Kartik Padave

A022

70362019039

Experiment 7

Aim

To implement ElGamal Cipher.

Theory

In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher ElGamal in 1985. ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm (DSA) is a variant of the ElGamal signature scheme, which should not be confused with ElGamal encryption. ElGamal encryption can be defined over any cyclic group G, like multiplicative group of integers modulo n. Its security depends upon the difficulty of a certain problem in G related to computing discrete logarithms. The security of the ElGamal scheme depends on the properties of the underlying group G as well as any padding scheme used on the messages. If the computational Diffie–Hellman assumption (CDH) holds in the underlying cyclic group G, then the encryption function is one-way. If the decisional Diffie–Hellman assumption (DDH) holds in G, then ElGamal achieves semantic security. Semantic security is not implied by the computational Diffie–Hellman assumption alone. See decisional Diffie–Hellman assumption for a discussion of groups where the assumption is believed to hold. ElGamal encryption is unconditionally malleable, and therefore is not secure under chosen ciphertext attack. For example, given an encryption (C1, 2C2) of some (possibly unknown) message 2m, one can easily construct a valid encryption (C1, 2c2) of the message 2m. To achieve chosen-ciphertext security, the scheme must be further modified, or an appropriate padding scheme must be used. Depending on the modification, the DDH assumption may or may not be necessary. Other schemes related to ElGamal which achieve security against chosen ciphertext attacks have also been proposed. The Cramer–Shoup cryptosystem is secure under chosen ciphertext attack assuming DDH holds for G. Its proof does not

use the random oracle model. Another proposed scheme is DHAES, whose proof requires an assumption that is weaker than the DDH assumption.

Code

```
# Key Generation
p = int(input("Enter prime no.: "))
d = int(input("Enter decryption key: "))
e1 = int(input("Enter second part of decryption key: "))
e2 = pow(e1, d) % p
pu_key = [e1, e2, p]

# Encryption
r = int(input("Enter random integer: "))
pt_1 = int(input("Enter Plain Text: "))
c1 = pow(e1, r) % p
c2 = (pt_1 * pow(e2, r)) % p
c = [c1, c2]

# Decryption
i = 1
while True:
    if(i*pow(c1, d) % p == 1):
        break
    i += 1

pt_2 = (c2 * i) % p

print("-----------------------------------------------")
print(f"P = {p}")
print(f"D = {d}")
print(f"E1 = {e1}")
print(f"E2 = {e2}")
print(f"Public Key = {pu_key}")
print(f"R = {r}")
print(f"Plain Text = {pt_1}")
print(f"C1 = {c1}")
print(f"C2 = {c2}")
print(f"Cipher Text = {c}")
print(f"Plain Text: {pt_2}")

if pt_1 == pt_2:
    print("ElGammal Successful !!!")
else:
    print("ElGammal Unsuccessful")
```

Output

```
E:\Programs\College-Labs\CRYPTO-Lab\ProgFiles>python elgamal.py
Enter prime no.: 11
Enter decryption key: 3
Enter second part of decryption key: 2
Enter random integer: 4
Enter Plain Text: 7
-----------------------------------------------
P = 11
D = 3
E1 = 2
E2 = 8
Public Key = [2, 8, 11]
R = 4
Plain Text = 7
C1 = 5
C2 = 6
Cipher Text = [5, 6]
Plain Text: 7
ElGammal Successful !!!
```

Conclusion

Hence, we were able to implement ElGamal.