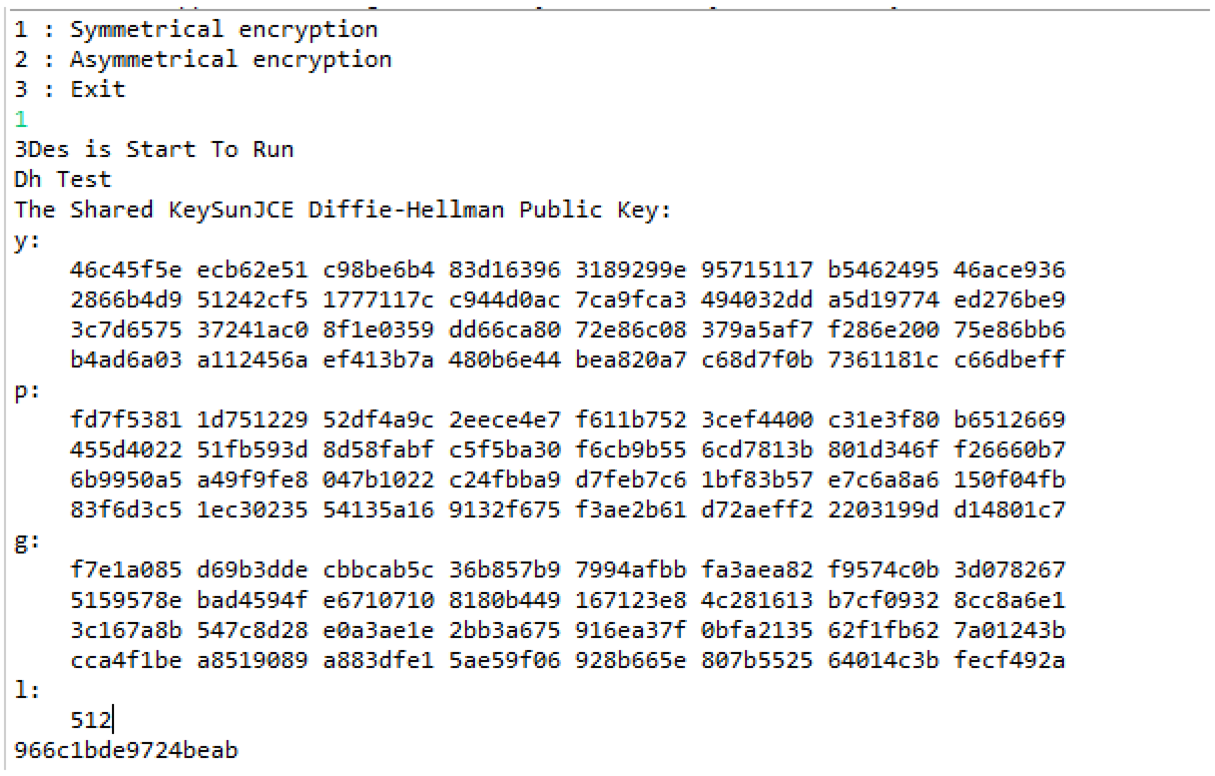
# 471-Cryptography Homework#2

**Ahmet Eroğlu**

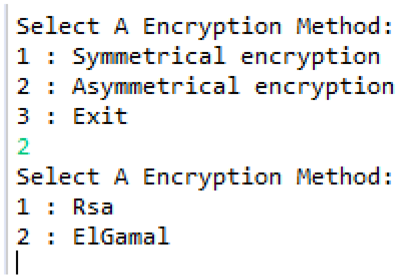
**210201010**



```
1 : Symmetrical encryption
2 : Asymmetrical encryption
3 : Exit
1
3Des is Start To Run
Dh Test
The Shared KeySunJCE Diffie-Hellman Public Key:
y:
    46c45f5e ecb62e51 c98be6b4 83d16396 3189299e 95715117 b5462495 46ace936
    2866b4d9 51242cf5 1777117c c944d0ac 7ca9fca3 494032dd a5d19774 ed276be9
    3c7d6575 37241ac0 8f1e0359 dd66ca80 72e86c08 379a5af7 f286e200 75e86bb6
    b4ad6a03 a112456a ef413b7a 480b6e44 bea820a7 c68d7f0b 7361181c c66dbeff
p:
    fd7f5381 1d751229 52df4a9c 2eece4e7 f611b752 3cef4400 c31e3f80 b6512669
    455d4022 51fb593d 8d58fabf c5f5ba30 f6cb9b55 6cd7813b 801d346f f26660b7
    6b9950a5 a49f9fe8 047b1022 c24fbba9 d7feb7c6 1bf83b57 e7c6a8a6 150f04fb
    83f6d3c5 1ec30235 54135a16 9132f675 f3ae2b61 d72aeff2 2203199d d14801c7
g:
    f7e1a085 d69b3dde cbbcab5c 36b857b9 7994afbb fa3aea82 f9574c0b 3d078267
    5159578e bad4594f e6710710 8180b449 167123e8 4c281613 b7cf0932 8cc8a6e1
    3c167a8b 547c8d28 e0a3ae1e 2bb3a675 916ea37f 0bfa2135 62f1fb62 7a01243b
    cca4f1be a8519089 a883dfe1 5ae59f06 928b665e 807b5525 64014c3b fecf492a
l:
    512
966c1bde9724beab
```

**Symmetrical Encryption Output Example ( 3DES )**

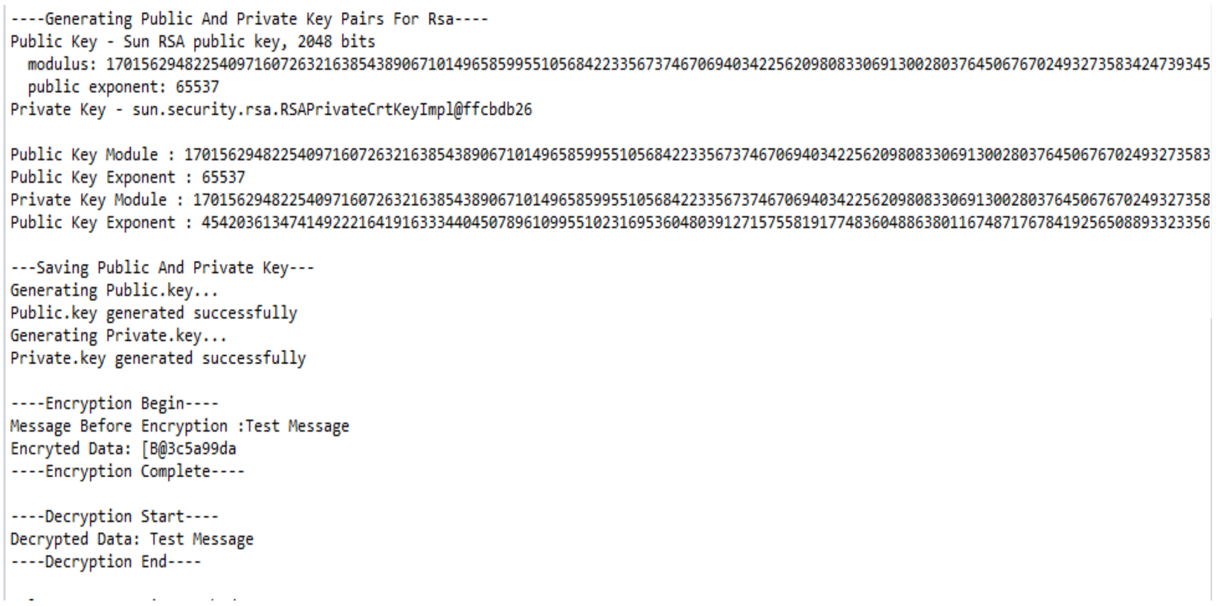
First I find symmetric encryption 3DES and implement my system. Then write some code for Diffie-Hellman Key Exchange algorithm.



```
Select A Encryption Method:
1 : Symmetrical encryption
2 : Asymmetrical encryption
3 : Exit
2
Select A Encryption Method:
1 : Rsa
2 : ElGamal
```

## Menu

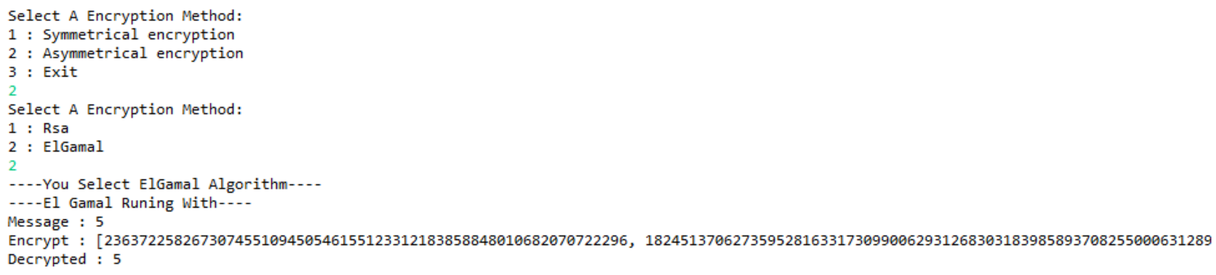
I implement a menu that users can be select and use which encryption method will use.



```
----Generating Public And Private Key Pairs For Rsa----
Public Key - Sun RSA public key, 2048 bits
  modulus: 1701562948225409716072632163854389067101496585995510568422335673746706940342256209808330691300280376450676702493273583424739345
  public exponent: 65537
Private Key - sun.security.rsa.RSAPrivateCrtKeyImpl@ffcbdb26

Public Key Module : 1701562948225409716072632163854389067101496585995510568422335673746706940342256209808330691300280376450676702493273583
Public Key Exponent : 65537
Private Key Module : 1701562948225409716072632163854389067101496585995510568422335673746706940342256209808330691300280376450676702493273583
Public Key Exponent : 4542036134741492221641916333440450789610995510231695360480391271575581917748360488638011674871767841925650889332335 6

---Saving Public And Private Key---
Generating Public.key...
Public.key generated successfully
Generating Private.key...
Private.key generated successfully

----Encryption Begin----
Message Before Encryption :Test Message
Encryted Data: [B@3c5a99da
----Encryption Complete----

----Decryption Start----
Decrypted Data: Test Message
----Decryption End----
```

**Rsa Example Output**

In the implementation on Rsa, first user create a public key and share it. Then the other user encrypted it with public key. The owner of the key decrypt the message with his own private key so owner of key can be read the message.



```
Select A Encryption Method:
1 : Symmetrical encryption
2 : Asymmetrical encryption
3 : Exit
2
Select A Encryption Method:
1 : Rsa
2 : ElGamal
2
----You Select ElGamal Algorithm----
----El Gamal Runing With----
Message : 5
Encrypt : [2363722582673074551094505461551233121838588480106820707222 96, 1824513706273595281633173099006293126830318398589370825500063128 9
Decrypted : 5
```

**ElGamal Example Output**