**Unit III**

# 3 Public Key and Management

## Syllabus

*Public Key Cryptography, RSA Algorithm : Working, Key length, Security, Key Distribution, Deffie-Hellman Key Exchange, Elliptic Curve : Arithmetic, Cryptography, Security, Authentication methods, Message Digest, Kerberos, X.509 Authentication service. Digital Signatures : Implementation, Algorithms, Standards (DSS), Authentication Protocol.*

## Contents

## 3.1 Public Key Cryptography

- Diffie and Hellman proposed a new type of cryptography that distinguished between encryption and decryption keys. One of the keys would be publicly known; the other would be kept private by its owner.
- These algorithms have the following important characteristic.
  1. It must be computationally easy to encipher or decipher a message given the appropriate key.
  2. It must be computationally infeasible to derive the private key from the public key.
  3. It must be computationally infeasible to determine the private key from a chosen plaintext attack.
- *A public key encryption* scheme has six ingredients. *Fig. 3.1.1* shows public key cryptography. (See Fig. 3.1.1 on next page)

1. **Plaintext :** It is input to algorithm and in a readable message or data.
2. **Encryption algorithm :** It performs various transformations on the plaintext.
3. **Public and private keys :** One key is used for encryption and other is used for decryption.
4. **Ciphertext :** This is the scrambled message produced as output. It depends on the plaintext and the key.
5. **Decryption algorithm :** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

- The essential steps are the following :
  1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
  2. Each user places one of the two keys in a public register. This is the public key. The companion key is kept private.
  3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
  4. Alice decrypts the message using her private key.
- The public key is accessed to all participants and private key is generated locally by each participant.
- System controls its private key. At any time, a system can change its private key. Fig. 3.1.2 shows the process of public key algorithm.
- A message from source which is in a plaintext $X = (X_1, X_2, \ldots X_m)$. The message is intended for destination which generates a related pair of keys a public key $KU_b$, and a private key $KR_b$.
- Private key is secret key and known only to $Y_1$. With the message X and encryption key $KU_b$ as input, $Y_1$ forms the ciphertext.

$$Y = (Y_1, Y_2, Y_3 \ldots . Y_n)$$

$$Y = E_{KU_b}(X)$$

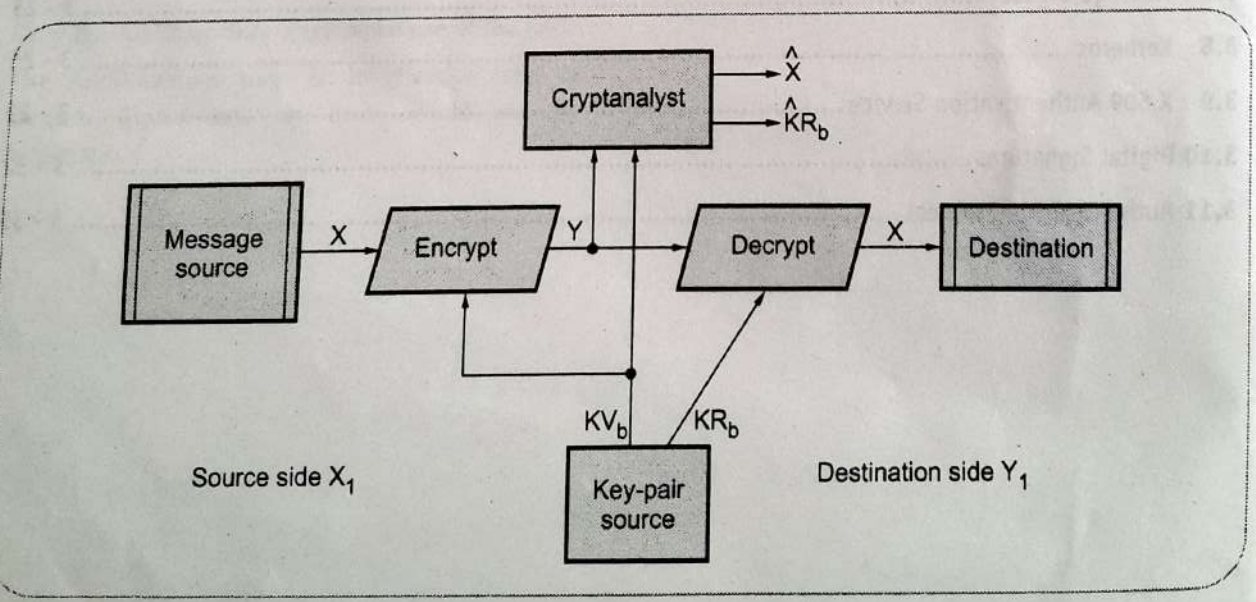


**Fig. 3.1.2 Public key cryptosystem secrecy**

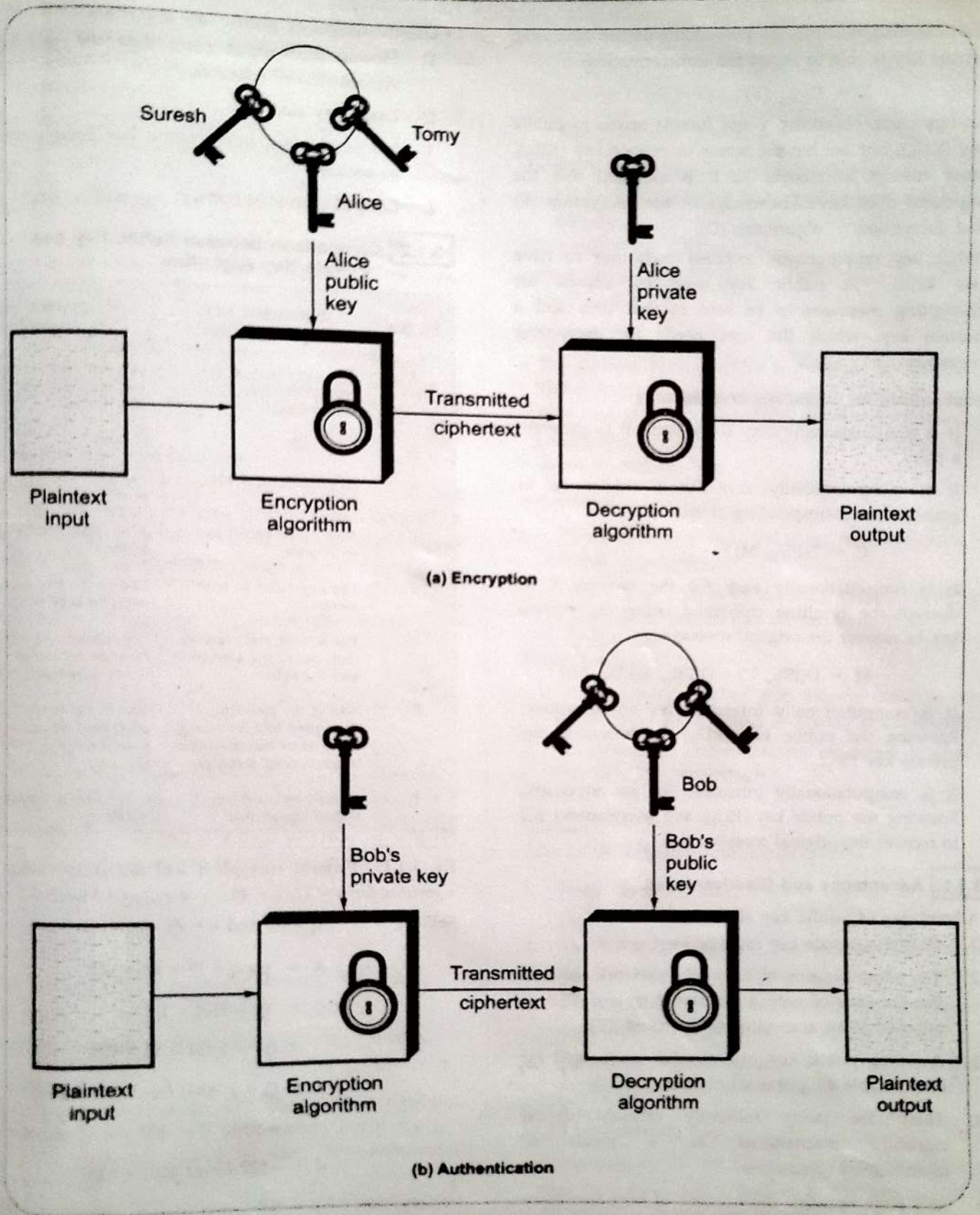


Fig. 3.1.1 Public key cryptography

- The intended receiver, in possession of the matching private key is able to invert the transformation.

$$X = D_{KR_b}(Y)$$

- An opponent, observing Y and having access to public key ($KU_b$), but not having access to private key ($KR_b$), must attempt to recover X. It is assumed that the opponent does have knowledge of the encryption (E) and decryption algorithms (D).
- Public key cryptography requires each user to have two keys : A public key used by anyone for encrypting messages to be sent to that user and a private key, which the user needs for decrypting messages.

**Requirements for public key cryptography**

1. It is computationally easy for a party B to generate a pair.
2. It is computationally easy for a sender A, to generate the corresponding ciphertext :

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message :

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key ($PU_b$) to determine the private key $PR_b$.
5. It is computationally infeasible for an adversary, knowing the public key ($PU_b$) and a ciphertext (C) to recover the original message (M).

## 3.1.1 Advantages and Disadvantages

- **Advantages of public key algorithm**
  1. Only the private key must be kept secret.
  2. The administration of keys on a network requires the presence of only a functional trusted TTP as opposed to an unconditionally trusted TTP.
  3. A private/public key pair remains unchanged for considerable long periods of time.
  4. There are many relatively efficient digital signature mechanisms as a result of asymmetric-key schemes.
  5. In a large network the number of keys necessary may be considerably smaller than in the symmetric-key scenario.

- **Disadvantages of public key algorithm**
  1. Slower throughput rates than the best known symmetric-key schemes.
  2. Large key size.
  3. No asymmetric-key scheme has been proven to be secure.
  4. Lack of extensive history.

## 3.1.2 Comparison between Public Key and Private Key Algorithm

| Sr. No. | Symmetric key cryptography | Asymmetric key cryptography |
|---|---|---|
| 1. | Same key is used for encryption and decryption. | One key for encryption and other key for decryption. |
| 2. | Very fast. | Slower. |
| 3. | Key exchange is big problem. | Key exchange is not a problem. |
| 4. | Also called **secret key** encryption. | Also called **public key** encryption. |
| 5. | The key must be kept secret. | One of the two keys must be kept secret. |
| 6. | The sender and receiver must share the algorithm and the key. | The sender and receiver must each have one of the matched pair of keys. |
| 7. | Size of the resulting encrypted text is usually same as or less than the original clear text size. | Size of the resulting encrypted text is more than the original clear text size. |
| 8. | Cannot be used for digital signatures. | Can be used for digital signature. |

**Ex. 3.1.1** *Perform encryption and decryption using RSA algorithm for p = 17, q = 11, e = 7 and M = 2.*

**Sol. :** P = 17 q = 31 and e = 7

$$n = p \times q = 17 \times 31 = 527$$

$$\phi(n) = (p - 1)(q - 1)$$

$$= (17 - 1)(31 - 1) = 480$$

$$d = (1 + k\,\phi(n)) / e = (1 + 480k) / 7$$

$$= -959 / 7 = -137 \quad (\text{for } k = -2)$$

$$d = -137 \pmod{480} = 343$$

$$\text{Encryption } (C) = M^e \pmod{n} = 2^7 \pmod{527} = 128$$

$$\text{Decryption } M = C^d \pmod{n} = 128^{343} \pmod{527} = 2$$

**Review Questions**

1. *Explain various public key distribution approaches.*
2. *What are different approaches of public key distribution ? Explain any one.*
3. *Compare between symmetric key encryption and asymmetric key encryption.*

## 3.2 RSA Algorithm

- RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and n – 1 for some n.
- A typical size for n is 1024 bits.
- The RSA algorithm developed in 1977 by Rivest, Shamir, Adleman (RSA) at MIT. RSA algorithm is public key encryption type algorithm. In this algorithm, one user uses a public key and other user uses a secret (private key) key.
- In the RSA algorithm each station independently and randomly chooses two large primes p and q number, and multiplies them to produce n = pq which is the modulus used in the arithmetic calculations of the algorithm.
- The details of the RSA algorithm are described as follows :
- **Key generation :**
  1) Pick two large prime numbers p and q, $p \neq q$;
  2) Calculate $n = p \times q$;
  3) Calculate $\phi(n) = (p - 1)(q - 1)$;
  4) Pick e, so that gcd $(e, \phi(n)) = 1$, $1 < e < \phi(n)$;
  5) Calculate d, so that $d \cdot e \bmod \phi(n) = 1$, i.e. d is the multiplicative inverse of e in mod $\phi(n)$;
  6) Get public key as $K_U = \{e, n\}$;
  7) Get private key as $K_R = \{d, n\}$.
- **Encryption :**

For plaintext block $P < n$, its ciphertext $C = P^e \bmod n$.

- **Decryption :**

For ciphertext block C, its plaintext is $P = C^d \bmod n$.

**Why RSA works :**

- As we have seen from the RSA design, RSA algorithm uses modular exponentiation operation. For $n = p \cdot q$, e which is relatively prime to $\phi(n)$, has exponential inverse in mod n.
- Its exponential inverse d can be calculated as the multiplicative inverse of e in mod $\phi(n)$. The reason is illustrated as follows :

Based on Euler's theorem, for y which satisfies $y \bmod \phi(n) = 1$, the following equation holds :

$$x^y \bmod n = x \bmod n$$

AS $d \cdot e \bmod \phi(n) = 1$, we have that $p^{ed} \equiv P \bmod n$. So the correctness of RSA cryptosystem is shown as follows :

- **Encryption :** $C = P^e \bmod n$;
- **Decryption :** $P = C^d \bmod n = (P^e)^d \bmod n = P^{ed} \bmod n = P \bmod n = P$.

**Why RSA is secure :**

- The premise behind RSA's security is the assumption that factoring a big number (n into p and q) is hard. And thus it is difficult to determine $\phi(n)$. Without the knowledge of $\phi(n)$, it would be hard to derive d based on the knowledge of e.

**Advantages**

1. RSA can be used both for encryption as well as for digital signatures.
2. Trapdoor in RSA is in knowing value of n but not knowing the primes that are factors of n.

**Disadvantages**

1. If any one of p, q, m, d is known, then the other values can be calculated. So secrecy is important.
2. To protect the encryption, the minimum number of bits in n should be 2048.

### 3.2.1 Attacks on RSA

Attacks on RSA algorithm are as follows :

1. **Brute force :** This involves trying all possible private keys.
2. **Mathematical attacks :** This involves the factoring the product of two primes.
3. **Timing attacks :** These depends on the running time of the description algorithm.
4. **Chosen ciphertext attacks :** This type of attack exploits properties of the RSA algorithm.

#### 3.2.1.1 Computing $\phi(n)$

- Computing $\phi(n)$ is no easier that factoring n. For, if n and $\phi(n)$ are known, and n is the product of two primes p, q, then n can be easily factored, by solving the two equations.

$$n = pq \quad \ldots (3.2.1)$$

$$\phi(n) = (p - 1)(q - 1) \quad \ldots (3.2.2)$$

for the two unknowns p and q.

- If we substitute $q = n/p$ into the equation (3.2.2), we obtain a quadratic equation in the unknown value p :

$$p^2 - (n - \phi(n) + 1)p + n = 0 \quad \ldots (3.2.3)$$

- The two roots of equation (3.2.3) will be p and q, the factors of n. If a cryptanalyst can learn the value of $\phi(n)$, then he can factor 'n' and break the system.

### 3.2.1.2 Timing Attacks

- Kocher described a new attack on RSA in 1995.
- If the attacker Eve knows Alice's hardware in sufficient detail and is able to measure the decryption times for several known cipher-texts, she can deduce the decryption key (d) quickly. This attack can also be applied against the RSA signature scheme.
- In 2003, Boneh and Brumley demonstrated a more practical attack capable of recovering RSA factorizations over a network connection. This attack takes advantage of information leaked by the Chinese remainder theorem optimization used by many RSA implementations.
- One way to thwart these attacks is to ensure that the decryption operation takes a constant amount of time for every cipher-text. However, this approach can significantly reduce performance.
- There is simple counter-measures against timing attacks :

1. **Constant exponentiation time :** Ensure that all exponentiations take the same time, but this will degrade performance.
2. **Random delay :** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.
3. **Blinding :** Multiply the cipher-text by a random number before performing exponentiation. This process prevents the attacker from knowing what cipher-text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack. RSA data security reports a 2 % to 10 % performance penalty for blinding.

### 3.2.1.3 Mathematical Attacks

- We can identify three approaches to attacking RSA mathematically :

1. Factor n into two prime factors, this enables calculation of $\phi(n) = (p - 1)(q - 1)$, which in turn, enables determination of $d = e^{-1} \bmod \phi(n)$.
2. Determine $\phi(n)$ directly, without first determining p and q.
3. Determine d directly, without first determining $\phi(n)$.

- Most discussions of cryptanalysis of RSA have focused on the task of factoring n into its two prime numbers. Determining $\phi(n)$ given n is equivalent to factoring n.
- With presently known algorithms, determining d given e and n appears to at least as time consuming as the factoring problem.

### 3.2.1.4 Adaptive Chosen Cipher-text Attacks

- In 1998, Daniel Bleichenbacher described the first practical adaptive chosen cipher-text attack, against RSA-encrypted messages using the PKCS#1 v1 padding scheme.
- Due to flaws with the PKCS#1 scheme, Bleichenbacher was able to mount a practical attack against RSA implementations of the Secure Socket Layer protocol and to recover session keys.
- As a result of this work, cryptographers now recommend the use of provably secure padding schemes such as Optimal Asymmetric Encryption padding and RSA laboratories has released new versions of PKCS#1 that are not vulnerable to these attacks.

---

**Ex. 3.2.1** *For the given values $p = 19$, $q = 23$ and $e = 3$ find n, $\phi(n)$ and d using RSA algorithm.*

**Sol. :**

$$n = p * q$$

$$n = 19 \times 23$$

$$n = 437$$

$$\phi(n) = (p - 1) * (q - 1)$$

$$\phi(n) = 18 \times 22$$

$$\phi(n) = 396$$

$$e.d. = 1 \bmod \phi(n)$$

$$3d = 1 \bmod 396$$

$$d = \frac{1}{3}$$

**Ex. 3.2.2** *Using the RSA algorithm, encrypt the following :*
*i) p = 3, q = 11, e = 7, M = 12*
*ii) p = 7, q = 11, e = 17, M = 25*
*iii) Find the corresponding ds for i) and ii) and decrypt the ciphertext.*

**Sol. : i)**

$$n = p * q$$

$$n = 3 * 11 = 33$$

$$\phi(n) = (p - 1)(q - 1)$$

$$\phi(n) = 2 * 10 = 20$$

$$e \cdot d = 1 \bmod \phi(n)$$

$$7 \cdot d = 1 \bmod 20$$

$$d = 3$$

Ciphertext (C) $= M^e \bmod n$

$$= 12^7 \bmod 33$$

$$C = 12$$

**ii)** $n = p * q = 7 * 11 = 77$

$$\phi(n) = (p - 1) * (q - 1) = 6 \times 10 = 60$$

$$e \cdot d = 1 \bmod \phi(n) \Rightarrow 17\ d = 1 \bmod 77$$

$$d = 3$$

Ciphertext (C) $= M^e \bmod n$

$$= 25^{17} \bmod 77 \Rightarrow 77 \Rightarrow c = 9$$

$$C = 12$$

**iii) Decryption :**

$$M = c^d \bmod n$$

In case (i) $M = 12^3 \bmod 33 = 12$

In case (ii) $M = 9^{57} \bmod 77 = 25$

**Ex. 3.2.3** *In RSA system the public key of a given user is e = 7 and n = 187*
*i) What is the private key of this user ?*
*ii) If the intercepted ciphertext is c = 11 and sent to a user whose public key is e = 7 and n = 187. What is the plaintext ?*
*iii) What are the possible approaches to defeating the RSA algorithm ?*

**Sol. : i)** $n = p * q$

$$n = 11 \times 17 \Rightarrow 187$$

$$\phi(n) = (p - 1)(q - 1)$$

$$= (17 - 1)(11 - 1) = 16 \times 10 = 160$$

$$e \cdot d = 1 \bmod \phi(n)$$

$$7\ d = 1 \bmod 160$$

$$7 \times 23 = 1 \bmod 160$$

Public key PU

$$(e, n) = 7, 187$$

Private key PR

$$(d, n) = 23, 187$$

**ii)** $c = 11, \quad e = 7, \quad n = 187$

Plaintext $p = c^d \bmod n$

$$= 11^{23} \bmod 187$$

$$= 79720245 \bmod 187$$

$\therefore$ Plaintext = 88

**Ex. 3.2.4** *Explain about the RSA algorithm with example as : p = 11, q = 5, e = 3 and PT = 9*

**Sol. :** $p = 11, \quad q = 5$

$$n = p * q = 11 \times 5 = 55$$

$$\phi(n) = (p-1) * (q-1) = 10 * 4$$

$$= 40$$

$$e = 3 \text{ and } m = 9$$

$$\gcd(\phi(n), e) = \gcd(40, 3) = 1$$

$$d \equiv e^{-1} (\bmod \phi(n))$$

$$d \times e^{-1} (\bmod \phi(n)) = 1$$

$$3d \bmod 40 = 1$$

$$d = 27$$

public key $pu = \{e, n\} = \{3, 55\}$

private key $pr = \{d, n\} = \{27, 55\}$

Encryption : $C = M^e \bmod n$

$$= 9^3 \bmod 55 = 14$$

decryption : $M = c^d \bmod n$

$$M = 14^{27} \bmod 55 = 9$$

**Ex. 3.2.5** *Perform encryption and decryption using RSA algorithm. p = 7, q = 11, e = 17 and M = 8.*

**Sol. : RSA algorithm :**

$$N = p \times q$$
$$= 7 \times 11$$
$$= 77$$

Calculate $\phi(n) = (p - 1)(q - 1)$
$$= (7 - 1)(11 - 1)$$
$$= 6 \times 10$$
$$= 60$$

So, $e = 17$

Determine d such that

$$ed = 1 \bmod \phi(n)$$
$$17d = 1 \bmod 60$$

According to GCD :

$$60 = 17 * 3 + 9$$
$$17 = 9 * 1 + 8$$
$$9 = 8 * 1 + 1$$
$$8 = 1 * 8 + 0$$

Therefore, we have :

$$1 = 9 - 8$$
$$= 9 - (17 - 9)$$
$$= 9 - (17 - (60 - 17 * 3))$$
$$= 60 - 17 * 3 - (17 - 60 + 17 * 3)$$
$$= 60 - 17 * 3 + 60 - 17 * 4$$
$$= 60 * 2 - 17 * 7$$

Hence, we get,

$$d = e^{-1} \bmod f(n)$$
$$= e^{-1} \bmod 60$$
$$= -7 \bmod 60$$
$$= (53 - 60) \bmod 60$$
$$= 53$$

So, the public key is (17, 77) and the private key is (53, 77)

**Encryption :**

Ciphertext (C) $= M^e \bmod N$
$$= (8)^{17} \bmod 77$$
$$C = 57$$

**Ex. 3.2.6** *In a public key cryptosystem using RSA, given N = 187 and the encryption key (E) as 17, find out the corresponding private key (D).*

**Sol. :** $N = 187$

$N = p \times q = 17 \times 11$

$N = 187$

So $p = 17, \quad q = 11$

$\phi(n) = (p - 1) \times (q - 1)$
$= (17 - 1) \times (11 - 1) = 160$

$ED = 1 \bmod \phi(n)$

$17\ d = 1 \bmod 160$

$d = 113$

**Ex. 3.2.7** *Let the given data be - Prime numbers p = 11, q = 19 and the plain text to be sent is 4. Assume public key e as 23. Using RSA algorithm determine the cipher text for the given plain text. Also perform the reverse process of finding the plain text. Also perform the reverse process of finding the plain text from the cipher text.*

**Sol. :** Given p = 11, q = 19, plain text (m) = 40,

Public key e = 23

$n = p \times q = 11 \times 19 = 209$

$\phi(n) = (p - 1) \times (q - 1)$
$= (11 - 1) \times (19 - 1) = 180$

**Encryption :**

$C = M^e \pmod n$
$= (40)^{23} \bmod 180$
$C = 160$

**Ex. 3.2.8 :** *For the given parameters 'P' = 3 and 'Q' = 19 find the value of 'e' and 'd' using RSA algorithm and encrypt message 'M' = 6.*

**Sol. :** P = 3 Q = 19

$N = PQ$

$= 3 \times 19$

$N = 57$

Calculate $\phi(n) = (P - 1)(Q - 1)$

$= (3 - 1)(19 - 1)$

$= 36$

Public key 'e' is calculated by using Euclid algorithm. Using 36, GCD is calculated and 5 and 7 gives GCD = 1

So you can select e = 5 or 7. Here we selected e = 7

So public key (7, 57)

Private key generation (d) :

Determine d such that $ed = 1 \bmod \phi(n)$

$7d = 1 \pmod{36}$

$7 \times 31 = 1 \pmod{36}$

So d = 31

Encryption of message :

Ciphertext (C) $= M^e \bmod n$

$= 6^7 \bmod 57$

$C = 9$

---

**Ex. 3.2.9 :** *Use RSA algorithm to encrypt the plaintext " 3 " use following parameters p = 11, q = 3, e = 13.*

**Sol. :** p = 11, q = 3, e = 13

Plaintext = 3

$N = p \times q = 11 \times 3 = 33$

$\phi(n) = (p-1)(q-1) = (11-1)(3-1) = 20$

Given e = 13;

Determine d such that

$ed = 1 \bmod \phi(n)$

$13d = 1 \bmod 20$

$13 \times 17 = 1 \bmod 20$

$221 = 1 \bmod 20$

So d = 17

Ciphertext $(C) = M^e \bmod (n)$

$C = 3^{13} \bmod 33$

$C = 27$

**Review Questions**

1. *For the given parameters 'P' = 3 and 'Q' = 19 find the value of 'e' and 'd' using RSA algorithm and encrypt message 'M' = 6.*
2. *What is public key cryptography ? Explain RSA algorithm used for public key cryptography.*
3. *Use RSA algorithm to encrypt the plaintext " 3 " use following parameters p = 11, q = 3, e = 13.*
4. *Explain RSA algorithm with suitable example.*
5. *Explain operation of RSA public key encryption algorithm.*

## 3.3 Key Distribution

- The purpose of public key cryptography is
  1. The distribution of public keys.
  2. The use of public key encryption to distribute secret keys.

### 3.3.1 Distribution of Public Keys

- Different methods have been proposed for the distribution of public keys. These are
  1. Public announcement.
  2. Publicly available directory.
  3. Public key authority.
  4. Public key certificates.

**1. Public announcement**

- In public key algorithm, any participant can **send** his or her public key to any other participant or **broadcast** the key to the community at large.
- Fig. 3.3.1 shows the public key distribution.
- Because of the growing popularity of PGP, which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newgroups and Internet mailing lists.

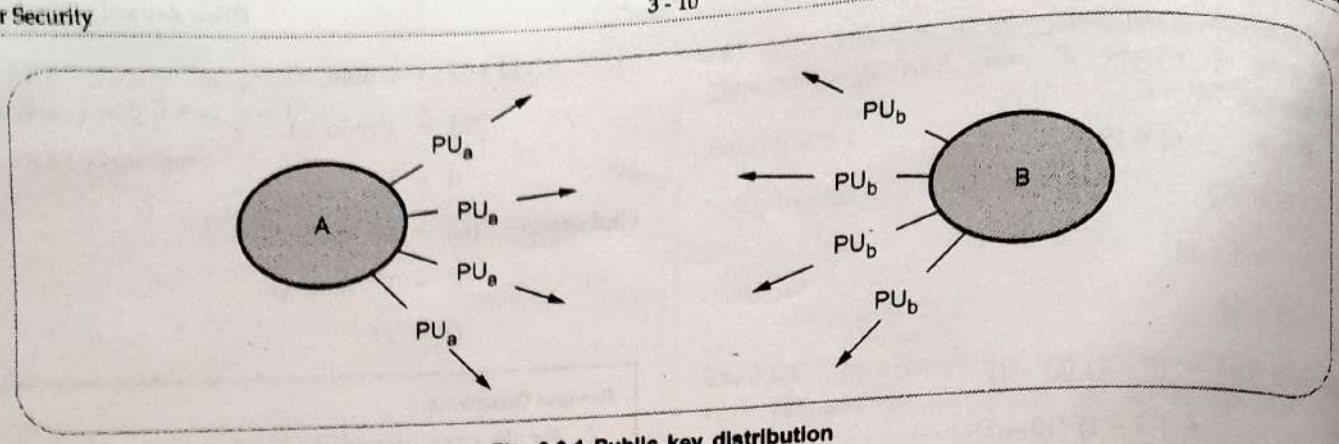


**Fig. 3.3.1 Public key distribution**

- The disadvantage is that, anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key.

**2. Public available directory**

- Greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.
- Fig. 3.3.2 shows public key publication.
- Such a scheme would include the following elements :
  1. The authority maintains a directory with a {name, public key} entry for each participant.
  2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
  3. A participant may replace the existing key with a new one at any time.
  4. Participants could also access the directory electronically.

**3. Public key authority**

- Fig. 3.3.3 shows public key distribution scenario. (See Fig. 3.3.3 on next page)
- Following steps occur in public key distribution.
  1. A sends a timestamped message to the public key authority containing a request for the current public key of B.
  2. The authority responds with a message that is encrypted using the authority's private key, $PR_{auth}$. The message also contains B's public key ($PU_b$), original request and timestamp.
  3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A ($ID_A$) and a nonce ($N_1$) which is used to identify this transaction uniquely.
  4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
  5. Public keys have been securely delivered to A and B and they may begin their protected exchange.

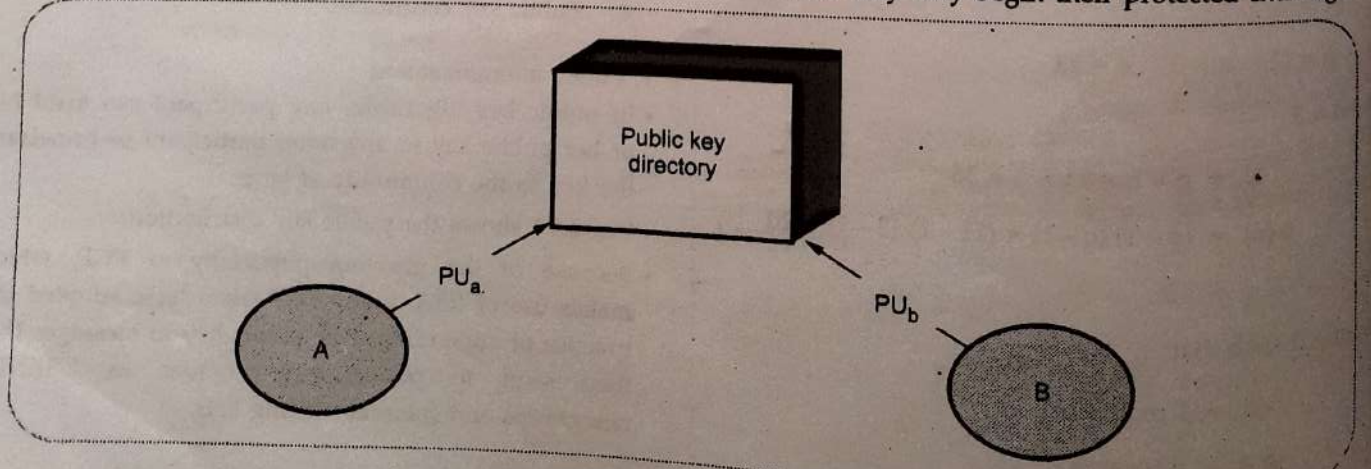


**Fig. 3.3.2 Public key publication**

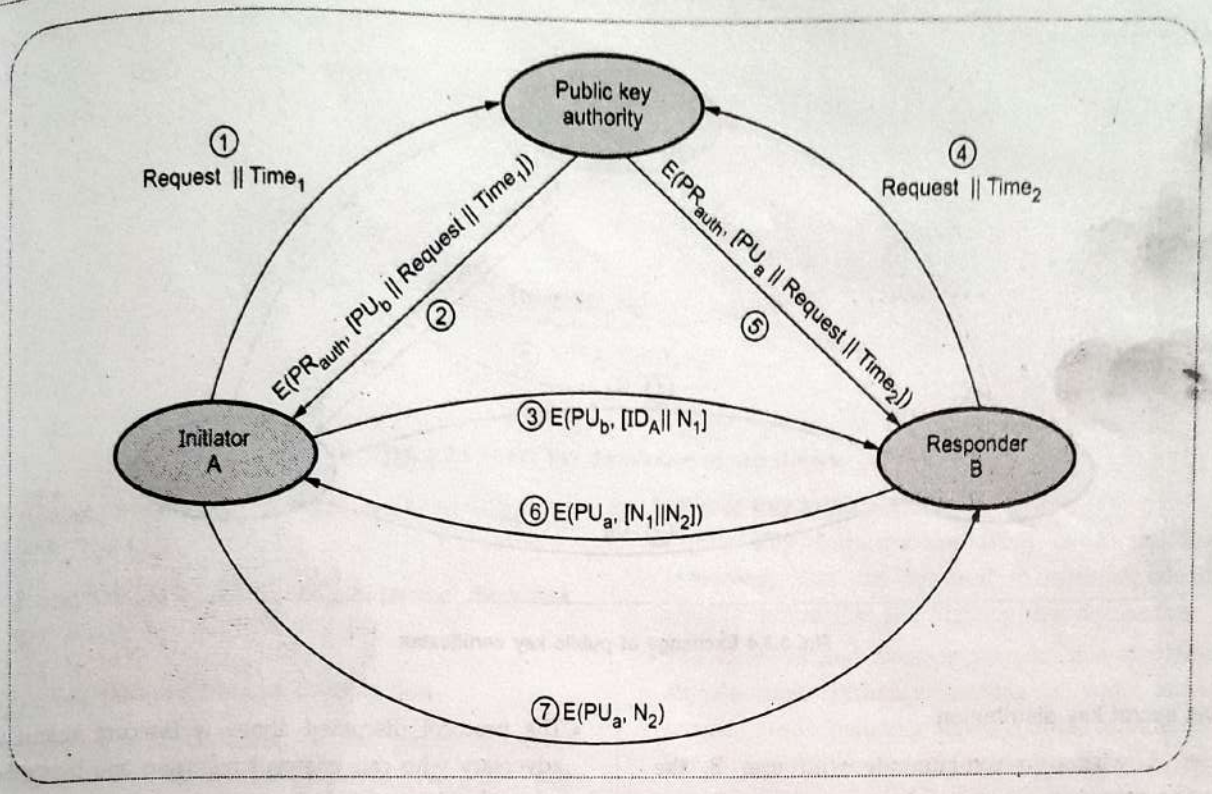


**Fig. 3.3.3 Public key distribution scenario**

6. B sends a message to A encrypted with $PU_a$ and containing A's nonce ($N_1$) as well as a new nonce generated by B($N_2$).
7. A returns $N_2$, encrypted using B's public key, to assure B that its correspondent is A.

**Drawback**

Public key authority could be somewhat of a bottleneck in the system. The directory of name and public keys maintained by the authority is vulnerable to tampering.

**4. Public key certificates**

- Certificates can be used by participants to exchange keys without contacting a public key authority. Certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party.
- The third party is a certificate authority, such as government agency or a financial institution, that is trusted by the user community.
- A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate.
- Requirements on this scheme :

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

- A certificate scheme is illustrated in Fig. 3.3.4. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
- For participant A, the authority provides a certificate of the form

$$C_A = E(PR_{auth}, [T \| ID_A \| PU_a])$$

where $PR_{auth}$ is the private key used by the authority and T is a timestamp.

## 3.3.2 Distribution of Secret Keys using Public Key Cryptography

- Public key encryption provides for the distribution of secret key to be used for conventional encryption.

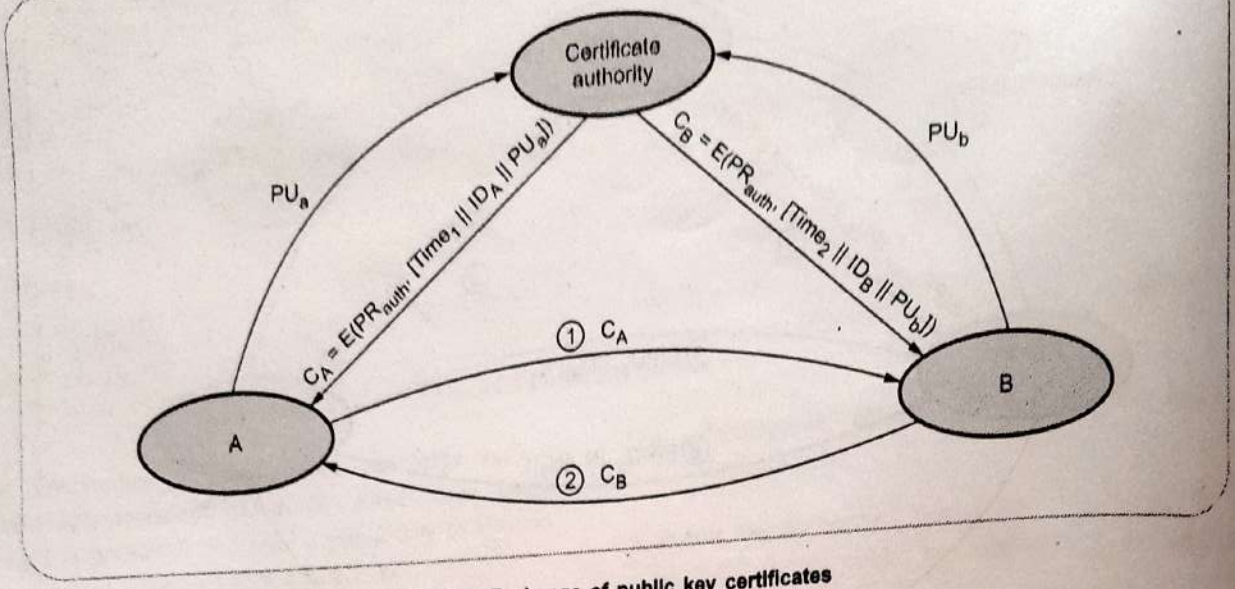


**Fig. 3.3.4 Exchange of public key certificates**

**Simple secret key distribution**

If user A wishes to communicate with user B, the following procedure is employed :

1. User A generates a public/private key pair {$PU_a$, $PR_a$} and transmits a message to user B consisting of $PU_a$ and an identifier of A, $ID_A$.
2. User B generates a secret key ($K_s$) and transmits it to user A, encrypted with A's public key.
3. User A computes D($PR_a$, E($PU_a$, $K_s$)) to recover the secret key. Because only A can decrypt the message, only user A and user B know the identity of $K_s$.
4. User A discards $PU_a$ and $PR_a$ and user B discards $PU_a$.
5. Fig. 3.3.5 shows use of public key encryption.

- User A and B can now securely communicate using conventional encryption and the session key $K_s$. At the completion of the exchange, both user A and B discard $K_s$.
- The protocol discussed above is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message. Such an attack is known as a **man in middle attack.**

**Secret key distribution with confidentiality and authentication**

- Fig. 3.3.6 shows the public key distribution of secret keys. (See Fig. 3.3.6 on next page.)
- It provides protection against both passive and active attacks.

1. A uses B's public key to encrypt a message to B containing an identifier of A ($ID_A$) and a nonce ($N_1$), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with $PU_a$ and containing A's nonce ($N_1$) as well as a new nonce generated by B($N_2$).
3. A returns $N_2$, encrypted using B's public key, to assure B that its correspondent is A.

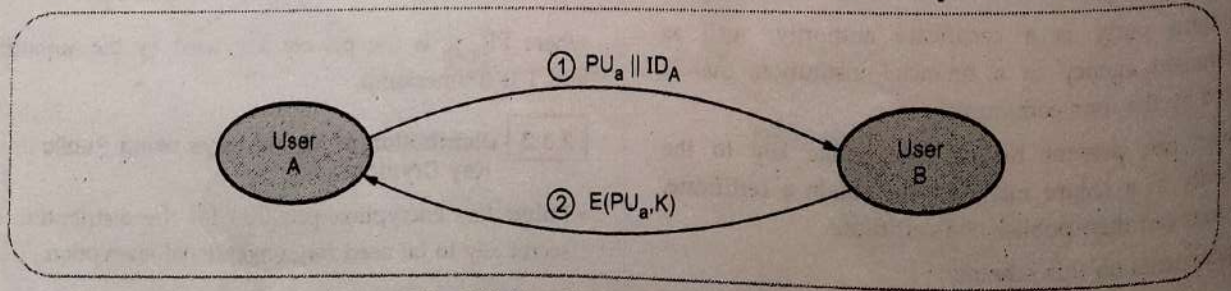


**Fig. 3.3.5 Use of public key encryption**

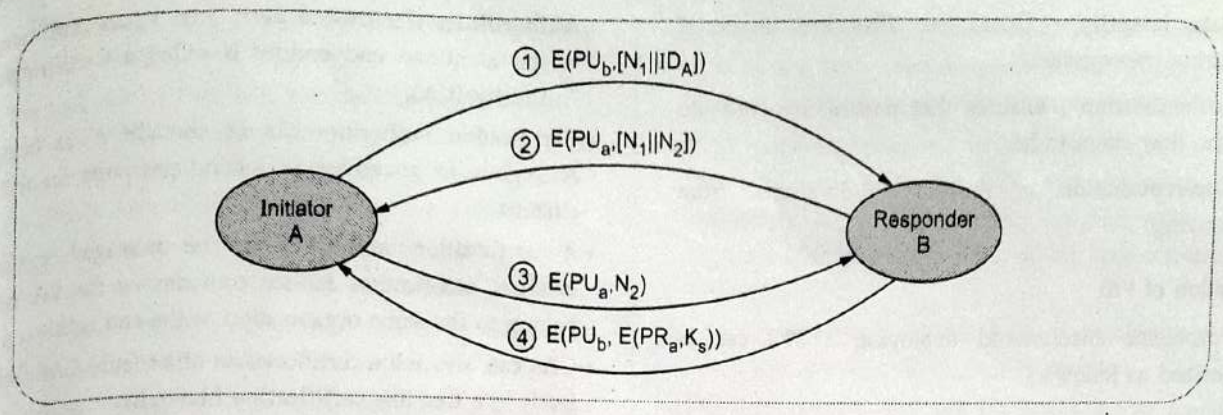


**Fig. 3.3.6 Public key distribution of secret keys**

4. A selects a secret key $K_s$ and sends $M = E(PU_b, E(PR_a, K_s))$ to B.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

## 3.3.3 Key Distribution and Certification

- Management and handling of the pieces of secret information is generally referred to as **key management.**
- Activities of key management includes selection, exchange, storage, certification, revocation, changing, expiration and transmission of the key.
- Key management is the set of processes and mechanisms which support key establishment and maintenance of ongoing keying relationship between parties, including replacing older key with new keys.
- Two major issues in key management are :

1. Key life time 2. Key exposure

Key life time - limit of use which can be measured as a duration of time.

**Issue related to key :**

1. Users must be able to obtain securely a key pair suited to their efficiency and security needs.
2. Keys need to be valid only until a specified expiration date.
3. The expiration date must be chosen properly and publicized securely.
4. User must be able to store their private keys securely.
5. Certificates must be unforgettable, obtainable in a secure manner.

**1. Public Key Infrastructure**

- Public Key Infrastructure (PKI) is a well-known technology that can be used to establish identities, encrypt information and digitally sign documents.
- PKI identifies and manages relationships of parties in an electronic exchange, serving a wide array of security needs including access control, confidentiality, integrity, authentication and non-repudiation.
- PKI also uses unique Digital Certificates (DC) to secure eCommerce, email, data exchange and Virtual Private Networks (VPN) and intranets and is also used to verify the identity and privileges of each user.
- The Certificate Authority (CA) provides a full life-cycle management of public keys and certificates, including issuance, authentication, storage, retrieval, backup, recovery, updating and revocation to the PKI.
- All users of PKI must have a registered identity, which is stored in a digital certificate issued by a CA.
- Remote users and sites using public private keys and public key certificates can authenticate each other with a high degree of confidence.
- Authentication is dependent on three conditions :
  1. It must be established that each party have a private key that has not been stolen or copied from the owner.
  2. The certificate must be issued to the owner in accord with the stated policy of the certificate issuer.
  3. The policies of the certificate issuer must be satisfactory to the parties so as to verify identity.

**Benefits of PKI**

1. **Confidential communication** : Only intended recipients can read files.

2. **Data integrity** : Guarantees files are unaltered during transmission.
3. **Authentication** : Ensures that parties involved are who they claim to be.
4. **Non-repudiation** : Prevents individuals from denying.

**Limitation of PKI**

The problems encountered deploying a PKI can be categorized as follows :

1. Public key infrastructure is new
2. Lack of standards
3. Shortage of trained personnel
4. Public key infrastructure is mostly about policies.

**2. Certificate**

- **Certificates** are digital documents that are used for secure authentication of communicating parties.
- A certificate binds identity information about an entity to the entity's public key for a certain validity period.
- A certificate is digitally signed by a Trusted Third Party (TTP) who has verified that the key pair actually belongs to the entity.
- Certificates can be thought of as analogous to passports that guarantee the identity of their bearers.
- **Authorities** : The trusted party who issues certificates to the identified end entities is called a **Certification Authority (CA).**
- Certification authorities can be thought of as being analogous to governments issuing passports for their citizens.
- A certification authority can be managed by an external certification service provider or the CA can belong to the same organization as the end entities.
- CAs can also issue certificates to other (sub) CAs. This leads to a tree-like **certification hierarchy.**
- The highest trusted CA in the tree is called a root CA.
- In large organizations, it may be appropriate to delegate the responsibility for issuing certificates to several different certificate authorities.
- For example, the number of certificates required may be too large for a single CA to maintain; different organizational units may have different policy requirements; or it may be important for a CA to be physically located in the same geographic area as the people to whom it is issuing certificates.
- The X.509 standard includes a model for setting up a hierarchy of the certification authority.
- Fig. 3.3.7 shows the hierarchy of certificate authorities.
- In the Fig. 3.3.7, the root CA is at the top of the hierarchy. The root CA's certificate is a self-signed

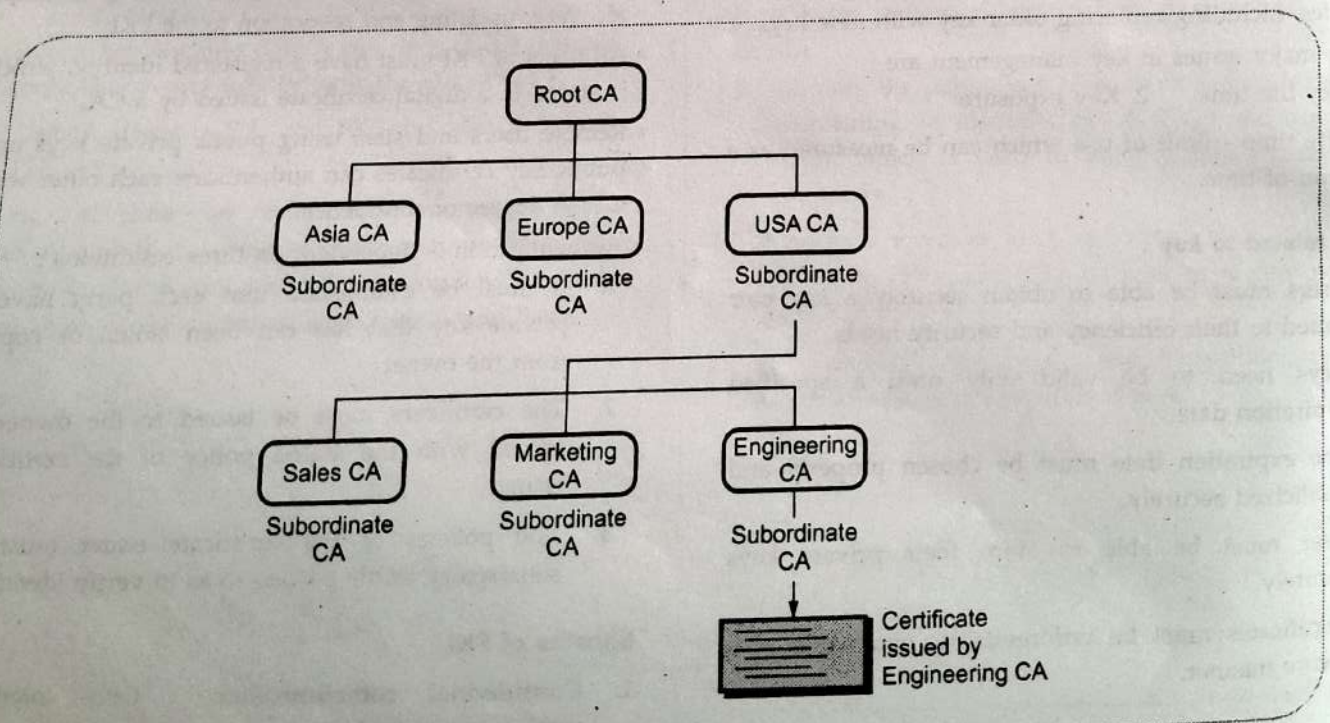


**Fig. 3.3.7 Hierarchy of CA**

certificate : That is, the certificate is digitally signed by the same entity.

- The CAs, that are directly subordinate to the root CA, have CA certificates signed by the root CA. CAs under the subordinate CAs in the hierarchy have their CA certificates signed by the higher-level subordinate CAs.
- Organizations have a great deal of flexibility in terms of the way they set up their CA hierarchies.
- **Certificate chains** : Certificate chain is series of certificates issued by successive CAs.
- In some cases, a CA can delegate the actual identification of end entities as well as some other administrative tasks to a separate entity, the **Registration Authority (RA).**

**Verifying certificates**

- When authentication is required, the entity presents a signatures it has generated from authentication data using its private key, and a certificate corresponding to that key.
- The receiving entity can verify the signature with the public key of the sender contained in the certificate.
- Next the receiving entity must verify the certificate itself by checking the validity time of the certificate and the signature of the CA in the certificate.
- If the CA is a sub CA, the receiving entity must also verify the signatures of all higher-level CAs up to the root CA.
- The list of certificates needed for verification is called a **certification path.**
- If all signatures are valid and the receiving entity trusts the root CA, the first entity will be authenticated successfully.
- If a private key of an end entity is compromised or the right to authenticate with a certificate is lost before its natural expiration date, the CA must revoke the certificate and inform all PKI users about this.
- The CA will periodically publish a **Certificate Revocation List (CRL).**
- The CRL is a list identifying the revoked certificates and it is signed by the CA.
- The end entities should check the latest CRL whenever they are verifying a validity of a certificate.

**3. Key length and encryption strength**

- The strength of encryption depends on both the cipher used and the length of the key.
- Encryption strength is often described in terms of the size of the keys used to perform the encryption : In general, longer keys provide stronger encryption.
- Key length is measured in bits. For example, 128-bit keys for use with the RC4 symmetric-key cipher supported by SSL provide significantly better cryptographic protection than 40-bit keys for use with the same cipher.
- Roughly speaking, 128-bit RC4 encryption is $3 \times 10^{26}$ times stronger than 40-bit RC4 encryption.
- Different ciphers may require different key lengths to achieve the same level of encryption strength.
- The RSA cipher used for public-key encryption, for example, can use only a subset of all possible values for a key of a given length, due to the nature of the mathematical problem on which it is based.
- Other ciphers, such as those used for symmetric key encryption, can use all possible values for a key of a given length, rather than a subset of those values.
- Thus a 128-bit key for use with a symmetric key encryption cipher would provide stronger encryption than a 128-bit key for use with the RSA public-key encryption cipher.

### 3.3.4 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. **Key distribution** 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 a number of ways, as follows.
  1. User A can select a key and physically deliver it to user B.
  2. A third party can select the key and physically deliver it to user A and user B.
  3. If user A and user 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 user A and user B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to user A and user B.
- For manual delivery of key, **options 1 and 2** are used. These options are suitable for **link encryption.**

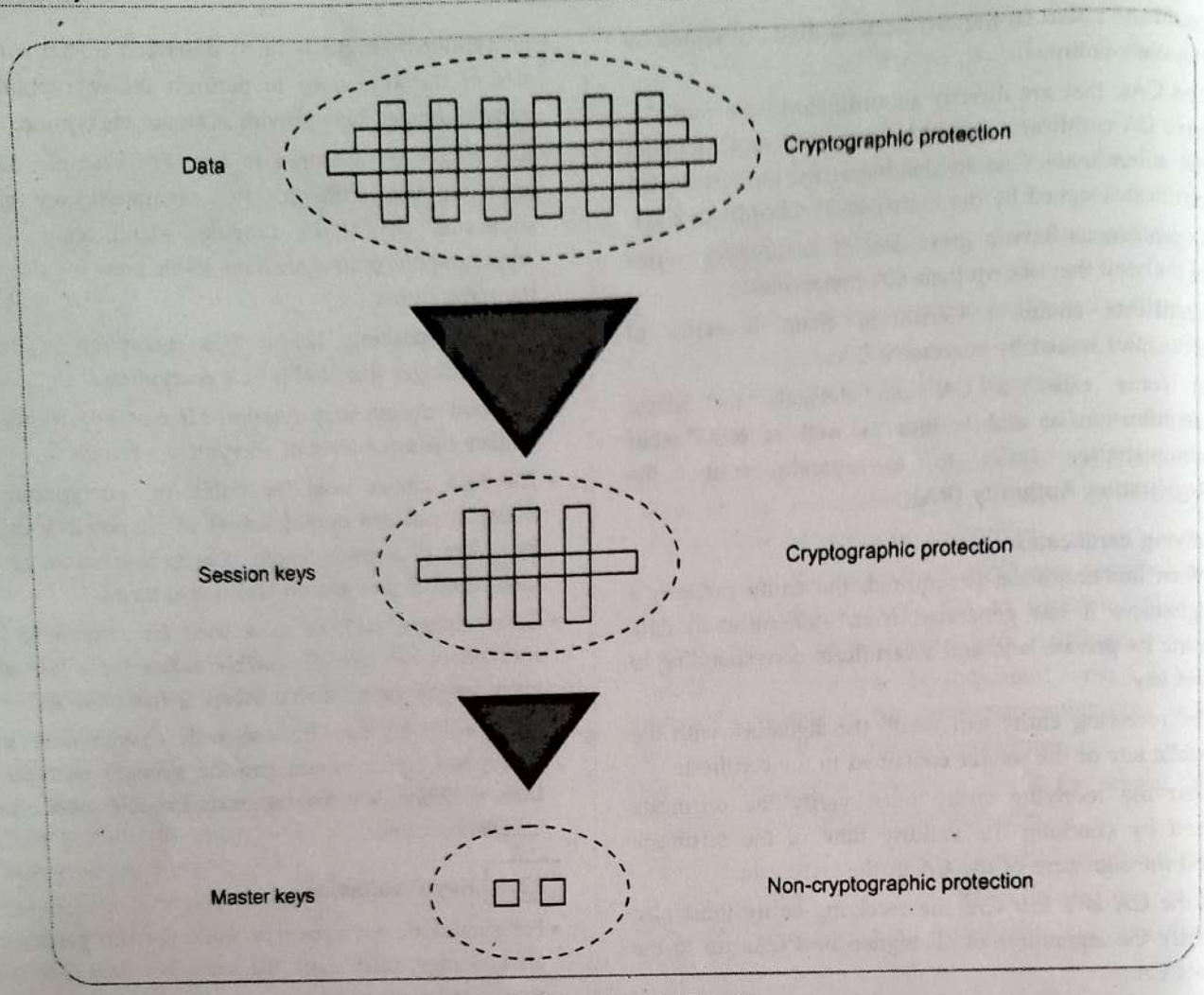


**Fig. 3.3.8 Use of a key hierarchy**

- Option 3 is suitable for link encryption or end-to-end encryption.
- For end-to-end encryption, some variation on option 4 has been widely adopted.
- The use of a key distribution center is based on the use of a hierarchy of keys. Minimum two levels of keys are used. Fig. 3.3.8 shows the use of a key hierarchy.
- Communication between end systems is encrypted using a temporary key, often referred to as a **session key**. The **session key** is used for the duration of a logical connection, such as a frame relay connection, or transport connection and then discarded.
- 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. For each end user, there is a unique master key that it shares with the key distribution center.

**A key distribution scenario**

- User A wishes to establish a logical connection with user B and requires a one time session key to protect the data transmitted over the connection. User A has a master key ($K_a$), known only to itself and the KDC. User B shares the master key $K_b$ 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 ($N_1$) for this transaction.
2. KDC responds with a message encrypted using $K_a$.
3. A stores the session key for use in the upcoming session and forward to B the information that originated at the KDC for B :
4. User B sends a nonce $N_2$ to A.

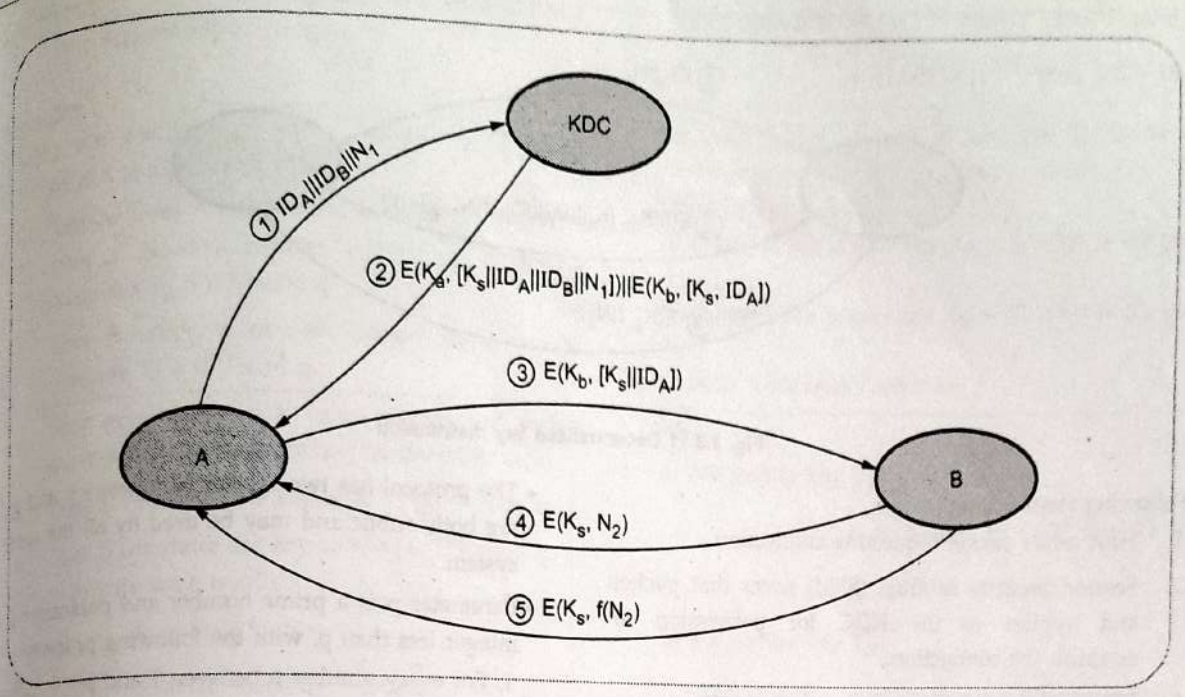


**Fig. 3.3.9 Key distribution scenario**

- Fig. 3.3.9 shows the key distribution scenario.
- Steps 1, 2 are used for key distribution and steps 3, 4, 5 for authentication.

**Session key lifetime**

1. **For connection-oriented protocol**
   - Use the same session key for the length of time that the connection is open. Use new session key for each new session.
   - For long lifetime, change the session key periodically.
2. **For connectionless protocol**
   - The most secure approach is to use a new session key for each exchange. For connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination.

**Transparent key control scheme**

- Fig. 3.3.10 shows automatic key distribution for connection - oriented protocol.
- Assume that communication make use of a connection-oriented end-to-end protocol, such as TCP.

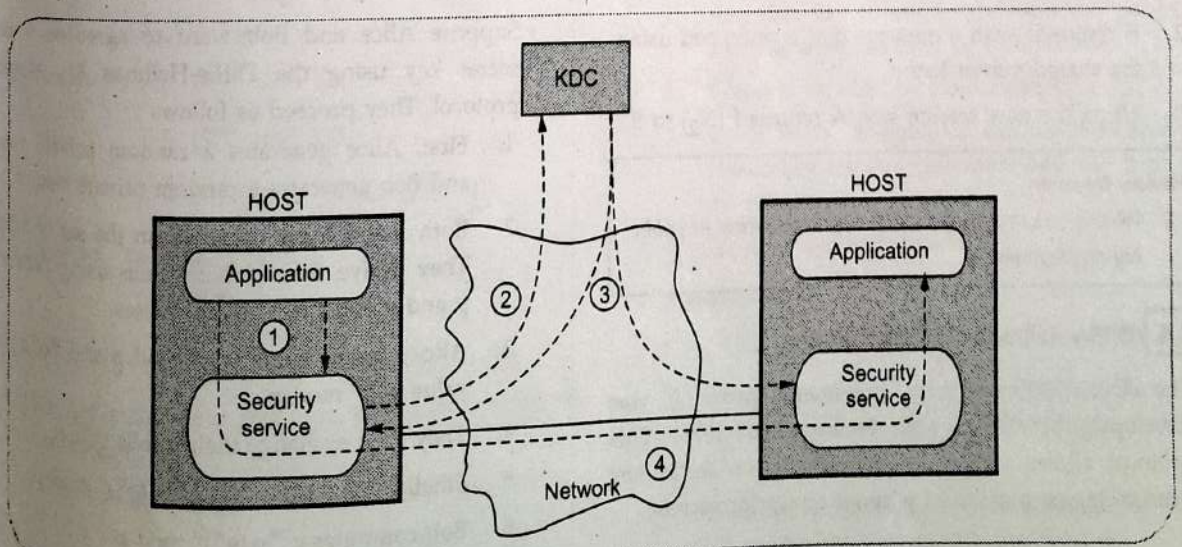


**Fig. 3.3.10 Automatic key distribution for connection-oriented protocol**

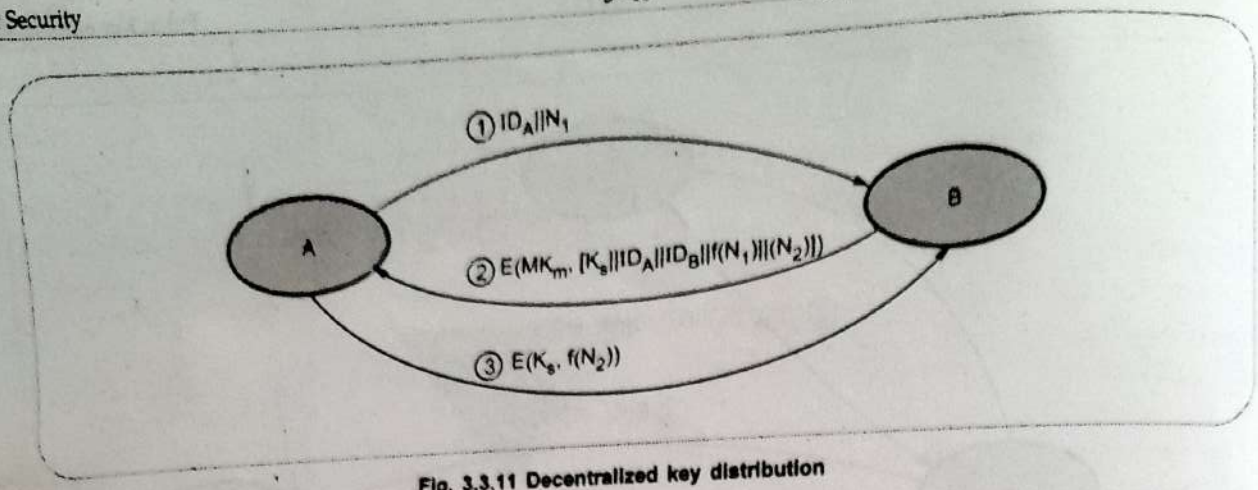


**Fig. 3.3.11 Decentralized key distribution**

- Following steps occurs :
  1. Host sends packet requesting connection.
  2. Session Security Module (SSM) saves that packet and applies to the KDC for permission to establish the connection.
  3. KDC distributes session key to both hosts.
  4. The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems.

**Decentralized key control**

- Decentralized approach requires that each end system be able to communicate in a secure manner with all potential parter end systems for purposes of session key distribution.
- A session key may be established with the following sequence of steps.
  1. A issues a request to B for a session key and includes a none, $N_1$.
  2. B responds with a message that is encryped using the shared master key.
  3. Using the new session key, A returns f ($N_2$) to B.

**Review Question**

1. *What are the methods used in key distribution in public key cryptography.*

## 3.4 Diffie-Hellman Key Exchange

- The Diffie-Hellman key agreement protocol was developed by Diffie and Hellman in 1976. This protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.
- The protocol has two system parameters p and g. Th are both public and may be used by all the users in system.
- Parameter p is a prime number and parameter g is integer less than p, with the following property :
  1. For every number n between 1 and p – 1 inclusive
  2. There is a power k of g such that $n = g^k \bmod p$.
- The protocol depends on the discrete logarith problem for its security. It assumes that it computationally infeasible to calculate the share secret key $k = g^{ab} \bmod p$ given the two public value $g^a \bmod p$ and $g^b \bmod p$ when the prime p sufficiently large.
- The Diffie-Hellman key exchange is vulnerable to man-in-the-middle attack. This vulnerability is presen because Diffie-Hellman key exchange does no authenticate the participants. Possible solutions includ the use of digital signatures and other protoc variants.
- Suppose Alice and Bob want to agree on a share secret key using the Diffie-Hellman key agreemen protocol. They proceed as follows :
  1. First, Alice generates a random private value a and Bob generates a random private value b.
  2. Both a and b are drawn from the set of integers They derive their public values using parameters p and g and their private values.
  3. Alice's public value is $g^a \bmod p$ and Bob's publ value is $g^b \bmod p$.
  4. They then exchange their public values.
  5. Finally, Alice computes $g^{ab} = (g^b)^a \bmod p$.
  6. Bob computes $g^{ba} = (g^a)^b \bmod p$.

7. Since $g^{ab} = g^{ba} = k$, Alice and Bob now have a shared secret key k.

**Algorithm :**

- Select two numbers (1) prime number q (2) $\alpha$ an integer that is a primitive root of q.
- Suppose the users A and B wish to exchange a key.
  1. User A select a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$.
  2. User B selects a random integer $X_B < q$ and compute $Y_B = \alpha^{X_B} \bmod q$.
  3. Both side keeps the X value private and makes the Y value available publicly to the other side.
  4. User A computes the key as $K = (Y_B)^{X_B} \bmod q$.
  5. User B computes the key as $K = (Y_A)^{X_B} \bmod q$.
- Both side gets same results :

$$K = (Y_B)^{X_A} \bmod q$$

$$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$$

$$= (\alpha^{X_B})^{X_A} \bmod q = \alpha^{X_B X_A} \bmod q$$

$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$$

$$= (Y_A)^{X_B} \bmod q$$

**Example**

- Key exchange is based on the use of the prime number and a primitive root of prime number.
- Prime number $q = 353$

  Primitive root $\alpha = 3$
- A and B select secret keys.

  $X_A = 97$ $X_B = 233$
- Calculates the public keys

A computes $Y_A = \alpha^{X_A} \bmod q$

$= (3)^{97} \bmod 353$

$= (1.9080 \times 10^{97}) \bmod 353 = 40$

B computes $Y_B = \alpha^{X_B} \bmod q$

$= (3)^{233} \bmod 353$

$= (1.4765 \times 10^{111}) \bmod 353 = 248$

- After they exchange public keys, each can compute the common secret key.

A computes $K = (Y_B)^{X_A} \bmod q = (248)^{97} \bmod 353$

$= (1.8273 \times 10^{232}) \bmod 353 = 160$

B computes $K = (Y_A)^{X_B} \bmod q = (40)^{233} \bmod 353$

$= (1.9053 \times 10^{373}) \bmod 353 = 160$

---

**Ex. 3.4.1** *User A and B use the Diffie-Hellman key exchange technique with a common prime q = 71 and a primitive root $\alpha = 7$.*

*a) If user A has private key $X_A = 5$, what is A's public key $Y_A$ ?*

*b) If user B has private key $X_B = 12$, what is B's public key $Y_B$ ?*

*c) What is the shared secret key ?*

**Sol. :**

**a) A's public key $Y_A$**

$$Y_A = \alpha^{X_A} \bmod q = (7)^5 \bmod 71$$

$$= 16807 \bmod 71 = 51$$

**b) B's public key $Y_B$**

$$Y_B = \alpha^{X_B} \bmod q = (7)^{12} \bmod 71$$

$$= 13841287201 \bmod 71 = 4$$

**c) Shared secret key**

i) At user A $K = (Y_B)^{X_A} \bmod q$

$= (4)^5 \bmod 71 = 1024 \bmod 71$

$K = 30$

The man in middle attack can work against the Diffie-Hellman key exchange algorithm, causing it to fail.

**Advantages**

1. Any user can choose a random x and publish $g^x$ in a public database such as a phone book.
2. Phone book must be maintained by a TTP.
3. Other users can look up the database and get the public key for the individual and use it to encrypt the message.
4. Ideal for use with emails.

**Disadvantages**

1. Does not protect against man-in-the-middle attacks.
2. Even can intercept all traffic between Alice and Bob and generate separate keys for communication with them.
3. If Allice sends an encrypted message for Bob with his public key, Eve simply forwards it.

4. For large prime p, (p – 1) is an even number and so $Z_p^*$ will have an subgroup of order 2.

**Ex. 3.4.2** *If generator g = 2 and n or P = 11, using Diffie-Hellman algorithm solve the following :*
*i) Show that 2 is a primitive root of 11.*
*ii) If A has a public key '9' what is A's private key ?*
*iii) If B has a public key '3' what is B's private key ?*
*iv) Calculate the shared secret key.*

**Sol. :** i)

$2^1 \bmod 11 = 2$

$2^2 \bmod 11 = 4$

$2^3 \bmod 11 = 8$

$2^4 \bmod 11 = 5$

$2^5 \bmod 11 = 10$

$2^6 \bmod 11 = 9$

$2^7 \bmod 11 = 7$

$2^8 \bmod 11 = 3$

$2^9 \bmod 11 = 6$

Using 2 as integer, we get all the integer values between 1 to 11. So 2 is a primitive root of 11.

ii) Public key = 9

$2^6 \bmod 11 = 9$

$X_A = 6$

iii) $Y_B = (11)^6 \bmod 9$

$Y_B = 1$

**iv) Shared secret key :**

$K = (Y_B)^{XA} \bmod q$

$K = 3^6 \bmod 11$

$K = 3$

**Ex. 3.4.3 :** *Consider a Diffie-Hellman scheme with a common prime q = 11 and a primitive root α = 2.*
*i) If user A has the public key $Y_A$ = 9; what is A's private key $X_A$ ?*
*ii) If user A has a public key $Y_A$ = 3 ; what is the shared secret key $X_A$ ?*

**Sol. :** i) q = 11, α = 2, $Y_A$ = 9, $X_A$ = ?

$2 \bmod 11 = 2$

$2^2 \bmod 11 = 4$

$2^3 \bmod 11 = 8$

$2^4 \bmod 11 = 5$

$2^5 \bmod 11 = 10$

$2^6 \bmod 11 = 9$

$2^7 \bmod 11 = 7$

$2^8 \bmod 11 = 3$

Since $2^e \bmod 11$ for $0 < i < 11$ contains all numbers from 1 to 11 – 1, the size of this set is equal to $\phi(q)$, the order of q.

From the above values $2^6 \bmod 11 = 9$ therefore $X_A = 6$

ii) From the above values

$\alpha^{X_A} \bmod 11 = Y_A$

$2^{X_A} \bmod 11 = 3$

$2^{X_8} \bmod 11 = 3$

$\therefore \quad X_A = 8$

**Review Question**

1. *Explain "Diffie-Hellman" key exchange algorithm with suitable example.*

## 3.5 Elliptic Curve

- An elliptic curve is a set of points on the cooridnate plane satisfying an equation of the form $y^2 + axy + by = x^3 + cx^2 + dx + e$. In order to use elliptic curves for say, Diffie-Hellman, there needs to be some mathematical operation on two points in the set that will always produce a point also in the set.
- ECC can be done with at least two types of arithmetic, each of which gives different definitions of multiplication. The two types of arithmetic are
  1. Zp arithmetic
  2. GF ($2^n$) arithmetic, which can be done with shifts and ⊕ s.
- To form a cryptographic system using elliptic curves, we need to find a hard problem corresponding to factoring the product of two primes or taking the discrete logarithm.

- Consider the equation $Q = KP$ where $Q, P \in E_P(a, b)$ and $K < P$. It is relatively easy to calculate Q given K and P, but it is relatively hard to determine K given Q and P. This is called the **discrete logarithm problem for elliptic curves.**

**Ex. 3.5.1** *Consider the group $E_{23}(9, 17)$. This is the group defined by the equation $y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$. What is the discrete logarithm K of $Q = (4, 5)$ to the base $P = (16, 5)$ ?*

**Sol. :** The brute-force method is to compute multiples of P until Q is found.

Thus,

$P = (16, 5)$

$2P = (20, 20)$

$3P = (14, 14)$

$4P = (19, 20)$

$5P = (13, 10)$

$6P = (7, 3)$

$7P = (8, 7)$

$8P = (12, 17)$

$9P = (4, 5)$

Because $9P = (4, 5) = Q$, the discrete logarithm $Q = (4, 5)$ to the base $P = (16, 5)$ is $K = 9$.

**Analog of Diffie-Hellman key exchange**

A key exchange between users A and B can be accomplished as follows

1. A selects an integer $n_A$ less than n. This is A's private key. A then generates a public key $P_A = n_A \times G$; the public key is a point in $E_q$ (a, b).
2. B similarly selects a private key $n_B$ and computes a public key $P_B$.
3. A generates the secret key $K = n_A \times P_B$.

   B generates the secret key $K = n_B \times P_A$.

The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G)$$

$$= n_B \times (n_A \times G)$$

$$= n_B \times P_A$$

**Elliptic curve encryption and decryption**

- For an encryption/decryption, system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key $n_A$ and generates a public key $P_A = n_A \times G$.
- To encrypt and send message $P_m$ to user B, A chooses a random positive integer K and produces the ciphertext $C_m$ consisting of the pair of points

$$C_m = \{KG, P_m + KP_B\}$$

- To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point

$$= P_m + KP_B - n_B(KG)$$

$$= P_m + K(n_B G) - n_B(KG)$$

$$= P_m$$

**Review Question**

1. *Describe elliptic curve cryptography.*

## 3.6 Authentication Methods

- Authentication is a service used to provide the identity of an entity.
- Identification and authentication is the process that can be used to identify and verify the users on their secure systems. In secure system, the user must identify himself/herself, then the system will authenticate the identity before using the system.
- However, authentication verifies the user who is requesting access process of determining the identity of a user that is attempting to access a system

### 3.6.1 Password Based Authentication Methods

- Password is a front line protection against the unauthorized access (intruder) to the system. A password authenticate the identifier (ID) and provides security to the system. Therefore almost all systems are password protected.

**1] Password vulnerability**

- Passwords are extremely common. Passwords can often be guessed. Use of mechanisms to keep passwords secret does not guarantee that the system security can not be broken. It only says that it is difficult to obtain passwords. The intruder can always use a trial and error method. A test of only a limited

set of potential strings tends to reveal most passwords because there is a strong tendency for people to choose relatively short and simple passwords that they can remember. Some techniques that may be used to make the task of guessing a password difficult are as follows

1. Longer passwords.
2. Salting the password table.
3. System assistance in password selection.

- The length of a password determines the ease with which a password can be found by exhaustion. For example, 3-digit password provides 1000 variations whereas a four digit passwords provides 10,000 variations. Second method is the system assistance. A password can be either system generated or user selected. User selected passwords are often easy to guess. A system can be designed to assist users in using passwords that are difficult to guess.

**2] Encrypted passwords**

- Instead of storing the names and passwords in plain text form, they are encrypted and stored in cipher text form in the table. In this case, instead of directly using a user specified name and password for table lookup, they are first encrypted and then the results are used for table lookup. If the stored encoded password is seen, it can not be loaded, so the password cannot be determined. The password file does not need to be kept secret.

**3] One time passwords**

- Set of paired passwords solve the problem of password sniffing. When a session begins, the system randomly selects and presents one part of a password pair; user must supply the other part. In this, user is challenged and must respond with the correct answer to that challenge. In this method, the password is different in each instance. One time passwords are among the only ways to prevent improper authentication due to password exposure. Commercial implementations of one time password system such as secur ID, use hardware calculators.

**Password selection strategies**

- Too short password is too easy to guess. If the password is 8 random character, it is impossible to crack the password. In order to eliminate gaussable passwords four basic techniques are suggested.

1. User education
2. Computer generated password
3. Reactive password checking
4. Proactive password checking

### 3.6.2 Extensible Authentication Protocol

- Extensible Authentication Protocol is a universal authentication framework frequently used in wireless networks and Point-to-Point connections.
- The Extensible Authentication Protocol is a protocol commonly used in 802.1x to authenticate users.
- WPA and WPA2 standard has officially adopted five EAP types as its official authentication mechanisms.
- EAP is a way for a supplicant to authenticate, usually against a back-end RADIUS server. EAP comes from the dial access world and PPP.
- EAP sits inside PPP's authentication protocol. It provides a generalized framework for all sorts of authentication methods.
- EAP is supposed to head off proprietary authentication systems and let everything from passwords to challenge-response tokens and PKI certificates work smoothly.
- With a standardized EAP, interoperability and compatibility across authentication methods becomes simpler.
- Only the client and the authentication server have to be coordinated.
- By supporting EAP authentication, a RAS server (in wireless this is the AP) gets out of the business of actively participating in the authentication dialog.
- EAP supports multiple authentication methods, such as token card, Kerberos, one-time password, certificate, public key authentication, and smart card.

### 3.6.3 Biometric Authentication

**Biometric identification systems** can be grouped based on the main physical characteristic that lends itself to biometric identification.

- **Fingerprint identification** : Fingerprint ridges are formed in the womb; you have fingerprints by the fourth month of fetal development. Once formed, fingerprint ridges are like a picture on the surface of a

balloon. As the person ages, the fingers do get larger. However, the relationship between the ridges stays the same, just like the picture on a balloon is still recognizable as the balloon is inflated.

- **Hand geometry :** Hand geometry is the measurement and comparison of the different physical characteristics of the hand. Although hand geometry does not have the same degree of permanence or individuality as some other characteristics, it is still a popular means of biometric authentication.
- **Palm vein authentication :** This system uses an infrared beam to penetrate the users hand as it is waved over the system; the veins within the plam of the user are returned as black lines. Palm vein authentication has a high level of authentication accuracy due to the complexity of vein patterns of the palm. Because the palm vein patterns are internal to the body, this would be a difficult system to counterfeit. Also, the system is contactless and therefore hygienic for use in public areas.
- **Retina scan :** A retina scan provides an analysis of the capillary blood vessels located in the back of the eye; the pattern remains the same throughout life. A scan uses a low intensity light to take an image of the pattern formed by the blood vessels. Retina scans were first suggested in the 1930's.
- **Iris scan :** An iris scan provides an analysis of the rings, furrows and freckles in the colored ring that surrounds the pupil of the eye. More than 200 points are used for comparison.

**Face recognition :** Facial characteristics are depends on the size and shape of facial characteristics, and their relationship to each other. Although this method is the one that human beings have always used with each other, it is not easy to automate it. Typically, this method uses relative distances between common landmarks on the face to generate a unique "faceprint".

**Signature :** Although the way you sign your name does change over time, and can be consciously changed to some extent, it provides a basic means of identification.

- **Voice analysis :** The analysis of the pitch, tone, cadence and frequency of a person's voice.

**Advantages**

There are a number of advantages to this technology

- Biometric identification can provide extremely accurate, secured access to information; fingerprints, retinal and iris scan produce absolutely unique data sets when do properly.
- Current methods like password verification have many problems (people write them down, they forget them, they make up easy-to-hack passwords).
- Automated biometric identification can be done very rapidly and uniformly, with a minimum of training.
- Your identity can be verified without resort to documents that may be stolen, lost or altered.

**Review Question**

1. *What is authentication ? Explain various methods for authentication.*

## 3.7 Message Digest

- A message-digest algorithm is also called **a hash function** or a cryptographic hash function.
- It accepts a message as input and generates a fixed-length output, which is generally less than the length of the input message. The output is called a **hash value**, a fingerprint or a message digest.
- Message Digest 5 (MD5) processes the input text in 512-bit blocks. These blocks are further divided into 16 32-bit sub blocks.
- MD5 is a 128-bit hash.
- The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private key under a public-key cryptosystem such as RSA.

### 3.7.1 MD5 Description

- Suppose if we have b-bit message as input, and that we wish to find its message digest. Here b is an arbitrary non-negative integer; b may be zero, it need not be a multiple of eight and it may be arbitrarily

large. The bits of the message written down as follows:

m_0 m_1 ... m_{b-1}

- The following five steps are performed to compute the message digest of the message.

**Step 1 : Append Padding Bits**

- The message is "padded" so that its length is congruent to 448, modulo 512. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.
- Padding is performed as follows : A single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

**Step 2 : Append Length**

- A 64-bit representation of b is appended to the result of the previous step. In the unlikely event that b is greater than $2^{64}$, and then only the low-order 64 bits of b are used.
- At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words.
- Let M[0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

**Step 3 : Initialize MD Buffer**

- A four-word buffer (A, B, C, and D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal :

**Word A : 01 23 45 67**

**Word B : 89 ab cd ef**

**Word C : fe dc ba 98**

**Word D : 76 54 32 10**

**Step 4 : Process Message in 16-Word Blocks**

- We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

- In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of v since XY and not(X)Z will never have 1's in the same bit position.
- It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X, Y, Z) will be independent and unbiased.
- The functions G, H and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be independent and unbiased.
- This step uses a 64-element table T[1 ... 64] constructed from the sine function. Let T[i] denote the i-th element of the table, which is equal to the integer part of 4294967296 times abs(sin(i)), where i is in radians.

**Step 5 : Output**

- The message digest produced as output is A, B, C, and D. That is, we begin with the low-order byte of A, and end with the high-order byte of D.

### 3.7.2 Differences between MD4 and MD5

The following are the differences between MD4 and MD5 :

1. A fourth round has been added.
2. Each step now has a unique additive constant.
3. The function g in round 2 was changed from (XY v XZ v YZ) to (XZ v Y not(Z)) to make g less symmetric.
4. Each step now adds in the result of the previous step. This promotes a faster "avalanche effect".
5. The order in which input words are accessed in rounds 2 and 3 is changed, to make these patterns less like each other.
6. The shift amounts in each round have been approximately optimized, to yield a faster "avalanche effect." The shifts in different rounds are distinct.

### 3.7.3 Comparison between MD5 and SHA

| Sr. No. | MD5 | SHA |
|---|---|---|
| 1. | MD length is 128-bits | Length is 160-bits |
| 2. | Speed is faster than SHA | Slower than MD5 |
| 3. | Number of iteration is 64 | Number of iteration is 80 |
| 4. | Buffer space is 128-bits | Buffer space is 160-bits |
| 5. | MD5 is vulnerable to cryptanalytic attacks | SHA-1 appears not to be vulnerable to cryptanalytic attack |
| 6. | MD5 uses a little endian scheme | SHA-1 uses a big endian scheme |
| 7. | Simple to implement and do not need any large programs or complex table | Simple to implement and do not need any large programs or complex table. |
| 8. | No limit on maximum message size. | Maximum message size is $2^{64} - 1$ bits. |

**Review Questions**

1. *Explain operation of MD5 message digest algorithm.*
2. *What is message digest ? Compare MD5 with SHA - 1.*

## 3.8 Kerberos

- Kerberos is an **authentication protocol**. It provides a way to authenticate clients to services to each other through a trusted third party.
- Kerberos makes the assumption that the connection between a client and service is insecure. Passwords are encrypted to prevent others from reading them. Clients only have to authenticate once during a pre-defined lifetime.
- Kerberos was designed and developed at MIT by Project Athena. Currently, Kerberos is upto Version 5. Version 4 being the first version to be released outside of MIT.
- Kerberos has been adopted by several private companies as well as added to several operating systems.
- Its creation was inspired by client-server model replacing time-sharing model. Kerberos is a network authentication protocol designed to allow users, clients and servers, authenticate themselves to each other.
- This **mutual authentication** is done using **secret-key cryptography** with parties proving to each other their identity across an insecure network connection.
- Communication between the client and the server can be secure after the client and server have used Kerberos to prove their identity.
- From this point on, subsequent communication between the two can be encrypted to assure privacy and data integrity.

**Requirement of Kerberos**

- Kerberos client/server authentication requirements are :
  1. **Security** : That Kerberos is strong enough to stop potential eavesdroppers from finding it to be a weak link.
  2. **Reliability** : That Kerberos is highly reliable employing a distributed server architecture where one server is able to back up another. This means that Kerberos systems are fail safe, meaning graceful degradation, if it happens.
  3. **Transparency** : That user is not aware that authentication is taking place beyond providing passwords.
  4. **Scalability** : Kerberos systems accept and support new clients and servers.

  To meet these requirements, Kerberos designers proposed a third-party trusted authentication service to arbitrate between the client and server in their mutual authentication.

### 3.8.1 Kerberos Terminology

- Kerberos has its own terminology to define various aspects of the service.
  1. **Authentication Server (AS)** : A server that issues tickets for a desired service which are in turn given to users for access to the service.
  2. **Client** : An entity on the network that can receive a ticket from Kerberos.
  3. **Credentials** : A temporary set of electronic credentials that verify the identity of a client for a particular service. It also called a ticket.
  4. **Credential cache or ticket file** : A file which contains the keys for encrypting communications between a user and various network services.
  5. **Crypt hash** : A one-way hash used to authenticate users.

6. **Key** : Data used when encrypting or decrypting other data.
7. **Key Distribution Center (KDC)** : A service that issue Kerberos tickets and which usually run on the same host as the Ticket-Granting Server (TGS).
8. **Realm** : A network that uses Kerberos composed of one or more servers called KDCs and a potentially large number of clients.
9. **Ticket-Granting Server (TGS)** : A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.
10. **Ticket-Granting Ticket (TGT)** : A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

### 3.8.2 Kerberos Version 4

- Kerberos version 4 uses DES for providing authentication service. Some aspect of version 4 are
  A) Simple Authentication Dialogue.
  B) More Secure Authentication Dialogue.

#### 3.8.2.1 Simple Authentication Dialogue

- For a secure transaction, server should confirm the client and its request. In unprotected network it creates burden on server, therefore an Authentication Server (AS) is used. The Authentication Server (AS) maintains password of all users in centralized database. Also the authentication server shares a unique secret key with each server.
- Let
  Client is represented as C
  Authentication server is represented as AS
  Server is represented as V
  Identifier of user on C is represented as $ID_C$
  Identifier of V is represented as $ID_V$
  Password of user on C is $P_C$
  Network address of C is represented as $AD_C$
  Secret encryption key shared by AS and V is $K_V$
  Then consider a hypothetical dialogue.

| | Sender and receiver | | Contents of message |
|---|---|---|---|
| 1. | C → AS | : | IDc \|\| Pc \|\| IDv |
| 2. | AS → C | : | Ticket |
| 3. | C → V | : | IDc \|\| Ticket |
| 4. | Ticket | = | E [Kv, (IDc \|\| ADc \|\| IDv )] |

**Explanation**

1. **Client Clogs on to workstation requesting to access to server V** : The workstation requests user's password and sends message to AS including user ID + Server ID + User password. The AS checks this message with database and verifies it.
2. **AS issues ticket** : On verifying the tests. AS issues ticket containing user ID + Server ID + Network address.

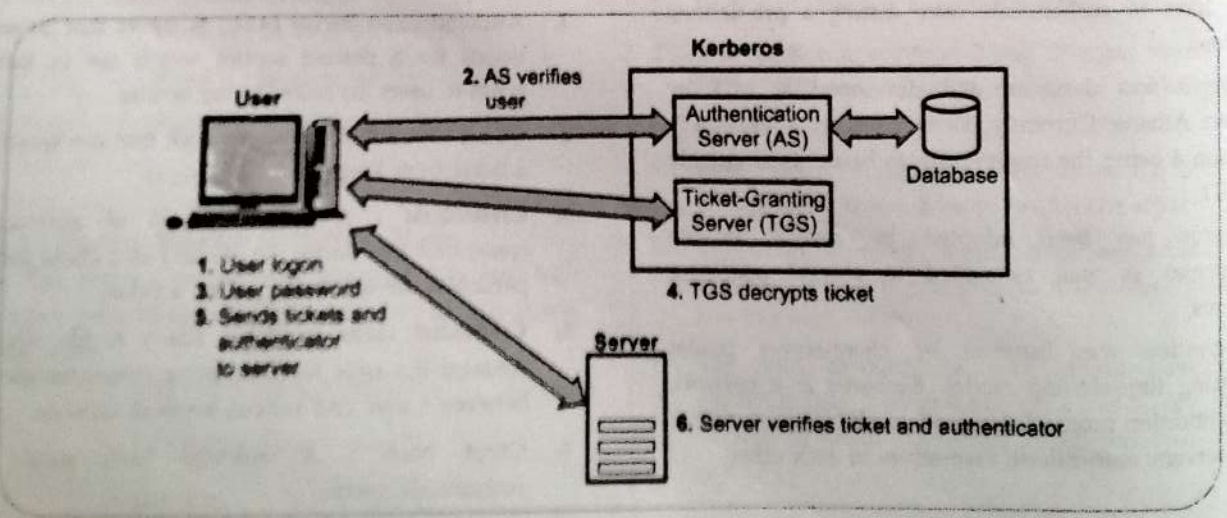


Fig. 3.8.1 Overview of kerberos

3. **Client C applies server V :** With this ticket, client C asks server V for access. Server V decrypts the ticket and verify the aunthenticity of data then grants the requested service. In above hypothetical dialogue, symbol || represents concatenation.

### 3.8.2.2 Secure Authentication Dialogue

- Kerberos version 4 protocol ensures secure authentication dialogue involving three sessions.
  - i] **Authentication Service :** Exchange to obtain ticket-granting ticket.
  - ii] **Ticket-granting Service :** Exchange to obtain service granting ticket.
  - iii] **Client/server authentication :** Exchange to obtain service.
- Each of the above session has two steps, as shown in table below

| Session | Step | Sender-Receiver |
|---|---|---|
| i] | 1. | C → AS |
| | 2. | AS → C |
| ii] | 3. | C → TGS (Ticket-granting server) |
| | 4. | TGS → C |
| iii] | 5. | C → V |
| | | V → C |

- Fig. 3.8.1 shows how the steps are executed in Kerberos version 4.

### 3.8.2.3 Kerberos Realms

- The constituents of a full-service Kerberos environment are
  a) A Kerberos server
  b) Clients
  c) Number of application server.
- **Requirements of Kerberos sever :**
  a) Kerberos server should have user ID.
  b) Hashed password for all users.
  c) All users should be registered with Kerberos server.
  d) Kerberos server should have secret key with each server.
  e) All servers should be registered with Kerberos server.
- A **Kerberos realm** is referred as is the environment where
  - All nodes share same secured database.
  - Changing and accessing the Kerberos database requires Kerberos master password.
  - A read only copy of Kerberos database resides in computer system.
- Networks have different realms under different administrative organizations. The users of one realm may access the servers in other realm provided the users are authenticated. The interoperating Kerberos shares a secret key with the server in other realm.

### 3.8.3 Kerberos Version 5

- Version 4 of Kerberos have some environmental shortcoming and technical deficiencies.

**Environmental shortcomings of version 4**

1. Encryption system dependence
2. Internet protocol dependence
3. Message byte ordering
4. Ticket lifetime
5. Authentication forwarding
6. Inter realm authentication.

**Technical deficiencies of version 4**

1. Double encryption
2. PCBC (Propagating Cipher Block Chaining) encryption
3. Session keys
4. Password attacks

### 3.8.3.1 Version 5 Authentication Dialogue

The Kerberos version 5 message exchange involves three session, these are

1. Authentication service exchange
2. Ticket - granting exchnage
3. Client/server authentication exchange

- Each session has two steps. Table 3.8.1 summarizes session, steps and their functions.

| | Session | Step | Function |
|---|---|---|---|
| i] | Application service exchange | C → AS<br>AS → C | To obtain ticket-granting ticket. |
| ii] | Ticket-granting service exchange | C → TGS<br>TGS → C | To obtain service-granting ticket. |
| iii] | Client/Server authentication exchange | C → V<br>V → C | To obtain service. |

**Table 3.8.1**

- The flags field is expanded in ticket in version 5 of Kerberos. Various flags that may be included in a ticket, are

i) INITIAL ii) PRE-AUTHENT
iii) HW-AUTHENT iv) RENEWABLE
v) MAY-POSTDATE vi) POSTDATED
vii) INVALID viii) PROXIABLE
ix) PROXY x) FORWARDABLE
xi) FORWARDED

### 3.8.4 Comparison between Kerberos Versions 4 and 5

| Parameters | Kerberos Versions 4 | Kerberos Versions 5 |
|---|---|---|
| Encryption algorithms used | DES only | DES and other encryptions |
| Ticket lifetime | 5 min units, Maximum = 1280 minutes | Start and end time is arbitrary |
| Message byte ordering | Tagged message with ordering | Abstract syntax notation on basis encoding rules. |
| Password attack | Initial request in clear and use it for offline attack. | Need to send pre-authentication data |
| Two times encryption | Supported | Not supported |
| Session keys | Replay risk using repeated ticket | Sub session key once only |
| Hierarchy of realms | Limits to pairs | Transition allowed |

### 3.8.5 Strengths of Kerberos

1. Passwords are never sent across the network unencrypted. This prevents those unscrupulous people from being able to read the most important data sent over the network.
2. Clients and applications services mutually authenticate. Mutual authentication allows for both ends to know that they truly know whom they are communicating with.
3. Tickets have a limited lifetime, so if they are stolen, unauthorized use is limited to the time frame that the ticket is valid.
4. Authentication through the AS only has to happen once. This makes the security of Kerberos more convenient.
5. Shared secret keys between clients and services are more efficient than public-keys.
6. Many implementations of Kerberos have a large support base and have been put through serious testing.
7. Authenticators, created by clients, can only be used once. This feature prevents the use of stolen authenticators.

### 3.8.6 Weakness of Kerberos

1. Kerberos only provides authentication for clients and services.
2. Kerberos 4 uses DES, which has been shown to be vulnerable to brute-force-attacks with little computing power.
3. The principal-key database on the KDC has to be hardened or else bad things can happen.
4. Like any security tool, it is also vulnerable to users making poor password choices.
5. Kerberos doesn't work well in a time-sharing environment.
6. Kerberos requires a continuously available Kerberos Server. If the Kerberos Server goes down, the Kerberos network is unusable.
7. Kerberos does not protect against modifications to system software like Trojan horses.

### 3.8.7 Difference between Kerberos and SSL

| Sr. No. | Kerberos | SSL |
|---|---|---|
| 1. | Uses private key encryption. | Uses public key encryption. |
| 2. | Based on the trusted third party. | Based on certificate. |
| 3. | Ideal for network environment. | Ideal for the WWW. |
| 4. | Key revocation can be accomplished by disabling a user at the authentication server. | Key revocation requires revocation server to keep track of bad certificate. |
| 5. | Password resides in user's minds where they are usually not subject to secret attack. | Certificates sit on a user hard drive where they are subject to being cracked. |
| 6. | Kerberos open source and free available. | Uses patented material, so the service is not free. |

**Review Questions**

1. *Explain the operation of Kerberos.*
2. *What is kerberos ? Explain its operation.*

## 3.9 X.509 Authentication Service

- X.509 is part of X.500 recommendations for directory service i.e. set of servers which maintains a database of information about users and other attributes.
- X.509 defines authentication services e.g. certificate structure and authentication protocols. Also X.509 also defines alternative authentication protocols base on use of public-key certificates. The X.509 certificate format is emplied in S/MIME, IP security, SET and SSL/TLS.
- X.509 standard uses RSA algorithm and hash function for digital signature. Fig. 3.9.1 shows generation of public key certificate.

### 3.9.1 X.509 Format of Certificate

- The current version of the standard is version 3, called as X.509V3. The general format of digital certificate X.509V3 is shown in Fig. 3.9.2.

| | |
|---|---|
| 1 | Version |
| 2 | Certificate Serial Number |
| 3 | Signature Algorithm Identifier |
| 4 | Issuer Name |
| 5 | Period of Validity |
| 6 | Subject Name |
| 7 | Subject's Public Key Info. |
| 8 | Issuer Unique Identifier |
| 9 | Subject Unique Identifier |
| 10 | Extensions |
| 11 | Signature |

**Fig. 3.9.2 X.509 Digital certificate format version 3**

1. **Version** : Identifies successive versions of certificate format the default is version.
2. **Certificate Serial Number** : It contains an unique integer number, which is generated by Certification Authority (CA).
3. **Signature Algorithm Identifier** : Identifies the algorithm used by the CA to sign the certificate.
4. **Issuer Name** : Identifies the distinguished name of the CA that created and signed this certificate.
5. **Period of Validity** : Consists of two date-time values (not before and not after) within which the certificate is valid.
6. **Subject Name** : It specifies the name of the user to whom this certificate is issued.

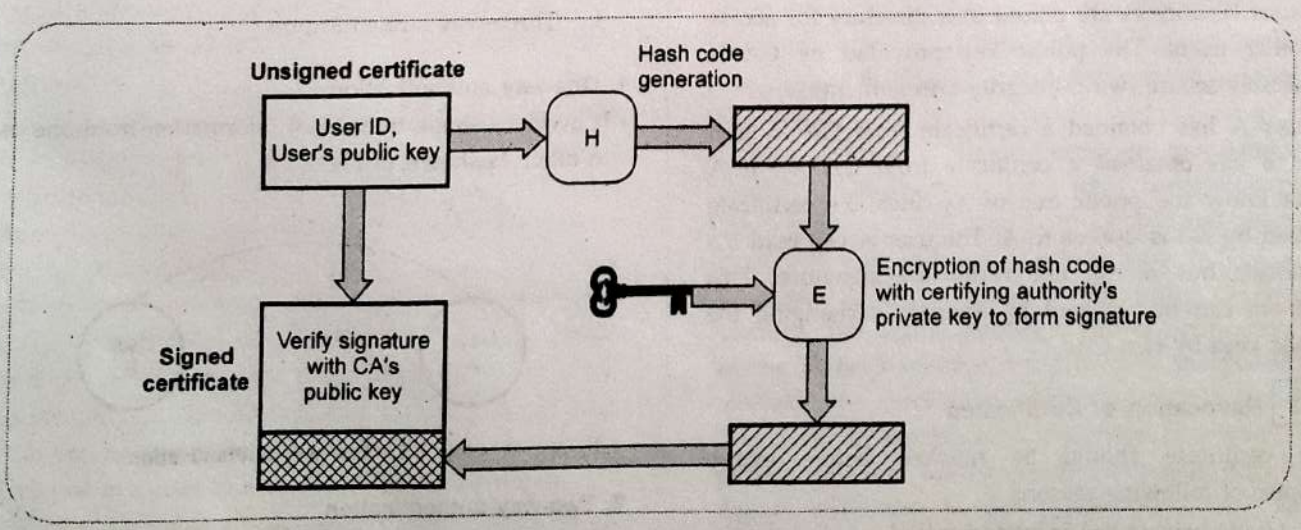


**Fig. 3.9.1 Public key certificate**

7. **Subject's Public Key Information** : It contains public key of the subject and algorithms related to that key.
8. **Issuer Unique Identifier** : It is an optional field which helps to identify a CA uniquely if two or more CAs have used the same Issuer Name.
9. **Subject Unique Identifier** : It is an optional field which helps to identify a subject uniquely if two or more subjects have used the same Subject Name.
10. **Extensions** : One or more fields used in version 3. These extensions convey additional information about the subject and issuer keys.
11. **Signature** : It contains hash code of the fields, encrypted with the CA's private key. It includes the signature algorithm identifier.

**Standard notations for defining a certificate**

$$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, T_A A, A_P\}$$

where,

$CA\langle\langle A\rangle\rangle$ indicates the certificate of user A issued by certification authority CA.

$CA\{V \ldots\ldots A_P\}$ indicates signing of $V \ldots\ldots A_P$ by CA.

## 3.9.2 Obtaining User's Certificate

- The characteristics of user certificate are -
  1. Any user who can access public key of CA can verify user public key.
  2. Only certification Authority (CA) can modify the certificate.
- All user certificates are placed in a directory for access of other users. The public key provided by CA is absolutely secure (w.r.t. integrity and authenticity).
- If user A has obtained a certificate from CA $X_1$ and user B has obtained a certificate from CA $X_2$. If A don't know the public key of $X_2$, then B's certificate (issued by $X_2$) is useless to A. The user A can read B's certificate but A can not verify the signature. This problem can be resolved by securely exchanging the public keys by two CAs.

## 3.9.3 Revocation of Certificates

- The certificate should be revoked before expiry because of following reasons :
  1. User's private key is compromised.
  2. User is not certified by CA.
  3. CA's certificate is compromised.
- Each CA has a list of all revoked but not expired certificates. The Certificate Revocation List (CRL) is posted in directory signed by issuer and includes issuer's name, date of creation, date of next CRL. Fig. 3.9.3 Certificate revocation list. Each certificate has unique serial number of identify the certificate.

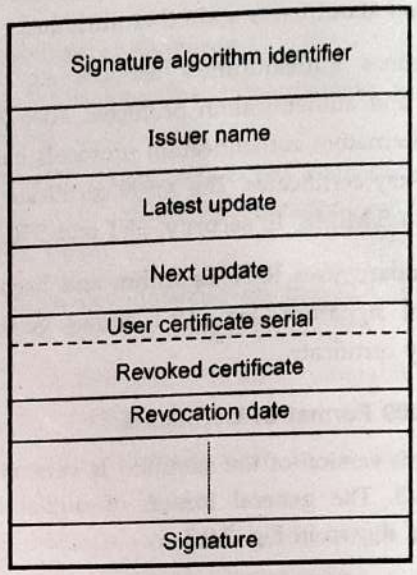


**Fig. 3.9.3 Certificate revocation list**

## 3.9.4 Authentication Procedures

- X.509 supports three types of authenticating using public key signatures. The types of authentication are
  1. One-way authentication
  2. Two-way authentication
  3. Three-way authentication

**1. One-way authentication**

- It involves single transfer of information from one user to other as shown in Fig. 3.9.4.

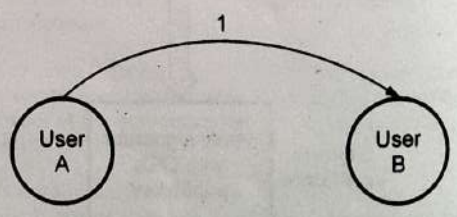


**Fig. 3.9.4 One way authentication**

**2. Two-way authentication**

- Two-way authentication allows both parties to communicate and verify the identity of the user.

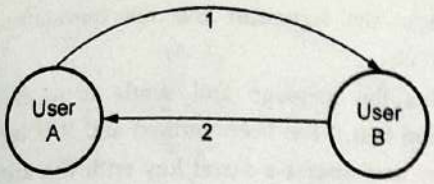


Fig. 3.9.5 Two-way authentication

**3. Three-way authentication**

- Three-way authentication is used where synchronized clockes are not available Fig. 3.9.6 shows three-way authentication.

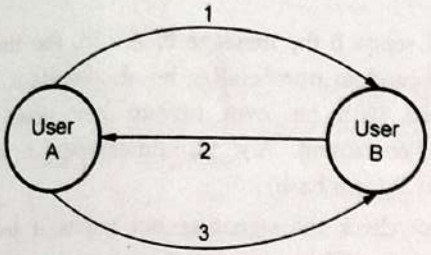


Fig. 3.9.6 Three-way authentication

## 3.9.5 Digital Certificate

- A data structure that securely binds an individual or entity to a public key used in cryptographic operations such as digital signatures or asymmetric encryption.
- To obtain digital certificate an organization must apply to a certification authority which is responsible for validating and ensuring the authenticity of requesting organization. The certificate will identify the name of the organization, a serial number, the validity date and the organization's public key where encryption to / from that organization is required.
- In addition, the digital certificate will also contain the digital signature of the certification authority to allow any recipient to confirm the authenticity of the digital certificate.
- A digital certificate is an ID that is carried with a file. To validate a signature, a certifying authority validates information about the software developers and then issues them digital certificates. The digital certificate contains information about the person to whom the certificate was issued, as well as information about the certifying authority that issued it. When a digital certificate is used to sign programs, ActiveX controls, and documents, this ID is stored with the signed item in a secure and verifiable form so that it can be displayed to a user to establish a trust relationship.

A digital certificate allows unique identification of an entity; it is essentially an electronic ID card, issued by a trusted third party. Digital certificates form part of the ISO authentication framework, also known as the X.509 protocol. This framework provides for authentication across networks.

- A digital certificate serves two purposes: it establishes the owner's identity, and it makes the owner's public key available. A digital certificate is issued by a Certification Authority (CA). It is issued for only a limited time, and when its expiry date has passed, it must be replaced.
- A digital certificate consists of :
  1. The public key of the person being certified
  2. The name and address of the person being certified, also known as the Distinguished Name (DN)
  3. The digital signature of the CA
  4. The issue date
  5. The expiry date
- The Distinguished Name is the name and address of a person or organization. You enter your Distinguished Name as part of requesting a certificate. The digitally-signed certificate includes not only your own Distinguished Name, but the Distinguished Name of the CA, which allows verification of the CA.
- To communicate securely, the receiver in a transmission must trust the CA that issued the certificate that the sender is using. This means that when a sender signs a message, the receiver must have the corresponding CA's signer certificate and public key designated as a trusted root key. For example, your web browser has a default list of signer certificates for trusted CAs. If you want to trust certificates from another CA, you must receive a certificate from that CA and designate it as a trusted root key.
- If you send your digital certificate containing your public key to someone else, what keeps that person from misusing your digital certificate and posing as you ? The answer is : your private key.
- A digital certificate alone is not proof of anyone's identity. The digital certificate allows verification only of the owner's identity, by providing the public key needed to check the owner's digital signature. Therefore, the digital certificate owner must protect the private key that belongs with the public key in the digital certificate. If the private key were stolen, anyone could pose as the legitimate owner of the digital certificate.

## 3.10 Digital Signatures

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature is formed by taking the hash of the message and encrypting the message with the creator's private key.

**Requirements**

- Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other.
- In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature is analogous to the handwritten signature.
- It must have the following properties
  1. It must verify the author and the date and time of the signature.
  2. It must to authenticate the contents at the time of the signature.
  3. It must be verifiable by third parties, to resolve disputes.
- The digital signature function includes the authentication function. On the basis of these properties, we can formulate the following requirements for a digital signature.
- Must be a bit pattern depending on the message being signed.
- Signature must use some information unique to the sender to prevent forgery and denial.
- Computationally easy to produce a signature.
- Computationally easy to recognize and verify the signature.
- Computationally infeasible to forge a digital signature.
  a) either by constructing a new message for an existing digital signature.
  b) or by constructing a fraudulent digital signature for given message.
- Practical to retain a copy of the digital signature in storage.

**Two general schemes for digital signatures**

1) Direct 2) Arbitrated

### 3.10.1 Arbitrated Digital Signatures

Every signed message from A to B goes to an arbiter BB (Big Brother) that everybody trusts.

- BB checks the signature and the timestamp, origin, content, etc.
- BB dates the message and sends it to B with an indication that it has been verified and it is legitimate.

**e.g. Every user shares a secret key with the arbiter**

- A sends to BB in an encrypted form the plaintext P together with B's id, a timestamp and a random number RA.
- BB decrypts the message and thus makes sure it comes from A; it also checks the timestamp to protect against replays.
- BB then sends B the message P, A's id, the timestamp and the random number RA; he also sends a message encrypted with his own private key (that nobody knows) containing A's id, timestamp t and the plaintext P (or a hash).
- B cannot check the signature but trusts it because it comes from BB-he knows that because the entire communication was encrypted with KB.
- B will not accept the messages or messages containing the same RA to protect against replay.
- In case of dispute, B will show the signature he got from BB (only B may have produced it) and BB will decrypt it.

### 3.10.2 Direct Digital Signature

- This involves only the communicating parties and it is based on public keys.
- The sender knows the public key of the receiver.
- Digital signature : Encrypt the entire message (or just a hash code of the message) with the sender's private key.
- If confidentiality is required : Apply the receiver's public key or encrypt using a shared secret key.
- In case of a dispute the receiver B will produce the plaintext P and the signature E(KRA, P) - the judge will apply KUA and decrypt P and check the match : B does not know KRA and cannot have produced the signature himself.

**Weaknesses**

- The scheme only works as long as KRA remains secret : If it is disclosed (or A discloses it herself), then the argument of the judge does not hold : Anybody can produce the signature.
- **Attack :** To deny the signature right after signing, simply claim that the private key has been lost-similar to claims of credit card misuse.

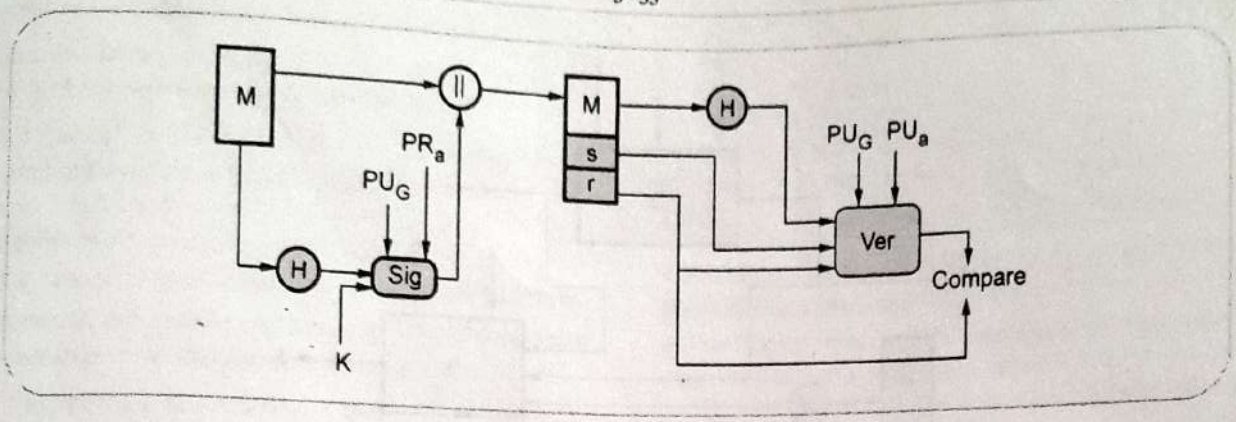


Fig. 3.10.1 DSS approach

i.e. If A changes her public-private keys (she can do that often) the judge will apply the wrong public key to check the signature.

- **Attack** : To deny the signature change your public-private key pair-this should not work if a PKI is used because they may keep trace of old public keys.

  i.e. A should protect her private key even after she changes the key.

- **Attack :** Eve could get hold of an old private key and sign a document with an old timestamp.

### 3.10.3 Digital Signature Standard

- The Digital Signature Standard (DSS) makes use of the Secure Hash Algorithm (SHA) and presents a new digital signature technique, the Digital Signature Algorithm (DSA). DSS cannot be used for encryption or key exchange. Fig. 3.10.1 shows the DSS approach.
- It uses a hash function. The hash code is provided as input to a signature function along with a random number K generated for this particular signature.
- The signature function also depends on the sender's private key ($PR_a$) and a set of parameters known to a group of communicating principles.
- The result is a signature consisting of two components, labeled s and r.
- At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function.
- Fig. 3.10.2 shows the RSA approach. In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted.
- The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid.

### 3.10.4 Digital Signature Algorithm

- There are three parameters that are public and can be common to a group of users. **Prime number q** is chosen and it is **160-bit**. A **prime number p** is selected with a length between **512** and **1024 bits** such that q divides (P – 1).

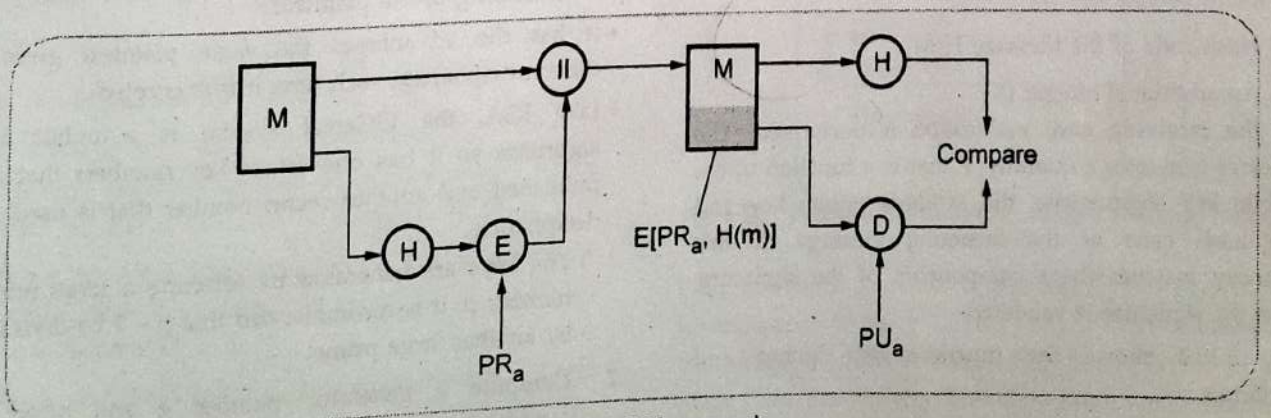


Fig. 3.10.2 RSA approach

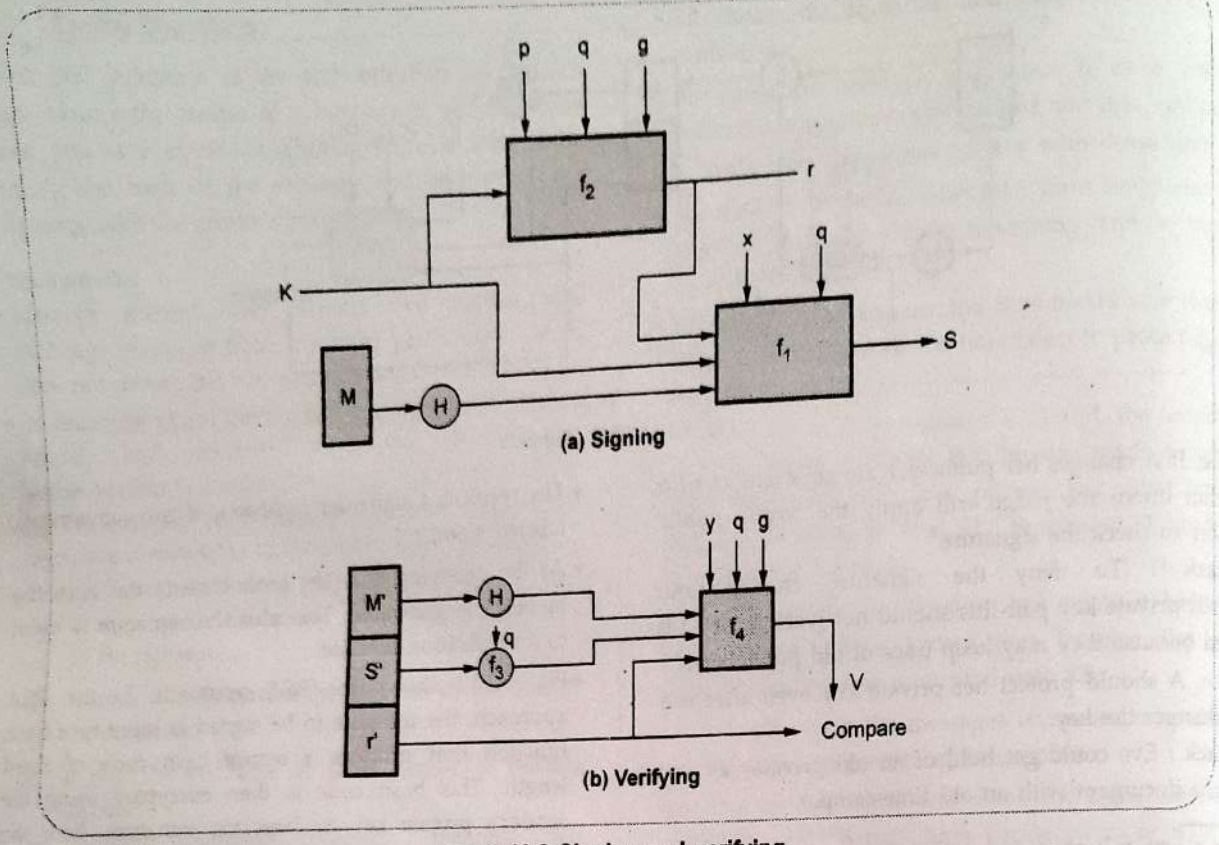


**Fig. 3.10.3 Signing and verifying**

- g is chosen to be of the form $h^{(p-1)/q}$ mod p where h is an integer between 1 and (p – 1).
- With these number, user selects a private key and generate a public key. The private key x must be a number from 1 to (q – 1) and should be chosen randomly or pseudorandomly.
- The public key is calculated from the private key as $y = g^x$ mod p.
- To create a signature, a user calculates two quantities, **rands**, that are functions of

  i) Public key components (p, q, g)

  ii) User's private key (x)

  iii) Hash code of the message H(M)

  iv) An additional integer (K)
- **At the receiving end**, verification is performed. The receiver generates a quantity V that is a function of the public key components, the sender's public key and the hash code of the incoming message. If this quantity matches the r components of the signature, then the signature is validated.
- Fig. 3.10.3 shows the functions of signing and verifying.

### 3.10.5 ELGamal Digital Signatures

- The ElGamal algorithm provides an alternative to the RSA for public key encryption.

1. Security of the RSA depends on the difficult of factoring large integers.
2. Security of the ElGamal algorithm depends on the difficulty of computing discrete logs in a large prime modulus.

- ElGamal has the disadvantage that the ciphertext is twice as long as the plaintext.
- It has the advantages the same plaintext gives a different ciphertext each time it is encrypted.
- Like RSA, the ElGamal system is a public key algorithm so it has one set of key numbers that are published and another secret number that is used for deciphering.

1. The keys are generated by selecting a large prime number p. It is recommended that p – 1 be divisible by another large prime.
2. Compute a generator number g and select a random integer "a" less than p – 1.

3. With these numbers compute $b = g^a \pmod p$.
4. The public key consists of the three number (p, g, b) and the secret key is the number a.
5. To find "a" given the public key, an attacker must be able to solve the discrete logarithm problem.

**Encryption :**

- If Bob wants to send a message to Alice he begins by looking up her public key (p, g, b) and representing the message as an integer m in the range 0 to p – 1.
- He then selects a random key, k that is less than p – 1.
- Using these numbers, Bob computes two numbers :

$$c_1 = g^k \quad \text{and} \quad c_2 = mb^k$$

- He sends $(c_1, c_2)$ to Alice.

**Decryption :**

- When Alice receives the cipher-text, she will recover the plaintext using her secret key, "a" to compute :

$$m = c_2 c_1^{-a} \bmod p$$

- This works because :

$$c_2 c_1^{-a} = mb^k (g^k)^{-a}\, mb^k\, (g^a)^k\, (g^k)^{-a}$$
$$= mg^{ak} g^{-ak} = m \pmod p$$

- Bob should choose a different random integer k for each message he sends to Alice. If M is a longer message, so it is divided into blocks, he should choose a different k for each block.
- Say he encrypts two messagers (or blocks) $M_1$ and $M_2$, using the same k, producing cipher-texts.
- Eve intercepts both cipher-text messages and discovers one plaintext message $M_1$, she can compute the other plaintext message $M_2$.

**Example :** Alice selected her initial prime number p = 11, found the primitive element g = 7 and selected her random secret key a = 2, then her public key is :

$$b = 7^2 \bmod 11 = 5$$

- She would publish her public key : (11, 7, 5)
- Bob wants to send the letter "a" to Alice

1. He first breaks it up into a set of numbers where each number is less than 11 (the value of p).
2. Since the ASCII representation of "a" is 01100001, he might break it up into four messages (01 10 00 01) or in decimal (1, 2, 0, 1).
3. Next, he would select a random number k = 3 and then compute and send to Alice :

| m | $c_1$ | $c_2$ |
|---|---|---|
| 1 | $7^3 \bmod 11 = 2$ | $1 \times 5^3 \bmod 11 = 4$ |
| 2 | $7^3 \bmod 11 = 2$ | $2 \times 5^3 \bmod 11 = 8$ |
| 0 | $7^3 \bmod 11 = 2$ | $0 \times 5^3 \bmod 11 = 0$ |
| 1 | $7^3 \bmod 11 = 2$ | $1 \times 5^3 \bmod 11 = 4$ |

- The cipher-text is ((2, 4), (2, 8), (2, 0), (2, 4)).

**Deciphering a Message**

- When Alice receives this message from Bob, she uses her secrete key a = 2 as follows :

(2, 4) : $m = 4(2) - 2 = 4(4) - 1 = 12 \bmod 11 = 1$ (4 and 3 are inverse mod 11)

(2, 8) : $m = 8(2) - 2 = 8(4) - 1 = 24 \bmod 11 = 2$

(2, 0) : $m = 0(2) - 2 = 0(4) - 1 = 0 \bmod 11 = 0$

(2, 4) : m = 1

Alice reassembles the message into the letter "a".

**Review Question**

1. *Explain digital signature algorithm.*

## 3.11 Authentication Protocol

### 3.11.1 Mutual Authentication

It is often necessary for both communicating themselves to each other.

#### 3.11.1.1 Based on a Shared Secret Key

In this protocol, a secret key is shared with both party. i.e. source and destination. One party sends random number to the other, other side transforms it in a special way and then returns a result. This type of protocols are called challenge-response protocols. The working of this protocol is as follows.

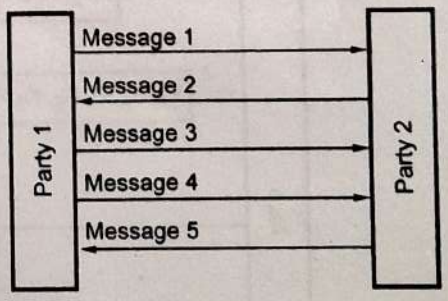


**Fig. 3.11.1 Two way authentication using a challenge-response protocol**

First the party 1 sends a message 1 to party 2 i.e. identification of party 1. The party 2 needs to find out the message which it received is from party 1 or any other third party. Party 2 sends a large random number

to party 1 in plaintext. The party 1 then encrypts the message with the key which shares with party 2 and sends the ciphertext back in message 3. When party 2 receives this message, they know that message is from party 1 because of the shared secret key. Uptill now party 2 is sure only about communication, but party 1 is not sure about the communication between him and party 2. The party 1 sends a random number to party 2 as plaintext in message 4. When party 2 responds with secret key, party 1 knows they are communicating with party 2. This protocol has some disadvantages. It is slower and contains extra messages. These can be eliminated by combining information.

### 3.11.1.2 Using Public Key Cryptography

In this method, A sends a random number $R_A$ and identity by encrypting. A uses B's public-key $E_B$ for sending message. When B receives this messages, B sends A back a message containing A's random number $R_A$ and his own random number $R_B$ and a proposed session key, $K_s$. When A gets message 2, A decrypts it using private key. After examining the message 2, A finds out the random number $R_A$. A knows that message 2 is from B only. Then A agrees to the session by sending back message 3 to B. When B reads $R_B$ encrypted with the session key which is generated by B, B knows that A got message 2 and verified $R_A$.

This protocol does has some disadvantage. It assumes that both user (A and B) already know each others public keys.

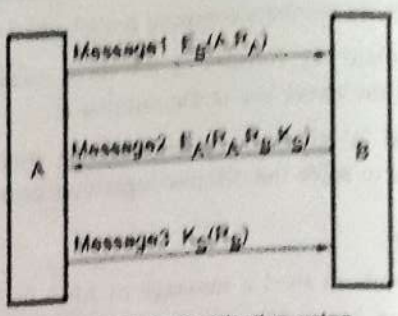


Fig. 3.11.2 Authentication using public key cryptography

### 3.11.2 Needham Schroeder Protocol

- The Needham Schroeder protocol refers to two methods of communication protocols through an insecure network.

1. Needham Schroeder symmetric key protocol, which is based on symmetric encryption algorithm to establish a session key between two parties in a network.
2. Needham Schroeder public-key protocol, based on the public key cryptography to provide mutual authentication between two communication parties over a network.

**Needham Schroeder public key authentication protocol**

- The Needham Schroeder public key authentication protocol aims to provide a mutual authentication between two parties Alice (A) and Bob (B).
- Both parties want to insure each other identity before starting to communicate. The protocol is as follows :

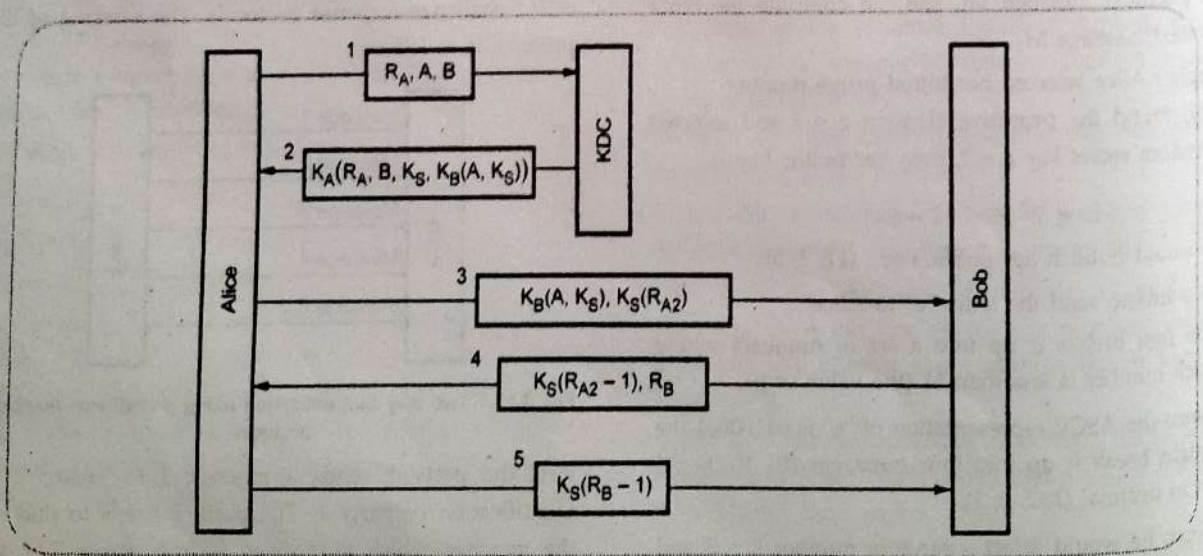


Fig. 3.11.3 Needham Schroeder authentication protocol

a. $K_A$ and $K_B$ are Alice's public key and Bob's public key respectively,

b. $N_A$ and $N_B$ are nonces generated by A and B respectively.

1. $A \rightarrow B : \{N_A, A\}_{K_B}$ (Init)

Alice generates a nonce $N_A$ and sends it to Bob with her identity. Everything is encrypted using Bob's public key.

2. $B \rightarrow A : \{N_A, N_B\}_{K_A}$ (Challenge)

Bob generates a nonce $N_B$, and sends it to Alice with $N_A$ he has just received. It is a way to prove that he is really the owner of the private key corresponding to $K_B$. In other word, this mechanism is implemented in order to authenticate Bob. Sending back to Alice $N_A$ is also a way to avoid a replay of this message.

3. $A \rightarrow A : \{N_B\}_{K_B}$ (Response)

Alice decrypts the message and check if it contains the right value of $N_A$. Then, she sends back $N_B$ to Bob to prove her ability to decrypt with her private key, and so to authenticate herself.

**Review Questions**

1. *Explain digital signature standard.*
2. *Explain digital signature algorithm.*
3. *Explain operation of DES algorithm in detail.*

❑❑❑