Name:- Farhan Abbasi

Roll No:- 20P-0044

Section:- 7B

Course:- Information Security

Elgamal encryption algorithm is a public key cryptosystem that provides both encryption & digital signature generation & verification

Code:

```
import random
import hashlib

# function to compute modular exponentiation (base^exp mod mod)
def mod_exp(base, exp, mod):
    result = 1
    while exp > 0:
        if exp % 2 == 1:
            result = (result * base) % mod
        base = (base * base) % mod
        exp //= 2
```

```
    return result

# function to generate prime number
def generate_prime(bits):
    while True:
        num = random.getrandbits(bits)
        if num > 1 and all(num % i
                != 0 for i in range
                (2, int(num**0.5)+1)):
            return num.
# Key Generation
def generate_keys(bits):
    p = generate_prime(bits) # Large prime number
    g = random.randint(2, p-1)
    x = random.randint(1, p-2)
    y = modexp(g, x, p)
    return p, g, x, y
# Encryption
def encrypt(p, g, y, plaintext):
    k = random.randint(1, p-2)
    c1 = modexp(g, k, p)
    s = mod_exp(y, k, p)
    c2 = (plaintext * s) % p
    return c1, c2
```

```
# Decryption
def decrypt(p, x, c1, c2):
    s = mod_exp(c1, x, p)
    s_inverse = pow(s, -1, p)
    plaintext = (c2 * s_inverse) % p
    return plaintext
# Digital Signature Creation
def sign(p, g, x, message):
    k = random.randint(1, p-2)
    r = mod_exp(g, k, p)
    hash_message = int(hashlib.sha256(
        message.encode()).hexdigest(), 16)
    s = (hash_message - x * r) * pow(k, -1, p-1)
        % (p-1)
    return r, s
def verify(p, g, y, message, r, s):
    hash_message = int(hashlib.sha256(messag.encod
                    hexdigest(), 16)
    left = mod_exp(y, r, p) * mod_exp(r, s, p)
                    % p
    right = mod_exp(g, hash_message, p)
    return left == right.
```

Example Usage

```
if __name__ == "__main__":
    p, g, x, y = generate_keys(bits=64)
    # Encrypt a message:
    plain_text = 42
    print("Original message:", plain_text)
    c1, c2 = encrypt(p, g, y, plaintext)
    decrypted_text = decrypt(p, x, c1, c2)
    print("Decrypted message:", decrypted_text)

    # Generate & verify digital signature
    message = "Hello, world!"
    r, s = sign(p, g, x, message)
    is_valid_signature = verify(p, g, y, message, r, s)
    print(is_valid_signature)
```