# Unit 3

# Asymmetric Ciphers

# (8 Hours)

*3.4. Number Theory: Prime Numbers, Fermat's Theorem, Euler's Theorem, Primility Testing, Miller-Rabin Algorithm, Extended Euclidean Theorem, Discrete Logarithms*

*3.5. Public Key Cryptosystems, Applications of Public Key Cryptosystems*

*3.6. Distribution of public key, Distribution of secret key by using public key cryptography, Diffie-Helman Key Exchange, Man-in-the-Middle Attack*

*3.7. RSA Algorithm*

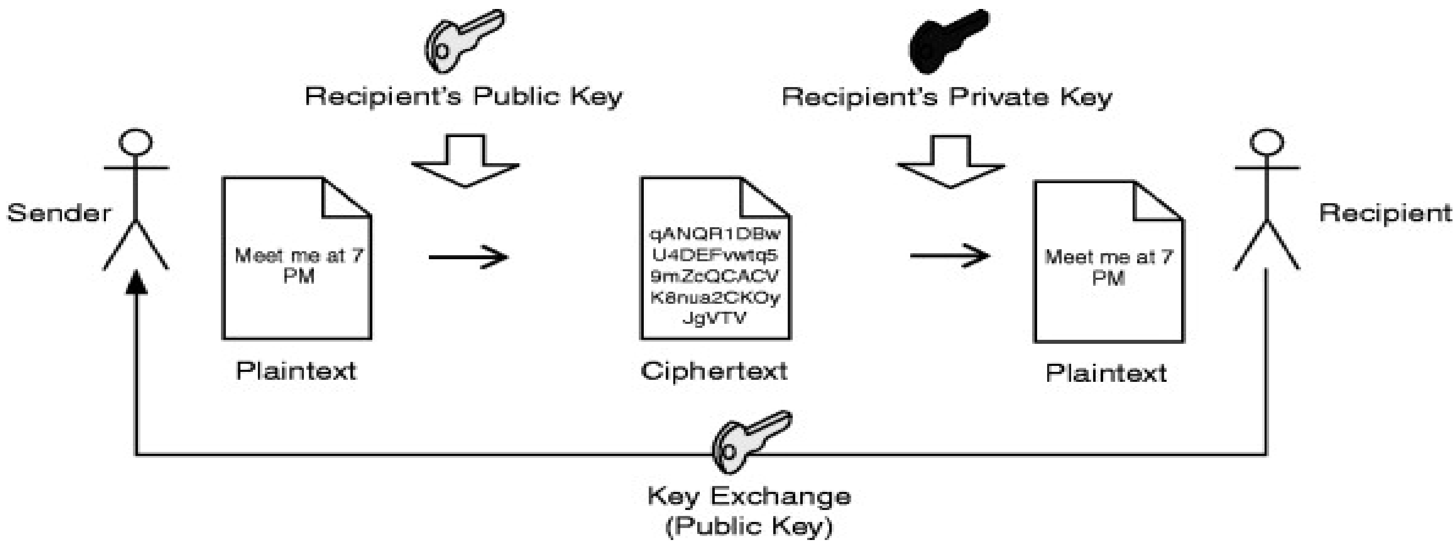
*3.8. Elgamal Cryptographic System*

**Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist**

# Public Key Cryptosystems

A **Public Key Cryptosystem** (also called **Asymmetric Cryptosystem**) is a cryptographic system that uses **two mathematically related keys**:

- **Public Key** → used for encryption (shared openly)
- **Private Key** → used for decryption (kept secret)




**Working Principle**

- Data encrypted with the **public key** can only be decrypted using the corresponding **private key**.
- Data signed with the **private key** can be verified using the **public key**.

| **Key Features** | **Examples of Public Key Cryptosystems** |
|---|---|
| • Eliminates the problem of secure key distribution<br>• Based on complex mathematical problems<br>• Provides confidentiality, authentication, and non-repudiation | • RSA<br>• Diffie–Hellman<br>• ElGamal<br>• ECC (Elliptic Curve Cryptography) |

# Applications of Public Key Cryptosystems

### 1. Secure Data Communication

Public key cryptography ensures **confidential communication** over insecure networks like the Internet.

📌 Example: Secure web browsing using HTTPS.

---

### 2. Key Distribution

Public key cryptography is widely used to **securely exchange secret keys** used in symmetric encryption.

📌 Example: SSL/TLS uses public key encryption to exchange session keys.

---

### 3. Digital Signatures

Public key cryptosystems are used to **sign digital documents**, ensuring:

- Authentication
- Integrity
- Non-repudiation

📌 Example: Software updates, legal documents.

---

### 4. Authentication

Used to verify the identity of users or systems.

📌 Example: Login systems using public key certificates.

---

### 5. Secure Email

Public key cryptography secures email content and verifies sender identity.

📌 Example: PGP (Pretty Good Privacy).

---

**6. Electronic Commerce (E-Commerce)**

Used to secure online transactions such as:

- Credit card payments
- Online banking

📌 Example: Secure payment gateways.

---

**7. Digital Certificates**

Public key cryptography supports **certificate authorities (CA)** to verify and bind identities to public keys.

📌 Example: SSL certificates issued by trusted CAs.

---

**8. Software Distribution**

Ensures that software is genuine and not altered.

📌 Example: Code signing in operating systems.

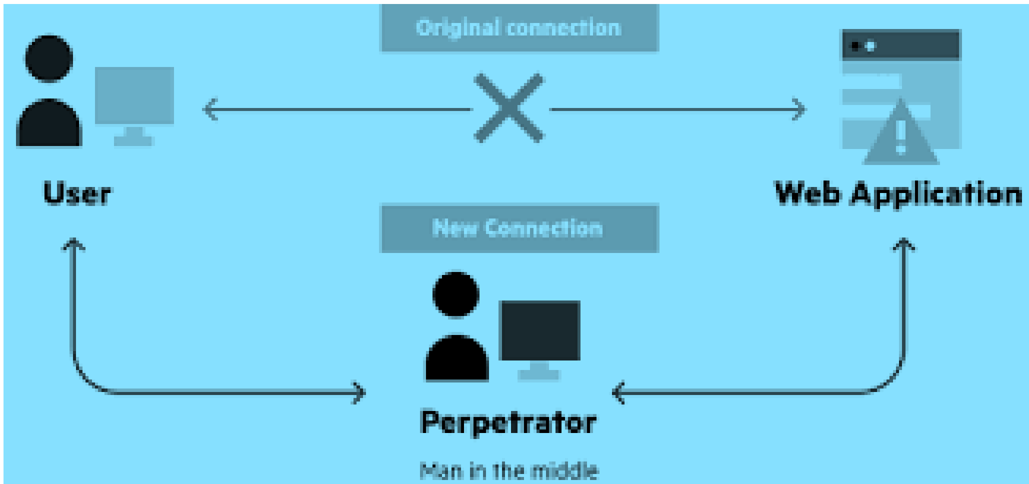
---

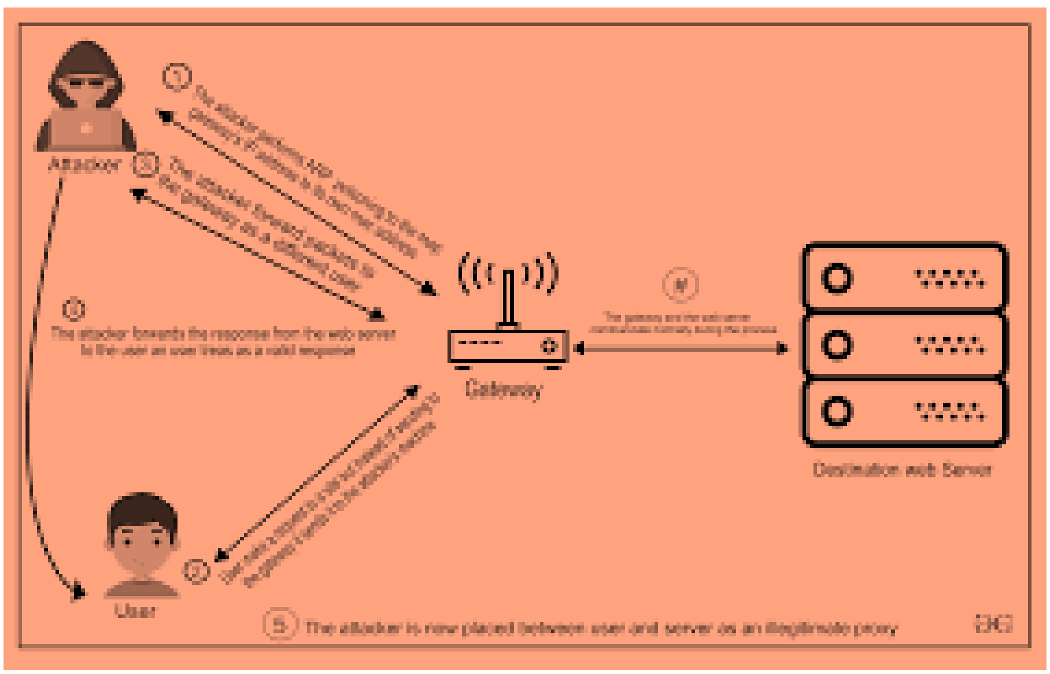
| **Advantages of Public Key Cryptosystems** | **Limitations** |
|---|---|
| • Secure key exchange<br>• Supports digital signatures<br>• High security<br>• Suitable for open networks | • Slower than symmetric cryptography<br>• Requires more computation power |

# Man-in-the-Middle Attack



A **Man-in-the-Middle attack** occurs when an attacker secretly intercepts and possibly alters communication between two parties.




**MITM in Diffie–Hellman**

- Attacker intercepts key exchange messages.
- Establishes separate keys with both users.
- Users believe they are communicating securely, but attacker reads/modifies data.

**Effects**

- Loss of confidentiality
- Data manipulation
- Identity impersonation

---

**Prevention of MITM Attack**

- Digital signatures
- Authentication using certificates
- Secure protocols (TLS, HTTPS)
- Public key authentication

# RSA Algorithm

RSA is a public key cryptographic algorithm used for **secure data transmission** and **digital signatures**. It uses two large prime numbers to generate a public and private key pair. The public key encrypts data, while the private key decrypts it. RSA's security relies on the difficulty of **factoring large composite numbers**.

📘 **Case Scenario RSA**

Ram wants to send a secure message to Shyam. Shyam chooses prime numbers **p** = **7** and **q** = **11**, giving **n** = **77** and **φ(n)** = **60**. He selects **e** = **7** and computes **d** = **43**. Ram encrypts message **M** = **9** as **C** = **$9^7$ mod 77** = **37** using the public key. Shyam decrypts it using **M** = **$37^{43}$ mod 77** = **9**, recovering the original message.

🔷 **Algorithm of RSA**

**Key Generation**

**Step 1:** Choose two prime numbers **p** and **q**.

**Step 2:** Compute $n = p \times q$

**Step 3:** Compute Euler's Totient $\phi(n) = (p-1)(q-1)$

**Step 4:** Choose public key **e** such that $1 < e < \phi(n), \gcd(e, \phi(n)) = 1$

**Step 5:** Compute private key **d** such that $ed \equiv 1 (\bmod \phi(n))$

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

**Decryption** $$M = C^d \bmod n$$

## 🔢 RSA Numerical Example

Ram wants to send a secure message to Shyam. Shyam chooses prime numbers **p = 7** and **q = 11**, giving **n = 77** and **φ(n) = 60**. He selects **e = 7** and computes **d = 43**. Ram encrypts message **M = 9** as **C = $9^7$ mod 77 = 37** using the public key. Shyam decrypts it using **M = $37^{43}$ mod 77 = 9**, recovering the original message.

**Given:**
p = 7, q = 11
Message M = 9

### Key Generation

| Step | Operation | Calculation | Result |
|---|---|---|---|
| 1 | Compute n | 7 × 11 | 77 |
| 2 | Compute φ(n) | 6 × 10 | 60 |
| 3 | Choose e | gcd(7, 60) = 1 | e = 7 |
| 4 | Compute d | 7d ≡ 1 (mod 60) | d = 43 |

---

### Encryption

| Step | Formula | Calculation | Result |
|---|---|---|---|
| 5 | $C = M^e \bmod n$ | $9^7 \bmod 77$ | 37 |

---

### Decryption

| Step | Formula | Calculation | Result |
|---|---|---|---|
| 6 | $M = C^d \bmod n$ | $37^{43} \bmod 77$ | 9 |

✅ **Original message recovered = 9**

# Diffie-Helman Key Exchange

Diffie–Hellman is a key exchange technique that allows two users to **generate a shared secret key over an insecure channel**. It uses a public prime number and a generator. Each user selects a private key and exchanges computed public values. The shared secret key is never transmitted, making the method secure. Its security depends on the **discrete logarithm problem**.

📘 **Case Scenario: Diffie–Hellman**

Two users, **Asha** and **Bikram**, want to share a secret key securely. They agree on public values **p = 23** and **g = 5**. Asha chooses private key **a = 6** and sends **$5^6$ mod 23 = 8**. Bikram chooses **b = 15** and sends **$5^{15}$ mod 23 = 19**. Asha computes **$19^6$ mod 23 = 2**, and Bikram computes **$8^{15}$ mod 23 = 2**. Both obtain the same secret key **2**.

🔷 **Algorithm of Diffie–Hellman Key Exchange**

**Step 1:** Select a large prime number **p** and a primitive root **g** (public).
**Step 2:** Sender selects private key **a**.
**Step 3:** Receiver selects private key **b**.
**Step 4:** Sender computes public value $\boldsymbol{A = g^a mod p}$
**Step 5:** Receiver computes public value $\boldsymbol{B = g^b mod p}$

**Step 6:** Exchange A and B.
**Step 7:** Sender computes shared key $\boldsymbol{K = B^a mod p}$
**Step 8:** Receiver computes shared key $\boldsymbol{K = A^b mod p}$

## 🔢 Diffie–Hellman Numerical Example

Two users, **Asha** and **Bikram**, want to share a secret key securely. They agree on public values **p = 23** and **g = 5**. Asha chooses private key **a = 6** and sends **$5^6$ mod 23 = 8**. Bikram chooses **b = 15** and sends **$5^{15}$ mod 23 = 19**. Asha computes **$19^6$ mod 23 = 2**, and Bikram computes **$8^{15}$ mod 23 = 2**. Both obtain the same secret key **2**.

**Given:**
p = 23, g = 5
Private keys: a = 6, b = 15

| Step | Operation | Calculation | Result |
|---|---|---|---|
| 1 | Public value A | $5^6$ mod 23 | 8 |
| 2 | Public value B | $5^{15}$ mod 23 | 19 |
| 3 | Shared key (Sender) | $19^6$ mod 23 | 2 |
| 4 | Shared key (Receiver) | $8^{15}$ mod 23 | 2 |

✅ **Shared Secret Key = 2**

```
controlplane:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tI6NdC0CNakrh878WcoHcEPUoMkCuFPTxuLlR4dMWEY root@controlplane
The key's randomart image is:
+---[RSA 3072]----+
|o  +oOEo         |
|o.+oBo*..        |
|.++B.o ..        |
|o.o =... o       |
| . + +o S .      |
|  o +. B .       |
| + o .+ o        |
|  +. +.          |
|   .=.           |
+----[SHA256]-----+
controlplane:~$
```

```
controlplane:~$
controlplane:~$ cat /root/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
```


```
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAxbN+pd+41mRpgKWhRD29EEplsVGu5JFEL9LWg5g9LuTTcu3EK21O
VhgYvsJ4hx3nEyaNglQ/nBkF1qK/xbycTE/c/O+6nWuwEfXtz4/NMjCqnDtRC2X5afCiZ8
KK/JHQEVVp06fNgFD/N31YW0I/i4xLXjdc/CJzo+F9Zx9SEQA7VmH/yIJ5+HT5DP4wcjU2
jtp3KNWPp0FDQXJVrBnYyuznbqFFInwRQnE8p5kXXIGBkvDn2PJkxODVJUUPPl41V9cHca
sLUlELHNW24bhNmm9//ZRQugYwjTbtfgMdu3ast+Df2W0QvfDQWoWxHpTVkTcSxXmk3WsO
DS9XQGHOj4NHbh4KGLNaNYAskN9RvXFTRYixdvJ7kFwS3CmsuTqaid2Km82jmH0cb8nrBW
IPsq/mECpW5j+PJdlt4WVRu4imIzK9sjXb2rMedaw5dx6vnqsZoEpjtzAQdSNNYmfdv0OP
Hbb7KPIYjNwTwIzpvLZD7DV9/sZVhybZZsd9JyGlAAAFiOBevnDgXr5wAAAAB3NzaC1yc2
EAAAGBAMWzfqXfuNZkaYCloUQ9vRBKZbFRruSRRC/S1oOYPS7k03LtxCttTlYYGL7CeIcd
5xMmjYJUP5wZBdaiv8W8nExP3Pzvup1rsBH17c+PzTIwqpw7UQtl+WnwomfCivyR0BFVad
OnzYBQ/zd9WFtCP4uMS143XPwic6PhfWcfUhEAO1Zh/8iCefh0+Qz+MHI1No7adyjVj6dB
Q0FyVawZ2Mrs526hRSJ8EUJxPKeZF1yBgZLw59jyZMTg1SVFDz5eNVfXB3GrC1JRCxzVtu
G4TZpvf/2UULoGMI027X4DHbt2rLfg39ltEL3w0FqFsR6U1ZE3EsV5pN1rDg0vV0Bhzo+D
R24eChizWjWALJDfUb1xU0WIsXbye5BcEtwprLk6mondipvNo5h9HG/J6wViD7Kv5hAqVu
Y/jyXZbeFlUbuIpiMyvbI129qzHnWsOXcer56rGaBKY7cwEHUjTWJn3b9Djx22+yjyGIzc
E8CM6by2Q+w1ff7GVYcm2WbHfSchpQAAAAMBAAEAAAGALz5WfHL665ixMYHjCLisLnFo3b
fEUHB6V+dNRcMik8sw0F0LLMmBF8fIWgWE4dz+dDAZTtStIrNEFudrZmhM/Vt0Vqk+8hFi
0NfxDVGy2lSQ4Ka4Fj43HN7ZIGx35CVAEHRVxrNpw0kXw/3bGGu7XbCX7hrSyJm/AqGG4x
az+MYSZ/EhO5waY4ooXbd8gj4/HGIOaZ6nJnPNC52z5V2D1fJI18IIEFaPlU4UDYTi9UMp
YKDa9pfuBrInnTFklCXUs6LFERylvKpP77Ntayy3BfWT6JCuQDxWevRxYF2eKk09O3Nto6
VBetbsLWTXoVUMW2X9P4WfgU4rlRIm2kVgwV3IMOHa2RXHyBDeIi3jeS2Qz2rGRW5/89u2
8cKnzOXmIYXpC7AjVKai4ubCeuFZ7yNExvhjLP8RC90LQABZkJsBl52aGNyEyQ93N7IMWK
icq2s/jAxKosvtkfjcb+ziDQ5Hdck8qglxNr6H+pV1K3qXI0VY98PFFXgaZ5op4/LhAAAA
wGAfs0dEd7SwjsJS0i5WY0859EWZVZrnFoR7vhY2JBvMUroS+2eIPl1HkMoMV9eBfaHQBK
nJq8QRqmt4WxKIjHWM8kEFbBCh9xC56CJBav13wo3XTpvB2byjDVoYmGwAxPEq6fMnHBdD
ns5YTKunCg+hil3rGggfszz9CmsOP3MdUfeYd7Va93tYHqhYxGsgrIUc0EMMLtAhrJDthJ
1I89XmgiPb0+fKCqOG+W/bIZptNLXmbavA5fnKEuIFKRa1RgAAAMEA8v4pWB/psaPFH6t8
juk7d8hSL38BnFYR8YUUm+UoiLXu1GWDiN7SEqKM1gG/gDF9qntr1/mGGzNtzGERDY+p4d
0g0sbK9IRa0msL0OlV2jRJz/VQ8aYgBNzNDTkep9PDaLHS5sOHzAiqSi8HK/sIqXaWTX2d
```


```
controlplane:~$ cat /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDFs36l37jWZGmApaFEPb0QSmWxUa7kkUQv0taDmD0u5NNy7cQrb
U5WGBi+wniHHecTJo2CVD+cGQXWor/FvJxMT9z877qda7AR9e3Pj80yMKqcO1ELZflp8KJnwor8kdARVWnTp82AUP
83fVhbQj+LjEteN1z8InOj4X1nH1IRADtWYf/Ignn4dPkM/jByNTaO2nco1Y+nQUNBclWsGdjK7OduoUUifBFCcTy
nmRdcgYGS8OfY8mTE4NUlRQ8+XjVX1wdxqwtSUQsc1bbhuE2ab3/9lFC6BjCNNu1+Ax27dqy34N/ZbRC98NBahbEe
lNWRNxLFeaTdaw4NL1dAYc6Pg0duHgoYs1o1gCyQ31G9cVNFiLF28nuQXBLcKay5OpqJ3YqbzaOYfRxvyesFYg+yr
+YQKlbmP48l2W3hZVG7iKYjMr2yNdvasx51rDl3Hq+eqxmgSmO3MBB1I01iZ92/Q48dtvso8hiM3BPAjOm8tkPsNX
3+xlWHJtlmx30nIaU= root@controlplane
controlplane:~$
```

# Distribution of public key



Public key distribution is the process of **making a user's public key available** to others in a secure and trustworthy way.

**Methods of Public Key Distribution**

**a) Public Announcement**

- User publishes public key openly (website, email).
- ❌ Not secure (can be replaced by attacker).

**b) Publicly Available Directory**

- A trusted directory stores user identities with public keys.
- Users can retrieve keys when needed.

**c) Public Key Authority**

- Central trusted authority provides public keys on request.
- Ensures authenticity and freshness.

**d) Certificates (Most Common)**

- Public keys are distributed using **digital certificates** issued by a **Certificate Authority (CA)**.
- Certificate contains user identity and public key, signed by CA.

✅ **Most secure and widely used method**

# Distribution of secret key by using public key cryptography

| | |
|---|---|
| Public key cryptography can be used to **securely exchange a symmetric (secret) key**.<br><br>**Steps**<br><br>1. Sender generates a random secret key.<br>2. Encrypts the secret key using receiver's **public key**.<br>3. Encrypted key is sent over the network. | **Purpose**<br><br>• Combines speed of symmetric encryption with security of asymmetric encryption.<br>• Used in **hybrid cryptosystems**.<br><br>📌 **Example:** SSL/TLS |

| | |
|---|---|
| 4. Receiver decrypts it using **private key**.<br>5. Both parties now share the same secret key. |  |

# Elgamal Cryptographic System

The **ElGamal Cryptographic System** is an **asymmetric (public key) encryption scheme** based on the **Discrete Logarithm Problem**. It provides confidentiality and is commonly used in secure communications. Unlike RSA, ElGamal produces **two ciphertext values**, which increases security.

---

### ⚙ ElGamal Algorithm

**Key Generation**

1. Choose a large prime number **p**.
1. Choose a generator **g** of the multiplicative group modulo **p**.
2. Choose a private key **x**, where $\mathbf{1 < x < p - 1}$.
3. Compute public key $\boldsymbol{y = g^x mod p}$

**Public Key:** (p, g, y)
**Private Key:** x

**Encryption**

To encrypt message **M**:

1. Choose a random number **k**, where $\mathbf{1 < k < p - 1}$.
2. Compute $\boldsymbol{C_1 = g^k mod p}$
3. Compute $\boldsymbol{C_2 = M \times y^k mod p}$
   **Ciphertext: ($C_1$, $C_2$)**

---

**Decryption**

To decrypt ciphertext ($C_1$, $C_2$):

1. Compute

$$S = C_1^x \, mod p$$

2. Compute modular inverse $S^{-1}$.

3. Recover message

$$M = C_2 \times S^{-1} mod p$$

---

### 🔢 Numerical Example (Step-wise Table)

**Given:**
p = 23, g = 5
Private key x = 6
Message M = 10
Random number k = 7

---

### 🔑 Key Generation

| Step | Calculation | Result |
|---|---|---|
| $y = g^x \bmod p$ | $5^6 \bmod 23$ | 8 |

**Public Key:** (23, 5, 8)
**Private Key:** 6

---

### 🔐 Encryption

| Step | Formula | Calculation | Result |
|---|---|---|---|
| $C_1$ | $g^k \bmod p$ | $5^7 \bmod 23$ | 17 |
| $C_2$ | $M \times y^k \bmod p$ | $10 \times 8^7 \bmod 23$ | 21 |

**Ciphertext = (17, 21)**

🔒 **Decryption**

| Step | Formula | Calculation | Result |
|---|---|---|---|
| S | $C_1^x$ mod p | $17^6$ mod 23 | 12 |
| $S^{-1}$ | Modular inverse of 12 mod 23 | 2 | |
| M | $C_2 \times S^{-1}$ mod p | 21 × 2 mod 23 | 10 |

✅ **Original message recovered = 10**

---

⭐ **Key Features**

- Based on **Discrete Logarithm Problem**
- Produces **two-part ciphertext**
- More secure but larger ciphertext than RSA

# Sdsds

## Sdsds

**Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist**