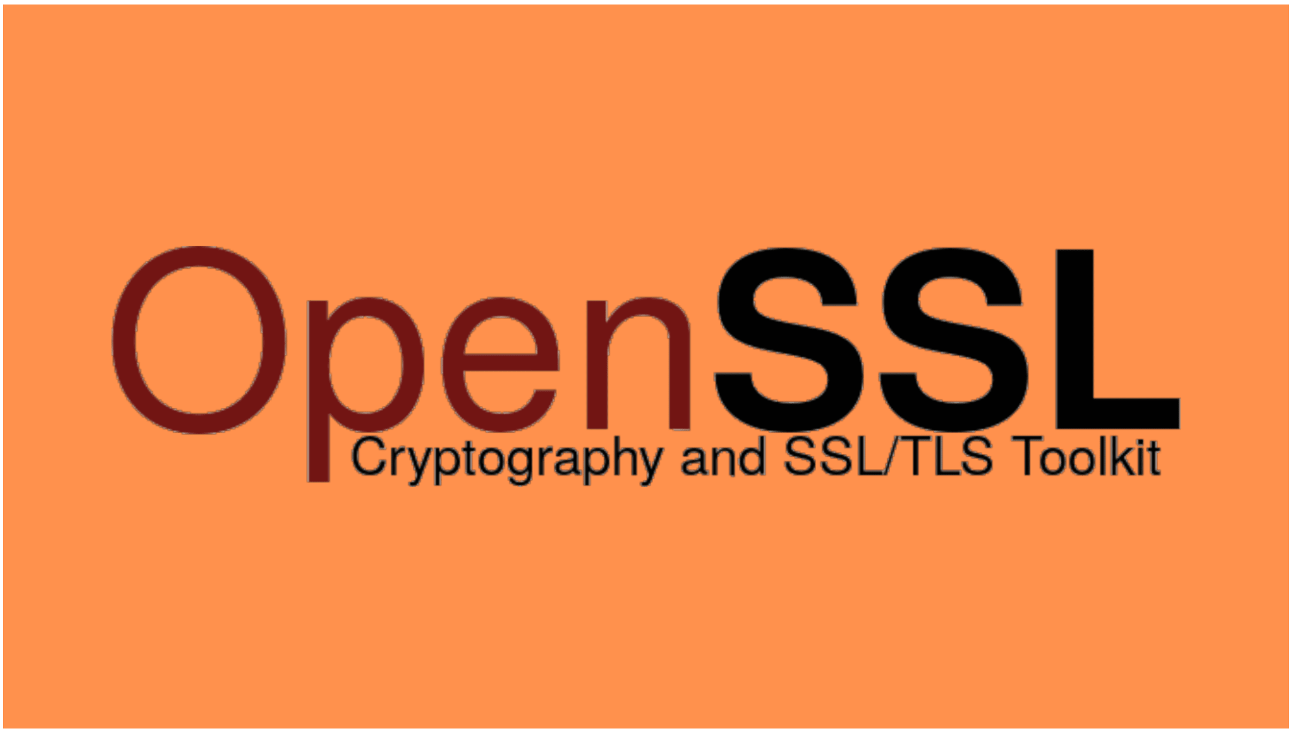


The Most Useful OpenSSL Commands to Work With SSL Certificates

Dal sito: **https://thesecmaster.com/the-most-useful-openssl-commands-to-work-with-ssl-certificates/**

OpenSSL is an open-source software library designed to secure digital communications and certificates. It is used by many websites and organizations worldwide to encrypt information in transit, such as emails, web traffic, and other data exchanged over the internet or computer networks. OpenSSL also provides a way for trusted entities to sign documents or digital certificates in order to verify the authenticity and integrity of the data. OpenSSL provides an invaluable security layer that helps to protect online transactions from malicious actors.
This article provides an overview of commonly used OpenSSL commands to work with SSL certificates. It covers various operations such as generating a new certificate, checking the details of an existing certificate, converting the certificate into different format, debugging when there is an error occurs, and pretty much everything that you need to know about OpenSSL.
The commands we have provided here can be more useful for creating, managing, and troubleshooting SSL certificates on various platforms. All this content made this article a valuable resource for system administrators and security professionals. The article also provides examples of how to use these commands in real-world scenarios with screenshots to give you a practical solution, helping readers to quickly and efficiently work with SSL certificates using OpenSSL.

Table of Contents

What is OpenSSL?

OpenSSL is an open-source software library that provides cryptographic protocols and security algorithms for implementing secure communications over computer networks. It can be used to protect data from eavesdropping, and encryption of email messages, payment transactions other sensitive information. OpenSSL supports a wide range of cryptographic functions, including digital signature, key exchange, and public-key encryption schemes. It is widely used for web-based applications and can be integrated into a variety of software programs.

The OpenSSL library is available on Linux, macOS, and Windows, making it a popular choice amongst developers who need secure communication protocols. By leveraging the functions of OpenSSL, developers can create highly secure applications that protect user data from malicious actors.

What is OpenSSL Used For?

OpenSSL is a powerful and versatile tool that can be used for a wide range of tasks. Some of the things that can be done using OpenSSL include the following:

1. **Creating and managing SSL certificates:** OpenSSL allows users to easily create and manage SSL certificates, which can be used to prove the identity of the entity.
2. **Creating and verifying digital signatures:** OpenSSL's libraries can be used to create digital signatures, which can be used to authenticate the identity of the sender and the integrity of the message.
3. **Generating private keys and certificate signing requests:** OpenSSL can be used to generate private keys and CSRs, which are used in the process of obtaining an SSL certificate from a certificate authority.
4. **Creating and managing Certificate Authorities and Certificate Revocation Lists:** OpenSSL can be used to create and manage CAs and CRLs, which are used to issue and revoke SSL certificates that have been compromised or are no longer needed.
5. **Converting certificate formats:** OpenSSL can be used to convert certificates between different formats, such as CRT, CER, PEM, DER, CRT, PKCS7, and PKCS#12.
6. **Inspecting SSL Certificates:** OpenSSL can be used to check the details of existing certificates, such as the validity period, the subject and issuer, and other details.
7. **SSL/TLS Testing:** OpenSSL can be used to test the SSL/TLS configurations of servers and clients.

Please read these posts to learn more about OpenSSL and Digital Certificate:

- How To Generate A CSR For A Multi-Domain SSL Certificate Using OpenSSL?
- Step-By-Step Procedure To Install OpenSSL On The Windows Platform
- How To Set Up A Certificate Authority On Ubuntu Using OpenSSL?
- What Is A PKI Certificate? What Are The Different Types Of PKI Certificates?
- What Are The Different Types Of Certificate Authority
- What Is The Difference Between A Standalone And An Enterprise CA

How Do You Check That OpenSSL is Installed on Your Machine?

This verification procedure depends on the type of operating system platform. The different operating systems will have different procedures.

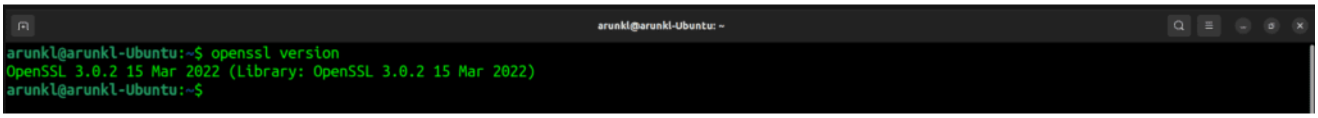
1. **Windows**: On Windows, you can check if OpenSSL is installed by going to the Add/Remove Programs underneath Control Panel and searching for “OpenSSL”. If it’s installed, it will appear in the list of installed programs.
2. **macOS**: you can check by searching for OpenSSL in the Applications folder.
3. **Linux**: Open a terminal window and type “openssl version” (without quotes). If OpenSSL is installed, it will display the version number of your installed version.

Generally, you can use the Linux method on Windows and mac. It works on all the operating systems. Check the OpenSSL documentation: If you are not sure where OpenSSL is installed, you can check the OpenSSL documentation. This usually contains information on how to install OpenSSL on various platforms.

Commands to Check the Version of OpenSSL

Run this command on the terminal to check the version of OpenSSL.

```
$ openssl version
```



```
arunkl@arunkl-Ubuntu:~$ openssl version
OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
arunkl@arunkl-Ubuntu:~$
```

Important Abbreviations Related to the Digital Certificates:
Here are some important abbreviations related to certificates:

1. **SSL**: Secure Sockets Layer, a protocol used to establish secure connections over the internet.
2. **TLS**: Transport Layer Security, a successor to SSL, which is used to establish secure connections over the internet.
3. **CA**: Certificate Authority, an organization that issues digital certificates used to establish trust in SSL/TLS connections.
4. **CSR**: Certificate Signing Request, a file that contains information about the certificate holder and the public key. It is used to apply for a certificate from a CA.
5. **CRT**: Certificate, a digital certificate issued by a CA that contains information about the certificate holder and the public key.
6. **PEM**: Privacy-enhanced Electronic Mail, a Base64 encoded file format that is used to store SSL/TLS certificates, private keys, and other cryptographic objects.
7. **DER**: Distinguished Encoding Rules, a binary format used to store SSL/TLS certificates, private keys, and other cryptographic objects.
8. **PKCS**: Public-Key Cryptography Standards, a set of standards for public-key cryptography, which includes formats for storing certificates and keys.
9. **X.509**: A standard that defines the format of digital certificates used in SSL/TLS connections.
10. **CRL**: Certificate Revocation List, a list of revoked certificates that is used to revoke SSL/TLS certificates that have been compromised or are no longer needed.
11. **SAN**: Subject Alternative Name, a field in an SSL/TLS certificate that allows multiple hostnames or IP addresses to be associated with a single certificate.
12. **OV**: Organization Validation, a type of SSL/TLS certificate that requires additional validation of the organization’s identity before the certificate is issued.
13. **EV**: Extended Validation, a type of SSL/TLS certificate that requires the highest level of validation of the organization’s identity before the certificate is issued.
14. **OCSP**: Online Certificate Status Protocol, a protocol used to check the revocation status of a certificate in real time.
15. **CAA**: Certificate Authority Authorization, a DNS record that specifies which CAs are authorized to issue certificates for a domain.
16. **ECC**: Elliptic Curve Cryptography, a type of public-key cryptography that uses elliptic curve mathematics to provide the same level of security as traditional methods with smaller key sizes.
17. **RSA**: Rivest–Shamir–Adleman, a widely used public-key encryption algorithm that is based on the mathematical properties of large prime numbers.
18. **DH**: Diffie-Hellman, a key-agreement protocol that allows two parties to establish a shared secret over an insecure communication channel.
19. **AES**: Advanced Encryption Standard, a widely used symmetric encryption algorithm that can be used to encrypt and decrypt data.

20. **SHA**: Secure Hash Algorithm, a widely used cryptographic hash function that can be used to create a unique digital fingerprint of a message or data.
21. **HSM**: Hardware Security Module, a physical device that can store and manage cryptographic keys, and perform cryptographic operations.
22. **PKI**: Public Key Infrastructure, a set of policies, procedures, and technologies used to manage digital certificates and public-key encryption.
23. **DSA**: Digital Signature Algorithm, a standard for digital signatures, based on the mathematical properties of modular arithmetic and the discrete logarithm problem.
24. **DH**: Diffie-Hellman, a key-agreement protocol that allows two parties to establish a shared secret over an insecure communication channel.
25. **ECDSA**: Elliptic Curve Digital Signature Algorithm, a digital signature algorithm based on elliptic curve cryptography, that can be used to create digital signatures.

How to Generate A Self-Signed Certificate Using OpenSSL?

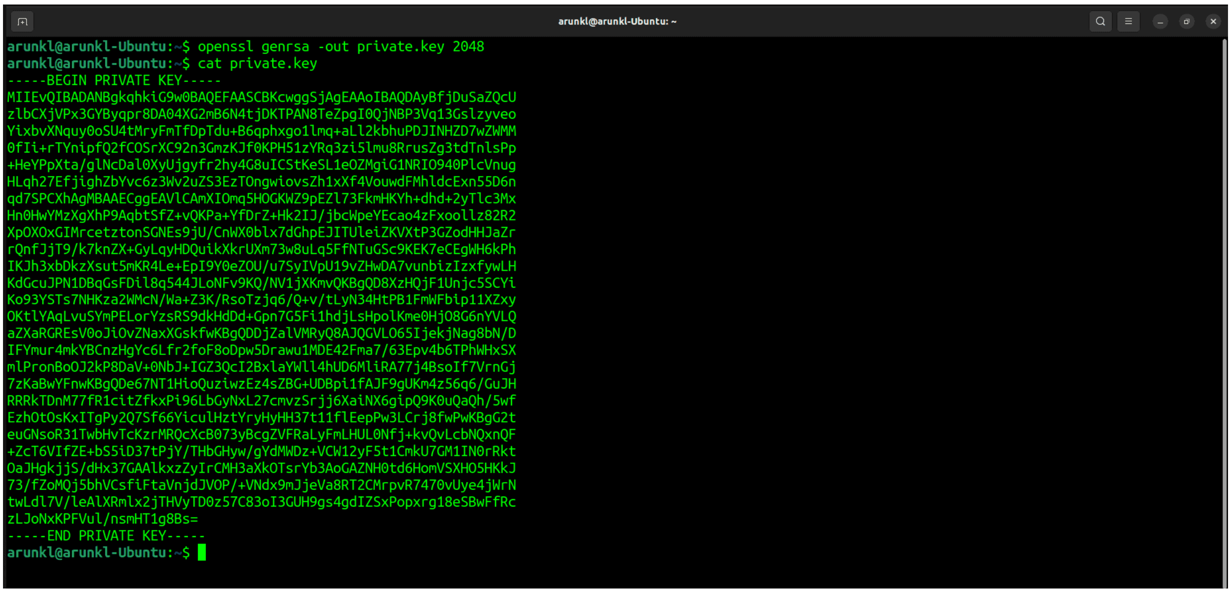
Generating a self-signed certificate using OpenSSL is a relatively simple process. The first step is to generate the key pair, which has a private key as well as a public key. This will be used to sign the certificate in Step 4. The second step is to extract the public key from the key pair. The third step is to generate a Certificate Signing Request (CSR). This will be used by the certificate authority (CA) to create the self-signed certificate. You will be prompted to enter a variety of information, such as the common name, organization name, organization unit, country code, email address, and many more. Finally, generate the self-signed certificate using the private key and CSR. Simple, Isn't it?
Time needed: 15 minutes.

How to Generate A Self-Signed Certificate Using OpenSSL?

1. **Generate Key pair using OpenSSL**
   To create the key pair, run this command in your terminal:

   $ openssl genrsa -out private.key 2048

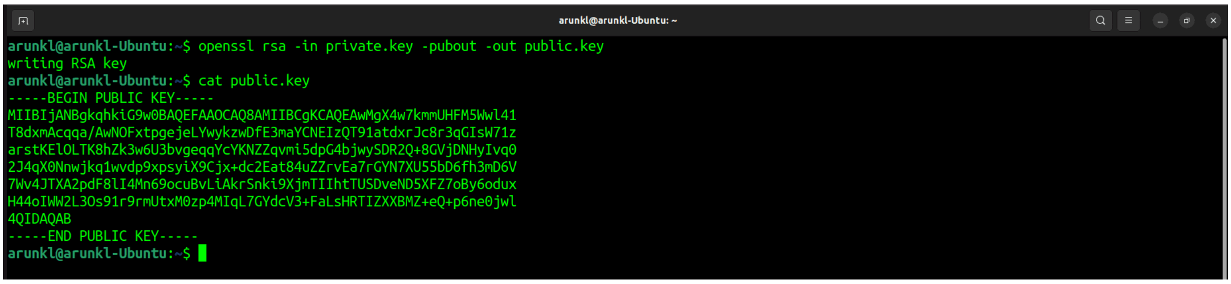
   This command will generate an RSA key pair with a length of 2048.



```
arunkl@arunkl-Ubuntu:~$ openssl genrsa -out private.key 2048
arunkl@arunkl-Ubuntu:~$ cat private.key
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDAyBfjDuSaZQcU
zlbCXjVPx3GYByqpr8DA04XG2mB6N4tjDKTPAN8TeZpgI0QjNBP3Vq13Gslzyveo
YixbvXNquy0oSU4tMryFmTfDpTdu+B6qphxgo1lmq+aLl2kbhuPDJINHZD7wZWMM
0fIi+rTYnipfQ2fCOSrXC92n3GmzKJf0KPH51zYRq3zi5lmu8RrusZg3tdTnlsPp
+HeYPpXta/glNcDal0XyUjgyfr2hy4G8uICStKeSL1eOZMgiG1NRIO940PlcVnug
HLqh27EfjighZbYvc6z3Wv2uZS3EzTOngwiovsZh1xXf4VouwdFMhldcExn55D6n
qd7SPCXhAgMBAAECggEAVlCAmXIOmq5HOGKWZ9pEZl73FkmHKYh+dhd+2yTlc3Mx
Hn0HwYMzXgXhP9AqbtSfZ+vQKPa+YfDrZ+Hk2IJ/jbcWpeYEcao4zFxoollz82R2
XpOXOxGIMrcetztonSGNEs9jU/CnWX0blx7dGhpEJITUleiZKVXtP3GZodHHJaZr
rQnfJjT9/k7knZX+GyLqyHDQuikXkrUXm73w8uLq5FfNTuGSc9KEK7eCEgWH6kPh
IKJh3xbDkzXsut5mKR4Le+EpI9Y0eZOU/u7SyIVpU19vZHwDA7vunbizIzxfywLH
KdGcuJPN1DBqGsFDil8q544JLoNFv9KQ/NV1jXKmvQKBgQD8XzHQjF1Unjc5SCYi
Ko93YSTs7NHKza2WMcN/Wa+Z3K/RsoTzjq6/Q+v/tLyN34HtPB1FmWFbip11XZxy
OKtlYAqLvuSYmPELorYzsRS9dkHdDd+Gpn7G5Fi1hdjLsHpolKme0HjO8G6nYVLQ
aZXaRGREsV0oJiOvZNaxXGskfwKBgQDDjZalVMRyQ8AJQGVLO65IjekjNag8bN/D
IFYmur4mkYBCnzHgYc6Lfr2foF8oDpw5Drawu1MDE42Fma7/63Epv4b6TPhWHxSX
mlPronBoOJ2kP8DaV+0NbJ+IGZ3QcI2BxlaYWll4hUD6MliRA77j4BsoIf7VrnGj
7zKaBwYFnwKBgQDe67NT1HioQuziwzEz4sZBG+UDBpi1fAJF9gUKm4z56q6/GuJH
RRRkTDnM77fR1citZfkxPi96LbGyNxL27cmvzSrjj6XaiNX6gipQ9K0uQaQh/5wf
EzhOtOsKxITgPy2Q7Sf66YiculHztYryHyHH37t11flEepPw3LCrj8fwPwKBgG2t
euGNsoR31TwbHvTcKzrMRQcXcB073yBcgZVFRaLyFmLHUL0Nfj+kvQvLcbNQxnQF
+ZcT6VIfZE+bS5iD37tPjY/THbGHyw/gYdMWDz+VCW12yF5t1CmkU7GM1IN0rRkt
OaJHgkjjS/dHx37GAAlkxzZyIrCMH3aXkOTsrYb3AoGAZNH0td6HomVSXHO5HKkJ
73/fZoMQj5bhVCsfiFtaVnjdJVOP/+VNdx9mJjeVa8RT2CMrpvR7470vUye4jWrN
twLdl7V/leAlXRmlx2jTHVyTD0z57C83oI3GUH9gs4gdIZSxPopxrg18eSBwFfRc
zLJoNxKPFVul/nsmHT1g8Bs=
-----END PRIVATE KEY-----
arunkl@arunkl-Ubuntu:~$
```

2. **Extract the public key from the key pair**
   Run this command to extract the public key from the key pair generated in step 1.

   $ openssl rsa -in private.key -pubout -out public.key



```
arunkl@arunkl-Ubuntu:~$ openssl rsa -in private.key -pubout -out public.key
writing RSA key
arunkl@arunkl-Ubuntu:~$ cat public.key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwMgX4w7kmmUHFM5Wwl41
T8dxmAcqqa/AwNOFxtpgejeLYwykzwDfE3maYCNEIzQT91atdxrJc8r3qGIsW71z
arstKElOLTK8hZk3w6U3bvgeqqYcYKNZZqvmi5dpG4bjwySDR2Q+8GVjDNHyIvq0
2J4qX0Nnwjkq1wvdp9xpsyiX9Cjx+dc2Eat84uZZrvEa7rGYN7XU55bD6fh3mD6V
7Wv4JTXA2pdF8lI4Mn69ocuBvLiAkrSnki9XjmTIIhtTUSDveND5XFZ7oBy6odux
H44oIWW2L3Os91r9rmUtxM0zp4MIqL7GYdcV3+FaLsHRTIZXXBMZ+eQ+p6ne0jwl
4QIDAQAB
-----END PUBLIC KEY-----
arunkl@arunkl-Ubuntu:~$ 
```

3. **Generate a Certificate Signing Request (CSR)**

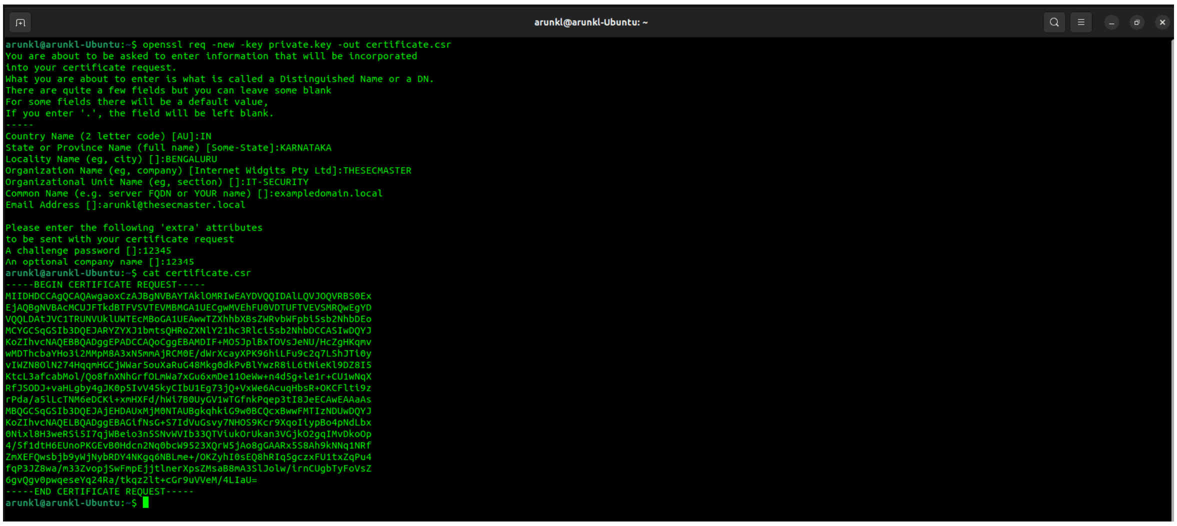
   The next step is to generate a Certificate Signing Request (CSR). This will be used by the certificate authority (CA) to create the self-signed certificate. To generate the CSR, run this command in your terminal:

   You will be prompted to enter a variety of information, such as the common name, organization name, organization unit, country code, email address, optional password, and many more. Enter the valid input it asks to generate the CSR. For example, the country name should be two char country codes. The common name should be the FQDN or IP to which you are going to get the certificate. The CSR is now ready for the CA to generate a self-signed certificate.

   $ openssl req -new -key private.key -out certificate.csr

   We suggest verifying the CSR generated before you submit it to the Certificate Authority. Run this command to verify the details of your CSR.

   $ openssl req -text -in certificate.csr -noout -verify



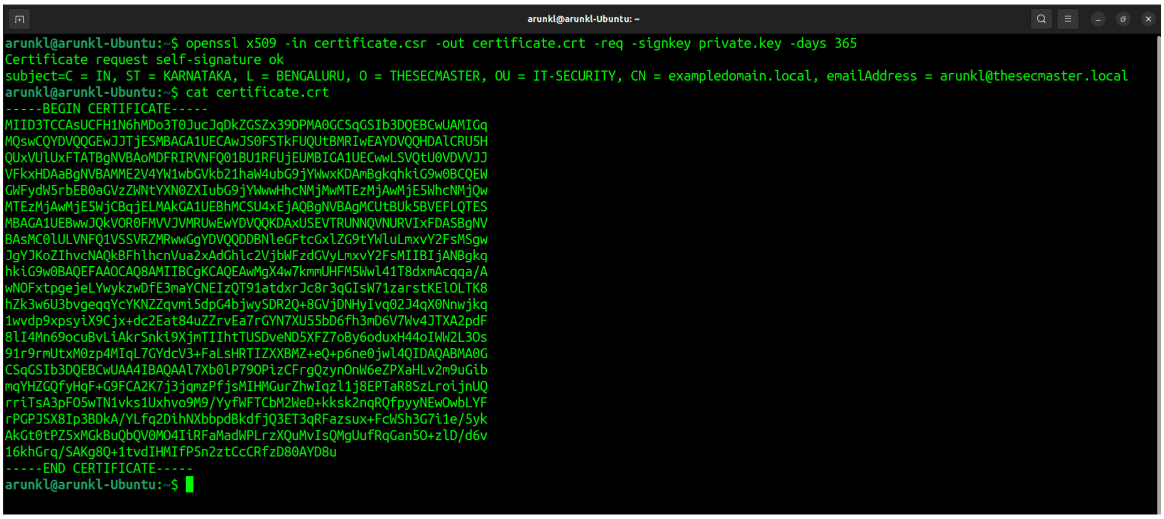
```
arunkl@arunkl-Ubuntu:~$ openssl req -new -key private.key -out certificate.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can 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]:IN
State or Province Name (full name) [Some-State]:KARNATAKA
Locality Name (eg, city) []:BENGALURU
Organization Name (eg, company) [Internet Widgits Pty Ltd]:THESECMASTER
Organizational Unit Name (eg, section) []:IT-SECURITY
Common Name (e.g. server FQDN or YOUR name) []:exampledomain.local
Email Address []:arunkl@thesecmaster.local

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:12345
An optional company name []:12345
arunkl@arunkl-Ubuntu:~$ cat certificate.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIDHDCCAgQCAQAwgaoxCzAJBgNVBAYTAklOMRIwEAYDVQQIDAlLQVJOQVRBS0Ex
EjAQBgNVBAcMCUJFTkdBTFVSVTEVMBMGA1UECgwMVEhFU0VDTUFTVEVSMRQwEgYD
VQQLDAtJVC1TRUNVUklUWTEcMBoGA1UEAwwTZXhhbXBsZWRvbWFpbi5sb2NhbDEo
MCYGCSqGSIb3DQEJARYZYXJ1bmtsQHRoZXNlY21hc3Rlci5sb2NhbDCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAMDIF+MO5JplBxTOVsJeNU/HcZgHKqmv
wMDThcbaYHo3i2MMpM8A3xN5mmAjRCM0E/dWrXcayXPK96hiLFu9c2q7LShJTi0y
vIWZN8OlN274HqqmHGCjWWar5ouXaRuG48Mkg0dkPvBlYwzR8iL6tNieKl9DZ8I5
KtcL3afcabMol/Qo8fnXNhGrfOLmWa7xGu6xmDe11OeWw+n4d5g+le1r+CU1wNqX
RfJSODJ+vaHLgby4gJK0p5IvV45kyCIbU1Eg73jQ+VxWe6AcuqHbsR+OKCFlti9z
rPda/a5lLcTNM6eDCKi+xmHXFd/hWi7B0UyGV1wTGfnkPqep3tI8JeECAwEAAaAs
MBQGCSqGSIb3DQEJAjEHDAUxMjM0NTAUBgkqhkiG9w0BCQcxBwwFMTIzNDUwDQYJ
KoZIhvcNAQELBQADggEBAGifNsG+S7IdVuGsvy7NHOS9Kcr9XqoIiypBo4pNdLbx
0Nixl8H3weRSi5I7qjWBeio3n5SNvWVIb33QTViukOrUkan3VGjkO2gqIMvDkoOp
4/5f1dtH6EUnoPKGEvB0Hdcn2Nq0bcW9523XQrW5jAo8gGAARx5S8Ah9kNNq1NRf
ZmXEFQwsbjb9yWjNybRDY4NKgq6NBLme+/OKZyhI0sEQ8hRIq5gczxFU1txZqPu4
fqP3JZ8wa/m33ZvopjSwFmpEjjtlnerXpsZMsaB8mA3SlJolw/irnCUgbTyFoVsZ
6gvQgv0pwqeseYq24Ra/tkqz2lt+cGr9uVVeM/4LIaU=
-----END CERTIFICATE REQUEST-----
arunkl@arunkl-Ubuntu:~$ 
```

4. **Generate the self-signed certificate**

   Finally, generate the self-signed certificate using the private key and CSR. Run this command to generate the self-signed certificate on the terminal:

   $ openssl x509 -in certificate.csr -out certificate.crt -req -signkey private.key -days 365



```
arunkl@arunkl-Ubuntu:~$ openssl x509 -in certificate.csr -out certificate.crt -req -signkey private.key -days 365
Certificate request self-signature ok
subject=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = exampledomain.local, emailAddress = arunkl@thesecmaster.local
arunkl@arunkl-Ubuntu:~$ cat certificate.crt
-----BEGIN CERTIFICATE-----
MIID3TCCAsUCFH1N6hMDo3T0JucJqDkZGSZx39DPMA0GCSqGSIb3DQEBCwUAMIGq
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxHDAaBgNVBAMME2V4YW1wbGVkb21haW4ubG9jYWwxKDAmBgkqhkiG9w0BCQEW
GWFydW5rbEB0aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTEzMjAwMjE5WhcNMjQw
MTEzMjAwMjE5WjCBqjELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTES
MBAGA1UEBwwJQkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNV
BAsMC0lULVNFQ1VSSVRZMRwwGgYDVQQDDBNleGFtcGxlZG9tYWluLmxvY2FsMSgw
JgYJKoZIhvcNAQkBFhlhcnVua2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwMgX4w7kmmUHFM5Wwl41T8dxmAcqqa/A
wNOFxtpgejeLYwykzwDfE3maYCNEIzQT91atdxrJc8r3qGIsW71zarstKElOLTK8
hZk3w6U3bvgeqqYcYKNZZqvmi5dpG4bjwySDR2Q+8GVjDNHyIvq02J4qX0Nnwjkq
1wvdp9xpsyiX9Cjx+dc2Eat84uZZrvEa7rGYN7XU55bD6fh3mD6V7Wv4JTXA2pdF
8lI4Mn69ocuBvLiAkrSnki9XjmTIIhtTUSDveND5XFZ7oBy6oduxH44oIWW2L3Os
91r9rmUtxM0zp4MIqL7GYdcV3+FaLsHRTIZXXBMZ+eQ+p6ne0jwl4QIDAQABMA0G
CSqGSIb3DQEBCwUAA4IBAQAAl7Xb0lP79OPizCFrgQzynOnW6eZPXaHLv2m9uGib
mqYHZGQfyHqF+G9FCA2K7j3jqmzPfjsMIHMGurZhwIqzl1j8EPTaR8SzLroijnUQ
rriTsA3pFO5wTN1vks1Uxhvo9M9/YyfWFTCbM2WeD+kksk2nqRQfpyyNEwOwbLYF
rPGPJSX8Ip3BDkA/YLfq2DihNXbbpdBkdfjQ3ET3qRFazsux+FcWSh3G7i1e/5yk
AkGt0tPZ5xMGkBuQbQV0MO4IiRFaMadWPLrzXQuMvIsQMgUufRqGan5O+zlD/d6v
16khGrq/SAKg8Q+1tvdIHMIfP5n2ztCcCRfzD80AYD8u
-----END CERTIFICATE-----
arunkl@arunkl-Ubuntu:~$
```

Most Useful OpenSSL Commands to Work With SSL Certificates

Before we head toward the most useful OpenSSL commands to work with SSL certificates, let's see the structure of OpenSSL commands.

Structure of a typical OpenSSL command:

The structure of a simple OpenSSL command typically looks like this:

```
openssl <command> <options>
```

The main command is 'openssl' which is followed by a subcommand <command> and <options>. It is allowed to use multiple subcommands and options in a single line so that multiple things can be performed in a single command.
The subcommand can be any of the cryptographic functions supported by the OpenSSL library. Some examples of subcommands include genpkey, req, x509, pkcs12, rsa, verify, etc. These subcommands are used to perform specific cryptographic operations.
Options <options> are any additional arguments required to work the command. Typically options are used to give input or get the output of the comment. Examples: -req, -days, -in, -out, etc.
If you struggle to identify the 'subcommands' over 'options' in lengthy OpenSSL commands, here is the tip. The key identifier that differentiates 'subcommands' from 'options' is the '-' symbol.

Example for a simple OpenSSL command:

```
$ openssl genrsa -out private.key 2048
```

In the above command, 'genrsa' is the subcommand, '-out' is an option, 'private.key' specifies the output file name, and '2048' is the key pair length.

Example of an OpenSSL with multiple subcommands and options:

```
$ openssl x509 -in certificate.csr -out certificate.crt -req -signkey private.key -days 365
```

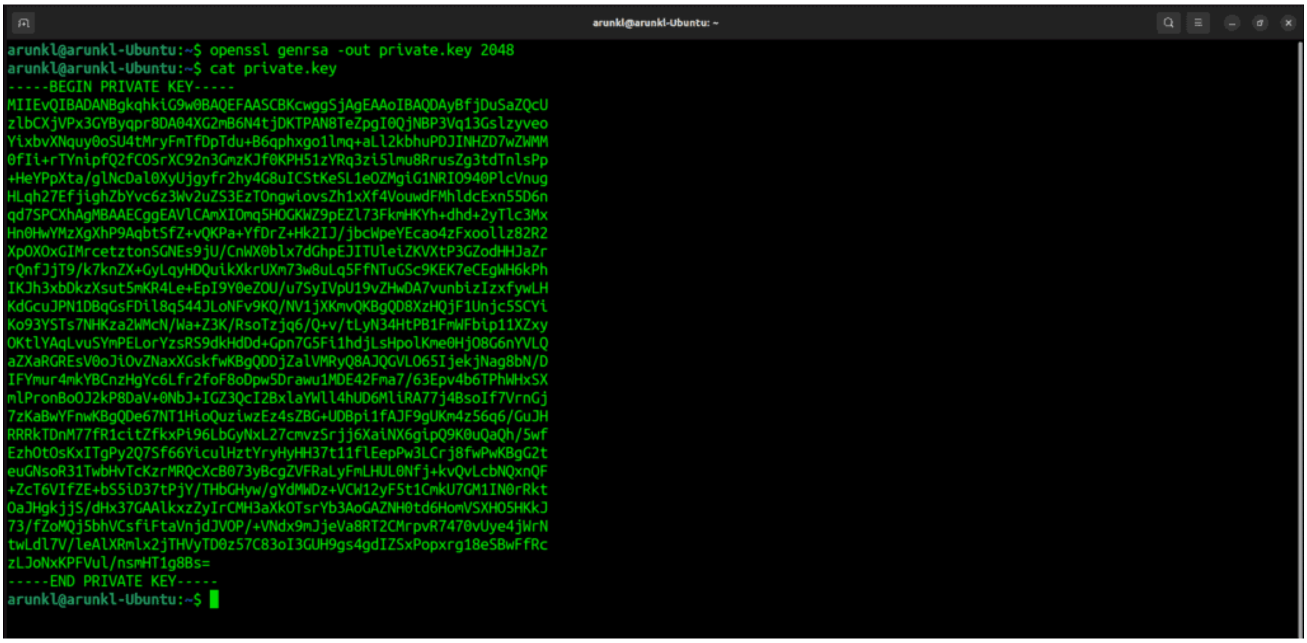
OpenSSL Commands to Generate SSL Certificates and Keys

In this section, we are going to see the Most Useful OpenSSL Commands to Work With SSL Certificates.

#1. OpenSSL Command to Generate a Private Key or Key Pair

```
$ openssl genrsa -out private.key 2048
```

This command generates a new RSA key pair and saves it in a file named “private.key” in the current directory.

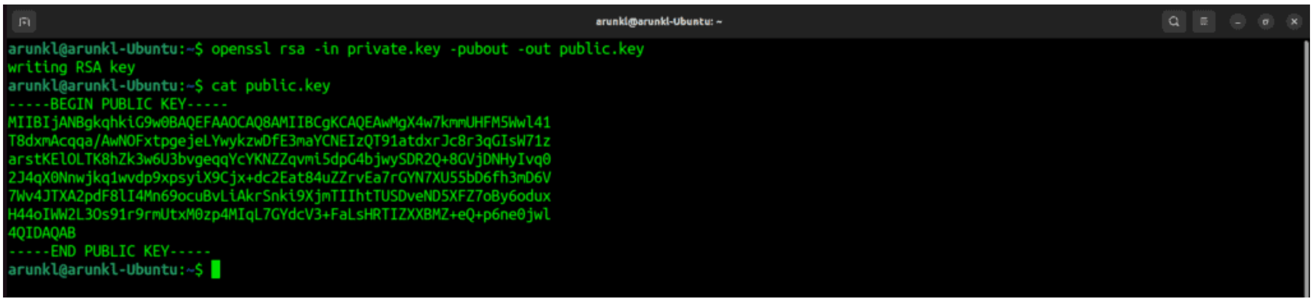


```
arunkl@arunkl-Ubuntu:~$ openssl genrsa -out private.key 2048
arunkl@arunkl-Ubuntu:~$ cat private.key
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDAyBfjDuSaZQcU
zlbCXjVPx3GYByqpr8DA04XG2mB6N4tjDKTPAN8TeZpgI0QjNBP3Vq13Gslzyveo
YixbvXNquy0oSU4tMryFmTfDpTdu+B6qphxgo1lmq+aLl2kbhuPDJINHZD7wZWMM
0fIi+rTYnipfQ2fCOSrXC92n3GmzKJf0KPH51zYRq3zi5lmu8RrusZg3tdTnlsPp
+HeYPpXta/glNcDal0XyUjgyfr2hy4G8uICStKeSL1eOZMgiG1NRIO940PlcVnug
HLqh27EfjighZbYvc6z3Wv2uZS3EzTOngwiovsZh1xXf4VouwdFMhldcExn55D6n
qd7SPCXhAgMBAAECggEAVlCAmXIOmq5HOGKWZ9pEZl73FkmHKYh+dhd+2yTlc3Mx
Hn0HwYMzXgXhP9AqbtSfZ+vQKPa+YfDrZ+Hk2IJ/jbcWpeYEcao4zFxoollz82R2
XpOXOxGIMrcetztonSGNEs9jU/CnWX0blx7dGhpEJITUleiZKVXtP3GZodHHJaZr
rQnfJjT9/k7knZX+GyLqyHDQuikXkrUXm73w8uLq5FfNTuGSc9KEK7eCEgWH6kPh
IKJh3xbDkzXsut5mKR4Le+EpI9Y0eZOU/u7SyIVpU19vZHwDA7vunbizIzxfywLH
KdGcuJPN1DBqGsFDil8q544JLoNFv9KQ/NV1jXKmvQKBgQD8XzHQjF1Unjc5SCYi
Ko93YSTs7NHKza2WMcN/Wa+Z3K/RsoTzjq6/Q+v/tLyN34HtPB1FmWFbip11XZxy
OKtlYAqLvuSYmPELorYzsRS9dkHdDd+Gpn7G5Fi1hdjLsHpolKme0HjO8G6nYVLQ
aZXaRGREsV0oJiOvZNaxXGskfwKBgQDDjZalVMRyQ8AJQGVLO65IjekjNag8bN/D
IFYmur4mkYBCnzHgYc6Lfr2foF8oDpw5Drawu1MDE42Fma7/63Epv4b6TPhWHxSX
mlPronBoOJ2kP8DaV+0NbJ+IGZ3QcI2BxlaYWll4hUD6MliRA77j4BsoIf7VrnGj
7zKaBwYFnwKBgQDe67NT1HioQuziwzEz4sZBG+UDBpi1fAJF9gUKm4z56q6/GuJH
RRRkTDnM77fR1citZfkxPi96LbGyNxL27cmvzSrjj6XaiNX6gipQ9K0uQaQh/5wf
EzhOtOsKxITgPy2Q7Sf66YiculHztYryHyHH37t11flEepPw3LCrj8fwPwKBgG2t
euGNsoR31TwbHvTcKzrMRQcXcB073yBcgZVFRaLyFmLHUL0Nfj+kvQvLcbNQxnQF
+ZcT6VIfZE+bS5iD37tPjY/THbGHyw/gYdMWDz+VCW12yF5t1CmkU7GM1IN0rRkt
OaJHgkjjS/dHx37GAAlkxzZyIrCMH3aXkOTsrYb3AoGAZNH0td6HomVSXHO5HKkJ
73/fZoMQj5bhVCsfiFtaVnjdJVOP/+VNdx9mJjeVa8RT2CMrpvR7470vUye4jWrN
twLdl7V/leAlXRmlx2jTHVyTD0z57C83oI3GUH9gs4gdIZSxPopxrg18eSBwFfRc
zLJoNxKPFVul/nsmHT1g8Bs=
-----END PRIVATE KEY-----
arunkl@arunkl-Ubuntu:~$ 
```

#2. OpenSSL Command to Extract the Pubic Key from the Key pair

```
$ openssl rsa -in private.key -pubout -out public.key
```

This command extracts the public key from the ‘private.key’ key pair and saves it in ‘public.key’ file.

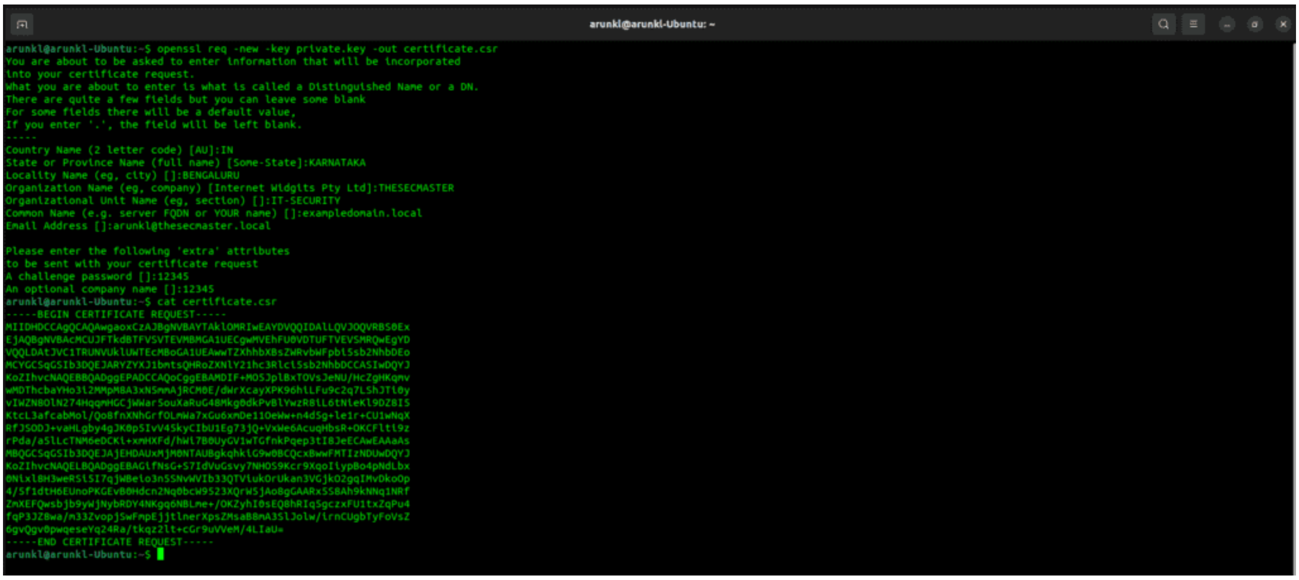


```
arunkl@arunkl-Ubuntu:~$ openssl rsa -in private.key -pubout -out public.key
writing RSA key
arunkl@arunkl-Ubuntu:~$ cat public.key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwMgX4w7kmmUHFM5Wwl41
T8dxmAcqqa/AwNOFxtpgejeLYwykzwDfE3maYCNEIzQT91atdxrJc8r3qGIsW71z
arstKElOLTK8hZk3w6U3bvgeqqYcYKNZZqvmi5dpG4bjwySDR2Q+8GVjDNHyIvq0
2J4qX0Nnwjkq1wvdp9xpsyiX9Cjx+dc2Eat84uZZrvEa7rGYN7XU55bD6fh3mD6V
7Wv4JTXA2pdF8lI4Mn69ocuBvLiAkrSnki9XjmTIIhtTUSDveND5XFZ7oBy6odux
H44oIWW2L30s91r9rmUtxM0zp4MIqL7GYdcV3+FaLsHRTIZXXBMZ+eQ+p6ne0jwl
4QIDAQAB
-----END PUBLIC KEY-----
arunkl@arunkl-Ubuntu:~$ 
```

#3. OpenSSL Commands to a Generate CSR

```
$ openssl req -new -key private.key -out certificate.csr
```

This command will prompt you to enter a variety of information, such as the common name, organization name, organization unit, country code, email address, optional password, and many more. Enter the valid input it asks to generate the CSR. This command generates a CSR and saves it in a file named ‘certificate.csr.’



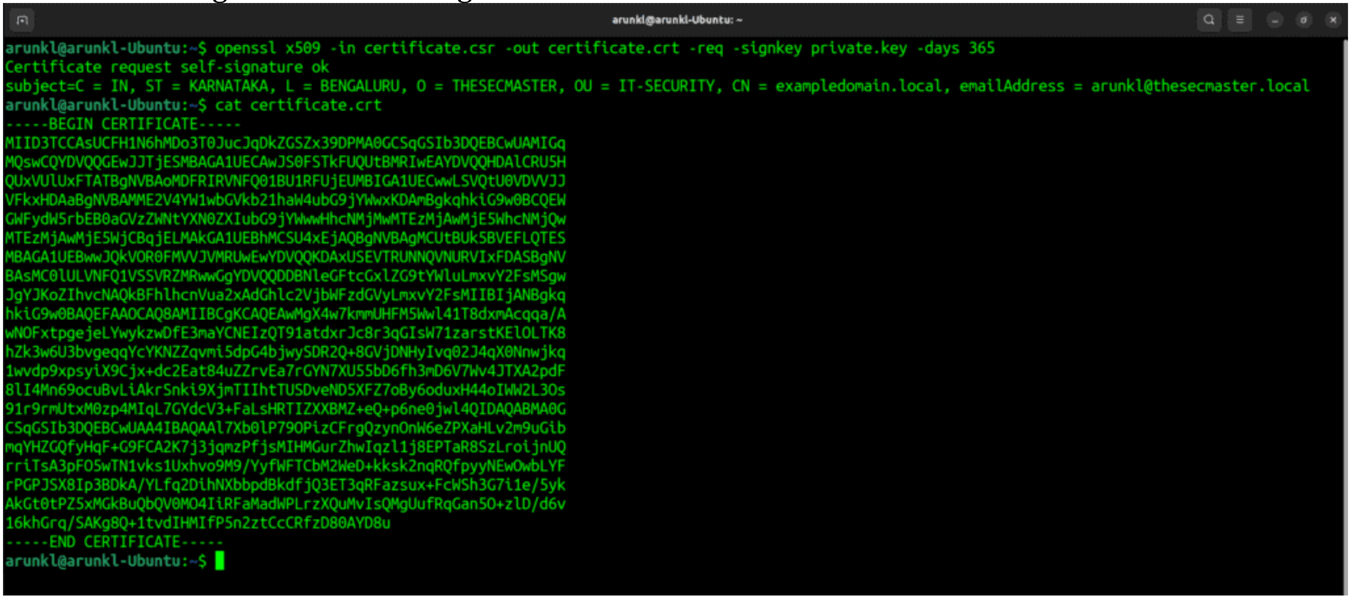
```
arunkl@arunkl-Ubuntu:~$ openssl req -new -key private.key -out certificate.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can 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]:IN
State or Province Name (full name) [Some-State]:KARNATAKA
Locality Name (eg, city) []:BENGALURU
Organization Name (eg, company) [Internet Widgits Pty Ltd]:THESECMASTER
Organizational Unit Name (eg, section) []:IT-SECURITY
Common Name (e.g. server FQDN or YOUR name) []:exampledomain.local
Email Address []:arunkl@thesecmaster.local

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:12345
An optional company name []:12345
arunkl@arunkl-Ubuntu:~$ cat certificate.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIDHDCCAgQCAQAwgaoxCzAJBgNVBAYTAklOMRIwEAYDVQQIDAlLQVJOQVRBS0Ex
EjAQBgNVBAcMCUJFTkdBTFVSVTEVMBMGA1UECgwMVEhFU0VDTUFTVEVSMRQwEgYD
VQQLDAtJVC1TRUNVUklUWTEcMBoGA1UEAwwTZXhhbXBsZWRvbWFpbi5sb2NhbDEo
MCYGCSqGSIb3DQEJARYZYXJ1bmtsQHRoZXNlY21hc3Rlci5sb2NhbDCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAMDIF+MO5JplBxTOVsJeNU/HcZgHKqmv
wMDThcbaYHo3l2MMpMBA3xN5mmAjRCM0E/dWrXcayXPK96hlLFu9c2q7LShJTi0y
vIWZNBOlN274HqqmHGCjWWar5ouXaRuG48Mkg0dkPvBlYwzR8iL6tNieKl9DZ8I5
KtcL3afcabMol/QoBfnXNhGrfOLmWa7xGu6xmDe11OeWw+n4d5g+le1r+CU1wNqX
RfJSODJ+vaHLgby4gJK0p5IvV45kyCIbU1Eg73jQ+VxWe6AcuqHbsR+OKCFlti9z
rPda/aSlLcTNM6eDCKi+xmHXFd/hWi7B0UyGV1wTGfnkPqep3tI8JeECAwEAAaAs
MBQGCSqGSIb3DQEJAjEHDAUxMjM0NTAUBgkqhkiG9w0BCQcxBwwFMTIzNDUwDQYJ
KoZIhvcNAQELBQADggEBAGifNsG+S7IdVuGsvy7NHOS9Kcr9XqoIiypBo4pNdLbx
0Nixl8H3weRSiSI7qjWBeio3n5SNvWVIb33QTViukOrUkan3VGjkO2gqIMvDkoOp
4/Sf1dtH6EUnoPKGEvB0Hdcn2Nq0bcW9523XQrW5jAoBgGAARx5S8Ah9kNNq1NRf
ZmXEFQwsbjb9yWjNybRDY4NKgq6NBLme+/OKZyhI0sEQ8hRIq5gczxFU1txZqPu4
fqP3JZBwa/m33ZvopjSwFmpEjjtlnerXpsZMsaB8mA3SlJolw/irnCUgbTyFoVsZ
6gvQgv0pwqeseYq24Ra/tkqz2lt+cGr9uVVeM/4LIaU=
-----END CERTIFICATE REQUEST-----
arunkl@arunkl-Ubuntu:~$ 
```

#4. OpenSSL Command to Generate a self-signed Certificate

```
$ openssl x509 -in certificate.csr -out certificate.crt -req -signkey private.key -days 365
```

This command generates a self-signed certificate and saves it in a file named ‘certificate.crt.’

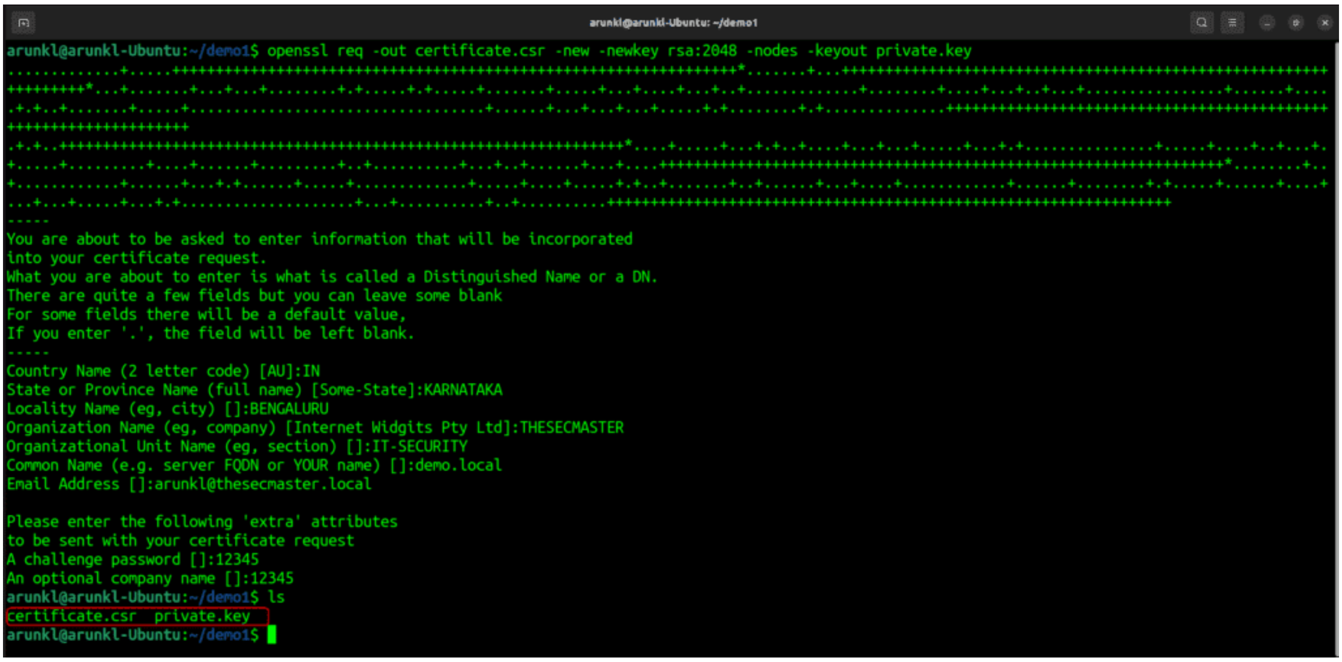


```
arunkl@arunkl-Ubuntu:~$ openssl x509 -in certificate.csr -out certificate.crt -req -signkey private.key -days 365
Certificate request self-signature ok
subject=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = exampledomain.local, emailAddress = arunkl@thesecmaster.local
arunkl@arunkl-Ubuntu:~$ cat certificate.crt
-----BEGIN CERTIFICATE-----
MIID3TCCAsUCFH1N6hMDo3T0JucJqDkZGSZx39DPMA0GCSqGSIb3DQEBCwUAMIGq
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxHDAaBgNVBAMME2V4YW1wbGVkb21haW4ubG9jYWwxKDAmBgkqhkiG9w0BCQEW
GWFydW5rbEB0aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTEzMjAwMjE5WhcNMjQw
MTEzMjAwMjE5WjCBqjELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTES
MBAGA1UEBwwJQkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNV
BAsMC0lULVNFQ1VSSVRZMRwwGgYDVQQDDBNleGFtcGxlZG9tYWluLmxvY2FsMSgw
JgYJKoZIhvcNAQkBFhlhcnVua2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwMgX4w7kmmUHFM5Wwl41T8dxmAcqqa/A
wNOFxtpgejeLYwykzwDfE3maYCNEIzQT91atdxrJc8r3qGIsW71zarstKElOLTK8
hZk3w6U3bvgeqqYcYKNZZqvmi5dpG4bjwySDR2Q+8GVjDNHyIvq02J4qX0Nnwjkq
1wvdp9xpsyiX9Cjx+dc2Eat84uZZrvEa7rGYN7XU55bD6fh3mD6V7Wv4JTXA2pdF
8lI4Mn69ocuBvLiAkrSnki9XjmTIIhtTUSDveND5XFZ7oBy6oduxH44oIWW2L3Os
91r9rmUtxM0zp4MIqL7GYdcV3+FaLsHRTIZXXBMZ+eQ+p6ne0jwl4QIDAQABMA0G
CSqGSIb3DQEBCwUAA4IBAQAAl7Xb0lP79OPizCFrgQzynOnW6eZPXaHLv2m9uGib
mqYHZGQfyHqF+G9FCA2K7j3jqmzPfjsMIHMGurZhwIqzl1j8EPTaR8SzLroijnUQ
rriTsA3pFO5wTN1vks1Uxhvo9M9/YyfWFTCbM2WeD+kksk2nqRQfpyyNEwOwbLYF
rPGPJSX8Ip3BDkA/YLfq2DihNXbbpdBkdfjQ3ET3qRFazsux+FcWSh3G7i1e/5yk
AkGt0tPZ5xMGkBuQbQV0MO4IiRFaMadWPLrzXQuMvIsQMgUufRqGan5O+zlD/d6v
16khGrq/SAKg8Q+1tvdIHMIfP5n2ztCcCRfzD80AYD8u
-----END CERTIFICATE-----
arunkl@arunkl-Ubuntu:~$ 
```

#5. OpenSSL Command to Generate a new Private Key and Certificate Signing Request in a Single Command

```
openssl req -out certificate.csr -new -newkey rsa:2048 -nodes -keyout private.key
```

This command generates a new private key and CSR and saves them in a file named ‘certificate.csr’ and ‘private.key.’



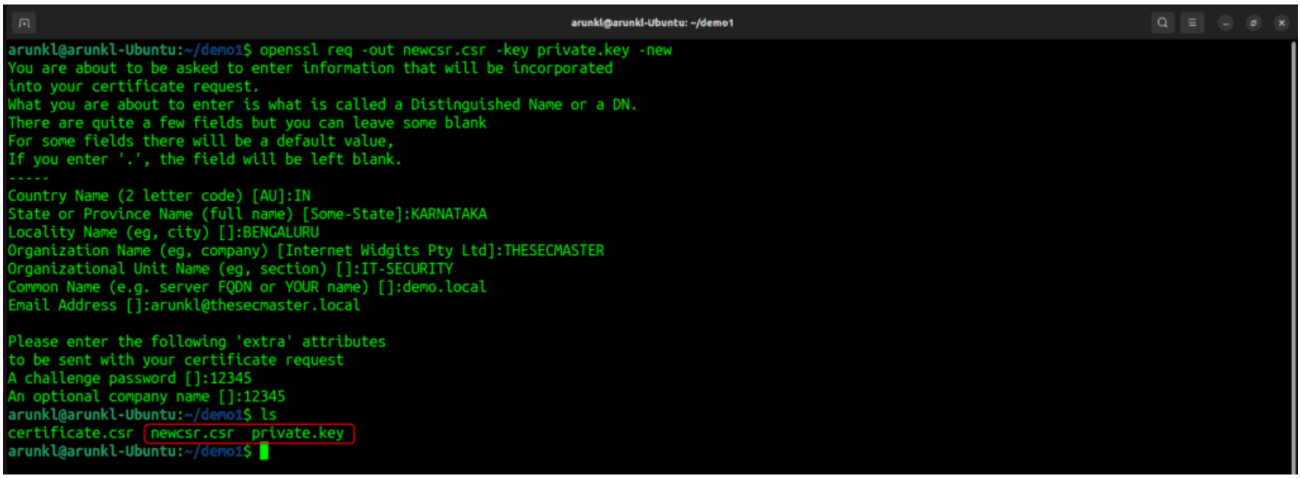
```
arunkl@arunkl-Ubuntu:~/demo1$ openssl req -out certificate.csr -new -newkey rsa:2048 -nodes -keyout private.key
..............+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.......+...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++*...+.......+...+...+........+.+.....+.+.....+........+.....+...+....+...+..+.............+........+....+...+..+...+...............+.......+....
.+.+..+.......+.....+..................................+......+...+...+...+.....+.+.........+.+.............+++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
.+.+..+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*....+.....+...+.+..+....+...+.....+...+.+..............+.....+....+..+...+.
+.....+.........+....+......+.........+..+..........+...+..+......+...+....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.........+..
+............+......+...+.+......+.....+..............+.....+....+.....+.+..+........+..+......+...+....+.............+......+.......+.+.....+......+....+
...+...+.....+...+.+.....................+...+..........+..+.........+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
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]:IN
State or Province Name (full name) [Some-State]:KARNATAKA
Locality Name (eg, city) []:BENGALURU
Organization Name (eg, company) [Internet Widgits Pty Ltd]:THESECMASTER
Organizational Unit Name (eg, section) []:IT-SECURITY
Common Name (e.g. server FQDN or YOUR name) []:demo.local
Email Address []:arunkl@thesecmaster.local

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:12345
An optional company name []:12345
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.csr  private.key
arunkl@arunkl-Ubuntu:~/demo1$
```

#6. OpenSSL Command to Generate a Certificate Signing Request (CSR) for an existing Private Key

```
openssl req -out newcsr.csr -key private.key -new
```

This command generates a new CSR and saves it in a file named ‘newcsr.csr’ for the existing private key.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl req -out newcsr.csr -key private.key -new
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]:IN
State or Province Name (full name) [Some-State]:KARNATAKA
Locality Name (eg, city) []:BENGALURU
Organization Name (eg, company) [Internet Widgits Pty Ltd]:THESECMASTER
Organizational Unit Name (eg, section) []:IT-SECURITY
Common Name (e.g. server FQDN or YOUR name) []:demo.local
Email Address []:arunkl@thesecmaster.local

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:12345
An optional company name []:12345
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.csr  newcsr.csr  private.key
arunkl@arunkl-Ubuntu:~/demo1$
```

#7. OpenSSL Command to Generate a Certificate Signing Request (CSR) based on an existing Certificate

```
openssl x509 -x509toreq -in certificate.crt -out newcsr2.csr -signkey private.key
```

This command generates a new Certificate Signing Request (CSR) and save it in a file named ‘newcsr2.csr.’

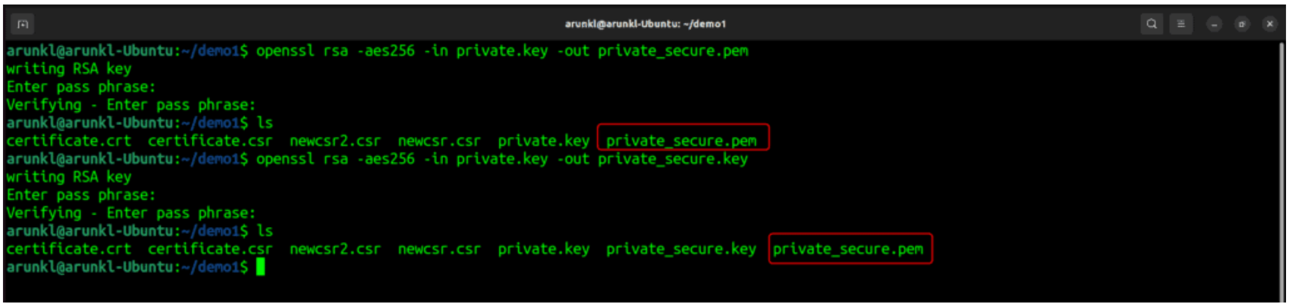
#8. OpenSSL Command to Encrypt or add Passphrase to a Private Key

```
openssl rsa -aes256 -in private.key -out private_secure.pem
```

OR

```
openssl rsa -aes256 -in private.key -out private_secure.key
```

You can keep the encrypted key pair in .key or .pem format. Both are valid.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl rsa -aes256 -in private.key -out private_secure.pem
writing RSA key
Enter pass phrase:
Verifying - Enter pass phrase:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  newcsr2.csr  newcsr.csr  private.key  private_secure.pem
arunkl@arunkl-Ubuntu:~/demo1$ openssl rsa -aes256 -in private.key -out private_secure.key
writing RSA key
Enter pass phrase:
Verifying - Enter pass phrase:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  newcsr2.csr  newcsr.csr  private.key  private_secure.key  private_secure.pem
arunkl@arunkl-Ubuntu:~/demo1$
```

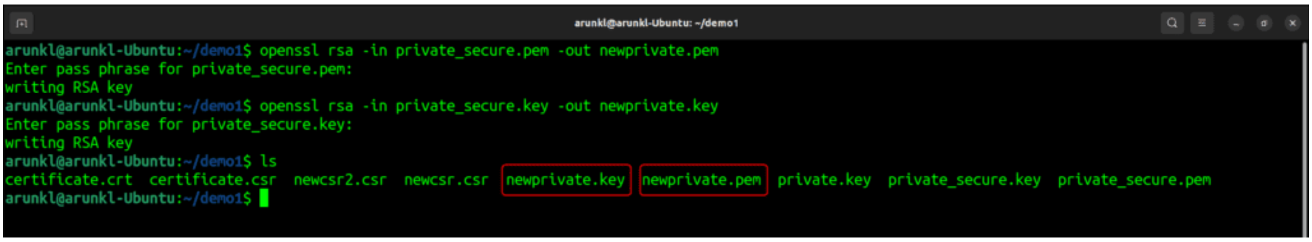
#9. OpenSSL Command to Decrypt or Remove a Passphrase from a Private Key

```
openssl rsa -in private_secure.pem -out newprivate.pem
```

OR

```
openssl rsa -in private_secure.key -out newprivate.key
```

This command removes the passphrase and decrypts the private key, and saves it in a file named newprivate.key.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl rsa -in private_secure.pem -out newprivate.pem
Enter pass phrase for private_secure.pem:
writing RSA key
arunkl@arunkl-Ubuntu:~/demo1$ openssl rsa -in private_secure.key -out newprivate.key
Enter pass phrase for private_secure.key:
writing RSA key
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  newcsr2.csr  newcsr.csr  newprivate.key  newprivate.pem  private.key  private_secure.key  private_secure.pem
arunkl@arunkl-Ubuntu:~/demo1$
```

OpenSSL Commands to Convert SSL Certificates

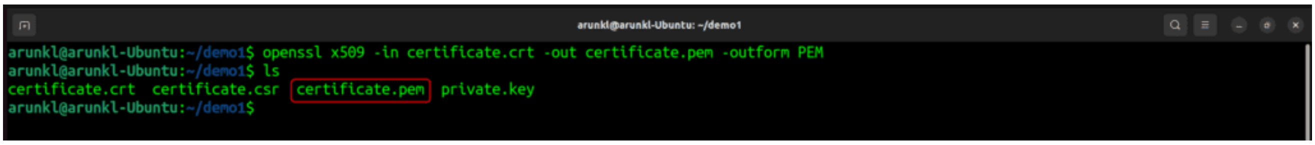
There could be several reasons why you may need to convert SSL certificates from one format to another. The main reason would be compatibility. Different systems and applications may require certificates to be in a specific format. Some systems or applications may not be able to handle certain file formats. In such cases, it is necessary to convert the certificate to a different format that the system or application can understand. For example, some web servers may require certificates to be in PEM format, while others may require them to be in PKCS#12 format.

OpenSSL can be used to convert SSL certificates between different formats. Let's see how to convert from one file format to another.

#1. OpenSSL Commands to Convert a Certificate from CRT to PEM

```
openssl x509 -inform der -in certificate.crt -out certificate.pem
```

This command will convert the certificate in CRT format named "certificate.crt" to PEM format and save it in a file named "certificate.pem" in the current directory. Note: this command can be used to convert .cer certificates.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl x509 -in certificate.crt -out certificate.pem -outform PEM
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$
```

PEM Certificate vs PEM Key file

Both the PEM Certificate and PEM Key file are different entities and made for different purpose. They can't be interchangeable. Don't be confused between them.
PEM stands for Privacy Enhanced Mail. PEM Certificate is a file that is used to store X.509 certificates. Where as PEM Key file is a file used to store private and publick key pair.
A PEM certificate typically contains the public key of a certificate and not the private key.
A PEM certificate file typically contains the following information:

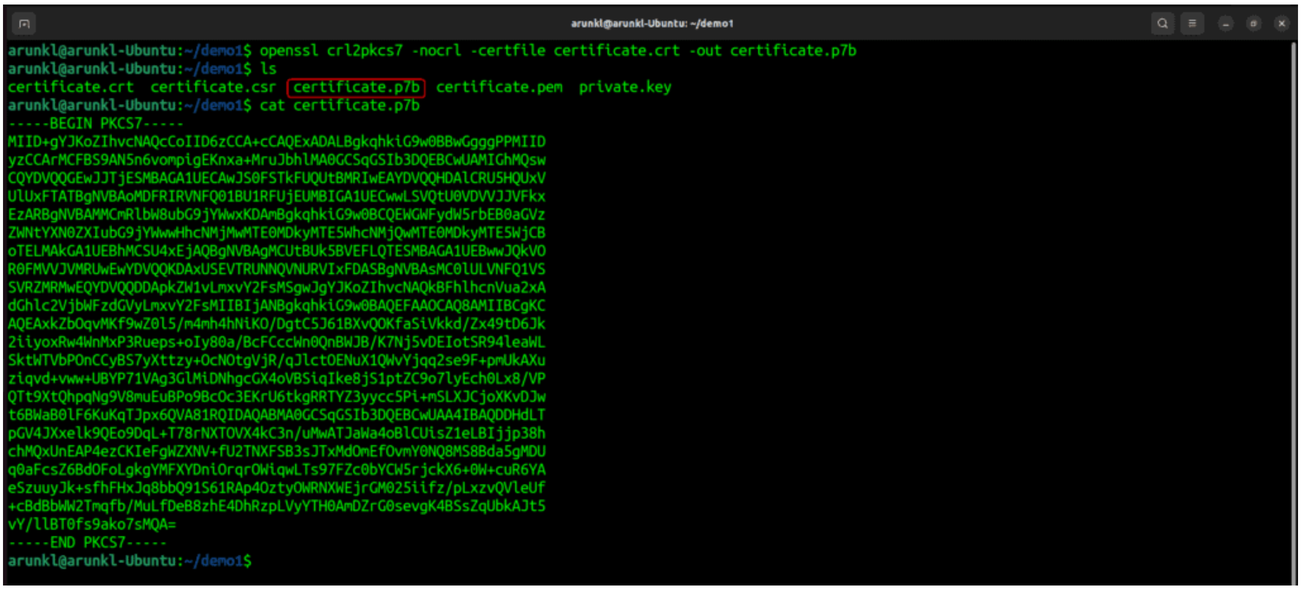
- The X.509 certificate in base64 encoded format
- The certificate's public key
- Optionally, any intermediate CA certificates

Use 'cat' command to differentiate the PEM Key file from the PEM Certificate. A PEM Key file contains a private key would typically have a header that says "BEGIN RSA PRIVATE KEY" or "BEGIN PRIVATE KEY".

#2. OpenSSL Commands to Convert a Certificate from CRT to PKCS7

```
openssl crl2pkcs7 -nocrl -certfile certificate.crt -out certificate.p7b
```

This command will convert the certificate in CRT format named "certificate.crt" to PKCS7 format and save it in a file named "certificate.p7b" in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl crl2pkcs7 -nocrl -certfile certificate.crt -out certificate.p7b
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.p7b
-----BEGIN PKCS7-----
MIID+gYJKoZIhvcNAQcCoIID6zCCA+cCAQExADALBgkqhkiG9w0BBwGgggPPMIID
yzCCArMCFBS9AN5n6vompigEKnxa+MruJbhlMA0GCSqGSIb3DQEBCwUAMIGhMQsw
CQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5HQUxV
UlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJVFkx
EzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0aGVz
ZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MDkyMTE5WhcNMjQwMTE0MDkyMTE5WjCB
oTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJQkVO
R0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNFQ1VS
SVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVua2xA
dGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
AQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49tD6Jk
2iiyoxRw4WnMxP3Rueps+oIy80a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94leaWL
SktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmUkAXu
ziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx8/VP
QTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXKvDJw
t6BWaB0lF6KuKqTJpx6QVA81RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQDDHdLT
pGV4JXxelk9QEo9DqL+T78rNXTOVX4kC3n/uMwATJaWa4oBlCUisZ1eLBIjjp38h
chMQxUnEAP4ezCKIeFgWZXNV+fU2TNXFSB3sJTxMdOmEfOvmY0NQ8MS8Bda5gMDU
q0aFcsZ6BdOFoLgkgYMFXYDniOrqrOWiqwLTs97FZc0bYCW5rjckX6+0W+cuR6YA
eSzuuyJk+sfhFHxJq8bbQ91S61RAp4OztyOWRNXWEjrGM025iifz/pLxzvQVleUf
+cBdBbWW2Tmqfb/MuLfDeB8zhE4DhRzpLVyYTH0AmDZrG0sevgK4BSsZqUbkAJt5
vY/llBT0fs9ako7sMQA=
-----END PKCS7-----
arunkl@arunkl-Ubuntu:~/demo1$
```

#3. OpenSSL Commands to Convert a Certificate from CRT to PKCS#12

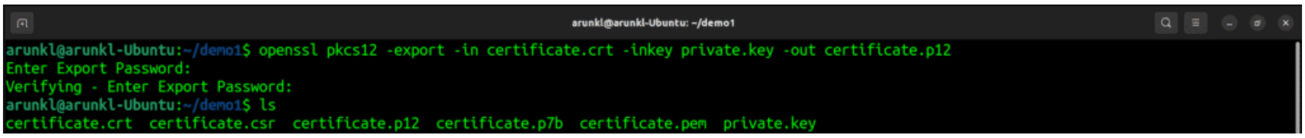
PKCS#12 and PFX are both file formats that are used to store X.509 certificates and private keys. The main difference between them is the file extension, PKCS#12 uses the file extension .p12 or .pfx, and PFX uses the file extension .pfx. Both formats are used for the same purpose, which is to provide a secure way to store and transport digital certificates and private keys. PKCS#12 is the standard for storing the certificate and private key developed by RSA Laboratories, while PFX is Microsoft's variant of the PKCS#12 standard.

```
openssl pkcs12 -export -in certificate.crt -inkey private.key -out certificate.p12
```

OR

```
openssl pkcs12 -export -in certificate.crt -inkey private.key -out certificate.pfx
```

This command will convert the certificate in CRT format named “certificate.crt” and private key named “private.key” to PKCS#12 format and save it in a file named “certificate.p12” in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs12 -export -in certificate.crt -inkey private.key -out certificate.p12
Enter Export Password:
Verifying - Enter Export Password:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p12  certificate.p7b  certificate.pem  private.key
```

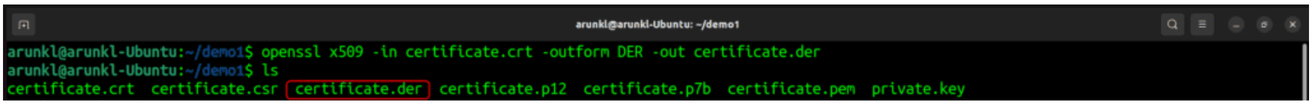
It’s important to note that PKCS#12 format is also known as P12 format, and it is used to store one or more certificates and private key. It’s encrypted by default and can’t be decoded in base64, and It will prompt you to enter a password to protect the certificate and key.

It’s important to note that PKCS#12 format is also known as P12 format, and it is used to store one or more certificates and private key. It’s encrypted by default and can’t be decoded in base64, and It will prompt you to enter a password to protect the certificate and key.

#4. OpenSSL Commands to Convert a Certificate from CRT to DER

```
openssl x509 -in certificate.crt -outform DER -out certificate.der
```

This command will convert the certificate in CRT format named “certificate.crt” to DER format and save it in a file named “certificate.der” in the current directory.



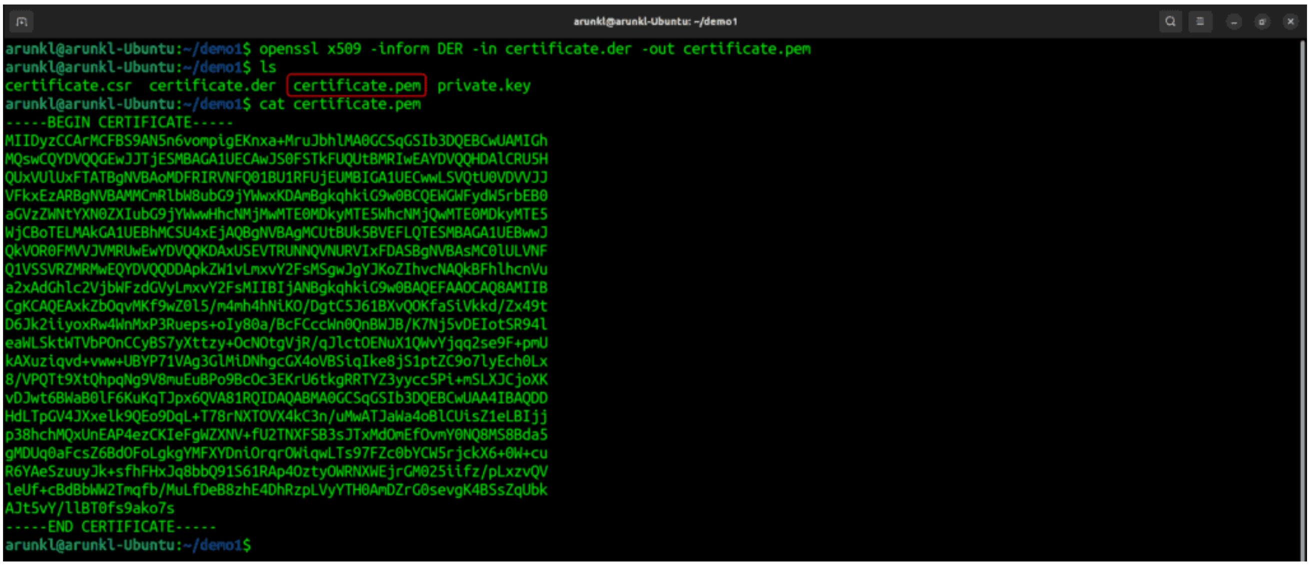
```
arunkl@arunkl-Ubuntu:~/demo1$ openssl x509 -in certificate.crt -outform DER -out certificate.der
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.p7b  certificate.pem  private.key
```

It’s important to note that DER is a binary format and can’t be decoded in base64, and it is typically used to store X.509 certificates and CRLs (Certificate Revocation Lists) in a compact and efficient way.

#5. OpenSSL Commands to Convert a Certificate from DER to PEM

```
openssl x509 -inform DER -in certificate.der -out certificate.pem
```

This command will convert the certificate in DER format named “certificate.der” to PEM format and save it in a file named “certificate.pem” in the current directory.

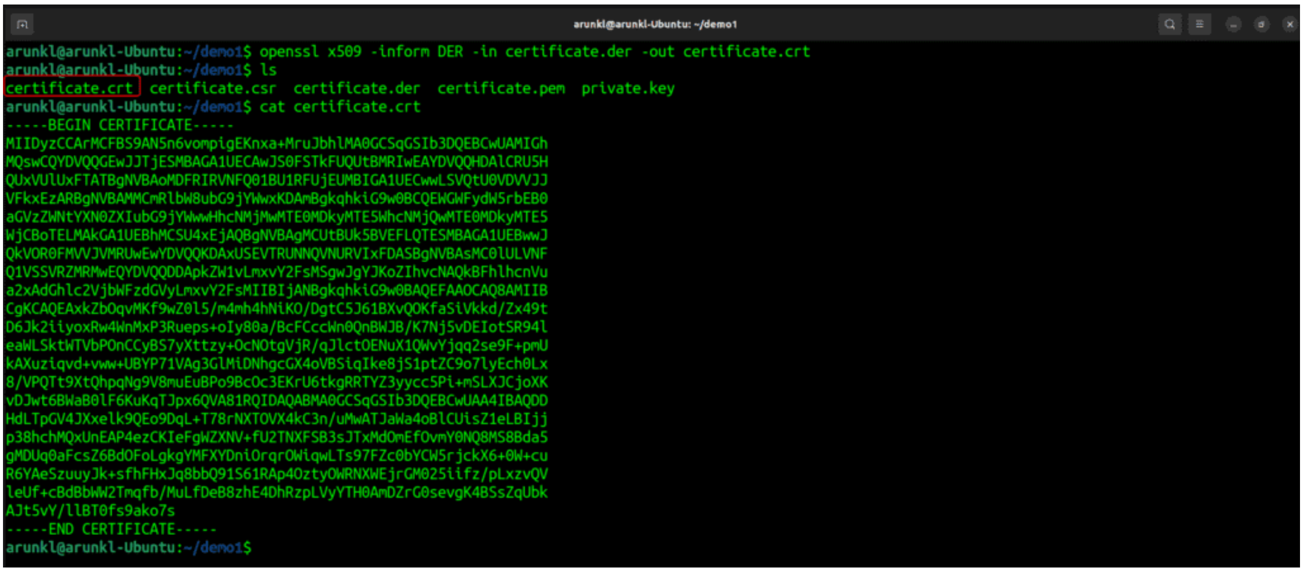


```
arunkl@arunkl-Ubuntu:~/demo1$ openssl x509 -inform DER -in certificate.der -out certificate.pem
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.csr  certificate.der  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.pem
-----BEGIN CERTIFICATE-----
MIIDyzCCArMCFBS9AN5n6vompigEKnxa+MruJbhlMA0GCSqGSIb3DQEBCwUAMIGh
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxEzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0
aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MDkyMTE5WhcNMjQwMTE0MDkyMTE5
WjCBoTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJ
QkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNF
Q1VSSVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVu
a2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49t
D6Jk2iiyoxRw4WnMxP3Rueps+oIy80a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94l
eaWLSktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmU
kAXuziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx
8/VPQTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXK
vDJwt6BWaB0lF6KuKqTJpx6QVA81RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQDD
HdLTpGV4JXxelk9QEo9DqL+T78rNXTOVX4kC3n/uMwATJaWa4oBlCUisZ1eLBIjj
p38hchMQxUnEAP4ezCKIeFgWZXNV+fU2TNXFSB3sJTxMdOmEfOvmY0NQ8MS8Bda5
gMDUq0aFcsZ6BdOFoLgkgYMFXYDniOrqrOWiqwLTs97FZc0bYCW5rjckX6+0W+cu
R6YAeSzuuyJk+sfhFHxJq8bbQ91S61RAp4OztyOWRNXWEjrGM025iifz/pLxzvQV
leUf+cBdBbWW2Tmqfb/MuLfDeB8zhE4DhRzpLVyYTH0AmDZrG0sevgK4BSsZqUbk
AJt5vY/llBT0fs9ako7s
-----END CERTIFICATE-----
arunkl@arunkl-Ubuntu:~/demo1$
```

#6. OpenSSL Commands to Convert a Certificate from DER to CRT

```
openssl x509 -inform DER -in certificate.der -out certificate.crt
```

This command will convert the certificate in DER format named “certificate.der” to CRT format and save it in a file named “certificate.crt” in the current directory.



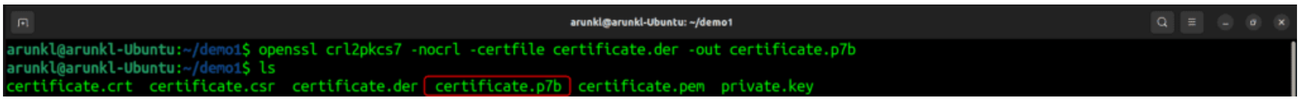
```
arunkl@arunkl-Ubuntu:~/demo1$ openssl x509 -inform DER -in certificate.der -out certificate.crt
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.crt
-----BEGIN CERTIFICATE-----
MIIDyzCCArMCFBS9AN5n6vompigEKnxa+MruJbhlMA0GCSqGSIb3DQEBCwUAMIGh
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxEzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0
aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MDkyMTE5WhcNMjQwMTE0MDkyMTE5
WjCBoTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJ
QkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNF
Q1VSSVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVu
a2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49t
D6Jk2iiyoxRw4WnMxP3Rueps+oIy80a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94l
eaWLSktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmU
kAXuziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx
8/VPQTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXK
vDJwt6BWaB0lF6KuKqTJpx6QVA81RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQDD
HdLTpGV4JXxelk9QEo9DqL+T78rNXTOVX4kC3n/uMwATJaWa4oBlCUisZ1eLBIjj
p38hchMQxUnEAP4ezCKIeFgWZXNV+fU2TNXFSB3sJTxMdOmEfOvmY0NQ8MS8Bda5
gMDUq0aFcsZ6BdOFoLgkgYMFXYDniOrqrOWiqwLTs97FZc0bYCW5rjckX6+0W+cu
R6YAeSzuuyJk+sfhFHxJq8bbQ91S61RAp4OztyOWRNXWEjrGM025iifz/pLxzvQV
leUf+cBdBbWW2Tmqfb/MuLfDeB8zhE4DhRzpLVyYTH0AmDZrG0sevgK4BSsZqUbk
AJt5vY/llBT0fs9ako7s
-----END CERTIFICATE-----
arunkl@arunkl-Ubuntu:~/demo1$
```

It’s important to note that the CRT file format is just a container format, and the data inside it can be encoded in various ways, and it is commonly used to store X.509 certificates.

#7. OpenSSL Commands to Convert a Certificate from DER to PKCS7

```
openssl crl2pkcs7 -nocrl -certfile certificate.der -out certificate.p7b
```

This command will convert the certificate in DER format named “certificate.der” to PKCS7 format and save it in a file named “certificate.p7b” in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl crl2pkcs7 -nocrl -certfile certificate.der -out certificate.p7b
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p7b  certificate.pem  private.key
```

It's important to note that PKCS7 format is also known as P7B format, and it is used to store one or more certificates, typically including the end-user certificate and any intermediate CA certificates, but it doesn't contain the private key.

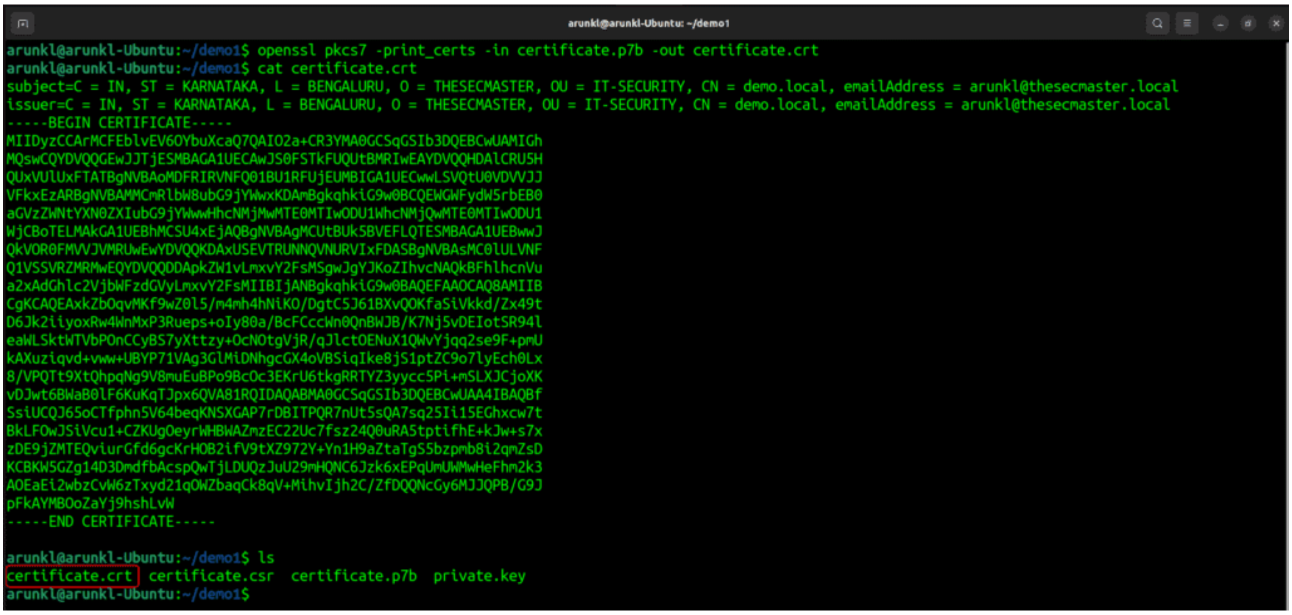
#8. OpenSSL Commands to Convert a Certificate from DER to PKCS#12

Practically, it is not possible to convert DER to PKCS#12, since DER is an unreadable binary file. The Certificate should be in either CRT or CER to convert to PKCS#12 or P12. Please refer to the #3 command for more information.

#9. OpenSSL Commands to Convert a Certificate from PKCS7 to CRT

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.crt
```

This command will convert the certificate in PKCS7 format named "certificate.p7b" to CRT format and save it in a file named "certificate.crt" in the current directory.



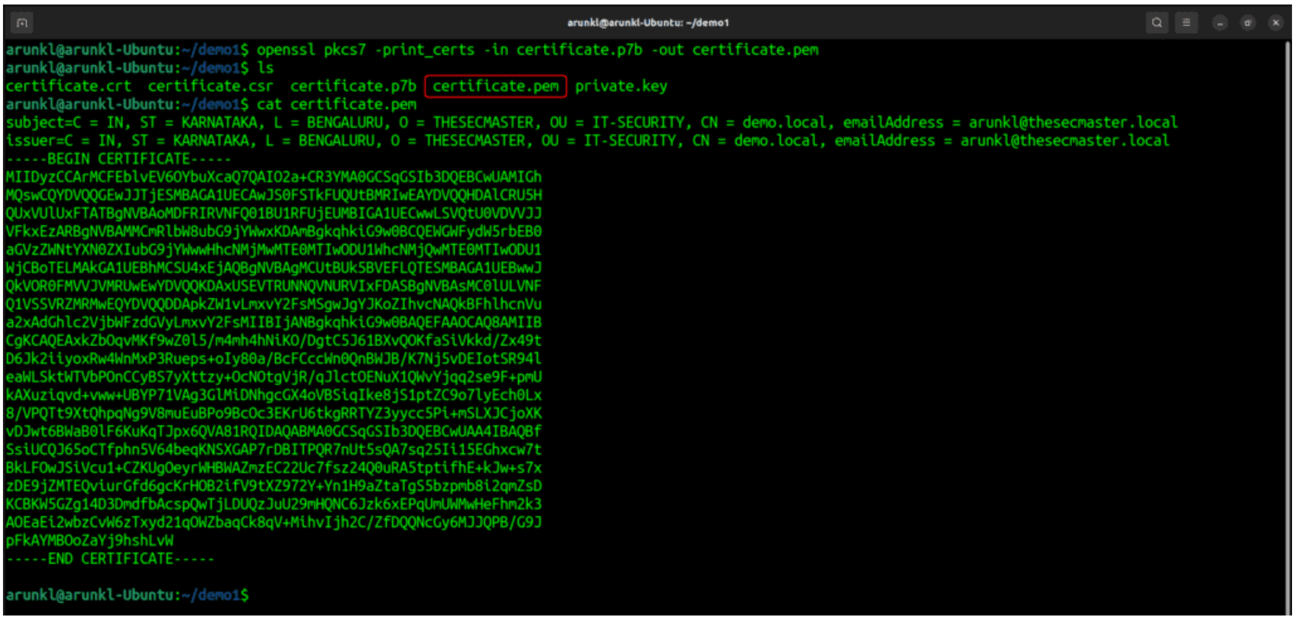
```
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs7 -print_certs -in certificate.p7b -out certificate.crt
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.crt
subject=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
issuer=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
-----BEGIN CERTIFICATE-----
MIIDyzCCArMCFEblvEV6OYbuXcaQ7QAIO2a+CR3YMA0GCSqGSIb3DQEBCwUAMIGh
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxEzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0
aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MTIwODU1WhcNMjQwMTE0MTIwODU1
WjCBoTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJ
QkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNF
Q1VSSVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVu
a2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49t
D6Jk2iiyoxRw4WnMxP3Rueps+oIy80a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94l
eaWLSktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmU
kAXuziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx
8/VPQTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXK
vDJwt6BWaB0lF6KuKqTJpx6QVA81RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBf
SsiUCQJ65oCTfphn5V64beqKNSXGAP7rDBITPQR7nUt5sQA7sq25Ii15EGhxcw7t
BkLFOwJSiVcu1+CZKUgOeyrWHBWAZmzEC22Uc7fsz24Q0uRA5tptifhE+kJw+s7x
zDE9jZMTEQviurGfd6gcKrHOB2ifV9tXZ972Y+Yn1H9aZtaTgS5bzpmb8i2qmZsD
KCBKW5GZg14D3DmdfbAcspQwTjLDUQzJuU29mHQNC6Jzk6xEPqUmUWMwHeFhm2k3
AOEaEi2wbzCvW6zTxyd21qOWZbaqCk8qV+MihvIjh2C/ZfDQQNcGy6MJJQPB/G9J
pFkAYMBOoZaYj9hshLvW
-----END CERTIFICATE-----

arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p7b  private.key
arunkl@arunkl-Ubuntu:~/demo1$
```

#10. OpenSSL Commands to Convert a Certificate from PKCS7 to PEM

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.pem
```

This command will convert the certificate in PKCS7 format named "certificate.p7b" to PEM format and save it in a file named "certificate.pem" in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs7 -print_certs -in certificate.p7b -out certificate.pem
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.pem
subject=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
issuer=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
-----BEGIN CERTIFICATE-----
MIIDyzCCArMCFEblvEV6OYbuXcaQ7QAIO2a+CR3YMA0GCSqGSIb3DQEBCwUAMIGh
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxEzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0
aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MTIwODU1WhcNMjQwMTE0MTIwODU1
WjCBoTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJ
QkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNF
Q1VSSVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVu
a2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49t
D6Jk2iiyoxRw4WnMxP3Rueps+oIy80a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94l
eaWLSktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmU
kAXuziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx
8/VPQTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXK
vDJwt6BWaB0lF6KuKqTJpx6QVA81RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBf
SsiUCQJ65oCTfphn5V64beqKNSXGAP7rDBITPQR7nUt5sQA7sq25Ii15EGhxcw7t
BkLFOwJSiVcu1+CZKUgOeyrWHBWAZmzEC22Uc7fsz24Q0uRA5tptifhE+kJw+s7x
zDE9jZMTEQviurGfd6gcKrHOB2ifV9tXZ972Y+Yn1H9aZtaTgS5bzpmb8i2qmZsD
KCBKW5GZg14D3DmdfbAcspQwTjLDUQzJuU29mHQNC6Jzk6xEPqUmUWMwHeFhm2k3
AOEaEi2wbzCvW6zTxyd21qOWZbaqCk8qV+MihvIjh2C/ZfDQQNcGy6MJJQPB/G9J
pFkAYMBOoZaYj9hshLvW
-----END CERTIFICATE-----

arunkl@arunkl-Ubuntu:~/demo1$
```

#11. OpenSSL Commands to Convert a Certificate from PKCS7 to PKCS#12

It is not possible to directly convert a certificate from PKCS7 format to PKCS#12 format. Because PKCS7 format is used for storing one or more certificates, typically including the end-user certificate and any intermediate CA certificates, but it doesn't contain the private key. At the same time, PKCS#12 format is used to store one or more certificates and a private key.

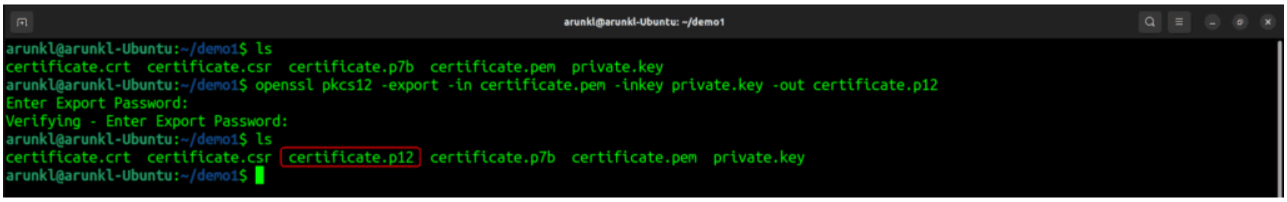
In order to convert a certificate from PKCS7 format to PKCS#12 format, you will need to first extract the private key and the certificate in PEM format and then use the OpenSSL command to create a PKCS#12 file which includes both the certificate and the private key.

Here is an example command:

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.pem


openssl pkcs12 -export -in certificate.pem -inkey private.key -out certificate.p12
```

This command will convert the certificate in PKCS7 format named "certificate.p7b" to PEM format using the first command and then using the second command. It will convert the PEM format certificate and private key to PKCS#12 format and save it in a file named "certificate.p12" in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs12 -export -in certificate.pem -inkey private.key -out certificate.p12
Enter Export Password:
Verifying - Enter Export Password:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ 
```

#12. OpenSSL Commands to Convert a Certificate from PKCS7 to DER

It is not possible to directly convert a certificate from PKCS7 format to DER format. Because PKCS7 format is used for storing one or more certificates, typically including the end-user certificate and any intermediate CA certificates, but it doesn't contain the private key. While DER format is a binary format, and it is typically used to store X.509 certificates and CRLs (Certificate Revocation Lists) in a compact and efficient way.

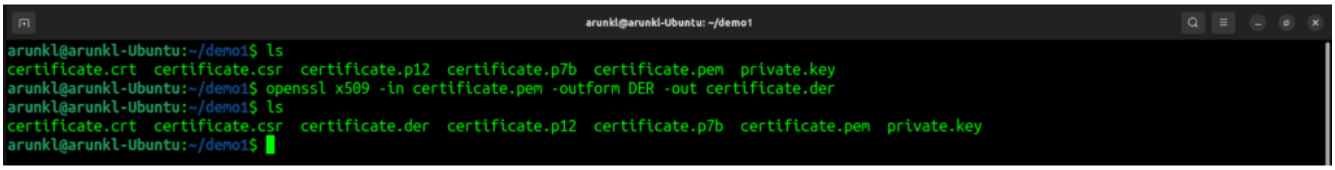
In order to convert a certificate from PKCS7 format to DER format, you will need to first extract the certificate in PEM format and then use the OpenSSL command to convert PEM to DER format.

Here is an example command:

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.pem


openssl x509 -in certificate.pem -outform DER -out certificate.der
```

This command will convert the certificate in PKCS7 format named “certificate.p7b” to PEM format using the first command, and then using the second command, it will convert the PEM format certificate to DER format and save it in a file named “certificate.der” in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ openssl x509 -in certificate.pem -outform DER -out certificate.der
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ 
```

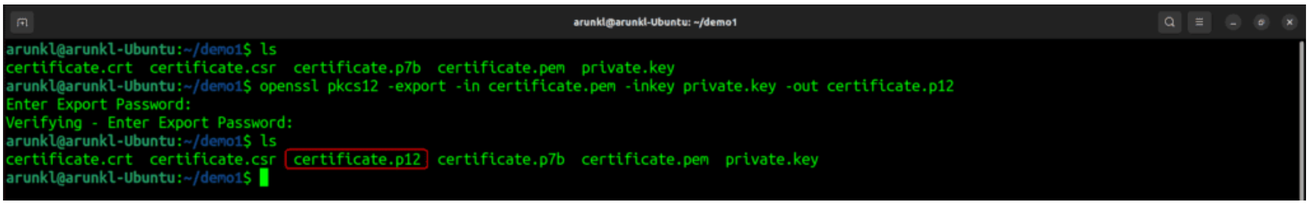
#13. OpenSSL Commands to Convert a Certificate from PEM to CRT

The PEM format and CRT format are both base64 container formats that are used to store X.509 certificates. PEM is used to store public certificates or the entire certificate chain (private key, public key, root certificates). So, a certificate in the PEM format can be used as it is in a CRT format because it’s the same format, and it just depends on the file extension. PEM can be used where ever CRT is being used. No conversion is required.

#14. OpenSSL Commands to Convert a Certificate from PEM to PKCS#12

```
openssl pkcs12 -export -in certificate.pem -inkey private.key -out certificate.p12
```

This command will convert the certificate in PEM format named “certificate.pem” and the private key named “private.key” to PKCS#12 format and save it in a file named “certificate.p12” in the current directory.



```
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs12 -export -in certificate.pem -inkey private.key -out certificate.p12
Enter Export Password:
Verifying - Enter Export Password:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ 
```

#15. OpenSSL Commands to Convert a Certificate from PEM to PKCS7

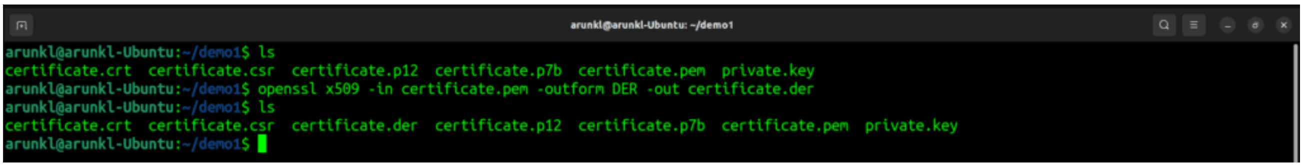
It is not possible to directly convert a certificate from PEM format to PKCS7 format. Because PKCS7 format is used for storing one or more certificates, typically including the end-user certificate and any intermediate CA certificates, but it doesn’t contain the private key. While PEM format can include just the public certificate or the entire certificate chain (private key, public key, root certificates), and it is base64 encoded.

In order to convert a certificate from PEM format to PKCS7 format, you will need to first extract the public certificate and any intermediate CA certificates from the PEM file, then use the OpenSSL command to create a PKCS7 file.

#16. OpenSSL Commands to Convert a Certificate from PEM to DER

```
openssl x509 -in certificate.pem -outform DER -out certificate.der
```

This command will convert the certificate in PEM format named “certificate.pem” to DER format and save it in a file named “certificate.der” in the current directory.

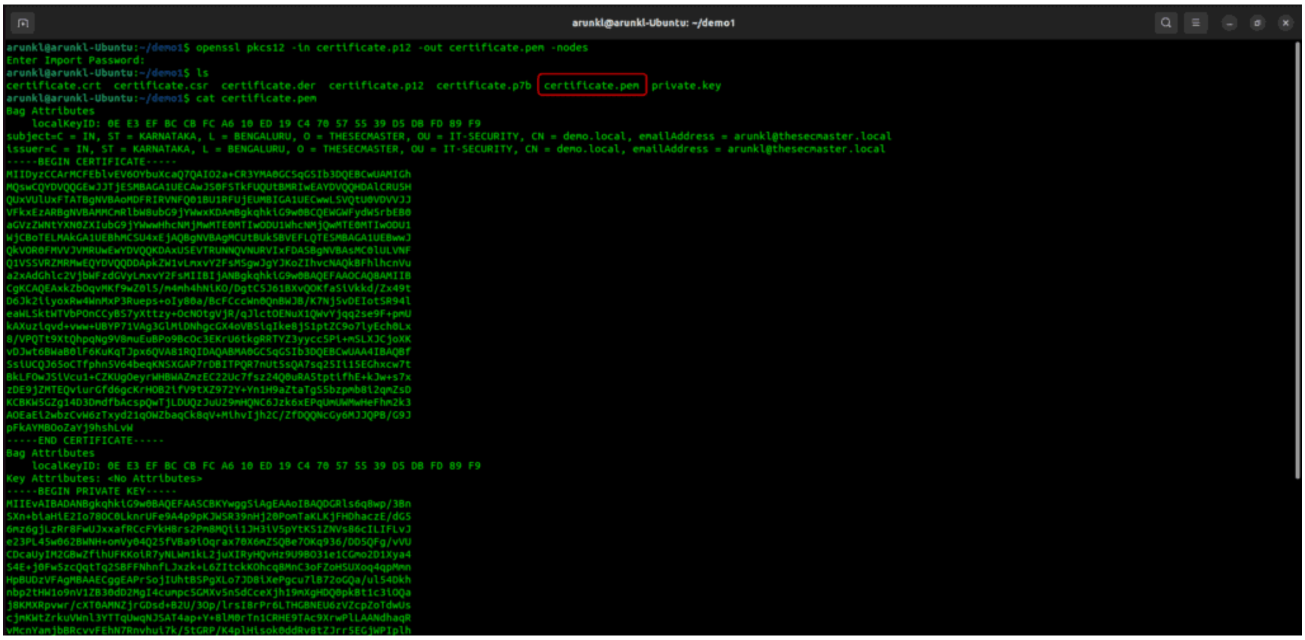


```
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ openssl x509 -in certificate.pem -outform DER -out certificate.der
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ 
```

#17. OpenSSL Commands to Convert a Certificate from PKCS#12 to PEM

```
openssl pkcs12 -in certificate.p12 -out certificate.pem -nodes
```

This command will convert the certificate in PKCS#12 format named “certificate.p12” to PEM format and save it in a file named “certificate.pem” in the current directory. It will prompt you to enter the password that was used to encrypt the certificate in PKCS#12 format. If the certificate was not encrypted, you don’t have to use the `–nocerts` option.



```
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs12 -in certificate.p12 -out certificate.pem -nodes
Enter Import Password:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.pem
Bag Attributes
    localKeyID: 0E E3 EF BC CB FC A6 10 ED 19 C4 70 57 55 39 D5 DB FD 89 F9
subject=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
issuer=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
-----BEGIN CERTIFICATE-----
MIIDyzCCArMCFEblvEV6OYbuXcaQ7QAIO2a+CR3YMA0GCSqGSIb3DQEBCwUAMIGh
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxEzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0
aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MTIwODU1WhcNMjQwMTE0MTIwODU1
WjCBoTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJ
QkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNF
Q1VSSVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVu
a2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49t
D6Jk2ilyoxRw4WnMxP3Rueps+oIyB0a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94l
eaWLSktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmU
kAXuziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx
8/VPQTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXK
vDJwt6BWaB0lF6KuKqTJpx6QVAB1RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBf
SsiUCQJ65oCTfphn5V64beqKNSXGAP7rDBITPQR7nUt5sQA7sq25Ii15EGhxcw7t
BkLFOwJSiVcu1+CZKUgOeyrWHBWAZmzEC22Uc7fsz24Q0uRA5tptifhE+kJw+s7x
zDE9jZMTEQviurGfd6gcKrHOB2ifV9tXZ972Y+Yn1H9aZtaTgS5bzpmb8i2qmZsD
KCBKW5GZg14D3DmdfbAcspQwTjLDUQzJuU29mHQNC6Jzk6xEPqUmUWMwHeFhm2k3
AOEaEi2wbzCvW6zTxyd21qOWZbaqCkBqV+MihvIjh2C/ZfDQQNcGy6MJJQPB/G9J
pFkAYMBOoZaYj9hshLvW
-----END CERTIFICATE-----
Bag Attributes
    localKeyID: 0E E3 EF BC CB FC A6 10 ED 19 C4 70 57 55 39 D5 DB FD 89 F9
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDGRls6q8wp/3Bn
SXn+biaHiE2Io78OC0LknrUFe9A4p9pKJWSR39nHj20PomTaKLKjFHDhaczE/dG5
6mz6gjLzRr8FwUJxxafRCcFYkH8rs2PmBMQii1JH3iV5pYtKS1ZNVs86cILIFLvJ
e23PL45w062BWNH+omVy04Q25fVBa9iOqrax70X6mZ5QBe7OKq936/DD5QFg/vVU
CDcaUyIM2GBwZfihUFKKoiR7yNLWm1kL2juXIRyHQvHz9U9BO31e1CGmo2D1Xya4
S4E+j0Fw5zcQqtTq2SBFFNhnfLJxzk+L6ZItckKOhcqBMnC3oFZoH5UXoq4qpMmn
HpBUDzVFAgMBAAECggEAPrSojIUhtBSPgXLo7JD8iXePgcu7lB72oGQa/ul54Dkh
nbp2tHW1o9nV1ZB30dD2MgI4cumpc5GMXv5n5dCceXjh19mXgHDQ0pkBt1c3iOQa
j8KMXRpvwr/cXT0AMNZjrGDsd+B2U/3Op/lrsI8rPr6LTHGBNEU6zVZcpZoTdwUs
cjmKWtZrkuVWnl3YTTqUwqNJSAT4ap+Y+BlM0rTn1CRHE9TAc9XrwPlLAANdhaqR
vMcnYamjbBRcvvFEhN7Rnvhui7k/5tGRP/K4plHisok0ddRv8tZJrr5EGjWPIplh
```

#18. OpenSSL Commands to Convert a Certificate from PKCS#12 to CRT

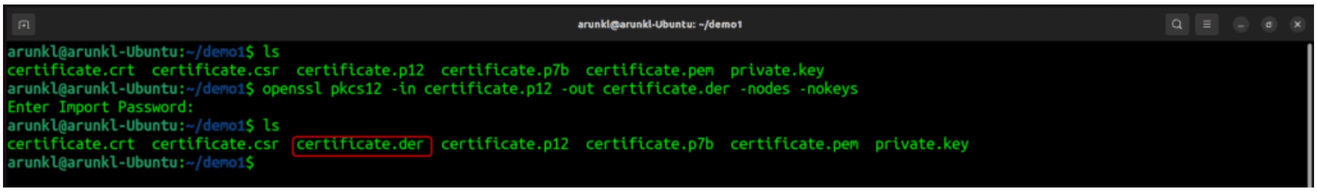
The CRT format and PKCS#12 format are both container formats that are used to store X.509 certificates. The CRT file format is used to store X.509 certificates, While PKCS#12 format is used to store one or more certificates and private key. It’s also encrypted by default.

It’s important to note that a certificate in the PKCS#12 format can be used as it is in a CRT format because it’s the same format, and it just depends on the file extension.

#19. OpenSSL Commands to Convert a Certificate from PKCS#12 to DER

```
openssl pkcs12 -in certificate.p12 -out certificate.der -nodes -nokeys
```

This command will convert the certificate in PKCS#12 format named “certificate.p12” to DER format and save it in a file named “certificate.der” in the current directory. It will prompt you to enter the password that was used to encrypt the certificate in PKCS#12 format.



```
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs12 -in certificate.p12 -out certificate.der -nodes -nokeys
Enter Import Password:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$
```

#20. OpenSSL Commands to Convert a Certificate from PKCS#12 to PKCS7

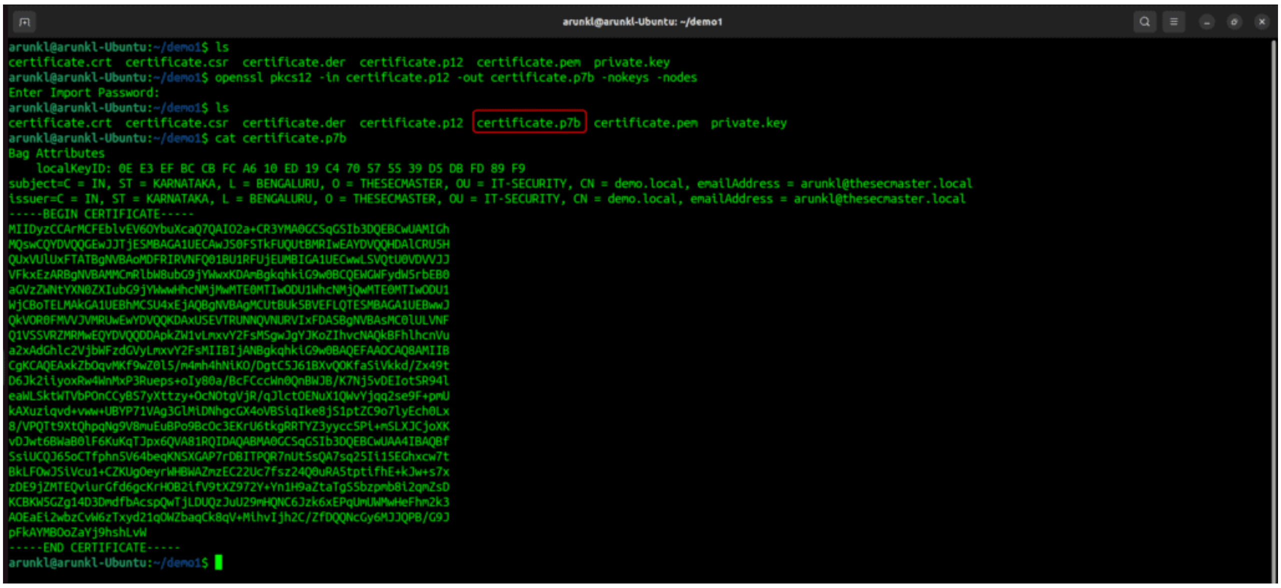
It is not possible to directly convert a certificate from PKCS#12 format to PKCS7 format. Because PKCS7 format is used for storing one or more certificates, typically including the end-user certificate and any intermediate CA certificates, but it doesn’t contain the private key. At the same time, PKCS#12 format is used to store one or more certificates and private key. It’s also encrypted by default.

In order to convert a certificate from PKCS#12 format to PKCS7 format, you will need to first extract the public certificate and any intermediate CA certificates from the PKCS#12 file using OpenSSL, then use the OpenSSL command to create a PKCS7 file.

Here is an example command:

```
openssl pkcs12 -in certificate.p12 -out certificate.p7b -nokeys -nodes
```

This command will convert the certificate in PKCS#12 format named “certificate.p12” to PKCS7 format and save it in a file named “certificate.p7b” in the current directory. It will prompt you to enter the password that was used to encrypt the certificate in PKCS#12 format.



```
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ openssl pkcs12 -in certificate.p12 -out certificate.p7b -nokeys -nodes
Enter Import Password:
arunkl@arunkl-Ubuntu:~/demo1$ ls
certificate.crt  certificate.csr  certificate.der  certificate.p12  certificate.p7b  certificate.pem  private.key
arunkl@arunkl-Ubuntu:~/demo1$ cat certificate.p7b
Bag Attributes
    localKeyID: 0E E3 EF BC CB FC A6 10 ED 19 C4 70 57 55 39 D5 DB FD 89 F9
subject=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
issuer=C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = demo.local, emailAddress = arunkl@thesecmaster.local
-----BEGIN CERTIFICATE-----
MIIDyzCCArMCFEblvEV6OYbuXcaQ7QAIO2a+CR3YMA0GCSqGSIb3DQEBCwUAMIGh
MQswCQYDVQQGEwJJTjESMBAGA1UECAwJS0FSTkFUQUtBMRIwEAYDVQQHDAlCRU5H
QUxVUlUxFTATBgNVBAoMDFRIRVNFQ01BU1RFUjEUMBIGA1UECwwLSVQtU0VDVVJJ
VFkxEzARBgNVBAMMCmRlbW8ubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0
aGVzZWNtYXN0ZXIubG9jYWwwHhcNMjMwMTE0MTIwODU1WhcNMjQwMTE0MTIwODU1
WjCBoTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUtBUk5BVEFLQTESMBAGA1UEBwwJ
QkVOR0FMVVJVMRUwEwYDVQQKDAxUSEVTRUNNQVNURVIxFDASBgNVBAsMC0lULVNF
Q1VSSVRZMRMwEQYDVQQDDApkZW1vLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVu
a2xAdGhlc2VjbWFzdGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxkZbOqvMKf9wZ0l5/m4mh4hNiKO/DgtC5J61BXvQOKfaSiVkkd/Zx49t
D6Jk2iiyoxRw4WnMxP3Rueps+oIy80a/BcFCccWn0QnBWJB/K7Nj5vDEIotSR94l
eaWLSktWTVbPOnCCyBS7yXttzy+OcNOtgVjR/qJlctOENuX1QWvYjqq2se9F+pmU
kAXuziqvd+vww+UBYP71VAg3GlMiDNhgcGX4oVBSiqIke8jS1ptZC9o7lyEch0Lx
8/VPQTt9XtQhpqNg9V8muEuBPo9BcOc3EKrU6tkgRRTYZ3yycc5Pi+mSLXJCjoXK
vDJwt6BWaB0lF6KuKqTJpx6QVA81RQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBf
SsiUCQJ65oCTfphn5V64beqKNSXGAP7rDBITPQR7nUt5sQA7sq25Ii15EGhxcw7t
BkLFOwJSiVcu1+CZKUgOeyrWHBWAZmzEC22Uc7fsz24Q0uRA5tptifhE+kJw+s7x
zDE9jZMTEQviurGfd6gcKrHOB2ifV9tXZ972Y+Yn1H9aZtaTgS5bzpmb8i2qmZsD
KCBKW5GZg14D3DmdfbAcspQwTjLDUQzJuU29mHQNC6Jzk6xEPqUmUWMwHeFhm2k3
AOEaEi2wbzCvW6zTxyd21qOWZbaqCk8qV+MihvIjh2C/ZfDQQNcGy6MJJQPB/G9J
pFkAYMBOoZaYj9hshLvW
-----END CERTIFICATE-----
arunkl@arunkl-Ubuntu:~/demo1$ 
```

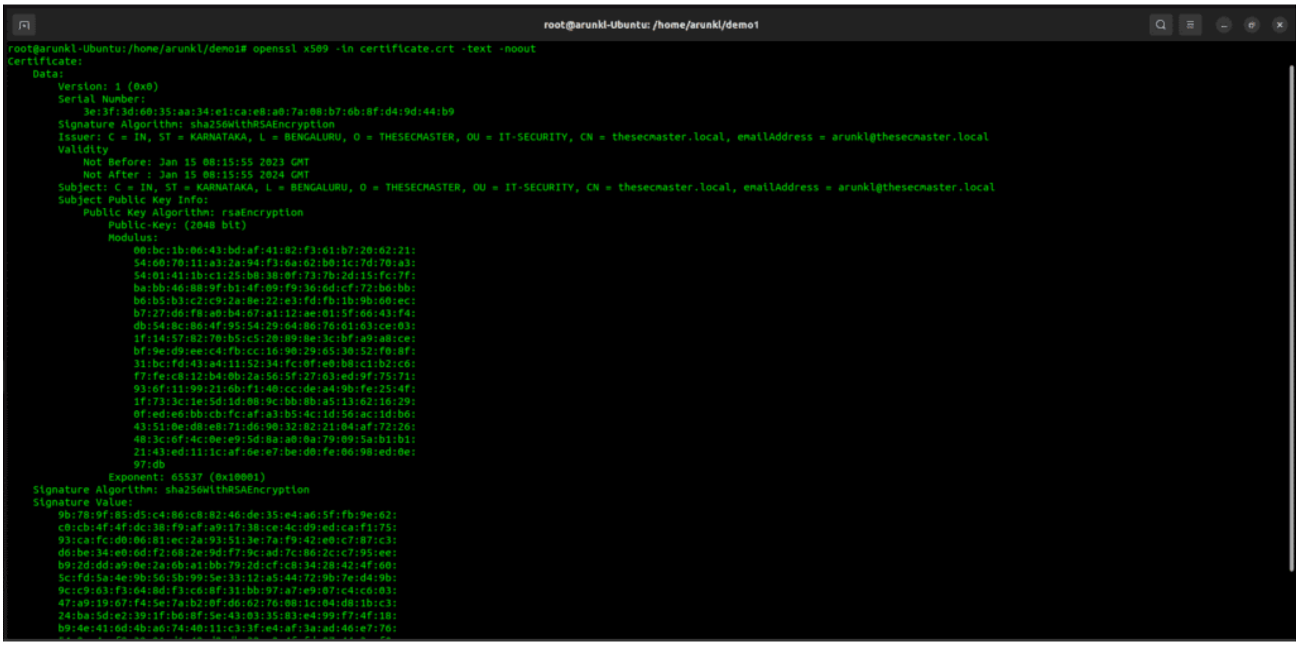
OpenSSL Commands to Debug SSL Certificates and Keys

OpenSSL is a powerful tool that can be used to debug SSL certificates and keys. In this section, we tried showing a few important commands that you can try when you are ended up in some trouble.

#1. OpenSSL Command to Verify the Certificate

```
openssl x509 -in certificate.crt -text -noout
```

This command will display the details of the certificate, including the subject, issuer, and the public key.

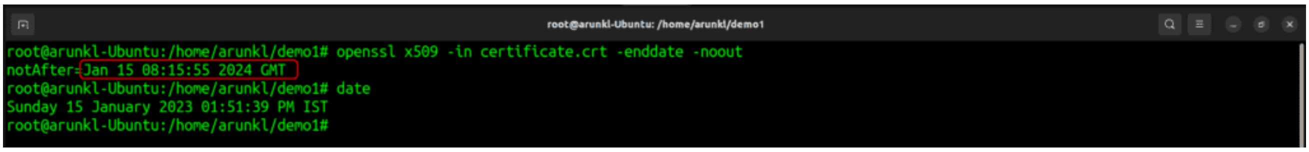


```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -in certificate.crt -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            3e:3f:3d:60:35:aa:34:e1:ca:e8:a0:7a:08:b7:6b:8f:d4:9d:44:b9
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = thesecmaster.local, emailAddress = arunkl@thesecmaster.local
        Validity
            Not Before: Jan 15 08:15:55 2023 GMT
            Not After : Jan 15 08:15:55 2024 GMT
        Subject: C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = thesecmaster.local, emailAddress = arunkl@thesecmaster.local
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:bc:1b:06:43:bd:af:41:82:f3:61:b7:20:62:21:
                    54:60:70:11:a3:2a:94:f3:6a:62:b0:1c:7d:70:a3:
                    54:01:41:1b:c1:25:b8:38:0f:73:7b:2d:15:fc:7f:
                    ba:bb:46:88:9f:b1:4f:09:f9:36:6d:cf:72:b6:bb:
                    b6:b5:b3:c2:c9:2a:8e:22:e3:fd:fb:1b:9b:60:ec:
                    b7:27:d6:f8:a0:b4:67:a1:12:ae:01:5f:66:43:f4:
                    db:54:8c:86:4f:95:54:29:64:86:76:61:63:ce:03:
                    1f:14:57:82:70:b5:c5:20:89:8e:3c:bf:a9:a8:ce:
                    bf:9e:d9:ee:c4:fb:cc:16:90:29:65:30:52:f0:8f:
                    31:bc:fd:43:a4:11:52:34:fc:0f:e0:b8:c1:b2:c6:
                    f7:fe:c8:12:b4:0b:2a:56:5f:27:63:ed:9f:75:71:
                    93:6f:11:99:21:6b:f1:40:cc:de:a4:9b:fe:25:4f:
                    1f:73:3c:1e:5d:1d:08:9c:bb:8b:a5:13:62:16:29:
                    0f:ed:e6:bb:cb:fc:af:a3:b5:4c:1d:56:ac:1d:b6:
                    43:51:0e:d8:e8:71:d6:90:32:82:21:04:af:72:26:
                    48:3c:6f:4c:0e:e9:5d:8a:a0:0a:79:09:5a:b1:b1:
                    21:43:ed:11:1c:af:6e:e7:be:d0:fe:06:98:ed:0e:
                    97:db
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        9b:78:9f:85:d5:c4:86:c8:82:46:de:35:e4:a6:5f:fb:9e:62:
        c0:cb:4f:4f:dc:38:f9:af:a9:17:38:ce:4c:d9:ed:ca:f1:75:
        93:ca:fc:d0:06:81:ec:2a:93:51:3e:7a:f9:42:e0:c7:87:c3:
        d6:be:34:e0:6d:f2:68:2e:9d:f7:9c:ad:7c:86:2c:c7:95:ee:
        b9:2d:dd:a9:0e:2a:6b:a1:bb:79:2d:cf:c8:34:28:42:4f:60:
        5c:fd:5a:4e:9b:56:5b:99:5e:33:12:a5:44:72:9b:7e:d4:9b:
        9c:c9:63:f3:64:8d:f3:c6:8f:31:bb:97:a7:e9:07:c4:c6:03:
        47:a9:19:67:f4:5e:7a:b2:0f:d6:62:76:08:1c:04:d8:1b:c3:
        24:ba:5d:e2:39:1f:b6:8f:5e:43:03:35:83:e4:99:f7:4f:18:
        b9:4e:41:6d:4b:a6:74:40:11:c3:3f:e4:af:3a:ad:46:e7:76:
```

#2. OpenSSL Commands to Check the Expiration date of a Certificate

```
openssl x509 -in certificate.crt -enddate -noout
```

This command will display the expiration date of the certificate.

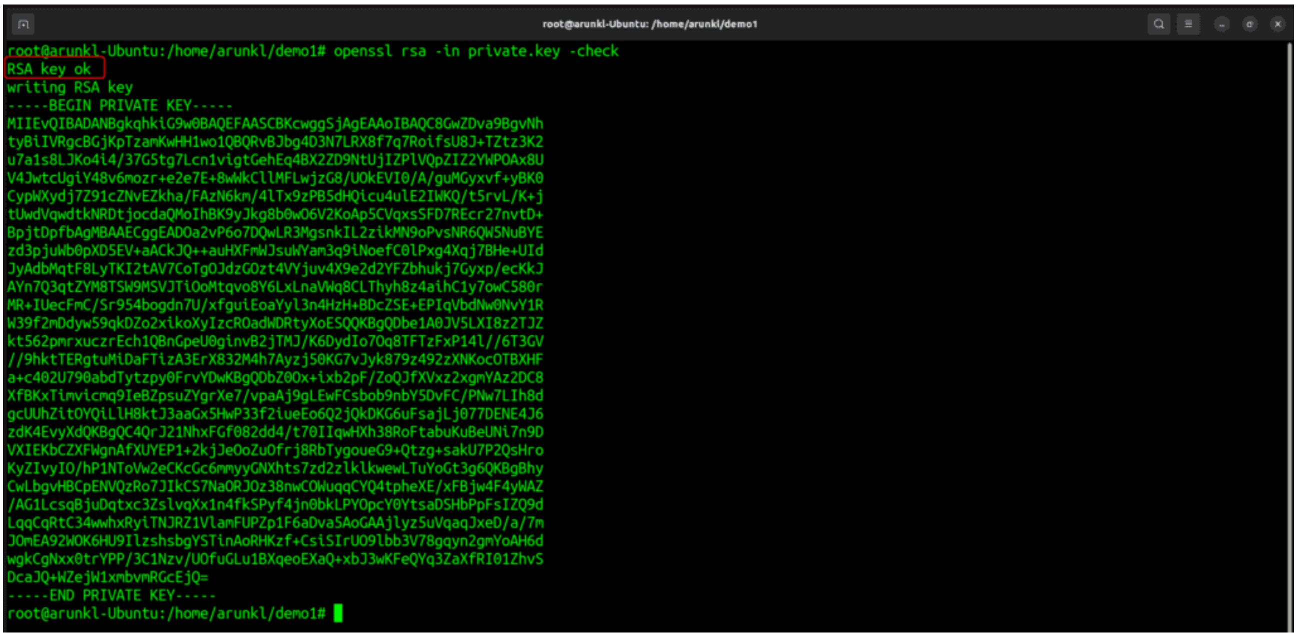


```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -in certificate.crt -enddate -noout
notAfter=Jan 15 08:15:55 2024 GMT
root@arunkl-Ubuntu:/home/arunkl/demo1# date
Sunday 15 January 2023 01:51:39 PM IST
root@arunkl-Ubuntu:/home/arunkl/demo1#
```

#3. OpenSSL Command to Check a Private key

```
openssl rsa -in private.key -check
```

This command will check the private key and display any errors or warnings.

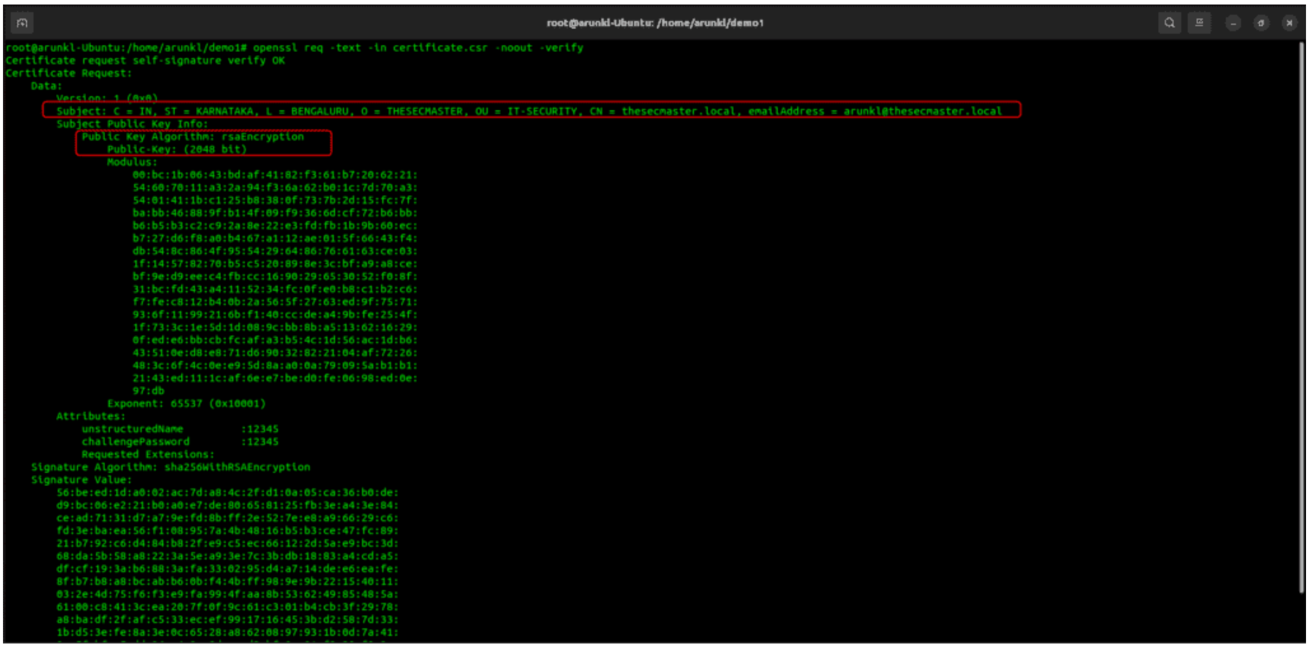


```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl rsa -in private.key -check
RSA key ok
writing RSA key
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC8GwZDva9BgvNh
tyBiIVRgcBGjKpTzamKwHH1wo1QBQRvBJbg4D3N7LRX8f7q7RoifsU8J+TZtz3K2
u7a1s8LJKo4i4/37G5tg7Lcn1vigtGehEq4BX2ZD9NtUjIZPlVQpZIZ2YWPOAx8U
V4JwtcUgiY48v6mozr+e2e7E+8wWkCllMFLwjzG8/UOkEVI0/A/guMGyxvf+yBK0
CypWXydj7Z91cZNvEZkha/FAzN6km/4lTx9zPB5dHQicu4ulE2IWKQ/t5rvL/K+j
tUwdVqwdtkNRDtjocdaQMoIhBK9yJkg8b0wO6V2KoAp5CVqxsSFD7REcr27nvtD+
BpjtDpfbAgMBAAECggEADOa2vP6o7DQwLR3MgsnkIL2zikMN9oPvsNR6QW5NuBYE
zd3pjuWb0pXD5EV+aACkJQ++auHXFmWJsuWYam3q9iNoefC0lPxg4Xqj7BHe+UId
JyAdbMqtF8LyTKI2tAV7CoTgOJdzGOzt4VYjuv4X9e2d2YFZbhukj7Gyxp/ecKkJ
AYn7Q3qtZYM8TSW9MSVJTiOoMtqvo8Y6LxLnaVWq8CLThyh8z4aihC1y7owC580r
MR+IUecFmC/Sr954bogdn7U/xfguiEoaYyl3n4HzH+BDcZSE+EPIqVbdNw0NvY1R
W39f2mDdyw59qkDZo2xikoXyIzcROadWDRtyXoESQQKBgQDbe1A0JV5LXI8z2TJZ
kt562pmrxuczrEch1QBnGpeU0ginvB2jTMJ/K6DydIo7Oq8TFTzFxP14l//6T3GV
//9hktTERgtuMiDaFTizA3ErX832M4h7Ayzj50KG7vJyk879z492zXNKocOTBXHF
a+c402U790abdTytzpy0FrvYDwKBgQDbZ0Ox+ixb2pF/ZoQJfXVxz2xgmYAz2DC8
XfBKxTimvicmq9IeBZpsuZYgrXe7/vpaAj9gLEwFCsbob9nbY5DvFC/PNw7LIh8d
gcUUhZitOYQiLlH8ktJ3aaGx5HwP33f2iueEo6Q2jQkDKG6uFsajLj077DENE4J6
zdK4EvyXdQKBgQC4QrJ21NhxFGf082dd4/t70IIqwHXh38RoFtabuKuBeUNi7n9D
VXIEKbCZXFWgnAfXUYEP1+2kjJeOoZuOfrj8RbTygoueG9+Qtzg+sakU7P2QsHro
KyZIvyIO/hP1NToVw2eCKcGc6mmyyGNXhts7zd2zlklkwewLTuYoGt3g6QKBgBhy
CwLbgvHBCpENVQzRo7JIkCS7NaORJOz38nwCOWuqqCYQ4tpheXE/xFBjw4F4yWAZ
/AG1LcsqBjuDqtxc3ZslvqXx1n4fkSPyf4jn0bkLPYOpcY0YtsaDSHbPpFsIZQ9d
LqqCqRtC34wwhxRyiTNJRZ1VlamFUPZp1F6aDva5AoGAAjlyz5uVqaqJxeD/a/7m
JOmEA92WOK6HU9IlzshsbgYSTinAoRHKzf+CsiSIrUO9lbb3V78gqyn2gmYoAH6d
wgkCgNxx0trYPP/3C1Nzv/UOfuGLu1BXqeoEXaQ+xbJ3wKFeQYq3ZaXfRI01ZhvS
DcaJQ+WZejW1xmbvmRGcEjQ=
-----END PRIVATE KEY-----
root@arunkl-Ubuntu:/home/arunkl/demo1#
```

#4. OpenSSL Command to Verify the Certificate Signing Request (CSR)

```
openssl req -text -in certificate.csr -noout -verify
```

This command will display the details of the certificate signing request (CSR), including the subject, issuer, and public key.



```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl req -text -in certificate.csr -noout -verify
Certificate request self-signature verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = IN, ST = KARNATAKA, L = BENGALURU, O = THESECMASTER, OU = IT-SECURITY, CN = thesecmaster.local, emailAddress = arunkl@thesecmaster.local
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:bc:1b:06:43:bd:af:41:82:f3:61:b7:20:62:21:
                    54:60:70:11:a3:2a:94:f3:6a:62:b0:1c:7d:70:a3:
                    54:01:41:1b:c1:25:b8:38:0f:73:7b:2d:15:fc:7f:
                    ba:bb:46:88:9f:b1:4f:09:f9:36:6d:cf:72:b6:bb:
                    b6:b5:b3:c2:c9:2a:8e:22:e3:fd:fb:1b:9b:60:ec:
                    b7:27:d6:f8:a0:b4:67:a1:12:ae:01:5f:66:43:f4:
                    db:54:8c:86:4f:95:54:29:64:86:76:61:63:ce:03:
                    1f:14:57:82:70:b5:c5:20:89:8e:3c:bf:a9:a8:ce:
                    bf:9e:d9:ee:c4:fb:cc:16:90:29:65:30:52:f0:8f:
                    31:bc:fd:43:a4:11:52:34:fc:0f:e0:b8:c1:b2:c6:
                    f7:fe:c8:12:b4:0b:2a:56:5f:27:63:ed:9f:75:71:
                    93:6f:11:99:21:6b:f1:40:cc:de:a4:9b:fe:25:4f:
                    1f:73:3c:1e:5d:1d:08:9c:bb:8b:a5:13:62:16:29:
                    0f:ed:e6:bb:cb:fc:af:a3:b5:4c:1d:56:ac:1d:b6:
                    43:51:0e:d8:e8:71:d6:90:32:82:21:04:af:72:26:
                    48:3c:6f:4c:0e:e9:5d:8a:a0:0a:79:09:5a:b1:b1:
                    21:43:ed:11:1c:af:6e:e7:be:d0:fe:06:98:ed:0e:
                    97:db
                Exponent: 65537 (0x10001)
        Attributes:
            unstructuredName         :12345
            challengePassword        :12345
            Requested Extensions:
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        56:be:ed:1d:a0:02:ac:7d:a8:4c:2f:d1:0a:05:ca:36:b0:de:
        d9:bc:06:e2:21:b0:a0:e7:de:80:65:81:25:fb:3e:a4:3e:84:
        ce:ad:71:31:d7:a7:9e:fd:8b:ff:2e:52:7e:e8:a9:66:29:c6:
        fd:3e:ba:ea:56:f1:08:95:7a:4b:48:16:b5:b3:ce:47:fc:89:
        21:b7:92:c6:d4:84:b8:2f:e9:c5:ec:66:12:2d:5a:e9:bc:3d:
        68:da:5b:58:a8:22:3a:5e:a9:3e:7c:3b:db:18:83:a4:cd:a5:
        df:cf:19:3a:b6:88:3a:fa:33:02:95:d4:a7:14:de:e6:ea:fe:
        8f:b7:b8:a8:bc:ab:b6:0b:f4:4b:ff:98:9e:9b:22:15:40:11:
        03:2e:4d:75:f6:f3:e9:fa:99:4f:aa:8b:53:62:49:85:48:5a:
        61:00:c8:41:3c:ea:20:7f:0f:9c:61:c3:01:b4:cb:3f:29:78:
        a8:ba:df:2f:af:c5:33:ec:ef:99:17:16:45:3b:d2:58:7d:33:
        1b:d5:3e:fe:8a:3e:0c:65:28:a8:62:08:97:93:1b:0d:7a:41:
```

#5. OpenSSL Commands to Check the Hash Value of A Certificate

OpenSSL can be used to calculate the hash value of an X.509 certificate. A hash value is a unique value that is calculated based on the content of the certificate. It can be used to check the integrity of the certificate and to verify that it has not been tampered with. Here are some common OpenSSL commands that can be used to check the hash value of a certificate:

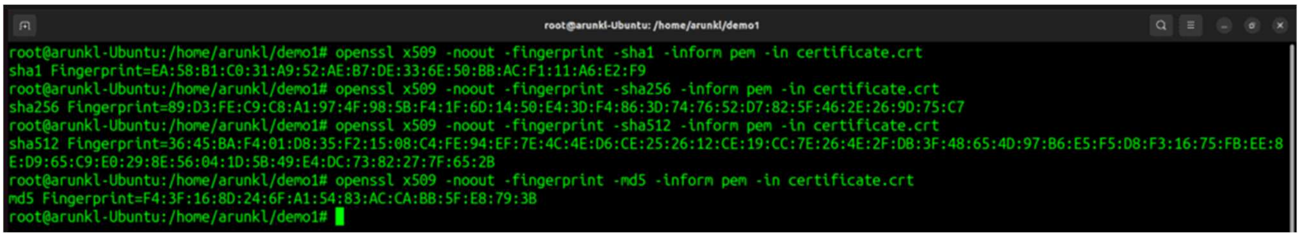
```
openssl x509 -noout -fingerprint -md5 -inform pem -in certificate.crt
openssl x509 -noout -fingerprint -sha1 -inform pem -in certificate.crt


openssl x509 -noout -fingerprint -sha256 -inform pem -in certificate.crt


openssl x509 -noout -fingerprint -sha512 -inform pem -in certificate.crt
```

Note: It’s important to note that the above command will work only if the certificate is in PEM format. If you have a certificate in DER format, you need to use the -inform DER option.

```
openssl x509 -noout -fingerprint -sha1 -inform der -in certificate.crt
```



```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -noout -fingerprint -sha1 -inform pem -in certificate.crt
sha1 Fingerprint=EA:58:B1:C0:31:A9:52:AE:B7:DE:33:6E:50:BB:AC:F1:11:A6:E2:F9
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -noout -fingerprint -sha256 -inform pem -in certificate.crt
sha256 Fingerprint=89:D3:FE:C9:C8:A1:97:4F:98:5B:F4:1F:6D:14:50:E4:3D:F4:86:3D:74:76:52:D7:82:5F:46:2E:26:9D:75:C7
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -noout -fingerprint -sha512 -inform pem -in certificate.crt
sha512 Fingerprint=36:45:BA:F4:01:D8:35:F2:15:08:C4:FE:94:EF:7E:4C:4E:D6:CE:25:26:12:CE:19:CC:7E:26:4E:2F:DB:3F:48:65:4D:97:B6:E5:F5:D8:F3:16:75:FB:EE:8
E:D9:65:C9:E0:29:8E:56:04:1D:5B:49:E4:DC:73:82:27:7F:65:2B
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -noout -fingerprint -md5 -inform pem -in certificate.crt
md5 Fingerprint=F4:3F:16:8D:24:6F:A1:54:83:AC:CA:BB:5F:E8:79:3B
root@arunkl-Ubuntu:/home/arunkl/demo1# 
```

#6. OpenSSL Command to Verify the SSL/TLS version Accepted by a Site

```
openssl s_client -connect host:port
```

This command will initiate an SSL connection to the specified host and port and display the details of the SSL connection, including the certificate chain and the cipher suite.

```
openssl s_client -connect host:port -ssl2
```

This command will initiate an SSL connection to the specified host and port using SSL2 and display the details of the SSL connection.

```
openssl s_client -connect host:port -ssl2
```

This command will initiate an SSL connection to the specified host and port using SSL3 and display the details of the SSL connection.

```
openssl s_client -connect host:port -tls1_1
```

This command will initiate an SSL connection to the specified host and port using TLS 1.1 and display the details of the SSL connection.

```
openssl s_client -connect host:port -tls1_2
```

This command will initiate an SSL connection to the specified host and port using TLS 1.2 and display the details of the SSL connection.

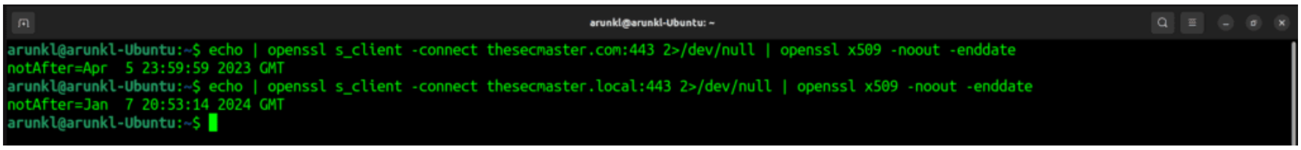
```
openssl s_client -connect host:port -tls1_3
```

This command will initiate an SSL connection to the specified host and port using TLS 1.3 and display the details of the SSL connection.

#7. OpenSSL Commands to Check the Certificate Expiry of a Site

```
echo | openssl s_client -connect example.com:443 2>/dev/null | openssl x509 -noout -enddate
```

Where “example.com” is the URL you want to check the SSL certificate for and “443” is the port number for HTTPS connections.
This command uses the `openssl s_client` command to initiate an SSL connection to the specified URL and port and the `openssl x509` command to extract the expiration date of the certificate.



```
arunkl@arunkl-Ubuntu:~$ echo | openssl s_client -connect thesecmaster.com:443 2>/dev/null | openssl x509 -noout -enddate
notAfter=Apr  5 23:59:59 2023 GMT
arunkl@arunkl-Ubuntu:~$ echo | openssl s_client -connect thesecmaster.local:443 2>/dev/null | openssl x509 -noout -enddate
notAfter=Jan  7 20:53:14 2024 GMT
arunkl@arunkl-Ubuntu:~$
```

#8. OpenSSL Commands to Check A Particular Cipher is Accepted by a Site

```
openssl s_client -connect example.com:443 -cipher ECDHE-RSA-AES256-GCM-SHA384
```

Where “example.com” is the URL you want to check the SSL certificate for, “443” is the port number for HTTPS connections, and “ECDHE-RSA-AES256-GCM-SHA384” is the cipher you want to check for.

#9. OpenSSL Commands to List all the Cipher Supported by a Server

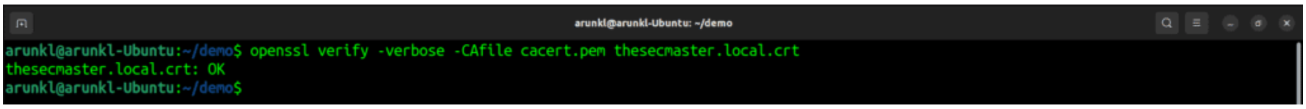
```
openssl ciphers -v 'ALL:COMPLEMENTOFALL' | grep -i -o -w -E 'TLS_.*'
```

It’s important to note that this command will only check the ciphers that are supported by the server and not the client.

#10. OpenSSL Command to Verify the Certificate Chain

```
openssl verify -verbose -CAfile ca-bundle.crt certificate.crt
```

Where “ca-bundle.crt” is the file that contains the root and intermediate CA certificates, and “certificate.crt” is the end-entity certificate file. This command uses the `openssl verify` command to verify the certificate chain, using the CA certificates from the file “ca-bundle.crt”. It will display the result of the verification process and indicate whether the certificate chain is valid or not.



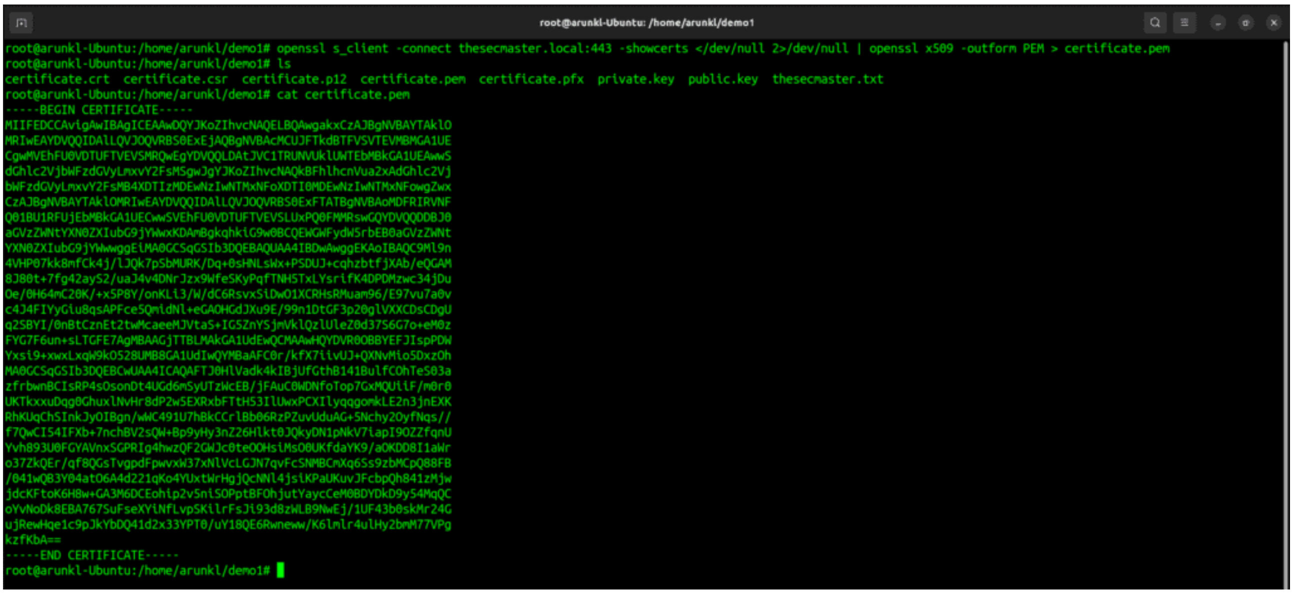
```
arunkl@arunkl-Ubuntu:~/demo$ openssl verify -verbose -CAfile cacert.pem thesecmaster.local.crt
thesecmaster.local.crt: OK
arunkl@arunkl-Ubuntu:~/demo$
```

#11. OpenSSL Commands to Retrieve or Download the Certificate of a Site

```
openssl s_client -connect example.com:443 -showcerts </dev/null 2>/dev/null | openssl x509
-outform PEM > certificate.pem
```

This command uses the `openssl s_client` command to initiate an SSL connection to the specified URL and port and the `–showcerts` option to display the entire certificate chain.

The `openssl x509` command is used to convert the certificate from DER format to PEM format and the > symbol is used to redirect the output to a file named "certificate.pem"



```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl s_client -connect thesecmaster.local:443 -showcerts </dev/null 2>/dev/null | openssl x509 -outform PEM > certificate.pem
root@arunkl-Ubuntu:/home/arunkl/demo1# ls
certificate.crt  certificate.csr  certificate.p12  certificate.pem  certificate.pfx  private.key  public.key  thesecmaster.txt
root@arunkl-Ubuntu:/home/arunkl/demo1# cat certificate.pem
-----BEGIN CERTIFICATE-----
MIIFEDCCAvigAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwgakxCzAJBgNVBAYTAklO
MRIwEAYDVQQIDAlLQVJOQVRBS0ExEjAQBgNVBAcMCUJFTkdBTFVSVTEVMBMGA1UE
CgwMVEhFU0VDTUFTVEVSMRQwEgYDVQQLDAtJVC1TRUNVUklUWTEbMBkGA1UEAwwS
dGhlc2VjbWFzdGVyLmxvY2FsMSgwJgYJKoZIhvcNAQkBFhlhcnVua2xAdGhlc2Vj
bWFzdGVyLmxvY2FsMB4XDTIzMDEwNzIwNTMxNFoXDTI0MDEwNzIwNTMxNFowgZwx
CzAJBgNVBAYTAklOMRIwEAYDVQQIDAlLQVJOQVRBS0ExFTATBgNVBAoMDFRIRVNF
Q01BU1RFUjEbMBkGA1UECwwSVEhFU0VDTUFTVEVSLUxPQ0FMMRswGQYDVQQDDBJ0
aGVzZWNtYXN0ZXIubG9jYWwxKDAmBgkqhkiG9w0BCQEWGWFydW5rbEB0aGVzZWNt
YXN0ZXIubG9jYWwwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC9Ml9n
4VHP07kk8mfCk4j/lJQk7pSbMURK/Dq+0sHNLsWx+PSDUJ+cqhzbtfjXAb/eQGAM
8J80t+7fg42ayS2/uaJ4v4DNrJzx9WfeSKyPqfTNH5TxLYsrifK4DPDMzwc34jDu
Oe/0H64mC20K/+x5P8Y/onKLi3/W/dC6RsvxSiDwO1XCRHsRMuam96/E97vu7a0v
c4J4FIYyGiu8qsAPFce5QmidNl+eGAOHGdJXu9E/99n1DtGF3p20glVXXCDsCDgU
q2SBYI/0nBtCznEt2twMcaeeMJVtaS+IG5ZnYSjmVklQzlUleZ0d37S6G7o+eM0z
FYG7F6un+sLTGFE7AgMBAAGjTTBLMAkGA1UdEwQCMAAwHQYDVR0OBBYEFJIspPDW
Yxsi9+xwxLxqW9kO528UMB8GA1UdIwQYMBaAFC0r/kfX7iivUJ+QXNvMio5DxzOh
MA0GCSqGSIb3DQEBCwUAA4ICAQAFTJ0HlVadk4kIBjUfGthB141BulfCOhTeS03a
zfrbwnBCIsRP4sOsonDt4UGd6mSyUTzWcEB/jFAuC0WDNfoTop7GxMQUiiF/m0r0
UKTkxxuDqg0GhuxlNvHr8dP2w5EXRxbFTtH53IlUwxPCXIlyqqgomkLE2n3jnEXK
RhKUqChSInkJyOIBgn/wWC491U7hBkCCrlBb06RzPZuvUduAG+5Nchy2OyfNqs//
f7QwCI54IFXb+7nchBV2sQW+Bp9yHy3nZ26Hlkt0JQkyDN1pNkV7iapI9OZZfqnU
Yvh893U0FGYAVnxSGPRIg4hwzQF2GWJc0teOOHsiMsO0UKfdaYK9/aOKDD8I1aWr
o37ZkQEr/qf8QGsTvgpdFpwvxW37xNlVcLGJN7qvFcSNMBCmXq6Ss9zbMCpQ88FB
/041wQB3Y04atO6A4d221qKo4YUxtWrHgjQcNNl4jsiKPaUKuvJFcbpQh841zMjw
jdcKFtoK6H8w+GA3M6DCEohip2v5niSOPptBFOhjutYaycCeM0BDYDkD9y54MqQC
oYvNoDk8EBA767SuFseXYiNfLvpSKilrFsJi93d8zWLB9NwEj/1UF43b0skMr24G
ujRewHqe1c9pJkYbDQ41d2x33YPT0/uY18QE6Rwneww/K6lmlr4ulHy2bmM77VPg
kzfKbA==
-----END CERTIFICATE-----
root@arunkl-Ubuntu:/home/arunkl/demo1#
```

#12. OpenSSL Commands to Verify the Same Public Key File in Key Pair, CSR, and Certificate

It is important to verify that the public key and the certificate match to ensure the security of the SSL/TLS connection. OpenSSL can be used to verify that the public key and the certificate match.

```
openssl pkey -pubout -in private.key | openssl sha256
```

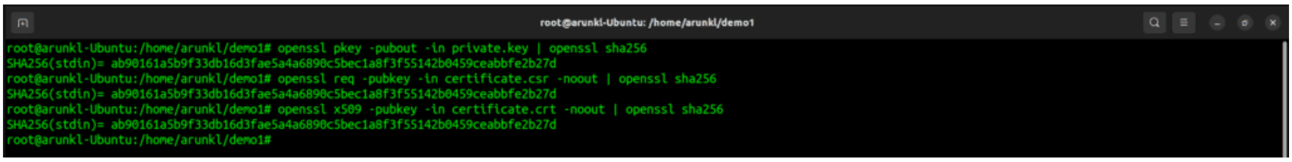
This command extracts the public key from the 'private.key' key pair and generates the sha256 hash of the public key.

```
openssl req -pubkey -in certificate.csr -noout | openssl sha256
```

This command extracts the public key from the 'certificate.csr' CSR and generates the sha256 hash of the public key.

```
openssl x509 -pubkey -in certificate.crt -noout | openssl sha256
```

This command extracts the public key from the 'certificate.crt' certificate and generates the sha256 hash of the public key.



```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl pkey -pubout -in private.key | openssl sha256
SHA256(stdin)= ab90161a5b9f33db16d3fae5a4a6890c5bec1a8f3f55142b0459ceabbfe2b27d
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl req -pubkey -in certificate.csr -noout | openssl sha256
SHA256(stdin)= ab90161a5b9f33db16d3fae5a4a6890c5bec1a8f3f55142b0459ceabbfe2b27d
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -pubkey -in certificate.crt -noout | openssl sha256
SHA256(stdin)= ab90161a5b9f33db16d3fae5a4a6890c5bec1a8f3f55142b0459ceabbfe2b27d
root@arunkl-Ubuntu:/home/arunkl/demo1#
```

#13. OpenSSL Commands to Verify the Same Private Key File in Key Pair, CSR, and Certificate

It is important to verify that the private key and the certificate match to ensure the security of the SSL/TLS connection. OpenSSL can be used to verify that the private key and the certificate match.

```
openssl rsa -noout -modulus -in private.key | openssl sha256
```

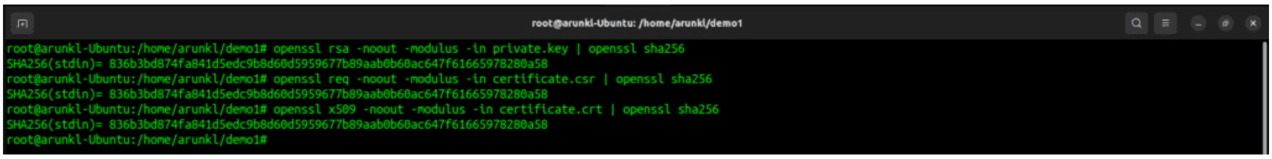
This command extracts the private key from the ‘private.key’ key pair and generates the sha256 hash of the private key.

```
openssl req -noout -modulus -in certificate.csr | openssl sha256
```

This command extracts the private key from the ‘certificate.csr’ CSR and generates the sha256 hash of the private key.

```
openssl x509 -noout -modulus -in certificate.crt | openssl sha256
```

This command extracts the private key from the ‘certificate.crt’ certificate and generates the sha256 hash of the private key.



```
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl rsa -noout -modulus -in private.key | openssl sha256
SHA256(stdin)= 836b3bd874fa841d5edc9b8d60d5959677b89aab0b60ac647f61665978280a58
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl req -noout -modulus -in certificate.csr | openssl sha256
SHA256(stdin)= 836b3bd874fa841d5edc9b8d60d5959677b89aab0b60ac647f61665978280a58
root@arunkl-Ubuntu:/home/arunkl/demo1# openssl x509 -noout -modulus -in certificate.crt | openssl sha256
SHA256(stdin)= 836b3bd874fa841d5edc9b8d60d5959677b89aab0b60ac647f61665978280a58
root@arunkl-Ubuntu:/home/arunkl/demo1#
```

What is the difference between SSL and OpenSSL?

SSL is a protocol that was widely used to establish secure connections over the internet until it was replaced by its successor, TLS (Transport Layer Security). SSL provides a secure communication channel between a client and a server by encrypting the data that is transmitted. It is designed to ensure the confidentiality, integrity, and authenticity of data exchanged over the internet.

OpenSSL, on the other hand, is an open-source implementation of the SSL and TLS protocols. It provides a collection of libraries and tools that can be used to secure internet communication, protect sensitive data, and perform other cryptographic operations. OpenSSL is designed to be a versatile tool that can be used in many different scenarios and environments.

In summary, SSL is a protocol used to establish secure connections over the internet, while OpenSSL is an open-source implementation of the SSL and TLS protocols that provides a collection of libraries and tools for cryptographic operations.

Is OpenSSL free to use?

Yes, OpenSSL is free and open-source software. It can be downloaded and used without any costs. It is available under a dual-license model, consisting of the OpenSSL License and the SSLeay License. For the OpenSSL 3.0 release and later releases derived from that, the Apache License v2 applies. For any release made before OpenSSL 3.0 (namely the 1.1.1, 1.1.0, 1.0.2, and all prior releases, including those not currently supported), the dual OpenSSL and SSLeay license applies. It is important to consult the OpenSSL license and legal advice before using it in a commercial or production environment.

Is OpenSSL available for Windows?

OpenSSL has no pre-compelled OpenSSL libraries for any platform. It has released only binary source code. It’s the Linux distributions that released a pre-compelled version for their distributions. However, Some third-party services have offered to provide pre-compelled OpenSSL libraries for Win32/64. One such service offering a pre-compelled OpenSSL library for Microsoft Windows is slproweb.com/products/Win32OpenSSL.html.

We hope this post would help you know what is OpenSSL, what OpenSSL is used for, how to generate a sedl-singned certificate, how to convert an SSL certificate from one to another format, troubleshooting, debugging, and other most useful OpenSSL commands to work with SSL certificates. Please share this post if you find this interested. Visit our social media page on Facebook, LinkedIn, Twitter, Telegram, Tumblr, & Medium and subscribe to receive updates like this.