Exercise 2

Reeshabh Kumar Ranjan
2017086

1 SET-1

1.1 PART-1

1.1.1 CREATING CA CERTIFICATES AND KEYS

Command used:

```
openssl req -newkey rsa:2048 -nodes -keyout ca.key -x509 -days 365 -out ca.crt
```

Command description:

1. req: OpenSSL utility to generate certificate and certificate requests.
2. -newkey rsa:2048 -nodes -keyout ca.key: Generate a 2048 bit long private key, and do not encrypt it with a password. Save it as ca.key.
3. -x509 -days 365 -out ca.crt: Create a self-signed certificate, valid for 1 year since time of creation. Save it as ca.crt.

1.1.2 CREATING SERVER/CLIENT CERTIFICATE REQUEST AND KEYS

Command used:

```
openssl req -newkey rsa:2048 -nodes -keyout server.key -x509 -out server.csr
```

Command description: New commands:

1. -out: save the certificate signing-request as server.csr.

Note: Repeat the same step for client.

1.1.3 SIGNING SERVER/CLIENT CERTIFICATE REQUEST USING CA'S PRIVATE KEY

Command used:

```
openssl x509 -req -in ../server/server.crt -CA ca.crt -CAkey ca.key
-CAcreateserial -out ../server/server.crt -days 365 -sha256
```

Command description: New commands:

1. x509: Use certificate signing utility.
2. -req: Means that the current command works with a certificate signing request.
3. -CA: To specify the CA's certificate file to be used for signing.
4. -CAkey: To specify the CA's private key to be used for signing.
5. -CAcreateserial: Generates a file containing the serial-number associated with the current CSR.
6. -out: Specifies the output file path (for the signed certificate).
7. -days: Specifies the number of days of validity
8. -sha256: Specifies to use SHA-256 as the hashing algorithm.

Note: Repeat the same steps as above for client.

1.2 PART-2

1.2.1 LAUNCHING THE SERVER

Command used:

```
openssl s_server -accept 9999 -cert server.crt -key server.key -debug
```

Command description:

1. s_server: Use the server provided in the OpenSSL package.
2. -accept: specify the port on which to listen to connections
3. -cert: specify the path to server certificate file to be presented to incoming connections
4. -key: the private key to use for the communication
5. -debug: output debug information to console

1.2.2 LAUNCHING THE CLIENT

Command used:

```
openssl s_client -connect localhost:9999 -verify 5 -CAfile ../ca/ca.crt
-debug
```

Command description: (new commands)

1. s_client: use the client service provided in OpenSSL package.
2. -connect: specifies the network address and port to connect to.
3. -verify 5: tries to verify the server, specifying the verification depth (chain of CA authorities).
4. -CAfile: specifies the CA's certificate to use

1.3 PART-3

1.3.1 LAUNCHING THE SERVER

Command used:

```
openssl s_server -accept 9999 -cert server.crt -key server.key -Verify 5
-CAfile ../ca/ca.crt -debug
```

Command description: (new commands)

1. -Verify 5: specifies that the client must present its certificate, with the verification depth of 5.

1.3.2 LAUNCHING THE CLIENT

Command used:

```
openssl s_client -connect 9999 -cert client.crt -key client.key -verify 5
-CAfile ../ca/ca.crt -debug
```

1.4 PART-4

1.4.1 LAUNCHING THE SERVER

Command used:

```
openssl s_server -accept 9999 –www -cert server.crt -key server.key -Verify 5
-CAfile ../ca/ca.crt -debug
```

Command description: (new commands)

1. -www: The status message sent back to the client about various protocol details is in HTML format, hence a browser can act like a client.

1.4.2 SETTING UP THE BROWSER: FIREFOX

Steps to load the certificates:

1. Go to Firefox preferences/settings.
2. Search for "certificates".
3. Click "View Certificates".
4. Import CA's certificate in Authority certificates.
    - Check "Trust this CA to identify websites."
    - Check "Trust this CA to identify email users."
    - Click OK.
5. Import client's certificate in "Your Certificates".

Now open this link in Firefox:

```
https://localhost:9999
```

When prompted, provide the client-certificate. After this, a response from the server will be displayed in the browser, containing information regarding certificates and ciphers supported. The "lock" icon on the top shows that the connection is secure, hence the CA is trusted.

2 SET-2

2.1 PART-1

2.1.1 INSTALLING ECRYPTS

Command used:

```
sudo apt install ecryptfs-utils
```

2.2 PART-2

2.2.1 CREATING SUB-DIRECTORY

Command used:

```
mkdir encrypted
```

2.3 PART-3, 4

2.3.1 MOUNTING THE SUB-DIRECTORY USING ECRYPTFS

Command used:

```
sudo mount -t ecryptfs encrypted/ encrypted
```

Command description:

1. -t ecryptfs: Specifies the filesystem type, here as "ecryptfs".
2. encrypted/ encrypted: The first location is treated as a device, and the second one is treated as the directory to which the device is mounted.

2.4 PART -5

2.4.1 CREATING FILE TEMP1

Command used:

```
echo "Hello world!" > temp1
```

2.5 PART-6

2.5.1 UNMOUNTING DIRECTORY

Command used:

```
sudo umount encrypted
```

Command description:

1. Takes in the directory from which the mounted filesystem has to be detached from.

3 SET-3

3.1 PART-1

3.1.1 CREATING REESHABH.1.TXT

Command used

```
touch reeshabh.1.txt
```

Used vim text-editor to add info.

3.2 PART-2

3.2.1 ENCRYPTING THE CONTENTS OF THE FILE

Commands used:

```
$ openssl enc -pbkdf2 -aes-256-cbc -pass pass:reeshabh17086 -in
  reeshabh.1.txt -out reeshabh.2.txt.encr
$ rm reeshabh.1.txt
$ mv reeshabh.1.txt.encr reeshabh.1.txt
```

Command description:

1. enc: Use the symmetric cipher service in OpenSSL package.
2. -pbkdf2: Use the password based key-derivation function algorithm version 2.
3. -aes-256-cbc: Use AES-256 CBC mode as encryption algorithm.
4. -pass: Specify the password for deriving salt, key and iv.
5. -in: The file to be encrypted.
6. -out: The output path for the encrypted file.

Note: The passphrase used is "reeshabh17086".

3.3 PART-3

3.3.1 CREATING RANDOM SALT, KEY AND IV

Script used

```
#!bin/bash
echo -n "salt: "
openssl rand -hex 8
echo -n "key: "
openssl rand -hex 32
echo -n "iv: "
openssl rand -hex 16
```

The above code generates a salt of 16 bytes, key of 64 bytes and IV of 32 bytes. It is saved into the file credentials.txt.

3.3.2 ENCRYPTING USING THE GENERATED CREDENTIALS

Command used:

```
openssl enc -aes-256-cbc -salt -S <salt> -K <key> -iv <iv> -in
reeshabh.2.txt -out reeshabh.2.txt.encr
```

Command description: (new commands)

1. -salt: Specifies that a salt needs to be used.
2. -S: Specifies the salt to be used.
3. : -iv: Specifies the initialization-vector to be used.

3.4 PART-4

3.4.1 DECRYPTING REESHABH.1.TXT

Command used:

```
openssl enc -pbkdf2 -aes-256-cbc -d -pass pass:reeshabh17086 -in
reeshabh.1.txt -out reeshabh.1.txt.decr
```

Command description: (new commands)

1. -d: Specifies decryption. This automatically reads the salt present in the header of the file to be decrypted, and uses the salt and the password to derive the key and the iv.

3.4.2 DECRYPTING REESHABH.2.TXT

Command used:

```
openssl enc -aes-256-cbc -d -salt -S <salt> -K <key> -iv <iv> -in
reeshabh.2.txt -out reeshabh.2.txt.decr
```