Department of INFORMATION AND COMMUNICATION ENGINEERING

Faculty of Engineering

(BS CYBER SECURITY AND DIGITAL FORENSIC) FM 1

SEMESTER PROJECT

Subject:

Number Theory and Cryptography

Submitted to:

Dr . ASJAD AMIN

Submitted by:

NAME	*Roll Number*
1) **SHAFAT-E-RASOOL**	**(03028)**
2) MUHAMMAD FAHAD SAEED	**(03015)**
3) TERHEEM AMNA	**(03012)**

ElGamal Encryption Scheme:

An asymmetric encryption scheme based on the difficulty of computing discrete logarithms. The ElGamal encryption scheme is a public-key cryptosystem that provides both encryption and digital signature functionality. It was named after its inventor, Taher ElGamal, and was introduced in the 1980s as an alternative to the widely used 'RSA' encryption algorithm. ElGamal encryption is based on the mathematical properties of modular exponentiation and the difficulty of solving the discrete logarithm problem.

The security of the ElGamal encryption scheme relies on the computational difficulty of solving the discrete logarithm problem, which means finding x given (p, g, and y) in the equation y ≡ g^x (mod p). This problem is believed to be hard, making ElGamal a secure choice for public-key encryption. ElGamal encryption is typically slower and requires longer cipher-texts compared to RSA for the same level of security. It is often used in hybrid encryption systems where a symmetric encryption algorithm (e.g., AES) is used to encrypt the actual message, and ElGamal is used to encrypt the symmetric key.

In addition to encryption, ElGamal can also be used for digital signatures, where the private key signs a message, and the corresponding public key can be used to verify the signature. This makes ElGamal a versatile cryptographic tool for securing communication and data integrity.

`generate_keys` **Function:**

```
def prime(p):
        if p>1:
                for i in range(2,p):
                        if((p%i) ==0):
                                print("number is not prime",p)
                                break
                else:
                        print("number is prime ",p)
        else:
                print("number is not prime ",p)
def root(p):
   for i in range(1, p-1):
      for r in range(1, p-1):
         if(pow (r,i) % p != 1):
           print("primative root is = ",r)
           return r
p=int (input ("enter a larger prime number = "))
```

```
prime(p)
r=root(p)
k = int(input("Enter a secret key (k): "))
a=pow(r,k)%p #now computing a
print( "a is =",a )
```

In this part we check prime number & find primitive root of the prime number and taken secret key for receiver and computing a for encryption.

encrypt Function:

```
print ("encryption")
ks = int (input("enter a random integers for encryption ks= "))
m = int (input("enter a integers for encryption which is less then prime number  m= "))
s=pow(r,ks)%p
print ("s" ,s)
t=(pow(a,ks)*m)%p
print ("t",t)
```

In this part we need a integers for encryption a plain-text and find the value of s and t.

decrypt Function:

```
print ("decryption")
D = ((t * pow(s, -k, p)) % p)
print ("Decrypt message =",D)
print("\nSTUDENT  NAMES                ROLL NUMBER")
print("SHAFAT-E-RASOOL              S22BINCE1M03028")
print("MUHAMMAD FAHAD SAEED         S22BINCE1M03015")
print("TEHREEM AMNA                 S22BINCE1M03012")
```

Only decrypt the message using the value of s, t , k and p.

PROGRAM OF ***ElGamal Encryption Scheme*** in python which are given below.

```
def prime(p):
    if p>1:
        for i in range(2,p):
            if((p%i) ==0):
                print("number is not prime",p)
                break
```

```
                else:
                        print("number is prime ",p)
        else:
                print("number is not prime ",p)
def root(p):
   for i in range(1, p-1):
      for r in range(1, p-1):
         if(pow (r,i) % p != 1):
           print("primative root is = ",r)
           return r
p=int (input ("enter a larger prime number = "))
prime(p)
r=root(p)
k = int(input("Enter a secert key (k): "))
a=pow(r,k)%p #now computing a
print( "a is =",a )
print ("encryption")
ks = int (input("enter a random integres for encryption ks= "))
m = int (input("enter a integres for encryption which is less then prime number  m= "))
s=pow(r,ks)%p
print ("s" ,s)
t=(pow(a,ks)*m)%p
print ("t",t)
print ("decryption")
D = ((t * pow(s, -k, p)) % p)
print ("Decrypt message =",D)
print("\nSTUDENT  NAMES                ROLL NUMBER")
```

print("SHAFAT-E-RASOOL S22BINCE1M03028")

print("MUHAMMAD FAHAD SAEED S22BINCE1M03015")

print("TEHREEM AMNA S22BINCE1M03012")

TESTING

1)

```
┌──(shrs㉿192)-[~]
└─$ python3 12.py
enter a prime number = 11
Public Key (p, g, A): 11 4 3
Private Key (a): 9
Enter an integer to encrypt which is less then your  chosen primer number : 3
Ciphertext (C1, C2): 5 5
Decrypted Plaintext: 3
     NAME                student id
SHAFAT-E-RASOOL        S22BINCE1M03028
MUHAMMAD FAHAD SAEED   S22BINCE1M03015
TEHREEM AMNA           S22BINCE1M03012
```

2)

```
┌──(shrs㉿192)-[~]
└─$ python3 12.py
enter a prime number = 31
Public Key (p, g, A): 31 8 4
Private Key (a): 14
Enter an integer to encrypt which is less then your  chosen primer number : 28
Ciphertext (C1, C2): 1 28
Decrypted Plaintext: 28
     NAME                student id
SHAFAT-E-RASOOL        S22BINCE1M03028
MUHAMMAD FAHAD SAEED   S22BINCE1M03015
TEHREEM AMNA           S22BINCE1M03012
```

3)

```
┌──(shrs㉿192)-[~]
└─$ python3   12.py
enter a prime number = 17
Public Key (p, g, A): 17 2 13
Private Key (a): 6
Enter an integer to encrypt which is less then your  chosen primer number : 15
Ciphertext (C1, C2): 4 2
Decrypted Plaintext: 15
     NAME                  student id
SHAFAT-E-RASOOL          S22BINCE1M03028
MUHAMMAD FAHAD SAEED     S22BINCE1M03015
TEHREEM AMNA             S22BINCE1M03012
```