

mod17_ex01: Creating a CA

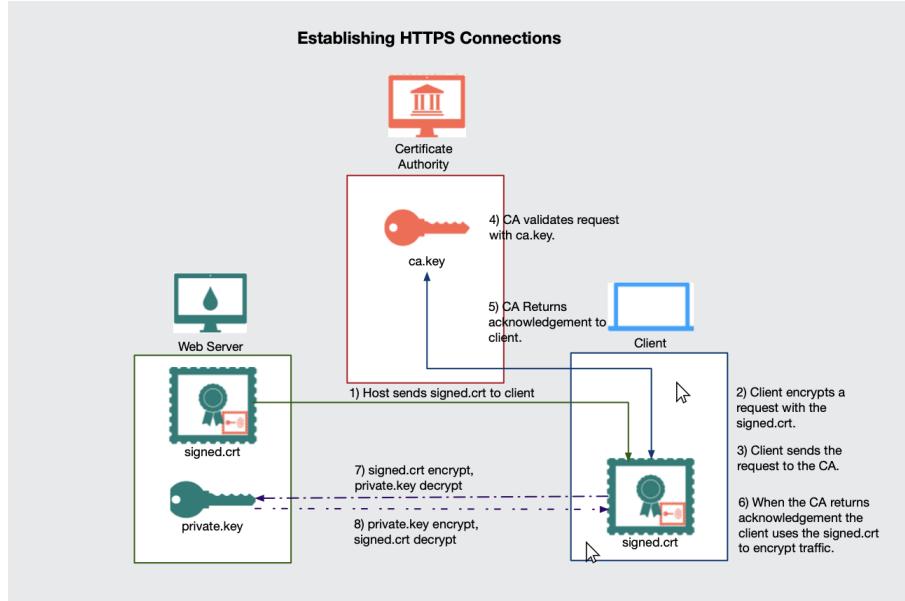
The purpose of this exercise is to create a simulated Certificate Authority and to sign a certificate signing request. This exercise presents the Transport Security Layer and introduces administrative skills for the `openssl` command.

Reference Information

The following documents provide information related to this exercise.

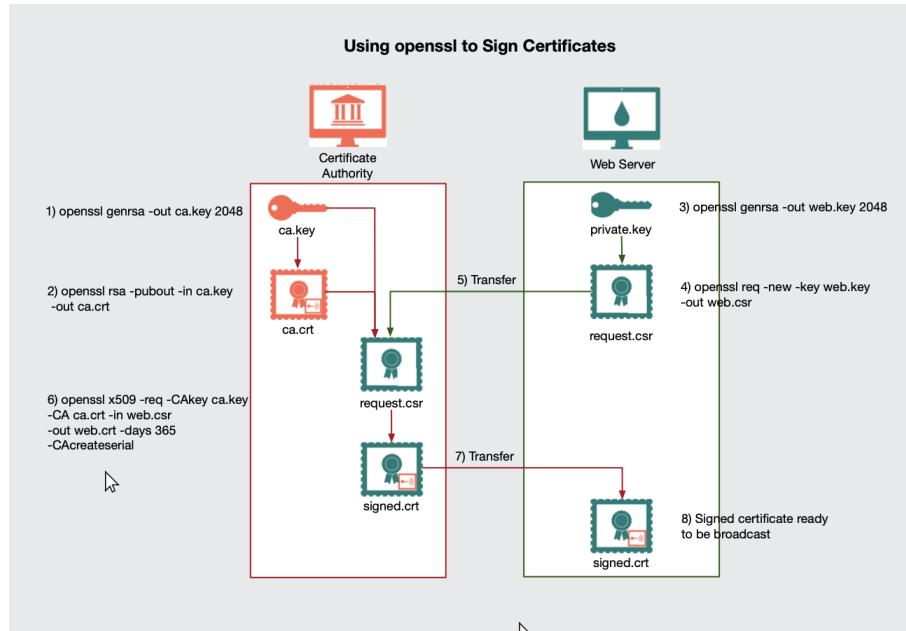
- TBD

1. Transport Layer Security (TLS) is used to establish HTTPS connectivity.



1. The web server sends a signed certificate to the client. The signed certificate is signed by a trusted Certificate Authority (CA).
2. The client reads the contact information for the CA from the signed certificate and uses this information to contact the CA.
3. The client encrypts a request using the signed certificate and sends this request to the CA.
4. The CA will use its private key to decrypt this request.
5. If the request decrypts the CA sends back acknowledgement to the client.
6. The client will now establish secure network communications with the web server.
7. The client uses the signed certificate to encrypt traffic and the web server uses the private key to decrypt.
8. In reverse, the web server will use the private key to encrypt and the client will use the signed certificate to decrypt.

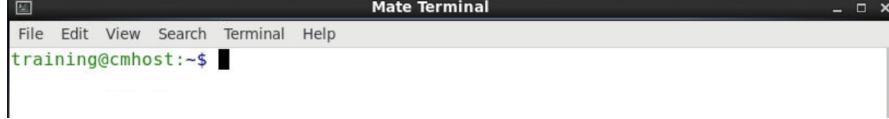
2. The openssl command manages TLS certificates.



The openssl command is used to generate private keys (.key), certificate signing requests (.csr), and to sign certificate signing requests (.csr) creating a signed public certificates (.crt).

1. The openssl command generates a private key (.key) for the CA.
2. The openssl command uses the private key to create a public key (.crt) for the CA.
3. The openssl command creates a private key (.key) for the web server.
4. The openssl command uses the web servers private key to create a signing request (.csr).
5. The signing request (.csr) is transferred to the CA.
6. The openssl command uses the CA private key (.key) and the CA public key (.crt) to sign the request (.csr).
7. The signed certificate (.crt) is transferred to the webserver.
8. The signed certificate (.crt) is ready to broadcast to any client.

3. Open a Mate terminal as the user training.



4. Create a keypair for the CA.

A key pair is an unpredictable (meaning large and random) number is used to generate an acceptable pair of keys suitable for use by an asymmetric key algorithm. The private key has the bulk of the encryption algorithm, typically containing either 1024 or 2048 characters of the encryption algorithm. The private key is identified with a file extension of .key. The public key contains fewer characters. The public key commonly has an extension of .crt. The two keys are referred to as a key pair.

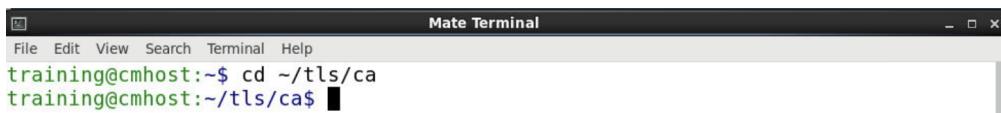
4.1 Open a Mate terminal. Create four directories - ca, web, local, and client.



```
Mate Terminal
File Edit View Search Terminal Help
training@cmhost:~$ rm -r -f tls
training@cmhost:~$ mkdir -p ~/tls/ca ~/tls/client ~/tls/local ~/tls/web
training@cmhost:~$ ls ~/tls
ca client local web
training@cmhost:~$
```

```
$ mkdir -p ~/tls/ca ~/tls/client ~/tls/local ~/tls/web
$ ls ~/tls
```

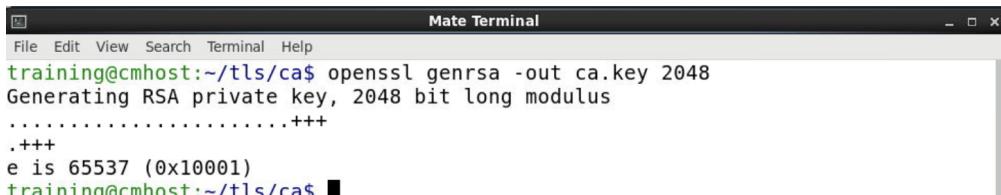
4.2 Change directory to ca.



```
Mate Terminal
File Edit View Search Terminal Help
training@cmhost:~$ cd ~/tls/ca
training@cmhost:~/tls/ca$
```

```
$ cd ~/tls/ca
```

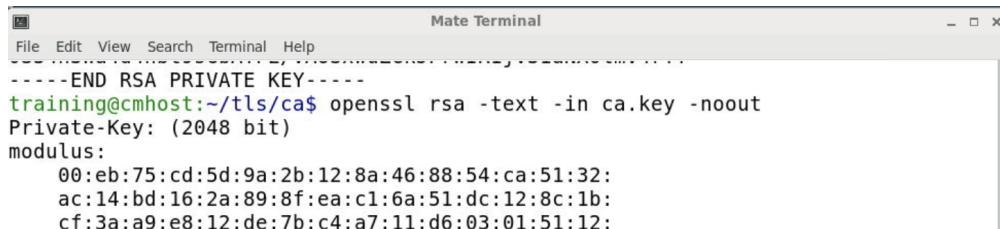
4.3 Use the openssl command to create a private key for the CA.



```
Mate Terminal
File Edit View Search Terminal Help
training@cmhost:~/tls/ca$ openssl genrsa -out ca.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.e++
e is 65537 (0x10001)
training@cmhost:~/tls/ca$
```

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

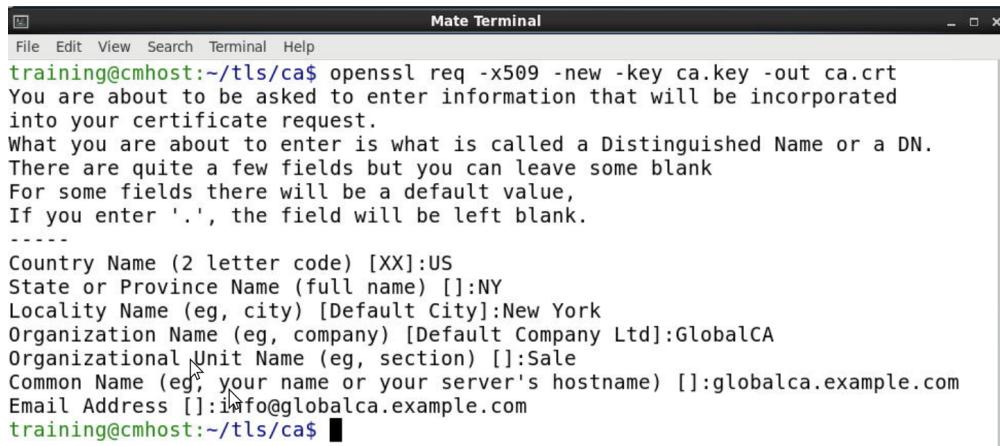
4.4 Verify the private key. The output from both commands must match.



```
Mate Terminal
File Edit View Search Terminal Help
-----END RSA PRIVATE KEY-----
training@cmhost:~/tls/ca$ openssl rsa -text -in ca.key -noout
Private-Key: (2048 bit)
modulus:
00:eb:75:cd:5d:9a:2b:12:8a:46:88:54:ca:51:32:
ac:14:bd:16:2a:89:8f:ea:c1:6a:51:dc:12:8c:1b:
cf:3a:a9:e8:12:de:7b:c4:a7:11:d6:03:01:51:12:
```

```
$ cat ca.key
$ openssl rsa -check -in ca.key
$ openssl rsa -text -in ca.key -noout
```

4.5 Use the CA's private key to create the CA's public certificate. Use the provided information. This information will be encoded into ca.crt.



```
Mate Terminal
File Edit View Search Terminal Help
training@cmhost:~/tls/ca$ openssl req -x509 -new -key ca.key -out ca.crt
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) [XX]:US
State or Province Name (full name) []:NY
Locality Name (eg, city) [Default City]:New York
Organization Name (eg, company) [Default Company Ltd]:GlobalCA
Organizational Unit Name (eg, section) []:Sale
Common Name (eg, your name or your server's hostname) []:globalca.example.com
Email Address []:info@globalca.example.com
training@cmhost:~/tls/ca$
```

```
$ openssl req -x509 -new -key ca.key -out ca.crt
```

4.6 Verify the ca.crt certificate.

```
training@cmhost:~/tls/ca$ openssl x509 -text -in ca.crt -noout
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number:
        98:1a:0f:22:cb:e1:ca:de
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=NY, L=New York, O=GlobalCA, OU=Sale, CN=globalca.example.com/emailAddress=info@globalca.example.com
    Subject: C=US, ST=NY, L=New York, O=GlobalCA, OU=Sale, CN=globalca.example.com/emailAddress=info@globalca.example.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
                Modulus:
                    00:eb:75:cd:5d:9a:2b:12:8a:46:88:54:ca:51:32:
                        ac:14:bd:16:2a:89:8f:ea:c1:6a:51:dc:12:8c:1b:
```

```
$ openssl x509 -text -in ca.crt -noout
```

5. Create a private key for the web server.

5.1 Change directory to web server.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/ca$ cd ~/tls/web
training@cmhost:~/tls/web$
```

```
$ cd ~/tls/web
```

5.2 Create a private key for the web server.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ openssl genrsa -out web.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
training@cmhost:~/tls/web$
```

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

6. Create a Certificate Signing Request (.csr).

Note. There is syntax to create a .csr with a single command.

6.1 Use the private key (.key) to create a certificate signing request (.csr). Use the provided information.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ openssl req -new -key web.key -out web.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) [XX]:US
State or Province Name (full name) []:CA
Locality Name (eg, city) [Default City]:Santa Clara
Organization Name (eg, company) [Default Company Ltd]:Cloudride
Organizational Unit Name (eg, section) []:Edu
Common Name (eg, your name or your server's hostname) []:cloudride.example.com
Email Address []:nkelly@cloudride.example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
training@cmhost:~/tls/web$
```

```
$ openssl req -new -key web.key -out web.csr
```

6.2 Verify the content of the request.csr file. Scroll and review the contents of the request.csr. Scroll up to read the input information.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ openssl req -text -in web.csr -noout -verify
verify OK
Certificate Request:
Data:
    Version: 0 (0x0)
    Subject: C=US, ST=CA, L=Santa Clara, O=Cloudride, OU=Edu, CN=cloudride.e
example.com/emailAddress=nkelly@cloudride.example.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
            00:b9:14:30:0a:80:fe:dc:eb:28:e0:64:82:64:2d:
            a4:c0:f7:f3:75:d7:1f:b7:d6:48:33:03:5c:c5:f8:
            2e:6c:02:fa:44:dc:5d:c4:96:eb:6d:8e:99:ef:54:
            58:05:70:20:1b:0d:72:22:84:d9:d7:ee:a4:a2:52:
            75:0f:67:eb:c6:bf:85:4e:0a:32:cd:e4:17:75:2b:
            88:0d:48:ef:c4:7d:72:75:cc:8a:36:d3:4e:4b:24:
            e5:97:35:7a:cc:04:27:e7:2c:46:fb:eb:43:63:f5:
            99:d8:d0:d3:f9:8d:f4:37:02:3d:7b:e6:de:7c:a7:
            81:44:b7:4c:d0:1e:6c:71:85:f9:e7:6f:18:e6:da:
            4f:c6:59:0e:ad:4f:e5:9d:7c:ab:6f:a1:d5:b7:11:
            82:56:13:c3:03:3c:ec:ad:ff:ac:d0:83:ee:28:fc:
```

```
$ openssl req -text -in web.csr -noout -verify
```

6.3 Transfer the certificate signing request (.csr) to the CA.



```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ ls
web.csr  web.key
training@cmhost:~/tls/web$ cp web.csr ~/tls/ca/web.csr
training@cmhost:~/tls/web$
```

```
$ cp web.csr ~/tls/ca/web.csr
```

7. Sign a Certificate Signing Requests (.csr) creates a Signed Certificate (.cr).

The certificate signing request (.csr) is submitted to the Certificate Authority (CA), this is generally through a web UI. The CA will sign the certificate signing request with its own private key (ca.key). The CA creates a new public key encrypted with its private key without compromising the certificate signing request (.crt) itself. The CA issues a command to encode additional information, such as the hostname or username, to create a new public key. This new public key is returned to the originator.

7.1 Change directory to the CA.



```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ cd ~/tls/ca
training@cmhost:~/tls/ca$
```

```
$ cd ~/tls/ca
```

7.2 Use the CA's private key and public certificate to sign the certificate signing request (.csr). The information from ca.crt will be written into web.crt.



```
File Edit View Search Terminal Help
training@cmhost:~/tls/ca$ ls
ca.crt  ca.key  web.csr
training@cmhost:~/tls/ca$ openssl x509 -req -CAkey ca.key -CA ca.crt -in web.csr
-out web.crt -days 365 -CAcreateserial
Signature ok
subject=/C=US/ST=CA/L=Santa Clara/O=Cloudride/OU=Edu/CN=cloudride.example.com/emailAddress=nkelly@cloudride.example.com
Getting CA Private Key
training@cmhost:~/tls/ca$
```

```
$ openssl x509 -req -CAkey ca.key -CA ca.crt -in web.csr -out web.crt -days 365 -CAcreateserial
```

7.3 Transfer the signed public certificate back to the web server.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/ca$ cp web.crt ~/tls/web/web.crt
training@cmhost:~/tls/ca$
```

```
$ cp web.crt ~/tls/web/web.crt
```

8. Validate the signed public key.

8.1 Change directory to tls/web.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/ca$ cd ~/tls/web
training@cmhost:~/tls/web$
```

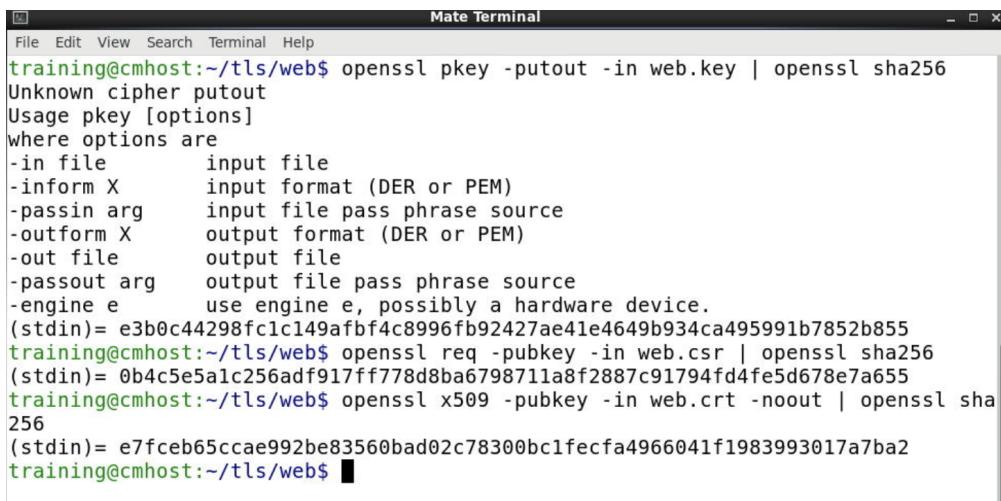
```
$ cd ~/tls/web
```

8.2 Review the Signed Certificate (.crt). Scroll up. Compare the information for the issuer and the subject.

```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ openssl x509 -text -in web.crt -noout
Certificate:
Data:
    Version: 1 (0x0)
    Serial Number:
        c7:c5:f2:65:99:98:cb:95
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, ST=NY, L=New York, O=GlobalCA, OU=Sale, CN=globalca.example.com/emailAddress=info@globalca.example.com
    Validity
        Not Before: Feb 27 18:45:13 2022 GMT
        Not After : Feb 27 18:45:13 2023 GMT
    Subject: C=US, ST=CA, L=Santa Clara, O=Cloudride, OU=Edu, CN=cloudride.example.com/emailAddress=nkelly@cloudride.example.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
                Modulus:
                    00:b9:14:30:0a:80:fe:dc:eb:28:e0:64:82:64:2d:a4:cc:f7:f3:75:d7:1f:b7:d6:48:33:03:5c:c5:f8:2e:6c:02:fa:44:dc:5d:c4:96:eb:6d:8e:99:ef:54:
```

```
$ openssl x509 -text -in web.crt -noout
```

8.3 Verify the keys match, extract the public key from each file and generate a hash output. All three files should share the same public key and the same hash value. You may also pipe to openssl md5.



```
File Edit View Search Terminal Help
training@cmhost:~/tls/web$ openssl pkey -pubout -in web.key | openssl sha256
Unknown cipher putout
Usage pkey [options]
where options are
-in file      input file
-inform X     input format (DER or PEM)
-passin arg   input file pass phrase source
-outform X    output format (DER or PEM)
-out file     output file
-passout arg  output file pass phrase source
-engine e     use engine e, possibly a hardware device.
(stdin)= e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
training@cmhost:~/tls/web$ openssl req -pubkey -in web.csr | openssl sha256
(stdin)= 0b4c5e5a1c256adf917ffff78d8ba6798711a8f2887c91794fd4fe5d678e7a655
training@cmhost:~/tls/web$ openssl x509 -pubkey -in web.crt -noout | openssl sha256
(stdin)= e7fce65ccae992be83560bad02c78300bc1fecfa4966041f1983993017a7ba2
training@cmhost:~/tls/web$ █
```

```
$ openssl pkey -pubout -in web.key | openssl sha256
$ openssl req -pubkey -in web.csr -noout | openssl sha256
$ openssl x509 -pubkey -in web.crt -noout | openssl sha256
```

9. Self-Signed Certificates.

A standard work-around to avoid having to send a .csr to a CA is to create and use a self-signed certificate. Here the generating authority is the same as the signing authority. For Internet purposes this is considered distinctly unsecure. Self-signed certificates are only used for testing, training, and development purposes.