Title: Cryptography Analysis and Implementation

Chaitanya Nagre – 20BCN7032

VIT – AP

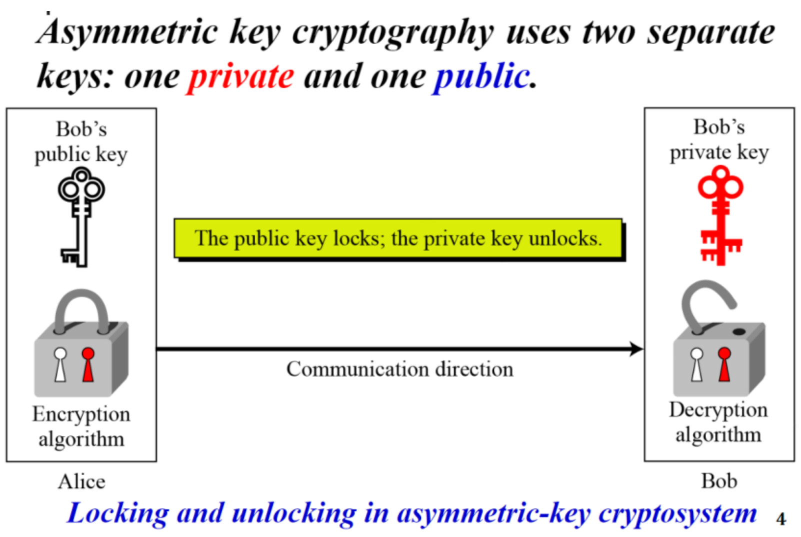
Asymmetric Cryptography: -

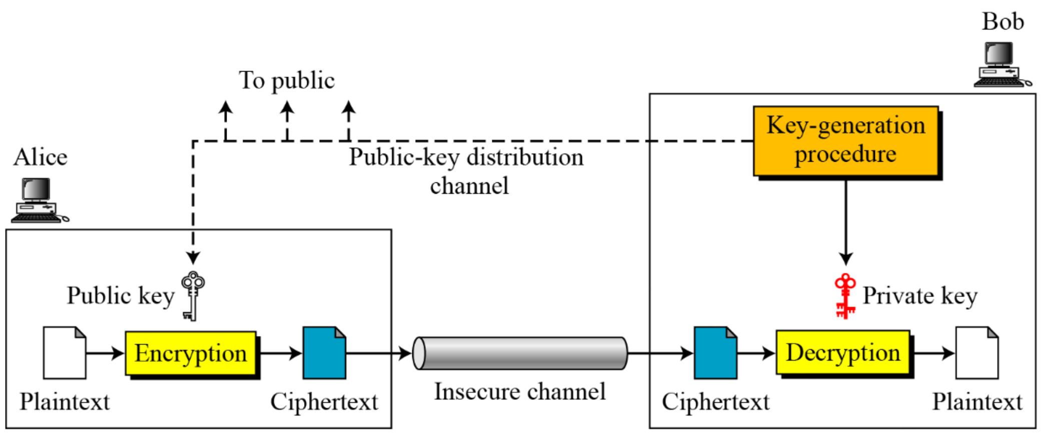
Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys -- one public key and one private key -- to encrypt and decrypt a message and protect it from unauthorized access or use.

The public key is shared with all the parties that must communicate with the sender.
The private key is kept secret from each party.
Though there is a mathematical connection between these private key and public key pairs, the public key cannot generate the private key.
Public key cryptography is commonly used in digital signatures for message authentication.
Senders use their private keys to digitally sign their messages to prove their authenticity.
Thus, the receiver knows exactly that the sender is a trusted third party.

Ex: -

- Elliptical Curve Cryptography (ECC)
- Rivest Shamir Adleman (RSA)
- The Diffie-Hellman exchange method
- TLS/SSL protocol

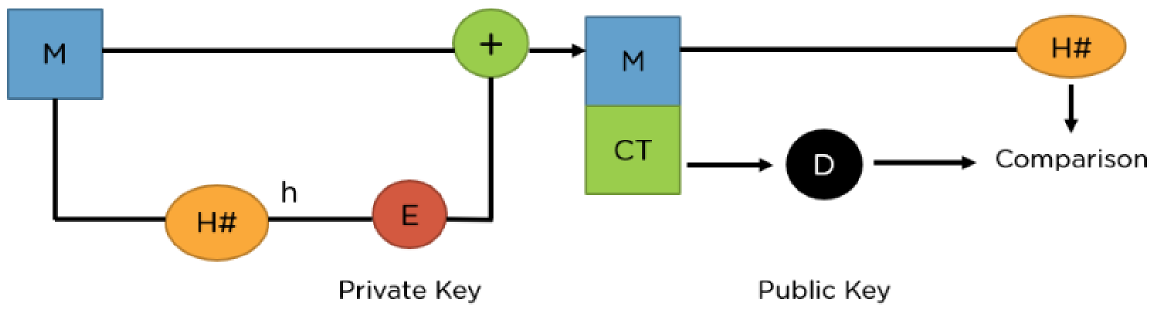

Asymmetric key cryptography uses two separate keys: one private and one public.
Bob's public key
Encryption algorithm
Alice
The public key locks; the private key unlocks.
Communication direction
Bob's private key
Decryption algorithm
Bob
Locking and unlocking in asymmetric-key cryptosystem
4



General idea of asymmetric-key cryptosystem

RSA Algorithm: -

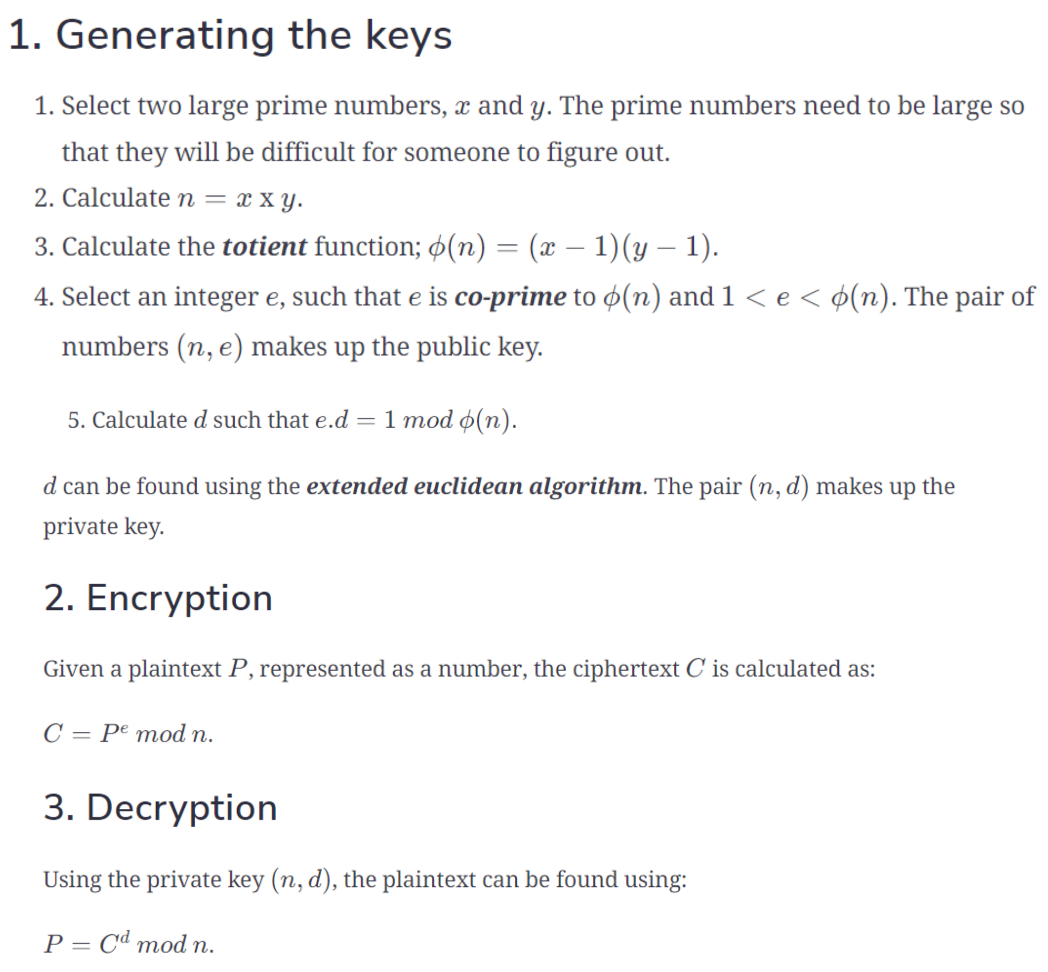
The RSA algorithm is a public-key signature algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. Their paper was first published in 1977, and the algorithm uses logarithmic functions to keep the working complex enough to withstand brute force and streamlined enough to be fast post-deployment. The image below shows it verifies the digital signatures using RSA methodology.



The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a *public* key and a *private* key (i.e., two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

Procedure: -

The RSA algorithm ensures that the keys, in the above illustration, are as secure as possible. The following steps highlight how it works:



1. Generating the keys

1. Select two large prime numbers, x and y. The prime numbers need to be large so that they will be difficult for someone to figure out.
2. Calculate $n = x$ x y.
3. Calculate the ***totient*** function; $\phi(n) = (x-1)(y-1)$.
4. Select an integer e, such that e is ***co-prime*** to $\phi(n)$ and $1 < e < \phi(n)$. The pair of numbers (n, e) makes up the public key.

5. Calculate d such that $e.d = 1 \ mod \ \phi(n)$.

d can be found using the ***extended euclidean algorithm***. The pair (n, d) makes up the private key.

2. Encryption

Given a plaintext P, represented as a number, the ciphertext C is calculated as:

$C = P^e \ mod \ n$.

3. Decryption

Using the private key (n, d), the plaintext can be found using:

$P = C^d \ mod \ n$.

Advantages of RSA: -

No Key Sharing: RSA encryption depends on using the receiver's public key, so you do not have to share any secret key to receive messages from others.

Proof of Authenticity: Since the key pairs are related to each other, a receiver cannot intercept the message since they will not have the correct private key to decrypt the information.

Faster Encryption: The encryption process is faster than that of the DSA algorithm.

Data Cannot Be Modified: Data will be tamper-proof in transit since meddling with the data will alter the usage of the keys. And the private key will not be able to decrypt the information, hence alerting the receiver of manipulation.

RSA encryption provides strong security based on the difficulty of factoring large composite numbers, making it difficult for unauthorized parties to decrypt encrypted data.

RSA encryption is widely adopted and standardized, making it compatible with various systems and enabling its use in digital signatures, secure data transmission, and other cryptographic protocols.

Disadvantages of RSA: -

Because RSA only employs asymmetric encryption and complete encryption requires both symmetric and asymmetric encryption, it might occasionally fail.

Sometimes, it is necessary for a third party to confirm the dependability of public keys.

RSA cannot be used for public data encryption, such as electoral voting.

Decryption requires intensive processing on the receiver's end.

It has slow data transfer rate due to large numbers involved.

Real World Scenario: -

Secure online transactions: RSA encryption is used in e-commerce platforms, online banking systems, and payment gateways to secure sensitive information, such as credit card details, during online transactions.

Secure messaging and chat applications: Many messaging and chat applications, including popular ones like WhatsApp and Signal, employ RSA encryption to protect the confidentiality of messages and ensure secure communication between users.

Virtual private networks (VPNs): RSA encryption is utilized in VPNs to establish secure connections between remote users and private networks, safeguarding data transmission and ensuring privacy.

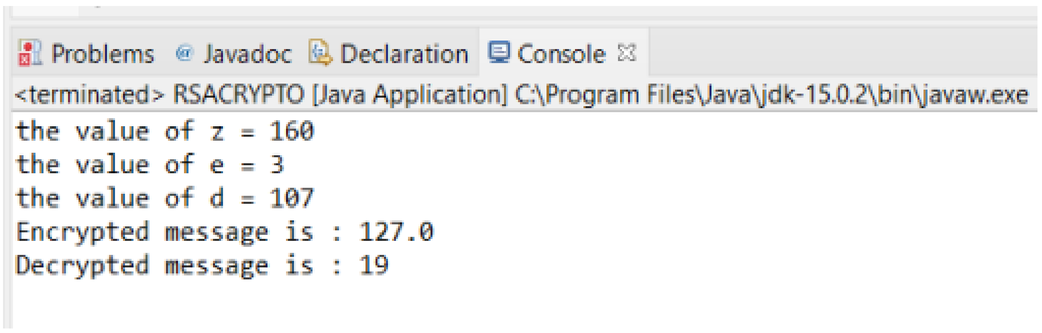
Implementation: -

Code: -

```
import java.math.*;

import java.util.*;

public class RSACRYPTO {

        public static void main(String args[]){

                int p, q, n, z, d = 0, e, i;
                int msg = 19 ;

                double c;

                BigInteger msgback;

                p = 11;

                q = 17;

                n = p * q;

                z = (p - 1) * (q - 1);

                System.out.println("the value of z = " + z);

                for (e = 2; e < z; e++) {

                        if (gcd (e, z) == 1) {

                                break;

                        }

                 }

                System.out.println("the value of e = " + e);

                for (i = 0; i <= 9; i++) {

                        int x = 1 + (i * z);

                        if (x % e == 0) {

                                d = x / e;

                                break;

                        }

                }

                System.out.println("the value of d = " + d);
```

```
                c = (Math.pow(msg, e)) % n;

                System.out.println("Encrypted message is: " + c);

                BigInteger N = BigInteger.valueOf(n);

                BigInteger C = BigDecimal.valueOf(c).toBigInteger();

                msgback = (C.pow(d)).mod(N);

                System.out.println("Decrypted message is: "+ msgback);

                }

        static int gcd (int e, int z) {

                if (e == 0)

                return z;

                else

                return gcd (z % e, e);

         }

}
```

Result: -



```
<terminated> RSACRYPTO [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe
the value of z = 160
the value of e = 3
the value of d = 107
Encrypted message is : 127.0
Decrypted message is : 19
```

Security Analysis of RSA: -

Potential Threats: -

1. Plain text attacks: -

It is classified into 3 subcategories: -

- Short message attack:
  In this we assume that attacker knows some blocks of plain text and tries to decode cipher text with the help of that. So, to prevent this pad the plain text before encrypting.

- Cycling attack:
  In this attack, the attacker thinks that the cipher text has been generated by using some permutation. He uses all possible permutations of plain text to decipher the cipher text by ‘cycling’ the permutations.

- Unconcealed Message attack:
  Sometimes it happens that plain text is same as cipher text after encryption. So, it must be checked or it will be of no use as the attacker will see right through it.

2. Chosen cipher attack: -

In this attacker can find out plain text based on cipher text using the Extended Euclidean Algorithm.

3. Factorization attack: -

If the attacker can know P and Q using N, then he can find out value of private key. This fails when N contains at least 300 longer digits in decimal terms, attacker will not able to find it. Hence this is infeasible for larger numbers.

4. Attacks on Encryption key: -

People well versed with the mathematics of RSA sometimes feel that it is quite easy because it can need a huge number for the public key or encryption key E. It also creates RSA more secure. Therefore, if it can decide to try and create the working of RSA faster by utilizing a small value for E, it can lead to potential attacks known as attacks on the encryption key and therefore it is suggested that it can use E as 216 + 1 = 65537 or a value nearer to this number.

5. Attacks on Decryption key: -

- Revealed decryption exponent attack:
  If attacker somehow guess decryption key D, not only the cipher text generated by encryption the plain text with corresponding encryption key is in danger, but even future messages are also in danger. So, it is advised to take fresh values of two prime numbers (i.e.: P and Q), N and E.

- Low decryption exponent attack:
  If we take smaller value of D in RSA this may occur so to avoid this take value of D = 2^16+1(at least).

Best Practice’s: -

Use a strong prime number generator to ensure that the prime numbers are unpredictable and cannot be easily guessed by an attacker.

Avoid using weak prime numbers, such as small primes or primes too close to each other.

Use a minimum length of 2048 bits for the RSA key.

Take necessary actions to protect against fault-based attacks, such as using tamper-resistant hardware.

Manage and secure the RSA keys properly using techniques like regular key rotation and different keys for different applications.

Keep the RSA algorithm up to date by regularly monitoring for vulnerabilities and updates.

Limitation’s: -

1) Using small primes.

This one is obvious, if the primes used are small enough then a computer will make easy work of factorising N.

2) Using primes that are very close.

This is quite a serious weakness because it makes a big flaw, even if you do use big enough primes. If p, q are relatively close (I know this is ambiguous but should be well understood) then searching for prime factors in the vicinity of $\sqrt{N}$ will reveal either of the factors in quite a quick time (depending on how close the factors are).

Alternatively, the methods of Fermat factorisation or more general number sieves can be used here to give a quick result in some cases.

3) Message is an observable e th power.

Sometimes (especially with very small values of e) it is likely that the ciphertext is the eth power of an integer. If this is the case then the plaintext can easily be recovered by taking e th roots of the corresponding integer equation.

Conclusion: -

RSA is a widely used cryptographic algorithm that was first introduced in 1977. It uses public and private key pairs to encrypt and decrypt data. Though RSA can be used in several applications, its computational complexity makes it unsuitable for encrypting large messages or files. Currently, RSA creates digital signatures and certificates for secure authentication, communication, web access and email messages and key exchanges.

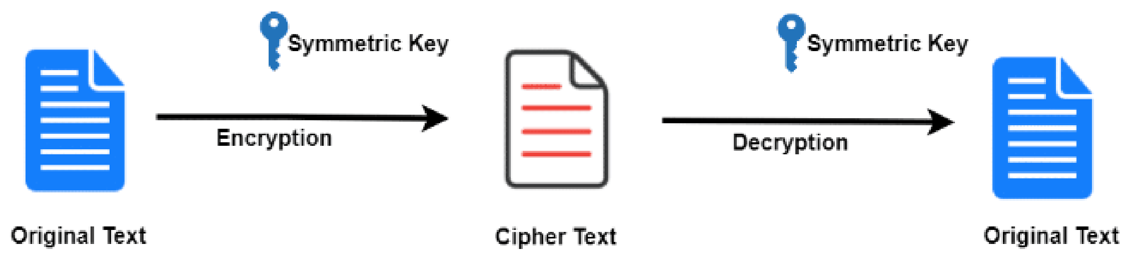
There are several vulnerabilities in RSA, such as side-channel attacks, inappropriate key lengths, weaknesses in prime numbers, fault-based attacks and risks introduced by stolen or lost keys. Thus, it is important to consider the recommendations described in the article when using RSA for your cryptographic applications.

Symmetric Cryptography: -

Symmetric-key cryptography involves encrypting and decrypting using the same cryptographic keys. Here, the sender and all receivers share a common secret key. The plaintext messages are transformed into cipher text using a particular encryption key. The receiver can use the same encryption key to decrypt the message using the shared secret key.

Ex: -

- Advanced Encryption Standard (AES)
- Data Encryption Standard (DES)
- Triple Data Encryption Standard (Triple DES)
- International Data Encryption Algorithm (IDEA)
- TLS/SSL protocol



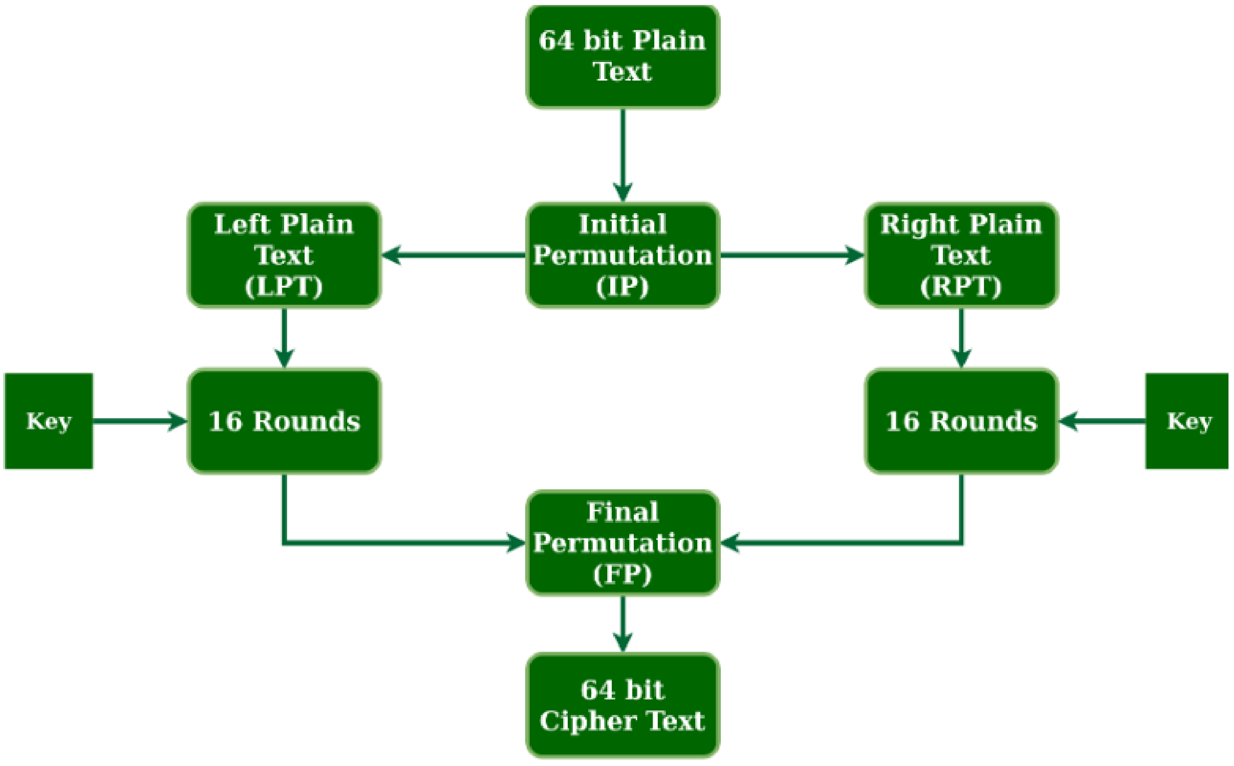
DES: -

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

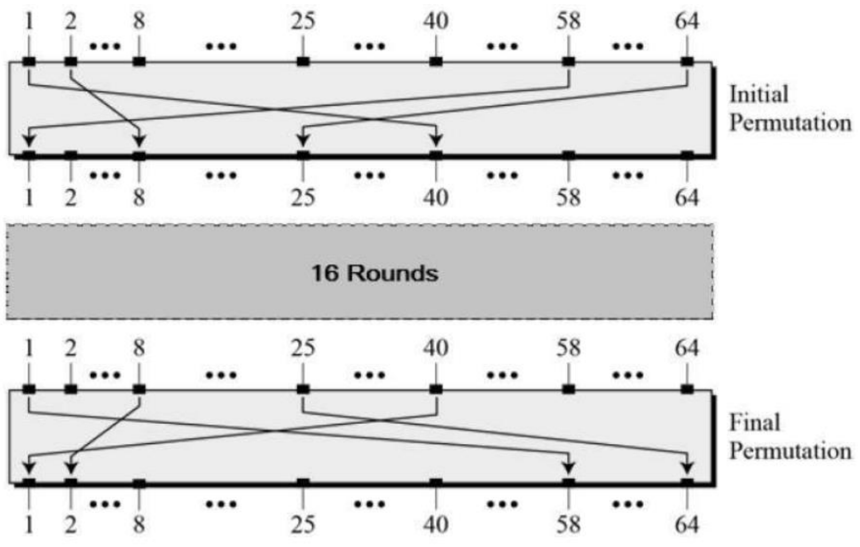


Broad Level Steps in DES

Procedure: -

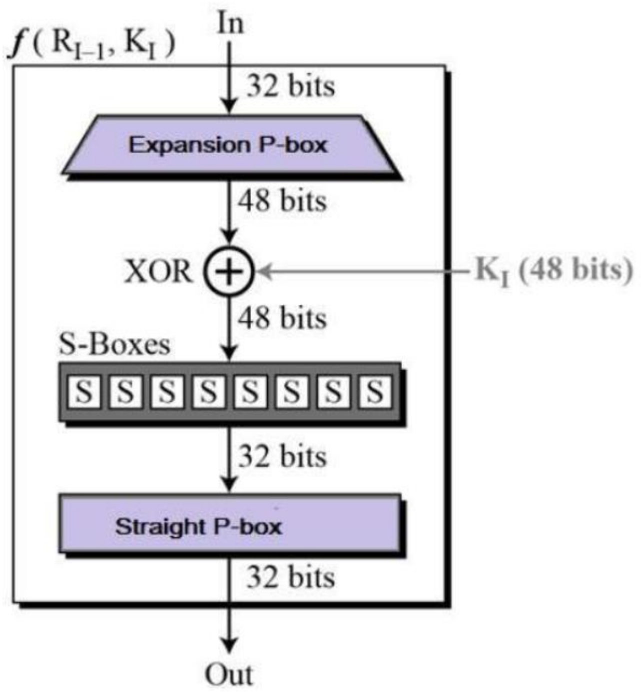
Initial and Final Permutation: -

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −

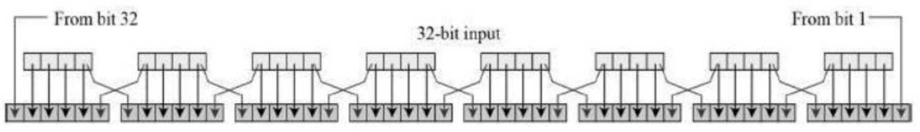


Round Function: -

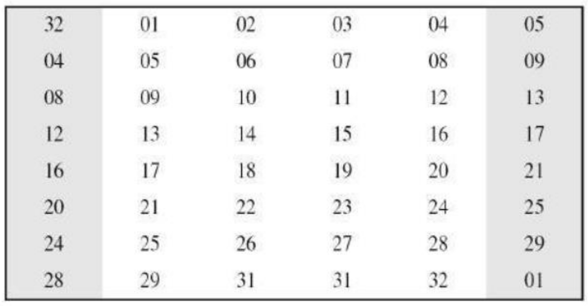
The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration –



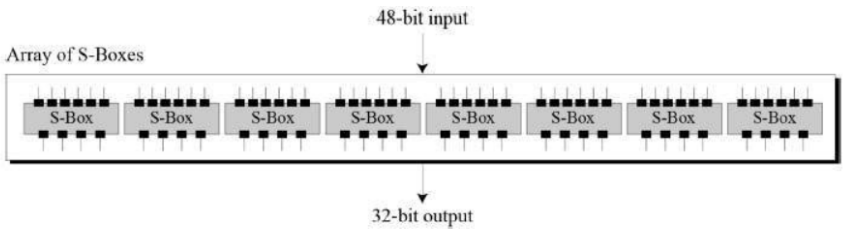
- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown –



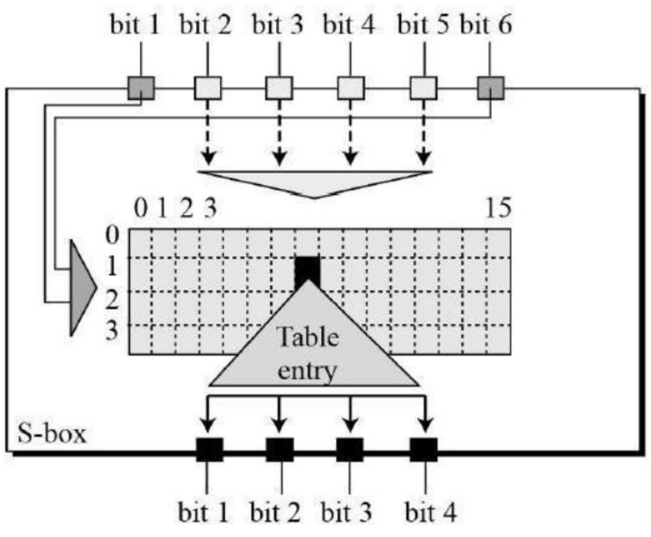
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −

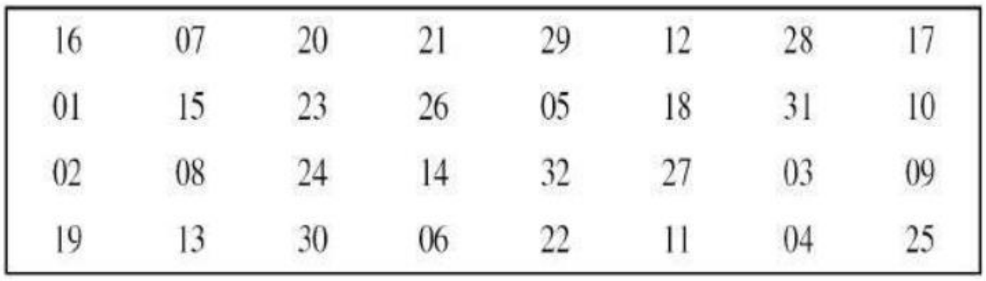


- The S-box rule is illustrated below −



- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32-bit section.

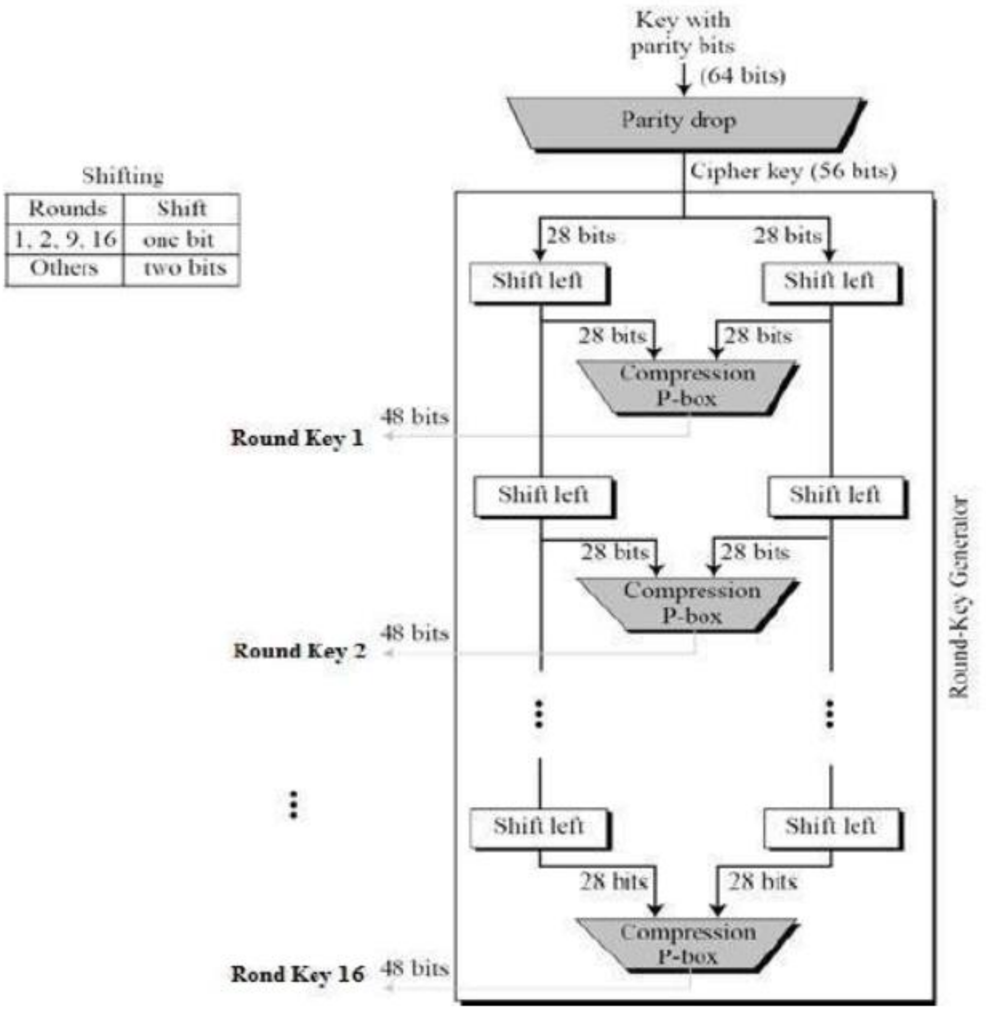
- **Straight Permutation** − The 32-bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:



16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation: -

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −



Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

The logic for Parity drops, shifting, and Compression P-box is given in the DES description.

Advantages: -

- DES has been around a long time (since 1977), even no actual weaknesses have been discovered and the most effective attack is still brute force.
- DES is an official United States Government standard. The Government is needed to re-certify, DES every five years and ask it be restored if essential.
- DES is also an ANSI and ISO standard. Because DES was designed to run on 1977 hardware, it is rapid in hardware and associatively quick in software.
- It supports functionality to save a file in an encrypted format which can only be accessed by supporting the correct password.
- It can change the system to create the directories password protected.
- It can review a short history of DES and represent the basic structures.
- It can define the building block component of DES.
- It can define the round keys generation process and to interpret data encryption standard.
- It can provide that private information is not accessed by other users.
- Some users can use the similar system and still can work individually.

Disadvantages: -

- The 56-bit key size is the largest defect of DES and the chips to implement one million of DES encrypt or decrypt operations a second are applicable (in 1993).
- Hardware implementations of DES are very quick.
- DES was not designed for application and therefore it runs relatively slowly.
- In a new technology, it is improving a several possibilities to divide the encrypted code, therefore AES is preferred than DES.

Real World Scenario: -

DES algorithm is to create triple DES legacy systems with three keys.

Hash Functions: -

A Hash Function is a function that converts a given numeric or alphanumeric key to a small practical integer value. The mapped integer value is used as an index in the hash table. In simple terms, a hash function maps a significant number or string to a small integer that can be used as the index in the hash table.

The pair is of the form (key, value), where for a given key, one can find a value using a "function" that maps keys to values. The key for a given object can be calculated using a function called a hash function. For example, given an array A, if i is the key, then we can find the value by simply looking up A[i].

Types of Hash functions: -

1. Division Method.
2. Mid Square Method.
3. Folding Method.
4. Multiplication Method.

Common Hashing Algorithms: -

- SHA-1
- SHA-2
- SHA-3
- MD5
- Whirlpool
- Blake 2
- Blake 3
- SHA-256

SHA – 256: -

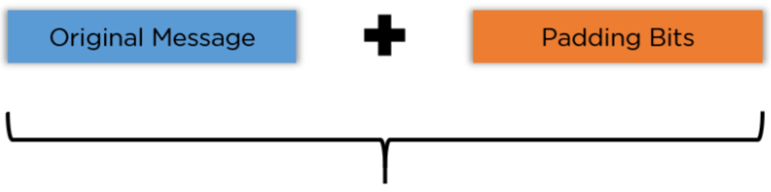
SHA-256 is part of the SHA-2 family of hash functions. It generates a 256-bit hash value, providing a high level of security. It is widely adopted and used in various applications, including blockchain technology, digital signatures, and password storage.

The significance of the 256 in the name stands for the final hash digest value, i.e., irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.

Procedure: -

1. **Padding Bits: -**

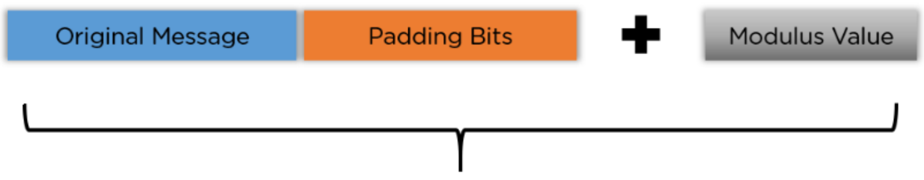
It adds some extra bits to the message, such that the length is exactly 64 bits short of a multiple of 512. During the addition, the first bit should be one, and the rest of it should be filled with zeroes.



Total length to be 64 bits less than multiple of 512

2. **Padding Length: -**

You can add 64 bits of data now to make the final plaintext a multiple of 512. You can calculate these 64 bits of characters by applying the modulus to your original cleartext without the padding.



Final Data to be Hashed as a multiple of 512

3. **Initialising the Buffers: -**

You need to initialize the default values for eight buffers to be used in the rounds as follows:

a = 0x6a09e667

b = 0xbb67ae85

c = 0x3c6ef372

d = 0xa54ff53a

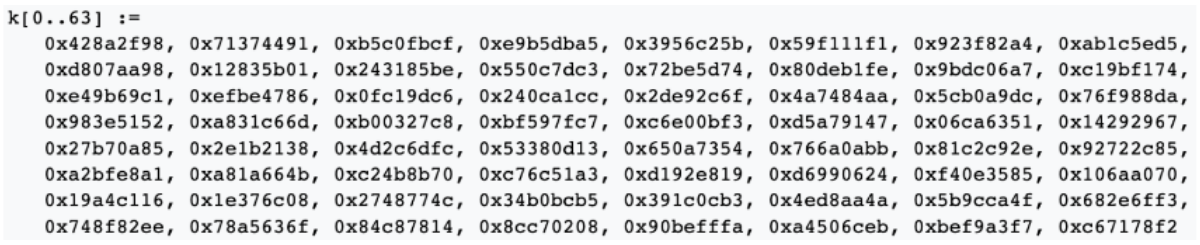
e = 0x510e527f

f = 0x9b05688c

g = 0x1f83d9ab

h = 0x5be0cd19

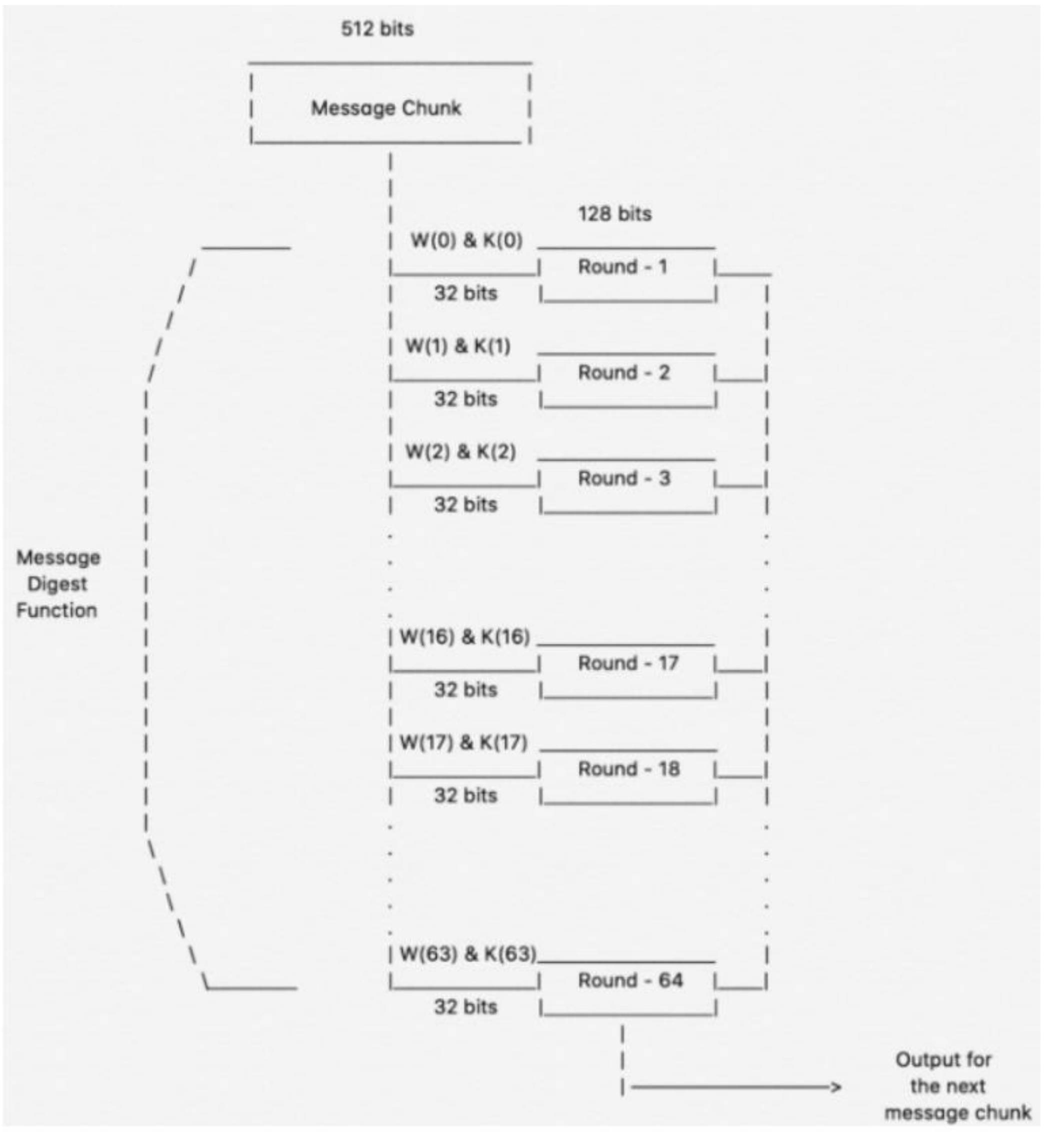
You also need to store 64 different keys in an array, ranging from K [0] to K [63]. They are initialized as follows:



```
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

4. Compression Functions: -

The entire message gets broken down into multiple blocks of 512 bits each. It puts each block through 64 rounds of operation, with the output of each block serving as the input for the following block. The entire process is as follows:



While the value of K[i] in all those rounds is pre-initialized, W[i] is another input that is calculated individually for each block, depending on the number of iterations being processed now.

Advantages: -

- SHA-256 can only be used in one direction. You can convert input to a hash, but it is impossible to convert a SHA-256 back to the contents of the hash.
- The SHA-256 hashing algorithm is regarded by many large organizations (including NIST) as the most secure and most useful hashing protocol in the world.
- Many governments (including the United States and Australia) use SHA-256, which makes the hashing algorithm reliable.
- SHA-256 is not vulnerable to pre-image and second-pre-image attacks (we'll spare you the technical details of these attacks).
- Generating hashes with SHA-256 is quite fast.

Disadvantages: -

- We are not sure whether SHA-256 will always remain an unbreakable (and therefore safe) hashing algorithm. Should more powerful computers ever be developed, they could break the SHA-256 protocol (but we don't know for sure).

Real World Scenario: -

- Digital Signature Verification
- Password Hashing
- SSL Handshake in browsing
- Integrity Checks

References: -

https://www.trentonsystems.com/blog/symmetric-vs-asymmetric-encryption

https://www.techtarget.com/searchsecurity/definition/asymmetric-cryptography#:~:text=Asymmetric%20encryption%20uses%20a%20mathematically,key%20is%20used%20for%20decryption.

https://www.simplilearn.com/tutorials/cryptography-tutorial/rsa-algorithm

https://www.educative.io/answers/what-is-the-rsa-algorithm

https://unacademy.com/content/gate-cse-it/rsa-full-form/

https://www.geeksforgeeks.org/security-of-rsa/

https://dedekindsparadise.wordpress.com/2011/07/24/limitations-of-rsa/

https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm

https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/

https://www.tutorialspoint.com/what-are-the-advantage-and-disadvantage-of-des#:~:text=DES%20is%20also%20an%20ANSI,by%20supporting%20the%20correct%20password.

https://intellipaat.com/blog/what-is-des-algorithm/#:~:text=Enroll%20today!-,Applications%20of%20DES%20Algorithm,legacy%20systems%20with%20three%20keys.

https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

https://wiki.rugdoc.io/docs/introduction-to-sha-256/