# Network and Cyber Security

Bachelor in Computer Engineering

Er. Anuj Sherchan

Assistant Professor

# Unit 3: Asymmetric Key Cryptography

- Outline:
- 3.1 Need of Public Key, Asymmetric Cipher Model
- 3.2 Prime Factorization Problem, RSA Encryption
- 3.3 Discrete Logarithmic Problem
- 3.3.1 Diffie Hellman Key
- 3.3.2 ElGamal Encryption
- 3.3.3 Elliptical Curve Cryptography

# Introduction

- **Asymmetric-key Encipherment**
- The asymmetric-key encipherment also called public-key encipherment or public-key cryptography, was introduced by Diffie and Hellman in 1976 to overcome the problem found in symmetric key cryptography.
- It uses two different keys for encryption and decryption.
- These two keys are referred to as the public key (used for encryption) and the private key (used for decryption).
- Each authorized user has a pair of public and private keys.
- The public key of each user is known to everyone, whereas the private key is known to its owner only.

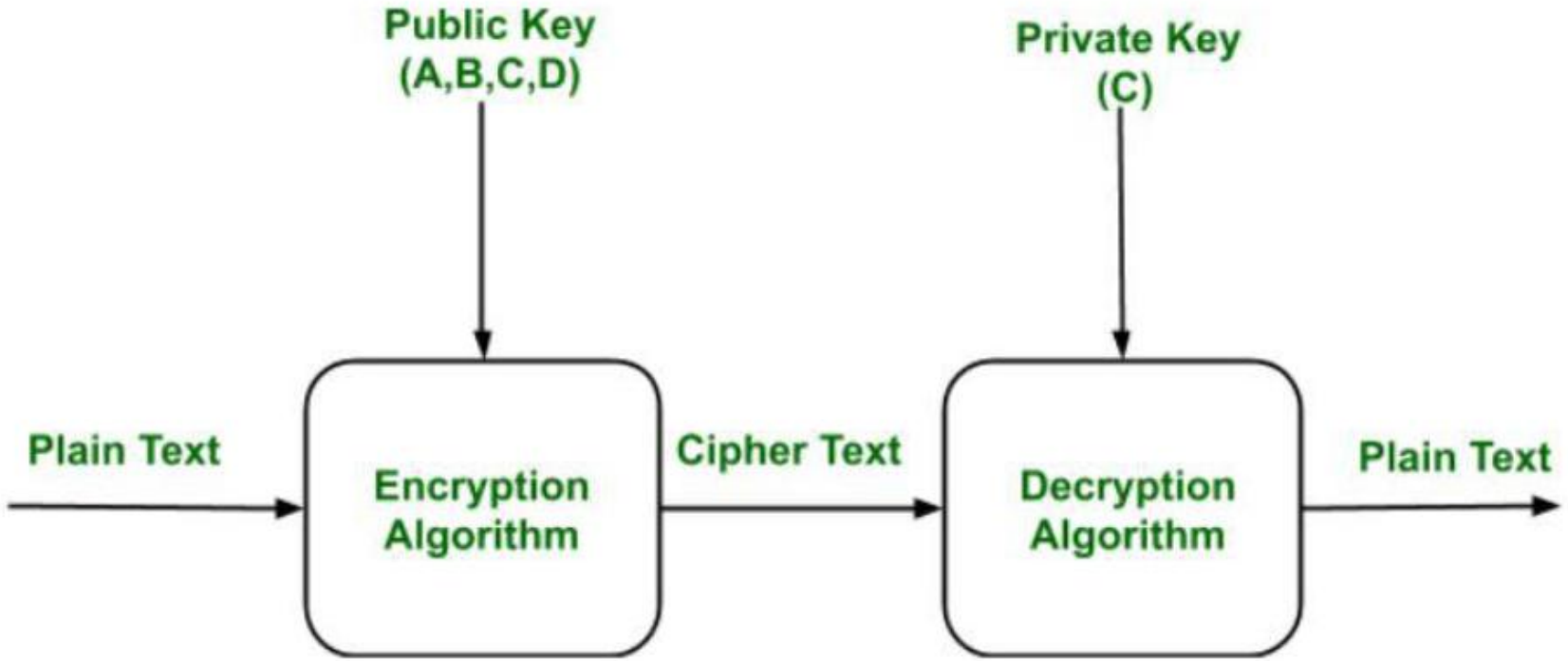
# Need of Public Key ,Asymmetric Cipher Model

- Public key cryptography provides a secure way to exchange information and authenticate users by using pairs of keys.
- The public key is used for encryption and signature verification, while the private key is used for decryption and signing.
- When the two parties communicate with each other to transfer the intelligible or sensible message, referred to as plaintext, is converted into apparently random unreadable for security purposes referred to as ciphertext.
- This system ensures that only the intended recipient can read an encrypted message and that a signed message truly comes from the claimed sender.
- Public key cryptography is essential for secure internet communications, allowing for confidential messaging, authentication of identities, and verification of data integrity.

# Need of Public Key ,Asymmetric Cipher Model

- **Public Key Encryption Working**
- **Key Pair Generation** : A user generates a pair of keys :
- **Public Key**: Shared openly. Anyone can use it to send an encrypted message.
- **Private Key**: Kept secret. Only the key owner can decrypt messages encrypted with the public key.
- **Encryption** : If someone wants to send a private message:
- They obtain the recipient’s public key.
- They encrypt the message using that public key.
- The encrypted message is sent over a network.
- **Decryption** : Upon receiving the message:
- The recipient uses their private key to decrypt the message and recover the original plaintext

# Need of Public Key ,Asymmetric Cipher Model




*Public Key Encryption*

# Need of Public Key ,Asymmetric Cipher Model

- **Public Key Encryption Practical Example: Secure Website (HTTPS)**
- When you visit a secure website like https://www.bank.com, public key encryption is used behind the scenes to encrypt data between your browser and the bank's server.
- **Bank's Server Has a Key Pair**
- **Private Key**: Secret, stored securely on the server.
- **Public Key**: Shared with anyone via an SSL certificate.
- **You Connect to the Website**
- Your browser gets the bank’s public key from its SSL certificate.
- It verifies the certificate is valid (issued by a trusted certificate authority).

# Need of Public Key ,Asymmetric Cipher Model

- **Encrypting the Session Key**
- Your browser creates a random symmetric key (used for actual data encryption).
- It encrypts this key using the bank's public key.
- Only the bank can decrypt it using its private key.
- **Secure Communication Begins**
- Now both your browser and the bank share a secret symmetric key.
- All further communication (login info, account data, etc.) is encrypted using this key.

# Need of Public Key ,Asymmetric Cipher Model

- **Why Public Key Encryption is Used**
- It ensures that only the server (with the private key) can read the symmetric key.
- Even if someone intercepts the traffic, they can’t decrypt the session key or data.
- **Applications of the Public Key Encryption**
- **SSL/TLS protocols** : They use public key encryption to securely exchange symmetric session keys between a web browser and a server.
- **Digital signature:** Digital signature is for senders authentication purpose. In this sender encrypt the plain text using his own private key.
- This step will make sure the authentication of the sender because receiver can decrypt the cipher text using senders public key only.

# Need of Public Key ,Asymmetric Cipher Model

- **Key exchange:** This algorithm can use in both Key-management and securely transmission of data.
- **SSH keys** : For secure login to remote servers use public/private key pairs for authentication.
- **Blockchain and Cryptocurrencies**: Users control wallets with private keys , public keys serve as wallet addresses.

# Prime factorization Problem, RSA Encryption

- **What is the Prime Factorization Problem?**
- It is the challenge of finding the prime numbers that multiply together to give a large integer.
  Example:
- $77 = 7 \times 11$
- This is easy for small numbers, but **extremely hard** when the number is hundreds or thousands of bits long.

# Prime factorization Problem, RSA Encryption

- **Why is it Important in Cryptography?**
- RSA encryption works by:
- Choosing two very large prime numbers p and q
- Multiplying them to get n= p×q
- Publishing n as part of the public key
- Even though anyone knows n, **they cannot easily find p and q**.
- Without p and q, it's practically impossible to compute the private key.

# Prime factorization Problem, RSA Encryption

- **Security Reason**
- Currently, **no efficient algorithm exists** to factor very large numbers (e.g., 2048-bit) on classical computers.
- So RSA's security = difficulty of factoring large composites.
- **Example (Simplified)**
- Public key contains:
- n = 2537
- To break RSA, attacker must find:
- 2537 = 43 × 59  <-- primes
- For tiny numbers this is easy, but **real RSA uses primes with 300+ digits each.**

# Prime factorization Problem, RSA Encryption

- **Threats & Future**
- **Quantum computing** (via Shor’s algorithm) could factor large numbers efficiently.
- This is why **post-quantum cryptography** is being developed.

# Prime factorization Problem, RSA Encryption

- **RSA Algorithm**
- Block cipher asymmetric algorithm developed by Rivest, Shamir & Adleman .
- It is the best known & widely used public-key scheme and based on exponentiation in a finite (Galois) field over integers modulo a prime.
- Its security due to cost of factoring large numbers Factorization takes O(e log n log log n ) operations (hard)
- Each user will be provided with pair of keys one of which is public key used for encryption and the other is private used for decryption.
- Plaintext and cipher text are integers between 0 and n – 1 for some n. (eg . 1024 bits)

# Prime factorization Problem, RSA Encryption

- **Ingredients of RSA Algorithm**
- The ingredients are the following:
- p, q two prime numbers
- n = p*q
- e, with gcd( $\phi$ (n),e) = 1;
- 1 < e < $\phi$
- d≡ e-1(mod $\phi$ (n))

(private, chosen)
(public, calculated)
(public, chosen)
(private, calculated)

# Prime factorization Problem, RSA Encryption

- **Steps for RSA Algorithm:**
- Select two large prime numbers, p and **q**.
- Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.
- Choose a number **e** less than **n**, such that n is relatively prime to **(p - 1) x (q -1).**
- It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1.
- Choose “e” such that 1<e < φ (n), e is prime to φ (n), **gcd (e,d(n)) =1**
- If **n = p x q,** then the public key is <e, n>.
- A plaintext message **m** is encrypted using public key <e, n>.
- To find cipher text from the plain text following formula issued to get cipher text C. **$C = m^e \bmod n$**

# Prime factorization Problem, RSA Encryption

- Here**, m** must be less than **n**.
- A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
- To determine the private key, we use the following formula to calculate the d such that:
- Calculate d=e^-1mod φ (n)
- e d =mod φ (n)
- **ed mod** φ (n)**= 1**
- **d= ((1+k φ (n)/e)**
- The private key is <d, n>.
- A ciphertext message **c** is decrypted using private key <d, n>.
- To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.
  **$m = c^d \bmod n$**

# Prime factorization Problem, RSA Encryption

- Example 1:
- This example shows how we can encrypt plaintext 9 using the RSA public-key encryption algorithm. This example uses prime numbers 7 and 11 to generate the public and private keys.
- **Explanation:**
- **Step 1:** Select two large prime numbers, p, and **q**.
- p = 7
- q = 11
- **Step 2:** Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.

# Prime factorization Problem, RSA Encryption

- First, we calculate **n = p x q** = 7 x 11=77
- **Step 3:** Choose a number **e** less that **n**, such that n is relatively prime to **(p - 1) x (q -1).**
- It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1.
- Choose "e" such that 1<e < φ (n), e is prime to φ (n), gcd (e,d(n)) =1. Second, we calculate **φ (n) = (p - 1) x (q-1)**
- φ (n) = (7 - 1) x (11 - 1)
- φ (n) = 6 x 10=60
- Let us now choose relative prime e of 60 as 7.
- Thus the public key is <e, n> = (7, 77)

# Prime factorization Problem, RSA Encryption

- **Step 4:** A plaintext message **m** is encrypted using public key <e, n>.
- To find ciphertext from the plain text following formula is used to get ciphertext C.
- To find ciphertext from the plain text following formula is used to get ciphertext C.
- **$C = m^e \bmod n$**
- $C = 9^7 \bmod 77$
- C = 37
- **Step 5:** The private key is <d, n>. To determine the private key, we use the following formula d such that:
- **$ed \bmod \varphi(n) = 1$ [d=(1+k φ (n))/e)]**
- 7d mod 60 = 1, which gives d = 43
- The private key is <d, n> = (43, 77)

# Prime factorization Problem, RSA Encryption

- **Step 6:** A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.
- **$m = c^d \bmod n$**
- $m = 37^{43} \bmod 77$
- $m = 9$
- In this example, Plain text = 9 and the ciphertext = 37
- Example 2:
- In an RSA cryptosystem, a particular A uses two prime numbers, 13 and 17, to generate the public and private keys. If the public of A is 35. Then the private key of A is ……?.

# Discrete Logarithmic Problem

- The **Discrete Logarithm Problem (DLP)** is a fundamental problem in cryptography that forms the basis for the security of many public-key systems.
- Given:
- A cyclic group G
- A generator g of the group
- An element h ∈ G
- The discrete logarithm problem is to find an integer x such that:

$$g^x \equiv h \pmod{p}$$

- We call x the **discrete logarithm** of h to the base g:

$$x = \log_g(h)$$

- This is analogous to ordinary logarithms, but instead of real numbers, it works in modular arithmetic or other algebraic groups.

# Discrete Logarithmic Problem

- **Why is it important?**
- The DLP is considered **computationally hard** when:
- p is a large prime (e.g., 2048-bit)
- g generates a large cyclic group
- **Why is it hard?**
- There is **no known efficient classical algorithm** to solve DLP for large, properly chosen groups.

# Discrete Logarithmic Problem

**Example (small numbers)**

Solve:

$$2^x \equiv 5 \pmod{11}$$

Try values:

| x | $2^x \bmod 11$ |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 5 ✅ |

So,

$x=4$

# Diffie Hellman Key

- **Diffie-Hellman Key Exchange** is a technique of securely transferring cryptographic keys over a public channel, devised by Ralph Merkle and named after Whitfield Diffie and Martin Hellman, and was one of the earliest public-key protocols.
- **Diffie-Hellman Key Exchange** is one of the earliest practical examples of public key exchange.
- This is the oldest publicly known paper that proposed the idea of a private key and a corresponding public key.
- It was published in 1976 by Diffie and Hellman.
- The point is to agree on a key that two parties can use for symbolic encryption in such a way that eavesdropper cannot obtain the key.

# Diffie Hellman Key

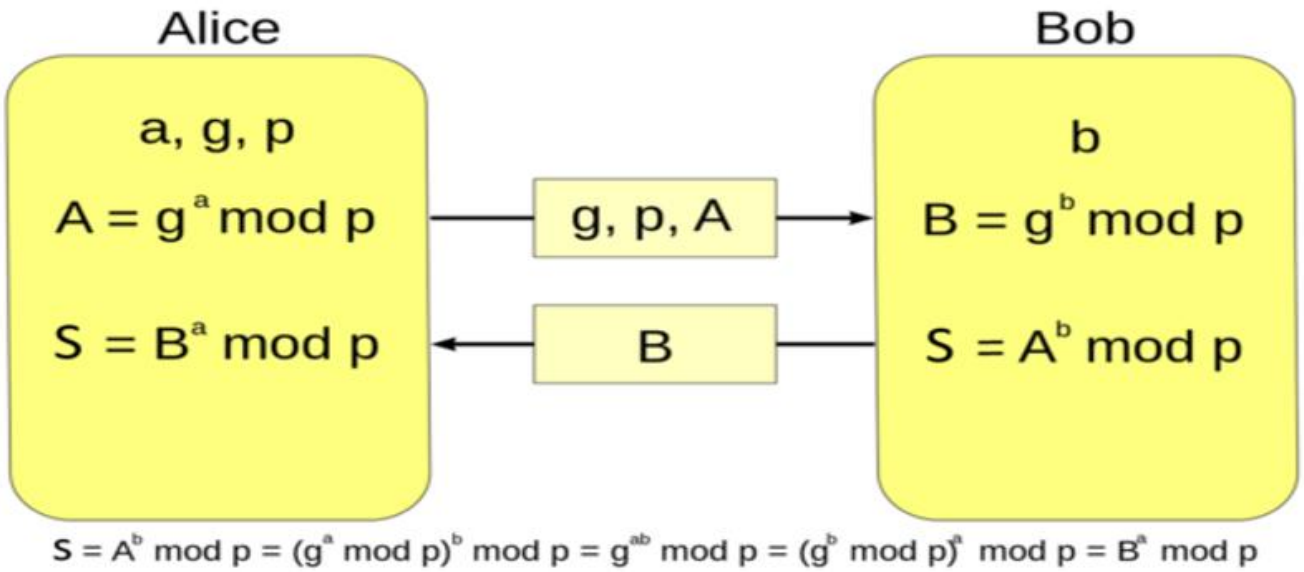



**Fig : Diffie Hellman Key Exchange Algorithm**

# Diffie Hellman Key

- Steps
- Alice and Bob agree on prime number p and a base g.
- Alice chooses a secret number a , and sends Bob (g^a mod p).
- Bob chooses a secret number b and sends Alice (g^b mod p).
- Alice computes ((g^b mod p)^a mod p)
- Bob computes ((g^a mod p)^b  mod p)
- Both Alice and Bob can use this number as their key.
- Notice that p and g need not be protected

# Diffie Hellman Key

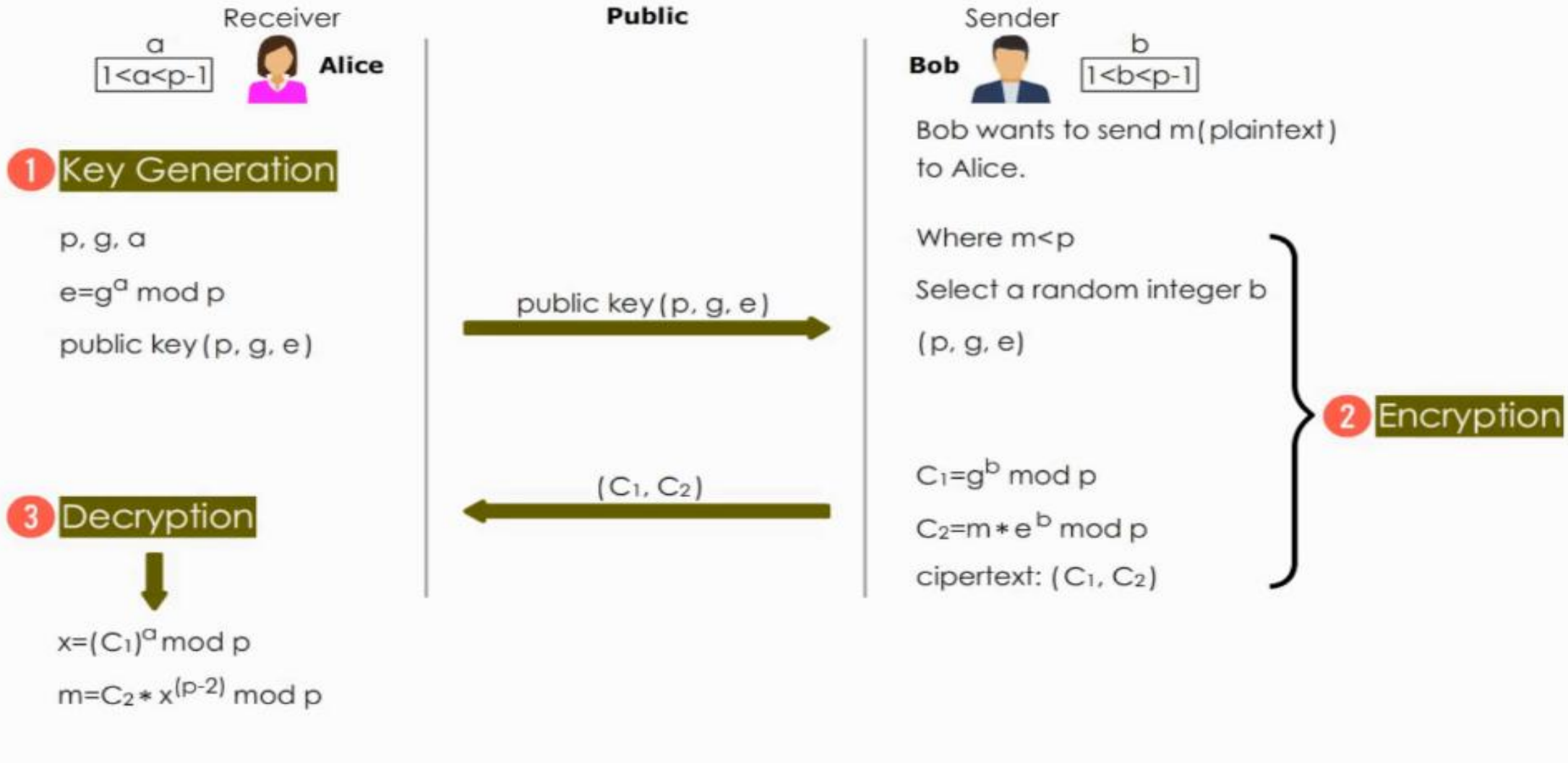
- Example
- Alice and Bob agree on  p=23 and g=5
- Alice chooses a=6 and sends 5^6 mod 23=8
- Bob chooses b=15 and sends 5^15 mod 23=19
- Alice computes 19^6 mod 23 =2
- Bob computes 8^15 mod 23=2
- Then 2 is the shared secret
- Clearly , much larger values of a ,b , p are required .
- An eaves dropper cannot discover this value even if she knows p and g and can obtain each of the messages.

# ElGamal Encryption

- **ElGamal Encryption** is a public-key cryptosystem.
- It uses asymmetric key encryption to communicate between two parties and encrypt the message.
- This cryptosystem is based on the difficulty of finding **discrete logarithms** in a cyclic group that is even if we know $g^a$ and $g^k$, it is extremely difficult to compute $g^{ak}$
- The ElGamal cryptographic algorithm is an asymmetric key encryption scheme based on the Diffie-Hellman key exchange.
- It was invented by Taher ElGamal in 1985.
- The algorithm is widely used for secure data transmission and has digital signatures and encryption applications.

# The ElGamal Cryptosystem

**Three steps with math formulas**



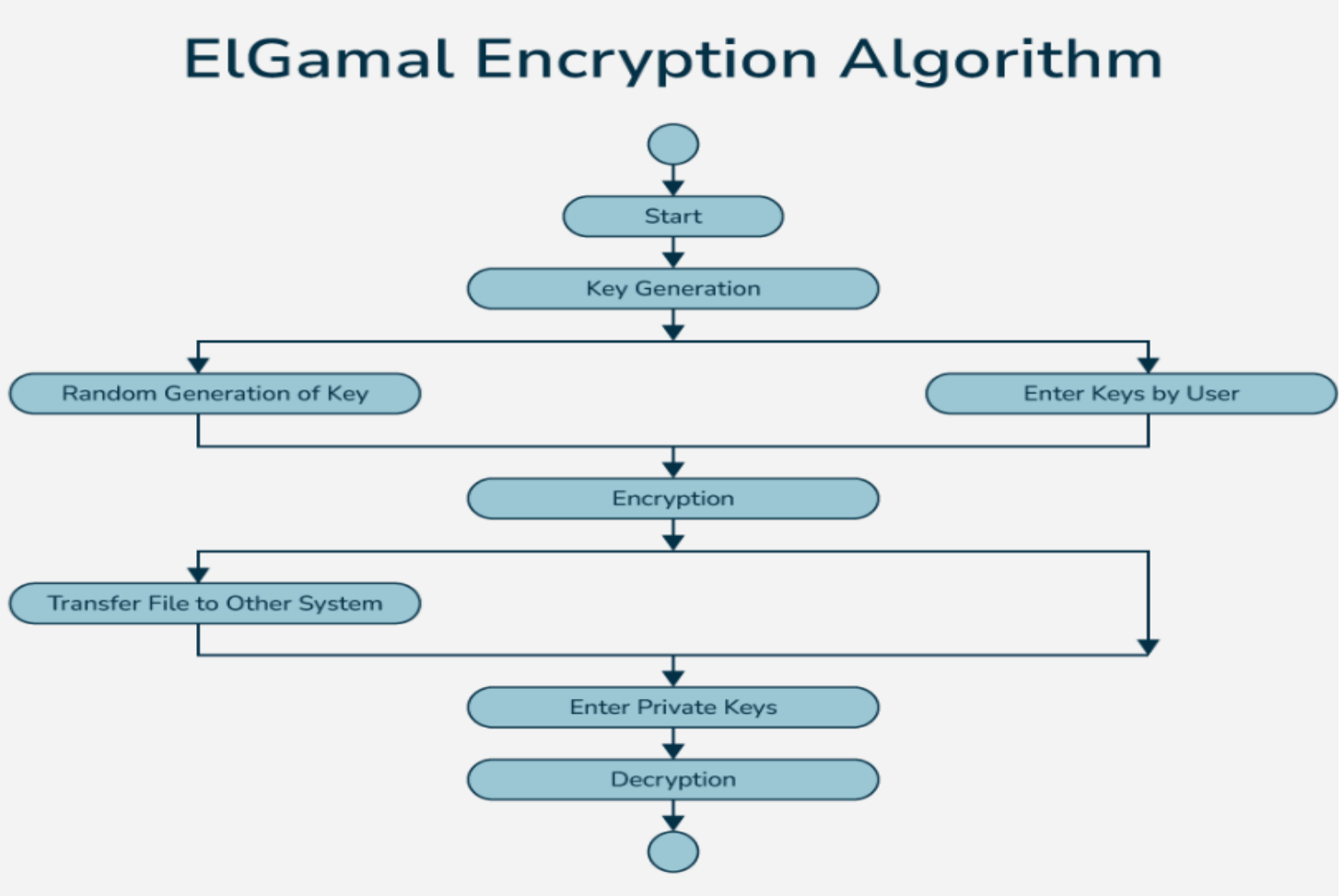
# ElGamal Encryption

- **Components of the ElGamal Algorithm**
- **Key Generation**:
  - **Public Parameters**: Select a large prime number p and a generator g of the multiplicative group $Z^*_p$.
  - **Private Key**: Select a private key x such that $1 \leq x \leq p-1$.
  - **Public Key**: Compute $h=g^x \bmod p$.
  - The public key is (p,g,h) and the private key is x.
- **Encryption**:
  - To encrypt a message M:
    - Choose a random integer k such that $1 \leq k \leq p-1$.
    - Compute $C_1 = g^k \bmod p$.
    - Compute $C_2 = M \cdot h^k \bmod p$.
    - The ciphertext is (c1,c2).

# ElGamal Encryption

- **Decryption**:
  - To decrypt the ciphertext (c1,c2) using the private key x:
    - Compute the shared secret $s = C_1^x \bmod p$.
    - Compute $s^{-1} \bmod p$ (the modular inverse of s).
    - Compute the original message $M = C_2 \cdot s^{-1} \bmod p$.

# ElGamal Encryption




*ElGamal Encryption Flowchart*

# ElGamal Encryption

## Idea of ElGamal Cryptosystem

Suppose Alice wants to communicate with Bob.

1. Bob generates public and private keys:
   - Bob chooses a very large number **q** and a cyclic group $\mathbf{F}_q$.
   - From the cyclic group $\mathbf{F}_q$, he choose any element **g** and an element **a** such that gcd(a, q) = 1.
   - Then he computes $h = g^a$.
   - Bob publishes **F**, $h = \mathbf{g^a}$, **q**, and **g** as his public key and retains **an** as a private key.

2. Alice encrypts data using Bob's public key :
   - Alice selects an element **k** from cyclic group **F** such that gcd(k, q) = 1.
   - Then she computes $p = g^k$ and $s = h^k = g^{ak}$.
   - She multiples s with M.
   - Then she sends $(p, M*s) = (g^k, M*s)$.

3. Bob decrypts the message :
   - Bob calculates $s' = p^a = g^{ak}$.
   - He divides M*s by $s'$ to obtain M as $s = s'$.

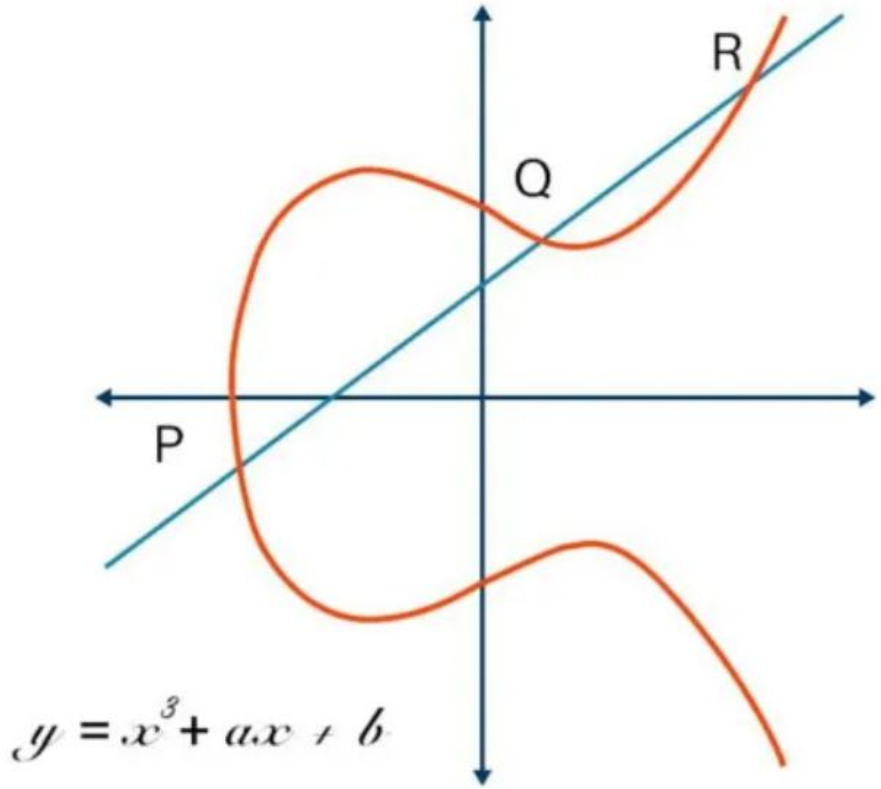
# ElGamal Encryption

- **Applications of ElGamal Encryption Algorithm**
- **Encryption**: ElGamal is used for encrypting messages where public key cryptography is required.
- **Digital Signatures**: A variant of ElGamal is used for creating digital signatures, ensuring message authenticity and integrity.

# Elliptical Curve Cryptography

- **Elliptic curve cryptography [ECC]**
- Elliptic curve cryptography [ECC] is a public-key cryptosystem just like RSA, Rabin, and El Gamal.
- Every user has a public and a private key.
- Public key is used for encryption/signature verification.
- Private key is used for decryption/signature generation.
- Elliptic curves are used as an extension to other current cryptosystems.
- Elliptic Curve Diffie-Hellman Key Exchange
- Elliptic Curve Digital Signature Algorithm

# Elliptical Curve Cryptography



# Elliptical Curve Cryptography

- **ECC- Algorithm**
- Both parties agree to some publicly-known data items
- The elliptic curve equation $y^2 = x^3 + ax + b \bmod p$

  -values of a and b such that $4a^3 + 27\,b^2 \neq 0$

  -prime, p
- The elliptic group is computed from the elliptic curve equation
- A base point, G, taken from the elliptic group
- Each user generates their public/private key pair
- Private Key = an integer, x selected from the interval [1, p-1]
- Public Key = product of private key and base point (Product = x*G)

# Elliptical Curve Cryptography

- **Example :**
- Suppose Alice wants to send Bob an encrypted message.
- Both agree on a base point, G.
- Alice and Bob create public/private keys.
- Alice: Private Key = $n_A$
- Public Key = $P_A = n_A * G$
- Bob : Private Key = $n_B$
- Public Key = $P_B = n_B * G$
- Alice takes plaintext message, M, and encodes it onto a point, $P_M$, from the elliptic group.

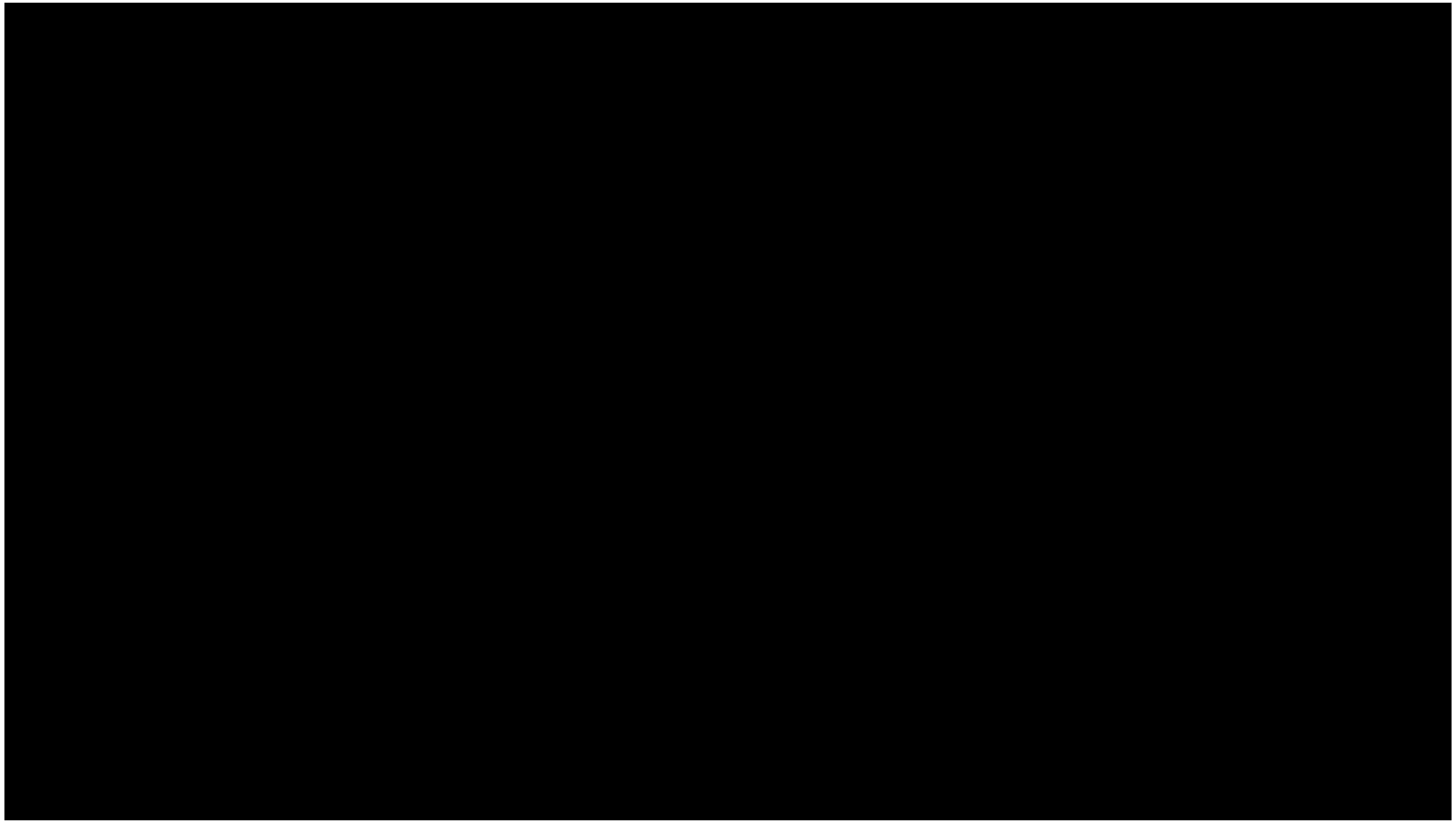
# Elliptical Curve Cryptography

- **Encryption** : Alice choose another random k – value from { 1,2,… p-1 } Cipher text : $C_m$ = { KG, $P_m$ + $KP_B$ }
- **Decryption** : by Bob
- Take the first point from $C_m$ - KG
- Multiply $K_G$ and private key of Bob : Product = $n_B$ KG
- Take the second point from $C_m$ and subtract the product from it $P_m$ + $KP_B$ - $n_B$ KG
- Substitute $P_B$ = $n_B$ * G
- Then $P_m$ + K $n_B$ * G - $n_B$ KG = $P_m$

# Elliptical Curve Cryptography

- ECC is particularly beneficial for application where:
- computational power is limited (wireless devices, PC cards)
- integrated circuit space is limited (wireless devices, PC cards)
- High speed is required.
- Intensive use of signing, verifying or authenticating is required.
- Signed messages are required to be stored or transmitted (especially for short messages).
- Bandwidth is limited (wireless communications and some computer networks). Advantages:
- Shorter key lengths
- Encryption, Decryption and Signature Verification speed up
- Storage and bandwidth saving

# Elliptical Curve Cryptography



# THANK YOU