CRYPTOGRAPHY

UNIT-II

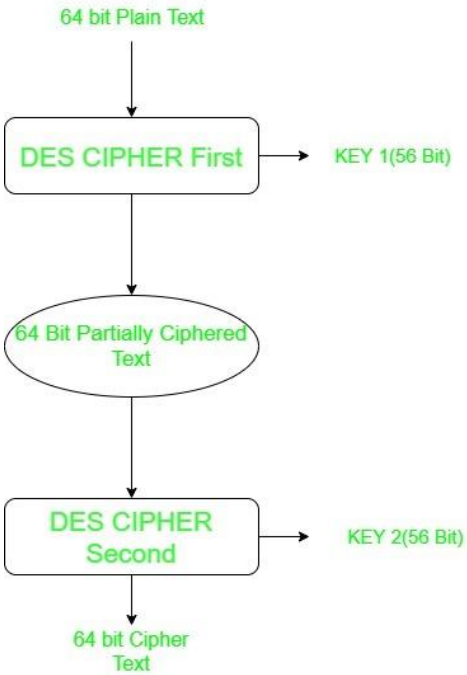
Conventional Encryption Algorithms

Double DES and Triple DES

As we know the Data encryption standard (DES) uses 56 bit key to encrypt any plain text which can be easily be cracked by using modern technologies. To prevent this from happening double DES and triple DES were introduced which are much more secured than the original DES because it uses 112 and 168 bit keys respectively. They offer much more security than DES.

Double DES:

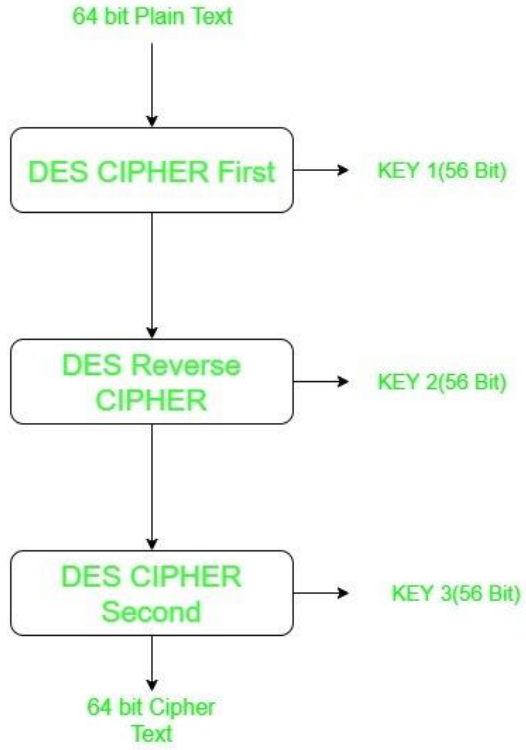
Double DES is a encryption technique which uses two instance of DES on same plain text. In both instances it uses different keys to encrypt the plain text. Both keys are required at the time of decryption. The 64 bit plain text goes into first DES instance which then converted into a 64 bit middle text using the first key and then it goes to second DES instance which gives 64 bit cipher text by using second key.



However double DES uses 112 bit key but gives security level of 2^56 not 2^112 and this is because of meet-in-the middle attack which can be used to break through double DES.

Triple DES:

Triple DES is a encryption technique which uses three instance of DES on same plain text. It uses there different types of key choosing technique in first all used keys are different and in second two keys are same and one is different and in third all keys are same.



Triple DES is also vulnerable to meet-in-the middle attack because of which it give total security level of 2^112 instead of using 168 bit of key. The block collision attack can also be done because of short block size and using same key to encrypt large size of text. It is also vulnerable to sweet32 attack.

Strength of Data encryption standard (DES)

Data encryption standard (DES) is a symmetric key block cipher algorithm. The algorithm is based on Feistel network. The algorithm uses a 56-bit key to encrypt data in 64-bit blocks. There are mainly two categories of concerns about the strength of Data encryption standard. They are:

1. Concerns about the particular algorithm used.
2. Concerns about the usage of key of size 56-bit.

The first concern regarding the algorithm used addresses the possibility of cryptanalysis by making use of the DES algorithm characteristics. A more severe concern is about the length of secret key used. There can be (approximately 7.2 × keys) possible keys with a key length of 56 bits. Thus, a brute force attack appears to be impractical. Assuming that on an average one has to search half the key space, to break the cipher text, a system performing one DES encryption per microsecond might require more than thousand years. But, the assumption of one DES encryption per microsecond is too conservative. In July

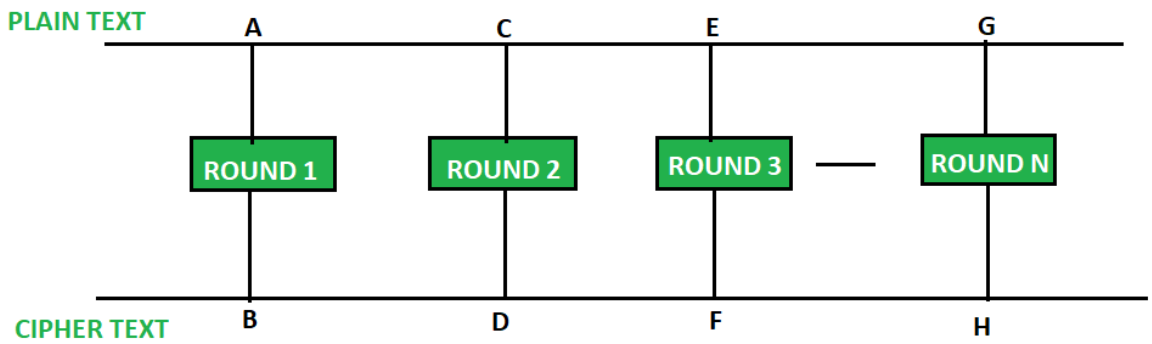
1998, DES was finally proved to be insecure when the Electronic Frontier Foundation (EFF) had broken a DES encryption. The encryption was broken with the help of a special-purpose “DES cracker” machine. It was reported that the attack took less than 3 days. Simply running through all possible keys won’t result in cracking the DES encryption. Unless known plain text is given, the attacker must be able to differentiate the plain text from other data. Some degree of knowledge about the target plain text and some techniques for automatically distinguishing plain text from garble are required to supplement the brute-force approach. If brute force attack is the only means to crack the DES encryption algorithm, then using longer keys will obviously help us to counter such attacks. An algorithm is guaranteed unbreakable by brute force if a 128- bit key is used. The differential cryptanalysis, linear cryptanalysis, are examples for statistical attacks on DES algorithm. Few of the important alternatives for DES are AES (Advanced Encryption Standard) and triple DES.

AES

AES stands for **Advanced Encryption Standard** and is a majorly used symmetric encryption algorithm. It is mainly used for encryption and protection of electronic data. It was used as the replacement of DES (Data encryption standard) as it is much faster and better than DES. AES consists of three block ciphers and these ciphers are used to provide encryption of data.

History

AES was developed by NIST (National Institute of Standards and Technology) in 1997. It was developed for replacing DES which was slow and was vulnerable to various attacks. So, therefore, a new encryption algorithm was made to overcome the shortcomings of DES. AES was then published on 26th November 2001.



Characteristics

- AES has keys of three lengths which are of 128, 192, 256 bits.
- It is flexible and has implementation for software and hardware.

- It provides high security and can prevent many attacks.
- It doesn’t have any copyright so it can be easily used globally.
- It consists of 10 rounds of processing for 128 bit keys.

Advantages

- It can be implemented on both hardware and software.
- It provides high security to the users.
- It provides one of the best open source solutions for encryption.
- It is a very robust algorithm.

Disadvantages

- It requires many rounds for encryption.
- It is hard to implement on software.
- It needs much processing at different stages.
- It is difficult to implement when performance has to be considered.

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a highly trusted **encryption algorithm** used to secure data by converting it into an unreadable format without the proper key. It is developed by the National Institute of Standards and Technology (NIST) in 2001. It is is widely used today as it is much stronger than DES and triple DES despite being harder to implement. **AES encryption** uses various **key lengths** (128, 192, or 256 bits) to provide strong protection against unauthorized access. This **data security** measure is efficient and widely implemented in securing **internet communication**, protecting **sensitive data**, and encrypting files. AES, a cornerstone of modern cryptography, is recognized globally for its ability to keep information safe from cyber threats.

- 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. AES relies on the substitution-permutation network principle, which is performed using a series of linked operations that involve replacing and shuffling the input data.

Working of The Cipher

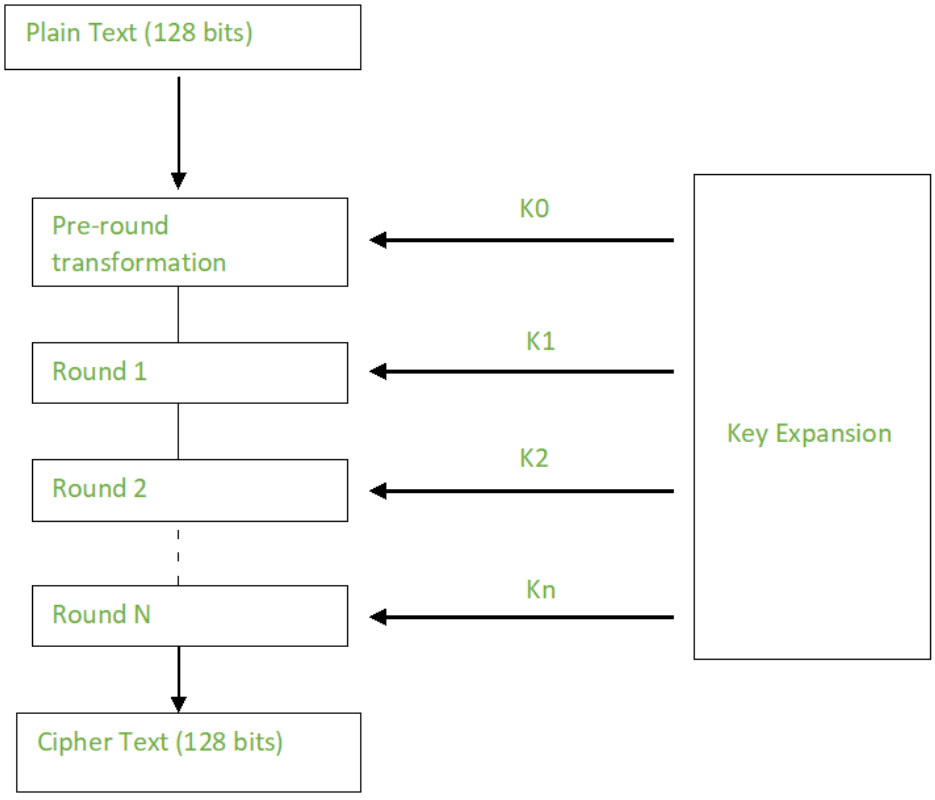
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

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

N (Number of Rounds)	Key Size (in bits)
10	128
12	192
14	256

Creation of Round Keys

A Key Schedule algorithm calculates 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.

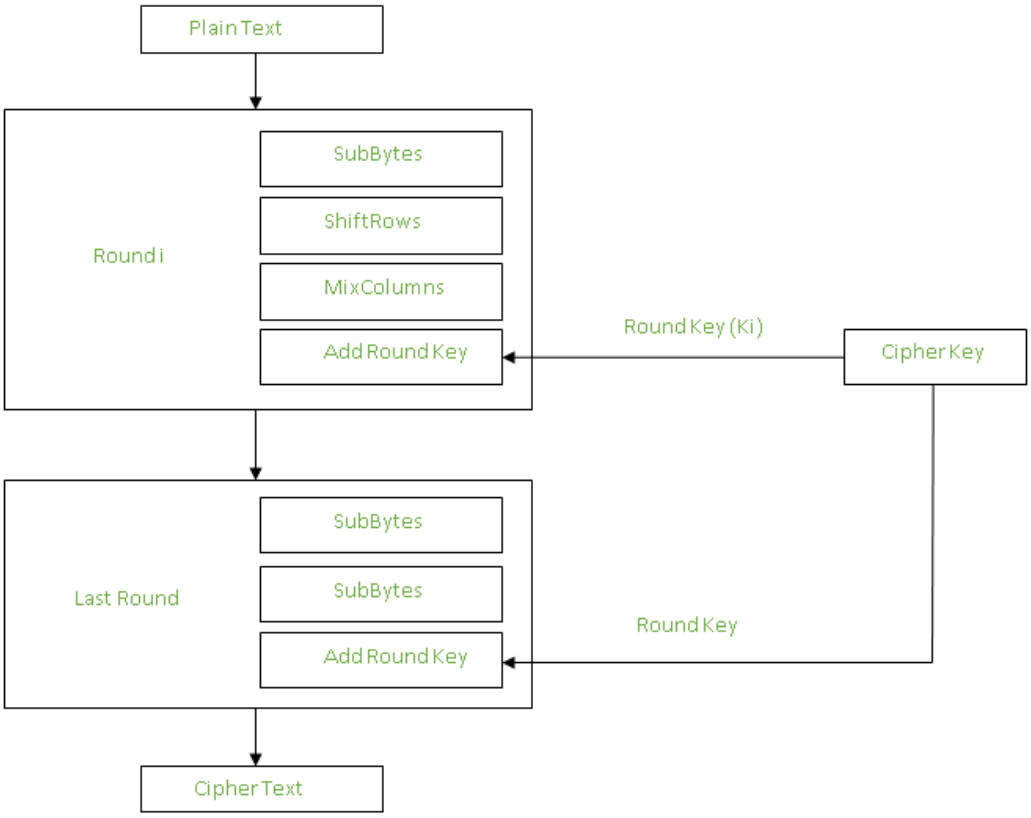


Creation of Round Keys (AES)

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]



Added Round Keys (AES)

Each round comprises of 4 steps :

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

Step1. Sub Bytes

This step implements the substitution.

In this step, each byte is substituted by another byte. It is 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.

Step2. Shift Rows

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]

Step 3: Mix Columns

This step is 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]

Step 4: Add Round Keys

- Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes are not considered as a grid but just as 128 bits of data.
- After all these rounds 128 bits of encrypted data are 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 of decryption are as follows :

- Add round key
- Inverse Mix Columns
- Shift Rows
- Inverse Sub Byte

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

Inverse Mix Columns

- This step is similar to the Mix Columns step in encryption but differs in the matrix used to carry out the operation.
- Mix Columns Operation each column is mixed independent of the other.
- Matrix multiplication is used. The output of this step is the matrix multiplication of the old values and a

constant matrix

[b0] = [14 11 13 9] [c0]
[b1]=[9 14 11 13] [c1]
[b2] =[13 9 14 11] [c2]
[b3]=[11 13 9 14] [c3]

Inverse Sub Bytes

- Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.
- Function Substitute performs a byte substitution on each byte of the input word. For this purpose, it uses an S-box.

Applications of AES

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 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.

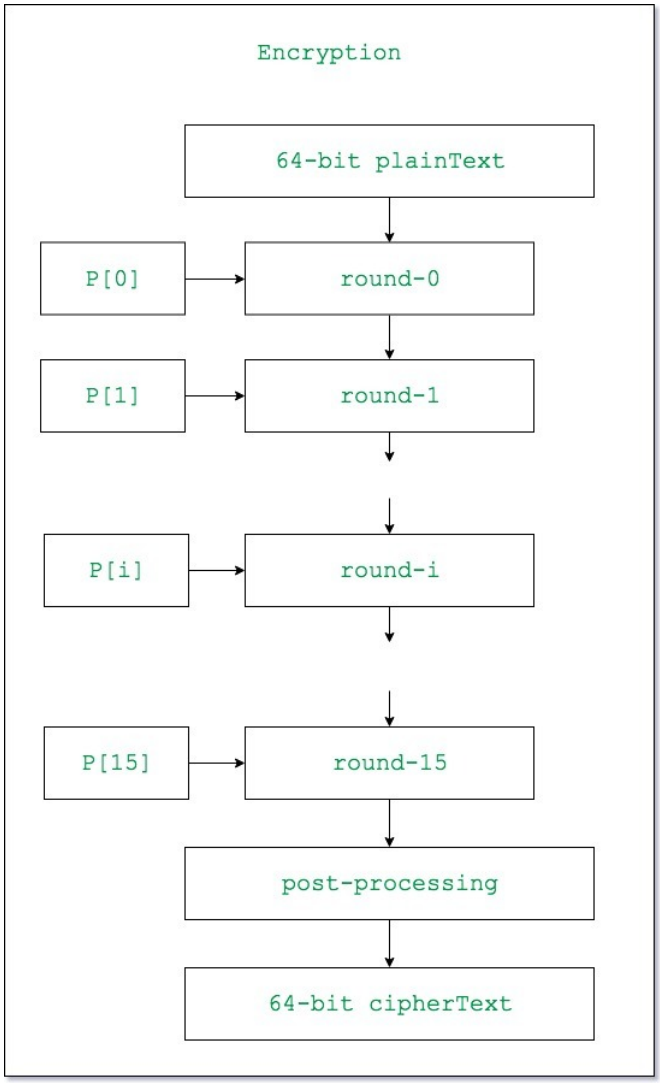
Blowfish Algorithm

Blowfish is an encryption technique designed by **Bruce Schneier** in 1993 as an alternative to DES Encryption Technique. It is significantly faster than DES and provides a good encryption rate with no effective cryptanalysis technique found to date. It is one of the first, secure block cyphers not subject to any patents and hence freely available for anyone to use. It is symmetric block cipher algorithm.

1. **blockSize**: 64-bits
2. **keySize**: 32-bits to 448-bits variable size
3. **number of subkeys**: 18 [P-array]
4. **number of rounds**: 16
5. **number of substitution boxes**: 4 [each having 512 entries of 32-bits each]

Blowfish Encryption Algorithm

The entire encryption process can be elaborated as:



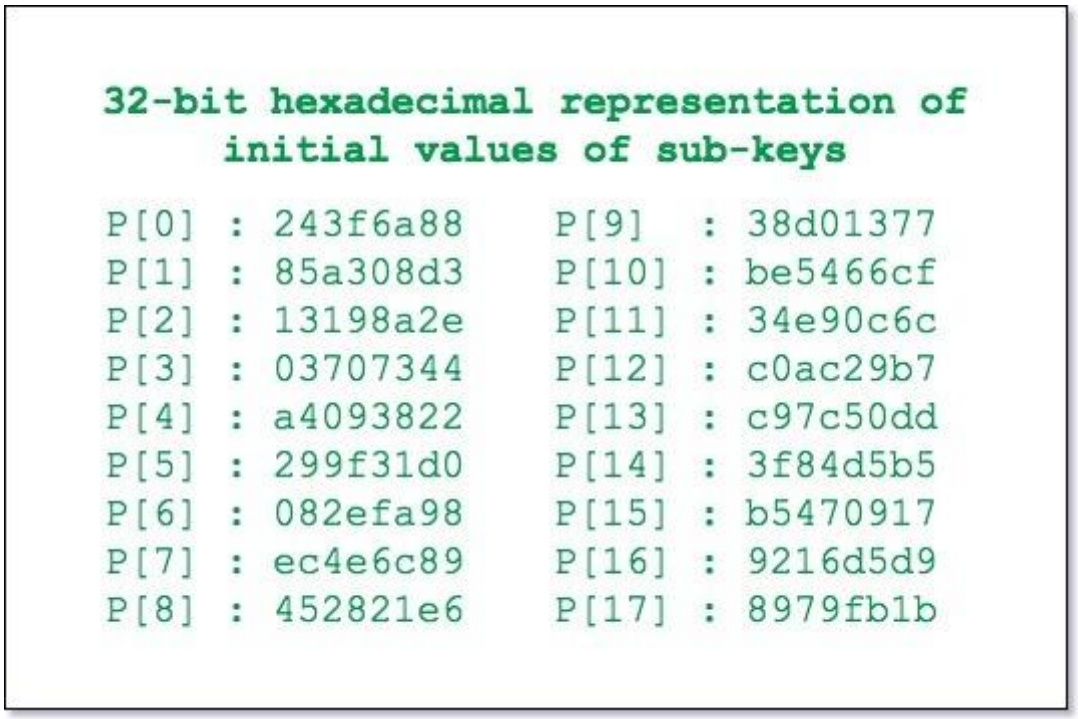
Lets see each step one by one:

Step1: Generation of subkeys:

- 18 subkeys{P[0]…P[17]} are needed in both encryption as well as decryption process and the same subkeys are used for both the processes.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

P[0] = "243f6a88"
P[1] = "85a308d3"
.
.

.
P[17] = "8979fb1b"



- Now each of the subkey is changed with respect to the input key as:

P[0] = P[0] xor 1st 32-bits of input key
P[1] = P[1] xor 2nd 32-bits of input key
.
.
.
P[i] = P[i] xor (i+1)th 32-bits of input key
(roll over to 1st 32-bits depending on the key length)
.
.
.
P[17] = P[17] xor 18th 32-bits of input key
(roll over to 1st 32-bits depending on key length)

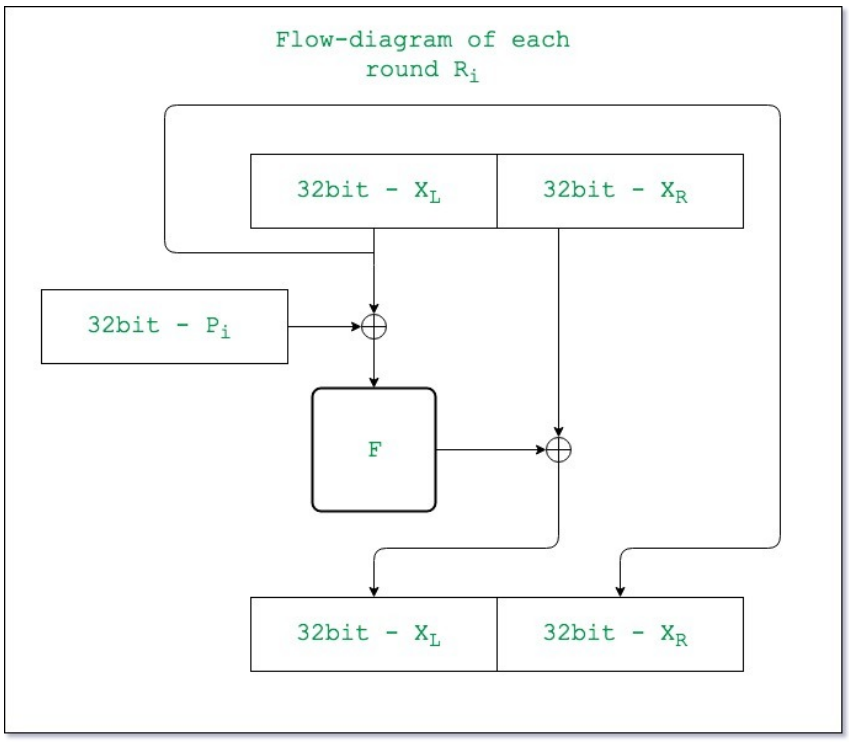
The resultant P-array holds 18 subkeys that is used during the entire encryption process

Step2: initialise Substitution Boxes:

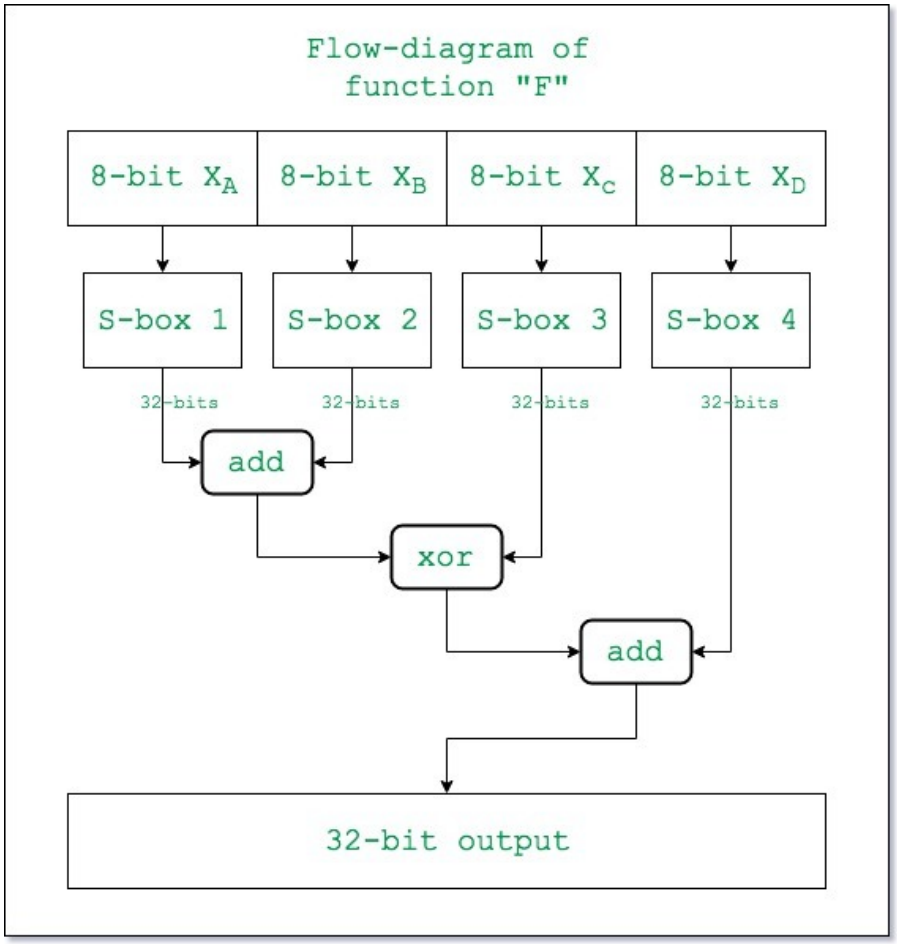
- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption as well as decryption process with each S-box having 256 entries{S[i][0]…S[i][255], 0&lei&le4} where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array.

Step3: Encryption:

- The encryption function consists of two parts:
  a. Rounds: The encryption consists of 16 rounds with each round (Ri) taking inputs the plainText(P.T.) from previous round and corresponding subkey(Pi). The description of each round is as follows:

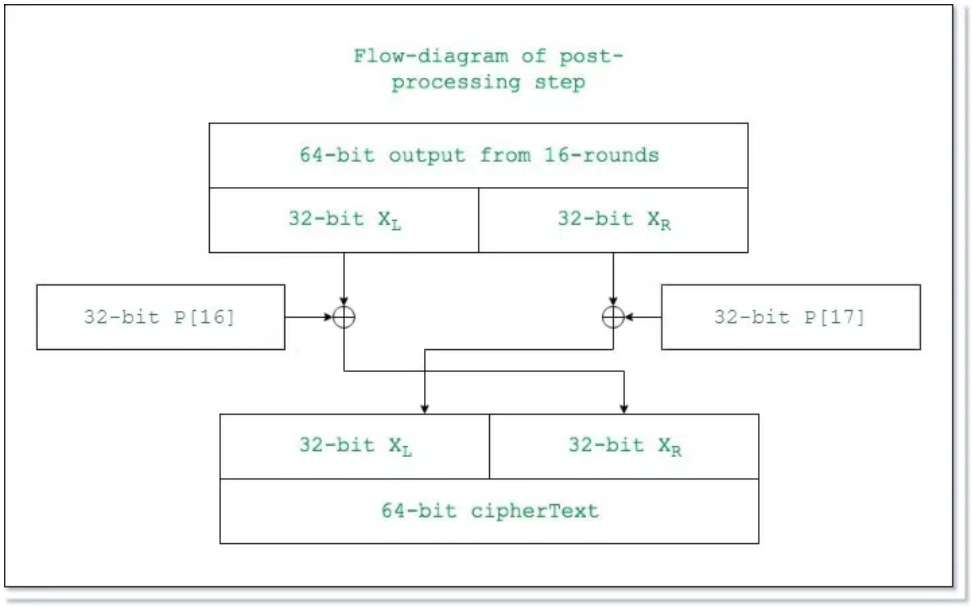


The description of the function ” F ” is as follows:



Here the function “add” is addition modulo 2^32.

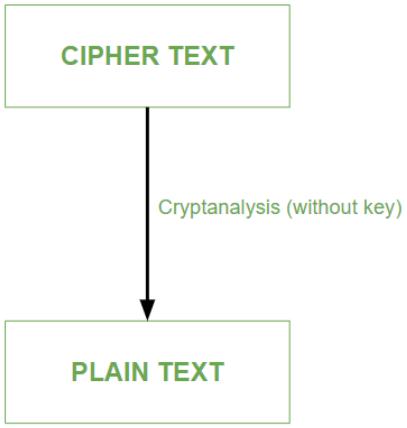
b. Post-processing: The output after the 16 rounds is processed as follows:



Differential and Linear Cryptanalysis

Cryptanalysis is the process of transforming or decoding communications from non-readable to readable format without having access to the real key. OR we may say it is the technique of retrieving the plain text of the communication without having access to the key. *Cryptoanalysis is the art, science, or practice of decrypting encrypted messages.* The secret key used for encryption and decoding is considered to be unknown to the cryptologists, mathematicians, and other scientists participating in the process. In contrast to a brute force attack, this form of analysis seeks vulnerabilities in a cryptosystem.

Cryptanalysis frequently comprises a direct evaluation of the cryptosystem in use, which is essentially an advanced concentrated mathematical attempt at decryption utilizing knowledge about the encryption scheme that is already available. They can employ intercepted encrypted messages (ciphertext), intercepted complete, partial, likely, or similar original messages (plaintext), or information (encrypted or original) that is known to be used adaptively in subsequent trials.



Process of cryptanalysis

Cryptanalysis is used to break cryptographic security systems and gain access to the contents of the encrypted messages, even if the cryptographic key is unknown.

Types of Cryptanalytic Attacks:

1. **Ciphertext only attack:**

    1. In this type of cryptanalytic attack, the attacker has the knowledge of only the ciphertext.
    2. The attacker has to detect the plain text using the ciphertext only.
    3. This type of attack is not very easy to be implemented.

2. **Known plain text only attack:**

1. In this type of cryptanalytic attack, the attacker has the knowledge of some plain text as well as ciphertext.
2. The attacker tries to decrypt the messages using these two.
3. This type of attack is somewhat easy to implement.

Different Forms of Cryptanalysis:

Cryptanalysis basically has two forms:

1. Linear Cryptanalysis:

Linear cryptanalysis is a general type of cryptanalysis based on discovering affine approximations to a cipher's action in cryptography. Block and stream ciphers have both been subjected to attacks. Linear cryptanalysis is one of the two most common attacks against block ciphers, with differential cryptanalysis being the other.

2. Differential Cryptanalysis:

Differential cryptanalysis is a sort of cryptanalysis that may be used to decrypt both block and stream ciphers, as well as cryptographic hash functions. In the widest sense, it is the study of how alterations in information intake might impact the following difference at the output. In the context of a block cipher, it refers to a collection of strategies for tracking differences across a network of transformations, finding where the cipher displays non-random behavior, and using such attributes to recover the secret key (cryptography key).

Difference between Linear Cryptanalysis and Differential Cryptanalysis

S. No.	**Linear Cryptanalysis**	**Differential Cryptanalysis**
1.	Linear cryptanalysis was basically invented by Matsui and Yamagishi in the year 1992.	Differential cryptanalysis was first defined in the year 1990 by Eli Biham and Adi Shamir.
2.	Linear cryptanalysis always works on a single bit (one bit at a time).	Differential cryptanalysis can work on multiple bits at a time.
3.	In the case of Linear cryptanalysis, ciphertext attack is a very big disadvantage.	In the case of differential cryptanalysis plain text attack is a very big disadvantage.

S. No.	Linear Cryptanalysis	Differential Cryptanalysis
4.	The use of linear cryptanalysis is to figure out what is the linear relationship present between some plaintext bits, ciphertext bits, and unknown key bits very easily.	The use of differential cryptanalysis is to get clues about some critical bits, reducing the need for an extensive search.
5.	Subsets of input attributes refer to the internal structures of a single input.	The underlying structure of each individual input is unimportant in this case since the input attributes are differential.
6.	The cryptanalyst decrypts each ciphertext using all available subkeys and analyses the resultant intermediate ciphertext to determine the random outcome for one encryption cycle.	After several encryption rounds, Cryptanalyst analyses the changes in the intermediate ciphertext obtained. The practice of combining assaults is known as differential linear cryptanalysis.
7.	Any random plaintext is selected in Linear Cryptanalysis.	Plaintext is Carefully chosen in Differential Cryptanalysis.
8.	Plaintext is used one by one in linear Cryptanalysis.	Plaintext is used in pairs in Differential Cryptanalysis.
9.	Complexity of attack is low in linear Cryptanalysis.	Complexity of attack is High in Differential Cryptanalysis
10.	Mathematical relation between plaintexts used has Linear	Mathematical relation between plaintexts used has Specific differences (such as XOR).

S. No.	Linear Cryptanalysis	Differential Cryptanalysis
	approximation (such as a series of XOR operations).	
11.	Goal of the attack is to identify the linear relation between some bits of the plaintext, some bits of the cipher text and some bits of the unknown key.	Goal of the attack is to Identify some bits of the unknown key.

RC5 Encryption Algorithm

RC5 is a symmetric key block encryption algorithm designed by Ron Rivest in 1994. It is notable for being simple, fast (on account of using only primitive computer operations like XOR, shift, etc.) and consumes less memory. **Example:**

Key : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Plain Text : 00000000 00000000
Cipher Text : EEDBA521 6D8F4B15

RC5 is a block cipher and addresses two word blocks at a time. Depending on input plain text block size, number of rounds and key size, various instances of RC5 can be defined and each instance is denoted as RC5-w/r/b where w=word size in bits, r=number of rounds and b=key size in bytes. Allowed values are:

Parameter	Possible Value
block/word size (bits)	16, 32, 64
Number of Rounds	0 – 255
Key Size (bytes)	0 – 255

Note – Since at a time, RC5 uses 2 word blocks, the plain text block size can be 32, 64 or 128 bits. Notation used in the algorithm:

Symbol	Operation
x <<< y	Cyclic left shift of x by y bits
+	Two's complement addition of words where addition is modulo 2w2*w*
^	Bit wise Exclusive-OR

Step-1: Initialization of constants P and Q. RC5 makes use of 2 magic constants P and Q whose value is defined by the word size w.

Word Size (bits)	P (Hexadecimal)	Q (Hexadecimal)
16	b7e1	9e37
32	b7e15163	9e3779b9
64	b7e151628aed2a6b	9e3779b97f4a7c15

For any other word size, P and Q can be determined as:

P = Odd((e-2)2w2*w*) Q = Odd((φ*ϕ*-2)2w2*w*)

Here, Odd(x) is the odd integer nearest to x, e is the base of natural logarithms and [Tex]\phi [/Tex]is the golden ratio. **Step-2:** Converting secret key K from bytes to words. Secret key K of size b bytes is used to initialize array L consisting of c words where c = b/u, u = w/8 and w = word size used for that particular instance of RC5. For example, if we choose w=32 bits and Key k is of size 96 bytes then, u=32/8=4, c=b/u=96/4=24. L is pre initialized to 0 value before adding secret key K to it.

```
for i=b-1 to 0
   L[i/u] = (L[u/i] <<< 8) + K[i]
```

Step-3: Initializing sub-key S. Sub-key S of size t=2(r+1) is initialized using magic constants P and Q.

```
S[0] = P
for i = 1 to 2(r+1)-1
   S[i] = S[i-1] + Q)
```

Step-4: Sub-key mixing. The RC5 encryption algorithm uses Sub key S. L is merely, a temporary array formed on the basis of user entered secret key. Mix in user's secret key with S and L.

```
i = j = 0
A = B = 0
do 3 * max(t, c) times:
   A = S[i] = (S[i] + A + B) <<< 3
   B = L[j] = (L[j] + A + B) <<< (A + B)
   i = (i + 1) % t
   j = (j + 1) % c
```

Step-5: Encryption. We divide the input plain text block into two registers A and B each of size w bits. After undergoing the encryption process the result of A and B together forms the cipher text block. RC5 Encryption Algorithm:

1. One time initialization of plain text blocks A and B by adding S[0] and S[1] to A and B respectively. These operations are mod[Tex]2^w [/Tex].
2. XOR A and B. A=A^B
3. Cyclic left shift new value of A by B bits.
4. Add S[2*i] to the output of previous step. This is the new value of A.
5. XOR B with new value of A and store in B.
6. Cyclic left shift new value of B by A bits.
7. Add S[2*i+1] to the output of previous step. This is the new value of B.
8. Repeat entire procedure (except one time initialization) r times.

```
A = A + S[0]
B = B + S[1]
for i = 1 to r do:
   A = ((A ^ B) <<< B) + S[2 * i]
   B = ((B ^ A) <<< A) + S[2 * i + 1]
return A, B
```

Alternatively, RC5 Decryption can be defined as:

```
for i = r down to 1 do:
   B = ((B - S[2 * i + 1]) >>> A) ^ A
   A = ((A - S[2 * i]) >>> B) ^ B
B = B - S[1]
A = A - S[0]
return A, B
```

Advantages:

High level of security: RC5 is designed to provide a high level of security against attacks, including brute-force attacks and differential cryptanalysis. It uses a variable-length key and can operate on block sizes of up to 256 bits, making it difficult for attackers to break the encryption.

Fast encryption and decryption: RC5 is known for its fast encryption and decryption speeds. It uses simple mathematical operations such as modular arithmetic and bit shifting, which can be efficiently implemented on modern CPUs and hardware.

Flexible key length: RC5 allows for a variable-length key, which can range from 0 to 2040 bits. This flexibility allows users to choose a key length that suits their security needs and resources.

Disadvantages:

Vulnerable to side-channel attacks: RC5 is vulnerable to side-channel attacks, such as timing attacks and power analysis attacks. These attacks exploit information leaked through the implementation of the algorithm, rather than attacking the algorithm itself.

Limited adoption: RC5 is not widely adopted in practice compared to other encryption algorithms, such as AES. This means that there may be fewer resources and tools available to support RC5 in various applications and systems.

Patent issues: RC5 was subject to a patent held by RSA Security, which limited its adoption and use in commercial applications. Although the patent has since expired, it may have contributed to the limited adoption of RC5 compared to other encryption algorithms

Placement of encryption function

The placement of encryption functions can be on the client-side, server-side, or both, depending on the specific application and security requirements, aiming to protect data at rest or in transit.

1. Client-Side Encryption:

- **Purpose:**

Encrypts data on the user's device (client) before it's transmitted to the server or stored on the server.

- **Benefits:**
    - Enhanced security: Even if the server is compromised, the data remains encrypted and unreadable without the decryption key.

    - Data privacy: Protects sensitive information from unauthorized access, even if the server is accessed by someone other than the intended recipient.
- **Example:**

Encrypting data in a web browser before sending it to a server, or encrypting files on a user's computer before uploading them to cloud storage.

2. Server-Side Encryption:

- **Purpose:** Encrypts data on the server-side, either before or after it is stored.
- **Benefits:**
    - Simplified security management: The server administrator manages the encryption keys and processes, rather than the user.
    - Data protection at rest: Protects data stored on the server from unauthorized access, even if the server is physically compromised.
- **Example:** Encrypting databases or files stored on a server, or encrypting data in transit between servers.

3. End-to-End Encryption:

- **Purpose:** Encrypts data on the client-side and decrypts it only on the recipient's client-side, ensuring that the data is never decrypted on the server.
- **Benefits:**
    - Maximum privacy: The server cannot access or read the encrypted data.
    - Enhanced security: Even if the server is compromised, the data remains unreadable.
- **Example:** Secure messaging apps that use end-to-end encryption.

4. Considerations for Placement:

- **Security requirements:**

The level of security required will determine the appropriate placement of encryption functions.

- **Performance:**

Encryption can add overhead, so it's important to consider the performance impact of encrypting data.

- **Complexity:**

Implementing encryption can be complex, so it's important to choose a solution that is appropriate for the application and the skills of the developers.

- **Key management:**

Securely managing encryption keys is crucial for maintaining the integrity of the encryption process.

- **Compliance:**

Some regulations may require the use of specific encryption methods or placement of encryption functions

Key Distribution

Two parties may exchange cryptographic keys through a procedure known as key exchange, also known as key distribution, in order to use a cryptographic algorithm.

For messages to be exchanged via encryption, both the sender and the recipient must be able to encrypt and decrypt them. Depending on the kind of encryption they want to use different technologies are required. Both will need a copy of the same codebook if they use a code. They will need the right keys if they utilise a cipher. Both will require a copy of the same key if the cipher uses symmetric keys. Both parties will need the public key of the other if the key cipher is asymmetric and has the public/private key characteristic.

Channel of Distribution

Key Distribution is possible in-band or out-of-band.

"Channel of distribution" means the way information or keys are swapped between two parties.

"Key exchange" is when two parties share secret codes or 'keys' to communicate securely.

"In-band" key exchange means the keys are swapped through the same communication channel being used for the actual data.

"Out-of-band" key exchange means the keys are shared through a separate, different communication channel from the one used for the actual data.

The Key Exchange Problem

The goal of the key distribution problem is to safely change the keys so that communications are only read by those who are intended to see them.

Messages were encrypted using just one encryption code in the past. But they needed to find a way to pass this key between them so that no one else could figure it out in order to communicate safely.

We now have a very advanced technology known as public-key cryptography. It makes use of two keys: a private one that is kept hidden and a public one that is freely shared. Messages can be encrypted with one key and decrypted with another.

A well-known technique that allows parties to freely distribute keys without compromising the security of their messages is the Diffie-Hellman key exchange. It is far more secure than exchanging secret codes in the past.

Now let us discuss all the problem for key exchange −

Symmetric Key Distribution

The conventional approach, known as symmetric key distribution, uses a single secret key that is shared by both sides. Before communicating, they exchange this key via a secure channel.

Public Key Distribution

With this approach, a public key and a private key are given to each users. While the private key is kept confidential, the public key is freely shared. The recipient's public key is used to encrypt messages, while their private key is used to decrypt them.

Diffie−Hellman key exchange

Based on ideas created by Ralph Merkle, Martin Hellman's PhD student, Whitfield Diffie and Hellman published the Diffie−Hellman key exchange (D−H) cryptography protocol in 1976. Users can safely exchange secret keys because of the protocol, even if someone else is keeping an eye on the communication channel. However, authentication-that is, the issue of knowing for sure the true identity of the person or "entity" on the other end of the communication channel-is not addressed by the D−H key exchange protocol on its own. Authentication is important when an adversary can track and modify messages within the communication channel (also known as man-in-the-middle or MITM attacks).

Public key infrastructure

The issue of identity authentication has been addressed with the proposal of public key infrastructures (PKIs). In their most common application, each user requests for a digital certificate from a "certificate authority" (CA) that is universally trusted. This certificate acts as an immutable means of identity verification for other users. Even in the event that the CA is hacked, the infrastructure is secure. However, a lot of PKIs offer a mechanism to revoke certificates in case such happens, making other users suspicious of them. Certificate revocation lists, against which any certificate can be compared, are often where revoked certificates are stored.

Legislation or regulations supporting PKIs have been passed in a number of nations and other jurisdictions, providing these digital certificates with some degree of legal standing.

Quantum key exchange

The use of unique features of small particles known as quantum physics in quantum key distribution makes secrets highly encrypted. These particles undergo minor modifications as we observe or quantify them.

Using this technology, an attempt to track on a discussion between two people will cause the particles to become impacted, notifying us to the possibility of an issue.

This technique only functions if Alice and Bob, the individuals, already set up a unique, secure means of communication.