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 analyzing protocols which 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. 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 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.

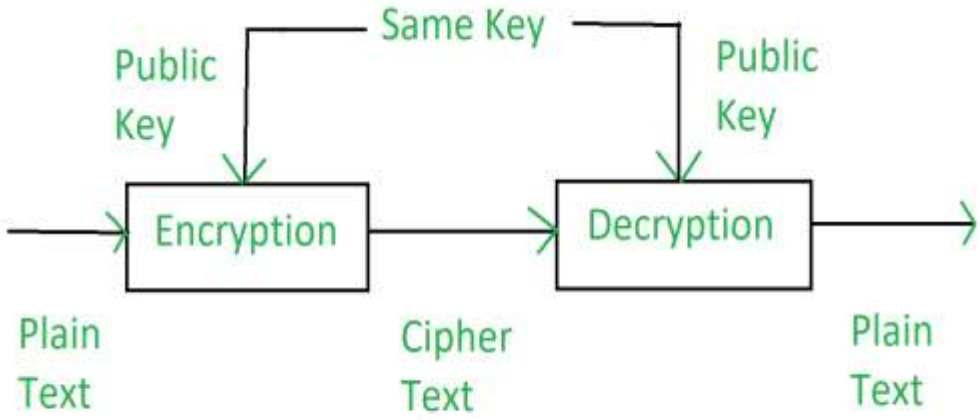
Classical Cryptography and Quantum Cryptography

Cryptography techniques can be categorized according to their basic principles or protocols they follow. But here we are going to concentrate on the two types of cryptography technique: **Classical Cryptography** and **Quantum Cryptography**. These are explained as following below.

1. Classical Cryptography:
Classical cryptography is based on the mathematics and it relies on the computational difficulty of factorizing large number. The security of classical cryptography is based on the high complexity of the mathematical problem for the instance factorization of large number.

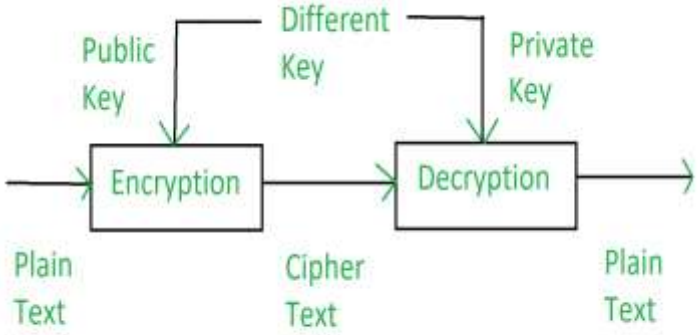
Classical Cryptography has two types of techniques:

1. **Symmetric Cryptography:**
   In the symmetric cryptography a single key is used for encrypting and decryption the data. This encryption key is private key. This is the limitation of this encryption technique that this private key must be distributed only among the authorized sender and receiver.



Symmetric Cryptography

2. **Asymmetric Cryptography:**
In the asymmetric cryptography a pair of key, i.e., public key and private key is used for encryption and decryption. A sender can use its public key to encrypt the data and on receiver end receiver can decrypt the data by using its private key. This technique overcomes the problem of key distribution.



Asymmetric Cryptography

Advantages of Classical Cryptography:

- While employing the one-time pad, it is unbreakable.
- It is easy to do manually, no computer required.
- It protects the plain text from casual snooping.

Disadvantages of Classical Cryptography:

- While employing the one-time pad, it is cumbersome and requires a personal meetup to exchange the pads.
- If not employing the OTP, anyone who is even remotely interested in knowing what you wrote and knows about cryptography will be able to break the encryption.

2. Quantum Cryptography:

Quantum Cryptography is based on physics and it relies on the laws of quantum mechanics. It is arising technology which emphasizes the phenomena of quantum physics in which two parties can have secure communication based on the invariabilities of the laws of the quantum mechanics.

There are 2 important elements of quantum mechanics on which quantum cryptography depends: **Heisenberg Uncertainty Principle** and **Photon Polarization Principle**. These are explained as following below.

1. **Heisenberg Uncertainty Principle:**
   This principle says that if you measure one thing, you cannot measure another thing accurately. For example, if you apply this principle to human, you could measure a person's height, but you can't measure his weight. The only odd thing about this principle is that it becomes true only for the instant at which you try to measure something. This principle is applied to the photons. Photons have wave like structure and are polarized or tilted in certain direction. While measuring photon polarization, all subsequent measurements are get affected by the choice of measures that we made for polarization. This principle plays the vital role to prevent the efforts of attacker in quantum cryptography.
2. **Photon Polarization Principle:**
   This principle refers that, an eavesdropper cannot copy the unique quantum bits, i.e., unknown quantum state, due to the no-cloning principle. If an attempt is made for measuring any properties, it will disturb the other information.

Advantages of Quantum Cryptography:

- It establishes secure communication by providing security based on fundamental laws of physics instead of mathematical algorithms or computing technologies used today.
- It is virtually unhackable.
- It is simple to use.
- Less resources are needed in order to maintain it.
- It is used to detect eavesdropping in QKD (Quantum Key Distribution). This is due to the fact that it is not possible to copy the data encoded in quantum state.
- The performance of such cryptography systems is continuously improved.

Disadvantages of Quantum Cryptography:

- The world wide implementation of this can take up lots of jobs and hence unemployment will increase.
- While traveling through the channel polarization of photon may change due to various causes.
- Quantum cryptography lacks many vital features such as digital signature, certified mail etc.
- The largest distance supported by QKD is about 250 KM at a speed of 16 bps through guided medium.

BLOCK CIPER

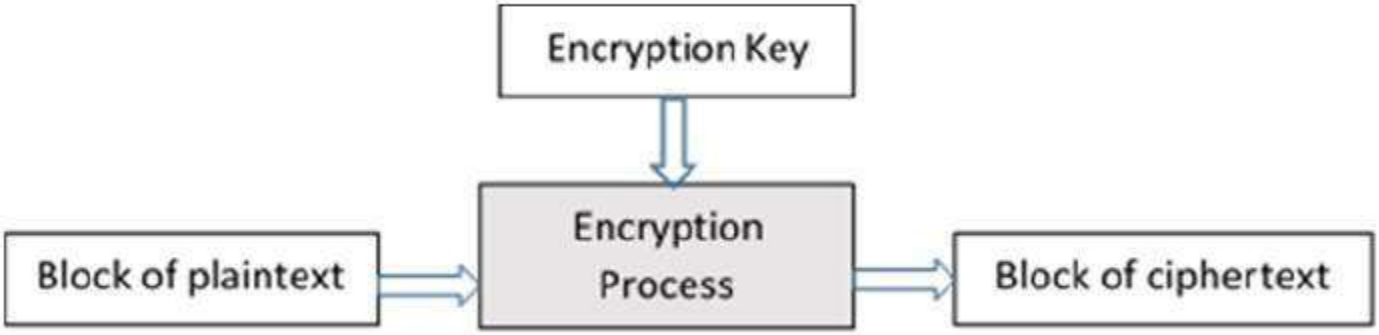
What is a block cipher?

A block cipher is a method of encrypting data in blocks to produce ciphertext using a cryptographic key and algorithm. The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time. Most modern block ciphers are designed to encrypt data in fixed-size blocks of either 64 or 128 bits.

How does a block cipher work?

A block cipher uses a symmetric key and algorithm to encrypt and decrypt a block of data. A block cipher requires an initialization vector (IV) that is added to the input plaintext in order to increase the keyspace of the cipher and make it more difficult to use brute force to break the key. The IV is derived from a random number generator, which is combined with text in the first block and the key to ensure all subsequent blocks result in ciphertext that does not match that of the first encryption block.

The basic scheme of a block cipher is depicted as follows −



A block cipher takes a block of plaintext bits and generates a block of ciphertext bits, generally of same size. The size of block is fixed in the given scheme. The choice of block size does not directly affect to the strength of encryption scheme. The strength of cipher depends up on the key length.

Block Size

Though any size of block is acceptable, following aspects are borne in mind while selecting a size of a block.

- **Avoid very small block size** − Say a block size is m bits. Then the possible plaintext bits combinations are then 2^m. If the attacker discovers the plain text blocks corresponding to some previously sent ciphertext blocks, then the attacker can launch a type of 'dictionary attack' by building up a dictionary of plaintext/ciphertext pairs sent using that encryption key. A larger block size makes attack harder as the dictionary needs to be larger.
- **Do not have very large block size** − With very large block size, the cipher becomes inefficient to operate. Such plaintexts will need to be padded before being encrypted.
- **Multiples of 8 bit** − A preferred block size is a multiple of 8 as it is easy for implementation as most computer processor handle data in multiple of 8 bits.

Padding in Block Cipher

Block ciphers process blocks of fixed sizes (say 64 bits). The length of plaintexts is mostly not a multiple of the block size. For example, a 150-bit plaintext provides two blocks of 64 bits each with third block of balance 22 bits. The last block of bits needs to be padded up with redundant information so that the length of the final block equal to block size of the scheme. In our example, the remaining 22 bits need to have additional 42 redundant bits added to provide a complete block. The process of adding bits to the last block is referred to as **padding**.

Too much padding makes the system inefficient. Also, padding may render the system insecure at times, if the padding is done with same bits always.

Block Cipher Schemes

There is a vast number of block ciphers schemes that are in use. Many of them are publically known. Most popular and prominent block ciphers are listed below.

- **Digital Encryption Standard (DES)** − The popular block cipher of the 1990s. It is now considered as a 'broken' block cipher, due primarily to its small key size.
- **Triple DES** − It is a variant scheme based on repeated DES applications. It is still a respected block ciphers but inefficient compared to the new faster block ciphers available.
- **Advanced Encryption Standard (AES)** − It is a relatively new block cipher based on the encryption algorithm **Rijndael** that won the AES design competition.
- **IDEA** − It is a sufficiently strong block cipher with a block size of 64 and a key size of 128 bits. A number of applications use IDEA encryption, including early versions of Pretty Good Privacy (PGP) protocol. The use of IDEA scheme has a restricted adoption due to patent issues.
- **Twofish** − This scheme of block cipher uses block size of 128 bits and a key of variable length. It was one of the AES finalists. It is based on the earlier block cipher Blowfish with a block size of 64 bits.
- **Serpent** − A block cipher with a block size of 128 bits and key lengths of 128, 192, or 256 bits, which was also an AES competition finalist. It is a slower but has more secure design than other block cipher.

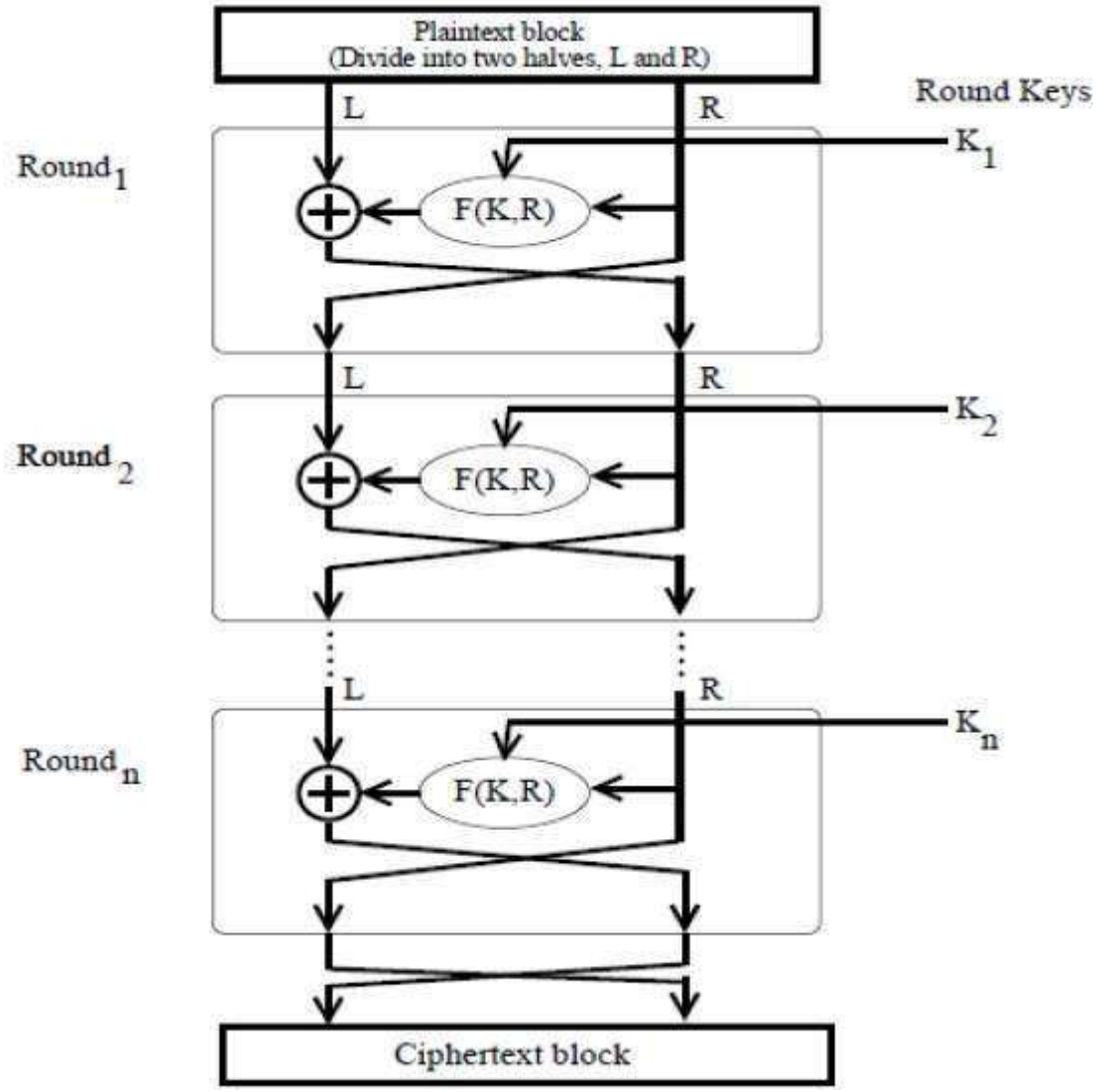
Feistel Block Cipher

Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived. DES is just one example of a Feistel Cipher. A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.

Encryption Process

The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext, each round consisting of a "substitution" step followed by a permutation step.

Feistel Structure is shown in the following illustration −



- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.
- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function 'f' that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a subkey) is derived from the encryption key. This means that each round uses a different key, although all these subkeys are related to the original key.
- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a 'round'. The number of rounds are specified by the algorithm design.
- Once the last round is completed then the two sub blocks, 'R' and 'L' are concatenated in this order to form the ciphertext block.

The difficult part of designing a Feistel Cipher is selection of round function 'f'. In order to be unbreakable scheme, this function needs to have several important properties that are beyond the scope of our discussion.

Decryption Process

The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.

The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.

The final swapping of 'L' and 'R' in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.

Number of Rounds

The number of rounds used in a Feistel Cipher depends on desired security from the system. More number of rounds provide more secure system. But at the same time, more rounds mean the inefficient slow encryption and decryption processes. Number of rounds in the systems thus depend upon efficiency–security tradeoff.

Data Encryption Standard

Data Encryption Standard (DES)

DES stands for Data Encryption Standard. There are certain machines that can be used to crack the DES algorithm. The DES algorithm uses a key of 56-bit size. Using this key, the DES takes a block of 64-bit plain text as input and generates a block of 64-bit cipher text.

The DES process has several steps involved in it, where each step is called a round. Depending upon the size of the key being used, the number of rounds varies. For example, a 128-bit key requires 10 rounds, a 192-bit key requires 12 rounds, and so on.

What is the DES Algorithm in Cyber Security?

The DES (Data Encryption Standard) algorithm is a symmetric-key block cipher created in the early 1970s by an IBM team and adopted by the National Institute of Standards and Technology (NIST). The algorithm takes the plain text in 64-bit blocks and converts them into ciphertext using 48-bit keys.

Since it's a symmetric-key algorithm, it employs the same key in both encrypting and decrypting the data. If it were an asymmetrical algorithm, it would use different keys for encryption and decryption

Initial Permutation (IP)

The plain text is divided into smaller chunks of 64-bit size. The IP is performed before the first round. This phase describes the implementation of the transposition process. For example, the 58th bit replaces the first bit, the 50th bit replaces the second bit, and so on. The resultant 64-bit text is split into two equal halves of 32-bit each called Left Plain Text (LPT) and Right Plain Text (RPT).

Step 1: Key Transformation

Step 2: Expansion Permutation

Triple DES Algorithm

Triple DES is a symmetric key-block cipher which applies the DES cipher in triplicate. It encrypts with the first key (k1), decrypts using the second key (k2), then encrypts with the third key (k3). There is also a two-key variant, where k1 and k3 are the same keys.

DES Algorithm Steps

The algorithm process breaks down into the following steps:

1. The process begins with the 64-bit plain text block getting handed over to an initial permutation (IP) function.
2. The initial permutation (IP) is then performed on the plain text.
3. Next, the initial permutation (IP) creates two halves of the permuted block, referred to as Left Plain Text (LPT) and Right Plain Text (RPT).
4. Each LPT and RPT goes through 16 rounds of the encryption process.
5. Finally, the LPT and RPT are rejoined, and a Final Permutation (FP) is performed on the newly combined block.
6. The result of this process produces the desired 64-bit ciphertext.

The encryption process step (step 4, above) is further broken down into five stages:

1. Key transformation
2. Expansion permutation
3. S-Box permutation
4. P-Box permutation
5. XOR and swap

For decryption, we use the same algorithm, and we reverse the order of the 16 round keys.

DES Modes of Operation

Experts using DES have five different modes of operation to choose from.

- Electronic Codebook (ECB). Each 64-bit block is encrypted and decrypted independently
- Cipher Block Chaining (CBC). Each 64-bit block depends on the previous one and uses an Initialization Vector (IV)
- Cipher Feedback (CFB). The preceding ciphertext becomes the input for the encryption algorithm, producing pseudorandom output, which in turn is XORed with plaintext, building the next ciphertext unit
- Output Feedback (OFB). Much like CFB, except that the encryption algorithm input is the output from the preceding DES
- Counter (CTR). Each plaintext block is XORed with an encrypted counter. The counter is then incremented for each subsequent block

DES Implementation and Testing

DES implementation requires a security provider. However, there are many available providers to choose from, but selecting one is the essential initial step in implementation. Your selection may depend on the language you are using, such as Java, Python, C, or MATLAB.

Once you decide on a provider, you must choose whether to have a random secret key generated by the KeyGenerator or create a key yourself, using a plaintext or byte array.

It's also essential to test the encryption to make sure it is properly implemented. You can find a testing procedure that will do the trick using the recurrence relation found on GitHub.

Now that we have come so far in our understanding of what is DES, let us next look into the reasons to learn DES.

Applications of DES Algorithm

In this section, we are going to learn about some of the applications of the DES Algorithm.

1. It is used in random number generation
2. It is deployed when not-so-strong encryption is needed
3. It is used to develop a new form of DES, called Triple DES (using a 168-bit key formed using three keys)

Difference Between DES and AES algorithms

DES	AES
Used to encrypt plain text of 64-bit	Used to encrypt plain text of 128-bit
The key is of 56-bit size.	The key is of different sizes such as 128-bits, 192-bits, and so on
Less secure than AES	More secure than DES
It can be broken by brute force attacks	To date, AES has not been attacked
It is based on Feistel network	It is based on permutation and substitution network

Advantages and Disadvantages of DES Algorithm

The advantages of the DES algorithm:

1. It is set as a standard by the US government.
2. When compared to the software, it works faster on hardware.
3. Triple DES, used a 168-bit key which is very hard to crack.

The disadvantages of the DES algorithm:

1. Weakly secured algorithm.
2. There is a threat from Brute force attacks.
3. A DES cracker machine known as Deep Crack is available in the market.

Stream Cipher

In stream cipher, one byte is encrypted at a time while in block cipher ~128 bits are encrypted at a time.

Initially, a key(k) will be supplied as input to pseudorandom bit generator and then it produces a random 8-bit output which is treated as keystream.

The resulted keystream will be of size 1 byte, i.e., 8 bits.

1. Stream Cipher follows the sequence of pseudorandom number stream.
2. One of the benefits of following stream cipher is to make cryptanalysis more difficult, so the number of bits chosen in the Keystream must be long in order to make cryptanalysis more difficult.
3. By making the key more longer it is also safe against brute force attacks.
4. The longer the key the stronger security is achieved, preventing any attack.
5. Keystream can be designed more efficiently by including more number of 1s and 0s, for making cryptanalysis more difficult.
6. Considerable benefit of a stream cipher is, it requires few lines of code compared to block cipher.

Encryption :
For Encryption,

- Plain Text and Keystream produces Cipher Text (Same keystream will be used for decryption.).
- The Plaintext will undergo XOR operation with keystream bit-by-bit and produces the Cipher Text.

Example –

```
Plain Text : 10011001
Keystream  : 11000011
``````````````````````
Cipher Text : 01011010
```

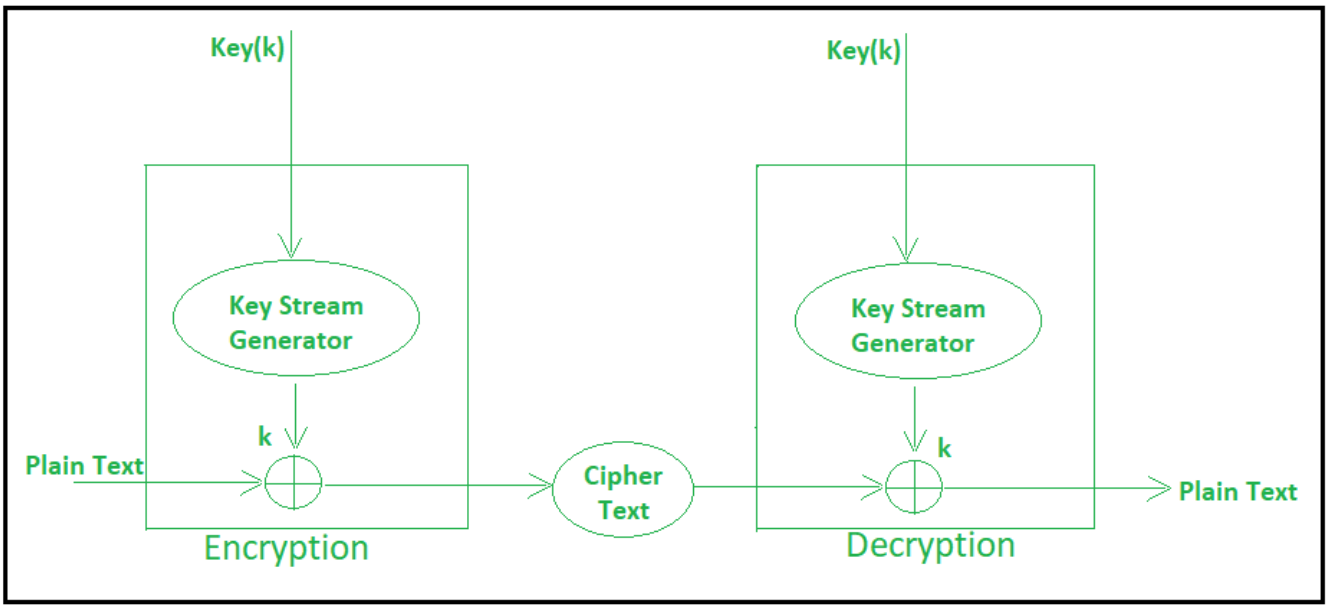
**Decryption :**
For Decryption,

- Cipher Text and Keystream gives the original Plain Text (Same keystream will be used for encryption.).
- The Ciphertext will undergo XOR operation with keystream bit-by-bit and produces the actual Plain Text.

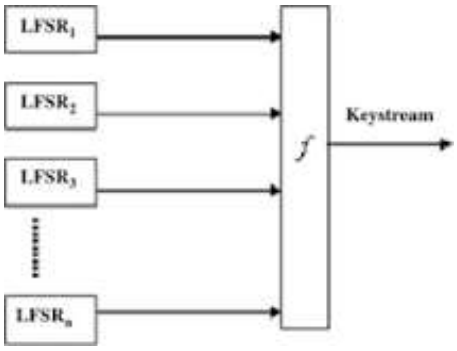
**Example –**

```
Cipher Text : 01011010
Keystream   : 11000011
``````````````````````
Plain Text  : 10011001
```

Decryption is just the reverse process of Encryption i.e. performing XOR with Cipher Text.



## What is LFSR based stream cipher?




LFSR based stream ciphers. **Linear feedback shift registers are used in many of the keystream generators proposed in literature due to their simple hardware structure**. They can produce sequences of large period and good statistical properties. An LFSR of length L consists of L elements capable of storing one bit each.

## Mathematical Background

In order to understand some of the cryptographic algorithms dealt with throughout this course, it is necessary to have some background in two areas of mathematics
1. Number Theory.
2. Abstract Algebra.

- New Advanced Encryption Standard (AES) relies on the subject of finite fields which forms a part of abstract algebra.
- Going to deal with Number Theory for the moment.

## ABSTRACT ALGEBRA

- Abstract algebra basically comprises Groups, Rings, Fields, Vector Spaces, Modules, and many other algebraic structures. It is not only useful in Cryptography but in Channel Coding, in the branch of Chemistry and Physics too. It is a tool like Groups where you can do arithmetic and algebraic operation under a closed set. One can compare two algebraic structures and find similarities between them and use these properties further.

- Now your question about AES and Galois Field. if you do arithmetic operation on the matrix of AES, say multiplying two bytes 0xEF

and 0xDC

- , without finite field your many arithmetic operations go beyond 1 byte of length. **Finite** field make sure that it will remain within the closed set by reducing to modulo.

- for example; if you take the finite field created by the prime number 11 ($\mathbf{Z}_{*11}$

) the set only contains {1,2,…,10}

elements. All operations are done within this set. Then only you can substitute bytes find inverses when you decrypt AES encrypted ciphertext.

if you don't use them, strictly speaking without irreducible polynomial (like in AES it is $x^8+x^4+x^3+x+1$

) or prime number, where it makes sure that all length of all operations ends up in byte and hence easy to find their **unique** inverses while decryption.

## NUMBER THEORY

# 1 Primality Testing and RSA

The first stage of key-generation for RSA involves finding two large primes p, q
Because of the size of numbers used, must find primes by trial and error
Modern primality tests utilize properties of primes eg:
$a^{n-1} = 1 \bmod n$ where GCD(a,n)=1
all primes numbers 'n' will satisfy this equation

some composite numbers will also satisfy the equation, and are called pseudo-primes.Most modern tests guess at a prime number 'n', then take a large number (eg 100) of numbers 'a', and apply this test to each. If it fails the number is composite, otherwise it is is probably prime.

here are a number of stronger tests which will accept fewer composites as prime than the above test. eg:

$$\text{GCD}(a,n) = 1, \quad \text{and} \quad \left(\frac{a}{n}\right) (\text{mod } n) = a^{\frac{(n-1)}{2}} (\text{mod } n)$$

where $\left(\frac{a}{n}\right)$ is the Jacobi symbol

**RSA Implementation in Practice**

Software implementations

generally perform at 1-10 bits/second on block sizes of 256-512 bits

two main types of implementations:

- on micros as part of a key exchange mechanism in a hybrid scheme
- on larger machines as components of a secure mail system

Harware Implementations

generally perform 100-10000 bits/sec on blocks sizes of 256-512 bits

all known implementations are large bit length conventional ALU units

2 Euler Totient Function [[phi]](n)

## Modular multiplicative inverse

Given two integers **A** and **M**, find the modular multiplicative inverse of **A** under modulo **M**.
The modular multiplicative inverse is an integer **X** such that:

A X ≅ 1 (mod M)

**Note:** The value of **X** should be in the range {1, 2, … M-1}, i.e., in the range of integer modulo **M**. ( Note that **X** cannot be 0 as A*0 mod M will never be 1). The multiplicative inverse of "A modulo M" exists if and only if A and M are relatively prime (i.e. if gcd(A, M) = 1)

**Examples:**

**Input:** A = 3, M = 11
**Output:** 4
**Explanation:** Since (4*3) mod 11 = 1, 4 is modulo inverse of 3(under 11).
One might think, 15 also as a valid output as "(15*3) mod 11"
is also 1, but 15 is not in range {1, 2, … 10}, so not valid.

**Input:** A = 10, M = 17
**Output:** 12
**Explanation:** Since (10*12) mod 17 = 1, 12 is modulo inverse of 10(under 17)

# Euclidean algorithms

The Euclidean algorithm is a way to find the greatest common divisor of two positive integers. GCD of two numbers is the largest number that divides both of them. A simple way to find GCD is to factorize both numbers and multiply common prime factors.

```
36 = 2 x 2 x 3 x 3
60 = 2 x 2 x 3 x 5

GCD = Multiplication of common factors
    = 2 x 2 x 3
    = 12
```

## Basic Euclidean Algorithm for GCD:

The algorithm is based on the below facts.

- If we subtract a smaller number from a larger one (we reduce a larger number), GCD doesn't change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.
- Now instead of subtraction, if we divide the smaller number, the algorithm stops when we find the remainder 0.

```
// C program to demonstrate Basic Euclidean Algorithm
#include <stdio.h>

// Function to return gcd of a and b
int gcd(int a, int b)
```

```
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}

// Driver code
int main()
{
    int a = 10, b = 15;

      // Function call
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a, b));
    a = 35, b = 10;
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a, b));
    a = 31, b = 2;
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a, b));
    return 0;
}
```

**Output**

```
GCD(10, 15) = 5
GCD(35, 10) = 5
GCD(31, 2) = 1
```

**Time Complexity:** O(Log min(a, b))
**Auxiliary Space:** O(Log (min(a,b))

## Extended Euclidean Algorithm:

Extended Euclidean algorithm also finds integer coefficients x and y such that: ax + by = gcd(a, b)

**Examples:**

**Input:** a = 30, b = 20
**Output:** gcd = 10, x = 1, y = -1
(Note that 30*1 + 20*(-1) = 10)

**Input:** a = 35, b = 15
**Output:** gcd = 5, x = 1, y = -2
(Note that 35*1 + 15*(-2) = 5)

The extended Euclidean algorithm updates the results of gcd(a, b) using the results calculated by the recursive call gcd(b%a, a). Let values of x and y calculated by the recursive call be $x_1$ and $y_1$. x and y are updated using the below expressions.

ax + by = gcd(a, b)
gcd(a, b) = gcd(b%a, a)
gcd(b%a, a) = $(b\%a)x_1 + ay_1$
ax + by = $(b\%a)x_1 + ay_1$
ax + by = $(b - [b/a] * a)x_1 + ay_1$
ax + by = $a(y_1 - [b/a] * x_1) + bx_1$

Comparing LHS and RHS,
$x = y_1 – \lfloor b/a \rfloor * x_1$
$y = x_1$

### How does Extended Algorithm Work?

As seen above, x and y are results for inputs a and b,

$a.x + b.y = gcd$ —-(1)

And $x_1$ and $y_1$ are results for inputs b%a and a

$(b\%a).x_1 + a.y_1 = gcd$

When we put $b\%a = (b – (\lfloor b/a \rfloor).a)$ in above,
we get following. Note that $\lfloor b/a \rfloor$ is floor(b/a)

$(b – (\lfloor b/a \rfloor).a).x_1 + a.y_1 = gcd$

Above equation can also be written as below

$b.x_1 + a.(y_1 – (\lfloor b/a \rfloor).x_1) = gcd$ —(2)

After comparing coefficients of ‘a’ and ‘b’ in (1) and
(2), we get following,
$x = y_1 – \lfloor b/a \rfloor * x_1$
$y = x_1$

### How is Extended Algorithm Useful?

The extended Euclidean algorithm is particularly useful when a and b are coprime (or gcd is 1). Since x is the modular multiplicative inverse of “a modulo b”, and y is the modular multiplicative inverse of “b modulo a”. In particular, the computation of the modular multiplicative inverse is an essential step in RSA public-key encryption method.

## Fermat’s Little Theorem

Fermat's little theorem is a fundamental theorem in elementary number theory, which provides compute powers of integers modulo prime numbers. It is a specific case of Euler's theorem, and is essential in applications of elementary number theory, such as primality testing and public-key cryptography. This is referred to as Fermat’s little theorem.

Fermat’s theorem is also called a Fermat’s little theorem defines that is P is prime and ‘a’ is a positive integer not divisible by P then −

$$a^{P-1} \equiv 1 \bmod P$$

Second condition says that if P is a prime and a is an integer then $a^P \equiv 1 \bmod P$.

**Proof** − $Z_p$ is the set of integer {0, 1…P-1} when multiplied by a modulo P, the result includes all the element of $Z_p$ in some sequence, moreover a x 0 ≡ 0 mod P. Thus, the (P-1) numbers {a mod P, 2a mod P,…((P-1) a mod P)} are only the number {1, 2,…(P-1)} in some order.

Multiplying the numbers in both steps and taking the result mod P gives

$$a \times 2a \times \ldots \times ((P-1)a) = [(a \bmod P) \times (2a \bmod P) \times \ldots \times ((P-1)a \bmod P)] \bmod P$$

$$= [1 \times 2 \times \ldots \times (P-1)] \bmod P$$

$$= (P-1)! \bmod P$$

But

$$a \times 2a \times \ldots \times ((P-1)a) = (P-1)!a^{P-1}$$

$$(P-1)!\, a^{P-1} \equiv (P-1)! \bmod P$$

$$a^{P-1} \equiv 1 \bmod P$$

Consider the set of positive integers less than p: {1, 2... p1} and multiply each element by a, modulo p, to receive the set X = {a mod p, 2a mod p . . . (p1) a mod p}. None of the elements of X is similar to zero because p does not divide a.

Furthermore no two of the integers in X are same. To see this, consider that (ja ≡ p) where 1 ≤ p1. Because p, it can delete a from both sides of the equation resulting in − j≡ p).

This final similarity is inaccessible due to j and k are both positive integers less than p. Therefore, it is understand that the (p1) elements of X are all positive integers, with no two elements same.

Numerical − Fermat's theorem states that if p is prime and a is a positive integer not divisible by P, then $a^{P-1} \equiv 1 \pmod p$.

Therefore, $3^{10} \equiv 1 \pmod{11}$.

Therefore, $3^{201} = (3^{10})^{20} \times 3 \equiv 3 \pmod{11}$.

Fermats little theorem sometimes is beneficial for quickly discovering a solution to some exponentiations. The following examples show the concept.

**Example1** − Find the result of $6^{10} \bmod 11$.

**Solution**

Here, we have $6^{10} \bmod 11 = 1$. This is the first version of Fermat's little theorem where p = 11.

**Example2** − Find the result of $3^{12} \bmod 11$.

**Solution**

Therefore the exponent (12) and the modulus (11) are not equal. With substitution this can be defined utilizing Fermat's little theorem.

$$3^{12} \bmod 11 = (3^{11} \times 3) \bmod 11 = (3^{11} \bmod 11)(3 \bmod 11) = (3 \times 3) \bmod 11 = 9$$

# EULERS THEOREM

Euler's theorem is a generalization of Fermat's little theorem handling with powers of integers modulo positive integers. It increase in applications of elementary number theory, such as the theoretical supporting structure for the RSA cryptosystem.

This theorem states that for every a and n that are relatively prime −

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

where $\phi$

(n) is Euler's totient function, which counts the number of positive integers less than n that are relatively prime to n.

Consider the set of such integers −

$R = \{x_1, x_2, \ldots x_{\phi}$

$_{(n)}\}$, i.e., each element xi of R is unique positive integer less than n with ged($x_i$, n) = 1. Then multiply each element by a and modulo n −

$S = \{(ax_1 \bmod n), (ax_2 \bmod n), \ldots (ax_{\phi}$

$_{(n)} \bmod n)\}$

Because a is relatively prime to n and $x_i$ is relatively prime to n, $ax_i$ must also be relatively prime to n. Therefore, all the members of S are integers that are less than n and that are relatively prime to n.

There are no duplicates in S.

If $ax_i \bmod n$ and $n = ax_j \bmod n$ then $x_i = x_j$

Therefore,

$$\Pi_{i=1}^{\phi(n)} (ax_i \bmod n) = \Pi_{i=1}^{\phi(n)} x_i$$

$$\Pi_{i=1}^{\phi(n)} ax_i \equiv \Pi_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} x [\Pi_{i=1}^{\phi(n)} x_i] = \Pi_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

## Euler Totient Function

Euler's Totient function is the mathematical multiplicative functions which count the positive integers up to the given integer generally known as as 'n' that are a prime number to 'n' and the function can be used to understand the number of prime numbers that ex$_i$st up to the given integer 'n'.

Euler's Totient function is also called as Euler's phi function. It plays an essential role in cryptography. It can discover the number of integers that are both smaller than n and relatively prime to n. These set of numbers defined by $Z_{*n}$

(number that are smaller than n and relatively prime to n).

Euler's totient function is beneficial in several ways. It can be used in the RSA encryption system, which can be used for security goals. The function deals with the prime number theory, and it is beneficial in the computation of large calculations also. The function can be utilized in algebraic computation and simple numbers.

The symbol used to indicate the function is ϕ, and it is also known as phi function. The function includes more theoretical use instead of practical use. The sensible requirement of the function is limited.

The function can be better understood through the several practical examples instead of only theoretical explanations. There are several rules for computing the Euler's totient function, and for different numbers, different rules are to be used.

The Euler totient function $\phi$

(n) calculates the number of elements in $Z_{*n}$

with the help of the following rules −

- $\phi$

- (1) = 0.

- $\phi$

- (P) = P − 1 if P is a Prime.

- $\phi$

(m x n) = $\phi$(m) x $\phi$

- (n) if m and n are relatively prime.

- $\phi$

- $(P^e) = P^e - P^{e-1}$ (if P is a prime. )

The following four rules can be combined to obtain the value of $\phi$

(n), factorize n as

$$n=P_{e11}xP_{e22}x\cdots P_{ekk}$$

$$\phi(n)=(P_{e11}-P_{e1-11})(P_{e22}-P_{e2-12})x\cdots x(P_{ekk}-P_{ek-1k})$$

The difficulty of finding $\phi$

(n) depends on the difficulty of finding the factorization of n.

# EULRS PHI FUNCTION

**Euler's phi (or totient) function** of a positive integer *n* is the number of integers in {1,2,3,...,*n*} which are relatively prime to *n*. This is usually denoted φ(*n*).

| integer $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\varphi(n)$ | 1 | 1 | 2 | 2 | 4 | 2 | 6 | 4 | 6 | 4 | 10 | 4 | 12 | 6 | 8 | 8 |

Clearly for primes $p$, $\varphi(p)=p-1$. Since $\varphi(x)$ is a multiplicative function, its value can be determined from its value at the prime powers:

**Theorem**

If $p$ is prime and $n$ is any positive integer, then $\varphi(p^n)$ is $p^{n-1}(p-1)$.

# ADVANCE 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.

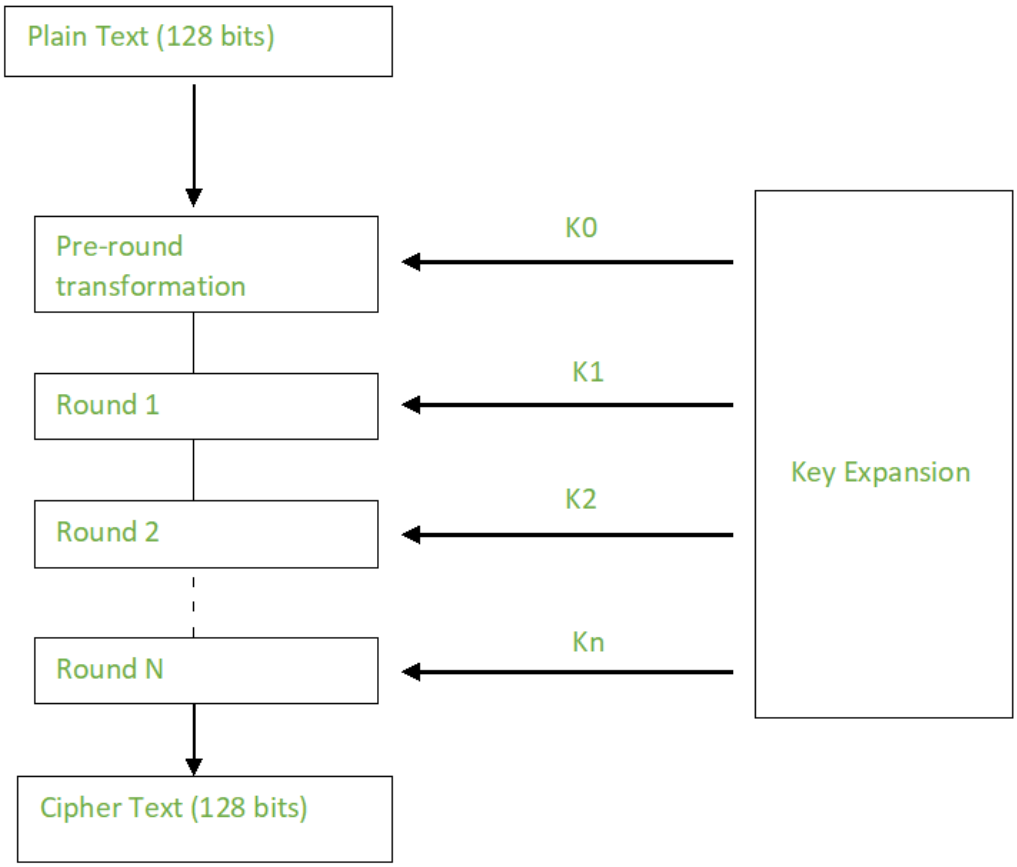
That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

**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.




**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.)

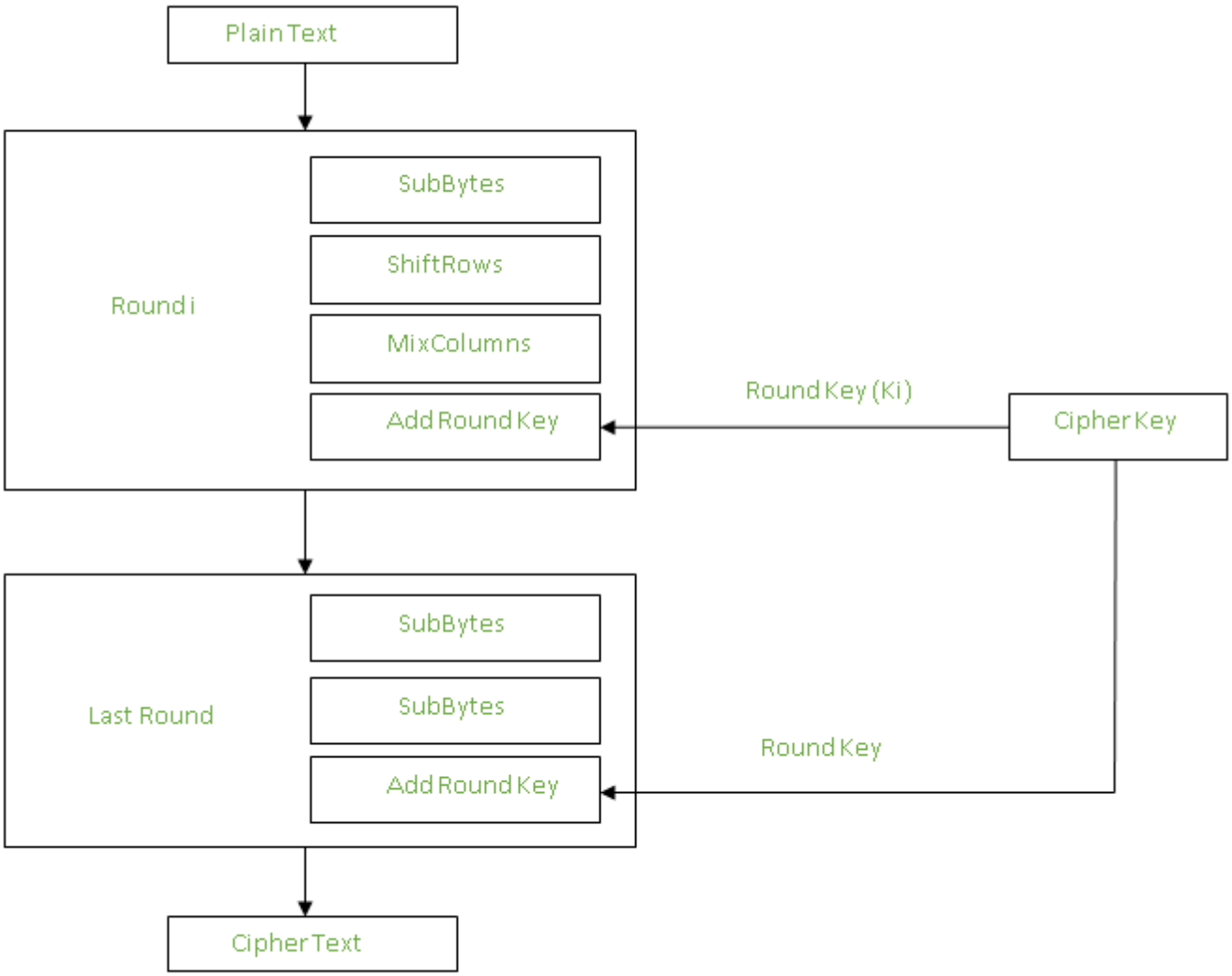
```
[ b0  | b1  | b2  | b3  ]          [ b0  | b1  | b2  | b3  ]
| b4  | b5  | b6  | b7  |    ->    | b5  | b6  | b7  | b4  |
| b8  | b9  | b10 | b11 |          | b10 | b11 | b8  | b9  |
[ b12 | b13 | b14 | b15 ]          [ b15 | b12 | b13 | b14 ]
```

**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.

```
[ 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 SubBytes :**
Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption

## PUBLIC KEY CRYPTOGRAPHY

The most commonly used implementations of public key cryptography (also known as public-key encryption and asymmetric encryption) are based on algorithms presented by Rivest-Shamir-Adelman (RSA) Data Security.

Public key cryptography involves a pair of keys known as a public key and a private key (a *public key pair*), which are associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published and the corresponding private key is kept secret. Data that is encrypted with the public key can be decrypted only with the corresponding private key.

RSA public key pairs can be any size. Typical sizes today are 1024 and 2048 bits.

Public key cryptography enables the following:

- Encryption and decryption, which allow two communicating parties to disguise data that they send to each other. The sender encrypts, or scrambles, the data before sending it. The receiver decrypts, or unscrambles, the data after receiving it. While in transit, the encrypted data is not understood by an intruder.
- Nonrepudiation, which prevents:
    - The sender of the data from claiming, at a later date, that the data was never sent
    - The data from being altered.

Figure 1 shows you a simplified view of how public key cryptography works.
Figure 1. Public-key encryption

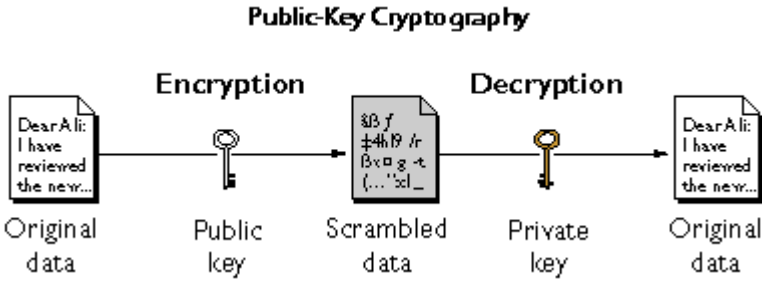



Figure 1 shows how you can freely distribute the public key so that only you (the owner of the private key) can read data that was encrypted with the public key. In general, to send encrypted data to someone, you must encrypt the data with that person's public key, and the person receiving the data decrypts it with the corresponding private key.

If you compare symmetric-key encryption with public-key encryption, you will find that public-key encryption requires more calculations. Therefore, public-key encryption is not always appropriate for large amounts of data. However, it is possible to use public-key encryption to send a symmetric key, which you can then use to encrypt additional data.

The reverse of what is shown in the previous figure also works. That is, data encrypted with your private key can be decrypted only with your public key. However, this is not a desirable way to encrypt sensitive data

because it means that anyone with your public key, which is by definition published, could decrypt the sensitive data. Despite this, private-key encryption is useful because it enables you to use your private key to sign data with your *digital signature*; anyone with your public key can be assured that only you sent the data. This is an important requirement for electronic commerce and other commercial applications of cryptography.

# Diffie-Hellman Key Exchange

## What is Diffie-Hellman Key Exchange (exponential key exchange)?

The Diffie-Hellman key exchange (also known as exponential key exchange) is a method for securely exchanging cryptographic keys over an insecure channel. It is a fundamental building block of many secure communication protocols, including SSL/TLS and SSH.

The Diffie-Hellman key exchange works by allowing two parties (Alice and Bob) to agree on a shared secret key over an insecure channel, without any other party being able to intercept the key or learn anything about it. The key exchange involves the following steps −

- Alice and Bob agree on two large prime numbers, p and g, and a public key exchange algorithm.
- Alice chooses a secret integer, a, and computes A = g^a mod p. She sends A to Bob.
- Bob chooses a secret integer, b, and computes B = g^b mod p. He sends B to Alice.
- Alice computes s = B^a mod p. Bob computes s = A^b mod p.
- Alice and Bob now both have shared secret keys, which they can use to establish a secure communication channel.

The security of the Diffie-Hellman key exchange relies on the fact that it is computationally infeasible for an attacker to determine the shared secret keys from the public values of p, g, A, and B. This allows Alice and Bob to exchange the key securely, even over an insecure channel.

## Where is Diffie-Hellman Key Exchange Used?

The Diffie-Hellman key exchange (also known as exponential key exchange) is a widely used and trusted technique for securely exchanging cryptographic keys over an insecure channel. It is used in many different contexts, including −

- **Secure communication protocols** −
- **Virtual private networks (VPNs)** − **Secure file transfer protocols** −
- **Other applications**

Overall, the Diffie-Hellman key exchange is an important and widely used technique for securely exchanging cryptographic keys and establishing secure communication channels. It is an essential component of many secure communication protocols and applications.

## How does Diffie-Hellman Key Exchange Work?

The Diffie-Hellman key exchange (also known as exponential key exchange) is a method for securely exchanging cryptographic keys over an insecure channel. It works by allowing two parties (Alice and Bob) to agree on a shared secret key without any other party being able to intercept the key or learn anything about it. The key exchange involves the following steps −

- Alice and Bob agree on two large prime numbers, p and g, and a public key exchange algorithm.
- Alice chooses a secret integer, a, and computes A = g^a mod p. She sends A to Bob.

- Bob chooses a secret integer, b, and computes B = g^b mod p. He sends B to Alice.
- Alice computes s = B^a mod p. Bob computes s = A^b mod p.
- Alice and Bob now both have the shared secret key s, which they can use to establish a secure communication channel.

The security of the Diffie-Hellman key exchange relies on the fact that it is computationally infeasible for an attacker to determine the shared secret key s from the public values of p, g, A, and B. This allows Alice and Bob to exchange the key securely, even over an insecure channel.

## Vulnerabilities of Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange (also known as exponential key exchange) is a widely used and trusted technique for securely exchanging cryptographic keys over an insecure channel. However, like all cryptographic systems, it is not completely immune to attacks and vulnerabilities. Some potential vulnerabilities of the Diffie-Hellman key exchange include −

- **Man-in-the-middle attacks** − If an attacker is able to intercept and modify the messages exchanged between Alice and Bob during the key exchange, they may be able to impersonate Alice or Bob and establish a secure channel with the other party. This can be prevented by using certificate-based authentication and/or by verifying the authenticity of the messages using message authentication codes (MACs).
- **Small subgroup attacks** − If the prime number p used in the key exchange has a small subgroup, an attacker may be able to use this to their advantage to recover the shared secret key. To prevent this, it is important to use a large prime number with no known small subgroups.
- **Exponent attacks** − If the secret exponents (a and b) used in the key exchange are not chosen randomly, an attacker may be able to use this to their advantage to recover the shared secret key. To prevent this, it is important to use a strong random number generator to generate the secret exponents.

## Examples of Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange (also known as exponential key exchange) is a widely used and trusted technique for securely exchanging cryptographic keys over an insecure channel. It is used in many different contexts, including secure communication protocols, virtual private networks (VPNs), secure file transfer protocols, and other applications where secure communication is required. Some examples of the use of the Diffie-Hellman key exchange include −

- **SSL/TLS** − The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols use the Diffie-Hellman key exchange to establish a secure channel between a client and a server. This allows the client and server to exchange encrypted messages over an insecure network, such as the Internet.
- **SSH** − The Secure Shell (SSH) protocol uses the Diffie-Hellman key exchange to establish a secure channel between a client and a server. This allows users to securely log in to a remote server and execute commands, transfer files, and perform other tasks over an insecure network.
- **VPNs** − Many VPN protocols, such as IPSec and OpenVPN, use the Diffie-Hellman key exchange to establish a secure connection between a client and a server. This allows the client and server to exchange encrypted traffic over an insecure network, such as the Internet.
- **SFTP** − The Secure File Transfer Protocol (SFTP) uses the Diffie-Hellman key exchange to establish a secure channel between a client and a server. This allows users to securely transfer files between two systems over an insecure network.

# Knapsack Cyrptosystem

Knapsack cryptosystems are **cryptosystems whose security is based on the hardness of solving the knapsack problem**. They remain quite unpopular because simple versions of these algorithms have been broken for several decades. However, that type of cryptosystem is a good candidate for post-quantum cryptography.

## Knapsack Encryption Algorithm in Cryptography

**Knapsack Encryption Algorithm** is the first general public key cryptography algorithm. It is developed by **Ralph Merkle** and **Mertin Hellman** in 1978. As it is a Public key cryptography, it needs two different keys. One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process. In this algorithm we will use two different knapsack problems in which one is easy and other one is hard. The easy knapsack is used as the private key and the hard knapsack is used as the public key. The easy knapsack is used to derived the hard knapsack.

For the easy knapsack, we will choose a **Super Increasing knapsack problem**. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

**Example –**

```
{1, 2, 4, 10, 20, 40} is a super increasing as
1<2, 1+2<4, 1+2+4<10, 1+2+4+10<20 and 1+2+4+10+20<40.
```

**Derive the Public key**

- **Step-1:**
  Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.

- **Step-2:**
  Choose two numbers n and m. Multiply all the values of private key by the number n and then find modulo m. The value of m must be greater than the sum of all values in private key, for example 110. And the number n should have no common factor with m, for example 31.

- **Step-3:**
  Calculate the values of Public key using m and n.

```
1x31 mod(110) = 31
2x31 mod(110) = 62
4x31 mod(110) = 14
10x31 mod(110) = 90
20x31 mod(110) = 70
40x31 mod(110) = 30
```

- Thus, our public key is {31, 62, 14, 90, 70, 30}
  And Private key is {1, 2, 4, 10, 20, 40}.

Now take an example for understanding the process of encryption and decryption.

**Example –**
Lets our plain text is 100100111100101110.

**1. Encryption :**
As our knapsacks contain six values, so we will split our plain text in a groups of six:

```
100100  111100  101110
```

Multiply each values of public key with the corresponding values of each group and take their sum.

```
100100  {31, 62, 14, 90, 70, 30}
1x31+0x62+0x14+1x90+0x70+0x30 = 121

111100  {31, 62, 14, 90, 70, 30}
1x31+1x62+1x14+1x90+0x70+0x30 = 197

101110  {31, 62, 14, 90, 70, 30}
1x31+0x62+1x14+1x90+1x70+0x30 = 205
```

So, our cipher text is 121 197 205.

**2. Decryption :**
The receiver receive the cipher text which has to be decrypt. The receiver also knows the values of m and n. So, first we need to find the , which is multiplicative inverse of n mod m i.e.,

```
n x        mod(m) = 131 x       mod(110) = 1       = 71
```

Now, we have to multiply 71 with each block of cipher text take modulo m.

```
121 x 71 mod(110) = 11
```

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e.,
1+10=11 so make that corresponding bits 1 and others 0 which is 100100.
Similarly,

```
197 x 71 mod(110) = 17
1+2+4+10=17 = 111100

And, 205 x 71 mod(110) = 35
1+4+10+20=35 = 101110
```

After combining them we get the decoded text.
100100111100101110 which is our plain text.

# RSA in cryptography

**RSA algorithm** is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

**An example of asymmetric cryptography:**

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. The client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

**The idea!** The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long,

but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.

**Let us learn the mechanism behind the RSA algorithm : >> Generating Public Key:**

Select two prime no's. Suppose **P = 53 and Q = 59.**
**Now First part of the Public key : n = P*Q = 3127.**
We also need a small exponent say **e :**
**But e Must be**
**An integer.**
**Not be a factor of n.**
**1 < e < Φ(n) [Φ(n) is discussed below],**
**Let us now consider it to be equal to 3.**
Our Public Key is made of n and e

**>> Generating Private Key:**

We need to calculate Φ(n) :
Such that **Φ(n) = (P-1)(Q-1)**
**so, Φ(n) = 3016**
Now calculate Private Key, **d :**
**d = (k*Φ(n) + 1) / e for some integer k**
**For k = 2, value of d is 2011.**

Now we are ready with our – Public Key ( n = 3127 and e = 3) and Private Key(d = 2011) Now we will encrypt **"HI"** :

Convert letters to numbers : H = 8 and I = 9
Thus **Encrypted Data c = $89^{e} \bmod n$.**
**Thus our Encrypted Data comes out to be 1394**

Now we will decrypt **1394 :**
**Decrypted Data = $c^{d} \bmod n$.**
**Thus our Encrypted Data comes out to be 89**
**8 = H and I = 9 i.e. "HI".**

# Primarily Testing

A primality test is an algorithm to decide whether an input number is prime. Some primality tests are deterministic. They always correctly decide if a number is prime or composite.

The fastest known deterministic primality test was invented in 2004. There are three computer scientists, such as Agrawal, Kayal, and Saxena, invented the AKS primality test that operated in $O\tilde{}(\log(n)^6)$ time, where $O\tilde{}(f(n))$ is represented as $O(f(n).\log(f(n))^k)$ for some integer k [1]. Although a significant breakthrough, this speed is rather slow when compared to information security requirement.

The advantage of prime numbers are that they are utilized in cryptography. One of the standard cryptosystem -RSA algorithm need a prime number as key which is generally over 1024 bits to provide higher security.

When handling with such large numbers, definitely doesn't create the following method any good. It is not simply to work with such large numbers particularly when the operations performed are / and % at the time of primality testing.

Therefore, the best primality testing algorithms that are produced can only decide if the provided number is a "probable prime" or composite.

There are the following types of Primality Testing which are as follows −

- **Determininstic Algorithm** − A deterministic primality testing algorithm accepts an integer and continually output a prime or a composite. This algorithm always provide a proper answer.
- **Divisibility Algorithm** − The simplest primality test is as follows −

  Given an input number n, checks whether any integer from 2 to n -1 divides n. If n is divisible by any m, then n is composite otherwise it is a prime. However, instead of testing all m upto n – 1, it is only important to test m upto √n. If n is composite then it can be factored into two values, at least one of which should be less than or same to √n.

- **Probabilistic Algorithm** − A probabilistic algorithm provide an answer that is correct most of time, but not all of the time. These tests decided whether n satisfies one or more conditions that all primes should satisfy. A probabilistic algorithm restore either a prime or a composite depends on the following rules −
  - If the integer to be tested is actually a prime, the algorithm definitely return a prime.
  - If the integer to be tested is actually a composite, it returns a composite with probability 1 − ε, but it can return a prime with the probability ε. The probability of mistakes can be enhanced if it can run the algorithm 'm' times and the probability of error reduce to $\Sigma^m$.

**Fermat Primality Test** − The Fermat Primality Test stems from Fermat's Little Theorem, which states that if n is prime, $a^{n-1} \equiv 1 \pmod n$. Given an input n and a < n, it can check whether $a^{n-1} \equiv 1 \pmod n$. If this is not true, thus n is composite and n is probably prime. Unfortunately, the Fermat Primality Test has a high cost of error, with too several composites being probably prime.

# ElGamal Cryptosystem

ElGamal cryptosystem can be defined as **the cryptography algorithm that uses the public and private key concepts to secure communication between two systems**. It can be considered the asymmetric algorithm where the encryption and decryption happen by using public and private keys.

**ElGamal encryption** is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.
This cryptosystem is based on the difficulty of finding **discrete logarithm** in a cyclic group that is even if we know $g^a$ and $g^k$, it is extremely difficult to compute $g^{ak}$.
**Idea of ElGamal cryptosystem**
Suppose Alice wants to communicate with Bob.

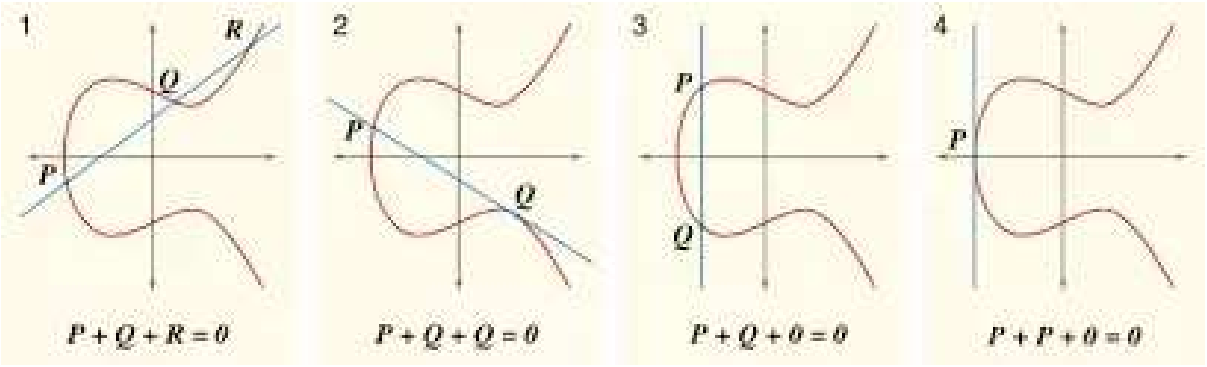
1. Bob generates public and private keys:
   - Bob chooses a very large number **q** and a cyclic group $\mathbf{F}_q$.
   - From the cyclic group $\mathbf{F}_q$, he choose any element **g** and an element **a** such that gcd(a, q) = 1.
   - Then he computes $h = g^a$.
   - Bob publishes **F**, $\mathbf{h = g^a}$, **q**, and **g** as his public key and retains **a** as private key.
2. Alice encrypts data using Bob's public key :
   - Alice selects an element **k** from cyclic group **F** such that gcd(k, q) = 1.
   - Then she computes $p = g^k$ and $s = h^k = g^{ak.}$
   - She multiples s with M.
   - Then she sends $(p, M*s) = (g^k, M*s)$.
3. Bob decrypts the message :
   - Bob calculates $s' = p^a = g^{ak}$.
   - He divides M*s by $s'$ to obtain M as $s = s'$.

# ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic curve cryptography is used to implement public key cryptography. It was discovered by Victor Miller of IBM and Neil Koblitz of the University of Washington in the year 1985. ECC popularly used an acronym for Elliptic Curve Cryptography. It is based on the latest mathematics and delivers a relatively more secure foundation than the first generation public key cryptography systems for example RSA.

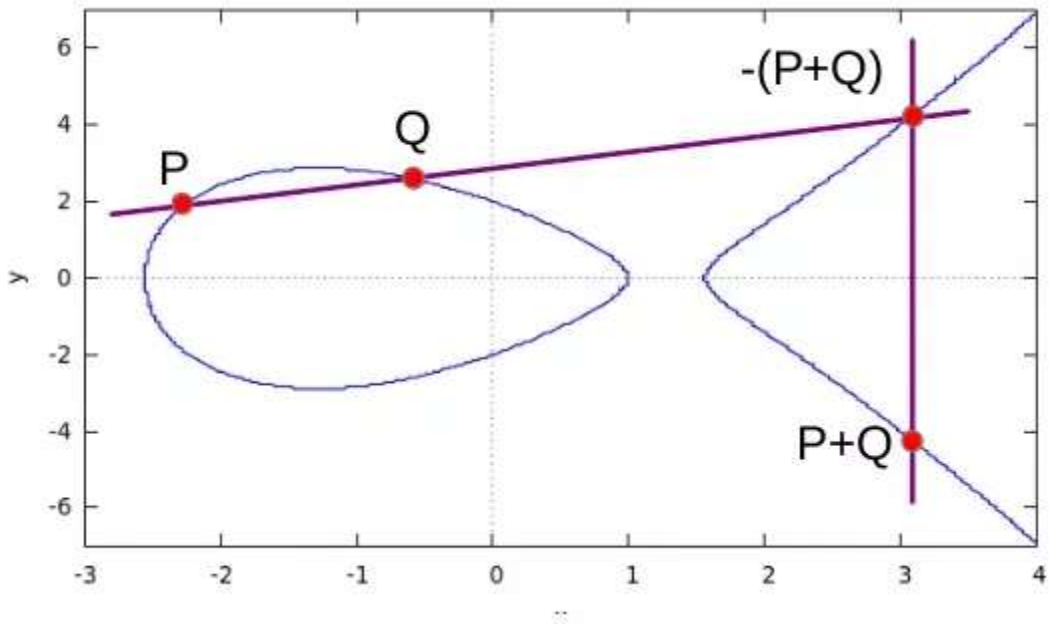
## Elliptic Curves

In 1985, cryptographic algorithms were proposed based on elliptic curves. An elliptic curve is the set of points that satisfy a specific mathematical equation. They are symmetrical.




## Uses

- Websites make extensive use of ECC to secure customers' hypertext transfer protocol connections.
- It is used for encryption by combining the key agreement with a symmetric encryption scheme.




- It is also used in several integer factorization algorithms like Lenstra elliptic-curve factorization.
- Time stamping uses an encryption model called a blind signature scheme. It is possible using Elliptic Curve Cryptography.

## ntroduction to the adoption curve

The adoption curve is a tool used by businesses to visualize and predict how customers will adopt a new product or service. The adoption curve has five stages− innovators, early adopters, early majority, late majority, and laggards. Each stage represents a different group of customers with different characteristics.

The adoption curve can be a helpful tool for businesses when launching a new product or service. By understanding the different stages of the adoption curve, businesses can tailor their marketing and sales efforts to target the right customers at the right time.

## The 5 stages of the adoption curve

It consists of five stages− Innovators, Early Adopters, Early Majority, Late Majority, and Laggards.

Innovators are the first people to adopt a new product or idea. They are risk−takers and are often seen as trendsetters. Early Adopters are the second group to adopt a new product or idea. They are more cautious than Innovators and often wait until a new product or idea has been proven to be successful before they adopt it.

Early Majority are the third group to adopt a new product or idea. They tend to be more skeptical than Innovators and Early Adopters and often wait until a new product or idea has been widely accepted before they adopt it. Late Majority are the fourth group to adopt a new product or idea. They are even more skeptical than the Early Majority and often wait until a new product or idea is proven to be very successful before they adopt it.

Laggards are the last group to adopt a new product or idea. They tend to be very set in their ways and are resistant to change. They may eventually come around to adopting a new product or idea, but it often takes them much longer than other groups.

## Product life cycle and the adoption curve

In order to understand the adoption curve, it is first important to understand the product life cycle. Every product has a beginning, a middle, and an end. The beginning is when the product is first introduced to the market. The middle is when the product gains popularity and becomes widely used. The end is when the product is no longer popular and sales begin to decline.

The adoption curve model was created in order to visualize how products are adopted by consumers over time. The model shows that there are four stages of adoption− innovators, early adopters, early majority, and late majority. Each stage represents a different group of consumers and their level of willingness to adopt a new product.

The innovators are the first group of people to adopt a new product. They are typically early adopters as well, but they also tend to be risk−takers who are willing to try new things. Early adopters are the second group of people to adopt a new product. They are usually trendsetters or opinion leaders within their social circle. Early majority adopters are typically more cautious than innovators or early adopters. They wait until a new product has been proven before they adopt it. Late majority adopters are even more cautious than early majority adopters. They wait until a new product is widely accepted before they adopt it.

## Why the adoption curve is important for businesses

Understanding where your target market falls on the adoption curve can help you tailor your marketing strategy accordingly and target your message to the right people. It can also help you predict how quickly your product or service will be adopted by the market.

The adoption curve is also a valuable tool for businesses when it comes to evaluating new products or services. By understanding where potential customers fall on the adoption curve, companies can better gauge whether a new offering is likely to be successful.

### How to use the adoption curve to your advantage

If you're trying to promote a new product or service, it's important to understand where your target audience falls on the adoption curve. This will help you tailor your marketing strategy to appeal to those who are most likely to be interested in what you have to offer.

For example, if you're targeting early adopters, you'll need to focus on creating a buzz around your product or service. This can be done through PR and social media campaigns. On the other hand, if you're targeting the late majority, you'll need to make sure your product or service is affordable and easy to use.

No matter what stage of the adoption curve your target audience is in, it's important to remember that not everyone will be ready to adopt your new product or service right away. It takes time for people to warm up to new ideas. By understanding the adoption curve, you can better manage your expectations and create a more effective marketing strategy.

# A generalization of the ElGamal public-key cryptosystem

The ElGamal cryptosystem is one of the most widely used public-key cryptosystems that depends on the difficulty of computing the discrete logarithms over finite fields. Over the years, the original system has been modified and altered in order to achieve a higher security and efficiency. In this paper, a generalization for the original ElGamal system is proposed which also relies on the discrete logarithm problem. The encryption process of the scheme is improved such that it depends on the prime factorization of the plaintext. Modular exponentiation is taken twice during the encryption; once with the number of distinct prime factors of the plaintext and then with the secret encryption key. If the plaintext consists of only one distinct prime factor, then the new method is similar to that of the basic ElGamal algorithm. The proposed system preserves the immunity against the Chosen Plaintext Attack (CPA).

# RABIN CRYPTOSYSTEM

**Rabin Cryptosystem** is an public-key cryptosystem invented by Michael Rabin. It uses asymmetric key encryption for communicating between two parties and encrypting the message.

The security of Rabin cryptosystem is related to the difficulty of factorization. It has the advantage over the others that the problem on which it banks has proved to be hard as **integer factorization**. It has the disadvantage also, that each output of the Rabin function can be generated by any of four possible inputs. if each output is a ciphertext, extra complexity is required on decryption to identify which of the four possible inputs was the true plaintext.

**Steps in Rabin cryptosystem**

**Key generation**

1. Generate two very large prime numbers, p and q, which satisfies the condition
   $p \neq q \rightarrow p \equiv q \equiv 3 \pmod 4$
   For example:
   p=139 and q=191
2. Calculate the value of n
   n = p.q
3. Publish n as public key and save p and q as private key

**Encryption**

1. Get the public key n.
2. Convert the message to ASCII value. Then convert it to binary and extend the binary value with itself, and change the binary value back to decimal m.
3. Encrypt with the formula:
   $C = m^2 \bmod n$
4. Send C to recipient.

**Decryption**

1. Accept C from sender.
2. Specify a and b with Extended Euclidean GCD such that, $a.p + b.q = 1$
3. Compute r and s using following formula:
   $r = C^{(p+1)/4} \bmod p$
   $s = C^{(q+1)/4} \bmod q$
4. Now, calculate X and Y using following formula:
   $X = ( a.p.r + b.q.s ) \bmod p$
   $Y = ( a.p.r – b.q.s ) \bmod q$
5. The four roots are, m1=X, m2=-X, m3=Y, m4=-Y
   Now, Convert them to binary and divide them all in half.
6. Determine in which the left and right half are same. Keep that binary's one half and convert it to decimal m. Get the ASCII character for the decimal value m. The resultant character gives the correct message sent by sender.

# Key management

In cryptography, it is a very tedious task to distribute the public and private keys between sender and receiver. If the key is known to the third party (forger/eavesdropper) then the whole security mechanism becomes worthless. So, there comes the need to secure the exchange of keys.

There are two aspects for Key Management:

1. Distribution of public keys.
2. Use of public-key encryption to distribute secrets.

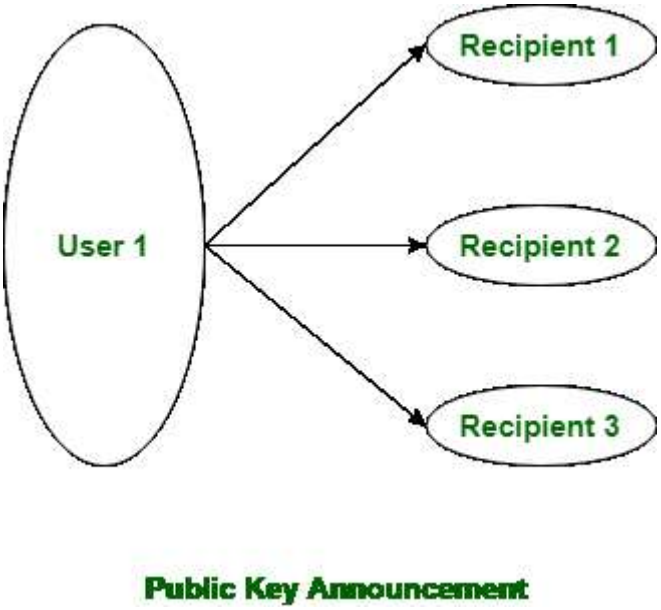
**Distribution of Public Key:**

The public key can be distributed in four ways:

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates.

These are explained as following below:

**1. Public Announcement:** Here the public key is broadcasted to everyone. The major weakness of this method is a forgery. Anyone can create a key claiming to be someone else and broadcast it. Until forgery is discovered can masquerade as claimed user.




**2. Publicly Available Directory:** In this type, the public key is stored in a public directory. Directories are trusted here, with properties like Participant Registration, access and allow to modify values at any time, contains entries like {name, public-key}. Directories can be accessed electronically still vulnerable to forgery or tampering.

**3. Public Key Authority:** It is similar to the directory but, improves security by tightening control over the distribution of keys from the directory. It requires users to know the public key for the directory. Whenever the keys are needed, real-time access to the directory is made by the user to obtain any desired public key securely.

**4. Public Certification:** This time authority provides a certificate (which binds an identity to the public key) to allow key exchange without real-time access to the public authority each time. The certificate is accompanied by some other info such as period of validity, rights of use, etc. All of this content is signed by the private key of the certificate authority and it can be verified by anyone possessing the authority's public key.
First sender and receiver both request CA for a certificate which contains a public key and other information and then they can exchange these certificates and can start communication.

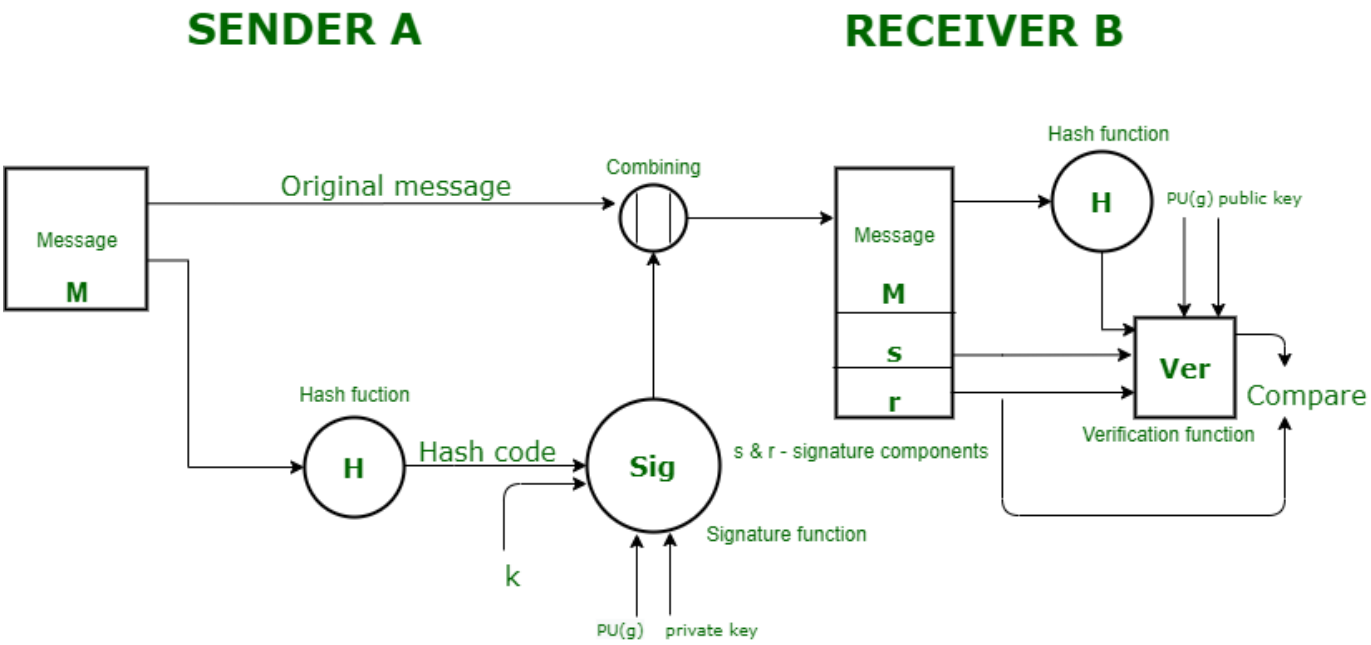
## KEY EXCHANGE

The key exchange method **specifies how one-time session keys are generated for encryption and for authentication, and how the server authentication is done**. The Diffie-Hellman Key Exchange is a method for exchanging secret keys over a non-secure medium without exposing the keys.

## Digital Signature Standard (DSS)

As we have studied, signature is a way of authenticating the data coming from a trusted individual. Similarly, digital signature is a way of authenticating a digital data coming from a trusted source.

**Digital Signature Standard (DSS)** is a Federal Information Processing Standard(FIPS) which defines algorithms that are used to generate digital signatures with the help of Secure Hash Algorithm(SHA) for the authentication of electronic documents. DSS only provides us with the digital signature function and not with any encryption or key exchanging strategies.




**Sender Side :**
In DSS Approach, a hash code is generated out of the message and following inputs are given to the signature function –

1. The hash code.
2. The random number 'k' generated for that particular signature.
3. The private key of the sender i.e., PR(a).
4. A global public key(which is a set of parameters for the communicating principles) i.e., PU(g).

These input to the function will provide us with the output signature containing two components – 's' and 'r'. Therefore, the original message concatenated with the signature is sent to the receiver.

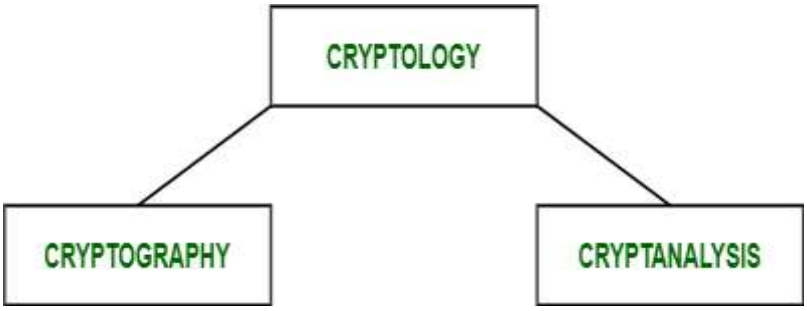
**Receiver Side :**
At the receiver end, verification of the sender is done. The hash code of the sent message is generated. There is a verification function which takes the following inputs –

1. The hash code generated by the receiver.
2. Signature components 's' and 'r'.
3. Public key of the sender.
4. Global public key.

The output of the verification function is compared with the signature component 'r'. Both the values will match if the sent signature is valid because only the sender with the help of it private key can generate a valid signature

## Cryptanalysis and Types of Attacks

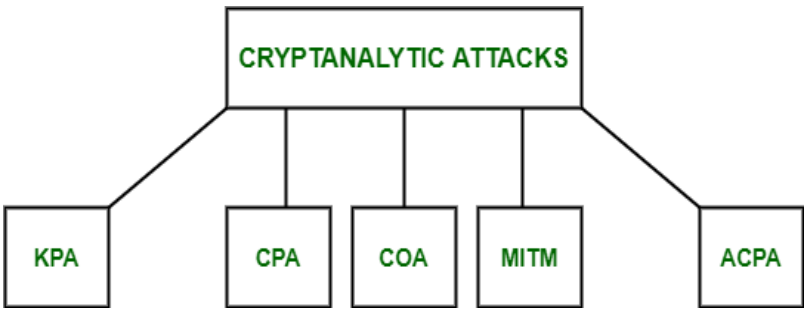
**Cryptology** has two parts namely, **Cryptography** which focuses on creating secret codes and **Cryptanalysis** which is the study of the cryptographic algorithm and the breaking of those secret codes. The person practicing Cryptanalysis is called a **Cryptanalyst**. It helps us to better understand the cryptosystems and also helps us improve the system by finding any weak point and thus work on the algorithm to create a more secure secret code. For example, a Cryptanalyst might try to decipher a ciphertext to derive the plaintext. It can help us to deduce the plaintext or the encryption key.




Parts Of Cryptology

To determine the weak points of a cryptographic system, it is important to attack the system. This attacks are called **Cryptanalytic attacks.** The attacks rely on nature of the algorithm and also knowledge of the general characteristics of the plaintext, i.e., plaintext can be a regular document written in English or it can be a code written in Java. Therefore, nature of the plaintext should be known before trying to use the attacks.

**Types of Cryptanalytic attacks :**



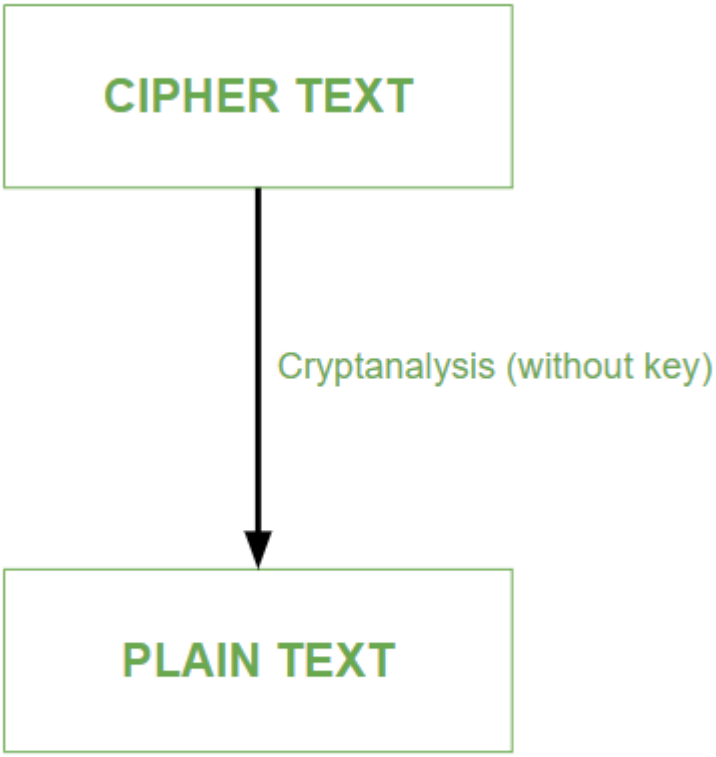

The Five Types of Cryptanalytic Attacks

- **Known-Plaintext Analysis (KPA) :**
  In this type of attack, some plaintext-ciphertext pairs are already known. Attacker maps them in order to find the encryption key. This attack is easier to use as a lot of information is already available.
- **Chosen-Plaintext Analysis (CPA) :**
  In this type of attack, the attacker chooses random plaintexts and obtains the corresponding ciphertexts and tries to find the encryption key. Its very simple to implement like KPA but the success rate is quite low.
- **Ciphertext-Only Analysis (COA) :**
  In this type of attack, only some cipher-text is known and the attacker tries to find the corresponding encryption key and plaintext. Its the hardest to implement but is the most probable attack as only ciphertext is required.
- **Man-In-The-Middle (MITM) attack :**
  In this type of attack, attacker intercepts the message/key between two communicating parties through a secured channel.
- **Adaptive Chosen-Plaintext Analysis (ACPA) :**
  This attack is similar CPA. Here, the attacker requests the cipher texts of additional plaintexts after they have ciphertexts for some texts.

# 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. |
| **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. |

| S.NO | Block Cipher | Stream Cipher |
|---|---|---|
| 1. | Block Cipher Converts the plain text into cipher text by taking plain text's block at a time. | Stream Cipher Converts the plain text into cipher text by taking 1 byte of plain text at a time. |
| 2. | Block cipher uses either 64 bits or more than 64 bits. | While stream cipher uses 8 bits. |

| | | |
|---|---|---|
| 3. | The complexity of block cipher is simple. | While stream cipher is more complex. |
| 4. | Block cipher Uses confusion as well as diffusion. | While stream cipher uses only confusion. |
| 5. | In block cipher, reverse encrypted text is hard. | While in-stream cipher, reverse encrypted text is easy. |
| 6. | The algorithm modes which are used in block cipher are ECB (Electronic Code Book) and CBC (Cipher Block Chaining). | The algorithm modes which are used in stream cipher are CFB (Cipher Feedback) and OFB (Output Feedback). |
| 7. | Block cipher works on transposition techniques like rail-fence technique, columnar transposition technique, etc. | While stream cipher works on substitution techniques like Caesar cipher, polygram substitution cipher, etc. |
| 8. | Block cipher is slow as compared to a stream cipher. | While stream cipher is fast in comparison to block cipher. |

# Shamir's Secret Sharing

(SSS) is a key distribution algorithm. It is named for the well-known Israeli cryptographer Adi Shamir who co-invented the Rivest–Shamir–Adleman (RSA) algorithm.

SSS divides a secret, such as a cryptokey, into parts called shares. The shares are distributed to a group of people who are parties to the conversation. The parts of the secret are brought together to reconstruct the secret, but an important feature of Shamir's Secret Sharing is that the total number of shares is not needed to reconstruct the secret. A number less than the total number, called the threshold, is required. This helps avoid failures in decrypting the closely-held information should just one or a few parties be unavailable.

SSS is practical in its solution to the key-sharing problems many arrangements face, and is therefore usually used to secure the keys to something that is encrypted or secure using other tools or algorithms. A simple illustration of SSS is that of a vault that only a corporate board may access. The passcode is encrypted by SSS, so a quorum (threshold) of board members is needed to authorize the display or release of the vault passcode. If a board member is traveling, but the threshold is met, SSS still allows for a reasonable assurance that the vault is secure.

### Example:

"SSS is provides a solution to the challenge of managing encryption keys. It solves the issue of all parties being necessary to access sensitive information but is still secure because a quorum of responsible parties must agree to a user's access."

# Identity-based encryption

**(IBE)** *is a form of public-key cryptography in which a third-party server uses a simple identifier, such as an e-mail address, to generate a public key that can be used for encrypting and decrypting electronic messages. Compared with typical public-key cryptography, this greatly reduces the complexity of the encryption process for both users and administrators. An added advantage is that a message recipient doesn't need advance preparation or specialized software to read the communication.*

### A Solution

One promising solution to these difficulties is called identity-based cryptography or identity-based encryption (IBE). In this process, which can be initiated by the sender, a unique identifier of the recipient (such as his e-mail address) is used to calculate a public key. A trusted third-party server, called the private-key generator, uses a cryptographic algorithm to calculate the corresponding private key from the public key. In this way, recipients can generate their own private keys directly from the server as needed, and they don't have to worry about distributing their public keys.

### How IBE Works

The success of IBE depends upon the third-party IBE server that generates private keys. The only information this server stores permanently is a secret master key -- a large random number that is exclusive to the security domain. The server uses this key to create a common set of public-key parameters (including the server's address) that are given to each user who installs the IBE software, and recipients' private keys as required.

### Pros and Cons of IBE

#### Pros

- No certificates needed. A recipient's public key is derived from his identity.
- No pre-enrollment required.
- Keys expire, so they don't need to be revoked. In a traditional public-key system, keys must be revoked if compromised.
- Less vulnerable to spam.
- Enables postdating of messages for future decryption.
- Enables automatic expiration, rendering messages unreadable after a certain date.

#### Cons

- Requires a centralized server. IBE's centralized approach implies that some keys must be created and held in escrow -- and are therefore at greater risk of disclosure.
- Requires a secure channel between a sender or recipient and the IBE server for transmitting the private key.

When a sender creates an encrypted message, the IBE software on his system uses three parameters to generate the public key for the message: a starting value, the current week number and the recipient's identity (normally the e-mail address). Because a calendar reference is included, the public key that is generated will automatically expire.

A user who receives an IBE-encrypted e-mail message but has not used the process before can request -- upon authentication -- a private key that allows him to decrypt all e-mails encrypted using his e-mail address as the public key.

## Attribute-based encryption

**ABE** is a generalisation of public-key encryption which enables fine grained access control of encrypted data using authorisation policies. The secret key of a user and the ciphertext are dependent upon attributes (e.g. their email address, the country in which they live, or the kind of subscription they have). In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext.[1]

A crucial security aspect of attribute-based encryption is collusion-resistance: An adversary that holds multiple keys should only be able to access data if at least one individual key grants access.

### Usage

Attribute-based encryption (ABE) can be used for log encryption.[14] Instead of encrypting each part of a log with the keys of all recipients, it is possible to encrypt the log only with attributes which match recipients' attributes. This primitive can also be used for broadcast encryption in order to decrease the number of keys used.[15] Attribute-based encryption methods are also widely employed in vector-driven search engine interfaces.[16]

## Challenges

Although the ABE concept is very powerful and a promising mechanism, ABE systems suffer mainly from two drawbacks: inefficiency and the lack of a straightforward attribute revocation mechanism.

Other main challenges are:

- Key coordination
- Key escrow
- Key revocation

## What is a side-channel attack?


A side-channel attack is a security exploit that aims to gather information from or influence the program execution of a system by measuring or exploiting indirect effects of the system or its hardware -- rather than targeting the program or its code directly. Most commonly, these attacks aim to exfiltrate sensitive information, including cryptographic keys, by measuring coincidental hardware emissions. A side-channel attack may also be referred to as a *sidebar attack* or an *implementation attack*.

As an illustration, imagine you're trying to determine where a person has driven their car. A typical attack channel would be to follow the car or use a Global Positioning System (GPS) tracker. A side-channel attack, on the other hand, would use measurements about the car to try and determine how it's used. For example, measuring changes in the amount of gas in the tank, car's weight, heat of the engine or passenger compartment, tire wear, paint scratches and the like may reveal information about the use of the car, places or distances it has traveled, or what is stored in the trunk -- all done without directly affecting the car or alerting its owner that they are under investigation.

Historically difficult to do, side-channel attacks are now more common because of several factors. Increasing sensitivity of measuring equipment has made it possible to gather extremely detailed data about a system while it is running. In addition, greater computing power and machine learning enable attackers to better understand the raw data they extract. This deeper understanding of targeted systems enables attackers to better exploit subtle changes in a system.

Attackers can also go after high-value targets, such as secure processors, Trusted Platform Module (TPM) chips and cryptographic keys. Even having only partial information can assist a traditional attack vector, such as a brute-force attack, to have a greater chance of success.

Side-channel attacks can be tricky to defend against. They are difficult to detect in action, often do not leave any trace and may not alter a system while it's running. Side-channel attacks can even prove effective against air-gapped systems that have been physically segregated from other computers or networks. Additionally, they may also be used against virtual machines (VMs) and in cloud computing environments where an attacker and target share the same physical hardware.

Still common, the targeting of electromagnetic radiation given off by devices is the earliest known type of side-channel attack.

## Types of side-channel attacks

Bad actors can implement side-channel attacks in several different ways, including the following.

### *Electromagnetic*

An attacker measures the electromagnetic radiation, or radio waves, given off by a target device to reconstruct the internal signals of that device. The earliest side-channel attacks were electromagnetic. van Eck phreaking and the National Security Agency's (NSA) Tempest system could reconstruct the entirety of a computer's screen. Attackers focus modern side-channel attacks on measuring the cryptographic operations of a system to try and derive secret keys. Software-defined radio (SDR) devices have lowered the barrier of entry for electromagnetic attackers, which can be performed through walls and without any contact with the target device.

### *Acoustic*

The attacker measures the sounds produced by a device. Proof-of-concept (POC) attacks have been performed that can reconstruct a user's keystrokes from an audio recording of the user typing. Hackers can obtain some information by listening to the sounds emitted by electronic components as well.

### *Power*

A hacker measures or influences the power consumption of a device or subsystem. By monitoring the amount and timing of power used by a system or one of its subcomponents, an attacker can infer activity of that system. Some attacks may cut or lower power to cause a system to behave in a way beneficial to the attacker, similar to Plundervolt attacks.

### *Optical*

An attacker uses visual cues to gain information about a system. Although rarely used against computers, some POC attacks have been performed where audio can be reconstructed from a video recording of an object vibrating in relation to sounds. Simple shoulder surfing attacks may also fall into this category.

### *Timing*

A bad actor uses the length of time an operation takes to gain information. The total time can provide data about the state of a system or the type of process it is running. Here, the attacker can compare the length of time of a known system to the victim system to make accurate predictions.

### *Memory cache*

An attacker abuses memory caching to gain additional access. Modern systems use data caching and pre-fetching to improve performance. An attacker can abuse these systems to access information that should be blocked. The Spectre and Meltdown vulnerabilities that primarily affected Intel processors exploited this channel.

### *Hardware weaknesses*

Hackers can use physical characteristics of a system to induce a behavior, cause a fault or exploit data remanence, which is data that persists after deletion. Row hammering attacks happen when an attacker causes a change in a restricted area of memory by quickly flipping, or hammering, another area of memory located close by on the physical random access memory (RAM) chip. Error correction code (ECC) memory can help prevent this attack. In a cold boot attack, the attacker quickly lowers the temperature of RAM, causing some of the information to be retained after power is removed so the attacker can read it back.

## How to prevent a side-channel attack

Organizations can implement a few best practice mitigations that may help protect against side-channel attacks. These attacks usually require specific detailed knowledge of a system to execute; therefore, a business should keep details related to implementation and vendors as a trade secret.

Address space layout randomization (ASLR) can prevent some memory- or cache-based attacks. Using business-grade equipment can also help to prevent systems from being exploited. Physical access to systems should be restricted as well. Businesses can also keep sensitive systems in shielded Faraday cages, and power conditioning equipment can shield against power attacks.

As extreme mitigations, increasing the amount of noise in a system will make it more difficult for an attacker to gain useful information. Furthermore, while the following ideas are often wasteful and not generally recommended, they may be useful in specific circumstances.

First, while performing a cryptographic process, some systems will simultaneously perform many unrelated and worthless similar processes to camouflage which ones the attacker may be interested in. A system may likewise run needless processes or components to hide the power or computational use so it is uncorrelated with the actual use. Turning on an additional source of electromagnetic field radiation may hide radio signals from attackers as well, and users can set the font color and text background to a similar color to make reconstructing it hard via van Eck phreaking or casual shoulder surfing.

# PGP

**(Pretty Good Privacy)**, is a popular program that is used to provide confidentiality and authentication services for electronic mail and file storage. It was designed by **Phil Zimmermann** way back in 1991. He designed it in such a way, that the best cryptographic algorithms such as RSA, Diffie-Hellman key exchange, DSS are used for the public-key encryption (or) asymmetric encryption; CAST-128, 3DES, IDEA are used for symmetric encryption and SHA-1 is used for hashing purposes. PGP software is an open source one and is not dependent on either the OS (Operating System) or the processor. The application is based on a few commands which are very easy to use.

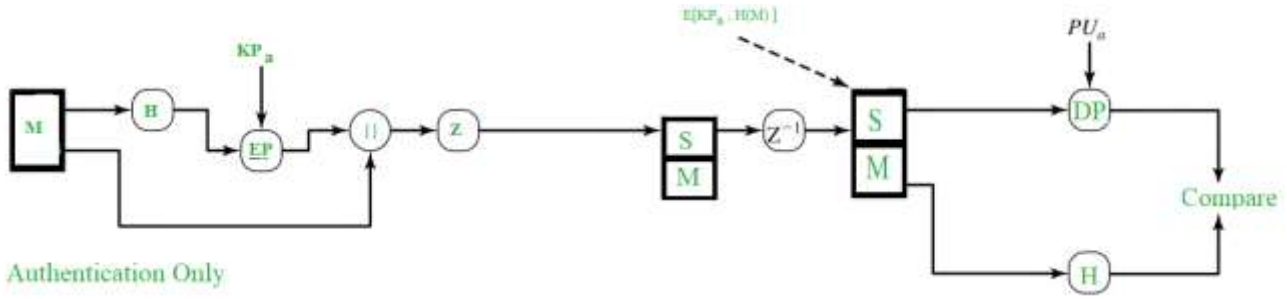
The following are the services offered by PGP:

1. Authentication
2. Confidentiality
3. Compression
4. Email Compatibility
5. Segmentation

In this article, we will see about Authentication and Confidentiality.

**1. Authentication:**
Authentication basically means something that is used to validate something as true or real. To login into some sites sometimes we give our account name and password, that is an authentication verification procedure.

In the email world, checking the authenticity of an email is nothing but to check *whether it actually came from the person it says*. In emails, authentication has to be checked as there are some people who spoof the emails or some spams and sometimes it can cause a lot of inconvenience. The Authentication service in PGP is provided as follows:



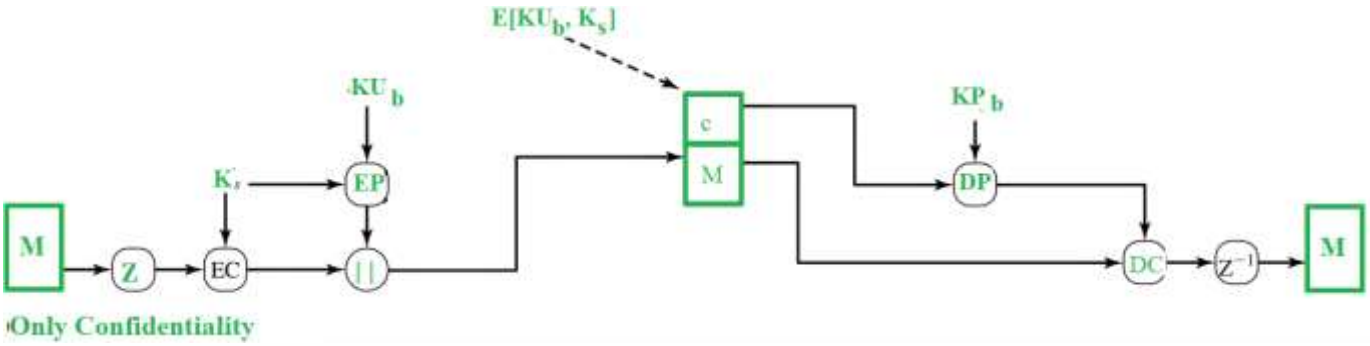

As shown in the above figure, the Hash Function (H) calculates the Hash Value of the message. For the hashing purpose, **SHA-1** is used and it produces a **160 bit** output hash value. Then, using the sender's private key ($KP_a$), it is encrypted and it's called as **Digital Signature**. The Message is then appended to the signature. All the process happened till now, is sometimes described as *signing the message* . Then the message is compressed to reduce the transmission overhead and is sent over to the receiver.

At the receiver's end, the data is decompressed and the message, signature are obtained. The signature is then decrypted using the sender's public key($PU_a$) and the hash value is obtained. The message is again passed to hash function and it's hash value is calculated and obtained.

Both the values, one from signature and another from the recent output of hash function are compared and if both are same, it means that the email is actually sent from a known one and is legit, else it means that it's not a legit one.

**2. Confidentiality:**
Sometimes we see some packages labelled as 'Confidential', which means that those packages are not meant for all the people and only selected persons can see them. The same applies to the email confidentiality as well. Here, in the email service, only the sender and the receiver should be able to read the message, that means the contents have to be kept secret from every other person, except for those two.

PGP provides that Confidentiality service in the following manner:



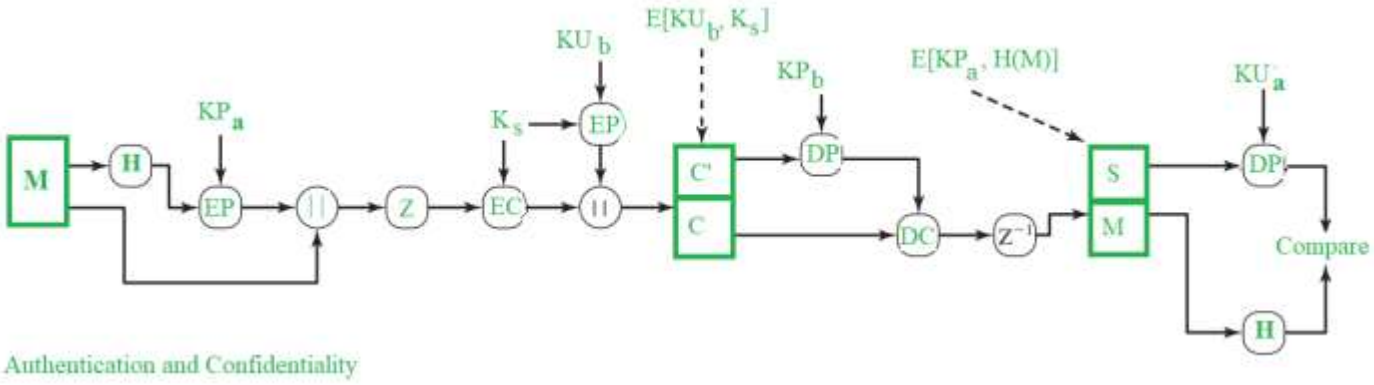

The message is first compressed and a 128 bit session key ($K_s$), generated by the PGP, is used to encrypt the message through symmetric encryption. Then, the session key ($K_s$) itself gets encrypted through public key encryption (EP) using receiver's public key($KU_b$) . Both the encrypted entities are now concatenated and sent to the receiver.

As you can see, the original message was compressed and then encrypted initially and hence even if any one could get hold of the traffic, he cannot read the contents as they are not in readable form and they can only read them if they had the session key ($K_s$). Even though session key is transmitted to the receiver and hence, is in the traffic, it is in encrypted form and only the receiver's private key ($KP_b$)can be used to decrypt that and thus our message would be completely safe.

At the receiver's end, the encrypted session key is decrypted using receiver's private key ($KP_b$) and the message is decrypted with the obtained session key. Then, the message is decompressed to obtain the original message (M).

RSA algorithm is used for the public-key encryption and for the symmetric key encryption, CAST-128(or IDEA or 3DES) is used.

Practically, **both** the Authentication and Confidentiality services are provided in parallel as follows :




Authentication and Confidentiality

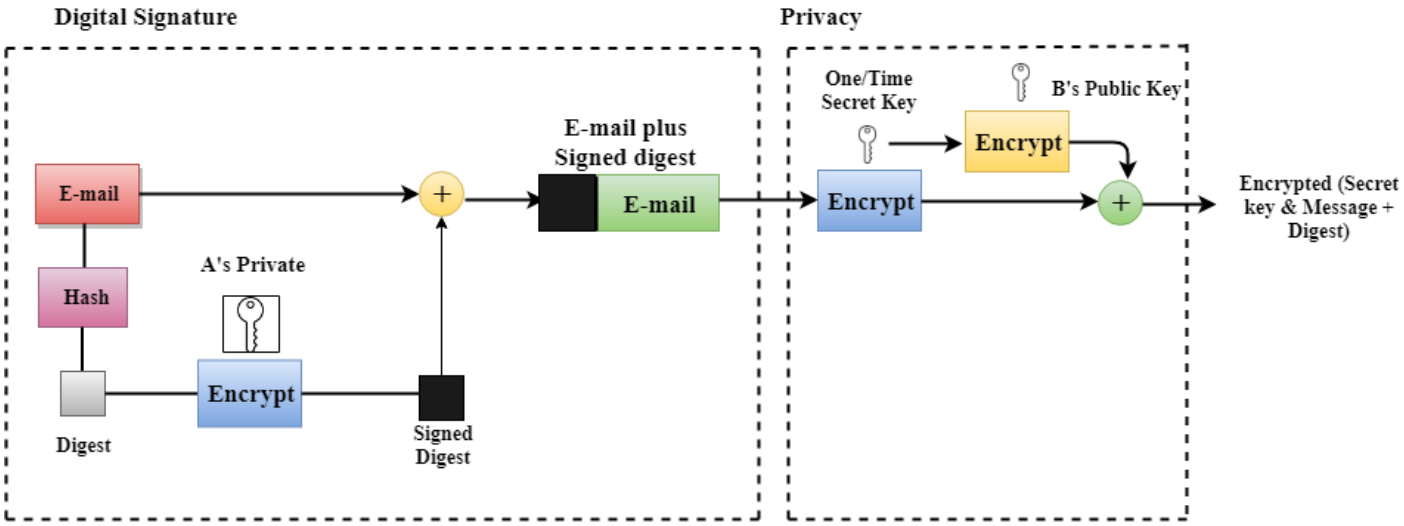
# PGP

- PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
- PGP was designed to provide all four aspects of security, i.e., privacy, integrity, authentication, and non-repudiation in the sending of email.
- PGP uses a digital signature (a combination of hashing and public key encryption) to provide integrity, authentication, and non-repudiation. PGP uses a combination of secret key encryption and public key encryption to provide privacy. Therefore, we can say that the digital signature uses one hash function, one secret key, and two private-public key pairs.
- PGP is an open source and freely available software package for email security.
- PGP provides authentication through the use of Digital Signature.
- It provides confidentiality through the use of symmetric block encryption.
- It provides compression by using the ZIP algorithm, and EMAIL compatibility using the radix-64 encoding scheme.

## Following are the steps taken by PGP to create secure e-mail at the sender site:

- The e-mail message is hashed by using a hashing function to create a digest.
- The digest is then encrypted to form a signed digest by using the sender's private key, and then signed digest is added to the original email message.
- The original message and signed digest are encrypted by using a one-time secret key created by the sender.
- The secret key is encrypted by using a receiver's public key.
- Both the encrypted secret key and the encrypted combination of message and digest are sent together.

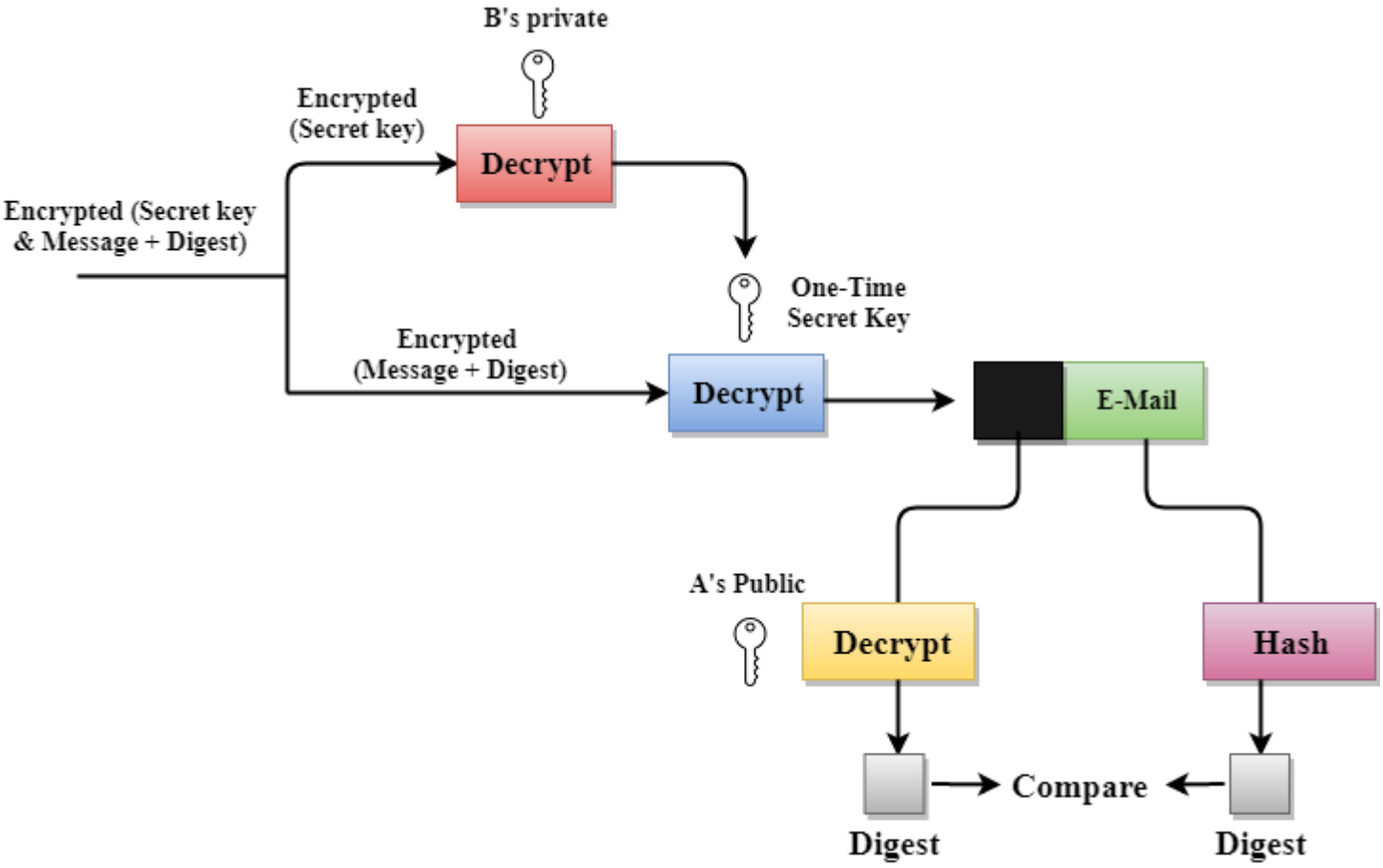
## PGP at the Sender site (A)




## Following are the steps taken to show how PGP uses hashing and a combination of three keys to generate the original message:

- The receiver receives the combination of encrypted secret key and message digest is received.
- The encrypted secret key is decrypted by using the receiver's private key to get the one-time secret key.
- The secret key is then used to decrypt the combination of message and digest.
- The digest is decrypted by using the sender's public key, and the original message is hashed by using a hash function to create a digest.
- Both the digests are compared if both of them are equal means that all the aspects of security are preserved.

## PGP at the Receiver site (B)



Disadvantages of PGP Encryption

- **The Administration is difficult:** The different versions of PGP complicate the administration.
- **Compatibility issues:** Both the sender and the receiver must have compatible versions of PGP. For example, if you encrypt an email by using PGP with one of the encryption technique, the receiver has a different version of PGP which cannot read the data.
- **Complexity:** PGP is a complex technique. Other security schemes use symmetric encryption that uses one key or asymmetric encryption that uses two different keys. PGP uses a hybrid approach that implements symmetric encryption with two keys. PGP is more complex, and it is less familiar than the traditional symmetric or asymmetric methods.
- **No Recovery:** Computer administrators face the problems of losing their passwords. In such situations, an administrator should use a special program to retrieve passwords. For example, a technician has physical access to a PC which can be used to retrieve a password. However, PGP does not offer such a special program for recovery; encryption methods are very strong so, it does not retrieve the forgotten passwords results in lost messages or lost files.

# Quantum Cryptography

Quantum cryptography is **a method of encryption that uses the naturally occurring properties of quantum mechanics to secure and transmit data in a way that cannot be hacked**. Cryptography is the process of encrypting and protecting data so that only the person who has the right secret key can decrypt it.

The **uncertainty principle** of quantum physics builds the earliest foundations for quantum cryptography. With quantum computers of the future being expected to solve discrete logarithmic problems and the popularly known cryptography methods such as AES, RSA, DES, quantum cryptography becomes the foreseen solution. In practice, it is used to establish a shared, secret and random sequence of bits to communicate between two systems let's say, Alice and Bob. This is known as **Quantum Key Distribution**. After this key is shared between Alice and Bob, further exchange of information can take place through known cryptographic strategies.

**Based On Heisenberg's Uncertainty Principle:**
**BB84 and variants –**
A single-photon pulse is passed through a polarizer. Alice can use a particular polarizer to polarize a single-photon pulse and encode binary value bits to the outcome of a particular type (vertical, horizontal, circular, etc) of a polarizer. On receiving the photon beam, Bob would guess the polarizer, and Bob can thus match the cases with Alice and know the correctness of his guesses. If Eve would have been trying to decode then due to polarization by Eve's polarizer would have caused discrepancies in match cases of Bob and Alice and thus they would know about eavesdropping. Thus in such a system if Eve tries to eavesdrop it will get to the notice of Alice and Bob.

- The B92 protocol has only two polarization states unlike four in the original BB84.
- BB84 has a similar protocol SSP that uses 6 states to encode the bits.
- SARG04 is another protocol that uses attenuated lasers and provides better results than BB84 in more than one photon system.

**Based On Quantum Entanglement:**
**E91 and Variants –**
There is a single source that emits a pair of entangled photons with Alice and Bob receiving each particle. Similar to the BB84 scheme Alice and Bob would exchange encoded bits and match cases for each photon transferred. But in this scenario, the outcome of the results of the match cases of Alice and Bob will be the opposite as a consequence of the Entanglement principle. Either of them will have complement bits in bit strings interpreted. One of them can then invert bits to agree upon a key. Since Bell's Inequality should not hold for entangled particles thus this test can confirm the absence of eavesdroppers. Since practically it is

not possible to have a third photon in entanglement with energy levels sufficient for nondetect ability, thus this system is fully secure.

- SARG04 and SSP protocol models can be extended to Entangled particles theory.

**Possible Attacks In Quantum Cryptography:**

- **Photon Number Splitting (PNS) Attack –**
  Since it is not possible to send a single photon thus a pulse is sent. Some of the photons from a pulse can be captured by Eve and after matching of bits by Alice and Bob, Eve can use the same polarizer as done by Bob and thus get the key without being detected.

- **Faked-State Attack –**
  Eve uses a replica of Bob's photon detector and thus captures the photons intended for Bob and further passed it to Bob. Though Eve knows about the encoded bit, Bob thinks that he received it from Alice.

| Quantum Cryptography | Post-Quantum Cryptography |
|---|---|
| Quantum cryptography, also known as Quantum Encryption or Quantum Security, is a term that explains how the laws of Quantum mechanics can be applied in cryptography. | Post-Quantum Cryptography is a set of techniques (typically public-key algorithms) that are expected to be secure against a quantum computer assault. |
| A quantum channel cannot be properly intercepted without detection, according to quantum mechanics. | Algorithms will be studied in order to evaluate their trustworthiness, but there is no assurance that someone will ultimately find a method to break them. |
| Implementation will necessitate the use of specialized hardware. | The majority of implementations will be software-only and will not necessitate the use of specialized hardware. |
| Only works with optical communications over a free space optical fiber. | It works with all types of digital communications mediums, including RF wireless networks and optical communications. |
| Higher costs due to the need for new hardware and communications infrastructure. | Software-based synthesis solutions will be relatively low-cost. |
| Receiving a Quantum channel, decoding to classical bits, and re-encrypting and broadcasting to another Quantum channel makes a repeater conceivable. | Compliant with today's digital repeater technology. |
| The number of options is really limited. Only line-of-sight nodes should be utilized. | Compatible with any form of mobile device communications. |
| It's possible that it may be utilized for digital signatures, but it's improbable. | Different versions of the standards are being developed expressly for the use of digital signatures. |

# Cryptography in Blockchain

One of the important questions that always comes to our mind is How blockchain is secure? and What makes blockchain secure? Blockchain security is built on two concepts Cryptography and Hashing. This article focuses on discussing these two important concepts in detail.

## Cryptography in Blockchain

Cryptography is a method of securing data from unauthorized access. In the blockchain, cryptography is used to secure transactions taking place between two nodes in a blockchain network. As discussed above, in a

blockchain there are two main concepts cryptography and hashing. Cryptography is used to encrypt messages in a P2P network and hashing is used to secure the block information and the link blocks in a blockchain.

Cryptography primarily focuses on ensuring the security of participants, transactions, and safeguards against double-spending. It helps in securing different transactions on the blockchain network. It ensures that only the individuals for whom the transaction data is intended can obtain, read and process the transaction.

## Role of Cryptography in Blockchain

Blockchain is developed with a range of different cryptography concepts. The development of cryptography technology promotes restrictions for the further development of blockchain.

- In the blockchain, cryptography is mainly used to protect user privacy and transaction information and ensure data consistency.
- The core technologies of cryptography include symmetric encryption and asymmetric encryption.
- Asymmetric cryptography uses digital signatures for verification purposes, every transaction recorded to the block is signed by the sender by digital signature and ensures that the data is not corrupted.

Cryptography plays a key role in keeping the public network secure, so making it fit to maintain the integrity and security of blockchain.

## Cryptography

Cryptography is a technique or a set of protocols that secure information from any third party during a process of communication. It is also made up of two Greek terms, Kryptos term meaning “hidden” and Graphein, a term meaning “to write”. Some terminologies related to Cryptography:

- **Encryption:** Conversion of normal text to a random sequence of bits.
- **Key:** Some amount of information is required to get the information of the cryptographic algorithm.
- **Decryption:** The inverse process of encryption, conversion of a Random sequence of bits to plaintext.
- **Cipher:** The mathematical function, i.e. a cryptographic algorithm which is used to convert plaintext to ciphertext(Random sequence of bits).

## Types of Cryptography

The two types of cryptography are:

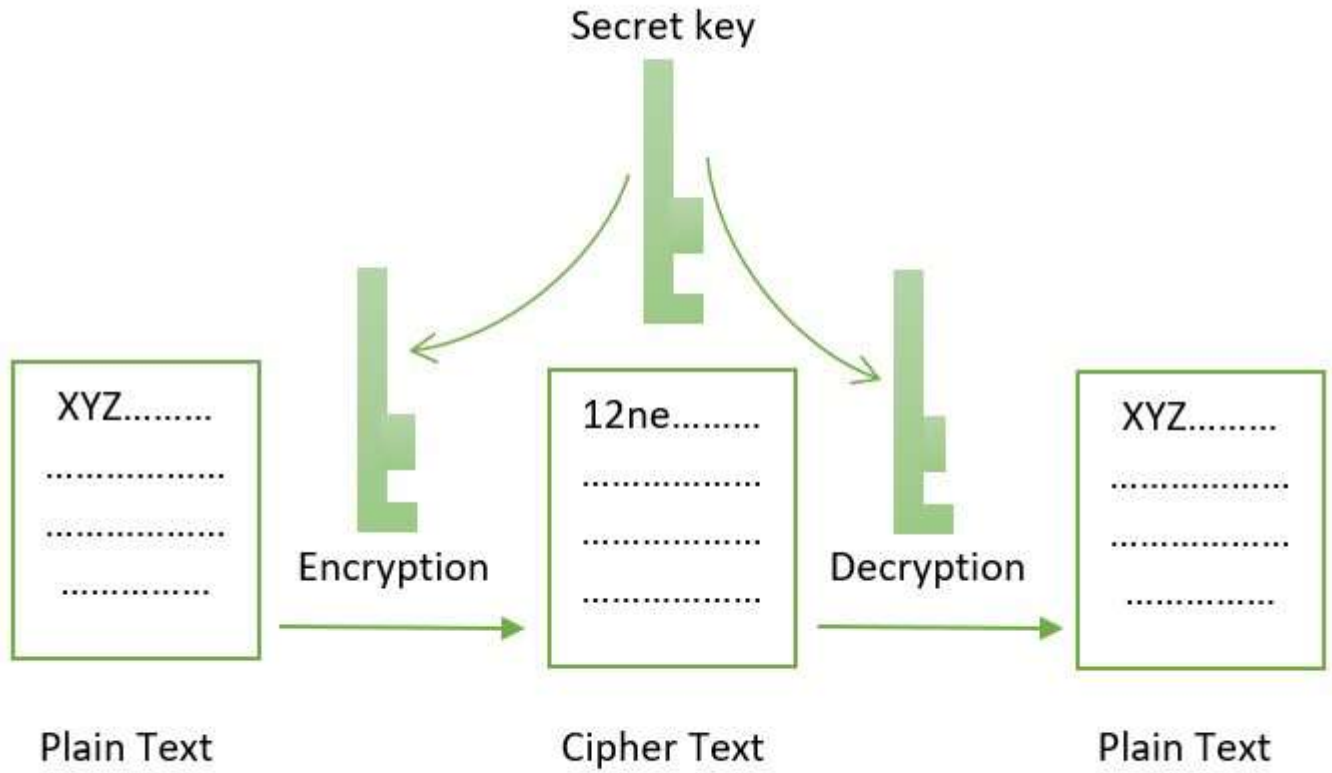
- **Symmetric-key cryptography.**
- **Asymmetric-key cryptography.**

Let’s discuss each of these topics in detail.

**1. Symmetric-key Encryption:** It focuses on a similar key for encryption as well as decryption. Most importantly, the symmetric key encryption method is also applicable to secure website connections or encryption of data. It is also referred to as secret-key cryptography. The only problem is that the sender and receiver exchange keys in a secure manner. The popular symmetric-key cryptography system is Data Encryption System(DES). The cryptographic algorithm utilizes the key in a cipher to encrypt the data and the data must be accessed. A person entrusted with the secret key can decrypt the data. Examples: AES, DES, etc.

**Features:**

- It is also known as Secret key cryptography.
- Both parties have the same key to keeping secrets.
- It is suited for bulk encryptions.
- It requires less computational power and faster transfer.

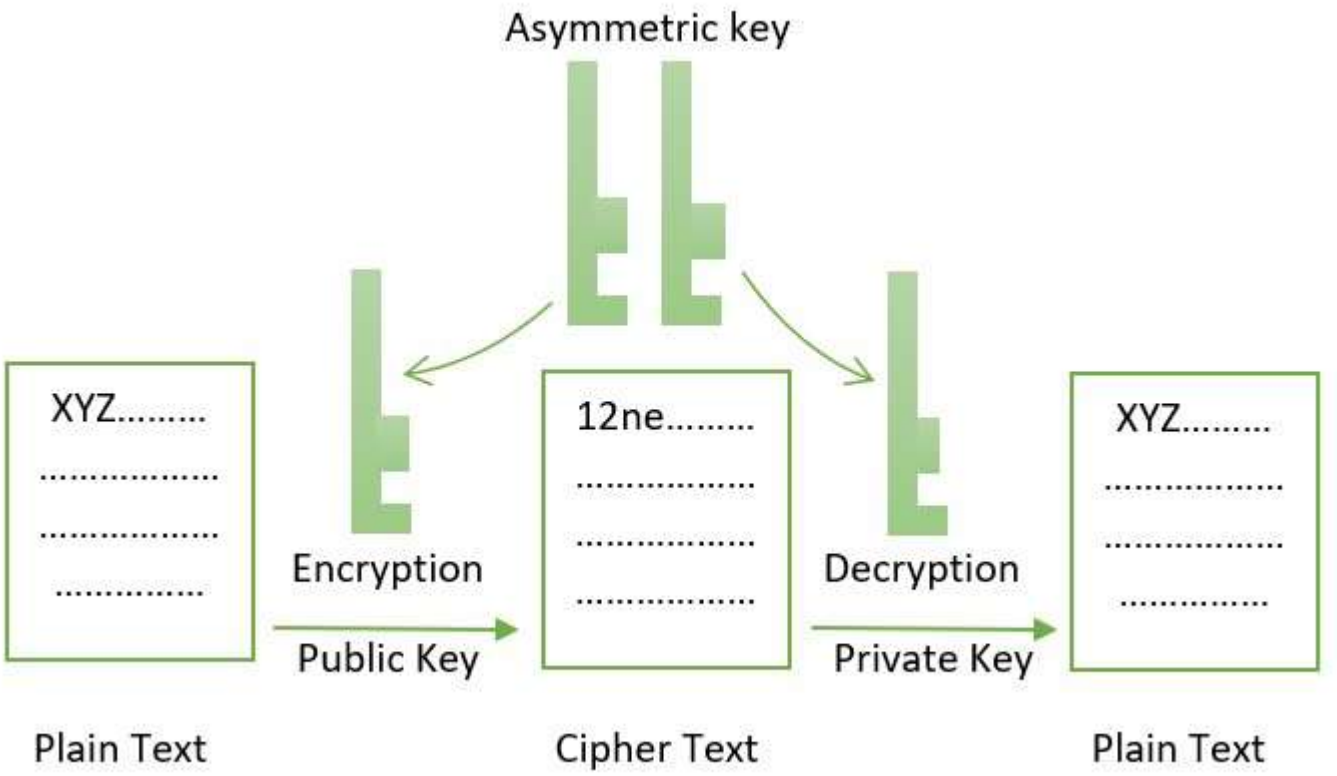



Symmetric Cryptography

**2. Asymmetric-key Encryption:** This cryptographic method uses different keys for the encryption and decryption process. This encryption method uses public and private key methods. This public key method help completely unknown parties to share information between them like email id. private key helps to decrypt the messages and it also helps in the verification of the digital signature. The mathematical relation between the keys is that the private key cannot be derived from the public key, but the public key can be derived from the private key. **Example:** ECC,DSS etc.

**Features:**

- It is also known as Public-key cryptography.
- It is often used for sharing secret keys of symmetric cryptography.
- It requires a long processing time for execution.
- Plays a significant role in website server authenticity.




Asymmetric Cryptography

## Wallets And Digital Signatures

A blockchain wallet is a special software or a hardware device that is used to keep the transaction information and personal information of the user. Blockchain wallets do not contain the actual currency. The wallets are used to keep private keys and maintain a transaction balance. Wallets are only a communication tool to communicate to carry out transactions with other users. The real data or currency is stored in blocks in the blockchain.

Digital signatures are like proofs that the user gives to the recipient and other nodes in the network to prove that it is a legitimate node in the network to carry out transactions. While initiating a transaction with other nodes in the blockchain network, the user first has to create a unique digital signature by combining the transaction data with the user's private key using a special algorithm. This process will guarantee the authenticity of the node and the integrity of the data.

## Cryptography Hash Function in Blockchain

One of the most notable uses of cryptography is cryptographic hashing. Hashing enables immutability in the blockchain. The encryption in cryptographic hashing does not involve any use of keys. When a transaction is verified hash algorithm adds the hash to the block, and a new unique hash is added to the block from the original transaction. Hashing continues to combine or make new hashes, but the original footprint is still accessible. The single combined hash is called the root hash. Hash Function helps in linking the block as well as maintaining the integrity of data inside the block and any alteration in the block data leads to a break of the blockchain. Some commonly used hashed function is MD5 and SHA-1.

**Properties of Cryptographic Hash:**

- For a particular message hash function does not change.
- Every minor change in data will result in a change in a major change in the hash value.

- The input value cannot be guessed from the output hash function.
- They are fast and efficient as they largely rely on bitwise operations.

**Benefits of Hash function in Blockchain:**

1. Reduce the bandwidth of the transaction.
2. Prevent the modification in the data block.
3. Make verification of the transaction easier.

### Use of Cryptographic Hash Functions

As the blockchain is also public to everyone it is important to secure data in the blockchain and keeps the data of the user safe from malicious hands. So, this can be achieved easily by cryptography.

- When the transaction is verified through a hash algorithm, it is added to the blockchain, and as the transaction becomes confirmed it is added to the network making a chain of blocks.
- Cryptography uses mathematical codes, it ensures the users to whom the data is intended can obtain it for reading and processing the transaction.
- Many new tools related to the application of cryptography in blockchain have emerged over the years with diverse functionalities.

### Benefits of Cryptography in Blockchain

There are a huge number of benefits of cryptography in blockchain some of them are stated below:

- **Encryption:** Cryptography uses asymmetric encryption to ensure that the transaction on their network guards the information and communication against unauthorized revelation and access to information.
- **Immutability:** This feature of cryptography makes it important for blockchain and makes it possible for blocks to get securely linked by other blocks and also to ensure the reliability of data stored in the blockchain, it also ensures that no attacker can derive a valid signature for unposed queries from previous queries and their corresponding signatures.
- **Security:** Cryptography makes the records of transactions easier using encryption of data, and accessing of data using public and private keys. Cryptographic hashing tampering with data is not possible, making blockchain more secure.
- **Scalability:** Cryptography makes the transaction irreversible giving the assurance that all users can rely on the accuracy of the digital ledger. It allows limitless transactions to be recorded securely in the network.
- **Non-repudiation:** The digital signature provides the non-repudiation service to guard against any denial of a message passed by the sender. This benefit can be associated with collision resistance i.e.; since every input value has a unique hash function so there is no clash between the messages that are sent and one message can be easily differentiated from the other.
- **Prevent hackers:** The digital signature prevents hackers from altering the data because if the data changes, the digital signature becomes invalid. With the help of cryptography, it protects the data from hackers and makes cryptography in blockchain unstoppable.

### Limitations of Cryptography in Blockchain

Below are some of the limitations of cryptography in the blockchain:

- **Information difficult to access:** Strongly encrypted and digitally signed information can be difficult to access even for a legitimate user at the most critical time of decision-making. The network can be attacked and rendered non-functional by an intruder.
- **High availability:** It is one of the fundamental aspects of information security, and cannot be ensured through the use of cryptography. Other methods are needed to guard against the threats such as denial of service or complete breakdown of the information systems.

- **No protection against vulnerabilities:** Cryptography does not guard against the vulnerabilities and threats that emerge from the poor design of protocols, procedures, and systems. These issues need to be fixed with the proper design of the defense infrastructure.
- **Expensive:** Cryptography needs huge time and money investments. Public key cryptography needs setting up and maintenance of public key infrastructure which requires huge investment. Addition of cryptographic techniques while sending messages and information processing adds to the delay.
- **Vulnerability:** The security of cryptographic techniques depends on the complexity and difficulty of the mathematical problem. Any breakthrough in solving such mathematical problems can make cryptographic techniques vulnerable.

# CRYPTOCURRENCY

A cryptocurrency is a digital or virtual currency secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology—a distributed ledger enforced by a disparate network of computers.

A defining feature of cryptocurrencies is that they are generally not issued by any central authority, rendering them theoretically immune to government interference or manipulation.

## Blockchain

Central to the appeal and functionality of Bitcoin and other cryptocurrencies is blockchain technology. As its name indicates, a blockchain is essentially a set of connected blocks of information on an online ledger. Each block contains a set of transactions that have been independently verified by each validator on a network.

Every new block generated must be verified by each node before being confirmed, making it almost impossible to forge transaction histories.1 The contents of the online ledger must be agreed upon by a network of individual nodes, or computers that maintain the ledger.

## Types of Cryptocurrency

Many cryptocurrencies were created to facilitate work done on the blockchain they are built on. For example, Ethereum's ether was designed to be used as payment for validation work done on the blockchain. When the blockchain transitioned to proof-of-stake in September 2022, ether (ETH) inherited an additional duty as the blockchain's staking mechanism. Ripple's XRP is designed to be used by banks to facilitate transfers between different geographies.

Most of the time, when you hear about cryptocurrency types, you hear the coin's name. However, coin names differ from coin types. Here are some of the types you'll find with some of the names of tokens in that category:

- **Utility**: XRP and ETH are two examples of utility tokens. They serve specific functions on their respective blockchains.
- **Transactional**: Tokens designed to be used as a payment method. Bitcoin is the most well-known of these.
- **Governance**: These tokens represent voting or other rights on a blockchain, such as Uniswap.
- **Platform**: These tokens support applications built to use a blockchain, such as Solana.
- **Security tokens**: Tokens representing ownership of an asset, such as a stock that has been tokenized (value transferred to the blockchain). MS Token is an example of a securitized token. If you can find one of these for sale, you can gain partial ownership of the Millenium Sapphire.3

## Are Cryptocurrencies Safe Investments?

Cryptocurrencies have attracted a reputation as unstable investments due to high investor losses as a result of scams, hacks, and bugs. Although the underlying cryptography is generally secure, the technical complexity of using and storing crypto assets can be a significant hazard to new users.

In addition to the market risks associated with speculative assets, cryptocurrency investors should be aware of the following risks:

- **User risk**: Unlike traditional finance, there is no way to reverse or cancel a cryptocurrency transaction after it has already been sent. By some estimates, about one-fifth of all bitcoins are now inaccessible due to lost passwords or incorrect sending addresses.11
- **Regulatory risks**: The regulatory status of some cryptocurrencies is still unclear, with many governments seeking to regulate them as securities, currencies, or both. A sudden regulatory crackdown could make it difficult to sell cryptocurrencies or cause a market-wide price drop.
- **Counterparty risks**: Many investors and merchants rely on exchanges or other custodians to store their cryptocurrency. Theft or loss by one of these third parties could result in losing one's entire investment.
- **Management risks**: Due to the lack of coherent regulations, there are few protections against deceptive or unethical management practices. Many investors have lost large sums to management teams that failed to deliver a product.
- **Programming risks**: Many investment and lending platforms use automated smart contracts to control the movement of user deposits. An investor using one of these platforms assumes the risk that a bug or exploit in these programs could cause them to lose their investment.
- **Market Manipulation**: Market manipulation remains a substantial problem in cryptocurrency, with influential people, organizations, and exchanges acting unethically.

## Advantages and Disadvantages of Cryptocurrency

Cryptocurrencies were introduced with the intent to revolutionize financial infrastructure. As with every revolution, however, there are tradeoffs involved. At the current stage of development for cryptocurrencies, there are many differences between the theoretical ideal of a decentralized system with cryptocurrencies and its practical implementation.

Some advantages and disadvantages of cryptocurrencies are as follows.

### Advantages

- Removes single points of failure
- Easier to transfer funds between parties
- Removes third parties
- Can be used to generate returns
- Remittances are streamlined

### Disadvantages

- Transactions are pseudonymous
- Pseudonymity allows for criminal uses
- Have become highly centralized
- Expensive to participate in a network and earn
- Off-chain security issues
- Prices are very volatile