
# Lab terminal

**Submitted by: Tayyaba Hussein**

**Registration: SP24-BSE-059**

**Course: Information security (lab)**

**Instructor: Ms. Ambreen Gul**

**Program: Software engineering**

## Question no 1:

**Scenario:** You are designing a secure email communication system that ensures confidentiality, integrity, and authenticity of messages. The system uses Elliptic Curve Cryptography (ECC) for encryption and the Digital Signature Algorithm (DSA) for signing messages.

Tasks: 1. Key Generation: 2. Encryption/Decryption using RSA.

3. Digital Signature and Verification using El Gamal.

### Key generation:

```
# ------------------------------------------------------------
print("1) ECC KEY GENERATION")

# Small elliptic curve parameters
p = 17                  # Prime number
G = (5, 1)              # Base point on curve

# Private key
d = 7

# Public key (simplified multiplication)
Q = ((d * G[0]) % p, (d * G[1]) % p)

print("ECC Private Key (d):", d)
print("ECC Public Key (Q):", Q)
print("\n-------------------------------------\n")
```

### Output:

```
1) ECC KEY GENERATION
ECC Private Key (d): 7
ECC Public Key (Q): (1, 7)

-------------------------------------
```

## Encryption/ decryption using RSA:

```python
# 2. RSA ENCRYPTION & DECRYPTION
print("2) RSA ENCRYPTION / DECRYPTION")

# Step 1: Choose two prime numbers
p = 11
q = 13

# Step 2: Compute n
n = p * q

# Step 3: Compute phi(n)
phi = (p - 1) * (q - 1)

# Step 4: Choose public key e
e = 7

# Step 5: Compute private key d
# (Manually chosen for demo)
d = 103   # Because (e * d) mod phi = 1

# Message
M = 9

# Encryption
C = pow(M, e, n)

# Decryption
M_decrypted = pow(C, d, n)

print("Original Message:", M)
print("Encrypted Message:", C)
print("Decrypted Message:", M_decrypted)
```

```
2) RSA ENCRYPTION / DECRYPTION
Original Message: 9
Encrypted Message: 48
Decrypted Message: 9
```

**Digital signature:**

```python
# 3. ELGAMAL DIGITAL SIGNATURE & VERIFICATION
print("3) ELGAMAL DIGITAL SIGNATURE")

# Public parameters
p = 23
g = 5

# Private key
x = 6

# Public key
y = pow(g, x, p)

# Message hash (simulated)
h = 10

# Random value k
k = 7

# Signature generation
r = pow(g, k, p)
k_inverse = pow(k, -1, p - 1)
s = (k_inverse * (h - x * r)) % (p - 1)

print("Signature (r, s):", (r, s))

# Verification
left = pow(g, h, p)
right = (pow(y, r, p) * pow(r, s, p)) % p

print("Verification Left Side:", left)
print("Verification Right Side:", right)

if left == right:
    print("Signature Verified ✓")
else:
    print("Signature Invalid ✗")
```

**Output:**

```
3) ELGAMAL DIGITAL SIGNATURE
Signature (r, s): (17, 12)
Verification Left Side: 9
Verification Right Side: 9
Signature Verified ✓
```

**Question no: 2**

**You have developed a project during this course as your final submission. Using that same project:**

**1. Justify your security method: Why is it suitable or better than other possible methods for your type of project?**

**2. Identify one possible vulnerability or weakness in your current system. How could an attacker misuse it?**

**3. Suggest one realistic improvement to enhance the security of your project. Briefly explain how it would work.**

## 1. Justification of Security Method (ElGamal)

The security method used in my project is **ElGamal cryptography**. ElGamal is suitable for my project because it is based on the **Discrete Logarithm Problem**, which is computationally very difficult to solve.

ElGamal provides the following security features:

- **Confidentiality**: Only the receiver with the private key can decrypt the message.

- **Security against brute-force attacks** due to large prime numbers.
- **Randomness**: Each encryption uses a random value, making the same message produce different cipher texts.

Compared to simpler methods, ElGamal is better because:

- It is more secure than basic symmetric encryption.
- It is widely used in cryptographic systems.
- It provides both **encryption and digital signature support**, making it suitable for secure communication projects.

### 2. Possible Vulnerability or Weakness

One possible vulnerability in my ElGamal system is **poor random number (k) selection**.

If the same random value **k** is reused or chosen weakly:

- An attacker can calculate the private key.
- The attacker can decrypt messages.
- The attacker can forge digital signatures.

This can allow the attacker to **impersonate the legitimate user** and compromise system security.

## 3. Suggested Security Improvement

A realistic improvement is to use a **cryptographically secure random number generator** for generating the random value **k**.

**Working:**

- A secure random generator produces unpredictable values.
- Each encryption or signature uses a unique random **k**.
- This prevents private key leakage.

This improvement increases resistance against:

- Key-recovery attacks
- Signature forgery
- Replay attacks