SECRET(SYMMETRIC) KEY CRYPTOGRAPHY

Structure

4.0 Cryptography Introduction

4.1 Types of Cryptography

4.2. Block Cipher Modes of Operation

4.0 Cryptography Introduction

Cryptography is the study and practice of techniques for secure communication in the presence of third parties called adversaries. It deals with developing and analysing protocols that prevents malicious third parties from retrieving information being shared between two entities thereby following the various aspects of information security. Secure Communication refers to the scenario where the message or data shared between two parties can't be accessed by an adversary. In Cryptography, an Adversary is a malicious entity, which aims to retrieve precious information or data thereby undermining the principles of information security. Data Confidentiality, Data Integrity, Authentication and Non-repudiation are core principles of modern-day cryptography.

1. **Confidentiality** refers to certain rules and guidelines usually executed under confidentiality agreements which ensure that the information is restricted to certain people or places.
2. **Data integrity** refers to maintaining and making sure that the data stays accurate and consistent over its entire life cycle.
3. **Authentication** is the process of making sure that the piece of data being claimed by the user belongs to it.
4. **Non-repudiation** refers to the ability to make sure that a person or a party associated with a contract or a communication cannot deny the authenticity of their signature over their document or the sending of a message.

Consider two parties Alice and Bob. Now, Alice wants to send a message m to Bob over a secure channel. So, what happens is as follows. The sender's message or sometimes called the Plaintext, is converted into an unreadable form using a Key k. The resultant text obtained is called the Ciphertext. This process is known as Encryption.

At the time of received, the Ciphertext is converted back into the plaintext using the same Key k, so that it can be read by the receiver. This process is known as Decryption.

Alice (Sender) Bob (Receiver)

C = E (m, k) ----> m = D (C, k)

Here, C refers to the Ciphertext while E and D are the Encryption and Decryption algorithms respectively. Let's consider the case of Caesar Cipher or Shift Cipher as an example. As the name suggests, in Caesar's Cipher each character in a word is replaced by another character under some defined rules. Thus, if A is replaced by D, B by E and so on. Then, each character in the word would be shifted by a position of 3 shown in table

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

For example:

Plaintext: *Meet me at home*

Ciphertext: *Phhw ph dw krph*

Note: *Even if the adversary knows that the cipher is based on Caesar's Cipher, it cannot predict the plaintext as it doesn't have the key in this case which is to shift the characters back by three places.*

There are several types of cryptography, each with its own unique features and applications. Some of the most common types of cryptography include:

1. Secret (Symmetric)-key cryptography: This type of cryptography involves the use of a single key to encrypt and decrypt data. Both the sender and receiver use the same key, which must be kept secret to maintain the security of the communication.

2. Public (Asymmetric)-key cryptography: Asymmetric-key cryptography, also known as public-key cryptography, uses a pair of keys – a public key and a private key – to encrypt and decrypt data. The public key is available to anyone, while the private key is kept secret by the owner.

Hash functions: A hash function is a mathematical algorithm that converts data of any size into a fixed-size output. Hash functions are often used to verify the integrity of data and ensure that it has not been tampered with.

Applications of Cryptography:

Cryptography has a wide range of applications in modern-day communication, including:

- **Secure online transactions:** Cryptography is used to secure online transactions, such as online banking and e-commerce, by encrypting sensitive data and protecting it from unauthorized access.
- **Digital signatures:** Digital signatures are used to verify the authenticity and integrity of digital documents and ensure that they have not been tampered with.
- **Password protection:** Passwords are often encrypted using cryptographic algorithms to protect them from being stolen or intercepted.

Military and intelligence applications: Cryptography is widely used in military and intelligence applications to protect classified information and communications.

Challenges of Cryptography:

While cryptography is a powerful tool for securing information, it also presents several challenges, including:

- **Key management**: Cryptography relies on the use of keys, which must be managed carefully to maintain the security of the communication.
- **Quantum computing:** The development of quantum computing poses a potential threat to current cryptographic algorithms, which may become vulnerable to attacks.
- **Human error:** Cryptography is only as strong as its weakest link, and human error can easily compromise the security of a communication.

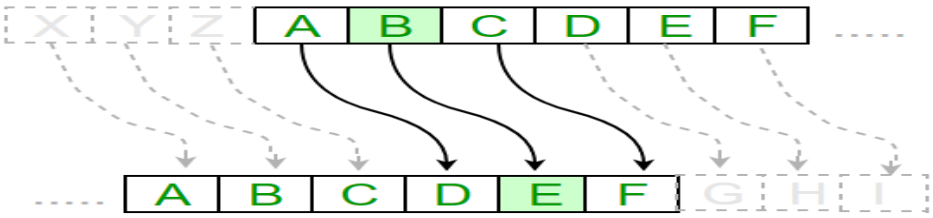
Cryptography

The word ‘cryptography’ originated from two greek words ‘Krypto’ means *hidden* and ‘graphene’ means *writing*.

Classical Cryptography

The roots are cryptography are found in Roman and Egyptian civilizations. Below are some of the ancient types of cryptography:

1. Caesar Cipher: The ancient Greeks were well known for the use of Ciphers. The Caesar Cipher or Shift Cipher is one of the earliest and simplest well-known cryptographic techniques. It is a form of Substitution Cipher where each character in a word is replaced by a fixed number of positions. For example with a shift of 3, A is replaced by D, B by E, and so on.



2. Vigenere Cipher: During the 16th century, Vigenere designed a cipher in which the encryption key is repeated multiple times spanning the entire message, and then the cipher text is generated by adding the message character with key character modulo 26. This approach is also vulnerable to attacks, where the secrecy of the message depends on the secrecy of the encryption key.

3. Enigma machine: Cryptography played a vital in the victory of Allied forces during World War I and World War II. World War II prominently saw the use of electromechanical cipher machines. The story of the Allied victory over the Germans by cracking the world-famous Enigma machine is well known. Like all rotor machines, Enigma is a combination of electro-mechanical subsystems. It consisted of somewhat three to five rotors. Whenever a key was pressed, one or more rotors rotated on the spindle, and accordingly, the key was scrambled to something else. The Enigma cipher was broken by Poland.

Data Encryption Standard (DES)

In the early 1970s, IBM realized that its customer base is requesting some type of encryption method to protect the data. They formed a crypto group headed by Horst-Feistel. This group designed a cipher called Lucifer. In 1973, the Nation Bureau of Standards (NBS) which is now known as the National Institute of Standards and

Technology (NIST) put out a proposal for the block cipher. Lucifer was eventually accepted and called **Data Encryption Standard (DES).**

- It is a symmetric-key algorithm based on the Feistel cipher and is used for the encryption of electronic data.
- It has a relatively small key size of 56-bits and is encrypted 64 bits or 8 characters at a time.
- In 1997, DES was broken by an exhaustive search attack.
- But, it was later discontinued as it was found to be insecure, especially against brute force attacks cause of its relatively small key size.

Advance Encryption Standard (AES)

In 1997, NIST again put out a proposal for a new block cipher. The Rijndael cipher is eventually accepted and renamed as **Advanced Encryption Standard (AES).**

- DES was replaced by Advance Encryption Standard or AES in 2001.
- Unlike DES, AES is based on a substitution-permutation network.
- AES is a sub-set of Rijndael.
- It is a family of ciphers with different key and block sizes.
- In the case of AES, the block size is 128 bits or 16 characters which means 16 characters can be encrypted at a time.
- It comes with three different key size variants: 128 bits, 192 bits, and 256 bits.

Features Of Cryptography are as follows:

1. **Confidentiality:** Information can only be accessed by the person for whom it is intended and no other person except him can access it.
2. **Integrity:** Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.
3. **Non-repudiation:** The creator/sender of information cannot deny his intention to send information at later stage.
4. **Authentication:** The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

4.1 Types of Cryptography

In general there are three types Of cryptography:

1. **Symmetric Key Cryptography:** It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system are Data Encryption System(DES) and Advanced Encryption System(AES).
2. **Hash Functions:** There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.
3. **Asymmetric Key Cryptography:** Under this system a pair of keys is used to encrypt and decrypt information. A receiver's public key is used for encryption and a receiver's private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone known his private key. The most popular asymmetric key cryptography algorithm is RSA algorithm.

Applications Of Cryptography:

1. **Computer passwords:** Cryptography is widely utilized in computer security, particularly when creating and maintaining passwords. When a user logs in, their password is hashed and compared to the hash that was previously stored. Passwords are hashed and encrypted before being stored. In this technique, the passwords are encrypted so that even if a hacker gains access to the password database, they cannot read the passwords.
2. **Digital Currencies:** To safeguard transactions and prevent fraud, digital currencies like Bitcoin also use cryptography. Complex algorithms and cryptographic keys are used to safeguard transactions, making it nearly hard to tamper with or forge the transactions.
3. **Secure web browsing:** Online browsing security is provided by the use of cryptography, which shields users from eavesdropping and man-in-the-

middle assaults. Public key cryptography is used by the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols to encrypt data sent between the web server and the client, establishing a secure channel for communication.

4. **Electronic signatures:** Electronic signatures serve as the digital equivalent of a handwritten signature and are used to sign documents. Digital signatures are created using cryptography and can be validated using public key cryptography. In many nations, electronic signatures are enforceable by law, and their use is expanding quickly.
5. **Authentication:** Cryptography is used for authentication in many different situations, such as when accessing a bank account, logging into a computer, or using a secure network. Cryptographic methods are employed by authentication protocols to confirm the user's identity and confirm that they have the required access rights to the resource.
6. **Cryptocurrencies:** Cryptography is heavily used by cryptocurrencies like Bitcoin and Ethereum to safeguard transactions, thwart fraud, and maintain the network's integrity. Complex algorithms and cryptographic keys are used to safeguard transactions, making it nearly hard to tamper with or forge the transactions.
7. **End-to-End Encryption:** End-to-end encryption is used to protect two-way communications like video conversations, instant messages, and email. Even if the message is encrypted, it assures that only the intended receivers can read the message. End-to-end encryption is widely used in communication apps like WhatsApp and Signal, and it provides a high level of security and privacy for users.

Advantages

1. **Access Control:** Cryptography can be used for access control to ensure that only parties with the proper permissions have access to a resource. Only those with the correct decryption key can access the resource thanks to encryption.
2. **Secure Communication:** For secure online communication, cryptography is crucial. It offers secure mechanisms for transmitting private information

like passwords, bank account numbers, and other sensitive data over the internet.

3. **Protection against attacks:** Cryptography aids in the defence against various types of assaults, including replay and man-in-the-middle attacks. It offers strategies for spotting and stopping these assaults.
4. **Compliance with legal requirements:** Cryptography can assist firms in meeting a variety of legal requirements, including data protection and privacy legislation.

4.2. Block Cipher Modes of Operation

In this chapter, we will discuss the different modes of operation of a block cipher. These are procedural rules for a generic block cipher. Interestingly, the different modes result in different properties being achieved which add to the security of the underlying block cipher.

A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

Electronic Code Book (ECB) Mode

This mode is a most straightforward way of processing a series of sequentially listed message blocks.

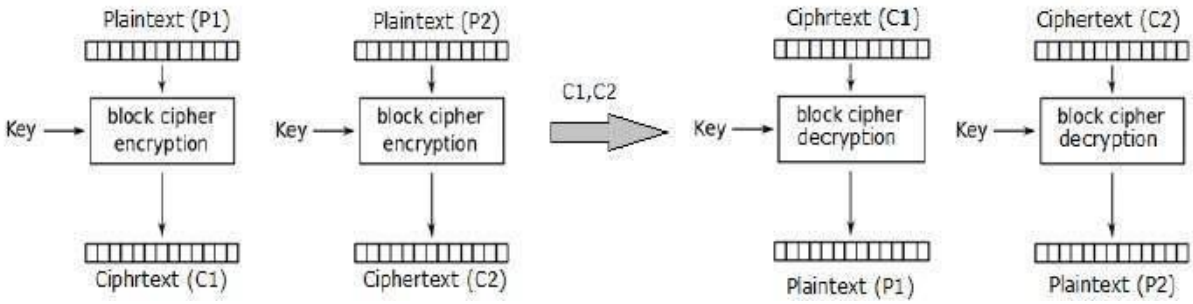
Operation

- The user takes the first block of plaintext and encrypts it with the key to produce the first block of ciphertext.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

The ECB mode is **deterministic**, that is, if plaintext block P1, P2,…, Pm are encrypted twice under the same key, the output ciphertext blocks will be the same.

In fact, for a given key technically we can create a codebook of ciphertexts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding ciphertext. Thus, the operation is analogous to the assignment

of code words in a codebook, and hence gets an official name − Electronic Codebook mode of operation (ECB). It is illustrated as follows −



Analysis of ECB Mode

In reality, any application data usually have partial information which can be guessed. For example, the range of salary can be guessed. A ciphertext from ECB can allow an attacker to guess the plaintext by trial-and-error if the plaintext message is within predictable.

For example, if a ciphertext from the ECB mode is known to encrypt a salary figure, then a small number of trials will allow an attacker to recover the figure. In general, we do not wish to use a deterministic cipher, and hence the ECB mode should not be used in most applications.

Cipher Block Chaining (CBC) Mode

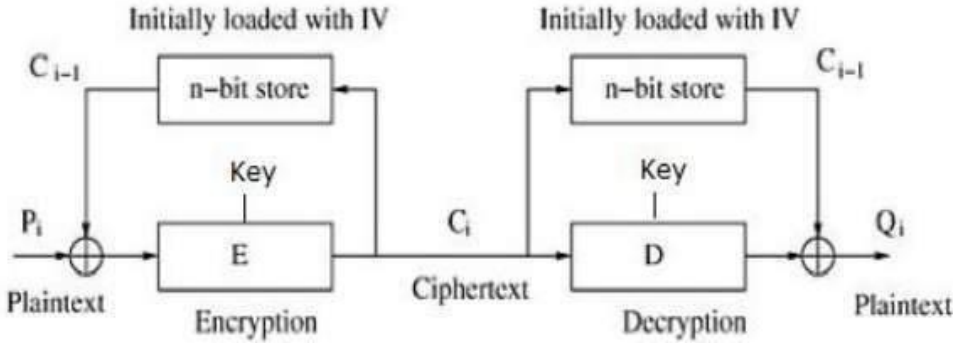
CBC mode of operation provides message dependence for generating ciphertext and makes the system non-deterministic.

Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows

- Load the n-bit Initialization Vector (IV) in the top register.
- XOR the n-bit plaintext block with data value in top register.
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.

- For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.



Analysis of CBC Mode

In CBC mode, the current plaintext block is added to the previous ciphertext block, and then the result is encrypted with the key. Decryption is thus the reverse process, which involves decrypting the current ciphertext and then adding the previous ciphertext block to the result.

Advantage of CBC over ECB is that changing IV results in different ciphertext for identical message. On the drawback side, the error in transmission gets propagated to few further block during decryption due to chaining effect.

It is worth mentioning that CBC mode forms the basis for a well-known data origin authentication mechanism. Thus, it has an advantage for those applications that require both symmetric encryption and data origin authentication.

Cipher Feedback (CFB) Mode

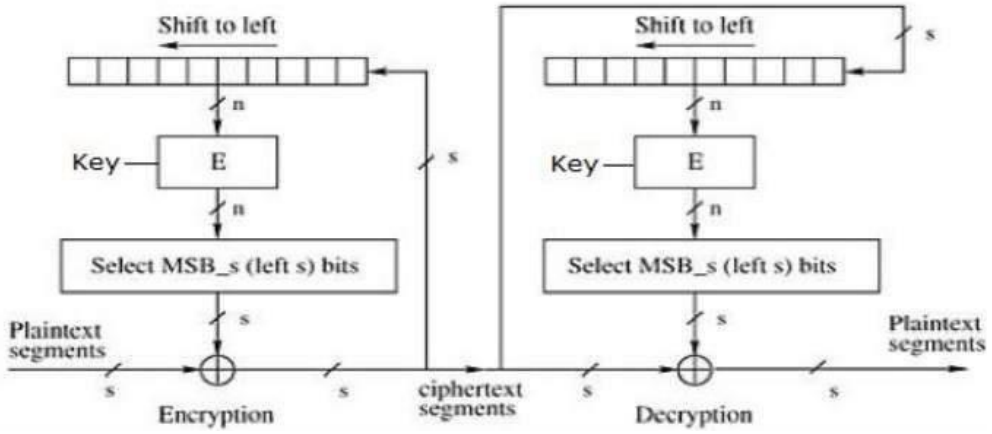
In this mode, each ciphertext block gets ‘fed back’ into the encryption process in order to encrypt the next plaintext block.

Operation

The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size ‘s’ bits where 1 < s < n. The CFB mode requires an initialization vector (IV) as the initial random n-bit input block. The IV need not be secret. Steps of operation are −

- Load the IV in the top register.

- Encrypt the data value in top register with underlying block cipher with key K.
- Take only ‘s’ number of most significant bits (left bits) of output of encryption process and XOR them with ‘s’ bit plaintext message block to generate ciphertext block.
- Feed ciphertext block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed.
- Essentially, the previous ciphertext block is encrypted with the key, and then the result is XORed to the current plaintext block.
- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption.



Analysis of CFB Mode

CFB mode differs significantly from ECB mode, the ciphertext corresponding to a given plaintext block depends not just on that plaintext block and the key, but also on the previous ciphertext block. In other words, the ciphertext block is dependent of message.

CFB has a very strange feature. In this mode, user decrypts the ciphertext using only the encryption process of the block cipher. The decryption algorithm of the underlying block cipher is never used.

Apparently, CFB mode is converting a block cipher into a type of stream cipher. The encryption algorithm is used as a key-stream generator to produce key-stream that is placed in the bottom register. This key stream is then XORed with the plaintext as in case of stream cipher.

By converting a block cipher into a stream cipher, CFB mode provides some of the advantageous properties of a stream cipher while retaining the advantageous properties of a block cipher.

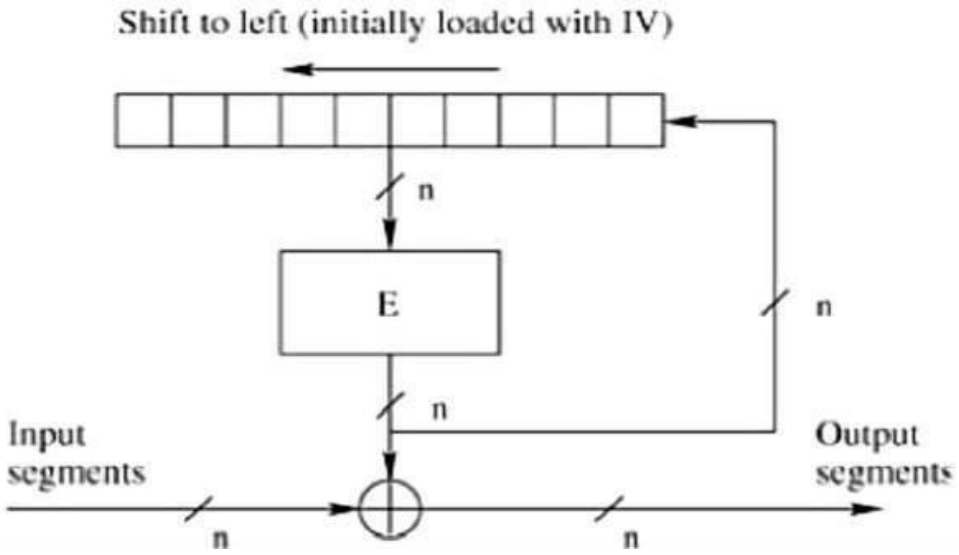
On the flip side, the error of transmission gets propagated due to changing of blocks.

Output Feedback (OFB) Mode

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

The operation is depicted in the following illustration −



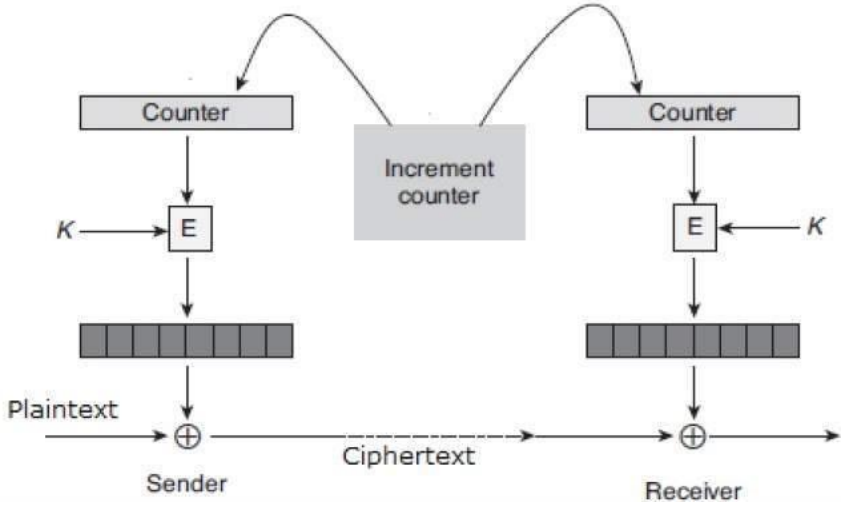
Counter (CTR) Mode

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are −

- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.
- Encrypt the contents of the counter with the key and place the result in the bottom register.
- Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode.
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.



Analysis of Counter Mode

It does not have message dependency and hence a ciphertext block does not depend on the previous plaintext blocks.

Like CFB mode, CTR mode does not involve the decryption process of the block cipher. This is because the CTR mode is really using the block cipher to generate a key-stream, which is encrypted using the XOR function. In other words, CTR mode also converts a block cipher to a stream cipher.

The serious disadvantage of CTR mode is that it requires a synchronous counter at sender and receiver. Loss of synchronization leads to incorrect recovery of plaintext.

However, CTR mode has almost all advantages of CFB mode. In addition, it does not propagate error of transmission at all.

SECRET (SYMMETRIC) KEY CRYPTOGRAPHY

STRUCTURE

5.0 Secret Key Cryptography

Let us briefly review Symmetric Key Cryptography. Symmetric Key Cryptography is referred to by various other terms, such as **Secret Key Cryptography** or **Private Key Cryptography.** In this scheme, only one key is used and the same key is used for both encryption and decryption of messages. Obviously, both the parties must agree upon the key before any transmission begins and nobody else should know about it. Figure 3.17 shows how symmetric key cryptography works. Basically at the sender's end (A), the key transforms the plain text message into a cipher text form. At the receiver's end (B), the same key is used to decrypt the encrypted message, thus deriving the original message out of it.

The first problem here is that of key agreement or key distribution. The problem is: in the first place, how do two parties agree on a key? One solution is that somebody from the sender's end physically visits the receiver and hand over the key. Another way is to courier a paper on which the key is written. Both are not exactly very convenient. A third way is to send the key over the network to B and ask for the confirmation. But then, if an intruder gets the message, he can interpret all the subsequent ones!

Regardless, because these drawbacks can be overcome using intelligent solutions as we shall see and also because symmetric key cryptography has several advantages as well, it is widely used in practice. However, we shall first discuss the most popular computed-based symmetric key cryptographic algorithms and think about solving the problems associated when we discuss asymmetric key cryptography subsequently.

5.1 Data Encryption Standard (DES)

Background and History

The **Data Encryption Standard (DES)**, also called as the Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO, has been a cryptographic algorithm used for over three decades. Of late, DES has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been slightly on the decline. DES is generally used in the ECB, CBC or the CFB mode.

The origins of DES go back to 1972, when in the US, the National Bureau of Standards (NBS), now known as the National Institute of Standards and Technology (NIST) embarked upon a project for protecting the data in computers and computer communications. They wanted to develop a single cryptographic algorithm. After two years, NBS realized that IBM's **Lucifer** could be considered as a serious candidate, rather than developing a fresh algorithm from scratch. After a few discussions, in 1975, the NBS published the details of the algorithm. Towards the end of 1976, the US Federal Government decided to adopt this algorithm and soon, it was renamed as *Data Encryption Standard (DES)*. Soon, other bodies also recognized and adopted DES as a cryptographic algorithm.

How DES Works

Basic Principles DES is a block cipher. It encrypts data in blocks of size 64 bits each. That is, 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is shown in Fig. 5.1.

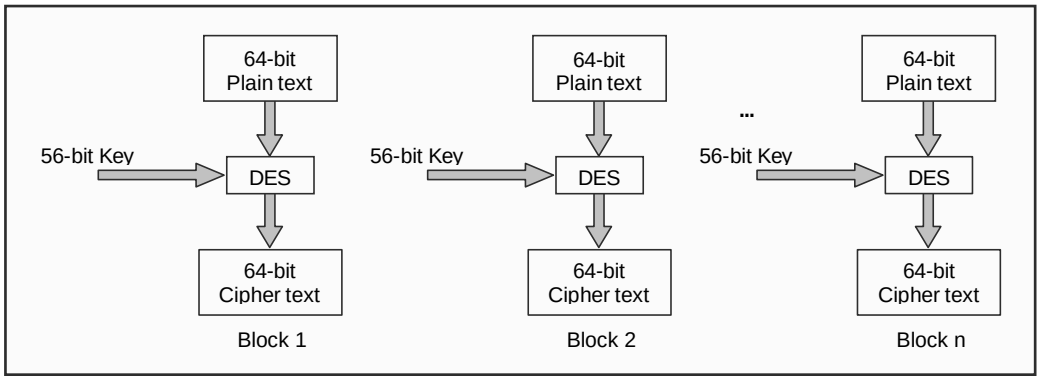


Fig. 5.1 *The conceptual working of DES*

We have mentioned that DES uses a 56-bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every eighth bit of the key is discarded to produce a 56-bit key. That is, bit positions 8, 16, 24, 32, 40, 48, 56 and 64 are discarded. This is shown in Fig. 5.2 with shaded bit positions indicating discarded bits. (Before discarding, these bits can be used for parity checking to ensure that the key does not contain any errors.)

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

Fig. 5.2 *Discarding of every 8th bit of the original key (Shaded bit positions are discarded)*

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key, as shown in Fig. 5.3.

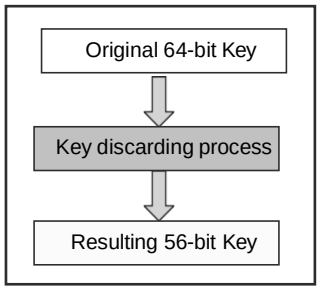


Fig.5.3 56-bit key from the original 64-bit key

Simplistically, DES is based on the two fundamental attributes of cryptography: substitution (also called as confusion) and transposition (also called as diffusion). DES consists of 16 steps, each of which is called as a **round**. Each *round* performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

1. In the first step, the 64-bit plain text block is handed over to an **Initial Permutation (IP)** function.
2. The Initial Permutation is performed on plain text.
3. Next, the Initial Permutation (IP) produces two halves of the permuted block; say Left Plain Text (LPT) and Right Plain Text (RPT).
4. Now, each of LPT and RPT go through 16 *rounds* of encryption process.
5. In the end, LPT and RPT are rejoined and a **Final Permutation (FP)** is performed on the combined block.
6. The result of this process produces 64-bit cipher text. This process is shown

diagrammatically in Fig. 5.4.

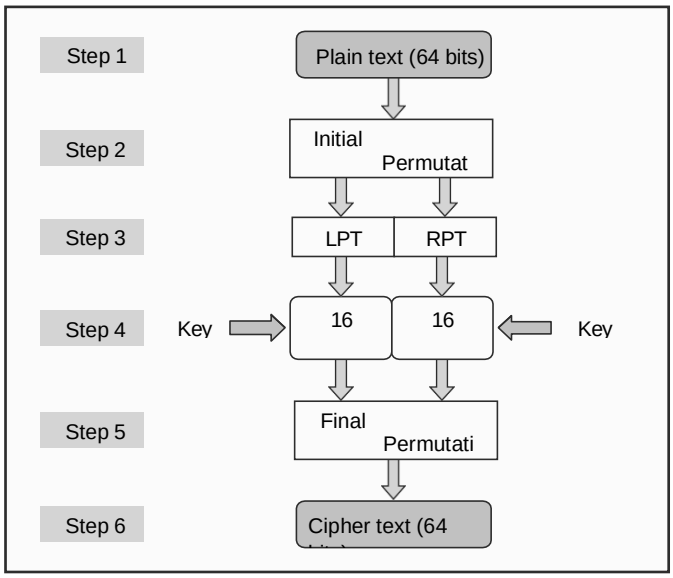


Fig. 5.4 Broad level steps in DES

Let us now understand each of these processes in detail.

Initial Permutation (IP) As we have noted, the Initial Permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in Fig. 5.5. For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text block, the second bit with the 58th bit of the original plain text block and so on. This is nothing but jugglery of bit positions of the original plain text block.

Bit position in the plain text block	To be overwritten with the contents of this bit position
1	58
2	60
3	42
..	..
64	7

Fig. 5.5 Idea of IP

The complete transposition table used by IP is shown in Fig. 5.6. This table (and all others in this chapter) should be read from left to right, top to bottom. For instance, we have noted that 58 in the first position indicates that the contents of the 58th bit in the original plain text block will overwrite the contents of the 1st bit position, during IP. Similarly, 1 is shown at the 40th position in the table, which means that the first bit will overwrite the 40th bit in the original plain text block. The same rule applies for all the other bit positions.

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

Fig. 5.6 Initial Permutation (IP) table

As we have noted, after IP is done, the resulting 64-bit permuted text block is divided into two half blocks. Each half block consists of 32 bits. We have called the left block as LPT and the right block as RPT. Now, 16 *rounds* are performed on these two blocks. This process is described as follows.

Rounds Each of the 16 rounds, in turn, consists of the broad level steps outlined in Fig. 5.7.

Let us discuss these details step-by-step.

Step 1: Key transformation We have noted that the initial 64-bit key is transformed into a 56-bit key by discarding every 8^{th} bit of the initial key. Thus, for each round, a 56-bit key is available. From this 56- bit key, a different 48-bit **sub-key** is generated during each *round* using a process called as **key transformation**.

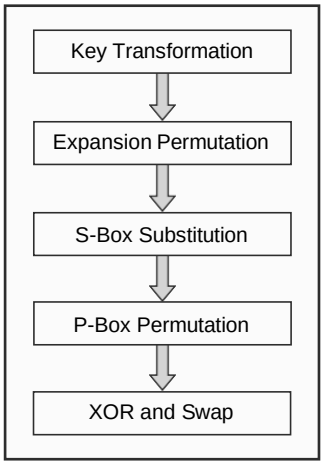


Fig. 5.7 Details of one round in DES

For this, the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round. For example, if the *round* number is 1, 2, 9 or 16, the shift is done by only position. For other rounds, the circular shift is done by two positions. The number of key bits shifted per round is shown in Fig. 5.8.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Fig. 5.8 Number of key bits shifted per round

After an appropriate shift, 48 of the 56 bits are selected. For selecting 48 of the 56 bits, the table shown in Fig. 5.9 is used. For instance, after the shift, bit number 14 moves into the first position, bit number 17 moves into the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce the 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub-set of the original 56-bit key, it is called as **compression permutation**.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Fig. 5.9 Compression permutation

Because of this compression permutation technique, a different subset of key bits is used in each round. That makes DES not so easy to crack.

Step 2: Expansion permutation Recall that after Initial Permutation, we had two 32-bit plain text areas, called as Left Plain Text (LPT) and Right Plain Text (RPT). During **expansion permutation**, the RPT is expanded from 32 bits to 48 bits. Besides increasing the bit size from 32 to 48, the bits are permuted as well, hence the name *expansion permutation*. This happens as follows:

1. The 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits. This is shown in Fig. 5.10.
2. Next, each 4-bit block of the previous step is then expanded to a corresponding 6-bit block. That is, per 4-bit block, 2 more bits are added. What are these two bits? They are actually the repeated first and the fourth bits of the 4-bit block. The second and the third bits are written down as they were in the input. This is shown in Fig. 5.11. Note that the first bit inputted

is outputted to the second output position and also repeats in output position 48. Similarly, the 32nd input bit is found in the 47th output position as well as in the first output position.

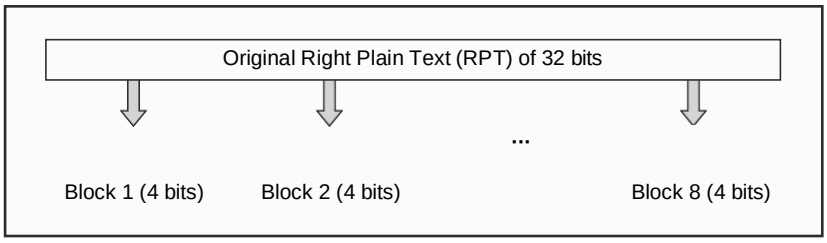


Fig. 5.10 Division of 32-bit RPT into eight 4-bit blocks

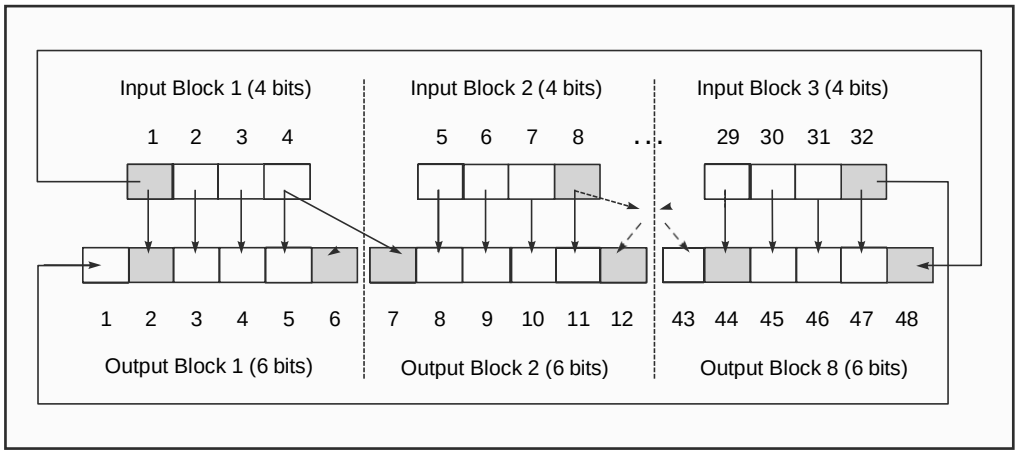


Fig. 5.11 *RPT expansion permutation process*

Clearly, this process results into expansion as well as permutation of the input bits while creating the output.

As we can see, the first input bit goes into the second and the 48th output positions. The second input bit goes into the third output position and so on. Consequently, we will observe that the expansion permutation has actually used the table shown in Fig. 5.12.

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

Fig. 5.12 RPT expansion permutation table

As we have seen, firstly, the *Key transformation* process compresses the 56-bit key to 48 bits. Then, the *Expansion permutation* process expands the 32-bit RPT to 48 bits. Now, the 48-bit key is XORed with the 48-bit RPT and the resulting output is given to

the next step, which is the **S-box Substitution** (which we shall discuss in the next section) as shown in Fig. 5.13.

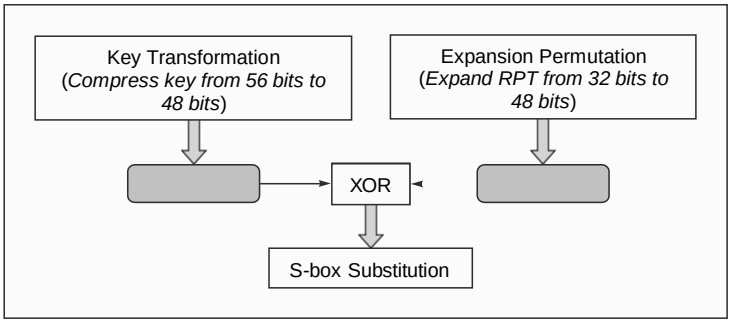


Fig. 5.13 *Way to S-box substitution*

Step 3: S-box substitution **S-box substitution** is a process that accepts the 48-bit input from the XOR operation involving the compressed key and expanded RPT and produces a 32-bit output using the substitution technique. The substitution is performed by eight **substitution boxes** (also called as **S- boxes**). Each of the eight S-boxes has a 6-bit input and a 4-bit output. The 48-bit input block is divided into 8 sub-blocks (each containing 6 bits) and each such sub-block is given to an S-box. The S-box transforms the 6-bit input into a 4-bit output, as shown in Fig. 5.14.

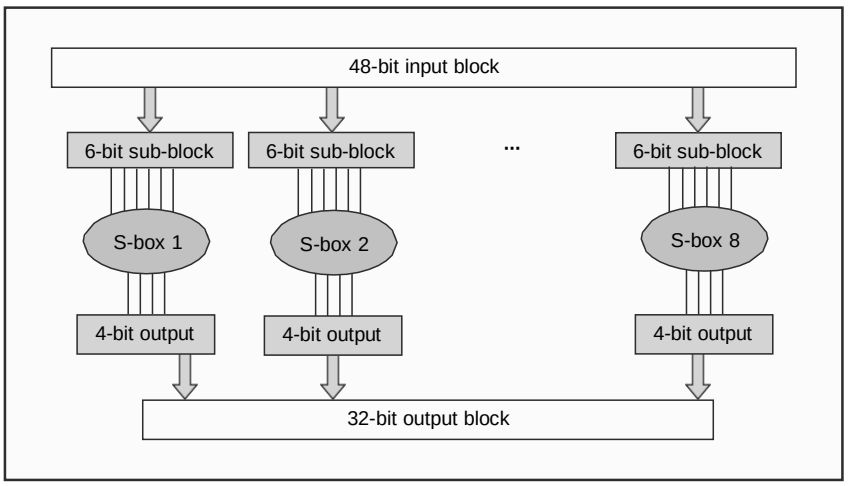


Fig. 5.14 S-box substitution

What is the logic used by S-box substitution for selecting only four of the six bits? We can conceptually think of every S-box as a table that has 4 rows (numbered 0 to 3) and 16 columns (numbered 0 to 15). Thus, we have 8 such tables, one for each S-box. At the intersection of every row and column, a 4-bit number (which will be the 4-bit output for that S-box) is present. This is shown in Fig. 5.15.

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

Fig. 5.15 *(a) S-box 1*

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

Fig. 5.15 *(b) S-box 2*

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

Fig. 5.15 *(c) S-box 3*

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

Fig. 5.15 *(d) S-box 4*

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

Fig. 5.15 *(e) S-box 5*

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

Fig. 5.15 *(f) S-box 6*

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

Fig. 5.15 *(g) S-box 7*

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

Fig. 5.15 *(h) S-box 8*

The 6-bit input indicates which row and column and therefore, which intersection

is to be selected thus, determining the 4-bit output. How is it done? Let us assume that the six bits of a S-box are indicated by b1, b2, b3, b4, b5 and b6. Now, bits b1 and b6 are combined to form a two-bit number. Two bits can store any decimal number between 0 (binary 00) and 3 (binary 11). This specifies the row number. The remaining four bits b2, b3, b4, b5 make up a four-bit number, which specifies the column number between decimal 0 (binary 0000) and 15 (binary 1111). Thus, the 6-bit input automatically selects the row number and column number for the selection of the output. This is shown in Fig. 5.16.

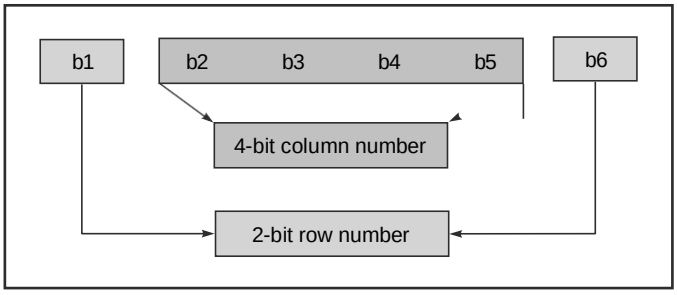


Fig. 5.16 *Selecting an entry in a S-box based on the 6-bit input*

Let us take an example now. Suppose the bits 5 to 8 of the 48-bit input (i.e. the input to the second S- box) contain a value 101101 in binary. Therefore, using our earlier diagram, we have (b1, b6) = 11 in binary (i.e. 3 in decimal) and (b2, b3, b4, b5) = 0110 in binary (i.e. 6 in decimal). Thus, the output of S- box 2 at the intersection of row number 3 and column number 6 will be selected, which is 4. (Remember to count rows and columns from 0, not 1). This is shown in Fig. 5.16.

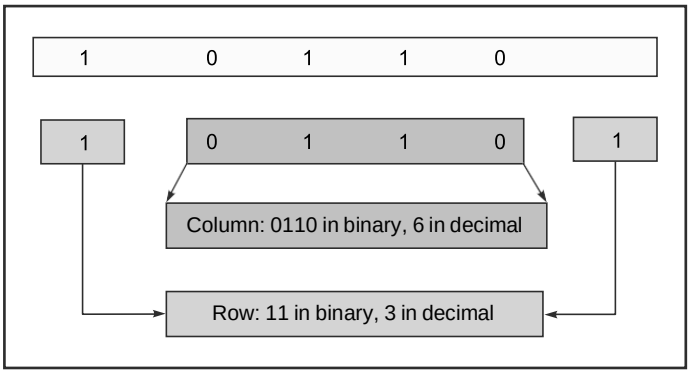


Fig. 5.16 *Example of selection of S-box output based on the input*

The output of each S-box is then combined to form a 32-bit block, which is given to the last stage of a *round*, the P-box Permutation, as discussed next.

Step 4: P-box permutation The output of S-box consists of 32 bits. These 32 bits

are permuted using a **P-box**. This straightforward permutation mechanism involves simple permutation (i.e. replacement of each bit with another bit, as specified in the P-box table, without any expansion or compression). This is called as **P-box Permutation**. The P-box is shown in Fig. 5.17. For example, a 16 in the first block indicates that the bit at position 16 of the original input moves to bit at position 1 in the output and a 10 in the block number 16 indicates that the bit at the position 10 of the original input moves to bit at the position 16 in the output.

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

Fig 5.17 *P-box permutation*

Step 5: XOR and Swap Note that we have been performing all these operations only on the 32-bit right half portion of the 64-bit original plain text (i.e. on the RPT). The left half portion (i.e. LPT) was untouched so far. At this juncture, the left half portion of the initial 64-bit plain text block (i.e. LPT) is XORed with the output produced by P-box permutation. The result of this XOR operation becomes the new right half (i.e. RPT). The old right half (i.e. RPT) becomes the new left half, in a process of swapping. This is shown in Fig. 5.18.

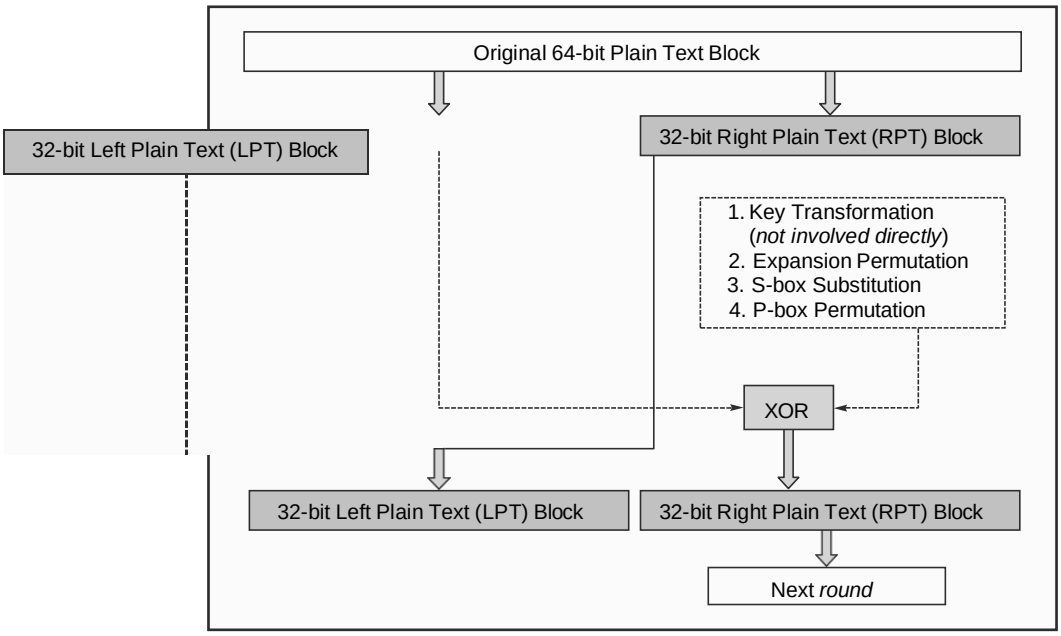


Fig. 5.18 *XOR and swap*

Final Permutation At the end of the 16 rounds, the **Final Permutation** is performed

(only once). This is a simple transposition, based on Fig. 5.19. For instance, the 40th input bit takes the position of the 1st output bit and so on.

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

Fig. 5.19 Final permutation

The output of the Final Permutation is the 64-bit encrypted block.

DES decryption From the above discussion of DES, we might get a feeling that it is an extremely complicated encryption scheme, therefore, the decryption using DES would employ a completely different approach. To most people's surprise, the same algorithm used for encryption in DES also works for decryption! The values of the various tables and the operations as well as their sequence are so carefully chosen that the algorithm is reversible. The only difference between the encryption and the decryption process is the reversal of key portions. If the original key *K* was divided into *K1*, *K2*, *K3*, *..., K16* for the 16 encryption *rounds*, then for decryption, the key should be used as *K16*, *K15*, *K14*, *..., K1*.

5.2 Double DES

Double DES is quite simple to understand. Essentially, it does twice what DES normally does only once. Double DES uses two keys, say *K1* and *K2*. It first performs DES on the original plain text using *K1* to get the encrypted text. It again performs DES on the encrypted text, but this time with the other key, i.e. *K2*. The final output is the encryption of encrypted text (i.e. the original plain text encrypted twice with two different keys). This is shown in Fig. 5.20.

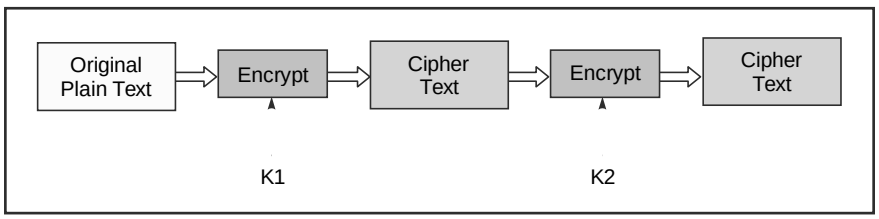


Fig. 5.20 Double DES encryption

Of course, there is no reason why double encryption cannot be applied to other cryptographic algorithms as well. However, in the case of DES, it is already quite popular, therefore, we have discussed this in the context of DES. It should also be quite simple to

imagine that the decryption process would work in exactly the reverse order, as shown in Fig. 5.21.

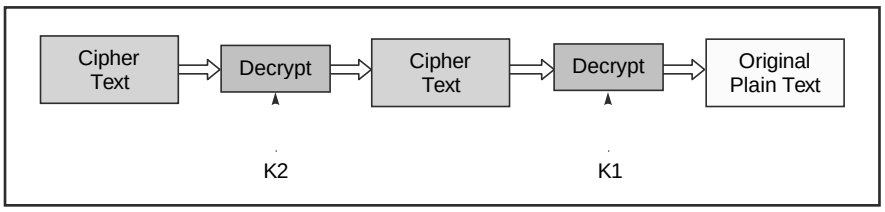


Fig. 5.21 *Double DES decryption*

The doubly encrypted cipher text block is first decrypted using the key *K2* to produce the singly encrypted cipher text. This cipher text block is then decrypted using the key *K1* to obtain the original plain text block.

If we use a key of just 1 bit, there are two possible keys (0 and 1). If we use a 2-bit key, there are four possible key values (00, 01, 10 and 11). In general, if we use an *n*-bit key, the cryptanalyst has to perform $2n$ operations to try out all the possible keys. If we use two different keys, each consisting of *n* bits, the cryptanalyst would need 2^{2n} attempts to crack the key. Therefore, on the face of it, we may think that since the cryptanalysis for the basic version of DES requires a search of 2^{56} keys, Double DES would require a key search of (2^{2*56}) i.e. 2^{128} keys. However, it is not quite true. Merkle and Hellman introduced the concept of the **meet-in-the-middle** attack. This attack involves encryption from one end, decryption from the other and matching the results in the middle, hence the name *meet-in-the-middle* attack.

5.3 Triple DES

Although the *meet-in-the-middle* attack on Double DES is not quite practical yet, in cryptography, it is always better to take the minimum possible chances. Consequently, Double DES seemed inadequate, paving the way for **Triple DES**. As we can imagine, Triple DES is *DES – three times*. It comes in two flavours: one that uses three keys and the other that uses two keys. We will study both, one-by-one.

- ***Triple DES with three keys*** The idea of Triple DES with three keys is illustrated in Fig. 5.22. As we can see, the plain text block P is first encrypted with a key K1, then encrypted with a second key K2 and finally with a third key, K3, where K1, K2 and K3 are all different from each other.

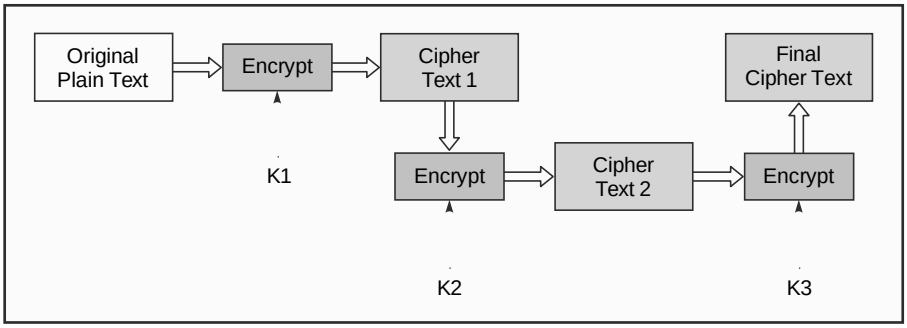


Fig. 5.22 Triple DES with three keys

Triple DES with three keys is used quite extensively in many products, including PGP and S/MIME. To decrypt the cipher text C and obtain the plain text P, we need to perform the operation $P = D_{K3}(D_{K2}(D_{K1}(C)))$.

- ***Triple DES with two keys*** Triple DES with three keys is highly secure. It can be denoted in the form of equation as $C = E_{K3}(E_{K2}(E_{K1}(P)))$. However, Triple DES with three keys also has the drawback of requiring 56 x 3 = 168 bits for the key, which can be slightly difficult to have in practical situations. A workaround suggested by Tuchman uses just two keys for Triple DES. Here, the algorithm works as follows:

1. Encrypt the plain text with key K1. Thus, we have $E_{K1}(P)$.
2. Decrypt the output of Step 1 above with key K2. Thus, we have $D_{K2}(E_{K1}(P))$.
3. Finally, encrypt the output of Step 2 again with key K1. Thus, we have $E_{K1}(D_{K2}(E_{K1}(P)))$.

To decrypt the cipher text C and obtain the original plain text P, we need to perform the operation $P = D_{K1}(E_{K2}(D_{K1}(C)))$.

There is no special meaning attached to the second step of decryption. Its only significance is that it allows Triple DES to work with two, rather than three keys. This is also called as **Encrypt-Decrypt- Encrypt (EDE)** mode. Triple DES with two keys is not susceptible to the *meet in the middle* attack, unlike Double DES as K1 and K2 alternate here.

5.4 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Points to remember

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

Working of the cipher:

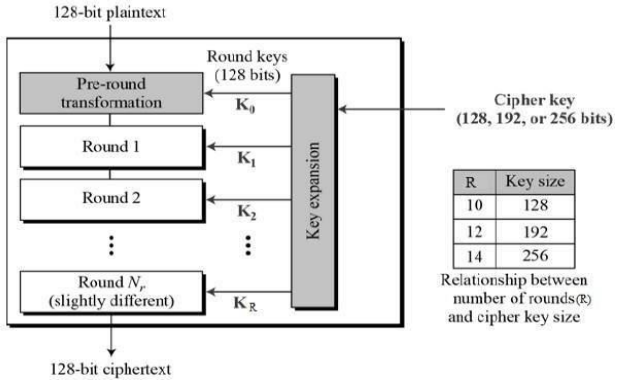
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows:

- 128 bit key – 10 rounds
- 192 bit key – 12 rounds
- 256 bit key – 14 rounds

Creation of Round keys:

A Key Schedule algorithm is used to calculate all the round keys from the key. So, the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Encryption:

AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.

[b0 | b4 | b8 | b12]
[b1 | b5 | b9 | b13]
[b2 | b6 | b10| b14]
[b3 | b7 | b11| b15]

Each round comprises of 4 steps :

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

The last round doesn't have the MixColumns round.

The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.

SubBytes:

This step implements the substitution.

In this step each byte is substituted by another byte. Its performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16 byte (4 x 4) matrix like before.

The next two steps implement the permutation.

ShiftRows:

This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

[b0	b1	b2	b3]		[b0	b1	b2	b3]
[b4	b5	b6	b7]	->	[b5	b6	b7	b4]
[b8	b9	b10	b11]		[b10	b11	b8	b9]
[b12	b13	b14	b15]		[b15	b12	b13	b14]

MixColumns:

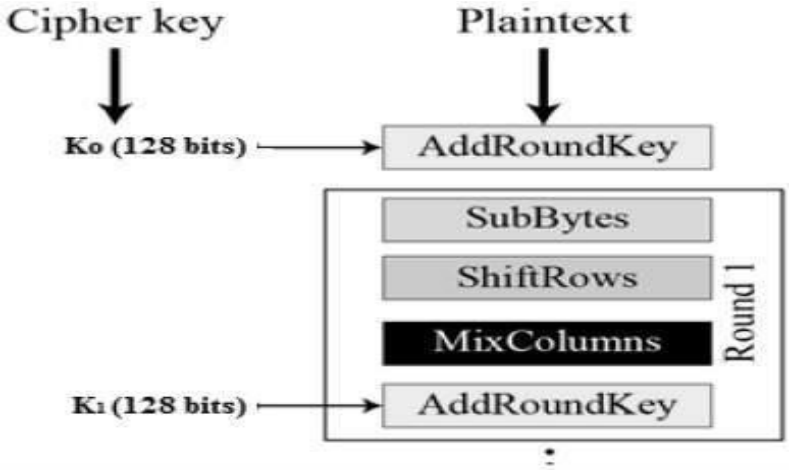
This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

This step is skipped in the last round.

[c0]		[2	3	1	1]	[b0]
[c1]	=	[1	2	3	1]	[b1]
[c2]		[1	1	2	3]	[b2]
[c3]		[3	1	1	2]	[b3]

Add Round Keys:

Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.



After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

Decryption:

The stages in the rounds can be easily undone as these stages have an opposite to it which

when performed reverts the changes.Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption is as follows :

- Add round key
- Inverse MixColumns
- ShiftRows
- Inverse SubByte

The decryption process is the encryption process done in reverse so i will explain the steps with notable differences.

Inverse MixColumns:

This step is similar to the MixColumns step in encryption, but differs in the matrix used to carry out the operation.

$$\begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

Inverse SubBytes:

Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.

Applications:

AES is widely used in many applications which require secure data storage and transmission. Some common use cases include:

- **Wireless security:** AES is used in securing wireless networks, such as Wi-Fi networks, to ensure data confidentiality and prevent unauthorized access.
- **Database Encryption:** AES can be applied to encrypt sensitive data stored in databases. This helps protect personal information, financial records, and other confidential data from unauthorized access in case of a data breach.
- **Secure communications:** AES is widely used in protocols like such as internet communications, email, instant messaging, and voice/video calls.It ensures that the data remains confidential.

- **Data storage:** AES is used to encrypt sensitive data stored on hard drives, USB drives, and other storage media, protecting it from unauthorized access in case of loss or theft.
- **Virtual Private Networks (VPNs):** AES is commonly used in VPN protocols to secure the communication between a user's device and a remote server. It ensures that data sent and received through the VPN remains private and cannot be deciphered by eavesdroppers.
- **Secure Storage of Passwords:** AES encryption is commonly employed to store passwords securely. Instead of storing plaintext passwords, the encrypted version is stored. This adds an extra layer of security and protects user credentials in case of unauthorized access to the storage.
- **File and Disk Encryption:** AES is used to encrypt files and folders on computers, external storage devices, and cloud storage. It protects sensitive data stored on devices or during data transfer to prevent unauthorized access.

AES instruction set is now integrated into the CPU (offers throughput of several GB/s)to improve the speed and security of applications that use AES for encryption and decryption. Even though its been 20 years since its introduction we have failed to break the AES algorithm as it is infeasible even with the current technology. Till date the only vulnerability remains in the implementation of the algorithm.

5.5 Attacks

Secret key cryptography, also known as symmetric key cryptography, relies on a single shared secret key between communicating parties for both encryption and decryption of data. While it is widely used and considered secure when implemented correctly, there are still some attacks that can threaten the security of secret key cryptography. Here are some common attacks:

1. **Brute-Force Attack:** In this attack, the attacker systematically tries all possible keys until the correct one is found. The security of secret key cryptography depends on the key length, and longer keys make brute-force attacks computationally infeasible.
2. **Known Plaintext Attack:** In a known plaintext attack, the attacker has

access to both the plaintext and the corresponding ciphertext. By analyzing multiple pairs of known plaintext-ciphertext, the attacker attempts to deduce the secret key.

3. **Chosen Plaintext Attack:** In a chosen plaintext attack, the attacker can choose specific plaintext inputs and obtain their corresponding ciphertexts. The goal is to use the obtained ciphertexts to deduce the secret key.
4. **Chosen Ciphertext Attack:** In a chosen ciphertext attack, the attacker has the ability to obtain the decryption of chosen ciphertexts under the secret key. This can be used to deduce the secret key.
5. **Cryptanalysis:** Cryptanalysis involves analyzing the cryptographic algorithm or implementation to identify weaknesses or vulnerabilities that could be exploited to deduce the secret key more efficiently than brute-force.
6. **Meet-in-the-Middle Attack:** This attack involves performing a brute-force search for the secret key by exploiting a known plaintext-ciphertext pair and splitting the search space in half.

To mitigate these attacks and ensure the security of secret key cryptography, several best practices should be followed:

Use strong encryption algorithms with appropriate key lengths.

Implement secure key management practices, including key generation, storage, and distribution.

Protect cryptographic keys from unauthorized access and tampering.

Regularly update cryptographic algorithms and libraries to address known vulnerabilities.

Implement countermeasures against side-channel attacks, such as constant-time algorithms and secure hardware design.