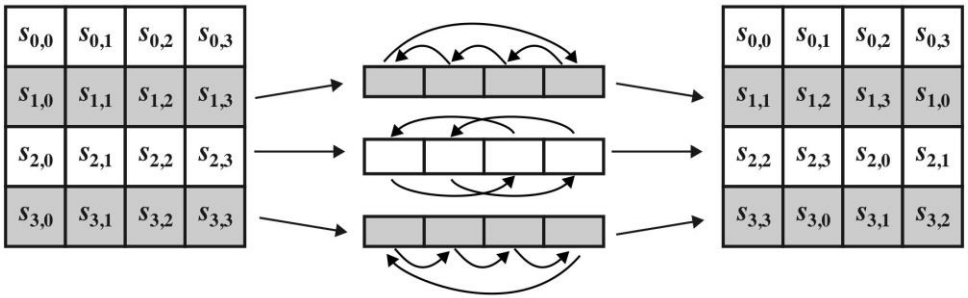
- The nonlinearity is due to the use of the multiplicative inverse

Shift Row Transformation a circular byte shift in each each

- – 1st row is unchanged
  - 2nd row does 1 byte circular shift to left –
  - 3rd row does 2 byte circular shift to left
  - – 4th row does 3 byte circular shift to left

decrypt does shifts to right ,since state is processed by columns, this step permutes bytes between the columns

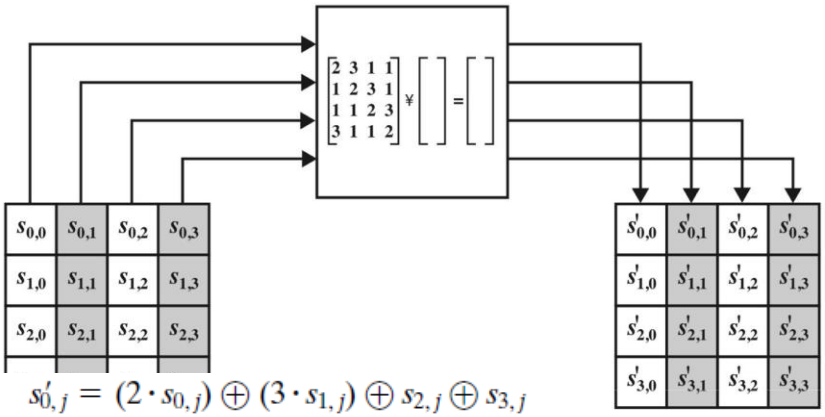
Shift Row Transformation



(a) Shift row transformation

Mix Column Transformation

The forward mix column transformation, called MixColumns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column. The transformation can be defined by the following matrix multiplication on State Each element in the product matrix is the sum of products of elements of one row and one column. In this case, the individual additions and multiplications are performed in GF(2^8).



The MixColumns transform[illegible]essed as

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

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

Add RoundKey Transformation

Add Round Key is XOR state with 128-bits of the round key

- again processed by column
- inverse for decryption is identical

since XOR is own inverse, just with correct round key

- designed to be as simple as possible

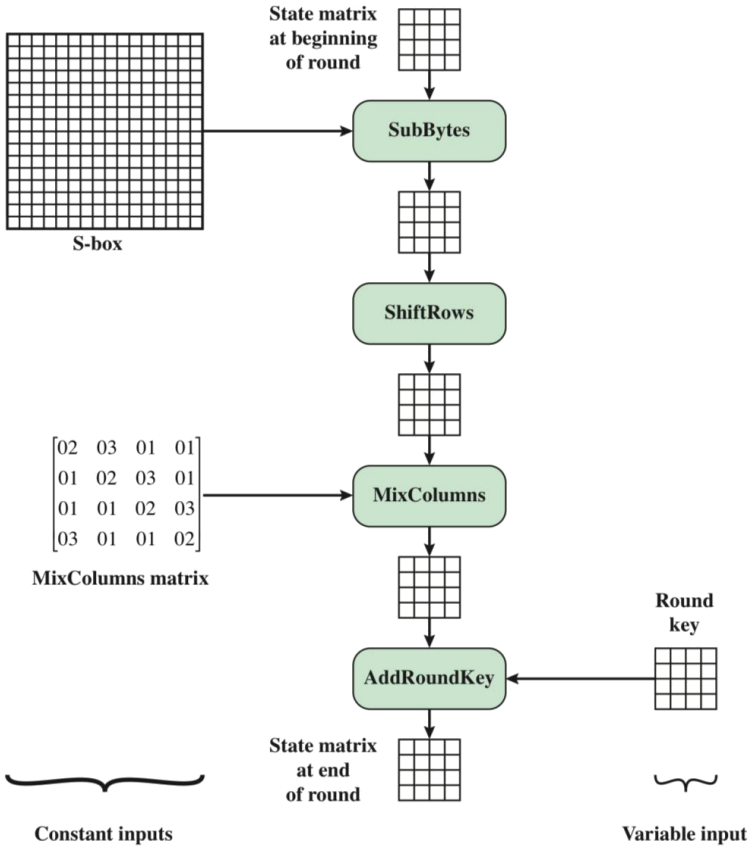


Figure 5.8 Inputs for Single AES Round

AES Key Expansion

Takes as input a four-word (16 byte) key and produces a linear array of 44 words (176) bytes This is sufficient to provide a four-word round key for the initial Add Round Key stage and each of the 10 rounds of the cipher.Key is copied into the first four words of the expanded key.The remainder of the expanded key is filled in four words at a time.Each added word *w*[i] depends on the immediately preceding word, *w[i – 1]*, and the word four positions back, w[i – 4].In three out of four cases a simple XOR is used.For a word whose position in the *w* array is a multiple of 4, a more complex function is used

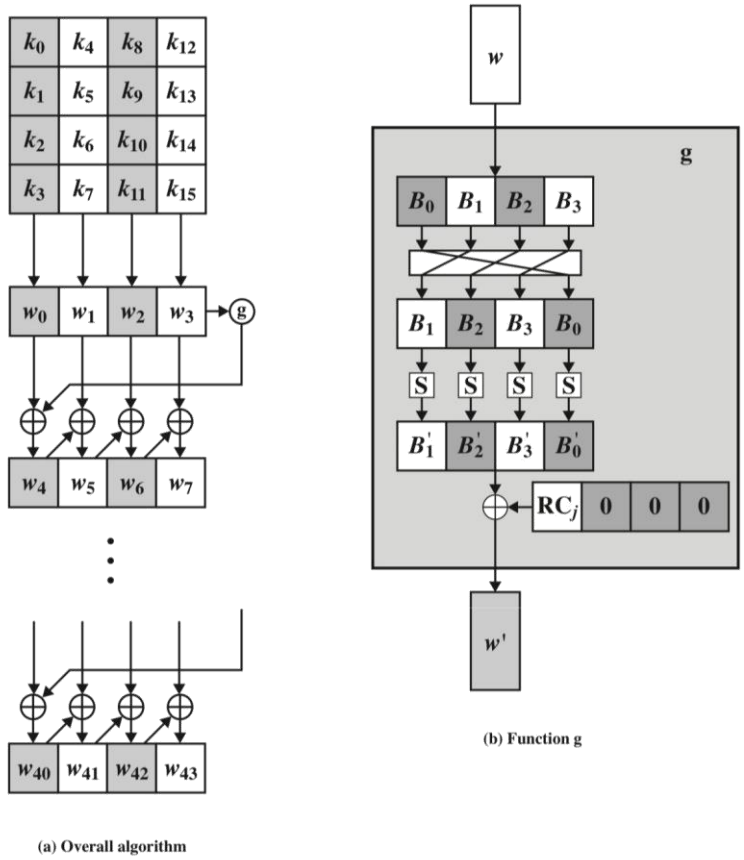


Figure 5.9 AES Key Expansion

AES decryption is not identical encryption since steps done in to reverse but can define an equivalent inverse cipher with steps as for encryption– but using inverses of each step – with a different key schedule. Works since result is unchanged when – swap byte substitution & shift rows

– swap mix columns & add (tweaked) round key

Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shifting
  - add round key works on byte XORs
  - mix columns requires matrix multiply in GF(2^8) which works on byte values, can be simplified to use a table lookup

Double DES

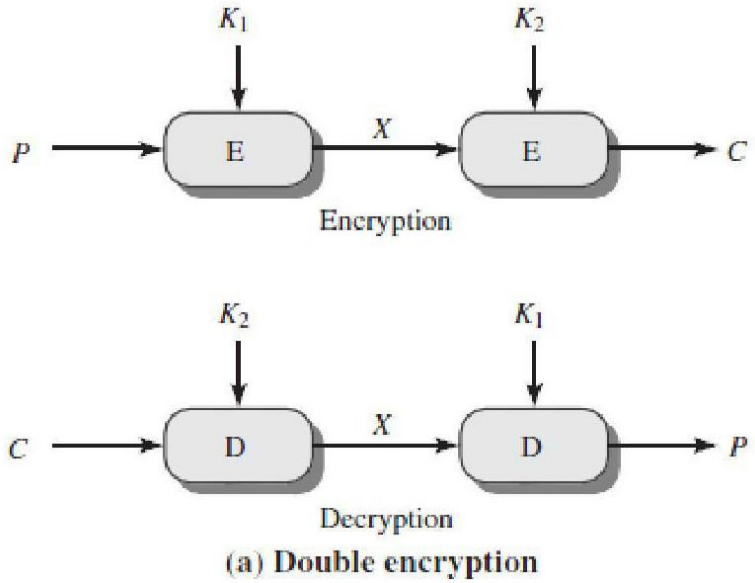
Double DES has two encryption stages and two keys. Given a plaintext P and two encryption keys, K1 and K2 and , ciphertext C is generated as

$$C = E(K2, E(K1, P))$$

Decryption requires that the keys be applied in reverse order

$$P = D(K1, D(K2, C))$$

this scheme apparently involves a key length of 56 * 2 = 112 bits, resulting in a dramatic increase in cryptographic strength



(a) Double encryption

Meet-In-The-Middle Attack

It is based on the observation that, if we have C = E(K2, E(K1, P))

then X = E(K1, P) = D(K2, C)

Given a known pair, (P, C) the attack proceeds as follows

- First, encrypt P for all 256 possible values of K1
- Store these results in a table and then sort the table by the values of X
- Next, decrypt C using all 256 possible values of K2
- As each decryption is produced, check the result against the table for a match.
- If a match occurs, then test the two resulting keys against a new known plaintext–cipher text pair.
- If the two keys produce the correct cipher text, accept them as the correct keys. For any given plaintext P, there are 264 possible cipher text values that could be produced by double DES, the foregoing procedure will produce about 248 false alarms on the first (P,C) pair.
- With an dditional 64 bits of known plaintext and ciphertext, the false alarm rate is reduced to 248-64 = 2-16

If the meet-in-the-middle attack is performed on two blocks of known plaintext–cipher text, the probability that the correct keys are determined is 1 – 2 -16. The result is that a known plaintext attack will succeed against double DES, which has a key size of 112 bits, with an effort on the order of 256, which is not much more than the 255 required for single DES

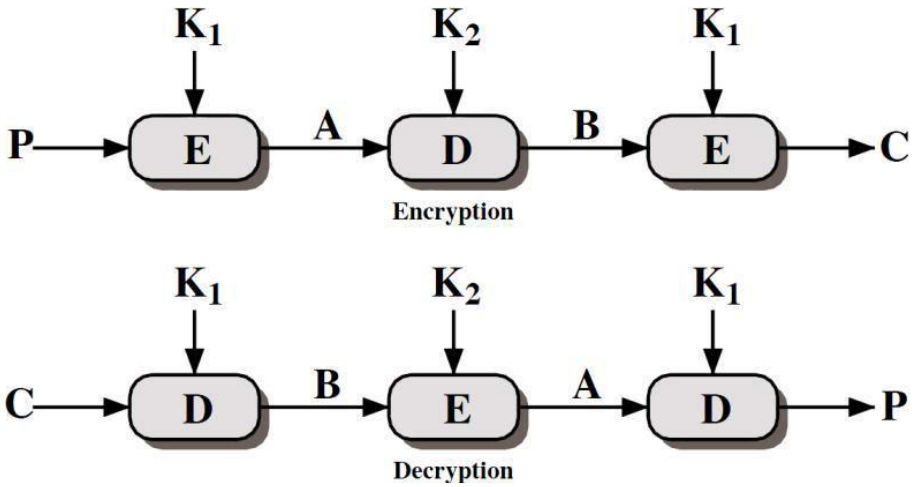
Triple DES

triple encryption method that uses only two keys. The function follows an encrypt-decrypt-encrypt (EDE) sequence

C = E(K1, D(K2, E(K1, P)))

There is no cryptographic significance to the use of decryption for the second stage. The advantage is that it allows users of 3DE to decrypt data encrypted by users of the older single DES:

C = E(K1, D(K1, E(K1, P))) = E(K1, P)



Attacks on TDES

Known-Plaintext Attack on Triple DES

Triple DES with Three Keys

Three-key 3DES has an effective key length of 168 bits and is defined as follows:

C = E(K3, D(K2, E(K1, P))) □

Backward compatibility with DES is provided by putting K3 = K2 or K1 = K2.

A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME

Blowfish Algorithm

Feature's of Blowfish

- Replace the overhead of AES, Triple DES & IDEA
- Data size: 64 bits
- Key size: Default 128 bits (vary 32bits to 448bits)
- No. of Rounds: 16 rounds
- Unpatented and unlicensed

Blowfish is a symmetric block encryption algorithm designed by Bruce Schneier in consideration with,

• **Fast:** It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.

• **Compact:** It can run in less than 5K of memory.

• **Simple:** It uses addition, XOR, lookup table with 32-bit operands.

• **Secure:** The key length is variable it can be in the range of 32~448 bits: default 128 bits key length.

• It is suitable for applications where the key does not change often, like communication link or an automatic file encryptor. Unpatented and royality-free.

The Blowfish Algorithm

There are two parts to this algorithm;

–A part that handles the expansion of the key.

–A part that handles the encryption of the data.

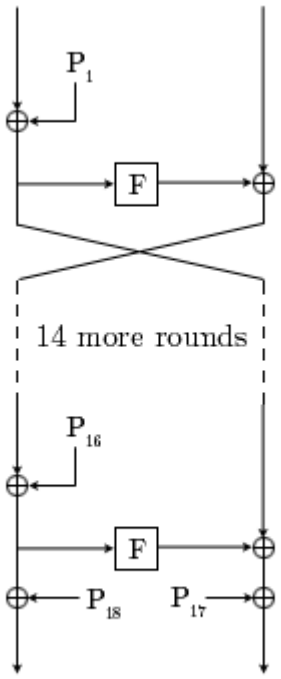
The expansion of the key: break the original key into a set of sub keys. Specifically, a key of no more than 448 bits is separated into 4168 bytes. There is a P-array and four 32-bit S-boxes. The P-array contains 18 32-bit sub keys, while each S-box contains 256 entries.

The encryption of the data: 64-bit input is denoted with an x, while the P-array is denoted with a Pi (where i is the iteration).

The Blowfish Algorithm: Key Expansion

Blowfish has a 64-bit block size and a key length of anywhere from 32 bits to 448 bits (32-448 bits in steps of 8 bits; default 128 bits).It is a 16-round Feistel cipher and uses large key-dependent S-boxes. It is similar in structure to CAST-128, which uses fixed S-boxes.

The diagram to shows the action of Blowfish. Each line represents 32 bits. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XOR ed with one of the two remaining unused P-entries.



The Blowfish Algorithm: Key Expansion

- Initialize the P-array and S-boxes
- XOR P-array with the key bits. For example, P1 XOR (first 32 bits of key), P2 XOR (second 32 bits of key), ...
- Use the above method to encrypt the all-zero string
- This new output is now P1 and P2
- Encrypt the new P1 and P2 with the modified sub keys
- This new output is now P3 and P4
- Repeat 521 times in order to calculate new sub keys for the P-array and the four S-boxes

The Blowfish Algorithm: Encryption

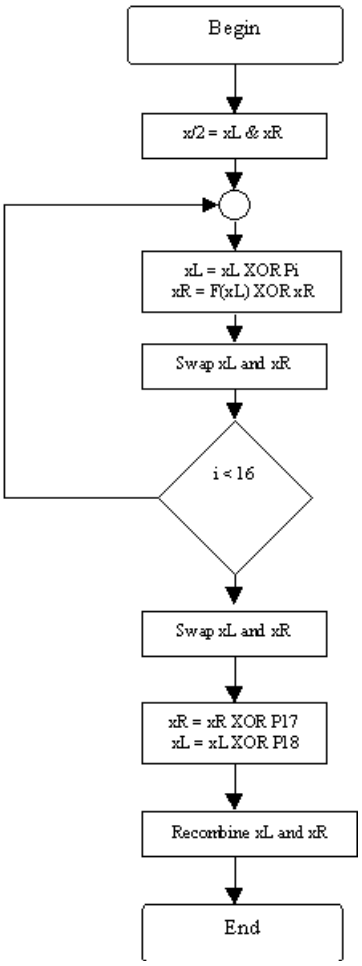
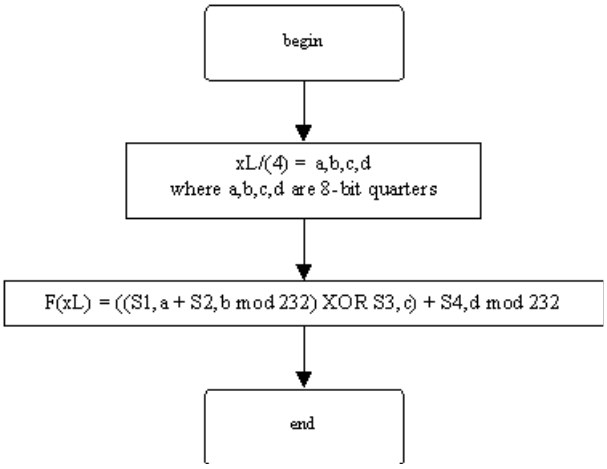


Diagram of Blowfish's F function



The diagram to the right shows Blowfish's F-function. The function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 232 and XORed to produce the final 32-bit output.

Since Blowfish is a Feistel network, it can be inverted simply by XORing P17 and P18 to the ciphertext block, then using the P-entries in reverse order.

Blowfish's key schedule starts by initializing the P-array and S-boxes with values derived from the hexadecimal digits of pi, which contain no obvious pattern.

•The secret key is then XORed with the P-entries in order (cycling the key if necessary). A 64-bit all-zero block is then encrypted with the algorithm as it stands.

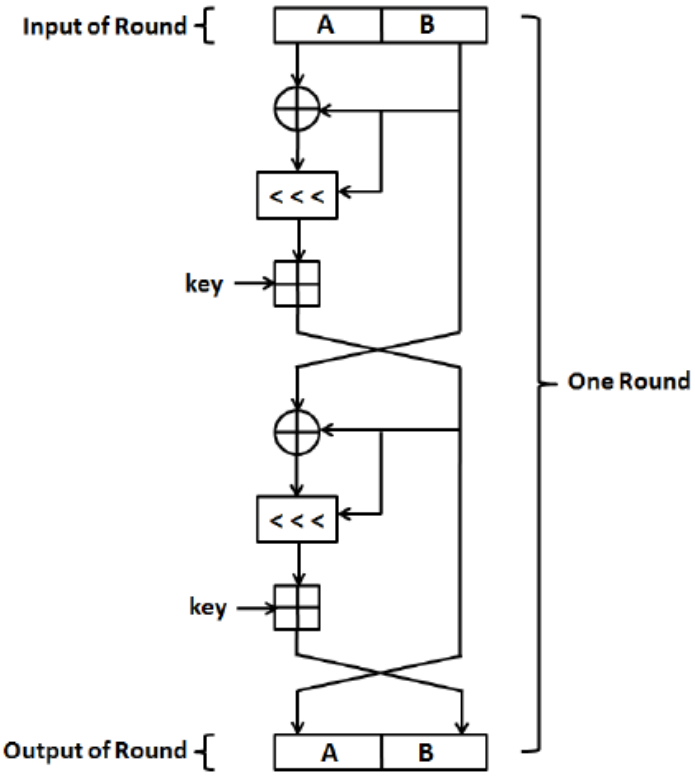
•The resultant ciphertext replaces P1 and P2. The ciphertext is then encrypted again with the new subkeys, and P3 and P4 are replaced by the new ciphertext. This continues, replacing the entire P-array and all the S-box entries.

•In all, the Blowfish encryption algorithm will run 521 times to generate all the subkeys - about 4KB of data is processed.

RC5 Algorithm – Block Cipher

RC5 Algorithm is designed by Ronald Rivest (of RSA fame) . It is used in various RSA Data Security Incorporation products. RC5 can vary key size / data size / no rounds.
Default data = 64 bits, key size = 128 bits & rounds = 12 rounds

- very clean and simple design
- easy implementation on various CPUs
- yet still regarded as secure



RC5 is a family of ciphers RC5-w/r/b
–w = word size in bits (16/32/64) nb data=2w

–r = number of rounds (0..255)
–b = number of bytes in key (0..255)

Nominal version is RC5-32/12/16
–ie 32-bit words so encrypts 64-bit data blocks
–using 12 rounds
–with 16 bytes (128-bit) secret key

Three parts:-
–Key Expansion
–Encryption Algorithm
–Decryption Algorithm

RC5 Algorithm – Key Expansion
•Requirements of key expansion
–Filling the expanded key table array S[0…t – 1] with random binary words
•"t" – Size of table "S" => 2 (r+1)
–S table is not an "S-box" like DES.
•Entries in S sequentially, one at a time.
–Random binary words are derived from the K

RC5 Algorithm :Encryption Algorithm
Two w-bit words are denoted as A and B
A = A + S[0];
B = B + S[1];
for i = 1 **to** r **do**
A = ((A ⊕ B) <<< B) + S[2 * i];
B = ((B ⊕ A) <<< A) + S[2 * i + 1];
The output is in the registers A and B.Work is done on both A and B, unlike DES where only half input is updated.

Decryption Algorithm :easily derived from encryption–Two w-bit words are denoted as A and B
for i = r **downto** 1 **do**
B = ((B – S[2 * i + 1]) >>> A) ⊕ A;
A = ((A – S[2 * i] >>> B) ⊕ B;
B = B - S[1];
A = A - S[0];

The output is in the registers A and B.

Private-Key Cryptography

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption: key distribution and digital signatures. The idea of public key schemes, and the first practical scheme, which was for key distribution only, was published in 1977 by Diffie & Hellman. The concept had been previously described in a classified report in 1970 by James Ellis (UK CESG) - and subsequently declassified [ELLI99]. Its interesting to note that they discovered RSA first, then Diffie-Hellman, opposite to the order of public discovery! There is also a claim that the NSA knew of the concept in the mid-60's [SIMM93].

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 key.

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

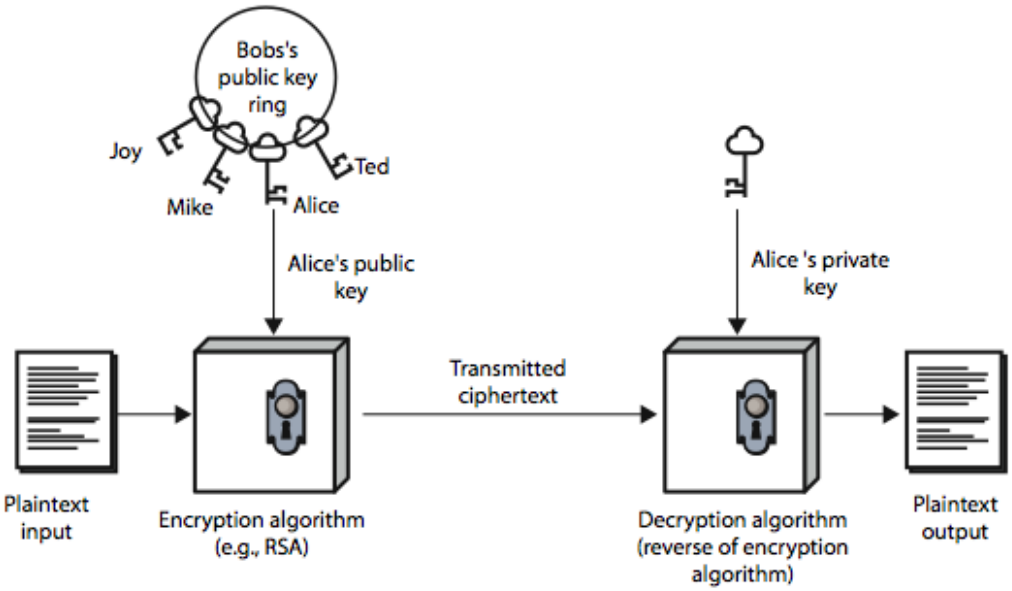
• **Plaintext:** This is the readable message or data that is fed into the algorithm as input.

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 cipher texts.

• **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.



(a) Encryption

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

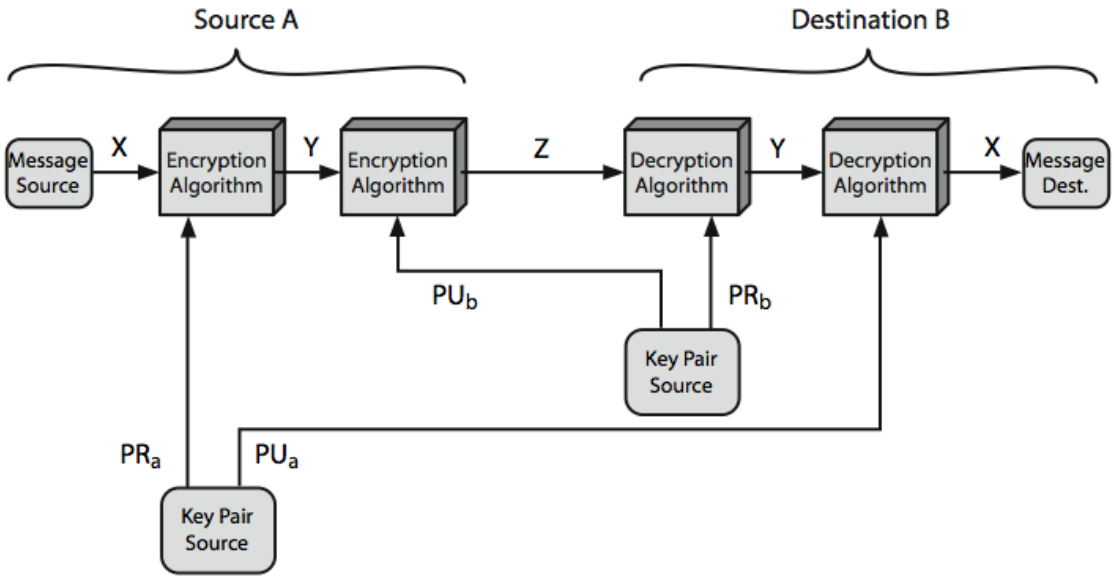
With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

Let us take a closer look at the essential elements of a public-key encryption scheme, using below Figure There is some source A that produces a message in plaintext, $X = [X1, X2, \ldots, XM]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, *PUb*, and a private key, *PRb*. *PRb* is known only to B, whereas *PUb* is publicly available and therefore accessible by A. With the message X and the encryption key *PUb* as input, A forms the cipher text $Y = [Y1, Y2, \ldots, YN]$:

$$Y = E(PUb, X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D(PRb, Y)$$



Public-Key Cryptosystem: Authentication and Secrecy

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

Public-key systems are characterized by the use of a cryptographic type of algorithm with two keys. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into the three categories:

• Encryption/decryption: The sender encrypts a message with the recipient's public key.

• Digital signature: The sender "signs"a message with its private key, either to the whole message or to a small block of data that is a function of the message.

• Key exchange: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

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

RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Rivest-Shamir-Adleman (RSA) Scheme

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. 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 Algorithm

RSA makes use of an expression with exponentials.Plaintext is encrypted in blocks with each block having a binary value less than some number *n*

•Encryption and decryption are of the following form, for some plaintext block *M* and ciphertext block C

$C = Me \bmod n$

$M = Cd \bmod n = (Me)d \bmod n = Med \bmod n$

•Both sender and receiver must know the value of *n*

•The sender knows the value of *e,* and only the receiver knows the value of *d*

•This is a public-key encryption algorithm with a public key of *PU={e,n}* and a private key of *PR={d,n}*

Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number *n*. That is, the block size must be less than or equal to log2(*n*) + 1; in practice, the block size is *i* bits, where 2*i* 6 *n* … 2*i*+1. Encryption and decryption are of the following form, for some b plaintext block *M* and cipher text block *C*.

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

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

For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of *e*, *d*, and *n* such that *Med* mod *n* = *M* for all *M* 6 *n*.
2. It is relatively easy to calculate *Me* mod *n* and *Cd* mod *n* for all values of *M* 6 *n*.
3. It is infeasible to determine *d* given *e* and *n*.

For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form

$$M^{ed} \bmod n$$

The preceding relationship holds if *e* and *d* are multiplicative inverses modulo f(*n*), where f(*n*) is the Euler totient function. It is shown in Chapter 8 that for *p*, *q* prime, f(*pq*) = (*p* - 1)(*q* - 1).

The relationship between *e* and *d* can be expressed as

$$ed \bmod \phi(n) = 1 \qquad \textbf{(9.1)}$$

This is equivalent to saying

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

That is, *e* and *d* are multiplicative inverses mod $\phi(n)$. Note that, according to the rules of modular arithmetic, this is true only if *d* (and therefore *e*) is relatively prime to $\phi(n)$. Equivalently, gcd($\phi(n)$, *d*) = 1. See Appendix R for a proof that Equation (9.1) satisfies the requirement for RSA.

We are now ready to state the RSA scheme. The ingredients are the following:

p, *q*, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

The private key consists of {*d*, *n*} and the public key consists of {*e*, *n*}. Suppose that user A has published its public key and that user B wishes to send the message *M* to A. Then B calculates

$C = M^e \bmod n$ and transmits C. On receipt of this cipher text, user A decrypts by calculating $M = C^d \bmod n$.

Figure summarizes the RSA algorithm

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

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

An example from [SING99] is shown in Figure 9.6.

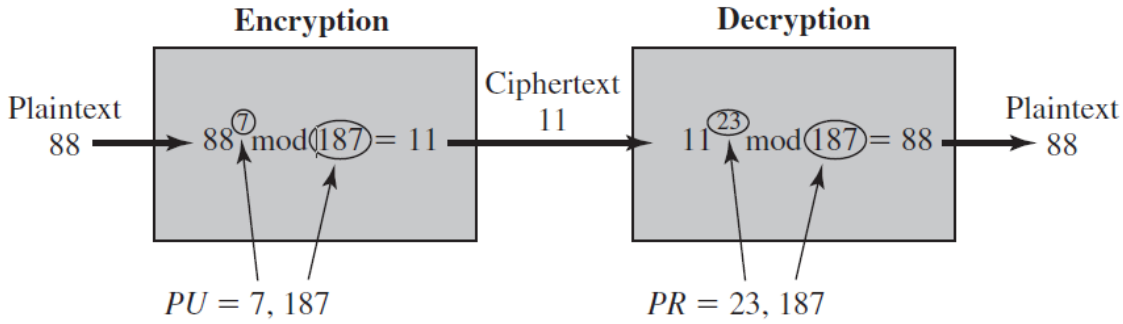


Figure 9.6 Example of RSA Algorithm

For this example, the keys were generated as follows.

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 * 11 = 187$.
3. Calculate $f(n) = (p - 1)(q - 1) = 16 * 10 = 160$.
4. Select e such that e is relatively prime to $f(n) = 160$ and less than $f(n)$; we choose $e = 7$.
5. Determine d such that de K 1 (mod 160) and d 6 160. The correct value is $d = 23$, because $23 * 7 = 161 = (1 * 160) + 1$; d can be calculated using the extended Euclid's algorithm.

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$.

The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 887 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

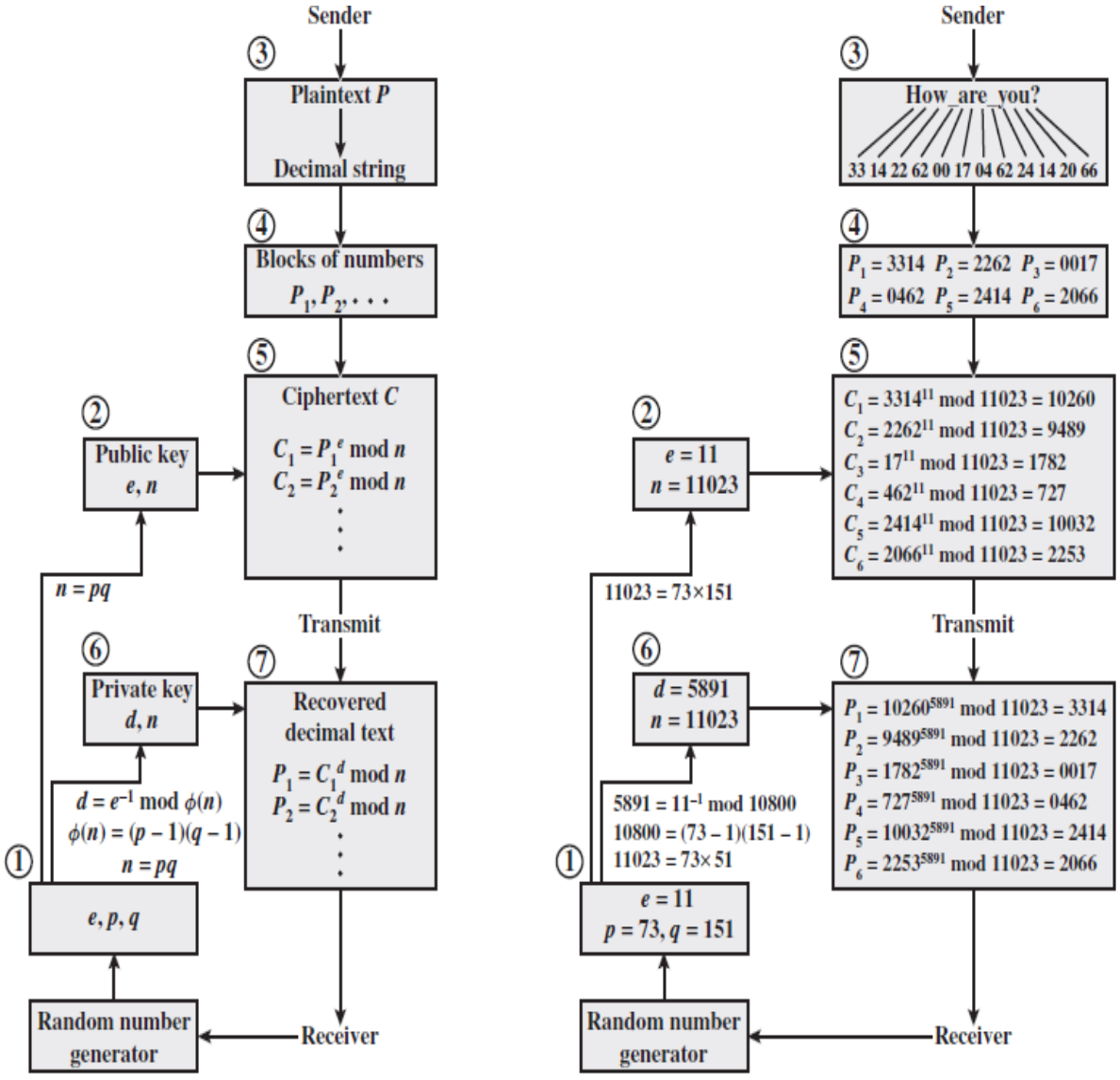
$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

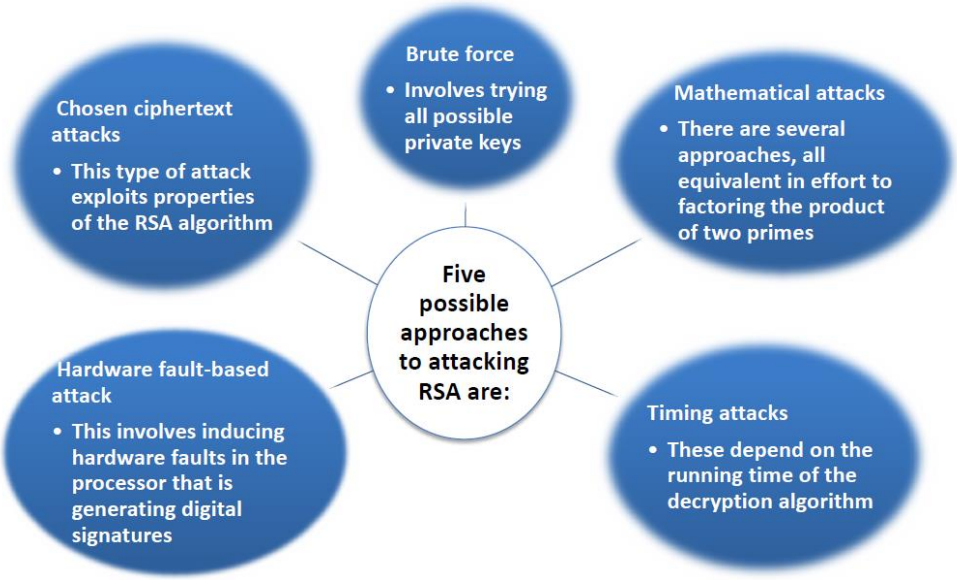


(a) General approach

(b) Example

Figure 9.7 RSA Processing of Multiple Blocks

The Security of RSA



Factoring Problem

•We can identify three approaches to attacking RSA mathematically:
–Factor *n* into its two prime factors. This enables calculation of ø(*n*) = (*p – 1) x* (*q* – 1), which in turn enables determination of *d = e-1 (mod* ø(n))

–Determine ø(n) directly without first determining *p* and *q.* Again this enables determination of *d = e-1 (mod* ø(n))

–Determine *d* directly without first determining ø(n)

Timing Attacks
•Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages

This attack is alarming for two reasons:
o It comes from a completely unexpected direction it is a cipher text-only attack

- A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.
- We can explain the attack using the modular exponentiation algorithm modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit

Working of this attack

- The attack proceeds bit-by-bit starting with the leftmost bit, bk
- Suppose that the first j bits are known
- For a given cipher text, the attacker can complete the first j iterations of the for loop.
- The operation of the subsequent step depends on the unknown exponent bit.

- if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow with a 1 bit, then this bit is assumed to be 1. If a number of observed execution times for the entire algorithm are fast, then this bit is assumed to be 0

Methods to overcome timing attacks

Constant exponentiation time

- Ensure that all exponentiations take the same amount of time before returning a result.
- This is a simple fix but does degrade performance

Random delay

Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.

Blinding

- Multiply the cipher text by a random
- number before performing exponentiation.
- This process prevents the attacker from knowing what cipher text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack

- **Explain Diffie-Hellman Key exchange algorithm with its merits and demerits. (APR/MAY 2011) (MAY/JUNE 2014) (APR/MAY 2010) (MAY/JUNE 2013) (NOV/DEC 2012)**
- **Users A and B use the Diffie-Hellman Key exachange technique with a common prime q=71 and a primitive root $\alpha = 7$. If the user A has private key X_A=5, what is A's public key Y_A? (8) (MAY/JUNE 2014)**
- **Explain Diffie-Hellman key exchange algorithm with an example. Consider a Diffie-Hellman scheme with a common prime q=353 and a primitive root α=3.Users A and B have private keys X_A=17 and X_B=21 respectively. What is theshared secret key K_1 and K_2? (16) (NOV-DEC 2014)**
- **How does Diffie-Hellman key exchange achieve security? (2) (MAY/JUNE 2007)**

Diffie-Hellman Key Exchange

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76b] and is generally referred to as Diffie-Hellman key exchange. A number of commercial products employ this key exchange technique.

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages. The algorithm itself is limited to the exchange of secret values.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. Recall from Chapter 8 that a primitive root of a prime number p is one whose powers modulo p generate all the integers from 1 to p - 1. That is, if a is a primitive root of the prime number p, then the numbers a mod p, a2 mod p,c, ap-1 mod 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 K ai (mod p) where 0 … i … (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 loga,p(b).

The Algorithm

Figure 10.1 summarizes the Diffie-Hellman key exchange algorithm. For this scheme, there are two publicly known numbers: a prime number q and an integer a that is a primitive root of q. Suppose the users A and B wish to create a shared key.

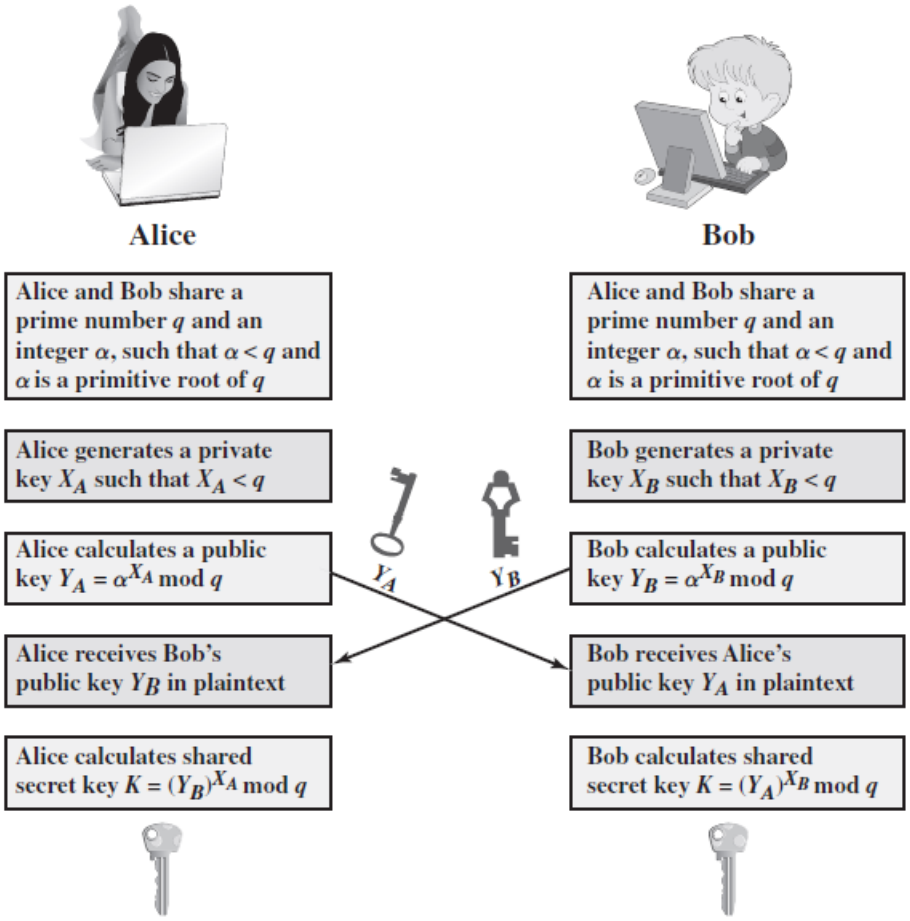


Figure 10.1 The Diffie-Hellman Key Exchange

User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. Thus, X_A is A's private key and Y_A is A's corresponding public key, and similarly for B. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = (Y_A)^{X_B} \bmod q$. These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{by the rules of modular arithmetic} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

The result is that the two sides have exchanged a secret value. Typically, this secret value is used as shared symmetric secret key. Now consider an adversary who can observe the key exchange and wishes to determine the secret key K. Because X_A and X_B are private, an adversary only has the following ingredients to work with: q, α, Y_A, and Y_B. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

$$X_B = \mathrm{dlog}_{\alpha,q}(Y_B)$$

The adversary can then calculate the key K in the same manner as user B calculates it. That is, the adversary can calculate K as

$$K = (Y_A)^{X_B} \bmod q$$

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Here is an example.Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case a = 3.
A and B select private keys $XA = 97$ and $XB = 233$, respectively.
Each computes its public key:
A computes $YA = 397 \bmod 353 = 40$.
B computes $YB = 3233 \bmod 353 = 248$.
After they exchange public keys, each can compute the common secret key:
A computes $K = (YB)XA \bmod 353 = 24897 \bmod 353 = 160$.
B computes $K = (YA)XB \bmod 353 = 40233 \bmod 353 = 160$.

Man-in-the-Middle Attack

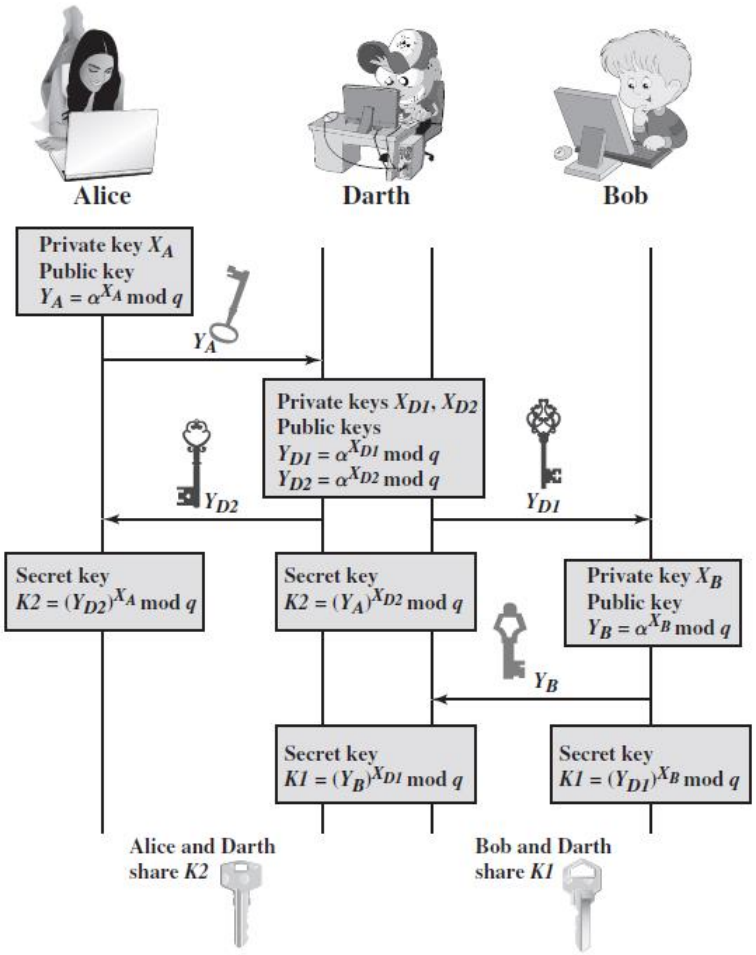


Figure 10.2 Man-in-the-Middle Attack

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.

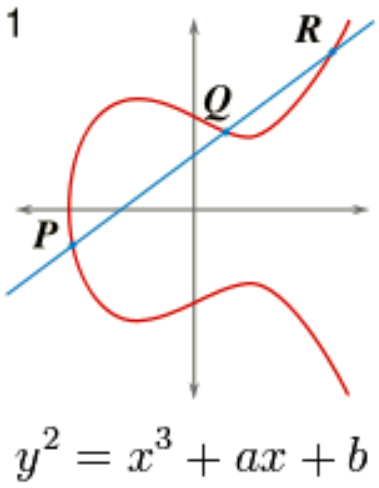
1. Alice sends an encrypted message M: E($K2$, M).
2. Darth intercepts the encrypted message and decrypts it to recover M.
3. Darth sends Bob $E(K1, M)$ or E($K1$, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob. The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates

1. **Explain how the elliptic curves are useful for cryptography? (16)(MAY/JUNE 2012)**
2. **Using Elliptic curve encryption/decryption scheme, key exchange between users A and B is accomplished. The cryptosystem parameters are, Ellyptic group of points E11 (1,6) and point G on the elliptic curve is G=(2,7). B'ssecret key is nB = 7. Now when. (i) A wishes to encrypt the message Pm = (10,9) and chooses the random value K=3. Determine the ciphertext Cm. (ii) How will B recover Pm form Cm. (iii) Find out B's public key PB. (16) (MAY/JUNE 2007)**

Elliptic Curve Cryptography

majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials. It imposes a significant load in storing and processing keys and messages • an alternative is to use elliptic curves . ECC offers same security with smaller bit sizes

An elliptic curve is defined by an equation in two variables x & y, with coefficients • consider a cubic elliptic curve of form – y2 = x3 + ax + b – where x,y,a,b are all real numbers – also define zero point O . ECC have addition operation for elliptic curve – geometrically sum of Q+R is reflection of intersection R



The equation of an elliptic curve is given as,**E -> Elliptic Curve,P -> Point on the curve ,n -> Maximum limit (This should be a prime number)**

- Elliptic curve cryptography uses curves whc variables & coefficients are finite
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - use integers modulo a prime
    - best in software
  - binary curves $E_{2m}(a,b)$ defined over GF(2^n)
    - use polynomials with binary coefficients
    - best in hardware

Global Public Elements	
$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation	
Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation	
Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A
$K = n_A \times P_B$

Calculation of Secret Key by User B
$K = n_B \times P_A$

Figure: ECC Diffie-Hellman Key Exchange

Key exchange using elliptic curves can be done as:

- First pick a large integer q, which is either a prime number p or an integer of the form 2^m, and elliptic curve parameters a and b. This defines the elliptic group of points $E_q(a, b)$.
- Next, pick a *base point* $G = (x_1, y_1)$ in $E_p(a, b)$ whose order is a very large

value n. The order n of a point G on an elliptic curve is the smallest positive integer n such that $nG = 0$ and G are parameters of the cryptosystem known to all participants.

- A key exchange between users A and B can be accomplished as:
  - A selects an integer n_A less than n. This is A‟s private key. A then generates a public key $P_A = n_A G$; the public key is a point in $E_q(a, b)$.
  - B similarly selects a private key n_B and computes a public key P_B.
  - A generates the secret key $k = n_A P_B$. B generates the secret key $k = n_B P_A$.

The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times n_B \times G = n_B \times n_A \times G = n_B \times P_A$$

- To break this scheme, an attacker would need to be able to compute k given G and kG, which is assumed to be hard.

Example:

Consider $p = 211$; $E_p(0,-4)$, which is equivalent to the curve $y^2 = x^3 - 4$; and $G = (2, 2)$. One can calculate that $240G = O$. A‟s private key is $n_A = 121$, so A‟s public key is $P_A = 121(2, 2) = (115, 48)$. B‟s private key is $n_B = 203$, so B‟s public key is $203(2, 3) = (130, 203)$. The shared secret key is $121(130, 203) = 203(115, 48) = (161, 69)$.

The secret key is a pair of numbers. If this key is to be used as a session key for conventional encryption, then a single number must be generated.

Elliptic Curve Encryption/Decryption

The first task in this system is to encode the plaintext message m to be sent as an (x, y) point P_m. As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key nA and generates a public key $P_A = n_A G$.

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = kG, P_m + kP_B$$

To decrypt the ciphertext, B multiplies the first point in the pair by B‟s private key and subtracts the result from the second point:

$$P_m + kP_B - n_B kG = P_m + k\, n_B\, G - n_B kG = P_m$$

A has masked the message Pm by adding kP_B to it. Nobody but A knows the value of k, so even though P_b is a public key, nobody can remove the mask kP_B. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

Example:
The global public elements are $q = 257$; $E_q(a, b) = E_{257}(0, -4)$, which is equivalent to the curve $y^2 = x^3 - 4$; and $G = (2, 2)$. Bob‟s private key is $n_B = 101$, and his public key is $P_B = n_B G = 101(2, 2) = (197, 167)$. Alice wishes to send a message to Bob that is encoded in the elliptic point $P_m = (112, 26)$. Alice chooses random integer $k = 41$ and computes $kG = 41(2, 2) = (136, 128)$, $kP_B = 41(197, 167) = (68, 84)$ and $P_m + kP_B = (112, 26) + (68, 84) = (246, 174)$. Alice sends the ciphertext $C_m = (C_1, C_2) = \{(136, 128), (246, 174)\}$ to Bob. Bob receives the ciphertext and computes $C_2 - n_B C_1 = (246, 174) - 101(136, 128) = (246, 174) - (68, 84) = (112, 26)$.

Security of Elliptic Curve Cryptography

The security of ECC depends on how difficult it is to determine k given kP and P. This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the Pollard rho method. A considerably smaller key size can be used for ECC compared to RSA. Furthermore, for equal key lengths, the computational effort required for ECC and RSA is comparable. Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA.

PART-A

1. What is difference between a block cipher and a stream cipher? (MAY/JUNE 2012), (APR/MAY 2015)

Stream Cipher Block Cipher

1. A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.

1. A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key. It has broader range of applications.

2. What is key distribution center? (MAY/JUNE 2012)

For symmetric key cryptography, the trusted intermediaryis called a **Key Distribution Center (KDC**), which is a single, trusted network entity with whom one

has established ashared secret key. A key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed. Each user must share a unique key with the key distribution center for purposes of key distribution. The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.

3. Mention the application of public key cryptography. (MAY/JUNE 2012)

• **Encryption/decryption:** The sender encrypts a message with the recipient‟s public key.

• **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

• **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

4. State whether symmetric and asymmetric cryptographic algorithms require key exchange. (MAY/JUNE 2014)

Both symmetric and asymmetric cryptographic algorithms require key exchange. Key exchange (also known as "key establishment") is any method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. If the cipher is a symmetric key cipher, both will need a copy of the same key. If an asymmetric key cipher with the public/private key property, both will need the other's public key.

5. Write down the difference between the public key and private key cryptosystems. (MAY/JUNE 2012)

Public key cryptosystems	**Private key cryptosystems**
1. One algorithm is used for encryption and decryption with pair of keys. 2. The sender and receiver must each have one of the matched pair of keys. One of two keys must be kept secret.	1. Same algorithm and same key is used for encryption and decryption. 2. Sender and receiver must share the algorithm and key. Key must be kept secret

6. Is it possible to use the DES algorithm to generate message authentication code? Justify. (NOV/DEC 2014)

Yes. It can use any block cipher chaining mode and use final block as a MAC. Data Authentication Algorithm(DAA) is a widely used MAC based on DES-CBC. Encrypt message using CBC mode and send just the final block as the MAC.

7. What is triple DES? (APR/MAY2010)

Triple DES involves repeating the DES algorithm three times on the plaintext using two or three different keys to produce the ciphertext.

8. Write down the purpose of the S-Boxes in DES. (NOV/DEC 2011)

S-Box is a nonlinear, invertible matrix in which each row defines a general reversible substitution. DES consists of 8 S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.

9. Define : Diffusion. (NOV/DEC 2011)

Diffusion is one of the basic building block for any cryptographic system. In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits. This is possible by using permutation.

10. List out the parameters of AES. (NOV/DEC 2011)

- Key Size (words/bytes/bits)
- Plaintext Block Size (words/bytes/bits)
- Number of Rounds
- Round Key Size (words/bytes/bits)
- Expanded Key Size (words/bytes)

11. State the difference between conventional encryption and public-key encryption. (NOV/DEC 2011)

Conventional encryption	**Public-key encryption**
Needed to Work: 1. The same algorithm with the same key Is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. Needed for Security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	Needed to Work: 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). Needed for Security: 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

12. State few applications of RC4 algorithm. (APR/MAY 2015)

RC4 is used in SSL/TLS. It is also used in WEP, the IEEE 802.11 wireless networking security standard. It can also be found in a number of other applications including email encryption products.

13. Define primitive root. (NOV/DEC 2012)

A primitive root of a prime number p is one whose powers modulo p generate all the integers from 1 to p - 1. That is, if a is a primitive root of the prime number p, then the numbers a mod p, a^2 mod p, …….. , a^{n-1} mod p are distinct and consist of the integers from 1 through p - 1 in some permutation.