1

COMPUTER FORENSIC FUNDAMENTALS

Unit Structure :

1.1 INTRODUCTION: NUMBER THEORY AND CRYPTOGRAPHY

The part of mathematics devoted to the study of the set of integers and their properties is known as number theory. In this chapter we will develop some of the important concepts of number theory including many of those used in computer science.

In modern cryptographic systems, the messages are represented by numerical values prior to being encrypted and transmitted. The encryption processes are mathematical operations that turn the input numerical values into output numerical values. Building, analyzing, and attacking these cryptosystems requires mathematical tools. The most important of these is number theory.

This chapter introduces several important applications of number theory. We also introduce the subject of cryptography. Number theory plays an essentially role both in classical cryptography, first used thousands of years ago, and modern cryptography, which plays an essential role in electronic communication. We will show how the ideas we develop can be used in cryptographical protocols, introducing protocols for sharing keys and for sending signed messages. Number theory, once considered the purest of subjects, has become an essential tool in providing computer and Internet security. This chapter presents the basic tools needed for the rest of the book.

1.2 DIVISIBILITY

Number theory is concerned with the properties of the integers. One of the most important is divisibility.

Definition.

Let a and b be integers with $a \neq 0$. *We say that a* divides *b, if there is an integer k such that b =*

ak. This is denoted by a|b. *Another way to express this is that b is a multiple of a.*

The positive divisors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24.

Examples: 3|15, -15|60, 7∤18

1.3 PRIME NUMBERS

Every integer greater than 1 is divisible by at least two integers, because a positive integer is divisible by 1 and by itself. Positive integers that have exactly two different positive integer factors are called **primes**.

An integer p greater than 1 is called *prime* if the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called *composite*.

Remark: The integer n is composite if and only if there exists an integer a such that $a \mid n$ and $1 <$

$a < n$.

EXAMPLE 1 The integer 7 is prime because its only positive factors are 1 and 7, whereas the integer 9 is composite because it is divisible by 3.

1.4 Greatest Common Divisor

The largest integer that divides both of two integers is called the **greatest common divisor** of these integers.

DEFINITION

Let *a* and *b* be integers, not both zero. The largest integer *d* such that *d* | *a* and *d* | *b* is called the *greatest common divisor* of *a* and *b*. The greatest common divisor of *a* and *b* is denoted by gcd*(a, b)*.

The greatest common divisor of two integers, not both zero, exists because the set of common divisors of these integers is nonempty and finite. One way to find the greatest common divisor of two integers is to find all the positive common divisors of both integers and then take the largest divisor.

EXAMPLE 1

What is the greatest common divisor of 24 and 36?

Solution: The positive common divisors of 24 and 36 are 1, 2, 3, 4, 6, and 12.

Hence, gcd (24, 36) = 12.

EXAMPLE 2

What is the greatest common divisor of 17 and 22?

Solution:

The integers 17 and 22 have no positive common divisors other than 1, so that gcd*(17, 22)* = 1.

The integers *a* and *b* are *relatively prime* if their greatest common divisor is 1.

Another way to find the greatest common divisor of two positive integers is to use the prime factorizations of these integers. Suppose that the prime factorizations of the positive integers *a* and *b* are

$$a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}, \; b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n},$$

where each exponent is a nonnegative integer, and where all primes occurring in the prime factorization of either *a* or *b* are included in both factorizations, with zero exponents if necessary. Then gcd*(a, b)* is given by

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \cdots p_n^{\min(a_n, b_n)},$$

where min*(x, y)* represents the minimum of the two numbers *x* and *y*. To show that this formula for gcd*(a, b)* is valid, we must show that the integer on the right-hand side divides both *a* and *b*, and that no larger integer also does. This integer does divide both *a* and *b*, because the power of each prime in the factorization does not exceed the power of this prime in either the factorization of *a* or that of *b*. Further, no larger integer can divide both *a*

and *b*, because the exponents of

the primes in this factorization cannot be increased, and no other primes can be included.

EXAMPLE 3

Because the prime factorizations of 120 and 500 are $120 = 2^3 .3 .5$ and $500 = 2^2 .5^3$, the greatest common divisor is

$$\gcd(120, 500) = 2^{\min(3,\,2)} \cdot 3^{\min(1,\,0)} \cdot 5^{\min(1,\,3)}$$

$$= 2^2 3^0 5^1$$

$$= 20.$$

Prime factorizations can also be used to find the **least common multiple** of two integers.

1.5 CONGRUENCES

One of the most basic and useful notions in number theory is modular arithmetic, or congruences.

Definition. *Let a , b , n be integers with n ≠0 . We say that*

a ≡ b (mod n)

(read: a is congruent to b mod n) if a - b is a multiple (positive or negative) of n.

Another formulation is that a= b (mod n) if a and b differ by a multiple of n. This can be rewritten as a ≡ b + nk for some integer k (positive or negative).

Examples.

32 ≡ 7 (mod 5), -12 ≡ 37 (mod 7),17 ≡ 1 7 (mod 13).

Congruence behaves very much like equality, In fact, the notation for congruence was intentionally chosen to resemble the notation for equality.

1.5.1 Linear Congruences

A congruence of the form

$ax \equiv b \pmod{m}$,

where *m* is a positive integer, *a* and *b* are integers, and *x* is a variable, is called a **linear congruence**. Such congruences arise throughout number theory and its applications.

How can we solve the linear congruence $ax \equiv b \pmod{m}$, that is, how can we find all integers *x* that satisfy this congruence? One method that we will describe uses an integer $\bar{a}$ such that $\bar{a}a \equiv 1 \pmod{m}$, if such an integer exists. Such an integer $\bar{a}$ is said to be an **inverse** of *a* modulo *m*.

1.6 THE EUCLIDEAN ALGORITHM

Computing the greatest common divisor of two integers directly from the prime factorizations of these integers is inefficient. The reason is that it is time-consuming to find prime factorizations. We will give a more efficient method of finding the greatest common divisor, called the **Euclidean algorithm**. This algorithm has been known since ancient times. It is named after the ancient Greek mathematician Euclid.

Before describing the Euclidean algorithm, we will show how it is used to find gcd(91, 287).

First, divide 287, the larger of the two integers, by 91, the smaller, to obtain

$287 = 91 \cdot 3 + 14.$

Any divisor of 91 and 287 must also be a divisor of $287 - 91 \cdot 3 = 14.$

Also, any divisor of 91 and 14 must also be a divisor of $287 = 91 \cdot 3 + 14.$
Hence, the greatest common divisor of 91

Theorem

If a and m are relatively prime integers and m>1, then an inverse of a modulo m exists.

Furthermore, this inverse is unique modulo m.

(That is, there is a unique positive integer a less than m that is an inverse of a modulo m and every other inverse of a modulo m is congruent to a

Congruences have the following properties:

1. $a \equiv b \pmod{n}$ if $n \mid (a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

To demonstrate the first point, if $n \mid (a - b)$, then $(a - b) = kn$ for some k.

So we can write $a = b + kn$. Therefore, (a mod n) = (remainder when $b + kn$ is divided by n) = (remainder when b is divided by n) = (b mod n).

$23 \equiv 8 \pmod{5}$	because	$23 - 8 = 15 = 5 \times 3$
$-11 \equiv 5 \pmod{8}$	because	$-11 - 5 = -16 = 8 \times (-2)$
$81 \equiv 0 \pmod{27}$	because	$81 - 0 = 81 = 27 \times 3$

and 287 is the same as the greatest common divisor of 91 and 14.
This means that the problem of finding gcd(91, 287) has been reduced to the problem of finding gcd(91, 14).

Next, divide 91 by 14 to obtain

$91 = 14 \cdot 6 + 7.$

Because any common divisor of 91 and 14 also divides 91 − 14 · 6 = 7 and any common divisor of 14 and 7 divides 91, it follows that gcd(91, 14) = gcd(14, 7).

Continue by dividing 14 by 7, to obtain

14 = 7 · 2.

Because 7 divides 14, it follows that gcd(14, 7) = 7. Furthermore, because gcd(287, 91) =

gcd(91, 14) = gcd(14, 7) = 7, the original problem has been solved.

We will use successive divisions to reduce the problem of finding the greatest common divisor of two positive integers to the same problem with smaller integers, until one of the integers is zero.

The Euclidean algorithm is based on the following result about greatest common divisors and the division algorithm.

EXAMPLE 4

Find the greatest common divisor of 414 and 662 using the Euclidean algorithm.

Solution: Successive uses of the division algorithm give:

$$662 = 414 \cdot 1 + 248$$
$$414 = 248 \cdot 1 + 166$$
$$248 = 166 \cdot 1 + 82$$
$$166 = 82 \cdot 2 + 2$$
$$82 = 2 \cdot 41.$$

Hence, gcd(414, 662) = 2, because 2 is the last nonzero remainder.

The Euclidean algorithm is expressed in pseudocode in Algorithm 1.

ALGORITHM 1 The Euclidean Algorithm.

```
procedure gcd(a, b: positive integers)
x := a
y := b
while y ≠ 0
    r := x mod y
    x := y
    y := r
return x {gcd(a, b) is x}
```

In Algorithm 1, the initial values of x and y are a and b, respectively. At each stage of the procedure, x is replaced by y, and y is replaced by x **mod** y, which is the remainder when x is divided by y. This process is repeated as long as $y \neq 0$. The algorithm terminates when $y = 0$, and the value of x at that point, the last nonzero remainder in the procedure, is the greatest common divisor of a and b.

The number of divisions required to find the greatest common divisor of a and b, where $a \geq b$, is $O(\log b)$.

1.7 MODULAR ARITHMETIC

Modular arithmetic is the branch of arithmetic mathematics related with the "mod" functionality. Basically, modular arithmetic is related with computation of "mod" of expressions. Expressions may have digits and computational symbols of addition, subtraction, multiplication, division or any other. Here we will discuss briefly about all modular arithmetic operations.

Quotient Remainder Theorem:

It states that, for any pair of integers a and b (b is positive), there exist two unique integers q and r such that:

$a = b \times q + r$ *where* $0 <= r < b$

Example:

If a = 20, b = 6

then q = 3, r = 2 20 = 6 x 3 + 2

1.7.1 Modular Addition

Rule for modular addition is:

(a + b) mod m = ((a mod m) + (b mod m)) mod m

Example:

(15 + 17) % 7

= ((15 % 7) + (17 % 7)) % 7

= (1 + 3) % 7

= 4 % 7

= 4

The same rule is to modular subtraction. We don't require much modular subtraction but it can also be done in the same way.

1.7.2 Modular Multiplication

The Rule for modular multiplication is:

(a x b) mod m = ((a mod m) x (b mod m)) mod m

Example:

(12 x 13) % 5

= ((12 % 5) x (13 % 5)) % 5

= (2 x 3) % 5

= 6 % 5

= 1

1.7.3 Modular Division

The modular division is totally different from modular addition, subtraction and multiplication. It also does not exist always.

(a / b) mod m is not equal to ((a mod m) / (b mod m)) mod m.

This is calculated using the following formula:

(a / b) mod m = (a x (inverse of b if exists)) mod m

1.7.4 Modular Inverse:

The modular inverse of a mod m exists only if a and m are relatively prime i.e. gcd(a, m) = 1. Hence, for finding the inverse of an under modulo m, if (a x b) mod m = 1 then b is the modular inverse of a.

Example:

a = 5, m = 7 (5 x 3) % 7 = 1 hence, 3 is modulo inverse of 5 under 7.

1.8 FERMAT'S LITTLE THEOREM

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then

$$a^{p-1} \equiv 1 \pmod{p} \qquad \text{.....(1)}$$

$a = 7, p = 19$
$7^2 = 49 \equiv 11 \pmod{19}$
$7^4 \equiv 121 \equiv 7 \pmod{19}$
$7^8 \equiv 49 \equiv 11 \pmod{19}$
$7^{16} \equiv 121 \equiv 7 \pmod{19}$
$a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}$

Example

```
P = an integer Prime number
a = an integer which is not multiple of P
Let a = 2 and P = 17

According to Fermat's little theorem
 2 17 - 1      ≡ 1 mod(17)
we got  65536 % 17 ≡ 1
that mean (65536-1) is an multiple of 17
```

An alternative form of Fermat's theorem is also useful: If p is prime and a is a positive integer, then

$a^p \equiv a \pmod{p}$ ……**(2)**

Note that the first form of the theorem [Equation (1)] requires that a be relatively prime to p, but this form does not.

$p = 5, a = 3$	$a^p = 3^5 = 243 \equiv 3 \pmod 5 = a \pmod p$
$p = 5, a = 10$	$a^p = 10^5 = 100000 \equiv 10 \pmod 5 \equiv 0 \pmod 5 = a \pmod p$

1.9 Euler's theorem

Euler's theorem states that for every a and n that are relatively prime:

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

$a = 3; n = 10; \phi(10) = 4;$	$a^{\phi(n)} = 3^4 = 81 = 1 \pmod{10} = 1 \pmod n$
$a = 2; n = 11; \phi(11) = 10;$	$a^{\phi(n)} = 2^{10} = 1024 = 1 \pmod{11} = 1 \pmod n$

As is the case for Fermat's theorem, an alternative form of the theorem is also useful:

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

1.10 The Chinese Remainder Theorem

The *Chinese remainder theorem*, named after the Chinese heritage of problems involving systems of linear congruences, states that when the moduli of a system of linear congruences are pairwise relatively prime, there is a unique solution of the system modulo the product of the moduli.

THEOREM

Let $m_1, m_2, \ldots, m_n$ be pairwise relatively

prime positive integers greater than one and $a_1, a_2, \ldots, a_n$ arbitrary integers. Then the system

$x \equiv a_1 \pmod{m_1}$, $x \equiv a_2 \pmod{m_2}$,

•

•

•

$x \equiv a_n \pmod{m_n}$

has a unique solution modulo $m = m_1 m_2 \cdots m_n$. (That is, there is a solution x with

$0 \leq x < m$, and all other solutions are congruent modulo m to this solution.)

1.11 Quadratic residue

In number theory, an integer q is called a quadratic residue modulo n if it is congruent to a perfect square modulo n; i.e., if there exists an integer x such that:

$$x^2 \equiv q \pmod{n}.$$

Otherwise, q is called a quadratic nonresidue modulo.

1.11.1 Quadratic Reciprocity Theorem

If p and q are distinct odd primes, then the quadratic reciprocity theorem states that the congruences

$$x^2 \equiv q \pmod{p}$$
$$x^2 \equiv p \pmod{q}$$

are both solvable or both unsolvable unless both p and q leave the remainder 3 when divided by

4 (in which case one of the congruences is solvable and the other is not). Written symbolically,

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4},$$

where

$$\left(\frac{p}{q}\right) \equiv \begin{cases} 1 & \text{for } x^2 \equiv p \pmod{q} \text{ solvable for } x \\ -1 & \text{for } x^2 \equiv p \pmod{q} \text{ not solvable for } x \end{cases}$$

is known as a Legendre symbol.

1.11.2 Jacobi Symbol

The Jacobi symbol, written (n/m) or $\left(\frac{n}{m}\right)$ is defined for positive odd m as

$$\left(\frac{n}{m}\right)=\left(\frac{n}{p_1}\right)^{a_1}\left(\frac{n}{p_2}\right)^{a_2}\cdots\left(\frac{n}{p_k}\right)^{a_k},$$

where

$$m=p_1^{a_1}\,p_2^{a_2}\cdots p_k^{a_k}$$

is the prime factorization of m and is (n/p_j) is the Legendre symbol. (The Legendre symbol is equal to $\pm$ 1 depending on whether n is a quadratic residue modulo m.) Therefore, when m is a prime, the Jacobi symbol reduces to the Legendre symbol. Analogously to the Legendre symbol, the Jacobi symbol is commonly generalized to have value

1.11.1 Quadratic Reciprocity Theorem

If p and q are distinct odd primes, then the quadratic reciprocity theorem states that the congruences

$$x^2\equiv q\,(\mathrm{mod}\,p)$$
$$x^2\equiv p\,(\mathrm{mod}\,q)$$

are both solvable or both unsolvable unless both p and q leave the remainder 3 when divided by

4 (in which case one of the congruences is solvable and the other is not). Written symbolically,

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right)=(-1)^{(p-1)(q-1)/4},$$

where

$$\left(\frac{p}{q}\right)\equiv\begin{cases}1 & \text{for } x^2\equiv p\,(\mathrm{mod}\,q) \text{ solvable for } x\\ -1 & \text{for } x^2\equiv p\,(\mathrm{mod}\,q) \text{ not solvable for } x\end{cases}$$

is known as a Legendre symbol.

1.11.2 Jacobi Symbol

The Jacobi symbol, written (n/m) or $\left(\frac{n}{m}\right)$ is defined for positive odd m as

$$\left(\frac{n}{m}\right)=\left(\frac{n}{p_1}\right)^{a_1}\left(\frac{n}{p_2}\right)^{a_2}\cdots\left(\frac{n}{p_k}\right)^{a_k},$$

where

$$m=p_1^{a_1}\,p_2^{a_2}\cdots p_k^{a_k}$$

is the prime factorization of m and is (n/p_i) is the Legendre symbol. (The Legendre symbol is equal to ± 1 depending on whether n is a quadratic residue modulo m.) Therefore, when m is a prime, the Jacobi symbol reduces to the Legendre symbol. Analogously to the Legendre symbol, the Jacobi symbol is commonly generalized to have value

$$\left(\frac{n}{m}\right)=0 \text{ if GCD } (m, n)\neq 1,$$

giving

$$\left(\frac{n}{n}\right)=0$$

as a special case. Note that the Jacobi symbol is *not defined* for $m \leq 0$ or m even. The Jacobi symbol is implemented in the Wolfram Language as Jacobi Symbol [*n*, *m*].

1.12 SUMMARY

After studying this chapter, we are able to:

- Understand the concept of divisibility.
- Understand how to use the Euclidean algorithm to find the greatest common divisor.
- Discuss key concepts relating to prime numbers.
- Understand Fermat's theorem.
- Understand Euler's theorem.
- Explain the Chinese remainder theorem.
- Understand Quadratic residue, Quadratic Reciprocity Theorem and Jacobi Symbol.

1.13 QUESTIONS

1) What is Euclidean Algorithm?

2) Explain Chinese Remainder Theorem.

3) Define Fermat's little theorem and Euler's theorem.

4) Explain following terms: a) Prime numbers b) Greatest Common Divisor

1.14 REFERENCES

1) Discrete Mathematics and Its Applications, Kenneth H. Rosen, 7th Edition, McGraw Hill, 2012.

2) Cryptography Theory and Practice, 3rd Edition, Douglas R. Stinson, 2005.

3) Network Security and Cryptography, Atul Kahate, McGraw Hill, 2003.

4) Cryptography and Network Security: Principles and Practices, William Stalling, Fourth Edition, Prentice Hall, 2013.

5) Introduction to Cryptography with coding theory, second edition, Wade Trappe, Lawrence C. Washington, Pearson, 2005.

❄❄❄❄❄❄❄

2

SIMPLE CRYPTOSYSTEMS

Unit Structure :

2.0 OBJECTIVE

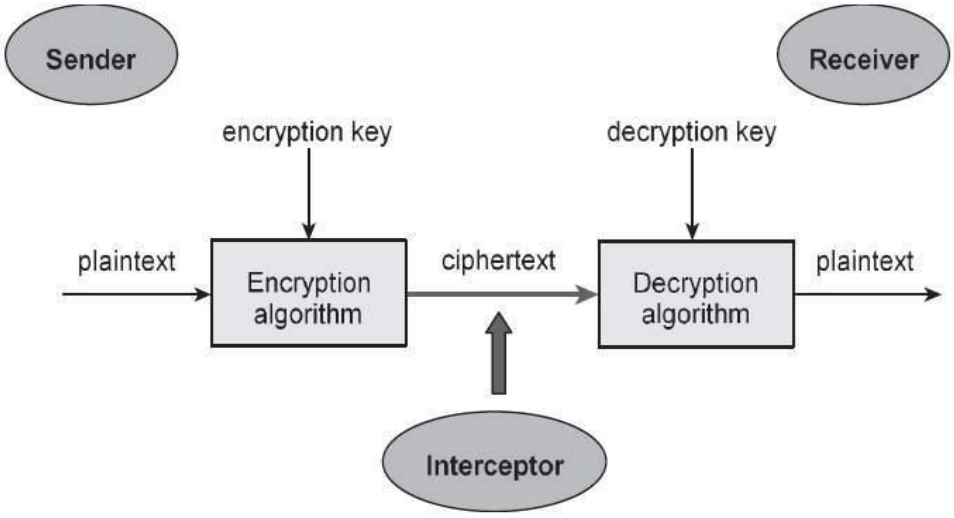
A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services.

A cryptosystem is also referred to as a cipher system.

In this chapter we are going to learn different cipher methods for encryption and decryption.

2.1 SIMPLE CRYPTOSYSTEMS

- A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services.
- A cryptosystem is also referred to as a cipher system.
- Simple model of a cryptosystem provides confidentiality to the information being transmitted.
- This basic model is shown below −



- The illustration shows a sender who wants to transfer some sensitive data to a receiver in such a way that any party intercepting or eavesdropping on the communication channel cannot extract the data.
- **The objective of this simple cryptosystem is that at the end of the process, only the sender and the receiver will know the plaintext.**
- **Components of a Cryptosystem:**
- The various components of a basic cryptosystem are as follows −
  - **Plaintext-**
    - It is the data to be protected during transmission.
  - **Encryption Algorithm-**
    - It is a mathematical process that produces a ciphertext for any given plaintext and encryption key.
    - It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
  - **Ciphertext-**
    - It is the scrambled version of the plaintext produced by the encryption algorithm using a specific encryption key.
    - The ciphertext is not guarded. It flows on public channels.
    - It can be intercepted or compromised by anyone who has access to the communication channel.
  - **Decryption Algorithm-**
    - It is a mathematical process that produces a unique plaintext for any given ciphertext and decryption key.
    - It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext.
    - The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.

- **Encryption Key-**
  - It is a value that is known to the sender.
  - The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
- **Decryption Key-**
  - It is a value that is known to the receiver.
  - The decryption key is related to the encryption key, but is not always identical to it.
  - The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.

Note :

1. Key space is a collection of all possible decryption keys in a cryptosystem.

2. An interceptor (an attacker) is an unauthorized entity who attempts to determine the plaintext. He can see the ciphertext and may know the decryption algorithm. He, however, must never know the decryption key.

Types of Cryptosystems:

- Two types of cryptosystems based on the manner in which encryption-decryption are:
  - Symmetric Key Encryption
  - Asymmetric Key Encryption
- The main difference between these cryptosystems is the relationship between the encryption and the decryption key.
- Logically, in any cryptosystem, both the keys are closely associated.
- It is practically impossible to decrypt the ciphertext with the key that is unrelated to the encryption key.
- Let's discuss each in detail:
  - **Symmetric Key Encryption**
    - The encryption process where the same keys are used for encrypting and decrypting the information is known as Symmetric Key Encryption.
    - The study of symmetric cryptosystems is referred to as symmetric cryptography.

- Symmetric cryptosystems are also sometimes referred to as secret key cryptosystems.
- A few well-known examples of symmetric key encryption methods are −

- Digital Encryption Standard (DES),
- Triple-DES (3DES), IDEA, and
- BLOWFISH.

- Features of cryptosystem based on symmetric key encryption are −

- Persons using symmetric key encryption must share a common key prior to exchange of information.
- Keys are recommended to be changed regularly to prevent any attack on the system.
- A robust mechanism needs to exist to exchange the key between the communicating parties.
- As keys are required to be changed regularly, this mechanism becomes expensive and cumbersome.
- In a group of n people, to enable two-party communication between any two persons, the number of keys required for group is $n \times (n - 1)/2$.
- Length of Key (number of bits) in this encryption is smaller and hence, process of encryption-decryption is faster than asymmetric key encryption.
- Processing power of a computer system required to run a symmetric algorithm is less.
- **Challenge of Symmetric Key Cryptosystem**
- There are two restrictive challenges of employing symmetric key cryptography.
  - **Key establishment −**
    - Before any communication, both the sender and the receiver need to agree on a secret symmetric key.
    - It requires a secure key establishment mechanism in place.
  - **Trust Issue −**
    - Since the sender and the receiver use the same symmetric key, there is an implicit requirement that the sender and the receiver 'trust' each other.
    - For example, it may happen that the receiver has lost the key to an attacker and the sender is not informed.

- **Asymmetric Key Encryption**
  - The encryption process where different keys are used for encrypting and decrypting the information is known as Asymmetric Key Encryption.
  - Though the keys are different, they are mathematically related and hence, retrieving the plaintext by decrypting ciphertext is feasible.
  - Features of this encryption scheme are as follows −
    - Every user in this system needs to have a pair of dissimilar keys, private key and public key.
    - These keys are mathematically related − when one key is used for encryption, the other can decrypt the ciphertext back to the original plaintext.
    - It requires putting the public key in a public repository and the private key as a well-guarded secret.
    - Hence, this scheme of encryption is also called Public Key Encryption.
    - Though public and private keys of the user are related, it is computationally not feasible to find one from another.
    - This is a strength of this scheme.
    - When Host1 needs to send data to Host2, he obtains the public key of Host2 from the repository, encrypts the data, and transmits.
    - Host2 uses his private key to extract the plaintext.
    - Length of Keys (number of bits) in this encryption is large and hence, the process of encryption-decryption is slower than symmetric key encryption.
    - Processing power of the computer system required to run an asymmetric algorithm is higher.

2.2 SHIFT CIPHER

- A shift cipher is a substitution cipher, the principle of which is to shift the letters by one or more values in the alphabet.
- Example:
  - The letter A shifted by 1 place in the alphabet becomes B
  - The Caesar cipher is a shift cipher, usually presented with a shift key of value 3.
- The shift cipher encryption uses an alphabet and a key (made up of one or more values) that shifts the position of its letters.
- A letter in position N in the alphabet, can be shifted by X into the letter located at position

- N + X (This is equivalent to using a substitution with a shifted alphabet).

- Example:
  - Take the letter E in position 5 in the alphabet

    ABCDEFGHIJKLMNOPQRSTUVWXYZ,

    it will be encrypted by a shift of 3 in position 8 or H.
- If the shifted position exceeds the number of letters in the alphabet, then take it at the beginning
- Example:
  - Z shifted by 1 gives A.
- It is thus possible to define different types of shifts, some shifts correspond to known encryption algorithms:
  - **A single shift**
    - all letters are shifted by the same value is called **Caesar Code.**
  - **A multiple shift**
    - according to a sequence or a key that is repeated the letters are shifted from each of the key values, is called **Vigenere Cipher.**
  - **A mathematical shift**
    - the easier is progressive, shifting the nth letter of the value n is the Trithemius Cipher or
    - if the shift is more complex Affine Cipher or even Hill Cipher.
- **Examples :**
  - Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - Shift: 23
  - Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW
  - Text : ATTACKATONCE
  - Shift: 4
  - Cipher: EXXEGOEXSRGI

Decrypt using the shift cipher:

- Decryption requires knowing the shift used and the alphabet.
- Take a letter in position N in the alphabet that has been encrypted by a shift of X, it must be shifted by -X to return to its original position N-X.

- Example:
    - The letter H in position 8 in the alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ,

      will be decrypted from a shift of 3 in position 8-3=5 or E.
- Example:
    - The word TIJGU is decoded with an offset of 1 as SHIFT

Advantages:

- Easy to implement and use thus, making it suitable for beginners to learn about encryption.
- Can be physically implemented, such as with a set of rotating disks or a set of cards, known as a scytale, which can be useful in certain situations.
- Requires only a small set of pre-shared information.
- Can be modified easily to create a more secure variant, such as by using multiple shift values or keywords.

Disadvantages:

- It is not secure against modern decryption methods.
- Vulnerable to known-plaintext attacks, where an attacker has access to both the encrypted and unencrypted versions of the same messages.
- The small number of possible keys means that an attacker can easily try all possible keys until the correct one is found, making it vulnerable to a brute force attack.
- It is not suitable for long text encryption as it would be easy to crack.
- It is not suitable for secure communication as it is easily broken.
- Does not provide confidentiality, integrity, and authenticity in a message.

2.3 SUBSTITUTION CIPHER:

- In a Substitution cipher, any character of plain text from the given fixed set of characters is substituted by some other character from the same set depending on a key.
- For example
    - with a shift of 1,
        - A would be replaced by B,
        - B would become C, and so on.
- The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25.

Encryption using a substitution cipher:

- For a substitution to be correct, it is necessary for the same element to be substituted by only one other.
- There is never more than one possibility of encryption or decryption.
- Encryption of a letter by a shift n can be described mathematically as.
- **Examples:**
    - Plain Text: I am studying Data Encryption
    - Key: 4
    - Output: M eq wxyhCmrk Hexe IrgvCtxmsr

Substitution cipher method:

- Substitution cipher is one of the most basic cryptography methods.
- Some of them are as follow::
    - **Ciphers by mono-alphabetic substitution,** with a disordered alphabet, one letter replaces another.
    - **Encryptions by poly-alphabetic substitution,** with several alphabets.
    - **Encryptions by homophonic substitution,** the same element can be substituted by several others.
    - **Substitution by dictionary, with words,** n-grams substituted by others.

Algorithm for Substitution Cipher:

- **Input:**
    - A String of both lower and upper case letters, called PlainText.
    - An Integer denoting the required key.
- **Procedure:**
    - Create a list of all the characters.
    - Create a dictionary to store the substitution for all characters.
    - For each character, transform the given character as per the rule, depending on whether we're encrypting or decrypting the text.
    - Print the new string generated.

2.4 AFFINE CIPHER

- Affine cipher is the name given to a substitution cipher whose key consists of 2 coefficients A and B constituting the parameters of a mathematical linear function f=Ax+B (called affine).
- Encryption using the Affine cipher:
  - Encryption uses a classic alphabet, and two integers, called coefficients or keys A and B, these are the parameters of the affine function Ax+B (which is a straight line/linear equation)
- Example:
  - The English Alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ.
  - For each letter of the alphabet is associated to the value of its position in the alphabet (starting at 0).
  - **Encrypt the plaintext: "prachi", using the key: A=5, B=3, using Affine cipher.**
  - Solution:

    C =E(A, B, p)

    =(5 p +3) mod 26

Lets list out all alphabet and their respective values, as shown in below table:

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

plaintext	p	r	a	c	h	i
p	15	17	0	2	7	8
5p+3	78	88	3	13	38	43
(5p+3) mod 26	0	10	3	13	12	17
Ciphertext C	a	k	d	n	m	r

Hence C= "akdnmr" for plaintext "prachi"

- Decryption using the Affine cipher:
  - In deciphering the ciphertext, we must perform the opposite (or inverse) functions on the ciphertext to retrieve the plaintext.
  - Once again, the first step is to convert each of the ciphertext letters into their integer values.
  - The decryption function is :

D (x) = a^-1 (x - b) mod m

a^-1 : modular multiplicative inverse of a modulo m. i.e., it satisfies the equation

1 = a a^-1 mod m .

For an affine

encryption with the function

$y = A x + B$

The reciprocal/inverse decryption function is expressed

$y' = A' x + B$

The value of A' depends on A but also on the alphabet's length. If it is a classic one, it is 26 characters long.
The values of A' are then:

A	1	3	5	7	9	11	15	17	19	21	23	25
A'	1	9	21	15	3	19	7	23	11	5	17	25

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- Example:

Example: Decrypt the ciphertext: "ihhwvc" , using the key: A=5, B=3, using Affine cipher.

Solution:

p = D(A,B,C)

$= A^{-1} (C - B) \bmod 26$

where $A^{-1} = 21$, as per above table.

Ciphertext C	i	h	h	w	v	c
C	8	7	7	22	21	2
C-3	5	4	4	19	18	-1
21(C-3)	105	84	84	399	378	-21
21(C-3) mod 26	1	6	6	9	14	5
Plaintext	b	g	g	j	o	f

Hence plaintext = "bggjof" for ciphertext C = "ihhwvc"

- **Example:**
  - Suppose that an Affine cipher E(x) = (ax + b) mod 26 in ciphers **h as X** and **q as Y**.
  - Find the cipher (that is, determine a and b).
  - Solution:

    C = E(p)

    =(**k1** x p + **k2**) mod 26,

suppose that

- k1=a,
- p=x,
- k2=b.

We see that

- h → X means E(7) = 23 and
- q → Y means E(16) = 24.

That is,

- a · 7 + b ≡ 23 (mod 26) and
- a · 16 + b ≡ 24 (mod 26).

Subtracting gives 16a − 7a ≡ 1 (mod 26)

So that 9a ≡ 1 (mod 26).

Therefore,

- a = 9 −1 (mod 26) = 3.
  - Finally, we substitute a = 3 into either of the earlier equations and solve for b, i.e.,

    3 · 7 + b ≡ 23 (mod 26) implies b = 2.
  - **Hence E(x) = (3x + 2) mod 26.**

2.5 VIGENÈRE CIPHER

- Vigenere Cipher is a method of encrypting alphabetic text.
- It uses a simple form of polyalphabetic substitution.
- A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets.
- The encryption of the original text is done using the Vigenère square or Vigenère table.
- The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.
- At different points in the encryption process, the cipher uses a different alphabet from one of the rows.
- The alphabet used at each point depends on a repeating keyword.

- **Example:**
- **Input :** Plaintext : GEEKSFORGEEKS

  Keyword : AYUSH
- **Output :** Ciphertext : GCYCZFMLYLEIM

- For generating keys, the given keyword is repeated in a circular manner until it matches the length of the plain text.
- The keyword "AYUSH" generates the key "AYUSHAYUSHAYU"
- The plain text is then encrypted using the process explained below.

- **Encryption:**
  - The first letter of the plaintext, G is paired with A, the first letter of the key.
  - So use row G and column A of the Vigenère square, namely G. Similarly, for the second letter of the plaintext, the second letter of the key is used, the letter at row E, and column Y is C.
  - The rest of the plaintext is enciphered in a similar fashion.

- **Decryption:**
  - Decryption is performed by going to the row in the table corresponding to the key, finding the position of the ciphertext letter in this row, and then using the column's label as the plaintext.
  - For example, in row A (from AYUSH), the ciphertext G appears in column G, which is the first plaintext letter.

- Next, we go to row Y (from AYUSH), locate the ciphertext C which is found in column E, thus E is the second plaintext letter.
- A more easy implementation could be to visualize Vigenère algebraically by converting [A-Z] into numbers [0–25].

- **Encryption**
- The plaintext(P) and key(K) are added modulo 26.
- Ei = (Pi + Ki) mod 26

- **Decryption**
- Di = (Ei - Ki + 26) mod 26
- Note: Di denotes the offset of the i-th character of the plaintext. Like offset of A is 0 and of B is 1 and so on.

Vigenere Cipher Table

- Its table consists of all the alphabets written 26 times in different rows.
- Each alphabet in every subsequent row and column is shifted cyclically to the left.
- This generates 26 Caesar Ciphers.

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

Vigenere Cipher Encoder:

- The first step in the autokey method is to decide on a priming key.
- Both sender and receiver have to agree on this key.
- The priming key is the single alphabet that is added to the beginning of messages to help make the key.
- The sender encrypts the message starting with writing the first letter of the plaintext on one line and the priming key under it.
- The rest of the plaintext is written as is, shifted one place to the right.

Plaintext	J	A	V	A
Key	R	J	A	V
Ciphertext	A	J	V	V

The steps involved are as follows:

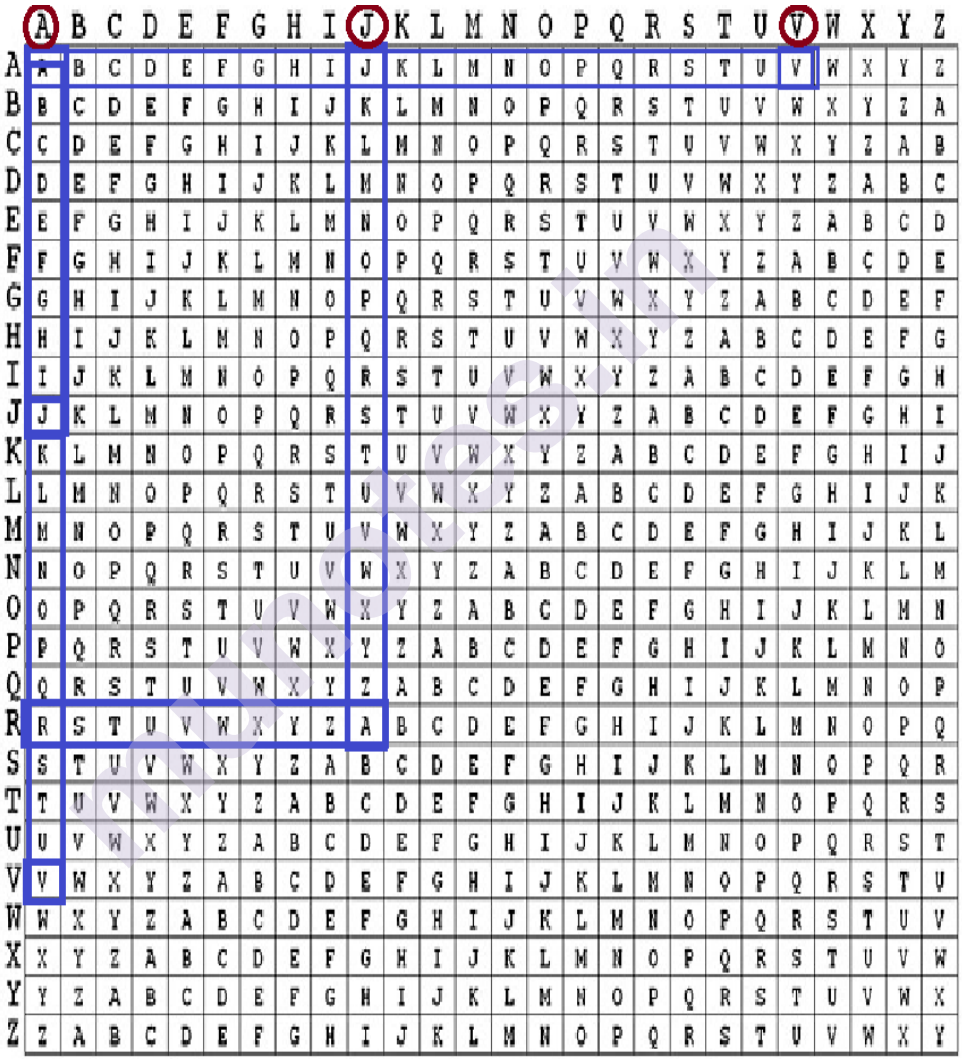
- Write the plaintext.
- Use the plaintext and the key letter to select a row and a column in the Vigenere table.
- The first letter of the plaintext is the first row and the key is the first column.
- For example,
  - if the plaintext is 'JAVA' and
  - the key is R,
  - then the first row will be the one that starts with J, and
  - the column will be the one that starts with R.
- The first letter of the ciphertext will be the letter where the first row and column intersect.
  - In the case of our example that will be the letter A.

Vigenere Cipher Decoder

The below-mentioned steps are followed to decipher the ciphertext:

- The first letter is selected using the priming key.
- The first letter of the ciphertext is located in this row.
- The first letter of the plaintext will be the letter where the first row and column intersect.
- This process is followed until the entire ciphertext is deciphered.

Key	R	J	A	V
Ciphertext	A	J	V	V
Plaintext	J	A	V	A



2.6 VERMIN CIPHER

Vernam Cipher is a method of encrypting alphabetic text.

It is one of the Substitution techniques for converting plain text into cipher text.

In this mechanism we assign a number to each character of the Plain-Text,

like (a = 0, b = 1, c = 2, … z = 25).

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Method to take key:

In the Vernam cipher algorithm, we take a key to encrypt the plain text whose length should be equal to the length of the plain text.

Encryption Algorithm:

- Assign a number to each character of the plain-text and the key according to alphabetical order.
- Bitwise XOR both the number (Corresponding plain-text character number and Key character number).
- Subtract the number from 26 if the resulting number is greater than or equal to 26, if it isn't then leave it.

Plaintext	J	A	V	A
The Key According To Alphabetical Order	9	0	21	0
Key	B	O	O	K
The Key According To Alphabetical Order	1	14	14	10
Bitwise XOR OPERATION	9 = 1001 1 = 0001 Bitwise XOR Result: J ⊕ B => 9 ⊕ 1 1000 = 8	0 = 0000 14 = 1110 Bitwise XOR Result: A ⊕ O =>0 ⊕ 14 10100 = 20	21 = 10101 14 = 1110 Bitwise XOR Result: V ⊕ O=>21⊕14 110101 = 53 53-26 = 27 27-26 = 1	0 = 0000 10 = 1010 Bitwise XOR Result: A ⊕ K => 0⊕10 10000 = 16
Ciphertext	I	U	B	Q

- If the resulting number is greater than 26, subtract 26 from it.
- Then convert the Cipher-Text character number to the Cipher-Text character.
- Similarly, do the same for the other corresponding characters,
- New Cipher-Text is after getting the corresponding character from the resulting number IS : “IUBQ”

2.7 HILL CIPHER:

- The Hill Cipher method was invented and developed in 1929 by Lester S. Hill.
- Hill Cipher uses multiple mathematical methods thus, figuring several primary methods in classical cryptography.
- Hill cipher is a polygraphic substitution cipher based on linear algebra.Each letter is represented by a number modulo 26.
- Often the simple scheme A = 0, B = 1, …, Z = 25 is used.
- To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26.
- To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.
- The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

Example

- As an example of the Hill Cipher technique, let’s encrypt the text, ‘SEA’, and, later, decrypt the resulting ciphertext.
- This will help us understand how the Hill Cipher works.
- To keep the example simple, here is a straightforward substitution scheme with the letter A mapped to 0, B mapped to 1, and so on and so forth.
- **Hill Cipher Encryption**
    - We have to encrypt the message ‘SEA’ (n=3).
    - The key is ‘SDYZKABSQ’, which in the form of an n x n matrix.:
    - The enciphered vector is given as:

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

$$= \begin{bmatrix} S & D & Y \\ Z & K & A \\ B & S & Q \end{bmatrix} \begin{bmatrix} S \\ E \\ A \end{bmatrix}$$

$$= \begin{bmatrix} 18 & 3 & 24 \\ 25 & 10 & 0 \\ 1 & 18 & 16 \end{bmatrix} \begin{bmatrix} 18 \\ 4 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 336 \\ 490 \\ 90 \end{bmatrix}$$

Now calculate mod 26,

336 mod 26 = 24

490 mod 26 = 22

90 mod 26 = 12

$$= \begin{bmatrix} 24 \\ 22 \\ 12 \end{bmatrix} \text{(mod 26)}$$

And the This results in the ciphertext of "YWM".

- **Hill Cipher Decryption**
  - For the purpose of decryption, the ciphertext will have to be turned back into a vector.
  - Simply, multiply it by the inverse matrix of the key matrix.
  - The inverse of the matrix will be:
  - To encrypt the text using hill cipher, we need to perform the following operation.

D(K, C) = (K-1 * C) mod 26

- Where ,
  - K is the key matrix and
  - C is the ciphertext in vector form.
  - Matrix multiplication of inverse of key matrix K and ciphertext C generates the decrypted plain text.

Advantages of Hill Cipher

- It perfectly conceals single-letter frequencies
- 3×3 Hill Ciphers are extremely effective when it comes to hiding both single-letter and two-letter frequency information.
- It is highly strong against attacks made on ciphertext except if the attack is through a known plaintext.

2.8 PERMUTATION CIPHER

- There are two common techniques used to construct ciphers:
  - substitution
    - Example:
- Affine ciphers,
- keyword ciphers,
- the Hill cipher,
- the Playfair cipher, and
- the Vigenère cipher
  - permutation.
- Substitution replaces plaintext letters or strings of letters by letters or numbers or symbols.
- Permutation uses the plaintext message letters but rearranges their order.
- Frequency analysis is a tool to identify the substitutions.
- Frequency analysis of a ciphertext message that has been enciphered using a permutation cipher reveals only plaintext frequencies.
- Permutation technique is one where the plaintext remains the same, but the order of characters is shuffled.
- One simple technique for encryption is based on the mathematical notation of permutation .
- The main idea of the permutation cipher is permutation of the position of letters.
- A permutation is a bijective map from n-element set into itself.
- Thus every permutation has an inverse.
- In Permutation cipher,K consists set of all possible permutations of N symbols 0,1,2,…N-1
- For each permutation K, defineC=AP and define P=A-1C where A-1 is the inverse permutation to A.
- This can be observed as follows

$$P \xrightarrow{A} C = AP \xrightarrow{A^{-1}} P$$

P is plain text

C is cipher text

***A* is permutation**

Encryption Algorithm

Step1: given P= plaintext

Step2: construct P^1 = block of plain text

Step3: Key K= permutation (bijection)

Step4: calculate $C^1 = P^1 0K$ = block of cipher text

Step5: write C= cipher text

Decryption Algorithm

Step1: take C= cipher text

Step2: $C^1 = P^1 0K$ = block of cipher text

Step3: Key K^{-1}= permutation (bijection)

Step4: calculate $P^1 = C^1 0K^{-1}$=block of plain text

Step5: write P= plaintext

Example:

P= 'MATHEMATICS' is the plain text

$P^1 = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ M & A & TH & EMA & T & IC & S \end{pmatrix}$ is block of plain text

KEY $K = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ 6 & 9 & 18 & 032 & 7 & 45 & 10 \end{pmatrix}$

Encryption: $C^1 = P^1 0K$ $\quad C^1 = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ M & A & TH & EMA & T & IC & S \end{pmatrix} \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ 6 & 9 & 18 & 032 & 7 & 45 & 10 \end{pmatrix}$

$C^1 = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ A & C & AI & MHT & T & EM & S \end{pmatrix}$

C='ACAIMHTTEMS' is cipher text

$C^1 = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ A & C & AI & MHT & T & EM & S \end{pmatrix}$ is block of Cipher text

KEY $K^{-1} = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ 4 & 2 & 65 & 890 & 7 & 31 & 10 \end{pmatrix}$

Decryption:: $P^1 = C^1 0K$

$P^1 = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ A & C & AI & MHT & T & EM & S \end{pmatrix} \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ 4 & 2 & 65 & 890 & 7 & 31 & 10 \end{pmatrix}$

$P^1 = \begin{pmatrix} 0 & 1 & 23 & 456 & 7 & 89 & 10 \\ M & A & TH & EMA & T & IC & S \end{pmatrix}$

P= 'MATHEMATICS' is the plain text

2.9 STREAM CIPHER

- Symmetric encryption algorithms are categorized into two:
  - block and
  - stream ciphers.

- A stream cipher is an encryption technique that works byte by byte to transform plain text into code that's unreadable to anyone without the proper key.
- Stream ciphers are linear, so the same key both encrypts and decrypts messages.
- And while cracking them can be difficult, hackers have managed to do it.
- For that reason, experts feel stream ciphers aren't safe for widespread use.
- Even so, plenty of people still lean on the technology to pass information through the internet.
- **Working of stream ciphers:**
    - All cryptographic methods aim to scramble data to hide it from outsiders.
    - But unlike their counterparts, stream ciphers work on each bit of data in a message rather than chunking the message into groups and encrypting them in blocks.
    - **Stream ciphers rely on:**
        - **Plaintext:** You must have a message you'd like to encode.
        - **Keystreams**: A set of random characters replaces those in the plaintext.

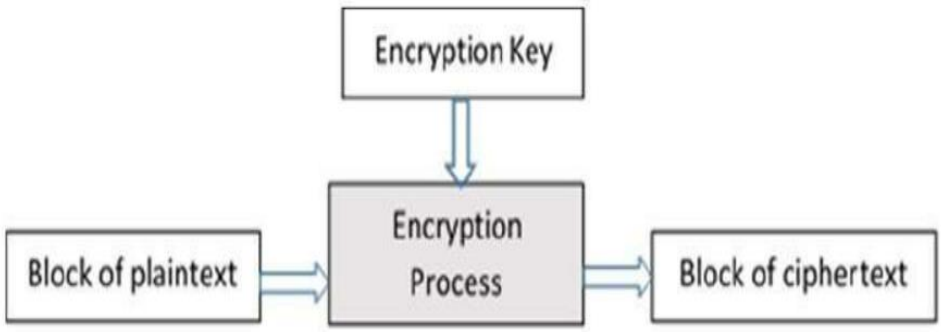
          They could be numbers, letters, or symbols.
        - **Ciphertext:** This is the encoded message.
- Generating a key is a complicated mathematical process.
- Even so, most computers can push through each step in seconds.
- Bits of plaintext enter the stream cipher, and the cipher manipulates each bit with the mathematical formula.
- The resulting text is completely scrambled, and the recipient can't read it without the proper key.
- With the right key, a recipient can push the ciphertext back through the stream cipher and transform the garbled data back to plaintext.
- **There are two main types of stream ciphers, and they each work slightly differently.**
    - **Synchronous stream ciphers:**
        - A secret key generates keystreams, and they're made independently of both the plaintext and the ciphertext.
    - **Self-synchronizing stream ciphers:**
        - They use a secret key, but they include another form of randomization to make hacking harder.

Advantages of Stream ciphers:

- **Speed**
  - This form of encryption is typically faster than others, including block ciphers.
- **Low complexity**
  - It's easy to incorporate stream ciphers into modern programs, and developers don't need complex hardware to make it happen.
- **Serial nature**
  - Some companies deal with messages written in a trickle. With their bit-by-bit processing, stream ciphers allow them to send information when it's ready rather than waiting for everything to be done.
- **Ease of use**
  - Stream ciphers are symmetrical encryption tools, so companies aren't forced to bother with public and private keys.
  - And mathematical concepts that underlie modern stream ciphers allow computers to determine the proper decryption key to use.

2.10 BLOCK CIPHERS

- Block ciphers break messages down into pieces, and then each piece moves through an encryption algorithm.
- A block cipher takes a block of plaintext bits and generates a block of ciphertext bits, generally of the same size.
- The size of the block is fixed in the given scheme.
- The choice of block size does not directly affect the strength of the encryption scheme.
- The strength of the cipher depends on the key length.



2.11 EXERCISE:

Answer the following:

1. Encrypt the plaintext: “its cool” , using the key: **k1**=5, **k2**=8, using Affine cipher.
2. Encrypt the plaintext: “affine cipher”, using the key: **k1**=5, **k2**=8, using Affine cipher.
3. Encrypt the plaintext :Plaintext: “A simple message “ Key: a = 7, b = 13 . Find Ciphertext
4. Decrypt the plaintext Ciphertext: NCDQWTQP FQ CEL NEFRMWLYN FA PTDQCN

   Key: a = 15, b = 3. Find Plaintext:
5. Encrypt the Plain-Text: RAMSWARUPK and Key: RANCHOBABA using **Vermin Cipher.**

REFERENCES :

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

http://www.ijstm.com/images/short_pdf/1456578528_427S.pdf

❄❄❄❄❄❄❄

3

CRYPTOGRAPHIC ALGORITHM MODE AND HASH FUNCTIONS

Unit Structure :

3.0 OBJECTIVE

- A basic component of many cryptographic algorithms is what is known as a hash function.
- When a hash function satisfies certain non-invertibility properties, it can be used to make many algorithms more efficient.
- In the following, we discuss the basic properties of hash functions and attacks on them.
- We also briefly discuss the random oracle model, which is a method of analyzing the security of algorithms that use hash functions.
- The focus of this chapter is on data integrity and cryptographic tools used to achieve the same.
- The chapter looks at security considerations for MACs.
- We look at a relatively recent approach known as authenticated encryption.
- We look at the use of cryptographic hash functions and MACs for pseudorandom number generation.

3.1 ALGORITHM MODES

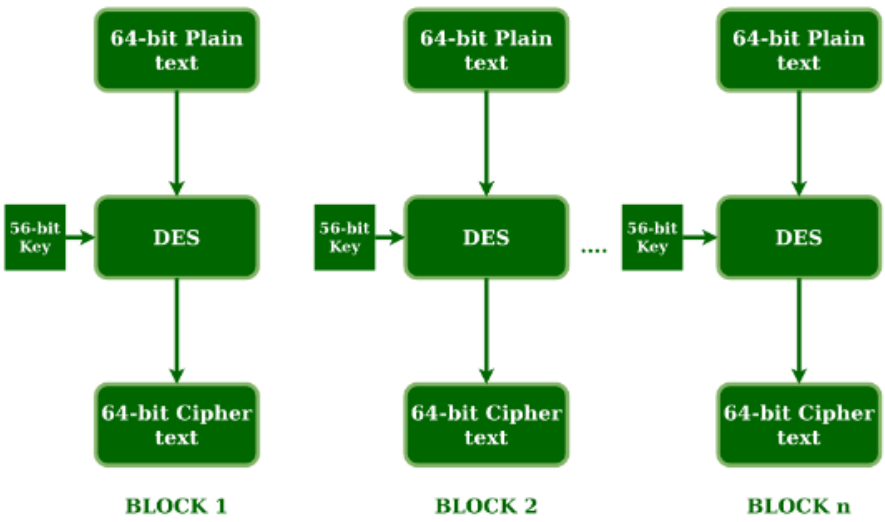
In this section we are going to see different algorithm modes. They are as follow:

- **DES**
- **Double DES**
- **Triple DES**
- **Meet-in-Middle Attack**
- **AES**
- **IDEA algorithm**

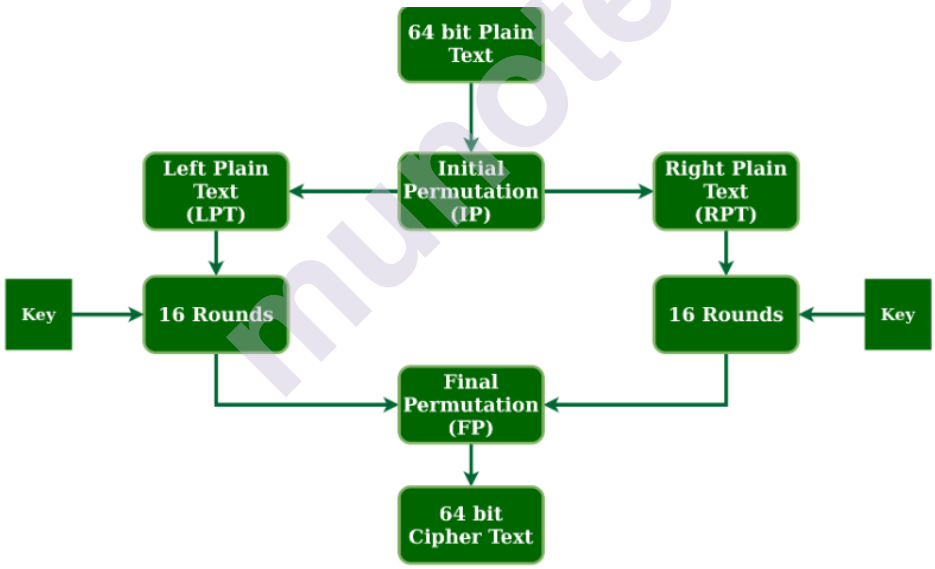
Let's discuss each one individually.

3.1.1 DES

- The DES Algorithm is a block cipher that uses symmetric keys to convert 64-bit plaintext blocks into 48-bit ciphertext blocks.
- The (DES) Data Encryption Standard Algorithm was developed by the IBM team in the 1970s. It has since been accepted by the National Institute of Standards and Technology (NSIT).
- The DES encryption algorithm uses symmetric keys, which means that the same key is used for encrypting and decrypting the data.
- Data encryption standard (DES) has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found slightly on the decline.
- DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext.
- The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits.
- The basic idea is shown in the figure:



- DES is based on the two fundamental attributes of cryptography:
  - substitution (also called confusion) and
  - transposition (also called diffusion).
- DES consists of 16 steps, each of which is called a round.
- Each round performs the steps of substitution and transposition.
- Let us now discuss the broad-level steps in DES.
  - In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
  - The initial permutation is performed on plain text.
  - Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).
  - Now each LPT and RPT go through 16 rounds of the encryption process.
  - In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
  - The result of this process produces 64-bit ciphertext.



Broad Level Steps in DES

3.1.2 Double DES

- Double DES is an encryption technique which uses two instances of DES on the same plain text.
- In both instances it uses different keys to encrypt the plain text.
- Both keys are required at the time of decryption. The 64 bit plain text goes into the first DES instance which is then converted into a 64 bit

middle text using the first key and then it goes to the second DES instance which gives 64 bit cipher text by using the second key.

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

3.1.3 Triple DES

- Triple DES is a block cipher that applies the DES algorithm thrice.
- It usually uses three different keys—k1, k2, and k3.
  - The first key, k1, is used to encrypt,
  - the second key, k2, is used to decrypt, and
  - The third key, k3, is used to encrypt again.
- The triple DES also has a variant that uses only two keys, where k1 and k3 are the same.
- Triple DES is also vulnerable to meet-in-the middle attack because of which it give total security level of 2^112 instead of using 168 bits of key.
- The block collision attack can also be done because of short block size and using the same key to encrypt a large size of text. It is also vulnerable to sweet32 attacks.
- **DES Algorithm Steps**
- Let us take a look at the steps involved in the DES algorithm:
  - The initial permutation (IP) function receives the 64-bit plaintext block.
  - The IP is performed on plaintext.
  - The IP then makes two halves of the block that has been permuted.
  - The two halves are known as left plan text (LPT) and right text (RPT).
  - All LPTs and RPTs are encrypted 16 times.
  - The LPT and RPT are joined, and then the final permutation (FP) is performed on this block.
  - The 64-bit ciphertext is now ready.
- **In the encryption process (step 4), there are five stages:**
  - Key transformation
  - Expansion permutation
  - S-Box permutation
  - P-Box permutation
  - XOR, and swap

- **In the decryption process, the same algorithm is used with the order of the 16 keys reversed.**

DES Modes of Operation

The following are the different DES modes of operation to choose from:

- **Electronic Codebook (ECB):**
  - In this mode, each block of 64-bits is independently encrypted and decrypted.
- **Cipher Block Chaining (CBC):**
  - In this mode, each block of 64-bits is dependent on the one before it.
  - It uses an initialization vector (IV).
- **Cipher Feedback (CFB):**
  - In this mode, the previous ciphertext is used as the input for the encryption algorithm.
  - This produces a pseudorandom output.
  - This output is then XORed along with the plaintext. This creates the next ciphertext unit.
- **Output Feedback (OFB):**
  - This mode is like CFB, except for the fact that the input for the encryption algorithm is the output of the previous DES.
- **Counter (CTR):**
  - In this mode, every block of plaintext gets XORed with a counter that has been encrypted. The counter is incremented for every next block.

Applications of DES Algorithm

- The DES algorithm is used whenever a not-very-strong encryption is needed.
- It can be used in random number generators or even as a permutation generator.
- One of the most important practical applications of the DES algorithm is to create triple DES legacy systems with three keys.

Advantages of DES Algorithm

- The algorithm has been in use since 1977.
- Technically, no weaknesses have been found in the algorithm.
- Brute force attacks are still the most efficient attacks against the DES algorithm.
- DES is the standard set by the US Government.

- The government recertifies DES every five years, and has to ask for its replacement if the need arises.
- The American National Standards Institute (ANSI) and International Organization for Standardization (ISO) have declared DES as a standard as well.
- This means that the algorithm is open to the public—to learn and implement.
- DES was designed for hardware; it is fast in hardware, but only relatively fast in software.

Disadvantages of DES Algorithm

- Probably the biggest disadvantage of the DES algorithm is the key size of 56-bit.
- There are chips available that can encrypt and decrypt a million DES operations in a second.
- A DES cracking machine that can search all the keys in about seven hours is available for $1 million.
- DES can be implemented quickly on hardware.
- But since it was not designed for software, it is relatively slow on it.
- It has become easier to break the encrypted code in DES as the technology is steadily improving.
- Nowadays, AES is preferred over DES.
- DES uses a single key for encryption as well as decryption as it is a type of symmetric encryption technique.
- In case that one key is lost, we will not be able to receive decipherable data at all.

3.1.4 Meet-in-Middle Attack

- The meet-in-the-middle attack is one of the types of known plaintext attacks.
- The intruder has to know some parts of plaintext and their ciphertexts.
- Using meet-in-the-middle attacks it is possible to break ciphers, which have two or more secret keys for multiple encryption using the same algorithm.
- For example, the 3DES cipher works in this way. Meet-in-the-middle attack was first presented by Diffie and Hellman for cryptanalysis of the DES algorithm.
- A cipher, which is to be broken using a meet-in-the-middle attack, can be defined as two algorithms, one for encryption and one for decryption.
- Each of them contains two simpler algorithms:
    - **C = Eb(kb, Ea(ka, P))**

- **P = Da(ka, Db(kb, C))**
- Where:

C is a ciphertext,

P is a plaintext,

E is an algorithm for encryption,

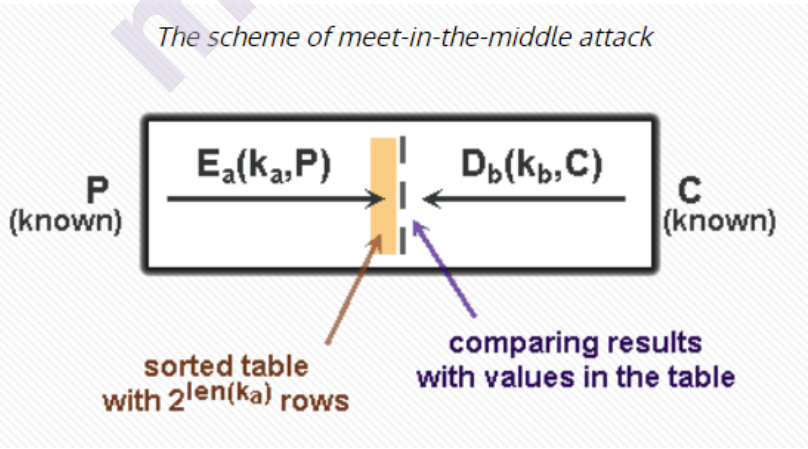
D is an algorithm for decryption,

ka and kb are two secret keys

- A following equation can be written for the cipher defined above:

  Db(kb, C) = Ea(ka, P)

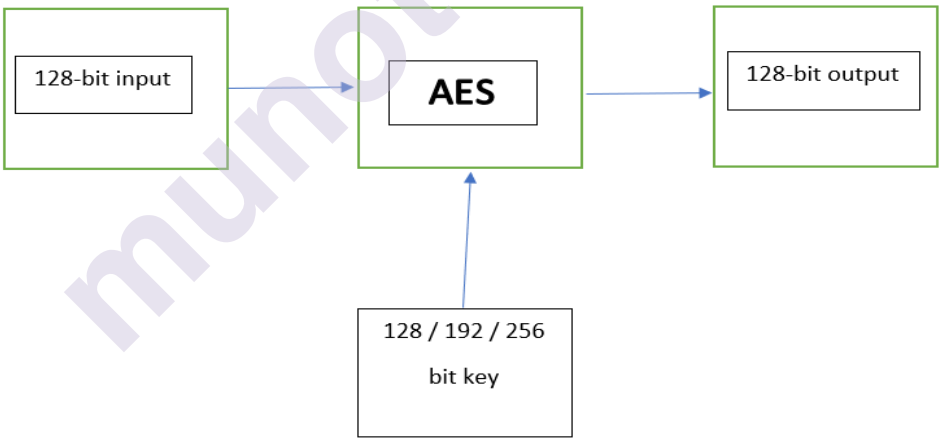
- **The first step of the attack is to create a table with all possible values for one side of the equation.**
  - One should calculate all possible ciphertexts of the known plaintext P created using the first secret key, so Ea(ka,P).
  - A number of rows in the table is equal to a number of possible secret keys.
  - It is good idea to sort the received table based on received ciphertexts Ea(ka,P), in order to simplify its further searching.
- **The second step of the attack is to calculate values of Db(kb,C) for the second side of the equation.**
  - One should compare them with the values of the first side of the equation, computed earlier and stored in the table.
  - The intruder searches for a pair of secret keys ka and kb, for which the value Ea(ka,P) found in the table and the just calculated value Db(kb,C) are the same.



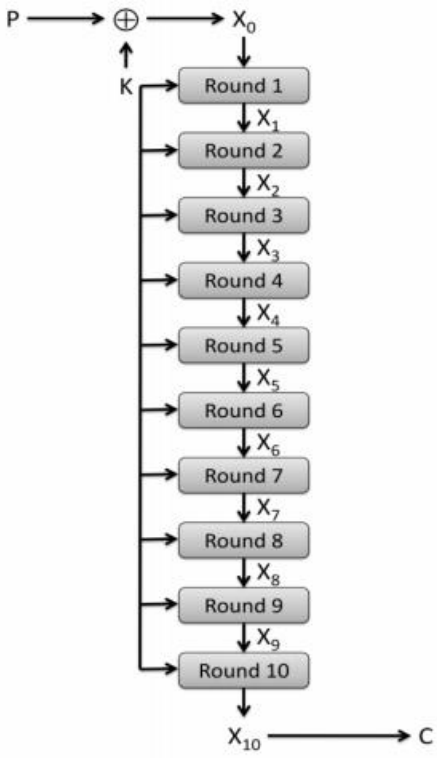
3.1.5 AES

- 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.

- As triple-DES was found to be slow, AES was created and is six times faster than the triple DES.
- It is one of the most widely used symmetric block cipher algorithms used nowadays.
- It works on bytes rather than bits.
- AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.
- 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.



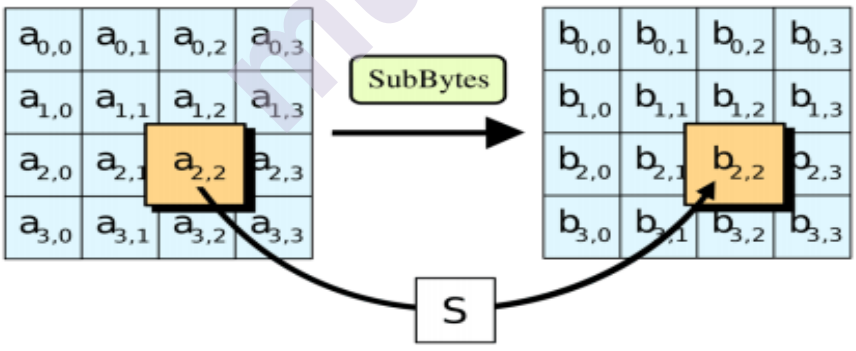
- The AES algorithm uses a substitution-permutation, or SP network, with multiple rounds to produce ciphertext.
- The number of rounds depends on the key size being used.
- A 128-bit key size dictates ten rounds, a 192-bit key size dictates 12 rounds, and a 256-bit key size has 14 rounds.
- Each of these rounds requires a round key, but since only one key is inputted into the algorithm, this key needs to be expanded to get keys for each round, including round 0.



- Steps in each round
- Each round in the algorithm consists of four steps.

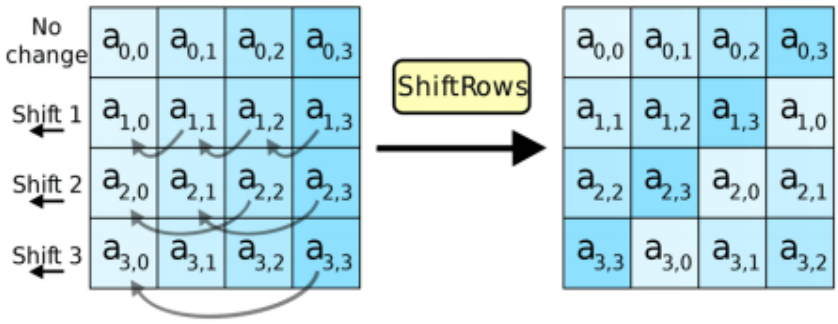
1. Substitution of the bytes

- In the first step, the bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).



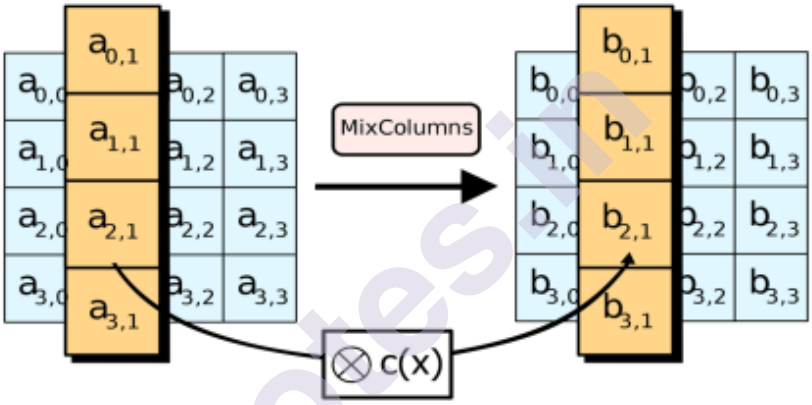
2. Shifting the rows

- Next comes the permutation step. In this step, all rows except the first are shifted by one, as shown below.



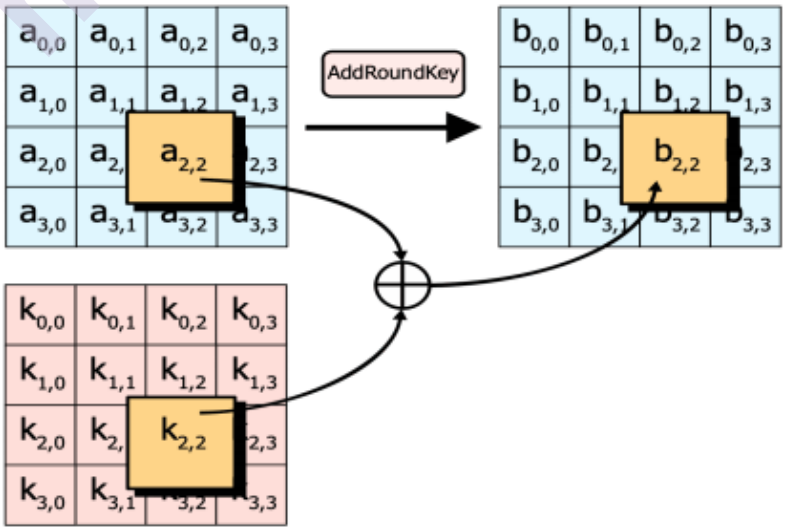
3. Mixing the columns

- In the third step, the Hill cipher is used to jumble up the message more by mixing the block's columns.



4. Adding the round key

- In the final step, the message is XORed with the respective round key.



- When done repeatedly, these steps ensure that the final ciphertext is secure.

AES modes

The AES algorithm has five modes of operation, namely:

- Electronic Code Book (ECB) mode
- Cipher Block Chaining (CBC) mode
- Cipher Feedback (CFB) mode
- Output Feedback (OFB) mode
- Counter (CTR) mode

Difference Between AES and DES Algorithms

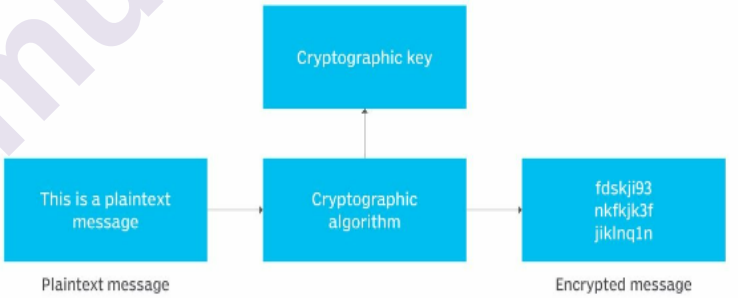
AES and DES are both symmetric ciphers. So, what is the difference between them? Let us find out.

Parameter	**AES**	**DES**
Meaning	AES stands for advanced encryption standard.	DES stands for data encryption standard.
Key Length	The key length can be 128 bits, 192 bits, or 256 bits.	The key length is 56 bits.
Rounds of Operations	The rounds of operations per key length are as follows: 128 bits: 10 192 bits: 12 256 bits: 14	There are 16 identical rounds of operations.
Network	AES is based on a substitution and permutation network.	DES is based on the Feistel network.
Security	AES is considered the standard encryption algorithm in the world and is more secure than DES.	DES is considered to be a weak encryption algorithm; triple DES is a more secure encryption algorithm.
Rounds	Key Addition, Mix Column, Byte Substitution, and Shift Row.	Substitution, XOR Operation, Permutation, and Expansion.
Size	AES can encrypt plaintext of 128 bits.	DES can encrypt plaintext of 64 bits.
Derived from	AES was derived from the Square Cipher.	DES was derived from the Lucifer Cipher.

Designed by	AES was designed by Vincent Rijmen and Joan Daemen.	DES was designed by IBM.
Known Attacks	There are no known attacks for AES.	Brute force attacks, differential cryptanalysis, and linear cryptanalysis.

3.1.6 IDEA algorithm

- The Simplified International Data Encryption Algorithm (IDEA) is a symmetric key block cipher that:
  - uses a fixed-length plaintext of 16 bits and
  - encrypts them in 4 chunks of 4 bits each
  - to produce 16 bits ciphertext.
  - The length of the key used is 32 bits.
  - The key is also divided into 8 blocks of 4 bits each.
- This algorithm involves a series of 4 identical complete rounds and 1 half-round.
- Each complete round involves a series of 14 steps that includes operations like:
  - Bitwise XOR
  - Addition modulo (2^4)
  - Multiplication modulo (2^4) +1



3.2 HASH FUNCTIONS AND DATA INTEGRITY

- To convert a numerical input value into another compressed numerical value a mathematical function used is known as a hash function.
- Values returned by a hash function are called message digest or simply hash values.

Features of Hash Functions

The typical features of hash functions are −

- **Fixed Length Output (Hash Value)**
  - Hash function converts data of arbitrary length to a fixed length.
  - This process is often referred to as hashing the data.
  - In general, the hash is much smaller than the input data, hence hash functions are sometimes called compression functions.
  - Since a hash is a smaller representation of a larger data, it is also referred to as a digest.
  - Hash function with n bit output is referred to as an n-bit hash function.
  - Popular hash functions generate values between 160 and 512 bits.
- **Efficiency of Operation**
  - Generally for any hash function h with input x, computation of h(x) is a fast operation.
  - Computationally hash functions are much faster than a symmetric encryption.

Properties of Hash Functions

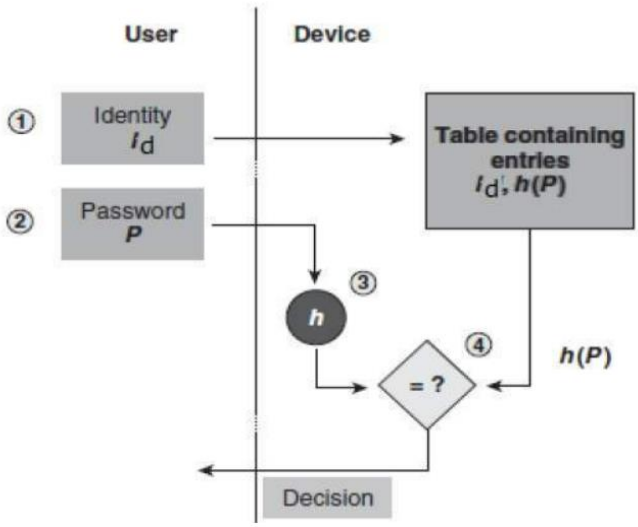
- In order to be an effective cryptographic tool, the hash function is desired to possess following properties −
- **Pre-Image Resistance**
  - This property means that it should be computationally hard to reverse a hash function.
  - In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
  - This property protects against an attacker who only has a hash value and is trying to find the input.
- **Second Pre-Image Resistance**
  - This property means given an input and its hash, it should be hard to find a different input with the same hash.
  - In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute a different value as legitimate value in place of original input value.

- **Collision Resistance**
  - This property means it should be hard to find two different inputs of any length that result in the same hash.
  - This property is also referred to as a collision free hash function.
  - In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, a hash function is a compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.
  - Also, if a hash function is collision-resistant then it is second pre-image resistant.

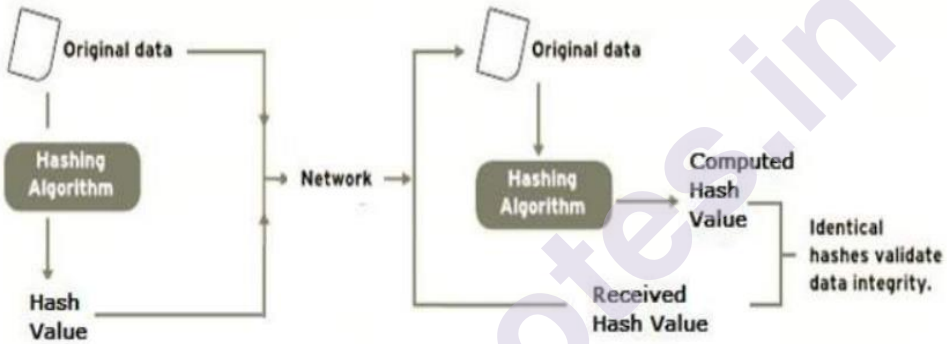
Applications of Hash Functions

There are two direct applications of hash function based on its cryptographic properties.

- **Password Storage**
  - Hash functions provide protection to password storage.
  - Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.
  - The Password file consists of a table of pairs which are in the form (user id, h(P)).
  - An intruder can only see the hashes of passwords, even if he accessed the password.
  - He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.
  - The process of logon is depicted in the following illustration −



- **Data Integrity Check**
  - Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files.
  - This application provides assurance to the user about correctness of the data.
  - The integrity check helps the user to detect any changes made to original file.
  - It however, does not provide any assurance about originality.
  - The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver.
  - This integrity check application is useful only if the user is sure about the originality of file.
  - The process is depicted in the following illustration −



DATA INTEGRITY

- Data integrity is a concept and process that ensures the accuracy, completeness, consistency (validity), and validity of an organization's data.
- The importance of data integrity increases as data volumes continue to increase exponentially.
- Major organizations are becoming more reliant on data integration and the ability to accurately interpret information to predict consumer behavior, assess market activity, and mitigate potential data security risks.

Threats to Data Integrity

- When sensitive information is exchanged, the receiver must have the assurance that the message has come intact from the intended sender and is not modified inadvertently or otherwise.
- There are two different types of data integrity threats,
  - **Passive threat**
    - This type of threats exists due to accidental changes in data.

- These data errors are likely to occur due to noise in a communication channel.
- Also, the data may get corrupted while the file is stored on a disk.
- Error-correcting codes and simple checksums like Cyclic Redundancy Checks (CRCs) are used to detect the loss of data integrity.
- In these techniques, a digest of data is computed mathematically and appended to the data.

- **Active threat**

- In this type of threats, an attacker can manipulate the data with malicious intent.
  - At simplest level, if data is without digest, it can be modified without detection.
  - The system can use techniques of appending CRC to data for detecting any active modification.
  - At a higher level of threat, attackers may modify data and try to derive new digest for modified data from existing digest.
  - This is possible if the digest is computed using simple mechanisms such as CRC.
  - Security mechanisms such as Hash functions are used to tackle the active modification threats.

Data integrity may be compromised through:

- **Human error,**
  - It can be malicious or unintentional
  - Human error offers a major data integrity risk to organizations.
  - This is often caused by,
    - users entering duplicate or incorrect data,
    - deleting data,
    - not following protocols, or
    - making mistakes with procedures put in place to protect information.
- **Transfer errors,**
  - If data is unable to transfer between database locations, it means there has been a transfer error.
  - These occur when pieces of data are in the destination table but not the source table of a relational database.
  - It including unintended alterations or data compromise during transfer from one device to another

- **Bugs, viruses/malware, hacking, and other cyber threats**
  - Hackers threaten organizations' data integrity by using software, such as malware, spyware, and viruses, to attack computers in an attempt to steal, amend, or delete user data.
- **Compromised hardware, such as a device or disk crash**
  - Compromised hardware can result in device or server crashes and other computer failures and malfunctions.
  - Consequently, data can be rendered incompletely or incorrectly, data access removed or limited, or data can become hard for users to work with.
- **Physical compromise to devices**
  - To compromise a device by Physically opening the device to gain access to its component parts.
  - Connecting a lead to access a physical port on the device.

Ensuring Data Integrity

Preventing the above issues and risks is reliant on preserving data integrity through processes such as:

- **Validate Input**
  - Data entry must be validated and verified to ensure its accuracy.
  - Validating input is important when data is provided by known and unknown sources, such as applications, end-users, and malicious users.
- **Remove Duplicate Data**
  - It is important to ensure that sensitive data stored in secure databases cannot be duplicated onto publicly available documents, emails, folders, or spreadsheets.
  - Removing duplicated data can help prevent unauthorized access to business-critical data or personally identifiable information (PII).
- **Back Up Data**
  - Data backups are crucial to data security and integrity.
  - Backing up data can prevent it from being permanently lost and should be done as frequently as possible.
  - Data backups are especially important for organizations that suffer ransomware attacks, enabling them to restore recent versions of their databases and documents.
- **Access Controls**
  - Applying appropriate access controls is also important to maintaining data integrity.
  - This is reliant on implementing a least-privileged approach to data access, which ensures users are only able to access data,

documents, folders, and servers that they need to do their job successfully.

- This limits the chances of hackers being able to impersonate users and prevents unauthorized access to data.

- **Always Keep an Audit Trail**
  - In the event of a breach occurring, it is crucial that organizations are able to quickly discover the source of the event.
  - An audit trail allows businesses to track what happened and how a breach occurred, and then find the source of the attack.

3.3 SECURE HASH ALGORITHM

Secure Hash Algorithm (SHA):

- It is the most widely used hash function.
- SHA is a modified version of MD5 and used for hashing information and certificates.
- A hashing algorithm shortens the input information into a smaller form that cannot be learned by utilizing bitwise operations, modular additions, and compression functions.
- SHAs also help in revealing if an original message was transformed in any way.
- By imputing the original hash digest, a user can tell if even an individual letter has been shifted, as the hash digests will be effectively different.
- The important element of SHAs is that they are deterministic.
- This defines that considering the hash function used is known, any computer or user can regenerate the hash digest.
- The determinism of SHAs is one of main reasons that each SSL certificate on the Internet is needed to have been hashed with a SHA-2 function.
- SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.
- When weaknesses were discovered in SHA, now known as SHA-0, a revised version was issued as FIPS 180-1 in 1995 and is referred to as SHA-1.
- SHA-1 produces a hash value of 160 bits.
- There are multiple instances of these tools that were set up to support better digital security.
- The first one, SHA-0, was invented in 1993. Like its successor, SHA-1, SHA-0 features 16-bit hashing.

Types of SHA

- There are several different forms of the Secure Hashing Algorithm. The following forms of SHA are mentioned below:
  - SHA-1
  - SHA-2
  - SHA-256
  - SHA-512
  - SHA-224
  - SHA-384

Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

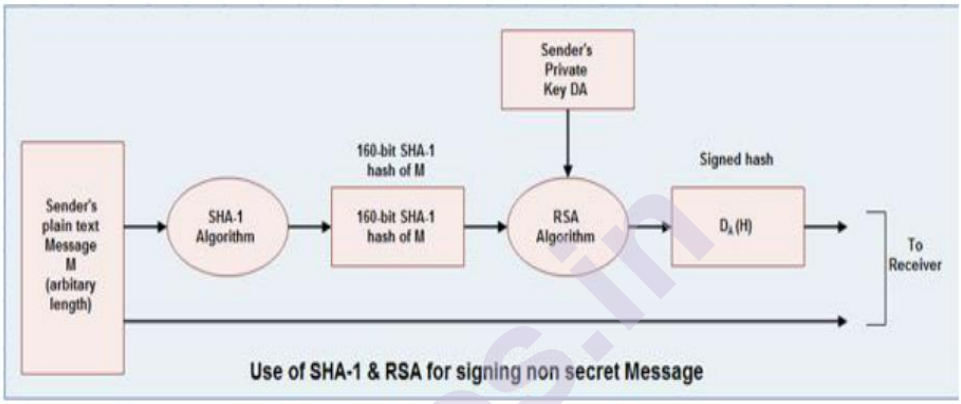
The features of the SHA algorithm are as follows:

- **Message Length:**
  - The length of the cleartext should be less than 264 bits.
  - The size needs to be in the comparison area to keep the digest as random as possible.
- **Digest Length:**
  - The length of the hash digest should be 256 bits in SHA 256 algorithm, 512 bits in SHA-512, and so on.
  - Bigger digests usually suggest significantly more calculations at the cost of speed and space.
- **Irreversible:**
  - By design, all hash functions such as the SHA 256 are irreversible.
  - You should neither get a plaintext when you have the digest beforehand nor should the digest provide its original value when you pass it through the hash function again.

Working of Secure Hash Algorithm:

Following steps are followed:

- Sender feeds a plaintext message into SHA-l algorithm and obtains a 160-bit SHA-l hash.
- Sender then signs the hash with his RSA private key and sends both the plaintext message and the signed hash to the receiver.
- After receiving the message, the receiver computes the SHA-l hash himself and also applies the sender's public key to the signed hash to obtain the original hash H.



Applications of SHA algorithm:

The SHA algorithm is being used in a lot of places, some of which are as follows:

- **Digital Signature Verification:**
  - Digital signatures follow asymmetric encryption methodology to verify the authenticity of a document/file.
  - Hash algorithms like SHA 256 go a long way in ensuring the verification of the signature.
- **Password Hashing:**
  - As discussed above, websites store user passwords in a hashed format for two benefits.
  - It helps foster a sense of privacy, and it lessens the load on the central database since all the digests are of similar size.
- **SSL Handshake:**
  - The SSL handshake is a crucial segment of the web browsing sessions, and it's done using SHA functions.
  - It consists of your web browsers and the web servers agreeing on encryption keys and hashing authentication to prepare a secure connection.

- **Integrity Checks:**
    - As discussed above, verifying file integrity has been using variants like SHA 256 algorithm and the MD5 algorithm.
    - It helps maintain the full value functionality of files and makes sure they were not altered in transit.

3.4 MESSAGE AUTHENTICATION CODE

- Message authentication is a mechanism or service used to verify the integrity of a message.
- Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.
- Symmetric encryption provides authentication among those who share the secret key.
- A message authentication code (MAC) is an algorithm that requires the use of a secret key.
- A MAC takes a variable-length message and a secret key as input and produces an authentication code.
- A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.
- One means of forming a MAC is to combine a cryptographic hash function in some fashion with a secret key.
- Another approach to constructing a MAC is to use a symmetric block cipher in such a way that it produces a fixed-length output for a variable- length input.

Message Authentication Code

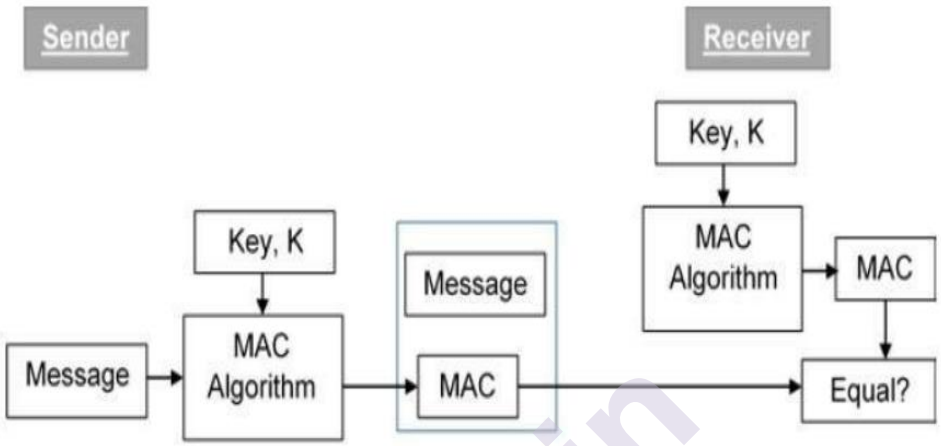
- An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC, that is appended to the message.
- This technique assumes that two communicating parties, say A and B, share a common secret key.
- When A has a message to send to B, it calculates the MAC as a function of the message and the key:

MAC=MAC(K,M)

- where
    - M= input message
    - C = MAC function
    - K= shared secret key
    - MAC = message authentication code

- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.

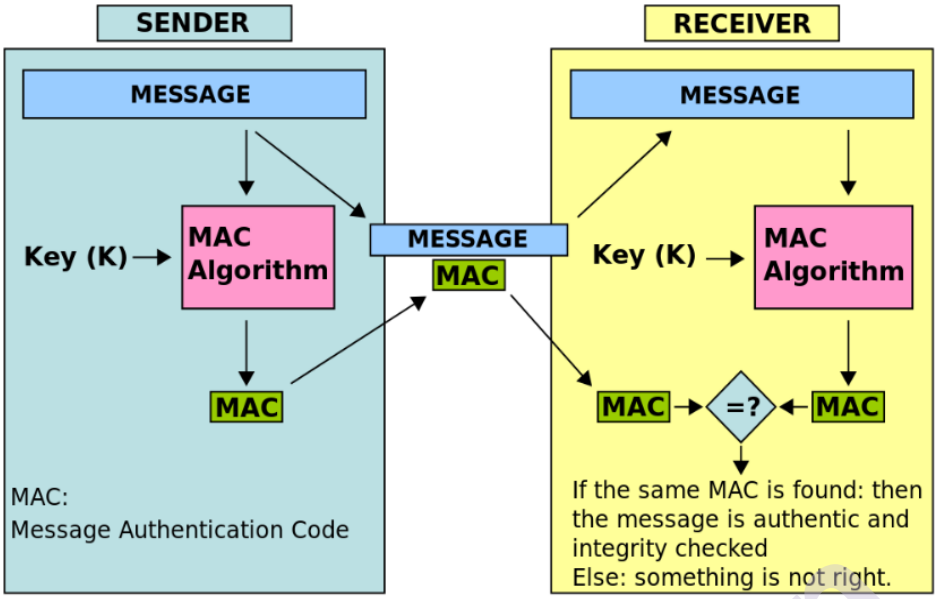
The MAC for authentication process:



Lets understand how it works:

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output.
    - The major difference between hash and MAC is that MAC uses a secret key during the compression.
- The sender forwards the message along with the MAC.
    - Here, we assume that the message is sent in the clear, as we are concerned with providing message origin authentication, not confidentiality.
    - If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender.
    - If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
    - If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified.
- As a bottom-line, a receiver safely assumes that the message is not genuine.

Example of MAC use:



- In this example, the sender of a message runs it through a MAC algorithm to produce a MAC data tag.
- The message and the MAC tag are then sent to the receiver.
- The receiver in turn runs the message portion of the transmission through the same MAC algorithm using the same key, producing a second MAC data tag.
- The receiver then compares the first MAC tag received in the transmission to the second generated MAC tag.
- If they are identical, the receiver can safely assume that the message was not altered or tampered with during transmission (data integrity).

Types of Message Authentication Codes (MACs):

- There are four different types of Message Authentication Codes:
  - Unconditionally secure
  - Hash-function based
  - Stream Cipher-based
  - Block Cipher-based

SECURITY OF MACS

- We can group attacks on MACs into two categories:
  - **Brute-force attacks:**
    - A brute-force attack on a MAC is a more difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs.
    - To attack a hash code, we can proceed in the following way.

- Given a fixed message x with n-bit hash code h=H(x) , a brute-force method of finding a collision is to pick a random bit string and check if H(y) = H(x).
- The attacker can do this repeatedly off line.
- Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the key and the tag.

○ **Cryptanalysis**

■ As with encryption algorithms and hash functions, cryptanalytic attacks on MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

■ The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.

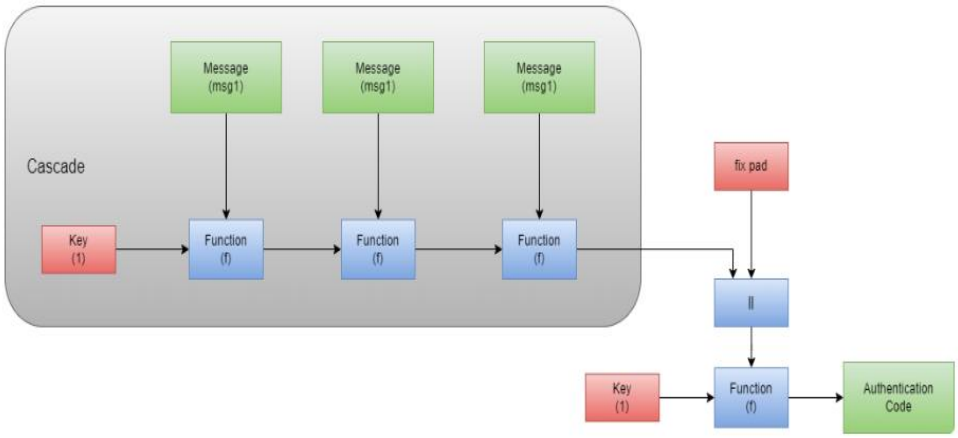
■ That is, an ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

The main differences between the two cryptographic primitives, Hash and MAC:

	Number of Inputs	Output	Algorithm	Security goals			Examples	Kind of keys	Applications
				Confidentiality	Integrity	Authentication			
Hash	A single input (the original message)	It is called a Hash or a digest, with a fixed size	Any change in message results in a different hash	No	Yes	No	SHA1, SHA2, SHA3, MD5	None	Store passwords, identify files, etc
MAC	Two inputs (the original mes- sage a secret key)	It is called a MAC or a Tag, with a size depending on the data inputs size	Any change in message or/and key results in a different MAC	No	Yes	Yes	HMAC, MAC	Symm-etric key	Financial cryptography, ETF, etc

3.5 NESTED MACS

- Nested Message Authentication Code commonly called NMAC.
- The MAC algorithm and the NMAC method are pretty similar.
- It makes use of the pseudorandom function F, which is slightly different.
- The integer returned by the function F represents the actual value of a secret key (thus, not the values of data blocks).
- **Working of NMAC or Nested MAC**

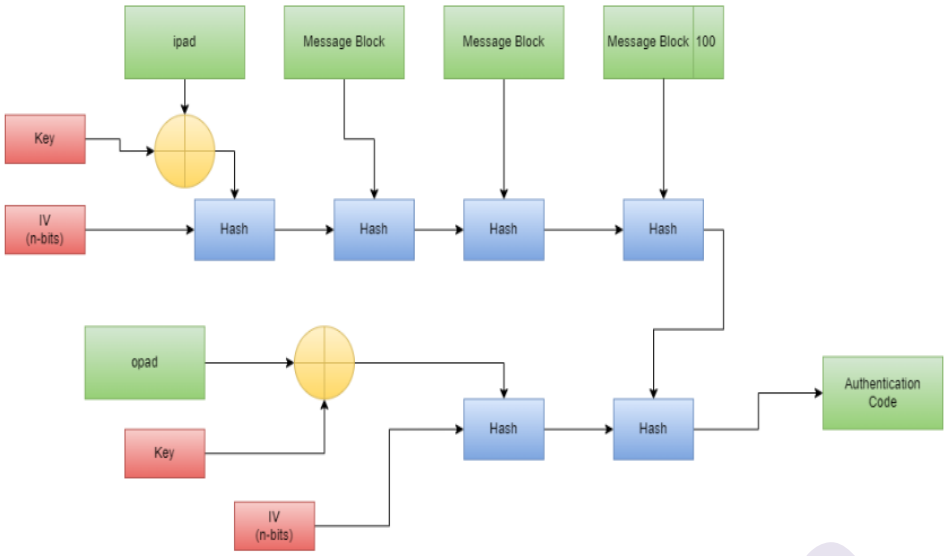


- Lets understand the working of NMAC or Nested Message Authentication Code.
  - In NMAC or Nested Authentication Code, as soon as the final data block is encrypted, the output is again encrypted using the second secret encryption key, as we do in the MAC algorithm.
  - Because the previous result of the last data block's encryption has the same number of bits as the secret key, the system adds a set of bits to ensure that the result is the same size as the data blocks.
  - This additional set is referred to as a fix pad. NMAC, or Nested Message Authentication Code, is typically used in systems where the size of secret keys is substantially more significant than the length of data blocks.
  - One final encryption is carried out in the same way that MAC protects the calculated code.
  - **The approach is known as a 'Cascade' since it does not require the final NMAC step when encrypting the subsequent blocks.**
  - An attacker can intercept the message and add any number of blocks to the message with the correctly calculated authentication code if the final algorithmic step isn't there (i.e. if encryption wasn't performed using the second key).
  - Then he may modify the message and generate a new authentication code to include it.
  - The attacker would enter the initial input for function F, which was just added, using the message's initial authentication code.
  - It would be best to change the secret key to maintain NMAC security periodically.
  - You can easily understand that a secret key is no longer secure after being sent a certain amount of messages, typically equal to the square of all secret key values.
  - The MAC algorithm and the NMAC algorithm use identical techniques for the addition of padding bits to the end of the final unfinished message block.

3.6 HMAC

- HMAC stands for Hash-Based Message Authentication Code.
- Hash-Based Message Authentication Code, or HMAC, is a type of MAC or message authentication code.
- It is created by applying a cryptographic hash function to the data (that has to be authenticated) and a private shared key.
- You can use it for data integrity and authentication like any other MAC.
- The parties participating in communication must verify the accuracy of the data.
- SFTP, HTTPS, FTPS, and many other transfer protocols use HMAC or Hash-Based Message Authentication Code.
- The cryptographic hash function can be MD-5, SHA-1, or SHA-256.
- HMACs and digital signatures use a common key and a hash function, making them almost identical.
- The keys used by HMACs and Signatures are different. HMACs generally use symmetric keys (exact copy), while Signatures use asymmetric keys (two different keys).
- There are three different kinds of authentication functions:
    - hashing operations,
    - message encryption, and
    - message authentication codes.
- **The main difference between MAC and hash MAC is the dependence on the key.**
    - In HMAC, the hash function and a key must be applied to the plain text.
    - The plain text message will be subject to the hash algorithm.
    - But first, we must compute S bits, append them to plain text, and then apply the hash function.
    - We use a key that the sender and recipient share to generate those S bits.

Working of HMAC or Hash-Based MAC:



HMAC = hashFunc(secret key + message)

- Now, let's try understanding the working of HMAC or Hash Based Message Authentication Code.
- HMACs give a shared private key to the client and the server only known to them.
- For each request, the client creates a unique hash (HMAC).
- When the client forwards or sends a request to the server, it hashes the data with a private key and includes it in the request.
- The system is secure because the message, as well as the key, is hashed separately.
- When the request is received, the server creates its HMAC.
- The system will compare both HMACS. In case they are equal, only then is the client assumed valid.

Objectives for HMAC:

- To use, without modifications, available hash functions.
- To use, hash functions that perform well in software and for which code is freely and widely available.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.

- To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

Advantages of HMAC:

- HMACs are perfect for high-performance systems like routers because they use hash functions, which can be calculated and validated quickly in contrast to public key systems.
- HMACs offer comparable greater security than digital signatures but are bulkier.
- HMACs are generally used in administrations that forbid public key systems.

Disadvantages of HMAC:

- HMACs make use of a shared key, which could result in non-repudiation.
- Attackers will easily be able to create unauthorized messages if the key of either the sender or the receiver is compromised.

Applications :

- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest.
- That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

3.7 EXERCISE

Answer the following:

1. What characteristics are needed in a secure hash function?
2. What is a message authentication code?
3. What is the difference between a message authentication code and a one-way hash
4. function?
5. In what ways can a hash value be secured so as to provide message authentication?

6. Is it necessary to recover the secret key in order to attack a MAC algorithm?

7. What characteristics are needed in a secure hash function?

https://isidore.co/CalibreLibrary/Washington,%20Lawrence%20C_/Introduction%20to%20Cryptography%20With%20Coding%20Theory%20(4971)/Introduction%20to%20Cryptography%20With%20Coding%20T%20-%20Washington,%20Lawrence%20C_.pdf

https://www.codingninjas.com/codestudio/library/nested-macs-and-hmac-in-cryptography

https://en.wikipedia.org/wiki/Message_authentication_code#An_example_of_MAC_use

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

https://intellipaat.com/blog/what-is-des-algorithm/#no8

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

❄❄❄❄❄❄❄

4

RSA CRYPTOSYSTEM

Unit Structure :

4.0 OBJECTIVES

This chapter delves into the fascinating subject of public key cryptography. Public key cryptography is also known as asymmetric cryptography, two key cryptography, and non-secret key cryptography, but we'll stay with public key cryptography for the time being.

The same key is used to encrypt and decode data in symmetric key cryptography. In public key cryptography, one key is used to encrypt and another key to decode, allowing the encryption key to be made public. This solves one of the most perplexing issues in symmetric key cryptography: how to safely transfer the symmetric key.

We'll look at the one of the most significant and extensively used public key cryptosystems in this chapter, RSA cryptosystems. We will also study few of the primality test methods to determine if a number is composite or probably prime and factorization methods.

4.1 INTRODUCTION

A public key cryptosystem is built on a one-way trap door operation. The term "one-way" refers to a function that is easy to compute in one direction but difficult (i.e., computationally infeasible) in the other. The "trap door" feature prevents an attacker from using public information to extract private information. Factoring is an example of a one-way function since it is relatively easy to produce two prime numbers p and q and compute their

product N = pq, but it is difficult to identify the factors p and q given a sufficiently big value of N.

Riya needs a key pair consisting of a public key and a matching private key to perform public key crypto. Anybody can use Riya's public key to encrypt a message intended solely for Riya's eyes, but only Riya can decode the message because, assuming Riya possesses his private key, only Bob can decrypt the message.

Inherently, public key cryptography is more mathematical than symmetric key encryption. This chapter assumes, in particular, a solid grasp of simple modular arithmetic.

4.2 RSA CRYPTOSYSTEM

This cryptosystem was one of the first. Even now, it is the most widely used cryptosystem. The technology was created by three academicians, Ron Rivest, Adi Shamir, and Len Adleman, and is hence known as the RSA cryptosystem.

We'll look at two parts of the RSA cryptosystem: key pair creation and encryption-decryption techniques.

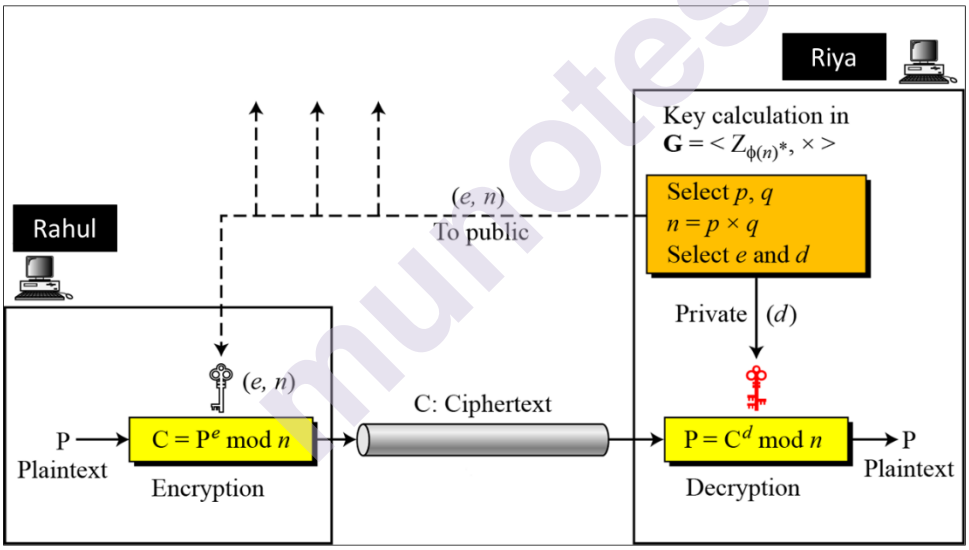


Figure 3.1: Encryption, decryption, and key generation in RSA

Generate the RSA modulus (n)

- Choose two huge prime numbers, p and q.
- Determine n=p*q. Let n to be a big value, generally at least 512 bits, for strong unbreakable encryption.

Find Derived Number (e)

- e must be larger than one and less than (p 1)(q 1).

- Apart for 1, there must be no common factor between e and (p 1)(q 1). In other words, e and (p - 1)(q - 1) are coprime integers.

Form the public key

- The RSA public key is formed by the pair of integers (n, e) and is made public.

- Curiously, even though n is part of the public key, the difficulty in factorising a huge prime number assures that the attacker cannot identify the two primes (p & q) necessary to produce n in finite time. This is an RSA strength.

Generate the private key

- Secret Key d is derived from p, q, and e. There is a distinct integer d for each given n and e.

- The inverse of e modulo (p - 1) is d. (q – 1). This indicates that d is an integer smaller than (p - 1)(q - 1) that, when multiplied by e, equals 1 modulo (p - 1). (q - 1).

- This relationship may be expressed numerically as follows:

$ed = 1 \bmod (p - 1)(q - 1)$

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

The sender encrypts a message M:

- gets the recipient's public key KU= {e, N}
- calculates : $C=M^e \bmod N$, where $0 \leq M < N$

To decrypt the ciphertext C the owner:

- uses their private key KR={d,p,q}
- computes: $M=C^d \bmod N$

It should be noted that the message M must be less than the modulus N. (block if needed)

RSA Analysis

RSA's security is dependent on the strengths of two distinct functions. The most prominent public-key cryptosystem is the RSA cryptosystem, the strength of which is predicated on the practical difficulty of factoring very big integers.

Encryption Function − It is considered as a one-way function of converting plaintext into ciphertext and it can be reversed only with the knowledge of private key d.

Key Generation − The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n. It is also a one-way function, going from p & q values to modulus n is easy but reverse is not possible.

RSA Example:

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)**
- ✓ 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 = 89e mod n**
- Thus, our Encrypted Data comes out to be 1394
- Now we will decrypt **1394**
- **Decrypted Data = cd mod n**
- Thus, our Encrypted Data comes out to be 89
- **8 = H and I = 9 i.e., "HI"**

4.3 PRIMARILY TESTING

In the RSA algorithm, primality testing is necessary for cryptography. In this asymmetric cryptography the use of prime numbers is critical, so to identify them. This crucial work is done by this testing. Basically, Primality testing functions to check whether the number provided for decryption or encryption is prime or not. Also, the function has several tests inside them like trial division. Primality testing also has the same counterpart used for composite numbers, for composite numbers to be unique the theorem is used which states that the divisor for this must be less than the square root of it. Primality testing is also considered the easiest testing in comparison to prime factorization. Also, the functions that are used for the proper functionality are the derived number, modulo number, and (p-1)(q-1).

4.4 Legendre and Jacobi Symbols

Legendre Symbol: This symbol is defined as a pair of integers a and p such that p is prime. It is denoted by (a/p) and calculated as:

= 0 if a%p = 0

(a/p) = 1 if there exists an integer k such that $k^2 = a(\text{mod } p)$

= -1 otherwise.

Euler proved that:

$(a/p) = a^{((p-1)/2)}\%p$ **Condition (i)**

Jacobian Symbol: This symbol is a generalization of the Legendre Symbol, where p is replaced by n where n is

n = p1k1 * .. * pnkn

, then the Jacobian symbol is defined as:

$(a/n) = ((a/p1)^{k1}) * ((a/p2)^{k2}) *.....* ((a/pn)^{kn})$

If n is taken as a prime number, then the Jacobian is equal to the Legendre symbol. These symbols have certain **properties** –
1) (a/n) = 0 if gcd(a,n) != 1, Hence (0/n) = 0. This is because if gcd(a,n) != 1, then there must be some prime pi such that pi divides both a and n. In that case (a/pi) = 0 [by definition of the Legendre Symbol].
2) (ab/n) = (a/n) * (b/n). It can be easily derived from the fact (ab/p) = (a/p)(b/p) (here (a/p) is the Legendry Symbol).
3) If a is even, then (a/n) = (2/n)*((a/2)/n). It can be shown that:

= 1 if n = 1 (mod 8) or n = 7 (mod 8)

(2/n) = -1 if n = 3 (mod 8) or n = 5 (mod 8)

= 0 otherwise

4) $(a/n) = (n/a) * (-1)^{((a - 1)(n - 1) / 4)}$ if a and n are both odd.

4.5 THE SOLOVAY-STRASSEN ALGORITHM

The Solovay–Strassen primality test is a probabilistic test to determine if a number is composite or probably prime.

Solovay-Strassen(n) choose a random integer a s.t. 1≤a≤n-1 x ← if x=0 then return ("n is composite") // gcd(x,n)≠1 y ← a (n-1)/2 mod n if (x=y) then return ("n is prime") // either n is a prime, or a pseudo-prime else return ("n is composite") // violates Euler's criterion If n is composite, it passes the test with at most ½ prob. Use multiple tests before accepting n as prime.

4.6 THE MILLER-RABIN ALGORITHM

Another efficient probabilistic algorithm for determining if a given number n is prime. – Write n-1 as 2 km, with m odd. – Choose a random integer a, 1 a n-1. – b ← a m mod n – if b=1 then return "n is prime" – compute b, b 2 ,b 4 ,…,b 2^(k-1) , if we find -1, return "n is prime" – return "n is composite" • A composite number pass the test with ¼ prob. • When t tests are used with independent a, a composite passes with (¼) t prob. • The test is fast, used very often in practice.

Why Miller-Rabin Algorithm works?

Claim: If the algorithm returns "n is composite", then n is not a prime. Proof: if we choose a and returns composite on n, then – a m≠1, a m≠-1, a 2m ≠ -1, a 4m ≠ -1, …, a 2^{k-1}m ≠ -1 (mod n) – suppose, for the sake of contradiction, that n is prime, – then a n-1=a 2^{k}m=1 (mod n) – then there are two square roots modulo n, 1 and -1 – then a 2^{k-1}m = a 2^{k-2}m = a 2m = a m = 1 (contradiction!) – so if n is prime, the algorithm will not return "composite

Example: Apply Miller-Rabin Algorithm using base 2 to test whether the number 341 is composite or not.

Solution: Using Miller-Rabin Algorithm, we can test the number 341 as follows −

Step1: $341 - 1 = 2^2 \times 85$. Thus p = 341, k = 2 and q = 85

Step2: x = 2 (given)

Step3: $S = x^q \bmod p$

$= 2^{85} \bmod 341 = (2^{10}) \times 2^5 \bmod 341$ 8

$= 2^{10} \bmod 341 \times 2^{13} \bmod 341$

$= 1 \times 8192 \bmod 341 = 8192 \bmod 341$

$= 8$

Step4: As 8 ≠ 1, we move to the next step.

Step5: For j = 1, $S = x^{2q} \bmod p$

$= 2^{170} \bmod 341 = (2^{20})^8 \times 2^{10} \bmod 341$

$= 2^{20} \bmod 341 \times 2^8 \bmod 341 \times 2^{10} \bmod 341$

= 1 x 256 x 1 = 256

Now, = 256 ≠ 1

and result is inconclusive

So, 341 is not a composite number.

Advantages

- This Algorithm can be used to test high numbers for primality.
- Due of its performance advantage over alternative primality tests, the Miller Rabin test will be the test of choice for numerous cryptographic applications.
- Miller Rabin tests are more dynamic than Euler and Solovay-Strassen tests, and the likelihood of failure is reduced.
- The fermat test indicates that there are too many liars for all Carmichael numbers n, and the error probability is close to one. This disadvantage is avoided in Miller Rabin.

4.7 FACTORING ALGORITHM: THE POLLARD P-1 ALGORITHM

The Pollard p-1 technique is a superior method for determining prime factors of any integer. It can quickly compute all of the unique prime factors by combining Modular Exponentiation with GCD. To summarise, Pollard's p 1 factoring technique is only efficient when one of the primes, say p, satisfies the characteristic that p 1 is a product of tiny primes (such a p 1 is called a smooth number). As a refresher, the algorithm definition in your textbook is as follows:

Algorithm : Our text's version of Pollard's p 1 Algorithm

input : N to be factored, and a bound B.
output: d, a non-trivial factor of N or failure.

a=2

j = 2

while j <= B do

$a = a^j \pmod N$

d = gcd(a – 1, N)

if 1 < d < N then

return d

end if

j = j + 1

end while

return f

As an example, let's factor the value $N = 6994241$ using Pollard's $p - 1$ algorithm:

j	a	$a^j \pmod N$	d	Comments
2	2	4	1	
3	4	64	1	
4	64	2788734	1	
5	2788734	3834705	1	
6	3834705	513770	1	
7	513770	443653	3361	Return 3361

Dividing, we find that $6994241 = 3361 \cdot 2081$. Further investigating, we find that $3361 - 1 = 2^5 \cdot 3 \cdot 5 \cdot 7$, which is a 7-smooth number (that is, contains no prime factors larger than 7).

As a hint on your homework, using this algorithm for problem 3.21, you will need to proceed to $j = 6$ for part (a), $j = 8$ for part (b), and $j = 19$ for part (c).

4.8 DIXON'S RANDOM SQUARES ALGORITHM

Dixon's Factorization is a method of integer factorization. This approach is taught in this article to discover the factors of a composite number. Dixon Factorization is founded on the well-known number theory statement that:

- If with it is likely that **gcd(x – y, n)** will be factor of **n**.

For example:

if N = 84923, by starting at 292, the first number greater than √N and counting up 5052 mod 84923 = 256 = 16^2 So (505 – 16)(505 + 16) = 0 mod 84923. Computing the greatest common divisor of 505 – 16 and N using Euclid's algorithm gives 163, which is a factor of N.

Dixon's Factorization Algorithm:

- **Step 1:** Pick a bound B and get the factor base (P) of all primes less than or equal to B.
- **Step 2:** Search for positive integer **z**, such that $z^2 \bmod(n)$ is B-Smooth.

$$z^2 \equiv \prod_{p_i \in P} p_i^{a^i} mod(n) \qquad \text{.....(1)}$$

B-Smooth: A positive integer is called B-Smooth if none of its prime factors is greater than B. **For example:**

720 has prime factorization as $2^4 * 3^2 * 5^1$ Therefore 720 is 5 smooth because none of its prime factors is greater than 5.

- **Step 3:** After establishing a significant number of these relations (typically a few more than the size of P), we combine these relations together using a linear algebra technique (e.g., Gaussian Elimination). This approach should be done until a sufficient number of smooth squares are obtained.

- **Step 4:** We get the final equation by multiplying all of these interactions together:

  $a^2 = b^2 \bmod(N)$

- **Step 5:** As a result, the factors of the preceding equation may be derived as follows:

  GCD(a - b, N), GCD(a + b, N)

Step by Step execution of the Dixon's Factorization Algorithm:

- Let's say, we want to factor N = 23449 using bound B = 7. Therefore, the factor base P = {2, 3, 5, 7}.
- Here, x = ceil(sqrt(n)) = 154. As a result, we randomly seek for numbers between 154 and N whose squares are B-Smooth.
- As previously stated, a positive integer is referred to be B-Smooth if none of its prime factors exceeds B. If we find two integers, 970 and 8621, whose squares have no prime factors bigger than 7.
- Starting here the first related squares we get are:
- 970^2 % 23499 = 2940 = $2^2 * 3 * 5 * 7^2$
- 8621^2 % 23499 = 11760 = $2^4 * 3 * 5 * 7^2$

  So, $(970 * 8621)^2 = (2^3 * 3 * 5 * 7^2)^2$ % 23499. That is: $14256^2 = 5880^2$ % 23499.
- Now, we find:

  gcd(14256 - 5880, 23449) = 131

  gcd(14256 + 5880, 23449) = 179
- Therefore, the factors are: **N = 131 * 179**

4.9 ATTACKS ON RSA

- **Plain text attacks**

The plain-text attacks are classified into three subcategories are as follows

Short message attack − It is possible that the attacker knows some blocks of plain text in the short message assault. If this assumption is correct, the attackers can experiment with encrypting each plain-text block to see if it yields the known cipher-text.

To avoid this short-message attack, it is proposed that the plain text be pad before encryption.

Cycling attack − Assume that the encrypted text was obtained by performing some kind of permutation on the plain-text. If this assumption is correct, the attacker can use the opposite operation, which is to constantly perform permutations on the known cypher text in order to obtain the original plain-text.

Unconcealed Message attack − In principle, encryption delivers cipher-text that is the same as the original plain-text in the case of some extremely restricted plain-text transmissions. If this emerges, the original plain-text communication cannot be kept private. As a result, this assault is known as an unconcealed message attack.

- **Chosen cipher attack** − The Extended Euclidean Algorithm can be used in the selected cypher attack to deduce plain text from cypher text.

- **Factorisation attack** − The whole security of RSA is based on the premise that the attacker will be unable to factor the number N into two components P and Q. If the attacker can deduce P or Q from the equation N = P x Q, he or she can deduce the private key.

  Assuming N has at least 300 decimal digits, the attacker cannot easily learn P and Q. As a result, the factorization assault fails.

- **Attack on the Encryption key** − Those who are familiar with the mathematics of RSA may believe that it is relatively simple since the public key or encryption key E might need a large amount.

  It also increases the security of RSA. As a result, if it decides to attempt and make RSA function quicker by using a tiny value for E, it may expose itself to potential assaults known as attacks on the encryption key, and it is advised that it use E as $2^{16} + 1 = 65537$ or a figure close to this amount.

- **Attacks on Decryption key** − The attack on the decryption key are as follows −

  - **Revealed decryption exponent attack** − Those who are familiar with the mathematics of RSA may believe that it is relatively simple since the public key or encryption key E might need a large amount. If the attacker can get the decryption key D, not only the cypher text created by encrypting the plain text with the associated encryption key E, but also future messages, are vulnerable. It is proposed that the sender adopt new numbers for P, Q, N, and E to avoid this exposed decryption exponent attack.

  - **Low decryption exponent attack** − It's intriguing to see how using a tiny number for decryption key D might make RSA function quicker. This can help the attacker estimate the decryption key D by launching a low decryption exponent attack.

4.10 THE RABIN CRYPTOSYSTEM

Michael Rabin designed the Rabin Cryptosystem, a public-key cryptosystem. It communicates and encrypts messages between two parties using asymmetric key encryption.

The complexity of factorization is connected to the security of the Rabin cryptosystem. It has the benefit over the others in that the issue it relies on has shown to be difficult as **integer factorization**. It also has the drawback that any Rabin function output can be created by any of four potential inputs. If each output is a ciphertext, decryption requires additional complexity to determine which of the four potential inputs was the genuine 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.

Example:

The exact key-generation procedure is as follows:

- Select two large distinct prime numbers p and q, such that p ☰ 3 mod 4, and q ☰ 3 mod 4(ex: p=139 and q=191).
- Calculate n = p*q.

  n = 139*191

  = 26549

Encryption

Algorithm:

- Obtain the public key n.

  n=26549
- Convert the message's value to ASCII. Then convert it to binary, multiply it by itself, and convert the binary value back to decimal m.

  m = "R" = 82 (R's ASCII value is 82)

  = $82_{10} = 1010010_2$;

  = 1010010 | 1010010_2; → double extend

  = 10578_{10}
- Encrypt with the following formula:

  $c = m^2 \bmod n$

  $c = 10578^2 \bmod 26549$

  c = 16598
- Send C to the intended recipient.

Decryption

- Accept the sender's C.

  c = 16598
- Using Extended Euclidean GCD, specify a and b so that

  x*p + y*q = GCD(p, q).

  x*p + y*q = 1

  139x + 191y = 1
- Using the following formula, compute r and s:

  $r = c^{(p+1)/4} \bmod p$

  $s = c^{(q+1)/4} \bmod q$

$r = 16598^{(139+1)/4} \bmod 139 = 125$

$s = 16598^{(191+1)/4} \bmod 191 = 118$

- Now, use the following formula to compute X and Y:

  $X = (x*p*r + b*q*s) \bmod p$

  $Y = (y*p*r – b*q*s) \bmod q$

  $X = 11*139*118 = 180422$

  $Y = -8 *191*125 = -191000$

- The four roots are as follows: m1=X, m2=-X, m3=Y, and m4=-Y.
- Now, convert all of them to binary and divide them in half.
- Decide which of the left and right halves is identical. One-half of that binary should be kept and converted to decimal m. Get the ASCII character that corresponds to the decimal number m. The generated character delivers the sender's message.

4.7 SUMMARY

In this chapter, we studied the most significant and extensively used public key cryptosystems in this chapter, RSA cryptosystems. We also studied few of the primality test methods to determine if a number is composite or probably prime and factorization methods.

❄❄❄❄❄❄❄

5

PUBLIC KEY CRYPTOSYSTEMS

Unit Structure :

5.1 OBJECTIVES

The objective of this chapter is to study other Public Key Cryptosystems such as The Diffie-Hellman Key Agreement, ElGamal Cryptosystem, The Pollard Rho Discrete Logarithm Algorithm, Elliptic Curves, Knapsack problem.

5.2 PUBLIC KEY CRYPTOSYSTEMS

Why might it be beneficial to make the key public? Each pair of users in a traditional symmetric key system requires a different key. Yet, with public key systems, anyone with a single public key may send a secret message to a user while the message is suitably secured from being read by an interceptor.

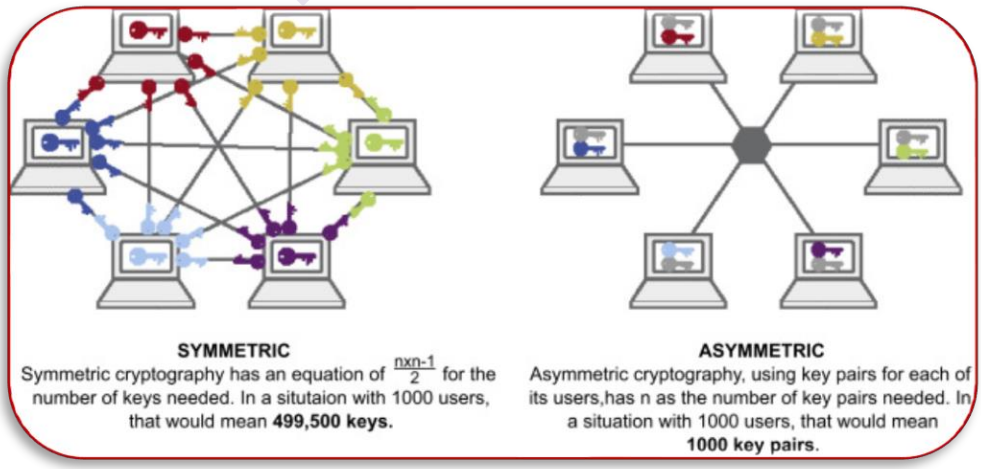


Figure 5.1: Conventional Symmetric Key System and Asymmetric Key System

Let us look at why this is the case. Remember that an n-user system requires n * (n - 1)/2 keys in general, and each user must track and remember a key for each other user with whom he or she wishes to connect. As the number of users grows, so does the number of keys, as seen in Figure 5.1. It is difficult to identify and distribute these keys. More important is the need to keep the keys that have already been released secure, because we cannot expect anyone to memorise that many keys.

On the other side, asymmetric encryption has been developed to overcome symmetrical encrypting's problems: the requirement to share a single key around which both data may be encrypted and decrypted. This innovative and safer approach uses two keys for its encryption, everything encrypted with one key can be decoded only with the second. These keys are mathematically connected and linked.

5.3 THE IDEA OF PUBLIC KEY CRYPTOGRAPHY

The most important properties of public key encryption scheme are −

- Encryption and decryption utilise different keys. This is a feature that distinguishes this system from symmetric encryption schemes.
- Each recipient has a unique decryption key, also known as his private key.
- The receiver must disclose an encryption key, known as his public key.
- With this approach, some guarantee of the validity of a public key is required to avoid spoofing by an attacker acting as the receiver. In general, this form of cryptosystem requires a trusted third party that confirms that a certain public key belongs to just one person or institution.
- The encryption technique is complicated enough that an attacker will be unable to deduce the plaintext from the ciphertext and the encryption (public) key.
- Although private and public keys are theoretically connected, calculating the private key from the public key is not possible. In reality, defining a connection between two keys is an intelligent element of every public-key cryptosystem.

The Concept is represented as:

1. $E_k(D_k(m))=m$ and $D_k(E_k(m))=m$ for every message m in M, the set of possible messages, every key k in K, the set of possible keys
2. For every m and every k, then values of $E_k(m)$ and $D_k(m)$ are easy to compute
3. For every k, if someone knows only the function E_k, it is computationally infeasible to find an algorithm to compute D_k
4. Given k, it's easy to find the functions E_k and D_k

5.4 THE DIFFIE-HELLMAN KEY AGREEMENT

The Diffie-Hellman key agreement is a sort of digital encryption in which two parties safely exchange cryptographic keys over a public channel without their conversations being broadcast over the internet. To encrypt and decode their messages, both sides employ symmetric cryptography. In 1976, Whitfield Diffie and Martin Hellman published one of the first practical applications of public key cryptography.

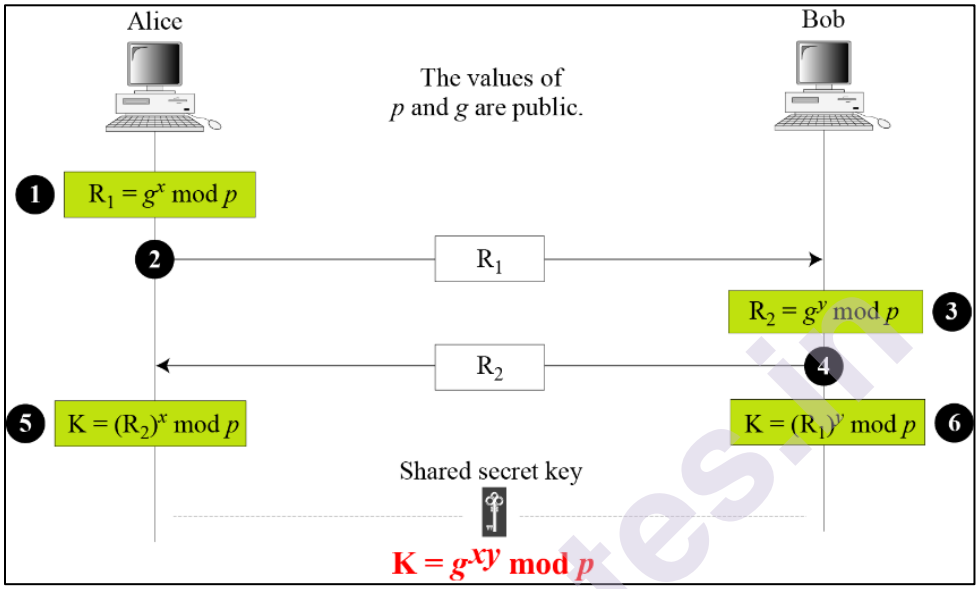


Figure 5.2: Diffie-Hellman method

- p: large prime number
- g: a generator number in the range $0 < g < p-1$.

1. Alice will then pick a secret number x in the range $0 < x < p-2$ and calculate $R_1 = g^x \bmod p$.
2. Alice will send X over to Bob.
3. Bob will pick a secret number y in the range $0 < y < p-2$ and calculate $R_2 = g^y \bmod p$.
4. Bob will send R_2 over to Alice.
5. This way, Alice and Bob will successfully exchange the necessary parameters to calculate the shared key. Alice calculates the key as $K1 = R_2^x \bmod p$. Bob calculates the key as $K2 = R_1^y \bmod p$. Alice and Bob have calculated the same key $K = g^{xy} \bmod p$ since K1 = K2.

The symmetric (shared) key in the Diffie-Hellman method is $K = g^{xy} \bmod p$.

Example:

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that g = 7 and p = 23. The steps are as follows:

1. ***Alice chooses x = 3 and calculates R_1 = 73 mod 23 = 21.***
2. ***Bob chooses y = 6 and calculates R_2 = 76 mod 23 = 4.***
3. ***Alice sends the number 21 to Bob.***
4. ***Bob sends the number 4 to Alice.***
5. ***Alice calculates the symmetric key K = 43 mod 23 = 18.***
6. ***Bob calculates the symmetric key K = 216 mod 23 = 18.***
7. ***The value of K is the same for both Alice and Bob; g^{xy} mod p = 718 mod 35 = 18.***

5.5 ELGAMAL CRYPTOSYSTEM

El Gamal created another public key algorithm in 1984 [ELG85, ELG86]. While this technique is not generally utilised directly, it is very important in the National Institute of Standards and Technology's Digital Signature Standard (DSS) [NIS92b, NIS94] (NIST). The difficulty of calculating discrete logarithms over finite fields is used in this approach. It is comparable to RSA in that it is based on arithmetic in finite fields, as is RSA.

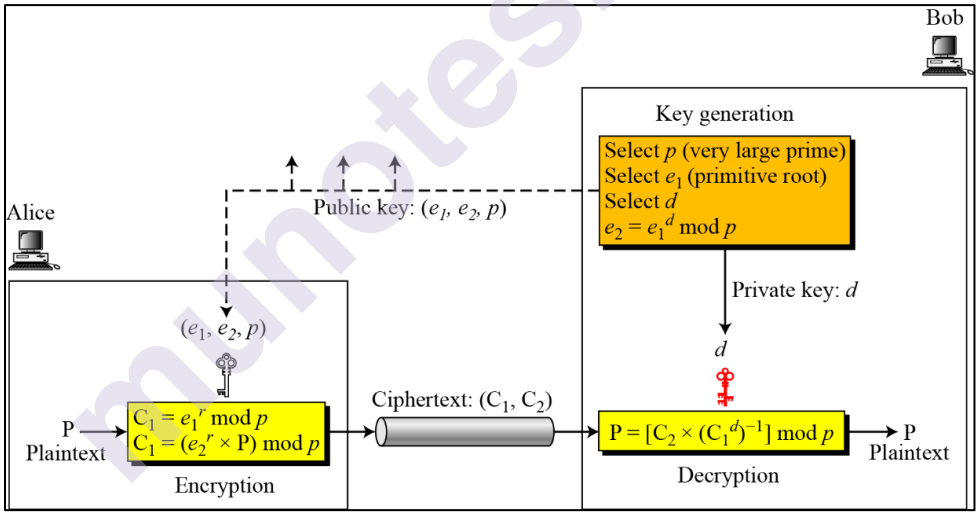


Figure 5.3: Key generation, encryption, and decryption in ElGamal

```
ElGamal_Key_Generation
{
   Select a large prime p
   Select d to be a member of the group G = < Zp*, × > such that 1 ≤ d ≤ p − 2
   Select e1 to be a primitive root in the group G = < Zp*, × >
   e2 ← e1^d mod p
   Public_key ← (e1, e2, p)          // To be announced publicly
   Private_key ← d                   // To be kept secret
   return Public_key and Private_key
}
```

Figure 5.4: Key Generation Algorithm in ElGamal

Example:

Here is a trivial example. Bob chooses p = 11 and e_1 = 2. and d = 3 $e_2 = e_1^d = 8$. So the public keys are (2, 8, 11) and the private key is 3. Alice chooses r = 4 and calculates C1 and C2 for the plaintext 7.

Plaintext: 7

$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

Bob receives the ciphertexts (5 and 6) and calculates the plaintext.

$$[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$$

Plaintext: 7

5.6 THE POLLARD RHO DISCRETE LOGARITHM ALGORITHM

Mathematically, for two real numbers, 'x' and 'y', the logarithm is logy x, which means ym = x, where 'm' is a number. An integer 'z' that solves the equation yz = x mod p is known as the discrete logarithm of 'x' to the base 'y'.

Pollard rho algorithm was introduced in 1978 by John Pollard. The main aim of this algorithm is to solve discrete logarithm problems. As in discrete logarithm problems, we are given an equation : $y^z = x$. So, this algorithm aims to compute the value of 'z', such that 'x' belongs to the cyclic group 'G' of 'y'.

A cyclic group is generally a group of elements generated by a single integer. For example, numerous of values will be generated for 'x' for different values of 'z' on 'y'. So, the group of all those values of 'x' is known as the cyclic group.

Algorithm

The algorithm computes integers a, b, A, and B such that $x^a y^b = x^A y^B$, where r $= x^{ai}y^{bi}$. Then we can easily compute the value of 'z' using equation (B - b)z = (A - a) (mod p).

To define such functions, divide group 'G' into approximately three disjoint subsets of equal size. Let these subsets be G0, G1, and G2. According to the algorithm, x, and y belongs to the set G, and G is a combination of G0, G1, and G2.

$X,Y \in G$

$G = G0 \cup G1 \cup G2$

To calculate the value of 'r', the function F is defined as –

$$F(r) = \begin{cases} yr, & r \in G0 \\ r^2, & r \in G1 \\ xr, & r \in G2 \end{cases}$$

To calculate the value of a, b, A and B, the equations used are -

$a_i \leftarrow g(x_{i-1}, a_{i-1})$

$b_i \leftarrow h(x_{i-1}, b_{i-1})$

$A_i \leftarrow g(f(x_{2i-2}), g(x_{2i-2}, a_{2i-2}))$

$B_i \leftarrow h(f(x_{2i-2}), h(x_{2i-2}, b_{2i-2}))$

So, we need two more functions, g and h. These functions are defined as -

$$H(r,x)=\begin{cases} x+1, & r \in G0 \\ 2x, & r \in G1 \\ x, & r \in G2 \end{cases}$$

5.7 ELLIPTIC CURVES

Elliptic-curve cryptography (ECC) is a public-key encryption technique that is based on the algebraic structure of elliptic curves over finite fields. When compared to non-EC encryption (based on ordinary Galois fields), ECC allows for fewer keys to guarantee equal security.

ECC is based on the properties of a set of values, for which operations can be performed on any two members of the group to produce a third member, which is derived from points where the line intersects the axes, as shown in the diagram below labelled A, B, and C with the green line and three blue dots. A point on the curve is produced by multiplying it by a number (C). Point D is obtained by taking point C and bringing it to the mirrored point on the opposite side of the x-axis. A line is traced back to our starting point A, intersecting at point E.

This technique can be repeated n times up to a certain limit. The n is the private key number, which defines how many times the equation should be performed before arriving at the final value needed to encrypt and decode data. The maximum specified value of the equation is related to the key size utilised.

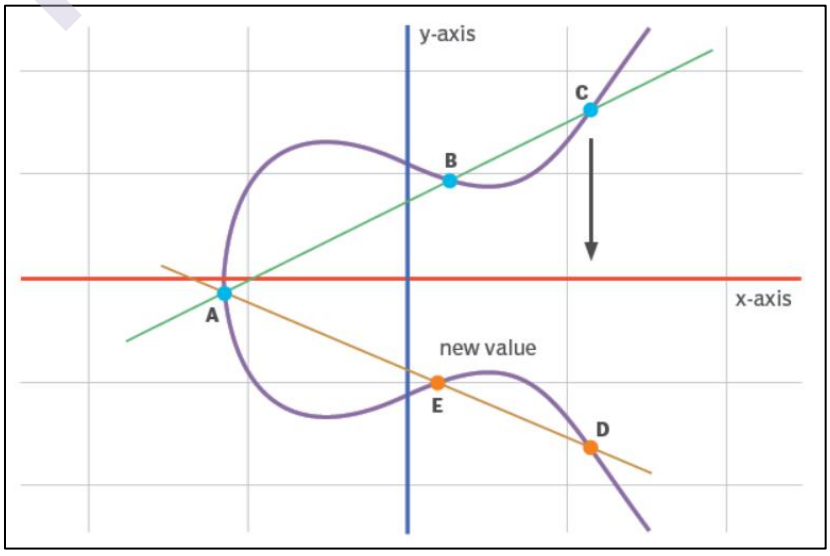


Figure 5.5: Elliptical Curve Showing Three Points of Intersection

Generic Procedures of ECC

- Both parties agree to some publicly-known data items – The elliptic curve equation
- values of a and b
- prime, p – The elliptic group computed from the elliptic curve equation – A base point, B, taken from the elliptic group
- Similar to the generator used in current cryptosystems
- Each user generates their public/private key pair – Private Key = an integer, x, selected from the interval [1, p-1] – Public Key = product, Q, of private key and base point
- (Q = x*B)

Applications of ECC

- Many devices are small and have limited storage and computational power
- Where can we apply ECC? – Wireless communication devices – Smart cards – Web servers that need to handle many encryption sessions
- Any application where security is needed but lacks the power, storage and computational power that is necessary for our current cryptosystems

5.8 KNAPSACK PROBLEM

The problem's name refers to packing stuff into a knapsack. Is it possible to choose some of the objects to pack so that their "sum" (the amount of space they occupy) exactly equals the knapsack capacity (the target)? The problem may be expressed as a case of adding integers. Is there a subset of nonnegative integers whose sum matches the objective given a set of nonnegative integers and a target?

Formally, given a set S = {a1, a2,..., an} and a target sum T, where each ai >= 0, we want to know if there is a selection vector, V = [v1, v2,..., vn], each of whose elements is 0 or 1, such that

$$T = \sum_{i=1}^{n} (ai * vi)$$

The selection vector V records a 1 for each element chosen for the sum and a 0 for each not chosen. Thus, each element of S can be used once or not at all.

For example, the set S might be {4, 7, 1, 12, 10}. A solution exists for target sum T = 17, since 17 = 4 + 1 + 12. The selection vector is V = [1,0,1,1,0]. No solution is possible for T = 25.

Definition

$a = [a_1, a_2, \ldots, a_k]$ and $x = [x_1, x_2, \ldots, x_k]$.

$$s = knapsackSum\ (a, x) = x_1a_1 + x_2a_2 + \cdots + x_ka_k$$

Given a and x, it is easy to calculate s. However, given s and a it is difficult to find x.

Super increasing Tuple: $ai \geq a1 + a2 + \ldots + ai-1$

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.

```
knapsackSum (x [1 … k], a [1 … k])
{
    s ← 0
    for (i = 1 to k)
    {
      s ← s + a_i × x_i
    }
    return s
}
```

```
inv_knapsackSum (s, a [1 … k])
{
    for (i = k down to 1)
    {
        if s ≥ a_i
        {
            x_i ← 1
            s ← s − a_i
        }
        else x_i ← 0
    }
    return x [1 … k]
}
```

Figure 5.5: Algorithm for Knapsack and inverse knapsack

Example:

As a very trivial example, assume that a = [17, 25, 46, 94, 201,400] and s = 272 are given. Table 10.1 shows how the tuple x is found using inv_knapsackSum routine in Algorithm 10.1. In this case x = [0, 1, 1, 0, 1, 0], which means that 25, 46, and 201 are in the knapsack.

i	a_i	s	$s \geq a_i$	x_i	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

5.8.1 Secret Communication with Knapsacks

The first general public key encryption algorithm was the Knapsack Encryption Algorithm. Ralph Merkle and Mertin Hellman invented it in 1978. Because it is a public key cryptography, it necessitates the use of two distinct keys. The first is a public key, which is used for encryption, and the second is a private key, which is used for decryption. This method will employ two separate knapsack problems, one easy and one difficult. The private key is the easy knapsack, whereas the public key is the hard knapsack. The hard knapsack is deduced from the easy knapsack.

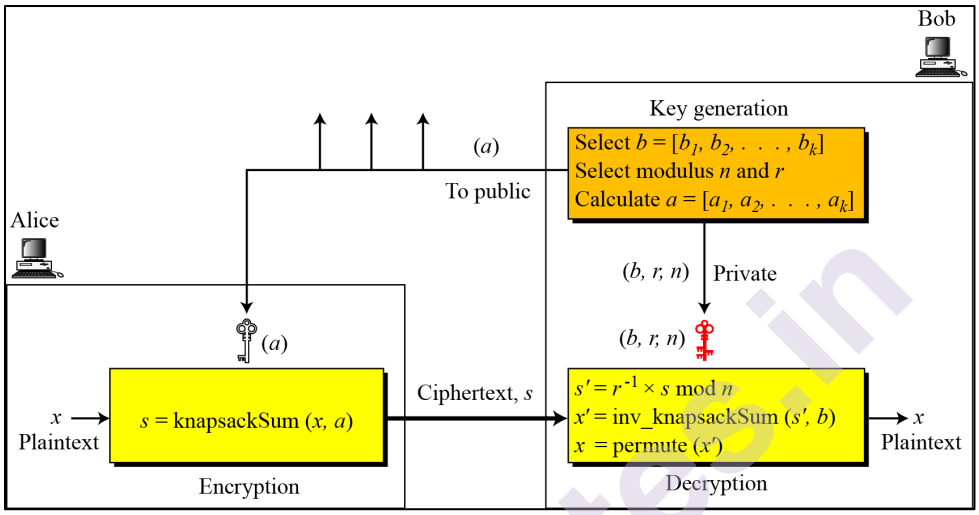


Figure 5.6: Secret communication with knapsack cryptosystem

- **Key Generation**
  - Create a superincreasing k-tuple b=[b1,b2,....bk]
  - Choose modulus n, such that n>b1+b2+...+bk
  - Select random integer r, such that r is relatively prime to n and 1≤r≤n-1
  - Create a temporary k-tuple t=[t1,t2,....,tk] in which ti=rxbi mod n
  - Select a permutation of k objects and find a new tuple a=permute(t)
  - The public key is k-tuple a and private key is n, r and the k-tuple b
- **Encryption**
  - Convert plaintext to k-tuple x=[x1,x2...,xk]
  - Knapsacksum routine is used to calculate s.
- **Decryption**
  - Calculate s'=r^{-1} x s mod n
  - Use inv_knapsacksum to create x'
  - Permute x' to find x

Example:

This is a trivial (very insecure) example just to show the procedure.

1. Key generation:
   a. Bob creates the superincreasing tuple b = [7, 11, 19, 39, 79, 157, 313].
   b. Bob chooses the modulus n = 900 and r = 37, and [4 2 5 3 1 7 6] as permutation table.
   c. Bob now calculates the tuple t = [259, 407, 703, 543, 223, 409, 781].
   d. Bob calculates the tuple a = permute (t) = [543, 407, 223, 703, 259, 781, 409].
   e. Bob publicly announces a; he keeps n, r, and b secret.
2. Suppose Alice wants to send a single character "g" to Bob.
   a. She uses the 7-bit ASCII representation of "g", $(1100111)_2$, and creates the tuple x = [1, 1, 0, 0, 1, 1, 1]. This is the plaintext.
   b. Alice calculates s = *knapsackSum* (a, x) = 2165. This is the ciphertext sent to Bob.

S=2399

3. Bob can decrypt the ciphertext, s = 216 ***S=2399***
   a. Bob calculates $s' = s \times r^{-1} \bmod n$ = 2165 × ***$2399x37^{-1}$ mod 900=527***
   b. Bob calculates x' = *Inv_knapsackSum* (s', b) = [1, 1, 0, 1, 0, 1, 1].
   c. Bob calculates x = *permute* (x') = [1, 1, 0, 0, 1, 1, 1]. He interprets the string $(1100111)_2$ as the character "g".

5.9 SUMMARY

This chapter provides a detailed look at various public key cryptosystem.

[*Note: Most of the figures from this chapter are taken from B. A. Fourozan "Cryptography and Network Security]

❄❄❄❄❄❄❄

6

KEY DISTRIBUTION AND KEY AGREEMENT SCHEME

Unit Structure :

6.1 OBJECTIVES

- To protect information from being stolen, compromised, or attacked.
- To protect sensitive information on the internet and on devices safeguarding them from attack, destruction, or unauthorized access.
- To ensure a risk-free and secure environment for keeping the data, network and devices guarded against cyber threats.
- To monitor any breaching activities and managing data at rest and data in motion.
- To implement measures to detect and prevent unauthorized access to systems, networks, and data.
- To develop policies, standards, and procedures to ensure compliance with applicable laws and regulations.

6.2 DEFINITION

Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks. These cyberattacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes.

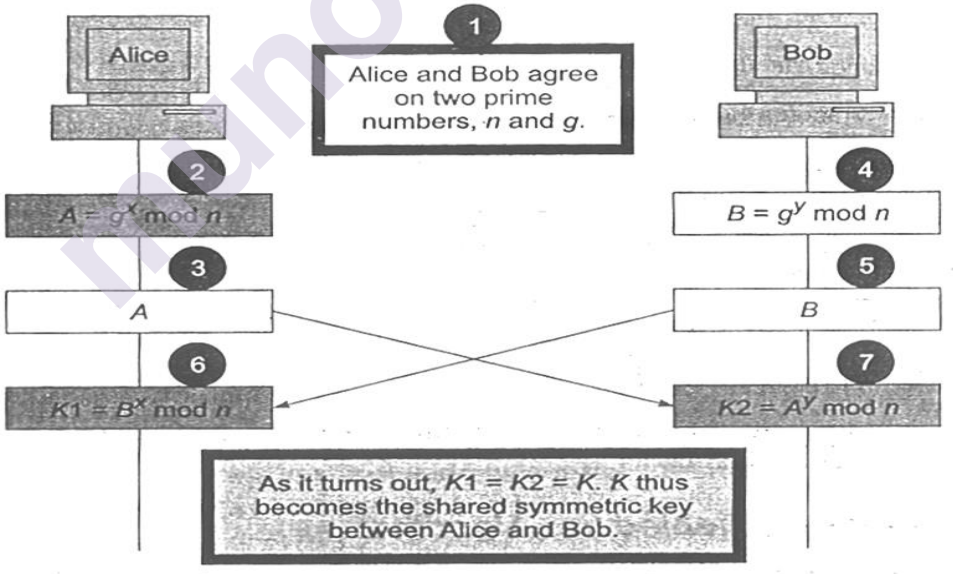
6.3 INTRODUCTION

4.Diffie-Hellman Key distribution and Key agreement scheme

Assume that Alice and Bob want to agree upon a key to be used for encrypting/decrypting messages that would be exchanged between them. Then, the Diffie-Hellman key exchange algorithm works as shown in figure. It suffers from an attack known as **Man-in-the middle attack/Bucket Brigade attack.**

This algorithm works as follows:

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number x, and calculates A such that: $A = g^x \bmod n$
3. Alice sends the number A to Bob.
4. Bob independently chooses another large random integer y and calculates B such that: $B = g^y \bmod n$
5. Bob sends the number B to Alice.
6. A now computes the secret key K1 as follows: $K1 = B^x \bmod n$
7. B now computes the secret key K2 as follows: $K2 = A^y \bmod n$



K1 is actually equal to K2! This means that K1 = K2 = K is the symmetric key, which Alice and Bob must keep secret and can henceforth use for encrypting/decrypting their messages with.

Example of the Algorithm

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.

   Let n = 11, g = 7.

2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$

   Let x = 3. Then, we have, $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$.

3. Alice sends the number A to Bob.

   Alice sends 2 to Bob.

4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$

   Let y = 6. Then, we have, $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$.

5. Bob sends the number B to Alice.

   Bob sends 4 to Alice.

6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$

   We have, $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$.

7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$

   We have, $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$.

6.5 KEY DISTRIBUTION PATTERNS

Key Distribution For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.

Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key.

For two parties A and B, key distribution can be achieved in several ways, as follows:

1. A can select a key and physically deliver it to B.

2. A third party can select the key and physically deliver it to A and B.

3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.

4. If A and B each has an encrypted connection to a third-party C, C can deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link.

However, for end-to-end encryption, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device

needs several keys supplied dynamically. The problem is especially difficult in a wide area distributed system.

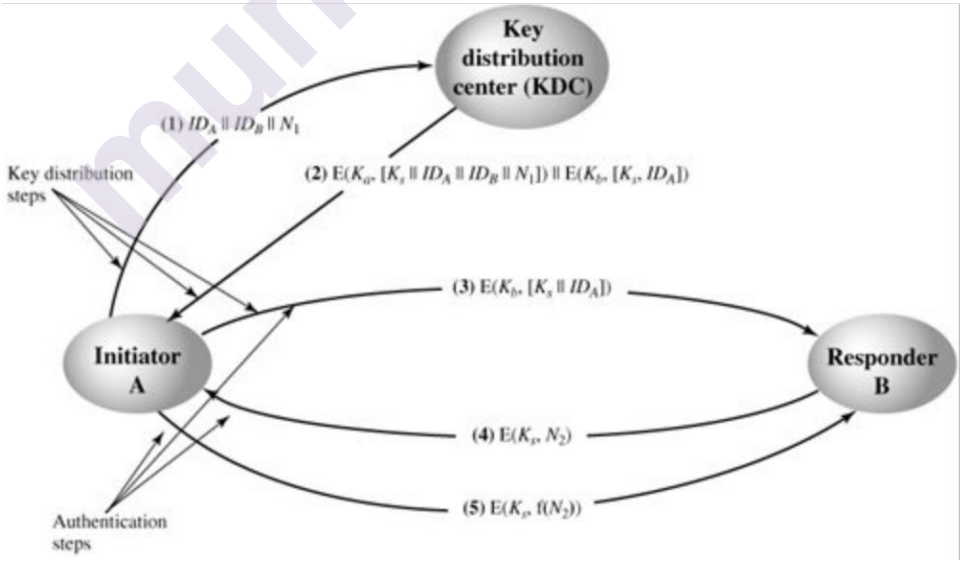
If end to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate. Thus, if there are N hosts, the number of required keys is [N(N-1)]/2. If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication. Thus, a network may have hundreds of hosts but thousands of users and processes.

The use of a key distribution centre is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used. Communication between end systems is encrypted using a temporary key, often referred to as a session key. Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded.

Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a master key that is shared by the key distribution center and an end system or user.

A Key Distribution Scenario:

The key distribution concept can be deployed in several ways. A typical scenario is illustrated in Figure. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).



Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, Ka, known only to itself and the KDC; similarly, B shares the master key Kb with the KDC.

The following steps occur:

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N1, for this transaction, which we refer to as a nonce. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using Ka Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

   - The one-time session key, Ks, to be used for the session
   - The original request message, including the nonce, to enable A to match this response with the appropriate request Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

   In addition, the message includes two items intended for B:

   - The one-time session key, Ks to be used for the session
   - An identifier of A (e.g., its network address), IDA These last two items are encrypted with Kb (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, E (Kb, [Ks || IDA]). Because this information is encrypted with Kb, it is protected from eavesdropping. B now knows the session key (Ks), knows that the other party is A (from IDA), and knows that the information originated at the KDC (because it is encrypted using Kb). At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N2, to A.

5. Also using Ks, A responds with f(N2), where f is a function that performs some transformation on N2.

   These steps assure B that the original message it received (step 3) was not a replay.

6.6 MITCHELL-PIPER KEY DISTRIBUTION PATTERN

It is a type of key distribution pattern.

6.7 STATION-TO-STATION PROTOCOL

In public-key cryptography, the Station-to-Station (STS) protocol is a cryptographic key agreement scheme. The protocol is based on classic Diffie–Hellman and provides mutual key and entity authentication. This protocol assumes that the parties have signature keys, which are used to sign messages, thereby providing security against man-in-the-middle attacks.

In addition to protect the established key from an attacker, the STS protocol uses no timestamps and provides perfect forward secrecy. It also entails two-way explicit key confirmation, making it an authenticated key agreement with key confirmation (AKC) protocol.

STS Setup

The following data must be generated before initiating the protocol.

An asymmetric signature keypair for each party Required for authentication. The public portion of this keypair may be shared prior to session establishment.

Key establishment parameters. The specification of a cyclic group p and a generator g for that group. These parameters may be public.

Basic STS

Supposing all setup data has been shared, the STS protocol proceeds as follows. If a step cannot be completed, the protocol immediately stops. All exponentials are in the group specified by p.

a. Alice generates a random number x and computes and sends the exponential gx to Bob.

b. Bob generates a random number y and computes the exponential gy.

c. Bob computes the shared secret key K = (gx)y.

d. Bob concatenates the exponentials (gy, gx) (order is important), signs them using his asymmetric (private) key B, and then encrypts the signature with K. He sends the ciphertext along with his own exponential gy to Alice.

e. Alice computes the shared secret key K = (gy)x.

f. Alice decrypts and verifies Bob's signature using his asymmetric public key.

g. Alice concatenates the exponentials (gx, gy) (order is important), signs them using her asymmetric (private) key A, and then encrypts the signature with K. She sends the ciphertext to Bob.

h. Bob decrypts and verifies Alice's signature using her asymmetric public key.

Alice and Bob are now mutually authenticated and have a shared secret. This secret, K, can then be used to encrypt further communication. The basic form of the protocol is formalized in the following three steps:

(1) Alice → Bob : gx

(2) Alice ← Bob : gy, EK(SB(gy, gx))

(3) Alice → Bob : EK(SA(gx, gy))

6.8 MTI KEY AGREEMENT SCHEME

In cryptography, a key-agreement protocol is a protocol whereby two or more parties can agree on a key in such a way that both influence the outcome. If properly done, this precludes undesired third parties from forcing a key choice on the agreeing parties. Protocols that are useful in practice also do not reveal to any eavesdropping party what key has been agreed upon.

Many key exchange systems have one party generate the key, and simply send that key to the other party—the other party has no influence on the key. Using a key-agreement protocol avoids some of the key distribution problems associated with such systems.

Protocols where both parties influence the final derived key are the only way to implement perfect forward secrecy.

6.9 PUBLIC-KEY INFRASTRUCTURE

Public key infrastructure or PKI is the governing body behind issuing digital certificates. It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

6.10 WHAT IS PKI?

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them.

Public key management further requires:

- **Keeping the private key secret:** Only the owner of a private key is authorized to use a private key. It should thus remain out of reach of any other person.
- **Assuring the public key:** Public keys are in the open domain and can be publicly accessed. When this extent of public accessibility, it becomes hard to know if a key is correct and what it will be used for. The purpose of a public key must be explicitly defined.

Public Key Infrastructure:

Public key infrastructure affirms the usage of a public key. PKI identifies a public key along with its purpose. It usually consists of the following components:

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system

Working on a PKI:

- **PKI and Encryption:** The root of PKI involves the use of cryptography and encryption techniques. Both symmetric & asymmetric encryption uses a public key. The challenge here is – "how do you know that the public key belongs to the right person or to the person you think it belongs to?".
- There is always a risk of MITM (Man in the middle). This issue is resolved by a PKI using digital certificates. It gives identities to keys to make the verification of owners easy and accurate.
- **Public Key Certificate or Digital Certificate:** Digital certificates are issued to people and electronic systems to uniquely identify them in the digital world. Here are a few noteworthy things about a digital certificate. Digital certificates are also called X.509 certificates. This is because they are based on the ITU standard X.509.
- The Certification Authority (CA) stores the public key of a user along with other information about the client in the digital certificate. The information is signed and a digital signature is also included in the certificate.
- The affirmation for the public key then thus be retrieved by validating the signature using the public key of the Certification Authority.
- **Certifying Authorities:** A CA issues and verifies certificates. This authority makes sure that the information in a certificate is real and correct and it also digitally signs the certificate. A CA or Certifying Authority performs these basic roles:
- Generates the key pairs – This key pair generated by the CA can be either independent or in collaboration with the client.
- Issuing of the digital certificates – When the client successfully provides the right details about his identity, the CA issues a certificate to the client. Then CA further signs this certificate digitally so that no changes can be made to the information.

- Publishing of certificates – The CA publishes the certificates so that the users can find them. They can do this by either publishing them in an electronic telephone directory or by sending them out to other people.
- Verification of certificate – CA gives a public key that helps in verifying if the access attempt is authorized or not.
- Revocation – In case of suspicious behavior of a client or loss of trust in them, the CA has the power to revoke the digital certificate.

PKIX Services

a. Registration

It is the process where an end-entity (subject) makes itself known to a CA. Usually, this is via an RA.

b. Initialization

This deals with the basic problems, such as who the end-entity is sure that it is talking to the right CA?

c. Certification

In this step, the CA creates a digital certificate for the end-entity and returns it to the end-entity maintains a copy of its own records and also copies it in public directories, if required.

d. Key pair recovery

Keys used for encryption may be required to be recovered later for decrypting some old documents. Key archival and recovery services can be provided by a CA or by an independent key recovery system.

e. Key generation

PKIX specifies that the end-entity should be able to generate private and public key pairs or the CA/RA should be able to do this for the end-entity (and then distribute these keys securely to the end-entity).

f. Key update

This allows a smooth transition from one expiring key pair to a fresh one, by the automatic renewal of digital certificates. However, there is a provision for manual digital certificates renewal request and response.

g. Cross-certification

Helps in establishing trust models, so that end-entities that are certified by different CAs can cross-verify each other.

h. Revocation

PKIX provides support for the checking of the certificate status in two modes: online (using OCSP) or offline (using CRL).

Challenges that a PKI Solves:

PKI owes its popularity to the various problems it solves. Some use cases of PKI are:

- Securing web browsers and communicating networks by SSL/TLS certifications.
- Maintaining Access Rights over Intranets and VPNs.
- Data Encryption/
- Digitally Signed Software
- Wi-fi Access Without Passwords

Other than these, one of the most important use cases of PKI is based around IoT (Internet of Things). Here are two industries that are using PKI for IoT devices:

- **Auto Manufacturers:** Cars these days have features like GPS, call for services, assistants, etc. These require communication paths where a lot of data is passed. Making these connections secure is very important to avoid malicious parties hacking into the cars. This is where PKI comes in.
- **Medical device Manufacturers:** Devices like surgical robots require high security. Also, FDA mandates that any next-generation medical device must be updatable so that bugs can be removed, and security issues can be dealt with. PKI is used to issues certificates to such devices.

Disadvantages of PKI:

- **Speed:** Since PKI uses super complex algorithms to create a secure key pair. So, it eventually slows down the process and data transfer.
- **Private Key Compromise:** Even though PKI can't be hacked very easily but a private key can be hacked by a professional hacker, since PKI uses Public and Private key to encrypt and decrypt data so with user's private key in hand and public key which is easily available the information can be decrypted easily.

6.11 UNIT END EXERCISE

1. Explain Diffie-Hellman Key distribution and Key agreement scheme.
2. Explain how Diffie-Hellman Key distribution suffers from an attach known as Man-in-the middle attack/Bucket Brigade attack.
3. What is meant by Key Distribution Patterns.
4. Explain a Key Distribution Scenario.

5. Describe Station-to-station protocol.
6. Explain the basic steps of Station-to-station protocol.
7. What is meant by MTI Key Agreement scheme.
8. What is Public-Key Infrastructure.
9. What are the components of Public-Key Infrastructure.
10. Describe the working of Public-Key Infrastructure.
11. What are the different roles of Certifying Authority.
12. Explain different PKIX Services
13. What are the challenges that PKIX Solves.
14. Write the advantages & disadvantages of PKI.

6.12 SUMMARY

This chapter focuses on key distribution patterns & schemes. It describes about the certifying authority, their roles & responsibilities, and public key infrastructure.

6.13 REFERENCE FOR FURTHER READING

Textbook:

- Discrete Mathematics and Its Applications, Kenneth H. Rosen, 7 th Edition, McGraw Hill, 2012.
- Cryptography Theory and Practice, 3rd Edition, Douglas R. Stinson, 2005.

Reference:

- Network Security and Cryptography, Atul Kahate, McGraw Hill, 2003.
- Cryptography and Network Security: Principles and Practices, William Stalling, Fourth Edition, Prentice Hall, 2013.
- Introduction to Cryptography with coding theory, second edition, Wade Trappe, Lawrence C. Washington, Pearson, 2005.

❄❄❄❄❄❄❄

KEY DISTRIBUTION AND KEY AGREEMENT SCHEME

Unit Structure :

7.1 OBJECTIVES

The objective of Cybersecurity is to protect information from being stolen, compromised or attacked. Cybersecurity can be measured by at least one of three goals-

- Protect the confidentiality of data.
- Preserve the integrity of data.
- Promote the availability of data for authorized users.

7.2 DEFINITION

The Confidentiality, Integrity & Availability (CIA) Triad is a security model developed to ensure the 3 goals of cybersecurity, which are Confidentiality, Integrity, and Availability of data and the network.

1. Confidentiality: Keeping the sensitive data private and accessible to only authorized users.

2. Integrity: Designed to protect the data from unauthorized access and ensure its reliability, completeness and correctness.

3. Availability: Authorized users can have access to system resources and data as and when they need it.

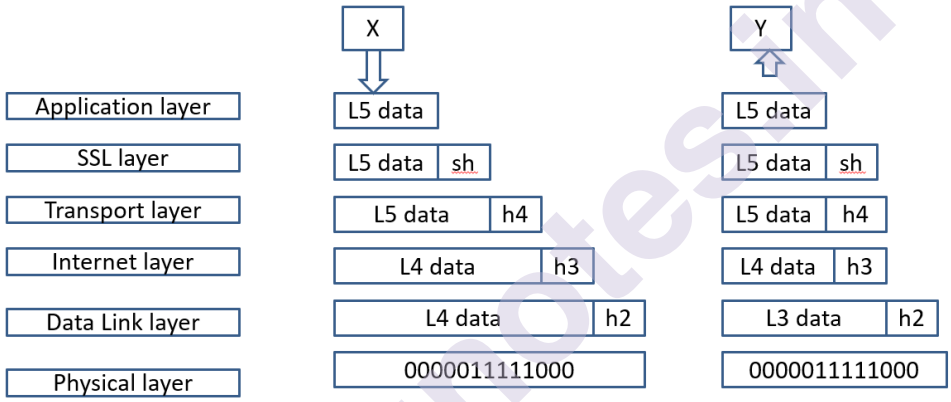
7.3 INTRODUCTION

Cyber Security refers to the technologies, processes and practices designed to protect networks, devices, app and data from any kind of cyber-attacks. Cyber security may also known as information technology (IT) security. Cyber Security is all about protecting your devices and network from unauthorized access or modification. The Internet is not only the chief source of information, but it is also a medium through which people do business.

7.4 SOCKET LAYER

It is also known as **Secure Socket Layer (SSL)**. It is an internet protocol used for the secure exchange of information b/w a web browser & a web server. it provides authentication & confidentiality.

Position of SSL in TCP/IP protocol suite.

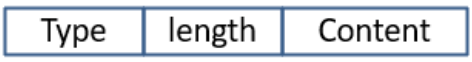


Working of SSL

It has got three protocols.

1.Handshake protocol

It is the protocol used by the client & server to communicate using an SSL enabled connection. It consists of series of messages b/w the client & the server. Its format is.

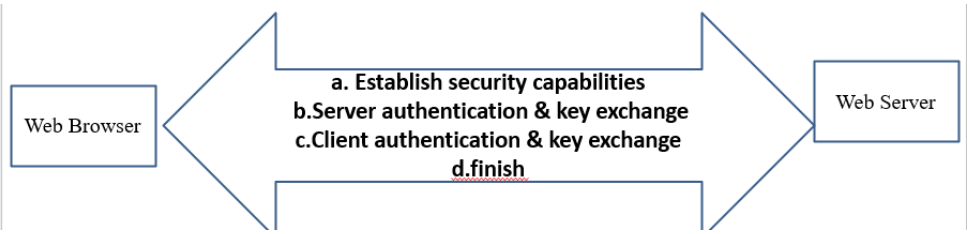


Type-it contains one of the ten possible message types. It could be hello request, client hello, server hello, server hello done etc.

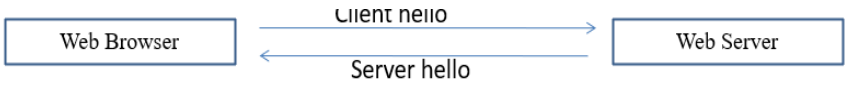
Length-It indicates the length of the message.

Content- It contains the parameter associated with this message.

Handshake protocol is consisting of 4 phases.



a. **Establishing security capabilities-**it is used to initiate a logical connection & establish the security capabilities associated with that connection. it has client & server hello.



It starts with a client hello from the client to server. It has following parameters:

Version-it identifies the highest version of SSL that the client can support.

Random-it generates a random number from the random number generator software built inside the computer.

Session id-a zero value in this indicates that the client wants to create a new connection with the server. & anon zero value indicates that there is already a connection b/w client & server.

Cipher Suite-This contains the list of cryptographic algorithms supported by the client (RSA, Diffie-Hellman etc.)

Compression Method-This contains the list of compression algorithms supported by the client.

the client sends the client hello message to the server & waits for the server's response. Accordingly, the server sends back a server hello message to the client.

The server hello message consists of:

Version- This identifies the lower of the versions suggested by the client & highest supported by the server.

Random-It generates a random number from the random number generator software built inside the computer.

Session id- if the session id value sent by the client was non-zero, the server uses the same value, otherwise the server creates a new session id.

Cipher suite- This contains the list of cryptographic algorithms, which the server selects one from the list.

Compression method-This contains the list of compression algorithms, which the server selects one from the list.

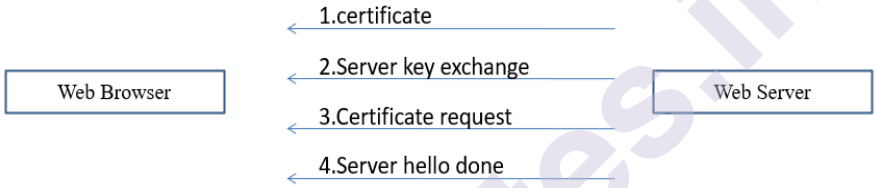
b. **Server authentication & key exchange-** The server initiates this phase & is the sender of the messages. the client is the recipient of all the messages. It consists of

Certificate –the server sends its digital certificate & the entire chain leading up to root CA to the client. This will help the client to authenticate server using server's public key from server's certificate.

Server key exchange- It is used only if the server does not send its digital certificate to the client. the server sends its public key to the client.

Certificate request-the server can request for the client's digital certificate.

Server hello done- this message It indicates to the client that its portion of the hello message is complete. client can now verify the certificates sent by the server. after sending this message server waits for the clients' response.

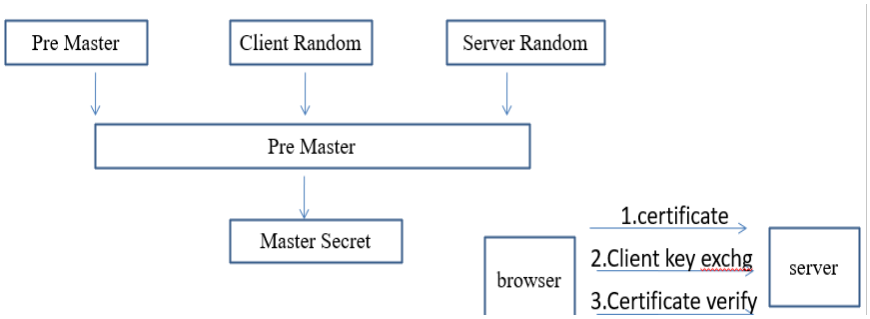


c. **Client authentication & key exchange-** client initiates this phase & is the sender of all messages. the server is the recipient of all messages. It consists of:

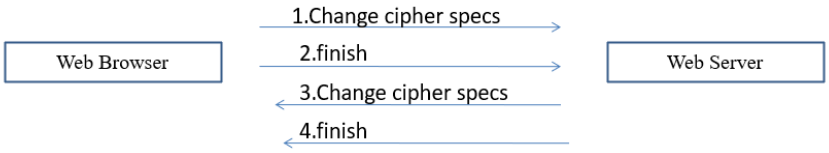
Certificate-this step is performed only if the server had requested for the client's digital certificate. & if the client does not have then it sends no certificate message to the server.

Client key Exchange-It allows the client to send information to the server. Client creates a 48-byte pre master secret & encrypts it with the server public key & sends encrypted pre-master secret to the server.

Certificates Verify-It is necessary only if the server had demanded client authentication. Thus, the client combines the pre-master secret with the random number after hashing them using MD5 & SHA-1 & with its private key.

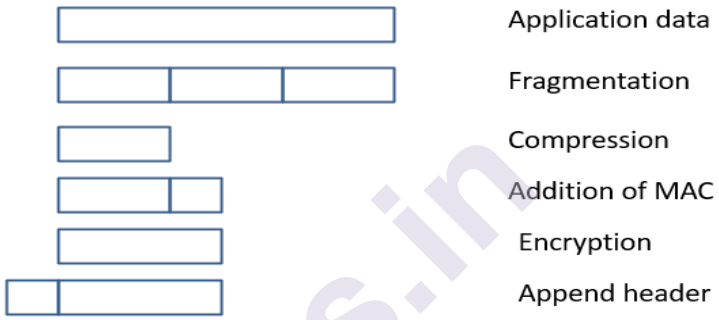


d. **Finish-**the client initiates this phase which the server ends. the first two messages are from client & the server responds back with the last two messages.



2. Record Protocol

After the client & server authenticated each other & have decided what algorithms to use for secure exchange of information, for this record protocol is.



Fragmentation-the original message is broken into blocks, so that the size of each block is less than or equal to 2^{14} bytes.

Compression-This should not result into the loss of the original data.i. e it must be a lossless compression.

Addition of MAC-using the shared secret key established in handshake protocol, the MAC is calculated.

Encryption-various encryption algorithms such as –AES, IDEA etc. are used.

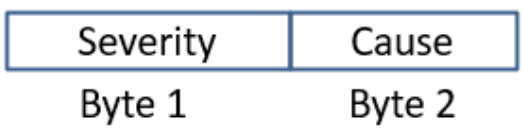
Append Header-a header is added to the encrypted block.

3. Alert Protocol

When either the client or the server detects an error, the detecting party sends an alert message to the other party. if the error is fatal both the parties immediately close the SSL connection.i.e transmission from both the ends is terminated .

They destroy the session identifiers, secret keys. Errors which are not severe ,do not results in the termination .instead they handle error& continue.

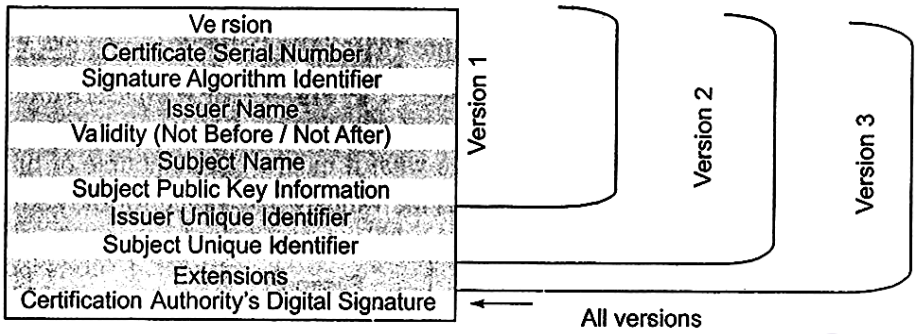
Each alert message consists of two bytes. the first byte signifies the type of error. if it is a warning, this byte contains 1. if the error is fatal, this byte contains 2.



7.5 CERTIFICATES

Digital Certificate

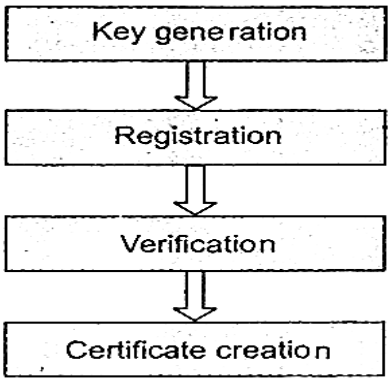
A standard called X.509 defines the structure of a digital certificate. Figure 1 shows the structure of a X.509 V3 digital certificate. Parties involved in digital certificate creation.



Field	Description
Version	Identifies a particular version of the X.509 protocol, which is used for this digital certificate. Currently, this field can contain 1, 2 or 3.
Certificate Serial Number	Contains a unique integer number, which is generated by the CA.
Signature Algorithm Identifier	Identifies the algorithm used by the CA to sign this certificate.
Issuer Name	Identifies the Distinguished Name (DN) of the CA that created and signed this certificate.
Validity (Not Before/Not After)	Contains two date-time values (Not Before and Not After), which specify the time frame within which the certificate should be considered valid. These values generally specify the date and time up to seconds or milliseconds.
Subject Name	Identifies the Distinguished Name (DN) of the end entity (i.e. the user or the organization) to whom this certificate refers. This field must contain an entry unless an alternative name is defined in Version 3 extensions.
Subject Public Key Information	Contains the subject's public key and algorithms related to that key. This field can never be blank.

7.6 CERTIFICATE LIFE CYCLE

Certificate creation -The creation of a digital certificate consists of several steps. These steps are:



Step 1: Key generation

The user/organization) who wants to obtain a certificate, there are two different approaches for this purpose:

(a) The subject can create a private key and public key pair using some software. The subject must keep the private key thus generated, secret. The subject then sends the public key along with other information and evidence about herself to the RA.

(b) RA can generate a key pair on the subject's (user's) behalf. This can happen in cases where either the user is not aware of the technicalities involved in the generation of a key pair,

the major disadvantages of this approach is the possibility of the RA knowing the private key of the user, and key to be exposed to others while in transit after it is generated and sent to the appropriate user.

Step 2: Registration

This step is required only if the user generates the key pair in the first step. If the RA generates the key pair on the user's behalf, this step will also be a part of the first step itself.

Assuming that the user has generated the key pair, the user now sends the public key and the associated registration information (e.g. subject name, as it is desired to appear in the digital certificate) and all the evidence about herself to the RA. For this, the software provides a wizard in which the user enters data and when all data is correct, submits it. This data then travels over the network/Internet to the RA. The format for the certificate requests has been standardized and is called Certificate Signing Request (CSR).

Step 3: Verification

After the registration process is complete, the RA must verify the user's credentials. This verification is in two respects, as follows:

(a) RA needs to verify the user's credentials such as the evidence provided are correct and that they are acceptable. If the user were an organization, then the RA would perhaps like to check the business records, historical documents and credibility proofs. If it is an individual user then simpler checks are in call, such as verifying the postal address, email id, phone number, passport or driving license details.

b) The second check is to ensure that the user who is requesting for the certificate does indeed possess the private key corresponding to the public key that is sent as a part of the certificate request to the RA. **This check is called checking the Proof Of Possession (POP) of the private key.** How can the RA perform this check? There are many approaches to this:

1) The RA can demand that the user must digitally sign his/her Certificate Signing Request (CSR) using his/her private key. If the RA can verify the signature (i.e. de-sign the CSR) correctly using the public key of the user, the RA can believe that the user indeed possesses the private key.

2) RA can create a random number challenge, encrypt it with the user's public key and send the encrypted challenge to the user. If the user can successfully decrypt the challenge using his/her private key, the RA can assume that the user possesses the right private key.

3) RA can generate a dummy certificate for the user, encrypt it using the user's public key and send it to the user. The user can decrypt it only if he/she can decrypt the encrypted certificate and obtain the plain-text certificate.

Step 4: Certificate creation

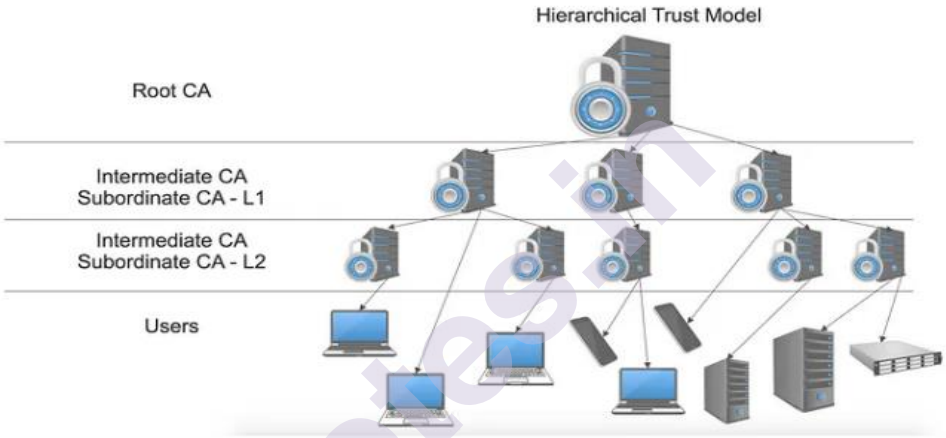
Assume all the steps have been successful, the RA passes on all the details of the user to the CA. The CA does its own verification and creates a digital certificate for the user. The CA sends the certificate to the user and also retains a copy of the certificate for its own record. The CA's copy of the certificate is maintained in a certificate directory. This is a central storage location maintained by the CA.

7.7 TRUST MODELS: STRICT HIERARCHY MODEL

A Trust Model is the collection of rules that inform application on how to solve the legitimacy of a Digital Certificate. According to the ITU-T X.509, Section 3.3.54, trust is defined as follows: "Generally an entity can 'trust' the second entity if the first entity makes the assumption that the second entity will behave exactly as the first entity expects."

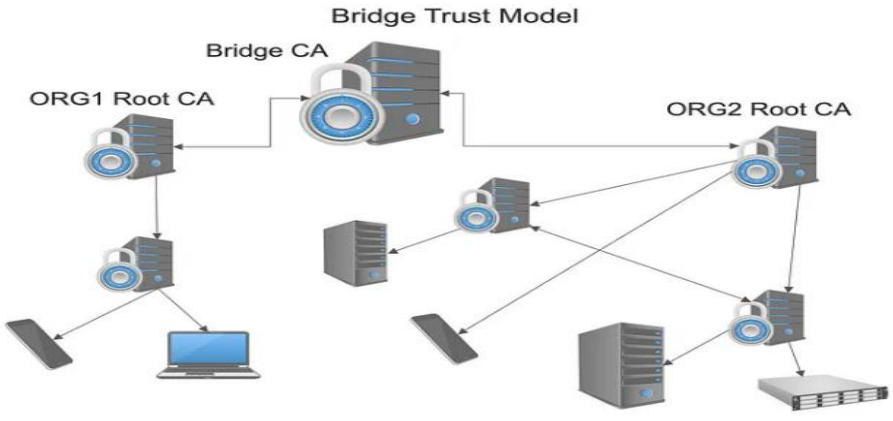
To implement a trust model that can cover all or some of these principles, one of the best ways is Public Key Infrastructure (PKI) and there are four types that are used to implement the trust model with PKI.

A. Hierarchical Trust Model: The hierarchical model or tree model is the most common model to implement the PKI. A root CA at the top provides all the information and the intermediate CAs are next in the hierarchy, and they only trust the information provided by the root. The root CA also trusts intermediate CAs that are in their level in the hierarchy. This arrangement allows a high level of control at all levels of the hierarchical tree this might be the most common implementation in a large organization that wants to extend its certificate-processing capabilities. Hierarchical models allow tight control over certificate-based activities.



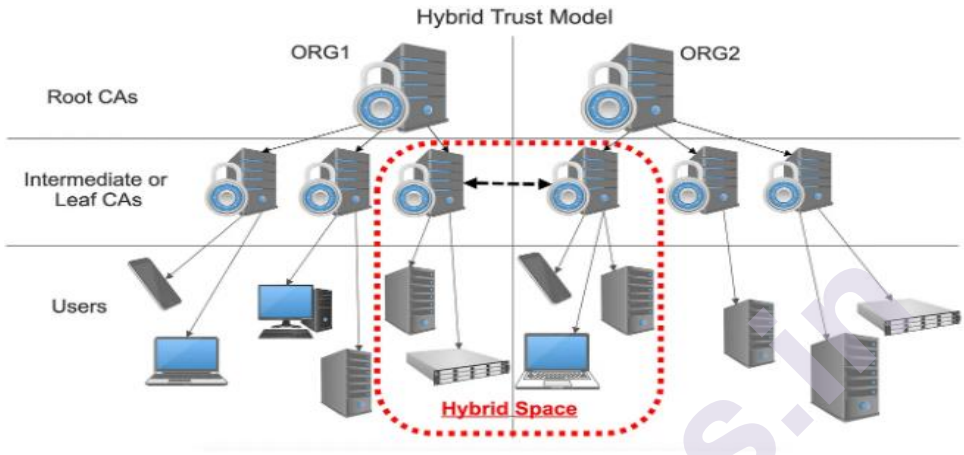
Hierarchical Trust Model

B. Bridge Trust Model: In Bridge Trust Model we have many P2P relations between Root CAs that the Root CAs can communicate with each other and allow cross-certificates. This implementation model allows a certification process to be established between Organizations (or departments). In this model, each intermediate CA trusts only the CAs above and below it but the CA structure can be expanded without creating additional layers of CAs. Additional flexibility and interoperability between organizations are the primary advantages of a bridge model.



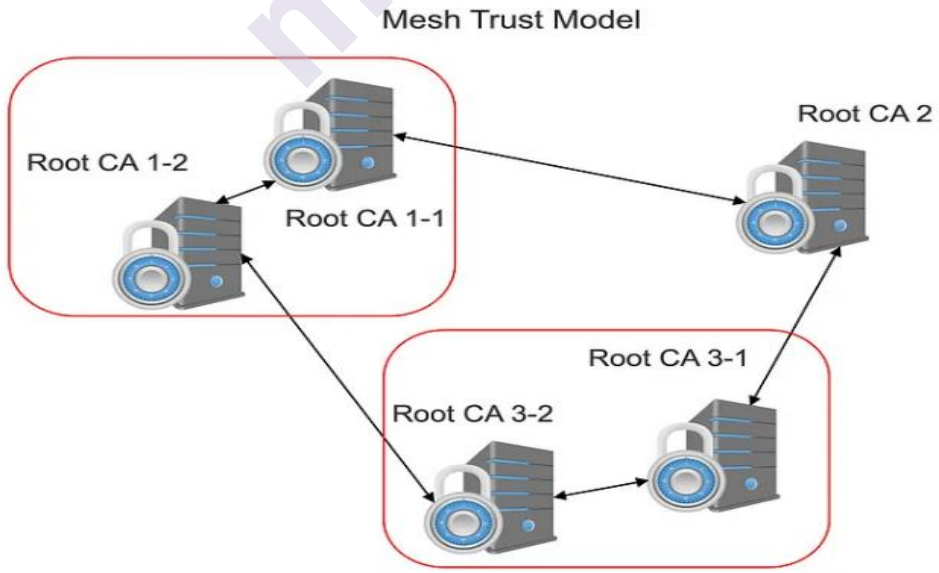
Bridge Trust Model

C. Hybrid Trust Model: Sometimes you need to link two or more organizations or departments in some part and separate other segments. When you need to make trust in some parts of two organizations but don`t want to be this trust in other segments of your organization. In these times the Hybrid Trust Model can be the best model for you. Notice that in this structure, the intermediate CAs which are out of the hybrid environment can trust only to direct Root CA and Intermediate CAs in the hybrid environment, trust to all Root CAs that connect to any intermediate CA in the hybrid environment.



Hybrid Trust Model

D. Mesh Trust Model: When you want to Implement a Hierarchical Trust Model with cross-certification checking or a web of Root CAs, the mesh trust model is the best choice. In the other sights, the mesh model migrates the concepts of bridge structure with multi-paths and multi Root CAs. Certifications in each one of Root CAs are authorized in all of Root, Intermediate, and leaf CAs and all end-users that connected to each one of CA chains.



Mesh Trust Model

7.8 PKIS

What is PKI and What is it used for?

The Public key infrastructure (PKI) is the set of hardware, software, policies, processes, and procedures required to create, manage, distribute, use, store, and revoke digital certificates and public-keys. PKIs are the foundation that enables the use of technologies, such as digital signatures and encryption, across large user populations. PKIs deliver the elements essential for a secure and trusted business environment for e-commerce and the growing Internet of Things (IoT).

PKIs help establish the identity of people, devices, and services – enabling controlled access to systems and resources, protection of data, and accountability in transactions. Next generation business applications are becoming more reliant on PKI technology to guarantee high assurance, because evolving business models are becoming more dependent on electronic interaction requiring online authentication and compliance with stricter data security regulations.

PKI Deployment

PKIs provide a framework that enables cryptographic data security technologies such as digital certificates and signatures to be effectively deployed on a mass scale. PKIs support identity management services within and across networks and underpin online authentication inherent in secure socket layer (SSL) and transport layer security (TLS) for protecting internet traffic, as well as document and transaction signing, application code signing, and time-stamping.

PKIs support solutions for desktop login, citizen identification, mass transit, mobile banking, and are critically important for device credentialing in the IoT. Device credentialing is becoming increasingly important to impart identities to growing numbers of cloud-based and internet-connected devices that run the gamut from smart phones to medical equipment.

7.9 THE WEB BROWSER MODEL

What is Web Browser Security?

A web browser is a software that allows you to access the websites, acting as a portal to the internet. Examples of web browsers are Internet Explorer and Safari. With increased threats and attacks, it was observed that certain attacks could be halted at the web browser setting. Thus web browser security is an essential criterion to safeguard user's data which could be easily accessible by attackers. An organization like LIFARS is providing Cyber Resiliency Program to immediately respond to cyber incidents and breaches.

Where to find the settings?

Different web browsers have different security settings to be defined. To check the correct setting is the responsibility of a user. These customized settings are useful to ensure that the information from the user is not as vulnerable to the attacker. Each web browser is different, hence setting options are available at different locations.

For example, In Internet Explorer, these settings can be found by clicking Tools on your menu bar, -> Internet Options -> Security tab -> Custom Level button.

In Firefox, the navigation path is Tools (menu bar) -> Options -> Content, Privacy-> Security tabs to explore the basic security options.

How do we keep web security sanity?

Various points to consider while making our web browser settings secure are as below:

a. Update your Web browser to the latest version: It is recommended to keep the browser updated in order to get any latest fix provided in the update.

b. Select Automatic clear history option when you exit from the browser

c. Configure browser Privacy and security settings: Review your browser's privacy and security settings to make sure the settings make your browser access secure with what's checked or unchecked. For example, look to see if your browser is blocking third-party cookies, which can enable advertisers to track your online activities.

d. Do not make any private data entry on Public Computers: It is important to note that making any private entry on public computers or using public internet makes the data vulnerable. Being set up for multi-factor authentication can also help mitigate the threat of your password stolen by a public computer. When you finish using a public computer, always clear the browser data, completely close out of the browser, and restart the computer.

e. Do not store passwords in your browser: To make the task easy, usually, users save their home banking, credit card or other confidential passwords on their browser. Though this makes it easier for the user to access their data but makes this data very much vulnerable for attack

f. Using encrypted connections to access websites: Be cautious of using an encrypted connection to access the website you are browsing. In most web browsers, you will see this in the address bar as "http" (not encrypted) or "https" (encrypted). Some browsers have stopped showing "http" or "https" and instead use a green lock icon representing encrypted and a red, yellow, or gray warning symbol for not encrypted.

7.10 PRETTY GOOD PRIVACY

It is widely used as compared to Privacy Enhanced Mail (PEM). It supports cryptographic function-encryption, non-repudiation & msg integrity.

Working

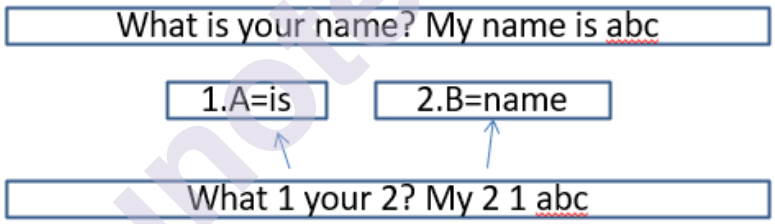
a. Digital Signature

It consists of the creation of a msg digest of the email msg using SHA-1 algorithm. the resulting msg digest is then encrypted with the sender's private key. the result is the sender's digital signature.

b. Compression

The input msg as well as the digital signature are compressed together to reduce the size of the final msg that will be transmitted. for this ZIP program is used.ZIP is based on Lempel-Ziv-algorithm.

This algorithm looks for repeated strings or words & stores them in variables. It then replaces the actual occurrence of the repeated word or string with a pointer to the corresponding variable. since a pointer requires only a few bits of memory as compared to the original string, this method results in the data being compressed.

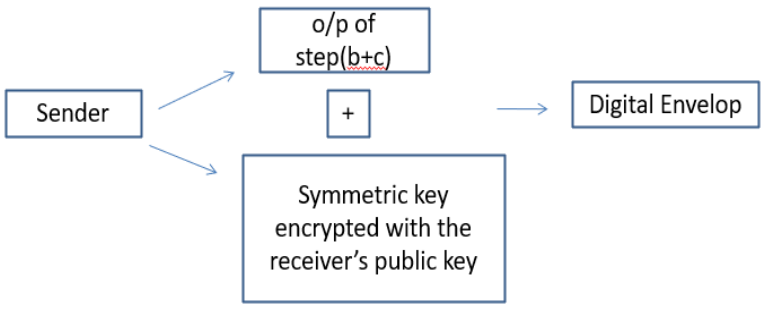


c. Encryption

The compressed output of step (b) are encrypted with a symmetric key. for this IDEA algorithm in CFB mode is used.

d. Digital Enveloping

In this the symmetric key used for encryption in step (c) is now encrypted with receiver's public key. The output of step (b+c) forms a digital envelop.

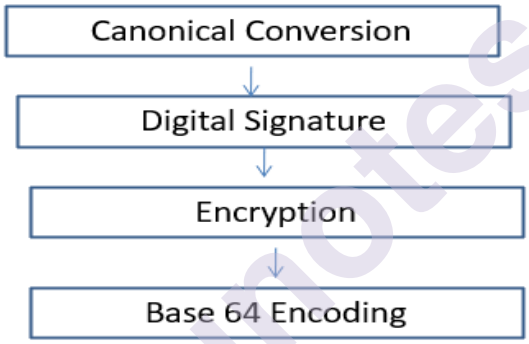


e. Base 64 Encoding

It is the last step. It transforms binary input into printable characters output. In this technique the binary input is processed using 24 bits. these 24 bits (8*3=24 bits) consist of 3 blocks with each block having 8 bits. this is converted into 8 bits in each block with total 3 blocks(8*3=24 bits).

Text	M								A								N							
ASCII	77								97								110							
Bit pattern (24 bts=8*3)	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
24 bits=4*6	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19								22								5							
Base64-encoded From predefined table	T								W								F							
																	U							

Overall steps:



7.11 UNIT END EXERCISE

1. Define CIA. Explain them.
2. What is Secure Socket Layer (SSL)/ Socket Layer (SL). What is the position of SSL in TCP/IP protocol suite?
3. Explain the working of SSL.
4. What is digital certificate. Describe all its fields.
5. Describe the Certificate Life cycle- Key generation, Registration, Verification, Certificate creation.
6. What is a Trust Model? Describe all the types.
7. What is PKI and What is it used for?
8. Explain PKI Deployment?
9. Explain Web Browser Security
10. How do to keep web security sanity?
11. What is Pretty Good Privacy? Describe its working

7.12 SUMMARY

This chapter discuss about the public key infrastructures, root certification authorities, web browser security and many more.

7.13 REFERENCE FOR FURTHER READING

Text book:

- Discrete Mathematics and Its Applications, Kenneth H. Rosen, 7th Edition, McGraw Hill, 2012.
- Cryptography Theory and Practice, 3rd Edition, Douglas R. Stinson, 2005.

Reference Book:

- Network Security and Cryptography, Atul Kahate, McGraw Hill, 2003.
- Cryptography and Network Security: Principles and Practices, William Stalling, Fourth Edition, Prentice Hall, 2013.
- Introduction to Cryptography with coding theory, second edition, Wade Trappe, Lawrence C. Washington, Pearson, 2005.

❄❄❄❄❄❄❄