CHAPTER 3

ASYMMETRIC CIPHERS

CHAPTER OUTLINE

After studying this chapter, the students will be able to understand the

- Number Theory: Prime Numbers, Fermat's Theorem, Euler's Theorem, Primility Testing, Miller-Rabin Algorithm, Extended Euclidean Theorem, Discrete Logarithms
- Public Key Cryptosystems, Applications of Public Key Cryptosystems
- Distribution of public key, Distribution of secret key by using public key cryptography, Diffie-Helman Key Exchange, Man-in-the-Middle Attack
- RSA Algorithm
- Elgamal Cryptographic System

BASIC NUMBER THEORY

PRIME NUMBERS

An integer p > 1 is a prime number if and only if its only divisors are ± 1 and ±p. Examples are 7 ,13...

Any integer a > 1 can be factored in a unique way as:

$A = {}_{p1}a1_{.p2}a2 \ldots\ldots\ldots\ldots {}_{pt}at$ where p1 < p2 < ... < pt are prime numbers and where each is a positive integer.

This is known as the fundamental theorem of arithmetic. Examples are

91 = 7 x 13 (factorization)

3600 = 24 x 32 x 52

11011 = 7 x 112 x 13

FERMAT'S THEOREM

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then $a^{p-1}=1 \pmod p$

(here a and p are relatively prime)

Example

a = 7, p = 19
$7^2 = 49 \quad 11 \pmod{19}$
$7^4 = 7^2 \times 7^2 \quad 11 \times 11 = 121 \quad 7 \pmod{19}$
$7^8 \quad 49 \quad 11 \pmod{19}$
$7^{16} = 121 = 7 \pmod{19}$
$a^{p1} = 7^{18} = 7^{16} \times 7^2 \quad 7 \times 11 \quad 1 \pmod{19}$

Alternative form of fermat theorem is $a^P = a \pmod p$

EULER'S TOTIENT FUNCTION

Before presenting Euler's theorem, we need to introduce an important quantity in number theory, referred to as Euler's totient function and written $\phi(n)$, defined as the number of positive integers less than n and relatively prime to n. By convention, $\phi(1) = 1$.

n	ϕ(n)
1	1
3	2
13	12
14	6
15	8
19	18
20	8

If n is prime number then ϕ(n)=n-1.

EULER'S THEOREM

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

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

Examples

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

An alternative form of Euler's theorem is

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

This does not require a be relatively prime to n. But n should be expressable in terms of multiple of unique prime numbers.

TESTING FOR PRIMALITY

For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random. Thus we are faced with the task of determining whether a given large number is prime. There is no simple yet efficient means of accomplishing this task.

MILLER-RABIN ALGORITHM

The algorithm due to Miller and Rabin is typically used to test a large number for primality. Before explaining the algorithm, we need some background.

First, any positive odd integer n>= 3 can be expressed as follows:

$$n\,1 = 2^k q \text{ with } k > 0, q \text{ odd}$$

Two Properties of Prime Numbers

The first property is stated as follows:

If p is prime and a is a positive integer less than p, then $a^2 \bmod p = 1$ if and only if either a mod p = 1 or a mod p = 1 mode p = p-1. By the rules of modular arithmetic (a mode p) (a mode p) = a^2 mod p. Thus if either a mode p = 1 or a mod p = 1, then a^2 mod p = 1. Conversely, if $a^2 \bmod p = 1$, then $(a \bmod p)^2 = 1$, which is true only for a mod p = 1 or a mod p = 1.

The second property is stated as follows:

Let p be a prime number greater than 2. We can then write $p\ 1 = 2^k q$, with k > 0 q odd. Let a be any integer in the range 1 < a < p 1. Then one of the two following conditions is true:

1. a^q is congruent to 1 modulo p. That is, $a^q \bmod p = 1$, or equivalently, $a^q 1 \pmod p$.
2. One of the numbers a^q, a^{2q}, a^{4q},..., q is congruent to 1 modulo p. That is, there is some = number j in the range (1 < = j < k) such that $a^{2j-1q} \bmod p = 1 \bmod p = p-1$, or equivalently, $a^{2j-1q} 1 \pmod p$.

Conclusion

If n is prime , then either the first element in the list of residues (a^q, a^{2q}, …, $a^{2^{k-1}q}$, $a^{2^k q}$) modulo n equals 1, or some elements in the list equals (n – 1), otherwise n is composite.

But it is the condition is met, it does not guarantee that the number is prime.

Example: $n = 2047 = 23 \times 89$

Now, n – 1 = 2046

From 2nd property,

$p - 1 = 2^k.q$

$2046 = 2^1 \times 1023$

Now, let a = 2 (a can be 1 < a < p – 1)

$2^{1023} \bmod 2047 = 1$ so according to second property, 2047 is prime number but is it snot.

This property can be used to develop primality testing algorithm.

Details of Miller Rabin algorithm

These considerations lead to the conclusion that if n is prime, then either the first element in the list of residues, or remainders, (aq, a^2q,..., , q, , q) modulo n equals 1, or some element in the list equals (n-1); otherwise n is composite (i.e., not a prime). On the other hand, if the condition is met, that does not necessarily mean that n is prime.

For example, if n = 2047 = 23 x 89, then n 1 = 2 x 1023. Computing, 21023 mod 2047 = 1, so that 2047 meets the condition but is not prime.

We can use the preceding property to devise a test for primality. The procedure TEST takes a candidate integer n as input and returns the result composite if n is definitely not a prime, and the result inconclusive if n may or may not be a prime.

TEST (n)

1. Find integers k, q, with k > 0, q odd, so that (n-1 = 2kq);
2. Select a random integer a, 1 < a < n-1;
3. If a_q mod n = 1 then return("inconclusive");
4. for j = 0 to k 1 do
5. If a_{2jq} mod n n 1 then return("inconclusive");
6. return("composite");

Example: *Test whether n = 221 is prime or not.*

If n is a prime number,

$(n - 1) = 2^k q$

$220 = 2^2 (55)$

So, k = 2 & q = 55

Now, choose a random integers $1 < a < n - 1$

Let a = 5

So, $a^q \bmod n = 5^{55} \bmod 221 = 112 \pmod{221}$

Now, let j = 1

So, $a^{2^1 q} \bmod n = a^{2 \times 55} \bmod 221$

$= 5^{2 \times 55} \bmod 221$

$= 168$

Now, the possible values of j were 0 & 1 (j = 0 to k – 1)

return "compsite".

Example: *n = 29*

If n is prime,

$(n - 1) = 28 = 2^2 \times 7$

So, k = 2, & q = 7

Now, let us choose a = 10

So, $a^q \bmod n = (10)^7 \bmod 29 = 17$ (which is $\neq 1 \bmod n$ or $-1 \bmod n$)

Now, $a^{2^j q} \bmod n = (10)^{2 \times 7} \bmod 29 = 28$ (i.e. n – 1) (i.e. – 1 mod n)

So, the result returns inconclusive i.e. the number may be prime.

DISCRETE LOGARITHMS

Powers of an integer, modulo n.

From Euler's theorem, for a and n that are relatively prim,

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

More generally,

If a and n are relatively prime then there is at least one integer m that satisfies following equation,

$$a^m \equiv 1 \pmod{n}$$

The value that satisfies is $m = \phi(n)$

If we consider all powers of a modulo 19, for all $a < 19$, we obtain a sequence of numbers.

If such sequences contains all numbers $1 < n < n - 1$ (i.e. between 1 and 18, it is said that base integer (via powers) the set of non zero integers modulo 19. Each such integer is called primitive root.

So, if 'a' is primitive root of n, then its powers,

$a, a^2, a^3, \ldots, a^{\phi(n)}$

are distinct (mod n) and all are relatively prime to n.

In case of prime number p, if a is primitive root of p, then a (mod p), a^2 (mod p), a^3 (mod p) ... a^{p-1} (mod p) are all distinct between 1 and p.

For e.g. if p = 19, then its primitive roots are 2, 3, 10, 13, 14, 15

All integers do not have a primitive number. Only the integers with primitive roots are those y from 2, 4, p^{α}, $2p^{qa}$ when p is any odd prime and α is a positive integer.

LOGARITHMS FOR MODULAR ARITHMETIC

The logarithm of a number is defined as power to which some positive base (except 1) must be raised in order to equal the numbers.

Properties:

$\log_x (1) = 0$

$\log_x (x) = 1$

$\log_x (yz) = \log_x (y) + (\log_x(z)$

$\log_x (y^r$

For any prime numbers p and a primitive root 'a' of 'p', we can find an unique exponent I such that

$B = a^i \bmod p$ when $1 \leq 1 \leq (p - 1)$

The exponent I is respond to as discrete logarithm of B for the base 'a mod p'.

$I = d \log_{a, p} (B)$

If we know value of a, I and p, the value of B can easily be computed.

But upon knowing B, a and p it is very difficult to compute i. It is considered to be as difficult as prime factorization. The problem is called discrete logarithm problem.

PUBLIC-KEY CRYPTOSYSTEMS

Sometimes referred to as asymmetric cryptography, public key cryptography is a class of cryptographic protocols based on algorithms. This method of cryptography requires two separate keys, one that is private or secret, and one that is public. Public key cryptography uses a pair of keys to encrypt and decrypt data to protect it against unauthorized access or use. Network users receive a public and private key pair from certification authorities. If other users want to encrypt data, they get the intended recipient's public key from a public directory. This key is used to encrypt the message, and to send it to the recipient. When the message arrives, the recipient decrypts it using a private key, to which no one else has access.

A public key cryptosystem must meet the following three conditions.

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.
4. The first cipher we consider to meet these requirements generates a shared session key. The second one provides both secrecy and authentication.

A public-key encryption scheme has six ingredients:

1. **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
2. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
3. **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
4. **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
5. **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

CHALLENGES OF PUBLIC KEY CRYPTOGRAPHY

Several private key cryptography methods are a great deal faster than the public key encryption method that currently is available. One way of overcoming this challenge with public key cryptography is to combine it with secret key systems to offer the security advantages of the public key system and the speed of the secret (private) key system.

Another challenge associated with public key cryptography is that it has been susceptible to attacks through spoofed or compromised certification authorities. When these attacks take place, cyber criminals impersonate nearly anyone by choosing a public key certificate from the compromised authority. This allows cyber criminals to connect a public key to the name of another user.

BENEFITS OF PUBLIC KEY CRYPTOGRAPHY

The increased data security provided by public key cryptography is its main benefit. Public key cryptography remains the most secure protocol (over private key cryptography) because users never need to transmit or reveal their private keys to anyone, which lessens the chances of cyber criminals discovering an individual's secret key during the transmission.

Public key cryptography also provides digital signatures that cannot be repudiated. Public key cryptography requires each user to be responsible for protecting his private key, whereas private key systems require users to share secret keys and perhaps even trust third parties for transmission. With the secret key system, it is possible for senders to claim the shared secret key was compromised by one of the parties involved in the process.

PUBLIC-KEY CRYPTOSYSTEM FOR SECRECY

There is some source A that produces a message in plaintext, $X = [X_1, X_2, ..., X_M,]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, PU_b, and a private key, PU_b. PU_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

With the message X and the encryption key PUb as input, A forms the ciphertext $Y = [Y_1, Y_2, ..., YN]$:

$Y = E(PU_b, X)$

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

$X = D(PR_b, Y)$

PUBLIC-KEY CRYPTOSYSTEM: AUTHENTICATION

In this case, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a digital signature. In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

Encryption to provide authentication:

$= E(PR_a, X)$ $Y = E(PU_a, Y)$

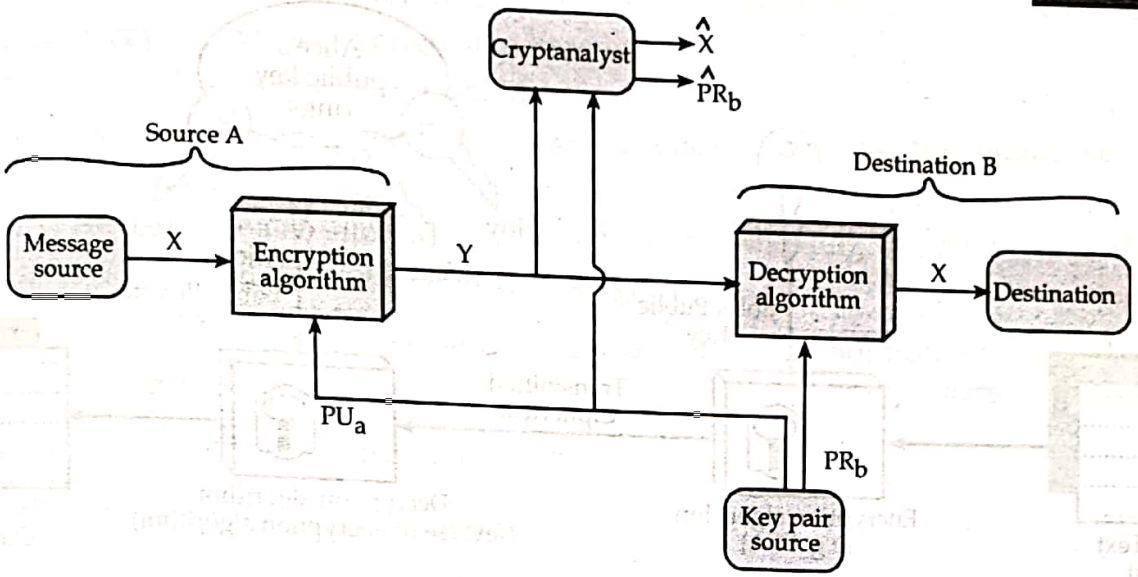


Figure 3.1: Public-key Cryptosystem: Secrecy

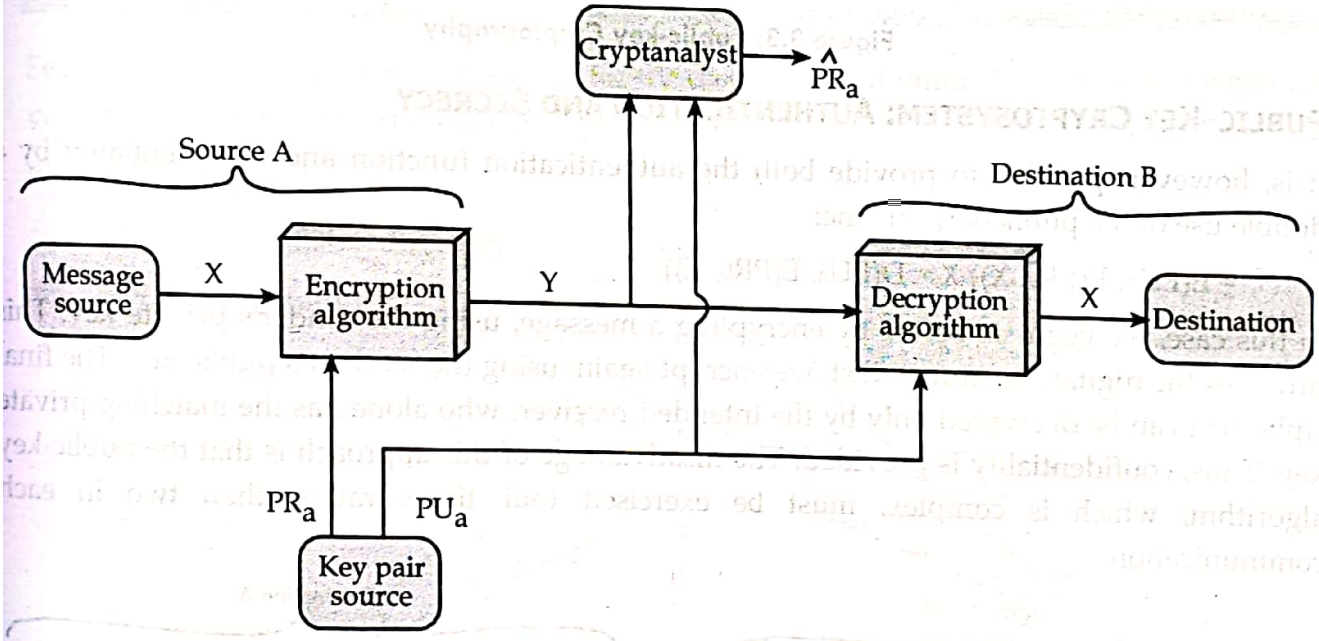
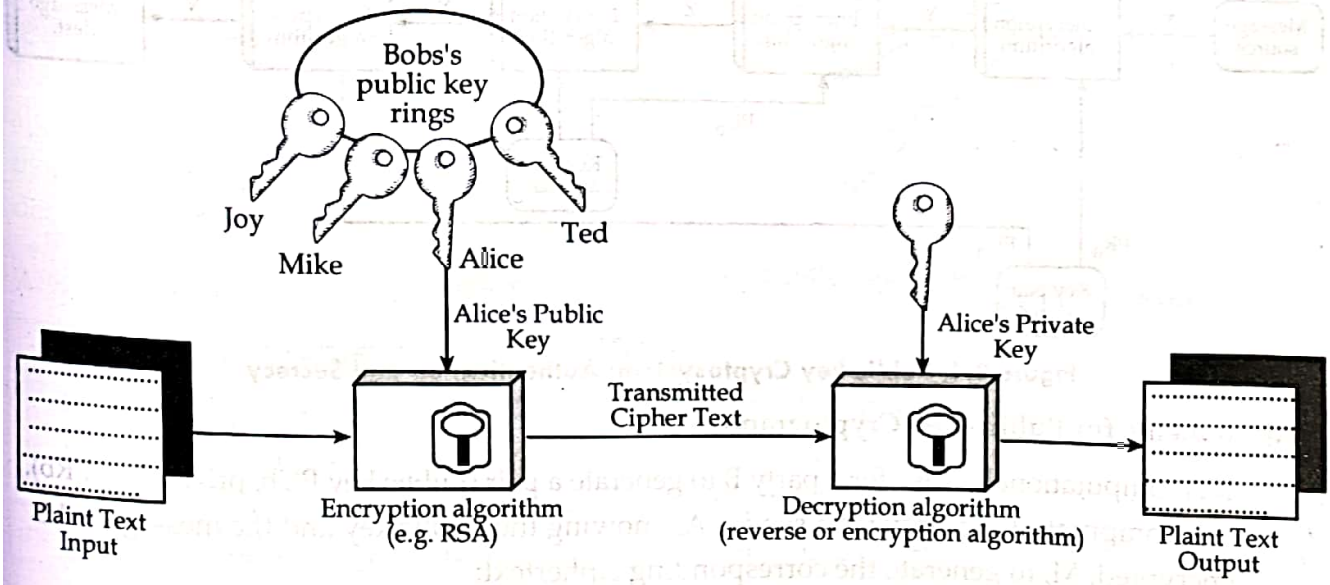
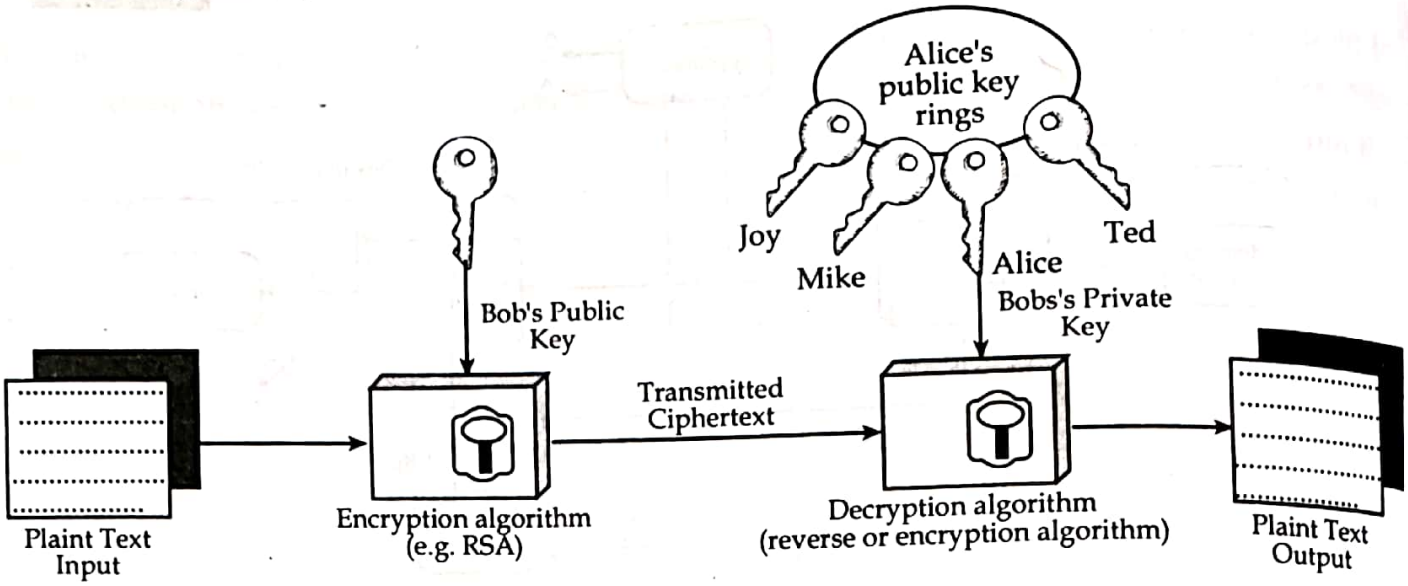


Figure 3.2: Public-key Cryptosystem: Authentication



(a) Encryption



(b)Authentication

Figure 3.3: Public-key Cryptography

PUBLIC-KEY CRYPTOSYSTEM: AUTHENTICATION AND SECRECY

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme:

$$= E(PU_b, E(PR_a, X))\ X = D(PU_a, E(PR_b, Z))$$

In this case, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided. The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

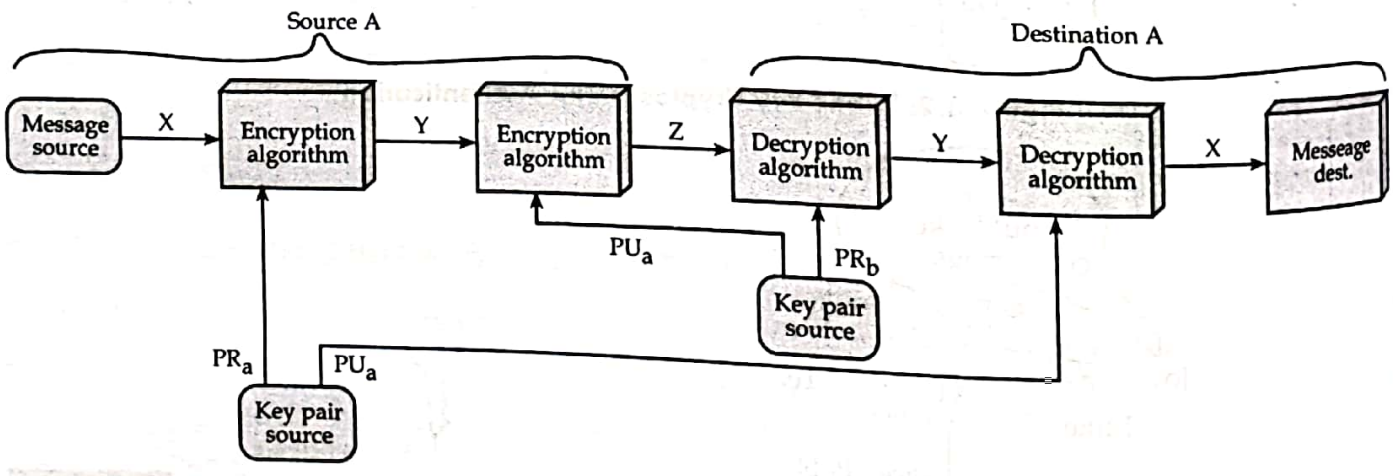


Figure 3.4: Public-key Cryptosystem: Authentication and Secrecy

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key PUb, private key PRb).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

   C = E(PUb, 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(PRb, C) = D[PRb, E(PUb, M)]
4. It is computationally infeasible for an adversary, knowing the public key, PUb, to determine the private key, PRb.
5. It is computationally infeasible for an adversary, knowing the public key, PUb, and a ciphertext, C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The two keys can be applied in either order:
   M = D[PUb, E(PRb, M)] = D[PRb, E(PUb, M)]

DISTRIBUTION OF PUBLIC KEYS

Several techniques have been proposed for the distribution of public keys. All these techniques can be grouped into the following general schemes:

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

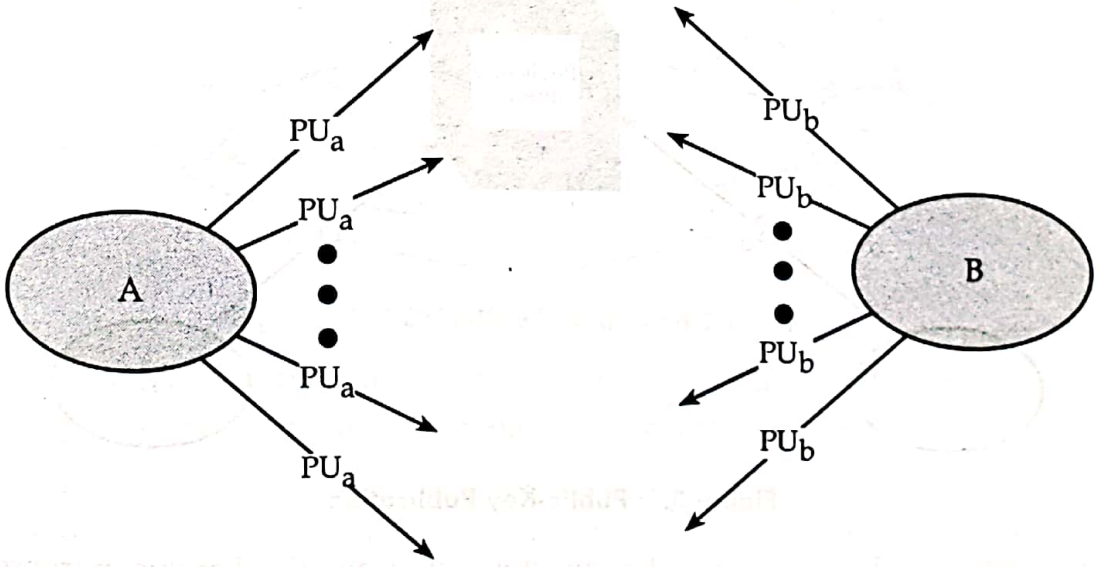


Figure 3.5: Uncontrolled Public Key control

PUBLIC ANNOUNCEMENT OF PUBLIC KEYS

On the face of it, the point of public key encryption is that the public key is public. Thus, if there is some broadly accepted public key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the keyto the community at large For exa mple, because of the growing popularity of PGP (pretty good privacy, 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 newsgroups and Internet mailing lists.

Although this approach is convenient, it has a major weakness. 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. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication

PUBLICLY AVAILABLE DIRECTORY

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public direc- tory would have to be the responsibility of some trusted entity or organization. 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 authenti- cated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been com- promised in some way.

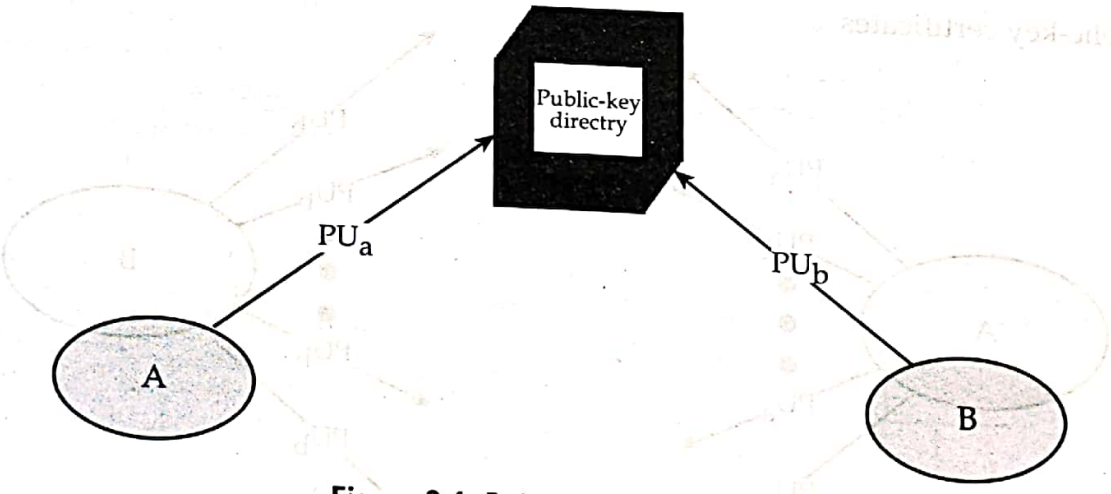


Figure 3.6: Public-Key Publication

4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

This scheme is clearly more secure than individual public announcements but still has vulnerab *ilities. If* an adversary succeeds in obtaining or computing the private key of the directory *authority,* the adversary could authoritatively pass out counterfeit public keys and *subsequently* impersonate any participant and eaves- drop on messages sent to any participant. *Another* way to achieve the same end is for the adversary to tamper with the records kept by *the authority.*

PUBLIC-KEY AUTHORITY

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

1. A sends a time stamped 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, PRauth. Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority.The message includes the following:
   - B's public key, PUb, which A can use to encrypt messages destined for B
   - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority

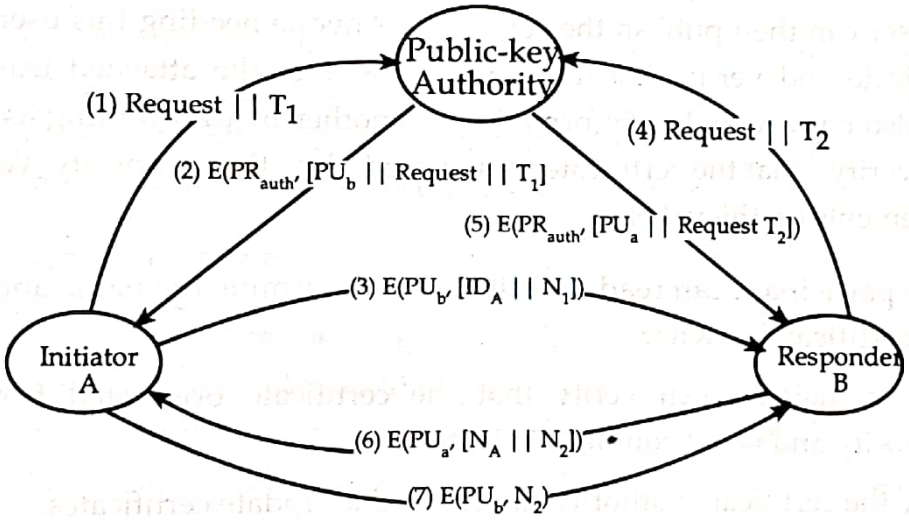


Figure 3.7: Public-Key Distribution Scenario

   - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), 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. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:
6. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B(N_2). Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.
7. A returns N_2, which is encrypted using B's public key, to assure B that its correspondent is A.

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use—a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

PUBLIC-KEY CERTIFICATES

The public-key authority could be some what of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

An alternative approach, is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party. Typically, the third party is a certificate authority, such as a 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. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

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

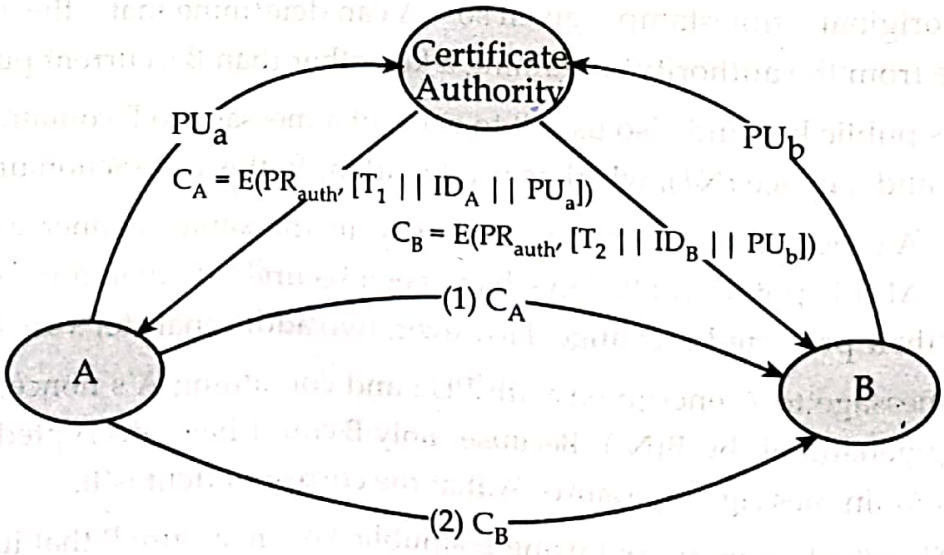


Figure 3.8: Exchange of Public-Key Certificates

Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

CA = E(PRauth, [T || IDA || PUa])

where PRauth is the private key used by the authority and T is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

D(PUauth, CA) = D(PUauth, E(PRauth, [T || IDA || PUa])) = (T || IDA || PUa)

The recipient uses the authority's public key, PUauth, to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements IDA and PUa provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired.

DIFFIE-HELLMAN KEY EXCHANGE

Diffie-Hellman (D-H) key exchange is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. Other names for Diffie-Hellman Key Exhange are *Diffie-Hellman Key Agreement, Diffie-Hellman Key Establishment, Diffie- Hellman Key Negotiation, Exponential Key Exchange.*

Description: The simplest and original implementation of the protocol uses the multiplicative group of integers modulo p, where p is a prime and g is primitive root of p. **Steps:**

1. Generate the global public elements p and g, where p is a prime number and $g < p$ is a primitive root of p.
2. User A selects a random integer number $X_A < p$, and computes $Y_A = g^{X_A} \bmod p$.
3. User B independently selects a random integer $X_B < p$, and computes $Y_B = g^{X_B} \bmod p$.
4. Each side keeps the X value private and makes the Y value available publicly to the other side.
5. User A generates secret key as $K = (Y_B)^{X_A} \bmod p$.
6. User B generates secret key as $K = (Y_A)^{X_B} \bmod p$

Why the key from both side same:

From user A, $K = (Y_B)^{X_A} \bmod p = (g^{X_B} \bmod p)^{X_A} \bmod p = (g^{X_B})^{X_A} \bmod p = g^{X_B X_A} \bmod p$

From user B, $K = (Y_A)^{X_B} \bmod p = (g^{X_A} \bmod p)^{X_B} \bmod p = (g^{X_A})^{X_B} \bmod p = g^{X_B X_A} \bmod p$

Above both the results are same.

Example: Alice and Bob agree to use a prime number p = 23 and base g=5.

Alice chooses a secret integer $X_A = 6$, then sends Bob ($Y_A = g^{X_A} \bmod p$): $5^6 \bmod 23 = 8$. Bob chooses a secret integer $X_B = 15$, then sends Alice ($Y_B = g^{X_B} \bmod p$): $5^{15} \bmod 23 = 19$.

Alice computes $(Y_B)^{X_A} \bmod p$: $19^6 \bmod 23 = 2$ and Bob computes $(Y_A)^{X_B} \bmod p$: $8^{15} \bmod 23 = 2$.

Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel. Of course, much larger values of X_A, X_B, and p would be needed to make this example secure, since it is easy to try all the possible values of g A B mod 23 (there will be, at most, 22 such values, even if X_A, X_B are large). If p were a prime of at least 300 digits, and X_A, X_B were at least 100 digits long, then even the best algorithms known today could not find a given only g, p, and $g^{X_A} \bmod p$, even using all of mankind's computing power. The problem is known as the discrete logarithm problem. *Note that g need not be large at all, and in practice is usually either 2 or 5.*

SECURITY: THE BUCKET BRIGADE/MAN IN THE MIDDLE ATTACK

In the original description, the Diffie-Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the- middle attack. A person in the middle may establish two distinct Diffie-Hellman key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing the attacker to decrypt (and read or store) then re-encrypt the messages passed between them. A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack.

RSA (RIVEST SHAMIR ADLEMAN)

RSA is an algorithm for public-key cryptography. It was the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations.

Operation: RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key.

RSA KEY GENERATION

1. Choose two distinct large random prime numbers p and q
2. Compute n = pq, n is used as the modulus for both the public and private keys
3. Compute the totient: $\varphi(n) = (p - 1)(q - 1)$.
4. Choose an integer e such that $1 < e < \varphi(n)$, and e and $\varphi(n)$ share no factors other than 1 i.e. e and $\varphi(n)$ are relatively prime)
5. e is released as the public key exponent
6. Compute d to satisfy the congruence relation $ed \equiv 1 \bmod \varphi(n)$; i.e. $de = 1 + k\varphi(n)$ for some integer k.
7. d is kept as the private key exponent

Notes on the above steps: Step 1: Numbers can be probabilistically tested for primality. Step 4: A popular choice for the public exponents is $e = 2^{16} + 1 = 65537$. Some applications choose smaller values such as e = 3, 5, 17 or 257 instead. This is done to make encryption and signature verification faster.

Steps 4 and 5 can be performed with the extended Euclidean algorithm; See Appendix for mathematical background.

ENCRYPTING MESSAGES

Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob send message M to Alice by turning M into a number m < n by using a reversible protocol called a padding scheme. He then computes the ciphertext c as: $c = m^e$ **mod** n. Bob then transmits c to Alice.

DECRYPTING MESSAGES

Alice can recover m from c by using her private key exponent d by the following computation: $m = c^d$ **mod** n. Given m, she can recover the original message M.

Example: Consider, p = 61 and q = 53 now, compute n = pq = 61 * 53 = 3233 Compute the totient $\varphi(n) = (p - 1)(q - 1) = (61-1)(53-1) = 3120$

Choose e > 1 relatively prime to 3120; e = 17

Compute d such that $ed \equiv 1 \bmod \varphi(n)$ e.g., by computing the modular multiplicative inverse of e modulo $\varphi(n)$: d = 2753 since 17 * 2753 = 46801 = 1 + 15 * 3120.

The public key is (n = 3233, e = 17).

For a padded message m the encryption function is: $c = m^e$ **mod** $n = m^{17}$ **mod** 3233.

The private key is (n = 3233, d = 2753). The decryption function is: $m = c^d$ **mod** $n = c^{2753}$ **mod** 3233.

For example, to encrypt m = 123, we calculate $c = 123^{17} \bmod 3233 = 855$ to decrypt c = 855, we calculate $m = 855^{2753} \bmod 3233 = 123$ Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation.

One More Example:

Consider primes p = 11, q = 3. Now, compute n = pq = 11.3 = 33 and totient φ(n)

= (p – 1) (q – 1)

= 10.2 = 20

Choose e=3; Check gcd(e, φ(n)) = gcd(3, 20) = 1 (i.e. 3 and 20 have no common factors except 1),

Compute d such that $ed \equiv 1 \pmod{\varphi(n)}$

i.e. find a value for d such that φ(n) divides (ed-1)

i.e. find d such that 20 divides 3d-1. Simple testing (d = 1, 2, ...) gives d = 7

Check: ed-1 = 3.7 - 1 = 20, which is divisible by φ(n)

Public key = (n, e) = (33, 3) Private key = (n, d) = (33, 7).

> The notation 'a ≡ b (mod n)' means a and b have the same remainder when divided by n, or, equivalently,
>
> a – b = nk for some integer k

This is actually the smallest possible value for the modulus n for which the RSA algorithm works.

Now say we want to encrypt the message m = 7, $c = m^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33 = 13$. Hence the ciphertext c = 13.

To check decryption we compute $m' = c^d \bmod n = 13^7 \bmod 33 = 7$.

Example

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

- Let two primes be p = 7 and q = 13. Thus, modulus n = pq = 7 x 13 = 91.
- Select e = 5, which is a valid choice since there is no number that is common factor of 5 and (p – 1)(q – 1) = 6 × 12 = 72, except for 1.
- The pair of numbers (n, e) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input p = 7, q = 13, and e = 5 to the Extended Euclidean Algorithm. The output will be d = 29.
- Check that the d calculated is correct by computing – $de = 29 \times 5 = 145 = 1 \bmod 72$
- Hence, public key is (91, 5) and private keys is (91, 29).

ENCRYPTION AND DECRYPTION

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

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

RSA Encryption

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

$C = P^e \bmod n$

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

$C = 105 \bmod 91$

RSA Decryption

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

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

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

$\text{Plaintext} = 8229 \bmod 91 = 10$

RSA ANALYSIS

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

- **Encryption Function:** It is considered as a one-way function of converting plaintext into ciphertext and it can be reversed only with the knowledge of private key d.
- **Key Generation:** The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n. It is also a one way function, going from p & q values to modulus n is easy but reverse is not possible.

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

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

ELGAMAL CRYPTOSYSTEM

Along with RSA, there are other public-key cryptosystems proposed. Many of them are based on different versions of the Discrete Logarithm Problem.

ElGamal cryptosystem, called Elliptic Curve Variant, is based on the Discrete Logarithm Problem. It derives the strength from the assumption that the discrete logarithms cannot be found in practical time frame for a given number, while the inverse operation of the power can be computed efficiently.

Let us go through a simple version of ElGamal that works with numbers modulo p. In the case of elliptic curve variants, it is based on quite different number systems.

Generation of ElGamal Key Pair

Each user of ElGamal cryptosystem generates the key pair through as follows:

- Choosing a large prime p. Generally a prime number of 1024 to 2048 bits length is chosen.
- Choosing a generator element g.
  - This number must be between 1 and p – 1, but cannot be any number.
  - It is a generator of the multiplicative group of integers modulo p. This means for every integer m co-prime to p, there is an integer k such that gk=a mod n.

    For example, 3 is generator of group 5 (Z5 = {1, 2, 3, 4}).

N	3n	3n mod 5
1	3	3
2	9	4
3	27	2
4	81	1

- Choosing the private key. The private key x is any number bigger than 1 and smaller than p−1.
- Computing part of the public key. The value y is computed from the parameters p, g and the private key x as follows:

  $y = g^x \bmod p$
- Obtaining Public key. The ElGamal public key consists of the three parameters (p, g, y).

  For example, suppose that p = 17 and that g = 6 (It can be confirmed that 6 is a generator of group Z17). The private key x can be any number bigger than 1 and smaller than 71, so we choose x = 5. The value y is then computed as follows:

  $y = 65 \bmod 17 = 7$
- Thus the private key is 62 and the public key is (17, 6, 7).

Encryption and Decryption

The generation of an ElGamal key pair is comparatively simpler than the equivalent process for RSA. But the encryption and decryption are slightly more complex than RSA.

ElGamal Encryption

Suppose sender wishes to send a plaintext to someone whose ElGamal public key is (p, g, y), then-

- Sender represents the plaintext as a series of numbers modulo p.
- To encrypt the first plaintext P, which is represented as a number modulo p. The encryption process to obtain the ciphertext C is as follows –
  - Randomly generate a number k;
  - Compute two values C1 and C2, where:

    $C1 = g^k \bmod p$

    $C2 = (P * y^k) \bmod p$
- Send the ciphertext C, consisting of the two separate values (C1, C2), sent together.
- Referring to our ElGamal key generation example given above, the plaintext P = 13 is encrypted as follows:
  - Randomly generate a number, say k = 10
  - Compute the two values C1 and C2, where –

    $C1 = 610 \bmod 17$

    $C2 = (13 * 710) \bmod 17 = 9$
- Send the ciphertext C = (C1, C2) = (15, 9).

ElGamal Decryption

- To decrypt the ciphertext (C1, C2) using private key x, the following two steps are taken:
  - Compute the modular inverse of (C1)x modulo p, which is (C1)-x , generally referred to as decryption factor.
  - Obtain the plaintext by using the following formula –

    $C2 \times (C1)^{-x} \bmod p = \text{Plaintext}$
- In our example, to decrypt the ciphertext C = (C1, C2) = (15, 9) using private key x = 5, the decryption factor is

  $15^{-5} \bmod 17 = 9$
- Extract plaintext $P = (9 \times 9) \bmod 17 = 13$.

ELGAMAL ANALYSIS

In ElGamal system, each user has a private key x. and has three components of public key – prime modulus p, generator g, and public $Y = g^x \bmod p$. The strength of the ElGamal is based on the difficulty of discrete logarithm problem.

The secure key size is generally > 1024 bits. Today even 2048 bits long key are used. On the processing speed front, Elgamal is quite slow, it is used mainly for key authentication protocols. Due to higher processing efficiency, Elliptic Curve variants of ElGamal are becoming increasingly popular.

ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

Elliptic Curve Cryptography (ECC) is a term used to describe a suite of cryptographic tools and protocols whose security is based on special versions of the discrete logarithm problem. It does not use numbers modulo p.

ECC is based on sets of numbers that are associated with mathematical objects called elliptic curves. There are rules for adding and computing multiples of these numbers, just as there are for numbers modulo p.

ECC includes a variants of many cryptographic schemes that were initially designed for modular numbers such as ElGamal encryption and Digital Signature Algorithm.

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

The shorter keys result in two benefits –

- Ease of key management
- Efficient computation

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

A COMPARISON OF RSA AND ELGAMAL SCHEMES

Let us briefly compare the RSA and ElGamal schemes on the various aspects.

RSA	ElGamal
It is more efficient for encryption.	It is more efficient for decryption.
It is less efficient for decryption.	It is more efficient for decryption.
For a particular security level, lengthy keys are required in RSA.	For the same level of security, very short keys are required.
It is widely accepted and used.	It is new and not very popular in market.

DISCUSSION EXERCISE

1. What is Euler's theorem? What is the totient of a prime number?
2. Describe RSA Algorithm and Estimate the encryption and decryption values for the RSA algorithm parameters.
3. In RSA algorithm, what is necessary condition that must be satisfied by the modulus n chosen for the generation of the public and private key pair? Also, is the modulus made public?
4. In a RSA system, a user has chosen the primes 5 and 19 to create a key pair. The public key is {e=5, n=?} and the private key is {d=?, n=?}. Decide the private key {d, n}. Show encryption and decryption process for the message "TOGA"
5. In a RSA system, a user has chosen the primes 5 and 19 to create a key pair. The public key is (5, n) and the private key is (d, n). Decide the private key (d, n). Show encryption and decryption process for the message "Drogba".
6. In a RSA system, a user has chosen the prime 5 and 19 to create a key pair. The public key is {e = 5, n = ?} and the private key is {d=?, n=?). Decide the private key {d, n}. Show encryption and decryption process for the message "TA".
7. In a RSA system, a user has chosen the primes 5 and 19 to create a key pair. The public key is (5, n) and the private key is (d, n). Decide the private key (d, n). Show encryption and decryption process for the message "ANT".
8. Consider a Diffie-Hellman scheme with a common prime p = 11 and primitive root g = 2.
   i. Show that 2 is a primitive root of 11.
   ii. If user A has public key $Y_a = 9$, what is A's private key X_a?
   iii. If user B has public key $Y_b = 3$, what is hared key K, shared with A.

9. How do you create public and private keys in the RSA algorithm for public-key cryptography?
10. In public key cryptosystem, each of the communicating parties, in general, should know the public keys of each other before attempting security encryptions. How this can be achieved?
11. What is Fermat's Little Theorem? What is the totient of a number?
12. What is discrete logarithm and when can we define it for a set of numbers?
13. What is the Diffie-Hellman algorithm for exchanging a secretsession key?
14. What do you mean by man-in middle attack? Is man-in-middle attack possible in Deffie-Hellman? How?
15. Miller-Rabin test says that if a candidate integer n is prime, it must satisfy one of two special conditions. What are those two conditions?
16. How Man-In-Middle attack is possible in Deffie-Hellman Algorithm. Support answer with a numerical computation. Chose the required parameters with your own assumptions
17. What do you mean by Elgamal cryptographic system? Explain.

❑❑❑