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

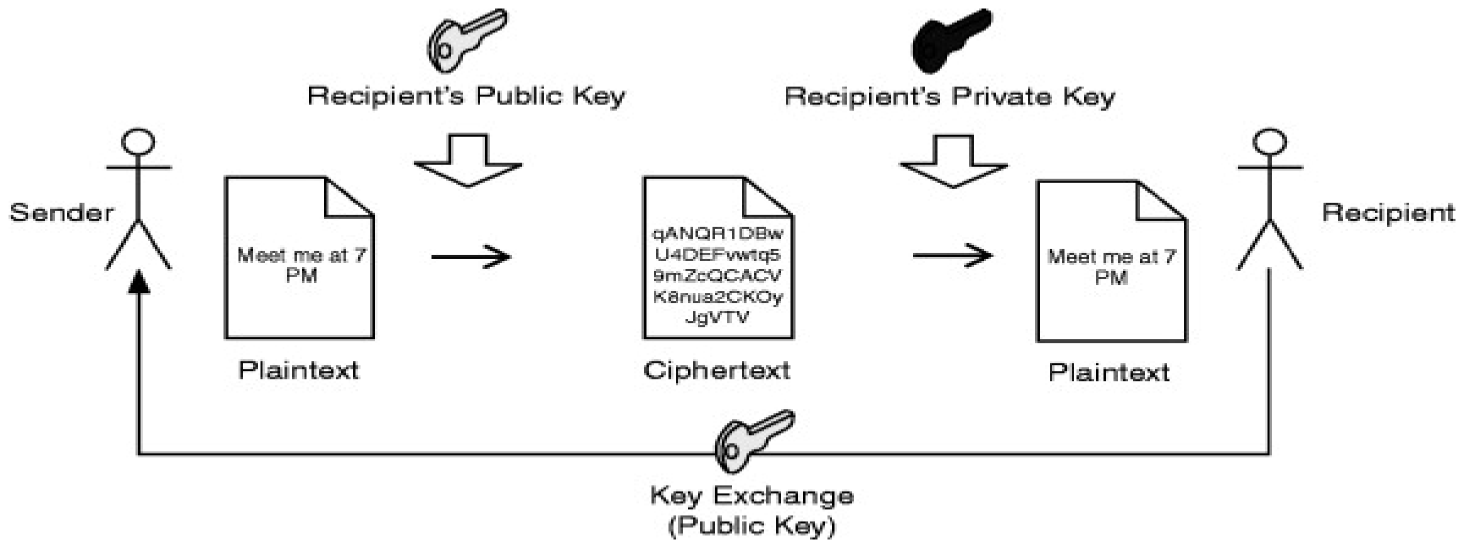
3.8. Elgamal Cryptographic System

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 • Based on complex mathematical problems • Provides confidentiality, authentication, and non-repudiation	• RSA • Diffie–Hellman • ElGamal • 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

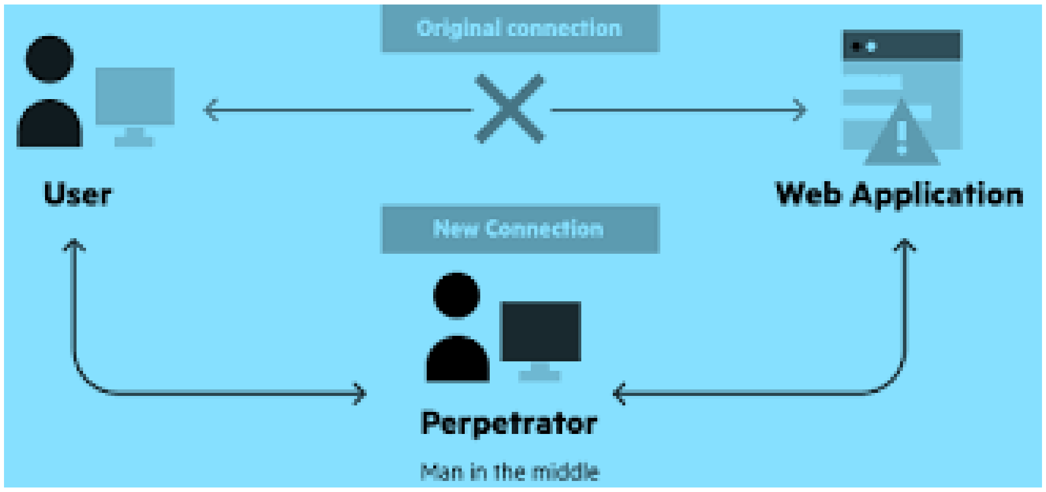
Ensures that software is genuine and not altered.

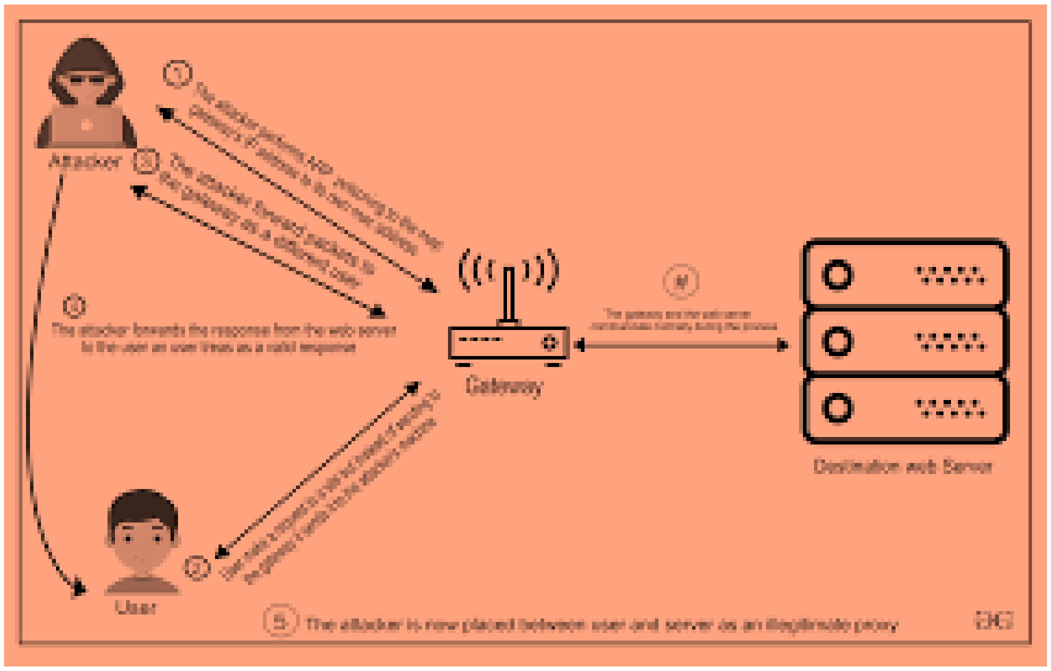
📌 Example: Code signing in operating systems.

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

Man-in-the-Middle Attack

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




Attacker
The attacker performs ARP poisoning to the user gateway's IP address to its own mac address
The attacker forward packets to the gateway as a different user
The attacker forwards the response from the web server to the user an user treats as a valid response
Gateway
Destination web Server
User
The attacker is now placed between user and server as an illegitimate proxy

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

Number A **number** is a mathematical value used to count, measure, or label objects. Examples: 1, 2, 3, −5, ½, √2	**Integer** An **integer** is a whole number with no fractional part. It includes **positive numbers, negative numbers, and zero**. **Set:** $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ Example: −4, 0, 7
Rational Number A **rational number** is any number that can be written in the form $\frac{p}{q}, q \neq 0$ Examples: ½, −3/4, 5, 0.25 (All integers are rational numbers.)	**Odd Number** An **odd number** is an integer that is **not divisible by 2**. **Form:** $2n + 1$ Examples**: 1, 3, 5, 7, 9**
Even Number An **even number** is an integer that **is divisible by 2**. **Form:** $2n$ Examples: **2, 4, 6, 8, 10**	
Prime Number A **prime number** is a natural number **greater than 1** that has **exactly two factors**: 1 and itself. Examples: 2, 3, 5, 7, 11 (Note: 2 is the **only even prime number**.)	**Co-Prime Numbers** Two numbers are **co-prime (relatively prime)** if their **GCD = 1**. Examples: • 8 and 15 → co-prime • 14 and 21 → not co-prime (GCD = 7)

🔐 Connection of Basic Number Concepts with Number Theory in Cryptography

Cryptography relies heavily on **number theory**, especially properties of integers and prime numbers, to ensure secure communication.

1. Numbers & Integers in Cryptography

Cryptographic algorithms work entirely with **integers**, not real or rational numbers.

📌 Examples:

- Modular arithmetic uses integers only.
- Encryption and decryption operations are performed on integer values.

👉 **Integers form the foundation of cryptographic computations.**

2. Rational Numbers

Rational numbers are **not directly used** in cryptography because encryption algorithms require **exact, discrete values**.

📌 Cryptography avoids fractions to prevent precision errors and ensure deterministic results.

👉 **Only integer arithmetic is used in cryptographic systems.**

3. Odd and Even Numbers

- Most cryptographic primes are **odd numbers**.
- Even numbers (except 2) are avoided because they are easily factorable.

📌 Example:

- RSA selects two **large odd primes**.
- Even numbers reduce security.

👉 **Odd numbers help maintain complexity and security.**

4. Prime Numbers

Prime numbers are the **backbone of modern cryptography**.

📌 Uses:

- RSA uses large primes to generate keys.
- Diffie–Hellman and ElGamal rely on prime modulus.
- Prime factorization is computationally hard.

👉 **Security depends on the difficulty of breaking prime-based problems.**

5. Co-Prime Numbers

Two numbers are co-prime if their **GCD = 1**.

📌 Uses:

6. Modular Arithmetic & Integers

Most cryptographic operations use:

$$amodn$$

- In RSA, public key **e** must be co-prime with φ(n).
- Modular inverse exists only when numbers are co-prime.

👉 **Co-primality ensures successful key generation and decryption.**

📌 Example:

- Encryption: $C = M^e \bmod n$
- Key exchange: $g^x \bmod p$

👉 **Modular arithmetic over integers enables secure encryption.**

🔢 **First 80 Prime Numbers**

2, 3, 5, 7, 11, 13, 17, 19, 23, 29,31, 37, 41, 43, 47, 53, 59, 61, 67, 71,73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173,179, 181, 191, 193, 197, 199, 211, 223, 227, 229,233, 239, 241, 251, 257, 263, 269, 271, 277, 281,283, 293, 307, 311, 313, 317, 331, 337, 347, 349,353, 359, 367, 373, 379, 383

GCD (Greatest Common Divisor)

The **GCD** of two integers **a** and **b** is the largest positive integer that divides both without remainder.

Euclidean Algorithm

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

Key Properties

- $\gcd(a, b) = \gcd(b, a)$
- If $\gcd(a, b) = 1$, then **a and b are co-prime**
- GCD helps determine the **existence of modular inverse**

Cryptography Use

- In RSA, choose e such that $\gcd(e, \phi(n)) = 1$
- Extended Euclidean Algorithm uses GCD to find modular inverse

Euler's Totient Function ϕ(n)

$\phi(n)$ is the number of **positive integers less than n** that are **co-prime with n**.

Formulas

- If $n = p$(prime): $\phi(p) = p - 1$
- If $n = p^k$: $\phi(p^k) = p^k - p^{k-1}$
- If $n = pq$(distinct primes):

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

- General form:

$$\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Examples

Example 1

Find gcd(24, 36)

36 mod 24 = 12
24 mod 12 = 0

✅ **GCD = 12**

Example 2

Find gcd(48, 18)

48 mod 18 = 12
18 mod 12 = 6
12 mod 6 = 0

✅ **GCD = 6**

Example 3

Find gcd(35, 64)

64 mod 35 = 29
35 mod 29 = 6
29 mod 6 = 5
6 mod 5 = 1
5 mod 1 = 0

✅ **GCD = 1 (Co-prime)**

Example 4

Find gcd(81, 57)

81 mod 57 = 24
57 mod 24 = 9
24 mod 9 = 6
9 mod 6 = 3
6 mod 3 = 0

✅ **GCD = 3**

Cryptography Use

- RSA uses $\phi(n)$to compute private key d
- Determines key relationships in modular arithmetic

Examples

Example 1

Find φ(11)

11 is prime

$$\phi(11) = 11 - 1 = 10$$

Example 2

Find φ(15)

15 = 3 × 5

$$\phi(15) = 15\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right)$$
$$= 15 \times \frac{2}{3} \times \frac{4}{5} = 8$$

Example 3

Find φ(16)

$16 = 2^4$

$$\phi(16) = 16\left(1 - \frac{1}{2}\right) = 8$$

Example 4

Find φ(21)

21 = 3 × 7

$$= 21 \times \frac{2}{3} \times \frac{6}{7} = 12$$

n	φ(n)	Numbers co-prime to n (< n)
1	1	{1}
2	1	{1}
3	2	{1, 2}
4	2	{1, 3}
5	4	{1, 2, 3, 4}
6	2	{1, 5}
7	6	{1, 2, 3, 4, 5, 6}
8	4	{1, 3, 5, 7}
9	6	{1, 2, 4, 5, 7, 8}
10	4	{1, 3, 7, 9}

Euler's Theorem/ Fermat Euler's Theorem/ Euler's Totient Theorem

Statement

If gcd $(a, n) = 1$, then: Coprime (GCD=1)

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

Special Case (Fermat's Theorem)

If $n = p$ is prime:

$$a^{p-1} \equiv 1 (\mathrm{mod} p)$$

Cryptography Use

- Basis of **RSA correctness**
- Ensures encrypted message decrypts to original message

📘 Fermat's Little Theorem

🔷 **Statement**

If **p is a prime number** and **a is any integer** such that

gcd $(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$

Equivalently, $a^p \equiv a \pmod{p}$

🔷 **Why it matters in Cryptography**

- Used in **primality testing**
- Foundation for **RSA, Diffie–Hellman, ElGamal**
- Helps simplify **large modular exponentiation**

✏️ **Numerical Examples**

✅ **Example 1**

Let $p = 7,\ a = 3$

$$3^6 = 729 \equiv 1 \pmod{7}$$

Examples

Example 1

Verify Euler's theorem for a = 3, n = 10

φ(10) = 5 x 2 = (5-1) x (2-z) = 4

$$3^4 = 81 \equiv 1(\mathrm{mod}10)$$

✅ Verified

Example 2

Verify for a = 7, n = 15

φ(15) = 5 x 3 = 4 x 2 = 8

$$7^8 \equiv 1(\mathrm{mod}15)$$

✅ Verified

Example 3

Verify for a = 2, n = 9

φ(9) =9−3=6

$$2^6 = 64 \equiv 1(\mathrm{mod}9)$$

✅ Verified

Example 4

Verify for a = 5, n = 14

φ(14) = 7 x 2 = (7-1) x (2-1) =6

$$5^6 = 15625 \equiv 1(\mathrm{mod}14)$$

✅ Verified

✓ Fermat's theorem verified

✅ **Example 2**

Let $p = 11,\ a = 2$

$$2^{10} = 1024 \equiv 1\ (\mathrm{mod}\ 11)$$

✅ **Example 3**

Let $p = 13,\ a = 5$

$$5^{12} \equiv 1\ (\mathrm{mod}\ 13)$$

❌ **Example 4 (Composite number)**

Let $n = 15,\ a = 2$

$$2^{14} \not\equiv 1\ (\mathrm{mod}\ 15)$$

🚫 Fermat's theorem **fails** for non-prime numbers

🔐 **Cryptography Insight**

- If $a^{p-1} \not\equiv 1\ (\mathrm{mod}\ p)$, then **p is NOT prime**
- Used in **Fermat Primality Test**

Example 5

$$p = 5, a = 2$$

$$2^{5-1} = 2^4 = 16 \equiv 1(\mathrm{mod}5)$$

Example 6

$$p = 17, a = 3$$

$$3^{16} = 43046721 \equiv 1(\mathrm{mod}17)$$

Example 7

$$p = 19, a = 4$$

$$4^{18} \equiv 1 (\bmod 19)$$

Example 8

$$p = 23, a = 5$$

$$5^{22} \equiv 1 (\bmod 23)$$

❌ Composite Number Examples (Fermat Fails)

Example 9

$$n = 9, a = 2$$

$$2^{8} = 256 \equiv 4 \not\equiv 1 (\bmod 9)$$

Example 10

$$n = 12, a = 5$$

$$5^{11} \equiv 1 \not\equiv 1 (\bmod 12) (\text{actually } 5^{11} \equiv 5 (\bmod 12))$$

Example 11

$$n = 18, a = 7$$

$$7^{17} \equiv 7 \not\equiv 1 (\bmod 18)$$

Example 12

$$n = 20, a = 3$$

$$3^{19} \equiv 3 \not\equiv 1 (\bmod 20)$$

🔷 Primality Testing

Definition:
Primality testing is the process of determining whether a given number n is **prime** (only divisible by 1 and itself) or **composite** (has divisors other than 1 and itself).

Euclidean Algorithm

Purpose:
Find the **Greatest Common Divisor (GCD)** of two integers $a$and b.

Principle: $\mathbf{gcd}\,(\boldsymbol{a}, \boldsymbol{b}) = \mathbf{gcd}\,(\boldsymbol{b}, \boldsymbol{a} \bmod \boldsymbol{b})$

1 Trial Division (Basic Method)

Repeat until remainder = 0. The last non-zero remainder is the GCD.

- Check divisibility of $n$by all integers from 2 to $\sqrt{n}$.
- **Example:**
$$n = 29$$
Check divisibility by 2, 3, 5 (since $\sqrt{29} \approx 5.39$) → not divisible → **prime**

Pros: Simple, accurate
Cons: Slow for large numbers

Examples Prime:
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47

Composite Numbers

Definition:
A **composite number** is a natural number greater than 1 that has **more than two positive divisors** (i.e., it has at least one divisor other than 1 and itself).

Properties:

- Divisible by numbers other than 1 and itself
- Can be **expressed as a product of primes**

Examples:

4,6,8,9,10,12,14,15,16,18,20,21,22,24,25

Feature	**Prime Number**	**Composite Number**
Divisors	2 (1 and itself)	More than 2
Examples	2, 3, 5, 7, 11	4, 6, 8, 9, 12
Factorization	Only itself × 1	Product of primes
Even Numbers	Only 2	Rest of even numbers

2 **Fermat Primality Test**

- Uses **Fermat's Little Theorem**:
If n is prime, $a^{n-1} \equiv 1 (\mathrm{mod} n)$for any a with $\gcd(a, n) = 1$
- **Steps:**

Step-by-Step Example

Find GCD of 252 and 105

1. $252 \div 105 = 2$remainder $42 \rightarrow 252 = 105 \cdot 2 + 42$
2. $105 \div 42 = 2$remainder $21 \rightarrow 105 = 42 \cdot 2 + 21$
3. $42 \div 21 = 2$remainder $0 \rightarrow$ Stop

✓ **GCD = 21**

Algorithm (Steps)

1. Divide $a$by b, get remainder r
2. Replace $a$by b, $b$by r
3. Repeat until $r = 0$
4. GCD = last non-zero remainder

Extended Euclidean Algorithm

Purpose:
Find integers $x$and $y$such that:

$$ax + by = \gcd(a, b)$$

- Very useful in **modular inverse calculation** for cryptography (RSA, ECC).

Step-by-Step Example

Find integers $x$and $y$such that:

$$252x + 105y = \gcd(252,105)$$

Step 1: Apply Euclidean Algorithm

$$252 = 105 \cdot 2 + 42$$
$$105 = 42 \cdot 2 + 21$$
$$42 = 21 \cdot 2 + 0$$

1. Pick a random $a < n$
2. Compute $a^{n-1} \bmod n$
3. If $\not\equiv 1$, n is **composite**
4. If $\equiv 1$, n is **probably prime**

Example:

$$n = 17, a = 3$$

$$3^{16} \equiv 1 (\bmod 17) \implies 17 \text{ is probably prime}$$

Note: Some composite numbers (Carmichael numbers) can pass this test → **not fully reliable**

3 Miller-Rabin Primality Test (Probabilistic)

- Advanced version of Fermat test, **more reliable**
- Checks if n passes multiple modular exponentiation conditions
- If n fails any, it is **composite**
- If n passes all, it is **probably prime**

Use: Common in cryptography for generating large primes (RSA keys)

4 AKS Primality Test (Deterministic)

- Always gives correct answer
- Polynomial-time deterministic test
- Less commonly used due to **slower practical performance**

5 Applications in Cryptography

- **RSA, Diffie-Hellman, ElGamal** all require **large prime numbers**
- Efficient primality tests help generate secure keys quickly

✓ GCD = 21

Step 2: Back-substitute to find x and y

1. $21 = 105 - 42 \cdot 2$
2. Replace $42 = 252 - 105 \cdot 2 \rightarrow$

$$21 = 105 - (252 - 105 \cdot 2) \cdot 2$$
$$21 = 105 - (252 \cdot 2 - 105 \cdot 4)$$
$$21 = 105 - 252 \cdot 2 + 105 \cdot 4$$
$$21 = -252 \cdot 2 + 105 \cdot 5$$

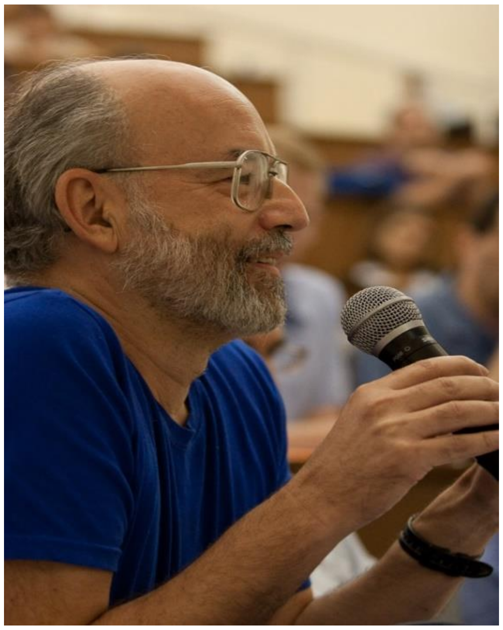
✓ **Solution:** $x = -2, y = 5$

$$252(-2) + 105(5) = 21$$

• Fermat test is fast but probabilistic → Miller-Rabin is preferred	

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**.



- The RSA (**Rivest–Shamir–Adleman**) cryptosystem is a family of public-key cryptosystems, one of the oldest widely used for secure data transmission.
- The initialism "RSA" comes from the surnames of **Ron Rivest, Adi Shamir and Leonard Adleman**, who publicly described the algorithm in 1977.
- An equivalent system was developed secretly in 1973 at Government Communications Headquarters (GCHQ), the British signals intelligence agency, by the English mathematician Clifford Cocks. That system was declassified in 1997.

📘 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 $1 < e < \phi(n), \quad \gcd(e, \phi(n)) = 1$

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

Encryption $C = M^e \bmod n$

Decryption $M = C^d \bmod n$

Resulting Keys:

- **Public Key:** (e, n)
- **Private Key:** (d, n)

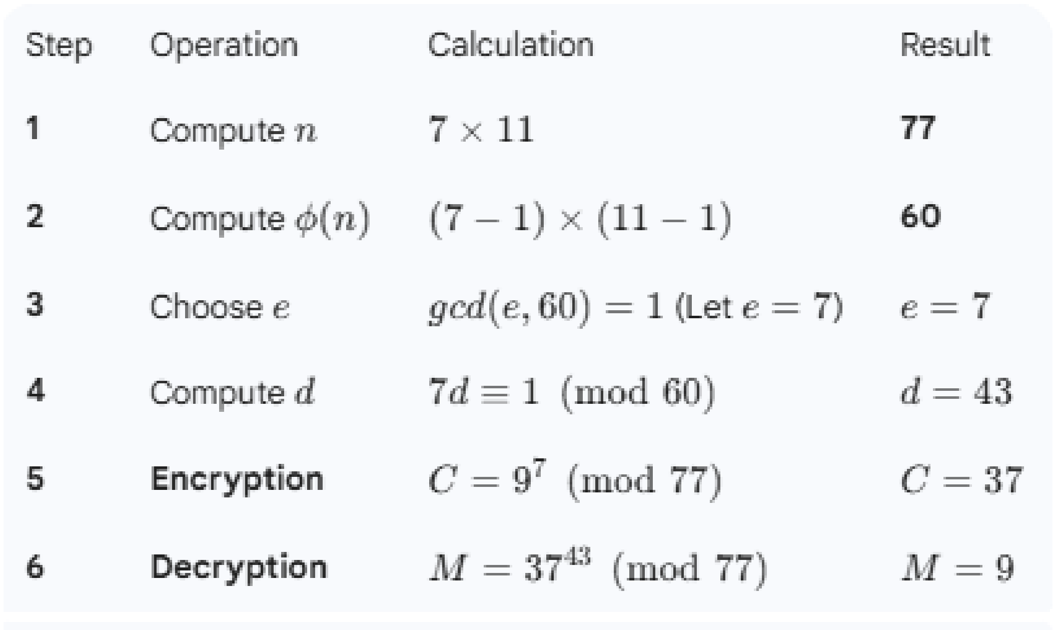
🔢 RSA Numerical Example

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

✅ **Original message recovered = 9**



Step	Operation	Calculation	Result
1	Compute n	7×11	**77**
2	Compute $\phi(n)$	$(7-1) \times (11-1)$	**60**
3	Choose e	$gcd(e, 60) = 1$ (Let $e = 7$)	$e = 7$
4	Compute d	$7d \equiv 1 \pmod{60}$	$d = 43$
5	**Encryption**	$C = 9^7 \pmod{77}$	$C = 37$
6	**Decryption**	$M = 37^{43} \pmod{77}$	$M = 9$

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 $A = g^a \bmod p$
Step 5: Receiver computes public value $B = g^b \bmod p$

Step 6: Exchange A and B.
Step 7: Sender computes shared key $K = B^a \bmod p$
Step 8: Receiver computes shared key $K = A^b \bmod p$

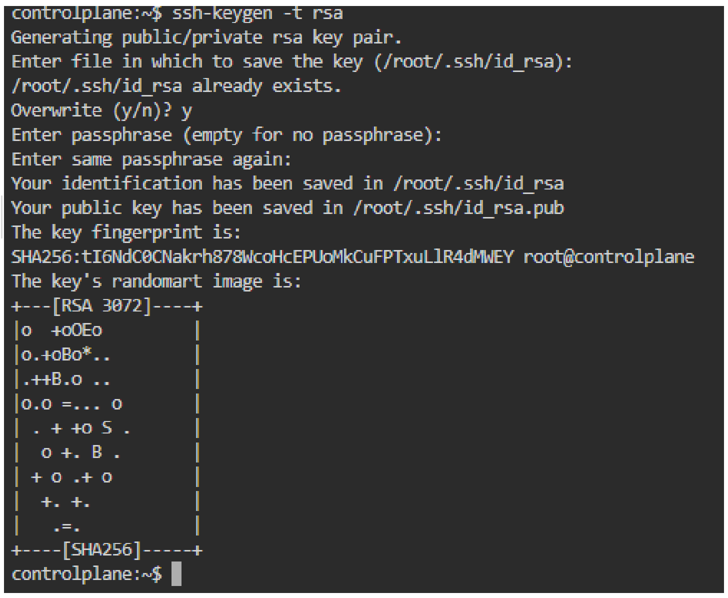
🔢 Diffie–Hellman Numerical Example

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**.

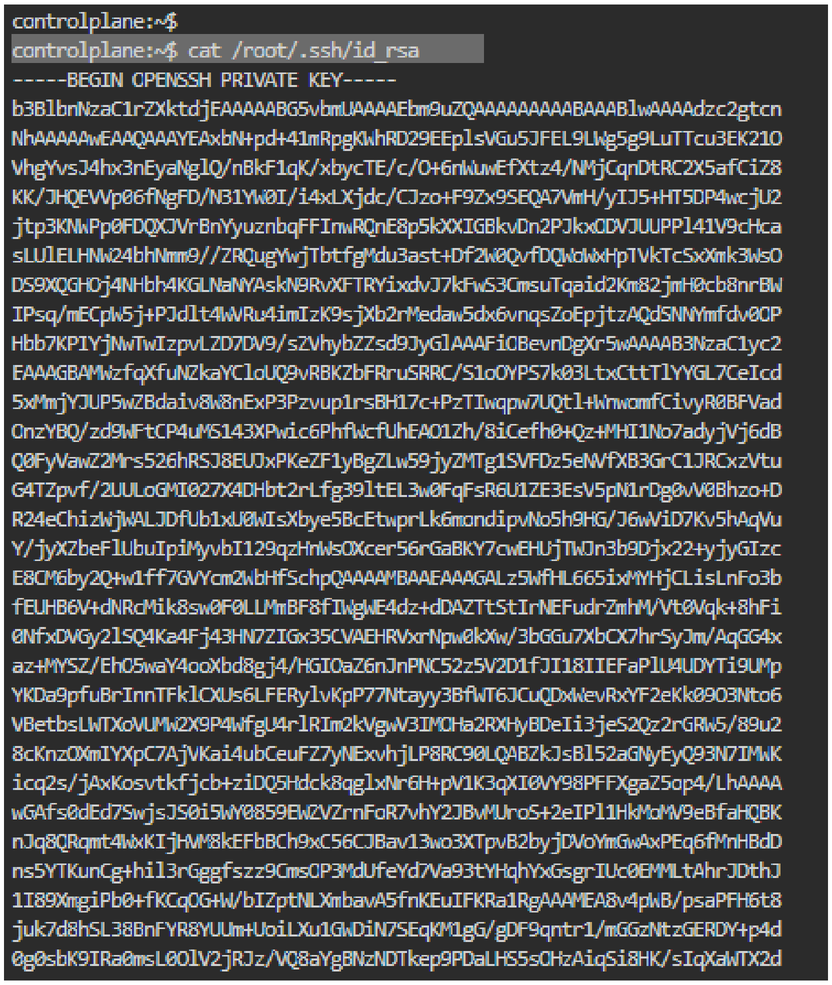
Given:
p = 23, g = 5
Private keys: a = 6, b = 15

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

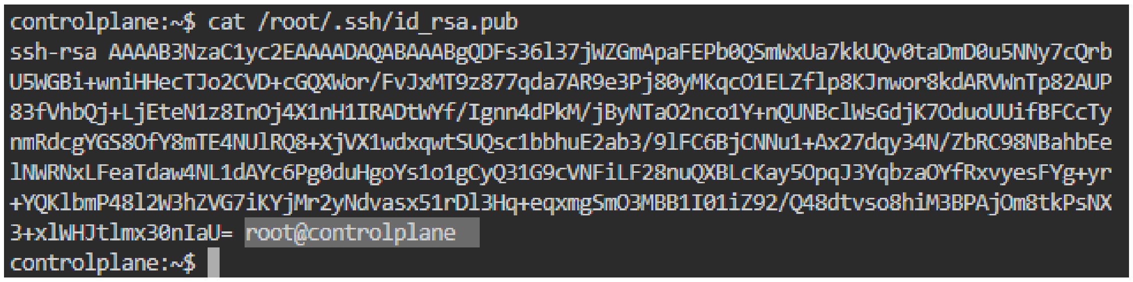
✅ **Shared Secret Key = 2**



```
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
nJq8QRqmt4WxKIjHVM8kEFbBCh9xC56CJBav13wo3XTpvB2byjDVoYmGwAxPEq6fMnHBdD
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**.

Steps

1. Sender generates a random secret key.
2. Encrypts the secret key using receiver's **public key**.
3. Encrypted key is sent over the network.

Purpose

- Combines speed of symmetric encryption with security of asymmetric encryption.
- Used in **hybrid cryptosystems**.

📌 **Example:** SSL/TLS

4. Receiver decrypts it using **private key**. 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 mod p	5^6 mod 23	8

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

🔐 Encryption

Step	**Formula**	**Calculation**	**Result**
C_1	g^k mod p	5^7 mod 23	17
C_2	M × y^k mod p	10 × 8^7 mod 23	21

Ciphertext = (17, 21)

🔒 **Decryption**

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