



OPENSSL AND POST-QUANTUM OPENSSL

[Group Assignment - 2]



CS6/745: Modern Cryptography

Submitted to: Dr. Yuliang Zheng

Submitted by:

1. Joyanta Mondal (jmondal)
2. Shouzab Khan (Skhan6)
3. Pooja Srinivas (psriniva)
4. Rajendra Mohan (rnavulur)

March 06, 2024

INTRODUCTION:

Transport Layer Security (TLS) protocol is a foundational element in ensuring the confidentiality and integrity of data in transit across networks. OpenSSL, the most significant library for implementing TLS, has long been the standard for securing communications on the internet. However, with the emerging threat of quantum computers capable of breaking traditional cryptographic algorithms, the imperative for quantum-resistant cryptography has never been more pressing.

In response to this emerging threat, the Open Quantum Safe project has introduced OQS-OpenSSL, an innovative extension of the OpenSSL library that incorporates post-quantum (PQ) cryptographic capabilities.

In this report, we explore the integration and application of both OpenSSL and OQS-OpenSSL. We use Mac devices to integrate these.

Part I. OpenSSL:

Download and Extract the OpenSSL Source Code:

First of all, we check on our computer which version is already installed with us. We have to use the command "openssl version" to see the version. It initially shows the "OpenSSL 1.1.1q 5 Jul 2022" version, but for this task, we update the openssl to its latest version. For that, we use the command

```
"wget https://www.openssl.org/source/openssl-3.2.1g.tar.gz"
```

After running this command, openssl in the computer updates to version 3.2.1.



The screenshot shows a terminal window titled "shouzabkhan -- zsh -- 137x24". The command "wget https://www.openssl.org/source/openssl-3.2.1.tar.gz" is run, and the output shows the progress of the download. The file is saved as "openssl-3.2.1.tar.gz.2". The download completes at 16.91M with a speed of 4.90MB/s in 3.5s. The terminal prompt "(base)" is visible at the end.

```
Last login: Sun Mar 10 07:23:48 on ttys000
[base] shouzabkhan@Shouzabs-Laptop ~ % wget https://www.openssl.org/source/openssl-3.2.1.tar.gz
--2024-03-10 08:07:03-- https://www.openssl.org/source/openssl-3.2.1.tar.gz
Resolving www.openssl.org (www.openssl.org)... 2600:1901:0:1812::, 34.36.58.177
Connecting to www.openssl.org (www.openssl.org)|2600:1901:0:1812::|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17733249 (17M) [application/x-tar]
Saving to: 'openssl-3.2.1.tar.gz.2'

openssl-3.2.1.tar.gz.2          100%[=====] 16.91M  4.90MB/s   in 3.5s

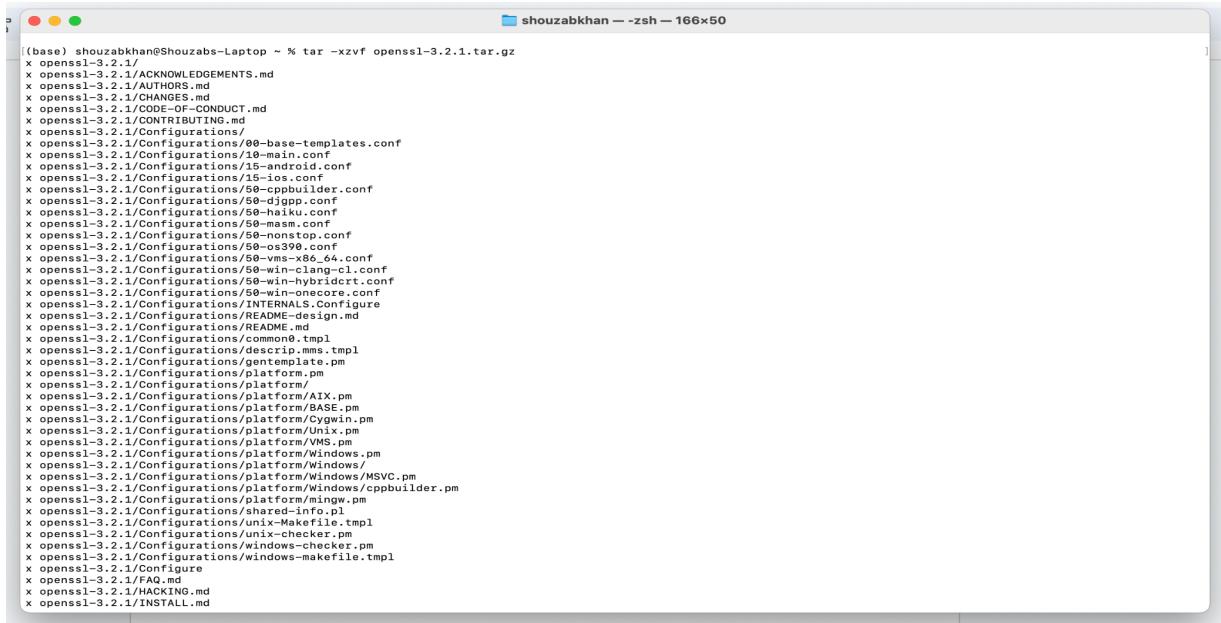
2024-03-10 08:07:08 (4.78 MB/s) - 'openssl-3.2.1.tar.gz.2' saved [17733249/17733249]

(base) shouzabkhan@Shouzabs-Laptop ~ %
```

Now when it is updated, we have to extract the source code tarball before running the command "./config".

For that, we run the command

```
"tar -xzvf openssl-3.2.1.tar.gz"
```



The screenshot shows a terminal window titled "shouzabkhan -- zsh -- 166x50". The command entered was "% tar -xzvf openssl-3.2.1.tar.gz". The output lists numerous files extracted from the tar archive, including configuration files for various platforms like Win32, Win64, AIX, BASE, Cygwin, UNIX, and Windows, along with AUTHORS, CHANGES, and CODE-OF-CONDUCT files.

```
(base) shouzabkhan@Shouzabs-Laptop ~ % tar -xzvf openssl-3.2.1.tar.gz
x openssl-3.2.1/
x openssl-3.2.1/ACKNOWLEDGEMENTS.md
x openssl-3.2.1/AUTHORS.md
x openssl-3.2.1/CHANGES.md
x openssl-3.2.1/CODE-OF-CONDUCT.md
x openssl-3.2.1/CONTRIBUTORS.md
x openssl-3.2.1/Configurations/
x openssl-3.2.1/Configurations/00-base-templates.conf
x openssl-3.2.1/Configurations/10-main.conf
x openssl-3.2.1/Configurations/15-android.conf
x openssl-3.2.1/Configurations/15-ios.conf
x openssl-3.2.1/Configurations/15-macosx-apple.conf
x openssl-3.2.1/Configurations/50-dipp.conf
x openssl-3.2.1/Configurations/50-haiku.conf
x openssl-3.2.1/Configurations/50-masm.conf
x openssl-3.2.1/Configurations/50-nonstop.conf
x openssl-3.2.1/Configurations/50-os390.conf
x openssl-3.2.1/Configurations/50-pcbsd.conf
x openssl-3.2.1/Configurations/50-win-cl.conf
x openssl-3.2.1/Configurations/50-win-hybridcert.conf
x openssl-3.2.1/Configurations/INTERNALS.Configure
x openssl-3.2.1/Configurations/openssl-internal-design.md
x openssl-3.2.1/Configurations/README.md
x openssl-3.2.1/Configurations/common0.tmpl
x openssl-3.2.1/Configurations/descrip.mms.tmpl
x openssl-3.2.1/Configurations/gentemplate.pm
x openssl-3.2.1/Configurations/platform.pm
x openssl-3.2.1/Configurations/platform/Win32.pm
x openssl-3.2.1/Configurations/platform/AIX.pm
x openssl-3.2.1/Configurations/platform/BASE.pm
x openssl-3.2.1/Configurations/platform/Cygwin.pm
x openssl-3.2.1/Configurations/platform/UNIX.pm
x openssl-3.2.1/Configurations/platform/Win64.pm
x openssl-3.2.1/Configurations/platform/Windows.pm
x openssl-3.2.1/Configurations/platform/Windows/MSVC.pm
x openssl-3.2.1/Configurations/platform/Windows/cppbuilder.pm
x openssl-3.2.1/Configurations/shell-init.p1
x openssl-3.2.1/Configurations/unix-Makefile.tmpl
x openssl-3.2.1/Configurations/unix-checker.pm
x openssl-3.2.1/Configurations/windows-checker.pm
x openssl-3.2.1/Configurations/windows-makefile.tmpl
x openssl-3.2.1/FAQ
x openssl-3.2.1/FAQ.md
x openssl-3.2.1/HACKING.md
x openssl-3.2.1/INSTALL.md
```

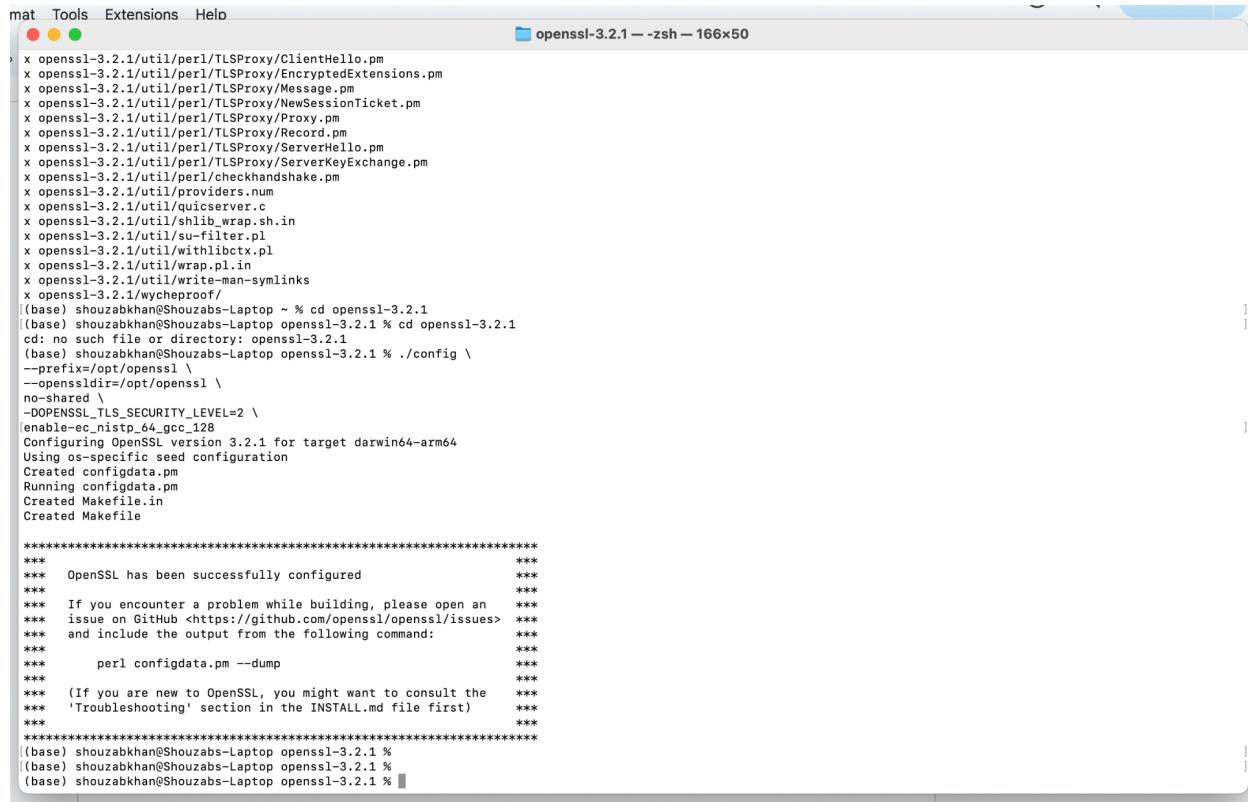
Now after that, we change directory to the extracted OpenSSL directory:

```
(base) shouzabkhan@Shouzabs-Laptop ~ % cd openssl-3.2.1
(base) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 % cd openssl-3.2.1
```

Till this step, we complete downloading and extracting the OpenSSL source code. After this step we have to configure, build and install OpenSSL.

Configure, Build and Install OpenSSL:

Now, we configure OpenSSL by running the `./config`. By this, OpenSSL will be successfully configured as shown in the screenshot of the terminal below.



The screenshot shows a terminal window titled "openssl-3.2.1 - zsh - 166x50". The terminal displays the configuration process of OpenSSL 3.2.1. The user runs "cd openssl-3.2.1" and then "./config". The configuration command outputs several lines of configuration options, including --prefix=/opt/openssl, --openssldir=/opt/openssl, and no-shared. It also sets the security level to 2 and enables the ec_nistp_64_gcc_128 option. The process continues with "Configuring OpenSSL version 3.2.1 for target darwin64-arm64". It creates configdata.pm, runs configdata.pm, and creates Makefile.in and Makefile. Finally, it outputs a success message: "OpenSSL has been successfully configured" and provides troubleshooting instructions.

```
mat Tools Extensions Help
openssl-3.2.1 - zsh - 166x50
x openssl-3.2.1/util/perl/TLSProxy/ClientHello.pm
x openssl-3.2.1/util/perl/TLSProxy/EncryptedExtensions.pm
x openssl-3.2.1/util/perl/TLSProxy/Message.pm
x openssl-3.2.1/util/perl/TLSProxy/NewSessionTicket.pm
x openssl-3.2.1/util/perl/TLSProxy/Proxy.pm
x openssl-3.2.1/util/perl/TLSProxy/Record.pm
x openssl-3.2.1/util/perl/TLSProxy/ServerHello.pm
x openssl-3.2.1/util/perl/TLSProxy/ServerKeyExchange.pm
x openssl-3.2.1/util/perl/checkhandshake.pm
x openssl-3.2.1/util/providers.num
x openssl-3.2.1/util/quicserver.c
x openssl-3.2.1/util/shlib_wrap.sh.in
x openssl-3.2.1/util/su-filter.pl
x openssl-3.2.1/util/withlibctx.pl
x openssl-3.2.1/util/wrap.pl.in
x openssl-3.2.1/util/write-man-symlinks
x openssl-3.2.1/wycheproof/
[(base) shouzabkhan@Shouzabs-Laptop ~ % cd openssl-3.2.1
[(base) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 % cd openssl-3.2.1
cd: no such file or directory: openssl-3.2.1 %
(base) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 % ./config \
--prefix=/opt/openssl \
--openssldir=/opt/openssl \
no-shared \
-DOPENSSL_TLS_SECURITY_LEVEL=2 \
enable-ec_nistp_64_gcc_128
Configuring OpenSSL version 3.2.1 for target darwin64-arm64
Using os-specific seed configuration
Created configdata.pm
Running configdata.pm
Created Makefile.in
Created Makefile

*****
*** OpenSSL has been successfully configured ***
*** If you encounter a problem while building, please open an issue on GitHub <https://github.com/openssl/openssl/issues>
*** and include the output from the following command:
*** perl configdata.pm --dump
*** (If you are new to OpenSSL, you might want to consult the 'Troubleshooting' section in the INSTALL.md file first)
***
*****[(base) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 %
[(base) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 %
(base) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 % ]]
```

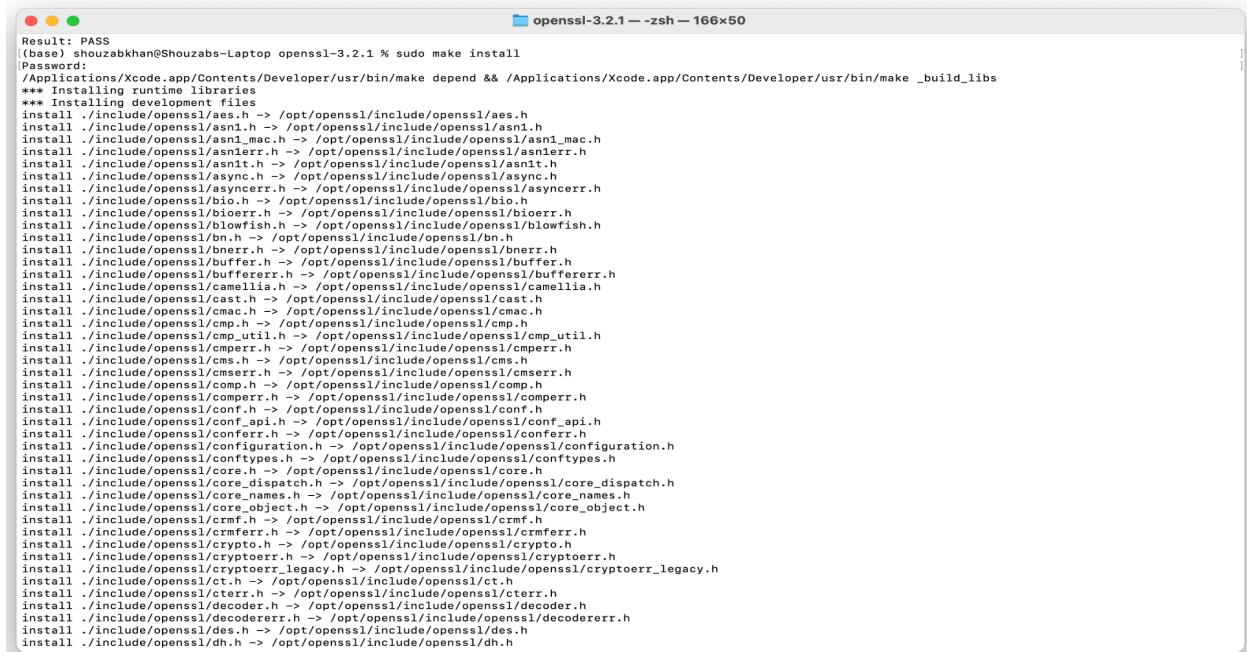
Now when the OpenSSL is configured, we have successfully built and installed OpenSSL. After giving the command “make” in the terminal, it starts building OpenSSL. Below the screenshot are showing the execution of command “make”

100

Now we run tests to ensure the build is successful. For that, we run the command "make test".

```
[ms] Tools Extensions Help  
openssl-3.2.1-zh - zsh - 166x50  
  
[bazu] ~
```

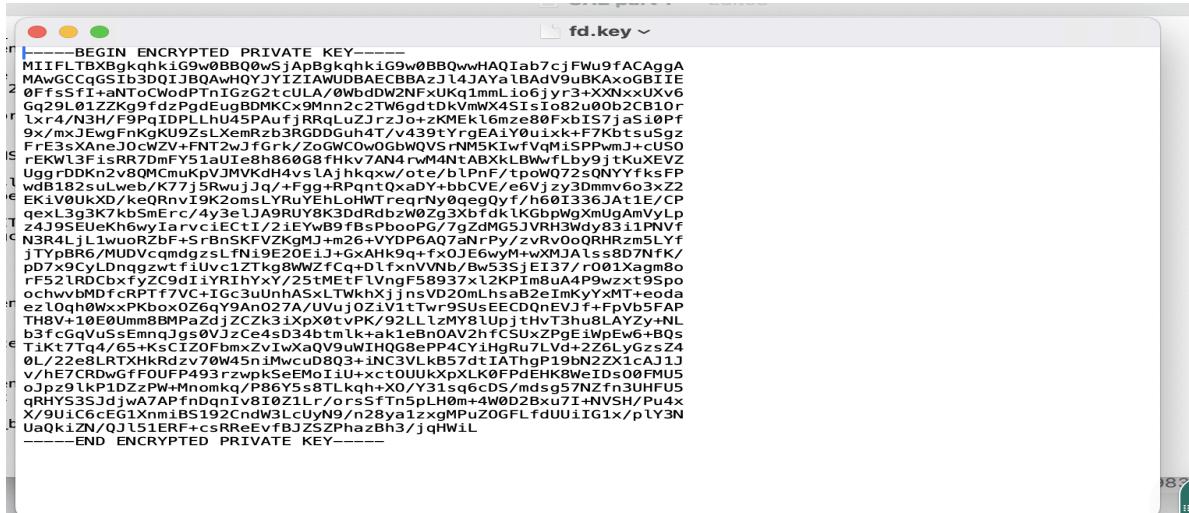
We can see that the result is passed when we run the tests. Now, the final step is to install OpenSSL. We put the command `"sudo make install"` after that it will require the password of your computer and it will start installing OpenSSL. Just shown below in the screenshot.



```
Result: PASS
(bash) shouzabkhan@Shouzabs-Laptop openssl-3.2.1 % sudo make install
[Password]
/Applications/Xcode.app/Contents/Developer/usr/bin/make depend && /Applications/Xcode.app/Contents/Developer/usr/bin/make _build_libs
*** Installing runtime libraries
*** Installing development files
install ./include/openssl/aes.h -> /opt/openssl/include/openssl/aes.h
install ./include/openssl/asn1.h -> /opt/openssl/include/openssl/asn1.h
install ./include/openssl/asn1_mac.h -> /opt/openssl/include/openssl/asn1_mac.h
install ./include/openssl/bioerr.h -> /opt/openssl/include/openssl/bioerr.h
install ./include/openssl/asn1t.h -> /opt/openssl/include/openssl/asn1t.h
install ./include/openssl/async.h -> /opt/openssl/include/openssl/async.h
install ./include/openssl/asyncerr.h -> /opt/openssl/include/openssl/asyncerr.h
install ./include/openssl/bio.h -> /opt/openssl/include/openssl/bio.h
install ./include/openssl/bioerr.h -> /opt/openssl/include/openssl/bioerr.h
install ./include/openssl/blowfish.h -> /opt/openssl/include/openssl/blowfish.h
install ./include/openssl/buffer.h -> /opt/openssl/include/openssl/buffer.h
install ./include/openssl/camellia.h -> /opt/openssl/include/openssl/camellia.h
install ./include/openssl/cast.h -> /opt/openssl/include/openssl/cast.h
install ./include/openssl/cmac.h -> /opt/openssl/include/openssl/cmac.h
install ./include/openssl/cmp.h -> /opt/openssl/include/openssl/cmp.h
install ./include/openssl/cmp_util.h -> /opt/openssl/include/openssl/cmp_util.h
install ./include/openssl/cmss.h -> /opt/openssl/include/openssl/cmss.h
install ./include/openssl/cmssr.h -> /opt/openssl/include/openssl/cmssr.h
install ./include/openssl/cms.h -> /opt/openssl/include/openssl/cms.h
install ./include/openssl/cmserr.h -> /opt/openssl/include/openssl/cmserr.h
install ./include/openssl/comp.h -> /opt/openssl/include/openssl/comp.h
install ./include/openssl/comppr.h -> /opt/openssl/include/openssl/comppr.h
install ./include/openssl/conf.h -> /opt/openssl/include/openssl/conf.h
install ./include/openssl/conf_api.h -> /opt/openssl/include/openssl/conf_api.h
install ./include/openssl/confferr.h -> /opt/openssl/include/openssl/confferr.h
install ./include/openssl/conftypes.h -> /opt/openssl/include/openssl/conftypes.h
install ./include/openssl/core.h -> /opt/openssl/include/openssl/core.h
install ./include/openssl/core_dispatch.h -> /opt/openssl/include/openssl/core_dispatch.h
install ./include/openssl/core_names.h -> /opt/openssl/include/openssl/core_names.h
install ./include/openssl/core_object.h -> /opt/openssl/include/openssl/core_object.h
install ./include/openssl/crmf.h -> /opt/openssl/include/openssl/crmf.h
install ./include/openssl/cryptoderr.h -> /opt/openssl/include/openssl/cryptoderr.h
install ./include/openssl/cryptoserr.h -> /opt/openssl/include/openssl/cryptoserr.h
install ./include/openssl/cryptoerr.h -> /opt/openssl/include/openssl/cryptoerr.h
install ./include/openssl/cryptoerr_legacy.h -> /opt/openssl/include/openssl/cryptoerr_legacy.h
install ./include/openssl/ct.h -> /opt/openssl/include/openssl/ct.h
install ./include/openssl/cterr.h -> /opt/openssl/include/openssl/cterr.h
install ./include/openssl/decoder.h -> /opt/openssl/include/openssl/decoder.h
install ./include/openssl/decodererr.h -> /opt/openssl/include/openssl/decodererr.h
install ./include/openssl/des.h -> /opt/openssl/include/openssl/des.h
install ./include/openssl/dh.h -> /opt/openssl/include/openssl/dh.h
```

Generate public - private key pairs and certificates:

Now we generate an RSA private key with a key size of 2048 bits and encrypt it using AES-128. The key is saved in a file named `fd.key`. We also protect it with a password called PEM phrase pass.



Now, we generate the public key so for that we run this command below:
openssl pkey -in fd.key -pubout -out fd-public.key

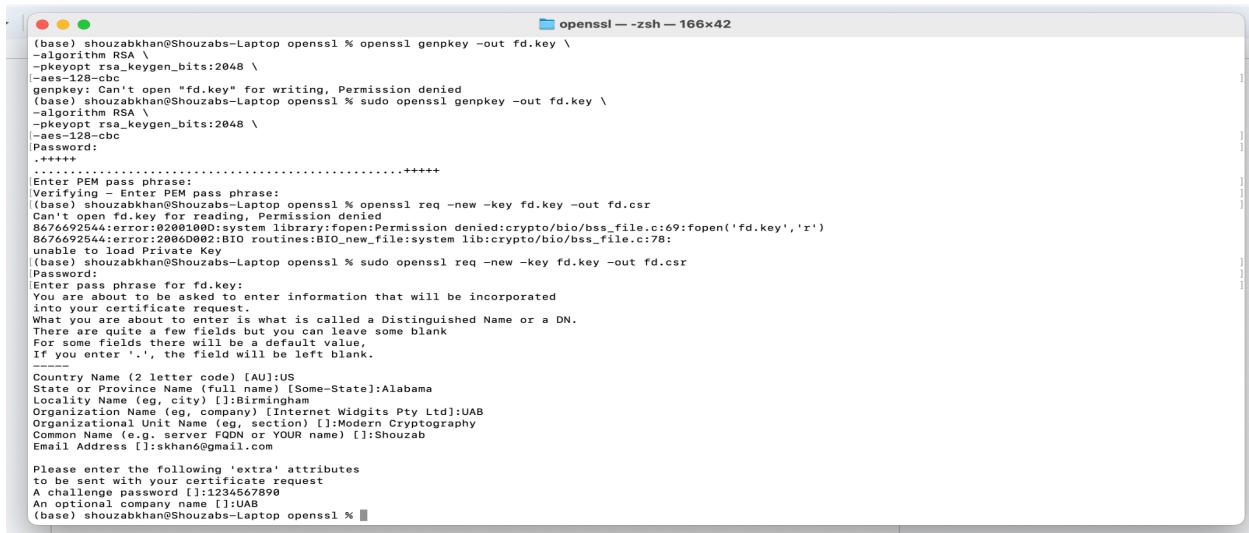
```
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo openssl pkey -in fd.key -pubout -out fd-public.key
Password:
Enter pass phrase for fd.key:
(base) shouzabkhan@Shouzabs-Laptop openssl % cat fd-public.key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAQCAQ8AMIIIBCgKCAQEAp7QYQDQPH7eptW+AawHz
ycQvO4s4sqemB1IvgCnYb08KZxvwShoWBcXHCFE38CpDtga8AVmUeBNLa/11BVFFP
E3rdruAAH1DeWfG1xHMSCRVr90CWxWqLLifyX6jU1C1EuTtTxUHS3wqC06i08Lu
YxDxBzmcplJFj4wxGx1wACzFD+XH2gtIsmmepCBxwqWV1AFNKjY0YRz0a+KKX
Xa8DZl1Df5xGkZ8646Mt6C4RvBzAxtRP906akkAvmSwnKY1KF/M11IF/sB2T2
4GVaRKTk1/KHGz9w7qgf2K+HnVyrb0WjWCQWc+uL3Gj7HAdGzcDgUQiowSWi0z3
EQIDAQAB
-----END PUBLIC KEY-----
(base) shouzabkhan@Shouzabs-Laptop openssl %
```

```
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo openssl genpkey -out fd.key \
-algorithm RSA \
-pkeyopt rsa_keygen_bits:2048 \
aes-128-cbc
Password:
.+++++
.....+++++
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
(base) shouzabkhan@Shouzabs-Laptop openssl %
```

- Generating Certificate Signing Request

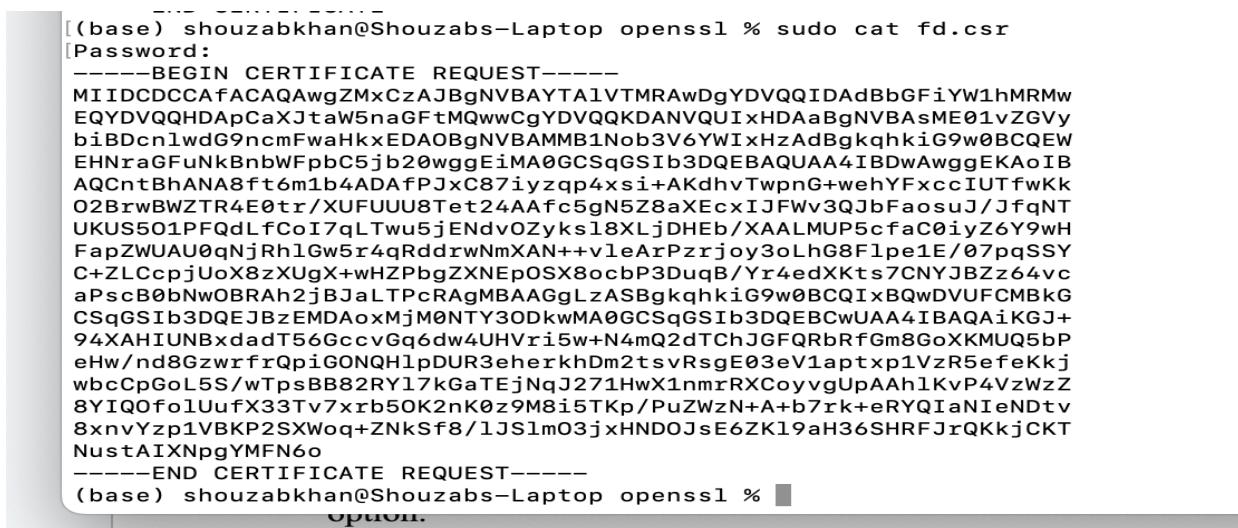
Now when the key is generated, we request for a certificate signing that key using the command "openssl req".

By this command, the terminal will ask for different questions for signing the certificate like PEM pass phrase, country name, state name, locality name, Organization name, Organization unit name, common name, email address, challenge password and optional company name. These were the questions through which the certificate will be signed.



```
(base) shouzabkhan@Shouzabs-Laptop openssl % openssl genpkey -out fd.key \
-algorithm RSA \
-pkeyopt rsa_keygen_bits:2048 \
-aes-128-cbc
genpkey: Can't open "fd.key" for writing, Permission denied
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo openssl genpkey -out fd.key \
-algorithm RSA \
-pkeyopt rsa_keygen_bits:2048 \
-aes-128-cbc
[base] Password:
+++++
Enter PEM pass phrase:
Verifyning - Enter PEM pass phrase:
(base) shouzabkhan@Shouzabs-Laptop openssl % openssl req -new -key fd.key -out fd.csr
Can't open fd.key for reading, Permission denied
8676402544:error:0200100e:system library:fopen:Permission denied:crypto/bio/bss_file.c:69:fopen('fd.key','r')
8676402544:error:0200100e:system library:BIO routines:BIO_new_file:cryptobio/bio/bss_file.c:78:
Unable to load Private Key
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo openssl req -new -key fd.key -out fd.csr
[base] Password:
Enter pass phrase for fd.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]:US
State or Province Name (full name) [Some-State]:Alabama
Locality Name (eg, city) []:Birmingham
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UAB
Organizational Unit Name (eg, section) []:Modern Cryptography
Common Name (e.g. server FQDN or YOUR name) []:Shouzab
Email Address []:khan0@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
(base) shouzabkhan@Shouzabs-Laptop openssl %
```



```
[base] shouzabkhan@Shouzabs-Laptop openssl % sudo cat fd.csr
[base] Password:
-----BEGIN CERTIFICATE REQUEST-----
MIIDCDCCAFACAAQAwgZMxCzAJBgNVBAYTA1VTMRAwDgYDVQQIDAjBbGFiYW1hMRMw
EQYDVQQHApCaXJtaW5naGFTMQwwCgYDVQQKDANVQUIxHDAaBgNVBAsME01vZGVy
biBDcnlwG9ncmFwaHkxEDAOBgNVBAMMB1Nob3V6YWIxHzAdBgkqhkiG9w0BCQEW
EHNraGFuNkBnbWFpbC5jb20wggeiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCntBhANA8ft6m1b4ADAfPJxC87iyzqp4xsi+AKd hvTwpnG+wehYFxccIUTfwKK
O2BrbwBwZTR4E0tr/XUFUUU8Tet24AAfc5gN5Z8aXEcxiJFWv3QJbFaosuJ/JfqNT
UKU$01PFQdLfCoI7qLTwu5jENdvOZyks18XLjDHEb/XAALMUP5cfaC0iyZ6Y9wh
FapZWUAU0qNjRhLGw5r4qRddrwNmXAN++vleArPzrjoy3oLhg8Flpe1E/07pqSSY
C+ZLCcpjUoX8zXUgX+wHZPbgZXNEpOSX8ocbP3DuqB/Yr4edXKts7CNYJBZz64vc
aPscB0bNwOBRAh2jBJalTPcRAgMBAAGgLzASBgkqhkiG9w0BCQIxBQwDVUFCKMBKG
CSqGSiB3DQEJBzEMDAoxMjM0NTY3ODkwMA0GCSqGSIb3DQEBCwUAA4IBAQAiKGJ+
94XAHIUNBxdadT56GccvGq6dw4UHVri5w+N4mQ2dTChJGFQRbRfGm8GoXKMUQ5bP
eHw/nd8GzwfrQpiGONQHlpDUR3eherkhDm2tsvRsgE03eV1aptxp1VzR5efeKkj
wbcCpGoL5S/wTpsBB82RY17kGaTeJNqJ271HwX1nmrxRCoyvgUpAAh1KvP4VzWzz
8YIQofolUufX33Tv7xr5OK2nK0z9M8i5TKp/PuZWzN+A+b7rk+eRYQIaNIENdtv
8xnvYzp1VBKP2SXWoq+ZNkSf8/1JS1mO3jxHND0JsE6ZK19ah36SHRFJrQKkjCKT
NustAIXNpgYMFN6o
-----END CERTIFICATE REQUEST-----
(base) shouzabkhan@Shouzabs-Laptop openssl %
```

Generating Self Signed Certificate:

Now if we want to self-sign the certificate, we can use the `openssl req` command with the `-x509` option.

The difference between these two was that, in CSR, they asked for the challenging password and optional company name but in self-signed certificates(CRT) they didn't ask for these two.

```
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo openssl req -x509 -new -key fd.key -out fd.crt
>Password:
Enter pass phrase for fd.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]:US
State or Province Name (full name) [Some-State]:AL
Locality Name (eg, city) []:BHM
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UAB
Organizational Unit Name (eg, section) []:CS
Common Name (e.g. server FQDN or YOUR name) []:Skhan
Email Address []:skhan6@gmail.com
(base) shouzabkhan@Shouzabs-Laptop openssl %
```

```
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo cat fd.crt
-----BEGIN CERTIFICATE-----
MIIDyTCCArGgAwIBAgIUEzDQVqfShoiCx3S8AoB0bVbr1E4wDQYJKoZIhvcNAQEL
BQAwdDELMakGA1UEBhMCVVMxCzAJBgNVBAgMAkFMMQwwCgYDVQQHDANCSE0xDDAK
BgNVBAoMA1VBQjELMAkGA1UECwwCQ1MxDjAMBgNVBAMMBVNraGFuMR8wHQYJKoZI
hvcNAQkBhBza2hhbjZAZ21haWwuY29tMB4XDTI0MDMxMDE3MTIzNFoXDTI0MDQw
OTE3MTIzNFowdDELMakGA1UEBhMCVVMxCzAJBgNVBAgMAkFMMQwwCgYDVQQHDANC
SE0xDDAKBgNVBAoMA1VBQjELMAkGA1UECwwCQ1MxDjAMBgNVBAMMBVNraGFuMR8w
HQYJKoZIhvcNAQkBhBza2hhbjZAZ21haWwuY29tMIIIBiJANBgkqhkiG9w0BAQE
AAOCAQ8AMIIBCgKCAQEAp7QYQDQPH7eptW+AAwHzycQv04ss6qeMbIvgCnYb08KZ
xvsHoWBcXHCFE38CpDtga8AVmU0eBNLa/11BVFFPE3rdAAH3OYDeWfG1xHMSCRV
r90CWxWqLLifyX6jU1C1EuTtTxUHS3wqC06i08LuYxDXbzmcpLJfFy4wxG/1wAC
zFD+XH2gtIsmemPcBxWqWV1AFNkjY0YZRsOa+KkXXa8DZlwDfvr5XgKz8646Mt6C
4RvBZaXtRP906akkAmSwnKY1KF/M11IF/sB2T24GVzRKTk1/KHGz9w7qgf2K+h
nVyrb0wjWCQWc+uL3Gj7HadGzcDgUQIdowSwi0z3EQIDAQAB01MwUTAdBgNVHQ4E
FgQU0JMN2F84TsSiwfNsRAY7yjcZc5QwHwYDVR0jBBgwFoAU0JMN2F84TsSiwfNs
RAy7yjcZc5QwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAP20/
ntK6BZ3OB1f8B2Pd1qbIytbiJG9VrxTH1bPhgVn6UPVy+cTq+h4/3zBYVkeUZgzR
PKMCYTjhG/100R7v5s00aPHuCV8AKY5UNH+A+Aa7FdqUJPX5ueNs81jb1hj/q1
NWbLdohRyfpBCvSPF/d34/tbaF9IQkKLz3YBnDw7e3AT3oD1TsZiOAY1te3uBAyG
DBVo0SqdVHDcTVjkpkIzhI7FaT7BWe7Sersjk0nyqkUcq0buuOUEdX1MNSBS9HV6
OVfeVzdnvhQ/tnSmL88+gmjJstPRSAT/oOTWYv/EyMRLIhdoL4FDoaW310K5JrL
hJoBwZFt79hyG06V3g==
-----END CERTIFICATE-----
(base) shouzabkhan@Shouzabs-Laptop openssl %
```

Create a private certification authority:

Root CA Configuration:

We include some basic information in a file to Root Certification Authority Configuration. The file name is set to be "root-ca.conf". The first part of the configuration file contains some basic CA information, such as the name and the base URL, and the components of the CA's distinguished name. Because the syntax is flexible, information needs to be provided only once.

```
[(base) shouzabkhan@Shouzabs-Laptop openssl % nano root-ca.conf
[(base) shouzabkhan@Shouzabs-Laptop openssl % sudo nano root-ca.conf
[Password:
[(base) shouzabkhan@Shouzabs-Laptop openssl % cat root-ca.conf
#CA Information
[ca_default]
name                = root-ca
domain_suffix       = www.cnn.com
[ca_dn]
countryName         = "US"
organizationName   = "UAB"
commonName          = "Shouzab"
[ca_default]
name                =
database           = $home/db/index
serial              = $home/db/serial
crlNumber           = $home/db/crlnumber
certificate        = $home/certs/cert.pem
private_key         = $home/private/$name.key
RANDFILE            = $home/private/random
nocs_crls_dir      = $home/certs
unique_subject     = no
copy_extensions    = none
default_days       = 3650
default_crl_days   = 30
default_md         = sha256
policy              = policy_c_o_match
[policy_c_o_match]
countryName         = match
stateOrProvinceName = optional
organizationName   = optional
organizationUnitName= optional
commonName          = supplied
[req]
default_bits       = 4096
encrypt_key        = yes
default_md         = sha256
```

Root CA Directory Structure:

Now we create a new Root CA directory Structure, this structure includes directories for certificates, a database, and private keys, as well as the initialization of some necessary files. By using the commands, we make the directories in the db directory just like shown in the below screenshot "Index" and "Serial".

```
[(base) shouzabkhan@Shouzabs-Laptop root-ca % ls
certs  db  private
[(base) shouzabkhan@Shouzabs-Laptop root-ca % touch db/serial
chmod u+w db/serial
touch: db/serial: Permission denied
chmod: db/serial: No such file or directory
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo touch db/serial
chmod u+w db/serial
Password:
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo openssl rand -hex 16 > db/serial
zsh: permission denied: db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % openssl rand -hex 16 > db/serial
zsh: permission denied: db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo touch db/serial
chmod u+w db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % ls
certs  db  private
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo openssl rand -hex 16 > db/serial
zsh: permission denied: db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo touch db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo chmod 600 db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo openssl rand -hex 16 > db/serial
zsh: permission denied: db/serial
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo sh -c 'openssl rand -hex 16 > db/serial'
[(base) shouzabkhan@Shouzabs-Laptop root-ca % echo 1001 > db/crlnumber
zsh: permission denied: db/crlnumber
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo echo 1001 > db/crlnumber
zsh: permission denied: db/crlnumber
[(base) shouzabkhan@Shouzabs-Laptop root-ca % sudo sh -c 'echo 1001 > db/crlnumber'
[(base) shouzabkhan@Shouzabs-Laptop root-ca % ls
certs  db  private
[(base) shouzabkhan@Shouzabs-Laptop root-ca % ls -l
total 0
drwxr-xr-x  2 root wheel 64 Mar 10 16:49 certs
drwxr-xr-x  5 root wheel 160 Mar 10 17:17 db
drwxr-xr-x  2 root wheel 64 Mar 10 16:49 private
[(base) shouzabkhan@Shouzabs-Laptop root-ca % certs/ db/ private/
zsh: permission denied: certs/
[(base) shouzabkhan@Shouzabs-Laptop root-ca % cd db
[(base) shouzabkhan@Shouzabs-Laptop db % ls
crlnumber  index  serial
[(base) shouzabkhan@Shouzabs-Laptop db % ]
```

Root CA Generation:

Now, we generate the root certificate authority private key and certificate.

```
(base) shouzabkhan@Shouzabs-Laptop openssl % sudo openssl req -new \
    -config root-ca.conf \
    -out root-ca.csr \
    -keyout private/root-ca.key
[Password:
 Generating a RSA private key
 .....+++++
 .....+++++
 writing new private key to 'private/root-ca.key'
 [Enter PEM pass phrase:
 [Verifying - Enter PEM pass phrase:
 -----
 (base) shouzabkhan@Shouzabs-Laptop openssl %
```

Test with OpenSSL:

By completing all the steps above from step 1 to step 3 in our step 4 is to test any domain with OpenSSL to see if we can get the server certificate or not so by executing the command

`"openssl s_client -connect www.cnn.com:443"` by this you can see in the screenshot below that we actually get the server certificate. It indicates that the whole process was successfully executed.

```
root-ca — openssl s_client -connect www.cnn.com:443 — 166x55
[base] shouzabkhan@Shouzabs-Laptop root-ca % openssl s_client -connect www.cnn.com:443
CONNECTED(0x0000005)
depth=2 OU = GlobalSign Root CA - R3, O = GlobalSign, CN = GlobalSign
verify return:1
depth=1 C = BE, O = GlobalSign nv-sa, CN = GlobalSign Atlas R3 DV TLS CA 2023 Q3
verify return:1
depth=0 CN = cnn.com
verify return:1
---
Certificate chain
  0 s:CN = cnn.com
    i:C = BE, O = GlobalSign nv-sa, CN = GlobalSign Atlas R3 DV TLS CA 2023 Q3
  1 s:C = BE, O = GlobalSign nv-sa, CN = GlobalSign Atlas R3 DV TLS CA 2023 Q3
    i:OU = GlobalSign Root CA - R3, O = GlobalSign, CN = GlobalSign
  ---
Server certificate
-----BEGIN CERTIFICATE-----
MIKqDCCCZCgAwIBAgIQAz3dCTUFVncaZ4TM/m6DFTANBgkqhkiG9w0BAQsFADBY
M0swCQDVQQEwJCRTEZMbcGA1UEChMRQ2xvYmfSsU2lnbiBudj1zYTEuCGWAEUE
AxMIR2xvYmfSsU2lnbiBBDgXhcbySMyBEV1BUFmQ8EqMjAyMyBRMzaEfW0yMzA5
MTIxOTM4MDVaFw0yNDEwMTMxOTM4MDRaMBIxEDA0BgNVBAMM2Nubz15jb20wggi
MA0GCSqS0ib3DQEBAQUAA4IBDwAwggEKAoIBAQDsZnl19Rp7hDYPjs4T0Ga39w5
BLHGsPhi41V4Hvt1me0/NMMmszIeNoY+aaDSM2dnegw296Ig1przSAQK8gDU6a
otUSmW08J+3xAbnn75DQ1BHjXF14EfjL4mIHmVV340+0w06owwFDUgxRzYnw1b
y6WEJfRyy70Mf6Elq0zXw2cMgfyuuq8ZEtgyddSr41/2/VxACBUDFYnqYbr9AR
qmJKvgz1sYUlaBj840Y3rnBnDCUkMW3qYT1mIDop+j4wLyMvHo9Qa0wY/bhI
ByhJtkdQy7xH2N02MoHQmaVo6x6w0l1cqszYiHND13SL31aj1MtU8aM13+edAgMB
AAgjgeyM1HrjCCBGGA1ldFQSCRF4wgRaggjbm4uY29tgg00lmfwSSjhm4u
Y29tggwqLmfwas5jbm4uaw+CHScuYXBpLmVsZwN0aw9udHjhY2t1c15jbm4uY29t
ghYqlfwaS5wbGF0Zm9ybS5jbm4uY29tghAqlMfryYWJpYy5jbm4uY29tghQqLmFy
d0VtaXMu0HvbnvylmNvbyIPK15jbm4u9ncy5jbm4uY29tghqqlmNsawWudC5hcbBs
ZXR2lmNubis5jb22CCSouY25ulmnbvYIK15jbm4uaw+CdyouY25uYXJhym1jlNmNv
byIOK15jbm5tb25leS5jb22CESouY25ulmnbvY09saXRp3MuY29tghYqLnvbmZpZy5v
dxR0dXjuZXiuY29tghEqLmRhdGEYX8pLmNubis5pb4IRK151zG1baW9ulmNubis5j
b22CFyoui2WRpdGlvbis5pLmNkbis5jbm4uY29tghwqLmVkaXRpb24uc3Rh2UubmV4
dc5jbm4uY29tgh0qLmVkaRpbb24uc3RhZ2Uylm51ehQuY25ulmNbVY1dkis1Z010
aw9ulnN0Yw1dMy5uZXh0LmNubis5jb22CEyoui2Wx1Y3Rpbd25zLmNubis5jb22CGsou
Zwx1Y3Rpbd25cmfja2VylmNubis5jb22CDcou228uY25ulmnbvYIPK15plmNkbis5j
bm4uY29tghYqLn1hcm1dHmubw9uZXkuY25ulm1vgg8qlm1vbmV5LmNubis5jb22C
Dioubmv4dC5jbm4uY29tghYqlm9ks5wbGF0Zm9yb5jbm4uY29tgg8qlm9idHr1
cm51c15jb22CEioucGxhd0Zvcm0uY25ulmNbVY1fk15zZWN0aw9ulWvnrlbnQu
bw9uZXkuY25ulmNbVY1UKi15zdGfnZS5uZXh0LmNubis5jb22CSouc3Rh2Uylm51
ehQuY25ulmNbVY1VKi5zdFnZTmubmV4dcC5jbm4uY29tghEqLnN0ZwxsX1iuY25u
lmNbVY1UKi50ZXjyYSs5uZXh0LmNubis5jb22CEcoudhJhdmvslmNubis5jb22CEyou
d3d3LmkjY2RulmNubis5jb22CD2Fws51dHauY25ulmNbVY1wY2xpZw50LmFwc0xl
dhYUy25ulmNbVYINY2suYXJhym1jlNmVbYIMY25ubw9uZXkuY29tgg9jbm5wb2xp
d61jcy5jb22CDWRjZmfuZG9tZS5jb22CHGdyX8c0cwuwdmVydgljYwzlmfwas5j
bm4uaW+CFGkuY2RulnRyYXZ1bc5jbm4uY29tgh1cmv2aW3Lmr1d15t251e55j
bm4uY29tghhwcmv2aW3lFnLm1vbmV5LmNubis5jb22CXbYzxPZxucmVm1v
bmV5LmNubis5jb22C038yZxZpZxcedJhaw4ubW9uZXkuY25ulmNbVY1acH1dm11
dzIucmVmLm1vbmV5LmNubis5jb22CD3VuZGyc2NvcmVklmNbVTA0BgNvHQBBAf8E
BAMCBaAwHQYDV01BVyFAY1kwYBQ0U AwEGCCsGAUFBwMCMB0GA1udbgQWBET9
Fy8eFhWRk9UjmQVNdvD81ZEHFTBX8gNVHSAEUD80MAgGBmeBDAECATBCBgorBgEE
AAyAygEDMDQwMgyIKwYBQ0UHagEWJmh0hdHz0i8vd3d3lmds2bJhbHnpZ24uY29t
L3JlcG9zaXRvcnkvAwGA1UdEwB/wQOMAwgZ4GCCsGAQUBBwEBB1GRMI0MEAG
```

```
Tools Extensions Help
root-ca -- zsh -- 166x55
XSRk1j10ZrVHcd0XLuAME/9+54Bm7TvRfI46hFCfu6FbQPIX3gg+5j+MZZSdIuQJ
d2XhMVAQY1pu27381/Ts2SuDx6v/cz8lV8D5o/xTtCpWAnLxM2bxSyVnYbk=
-----END CERTIFICATE-----
subject=CN = cnn.com

issuer=C = BE, O = GlobalSign nv-sa, CN = GlobalSign Atlas R3 DV TLS CA 2023 Q3

No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
SSL handshake has read 4452 bytes and written 377 bytes
Verification: OK
New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
Server public key is 2048 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
Post-Handshake New Session Ticket arrived:
SSL-Session:
Protocol : TLSv1.3
Cipher   : TLS_AES_128_GCM_SHA256
Session-ID: F32DA1C514F727E97C35AD19D2EEBEB119531B423B2748A0AC732F3FA639AAD2
Session-ID-ctx:
Resumption PSK: 03EAC947AA27F2940D0DACB5A242F2DFF29BB4D95151D02D1147B7753FE790AC
PSK identity: None
PSK identity hint: None
SRP username: None
TLS session ticket lifetime hint: 3600 (seconds)
TLS session ticket:
0000 - 62 4b c3 9f 8f 11 a7 ec-2e 4e 40 86 ab 8e 74 85 bK.....N@...t.
0010 - 90 84 07 d3 06 fd 5b c1-04 9d 8e 3b c7 73 b0 23 .....[....; .s.# 
0020 - 2f 76 16 f2 4e 41 c3 d3-da 95 85 d5 3a c6 20 bd /v..NA.....:.. .
0030 - 11 94 6b c5 c2 7f b7 83-8e 00 03 88 6a 57 4c df ..k.....jWL.
0040 - b4 8d 84 94 9c 2e db 44-5c 05 dd 54 93 ef 27 ce .....D\..T.'.
0050 - f1 1d 81 d6 46 b6 0e 92-18 09 77 2c 1d 29 bc f7 ....F.....w,..)..
0060 - 05 6d ac ba 67 75 0e 87-f3 de 09 5b 32 45 63 98 .m..gu.....[2Ec.
0070 - 17 af 79 35 95 d6 9a 52-14 b1 fd c1 c6 46 80 29 ..y5...R.....F.)
0080 - 4b 3c 88 69 76 1c a0 b9-85 5a 8d c4 1e ba 41 9f K<.iv.....Z....A.
0090 - 12 5b 4e 80 7b 4c e0 90-e2 e3 95 83 be 7a c0 b5 .[N.{L.....z.. 

Start Time: 1710119003
Timeout   : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
```

Getting the output as expected indicates that the SSL handshake process has completed successfully.

Part II. OQS-OpenSSL:

Step 1: Installation and build of OQS OpenSSL

1) On MacOS, firstly, we install brew package manager for simplified installation and managing software.

2) Then use, `brew install cmake ninja libtool openssl@1.1`

The above command is for brew package manager on macOS. It installs four software packages commonly used for building and managing software projects. This command equips the system with the necessary tools to build and manage software projects that might involve cryptography or rely on external libraries.

3) Command:

```
git clone --branch OQS-OpenSSL_1_1_1-stable  
https://github.com/open-quantum-safe/openssl.git <OPENSSL_DIR>
```

This command uses the git tool to clone a specific version of a software repository. In simpler terms, this command downloads the source code for the `OQS-OpenSSL_1_1_1-stable` branch of the `open-quantum-safe/openssl` repository and places it in the directory specified.

4) Command:

```
git clone --branch main  
https://github.com/open-quantum-safe/liboqs.git  
cd liboqs  
mkdir build && cd build  
cmake -GNinja -DCMAKE_INSTALL_PREFIX=<OPENSSL_DIR>/oqs ..  
ninja  
ninja install
```

- Download the source code for the liboqs library with PQC algorithms.
- Configure and build the library using cmake and ninja.
- Install the built library and potentially executables into a specific directory within OpenSSL installation.
- By completing these steps, the liboqs library with its PQC algorithms will be available on the system.

5) Command:

```
./Configure no-shared darwin64-arm64-cc  
make -j
```

This command sequence configures the build process to create a static library for an ARM64-based macOS system and then uses make to compile the source code into the library.

Step 2: Generate PQ public-private key pairs and certificates

1) Private Key: The below mentioned command generates a 2048-bit RSA private key and stores it in the file named fd.key.

Command for private key generation:

```
openssl genpkey -out fd.key \
    -algorithm RSA \
    -pkeyopt rsa_keygen_bits:2048 \
    -aes-128-cbc
```

```
Enter PEM pass phrase: (ex: hello)
Verifying - Enter PEM pass phrase: (ex: hello)
cat fd.key
```

```

+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
poojasrinivas@Poojas-Laptop oqs-openssl % cat fd.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFLTBXBgkqhkiG9w0BBQwwHAQIIlxPi39HpRkCAGgA
MwGCCqGSIB3DQIJBQAwHQYJYIZIAWUDBAECCBDtzFIYVvAbz9sm/JuJt0UBUIIE
0LlsGPSzExoQRlt61RWpu9Y1ThJCF3paZLVAb01VnOSjaWAs3Ww8us1x+eJQeJ/p
ijok+kz1hBZedw23STLSAnv0zAPbrNAnGms2DX/7ys00X7mrn2GH4HHJci7Sz6F1
C2CoRCH3JF/Cwn099j22g+P5MPmDi77EM1ERpSvYJi7uR1Y/hixkOnyU89ITZL
F0I/1+oiyY4Ht7OxU22gALDby+MrbkSIN7Hsp2Aj0hB1p+BLMj1PIbuX6gvF0kj
pDZkwRwoauIUs2rkgnfz7ueEHceh4ZspPY2oSZFR3yMwAyuaZ+LEECzYMAhv2FC
Gj8c1D/yvM+vPrgeMo/gD61rKBmdz93gmcpch5NIQXov3+IjiIXoSACKnErYYar
HE5sHnvdI1w2HU202uQpf49vNwf+QrV6tyyTRDrjAhg3CFTSn4enKsxUiceVUMJB
2+eDxVfyFT5S7NgH+y4uHgoHB1M3ycjEFPENYOnfQKKRBnQXv+jYFvUHOgEBNX
LC/si2PE1XqmtXqy1+4zsJ9/GcrgrnyCB41CTHZq66Pu/Fb0Scs7fhcmP20Mh4Y8Z
xQUvuywV5NCwQT500T6IUVlvBE4y9a0fTUvbA/sC4QpUq+8mfIf86e/giQle98Kj
xoui/CbsnvAttQVryVinsN28eQ52hFkfCoEetmfI1cqxDdTd1pIVYjbJ0k3rzAq8
wt9WZyCmEyYjRTriGkrQ30Wxkh0uP5wZqN1WW3SyvyJbS+SFn/xd4pKL0dwE5Mg
3772WQtN8W6YV95y7piQ5Mfq+ZKBFBgXSxET8vdP/zh1XPfhwjAkqMTq56aQNL
038XBu8BP0C/9A739/K/l0k4yP1EP4axRwv6v1yRz7IzuKTtu+Rqx/BgASzCyzk
igVmdrU9Kkc1G6x+TTGjCUOpq++tLA1s0oXipQsdwH9b2Xfc1HdM19AQsMin185/0
6NsmdVDACqisg04YXchMuhsAv/Yi7/0tloDZ5ycnGdOyeIAMRAS1lla1UfpBfAo5
xEt-Bt0ipXuXfohd60Jftx6g/0x+nErwEx6p/FA3vrXzkWjQIkLvgM2vrUyIewaf
sW0qjCrypklotlUeXQTh8ryRkSy3V/RyNwv2Q/TctmXifDkWaRjhTQyZLKAvu7k
sSXfOnV0m4fdHU/jKdoQqmCHTS1xUK04TzpUb8y5MwoiYQgg2t1xaJh3YvPe7Qj7
cV2vXGaprdox1Vt+QWJvUsBB/8u0A6qq4QVK7Z4Tehq1YDEdn9sSIYKkWdrvzqd
0SayI2QDvCuojfjZaopmNWkvwJjkX49LhOxNzUxn0e/G5tb4/x1YqwQmjVw1UKs/M
ujX2ANt8HrIXGrBe1bzEh1x/NskpHtghn4y/ksszVFcWR20iULUDGydBGFw4NAD
aj9YVWitnb7eMUJmJyAk28mLXANDamiX9VroG51ZxEC10eJMACUNhytZfldQr9n/
V2Pj13SS6ruflrRC5zFLvhwyubkh/Ipl0UzxWElMOYgbYpSLPwg4dPc0/90RSsu3
ECWSgk3JAvm6gGi1EEPbVka2zSnXR00HskpQGIXEMn1L3ihBRf/w0D6bEHuSIE0X
r2nvR0iaTr0k37/7mJqfrwx+jGa0q4ybWlrEy32j5t
-----END ENCRYPTED PRIVATE KEY-----
poojasrinivas@Poojas-Laptop oqs-openssl %

```

2) Public key generation Command:

```

openssl pkey -in fd.key -pubout -out fd-public.key
Enter pass phrase for fd.key:
cat fd-public.key

```

These commands extract the public key from a private key and display the public key's contents. When running this command, will be prompted to enter the passphrase associated with the private key in fd.key (if a passphrase was set during its creation). It provides the correct passphrase for successful extraction.

3) Certificate Generation Command:

```

openssl req -new -key fd.key -out fd.csr
openssl req -text -in fd.csr -noout

```

This command generates a Certificate Signing Request (CSR), a document used to request a digital certificate from a Certificate Authority (CA) and then display its contents for verification.

```
poojasrinivas@Poojas-Laptop oqs-openssl % openssl req -text -in fd.csr -noout
Certificate Request:
Data:
    Version: 1 (0x0)
    Subject: ST = Alabama, L = Birmingham, O = uab
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
                Modulus:
                    00:d2:e8:dc:8d:fa:1f:68:ff:a8:6f:fa:bd:2f:b0:
                    ea:8f:d0:f1:72:88:16:e4:04:2d:7f:d9:b3:f3:5f:
                    f5:d9:97:e7:c6:be:56:2f:d3:82:cd:8b:a8:e5:1b:
                    26:d9:cc:76:3e:45:8b:b9:8c:6c:d4:fe:84:f2:00:
                    83:74:54:f5:ec:77:16:92:62:a9:a7:e6:6c:95:55:
                    91:a2:97:fc:9f:8c:c7:be:48:92:30:34:d8:d7:af:
                    6b:e2:a6:b7:30:8f:62:37:93:5d:e4:9a:95:71:85:
                    7e:43:1d:a0:2b:18:f2:29:2b:e8:f4:59:26:77:00:
                    85:42:1c:29:73:67:b9:40:a2:77:16:af:a1:15:40:
                    d4:0c:db:ac:d8:63:bc:06:7b:48:a4:60:14:2b:2d:
                    3f:3e:02:74:08:d0:67:94:c0:aa:65:2f:da:2b:dd:
                    48:7c:f4:8e:6b:cd:f3:42:65:eb:ee:3b:09:3d:ea:
                    87:73:b7:3a:39:06:f6:6f:b8:8a:aa:83:d:a8:94:
                    d6:7d:e9:32:88:73:60:93:7f:9d:9c:f4:71:a1:cc:
                    37:d8:6f:f9:62:31:3f:92:6c:09:d4:9a:79:08:65:
                    73:45:ee:ea:67:44:1e:b3:7a:17:9c:0c:0a:cb:29:
                    09:75:c9:b0:6b:06:c0:20:64:53:c3:38:a2:4a:15:
                    a0:75
                Exponent: 65537 (0x10001)
Attributes:
    (none)
Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
04:c7:ec:24:27:ce:ca:92:ee:19:7f:3c:7d:9e:2f:99:4c:94:
ee:10:29:ef:f5:83:63:5f:78:db:a6:d2:fc:88:fb:83:0a:21:
19:b1:a1:14:41:23:a3:ec:06:88:ca:75:0d:06:0d:62:8f:36:
ce:e1:e4:89:e9:de:4f:67:3a:fe:f4:21:fd:bd:9c:24:56:f1:
bb:10:04:23:97:85:13:c7:60:36:31:67:9e:83:df:3d:dd:ba:
24:ad:98:54:66:2b:e1:86:8f:af:61:c6:f4:db:0e:be:76:fb:
1c:62:cf:6e:0a:94:76:ea:02:56:8f:cd:33:5e:e3:8a:98:45:
8b:22:18:dd:c0:26:5b:e7:ed:a9:ae:9e:48:80:e8:c4:e5:92:
59:aa:8e:7f:73:c5:22:6c:0a:5b:43:f2:3e:a8:1a:ac:dd:eb:
cb:66:82:3b:67:3a:e9:dc:07:d3:8e:5e:80:ad:dc:b3:11:44:
ff:28:fd:ad:03:8a:bb:95:c8:5d:6e:7c:28:39:a3:c6:30:d3:
52:fe:95:ed:04:f9:94:02:e8:82:61:44:ed:7f:34:27:d4:7c:
6a:b6:d2:5f:f8:00:c8:99:b0:03:78:2b:56:cb:ee:6e:ea:
f2:38:c0:d4:31:b8:95:a3:7e:b4:a3:59:53:b5:40:2e:c9:a1:
03:77:35:c0
```

```
poojasrinivas@Poojas-Laptop oqs-openssl %
```