Computer Graphics

educlash.com
Just another Way To Learn

Q 1) What do you understand by Network Security and why is it required? Explain principles of Network Security. Explain various types of attacks.

NETWORK SECURITY:

Network Security is the identification and mitigation of undesirable information flow. Individuals as well as organizations are becoming increasingly dependent on networks to carry on their activities. Networks have become assets like computers, data. Without these assets most organizations, individuals find it impossible to conduct day to day work. The network asset must be protected like other assets and surrounded with proper controls and appropriate security. Network security is needed to lay down appropriate measures to prevent the network from attacks.

PRINCIPLES OF NETWORK SECURITY:

CONFIDENTIALITY:

The principle of confidentiality specifies that only the sender and the intended recipient should be able to access the contents of a message. Confidentiality gets compromised if an unauthorized person is able to access a message. Here, the user of computer A sends a message to user of computer B. Another user C gets access to this message, which is not desired and therefore defeats the purpose of confidentiality. For example a confidential email message, sent by A to B, which is accessed by C without the permission of A or B.

AUTHENTICATION:

Authentication mechanism's help establish proof of identities. The authentication process ensures the origin of an electronic message or document is correctly identified. For instance, user C posing as user A sends a funds transfer request (form A's account to C's account) to bank B. The bank might transfer the funds from A's account to C's account thinking user A has requested for the funds transfer.

INTEGRITY:

When the contents of a message are changed after the sender sends it, but before it reaches the intended recipient, we say that the integrity of a message is lost. For example, a check is issued for Rs 100. In the next account statement the same check resulted in a payment of Rs 10000. This is a case of loss of message integrity.

REPUDIATION:

There are situations where a user sends a message and later on refuses that she had sent that message. For example user A could send a funds transfer request to bank B over the internet. After the bank performs the funds transfer as per A's instruction. A could claim that she never sent the funds transfer instruction to the bank. Thus A repudiates or denies her fund's transfer instruction. The principle of non-repudiation defeats such possibilities of denying something, having done it.

ACCESS CONTROL:

The principle of access control determines who should be able to access what. For example, we should be able to specify that user A can view the records in the database, but cannot update them. However user B might be allowed to make updates as well. An access control mechanism can be setup to ensure this. Access control is broadly related to role management and rule management. Role management concentrates on the user side, whereas rule management focuses on the resources side.

AVAILABILITY:

The principle of availability states that resources should be available to authorized parties at all times. For example, due to intentional actions of an unauthorized user C, an authorized user A may not be able to contact a server computer B. This would defeat the principle of availability. These attacks are further grouped into two types passive attacks and active attacks.

PASSIVE ATTACKS:

Passive attacks are those, wherein the attacker indulges in monitoring of data transmission. The attacker aims to obtain information that is in transit. The term passive indicates that the attacker does not attempt to perform any modifications to the data. Passive attacks are harder to detect, therefore passive attacks should be prevented rather than detected. Passive attacks are further classified into two categories release of message contents and traffic analysis.

RELEASE OF MESSAGE CONTENTS:

When we send a confidential message to our friend we desire that only he/she be able to access it. Otherwise the contents of the message are released against our wishes to someone else.

TRAFFIC ANALYSIS:

If we send multiple messages to our friend, a passive attacker could try to figure out similarities between them to come up with some sort of pattern that provides some clues regarding the communication that is taking place. Such attempts of analyzing encoded messages to come up with likely patterns are the work of the traffic analysis attack.

ACTIVE ATTACKS:

Active attacks are based on modification of the original message in some manner or the creation of a false message. These attacks cannot be prevented easily. However, they can be detected with some effort and attempts can be made to recover from them. Active attacks can be subdivided into four categories masquerade, replay, modification, and denial of service.

MASQUERADE

Masquerade is caused when an unauthorized entity pretends to be another entity. Consider three users A, B, C. User C might pose as user A and send a message to user B. User B might be led to believe that the message indeed came from user A. A masquerade attack usually includes one of the other forms of active attacks. For example the attack may involve capturing user's authentication sequence(eg user id and password). Later, those details can be replayed to gain illegal access to the computer system.

REPLAY ATTACK:

In replay attack, a user captures a sequence of events or some data units and resends them. For instance, suppose user A wants to transfer some amount to user C's bank account. Both users A and C have accounts with bank B. User A might send an electronic message to bank B, requesting for the funds transfer. User C could capture this message and send a copy of the same to the bank B. Bank B would have no idea that this is an unauthorized message and would treat this as a second and different, funds transfer request from user A. User C gets benefit of funds transfer twice, once authorized, once through a replay attack.

ALTERATION OF MESSAGES:

It means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example user A sends an electronic message Transfer 1000 Rs to D's account to bank B. User C might capture this and change it to Transfer 10000 Rs to C's account.

DENIAL OF SERVICE:

In these attacks, attempt is made to prevent legitimate user's from accessing some services, which they are eligible for. For example an unauthorized user might send too many login requests to a server using random user id's one after the other in quick session, so as to flood the network and deny other legitimate users from using the network facilities.

Q 2) Explain different types of Security Policy?

Ans. Security Policy**:** A security policy is the set of decisions that collectively determines an organization attitude towards security.

A security policy defines the boundaries of acceptable behavior and what the response to violations should be.

Security policy is the set of decisions/rules & regulation written or verbally understood that collectively determines organizations posture towards security. It delimits the boundaries, specifies acceptable & non acceptable behavior dictates what is ethical and what is non-ethical. States the degree of seriousness of the offence and also mention the consequences of the actions if it is violated. It focuses on different stack holders and satisfying their requirements in an efficient way. Organisation differs in their culture,structure and strategy. Thus security policy will also differ from organization to organization.

Security policy will decide on the following issues:

- What legal course of action will you follow if attacked?
- What will be considered as a cognizable crime?
- Can anyone be sued?
- Infringing on someone else’s rights?

A security policy holds guidelines or rules within itself that need to be enforced as to ensure a secured network. A good security policy generally takes care of four key aspects:

- Affordability
- Cultural issues
- Legality
- Functionality

There are mainly four types of security policies namely:

- Confidentiality policy
- Integrity policy
- Military security policy
- Commercial policy

1.Confidentiality Policy: This type of policy is concerned with the privacy of an organization. As the name suggest,it deals with the issues related to confidentiality of a system. At times, there is certain data that is required to be kept secret.This is achieved using the norms of the confidentiality policy failing to which the data is considered to be unsecured and unprotected.

2.Integrity Policy: This type of policy is concerened with the integrity issues of an organization. The data that is to be used by organization should not be tampered in any possible way. This is taken care with the help of integrity policy which ensures that right data moves in and out of the organization.

3.Military Security Policy: It is also known as government security policy. This security policy is developed primarily to provide confidentiality. The name comes from the military's need to keep information such as the data that a troop will sail,secret. This policy also considers integrity and availiability as an important part of the policy along with confidentiality being its key issued because the compromise of this feature would be catastrophic for the organization. Unauthorize disclosure can result in penalties that include jail or fines etc.

4.Commercial Security Policy: This is a security policy developed primarily to provide integrity.The name comes from the need for commercial firm to prevent tampering with the data, because they could not survive such compromises. For example, if the confidentiality of a bank's computer is compromised, a customers account balance may be revealed. This would certainly embarrass the bank, but the loss to the bank's data integrity would lead to financially ruinous effects. This policy gives impetus on integrity along with confidentiality and availiability as its other issue.

Some integrity policies use the notion of a transaction like database specifications which require the database to be in a consistent state. Such policies are called transaction-oriented integrity security policies.

Q 3) What are different modes of algorithm? Explain

Ans- An algorithm mode is a combination of a series of the basic algorithm steps on block chipper and some kind of feedback from the previous step.

There are four important algorithm modes as follows,

I. Electronic Code Book (ECB)
II. Cipher Block chaining (CBC)
III. Cipher Feedback (CFB)
IV. Output Feedback (OFB)

The first two modes operate on block cipher, whereas the latter two modes are block cipher modes, which can be used as if they are working on stream cipher.

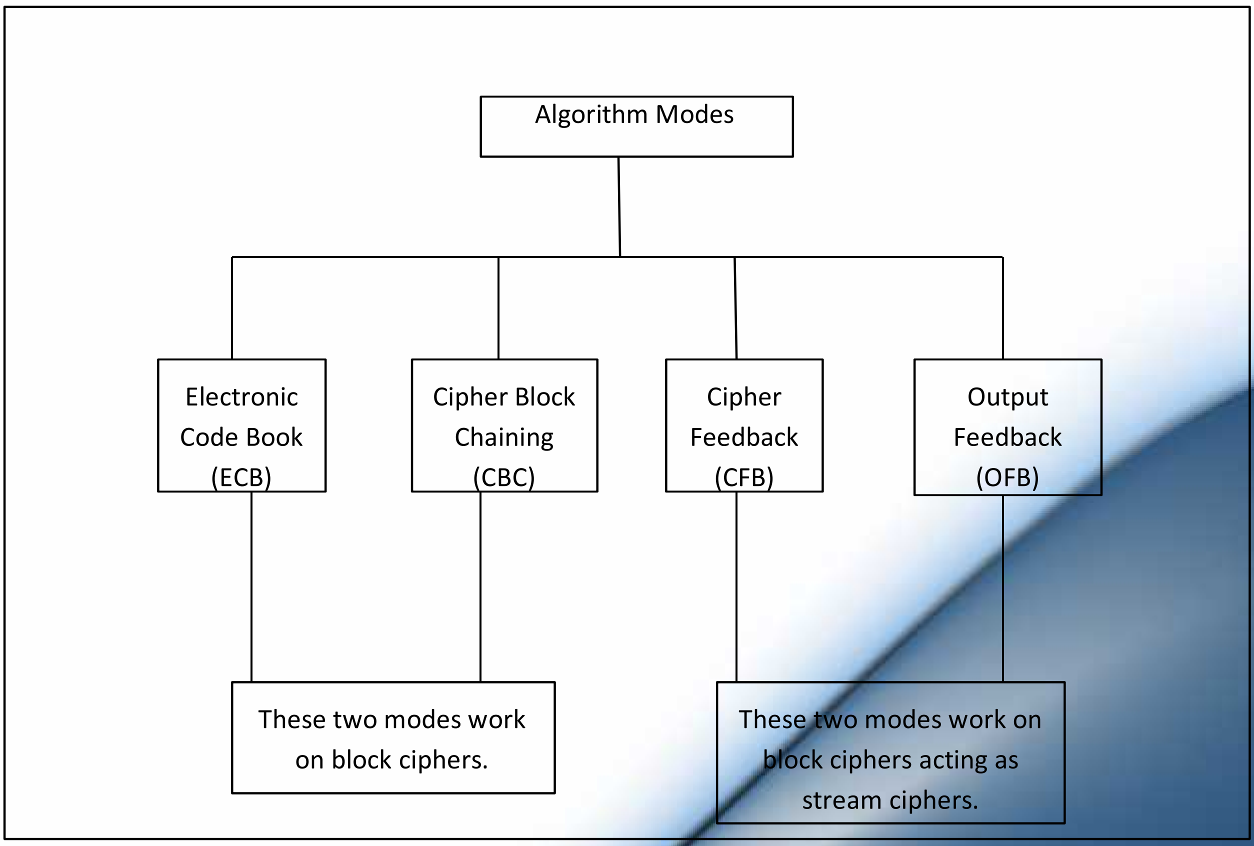

Algorithm Modes
Electronic Code Book (ECB)
Cipher Block Chaining (CBC)
Cipher Feedback (CFB)
Output Feedback (OFB)
These two modes work on block ciphers.
These two modes work on block ciphers acting as stream ciphers.

Fig. Algorithm modes

I. Electronic Code Book (ECB) mode:

It is the simplest mode of operation. Here, the incoming plain text message is divided into blocks of 64 bits each. Each such block is then encrypted independently of the other blocks. For all blocks in a message, the same key is used for encryption. The process is shown as follows:

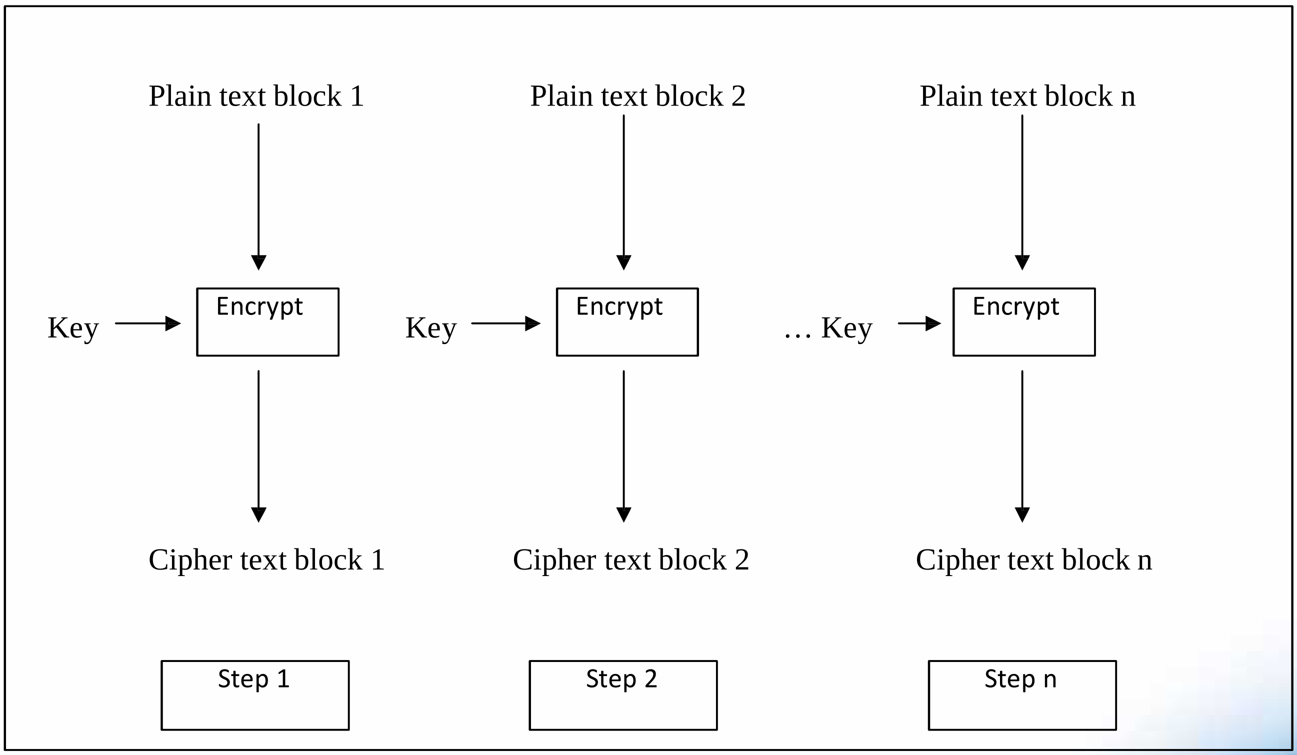


Fig. ECB mode-The encryption process

At the receiver's end, the incoming data is divided into 64 bit blocks and by using the same key as was used in encryption, each block is decrypted to produce the corresponding plain text block. The process is shown in fig.In ECB, since a single key is used for encrypting all the blocks of a message, if a plain text block repeats in the original message, the corresponding cipher text block will also repeat in the encrypted message. Therefore, ECB is suitable only for encrypting small messages, where the scope for repeating the same plain text block is quite less.

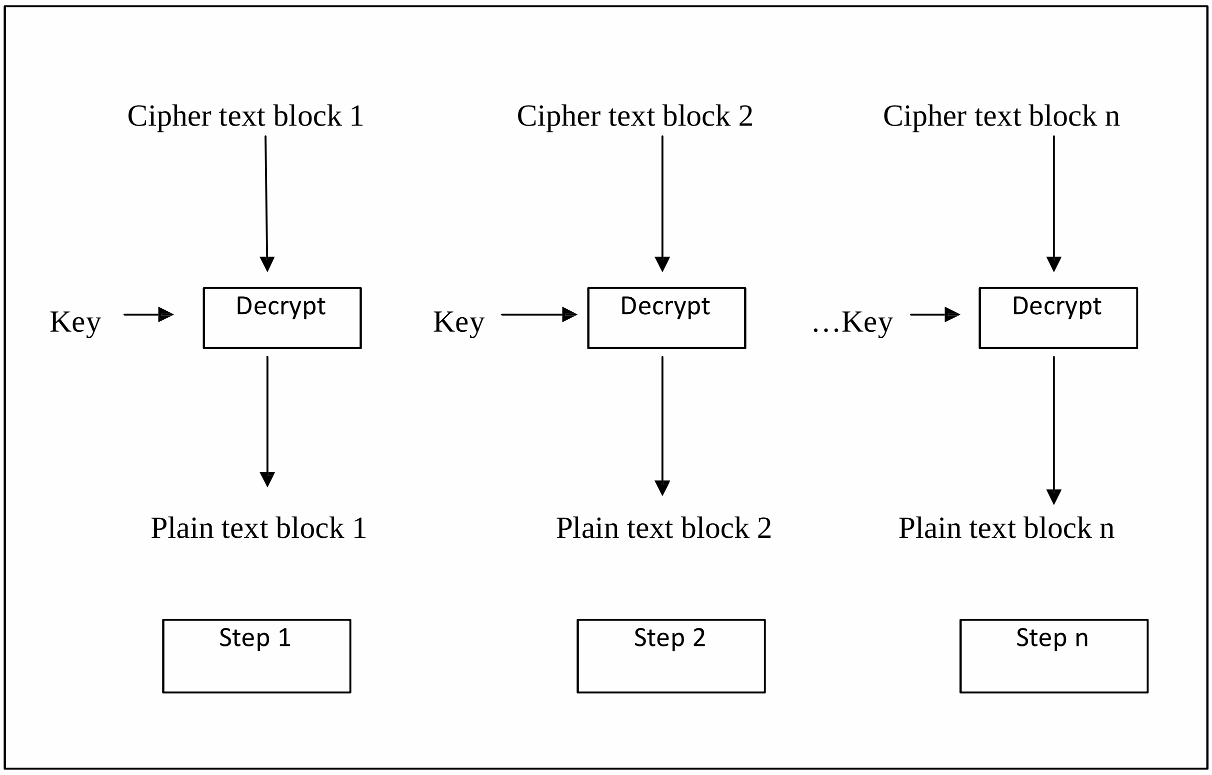


Fig. ECB mode-The decryption process

II. Cipher Block chaining (CBC) mode:

In case of ECB, within a given message, a plain text block always produces the same cipher text block. Thus, if a block of plain text occurs more than once in the input, the corresponding cipher text block will also occur more than once in the output, thus providing some clues to a cryptanalyst.

To overcome this problem, the Cipher Block chaining (CBC) mode ensures that even if a block of plain text repeats in the input, these two or more identical plain text blocks yield totally different cipher text blocks in the output. For this a feedback mechanism is used. Chaining adds a feedback mechanism to a block cipher.

In CBC the result of the encryption of the previous block are fed back into the encryption of the current block.

That is, each block is used to modify the encryption of the next block.

Thus, each block of cipher text is dependent on the corresponding current input plain text block as well as all the previous plain text blocks.

The encryption process of CBC is depicted in following fig.:

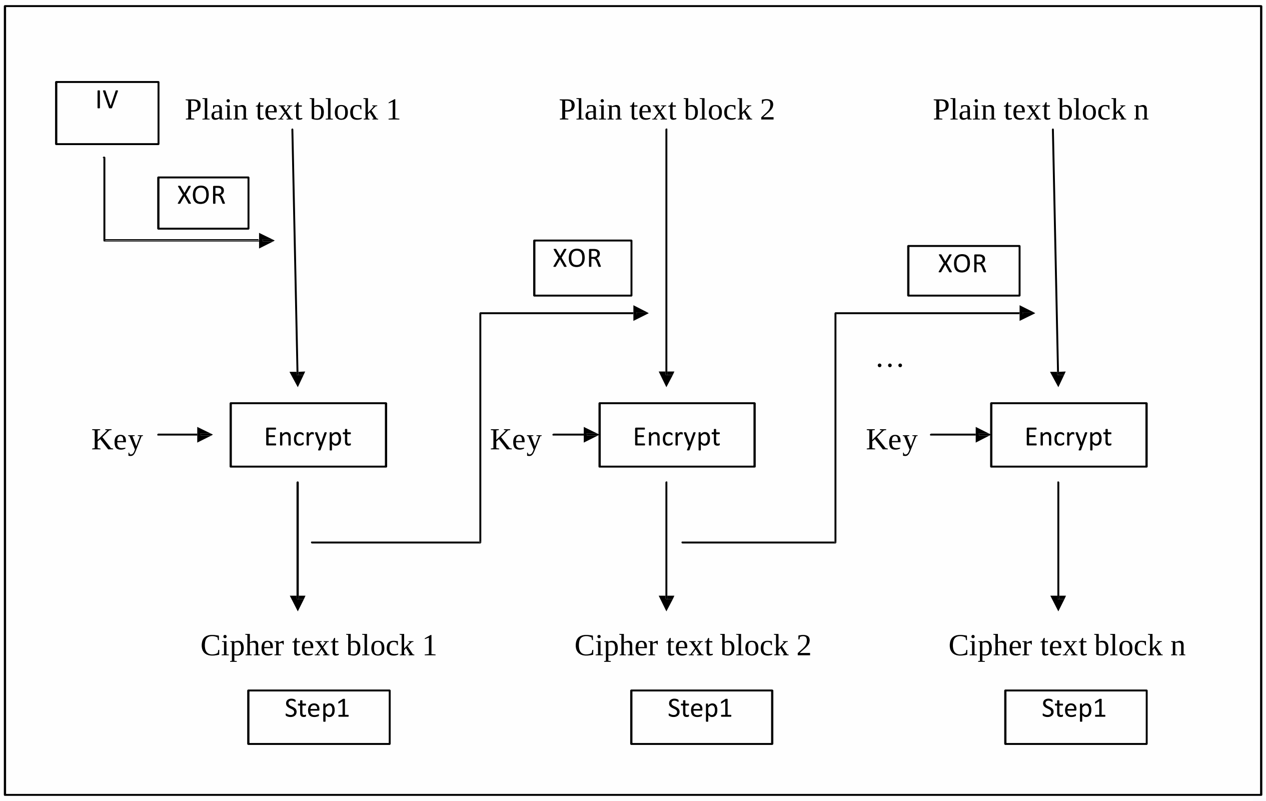


Fig. CBC mode – The encryption process

1. As shown in fig, the first step receives two inputs:
   The first block of plain text and a random block of text called as Initialization Vector (IV).
   a) The IV has no special meaning; it is simply used to make each message unique. Since the value of IV is randomly generated, the likelihood of it repeating in two different messages is quite rare. It is not mandatory to keep IV secret.
   b) The first block of cipher text and IV are combined using XOR and then encrypted using a key to produce the first cipher text block. The first cipher block then provided as a feedback to the next plain text block.
2. In the second step, the second plain text block is XORed with the output of step1, i.e. the first cipher text block. It is then encrypted with the same key, as used in step 1. This produces cipher text block 2.
3. In the third step, the third plain text block is XORed with the output of step 2, i.e. the second cipher text block. It is then encrypted with the same key, as used in step 1.

4. This process continues for all the remaining plain text blocks of the original message.

The decryption process works as follows:

1. The cipher text block 1is passed through the decryption algorithm using the same key, which was used during the encryption process for all the plain text blocks. The output of this step is then XORed with the IV. This process yields plain text block 1.
2. In step 2, the cipher text block 2 is decrypted, and its output is XORed with cipher text block 1, which yields plain text block 2.
3. This process continues for all the cipher text blocks in the encrypted message.

The decryption process is shown as follows:

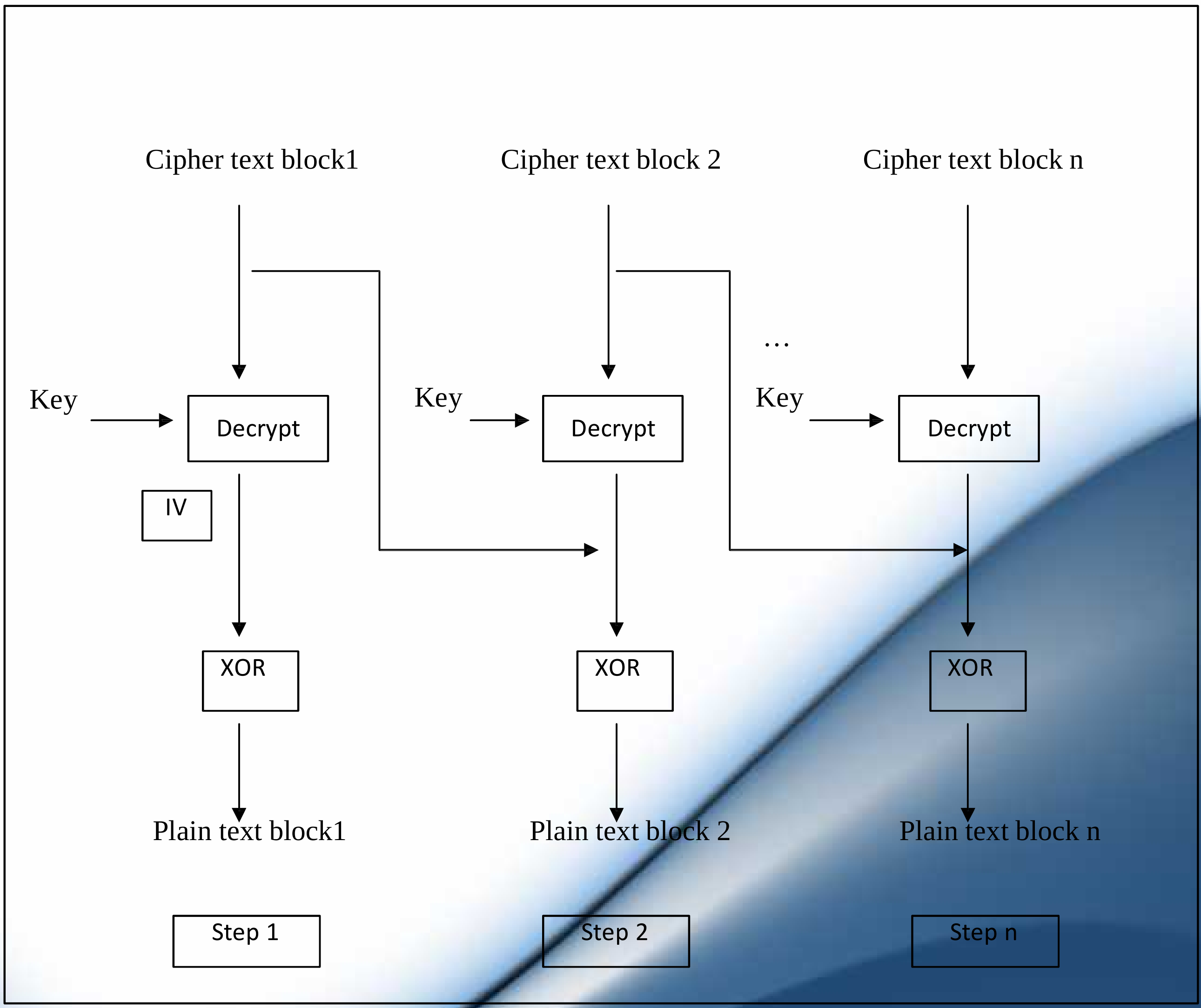


Fig. CBC mode- The decryption process

III. Cipher Feedback (CFB) mode:

Not all applications can work with blocks of data. Security is also required in applications that are character-oriented. The Cipher Feedback (CFB) mode is useful in such cases. In this mode data is encrypted in units that are smaller than a defined block size.

CFB works as follows;

Step 1:

Like CBC, a 64-bit IV is used in the case of CFB mode. The IV is kept in a shift register. It is encrypted in the first step to produce a corresponding 64-bit IV cipher text as shown in following fig.

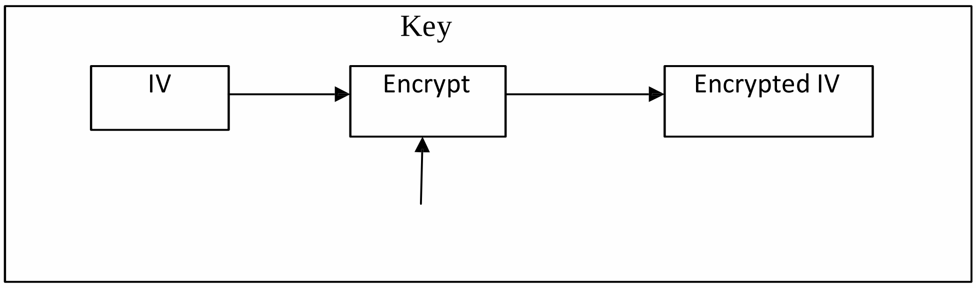


Fig.-CFB-Step 1

Step 2:

Now, the leftmost j bits of the encrypted IV are XORed with the first J bits of plain text. This produces the first portion of cipher text (say c) as shown in fig. c is then transmitted to the receiver.

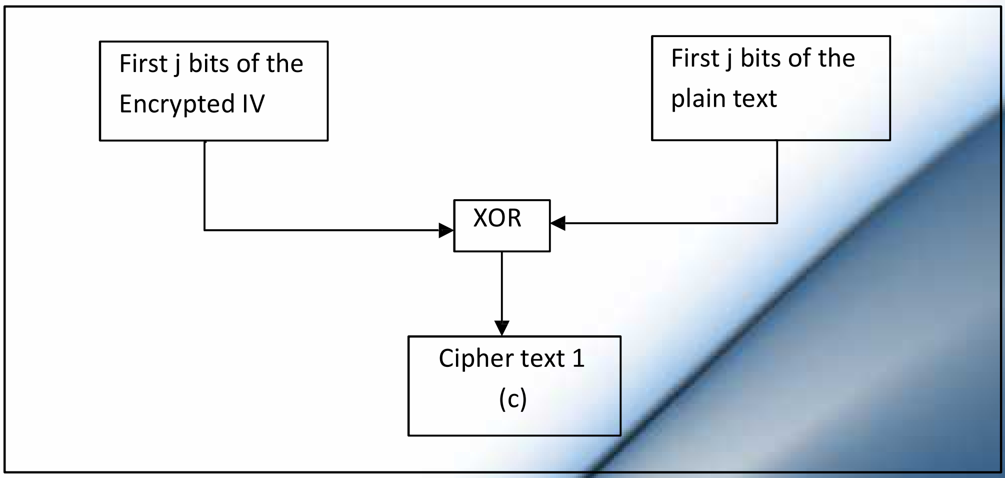

First j bits of the Encrypted IV
First j bits of the plain text
XOR
Cipher text 1 (c)

Fig.-Step 2

Step 3:

Now the bits of IV (i.e. the contents of shift register containing IV) are shifted left by j positions. Thus the rightmost j positions of the shift register now contain unpredictable data. These rightmost j positions are now filled with c. This is shown in following fig,

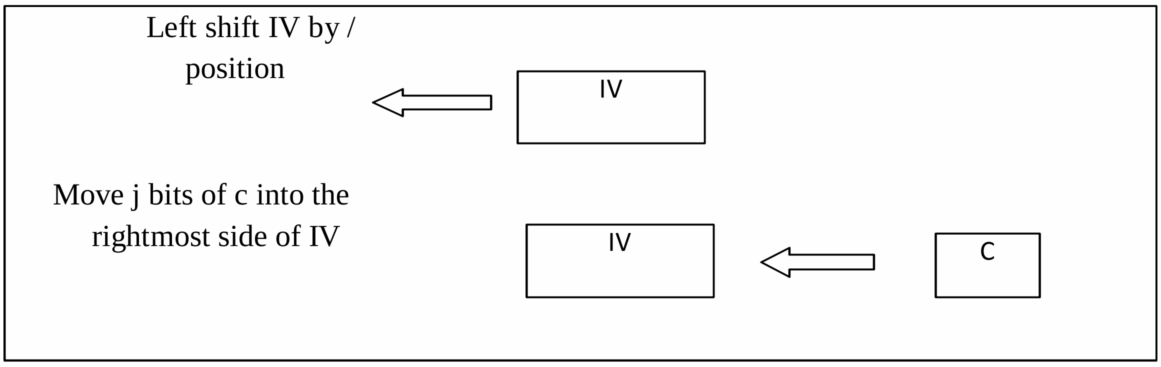

Left shift IV by / position
IV
Move j bits of c into the rightmost side of IV
IV
C

Fig. CFB-step 3

Step 4:
Now, Steps 1 through 3 continue until all the plain text units are encrypted. That is the following steps repeat:

- IV is encrypted.
- The leftmost j bits resulting from this encryption process are XORed with the next j bits of the plain text.
- The resulting cipher text portion is sent to the receiver.
- The shift register containing the IV is left-shifted by j bits.
- The j bits of the cipher text are inserted from right into the shift register containing the IV.

Following figure shows the overall conceptual view of the CFB mode.

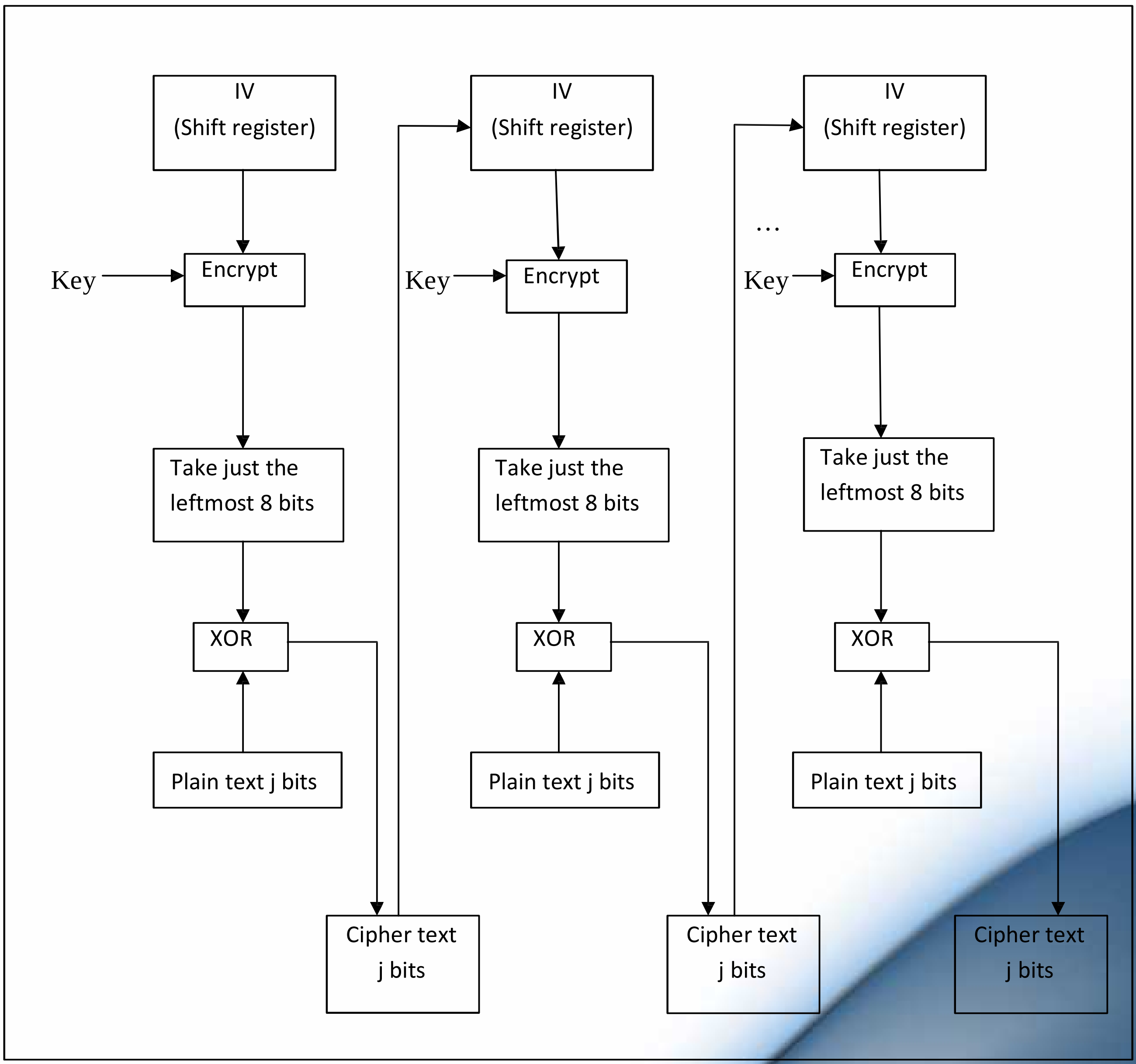


Fig. CFB-The overall encryption process

IV. <u>Output Feedback (OFB) mode:</u>

The OFB mode is extremely similar to the CFB. The only difference is that in case of CFB, the cipher text is fed into the next stage of encryption process but in case of OFB, the output of the IV encryption process is fed into the next stage of encryption process. Following figure shows the overall conceptual view of the CFB mode.

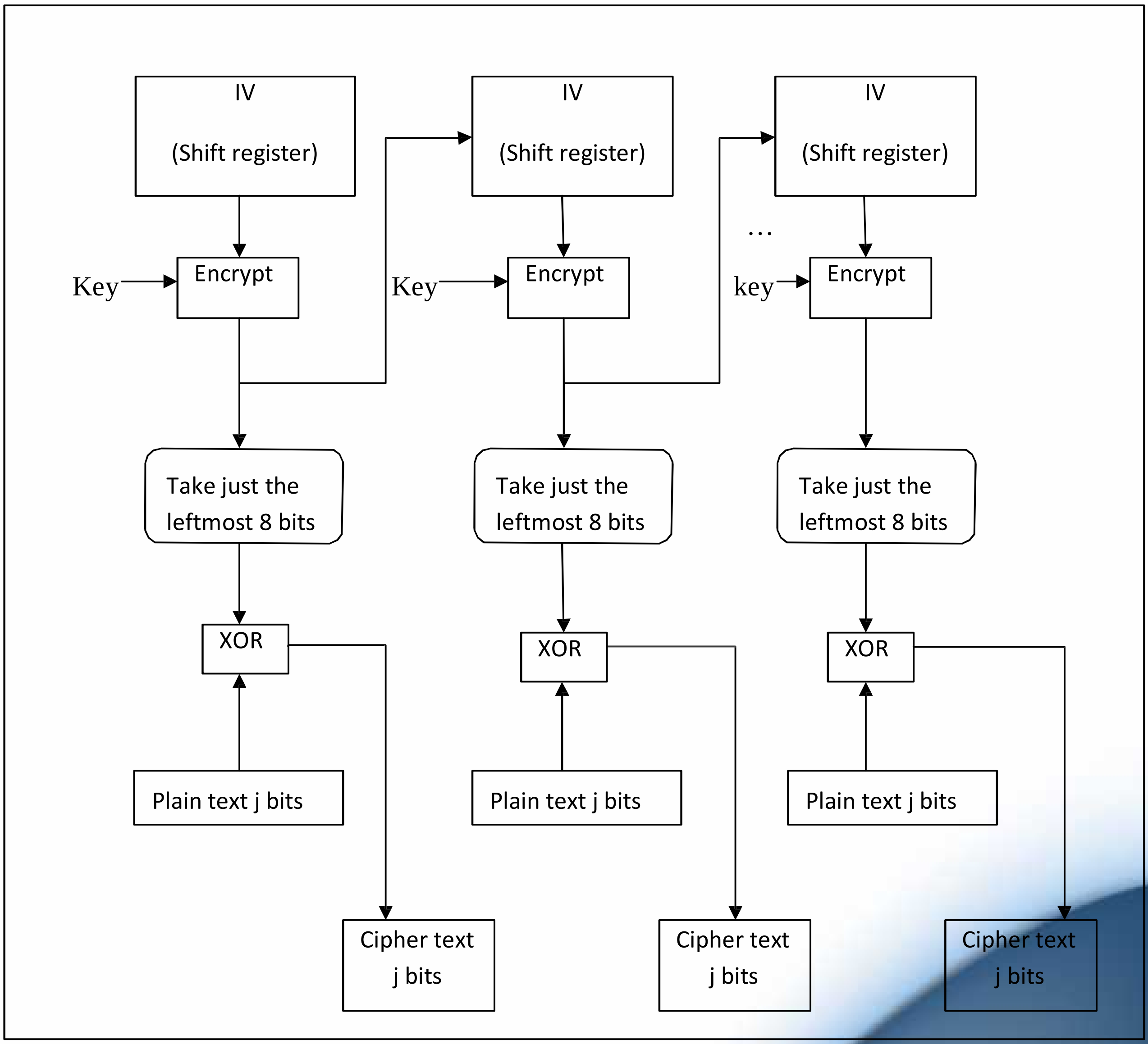


Fig. OFB-The overall encryption process

Q 4) What do you understand by Cryptography. Explain types Distinguish between symmetric and asymmetric cryptography .

Ans- :- Cryptography is an art of achieving security by encoding messages to make them non readable.

Within the context of any application-to-application communication, there are some specific security requirements, including:

- *Authentication:* The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
- *Privacy/confidentiality:* Ensuring that no one can read the message except the intended receiver.
- *Integrity:* Assuring the receiver that the received message has not been altered in any way from the original.
- *Non-repudiation:* A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. In all cases, the initial unencrypted data is referred to as *plaintext*. It is encrypted into *ciphertext*, which will be decrypted into usable plaintext.

There are several ways of classifying cryptographic algorithms. They will be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms are :

- Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption
- Public Key Cryptography (PKC): Uses one key for encryption and another for decryption
- Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information

1. Secret key cryptography:

With *secret key cryptography*, a single key is used for both encryption and decryption., the sender uses the key (or some set of rules) to encrypt the plaintext and sends the ciphertext to the receiver.

The receiver applies the same key (or ruleset) to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called *symmetric encryption*.

- With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, is the distribution of the key.
- Secret key cryptography schemes are generally categorized as being either *stream ciphers* or *block ciphers*.

- Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing.
- A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher.

2. Public-Key Cryptography

Public key cryptography depends upon the existence of *one-way functions*, or mathematical functions that are easy to compute whereas their inverse function is relatively difficult to compute.

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key.

One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it **does not matter which key is applied first**, but that both keys are required for the process to work Because a pair of keys are required, this approach is also called *asymmetric cryptography*.

In PKC, one of the keys is designated the *public key* and may be advertised as widely as the owner wants. The other key is designated the *private key* and is never revealed to another party. It is straight forward to send messages under this scheme. Suppose Alice wants to send Bob a message. Alice encrypts some information using Bob's public key; Bob decrypts the ciphertext using his private key. This method could be also used to prove who sent a message; Alice, for example, could encrypt some plaintext with her private key; when Bob decrypts using Alice's public key, he knows that Alice sent the message and Alice cannot deny having sent the message (*non-repudiation*).

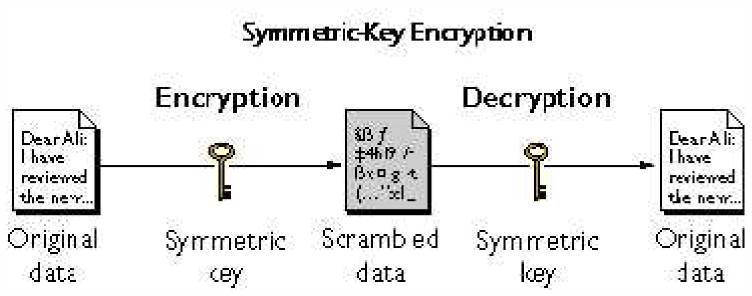
3.3. Hash Functions

Hash functions, also called *message digests* and *one-way encryption*, are algorithms that, in some sense, use no key (Figure 1C). Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a *digital fingerprint* of a file's contents, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file.

DES	IDEA
1. DES is efficient to implement in h/w but relatively slow if implemented in s/w. 2.56 bit key is used to generate 16, 48 bit round key. 3.Same key are used in reverse order deriving decryption. 4.Each DES S-box maps a 6-bits qty into a 4-bits qty. 5. DES has 16 round. 6.56 bits key is expanded to generate 16 keys.	1.IDEA was designed to compute in s/w. 2. Uses 128 bit key. 3.Decryption odd round mathematical inverses of keys is used. Even round same key used while encrypt are used No inverse of key. 4.Each Primitive maps 16-bits qty into a 16-bits qty. 5.Has 17-round odd no round are diff from even numbered round. 6.128-bit key is expanded into 52 keys.

Q 5) What are pitfalls of symmetric key cryptography?

Ans: Cryptography or **cryptology**; from Greek, "hidden, secret, *gráphin*, "writing is the practice and study of hiding information. Crytography can be Symmetric or public key and Asymmetric or private key. Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way) Symmetric-key algorithms can be divided into stream ciphers and block ciphers. Stream ciphers encrypt the bits of the message one at a time, and block ciphers take a number of bits and encrypt them as a single unit. Blocks of 64 bits have been commonly used. **Symmetric cryptography** uses a single private key to both encrypt and decrypt data. Any party that has the key can use it to encrypt and decrypt data. They are also referred to as block ciphers. Symmetric cryptography algorithms are typically fast and are suitable for processing large streams of data. The keys tend to be much smaller for the level of protection they afford.



Disadvantages

Need for secure channel for secret key exchange:

Sharing the secret key in the beginning is a problem in symmetric key encryption. It has to be exchanged in a way that ensures it remains secret. The disadvantage of symmetric cryptography is that it presumes two parties have agreed on a key and been able to exchange that key in a secure manner prior to communication. This is a significant challenge. Symmetric algorithms are usually mixed with public key algorithms to obtain a blend of security and speed. Therefore, symmetric cryptography is effective only if the symmetric key cipher is kept secret by the two parties involved. If anyone else finds the key, it affects both confidentiality and authentication. A person with an unauthorized symmetric key cipher not only can decrypt messages sent with that key, but can encrypt new messages and send them as if they came from one of the two parties who were originally using the key.

Too many keys:

- A new shared key has to be generated for communication with every different party. This creates a problem with managing and ensuring the security of all these keys. In a given system, each pair of system that wants to communicate must share individual private key. Thus the number of keys generated is far more than the keys generated in public key cryptography.

Origin and authenticity of message cannot be guaranteed:

- Since both sender and receiver use the same key, messages cannot be verified to have come from a particular user. This may be a problem if there is a dispute.
- Symmetric ciphers have historically been susceptible to known-plaintext attacks, chosen plaintext attacks, differential cryptanalysis and linear cryptanalysis

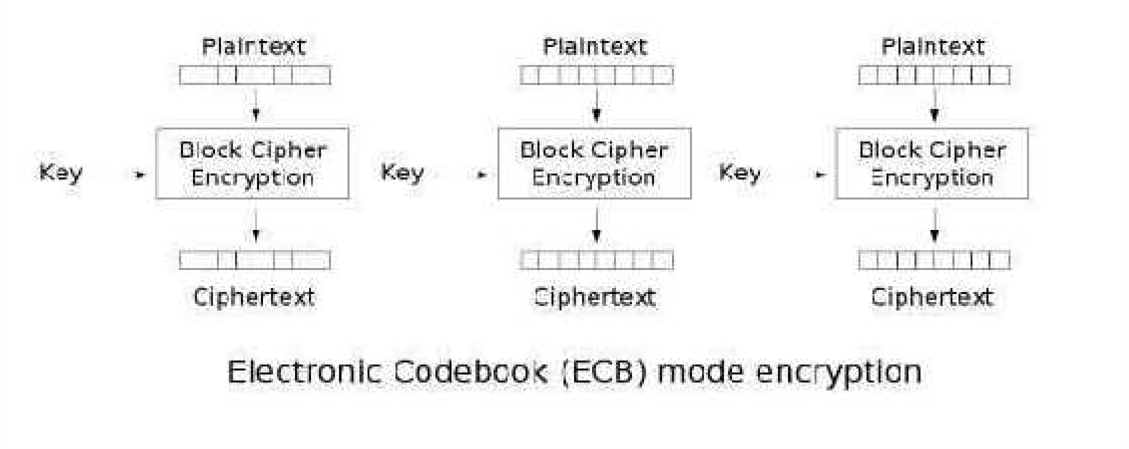
Q6) Explain different modes in which block and stream ciphers work?

Ans:

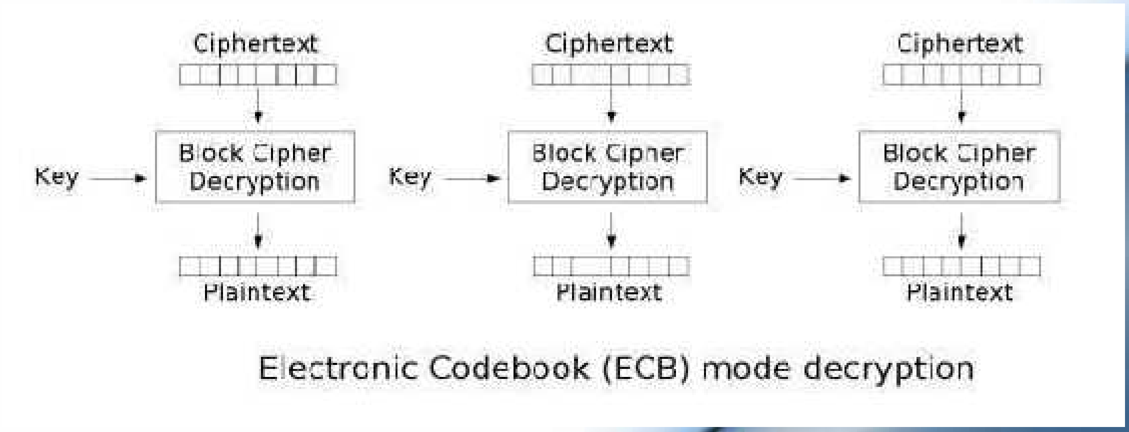
1. Electronic Code Book (ECB) Mode

It is the simplest mode of operation. The incoming plain text message is divided into block of 64 bits each. Each such block Is then encrypted independently of the other block.

The same key is used for encryption, of all blocks in the message.



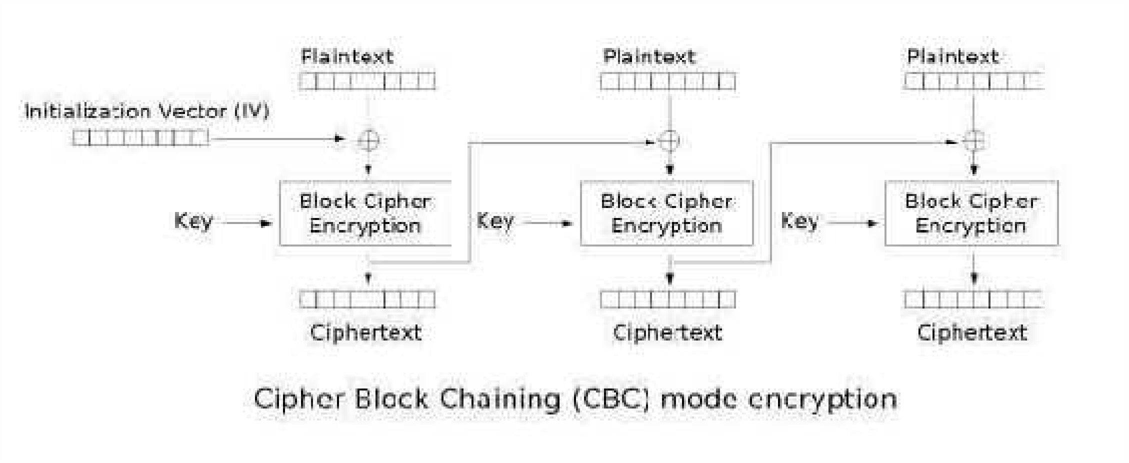
At the Receiver end, the incoming data is divided into 64-bits blocks and by using the same key as was used for encryption , each block is decrypted to produce the corresponding plain text block.

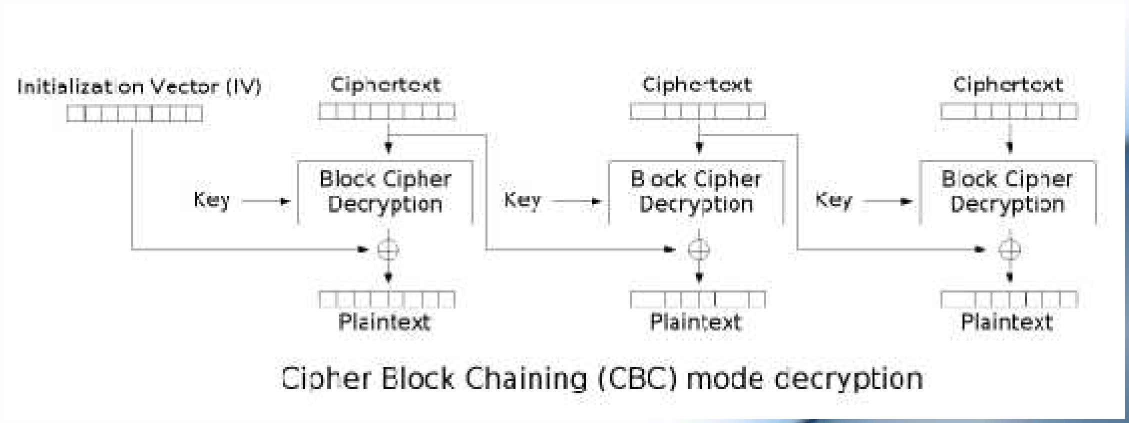


The disadvantage of this method is that identical plaintext blocks are encrypted into identical cipher text blocks; thus, it does not hide data patterns well. In some senses, it doesn't provide serious message confidentiality, and it is not recommended for use in cryptographic protocols at all.

2.Cipher Block Chaining (CBC) Mode

Cipher Block Chaining (CBC) is a mode of operation for a block cipher, one in which a sequence of bits is encrypted as a single unit or block with a cipher key applied to the entire block. Cipher block chaining uses what is known as an initialization vector (IV) of a certain length. CBC prevents the problems associated with Electronic Codebook (ECB), where every block of "plain text" maps to exactly one block of "cipher text" by having each encrypted block XORed with the previous block of cipher text. In this way, identical patterns in different messages are encrypted differently, depending upon the difference in the previous data.

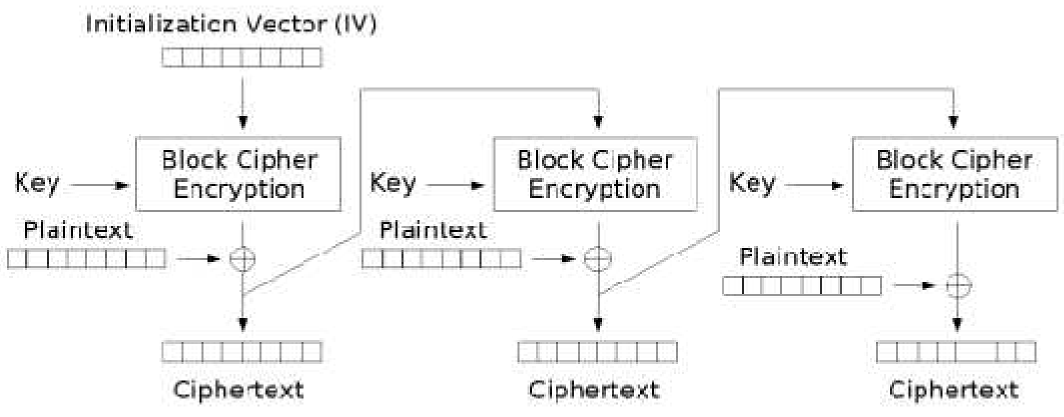




The Initialization Vector (IV) is simply used to make each message unique. Since the value o IN is randomly generated, the likelihood of it repeating in two different message is quite rare. IV helps in making the cipher texts somewhat unique or at least quite different from all other cipher texts in a different message. It is not mandatory to keep IV secret.

3. Cipher Feedback (CFB) Mode

In this mode,data is encrypted in units that are smaller(e.g. they could be of size 8 bits i.e. the size of a character typed by an operator) than defined block size which is usually 64 bits.



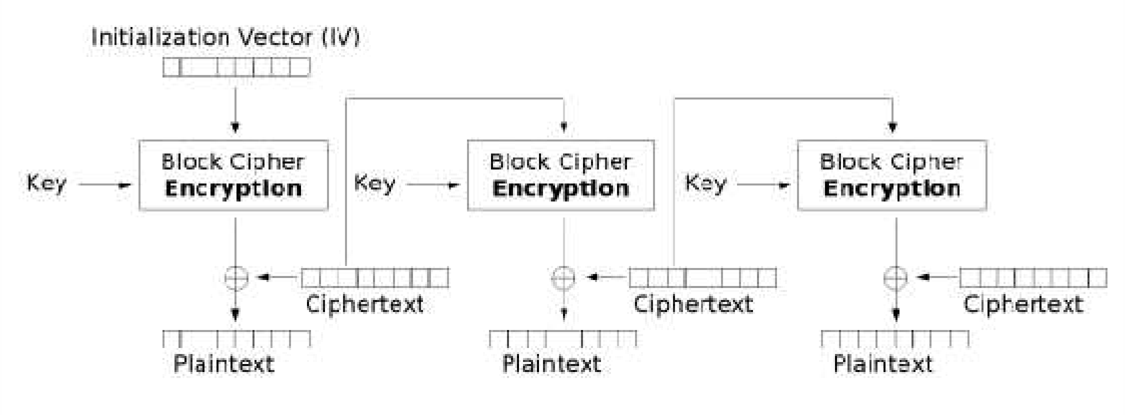
Cipher Feedback (CFB) mode encryption

i)Like CBC, a 64-bit IV is used in the case of CFB mode. The IV is kept in a shift register. It is encrypted to produce a corresponding 64-bit IV cipher text.

ii)Now , the leftmost j bits of the encrypted IV are XORed with the first j bits of the plain text. This produces the first portion of cipher text say C.

iii)Now, the bits of IV are shifted left by j positions. Thus the j positions of the shift regieter now contain unpredictable data. These rightmost j position are now filled with C.

iv) Now steps i) through iii) continue until all the plain text units are encrypted .

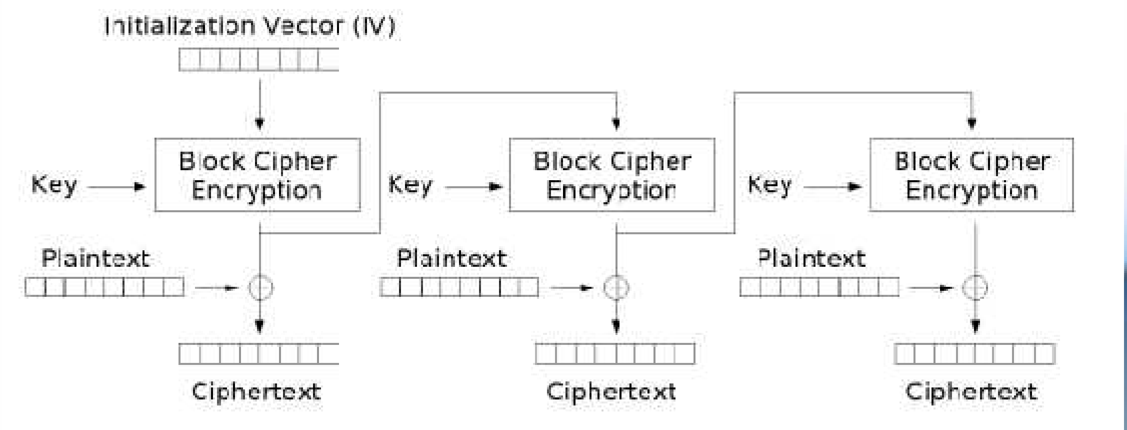


Cipher Feedback (CFB) mode decryption

4.Output Feedback (OFB) Mode

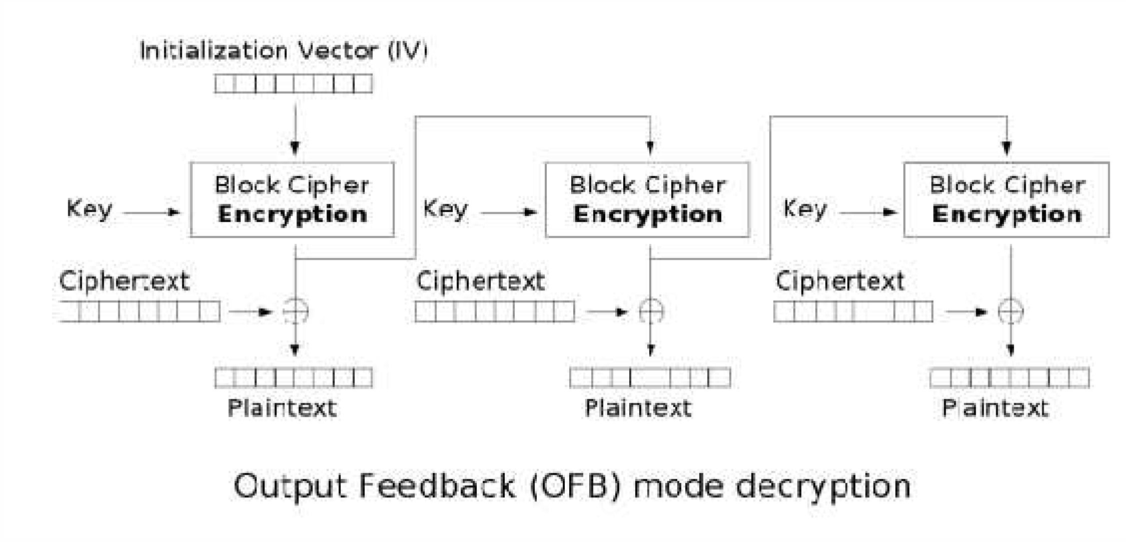
The only difference is that in case of CFB, the cipher text is fed into the next stage of encryption process. But in case of OFB , the output of the cipher text is fed into the next stage of encryption process.

We can state that in this mode if there are errors in individual bits, they remain errors on individual bits and do not corrupt the whole message. That is ,bit errors do not get propagated.



Output Feedback (OFB) mode encryption

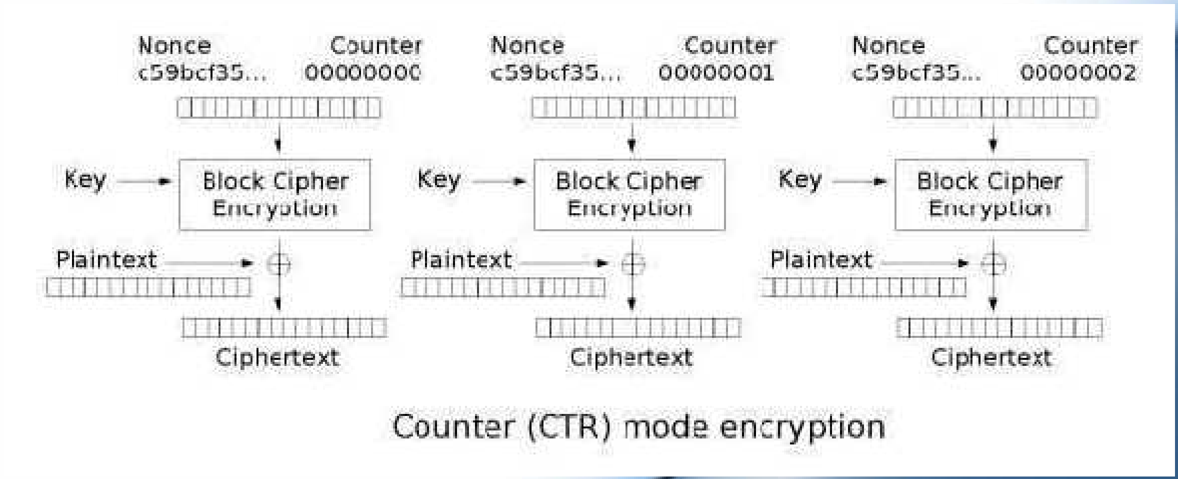
If a cipher text bit Ci is in error, only the decrypted value corresponding to these bits, i.e. Pi is wrong. Other bits are not affected.



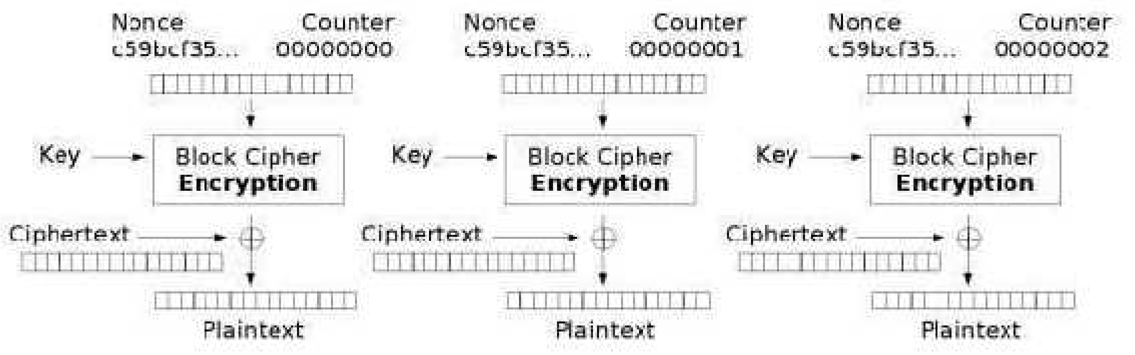
The possible drawback with the OFB is that an attacker can make necessary changes to the cipher text and the checksum o the message in a controlled fashion, these causes change the cipher text without it getting detected. In other words, the attacker changes both the cipher text and the checksum at the same time, hence is no way to detect this change.

5.Counter (CTR) Mode

It uses sequence numbers called ad\s counter as the inputs to the algorithm. After each block is encrypted, to fill the register, the next counter value is used. Usually , a constant is used as the initial counter value and is incremented for every iteration. The size of the counter block is same as that of the plain text block.



CTR mode has similar characteristics to OFB, but also allows a random access property during decryption. CTR mode is well suited to operation on a multi-processor machine where blocks can be encrypted in parallel. The IV/nonce and the counter can be concatenated, added, or XORed together to produce the actual unique counter block for encryption.



Counter (CTR) mode decryption

Q7) Give a overview of DES. Explain DES round. What are weak and semi-weak keys used in DES.
OR
Explain following steps in DES algorithm.

1. **Basic principle**
2. **Initial Permutation**
3. **Rounds**

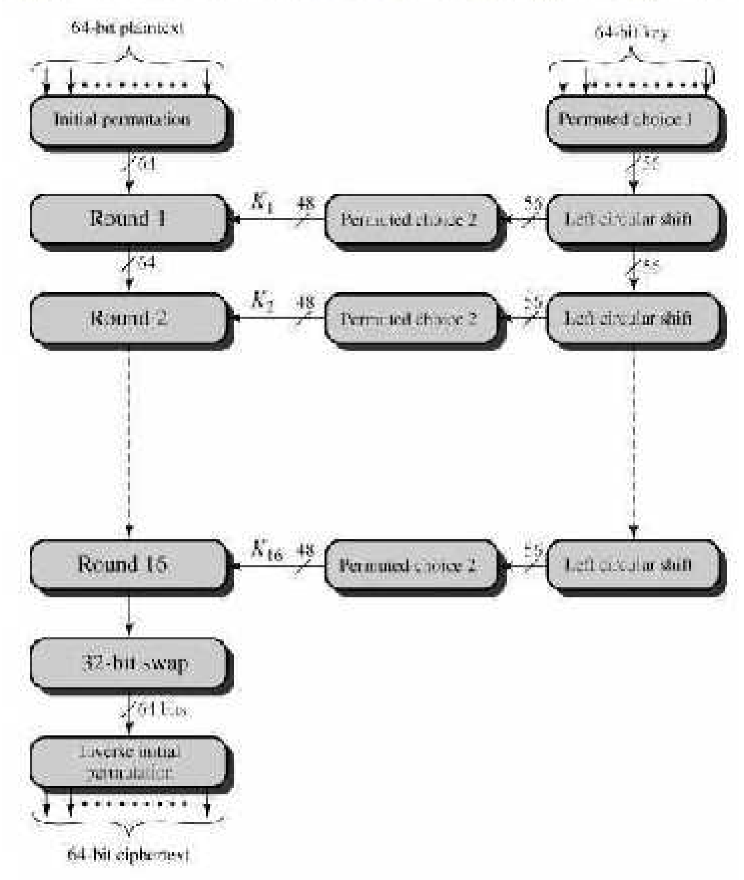
Ans:

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA).[6] For DES, 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.

DES Encryption

The overall scheme for DES encryption is illustrated in Figure 3.4. As with any encryption scheme, 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.

Figure 3.4. General Depiction of DES Encryption Algorithm



Looking at the left-hand side of the figure, we can see that the 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 last (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 pre-output. 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.

The right-hand portion of Figure 3.4 shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the 16 rounds, a *sub-key* (K_i) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different sub-key is produced because of the repeated shifts of the key bits.

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.

Details of Single Round

Figure 3.5 shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any cipher, the overall processing at each round can be summarized in the following formulas:

$L_i = R_{i-1}$

$R_i = L_{i-1} \text{ x } F(R_{i-1}, K_i)$

The round key K_i is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits. The resulting 48 bits are XORed with K_i. This 48-bit result passes through a substitution function that produces a 32-bit output, which is then permuted.

The role of the S-boxes in the function F is illustrated in Figure 3.6. 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 interpreted as follows: The first and last bits of the input to box S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i. 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 S_1 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.

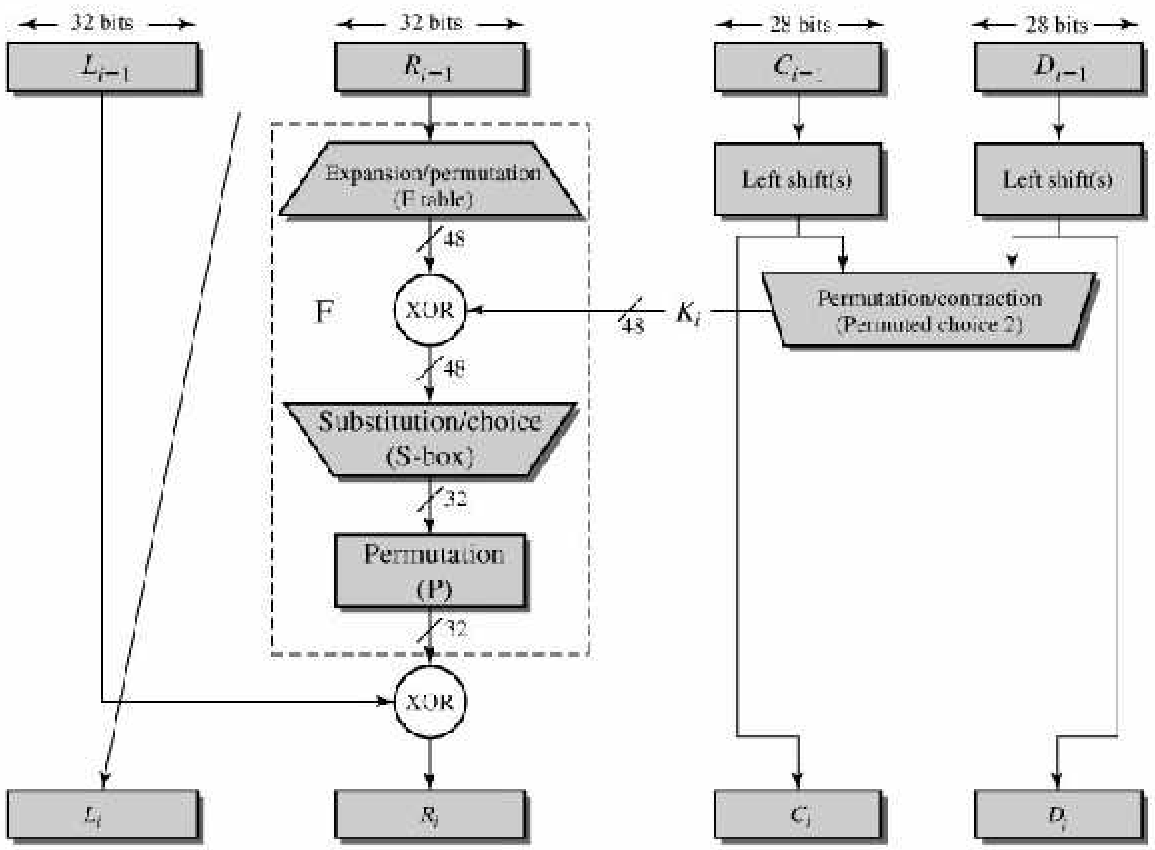
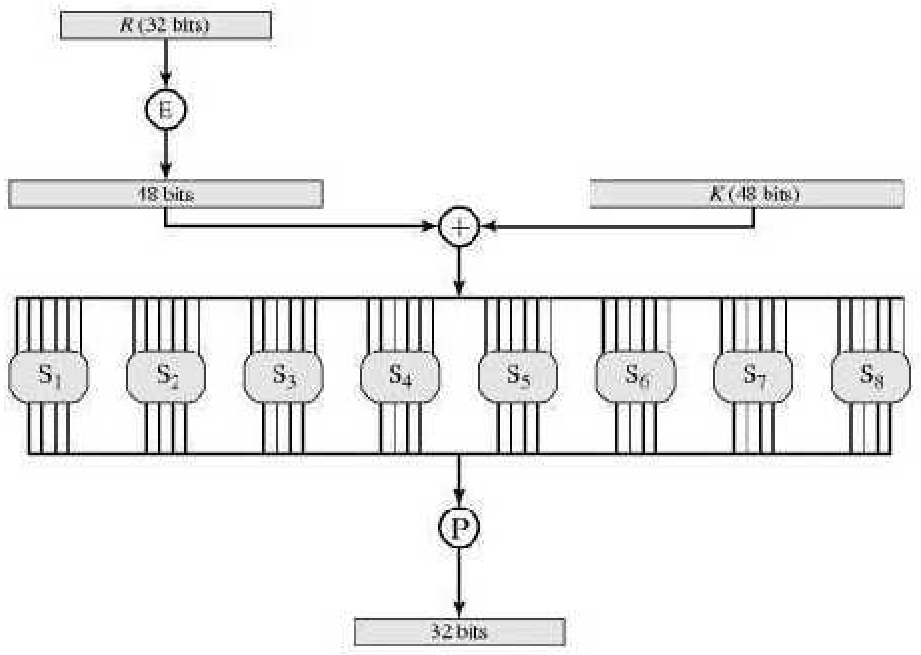


Figure 3.6. Calculation of F(R, K)

(This item is displayed on page 78 in the print version)



Weak and Semi-weak Keys

There are 16 DES keys that the security community warns people against using, because they have starnge properties. But the probability of randomly generating one of these keys is only $16/2^{56}$, which in our opinion is nothing to worry about. Its probably equally insecure to use a key with a value less than 1000, since an attacker might be likely to start searching for a keys from the bottom.

Generating the per-round keys that the key is subjected to an initial permutation to generate two 28-bit quantities, C_0 and D_0 . The 16 suspect keys are ones for which C_0 and D_0 are one of the four values: All ones, all zeros, alternatives ones and zeros, alternating zeros and ones. Since there are 4 possible values for each half, there are 16 possibilities in all. The 4 weak keys are the ones for which each of C_0 and D_0 are all ones or all zeros. Weak keys are their own inverses. The remaining 12 keys are the semi-weak keys. Each is the inverse of one of the others.

Q 8) Differentiate between:

- **DES & IDEA**
- **ECB Mode & CBC mode of DES**

Ans:

DES	IDEA
1. DES is efficient to implement in h/w but relatively slow if implemented in s/w. 2.56 bit key is used to generate 16, 48 bit round key. 3.Same key are used in reverse order deriving decryption. 4.Each DES S-box maps a 6-bits qty into a 4-bits qty. 5. DES has 16 round. 6.56 bits key is expanded to generate 16 keys.	1.IDEA was designed to compute in s/w. 2. Uses 128 bit key. 3.Decryption odd round mathematical inverses of keys is used. Even round same key used while encrypt are used No inverse of key. 4.Each Primitive maps 16-bits qty into a 16-bits qty. 5.Has 17-round odd no round are diff from even numbered round. 6.128-bit key is expanded into 52 keys.

ECB	CBC
1.Each block is encrypted to cipher text block. 2. Does not use IV. 3. Error is transmitted cipher text. 4.Repeated plain text, cipher text repeated 5. Less Expensive.	1.Each block is encrypted to cipher text block. Each cipher text block used to modify the encryption of next block. 2. Uses IV. 3.Error propagates in both cipher and plain text msg. 4.Different Cipher text block are three. 5.Overhead of transmitting and generating IV.

Q. 9) Explain the symmetric key encryption algorithm IDEA.

Ans. The International Data Encryption Algorithm (IDEA) is one of the strongest cryptographic algorithms. It was launched in 1990. Technically, IDEA is a block cipher. It works on 64-bit plain text blocks. The key size is 128 bits. IDEA is a symmetric key algorithm, that is, the same algorithm is used for encryption as well as decryption.

Working of IDEA:

The 64-bit input plain text block is divided into four portions of plain text, say P1 to P4. Thus, P1 to P4 are the inputs to the first round. There are eight such rounds. For the first round, we will have keys from Z1 to Z6. For the second round, we will have keys from Z7 to Z12. Finally, for the eighth round, the keys would be from Z43 to Z48. The final step consists of an Output Transformation, which has four sub-keys. The final output is produced by the Output Transformation step, which is four blocks of cipher text which are combined to form the final 64 bit cipher text block (C1 to C4).

Rounds:

Each round out of the eight rounds is performed as per the steps given below:

1. Multiply P1 and Z1
2. Add P2 and Z2
3. Add P3 and Z3
4. Multiply P4 and Z4
5. XOR the results of step 1 and step 3
6. XOR the results of step 2 and step 4
7. Multiply the results of step 5 with Z5
8. Add the results of step 6 and step 7
9. Multiply the results of step 6 with Z6
10. Add the results of step 7 and step 9
11. XOR the results of step 1 and step 9
12. XOR the results of step 3 and step 9
13. XOR the results of step 2 and step 10
14. XOR the results of step 4 and step 10

Sub-key Generation for a Round:

First round- The initial key consists of 128 bits from which 6 sub-keys Z1 to Z6 are generated for the first round. Since these keys consist of 16 bits each, the first 96 bits are used for the first round.

Second round- In the second round, the 32 unused bits (97-128) are used. For each round, 96 bits are required. Hence, to acquire the next 64 bits, the technique of key shifting is used.

Output Transformation:

This is a one-time operation. It takes place at the end of the eighth round. Output of eighth round is the input to this round. The 64-bit value is divided into four sub-blocks (say R1 to R4, each consisting of 16 bits).

The steps in Output Transformation are as follows:

1. Multiply R1 and Z1
2. Add R2 and Z2
3. Add R3 and Z3
4. Multiply R4 and Z4

We can show the entire working of IDEA as given below:

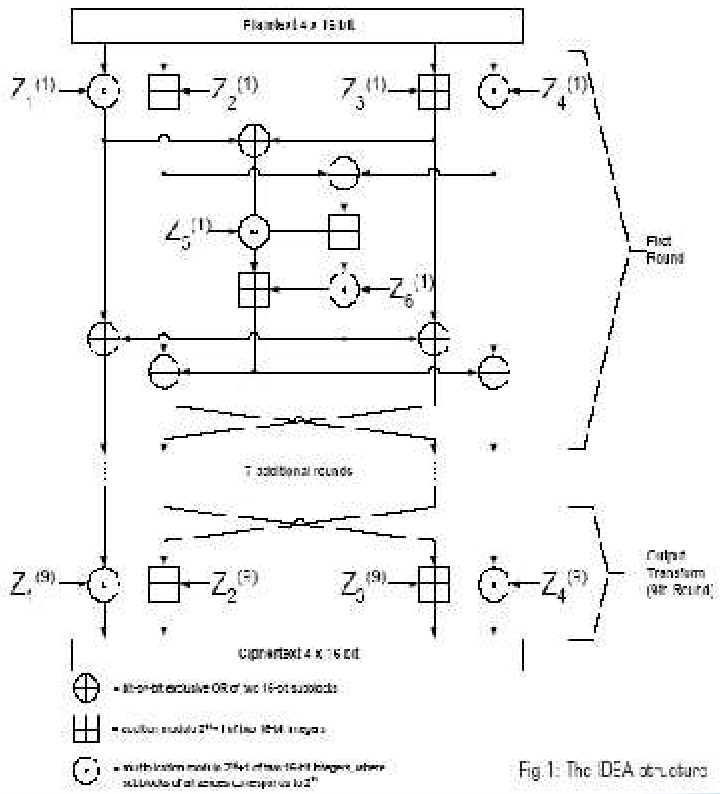


Fig 1: The IDEA structure

The Strength of IDEA:

IDEA uses a 128-bit key which is double the key size of DES. Thus, to break into IDEA, 2^{128} operations are required. Hence, more than 54×10^{23} years are required to break IDEA!

Q 10) What is the importance of message digest ? Explain MD2 and MD4 message digest scheme?

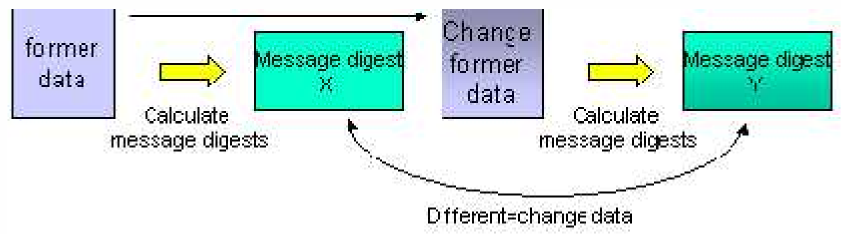
Ans:

1) A message digest is a compact digital signature for an arbitrarily long stream of binary data
2) Never generate the same signature for two different sets of input
3) Achieving such theoretical perfection would require a message digest as long as the input file
4) Compromise in favor of a digital signature of modest and usually fixed size created with an algorithm designed to make preparation of input text with a given signature computationally infeasible
5) Chances of two message digests being the same for two different inputs extremely remote
6) Message digest algorithms have much in common with techniques used in encryption

Importance of message digest:

1) Message digest functions also called *hash functions* , are used to produce digital summaries of information called message digests. Message digests (also called *hashes*) are commonly 128 bits to 160 bits in length and provide a digital identifier for each digital file or document. Message digest functions are mathematical functions that process information to produce a different message digest for each unique document. Identical documents have the same message digest; but if even one of the bits for the document changes, the message digest changes.
2) Because message digests are much shorter than the data from which the digests are generated and the digests have a finite length, duplicate message digests called *collisions* can exist for different data sets. However, good message digest functions use one-way functions to ensure that it is mathematically and computationally infeasible to reverse the message digest process and discover the original data. Finding collisions for good message digest functions is also mathematically and computationally infeasible but possible given enough time and computational effort. However, even if an attacker discovers a collision, it is highly improbable that the collision could be useful.
3) Message digests are commonly used in conjunction with public key technology to create digital signatures or "digital thumbprints" that are used for authentication, integrity, and nonrepudiation. Message digests also are commonly used with digital signing technology to provide data integrity for electronic files and documents
4) The generation of a digest is very fast and the digest itself is very small and can easily be encrypted and transmitted over the internet
5) It is very easy and fast (and therefore cheap) to check some data for validity

6) The algorithms are well known and implemented in most major programming languages, so they can be used in almost all environments

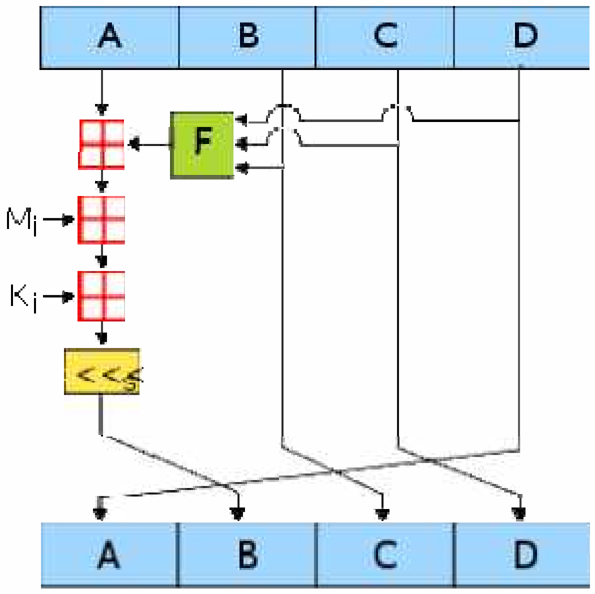


MD2:

1) **Message Digest Algorithm 2 (MD2)** is a cryptographic hash function developed by Ronald Rivest in 1989. The algorithm is optimized for 8-bit computers. MD2 is specified in RFC 1319. Although other algorithms have been proposed since, such as MD4, MD5 and SHA, even as of 2010 MD2 remains in use in public key infrastructures as part of certificates generated with MD2 and RSA
2) The 128-bit hash value of any message is formed by padding it to a multiple of the block length on the computer (128 bits or 16 bytes) and adding a 16-byte checksum to it. For the actual calculation, a 48-byte auxiliary block and a 256-byte S-table generated indirectly from the digits of the fractional part of pi are used (see nothing up my sleeve number). The algorithm runs through a loop where it permutes each byte in the auxiliary block 18 times for every 16 input bytes processed.
3) Once all of the blocks of the (lengthened) message have been processed, the first partial block of the auxiliary block becomes the hash value of the message.

MD4:

1) MD4 (Message-Digest algorithm 4) is a message digest algorithm (the fourth in a series) designed by Professor Ronald Rivest of MIT in 1990. It implements a cryptographic hash function for use in message integrity checks. The digest length is 128 bits. The algorithm has influenced later designs, such as the MD5, SHA-1 and RIPEMD algorithms.
2) The 128-bit (16-byte) MD4 hashes (also termed *message digests*) are typically represented as 32-digit hexadecimal numbers.



3) One MD4 operation : MD4 consists of 48 of these operations, grouped in three rounds of 16 operations. *F* is a nonlinear function; one function is used in each round. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation.

Q.11) What is the minimal & Maximum amount of padding that would be required in each message digest function MD2, MD4, MD5 , SHA, SHA-1?

Ans:

Padding for MD2:

The message is “padded”(extended) so that its length is congruent to 0, modulo 16. That is, the message is extended so that it is multiple of 16 byte long. Padding is always performed, even if the length of the message is already Congruent to 0, modulo 16. Padding is performed as follows : a single “i” bit is appended to the message so that the length is in byte of the padded message become congruent to 0,modulo 16. At this point the resulting message has length what it is exact multiple of 16 byte.

Let M[0…….N-1] denotes the bytes of the resulting message. Where N is a multiple of 16.

Padding For MD4:

The message is “padded”(extended) so that its length is congruent to 448 modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows : a single “1” bit is appended to the message, and then “0” bit appended so that the length is in bit of the padded message become congruent to 448,modulo 512. In all of at least one bit and at most 512 bits are appended.

Padding For MD5:

The message is “padded”(extended) so that its length is similar to 448 modulo 512.that is, the message is extended so that it is just 64 bits timid of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is always similar to 448,modulo 512. Padding is performed as follows : a single “1” bit is appended to the message, and then “0” bit appended so that the length is in bit of the padded message become congruent to 448,modulo 512. In all at least one bit and at most 512 bits are appended.

Padding For SHA:

Like MD5, the first step in SHA is to add padding to the end of original message in such a way that the length of the message is 64 bits short of a multiple of 512. Like MD5, the padding is always added, even if the message is already 64 bits short of a multiple of 512.

Padding For SHA-1:

SHA-1 pads message in same manner as MD4 and MD5, except that SHA-1 is not different for a message that is longer than 2^64 bits.If there were such a message it would take several hundred years to transmit it at 10 Gigabits per second, and it would take even longer to compute the SHA-1 message digest for it at 1000 MBPS.

Q 12) What is a message Digest? How is a message digest calculated using MD5 algorithm?

Ans: The electronic equivalent of the document & fingerprint pair is the message & message digest pair it is fingerprint or the summary of a message. It is similar to the concepts of Longitudinal Redundancy Check (LRC) or Cyclic Redundancy Check (CRC).

To preserve the integrity (i.e. to ensure that a message has not been tampered with after it leaves the sender but before it reaches the receiver) of a message, the message is passed through a message digest algorithm also called as hash function.

Thus we perform a hashing function (or message digest algorithm) over block of data to produce its hash or message digest, which is smaller in size than the original message. This concept is shown in figure.

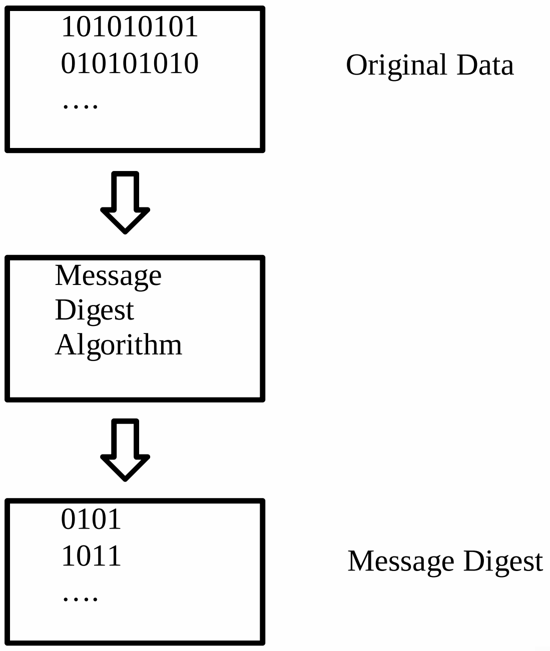


Figure: Message Digest Concept.

MD5 :

- MD5 is a message digest algorithm developed by Ron Rivest.
- MD5 actually has its roots in a series of message digest algorithm, which were predecessors of MD5, all developed by Ron Rivest.
- The original message digest algorithm was called as MD. He soon came up with next version MD2, MD3 & MD4 but those were quite weak, failure etc. consequently Ron Rivest released MD5.
- MD5 is quite fast & produces 128 bit message digest. MD5 has been able to successfully defend itself against collisions.

Working of MD5 :

Step 1: Padding The first step in MD5 is to add padding bits to the original message. The aim of this step is to make the length of the original message equal to a value, which is 64 bits less than an exact multiple of 512.

Thus, after padding, the original message will have a length of 448 bits (64 bits less than 512), 960 bits (64 bits less than 1024), 1472 bits (64 bits less than 1536), etc.

The padding consists of a single 1-bit, followed by as many 0-bits, as required. Note that padding is always added, even if the message length is already 64 bits less than a multiple of 512. Thus, if the message were already of length say 448 bits, we will add a padding of 512 bits to make its length 960 bits. Thus, the padding length is any value between 1 and 512.

The padding process is shown in below:

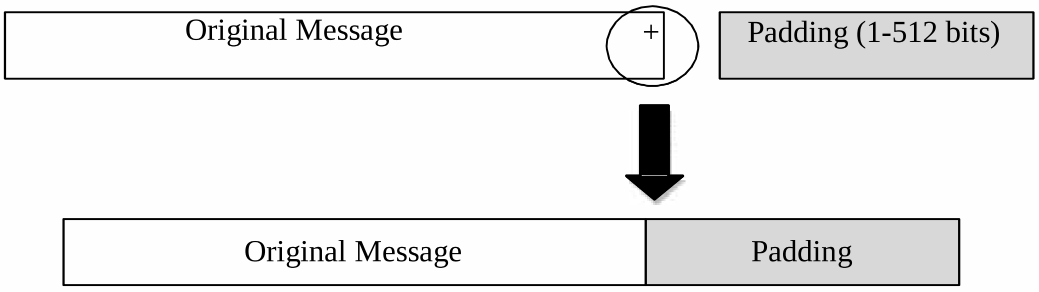


Figure: Padding Process

Step 2: Append length After padding bits are added, the next step is to calculate the original length of the message and add it to the end of message, after padding.

The length of the message is calculated, excluding the padding bits (i.e. it is the length before the padding bits were added). For instance, if the original message consisted of 1000 bits and we added a padding of 472 bits to make the length of the message 64 bits less than 1536 (a multiple of 512), the length is considered as 1000 and not 1472 for the purpose of this step.

This length of the original message is now expressed as a 64-bit value and these 64 bits are appended to the end of the original message + padding. This is shown in Figure below. Note that if the length of the message exceeds 2^{64} bits (i.e. 64 bits are not enough to represent the length, which is possible in the case of a really long message), we use only the low-order 64 bits of the length. That is, in effect, we calculate the length mod 2^{64} in that case.

We will realize that the length of the message is now an exact multiple of 512. This now becomes the message whose digest will be calculated.

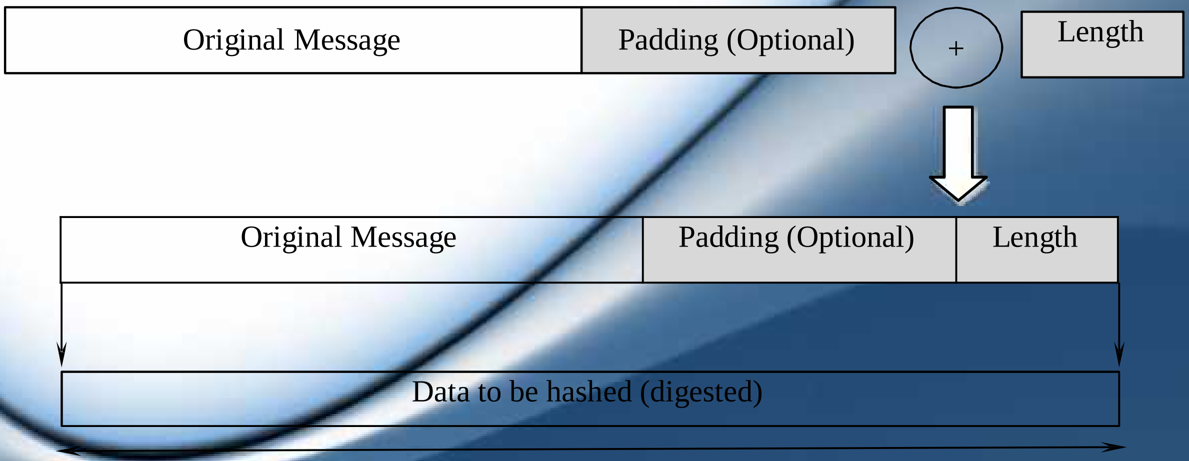


Figure: Append Length

Step 3: Divide the input into 512-bit blocks Now, we divide the input message into blocks, each of length 512 bits. This shown below:

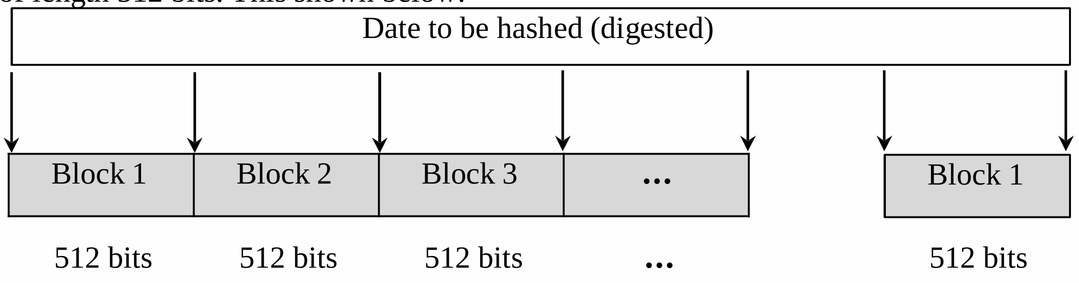


Figure: Data is divided into 512-bits blocks

Step 4: Initialize chaining variables In this step, four variables (called as **chaining variables)** are initialize. They are called as A, B, C and D. Each of these is a 32-bit number. The initial hexadecimal values of these chaining variables are shown below:

A	Hex	01	23	45	67
B	Hex	89	AB	CD	EF
C	Hex	FE	DC	BA	98
D	Hex	76	54	32	10

Figure: Chaining Variables

Step 5: Process blocks After all the initializations, the real algorithm begins. It is quite complicated and we shall discuss it step-by-step to simplify it to the maximum extent possible.

There is a loop that runs for as many 512-bit block as are in the message.

Step 5.1: Copy the four chaining variables into four corresponding variables, a, b, c and d (note the smaller case). Thus, we now have a = A, b = B, c = C and d = D. This is shown below:

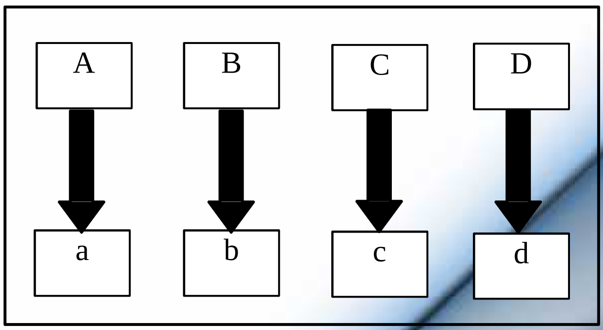


Figure: Copying chaining variables into temporary variables

Actually, the algorithm considers the combination of a, b, c and d as a 128-bit single register (which we shall call as abcd). This register (abcd) is useful in the actual algorithm operation for holding intermediate as well as final results. This is shown below:

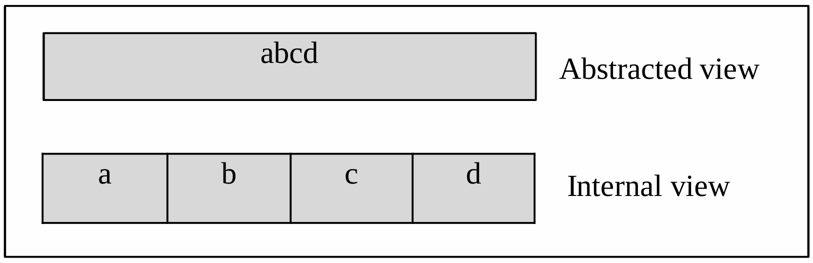


Figure: Abstracted view of the chaining variables

Step 5.2: Divide the current 512-bit block into 16 sub-blocks. Thus, each sub-block contains 32 bits, as shown below:

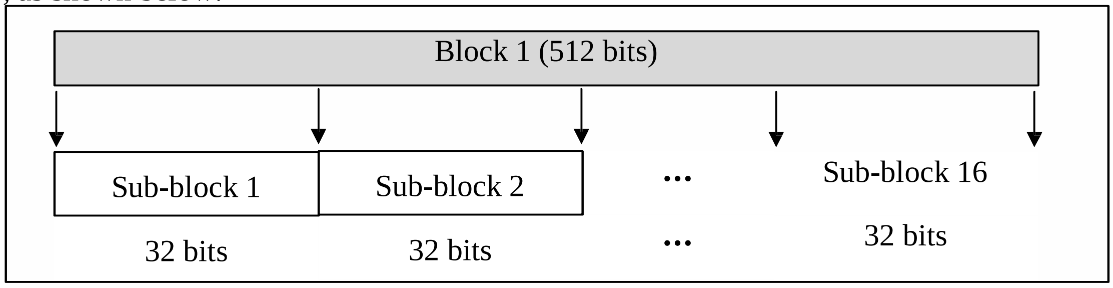


Figure: Sub-blocks within a block

Step 5.3: Now, we have four rounds. In each round, we process all the 16 sub-blocks belonging to a block. The inputs to each round are: (a) all the 16 sub-blocks, (b) the variables a, b, c, d and (c) some constants, designated as t. This is shown as below:

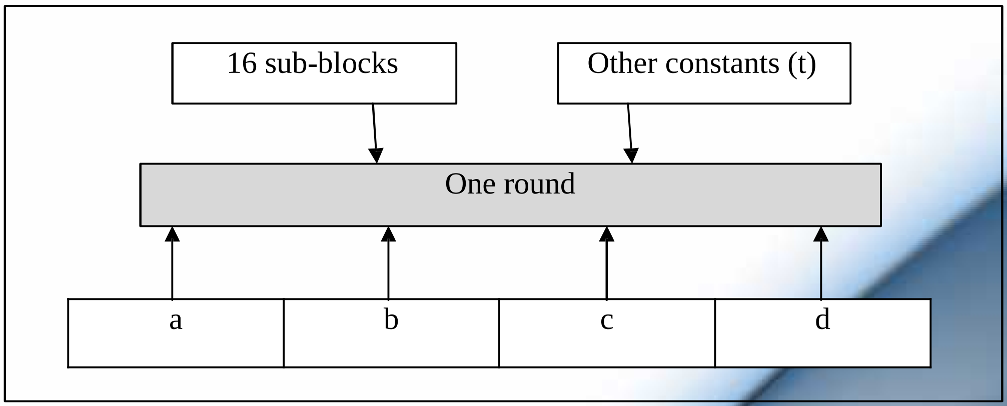


Figure: Conceptual process within a round

All the four rounds vary in one major way: Step 1 of the four round has different processing. The other steps in all the four rounds are the same.

- In each round, we have 16 input sub-blocks, named M[0], M[1], …,M[15] or in general, M[i], where I varies from 0 to 15. As we know, each sub-block consists of 32 bits.
- Also, t is an array of constants. It contains 64 elements, with each element consisting of 32 bits.

We denote the elements of this array t as t[1], t[2],.., t[64] or in general as t[k], where k varies from 1 to 64. Since there are four rounds, we use 16 out of the 64 values of t in each round.

Let us summarize these iterations of all the four rounds. In each case, the output of the intermediate as well as the final iteration is copied into the register abcd. Note that we have 16 such iterations in each round.

1. A process P is first performed on b, c and d. This process P is different in all the four rounds.
2. The variable a is added to the output of the process P (i.e. to the register abcd).
3. The message sub-block M[i] is added to the output of Step 2 (i.e. to the register abcd).
4. The constant t[k] is added to the output of Step 3 (i.e. to the register abcd).
5. The output of Step 4 (i.e. the contents of register abcd) is circular-left shifted by s bits. (The value of s keeps changing, as we shall study).
6. The variable b is added to the output of Step 5 (i.e. to the register abcd).
7. The output of Step 6 becomes the new abcd for the next step

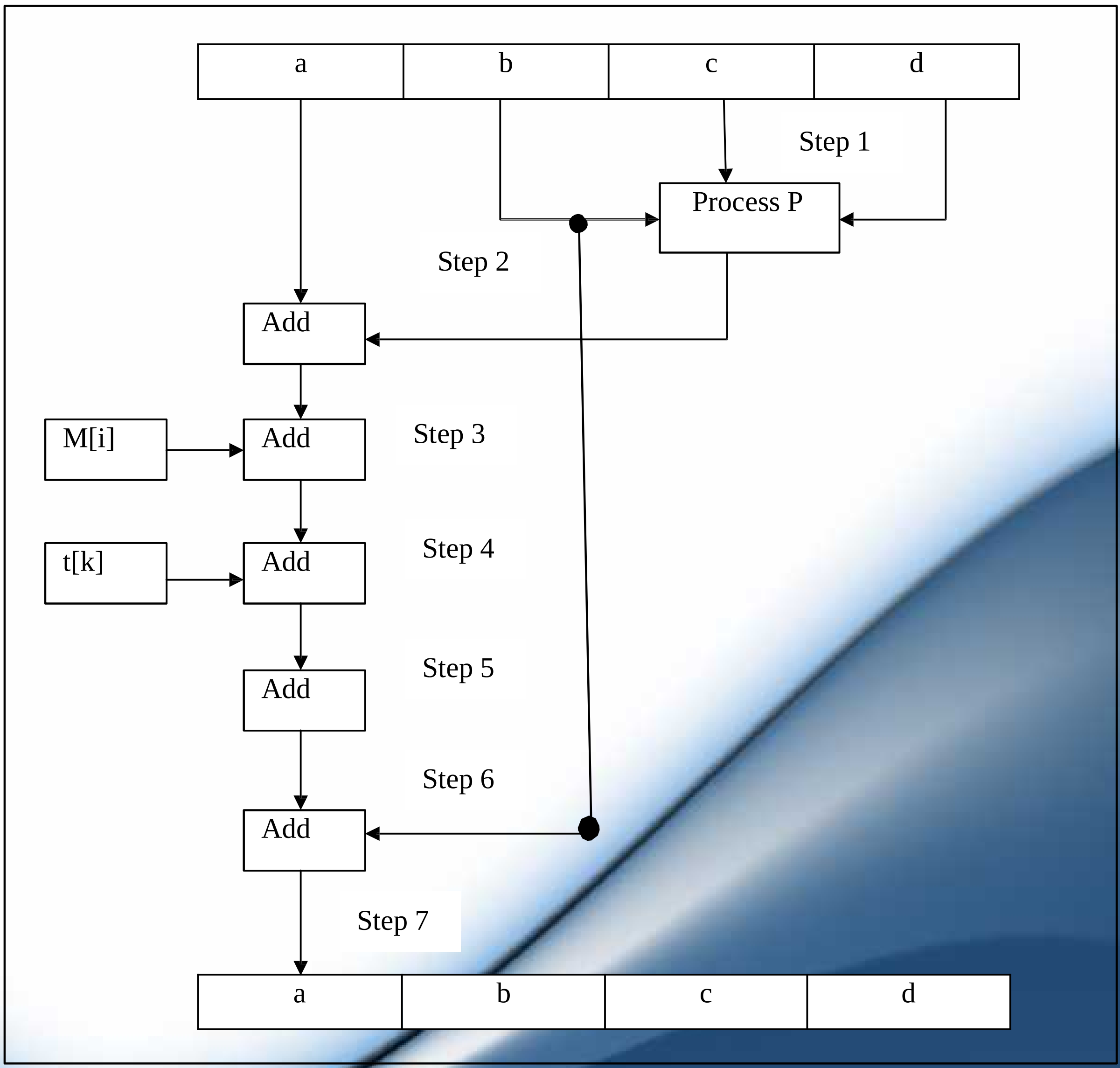


Figure: One MD5 operation

We can mathematically express a single MD5 operation as follows:

a = b + ((a + Process P (b, c, d) + M[i] + T[k]) <<< s)

Where,

a, b, c, d	= Chaining variables, as described earlier
Process P	= A non-linear operation, as described subsequently
M[i]	= M[q x 16 + i], which is the ith 32-bit word in the qth 512-bit block of the message
t[k]	= A constant, as discussed subsequently
<<< s	= Circular-left shift by s bits

Understanding the process P

The process P is nothing but some basic Boolean operations on b, c and d. This is shown below in the table.

Note that in the four rounds, only the process P differs. All the other steps remain the same. Thus, we can substitute the actual details of process P in each of the round and keep everything else constant.

Round	Process P
1	(b AND c) OR ((NOT b) AND (d))
2	(b AND d) OR (c AND (NOT d))
3	B XOR c XOR d
4	C XOR (b OR (NOT d))

Table: Process P in each round

Q13) What is Hash? Discuss briefly SHA-1 and compare it with predecessor MD5?

Ans:

Hash

- A Hash is a transformation of data into distilled forms that are unique to the data and is a one way function.
- In Hashing, a fixed length message digest is created out of a variable length message. A digest is normally much smaller than message.
- A hash function accept a variable size message M as input and produce fixed size output, referred to as Hash Code H(M).
- Hash code is a function of all the bits of message and provides an error detection capability: A change to any bit or bits in message result in a change to the hash code.

SHA-1

- SHA-1(Secure hash algorithm) was proposed by NIST as a message digests function.
- SHA-1 takes a message of length at most 2^{64} bits and produces a 160 bit output.
- It is similar to the MD5 message digest function, but it is a little slower to execute and presumably more secure.
- MD5 made four passes over each block of data; SHA-1 makes five. It also produce a 160 bit digest as opposed o the 128 of the MDs.

SHA-1 Message Padding

SHA-1 pads message in same manner as MD4 and MD5, except that SHA-1 is not different for a message that is longer than 2^{64} bits. If there were such a message it would take several hundred years to transmit it at 10 Gigabits per second, and it would take even longer to compute the SHA-1 message digest for it at 1000 MBPS.

Overview of SHA-1 Message Digest

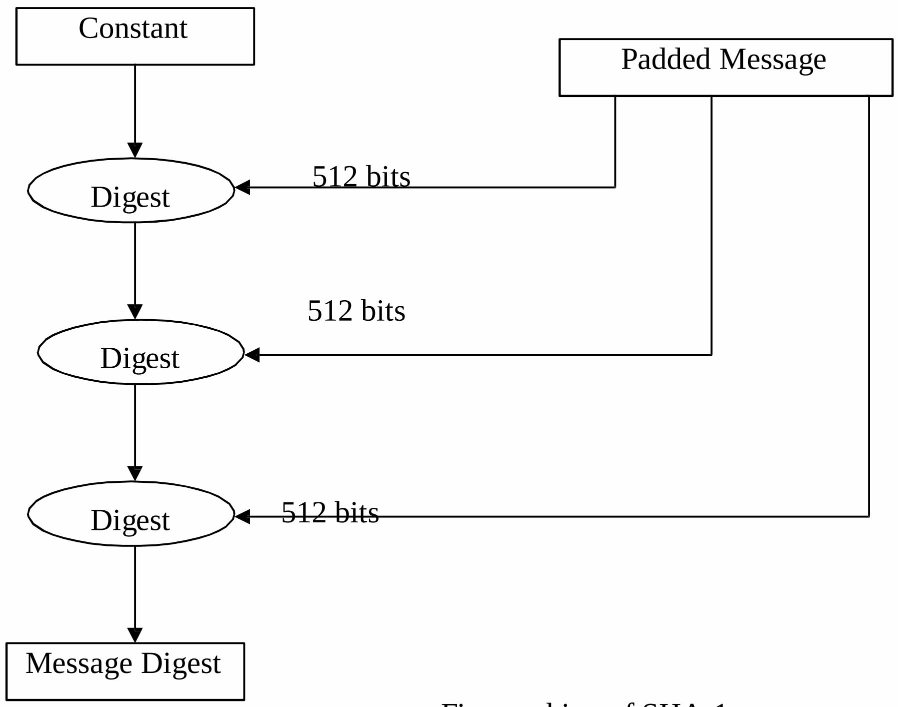


Fig: working of SHA-1

Like MD5, SHA-1 operates in stages. Each Stage mangles the pre-stage message digest by a sequence of operations based on the current message block.

At the end of the stage, each word of the mangled message digest is added to its pre-stage value to produce the post-stage value.

Therefore, the current value of the message digest must be saves at the beginning of the stage so that it can be added in the end of the stage.

The 160 bit message digest consist of 5 32-bit words. Consider these words are A,B,C,D and E.

Before first stage the values are A= 67452301_{16} , B=$efcdab89_{16}$, C= $98badcfe_{16}$,

D= 10325476_{16}, E=$c3d2e1f0_{16}$.After last stage, the value of A|B|C|D|E is the message digest for entire message.

SHA-1 Operation on a 512-bit Block

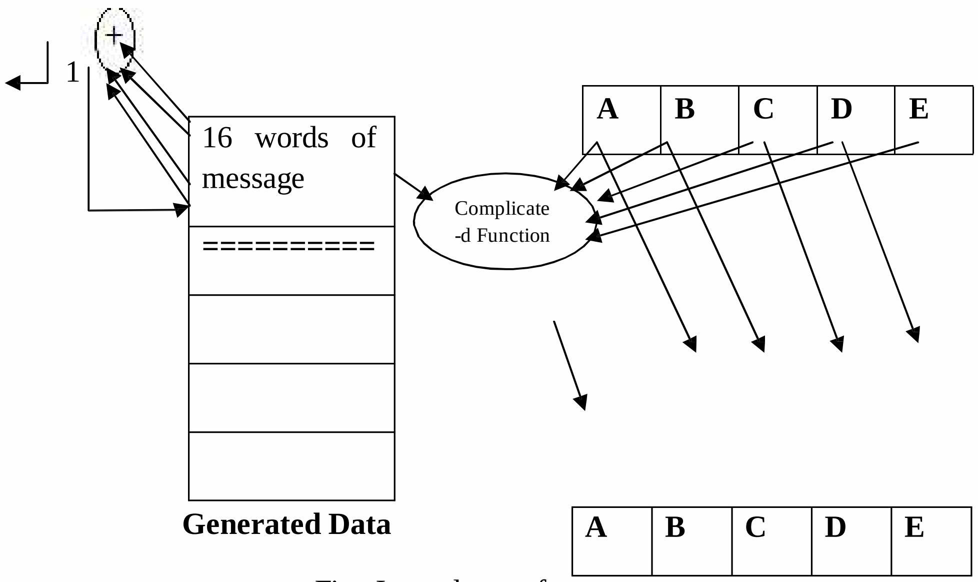


Fig. Inner loop of SHA-1 80 iteration per block

At the start of each stage, the 512-bit message block is used to create a 5×512-bit chunk as shown in above fig.

The first 512-bits of the chunk consist of the message block. The rest get filled in, a 32-bit word at a time, according to the bizarre rule that the n^{th} word is the XOR of words n-3, n-8, n-14 and n-16. In SHA-1 XOR of words n-3, n-8, n-14 and n-16 is rotated left one bit before stored as word n; this is only modification from the original SHA.

We have a buffer of eighty 32-bit words i.e. W_0 to W_{79}.

For t=0 through 79,modify A,B,C,D and E as follows:

B=old A, C= old B ↵ 30, D= old C , E= old D

Therefore new A depends on old A , B, C, D, and E.

A= E + (A ↵ 5) + W_t + K_t + f(t,B,C,D)

SHA-1 Features:

- The SHA1 is used to compute a message digest for a message or data file that is provided as input.
- The message or data file should be considered to be a bit string.
- The length of the message is the number of bits in the message (the empty message has length 0).
- If the number of bits in a message is a multiple of 8, for compactness we can represent the message in hex.
- The purpose of message padding is to make the total length of a padded message a multiple of 512.
- The SHA1 sequentially processes blocks of 512 bits when computing the message digest.
- The 64-bit integer is l, the length of the original message.
- The padded message is then processed by the SHA1 as n 512-bit blocks.

Comparison of MD5 and SHA-1

Points	MD5	SHA-1
Message Digest Length in bits	120	160
Attack to try and find the original message digest	Requires 2^{128} operations to break in	Requires 2^{160} operations to break in, therefore more secure
Attack to try and find two message producing the same message digest	Requires 2^{64} operations to break in	Requires 2^{80} operations to break in
Successful attack so far	There have been reported attempts to some extent	No such claim so far
Speed	Faster (64 iterations and 128 bit buffer)	Slower(80 iterations and 160 bit buffer)
Software Implementation	Simple, does not need any large programs or complex tables.	Simple, does not needs any large programs or complex tables.

Q14) Discuss various PKI trust models?
Ans:

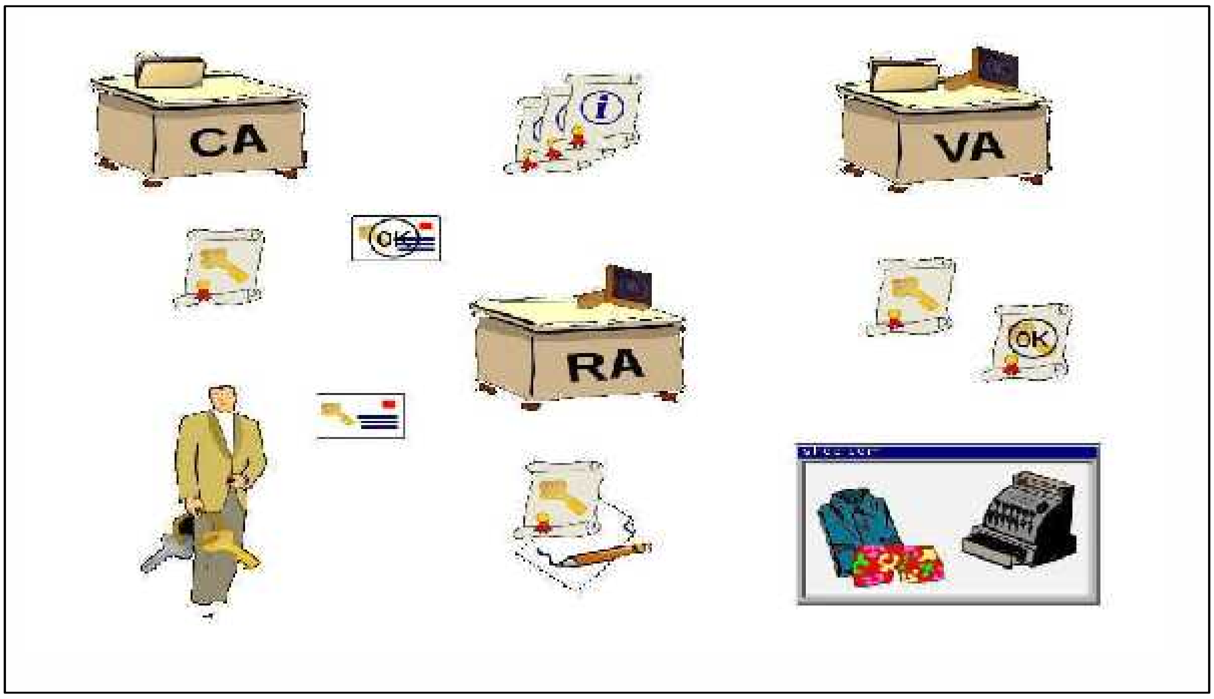


Diagram of a public key infrastructure

Introduction

Public Key Infrastructure (**PKI**) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates. n cryptography, a **PKI** is an arrangement that binds public keys with respective user identities by means of a certificate authority (**CA**). The user identity must be unique within each CA domain. The binding is established through the registration and issuance process, which, depending on the level of assurance the binding has, may be carried out by software at a CA, or under human supervision. The PKI role that assures this binding is called the Registration Authority (**RA**). For each user, the user identity, the public key, their binding, validity conditions and other attributes are made unforgettable in public key certificates issued by the CA.

PKI trust models

Monopoly Model

- In this model, the world chooses one organization, universally trusted by all companies, countries, universities, and other organizations to be the single CA for the world.
- The key of that one organization is embedded in all software and hardware as the PKI trust anchor. Everyone must get certificate from it. This model is a wonderfully simple

model, mathematically. This is favored by organizations hoping to be monopolist. However, there are problems with it:

- There is no one universally trusted organization.
- Given that all software and hardware would come preconfigured with the monopoly organization's key, it would be infeasible to ever change that key in case it were compromised, since that would involve reconfiguration of every piece of equipment and software.
- It would be expensive and insecure to have a remote organization certify your key. How would they know it was you? How would you be able to securely send them your public key? Although transmission of the public key does not require secrecy, it requires integrity. Otherwise the CA could be tricked into certifying the public key as yours.
- Once enough software and hardware was deployed so that it would be difficult for the world to switch organizations, the organization would have monopoly control, and could charge whatever it wanted for granted certificates.
- The entire security of the world rest on that one organization never having an incompetent or corrupt employee who might be bribed or tricked into issuing bogus certificates or divulging the CA's private key.

Monopoly plus Registration Authorities (RAs)

- This model is just like Monopoly model except that the single CA chooses other organization (Know as RAs) to securely check identities and obtains and vouch for public keys. The RA then securely communicates with the CA, perhaps by sending signed email with the information that would go into the certificate, and the CA can than issue a certificate because it trusts the RA.
- This model's advantage over the Monopoly model is that it is more convenient and secure to obtain certificates, since there are more places to go to get certified. However, all the other disadvantages of the monopoly model apply.

Delegated CAs

- In this model the trust anchor CA can issue certificates to other CAs, vouching for their keys and vouching for their trustworthiness as CAs. Users can then obtain certificates from one of the delegated CAs instead of having to go to the trust anchor CA.
- The difference between a delegated CA and an RA is whether Alice sees a chain of certificates from a trusted anchor to Bob's name, or sees a single certificate. Assuming a monopoly trust anchor, this has security and operational properties similar to Monopoly plus Registration Authorities (RAs). Chains of certificates through delegated CAs can be incorporated into any of the models.

Oligarchy

- This is the model commonly used in browsers. In this model, instead of having products preconfigured with one single key, the products come configured with many trust anchors, and a certificate issued by any one of them is accepted. Usually in such a model it is possible for the user to examine and edit the list of trust anchors, adding or deleting trust anchors.
- It has the advantage over the monopoly models that the organizations chosen as trust anchors will be in competition with each other, so the world might be spared monopoly pricing.

Anarchy Model

- This is the model used by PGP (pretty good privacy). Each user is responsible for configuring some trust anchors, for instance, public keys of people he has met and who have handed him a business card with PGP fingerprint (the message digest of the public key), and sent him email containing a public key with that digest. Then anyone can sign certificates for anyone else.
- Some organizations (for instance, MIT does this today) volunteer to keep a certificate database into which anyone can deposit certificates. To get key of someone whose key is not in your set of trust anchors, you can search through the public database to see if you can find a path from one of your trust anchors, you can search through the public database to see if you can find a path from one of your trust anchors to the name you want. This absolutely eliminates the monopoly pricing, but it is really unworkable on a large scale.

Top-Down with Name Constraints

- This model is similar to the monopoly model in that everyone must be configured with a preordained, never changing root key, and that root CA delegates to other CAs. However, the delegated CAs are only allowed to issue certificates for their portions of the namespace.
- In this model it is easy to find the path to a name (just follow the namespace from the root down). But it has the other problems of the monopoly model, in that everyone has to agree upon a root organization, and that organization and its key would be prohibitively expensive to ever replace

Bottom-Up with Name Constraints

- Each organization can create its own PKI and link it to others.
- Parent certifies child (downlink), but child certifies parent as well (uplink).
- Cross links are links from one node to another one.

THANK YOU