T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4057 Information System Security

PROJECT REPORT

Title of the Project

“Homework 1”

April 21 2024

Group Members

Aimen Daddi 150119879

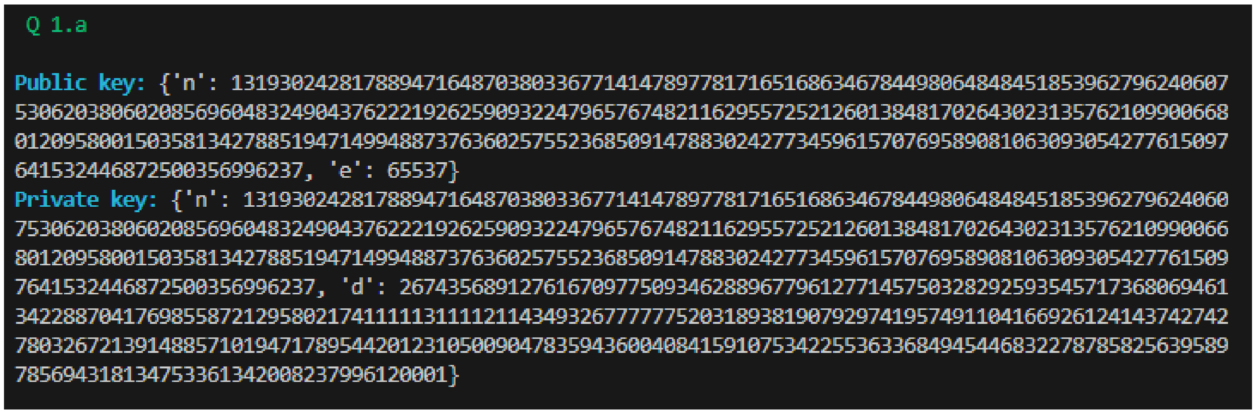
For this project we used Python Programming language to implement the following:

1. Generation of public-private key pairs.
2. Generation of Symmetric keys
3. Generation and Verification of Digital Signature
4. AES Encryption
5. Message Authentication Codes

To do that we used tools from Crypto and cryptography libraires.

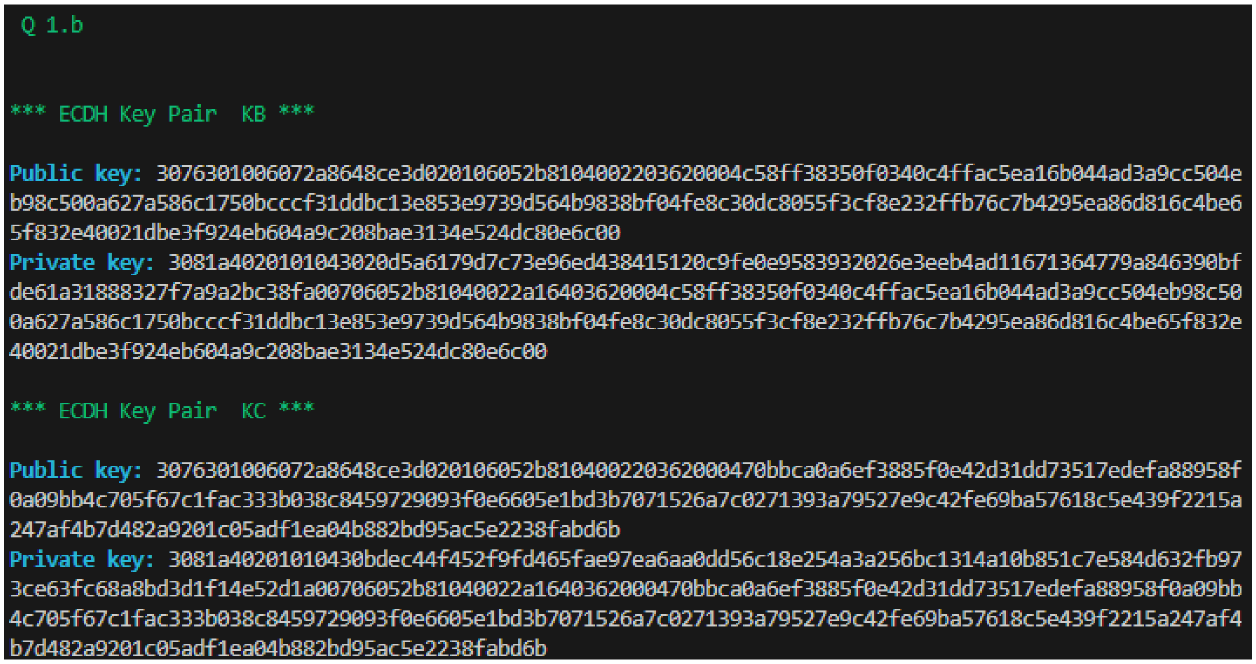
1) Generation of public-private key pairs

Created 1024 bit RSA key pair using Crypto library.



```
 Q 1.a

Public key: {'n': 13193024281788947164870380336771414789778171651686346784498064848451853962796240607
53062038060208569604832490437622219262590932247965767482116295572521260138481702643023135762109900668
01209580015035813427885194714994887376360257552368509147883024277345961570769589081063093054277615097
641532446872500356996237, 'e': 65537}
Private key: {'n': 1319302428178894716487038033677141478977817165168634678449806484845185396279624060
75306203806020856960483249043762221926259093224796576748211629557252126013848170264302313576210990066
80120958001503581342788519471499488737636025755236850914788302427734596157076958908106309305427761509
7641532446872500356996237, 'd': 267435689127616709775093462889677961277145750328292593545717368069461
34228870417698558721295802174111113111121143493267777775203189381907929741957491104166926124143742742
78032672139148857101947178954420123105009047835943600408415910753422553633684945446832278785825639589
785694318134753361342008237996120001}
```

Created ECDH keypair using SECP384R1 curve.

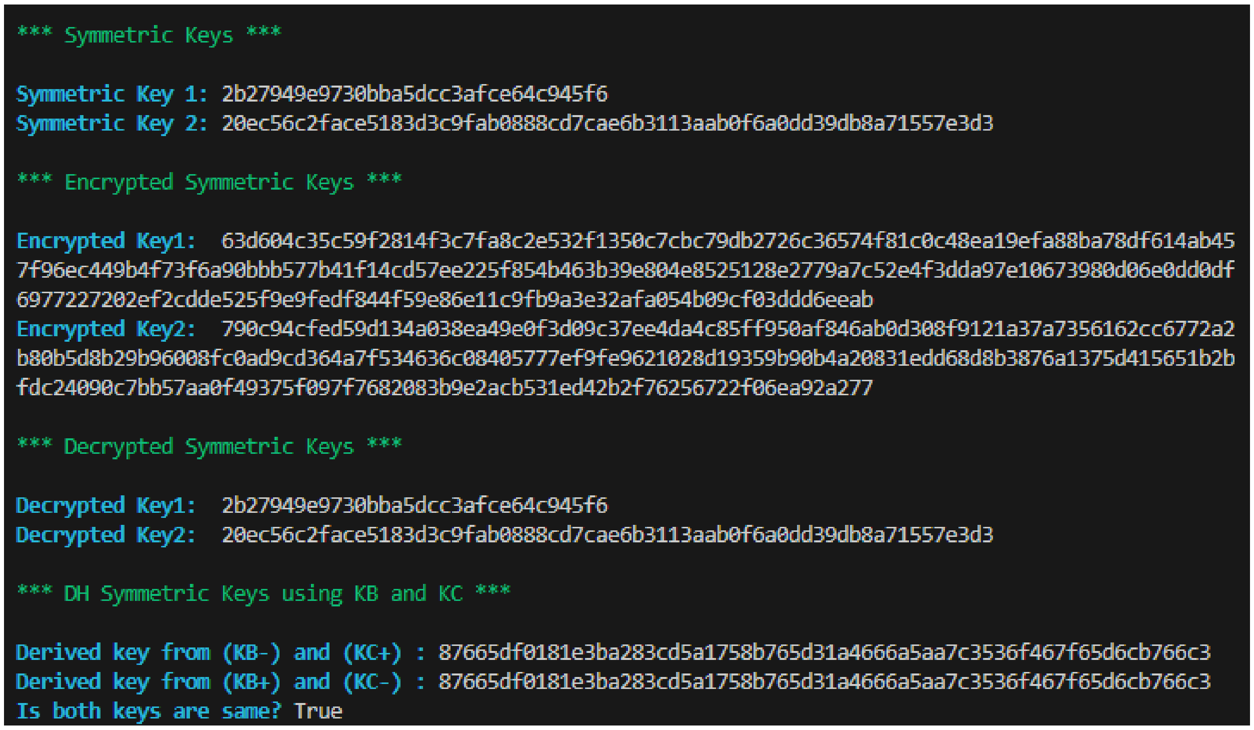


```
 Q 1.b


*** ECDH Key Pair  KB ***

Public key: 3076301006072a8648ce3d020106052b8104002203620004c58ff38350f0340c4ffac5ea16b044ad3a9cc504e
b98c500a627a586c1750bcccf31ddbc13e853e9739d564b9838bf04fe8c30dc8055f3cf8e232ffb76c7b4295ea86d816c4be6
5f832e40021dbe3f924eb604a9c208bae3134e524dc80e6c00
Private key: 3081a4020101043020d5a6179d7c73e96ed438415120c9fe0e9583932026e3eeb4ad11671364779a846390bf
de61a31888327f7a9a2bc38fa00706052b81040022a16403620004c58ff38350f0340c4ffac5ea16b044ad3a9cc504eb98c50
0a627a586c1750bcccf31ddbc13e853e9739d564b9838bf04fe8c30dc8055f3cf8e232ffb76c7b4295ea86d816c4be65f832e
40021dbe3f924eb604a9c208bae3134e524dc80e6c00

*** ECDH Key Pair  KC ***

Public key: 3076301006072a8648ce3d020106052b81040022036200047 0bbca0a6ef3885f0e42d31dd73517edefa88958f
0a09bb4c705f67c1fac333b038c8459729093f0e6605e1bd3b7071526a7c0271393a79527e9c42fe69ba57618c5e439f2215a
247af4b7d482a9201c05adf1ea04b882bd95ac5e2238fabd6b
Private key: 3081a40201010430bdec44f452f9fd465fae97ea6aa0dd56c18e254a3a256bc1314a10b851c7e584d632fb97
3ce63fc68a8bd3d1f14e52d1a00706052b81040022a164036200047 0bbca0a6ef3885f0e42d31dd73517edefa88958f0a09bb
4c705f67c1fac333b038c8459729093f0e6605e1bd3b7071526a7c0271393a79527e9c42fe69ba57618c5e439f2215a247af4
b7d482a9201c05adf1ea04b882bd95ac5e2238fabd6b
```

2) Generation of Symmetric keys

Created one 128-bit and one 256-bit symmetric key using PBKDF2. Then we encrypt them with Ka+ and decrypt them with Ka-.

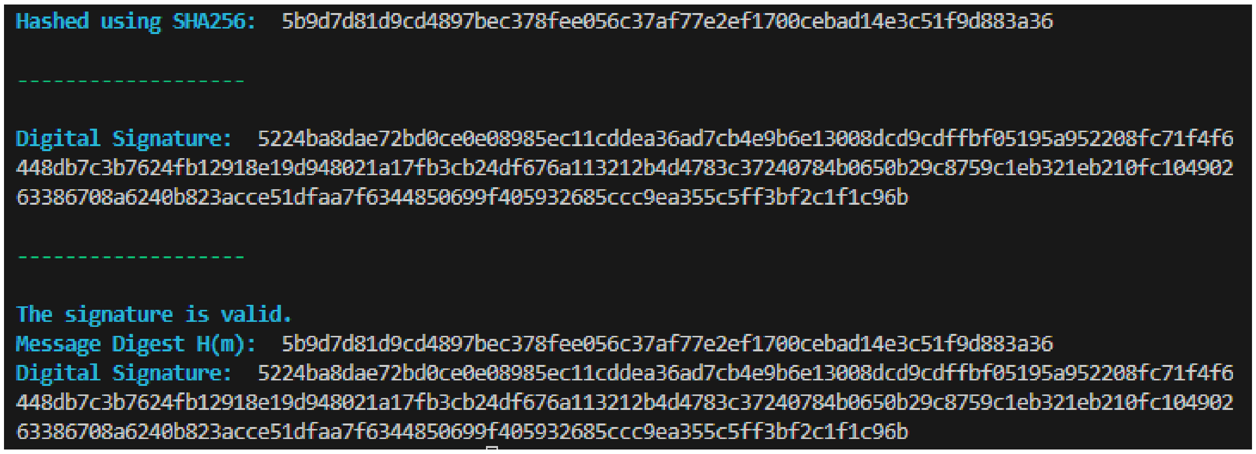
For part b we created a 256-bit symmetric key using DH with Kc+ and Kb-. Then we created another using Kb+ and Kc- and compared if they are the same.



```
*** Symmetric Keys ***

Symmetric Key 1: 2b27949e9730bba5dcc3afce64c945f6
Symmetric Key 2: 20ec56c2face5183d3c9fab0888cd7cae6b3113aab0f6a0dd39db8a71557e3d3

*** Encrypted Symmetric Keys ***

Encrypted Key1:  63d604c35c59f2814f3c7fa8c2e532f1350c7cbc79db2726c36574f81c0c48ea19efa88ba78df614ab45
7f96ec449b4f73f6a90bbb577b41f14cd57ee225f854b463b39e804e8525128e2779a7c52e4f3dda97e10673980d06e0dd0df
6977227202ef2cdde525f9e9fedf844f59e86e11c9fb9a3e32afa054b09cf03ddd6eeab
Encrypted Key2:  790c94cfed59d134a038ea49e0f3d09c37ee4da4c85ff950af846ab0d308f9121a37a7356162cc6772a2
b80b5d8b29b96008fc0ad9cd364a7f534636c08405777ef9fe9621028d19359b90b4a20831edd68d8b3876a1375d415651b2b
fdc24090c7bb57aa0f49375f097f7682083b9e2acb531ed42b2f76256722f06ea92a277

*** Decrypted Symmetric Keys ***

Decrypted Key1:  2b27949e9730bba5dcc3afce64c945f6
Decrypted Key2:  20ec56c2face5183d3c9fab0888cd7cae6b3113aab0f6a0dd39db8a71557e3d3

*** DH Symmetric Keys using KB and KC ***

Derived key from (KB-) and (KC+) : 87665df0181e3ba283cd5a1758b765d31a4666a5aa7c3536f467f65d6cb766c3
Derived key from (KB+) and (KC-) : 87665df0181e3ba283cd5a1758b765d31a4666a5aa7c3536f467f65d6cb766c3
Is both keys are same? True
```

3) Generation and Verification of Digital Signature

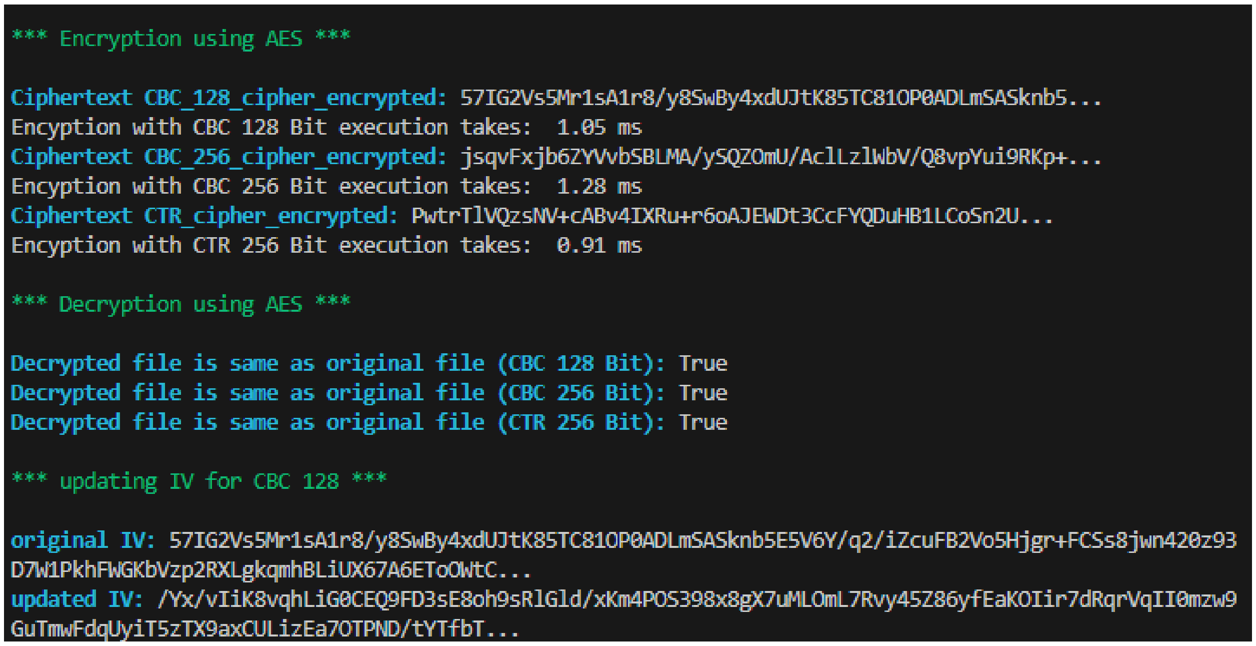
We applied SHA256 Hash algorithm to the contents of a image file. Then encrypted it using Ka- to get a digital signature. Then we verify it.



```
Hashed using SHA256:  5b9d7d81d9cd4897bec378fee056c37af77e2ef1700cebad14e3c51f9d883a36

-------------------

Digital Signature:  5224ba8dae72bd0ce0e08985ec11cddea36ad7cb4e9b6e13008dcd9cdffbf05195a952208fc71f4f6
448db7c3b7624fb12918e19d948021a17fb3cb24df676a113212b4d4783c37240784b0650b29c8759c1eb321eb210fc104902
63386708a6240b823acce51dfaa7f6344850699f405932685ccc9ea355c5ff3bf2c1f1c96b

-------------------

The signature is valid.
Message Digest H(m):  5b9d7d81d9cd4897bec378fee056c37af77e2ef1700cebad14e3c51f9d883a36
Digital Signature:  5224ba8dae72bd0ce0e08985ec11cddea36ad7cb4e9b6e13008dcd9cdffbf05195a952208fc71f4f6
448db7c3b7624fb12918e19d948021a17fb3cb24df676a113212b4d4783c37240784b0650b29c8759c1eb321eb210fc104902
63386708a6240b823acce51dfaa7f6344850699f405932685ccc9ea355c5ff3bf2c1f1c96b
```

4) AES Encryption

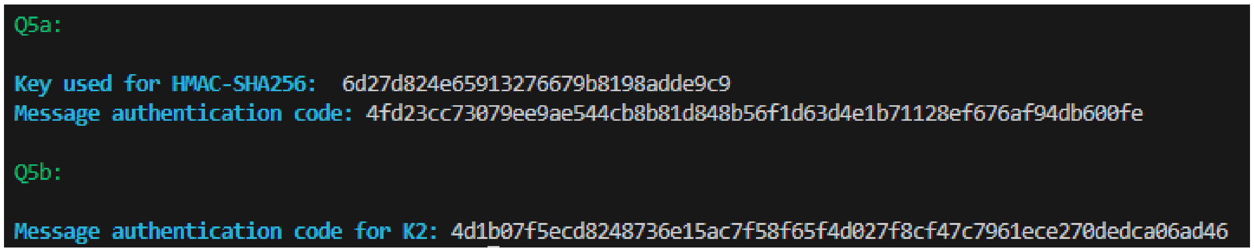
For this question we compare encryption time for i) AES (128 bit key) in CBC mode, ii) AES (256 bit key) in CBC mode, iii) AES (256 bit key) in CTR mode. We notice that CTR is faster at 0.91ms.



```
*** Encryption using AES ***

Ciphertext CBC_128_cipher_encrypted: 57IG2Vs5Mr1sA1r8/y8SwBy4xdUJtK85TC81OP0ADLmSASknb5...
Encyption with CBC 128 Bit execution takes:  1.05 ms
Ciphertext CBC_256_cipher_encrypted: jsqvFxjb6ZYVvbSBLMA/ySQZOmU/AclLzlWbV/Q8vpYui9RKp+...
Encyption with CBC 256 Bit execution takes:  1.28 ms
Ciphertext CTR_cipher_encrypted: PwtrTlVQzsNV+cABv4IXRu+r6oAJEWDt3CcFYQDuHB1LCoSn2U...
Encyption with CTR 256 Bit execution takes:  0.91 ms

*** Decryption using AES ***

Decrypted file is same as original file (CBC 128 Bit): True
Decrypted file is same as original file (CBC 256 Bit): True
Decrypted file is same as original file (CTR 256 Bit): True

*** updating IV for CBC 128 ***

original IV: 57IG2Vs5Mr1sA1r8/y8SwBy4xdUJtK85TC81OP0ADLmSASknb5E5V6Y/q2/iZcuFB2Vo5Hjgr+FCSs8jwn420z93
D7W1PkhFWGKbVzp2RXLgkqmhBLiUX67A6EToOWtC...
updated IV: /Yx/vIiK8vqhLiG0CEQ9FD3sE8oh9sRlGld/xKm4POS398x8gX7uMLOmL7Rvy45Z86yfEaKOIir7dRqrVqII0mzw9
GuTmwFdqUyiT5zTX9axCULizEa7OTPND/tYTfbT...
```

5) Message Authentication Codes

Used “message” as text message then generated a HMAC-SHA256 using K1. Then we used that to apply HMAC-SHA256 to K2.



```
Q5a:

Key used for HMAC-SHA256:  6d27d824e65913276679b8198adde9c9
Message authentication code: 4fd23cc73079ee9ae544cb8b81d848b56f1d63d4e1b71128ef676af94db600fe

Q5b:

Message authentication code for K2: 4d1b07f5ecd8248736e15ac7f58f65f4d027f8cf47c7961ece270dedca06ad46
```