Introduction to Cryptography Techniques

Assignment-1

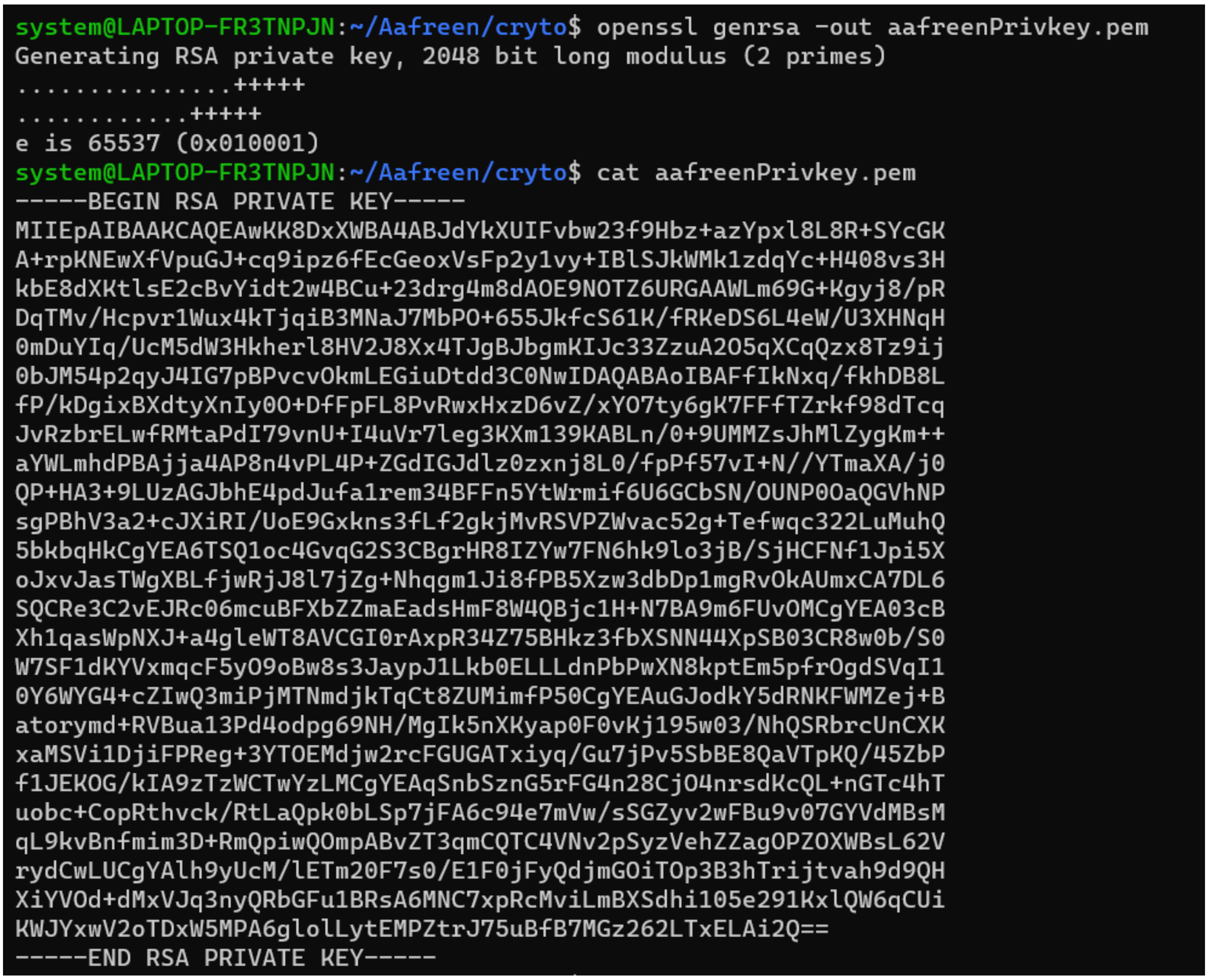
Name: AAFREEN.M

Reg.No:205001301

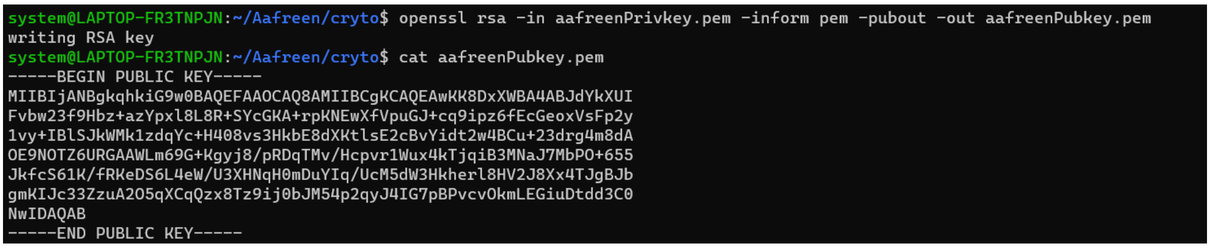
Question No: I

a. Implement one Key Generation algorithm and generate the key pairs and evaluate their performance in terms of Computational Time, Computational complexity. Identity the limitations of each algorithm

We are using RSA algorithm to create a private key, public key pair



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl genrsa -out aafreenPrivkey.pem
Generating RSA private key, 2048 bit long modulus (2 primes)
...............+++++
............+++++
e is 65537 (0x010001)
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat aafreenPrivkey.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAwKK8DxXWBA4ABJdYkXUIFvbw23f9Hbz+azYpxl8L8R+SYcGK
A+rpKNEwXfVpuGJ+cq9ipz6fEcGeoxVsFp2y1vy+IBlSJkWMk1zdqYc+H408vs3H
kbE8dXKtlsE2cBvYidt2w4BCu+23drg4m8dAOE9NOTZ6URGAAWLm69G+Kgyj8/pR
DqTMv/Hcpvr1Wux4kTjqiB3MNaJ7MbPO+655JkfcS61K/fRKeDS6L4eW/U3XHNqH
0mDuYIq/UcM5dW3Hkherl8HV2J8Xx4TJgBJbgmKIJc33ZzuA2O5qXCqQzx8Tz9ij
0bJM54p2qyJ4IG7pBPvcvOkmLEGiuDtdd3C0NwIDAQABAoIBAFfIkNxq/fkhDB8L
fP/kDgixBXdtyXnIy0O+DfFpFL8PvRwxHxzD6vZ/xYO7ty6gK7FFfTZrkf98dTcq
JvRzbrELwfRMtaPdI79vnU+I4uVr7leg3KXm139KABLn/0+9UMMZsJhMlZygKm++
aYWLmhdPBAjja4AP8n4vPL4P+ZGdIGJdlz0zxnj8L0/fpPf57vI+N//YTmaXA/j0
QP+HA3+9LUzAGJbhE4pdJufa1rem34BFFn5YtWrmif6U6GCbSN/OUNP0OaQGVhNP
sgPBhV3a2+cJXiRI/UoE9Gxkns3fLf2gkjMvRSVPZWvac52g+Tefwqc322LuMuhQ
5bkbqHkCgYEA6TSQ1oc4GvqG2S3CBgrHR8IZYw7FN6hk9lo3jB/SjHCFNf1Jpi5X
oJxvJasTWgXBLfjwRjJ8l7jZg+Nhqgm1Ji8fPB5Xzw3dbDp1mgRvOkAUmxCA7DL6
SQCRe3C2vEJRc06mcuBFXbZZmaEadsHmF8W4QBjc1H+N7BA9m6FUvOMCgYEA03cB
Xh1qasWpNXJ+a4gleWT8AVCGI0rAxpR34Z75BHkz3fbXSNN44XpSB03CR8w0b/S0
W7SF1dKYVxmqcF5yO9oBw8s3JaypJ1Lkb0ELLLdnPbPwXN8kptEm5pfrOgdSVqI1
0Y6WYG4+cZIwQ3miPjMTNmdjkTqCt8ZUMimfP50CgYEAuGJodkY5dRNKFWMZej+B
atorymd+RVBua13Pd4odpg69NH/MgIk5nXKyap0F0vKj195w03/NhQSRbrcUnCXK
xaMSVi1DjiFPReg+3YTOEMdjw2rcFGUGATxiyq/Gu7jPv5SbBE8QaVTpKQ/45ZbP
f1JEKOG/kIA9zTzWCTwYzLMCgYEAqSnbSznG5rFG4n28CjO4nrsdKcQL+nGTc4hT
uobc+CopRthvck/RtLaQpk0bLSp7jFA6c94e7mVw/sSGZyv2wFBu9v07GYVdMBsM
qL9kvBnfmim3D+RmQpiwQOmpABvZT3qmCQTC4VNv2pSyzVehZZagOPZOXWBsL62V
rydCwLUCgYAlh9yUcM/lETm20F7s0/E1F0jFyQdjmGOiTOp3B3hTrijtvah9d9QH
XiYVOd+dMxVJq3nyQRbGFu1BRsA6MNC7xpRcMviLmBXSdhi105e291KxlQW6qCUi
KWJYxwV2oTDxW5MPA6glolLytEMPZtrJ75uBfB7MGz262LTxELAi2Q==
-----END RSA PRIVATE KEY-----
```



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl rsa -in aafreenPrivkey.pem -inform pem -pubout -out aafreenPubkey.pem
writing RSA key
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat aafreenPubkey.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwKK8DxXWBA4ABJdYkXUI
Fvbw23f9Hbz+azYpxl8L8R+SYcGKA+rpKNEwXfVpuGJ+cq9ipz6fEcGeoxVsFp2y
1vy+IBlSJkWMk1zdqYc+H408vs3HkbE8dXKtlsE2cBvYidt2w4BCu+23drg4m8dA
OE9NOTZ6URGAAWLm69G+Kgyj8/pRDqTMv/Hcpvr1Wux4kTjqiB3MNaJ7MbPO+655
JkfcS61K/fRKeDS6L4eW/U3XHNqH0mDuYIq/UcM5dW3Hkherl8HV2J8Xx4TJgBJb
gmKIJc33ZzuA2O5qXCqQzx8Tz9ij0bJM54p2qyJ4IG7pBPvcvOkmLEGiuDtdd3C0
NwIDAQAB
-----END PUBLIC KEY-----
```

This is a public-key generation algorithm. Other examples are Elliptic Curve Digital Signature Algorithm, Edwards-curve Digital Signature Algorithm (EdDSA)

Compared to Elliptic Curve Digital Signature Algorithm, RSA is been for a long time and have been tested more than ECDSA. It is well studied and audited algorithm.

RSA is simpler compared to other algorithms. It is easy to implement in a public key infrastructure.

But ECDSA required shorter key length compared to RSA

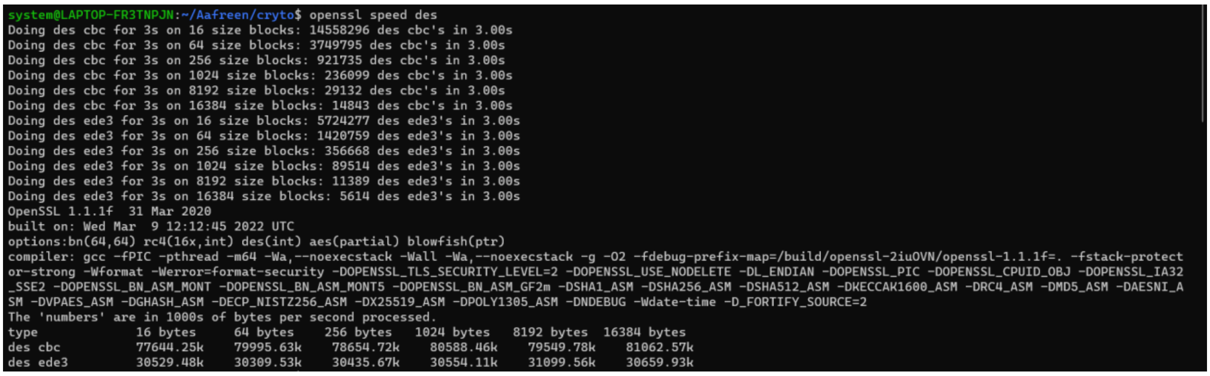
Private key algorithms is a single key infrastructure. A single key is used for communication.

Same key is used to encrypt and decrypt the message. There are many private key algorithms such as AES , DES.

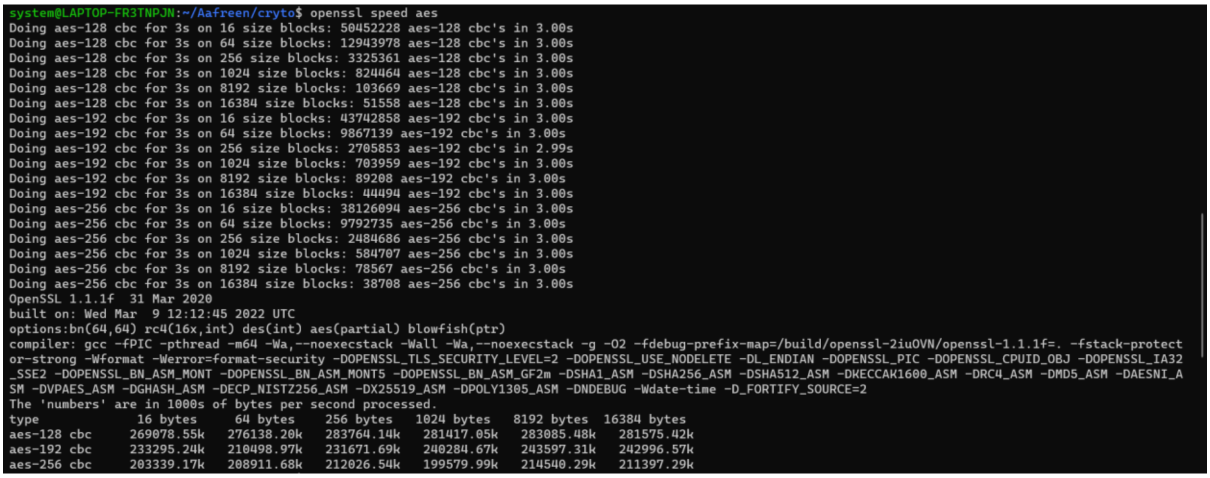
Public key cryptography is used to share the keys of private key cryptography.

Private key cryptography is faster than public key cryptography. Hence public key cryptography should be used to exchange keys and further communication should proceeded with private key cryptography.

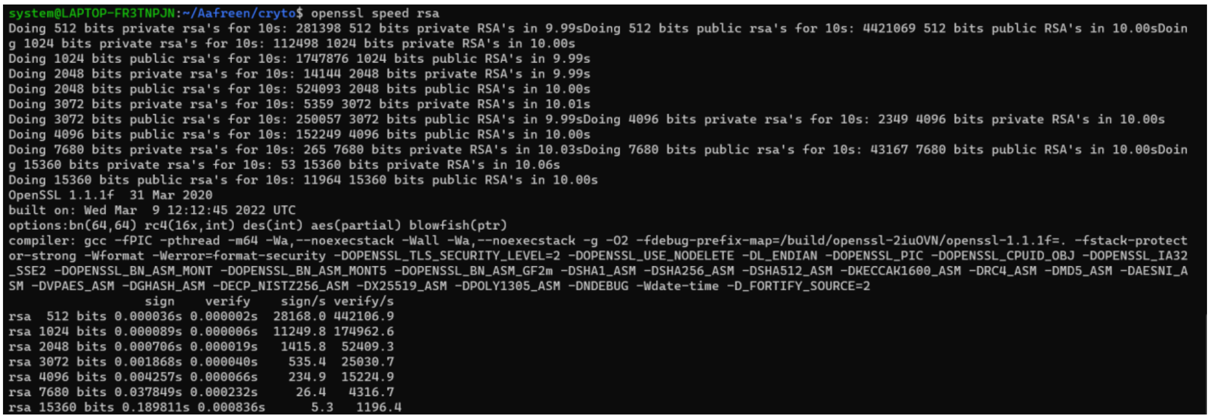
SPEED



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl speed des
Doing des cbc for 3s on 16 size blocks: 14558296 des cbc's in 3.00s
Doing des cbc for 3s on 64 size blocks: 3749795 des cbc's in 3.00s
Doing des cbc for 3s on 256 size blocks: 921735 des cbc's in 3.00s
Doing des cbc for 3s on 1024 size blocks: 236099 des cbc's in 3.00s
Doing des cbc for 3s on 8192 size blocks: 29132 des cbc's in 3.00s
Doing des cbc for 3s on 16384 size blocks: 14843 des cbc's in 3.00s
Doing des ede3 for 3s on 16 size blocks: 5724277 des ede3's in 3.00s
Doing des ede3 for 3s on 64 size blocks: 1420759 des ede3's in 3.00s
Doing des ede3 for 3s on 256 size blocks: 356668 des ede3's in 3.00s
Doing des ede3 for 3s on 1024 size blocks: 89514 des ede3's in 3.00s
Doing des ede3 for 3s on 8192 size blocks: 11389 des ede3's in 3.00s
Doing des ede3 for 3s on 16384 size blocks: 5614 des ede3's in 3.00s
OpenSSL 1.1.1f  31 Mar 2020
built on: Wed Mar  9 12:12:45 2022 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -fdebug-prefix-map=/build/openssl-2iuOVN/openssl-1.1.1f=. -fstack-protect
or-strong -Wformat -Werror=format-security -DOPENSSL_TLS_SECURITY_LEVEL=2 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32
_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_A
SM -DVPAES_ASM -DGHASH_ASM -DECP_NISTZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
des cbc          77644.25k    79995.63k    78654.72k    80588.46k    79549.78k    81062.57k
des ede3         30529.48k    30309.53k    30435.67k    30554.11k    31099.56k    30659.93k
```



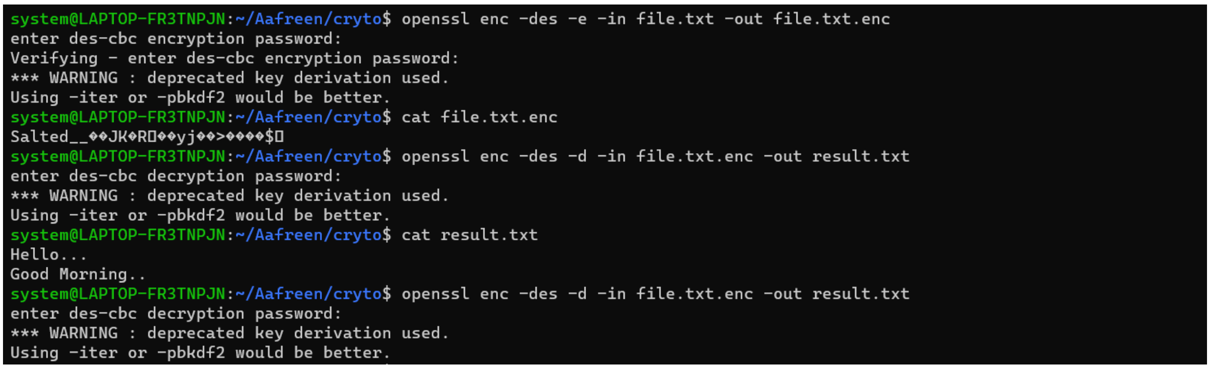
```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl speed aes
Doing aes-128 cbc for 3s on 16 size blocks: 50452228 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 64 size blocks: 12943978 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 256 size blocks: 3325361 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 1024 size blocks: 824464 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 8192 size blocks: 103669 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 16384 size blocks: 51558 aes-128 cbc's in 3.00s
Doing aes-192 cbc for 3s on 16 size blocks: 43742858 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 64 size blocks: 9867139 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 256 size blocks: 2705853 aes-192 cbc's in 2.99s
Doing aes-192 cbc for 3s on 1024 size blocks: 703959 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 8192 size blocks: 89208 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 16384 size blocks: 44494 aes-192 cbc's in 3.00s
Doing aes-256 cbc for 3s on 16 size blocks: 38126094 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 64 size blocks: 9792735 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 256 size blocks: 2484686 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 1024 size blocks: 584707 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 8192 size blocks: 78567 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 16384 size blocks: 38708 aes-256 cbc's in 3.00s
OpenSSL 1.1.1f  31 Mar 2020
built on: Wed Mar  9 12:12:45 2022 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -fdebug-prefix-map=/build/openssl-2iuOVN/openssl-1.1.1f=. -fstack-protect
or-strong -Wformat -Werror=format-security -DOPENSSL_TLS_SECURITY_LEVEL=2 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32
_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_A
SM -DVPAES_ASM -DGHASH_ASM -DECP_NISTZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-128 cbc     269078.55k   276138.20k   283764.14k   281417.05k   283085.48k   281575.42k
aes-192 cbc     233295.24k   210498.97k   231671.69k   240284.67k   243597.31k   242996.57k
aes-256 cbc     203339.17k   208911.68k   212026.54k   199579.99k   214540.29k   211397.29k
```



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl speed rsa
Doing 512 bits private rsa's for 10s: 281398 512 bits private RSA's in 9.99sDoing 512 bits public rsa's for 10s: 4421069 512 bits public RSA's in 10.00sDoin
g 1024 bits private rsa's for 10s: 112498 1024 bits private RSA's in 10.00s
Doing 1024 bits public rsa's for 10s: 1747876 1024 bits public RSA's in 9.99s
Doing 2048 bits private rsa's for 10s: 14144 2048 bits private RSA's in 9.99s
Doing 2048 bits public rsa's for 10s: 524093 2048 bits public RSA's in 10.00s
Doing 3072 bits private rsa's for 10s: 5359 3072 bits private RSA's in 10.01s
Doing 3072 bits public rsa's for 10s: 250057 3072 bits public RSA's in 9.99sDoing 4096 bits private rsa's for 10s: 2349 4096 bits private RSA's in 10.00s
Doing 4096 bits public rsa's for 10s: 152249 4096 bits public RSA's in 10.00s
Doing 7680 bits private rsa's for 10s: 265 7680 bits private RSA's in 10.03sDoing 7680 bits public rsa's for 10s: 43167 7680 bits public RSA's in 10.00sDoin
g 15360 bits private rsa's for 10s: 53 15360 bits private RSA's in 10.06s
Doing 15360 bits public rsa's for 10s: 11964 15360 bits public RSA's in 10.00s
OpenSSL 1.1.1f  31 Mar 2020
built on: Wed Mar  9 12:12:45 2022 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -fdebug-prefix-map=/build/openssl-2iuOVN/openssl-1.1.1f=. -fstack-protect
or-strong -Wformat -Werror=format-security -DOPENSSL_TLS_SECURITY_LEVEL=2 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32
_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_A
SM -DVPAES_ASM -DGHASH_ASM -DECP_NISTZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
                  sign    verify    sign/s verify/s
rsa  512 bits 0.000036s 0.000002s  28168.0 442106.9
rsa 1024 bits 0.000089s 0.000006s  11249.8 174962.6
rsa 2048 bits 0.000706s 0.000019s   1415.8  52409.3
rsa 3072 bits 0.001868s 0.000040s    535.4  25030.7
rsa 4096 bits 0.004257s 0.000066s    234.9  15224.9
rsa 7680 bits 0.037849s 0.000232s     26.4   4316.7
rsa 15360 bits 0.189811s 0.000836s      5.3   1196.4
```

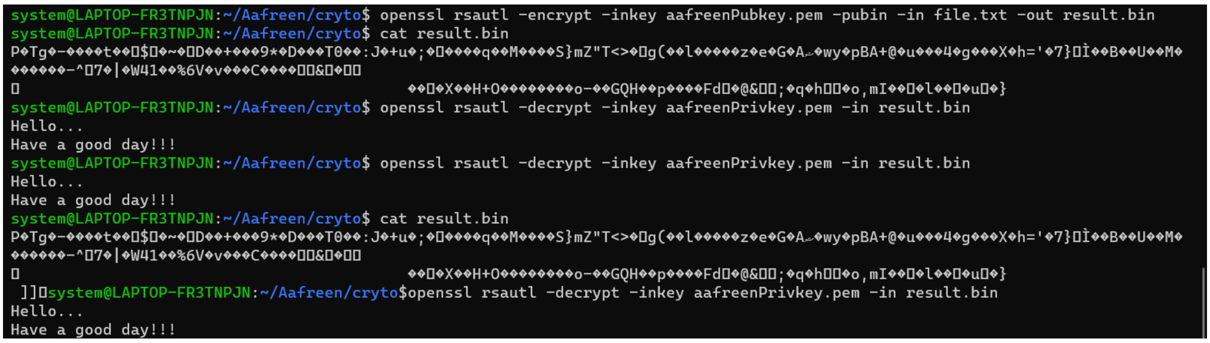
b) Using the above keys, encrypt a given message and evaluate various techniques to ensure the integrity of the data. Identity the deficiency of each approach.

Using DES to encrypt and decrypt



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl enc -des -e -in file.txt -out file.txt.enc
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat file.txt.enc
Salted__��JK�R▯��yj��>����$▯
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl enc -des -d -in file.txt.enc -out result.txt
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat result.txt
Hello...
Good Morning..
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl enc -des -d -in file.txt.enc -out result.txt
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Using RSA encryption

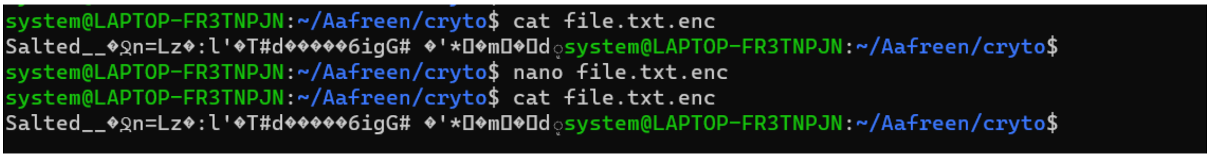


```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl rsautl -encrypt -inkey aafreenPubkey.pem -pubin -in file.txt -out result.bin
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat result.bin
P�Tg�-����t��▯$▯�~�▯D��+���9*�D���T0��:J�+u�;�▯����q��M����S}mZ"T<>�▯g(��l�����z�e�G�A�wy�pBA+@�u���4�g���X�h='�7}▯Ì��B��U��M�
������-^▯7�|�W41��%6V�v���C����▯▯&▯�▯▯
▯                                       ��▯�X��H+O���������o-��GQH��p����Fd▯�@&▯▯;�q�h▯▯�o,mI��▯�l��▯�u▯�}
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl rsautl -decrypt -inkey aafreenPrivkey.pem -in result.bin
Hello...
Have a good day!!!
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl rsautl -decrypt -inkey aafreenPrivkey.pem -in result.bin
Hello...
Have a good day!!!
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat result.bin
P�Tg�-����t��▯$▯�~�▯D��+���9*�D���T0��:J�+u�;�▯����q��M����S}mZ"T<>�▯g(��l�����z�e�G�A�wy�pBA+@�u���4�g���X�h='�7}▯Ì��B��U��M�
������-^▯7�|�W41��%6V�v���C����▯▯&▯�▯▯
▯                                       ��▯�X��H+O���������o-��GQH��p����Fd▯�@&▯▯;�q�h▯▯�o,mI��▯�l��▯�u▯�}
 ]]▯system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$openssl rsautl -decrypt -inkey aafreenPrivkey.pem -in result.bin
Hello...
Have a good day!!!
```

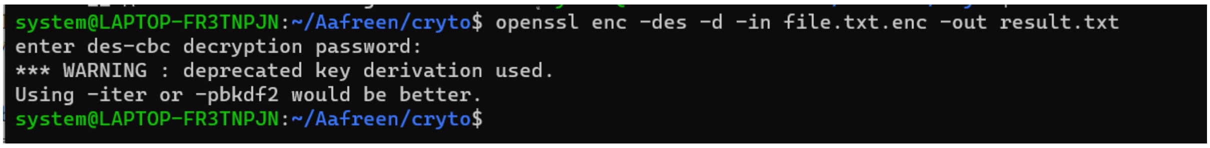
As we can see, by entering a wrong password or using wrong key, the contents of the cipher text is unavailable.

Hence the password(private key encryption) or private key should be known to decipher the text.

By changing the contents of cipher text, the content cannot be decipherd.



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat file.txt.enc
Salted__�♀n=Lz�:l'�T#d�����6igG# �'*▯�m▯�▯d♀system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ nano file.txt.enc
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ cat file.txt.enc
Salted__�♀n=Lz�:l'�T#d�����6igG# �'*▯�m▯�▯d♀system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$
```



```
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$ openssl enc -des -d -in file.txt.enc -out result.txt
enter des-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
system@LAPTOP-FR3TNPJN:~/Aafreen/cryto$
```

c) Identify the need for each approach and comprehend its significance.

DES -> Data encryption standard. It is a symmetric key block cipher.

Plaintext is divided into two halves in DES encryption, and then DES uses a 64-bit plaintext and a 56-bit key to generate a 64-bit ciphertext, which is an encrypted representation of the data.

The key length used for encryption in DES is 56 bits, although the block size is 64 bits .DES entails 16 rounds of identical procedures, regardless of key length.

Because the amount of operations in DES is fixed and no permutation combinations are permitted, it is easier to break the encryption, making it less secure than AES.

AES -> Advanced Encryption Standard developed after DES. AES is currently implemented world wide both in hardware and software.

AES uses a 128-bit plaintext and a 128-bit secret key to create a 128-bit block, which is then processed to produce 16 bytes (128-bit) ciphertext.

In the case of AES, the key length might be 128 bits, 192 bits, or 256 bits, with 10 rounds (128 bits), 12 rounds (192 bits), or 14 rounds (256 bits).

AES, on the other hand, is more secure than DES encryption and has become the de facto international standard.

RSA ->RSA is widely used for secure data transmission. 2 keys are generated. One is public known to all while other is private. Message to the owner is encrypted using public key and only the owner can decipher using private key.

The owner can use the private key as a digital signature which can be verified by the public using public key.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers. It is a slow algorithm compared to others.

Elliptic Curve Cryptography -> It is a key-based technique for encrypting data. ECC focuses on pairs of public and private keys for decryption and encryption of web traffic.

Unlike RSA , it is more powerful. ECC has grown in popularity due to its smaller key size and ability to maintain security. ECC bases its approach to public key cryptographic systems on how elliptic curves are structured algebraically over finite fields. Therefore, ECC creates keys that are more difficult, mathematically, to crack.