Sarvasiddhant Education Society's
SWAMINARAYAN SIDDHANTA INSTITUTE OF TECHNOLOGY
Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University
Nagpur Katol Highway Road, Khapri (Kothe),
Tal.Kalmeshwar, Nagpur, Maharashtra 441501
Department Of Computer Engineering
Session 2023-2024

CNS NOTES

CRYPTOGRAPHY & NETWORK SECURITY

SEMESTER: 7TH SEM (FINAL YEAR)

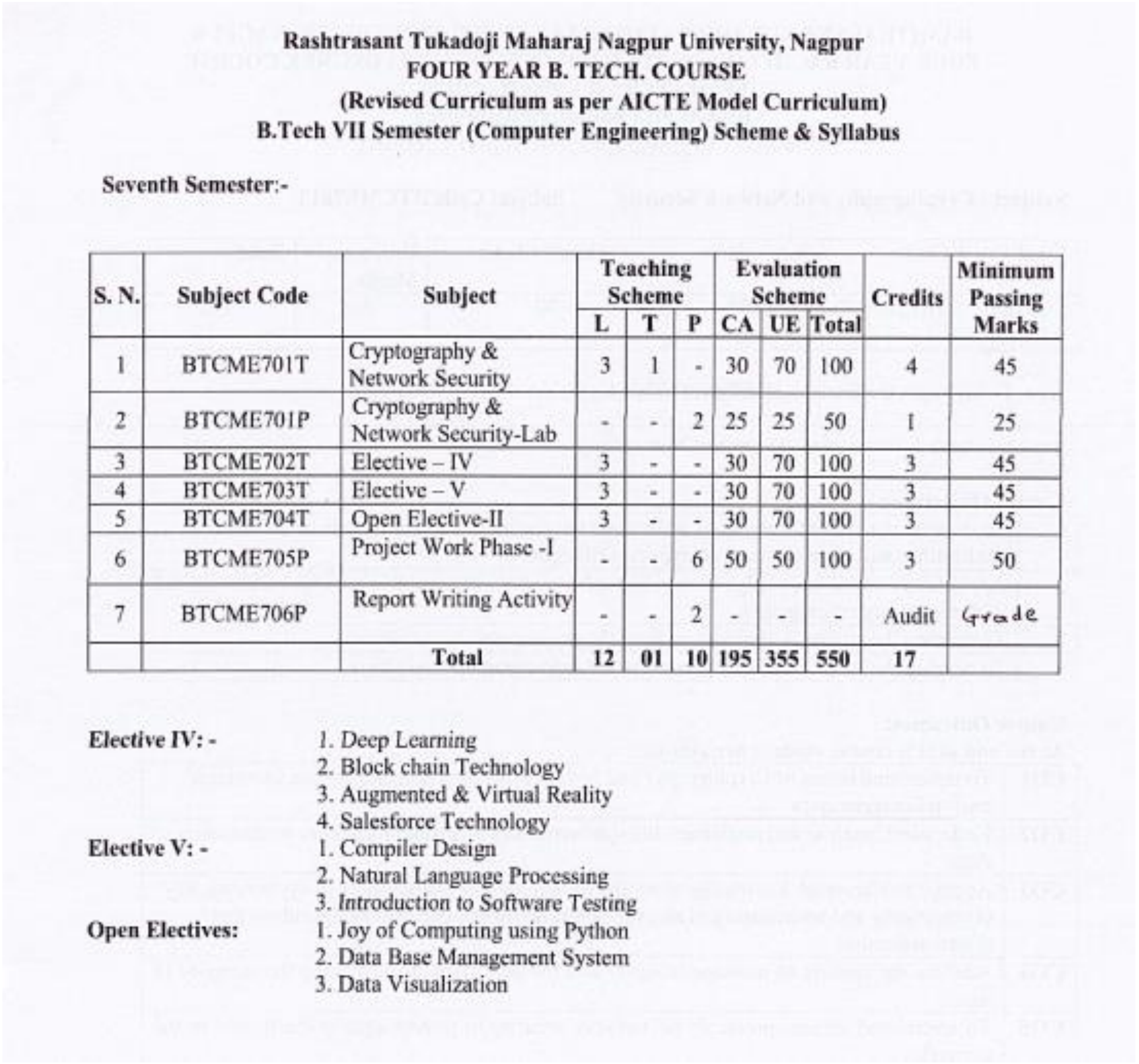
Subject Incharge: Prof. Ashvini Bais

Asst. Prof. (CE Dept.)

NOTES

CRYPTOGRAPHY & NETWORK SECURITY

Semester: 7th Sem (Final Year)

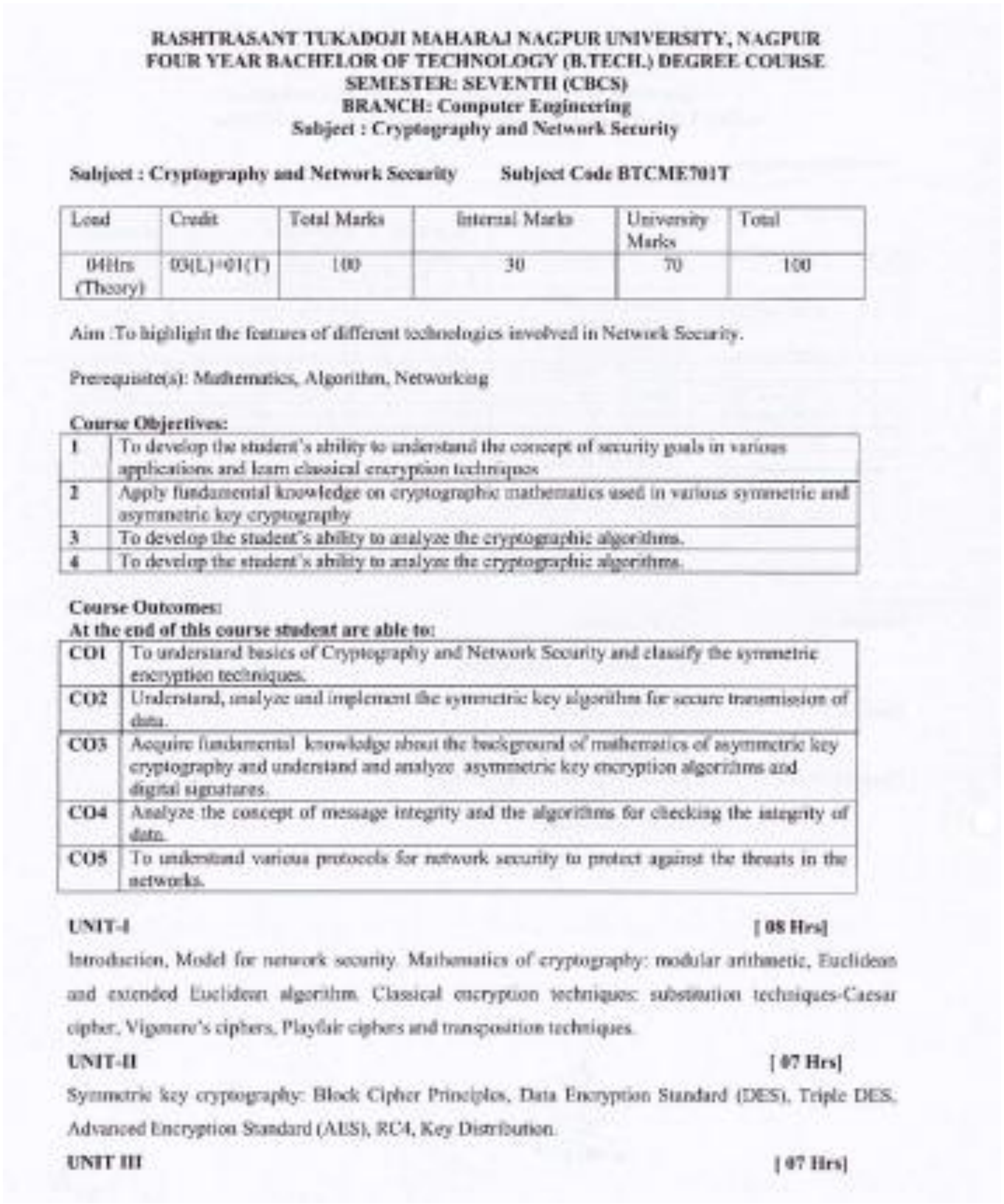


Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur
FOUR YEAR B. TECH. COURSE
(Revised Curriculum as per AICTE Model Curriculum)
B.Tech VII Semester (Computer Engineering) Scheme & Syllabus

Seventh Semester:-

S. N.	Subject Code	Subject	Teaching Scheme			Evaluation Scheme			Credits	Minimum Passing Marks
			L	T	P	CA	UE	Total		
1	BTCME701T	Cryptography & Network Security	3	1	-	30	70	100	4	45
2	BTCME701P	Cryptography & Network Security-Lab	-	-	2	25	25	50	1	25
3	BTCME702T	Elective – IV	3	-	-	30	70	100	3	45
4	BTCME703T	Elective – V	3	-	-	30	70	100	3	45
5	BTCME704T	Open Elective-II	3	-	-	30	70	100	3	45
6	BTCME705P	Project Work Phase -I	-	-	6	50	50	100	3	50
7	BTCME706P	Report Writing Activity	-	-	2	-	-	-	Audit	Grade
		Total	**12**	**01**	**10**	**195**	**355**	**550**	**17**	

Elective IV: -	1. Deep Learning
	2. Block chain Technology
	3. Augmented & Virtual Reality
	4. Salesforce Technology
Elective V: -	1. Compiler Design
	2. Natural Language Processing
	3. Introduction to Software Testing
Open Electives:	1. Joy of Computing using Python
	2. Data Base Management System
	3. Data Visualization



RASHTRASANT TUKADOJI MAHARAJ NAGPUR UNIVERSITY, NAGPUR
FOUR YEAR BACHELOR OF TECHNOLOGY (B.TECH.) DEGREE COURSE
SEMESTER: SEVENTH (CBCS)
BRANCH: Computer Engineering
Subject : Cryptography and Network Security

Subject : Cryptography and Network Security **Subject Code BTCME701T**

Load	Credit	Total Marks	Internal Marks	University Marks	Total
04Hrs (Theory)	03(L)+01(T)	100	30	70	100

Aim :To highlight the features of different technologies involved in Network Security.

Prerequisite(s): Mathematics, Algorithm, Networking

Course Objectives:

1	To develop the student's ability to understand the concept of security goals in various applications and learn classical encryption techniques
2	Apply fundamental knowledge on cryptographic mathematics used in various symmetric and asymmetric key cryptography
3	To develop the student's ability to analyze the cryptographic algorithms.
4	To develop the student's ability to analyze the cryptographic algorithms.

Course Outcomes:
At the end of this course student are able to:

CO1	To understand basics of Cryptography and Network Security and classify the symmetric encryption techniques.
CO2	Understand, analyze and implement the symmetric key algorithm for secure transmission of data.
CO3	Acquire fundamental knowledge about the background of mathematics of asymmetric key cryptography and understand and analyze asymmetric key encryption algorithms and digital signatures.
CO4	Analyze the concept of message integrity and the algorithms for checking the integrity of data.
CO5	To understand various protocols for network security to protect against the threats in the networks.

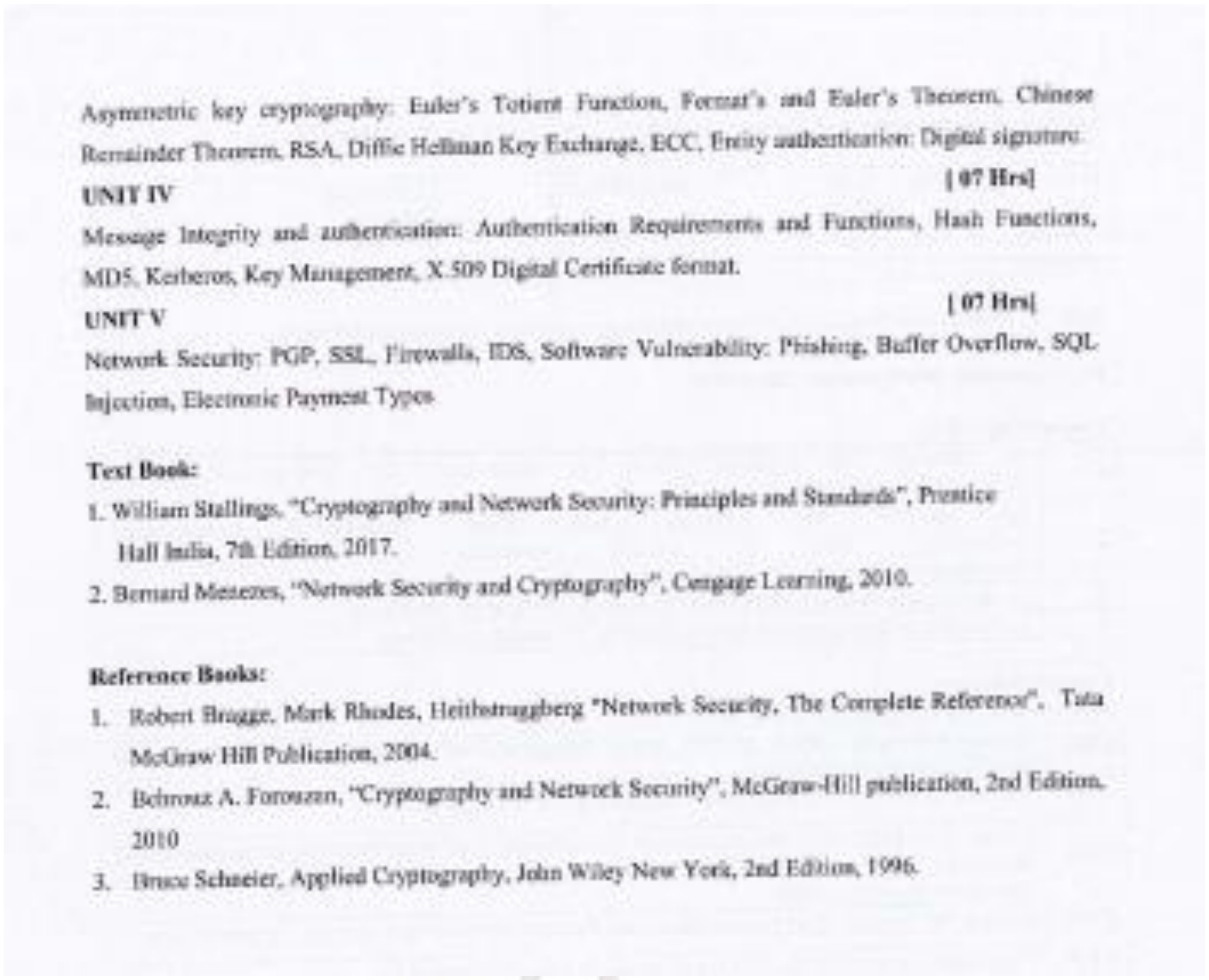
UNIT-I **[08 Hrs]**

Introduction, Model for network security. Mathematics of cryptography: modular arithmetic, Euclidean and extended Euclidean algorithm. Classical encryption techniques: substitution techniques-Caesar cipher, Vigenere's ciphers, Playfair ciphers and transposition techniques.

UNIT-II **[07 Hrs]**

Symmetric key cryptography: Block Cipher Principles, Data Encryption Standard (DES), Triple DES, Advanced Encryption Standard (AES), RC4, Key Distribution.

UNIT III **[07 Hrs]**



Asymmetric key cryptography: Euler's Totient Function, Fermat's and Euler's Theorem, Chinese Remainder Theorem, RSA, Diffie Hellman Key Exchange, ECC, Entity authentication: Digital signature

UNIT IV **[07 Hrs]**

Message Integrity and authentication: Authentication Requirements and Functions, Hash Functions, MD5, Kerberos, Key Management, X.509 Digital Certificate format.

UNIT V **[07 Hrs]**

Network Security: PGP, SSL, Firewalls, IDS, Software Vulnerability: Phishing, Buffer Overflow, SQL Injection, Electronic Payment Types

Text Book:

1. William Stallings, "Cryptography and Network Security: Principles and Standards", Prentice Hall India, 7th Edition, 2017.
2. Bernard Menezes, "Network Security and Cryptography", Cengage Learning, 2010.

Reference Books:

1. Robert Bragge, Mark Rhodes, Heithstraggberg "Network Security, The Complete Reference", Tata McGraw Hill Publication, 2004.
2. Behrouz A. Forouzan, "Cryptography and Network Security", McGraw-Hill publication, 2nd Edition, 2010
3. Bruce Schneier, Applied Cryptography, John Wiley New York, 2nd Edition, 1996.

UNIT III

Asymmetric Key Cryptography: Euler's Totient Function, Fermat's and Euler's Theorem, Chinese Remainder Theorem, RSA, Diffie Hellman Key Exchange, ECC, Entity Authentication: Digital Signature.

Asymmetric Key Cryptography

Euler's Totient Function-

It is denoted as 'ϕ(n)' and is commonly used to determine the number of prime numbers up to 'n'. This function provides insights into the properties and distribution of prime numbers, making it useful for analyzing number theory and cryptography.

Euler's totient function one may use to know how many prime numbers are coming up to the given integer 'n.' It is also called an arithmetic function. Two things are important for an application or use of Euler's totient function. One is that the gcd formed from the given integer 'n' should be multiplicative. The other is that the numbers of gcd should be the prime numbers only. The integer 'n' in this case should be more than 1. Calculating the Euler's totient function from a negative integer is impossible. The principle, in this case, is that for ϕ(n), the multiplicators called m and n should be greater than 1. Hence, denoted by 1<m<n and gcd (m, n) = 1. Sign ϕ is the sign used to denote the totient function.

Euler introduced this function in 1763. Initially, Euler used the Greek *π* for denotation of the function, but because of some issues, his denotation of Greek *π didn't* get recognition. And, he failed to give it the proper notation sign, i.e., ϕ. Hence, it cannot introduce the function. Further, ϕ was from Gauss's 1801 Disquisitiones Arithmeticae. The function is also known as the phi function. But J. J. Sylvester, in 1879, included the term totient for this function because of its properties and uses. The different rules deal with different kinds of integers, such as if integer p is a prime number, then which rule to apply, etc. Euler frames all the rules as practicable. Therefore, one can use it even today while dealing with the same.

Properties of Euler's Totient Function-

There are some different properties. Some of the properties of Euler's totient function are as under:

- Φ is the symbol used to denote the function.
- The function deals with the prime numbers theory.
- The function is applicable only in the case of positive integers.
- For ϕ (n), one can find two multiplicative prime numbers to calculate the function.
- The function is a mathematical function and useful in many ways.

- If integer ‘n’ is a prime number, then gcd (m, n) = 1.
- The function works on the formula $1< m< n$, where m and n are the prime and multiplicative numbers.
- In general, the equation is:

$$\mathbf{\Phi\ (mn) = \phi\ (m) * \phi\ (n)\ (1- 1 / m)\ (1 - 1/ n)}$$

- The function counts the number of positive integers less than the given integer, which is relatively prime numbers to the given integer.
- If the given integer p is prime, then $\phi\ (p) = p - 1$
- If the power of p is prime, then if $a = p^n$ is a prime power, then $\phi\ (p^n) = p^n - p^{(n-1)}$
- ϕ (n) is not one – one
- ϕ (n) is not onto.
- ϕ (n), $n > 3$ is always even.
- $\phi(10^n) = 4 * 10^{n-1}$

For Example- Calculate ϕ (7)?

Solution:

ϕ (7) = (1,2,3,4,5,6) = **6**

As all numbers are prime to 7, it made it easy to calculate the ϕ.

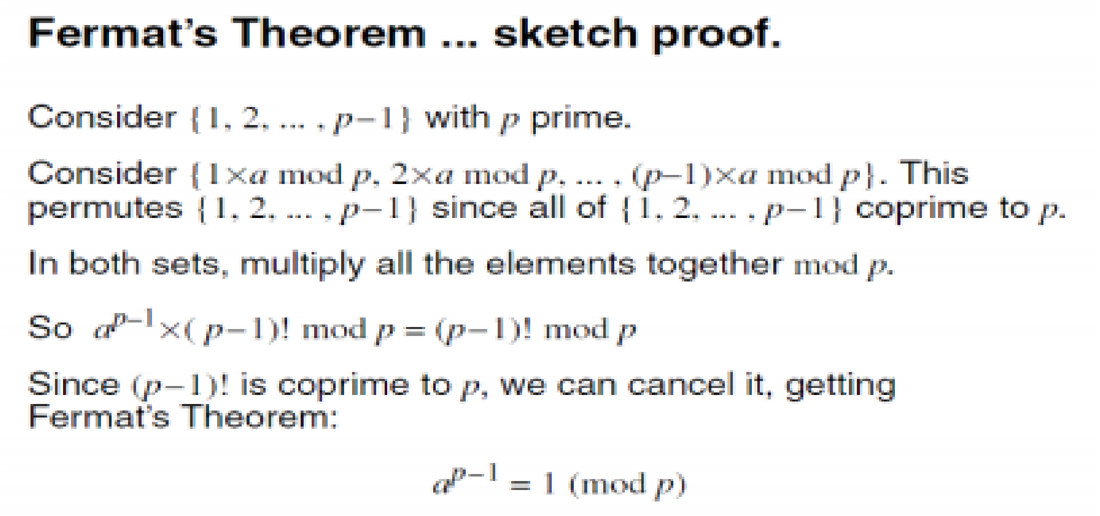
Applications-

- One may use the function to define the RSA encryption system used for internet security encryption.
- One may use it in prime numbers theory.
- One may use it in large calculations also.
- One may use it in applications of elementary number theory.

Fermat’s Theorem-

Fermat's “little” theorem states that if p is prime, then ap ≡ a (mod p) for all a. An alter- native form states that ap−1 ≡ 1 (mod p) when p is prime and a is any integer not divisible by p. (This last condition is needed for the alternative form, but not for the usual form.)

- ap-1 = 1 (mod p) – where p is prime and GCD(a,p) = 1
- Also known as Fermat’s Little Theorem
- Also have: ap = a (mod p)
- Useful in public key and primality testing



Fermat's Theorem ... sketch proof.

Consider $\{1, 2, \ldots, p-1\}$ with p prime.

Consider $\{1\times a \bmod p, 2\times a \bmod p, \ldots, (p-1)\times a \bmod p\}$. This permutes $\{1, 2, \ldots, p-1\}$ since all of $\{1, 2, \ldots, p-1\}$ coprime to p.

In both sets, multiply all the elements together $\bmod p$.

So $a^{p-1}\times(p-1)! \bmod p = (p-1)! \bmod p$

Since $(p-1)!$ is coprime to p, we can cancel it, getting Fermat's Theorem:

$$a^{p-1} = 1 \pmod p$$

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

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

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

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

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

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

a x 2a x …. x ((P-1)a)= [(a mod P) x (2a mod P) x …. x ((P-1) a mod P)] mod P

=[1 x 2 x…x (P-1)] mod P

= (P-1)! mod P

But

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

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

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

Consider the set of positive integers less than p: {1, 2... p1} and multiply each element by a, modulo p, to receive the set X = {a mod p, 2a mod p . . . (p1) a mod p}. None of the elements of X is similar to zero because p does not divide a. Furthermore no two of the integers in X are same. To see this, consider that (ja ≡ p) where 1 ≤ p1. Because p, it can delete a from both sides of the equation resulting in − j≡ p). This final similarity is inaccessible due to j and k are both positive integers less than p. Therefore, it is understand that the (p1) elements of X are all positive integers, with no two elements same.

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

Therefore, $3^{10} \equiv 1(\bmod 11)$.

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

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

Example 1 − Find the result of 6^{10} mod 11.

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

Example 2 − Find the result of 3^{12} mod 11.

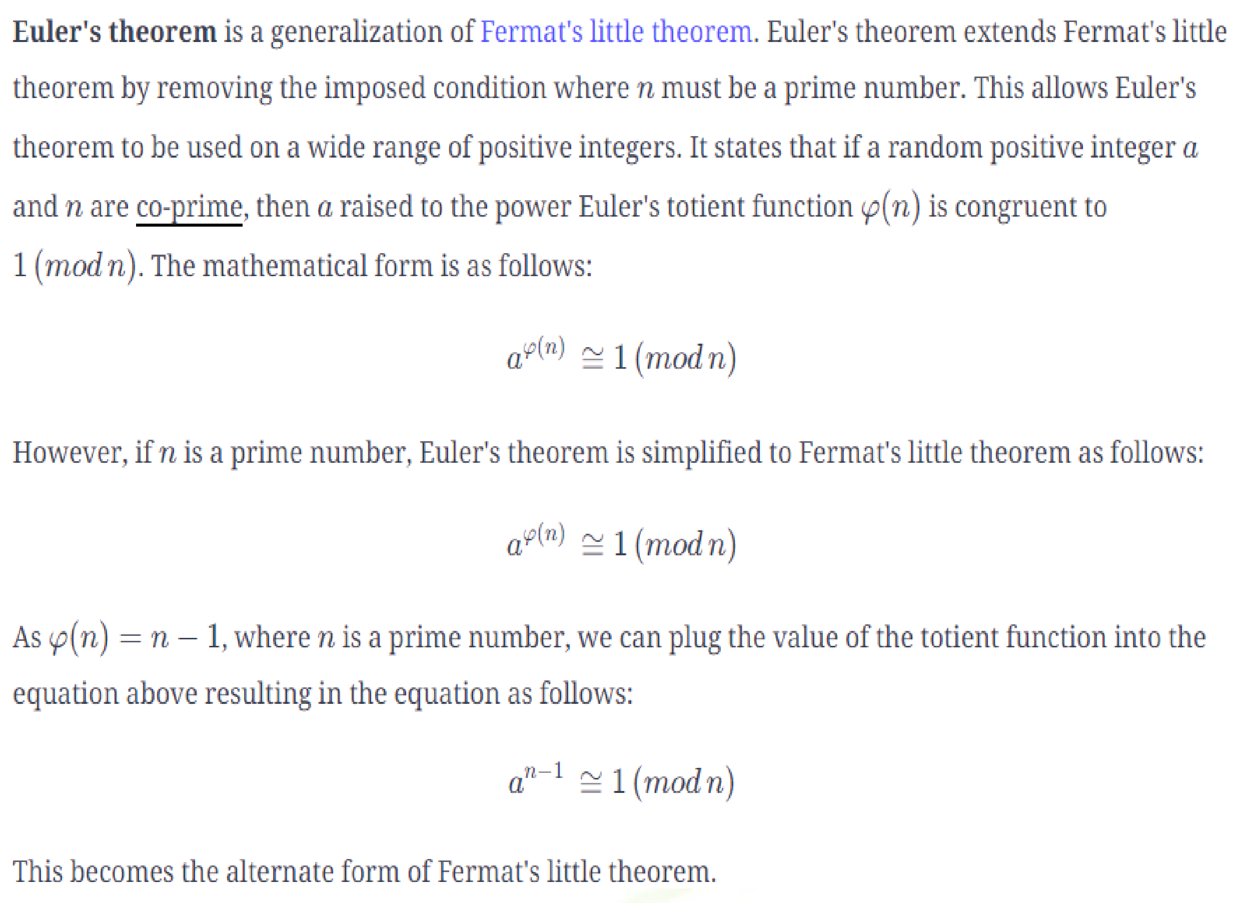
Therefore the exponent (12) and the modulus (11) are not equal. With substitution this can be defined utilizing Fermat's little theorem.

$3^{12} \bmod 11 = (3^{11} x 3) \bmod 11 = (3^{11} \bmod 11)(3 \bmod 11) = (3x3) \bmod 11 = 9$

Fermat's theorem is used as an underlying base for Euler's theorem. Euler's theorem is a generalization of Fermat's theorem. It extends Fermat's theorem, which allows it to be used as the base for the RSA algorithm. Thus, Fermat's theorem is a core part of modern-era cryptography.

Euler’s Theorem-

Euler Theorem deals with the concept of prime numbers, modulus/remainder, & congruency. It aims to provide a concept where coprime numbers can be correlated somehow to provide a value that can be used later as a hash value or for encryption key in cryptography. Euler's theorem is a generalization of Fermat's little theorem handling with powers of integers modulo positive integers. It increase in applications of elementary number theory, such as the theoretical supporting structure for the RSA cryptosystem.



Euler's theorem is a generalization of Fermat's little theorem. Euler's theorem extends Fermat's little theorem by removing the imposed condition where n must be a prime number. This allows Euler's theorem to be used on a wide range of positive integers. It states that if a random positive integer a and n are co-prime, then a raised to the power Euler's totient function $\varphi(n)$ is congruent to $1\,(mod\,n)$. The mathematical form is as follows:

$$a^{\varphi(n)} \cong 1\,(mod\,n)$$

However, if n is a prime number, Euler's theorem is simplified to Fermat's little theorem as follows:

$$a^{\varphi(n)} \cong 1\,(mod\,n)$$

As $\varphi(n) = n - 1$, where n is a prime number, we can plug the value of the totient function into the equation above resulting in the equation as follows:

$$a^{n-1} \cong 1\,(mod\,n)$$

This becomes the alternate form of Fermat's little theorem.

First version

The first version of Euler’s theorem is similar to the first version of the Fermat’s little theorem. if a and n are coprime,

Then Let a and m be coprime.

Then a φ(n) = 1 (mod n). The derivation of the Euler's formula for φ(n) proceeds in two steps. First, we consider the next simplest case φ(pa), where p is prime. Next, we establish the multiplicative property of φ: φ(n1n2) = φ(n1)φ(n2) for coprime m1 and m2. Since any integer can be (uniquely) represented in the form n = p1 a 1p2 a 2 ... pk a k, with distinct pi's, these two steps combined will lead to a closed form expression for φ.

$R = \{x_1, x_2, \ldots x_{\phi(n)}\}$, i.e., each element xi of R is unique positive integer less than n with ged(x_i, n) = 1. Then multiply each element by a and modulo n –

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

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

There are no duplicates in S.

If ax_i mod n and n = ax_j mod n then $x_i = x_j$

Therefore,

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

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

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

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

Example -Find the result of 6^{24} mod 35.

Solution: We have 6^{24} mod 35 = $6^{f(35)}$ mod 35 = 1.

Application of Euler's theorem-

Euler's theorem and Euler's totient function serve as a base for the modern RSA encryption algorithm. Euler's theorem is involved in the following processes of the RSA algorithm.

- Key generation process
- Encryption
- Decryption

Chinese Remainder Theorem-

The Chinese remainder theorem is a theorem of number theory, which states that if one knows the remainders of the Euclidean division of an integer *n* by several integers, then one can determine uniquely the remainder of the division of *n* by the product of these integers, under the condition that the divisors are pairwise coprime.

The theorem was first discovered in the 3rd century AD by the Chinese mathematician Sunzi in Sunzi Suanjing. The Chinese remainder theorem is widely used for computing with large integers, as it allows replacing a computation for which one knows a bound on the size of the result by several similar computations on small integers. The Chinese remainder theorem (expressed in terms of congruences) is true over every principal ideal domain. It has been generalized to any commutative ring, with a formulation involving ideals.

Let $n_1, \ldots, n_i, \ldots, n_k$ be integers greater than 1, which are often called moduli or divisors. Let us denote by N the product of the n_i. The Chinese remainder theorem asserts that if the n_i are pairwise coprime, and if $a_1, \ldots, a_k$ are integers such that $0 \leq a_i < n_i$ for every i, then there is one and only one integer x, such that $0 \leq x < N$ and the remainder of the Euclidean division of x by n_i is a_i for every i.

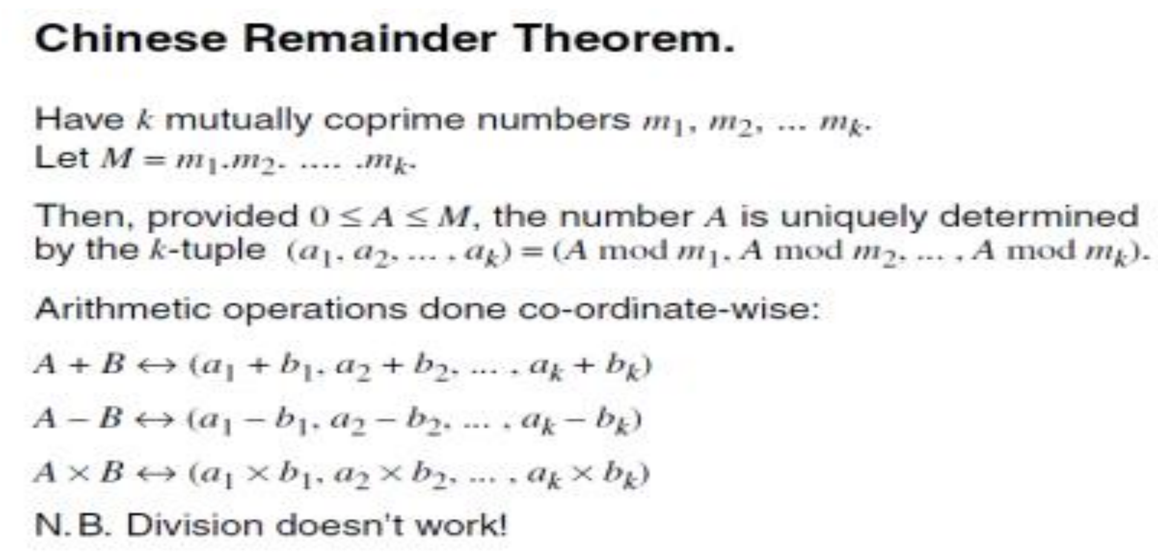
This may be restated as follows in term of congruences: If the n_i are pairwise coprime, and if $a_1, \ldots, a_k$ are any integers, then there exists an integer x such that and any two such x are congruent modulo N.

Chinese Remainder Theorem :

- Used to speed up modulo computations
- If working modulo a product of numbers – eg. mod M = m1m2..mk
- Chinese Remainder theorem lets us work in each modulus mi separately
- Since computational cost is proportional to size, this is faster than working in the full modulus M Chinese Remainder Theorem
- Can implement CRT in several ways

The Chinese remainder theorem asserts that if the ni are pairwise coprime, and if a1, ..., ak are integers such that 0 ≤ ai < ni for every i, then there is one and only one integer x, such that 0 ≤ x < N and the remainder of the Euclidean division of x by niis ai for every i.

To compute A (mod M) – first compute all ai = A mod mi separately – determine constants ci below, where Mi = M/mi – then combine results to get answer using:



Chinese Remainder Theorem.

Have k mutually coprime numbers $m_1, m_2, \dots m_k$.
Let $M = m_1.m_2. \dots .m_k$.

Then, provided $0 \le A \le M$, the number A is uniquely determined by the k-tuple $(a_1, a_2, \dots, a_k) = (A \bmod m_1, A \bmod m_2, \dots, A \bmod m_k)$.

Arithmetic operations done co-ordinate-wise:

$A + B \leftrightarrow (a_1 + b_1, a_2 + b_2, \dots, a_k + b_k)$

$A - B \leftrightarrow (a_1 - b_1, a_2 - b_2, \dots, a_k - b_k)$

$A \times B \leftrightarrow (a_1 \times b_1, a_2 \times b_2, \dots, a_k \times b_k)$

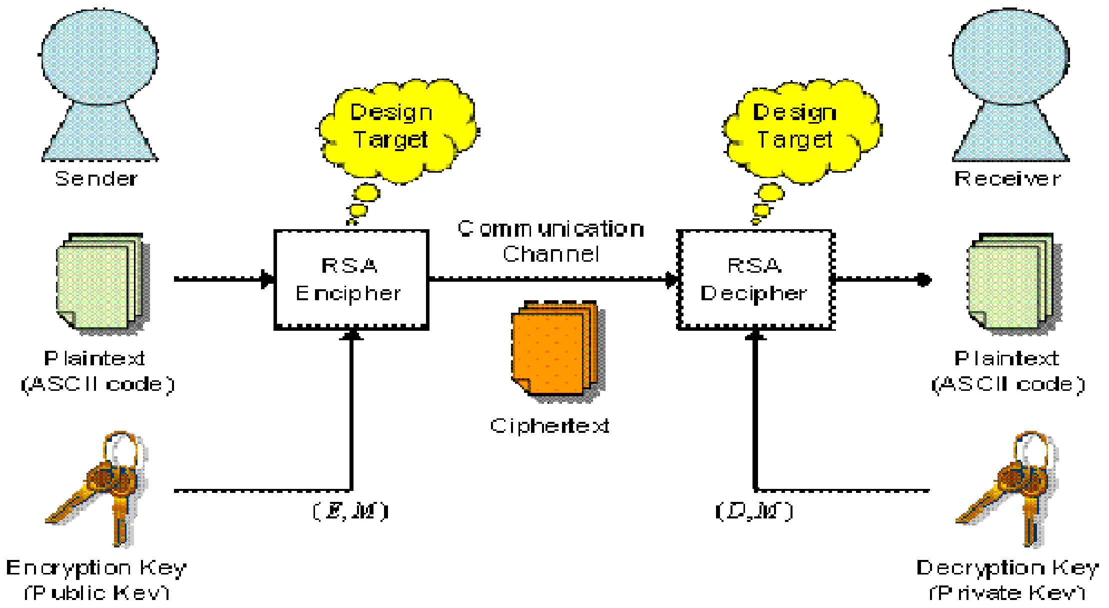
N.B. Division doesn't work!

RSA Algorithm-

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, but this was not declassified until 1997.

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, and if the public key is large enough, only someone with knowledge of the prime numbers can decode the message feasibly. Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem remains an open question.

RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed.



Generation of RSA Key Pair-

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below −

- **Generate the RSA modulus (n)**
  - Select two large primes, p and q.
  - Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.
- **Find Derived Number (e)**
  - Number **e** must be greater than 1 and less than (p − 1)(q − 1).
  - There must be no common factor for e and (p − 1)(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.
- **Form the public key**
  - The pair of numbers (n, e) form the RSA public key and is made public.
  - Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.
- **Generate the private key**

- Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
- Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1).
- This relationship is written mathematically 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.

Example:

An example of generating RSA Key pair is given below. (For ease of understanding, the primes p & q taken here are small values. Practically, these values are very high).

- Let two primes be p = 7 and q = 13. Thus, modulus n = pq = 7 x 13 = 91.
- Select e = 5, which is a valid choice since there is no number that is common factor of 5 and (p − 1)(q − 1) = 6 × 12 = 72, except for 1.
- The pair of numbers (n, e) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input p = 7, q = 13, and e = 5 to the Extended Euclidean Algorithm. The output will be d = 29.
- Check that the d calculated is correct by computing −

$$de = 29 \times 5 = 145 = 1 \bmod 72$$

- Hence, public key is (91, 5) and private keys is (91, 29).

Encryption and Decryption-

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy.

Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n. Hence, it is necessary to represent the plaintext as a series of numbers less than n.

RSA Encryption:

- Suppose the sender wish to send some text message to someone whose public key is (n,e).
- The sender then represents the plaintext as a series of numbers less than n.
- To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as −

$$C = P^e \bmod n$$

- In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n.
- Returning to our Key Generation example with plaintext P = 10, we get ciphertext C −

$$C = 10^5 \bmod 91$$

RSA Decryption:

- The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a cipher text C.
- Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.

$$\text{Plaintext} = C^d \bmod n$$

- Returning again to our numerical example, the ciphertext C = 82 would get decrypted to number 10 using private key 29 −

$$\text{Plaintext} = 82^{29} \bmod 91 = 10$$

RSA Analysis:

The security of RSA depends on the strengths of two separate functions. The RSA cryptosystem is most popular public-key cryptosystem strength of which is based on the practical difficulty of factoring the very large numbers.

- **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.

If either of these two functions are proved non one-way, then RSA will be broken. In fact, if a technique for factoring efficiently is developed then RSA will no longer be safe.

The strength of RSA encryption drastically goes down against attacks if the number p and q are not large primes and/ or chosen public key e is a small number.

Diffie Hellman Key Exchange-

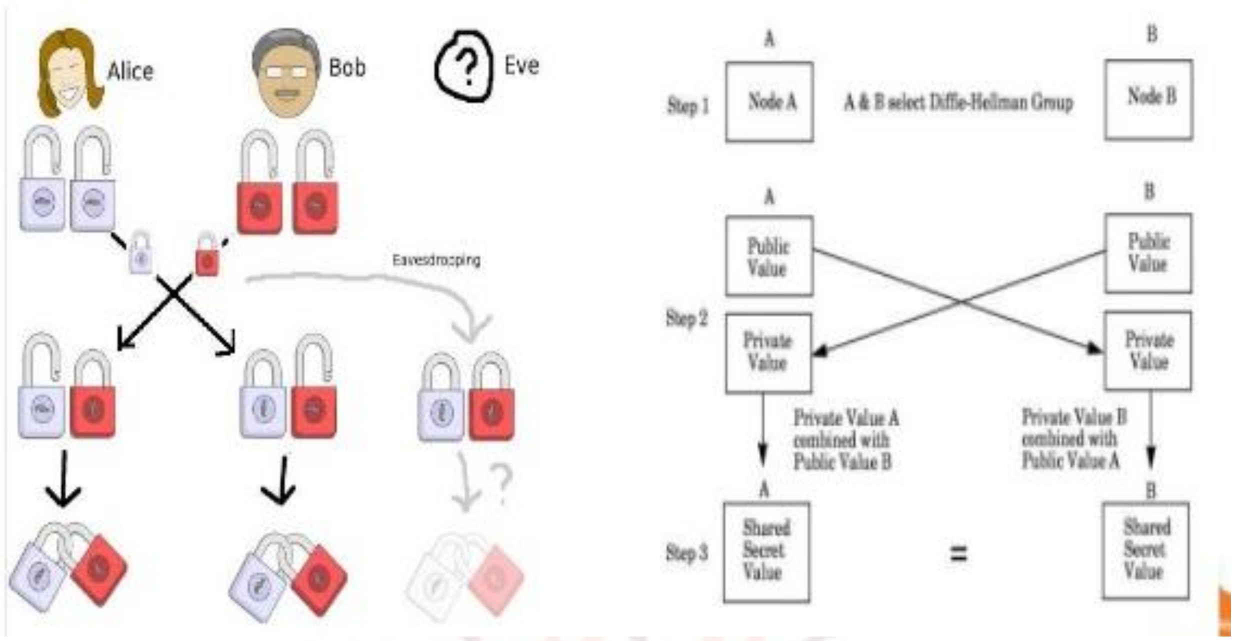
Diffie-Hellman key exchange is a method of digital encryption that securely exchanges cryptographic keys between two parties over a public channel without their conversation being transmitted over the internet. The two parties use symmetric cryptography to encrypt and decrypt their messages. Diffie-Hellman key exchange raises numbers to a selected power to produce decryption keys. The components of the keys are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming. The method doesn't share information during the key exchange. The two parties have no prior knowledge of each other, but the two parties create a key together.

Where is Diffie-Hellman key exchange used?

Diffie-Hellman key exchange's goal is to securely establish a channel to create and share a key for symmetric key algorithms. Generally, it's used for encryption, password-authenticated key agreement and forward security. Password-authenticated key agreements are used to prevent man-in-the-middle (MitM) attacks. Forward secrecy-based protocols protect against the compromising of keys by generating new key pairs for each session.

Diffie-Hellman key exchange is commonly found in security protocols, such as Transport Layer Security (TLS), Secure Shell (SSH) and IP Security (IPsec). For example, in IPsec, the encryption method is used for key generation and key rotation. Even though Diffie-Hellman key exchange can be used for establishing both public and private keys, the Rivest-Shamir-Adleman algorithm, or RSA algorithm, can also be used, since it's able to sign public key certificates.

Diffie –Hellman Key Exchange


Alice
Bob
Eve
Eavesdropping
?
A
B
Step 1
Node A
A & B select Diffie-Hellman Group
Node B
Step 2
Public Value
Private Value
Private Value A combined with Public Value B
Private Value B combined with Public Value A
Step 3
Shared Secret Value
=

How does Diffie-Hellman key exchange work?

To implement Diffie-Hellman, two end users, Alice and Bob, mutually agree on positive whole numbers *p* and *q*, such that *p* is a prime number and *q* is a generator of *p*. The generator *q* is a number that, when raised to positive whole-number powers less than *p*, never produces the same result for any two such whole numbers. The value of *p* may be large, but the value of *q* is usually small. Once Alice and Bob have agreed on *p* and *q* in private, they choose positive whole-number personal keys *a* and *b*. Both are less than the prime number modulus *p*. Neither user divulges their personal key to anyone; ideally, they memorize these numbers and don't write them down or store them anywhere. Next, Alice and Bob compute public keys *a** and *b** based on their personal keys according to the following formulas:

$a^* = q_a \bmod p$

$b^* = q_b \bmod p$

The two users can share their public keys *a** and *b** over a communications medium assumed to be insecure, such as the internet or a corporate wide area network. From these public keys, a number *x* can be generated by either user on the basis of their own personal keys. Alice computes *x* using the following formula:

x = (*b**) mod *p*

Bob computes *x* using the following formula:

x = (*a**) mod *p*

The value of *x* turns out to be the same according to either of the above two formulas. However, the personal keys *a* and *b*, which are critical in the calculation of *x*, haven't been transmitted over a public medium. Because it's a large and apparently random number, a potential hacker has almost no chance of correctly guessing *x*, even with the help of a powerful computer to conduct millions of trials. The two users can, therefore, in theory, communicate privately over a public medium with an encryption method of their choice using the decryption key *x*.

Examples of Diffie-Hellman key exchange

If two people, say Alice and Bob, want to communicate sensitive data over an open public network but want to avoid hackers or eavesdroppers, they can use Diffie-Hellman key exchange method for encryption. This open public network could be at a cafe, for example.

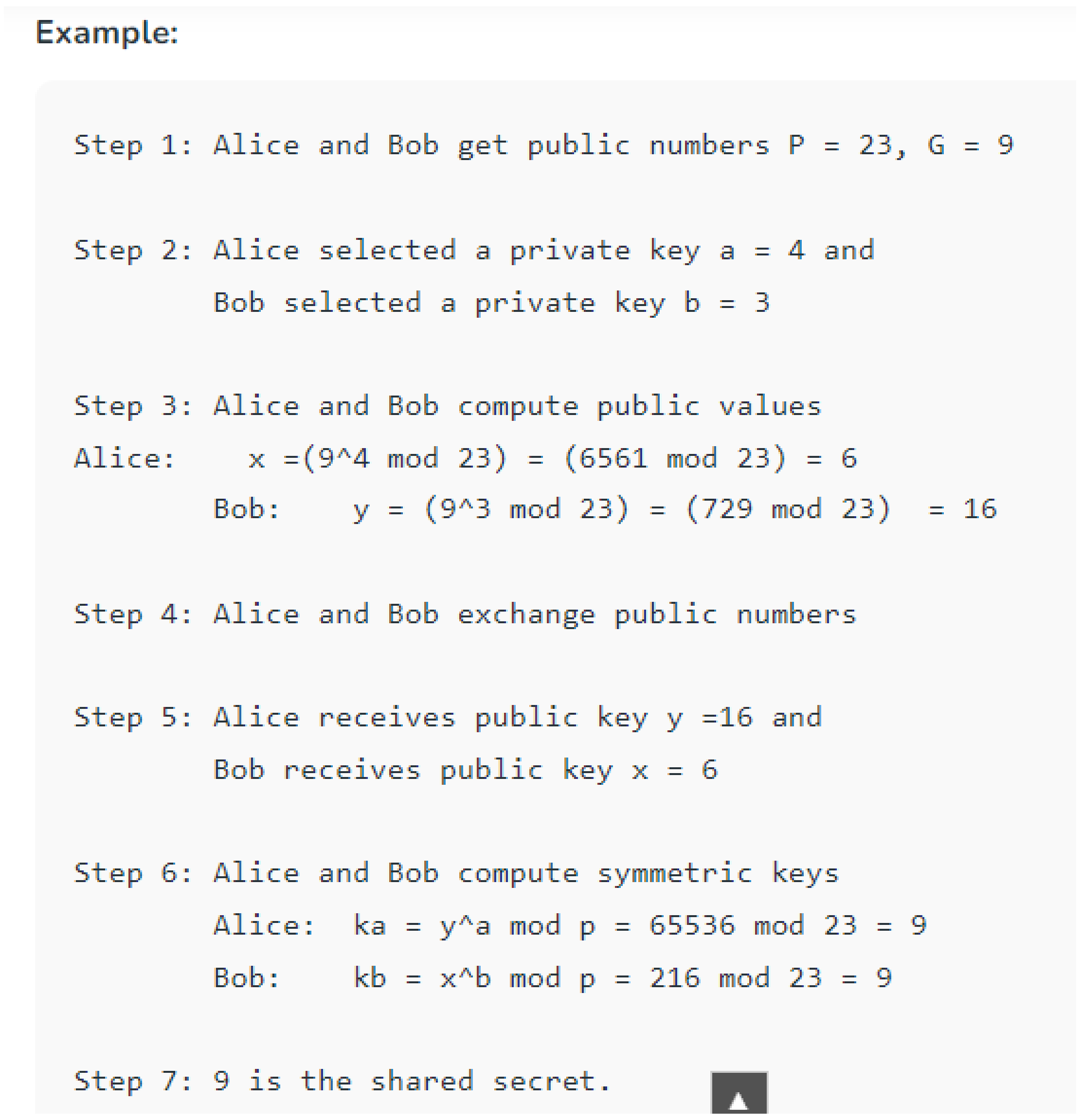
Alice and Bob privately choose a secret key, and a function is run on these keys to create a public key. The results -- and not the function -- are shared. Even if a third party is listening in, that third party won't have all the involved numbers, making it difficult to derive the function the numbers came from.

From here, Alice and Bob each run a new function using the results they received from the opposite party, their own secret number and the original prime value. Alice and Bob then arrive at a common shared secret key that a third party can't deduce. Alice and Bob are now free to communicate without worrying about third parties.

Diffie-Hellman algorithm:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

- For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.



Example:

```
Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
        Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice:    x =(9^4 mod 23) = (6561 mod 23) = 6
        Bob:    y = (9^3 mod 23) = (729 mod 23)  = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
        Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
        Alice:  ka = y^a mod p = 65536 mod 23 = 9
        Bob:    kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.
```

Elliptic Curve Cryptography (ECC)-

Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. They are also used in several integer factorization algorithms based on elliptic curves that have applications in cryptography, such as Lenstra elliptic-curve factorization.

Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers. The technology can be used in conjunction with most public key encryption methods, such as RSA, and Diffie-Hellman. According to some researchers, ECC can yield a level of security with a 164-bit key that other systems require a 1,024-bit key to achieve. Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for mobile applications. ECC was developed by Certicom, a mobile e-business security provider, and was recently licensed by Hifn, a manufacturer of integrated circuitry (IC) and network security products. RSA has been developing its own version of ECC. Many manufacturers, including 3COM, Cylink, Motorola, Pitney Bowes, Siemens, TRW, and VeriFone have included support for ECC in their products.

The properties and functions of elliptic curves have been studied in mathematics for 150 years. Their use within cryptography was first proposed in 1985, (separately) by Neal Koblitz from the University of Washington, and Victor Miller at IBM. An elliptic curve is not an ellipse (oval shape), but is represented as a looping line intersecting two axes (lines on a graph used to indicate the position of a point). ECC is based on properties of a particular type of equation created from the mathematical group (a set of values for which operations can be performed on any two members of the group to produce a third member) derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that is very valuable for cryptography purposes: they are relatively easy to perform, and extremely difficult to reverse.

The industry still has some reservations about the use of elliptic curves. Nigel Smart, a Hewlett Packard researcher, discovered a flaw in which certain curves are extremely vulnerable. However, Philip Deck of Certicom says that, while there are curves that are vulnerable, those implementing ECC would have to know which curves could not be used. He believes that ECC offers a unique potential as a technology that could be implemented worldwide and across all devices. According to Deck (quoted in Wired), "the only way you can achieve that is with elliptic curve." Elliptic Curve Cryptography (ECC) is a term used to describe a suite of cryptographic tools and protocols whose security is based on special versions of the discrete logarithm problem. It does not use numbers modulo p.

ECC is based on sets of numbers that are associated with mathematical objects called elliptic curves. There are rules for adding and computing multiples of these numbers, just as there are for numbers modulo p. ECC includes avariants of many cryptographic schemes that were initially designed for modular numbers such as ElGamal encryption and Digital Signature Algorithm.

It is believed that the discrete logarithm problem is much harder when applied to points on an elliptic curve. This prompts switching from numbers modulo p to points on an elliptic curve. Also an equivalent security level can be obtained with shorter keys if we use elliptic curve-based variants.

The shorter keys result in two benefits −

- Ease of key management

- Efficient computation

These benefits make elliptic-curve-based variants of encryption scheme highly attractive for application where computing resources are constrained.

Encryption algorithms-

- **Elliptic Curve Integrated Encryption Scheme (ECIES):** ECIES is a public-key authenticated encryption scheme that uses a KDF (key-derivation function) to generate a separate Medium Access Control key and symmetric encryption key from the ECDH shared secret. Because the ECIES algorithm incorporates a symmetric cipher, it can encrypt any amount of data. In practice, ECIES is used by standards such as Intelligent Transportation Systems.
- **EC-based ElGamal Elliptic Curve Cryptography:** ElGamal Elliptic Curve Cryptography is the public key cryptography equivalent of ElGamal encryption schemes that employ the Elliptic Curve Discrete Logarithm Problem. ElGamal is an asymmetric encryption algorithm that is used to send messages securely over long distances. Unfortunately, if the encrypted message is short enough, the algorithm is vulnerable to a Meet in the Middle attack.

Application of Elliptic Curve Cryptography-

- **Diffie-Hellman:** The basic public-key cryptosystem suggested for secret key sharing is the Diffie-Hellman protocol. If A (Alice) and B (Bob) initially agree on a given curve, field size, and mathematical type. They then distribute the secret key in the following manner. We can see that all we need to build the Diffie-Hellman protocol is scalar multiplication.

- **Elliptic Curve Digital Signature Algorithm (ECDSA):** ECC is one of the most widely utilized digital signature implementation approaches in cryptocurrencies. In order to sign transactions, both Bitcoin and Ethereum use the field inverse multiplication, but also arithmetic multiplication, inverse function, and modular operation.

- **Online application:** Moreover, ECC is not limited to cryptocurrencies. It is an encryption standard that will be utilized by most online apps in the future due to its reduced key size and efficiency. Most commonly used in cryptocurrencies such as Bitcoin and Ethereum, along with single-way encryption of emails, data, and software.

- **Blockchain application:** The cryptocurrency Bitcoin employs elliptic curve cryptography. Ethereum 2.0 makes heavy use of elliptic curve pairs with BLS signatures, as stated in the IETF proposed BLS specification, to cryptographically ensure that a specific Eth2 validator has really verified a specific transaction.

Entity Authentication

Digital Signature-

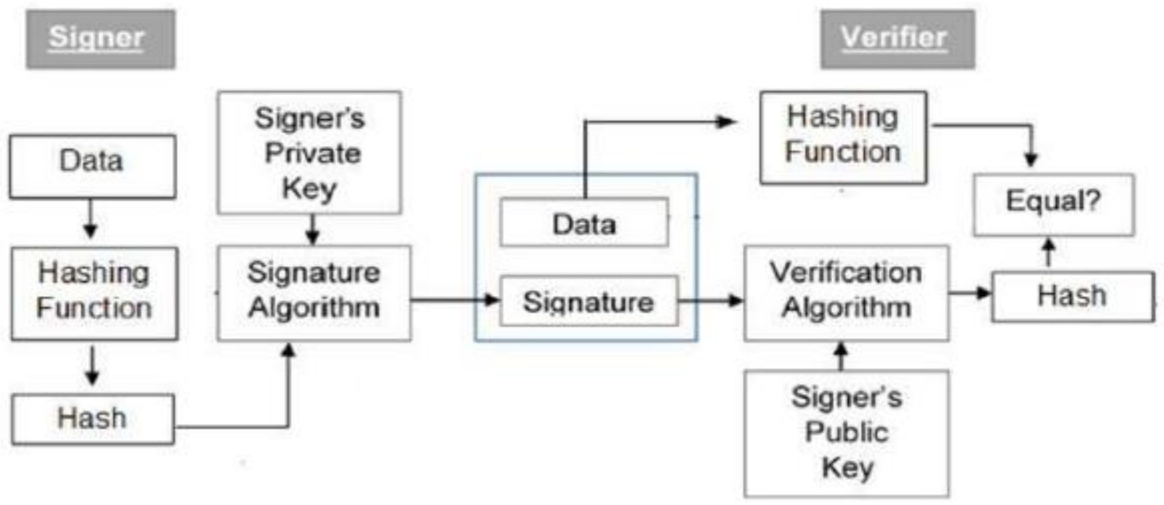
A digital signature is a cryptographic output used to verify the authenticity of data. A digital signature algorithm allows for two distinct operations: a signing operation, which uses a signing key to produce a signature over raw data.

A digital signature is a mathematical technique which validates the authenticity and integrity of a message, software or digital documents. It allows us to verify the author name, date and time of signatures, and authenticate the message contents. The digital signature offers far more inherent security and intended to solve the problem of tampering and impersonation (Intentionally copy another person's characteristics) in digital communications.

The computer-based business information authentication interrelates both technology and the law. It also calls for cooperation between the people of different professional backgrounds and areas of expertise. The digital signatures are different from other electronic signatures not only in terms of process and result, but also it makes digital signatures more serviceable for legal purposes. Some electronic signatures that legally recognizable as signatures may not be secure as digital signatures and may lead to uncertainty and disputes.

Model of Digital Signature-

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration −



The following points explain the entire process in detail −

- Each person adopting this scheme has a public-private key pair.

- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence signing a hash is more efficient than signing the entire data.

<u>Importance of Digital Signature-</u>

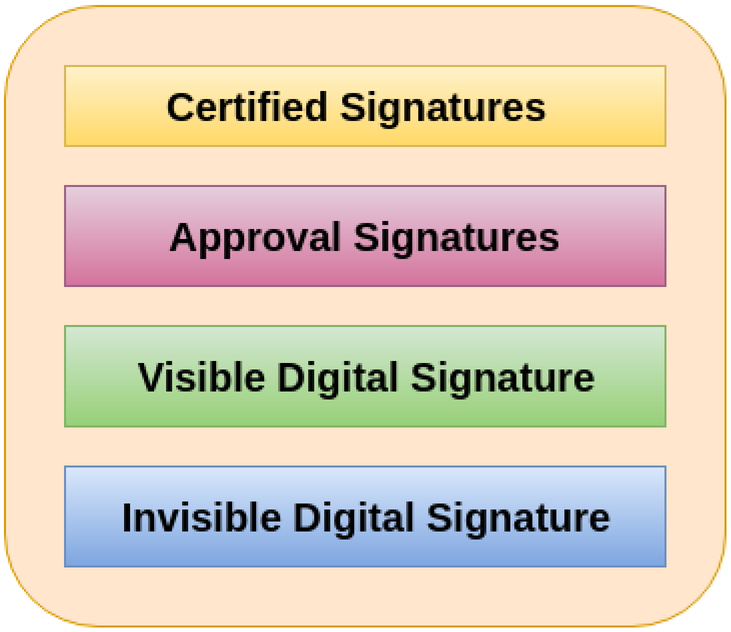
Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security. Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature −

- Message authentication − When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.

- Data Integrity − In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.

- Non-repudiation – Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

Types of Digital Signature-



Types of Digital Signature

1. **Certified Signatures**

The certified digital signature documents display a unique blue ribbon across the top of the document. The certified signature contains the name of the document signer and the certificate issuer which indicate the authorship and authenticity of the document.

2. **Approval Signatures**

The approval digital signatures on a document can be used in the organization's business workflow. They help to optimize the organization's approval procedure. The procedure involves capturing approvals made by us and other individuals and embedding them within the PDF document. The approval signatures to include details such as an image of our physical signature, location, date, and official seal.

3. **Visible Digital Signature**

The visible digital signature allows a user to sign a single document digitally. This signature appears on a document in the same way as signatures are signed on a physical document.

4. **Invisible Digital Signature**

The invisible digital signatures carry a visual indication of a blue ribbon within a document in the taskbar. We can use invisible digital signatures when we do not have or do not want to display our signature but need to provide the authenticity of the document, its integrity, and its origin.

Digital Signature-

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software, or digital document.

1. **Key Generation Algorithms**: Digital signature is electronic signatures, which assure that the message was sent by a particular sender. While performing digital transactions authenticity and integrity should be assured, otherwise, the data can be altered or someone can also act as if he was the sender and expect a reply.

2. **Signing Algorithms**: To create a digital signature, signing algorithms like email programs create a one-way hash of the electronic data which is to be signed. The signing algorithm then encrypts the hash value using the private key (signature key). This encrypted hash along with other information like the hashing algorithm is the digital signature. This digital signature is appended with the data and sent to the verifier. The reason for encrypting the hash instead of the entire message or document is that a hash function converts any arbitrary input into a much shorter fixed-length value. This saves time as now instead of signing a long message a shorter hash value has to be signed and moreover hashing is much faster than signing.

3. **Signature Verification Algorithms** : Verifier receives Digital Signature along with the data. It then uses Verification algorithm to process on the digital signature and the public key (verification key) and generates some value. It also applies the same hash function on the received data and generates a hash value. Then the hash value and the output of the verification algorithm are compared. If they both are equal, then the digital signature is valid else it is invalid.

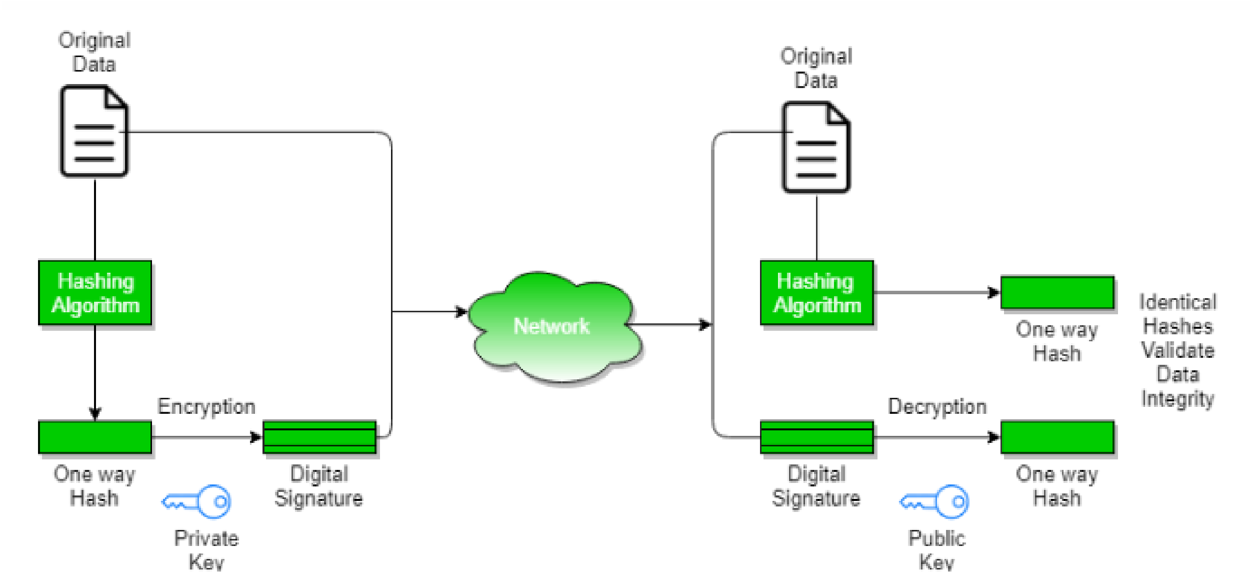
The steps followed in creating digital signature are :

1. Message digest is computed by applying hash function on the message and then message digest is encrypted using private key of sender to form the digital signature. (digital

signature = encryption (private key of sender, message digest) and message digest = message digest algorithm(message)).

2. Digital signature is then transmitted with the message.(message + digital signature is transmitted)
3. Receiver decrypts the digital signature using the public key of sender.(This assures authenticity, as only sender has his private key so only sender can encrypt using his private key which can thus be decrypted by sender's public key).
4. The receiver now has the message digest.
5. The receiver can compute the message digest from the message (actual message is sent with the digital signature).
6. The message digest computed by receiver and the message digest (got by decryption on digital signature) need to be same for ensuring integrity.

Message digest is computed using one-way hash function, i.e. a hash function in which computation of hash value of a message is easy but computation of the message from hash value of the message is very difficult.



Benefits of Digital Signatures-

- **Legal documents and contracts:** Digital signatures are legally binding. This makes them ideal for any legal document that requires a signature authenticated by one or more parties and guarantees that the record has not been altered.

- **Sales contracts:** Digital signing of contracts and sales contracts authenticates the identity of the seller and the buyer, and both parties can be sure that the signatures are legally binding and that the terms of the agreement have not been changed.

- **Financial Documents:** Finance departments digitally sign invoices so customers can trust that the payment request is from the right seller, not from a bad actor trying to trick the buyer into sending payments to a fraudulent account.

- **Health Data:** In the healthcare industry, privacy is paramount for both patient records and research data. Digital signatures ensure that this confidential information was not modified when it was transmitted between the consenting parties.

- Federal, state, and local government agencies have stricter policies and regulations than many private sector companies. From approving permits to stamping them on a timesheet, digital signatures can optimize productivity by ensuring the right person is involved with the proper approvals.

- **Shipping Documents:** Helps manufacturers avoid costly shipping errors by ensuring cargo manifests or bills of lading are always correct. However, physical papers are cumbersome, not always easily accessible during transport, and can be lost. By digitally signing shipping documents, the sender and recipient can quickly access a file, check that the signature is up to date, and ensure that no tampering has occurred.

<u>Drawbacks of Digital Signatures-</u>

- **Dependence on Key Management:** Digital signatures rely on the secure management of cryptographic keys. This means that the sender must keep their private key safe and secure from unauthorized access, while the recipient must verify the sender’s public key to ensure its authenticity. Any failure in key management can compromise the security of the digital signature.

- **Complexity:** Digital signatures require a complex process of key generation, signing, and verification. This can make them difficult to implement and use for non-technical users.

- **Compatibility:** Different digital signature algorithms and formats may not be compatible with each other, making it difficult to exchange signed messages across different systems and applications.

- **Legal Recognition:** Although digital signatures have legal recognition in many countries, their legal status may not be clear in all jurisdictions. This can limit their usefulness in legal or regulatory contexts.

- **Revocation:** In case of key compromise or other security issues, digital signatures must be revoked to prevent their misuse. However, the revocation process can be complex and may not be effective in all cases.

- **Cost:** Digital signatures may involve additional costs for key management, certificate issuance, and other related services, which can make them expensive for some users or organizations.

- **Limited Scope:** Digital signatures provide authentication and integrity protection for a message, but they do not provide confidentiality or protection against other types of attacks, such as denial-of-service attacks or malware.