Program : **B.E**

Subject Name: **Information Security**

Subject Code: **IT-8001**

Semester: **8th**

Unit II: Public-key Cryptography

The symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to the public key cryptosystems. The process of encryption and decryption is depicted in the below-given figure.

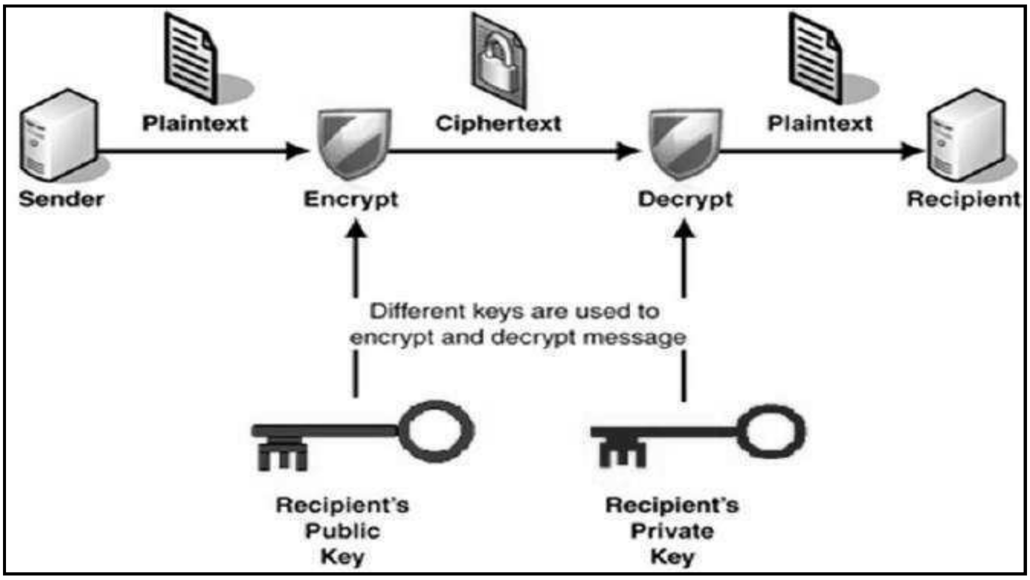

Fig: 2.1: public key encryption

The most important properties of public key encryption scheme are –

- Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
- Each receiver possesses a unique decryption key, generally referred to as his private key.
- The receiver needs to publish an encryption key, referred to as his public key.
- Some assurance of the authenticity of a public key is needed in this scheme to avoid spoofing by an adversary as the receiver. Generally, this type of cryptosystem involves a trusted third party which certifies that a public key belongs to a specific person or entity only.
- The encryption algorithm is sophisticated enough to prohibit an attacker from deducing the plaintext from the ciphertext and the encryption (public) key.
- Though private and public keys are related mathematically, it is not feasible to calculate the private key from the public key. The intelligent part of any public-key cryptosystem is in designing a relationship between two keys.

Modulo arithmetic

It turns out that prime numbers possess various useful properties when used in modular math. The RSA algorithm will take advantage of these properties.

- **Modular math** means that the only numbers under consideration are the non-negative integers less than the modulus. So, for mod *n*, only the integers from 0 to (*n* - 1) are valid operands and results of operations will always number from 0 to (*n* - 1).
- Think of a military time where the modulus is 2400. For instance, 2200 plus 400 (10:00 PM plus 4 hours) is not 2600. Once you reach 2400, you start over at 0. Hence, 2200 + 400 mod 2400 is 2600 - 2400 = 0200, or 2:00 in the morning. Likewise, if we start at 0, or midnight, 6 times 500 (say six 5-hour shifts) is not 3000, but 0600, or 6:00 AM the following day.
- Another aspect of modular math is the concept of a modular inverse. Two numbers are the modular inverse of each other if their product equals 1. For instance, 7 * 343 = 2401, but if our modulus is 2400, the result is

(7 * 343) mod 2400 = 2401 – 2400 = 1 mod 2400

Greatest common divisor

GCD of two numbers is the largest number that divides both. A simple way to find GCD is to factorize both numbers and multiply common factors.

36 = 2*2*3*3
60 = 2*2*3*5
GCD = Multiplication of common factor
= 2*2*3
= 12

Euclidean algorithm:

Euler's phi-function

In the eighteenth century, the mathematician Leonhard Euler (pronounced "Oiler") described □(*n*) as the number of numbers less than *n* that are relatively prime to *n*. The character is the Greek letter "phi" (in math circles it rhymes with "tea," in the academic organization Phi Beta Kappa it rhymes with "tie"). This is known as Euler's phi-function.

- Two numbers are **relatively prime** if they share only one factor, namely 1. For example, 10 and 21 are relatively prime. Neither is prime, but the numbers that evenly divide 10 are 1, 2, 5 and 10, whereas the numbers that evenly divide 21 are 1, 3, 7 and 21. The only number in both lists is 1, so the numbers are relatively prime.
- So j(6), for instance, is 2, since of all the numbers less than 6 (1, 2, 3, 4 and 5), only two of them (1 and 5) are relatively prime with 6. The numbers 2 and 4 shares with 6 a common factor other than 1, namely 2. And 3 and 6 shares 3 as a common factor.

Exponentiation

Exponentiation is taking numbers to powers, such as 2^3, which is 2 * 2 * 2 = 8. In this example, 2 is known as the **base** and 3 is the **exponent**. There are some useful algebraic identities in exponentiation. For instance,

$\mathbf{(b^x) * (b^y) = b^{x+y}}$

To illustrate this identity, let *b* = 2, *x* = 3 and *y* = 4.

$(2^3) * (2^4) = (2 * 2 * 2) * (2 * 2 * 2 * 2) = 2^7 = 2^{3+4}$

Another similar, useful identity is

$\mathbf{(b^x)^y = b^{xy}}$

Once again, to illustrate this identity, let *b* = 2, *x* = 3 and *y* = 4.

$(2^3)^4 = (2^3) * (2^3) * (2^3) * (2^3) = (2 * 2 * 2) * (2 * 2 * 2) * (2 * 2 * 2) * (2 * 2 * 2) = 2^{12} = 2^{3*4}$

Example

3 * 1 = 3 mod 10
3 * 2 = 6 mod 10
3 * 3 = 9 mod 10
3 * 4 = 12, 12 mod 10 = 12 - 10 = 2 mod 10
3 * 5 = 15, 15 mod 10 = 15 - 10 = 5 mod 10
3 * 6 = 18, 18 mod 10 = 18 - 10 = 8 mod 10
3 * 7 = 21, 21 mod 10 = 21 - (2 * 10) = 21 - 20 = 1 mod 10

Basic Euclidean Algorithm for GCD

The algorithm is based on below facts.

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

RSA Algorithm

The system was invented by three scholars **Ron Rivest, Adi Shamir,** and **Len Adleman** and hence, it is termed as RSA cryptosystem.

Modulo arithmetic - 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 robust and 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 co-prime.
- **Form the public key**
    - The pair of numbers (n, e) forms 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 mod (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 = p q = 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 × 5 = 145 = 1 mod 72
- Hence, public key is (91, 5) and private keys is (91, 29).

Encryption and Decryption

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 wishes 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 a 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 a public-key pair (n, e) has received a ciphertext C.
- Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.
  Plaintext = C^d mod n
- Returning to our numerical example, the ciphertext C = 82 would get decrypted to number 10 using private key 29 –
  Plaintext = 82^{29} mod 91 = 10

Hash functions

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length, but the output is always of fixed length.

Values returned by a hash function are called **message digest** or **hash values**. The following picture illustrated the hash function –

Design of Hashing Algorithms

Hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm. The size of each data block varies depending on the algorithm. Typically, the block sizes are from 128 bits to 512 bits.

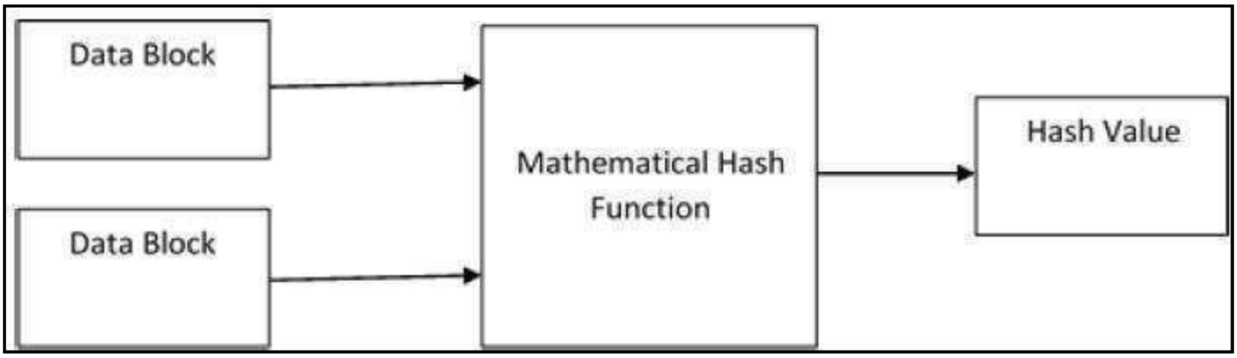

Fig: 2.2 Hashing

Popular Hash Functions

Message Digest (MD)

- The MD family comprises of hash functions MD2, MD4, MD5, and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.
- MD5 digests have been widely used in the software world to assure integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the data, so that a user can compare the checksum of the downloaded file to it.

Secure Hash Function (SHA)

The family of SHA comprises four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from the same family, there are structurally different.

- The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993.
- SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.
- In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful.
- SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending upon some bits in their hash value. Though SHA-2 is a strong hash function. Though significantly

different, its basic design still follows the design of SHA-1. Hence, NIST called for new competitive hash function designs.

- In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and excellent resistance for attacks.

RIPEMD

The RIPEMD is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by the open research community and generally known as a family of European hash functions.

- RIPEMD-160 is an improved version and the most widely used version in the family. The 256 and 320-bit versions reduce the chance of an accidental collision but do not have higher levels of security as compared to RIPEMD-128 and RIPEMD-160 respectively.

Whirlpool

It is a 512-bit hash function.

- It is derived from the modified version of the Advanced Encryption Standard (AES). One of the designers was Vincent Rijmen, a co-creator of the AES.
- Three versions of Whirlpool have been released; namely WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL.

Attack on collision resistance:

A collision attack is the ability to find two inputs that produce the same result, but that result is not known ahead of time. In a typical case (e.g., the attack on MD5) only a relatively small number of specific inputs are known to produce collisions.

A collision attack can be used in a relatively small number of specific scenarios (e.g., signed certificates)

- A hash function h is (t, ε) collision resistant if there exists no t-time probabilistic algorithm that outputs two messages x1 and x2 such that h (x1) = h (x2) with probability > ε
- A hash function h is (t, ε) weak collision resistant if there exists no t-time probabilistic algorithm A such that when given x, with probability > ε, it outputs x' such that x'≠ x and h(x') = h(x)

Diffie Hellman key exchange

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 received the key and from that generates a secret key after which they have the same secret key to encrypt.
  Step by Step Explanation

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.

Alice	BOB
Public Keys Available =P, G	Public Keys Available =P, G
Private key selected = a	Private Key Selected =b
Key generated =X=G^a mod P	Key generated =X=G^b mod P
Exchange of Generated key take place	
Key received = y	Key received = X
Generated Secret Key = K_a =y^a mod P	Generated Secret Key = K_b =y^b mod P

Table 1: Diffie Hellman key exchange

Note: Algebraically it can be shown that K_a=K_b

Users now have a symmetric secret key to encrypt.

Digital signature standard

- Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message. Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by the receiver as well as any third party.
- A digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer. In real world, the receiver of a message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of a digital signature scheme is depicted in the following fig 2.3

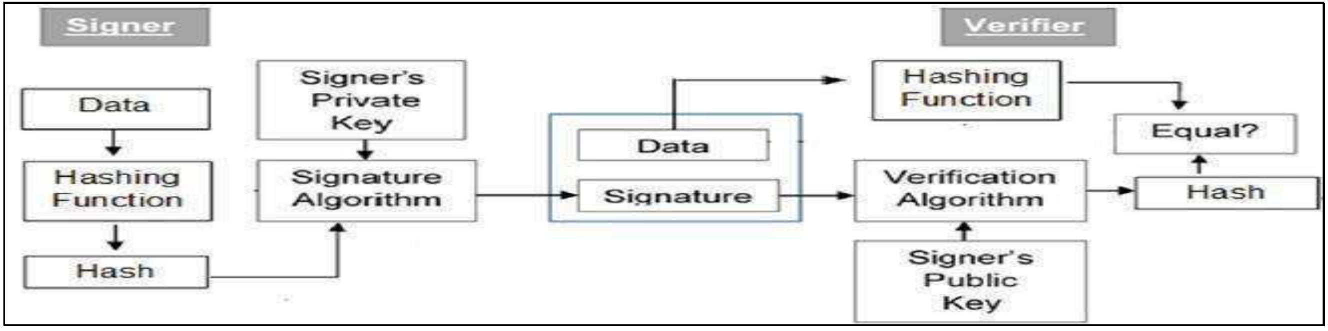

Fig: 2.3 Digital Signature

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 the 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 the same hash function on received data to generate a hash value.
- For verification, this hash value and output of the verification algorithm are compared. Based on the

comparison result, verifier decides whether the digital signature is valid.

- Since the digital signature is created by ‘private' key of the signer and no one else can have this key; the signer cannot repudiate signing the data in future.

Encryption with Digital Signature

In a public key encryption scheme, a public (encryption) key of the sender is available in the open domain, and hence anyone can spoof his identity and send an encrypted message to the receiver. This is depicted in the following figure 2.4.

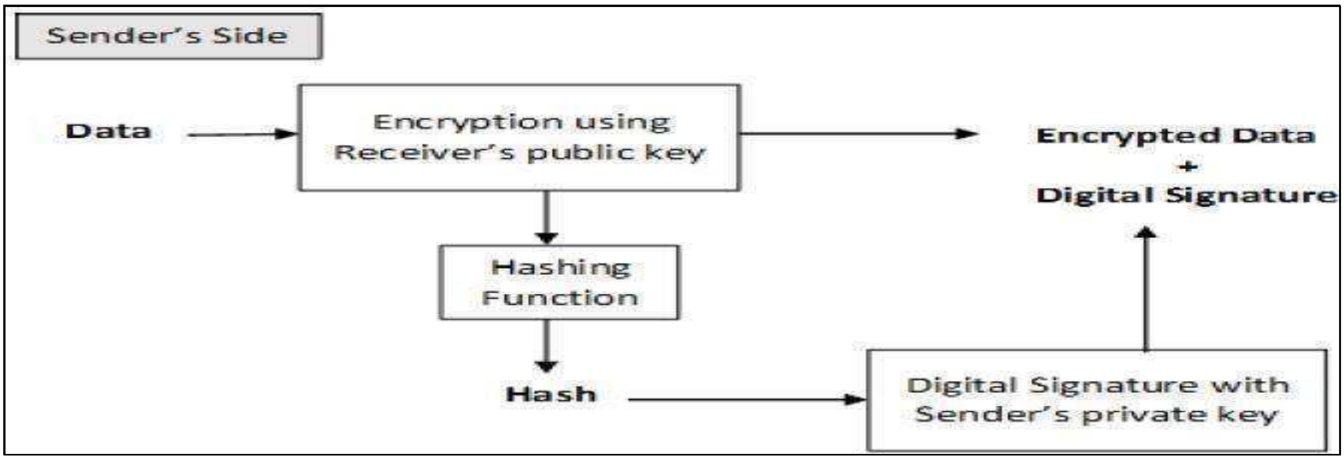
Sender’s Side
Data
Encryption using Receiver’s public key
Encrypted Data + Digital Signature
Hashing Function
Hash
Digital Signature with Sender’s private key

Fig: 2.4 Digital Signature

The receiver after receiving the encrypted data and signature on it first verifies the signature using sender's public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

Elliptic curve cryptography

Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography, based on the algebraic structure of elliptic curves over finite fields. ECC requires a smaller key as compared to non-ECC cryptography to provide equivalent security (a 256-bit ECC security have equal protection attained by 3072-bit RSA cryptography).

- Elliptic curve cryptography (ECC) is a modern type of public-key cryptography wherein the encryption key is made public, whereas the decryption key is kept private.
- This strategy uses the nature of elliptic curves to provide security for all manner of encrypted products.

Elliptic Curve Equation

$$y^2=x^3+ax+b \bmod p$$

- Here, y, x, a and b are all within Fp, i.e. they are integers modulo p.
- The coefficients a and b are the so-called characteristic coefficients of the curve -- they determine what points will be on the curve.
- curve coefficients have to fulfil one condition: $4a^3 + 27b^2$ not equal to 0

This condition guarantees that the curve will not contain any singularities.