II.2317 - Cybersecurity

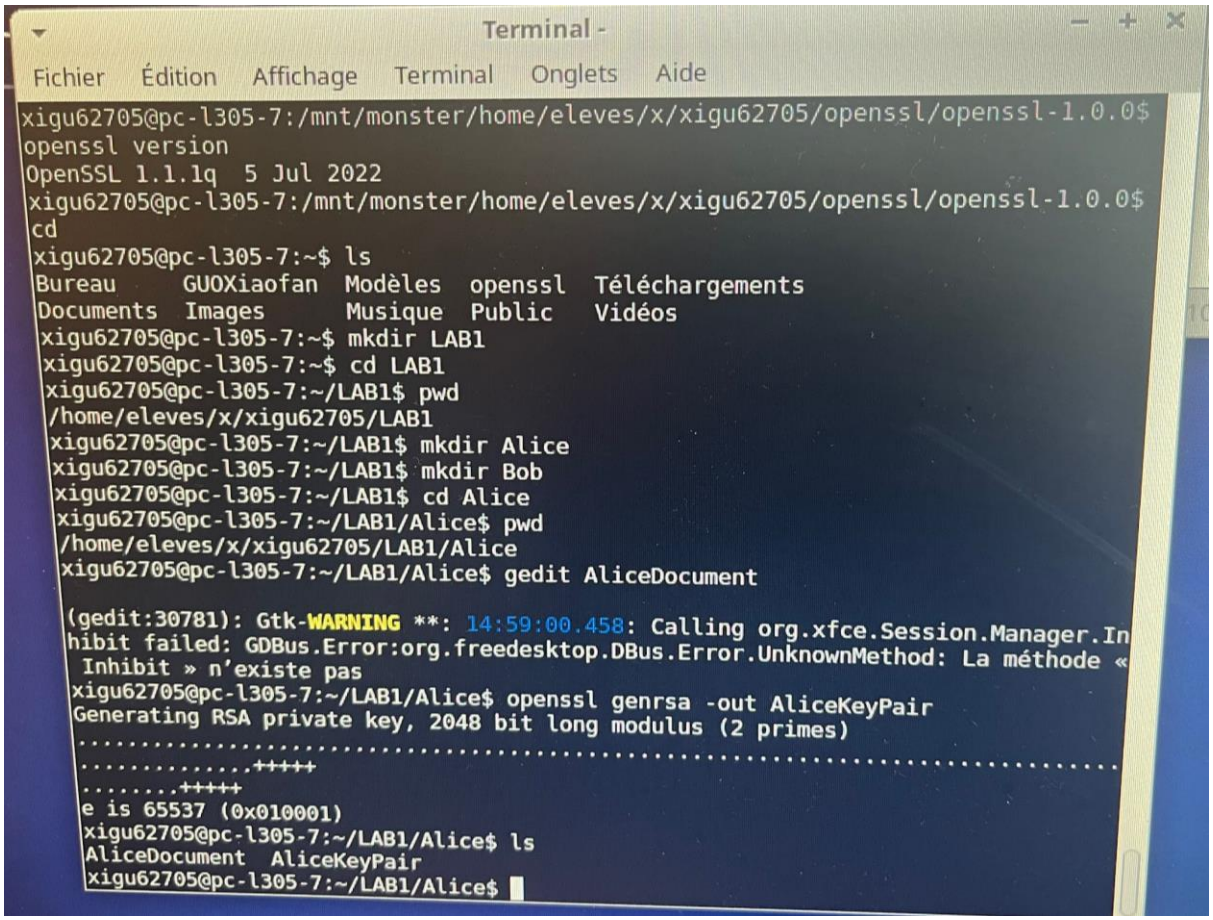
Practical Work 1

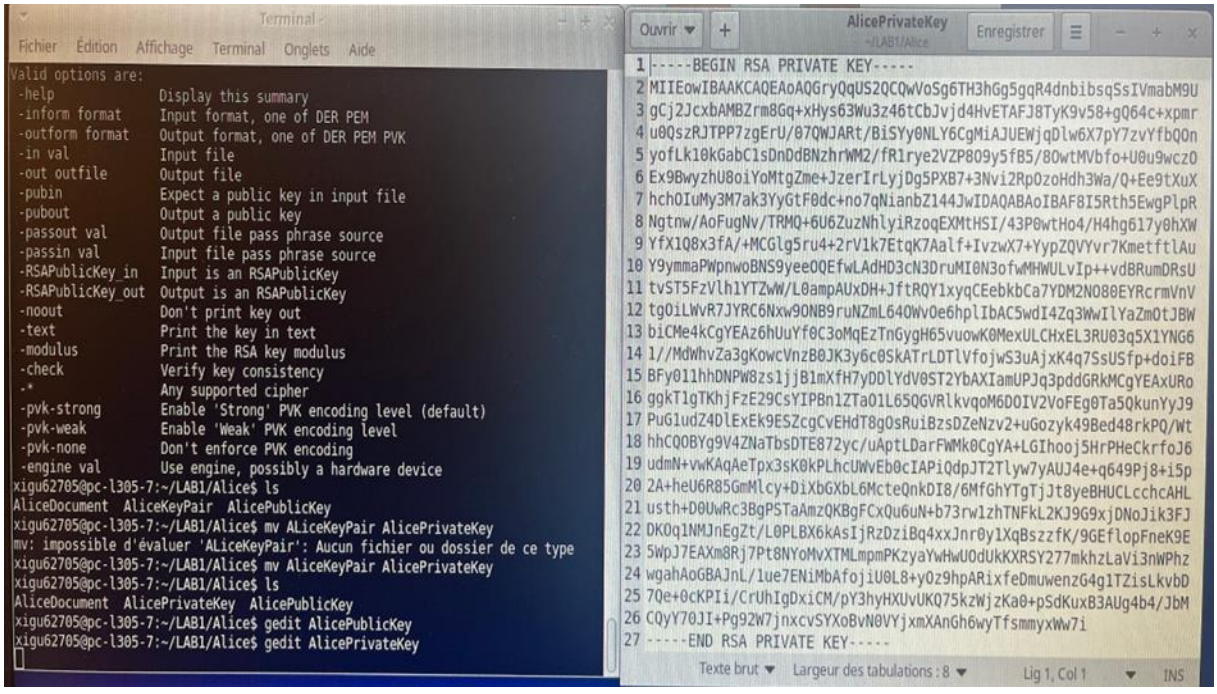
GUO Xiaofan

LAB 1

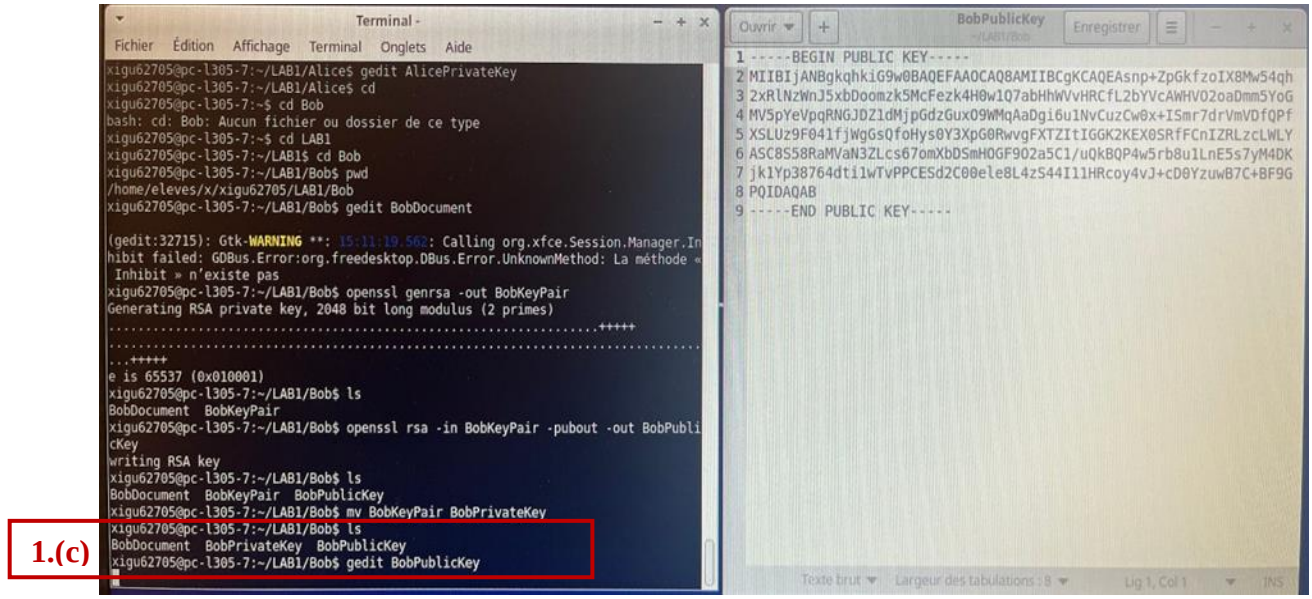
1 Introduction to OpenSSL

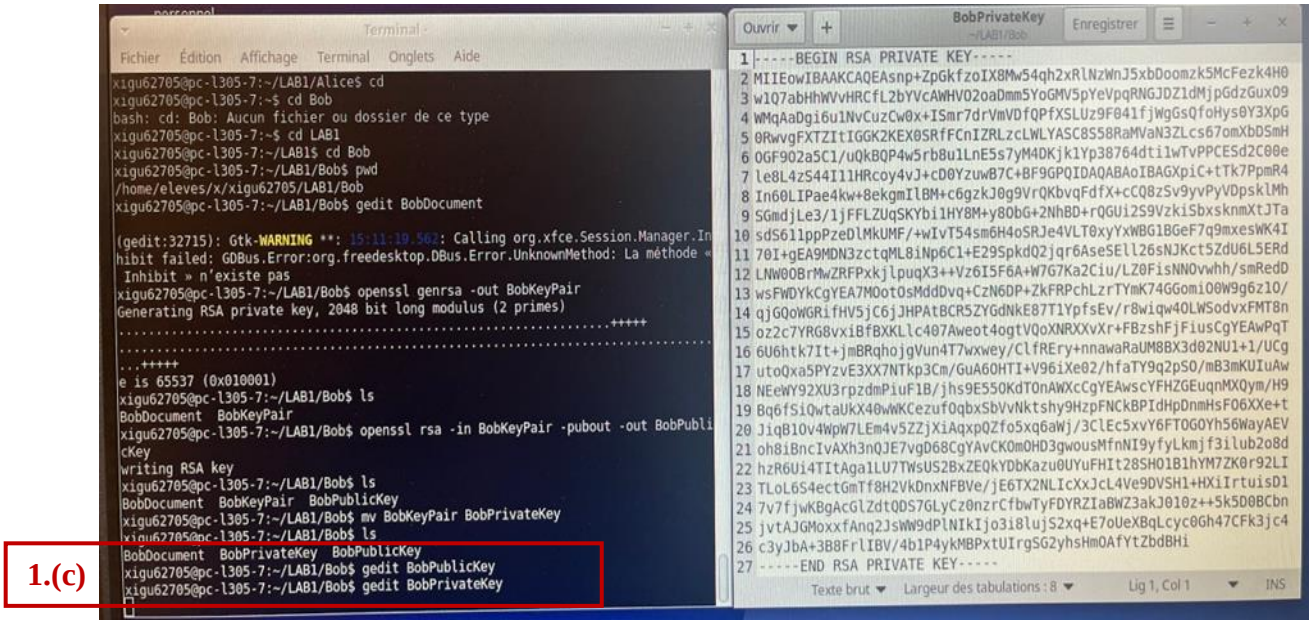
5. - 7.

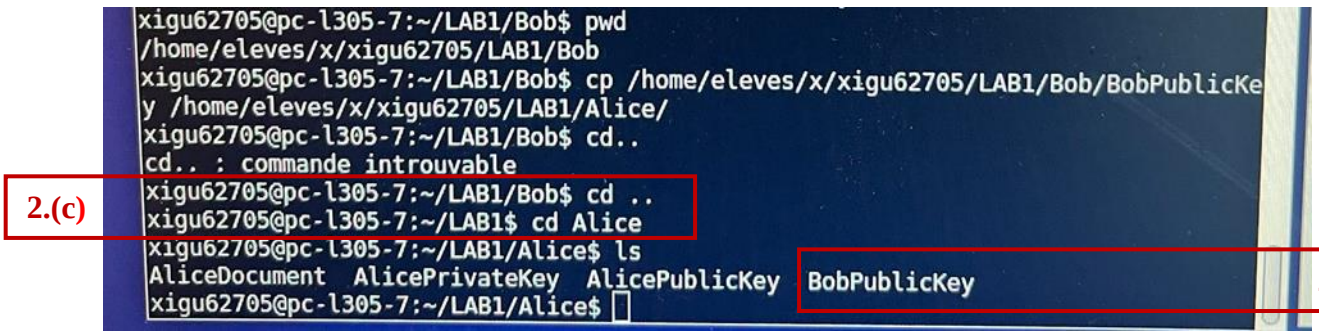


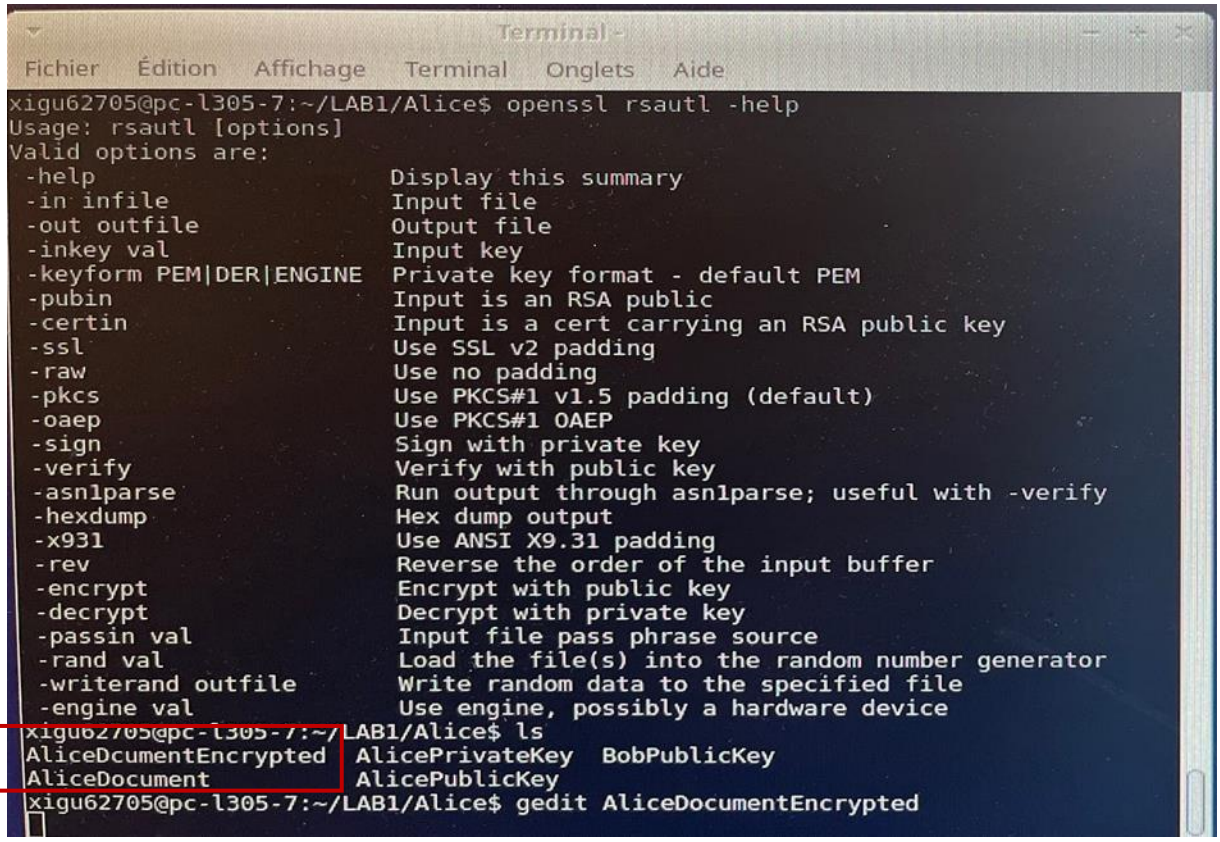


2 Exercise “Asymmetric cryptography”

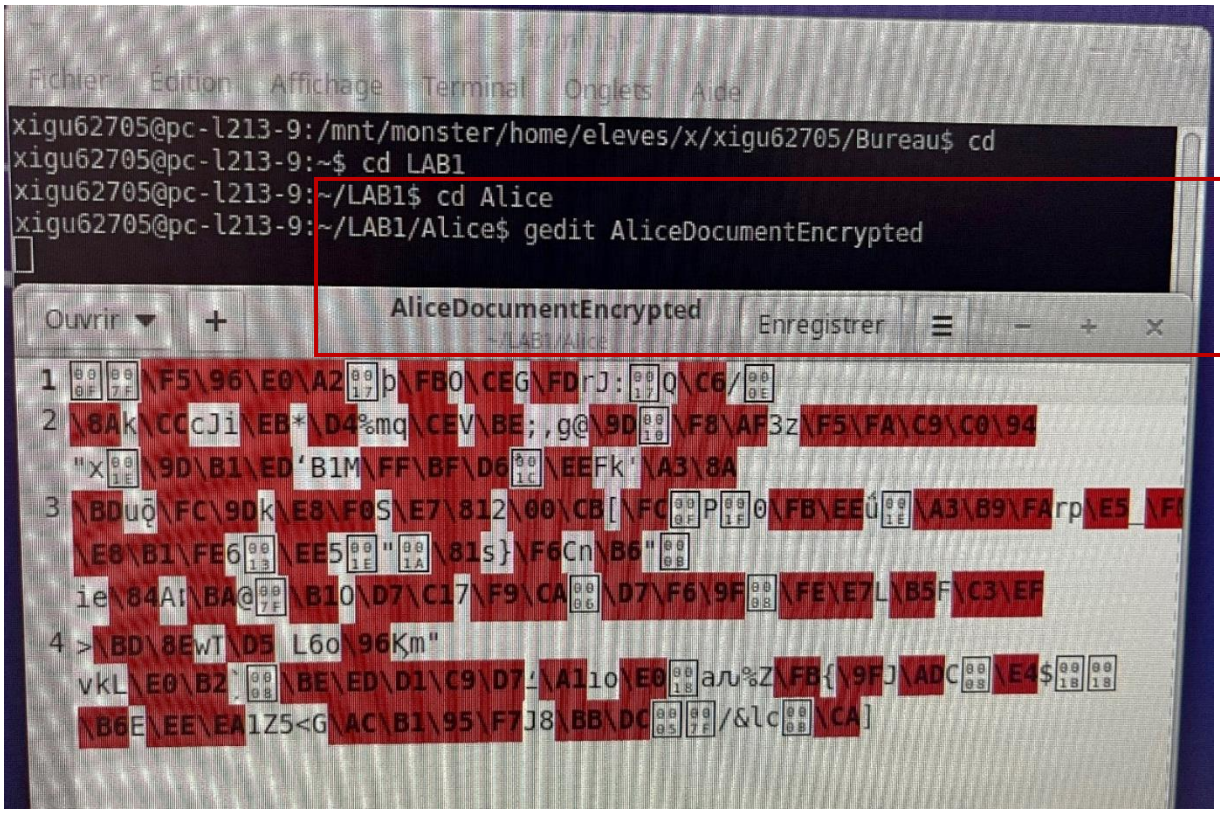






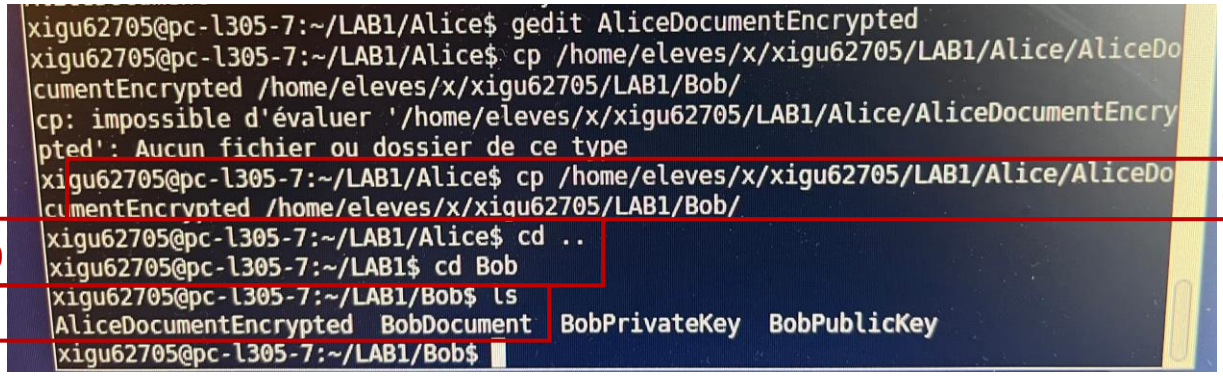


2.(g)



2.(h)

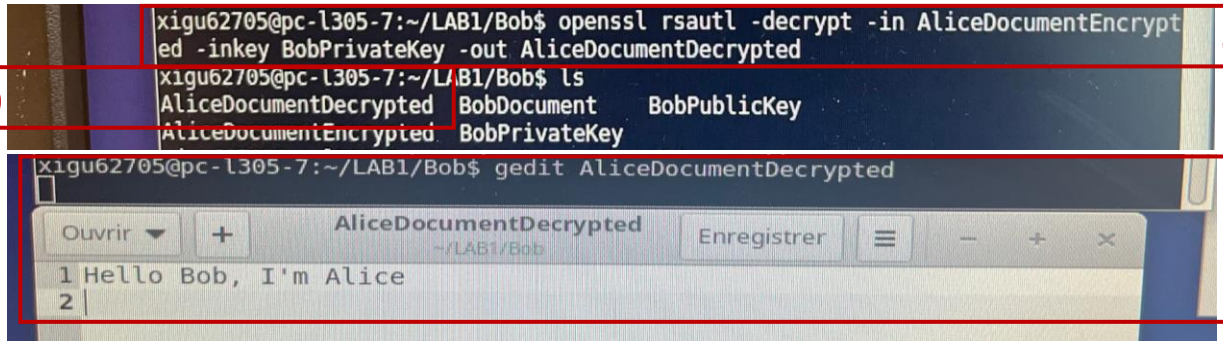
We cannot directly read the content in AliceDocumentEncrypted (it will appear as garbled characters)



3.(a)

3.(b)

3.(c)

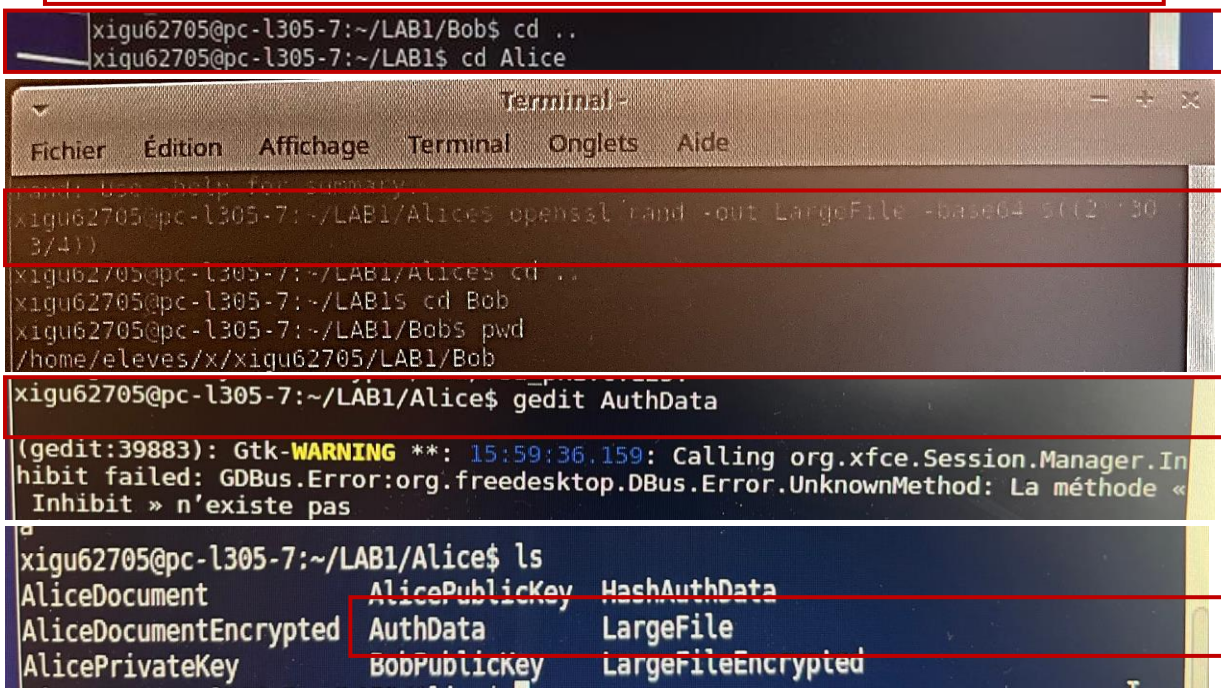


3.(d)

3.(e)

3.(f)

After completing the decoding operation, we can read the initial content written to AliceDocument in the AliceDocumentDecrypted file in Bob's folder.

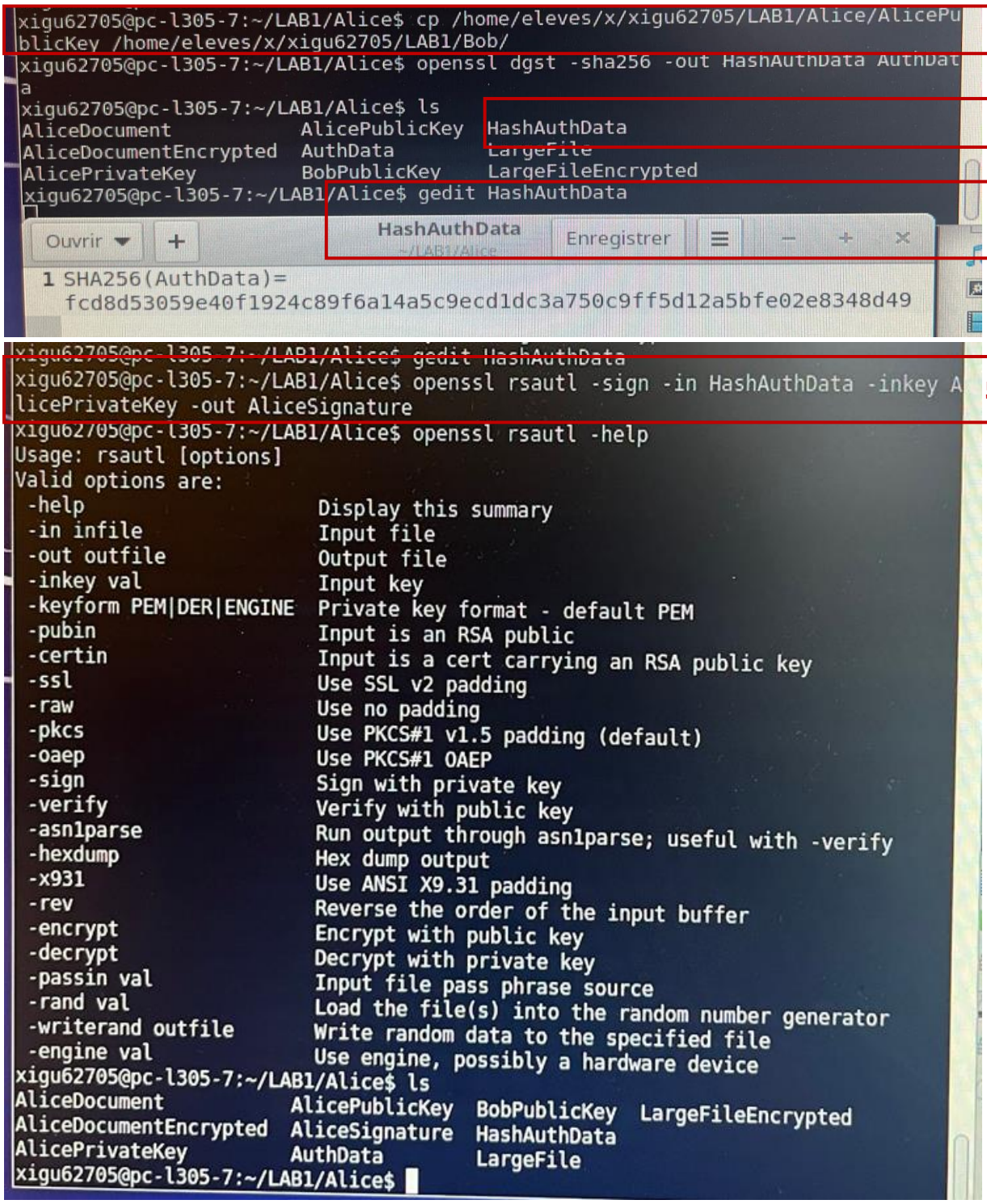


4.(a)

4.(c)

5.(a)

5.(b)

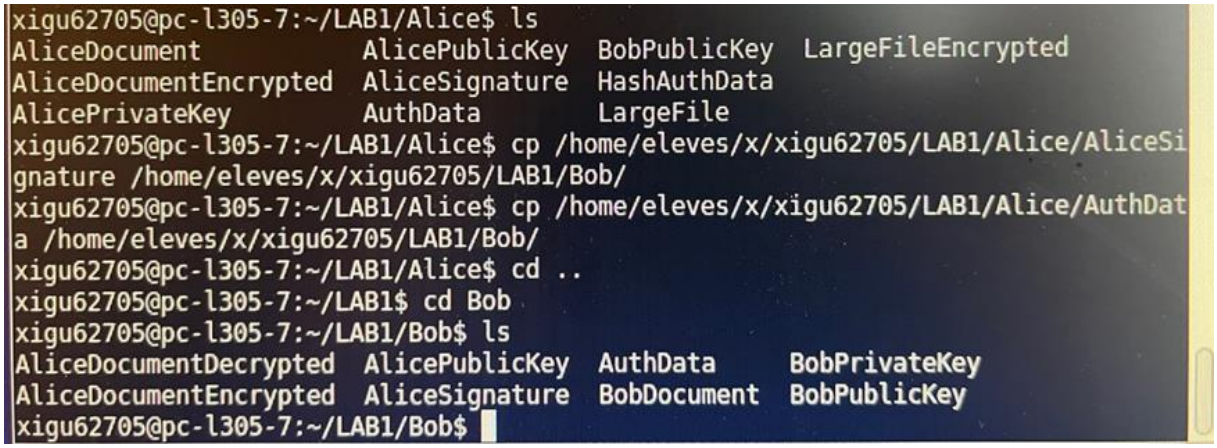

xigu62705@pc-l305-7:~/LAB1/Alice$ cp /home/eleves/x/xigu62705/LAB1/Alice/AlicePu
blicKey /home/eleves/x/xigu62705/LAB1/Bob/
xigu62705@pc-l305-7:~/LAB1/Alice$ openssl dgst -sha256 -out HashAuthData AuthDat
a
xigu62705@pc-l305-7:~/LAB1/Alice$ ls
AliceDocument AlicePublicKey HashAuthData
AliceDocumentEncrypted AuthData LargeFile
AlicePrivateKey BobPublicKey LargeFileEncrypted
xigu62705@pc-l305-7:~/LAB1/Alice$ gedit HashAuthData
Ouvrir
HashAuthData
Enregistrer
1 SHA256(AuthData)=
fcd8d53059e40f1924c89f6a14a5c9ecd1dc3a750c9ff5d12a5bfe02e8348d49
xigu62705@pc-l305-7:~/LAB1/Alice$ openssl rsautl -sign -in HashAuthData -inkey A
licePrivateKey -out AliceSignature
xigu62705@pc-l305-7:~/LAB1/Alice$ openssl rsautl -help
Usage: rsautl [options]
Valid options are:
-help Display this summary
-in infile Input file
-out outfile Output file
-inkey val Input key
-keyform PEM|DER|ENGINE Private key format - default PEM
-pubin Input is an RSA public
-certin Input is a cert carrying an RSA public key
-ssl Use SSL v2 padding
-raw Use no padding
-pkcs Use PKCS#1 v1.5 padding (default)
-oaep Use PKCS#1 OAEP
-sign Sign with private key
-verify Verify with public key
-asn1parse Run output through asn1parse; useful with -verify
-hexdump Hex dump output
-x931 Use ANSI X9.31 padding
-rev Reverse the order of the input buffer
-encrypt Encrypt with public key
-decrypt Decrypt with private key
-passin val Input file pass phrase source
-rand val Load the file(s) into the random number generator
-writerand outfile Write random data to the specified file
-engine val Use engine, possibly a hardware device
xigu62705@pc-l305-7:~/LAB1/Alice$ ls
AliceDocument AlicePublicKey BobPublicKey LargeFileEncrypted
AliceDocumentEncrypted AliceSignature HashAuthData
AlicePrivateKey AuthData LargeFile
xigu62705@pc-l305-7:~/LAB1/Alice$

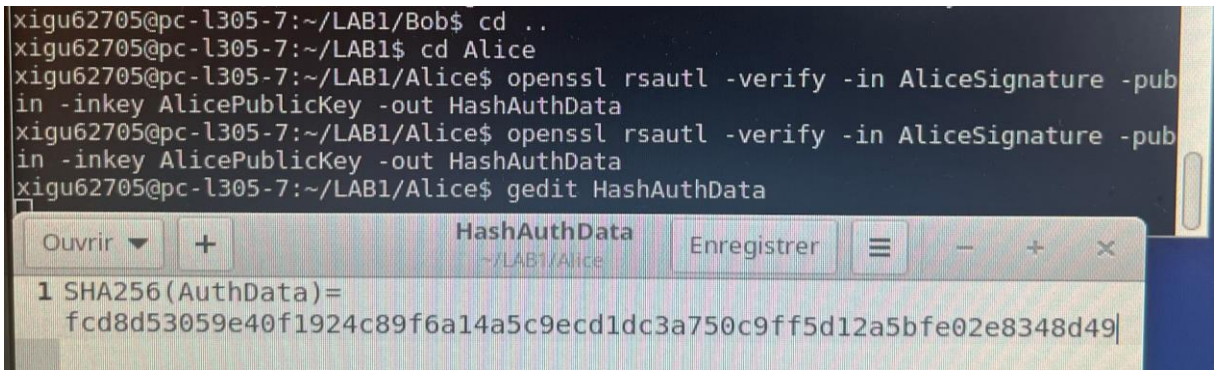
5.(c)

5.(e)

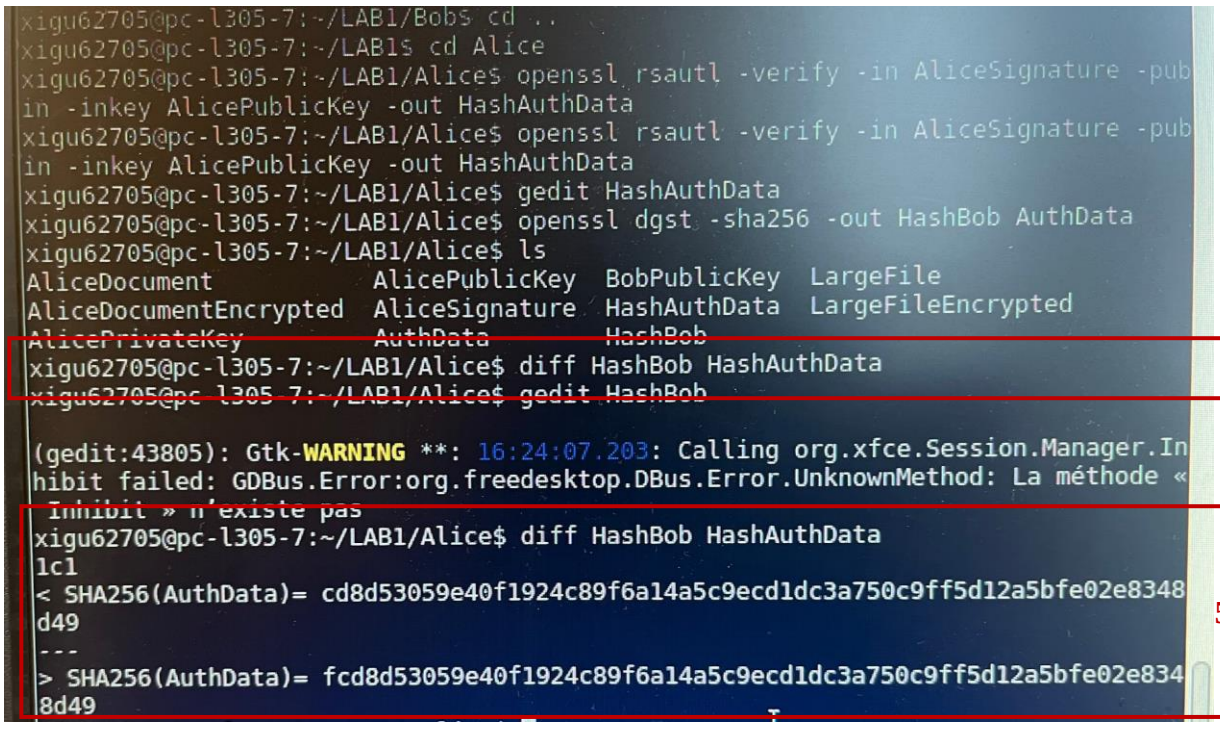
5.(f)

5.(g)


xigu62705@pc-l305-7:~/LAB1/Alice$ ls
AliceDocument AlicePublicKey BobPublicKey LargeFileEncrypted
AliceDocumentEncrypted AliceSignature HashAuthData
AlicePrivateKey AuthData LargeFile
xigu62705@pc-l305-7:~/LAB1/Alice$ cp /home/eleves/x/xigu62705/LAB1/Alice/AliceSignature /home/eleves/x/xigu62705/LAB1/Bob/
xigu62705@pc-l305-7:~/LAB1/Alice$ cp /home/eleves/x/xigu62705/LAB1/Alice/AuthData /home/eleves/x/xigu62705/LAB1/Bob/
xigu62705@pc-l305-7:~/LAB1/Alice$ cd ..
xigu62705@pc-l305-7:~/LAB1$ cd Bob
xigu62705@pc-l305-7:~/LAB1/Bob$ ls
AliceDocumentDecrypted AlicePublicKey AuthData BobPrivateKey
AliceDocumentEncrypted AliceSignature BobDocument BobPublicKey
xigu62705@pc-l305-7:~/LAB1/Bob$


xigu62705@pc-l305-7:~/LAB1/Bob$ cd ..
xigu62705@pc-l305-7:~/LAB1$ cd Alice
xigu62705@pc-l305-7:~/LAB1/Alice$ openssl rsautl -verify -in AliceSignature -pubin -inkey AlicePublicKey -out HashAuthData
xigu62705@pc-l305-7:~/LAB1/Alice$ openssl rsautl -verify -in AliceSignature -pubin -inkey AlicePublicKey -out HashAuthData
xigu62705@pc-l305-7:~/LAB1/Alice$ gedit HashAuthData
Ouvrir
HashAuthData
~/LAB1/Alice
Enregistrer
1 SHA256(AuthData)=
fcd8d53059e40f1924c89f6a14a5c9ecd1dc3a750c9ff5d12a5bfe02e8348d49


xigu62705@pc-l305-7:~/LAB1/Alice$ openssl dgst -sha256 -out HashBob AuthData
xigu62705@pc-l305-7:~/LAB1/Alice$ ls
AliceDocument AlicePublicKey BobPublicKey LargeFile
AliceDocumentEncrypted AliceSignature HashAuthData LargeFileEncrypted
AlicePrivateKey AuthData HashBob
xigu62705@pc-l305-7:~/LAB1/Alice$


xigu62705@pc-l305-7:~/LAB1/Alice$ diff HashBob HashAuthData
(gedit:43805): Gtk-WARNING **: 16:24:07.203: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: La méthode « Inhibit » n'existe pas
xigu62705@pc-l305-7:~/LAB1/Alice$ diff HashBob HashAuthData
1c1
< SHA256(AuthData)= cd8d53059e40f1924c89f6a14a5c9ecd1dc3a750c9ff5d12a5bfe02e8348d49

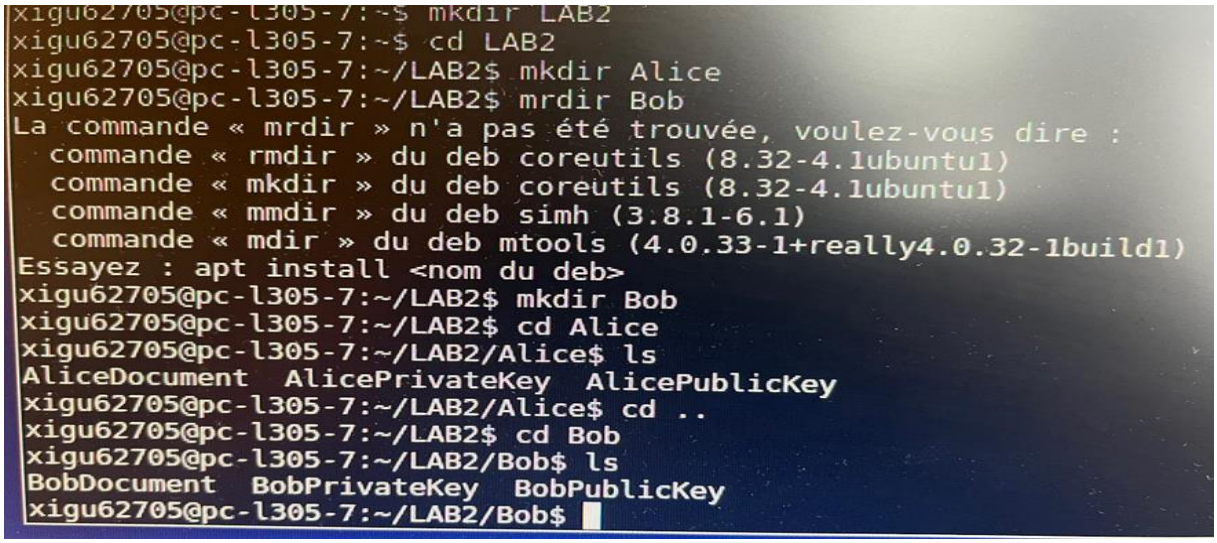
> SHA256(AuthData)= fcd8d53059e40f1924c89f6a14a5c9ecd1dc3a750c9ff5d12a5bfe02e8348d49

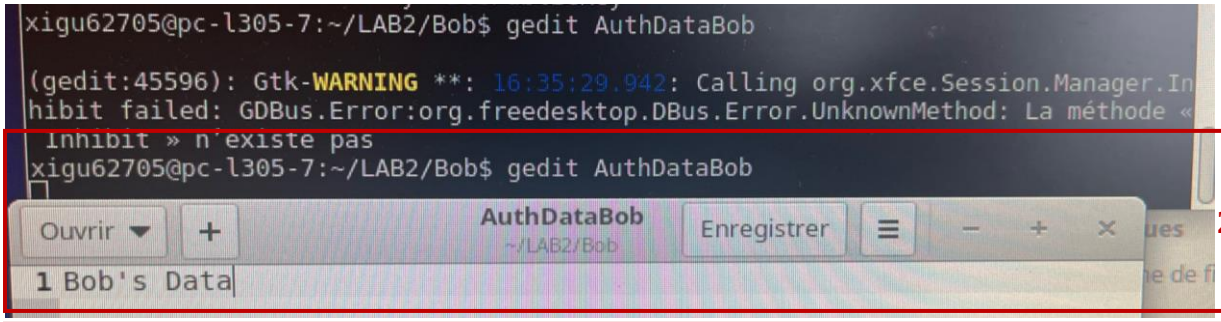
5.(n)

5.(o)

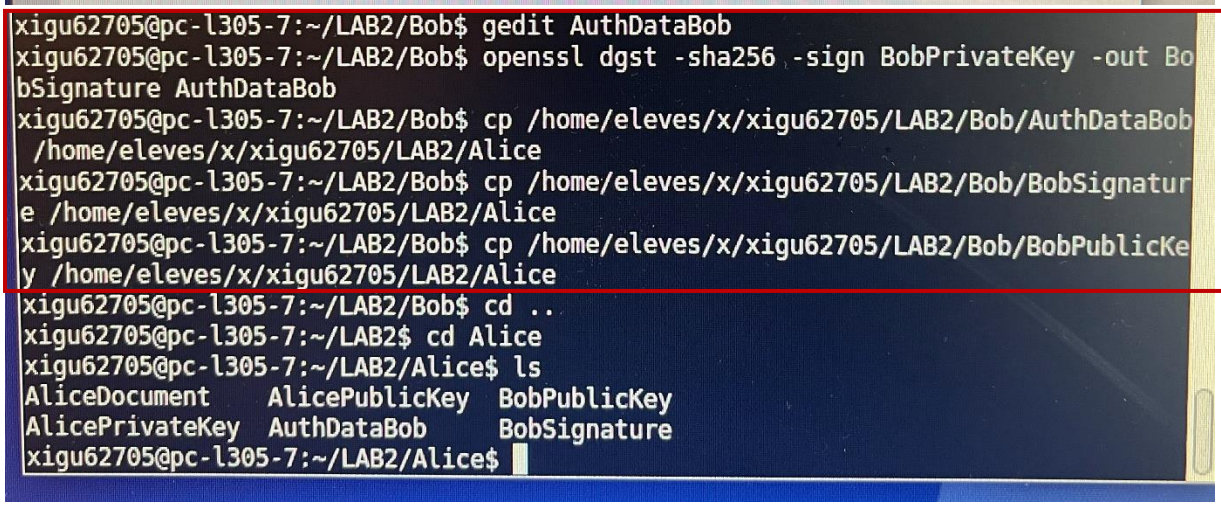
LAB 2

1 Exercise “Symmetric cryptography”

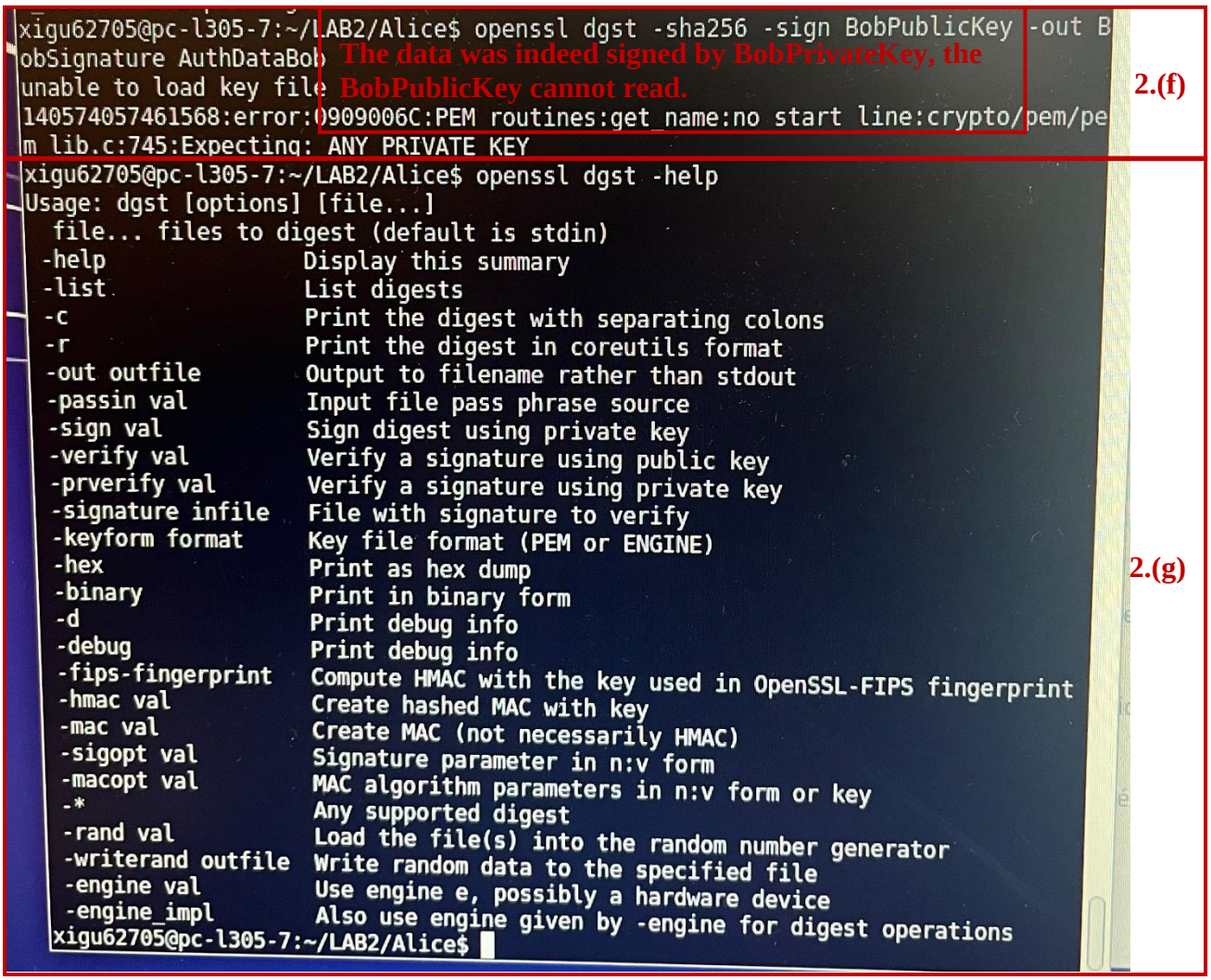


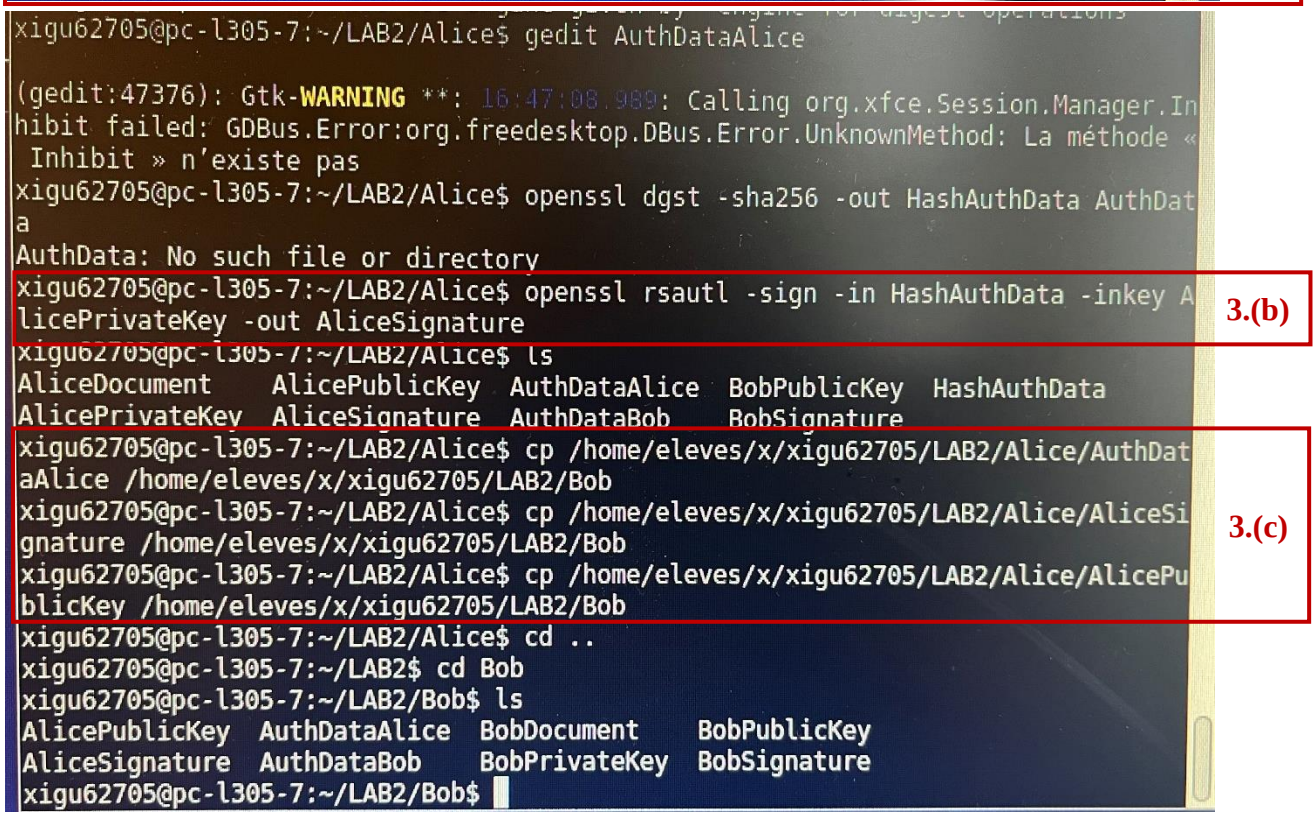


2.(b)



2.(c)


xigu62705@pc-l305-7:~/LAB2/Alice$ openssl dgst -sha256 -sign BobPublicKey -out B
obSignature AuthDataBob
unable to load key file
140574057461568:error:0909006C:PEM routines:get_name:no start line:crypto/pem/pe
m_lib.c:745:Expecting: ANY PRIVATE KEY
The data was indeed signed by BobPrivateKey, the BobPublicKey cannot read.
2.(f)
xigu62705@pc-l305-7:~/LAB2/Alice$ openssl dgst -help
Usage: dgst [options] [file...]
file... files to digest (default is stdin)
-help Display this summary
-list List digests
-c Print the digest with separating colons
-r Print the digest in coreutils format
-out outfile Output to filename rather than stdout
-passin val Input file pass phrase source
-sign val Sign digest using private key
-verify val Verify a signature using public key
-prverify val Verify a signature using private key
-signature infile File with signature to verify
-keyform format Key file format (PEM or ENGINE)
-hex Print as hex dump
-binary Print in binary form
-d Print debug info
-debug Print debug info
-fips-fingerprint Compute HMAC with the key used in OpenSSL-FIPS fingerprint
-hmac val Create hashed MAC with key
-mac val Create MAC (not necessarily HMAC)
-sigopt val Signature parameter in n:v form
-macopt val MAC algorithm parameters in n:v form or key
-* Any supported digest
-rand val Load the file(s) into the random number generator
-writerand outfile Write random data to the specified file
-engine val Use engine e, possibly a hardware device
-engine_impl Also use engine given by -engine for digest operations
xigu62705@pc-l305-7:~/LAB2/Alice$
2.(g)


xigu62705@pc-l305-7:~/LAB2/Alice$ gedit AuthDataAlice
(gedit:47376): Gtk-WARNING **: 16:47:08.989: Calling org.xfce.Session.Manager.In
hibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: La méthode «
Inhibit » n'existe pas
xigu62705@pc-l305-7:~/LAB2/Alice$ openssl dgst -sha256 -out HashAuthData AuthDat
a
AuthData: No such file or directory
xigu62705@pc-l305-7:~/LAB2/Alice$ openssl rsautl -sign -in HashAuthData -inkey A
licePrivateKey -out AliceSignature
3.(b)
xigu62705@pc-l305-7:~/LAB2/Alice$ ls
AliceDocument AlicePublicKey AuthDataAlice BobPublicKey HashAuthData
AlicePrivateKey AliceSignature AuthDataBob BobSignature
xigu62705@pc-l305-7:~/LAB2/Alice$ cp /home/eleves/x/xigu62705/LAB2/Alice/AuthDat
aAlice /home/eleves/x/xigu62705/LAB2/Bob
xigu62705@pc-l305-7:~/LAB2/Alice$ cp /home/eleves/x/xigu62705/LAB2/Alice/AliceSi
gnature /home/eleves/x/xigu62705/LAB2/Bob
xigu62705@pc-l305-7:~/LAB2/Alice$ cp /home/eleves/x/xigu62705/LAB2/Alice/AlicePu
blicKey /home/eleves/x/xigu62705/LAB2/Bob
3.(c)
xigu62705@pc-l305-7:~/LAB2/Alice$ cd ..
xigu62705@pc-l305-7:~/LAB2$ cd Bob
xigu62705@pc-l305-7:~/LAB2/Bob$ ls
AlicePublicKey AuthDataAlice BobDocument BobPublicKey
AliceSignature AuthDataBob BobPrivateKey BobSignature
xigu62705@pc-l305-7:~/LAB2/Bob$

AlicePublicKey can be used to verify AliceSignature

```
xigu62705@pc-l305-7:~/LAB2/Bob$ openssl rsautl -verify -in AliceSignature -pubin
 -inkey AlicePublicKey -out HashAuthData
xigu62705@pc-l305-7:~/LAB2/Bob$ openssl dgst -sha256 -out HashBob AuthData
AuthData: No such file or directory
xigu62705@pc-l305-7:~/LAB2/Bob$ openssl dgst -sha256 -out HashBob AuthData
xigu62705@pc-l305-7:~/LAB2/Bob$ diff HashBob HashAuthData
1d0
< SHA256(AuthData)= fcd8d53059e40f1924c89f6a14a5c9ecd1dc3a750c9ff5d12a5bfe02e834
8d49
xigu62705@pc-l305-7:~/LAB2/Bob$
```

3.(f)

```
xigu62705@pc-l305-7:~/LAB2/Bob$ openssl rand -hex -out SymKey 64
xigu62705@pc-l305-7:~/LAB2/Bob$ ls
AlicePublicKey  AuthDataAlice  BobPrivateKey  HashAuthData
AliceSignature  AuthDataBob    BobPublicKey   HashBob
AuthData        BobDocument    BobSignature   SymKey
xigu62705@pc-l305-7:~/LAB2/Bob$
xigu62705@pc-l305-7:~/LAB2/Bob$
```

4.(b)

4.(c)

```
xigu62705@pc-l305-7:~/LAB2/Bob$ openssl rsautl -encrypt -pubin -inkey AlicePubli
cKey -in SymKey -out SymKeyEncrypted
xigu62705@pc-l305-7:~/LAB2/Bob$ ls
AlicePublicKey  AuthDataAlice  BobPrivateKey  HashAuthData  SymKeyEncrypted
AliceSignature  AuthDataBob    BobPublicKey   HashBob
AuthData        BobDocument    BobSignature   SymKey
xigu62705@pc-l305-7:~/LAB2/Bob$ cp /home/eleves/x/xigu62705/LAB2/Bob/SymKeyEncry
pted /home/eleves/x/xigu62705/LAB2/Alice
xigu62705@pc-l305-7:~/LAB2/Bob$ cd ..
xigu62705@pc-l305-7:~/LAB2$ cd Alice
xigu62705@pc-l305-7:~/LAB2/Alice$ ls
AliceDocument    AlicePublicKey  AuthDataAlice  BobPublicKey  HashAuthData
AlicePrivateKey  AliceSignature  AuthDataBob    BobSignature  SymKeyEncrypted
xigu62705@pc-l305-7:~/LAB2/Alice$
```

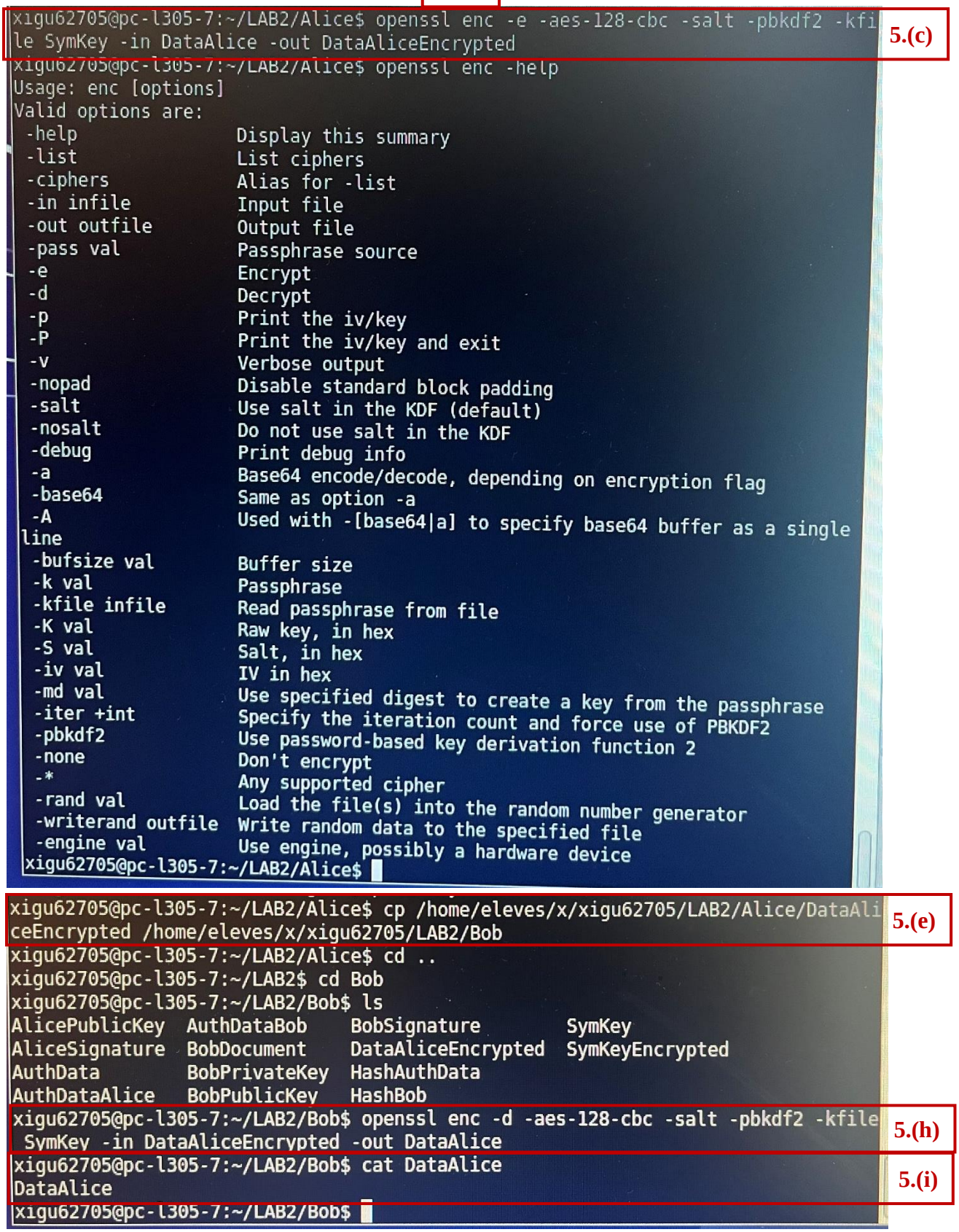
4.(d)

4.(f)

```
xigu62705@pc-l305-7:~/LAB2/Alice$ openssl rsautl -decrypt -inkey AlicePrivateKey
-in SymKeyEncrypted -out SymKey
rsautl: Use -help for summary.
```

4.(i)

5.(j)


xigu62705@pc-l305-7:~/LAB2/Alice$ openssl enc -e -aes-128-cbc -salt -pbkdf2 -kfile SymKey -in DataAlice -out DataAliceEncrypted
5.(c)
xigu62705@pc-l305-7:~/LAB2/Alice$ openssl enc -help
Usage: enc [options]
Valid options are:
-help Display this summary
-list List ciphers
-ciphers Alias for -list
-in infile Input file
-out outfile Output file
-pass val Passphrase source
-e Encrypt
-d Decrypt
-p Print the iv/key
-P Print the iv/key and exit
-v Verbose output
-nopad Disable standard block padding
-salt Use salt in the KDF (default)
-nosalt Do not use salt in the KDF
-debug Print debug info
-a Base64 encode/decode, depending on encryption flag
-base64 Same as option -a
-A Used with -[base64|a] to specify base64 buffer as a single line
-bufsize val Buffer size
-k val Passphrase
-kfile infile Read passphrase from file
-K val Raw key, in hex
-S val Salt, in hex
-iv val IV in hex
-md val Use specified digest to create a key from the passphrase
-iter +int Specify the iteration count and force use of PBKDF2
-pbkdf2 Use password-based key derivation function 2
-none Don't encrypt
-* Any supported cipher
-rand val Load the file(s) into the random number generator
-writerand outfile Write random data to the specified file
-engine val Use engine, possibly a hardware device
xigu62705@pc-l305-7:~/LAB2/Alice$
xigu62705@pc-l305-7:~/LAB2/Alice$ cp /home/eleves/x/xigu62705/LAB2/Alice/DataAliceEncrypted /home/eleves/x/xigu62705/LAB2/Bob
5.(e)
xigu62705@pc-l305-7:~/LAB2/Alice$ cd ..
xigu62705@pc-l305-7:~/LAB2$ cd Bob
xigu62705@pc-l305-7:~/LAB2/Bob$ ls
AlicePublicKey AuthDataBob BobSignature SymKey
AliceSignature BobDocument DataAliceEncrypted SymKeyEncrypted
AuthData BobPrivateKey HashAuthData
AuthDataAlice BobPublicKey HashBob
xigu62705@pc-l305-7:~/LAB2/Bob$ openssl enc -d -aes-128-cbc -salt -pbkdf2 -kfile SymKey -in DataAliceEncrypted -out DataAlice
5.(h)
xigu62705@pc-l305-7:~/LAB2/Bob$ cat DataAlice
DataAlice
5.(i)
xigu62705@pc-l305-7:~/LAB2/Bob$

2 Exercise “Certificates X509”

```
xigu62705@pc-l305-7:~/LAB2/Bob$ cd ..
xigu62705@pc-l305-7:~/LAB2$ openssl s_client -connect www.lcl.fr:443 > Certifica
teLCL
depth=2 C = US, ST = New Jersey, L = Jersey City, O = The USERTRUST Network, CN
= USERTrust RSA Certification Authority
verify return:1
depth=1 C = GB, ST = Greater Manchester, L = Salford, O = Sectigo Limited, CN =
Sectigo RSA Organization Validation Secure Server CA
verify return:1
depth=0 C = FR, ST = Auvergne-Rh\C3\B4ne-Alpes, O = CREDIT LYONNAIS SA, CN = www
.lcl.fr
verify return:1
```

1.-2.

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -noout -in CertificateLCL -issuer
issuer=C = GB, ST = Greater Manchester, L = Salford, O = Sectigo Limited, CN = S
ectigo RSA Organization Validation Secure Server CA
```

3.

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -help
Usage: x509 [options]
Valid options are:
 -help                       Display this summary
 -inform format              Input format - default PEM (one of DER or PEM)
 -in infile                  Input file - default stdin
 -outform format             Output format - default PEM (one of DER or PEM)
 -out outfile                Output file - default stdout
 -keyform PEM|DER|ENGINE     Private key format - default PEM
 -passin val                 Private key password/pass-phrase source
 -serial                     Print serial number value
 -subject_hash               Print subject hash value
 -*                          Any supported digest
 -subject_hash_old           Print old-style (MD5) subject hash value
 -issuer_hash_old            Print old-style (MD5) issuer hash value
 -engine val                 Use engine, possibly a hardware device
 -preserve_dates             preserve existing dates when signing
```

4.

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -in CertificateLCL -noout -dates
notBefore=Dec 15 00:00:00 2022 GMT
notAfter=Dec 15 23:59:59 2023 GMT
```

5.

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -in CertificateLCL -noout -text | grep
"Signature Algorithm"
        Signature Algorithm: sha256WithRSAEncryption
    Signature Algorithm: sha256WithRSAEncryption
```

6.

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -in CertificateLCL -noout -serial
serial=CDB5B1A7D3A2D065D811C0643FC339A8
```

7.

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -in CertificateLCL -pubkey -noout
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArJRun3tjz3eGvOkUGd4S
XXoCVxWjyfuOQjml0Du6++cu6L5rAFPxFlCINETaiBp0T8IvszJn1vqc9jGhAfDK
kqCn1py9xU14dIlP4yOUNsseufJVH6qb2C99aHF+yguFMP1K2VkhUEv7fYsKBG6Z
l4SQmyVh4Ikktxms1v1bewi/+9BM8THVm6yW/OjvA/xk4U9meFowL2GFUB3XsqLF
wCp2GwNHNQWjuoNCsdxOLKYix4qqqHX1mq/hW4vOOz1LmXdD8p2e7w7B/Mu6ZuSb
M6TB8CGkhYPVIkn1wrMU796C8USzz9trtAu4fw6oso+XxiZBoqla7KIv75h32jc8
NQIDAQAB
-----END PUBLIC KEY-----
xigu62705@pc-l305-7:~/LAB2$
```

8.

```
xigu62705@pc-l305-7:~/LAB2$ openssl genrsa -out ServerKeyPair -passout pass:pass
word 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................+++++
.....................................................+++++
e is 65537 (0x010001)
xigu62705@pc-l305-7:~/LAB2$ openssl rsa -pubout -in ServerKeyPair -out ServerPub
licKey
writing RSA key
xigu62705@pc-l305-7:~/LAB2$ mv ServerKeyPair ServerPrivateKey
xigu62705@pc-l305-7:~/LAB2$
```

9.(b)

```
e is 65537 (0x010001)
xigu62705@pc-l305-7:~/LAB2$ openssl rsa -pubout -in ServerKeyPair -out ServerPub
licKey
writing RSA key
xigu62705@pc-l305-7:~/LAB2$ mv ServerKeyPair ServerPrivateKey
xigu62705@pc-l305-7:~/LAB2$ openssl req -new -key ServerPrivatekey -out ServerRe
quest.csr
Can't open ServerPrivatekey for reading, No such file or directory
139862413813568:error:02001002:system library:fopen:No such file or directory:cr
ypto/bio/bss_file.c:69:fopen('ServerPrivatekey','r')
139862413813568:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio
/bss_file.c:76:
unable to load Private Key
xigu62705@pc-l305-7:~/LAB2$ openssl req -new -key ServerPrivateKey -out ServerRe
quest.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile de France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:EPITA
Organizational Unit Name (eg, section) []:First year
Common Name (e.g. server FQDN or YOUR name) []:myserver.fr
Email Address []:xiaofan.guo@eleve.isep.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
xigu62705@pc-l305-7:~/LAB2$
```

9.(c)

9.(e)

```
xigu62705@pc-l305-7:~/LAB2$ openssl genrsa -aes256 -out CAKeyPair -passout pass:
password 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................++++
.....................................................................................
.....................++++
e is 65537 (0x010001)
xigu62705@pc-l305-7:~/LAB2$
```

9.(f)

```
xigu62705@pc-l305-7:~/LAB2$ openssl req -x509 -new -key CAKeyPair -out CACertifi
cate.crt -days 500
Enter pass phrase for CAKeyPair:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile de France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyCA
Organizational Unit Name (eg, section) []:MyCA
Common Name (e.g. server FQDN or YOUR name) []:MyCA
Email Address []:CAemail
xigu62705@pc-l305-7:~/LAB2$
```

9.(g)

```
xigu62705@pc-l305-7:~/LAB2$ openssl x509 -req -in ServerRequest.csr -CA CACertif
icate.crt -CAkey CAKeyPair -CAcreateserial -out ServerCertificate.crt -days 500
-sha256
Signature ok
subject=C = FR, ST = Ile de France, L = Paris, O = EPITA, OU = First year, CN =
myserver.fr, emailAddress = xiaofan.guo@eleve.isep.fr
Getting CA Private Key
Enter pass phrase for CAKeyPair:
```

9.(h)

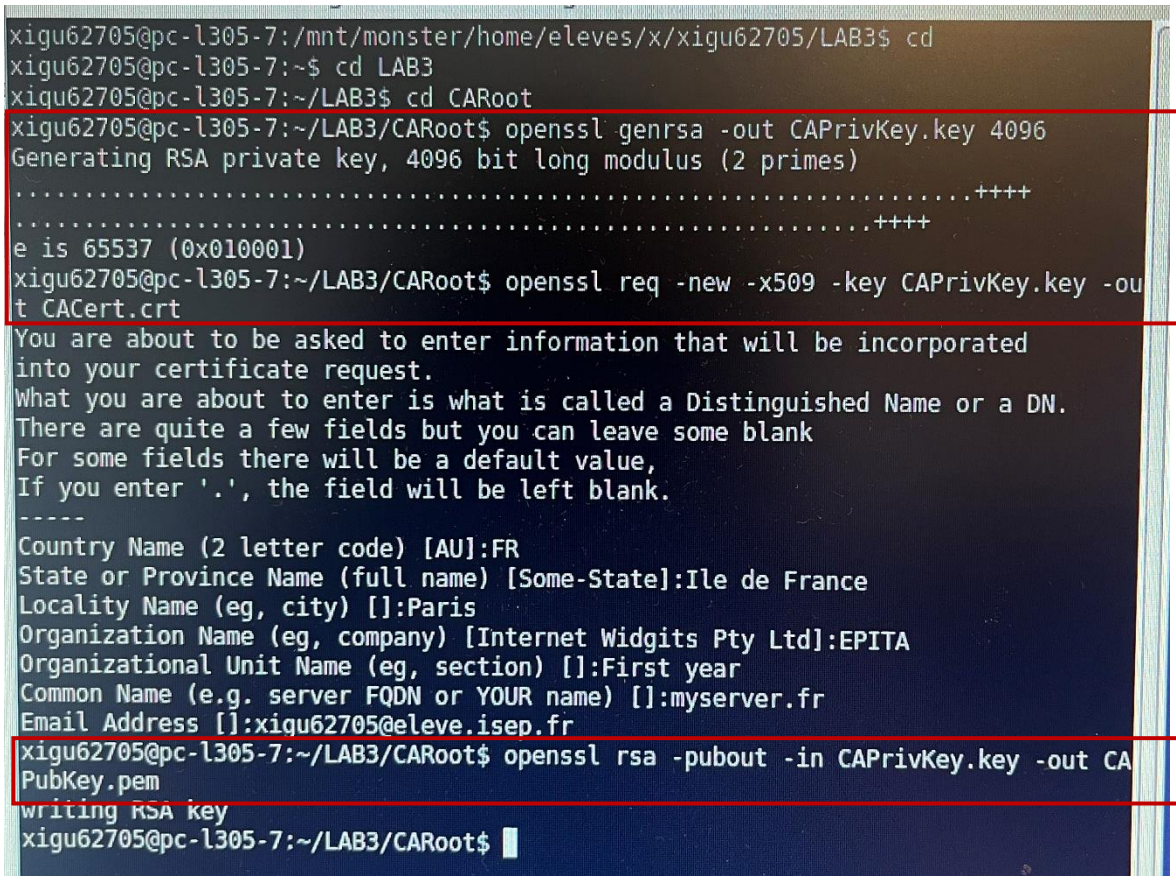
```
xigu62705@pc-l305-7:~/LAB2$ openssl verify -CAfile CACertificate.crt ServerCerti
ficate.crt
ServerCertificate.crt: OK
xigu62705@pc-l305-7:~/LAB2$
```

9.(i)

Verify the certificate of the server by entering the OpenSSL command is successful:
ServerCertificate.crt: OK

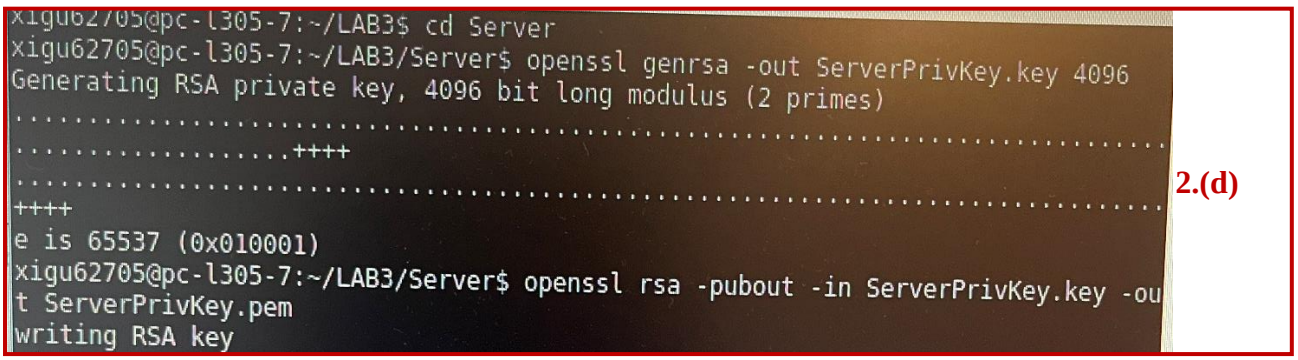
LAB 3

1 Exercise “Preparation of cryptographic security elements foreach actor”

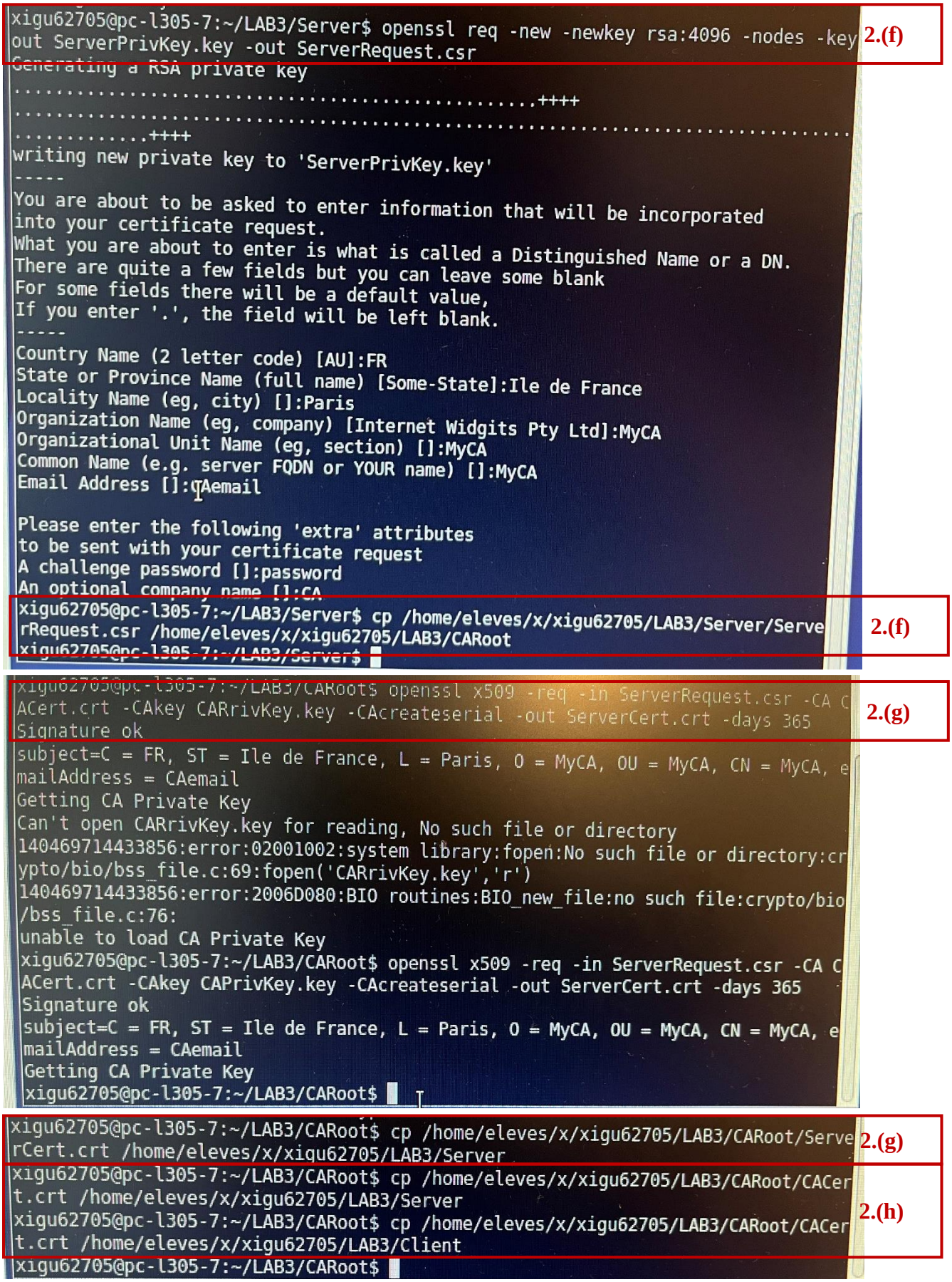


2.(c)

2.(c)



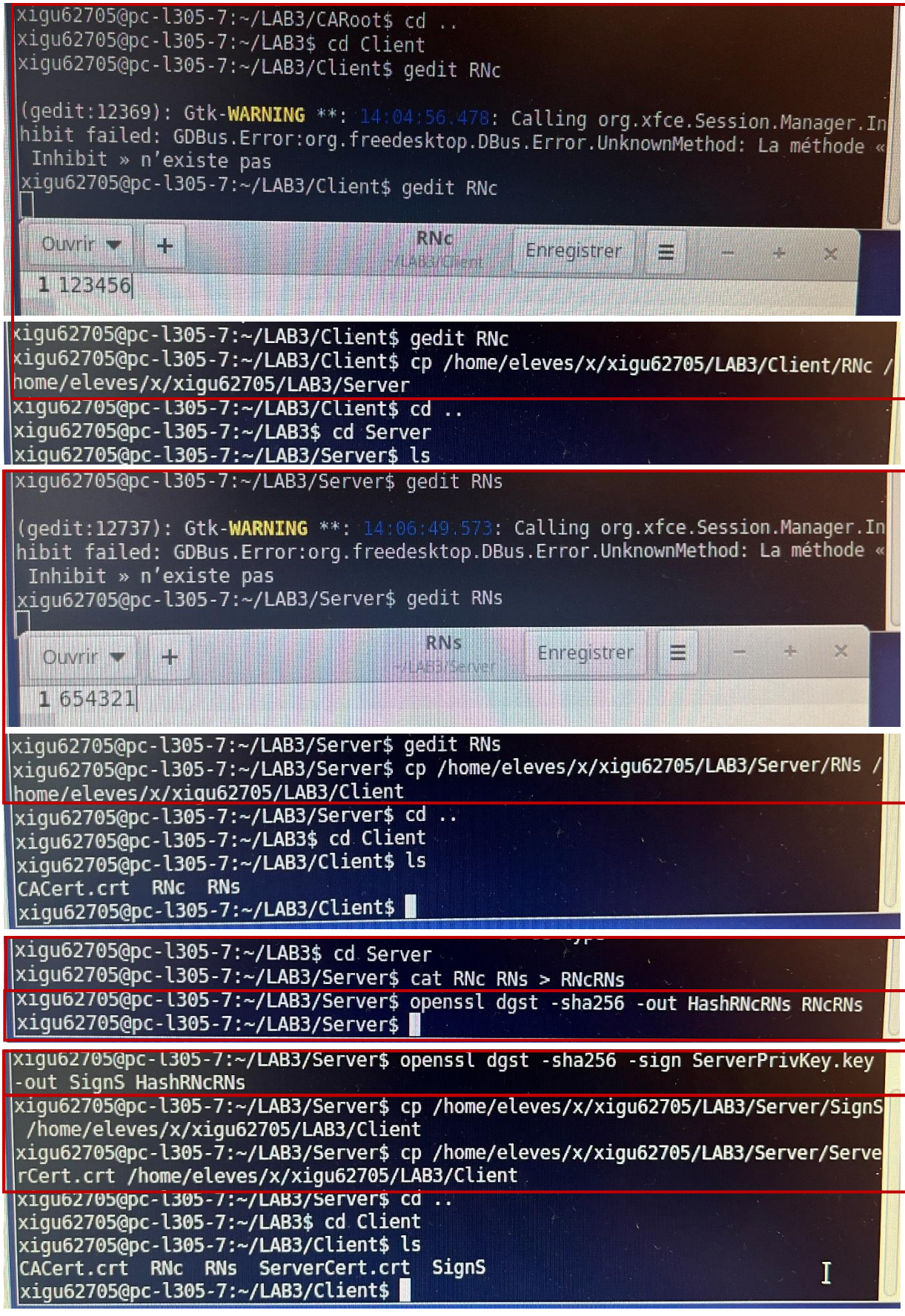
2.(d)


xigu62705@pc-l305-7:~/LAB3/Server$ openssl req -new -newkey rsa:4096 -nodes -key
out ServerPrivKey.key -out ServerRequest.csr
Generating a RSA private key
.....++++
.....++++
writing new private key to 'ServerPrivKey.key'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile de France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyCA
Organizational Unit Name (eg, section) []:MyCA
Common Name (e.g. server FQDN or YOUR name) []:MyCA
Email Address []:CAemail
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:password
xigu62705@pc-l305-7:~/LAB3/Server$ cp /home/eleves/x/xigu62705/LAB3/Server/Serve
rRequest.csr /home/eleves/x/xigu62705/LAB3/CARoot
2.(f)
xigu62705@pc-l305-7:~/LAB3/CARoot$ openssl x509 -req -in ServerRequest.csr -CA C
ACert.crt -CAkey CARrivKey.key -CAcreateserial -out ServerCert.crt -days 365
Signature ok
subject=C = FR, ST = Ile de France, L = Paris, O = MyCA, OU = MyCA, CN = MyCA, e
mailAddress = CAemail
Getting CA Private Key
Can't open CARrivKey.key for reading, No such file or directory
140469714433856:error:02001002:system library:fopen:No such file or directory:cr
ypto/bio/bss_file.c:69:fopen('CARrivKey.key','r')
140469714433856:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio
/bss_file.c:76:
unable to load CA Private Key
xigu62705@pc-l305-7:~/LAB3/CARoot$ openssl x509 -req -in ServerRequest.csr -CA C
ACert.crt -CAkey CAPrivKey.key -CAcreateserial -out ServerCert.crt -days 365
Signature ok
subject=C = FR, ST = Ile de France, L = Paris, O = MyCA, OU = MyCA, CN = MyCA, e
mailAddress = CAemail
Getting CA Private Key
2.(g)
xigu62705@pc-l305-7:~/LAB3/CARoot$ cp /home/eleves/x/xigu62705/LAB3/CARoot/Serve
rCert.crt /home/eleves/x/xigu62705/LAB3/Server
2.(g)
xigu62705@pc-l305-7:~/LAB3/CARoot$ cp /home/eleves/x/xigu62705/LAB3/CARoot/CACer
t.crt /home/eleves/x/xigu62705/LAB3/Server
xigu62705@pc-l305-7:~/LAB3/CARoot$ cp /home/eleves/x/xigu62705/LAB3/CARoot/CACer
t.crt /home/eleves/x/xigu62705/LAB3/Client
2.(h)
xigu62705@pc-l305-7:~/LAB3/CARoot$

2 Exercise “TLS Protocol”



```
xigu62705@pc-l305-7:~/LAB3/Client$ openssl verify -CAfile CACert.crt ServerCert.
crt
ServerCert.crt: OK
xigu62705@pc-l305-7:~/LAB3/Client$
```

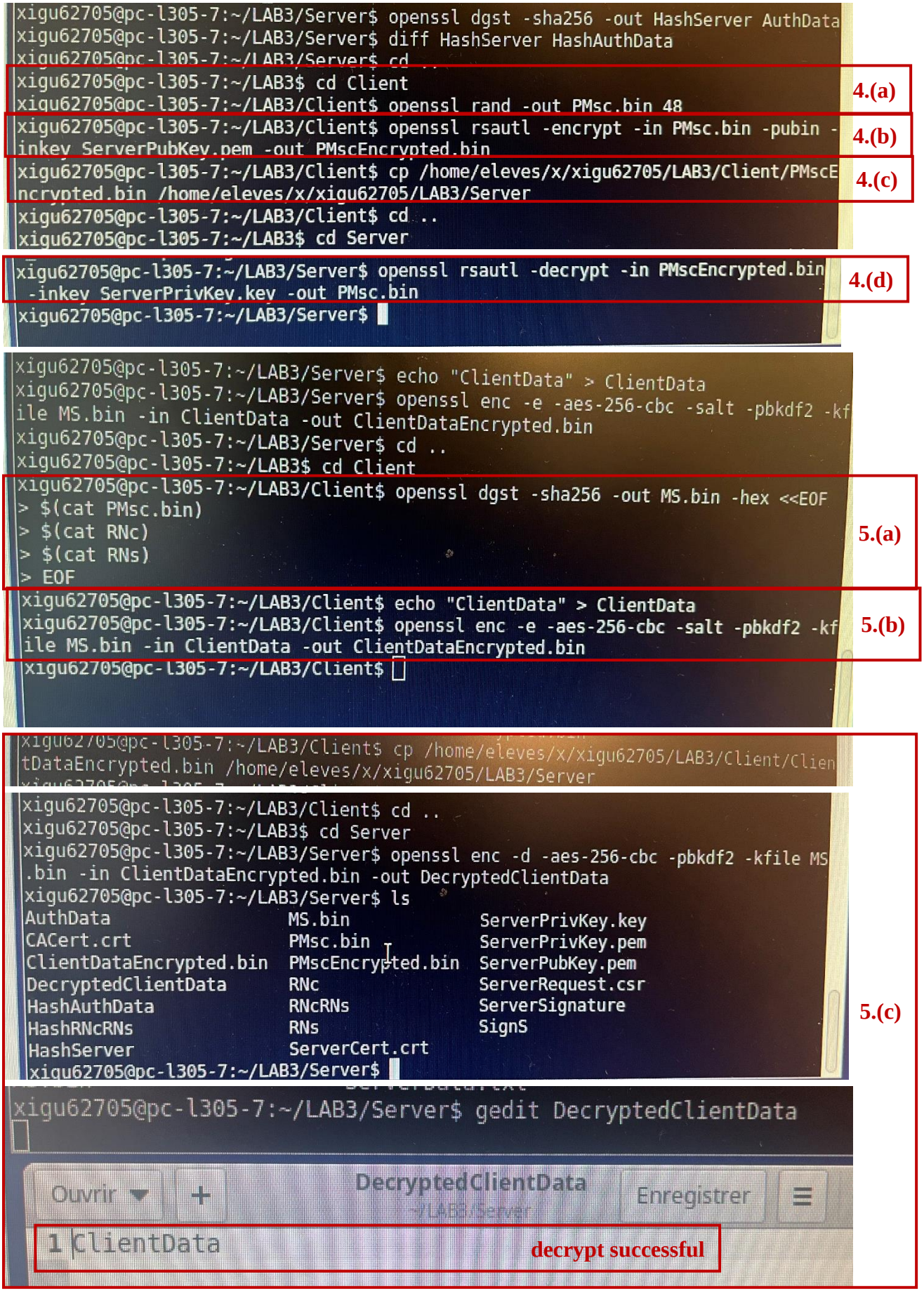
3.(e)

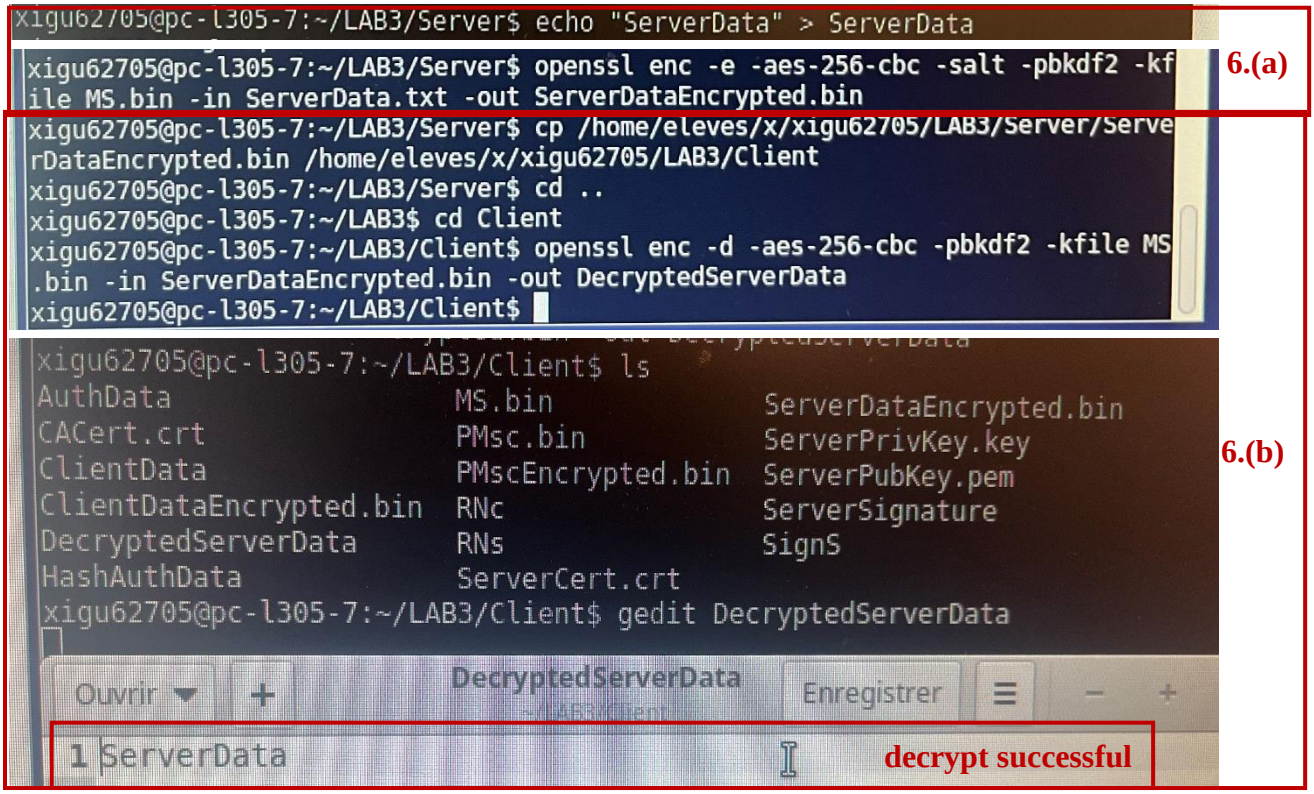
```
CACert.crt  RNc  RNs  ServerCert.crt  Sign$
xigu62705@pc-l305-7:~/LAB3/Client$ openssl verify -CAfile CACert.crt ServerCert.
crt
ServerCert.crt: OK
xigu62705@pc-l305-7:~/LAB3/Client$ openssl x509 -pubkey -noout -in ServerCert.cr
t > ServerPubKey.pem
xigu62705@pc-l305-7:~/LAB3/Client$ echo "This is the content of AuthData" > Auth
Data
xigu62705@pc-l305-7:~/LAB3/Client$ cat AuthData
This is the content of AuthData
xigu62705@pc-l305-7:~/LAB3/Client$ openssl dgst -sha256 -out HashAuthData AuthDt
a
AuthDta: No such file or directory
xigu62705@pc-l305-7:~/LAB3/Client$ openssl dgst -sha256 -out HashAuthData AuthDa
ta
xigu62705@pc-l305-7:~/LAB3/Client$ cat HashAuthData
SHA256(AuthData)= 0de15aaae3a07a2a80b6d701d459b982a81f0fd46e3db66edd041a74a46b09
3d
xigu62705@pc-l305-7:~/LAB3/Client$ openssl rsautl -sign -in HashAuthData -inkey
ServerPubKey.pem -out ServerSignature
unable to load Private Key
139968416565056:error:0909006C:PEM routines:get_name:no start line:crypto/pem/pe
m_lib.c:745:Expecting: ANY PRIVATE KEY
xigu62705@pc-l305-7:~/LAB3/Client$ openssl rsautl -sign -in HashAuthData -inkey
ServerPubKey.key -out ServerSignature
Can't open ServerPubKey.key for reading, No such file or directory
140305119586112:error:02001002:system library:fopen:No such file or directory:cr
ypto/bio/bss_file.c:69:fopen('ServerPubKey.key','r')
140305119586112:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio
/bss_file.c:76:
unable to load Private Key
xigu62705@pc-l305-7:~/LAB3/Client$ openssl rsautl -sign -in HashAuthData -inkey
ServerPrivKey.key -out ServerSignature
xigu62705@pc-l305-7:~/LAB3/Client$ cat ServerSignature
@@\@@@@@@@"@=@@N@@@@jUYu@@@@
                              /@@7@@&Q@Q@! S@xR   j}@@;@
@@@/x(72@@w@@m@@@IH@@@$4@t7@@P@zp@v@@@@t@F@H+@@j@!@@@@@@@_?@@n2@}俯@@@@]@1i@X@@@
P@K@y!@@@@t@8L@t@@@mC@n@Z1@K@@[)a@@\!@@
;Ï@@d_@{@PZ@@W@@v@`i0PU@cSJt>Q#@5O"@1@/@@@aT%@,@@\
                                                  m
N?xè@@@@@"@''{ε&@@@u@@@@@C@L%l<@@7@_{@U@E@)pV@@@@#È@"T@@j@@@R@@b@@L@4@n9(@v=@]3^
ʁB@@@@@CX@b@j@@k'@@l@e@@@@ˇRς,@@@@`!@T3@@2Vg@V@-@]<@8Z@@g@@4@@r@@@y@a}h8ò@Γ@@@@&
xigu62705@pc-l305-7:~/LAB3/Client$ cp /home/eleves/x/xigu62705/LAB3/Client/Serve
rSignature /home/eleves/x/xigu62705/LAB3/Server
xigu62705@pc-l305-7:~/LAB3/Client$ cd LAB3
bash: cd: LAB3: Aucun fichier ou dossier de ce type
xigu62705@pc-l305-7:~/LAB3/Client$ cd ..
xigu62705@pc-l305-7:~/LAB3$ cp /home/eleves/x/xigu62705/LAB3/Client/AuthData /ho
me/eleves/x/xigu62705/LAB3/Server
xigu62705@pc-l305-7:~/LAB3$ cd Server
xigu62705@pc-l305-7:~/LAB3/Server$ openssl rsautl -verify -in ServerSignature -p
ubin -inkey ServerPrivKey.key -out HashAuthData
unable to load Public Key
xigu62705@pc-l305-7:~/LAB3/Server$ openssl rsautl -verify -in ServerSignature -p
ubin -inkey ServerPrivKey.pem -out HashAuthData
RSA operation error
139748771997504:error:0407008A:rsa routines:RSA_padding_check_PKCS1_type_1:inval
id padding:crypto/rsa/rsa_pk1.c:67:
139748771997504:error:04067072:rsa routines:rsa_ossl_public_decrypt:padding chec
k failed:crypto/rsa/rsa_ossl.c:588:
xigu62705@pc-l305-7:~/LAB3/Server$ openssl dgst -sha256 -out HashServer AuthData
xigu62705@pc-l305-7:~/LAB3/Server$ diff HashServer HashAuthData
xigu62705@pc-l305-7:~/LAB3/Server$
```

3.(f)-1

3.(f)-2

Comparison successful


xigu62705@pc-l305-7:~/LAB3/Server$ openssl dgst -sha256 -out HashServer AuthData
xigu62705@pc-l305-7:~/LAB3/Server$ diff HashServer HashAuthData
xigu62705@pc-l305-7:~/LAB3/Server$ cd ..
xigu62705@pc-l305-7:~/LAB3$ cd Client
xigu62705@pc-l305-7:~/LAB3/Client$ openssl rand -out PMsc.bin 48
4.(a)
xigu62705@pc-l305-7:~/LAB3/Client$ openssl rsautl -encrypt -in PMsc.bin -pubin -
inkey ServerPubKey.pem -out PMscEncrypted.bin
4.(b)
xigu62705@pc-l305-7:~/LAB3/Client$ cp /home/eleves/x/xigu62705/LAB3/Client/PMscE
ncrypted.bin /home/eleves/x/xigu62705/LAB3/Server
4.(c)
xigu62705@pc-l305-7:~/LAB3/Client$ cd ..
xigu62705@pc-l305-7:~/LAB3$ cd Server
xigu62705@pc-l305-7:~/LAB3/Server$ openssl rsautl -decrypt -in PMscEncrypted.bin
-inkey ServerPrivKey.key -out PMsc.bin
4.(d)
xigu62705@pc-l305-7:~/LAB3/Server$
xigu62705@pc-l305-7:~/LAB3/Server$ echo "ClientData" > ClientData
xigu62705@pc-l305-7:~/LAB3/Server$ openssl enc -e -aes-256-cbc -salt -pbkdf2 -kf
ile MS.bin -in ClientData -out ClientDataEncrypted.bin
xigu62705@pc-l305-7:~/LAB3/Server$ cd ..
xigu62705@pc-l305-7:~/LAB3$ cd Client
xigu62705@pc-l305-7:~/LAB3/Client$ openssl dgst -sha256 -out MS.bin -hex <<EOF
> $(cat PMsc.bin)
> $(cat RNc)
> $(cat RNs)
> EOF
5.(a)
xigu62705@pc-l305-7:~/LAB3/Client$ echo "ClientData" > ClientData
xigu62705@pc-l305-7:~/LAB3/Client$ openssl enc -e -aes-256-cbc -salt -pbkdf2 -kf
ile MS.bin -in ClientData -out ClientDataEncrypted.bin
5.(b)
xigu62705@pc-l305-7:~/LAB3/Client$
xigu62705@pc-l305-7:~/LAB3/Client$ cp /home/eleves/x/xigu62705/LAB3/Client/Clien
tDataEncrypted.bin /home/eleves/x/xigu62705/LAB3/Server
xigu62705@pc-l305-7:~/LAB3/Client$ cd ..
xigu62705@pc-l305-7:~/LAB3$ cd Server
xigu62705@pc-l305-7:~/LAB3/Server$ openssl enc -d -aes-256-cbc -pbkdf2 -kfile MS
.bin -in ClientDataEncrypted.bin -out DecryptedClientData
xigu62705@pc-l305-7:~/LAB3/Server$ ls
AuthData MS.bin ServerPrivKey.key
CACert.crt PMsc.bin ServerPrivKey.pem
ClientDataEncrypted.bin PMscEncrypted.bin ServerPubKey.pem
DecryptedClientData RNc ServerRequest.csr
HashAuthData RNcRNs ServerSignature
HashRNcRNs RNs SignS
HashServer ServerCert.crt
xigu62705@pc-l305-7:~/LAB3/Server$
5.(c)
xigu62705@pc-l305-7:~/LAB3/Server$ gedit DecryptedClientData
Ouvrir
DecryptedClientData
Enregistrer
1 ClientData
decrypt successful


xigu62705@pc-l305-7:~/LAB3/Server$ echo "ServerData" > ServerData
xigu62705@pc-l305-7:~/LAB3/Server$ openssl enc -e -aes-256-cbc -salt -pbkdf2 -kf
ile MS.bin -in ServerData.txt -out ServerDataEncrypted.bin
xigu62705@pc-l305-7:~/LAB3/Server$ cp /home/eleves/x/xigu62705/LAB3/Server/Serve
rDataEncrypted.bin /home/eleves/x/xigu62705/LAB3/Client
xigu62705@pc-l305-7:~/LAB3/Server$ cd ..
xigu62705@pc-l305-7:~/LAB3$ cd Client
xigu62705@pc-l305-7:~/LAB3/Client$ openssl enc -d -aes-256-cbc -pbkdf2 -kfile MS
.bin -in ServerDataEncrypted.bin -out DecryptedServerData
xigu62705@pc-l305-7:~/LAB3/Client$
6.(a)
xigu62705@pc-l305-7:~/LAB3/Client$ ls
AuthData
CACert.crt
ClientData
ClientDataEncrypted.bin
DecryptedServerData
HashAuthData
MS.bin
PMsc.bin
PMscEncrypted.bin
RNc
RNs
ServerCert.crt
ServerDataEncrypted.bin
ServerPrivKey.key
ServerPubKey.pem
ServerSignature
SignS
xigu62705@pc-l305-7:~/LAB3/Client$ gedit DecryptedServerData
Ouvrir
DecryptedServerData
Enregistrer
1 ServerData
decrypt successful
6.(b)