2 Block Ciphers and Public Key Cryptography

Module 2

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Block cipher principles
- Data Encryption Standard (DES)
- Triple DES
- Block cipher modes of operation
- Advanced Encryption Standard (AES)
- Blowfish
- RC5 algorithm
- Public key cryptography
  - Principles of public key cryptosystems
  - RSA Algorithm
  - Knapsack Algorithm
  - ElGamal Algorithm
  - Key management
- Diffie Hellman Key exchange

2.1 Concept Building - Types of Symmetric Algorithms (Ciphers)

Q. What are Block Cipher Modes. Describe any two in detail. **MU - Dec. 18, 10 Marks**

Symmetric key based algorithms (ciphers) can work either on blocks of bits (characters) or one bit at a time.

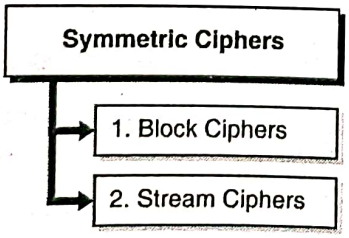


Fig. 2.1.1 : Symmetric Ciphers

✍ ***Definition :*** *The algorithms that work on blocks are called block ciphers.*

✍ ***Definition :*** *The algorithms that work on one-bit at a time are called stream ciphers.*

2.1.1 Block Ciphers

In block ciphers, the information that needs to be encrypted is broken into smaller and equal block sizes. Then, the encryption operation (substitution and transposition) is applied to each block. The resultant ciphertext from each block is then combined to produce the encrypted information.

Fig. 2.1.2 illustrates simplified block diagram of how a block cipher works. The information is broken into equal size blocks and then the encryption operation is carried out on each block. If the block size has lesser number of characters than required to form a block, then padding is done to fill the block. Padding is just filling some temporary information to form a block. Finally, the resulting encrypted information from each block is combined to get the overall encrypted message.

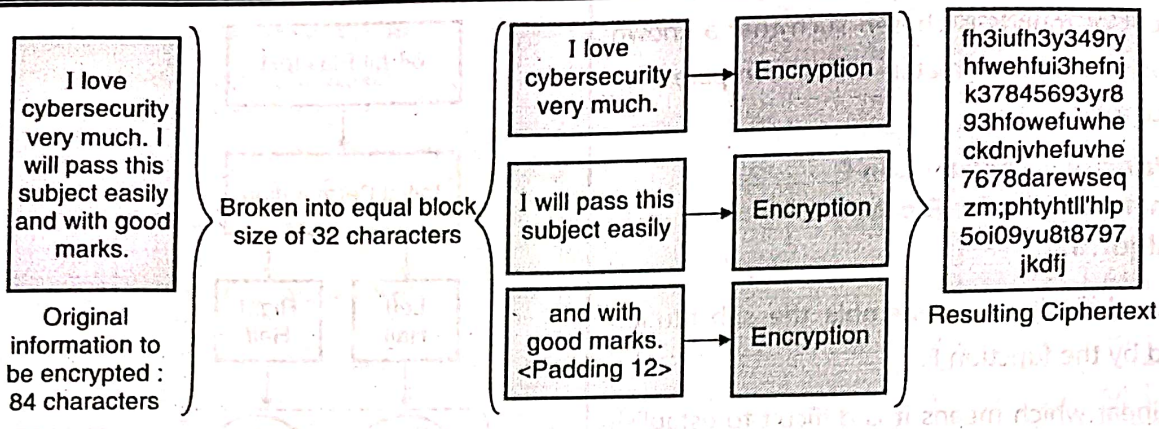


Fig. 2.1.2 : Block Ciphers

Data Encryption Standard (DES) and Advanced Encryption Standard(AES) are two of the examples of Symmetric Block Ciphers.

2.1.2 Stream Ciphers

Unlike block ciphers, stream ciphers work on one bit of plaintext at a time. Each bit of plaintext is combined with the bits of security key and then XORed to get ciphertext.

> **Note :** If you recall from your logical design classes, the Table 2.1.1 is the truth table of XOR. For result to be 1, both the inputs should be different.

Table 2.1.1 : XOR Truth Table

Sr. No	Input X	Input Y	Output Z (XOR X, Y)
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

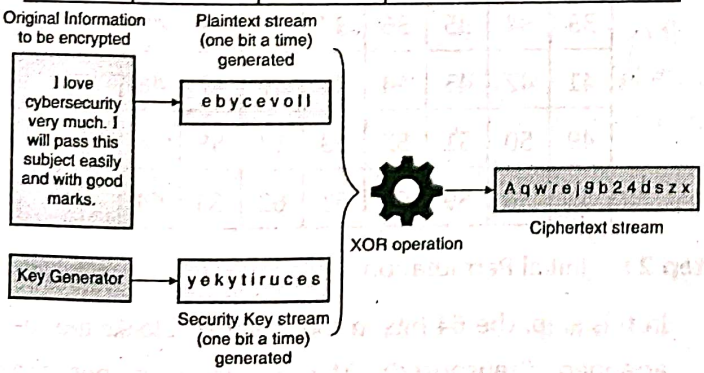


Fig. 2.1.3 : Stream Ciphers

RC4 is an example of stream cipher.

2.1.3 Comparison between Block and Stream Cipher

Sr. No.	Comparison Attribute	Block Cipher	Stream Cipher
1.	Security	High	Low
2.	Speed	Low	High
3.	Application	Non-real time such as documents	Real time data such as Voice
4.	Commonly used	Yes	No

2.2 Block Cipher Principles

There are three critical components in designing a block cipher as shown in Fig. 2.2.1.

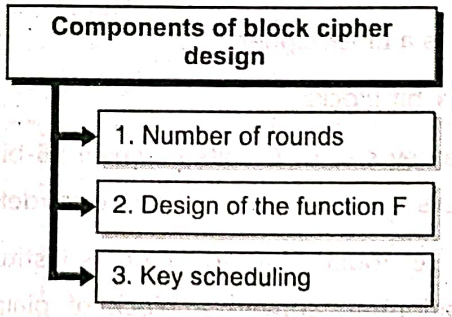


Fig. 2.2.1 : Components of block cipher design

Following are the principles around each of these.

1. Design Principles for Number of Rounds in the Block Cipher Algorithm

(i) The greater the number of rounds, the more difficult it is to perform cryptanalysis.

(ii) The number of rounds is chosen such that a known cryptanalysis takes a greater effort compared to brute-force attack.

2. Design Principles for function F (Feistel network) in the Block Cipher Algorithm

(i) It must be difficult to re-assemble the substitution performed by the function F.

(ii) F is non-linear which means it is difficult to establish any relation between input to F and output from F.

(iii) F should have high avalanche effect.

3. Principles for Key scheduling

(i) Subkey selection should be such that it is difficult to work backwards to derive the main key.

(ii) Subkeys should be hard to guess as well.

(iii) The key schedule should produce avalanche effect.

2.3 Data Encryption Standard (DES)

✍ ***Definition :*** *Data Encryption Standard (DES) is a symmetric key based block cipher standard used for encryption and decryption.*

It came into existence and usage around Nov 1976 and was predominantly used in the industry until 2002.

Major attributes of DES

- It is a symmetric key based algorithm
- It works as a block cipher
- It uses 64-bit blocks
- It uses a key size of 64-bits in which 56-bits are the actual keys and 8-keys are used for error detection
- It uses 16 rounds of operation (substitution and transposition) to convert a block of plaintext into ciphertext
- DES is now considered insecure and obsolete due to its short key-size (56-bits only)

2.3.1 Block Diagram and Internals of DES

Fig. 2.3.1 illustrates simplistic view of DES. Let us understand what happens at each stage.

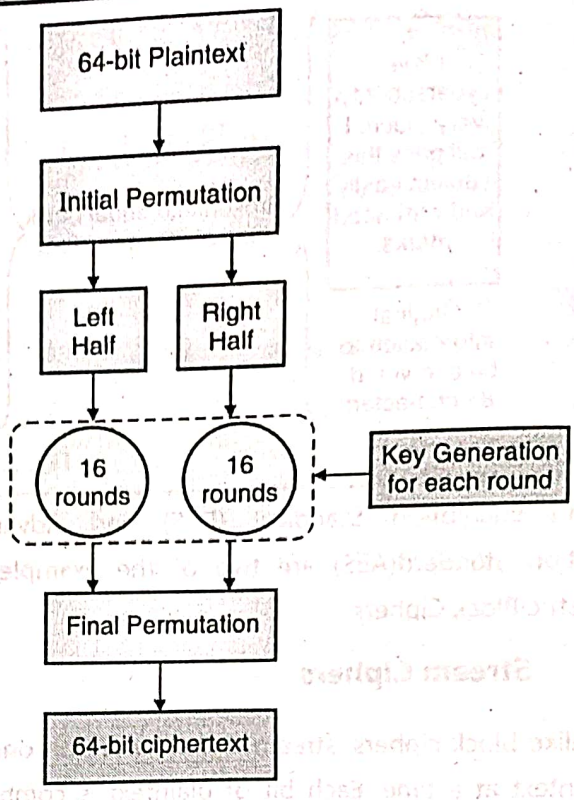


Fig. 2.3.1 : Block diagram of DES

Step 1 : Creation of 64-bit blocks

In this step, the plaintext information to be encrypted is broken into 64-bit blocks. DES is a block cipher and block creation is similar to as explained in the earlier section.

Table 2.3.1 : 64 bits of plaintext

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

Step 2 : Initial Permutation

In this step, the 64 bits in the plaintext blocks are re-arranged (transposed). This is done as per the diffusion property of the cipher to ensure that any small variance in plaintext produces a large variance in the ciphertext.

Table 2.3.2 : Initial Permutation (re-arrange bits of plaintext)

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

Step 3 : Left Half and Right Half Split

In this step, the bits from the Initial Permutation stage are split into two parts – left half and right half each containing 32-bits. These individual 32-bit blocks are then continuously worked through the 16 rounds of operation.

Step 4 : Subkey Key Generation

For the 16 rounds of operation, a unique subkey is derived for each round from the 56-bit key. The key is derived using complex mathematical functions. Each generated subkey is 48-bit long.

Step 5 : Rounds

Left half and the right half both individually go through 16 rounds of encryption operation. In each of the rounds, the derived subkey is used to produce temporary ciphertext. This temporary ciphertext produced after each round is used in the next round until the final round is complete. Each round consists of substitutions and successive permutations.

Step 6 : Final Permutation

In the last stage, we need to bring the bits back to their respective positions. The bit positions were changed at the initial permutation stage.

Table 2.3.3 : Final permutation (re-arrange bits of ciphertext)

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

Step 7 : Final Ciphertext

Once all the steps are done, you get the final ciphertext for the plaintext given the security key of your choice via DES.

Exam Tip : If you hear that an algorithm is broken or is insecure, it means that it is computationally feasible to find out the key used for encryption. Note that cryptography heavily depends upon our understanding of mathematics and the computation power available today. What is secure and infeasible today, could be insecure and feasible to crack in future.

2.3.2 Block Cipher Modes of Operation (for DES and other Block Ciphers in General)

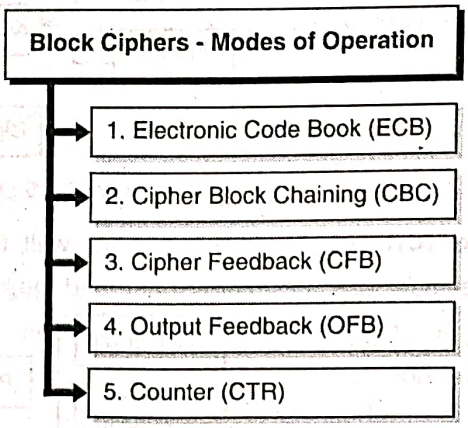


Fig. 2.3.2 : Modes of Operation for Block Ciphers

DES and other block ciphers can potentially work in several modes. Let's review them carefully.

1. **Electronic Code Book (ECB) Mode :** In this mode, the same key is used to encrypt all the blocks. Key derivatives or subkeys are not used. Additionally, each block is treated separately and the ciphertext of previous block does not influence successive blocks.

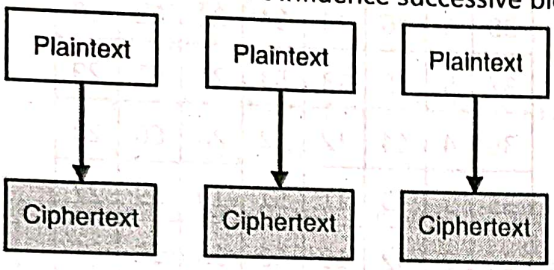


Fig. 2.3.3 : Electronic Code Book (ECB) Mode

2. **Cipher Block Chaining (CBC) Mode :** In this mode, the ciphertext of previous block is used with the next plaintext block. The two blocks (ciphertext of previous block and plaintext of next block) are XORed and then passed through the encryption operation. This generates a lot more randomness in the final ciphertext.

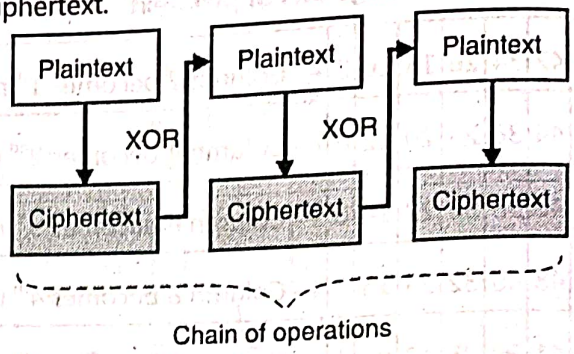


Fig. 2.3.4 : Cipher Block Chaining (CBC) Mode

3. **Cipher Feedback (CFB) Mode :** In this mode, the block cipher works like a stream cipher. The ciphertext from the previous block is XORed with the key (keystream) for the next block. This way the key increasingly becomes random and brings more randomness in the overall encryption process.

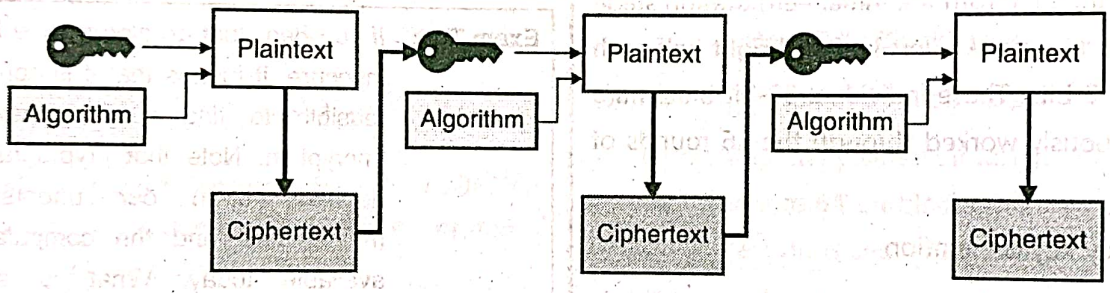


Fig. 2.3.5 : Cipher Feedback (CFB) Mode

4. **Output Feedback (OFB) Mode :** In this mode, the block cipher works like a stream cipher. The keystream used in the previous block is XORed with the keystream of the next block.

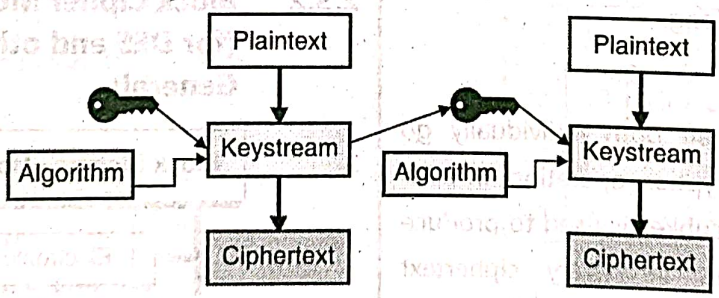


Fig. 2.3.6 : Output Feedback (OFB) Mode

5. **Counter (CTR) Mode :** In this mode as well, the block cipher works like a stream cipher. The key is converted into keystream (as used in stream cipher) and the keystream is XORed with a counter that increases for every block.

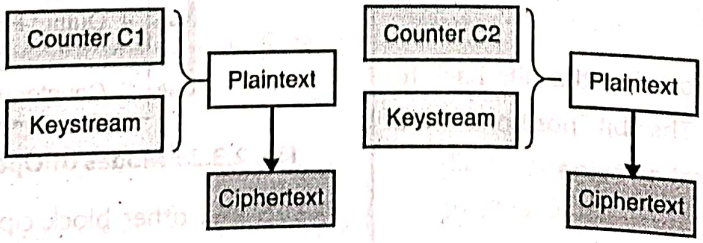


Fig. 2.3.7 : Counter (CTR) Mode

2.3.3 Comparison between Modes of Operation

Sr. No.	Mode	ECB	CBC	CFB	OFB	CTR
1.	In-Parallel block encryption	Yes	No	No	No	Yes
2.	Suited for	Small Information	Any size of information	Small Information	Small Information	Any size of information
3.	Security and randomness	Low	High	High	High	High
4.	Speed	High	Medium	Medium	Medium	High
5.	Complexity	Low	High	High	High	Low
6.	Works like stream cipher?	No	No	Yes	Yes	Yes

Weakness in DES

1. **Small key size**

   56-bits of keys have a keyspace (possible values) of 2^{56}. While that might seem a lot, it is actually not given the compute power we have today. In 1990s, the compute power we had was significantly lower and hence was considered secure at that time.

   ✍ ***Definition :*** *The type of attack where each combination is tried in an attempt to find the right combination is also called as brute force attack.*

2. **Prone to linear cryptanalysis**

   DES has been proven to be susceptible to linear cryptanalysis.

3. **Prone to differential cryptanalysis**

   DES has been proven to be susceptive to differential cryptanalysis.

2.3.4 Double DES

– In order to strengthen DES, it was considered to increase the key size to 112-bits effectively. The way it was chosen to do so was to use 2 keys of 56-bits each. Let's call them K1 and K2.

Mathematically, it can be denoted as follows:

Ciphertext C = Encryption(K2, Encryption(K1, Plaintext P))

Plaintext P = Decryption(K1, Decryption(K2, Ciphertext C))

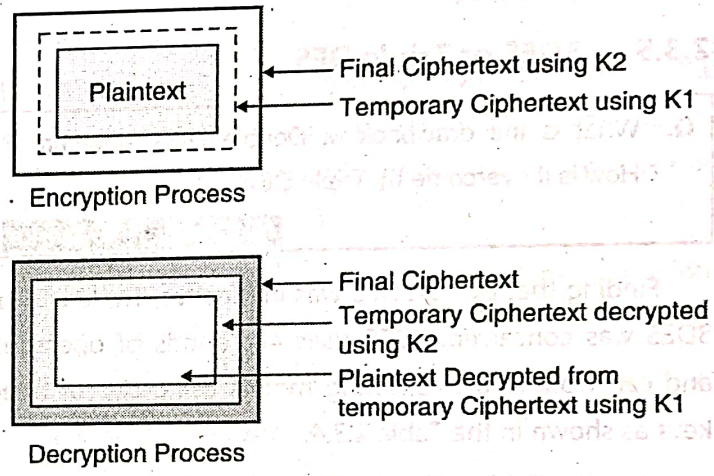


Fig. 2.3.8 : Double DES

– During the encryption process, first the plaintext is encrypted with Key K1 and then the result is again encrypted with Key K2 to get the final ciphertext for plaintext.

– During the decryption process, first the Key K2 is used to decrypt to get the ciphertext that Key K1 can decrypt to get the plaintext.

– However, Double DES was proven to be ineffective. Meet in the middle attack was shown to reduce the complexity to just 2^{57}(2^{56} attempts made twice, hence $2x2^{56} = 2^{57}$) instead of 2^{112} as originally thought.

– So, using K1 if you could derive temporary ciphertext using encryption process and using K2 if you could also derive the same temporary ciphertext using decryption process, you have found a match and the keys you chose (K1 and K2) are now known to you. Hence, you could effectively find both the keys and break Double DES without original thought of complexity of 112 bits.

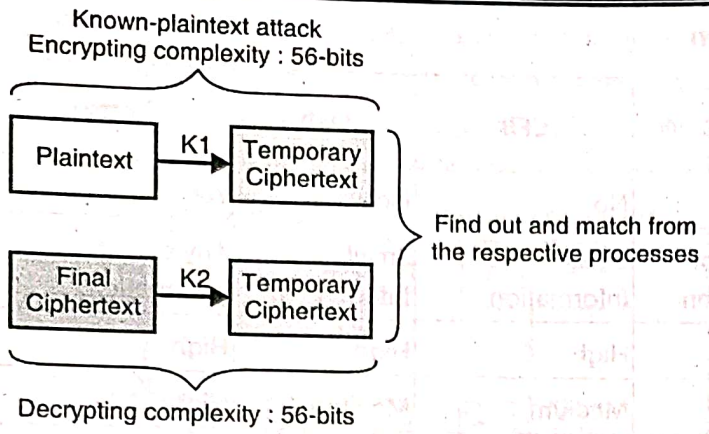


Fig. 2.3.9 : Complexity in Double DES

– Hence, Double DES was not adopted in the industry and is not used.

2.3.5 3DES or Triple DES

Q. What is the drawback of Double DES algorithm ? How is it overcome by Triple DES ?

MU - May 19, 5 Marks

Finding that Double DES was ineffective, Triple DES or 3DES was conceived. 3DES uses 48 rounds of operation and can work in the following modes using two or three keys as shown in the Table 2.3.4.

Table 2.3.4 : 3DES or Triple DES

Sr. No.	Mode	Number of keys	Key 1	Key 2	Key 3
1.	DES-EEE3	3	Encryption	Encryption	Encryption
2.	DES-EDE3	3	Encryption	Decryption	Encryption
3.	DES-EEE2	2	Encryption	Encryption	Encryption Using Key 1
4.	DES-EDE2	2	Encryption	Decryption	Encryption Using Key 1

You might wonder how decryption helps? Note that if you encrypt a plaintext using a key (say K1) and run the decryption process using a different key (say K2), the text (from encryption process using K1) becomes more random. The use of a different key in the decryption process from the encryption process brings added randomness and hence helps to make attacks such as linear or differential cryptanalysis extremely hard.

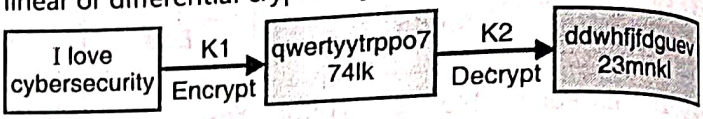


Fig. 2.3.10 : Decryption process

2.4 Advanced Encryption Standard (AES)

✍ ***Definition :*** *Advanced Encryption Standard (AES) is a symmetric key based block cipher standard used for encryption and decryption.*

The standard became effective on May 26, 2002 and is predominantly used in the industry today due to its strong cipher properties. AES replaced DES as the new standard when DES was found to be insecure and vulnerable to various attacks.

Evaluation Criteria for AES

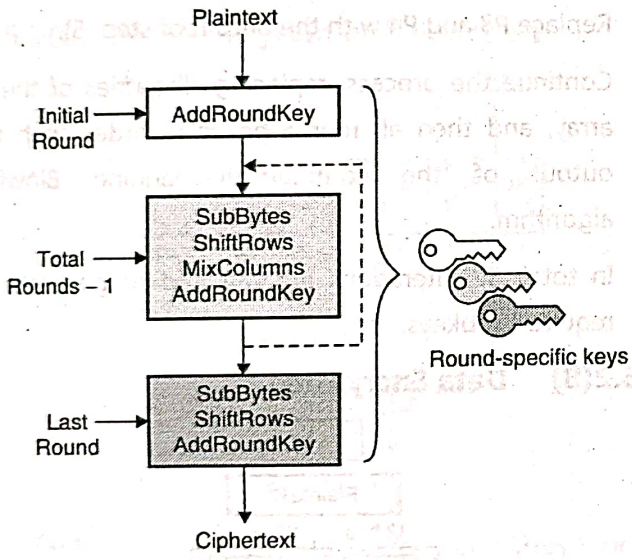
When looking for the next better alternative to DES, NIST defined the following acceptability requirements and evaluation criteria for AES.

1. AES shall be publicly defined.
2. AES shall be a symmetric block cipher.
3. AES shall be designed so that the key length may be increased as needed.
4. AES shall be implementable in both hardware and software.
5. AES shall either be a) freely available or b) available under terms consistent with the American National Standards Institute (ANSI) patent policy.
6. Algorithms which meet the above requirements will be judged based on the following factors
   (a) security (i.e., the effort required to cryptanalyse)
   (b) computational efficiency
   (c) memory requirements
   (d) hardware and software suitability
   (e) simplicity
   (f) flexibility
   (g) licensing requirements

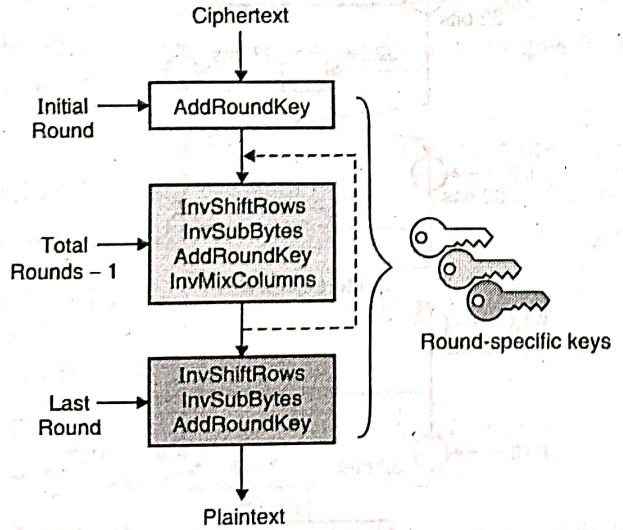
Major attributes of AES

- It is a symmetric key based algorithm
- It works as a block cipher
- It uses 128-bit blocks
- It can work with key sizes of 128, 192 and 256 bits
- Number of rounds of operation depends upon the key size
  - 128-bit key undergoes 10 rounds
  - 192-bit key undergoes 12 rounds
  - 256-bit key undergoes 14 rounds
- AES is considered highly secure due to its long key sizes and is used in the industry today

2.4.1 Block Diagram and Internals of AES



(a) AES Encryption Process



(b) AES Decryption Process

Fig. 2.4.1 : Block diagram of AES

Let's understand the blocks.

1. **AddRoundKey :** In this transformation step, a round key is generated and XORed with the intermediate (temporary) ciphertext. This block is used in both encryption as well as decryption process.
2. **SubBytes :** In this transformation step, the intermediate ciphertext undergoes various substitution operations. It is used for encryption process.
3. **ShiftRows :** In this transformation step, the intermediate ciphertext undergoes various row-wise transposition operations. It is used for encryption process.
4. **MixColumns :** In this transformation step, the intermediate ciphertext undergoes various column-wise transposition operations.It is used for encryption process.
5. **InvSubBytes :** This is inverse of SubBytes operation. It is used in the decryption process.
6. **InvShiftRows :** This is inverse of ShiftRows operation. It is used in the decryption process.
7. **InvMixColumns :** This is inverse of MixColumns operation. It is used in the decryption process.

2.4.2 Comparison between DES and AES

Q. Compare and contrast DES and AES.

MU - May 19, 10 Marks.

Sr. No.	Comparison Attribute	DES	AES
1.	Cryptographic Strength	Low	High
2.	Key Size	56-bit	128, 192 and 256 bits
3.	Block Size	64-bit	128-bit
4.	Rounds	16	10, 12, 14 - based on key size
5.	Usage	Obsolete - Not used	Currently used industry standard

2.5 Blowfish

Definition : Blowfish is a symmetric key based block cipher standard used for encryption and decryption.

Blowfish was designed in 1993 by Bruce Schneier as a fast and a free alternative to existing encryption algorithms such as DES and IDEA.

2.5.1 Major attributes of Blowfish

- It is a symmetric key based algorithm
- It works as a block cipher
- It uses 64-bit blocks
- It uses a variable length key size ranging from 32 bits to 448 bits
- It uses 16 rounds of operation to convert a block of plaintext into ciphertext
- It is Unpatented and royalty-free, and No license is required for its use

2.5.2 Block Diagram and Internals of Blowfish

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts

1. Key-expansion 2. Data- encryption

Key expansion converts a key into several subkeys. Data encryption occurs via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

2.5.2(A) Key Expansion and Subkey Generation

The subkeys are calculated using the Blowfish algorithm. The exact method is as follows:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (other than the initial 3). For example :

$$P_1 = 0 \times 243f6a88$$

$$P_2 = 0 \times 85a3308d3$$

$$P_3 = 0 \times 13198a2e$$

$$P_4 = 0 \times 03707344$$

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits.
3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P-array, and then all four S-boxes in order, with the output of the continuously-changing Blowfish algorithm.

In total, 521 iterations are required to generate all required subkeys.

2.5.2(B) Data Encryption

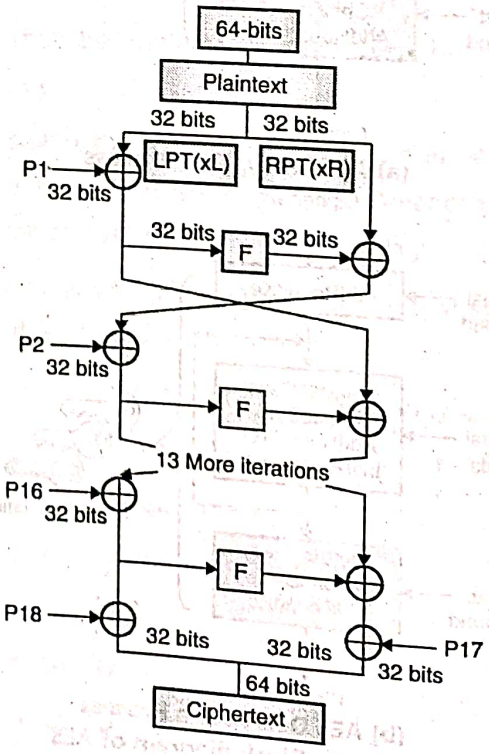


Fig. 2.5.1

Blowfish is a Feistel network consisting of 16 rounds. The input is a 64-bit block of plaintext. Let's call it x.

1. Divide x into two 32-bit halves: xL, xR
2. For i = 1 to 16: [16 rounds of operation]
   (a) xL = xL XOR Pi [From P-array derived from key expansion]
   (b) xR = F(xL) XOR xR [F is the Feistel Function]
   (c) Swap xL and xR
   (d) Next, I [Begin next round, steps a → c]
3. Swap xL and xR [Undo the last swap]
4. xR = xR XOR P17
5. xL = xL XOR P18
6. Recombine xL and xR

2.5.2(C) Block Diagram of F Function

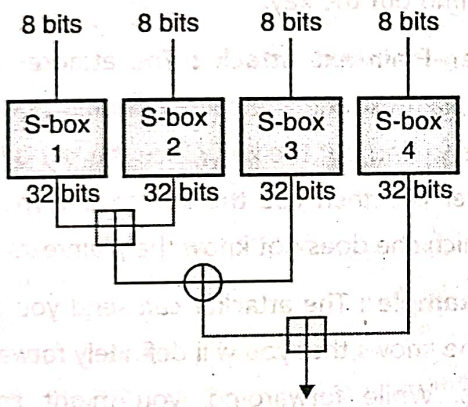


Fig. 2.5.2

The F function or the Feistel function utilizes S-boxes (substitution boxes) that work on the 32-bits of input. Each S-box works through 8-bits of inputs to give 32-bits output.

For example, xL(first 32-bits of 64-bit plaintext input) is divided into four eight-bit quarters: a, b, c, and d. Then, $F(xL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}$.

2.5.3 Strength of Blowfish

1. Faster than DES and IDEA
2. Support for large key size up to 448 bits making it suitable for various usage
3. It is unpatented and license-free, and is available free for all uses

2.5.4 Weakness of Blowfish

1. Only suitable for applications where the key does not change frequently
2. Some weak keys have been identified though they have not been broken
3. Other algorithms, such as AES, are frequently implemented in hardware and could be much faster than Blowfish

2.6 RC5 Algorithm

✍ ***Definition :*** *RC5 is a symmetric key based block cipher standard used for encryption and decryption.*

It was designed by Ronald Rivest in 1994. RC stands for "Rivest Cipher", or alternatively, "Ron's Code".

2.6.1 Major Attributes of RC5

- It is a symmetric key based algorithm
- It works as a block cipher
- It uses a variable block size – 32, 64 or 128
- It uses a variable length key size ranging from 0 to 2040 bits
- It uses variable number of rounds of operation – 0 to 255

2.6.2 Internals of RC5 Algorithm

RC5 is word-oriented. All of the basic computational operations have *w*-bit words as inputs and outputs. RC5 is a block-cipher with a two-word input (plaintext) block size and a two-word output (ciphertext) block size. The nominal choice for *w* is 32-bits for which RC5 has 64-bit plaintext and ciphertext block sizes.

The number *r* of rounds is the second parameter of RC5. Choosing a larger number of rounds provides an increased level of security. RC5 uses an "expanded key table" *S* that is derived from the user's supplied secret key. The size *t* of table *S* also depends on the number *r* of rounds. *S* has $t = 2(r+1)$ words.

2.6.3 Key Expansion

Key expansion consists of 3 steps:

1. Converting the Secret Key from Bytes to Words
2. Initializing the Array *S*
3. Mixing in the Secret Key

2.6.4 Encryption

Assume that the input block is given in two *w*-bit registers *A* and *B*. Assume that the key expansion step is already performed and thus the array S is already populated. The encryption algorithm can be outlined in the following pseudo code.

```
A = A + S[0];
B = B + S[1];
for i = 1 to r do
A = ((A xor B) <<< B) + S[2 * i];
B = ((B xor A) <<< A) + S[2 * i + 1];
```

The output is in the registers *A* and *B*.

2.7 Concept Building - Attacks on Cryptosystems

Now that you have a general understanding of the cryptosystems, let's learn some of the possible attacks on them.

Note : The attacks described here are common for any cryptographic algorithm be it DES, AES, RSA or any other. While for some algorithms it is comparatively easier and for others it is theoretically possible. Any specific attack against an algorithm is described in its respective section. Otherwise, you could mention and elaborate on the following attacks.

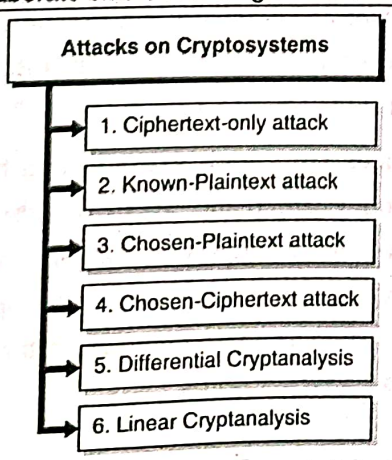


Fig. 2.7.1 : Attacks on Cryptosystems

1. **Ciphertext-only attack :** In this type of attack, the attacker has ciphertext of several messages. The algorithm is known, and the goal of the attack is to find out the key used in encryption. Once the key is found out, it is possible to decrypt messages that were encrypted using the key.

2. **Known-Plaintext attack :** The attacker knows the plaintext partially and the corresponding ciphertext. The goal is to find out the key. Once the key is known, the attacker can then use the key to decrypt ciphertext for which she does not know the plaintext.

– **For example :** You might be using a fixed greeting in your messages (for example, "Dear Friend") or you might be sending same message (for example, "Good morning") everyday to someone. The attacker could know this and also the corresponding ciphertext and try to find out the key.

3. **Chosen-Plaintext attack :** The attacker knows the exact plaintext and the corresponding ciphertext. The goal is to find out the key. Once the key is known, the attacker can then use the key to decrypt ciphertext for which she does not know the plaintext.

– **For example :** The attacker can send you a message that she knows that you will definitely forward to your friends. While forwarding, you might encrypt the message using your key. Now, the attacker can grab the ciphertext that you sent, and she already knows the plaintext message she sent you earlier.

4. **Chosen-Ciphertext attack :** In this attack, the attacker choses the ciphertext that she wants to be decrypted and know the corresponding plaintext. The goal again is to find out the key.

5. **Differential Cryptanalysis**

– In this attack, the attacker choses a pair of plaintexts and follows through each stage in their respective encryption process and compare the difference between the results at each stage.

– The key used in encrypting the pair is same. The goal again is to figure out the key by carefully studying the differences in results at various stages between the

pair. Since, the attacker chooses the plaintexts, differential analysis is considered to be a type of chosen-plaintext attack.

6. **Linear Cryptanalysis :** The attacker carries out a known-plaintext attack and tries to figure out the key. She evaluates the input and output at various stages of the encryption process and tries to find out the probability of specific key values.

2.7.1 Comparison between Differential and Linear Cryptanalysis

Sr. No.	Comparison Attribute	Differential Cryptanalysis	Linear Cryptanalysis
1.	Plaintext selection	Carefully chosen	Any random plaintext
2.	Plaintext used	In pairs	One by one
3.	Complexity of attack	High	Low
4.	Mathematical relation between plaintexts used	Specific differences (such as XOR)	Linear approximation (such as a series of XOR operations)
5.	Goal of the attack	Identify some bits of the unknown key	Identify the linear relation between some bits of the plaintext, some bits of the cipher text and some bits of the unknown key

2.8 Concept Building - Modular Arithmetic

✍ ***Definition :*** *The modular arithmetic deals with operations on integers specifically around remainders from division.*

Let's take a few examples.

Dividend	Divisor	Quotient	Remainder (Modulus)
15	5	3	0
15	4	3	3
15	3	5	0
15	2	7	1
15	1	15	0

So, for example, number 15 when divided by 2, gives you Quotient of 7 and Remainder of 1. This can be mathematically written as 15 mod 2 = 1.

Note : Before you proceed, try finding out a few mods (remainders from division) for numbers of your choice.

1. Congruence Property

Two numbers are said to be in congruence modulo, if they give out the same mod.

For example :

15 mod 2 = 1

17 mod 2 = 1

Hence, 15 is congruent to 17 modulo 2 i.e. 15 and 17 when undergo mod operation with 2, they both give same result of 1. They can be mathematically denoted as

$$15 \equiv 17 \pmod 2$$

2. Modular Addition

(A + B) mod C = (A mod C + B mod C) mod C

Let A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 + 15) mod 5 = 27 mod 5 = 2	= (12 mod 5 + 15 mod 5) mod 5 = (2 + 0) mod 5 = 2 mod 5 = 2

3. Modular operation on Negative numbers

If you come across modular operation on a negative number, make the number positive by repetitively adding mod until it becomes positive.

For example : If you have to find – 13 mod 5, keeping adding 5 to – 13 until you get a positive number. So,

$$-13 + 5 = -8 + 5 = -3 + 5 = 2$$

Now, do mod on the positive number you got. In this case, – 13 mod 5 becomes 2 mod 5. Hence, the answer is 2.

4. Modular Subtraction

$$(A - B) \bmod C = (A \bmod C - B \bmod C) \bmod C$$

Let A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 – 15) mod 5	= (12 mod 5 – 15 mod 5) mod 5
= –3 mod 5	= (2 – 0) mod 5
= 2	= 2 mod 5
	= 2

5. Modular Multiplication

$$(A * B) \bmod C = (A \bmod C * B \bmod C) \bmod C$$

Let A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 * 15) mod 5	= (12 mod 5 * 15 mod 5) mod 5
= 180 mod 5	= (2 *0) mod 5
= 0	= 0 mod 5
	= 0

6. Modular Inverse

Modular arithmetic does not have division operation. However, it has inverse. Inverse of a general number is 1 divided by that number.

For example: Inverse of 2 is $\frac{1}{2}$. In other words, a number when multiplied by its inverse would give 1. So, 2 multiplied by its inverse $\frac{1}{2}$ would give $2 * \frac{1}{2} = 1$.

So, modular inverse of A mod C is the value of B such that when A is multiplied by B and the mod C operation is carried out, it gives 1. Mathematically, it can be written as

$$A * B \equiv 1 \pmod{C}$$

Note : To understand the above, refer to the congruency equation. mod C operation on A*B should be same as mod C operation on 1.

Let's take an example.

Suppose you have to find modular inverse of 12 mod 5. Here, A = 12 and C = 5. Let's assume value of B from 0 onwards until we find (A*B) mod C = 1.

Table 2.8.1 : Modular inverse

Value of B	Operation	Result
0	(12 * 0) mod 5	0
1	(12 * 1) mod 5	2
2	(12 * 2) mod 5	4
3	(12 * 3) mod 5	1

Hence, modular inverse of 12 mod 5 is 3.

7. Prime Numbers

- These are whole numbers which are greater than 1 and are only divisible by 1 and itself.
- **For example :** 2, 3, 5, 7, 11 and various others.

8. Coprime Numbers

- Two integers a and b are said to be relatively prime, mutually prime, or coprime (also written co-prime) if the only positive integer (factor) that divides both of them is 1.
- **For Example :** 5 and 7 are coprime because their common factor is only 1 whereas 14 and 18 are not coprime because their common factor can also be 2.

9. Discrete Logarithm

- If a is an arbitrary integer relatively prime to n and g is a primitive root of n, then there exists exactly one number μ such that $a = g^{\mu} \pmod{n}$.
- The number μ is then called the discrete logarithm of a with respect to the base g modulo n and is denoted, $\mu = \text{ind}_g a \pmod{n}$.
- Discrete logarithms are quickly computable in a few special cases. However, no efficient method is known for computing them in general. Several important algorithms in public-key cryptography use discrete logarithms.

2.9 Greatest Common Divisor (GCD)

- Greatest Common Divisor (GCD) of two positive integers is the largest integer that can fully divide both the integers.

  For example : GCD (5, 10) is 5. GCD of (11, 13) is 1.

- GCD is usually found out by finding the factors of the respective integers and then choosing the common highest factor.

  For example : To find GCD (24, 70)

- **Factors of 24 :** 2 * 2 * 2 * 3 →(Factors could be 2, 3, 4, 6, 8, 12, 24)
- **Factors of 70 :** 2 * 5 * 7 →(Factors are only 2, 5, 7)

Hence, the largest common factor is 2. Hence, GCD (24, 70) = 2.

2.9.1 Euclid's or Euclidean Algorithm

Finding GCD for smaller numbers is quite straight forward. But, when it comes to finding GCD of large numbers, it might be a complex task. This is precisely where Euclid's (or Euclidean) algorithm helps.

Euclid's algorithm states that

gcd(a, b) = gcd(a mod b, b) if a > b

gcd(a, b) = gcd(a, b mod a) if b > a

2.9.2 Solved Examples

Ex. 2.9.1 : Find gcd(50, 65) using Euclidean algorithm.

Soln. : gcd(50, 65) = gcd(50, 65 mod 50)

[Because 65 is greater than 50]

= gcd(50, 15)

= gcd(50 mod 15, 15)

[Because 50 is greater than 15]

= gcd(5, 15)

= gcd(5, 15 mod 5)

[Because 15 is greater than 5]

= gcd(5, 0)

[Stop here once the mod of a term becomes 0]

gcd(50, 65) = 5 [Is the gcd(50, 65)]

Ex. 2.9.2 : Find gcd(464, 238) using Euclidean algorithm.

Soln. :

gcd(464, 238) = gcd(464 mod 238, 238)

= gcd(226, 238)

= gcd(226238 mod 226)

= gcd(226, 12)

= gcd(226 mod 12, 12)

= gcd(10, 12)

= gcd(10, 12 mod 10)

= gcd(10, 2)

= gcd(10 mod 2, 2)

= gcd(0, 2)

gcd(464, 238) = 2

Ex. 2.9.3 : Find gcd(105, 80).

Soln. :

gcd(105, 80) = gcd(105 mod 80, 80)

= gcd(25, 80 mod 25)

= gcd(25 mod 5, 5)

= gcd(0, 5)

gcd(105, 80) = 5

2.9.3 Extended Euclidean Algorithm

The Extended Euclidean Algorithm can be used to find the gcd of two numbers, and also to simultaneously express the gcd as a linear combination of these numbers. It helps to find values of coefficients x and y such that to satisfy the following equation : ax + by = gcd(a, b)

In the extended algorithm, the computation involves several entities. First, let's define them:

- **i(index) :** This would just be used to iterate (repeat) the process.

> **Note :** If b is greater than a, then assume a = b and b = a. Swap the values to avoid negatives

- **r (remainder) :** temporary placeholder variable for a and b

  r would be calculated as $r_{i+1} = r_{i-1} - q_i r_i$

- **q** would be calculated as $q_{i+1} = r_{i-1} / r_i$
- **s :** temporary placeholder for values while deriving coefficient x

  s would be calculated as $s_{i+1} = s_{i-1} - q_i s_i$
- **t :** temporary placeholder for values while deriving coefficient y

  t would be calculated as $t_{i+1} = t_{i-1} - q_i t_i$
- **x** would be

$$s_{i+1} = \frac{b}{\gcd(a,b)}$$

- **y** would be

$$t_{i+1} = \frac{a}{\gcd(a,b)}$$

Ex. 2.9.4 : For a = 161 and b = 42, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

$r_0 = 161$ and $r_1 = 42$

i starts with 0

First draw the initial table.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1

Starting steps

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 **= 3**	161 – 3 * 42 **= 35**	1 – 3 * 0 **= 1**	0 – 3 * 1 **= –3**

For the highlighted row,

- i = 1 (we are just doing the calculation for 2nd row, hence the value of i is still 1)
- q_1 can we written as q_i where i = 1

  So, $q_1 = r_{i-1} / r_i = r_0 / r_1 = 161 / 42 = 3$
- r_2 can be written as r_{i+1} where i = 1

  So, $r_2 = r_{i-1} - q_i r_i$ which means $r_2 = r_0 - q_1 * r_1$

  $r_2 = 161 - 3 * 42$

  $r_{i-1} = r_0$ which is 161

  $r_i = r_1$ which is 42
- Similarly, s_2 can be written as s_{i+1} where i = 1

  So, $s_2 = s_{i-1} - q_i s_i = s_0 - q_1 s_1 = 1 - 3*0 = 1$
- Similarly, t_2 can be written as t_{i+1} where i = 1

  So, $t_2 = t_{i-1} - q_i t_i = t_0 - q_1 t_1$

  $= 0 - 3*1 = -3$
- Similarly proceed to the next step.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 **= 3**	161 – 3 * 42 **= 35**	1 – 3 * 0 **= 1**	0 – 3 * 1 **= –3**
3	42 / 35 **= 1**	42 – 1 * 35 **= 7**	0 – 1 * 1 **= –1**	1 – 1 *–3 **= 4**

Here again

$q_i = q_2$, where i = 2

So, $q_2 = r_{i-1} / r_i = r_1 / r_2 = 42 / 35 = 1$

$r_3 = r_1 - q_2 * r_2 = 42 - 1 * 35 = 7$

$s_3 = s_1 - q_2 * s_2 = 0 - 1 * 1 = -1$

$t_3 = t_1 - q_2 * t_2 = 1 - 1 * -3 = 4$

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 = 3	161 – 3 * 42 = 35	1 – 3 * 0 = 1	0 – 3 * 1 = – 3
3	42 / 35 = 1	42 – 1 * 35 = 7	0 – 1 * 1 = – 1	1 * 1 *–3 = 4
4	35 / 7 = 5	35 – 5 * 7 = 0	Do not calculate	Do not calculate

$q_i = q_3$, where i = 3

So, $q_3 = r_2 / r_3 = 35 / 7 = 5$

$r_4 = r_2 - q_3 * r_3 = 35 - 5 * 7 = 0$

- Do not calculate s and t once you get r = 0
- Last calculated s and t become x and y respectively
- Last calculated r becomes gcd

So, according to extended Euclidean algorithm, for numbers 161 and 42

gcd(161, 42) = 7

In the equation ax + by = gcd(161,42)

x = – 1

y = 4

You can verify the answer by putting the values in the equation.

Left-hand side:

ax + by = 161 * – 1 – 42 * 4

= – 161 + 168

ax + by = 7 [which is equal to gcd(161, 42)]

Ex. 2.9.5 : For a = 256 and b = 5004, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. : First note that b > a. Hence, lets swap the values to make the calculations simple. We would re-swap them in the final answer.

So, assume a = 5004 and b = 256

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		5004	1	0
1		256	0	1

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		5004	1	0
1		256	0	1
2	5004 / 256 = 19	5004 – 19 * 256 = 140	1 – 19*0 = 1	0 – 19*1 = – 19
3	256 / 140 = 1	256 – 1 * 140 = 116	0 – 1*1 = –1	1 – 1*–19 = 20
4	140 / 116 = 1	140 – 1 * 116 = 24	1 – –1*1 = 2	–19 – 1*20 = – 39
5	116 / 24 = 4	116 – 24 * 4 = 20	– 1 – 4*2 = –9	20 – 39*4 = 176
6	24 / 20 = 1	24 – 20 * 1 = 4	2 – 9*1 = 11	– 39 – 1*176 = – 215
7	20 / 4 = 5	20 – 4* 5 = 0	Do not calculate	Do not calculate

gcd(256, 5004) = 4

We originally swapped the value. So, re-swap it.

Hence, x = – 215 and y = 11

Putting it in the equation, you get,

Left-Hand Side

ax + by = 256 * – 215 + 5004 * 11

= – 55,040 + 55,044

ax + by = 4 [which is equal to right hand side = gcd(256, 5004)]

Ex. 2.9.6 : For a = 86 and b = 14, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		86	1	0
1		14	0	1
2	86 / 14 = 6	86 – 6*14 = 2	1 – 6*0 = 1	0 – 6*1 = – 6
3	14 / 2 = 7	14 – 2*7 = 0	Do not calculate	Do not calculate

gcd(86,14) = 2

x = 1, y = – 6

To verify, let's put the above values in the equation:

ax + by = 86 * 1 – 14 * 6

= 86 – 84

= **2**

[this is gcd value that we got for 86,14]

Ex. 2.9.7 : For a = 999 and b = 9, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		999	1	0
1		9	0	1
2	999 / 9 = 111	999 – 111*9 = 0	Do not calculate	Do not calculate

gcd(999,9) = 9

x = 0, y = 1

Let's put those values in the equation :

ax + by = 999 * 0 + 1 * 9

= 9

[which matches the gcd value we got for 999,9]

Tip : It would perhaps be easy for you to calculate the first two columns q and r until you get 0 in r. That way you are focusing on one set of calculation at a time. Once you have q and r computed in the first two columns, calculate s and t by just substitution the values of q and r in the respective equations.

2.9.4 Multiplicative Inverse using Extended Euclidean Algorithm

If you recall our discussion from the previous section on modular inverse, you understand what inverse operation is. One application of the extended Euclidean Algorithm is to find out multiplicative inverse.

Definition : *A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus m*

In modular arithmetic, it can be written as, $ax \equiv 1 \pmod m$

According to the extended Euclidean Algorithm,

ax + my = gcd(a, m) = 1

ax + my = 1

ax – 1 = (– y)m

Dividing both sides by (mod m)

ax (mod m) – 1 (mod m) = (– y)m (mod m)

ax (mod m) – 1 (mod m) = 0

ax ≡ 1 (mod m)

Note : The multiplicative inverse of a modulo m exists if and only if a and m are coprime (i.e. if gcd(a, m) = 1)

So, if you come across a question where it is asked to calculate multiplicative inverse such that gcd(a, m) is not 1, do not attempt to solve the problem. Just calculate the gcd and show that the numbers are not coprime and hence the multiplicative inverse does not exist.

Ex. 2.9.8 : Find multiplicate inverse of 24140 mod 40902.

Soln. :

gcd(24140, 40902) = gcd(24140, 40902 mod 24140)

= gcd(24140, 16762)

= gcd(24140 mod 16762, 16762)

= gcd(7378, 16762)

= gcd(7378, 16762 mod 7378)

= gcd(7378, 2006)

= gcd(7378 mod 2006, 2006)

= gcd(1360, 2006)

= gcd(1360, 2006 mod 1360)

= gcd(1360 mod 646, 646)

= gcd(68, 646 mod 68)

= gcd(68 mod 34, 34)

= gcd(0, 34)

gcd(24140, 40902) = **34**

Here you find that gcd(24140, 40902) = 34. Hence, multiplicative inverse of 24140 mod 40902 does NOT exist.

Note : You can also calculate gcd using tabular method as you learnt in the extended Euclidean algorithm section to avoid repeating the gcd steps to calculate x and y if gcd = 1 does exist.

Ex. 2.9.9 : Find multiplicate inverse of 8 mod 11.

Soln. :

Since 8 < 11, let's swap the values for simplicity.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		11	1	0
1		8	0	1

Calculate next steps :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		11	1	0
1		8	0	1
2	11 / 8 = 1	11 – 8*1 = 3	1 – 1*0 = 1	0 – 1*1 = – 1
3	8 / 3 = 2	8 – 3*2 = 2	0 – 2*1 = – 2	1 – 2*– 1 = 3
4	3 / 2 = 1	3 – 2*1 = 1	1 – 1*– 2 = 3	– 1 – 1*3 = – 4
5	2 / 1 = 2	2 – 1*2 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

Hence, x = – 4 and y = 3 Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$8(-4) + 11(3) = 1$$

Since, you have to find multiplicative inverse in mod 11, divide both sides by mod 11.

$$8(-4) \bmod 11 + 11(3) \bmod 11 = 1 \bmod 11$$

$$8(-4) \bmod 11 + 0 = 1$$

Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$$-4 + 11 = 7$$

$$7 \bmod 11 = 7$$

Hence, $8(7) \bmod 11 = 1$

So, multiplicative inverse of 8 mod 11 is 7.

Note: You can also test your solution. If there are mistakes, re-visit the steps you took. In this example, 8*7 = 56 and 56 mod 11 = 1. Hence, you find that 7 is indeed multiplicative inverse of 7 in mod 11.

Ex. 2.9.10 : Find multiplicate inverse of 1234 mod 4321.

Soln. :

Since 1234 < 4321, let's swap the values for simplicity.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		4321	1	0
1		1234	0	1
2	4321 / 1234 = 3	4321 – 3*1234 = 619	1 – 3*0 = 1	0 – 3*1 = – 3
3	1234 / 619 = 1	1234 – 1*619 = 615	0 – 1*1 = – 1	1 – 1* – 3 = 4
4	619 / 615 = 1	619 – 1*615 = 4	1 – 1* – 1 = 2	– 3 – 1*4 = –7
5	615 / 4 = 153	615 – 4*153 = 3	– 1 – 153*2 = – 307	4 – 153*– 7 = 1075
6	4 / 3 = 1	4 – 1*3 = 1	2 – 1* – 307 = 309	–7 – 1*1075 = – 1082
7	3 / 1 = 3	3 – 3*1 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

Hence, x = – 1082 and y = 309

Putting it in the equation,

$$ax + by = 1$$

$$1234(-1082) + 4321(309) = 1$$

Dividing both sides by mod 4321, you get

$$1234 (-1082) \bmod 4321 + 4321 (309) \bmod 4321 = 1 \bmod 4321$$

$$1234 (-1082) \bmod 4321 + 0 = 1$$

Convert –1082 to positive

$$-1082 + 4321 = 3239$$

$$3239 \bmod 4321 = 3239$$

Hence,

$$1234 (3239) \bmod 4321 = 1$$

Or 3239 is multiplicative modular inverse of 1234 in mod 4321.

2.10 Public Key Cryptography

You learnt about the use of asymmetric keys earlier. Recall and revise that section before you proceed.

Public key cryptography relies on the use of such asymmetric keys for providing various cryptographic services such as encryption and digital signature.

✍ ***Definition :*** *Public key cryptography is a cryptographic scheme that uses two mathematically related keys, a public key and a private key, for providing various cryptographic services.*

2.10.1 Principles of Public Key Cryptosystems

Following are some basic principles of public key-based cryptosystems.

1. Public key cryptosystems require the use of two keys – a public key and a private key.
2. Public keys are widely known.
3. Private key is kept secret with its owner.
4. The two keys are mathematically related and form a key pair.
5. One key in the key pair cannot be used to derive the other key in the key pair.
6. Any key in the key pair can be used for encryption. The other key then must be used for decryption.

7. Sender and the receiver both must have their own key pairs.

In this section, you are going to learn about various asymmetric key based algorithms.

2.10.2 RSA Algorithm

RSA, named after its inventors Ron Rivest, Adi Shamir and Leonard Adleman, is an asymmetric key based algorithm. As you understand, RSA, or any other asymmetric key based algorithms can be used for confidentiality [encryption, decryption], authentication and non-repudiation.

RSA is based on finding prime factors for very large numbers. The length of numbers that we are referring to here is around 500 digits!

Sr. No.	RSA Key Length	Number of digits
1.	1024-bit	309
2.	2048-bit	617
3.	4096-bit	1233

Let's understand how RSA derives public and private keys and how does encryption and decryption process work based on the derived keys.

1. Choose two random large prime numbers, 'p' and 'q'
2. Multiply the numbers, $n = p*q$
3. Choose a random integer to be encryption key 'e' such that 'e' and $(p-1)(q-1)$ are relatively prime.
4. Decryption key is computed as $d = e^{-1} \bmod ([p-1]*[q-1])$
5. The public key = (n, e)
6. The private key = (n, d)
7. For encrypting message 'M' with public key (n, e), you get ciphertext $C = M^e \bmod n$.
8. For decrypting ciphertext with private key (n, d), you get plaintext $M = C^d \bmod n$.

Ex. 2.10.1 : Perform encryption and decryption using RSA algorithm with p = 7, q = 11, e = 17 and M = 8.

Soln. :

$$n = p * q$$

$$n = 7 * 11 = 77$$

$$r = (p-1) * (q-1)$$

$$r = 6*10 = 60$$

$$d = e^{-1} \bmod r$$

$$ed \equiv 1 \bmod 60$$

$$17d \equiv 1 \bmod 60$$

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 17 and 60)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		60	1	0
1		17	0	1
2	60 / 17 = 3	60 – 3*17 = 9	1 – 3*0 = 1	0 – 3*1 = – 3
3	17 / 9 = 1	17 – 1*9 = 8	0 – 1*1 = – 1	1 – 1*– 3 = 4
4	9 / 8 = 1	9 – 1*8 = 1	1 – 1*– 1 = 2	– 3 – 1*4 = – 7
5	8 / 1 = 8	8 – 1*8 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$$x = -7$$

$$y = 2$$

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$17(-7) + 60(2) = 1$$

Since, you have to find multiplicative inverse in mod 60, divide both sides by mod 60.

$$17(-7) \bmod 60 + 60(2) \bmod 60 = 1 \bmod 60$$

$$17(-7) \bmod 60 + 0 = 1$$

Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$$-7 + 60 = 53$$

$$53 \bmod 60 = 53$$

Hence, 17(53) mod 60 = 1

Hence, value of decrypting key, d = 53

Now, you have all the values needed for encryption and decryption.

As per RSA,

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

$$C = 8^{17} \bmod 77$$

$$C = 57$$

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

$$M = 57^{53} \bmod 77$$

$$M = 8$$

Ex. 2.10.2 : Given modulus n=91 and public key, e=5, find the values of p, q, phi(n), and d using RSA. Encrypt M=25. Also perform decryption. **MU - Dec. 18, 10 Marks**

Soln. :

Since, n is 91, assuming p and q were 7 and 13 respectively. (by factorizing 91).

$$r = (p-1) * (q-1)$$

$$r = 6*12 = 72$$

phi(n) can be calculated using the formula

$$phi(n) = n \times \left(1 - \frac{1}{7}\right) \times \left(1 - \frac{1}{13}\right)$$

[where 7 and 13 are prime factors of 91]

Hence, $phi(n) = 72$

$$d = e^{-1} \bmod r$$

$$ed \equiv 1 \bmod 72$$

$$5d \equiv 1 \bmod 72$$

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 5 and 72)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		72	1	0
1		5	0	1
2	72 / 5 = 14	72 – 5 * 14 = 2	1 – 0 * 14 = 1	0 – 1 * 14 = -14
3	5 / 2 = 2	5 – 2 * 2 = 1	0 – 1 * 2 = -2	1 – -14 * 2 = 29
4	2 / 1 = 2	2 – 1 * 2 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$$x = 29$$

$$y = -2$$

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$5 \times 29 + 72 \times (-2) = 1$$

Since, you have to find multiplicative inverse in mod 72, divide both sides by mod 72.

$$5 \times 29 \bmod 72 + 72 \times (-2) \bmod 72 = 1 \bmod 72$$

$$5 \times 29 \bmod 72 + 0 = 1 \bmod 72$$

$$5 \times 29 \bmod 72 = 1 \bmod 72$$

Hence, multiplicative is 29.

Therefore, the value of decrypting key, $d = 29$.

Now, you have all the values needed for encryption and decryption.

As per RSA,

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

$$C = 25^5 \bmod 91$$

$$\mathbf{C = 51} \text{ [encrypted message]}$$

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

$$M = 51^{29} \bmod 91$$

$$\mathbf{M = 25} \text{ [decrypted message]}$$

Ex. 2.10.3 : Given modulus n=221 and public key, e=7, find the values of p, q, phi(n), and d using RSA. Encrypt M = 5. **MU - May 19, 10 Marks**

Soln. :

Since, n is 221, assuming p and q were 13 and 17 respectively. (by factorizing 221).

$$r = (p-1) \times (q-1)$$

$$r = 12 \times 16 = 192$$

phi(n) can be calculated using the formula

$$phi\ (n) = n \times \left(1 - \frac{1}{13}\right) \times \left(1 - \frac{1}{17}\right)$$

Hence, $phi\ (n) = 192$

$$d = e^{-1} \bmod r$$

$$ed \equiv 1 \bmod 192$$

$$7d \equiv 1 \bmod 192$$

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 7 and 192)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		192	1	0
1		7	0	1
2	192 / 7 = 27	192 – 7 * 27 = 3	1 – 0 * 27 = 1	0 – 1*27 = – 27
3	7 / 3 = 2	7 – 3 * 2 = 1	0 – 1*2 = **– 2**	1 – (-27)*2 = **55**
4	3 / 1 = 3	3 – 1 *3 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$$x = 55$$

$$y = -2$$

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$7 \times 55 + 192 \times (-2) = 1$$

Since, you have to find multiplicative inverse in mod 192, divide both sides by mod 192.

$$7 \times 55 \bmod 192 + 192 \times (-2) \bmod 192 = 1 \bmod 192$$

$$7 \times 55 \bmod 192 - 0 = 1 \bmod 192$$

Hence, multiplicative is 55.

Therefore, the value of decrypting key, d = 55.

Now, you have all the values needed for encryption and decryption.

As per RSA,

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

$$C = 5^7 \bmod 192$$

$$\mathbf{C = 173}\ \text{[encrypted message]}$$

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

$$M = 173^{55} \bmod 192$$

$$\mathbf{M = 5}\ \text{[decrypted message]}$$

2.10.2(A) Attacks on RSA

1. **Brute-force Attack :** Here the attacker tries to find factors of 'n' by trying out various possibilities.
2. **Common Modulus**

– To avoid generating a different modulus n = p * q for each user one may wish to fix 'n' for all the users. It might seem like deriving the decrypting key 'd' is not possible for every encrypting key e by any other user since encrypting key 'e' is a randomly chosen value. But, the problem with this approach is that a particular user who knows her pair of 'e' and 'd' can successfully use her own pair to find the factors for common modulus.

– Once the factors are known, since the encrypting key 'e' is known to everyone (because it is public key), the decrypting key 'd' could be found out. Hence, you should not be using common modulus to generate keys for multiple users.

3. **Choosing Smaller Numbers :** The security of the algorithm comes from the fact that factoring large numbers is computationally intensive. Sometimes, to improve system performance, smaller numbers can be chosen which can significantly enhance the performance but at the cost of making the algorithm weaker. Hence, you should always choose large numbers to maintain the strength of the algorithm.

4. **Man in the Middle Attack :** The attacker could collect all the ciphertext coming out from the user's system (that is encrypted with her private key) and try to find the private key. The information known to the attacker is the ciphertext and public key.

2.10.3 Knapsack Algorithm

✍ ***Definition :*** *The Merkle – Hellman Knapsack algorithm is an asymmetric key based algorithm used for encryption and decryption.*

It was invented by Ralph Merkle and Martin Hellman in 1978.

2.10.3(A) Major Attributes of the Knapsack Algorithm

- It is based on public key cryptography meaning that it requires two keys – public and private
- Unlike RSA, it is one way. Public key is used for encryption and private key is used for decryption
- Hence, it cannot be used for digital signature and non-repudiation
- It is outdated and is not used anymore

2.10.3(B) Algorithm

Knapsack means a bag. The basic Knapsack problem aims at maximizing the weight of the knapsack taking values from a set of available weights.

✍ ***Definition :*** *Given a Knapsack of a maximum capacity of W and N items each with its own value and weight,* ***choose items*** *to place inside the Knapsack such that the final contents in the knapsack has the maximum value.*

In Merkle–Hellman Knapsack Algorithm, the keys are two knapsacks.

- The public key is a 'hard' knapsack A, and
- The private key is an 'easy', or super-increasing, knapsack B.

Two numbers, a multiplier and a modulus, can be used to convert the super-increasing knapsack B into the hard knapsack A.

So,

- Given a set of numbers *A* and a number *b*
- Find a subset of *A* which sums to *b*
- So, if you have a set of weights of 1, 4, 6, 8 and 15, and you want to get a weight of 29, you could thus use 6, 8 and 15 (6 + 8 + 15 = 29). So, the code would become 00111 (represented by not picking '1', not picking '4', picking '6', picking '8' and picking '15').

Plaintext	10001	11001	00001	00101
Knapsack	1,4,6,8,15	1,4,6,8,15	1,4,6,8,15	1,4,6,8,15
Ciphertext	1+15 = 16	1+4+15 = 20	15	6+15 = 21

2.10.4 ElGamal Algorithm

Q. Write short note on ElGamal Algorithm.

MU - May 19, 5 Marks

ElGamal is a public key algorithm that can be used for digital signatures, encryption, and key exchange. It is based on Diffie-Hellman algorithm. Unlike other asymmetric algorithms that are based on factoring large prime numbers, it is based on calculating discrete logarithms in a finite field. Although, El Gamal provides the same type of functionality as most of the other asymmetric algorithms, its main drawback is its slow performance.

ElGamal encryption consists of three components :

1. Key generation,
2. Encryption algorithm, and
3. Decryption algorithm

2.10.4(A) Key Generation

1. Select a large random prime p and a generator g.
2. Generate a random integer x such that $1 \le x \le p - 2$.
3. Compute $y = g^x \bmod p$
   (a) Public Key is (p, g, y)
   (b) Private Key is x

2.10.4(B) Encryption

Given a message m such that $0 \leq m < p$, then user Bobby can encrypt m as below:

1. Pick an integer k between 1 and p – 2
2. Compute the first component of the ciphertext $c_1 = g^k \bmod p$
3. Compute the second component of the ciphertext $c_2 = y^k * m \pmod p$
4. The ciphertext is the pair (c_1, c_2)

2.10.4(C) Decryption

User Alice can decrypt (c_1, c_2) as original message $m = c_1^{p-1-x} * c_2 \pmod p$

Let's demonstrate the ElGamal algorithm with a practice question.

Ex. 2.10.4 : Use initial values of p = 7, g = 5 and x = 2 and demonstrate ElGamal algorithm. Use other values as you prefer for your demonstration.

Soln. : Assume that the given values are for user Alex.

$$p = 7$$

$$g = 5$$

$$x = 2$$

Calculate Alex's keys

$$y = g^x \bmod p$$

$$y = 5^2 \bmod 7$$

$$y = 25 \bmod 7$$

$$y = 4$$

Alex's Public Key = (p, g, y) = (7, 5, 4)

Alex's Private Key = x = 2

Encrypt Message

Similarly, assume that the user Bobby's Private Key is $k = 4$ and the message she wants to send is $m = 6$.

She encrypts the message as

$$c_1 = g^k \bmod p$$

$$c_1 = 5^4 \bmod 7$$

$$c_1 = 625 \bmod 7$$

$$c_1 = 2$$

$$c_2 = y^k * m \pmod p$$

$$c_2 = 4^4 * 6 \pmod 7$$

$$c_2 = 1536 \bmod 7$$

$$c_2 = 3$$

Hence, Bobby sends ciphertext$(c_1, c_2) = (2, 3)$ to Alex.

Decrypt Message

Now, Alex wants to decrypt Bobby's message that she sent as ciphertext$(2, 3)$

$$m = c_1^{p-1-x} * c_2 \pmod p$$

$$m = 2^4 * 3 \pmod 7$$

$$m = 48 \bmod 7$$

$$m = 6$$

Hence, Alex could get the encrypted message from Bobby and could use his private key to decrypt the message and read it.

2.11 Key Management Techniques

Now that you understand various cryptographic methods and procedures, let us focus on the keys. As you know, keys play the critical role in delivering several cryptographic services. Without adequate and protected methods of key generation, distribution, and disposal, the entire cryptosystem could collapse, and you might not be able to use any cryptographic services.

2.11.1 What is a Key?

What is a Key? As per National Institute of Standards and Technology (NIST) :

✍ ***Definition :*** *A key is a parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce, reverse or verify the operation, while an entity without knowledge of the key cannot.*

Let us recall the keys used in various cryptographic services assuming that there are two parties involved in the communication and they both wish to use cryptographic services.

Cryptographic Service	Keys used
Symmetric Key Encryption / Decryption	One symmetric key
Asymmetric Key Encryption / Decryption	Two Public Private Key Pairs
Hashing	No Keys
MACs	One symmetric Key
Digital Signature	Two Public Private Key Pairs

2.11.2 Key States

It is important for you to understand that the keys could be in various states and can transition from one state to another state based on circumstances and scenarios.

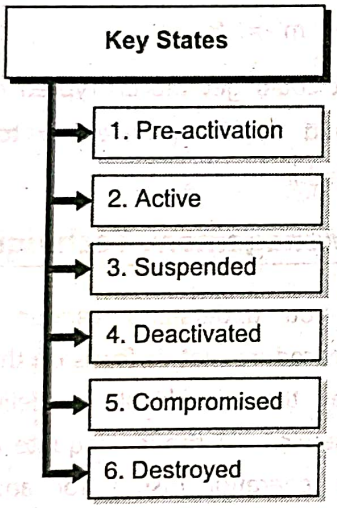


Fig. 2.11.1 : Key States

1. **Pre-activation state :** In this state, the key is generated but is not yet in use. This state is reached as soon as the key is generated.
2. **Active state :** In this state, the key can be used to provide cryptographic services.
3. **Suspended state :** The key in this state is restricted temporarily for use. The key can be transitioned to the active, de-active or destroyed state based on various criteria.
4. **Deactivated state :** In this state, the key is not used to carry out new cryptographic services. However, it may be used for previously consumed cryptographic services. For example, a deactivated key may not be used for encrypting new information but can still be used for decrypting previous information that it was used to encrypt when it was in the active state.
5. **Compromised state :** In this state, the key is exposed, and its secrecy can no more be guaranteed. Once a key is determined to be in this state, it is strongly advisable to suspend its use, carry out investigation to find the root cause of exposure and then destroy it.
6. **Destroyed state :** In this state, the key is safely disposed. The key is no more available to carry out any form of cryptographic services.

2.11.3 Cryptoperiod (Key lifetime)

All cryptographic keys come with a validity period. Some keys are short-lived, and some are long-lived.

✍ ***Definition :*** *A cryptoperiod is the time span during which a specific key is authorised for use by legitimate entities, or the keys for a given system will remain in effect.*

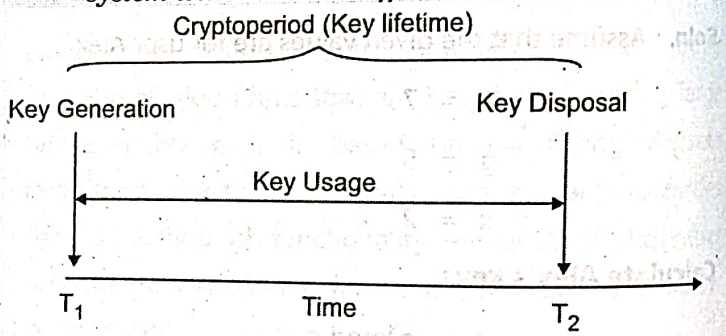


Fig. 2.11.2 : Cryptoperiod (Key Lifetime)

- Master Keys are usually long-lived. They are not used for providing cryptographic services themselves. Instead, they are used to periodically generate session keys.
- Session Keys are usually short-lived (up to a session). These are generated from the master keys and are used for providing cryptographic services such as encryption and decryption. These are frequently generated and destroyed.

2.11.4 Key Management Principles

Following are some general principles that you should follow for key management.

1. **Adequate key length :** The key length should be long enough to provide the necessary level of protection.
2. **Secure transmission :** The keys should be stored and transmitted by secure means.
3. **Difficult to guess :** The keys should be random, and hard to guess.
4. **Adequate Cryptoperiod :** The key's lifetime should correspond with the sensitivity of the data it is protecting. Low sensitive data may allow for a longer key lifetime, whereas more sensitive data might require a shorter key lifetime. Also, the more the key is used, the shorter its lifetime should be.
5. **Backup :** The keys should be backed up in case of emergencies.
6. **Adequate key disposal :** The keys should be properly destroyed when they expire.

2.11.5 Symmetric Key Distribution

Symmetric key distribution requires that the communication parties have the exact same key. The same key is used for cryptographic services between the parties.

Ways to distribute symmetric keys

Assume that there are two communicating parties – *A* and *B*. The symmetric key can be distributed in the following ways :

1. A and B can physically generate and distribute the key amongst each other.
2. Any third party C can physically generate and distribute the key to both A and B.
3. A and B can get connected to a key distribution center and get the key.

In reality, distributing the keys physically is infeasible. Hence, a third party, generally known as the Key Distribution Center, is tasked to distribute the keys amongst the communicating parties.

The symmetric key distribution can be achieved in two ways.

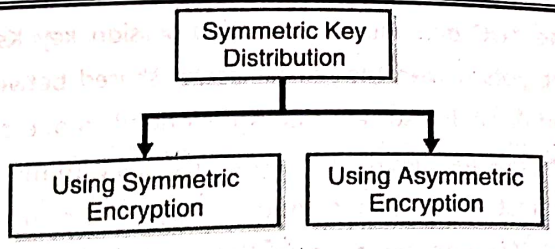


Fig. 2.11.3 : Symmetric Key Distribution

Let's learn about both of them.

2.11.5(A)Symmetric Key Distribution using Symmetric Encryption

Let's assume the following.

- *A* and *B* are the two communicating parties.
- There is a **trusted third party** known as Key Distribution Center (KDC) denoted by *K*.
- *A* and *K* have a shared master key *Ka*.
- Similarly, *B* and *K* have a shared master key *Kb*.
- *Na* is a random number called nonce generated by *A* to avoid replays.
- *Ida* is the unique identifier for the party *A*.
- *IDb* is the unique identifier for the party *B*.
- *Ks* is the desired session key that needs to reach both *A* and *B* so that they can securely communicate.

The steps for key distribution are as following.

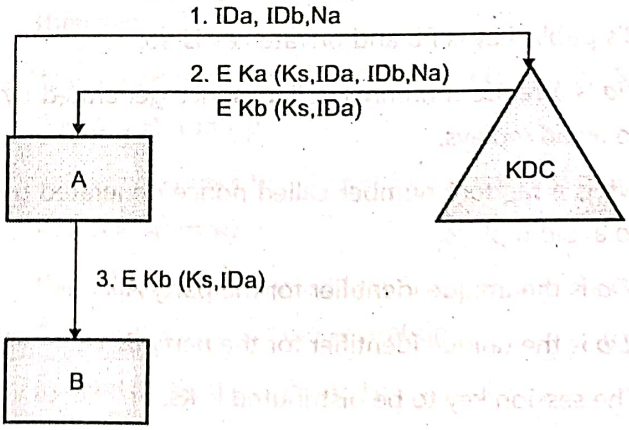


Fig. 2.11.4 : Symmetric Key Distribution using Symmetric Encryption

1. The communicating party A initiates a request with KDC by identifying itself as IDa and identifying the target party as IDb. It sends a nonce Na to ensure that the request is protected against the replay attacks.

2. The KDC generates the desired session key Ks and encrypts it with the master secret shared between it and A, Ka. It also sends back the original request from A to ensure that the response matches with the initial request. It also sends another copy of session key Ks and IDa encrypted using the master secret shared between it and B, Kb. This copy can only be decrypted by B.
3. A verifies the information received from KDC, stores the session key Ks and forwards the copy intended for B. B decrypts the copy, stores the session key Ks and gets to know that the communicating party is A with whom the received session key is to be used.

This way, both *A* and *B* receive the session key Ks and can then begin secure communication.

2.11.5(B) Symmetric Key Distribution using Asymmetric Encryption

Using symmetric key encryption to distribute symmetric keys requires KDC. Using asymmetric encryption for key distribution requires the use of public and private key pairs.

Let's assume the following.

- *A* and *B* are the two communicating parties.
- A's public key is Pa and private key is Sa.
- B's public key is Pb and private key is Sb.
- *Na* is a random number called nonce generated by *A* to avoid replays.
- *Nb* is a random number called nonce generated by *B* to avoid replays.
- *IDa* is the unique identifier for the party *A*.
- *IDb* is the unique identifier for the party *B*.
- The session key to be distributed is Ks.

The steps for key distribution are as following.

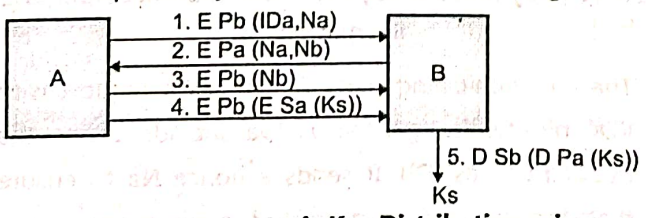


Fig. 2.11.5 : Symmetric Key Distribution using Asymmetric Encryption

1. A sends its identity IDa and a nonce Na encrypted with B's public key Pb.
2. B decrypts the message received from A and sends back Na and Nb encrypted with A's public key.
3. A decrypts the message received from B using her private key Sa, verifies Na, encrypts Nb with Pb and sends it back to B.
4. At this point, both A and B are authenticated to each other. A generates the session key Ks encrypts it with her private key Sa and then encrypts it again with B's public key Pb and sends it to B.
5. B first decrypts the message using her private key Sb and then decrypts the message again using A's public key Pa. B now gets the session key Ks and knows that only A could have used her private key to send the session key to her.

2.11.6 Asymmetric Key Distribution (Distribution of Public Keys)

Asymmetric keys involve public keys and private keys. Public keys are known in general. Private keys are kept secure and are in constant possession of their respective owners. The goal of asymmetric key distribution is to distribute public keys only. Private keys are not distributed.

Ways to distribute asymmetric keys

Asymmetric keys can be distributed using the following ways.

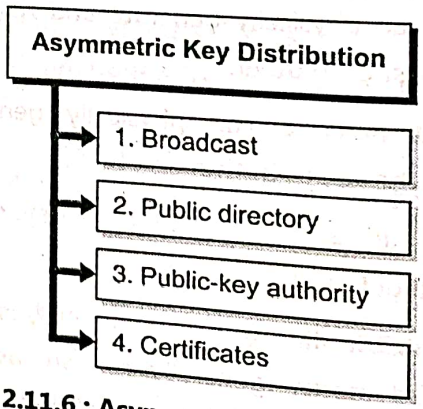


Fig. 2.11.6 : Asymmetric Key Distribution

1. **Broadcast :** Broadcast or public announcement of public keys is the simplest of all the distribution techniques. In this, the public keys are constantly

broadcasted (shared with the larger user community). As new users get into the community, they hear such key broadcast messages and store the public keys to use if and when required.

2. **Public directory :** A centralized and publicly open directory (very much like an old telephone directory) could be established to register and distribute public keys. Users can register new keys, change previous keys or delete the keys if required. The company maintaining the directory could exercise control to ensure that the directory is highly available, and the key information contained there-in is not destroyed or maligned.
3. **Public-key authority :** In Public-key authority, the central authority itself has a public and private key pair. When a user asks for the public key of the desired target user, the authority encrypts the public key information of the desired user with its own private key to securely distribute the public key.
4. **Certificates :** Certificate is the most widely used technique to distribute public keys in the industry today. A certificate typically consists of a public key, an identifier of the key owner, and is signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a corporate institution, that is trusted by the user community. You will learn about certificates in greater detail in the upcoming section.

2.11.7 Key Management using Diffie Hellman Key Exchange

Q. Write short note on Diffie Hellman algorithm.

MU - May 19, 5 Marks

If you recall, one of the challenges with using symmetric key algorithms was key distribution. How could sender and the receiver agree upon the key that would be used for encryption and decryption (recall that the symmetric key based algorithms use the SAME key for both encryption as well as decryption)? I asked you this question in the symmetric key section and here I am again to help you with the answer. Diffie-Hellman algorithm is one of the answers to the question.

✍ ***Definition :*** *The Diffie-Hellman algorithm provides a way of generating a shared secret between the sender and the receiver in such a way that the secret need not be exchanged or transferred over the communication medium.*

Basically, the sender and the receiver create the key together at their respective ends at the same time. The key, that they create at their respective ends, is mathematically computed to be the same. Hence, the key distribution need not happen, and the sender and the receiver can confidentially communicate using the key they created. You should note here that the Diffie-Hellman algorithm is NOT used for actual encryption and decryption process. It is used only for key generation.

Let's understand the steps involved in generating the shared key. Assume that there are two users Alex and Bobby who need to generate a shared key for securely communicating with each other.

1. Alex chooses two prime numbers 'g' and 'p' and also a secret number 'a'. He calculates value of 'A' such that $\mathbf{A = g^a \bmod p}$. He then sends 'g', 'p' and 'A' to Bobby. Note here that Alex does not share the secret number 'a' with Bobby.
2. Similarly, Bobby chooses a secret number band computes the value of B such that $\mathbf{B = g^b \bmod p}$. She then sends 'B' to Alex.
3. Alex computes the shared key at his end as Shared Key, $\mathbf{S = B^a \bmod p}$
4. Bobby computes the shared key at her end as Shared Key, $\mathbf{S = A^b \bmod p}$

The values of the shared key, 'S' derived in the step 3 and 4 are equal due to mod operation.

$$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$$
$$(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$$

It does not matter which step you do earlier. Both the keys created would be equal and can be used with any of the algorithms such as DES and AES to encrypt the information and communicate confidentially.

Let's understand the algorithm by solving a question.

Ex. 2.11.5 : Calculate shared key between two users if the initial chosen prime numbers are 5 and 7.

Soln. :

$g = 5$

$p = 7$

Assume that the user Alex chooses a secret number a = 2

$A = g^a \bmod p$

$A = 5^2 \bmod 7$

$A = 25 \bmod 7$

$A = 4$

Alex sends g, p and A to Bobby.

Assume that the user Bobby chooses a secret number b = 3

$B = g^b \bmod p$

$B = 5^3 \bmod 7$

$B = 125 \bmod 7$

$B = 6$

Bobby sends B to Alex

Now, both the users compute the shared key, S, at their respective ends. Alex calculates it as

$S = B^a \bmod p$

$S = 6^2 \bmod 7$

$S = 36 \bmod 7$

$S = 1$

Bobby calculates it as

$S = A^b \bmod p$

$S = 4^3 \bmod 7$

$S = 64 \bmod 7$

$S = 1$

So, the shared key, that can be used between Alex and Bobby, is S = 1.

Note: The example we took here involves very small numbers to make it easy for you to understand the steps involved in the key generation. Practically, the numbers used in the shared key generation are extremely large, possibly containing around 500 digits!

Ex. 2.11.6 : Given generator g= 2 and n= 11. Using Diffie Hellman algorithm solve the following :

1. Show that 2 is primitive root of 11
2. If A's public key is 9, what is A's private key?
3. If B's public key is 3, what is B's private key?
4. Calculate the shared secret key.

MU - Dec. 18, 10 Marks

Soln. :

1. Let's assume that 2 is a primitive root of 11. Then, powers of 2 mod 11 should produce all the integers from 1 to (11 – 1) which means it should produce {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

Integer	Mod operation	Resultant integer
2^0	1 mod 11	1
2^1	2 mod 11	2
2^2	4 mod 11	4
2^3	8 mod 11	8
2^4	16 mod 11	5
2^5	32 mod 11	10
2^6	64 mod 11	9
2^7	128 mod 11	7
2^8	256 mod 11	3
2^9	512 mod 11	6

You see that 2 is indeed a primitive root of 11. It produces all the numbers from 1 to 10 as expected.

1. $g = 2$

$p = 11$

For user A,

Public key YA = 9 which means

$A = g^x \bmod p$

$9 = 2^x \bmod 11$

From the primitive root table above, $2^6 \bmod 11$ gives 9.

Hence, the private key of user A is 6.

2. For user B,

Public key YB = 3 which means

$$B = g^y \bmod p$$

$$3 = 2^y \bmod 11$$

From the primitive root table above, $2^8 \bmod 11$ gives 3.

Hence, the private key of user B is 8.

Now, both the users compute the shared key, S, at their respective ends.

User A calculates it as

$$S = B^x \bmod p$$

$$S = 3^6 \bmod 11$$

$$S = 3$$

User B calculates it as

$$S = A^y \bmod p$$

$$S = 9^8 \bmod 11$$

$$S = 3$$

Hence, the shared key between User A and User B is 3.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Block Ciphers

Q. 1 Write a short note on block ciphers. **(4 Marks)**

Q. 2 List the design principles of block ciphers. **(6 Marks)**

Q. 3 What is Data Encryption Standard (DES)? List its major attributes. **(6 Marks)**

Q. 4 Draw a block diagram of DES and explain how it works. **(8 Marks)**

Q. 5 Write a short note on Electronic Code Book (ECB) Mode of Block Cipher operation. **(4 Marks)**

Q. 6 Write a short note on Cipher Block Chaining (CBC) Mode of Block Cipher operation. **(4 Marks)**

Q. 7 Write a short note on Cipher Feedback (CFB) Mode of Block Cipher operation. **(4 Marks)**

Q. 8 Write a short note on Output Feedback (OFB) Mode of Block Cipher operation. **(4 Marks)**

Q. 9 Write a short note on Counter (CTR) Mode of Block Cipher operation. **(4 Marks)**

Q. 10 Compare various modes of Block Cipher operations. **(8 Marks)**

Q. 11 Why is DES not recommended to be used today? **(4 Marks)**

Q. 12 List the weaknesses in DES. **(4 Marks)**

Q. 13 Write a short note on Double DES. **(6 Marks)**

Q. 14 Write a short note on Triple DES. **(6 Marks)**

Q. 15 Write a short note on 3DES. **(6 Marks)**

Q. 16 What is AES? List its major attributes. **(4 Marks)**

Q. 17 List the evaluation criteria used by NIST for selecting AES algorithm. **(6 Marks)**

Q. 18 Draw the block diagram of AES and explain its working. **(8 Marks)**

Q. 19 Compare AES and DES. Which one would you use and why? **(8 Marks)**

Q. 20 What is Blowfish? List its major attributes. **(6 Marks)**

Q. 21 Explain key expansion and subkey generation in Blowfish. **(8 Marks)**

Q. 22 Describe strengths and weaknesses of Blowfish. **(4 Marks)**

Q. 23 Describe RC5 Algorithm and its major attributes. **(6 Marks)**

Q. 24 Write the pseudo code for encryption using RC5 algorithm. **(6 Marks)**

Q. 25 Describe the various attacks possible on cryptosystems. **(8 Marks)**

Q. 26 Compare differential and linear cryptanalysis. **(8 Marks)**

Public Key Cryptography

Q. 27 List the principles of public key cryptosystems. **(6 Marks)**

Q. 28 Describe RSA algorithm and its steps. **(6 Marks)**

Q. 29 With an example of your choice, demonstrate RSA algorithm. **(8 Marks)**

Q. 30 Perform encryption and decryption using RSA algorithm with p = 7, q = 11, e = 17 and M = 8. **(8 Marks)**

Q. 31 Describe a few attacks on RSA. **(4 Marks)**

Q. 32 What is Knapsack algorithm? List its major attributes. **(4 Marks)**

Q. 33 With an example, explain Knapsack algorithm. **(6 Marks)**

Q. 34 Describe how ElGamal algorithm works. **(6 Marks)**

Q. 35 With an example of your choice, demonstrate how ElGamal algorithm works. **(6 Marks)**

Q. 36 Use initial values of p = 7, g = 5 and x =2 and demonstrate ElGamal algorithm. Use other values as you prefer for your demonstration. **(6 Marks)**

Q. 37 Describe the various key states. **(6 Marks)**

Q. 38 Write a short note on Cryptoperiod (Key lifetime). **(4 Marks)**

Q. 39 List the Key Management Principles. **(6 Marks)**

Q. 40 Describe Symmetric Key Distribution using Symmetric Encryption. **(6 Marks)**

Q. 41 Describe Symmetric Key Distribution using Asymmetric Encryption. **(6 Marks)**

Q. 42 Describe the ways to distribute asymmetric keys. **(4 Marks)**

Q. 43 What problem does Diffie Hellman Key exchange solve and how? **(4 Marks)**

Q. 44 With an example of your choice, demonstrate how Diffie Hellman Key exchange works. **(6 Marks)**

Q. 45 Calculate shared key between two users if the initial chosen prime numbers are 5 and 7. **(6 Marks)**

□□□