

# Cryptography: Assignment 1

---

**Name:** Albert Ferguson **SID:** 13611165

**Group:** N/A (solo)

## Part I: Setting up SSH

Firstly, there's a few dependencies to check. Namely, I don't have the SSH server configured installed on my system. So I go ahead and install that and connect to localhost as a test,

```

man openssl      albert@diodorus:~          albert@diodorus:/usr/lib/ssl
/usr/lib/ssl > ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:cWVeRvpj3aVw8tK/T4yBvv2+7gfJf9bASwQiDjsrobc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
albert@localhost's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-26-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

Expanded Security Maintenance for Applications is enabled.

0 updates can be applied immediately.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

- > ls ~/git/albert/assignment1
albert@diodorus 14:53:02
openssl_rootca.cnf report
- > echo Hello world!
albert@diodorus 14:53:18
Hello world!
- > █
albert@diodorus 14:53:29

```

You can tell I haven't connected to localhost before as the client prompts me to add it to my list of known hosts. After logging in, you can also see that I'm running Ubuntu 22.04.4 LTS and can browse to the directory where I'm editing this assignment.

Exiting out of this and checking the openssh server service shows it working as expected,

```

man openssl      albert@diodorus:/usr/lib/ssl          albert@diodorus:/usr/lib/ssl
/usr/lib/ssl > sudo systemctl status ssh
albert@diodorus:~          albert@diodorus:/usr/lib/ssl
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: active (running) since Sun 2024-03-31 14:49:48 AEDT; 5min ago
    Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 22411 (sshd)
      Tasks: 1 (limit: 18763)
     Memory: 1.8M
        CPU: 63ms
      CGroup: /system.slice/ssh.service
              └─22411 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 31 14:49:48 diodorus systemd[1]: Starting OpenBSD Secure Shell server...
Mar 31 14:49:48 diodorus sshd[22411]: Server listening on 0.0.0.0 port 22.
Mar 31 14:49:48 diodorus sshd[22411]: Server listening on :: port 22.
Mar 31 14:49:48 diodorus systemd[1]: Started OpenBSD Secure Shell server.
Mar 31 14:53:02 diodorus sshd[24294]: Accepted password for albert from 127.0.0.1 port 58596 ssh2
Mar 31 14:53:02 diodorus sshd[24294]: pam_unix(sshd:session): session opened for user albert(uid=1000) by (uid=0)
φ /usr/lib/ssl > █
albert@diodorus:~          albert@diodorus:/usr/lib/ssl
14:55:06
14:55:12

```

The daemon is running successfully without issue and shows my previous connection in the logs. You can also spot my hostname and the date here (happy easter).

I haven't configured pubkey auth to localhost, but I have done so to other hosts already. To add the localhost config, I'll edit my existing config file.

```
FORWARDX11 no
```

```
Host localhost
Hostname localhost
User albert
PubKeyAuthentication yes
IdentityFile /home/albert/.ssh/localhost
ForwardX11 no
```

Then I generate a new RSA keypair with the password ABC123,

The screenshot shows a terminal window titled "man openssl" with the command "ssh-keygen -t rsa" run by user "albert" at the prompt "albert@diodorus:~/.ssh". The output shows the key generation process, including prompts for saving the key to "/home/albert/.ssh/id\_rsa", entering a passphrase (left empty), and confirming it. It also displays the SHA256 fingerprint of the key and its randomart image.

```
man openssl
albert@diodorus:~/.ssh
~/.ssh > ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/albert/.ssh/id_rsa): localhost
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in localhost
Your public key has been saved in localhost.pub
The key fingerprint is:
SHA256:E6v795gu2C04tsZ0zNw4r+L6yB6qkqbIIzrNUCuRPOk albert@diodorus
The key's randomart image is:
+---[RSA 3072]---+
| |
| ...
| o+. . o
| .o... + oS
| oE. . B...
| *o o.=
| B=oO* o.= .o
| &+**=+o+=+..
+---[SHA256]---+
~/.ssh >
```

This is then added as an authorised key, which lets me ssh to localhost without using my account password,

```
man openssl          x      albert@diodorus:~/ssh          x      albert@diodorus:~          x
~/.ssh > mkdir ~/.ssh/authorized_keys
~/.ssh > rmdir ~/.ssh/authorized_keys
~/.ssh > touch ~/.ssh/authorized_keys
~/.ssh > cat ~/.ssh/localhost.pub >> ~/.ssh/authorized_keys
~/.ssh > head ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQCw2xq5gDt/WSKchE5YY/ZavVUsEEgyNjRCjXpf70KBmw3x0Pwzg44jiJchPo1me/uRLehpjw7onqo7bCZUha/OwO
B0g5d3rvNt9mhKwvWapt606vChz+8FcddT0jnsH/lhIePHVqgWN7BvF08sKywDL0EbJr81VSjjYwsAldb7l4E2+pKofpgnjBp0G/ZjUtn16n+9TuuhXDipKvFI
YBtTP12gKx3oMtvMEcW06sQ3FtsNqsoAPZ2W/b577AeoNyYhBgD5Eb0jvCrKo83iXomkgm3ZRABj7BAW9bHiHj/nkPKOM2FFszdL/Vj8Ad0wo1pJUKd00fmyv4NK
Si5yilI11MbQDSFPvqsrtuxDGU0YkD5twhWg1wxzgxA7AkMcixLkoFzq+9hwgZXGe/HKs0x0/HtJvkeYvxydEF4KkjbI3n4sUsCR5A6X+PHqt1N0dp8k93MuZYd
2m8MkAXW7t1Hx/PZ2kZzLaFFjs80u8iwFcaMhn7UhJYJW0= albert@diodorus
~/.ssh > █
```

```
man openssl          x      albert@diodorus:~/ssh          x      albert@diodorus:~          x
φ /usr/lib/ssl > cd ~/
~ > ssh localhost
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-26-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

Expanded Security Maintenance for Applications is enabled.

0 updates can be applied immediately.

Last login: Sun Mar 31 14:53:02 2024 from 127.0.0.1
~ > █
```

albert@diodorus 15:23:04

Notice I didn't provide the identity file in this command, as the config already set it.

## Part II: Creating and Configuring a Root CA

For this part, the root CA directory will be this assignment's directory `/git/albert/assignment1`. The CA full config used is in the appendix for reference

To setup the CA, I start by configuring the cwd,

```
mkdir certs crl newcerts private
chmod 700 private
touch index.txt serial
echo 1000 > serial
```

Then I created a configuration file by copying and modifying the demo one installed with OpenSSL,

```
~/git/albert/assignment1 master ?3 > head /usr/lib/ssl/openssl.cnf
#
# OpenSSL example configuration file.
# See doc/man5/config.pod for more info.
#
# This is mostly being used for generation of certificate requests,
# but may be used for auto loading of providers

# Note that you can include other files from the main configuration
# file using the .include directive.
#.include filename
~/git/albert/assignment1 master ?3 > cp /usr/lib/ssl/openssl.cnf ./openssl.cnf.bak
~/git/albert/assignment1 master ?3 > ls
certs crl index.txt newcerts openssl.cnf.bak openssl_rootca.cnf private report serial
~/git/albert/assignment1 master ?3 > █
```

16:05:11 16:05:13 16:05:34 16:05:35

I trimmed down some sections and configure the root cert and key directories,

```
# root key and cert
```

```
private_key = $dir/private/ca.key.pem
certificate = $dir/certs/ca.cert.pem
```

The sample file also recommended,

```
# recommended from docs/sample file

name_opt = ca_default
cert_opt = ca_default
```

I then configured a policy to match the assignment requirements, as we will always require these details,

```
[ policy ]
stateOrProvinceName = match
countryName = match
organizationName = match
organizationalUnitName = match
localityName = match
commonName = supplied
emailAddress = supplied
```

Finally, I configure the defaults for requested certs. Some notable parts,

- X509 extensions are enabled,
- a challenge password is required,
- the cert

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
string_mask = utf8only
default_md = sha256
attributes = req_attributes

# Extension to add when the -x509 option is used.

x509_extensions = v3_ca

[ req_distinguished_name ]

# See <https://en.wikipedia.org/wiki/Certificate\_signing\_request>.

countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name (full name) [Some-State]
localityName = Locality Name (eg, city)
0.organizationName = Organization Name (eg, company)
organizationalUnitName = Organizational Unit Name (eg, section)
```

```

commonName = Common Name (e.g. server FQDN or YOUR name)
emailAddress = Email Address

# specify defaults for this example

countryName_default = AU
stateOrProvinceName_default = NSW
localityName_default = SYD
0.organizationName_default = UTS
organizationalUnitName_default = FEIT
commonName_default = utscrypto.com.au
emailAddress_default = root@utscrypto.com.au

[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20

# an optional company name as requested in the assignment details

unstructuredName = An optional company name

```

As an extra I applied the default `[server_cert]` extension, as we're about to create a server cert for the next part.

Next, I create a 4096-bit root key with password `ABC123` and lock down its permissions as readonly,

```

openssl genrsa -aes256 -out ./private/ca.key.pem 4096
chmod 400 ./private/ca.key.pem

```

With that complete, then I create the root cert, ensuring to specify my custom config file. I set the expiry to some large amount of time in the future (7300 days = 20 years). I also set its password to `ABC123`,

```

openssl req -config openssl_rootca.cnf \
    -key ./private/ca.key.pem \
    -new -x509 -days 7300 -sha256 -extensions v3_ca \
    -out ./certs/ca.cert.pem

```

This then prompts for the cert information, which defaults to the expected configuration,

```
~/git/albert/assignment1 master ?3 > openssl req -config openssl_rootca.cnf \
  -key ./private/ca.key.pem \
  -new -x509 -days 7300 -sha256 -extensions v3_ca \
  -out ./certs/ca.cert.pem
Enter pass phrase for ./private/ca.key.pem:
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]:
State or Province Name (full name) [Some-State] [NSW]:
Locality Name (eg, city) [SYD]:
Organization Name (eg, company) [UTS]:
Organizational Unit Name (eg, section) [FEIT]:
Common Name (e.g. server FQDN or YOUR name) [utscrypto.com.au]:
Email Address [root@utscrypto.com.au]:
~/git/albert/assignment1 master ?3 > 
```

5s 16:34:03

I then lock this down to readonly for all and verify it with another openssl command,

```
chmod 444 certs/ca.cert.pem
openssl x509 -noout -text -in ./certs/ca.cert.pem
```

```
~/git/albert/assignment1 master ?3 > openssl x509 -noout -text -in ./certs/ca.cert.pem
16:36:13
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
    28:aa:0d:60:64:16:34:46:1e:f8:78:50:7f:ac:35:2c:b8:6c:98:ee
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = AU, ST = NSW, L = SYD, O = UTS, OU = FEIT, CN = utscrypto.com.au, emailAddress = root@utscrypto.com.au
Validity
    Not Before: Mar 31 05:34:03 2024 GMT
    Not After : Mar 26 05:34:03 2044 GMT
Subject: C = AU, ST = NSW, L = SYD, O = UTS, OU = FEIT, CN = utscrypto.com.au, emailAddress = root@utscrypto.com.au
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (4096 bit)
            Modulus:
                00:9f:a8:66:56:74:06:42:42:8a:42:2a:de:94:b1:
                b7:a5:c5:8e:7c:e4:63:af:b9:25:fa:20:8b:0b:b8:
                6b:45:57:86:b6:cd:26:b6:c8:27:bb:cf:b9:4b:5e:
                87:55:d9:66:04:35:cc:cb:76:95:bd:9e:2c:4c:2b:
                b7:36:be:7a:61:69:95:2a:ed:49:1a:b6:4c:6c:81:
                03:5f:0a:d8:d0:d1:96:68:23:88:d9:c6:71:25:b7:
                ca:ea:f9:07:af:73:13:e4:b7:80:96:e5:a1:de:4f:
                30:f5:56:3a:50:86:26:4a:26:5c:79:f0:bd:f1:75:
                77:51:ad:98:b0:67:a7:5a:e8:50:dd:80:f2:8d:40:
                97:f6:a1:ef:0e:da:94:09:3f:0e:97:b9:19:db:ef:
                8f:2d:a4:09:57:07:d3:9d:d3:37:61:5b:5c:cf:37:
                fb:15:7b:1e:ba:8d:3b:d7:7f:63:23:7d:46:ac:7f:
                0a:51:24:6e:56:6c:b6:62:4f:9c:f3:85:11:3d:54:
                3f:6c:45:23:85:79:ac:34:4f:44:84:02:53:77:6d:
                dc:32:d7:dd:56:56:cd:30:8a:25:0f:1a:93:f1:ef:
                2f:95:0c:6c:d3:62:80:b3:1d:d7:e3:cd:02:51:5c:
                48:6c:f3:84:ef:59:e1:11:39:0c:c9:3f:49:c8:8e:
                76:9e:87:6d:2a:d3:df:24:11:c4:38:fd:7f:a5:e3:
                95:04:7e:a5:9e:6c:57:64:51:f9:39:a1:9f:99:40:
                ee:3f:cd:83:55:42:11:44:70:42:86:e5:2b:6f:33:
                91:cd:14:8d:62:fa:b3:5c:f9:cb:1e:7d:b8:f4:8b:
                84:6c:46:32:55:06:b8:2b:9a:f5:44:06:03:d2:8e:
                b5:44:84:e8:d3:b3:d8:2a:d7:58:1b:6e:f9:05:7c:
                39:51:bd:92:bf:c7:6a:53:89:3a:fd:6b:1a:6f:d7:
                91:f1:e2:bf:e0:88:2d:87:c5:30:9f:18:8d:09:3c:
                38:e0:32:3b:a7:79:15:cb:34:ea:38:b7:ce:38:c2:
                c5:94:2e:2e:dc:91:fa:d6:44:ae:99:69:c8:aa:2d:
                ad:20:df:af:f4:3e:35:85:fd:b5:4c:c6:96:8f:4d:
                14:b0:c2:b2:5a:57:37:a4:59:fa:c4:36:8d:7a:c7:
                f2:73:2b:21:da:4f:3f:b9:aa:24:1a:17:a8:3f:91:
                b4:28:48:b5:e1:6f:ff:b8:c0:c9:49:89:d4:cb:2b:
                33:6b:20:be:29:c2:dd:e9:83:5d:6e:77:56:0d:bf:
                46:65:81:ce:05:9b:60:72:3a:8e:2f:26:9b:a8:18:
                a4:a2:17:e0:10:ce:54:3c:d1:7e:e4:91:96:ea:d3:
                73:7b:e1
            Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Key Identifier:
        A3:9A:F6:14:02:DB:C9:04:08:07:D3:B6:8F:6D:63:35:8F:CF:C1:1F
    X509v3 Authority Key Identifier:
        A3:9A:F6:14:02:DB:C9:04:08:07:D3:B6:8F:6D:63:35:8F:CF:C1:1F
    X509v3 Basic Constraints: critical
        CA:TRUE
    X509v3 Key Usage: critical
        Digital Signature, Certificate Sign, CRL Sign
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
    8b:31:89:bc:0a:62:ef:dc:3a:fa:1b:88:8f:73:71:86:8d:20:
    d6:73:b9:57:be:32:21:0d:19:38:5e:bb:c9:99:45:83:a9:22:
    bd:88:01:7b:06:71:dd:5b:4c:6a:2a:50:c7:1e:d2:c0:a7:fd:
```

The working directory now looks like so,

```
~/git/albert/assignment1 master ?3 > tree -f -I "report"
.
└── ./certs
    └── ./certs/ca.cert.pem
── ./crl
── ./index.txt
── ./newcerts
── ./openssl.cnf.bak
── ./openssl_rootca.cnf
── ./private
    └── ./private/ca.key.pem
── ./serial

4 directories, 6 files
~/git/albert/assignment1 master ?3 >
```

## Part III: Generating an X.509 Cert for a Server

To generate an RSA private key and CSR in one go, I use the `openssl` command (again setting the key to readonly)

```
openssl req -config openssl_rootca.cnf \
-new -newkey rsa:2048 \
-keyout ./albertferguson.key \
-out ./albertferguson.csr
chmod 400 ./albertferguson.key
```

Since I have omitted the `-nodes` / `-noenc` flags, the default private key encryption is DES. The command prompts a form as expected. I then verify the key,

```
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]:
State or Province Name (full name) [Some-State] [NSW]:
Locality Name (eg, city) [SYD]:
Organization Name (eg, company) [UTS]:
Organizational Unit Name (eg, section) [FEIT]:
Common Name (e.g. server FQDN or YOUR name) [utscrypto.com.au]:
Email Address [root@utscrypto.com.au]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
~/git/albert/assignment1 master ?3 > openssl req -in albertferguson.csr -noout -text
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = AU, ST = NSW, L = SYD, O = UTS, OU = FEIT, CN = utscrypto.com.au, emailAddress = root@utscrypto.com.au
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
            Modulus:
                00:98:c2:e3:cd:b1:e2:d8:64:88:65:6a:43:19:e6:
                fb:57:d7:60:4c:1c:2e:e6:04:d1:cf:42:ea:69:70:
                fc:24:02:a5:36:bc:59:10:14:1d:60:67:ff:0b:1e:
                e8:66:7b:7b:8a:ba:5b:7e:55:66:28:86:31:1c:04:
                35:4a:0d:10:5a:56:a5:ad:3e:d5:d4:38:b3:aa:17:
                ea:9d:09:bb:54:49:52:a1:64:b1:d0:20:b2:ec:f2:
                52:ea:b3:d8:9a:65:a4:c8:51:c6:9b:86:c1:da:e5:
                3f:8d:f6:1c:5d:2a:d8:20:d1:f0:d4:da:84:62:1f:
                a9:00:8a:48:a3:14:6a:9e:93:dc:6f:81:d6:b4:0d:
                a8:7d:a0:78:6b:ae:b8:63:6f:a9:dd:64:66:b8:bc:
                a4:c1:1b:6a:91:d5:bc:c0:99:f7:7e:e2:e9:73:12:
                18:59:3c:2b:0:a:ed:fc:45:89:1a:0a:df:b2:2f:62:
                28:59:ca:b4:9c:84:b5:c4:b8:33:1e:9b:eb:65:c2:
                b4:dc:f0:09:69:03:b0:4e:05:a0:a9:75:7f:39:41:
                b7:a8:4c:78:5b:7d:58:be:47:43:ad:65:28:49:c9:
                2a:2a:67:c8:3e:93:18:08:7f:66:0c:7f:06:56:e5:
                d2:af:b9:3f:b3:c3:c3:cf:b7:2f:b6:6e:e6:a2:ae:
                0e:49
            Exponent: 65537 (0x10001)
Attributes:
    (none)
Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
33:09:9f:55:b9:b4:fd:fc:f0:51:ae:63:78:60:dd:25:45:2b:
98:ef:0d:49:c7:35:6c:0b:53:f7:10:a1:e8:21:21:d5:c5:94:
44:21:d5:85:50:2b:23:fd:33:2e:ee:ec:34:19:af:d4:46:86:
0f:0f:94:f1:44:c8:65:44:fb:1d:74:93:b5:60:37:27:5d:5c:
79:3e:41:18:a1:cd:1a:ea:02:bc:e9:64:bb:bb:64:77:03:48:
db:93:1e:cc:d9:e5:5c:25:7b:b1:d5:15:0d:9d:a3:08:76:e6:
b5:5d:a1:bb:9b:6b:a3:a6:5b:68:06:60:63:f3:8b:a0:79:98:
48:e0:be:46:0d:26:f3:5c:91:ec:af:e6:df:f8:17:bb:c5:ae:
13:25:69:55:a7:4b:a8:13:be:b7:6d:db:44:61:91:29:45:83:
c0:d3:2a:0:d6:3a:71:fe:b4:48:e7:14:fa:6c:5b:11:1b:fc:
5c:b5:01:f5:a5:32:aa:ec:04:0b:27:b0:ab:26:bf:eb:8d:73:
c4:bc:5b:8b:20:ec:25:df:aa:85:97:af:d2:30:1c:30:51:6c:
75:e2:27:c5:e1:8e:a2:8b:5b:25:cd:60:26:24:8f:ed:ab:fd:
79:1b:73:68:11:29:df:a1:c4:d5:79:ed:44:ea:00:fb:1d:63:
68:e6:8d:05
~/git/albert/assignment1 master ?3 > 17:13:44
```

Next, I create a certificate for the local server using the above CSR. Notably, I'm invoking the server extensions

```
openssl ca -config openssl_rootca.cnf \
    -extensions server_cert -days 375 -notext -md sha256 \
    -in ./albertferguson.csr \
    -out ./albertferguson.cert
chmod 444 ./albertferguson.cert
```

Processing this and applying the chmod change,

```
Attributes:
  (none)
  Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
33:09:9f:55:b9:b4:fd:fc:f0:51:ae:63:78:60:dd:25:45:2b:
98:ef:0d:49:c7:35:6c:0b:53:f7:10:a1:e8:21:21:d5:c5:94:
44:21:d5:85:50:2b:23:fd:33:2e:ee:ec:34:19:af:d4:46:86:
0f:0f:94:f1:44:c8:65:44:fb:1d:74:93:b5:60:37:27:5d:5c:
79:3e:41:18:a1:cd:1a:ea:02:bc:e9:64:bb:bb:64:77:03:48:
db:93:1e:cc:d9:e5:5c:25:7b:b1:d5:15:0d:9d:a3:08:76:e6:
b5:5d:a1:bb:9b:6b:a3:a6:5b:68:06:60:63:f3:8b:a0:79:98:
48:e0:be:46:0d:26:f3:5c:91:ec:af:e6:df:f8:17:bb:c5:ae:
13:25:69:55:a7:4b:a8:13:be:b7:6d:db:44:61:91:29:45:83:
c0:d3:2a:d0:d6:3a:71:fe:b4:48:e7:14:fa:6c:5b:11:1b:fc:
5c:b5:01:f5:a5:32:a:ec:04:0b:27:b0:ab:26:bf:eb:8d:73:
c4:bc:5b:8b:20:ec:25:df:aa:85:97:af:d2:30:1c:30:51:6c:
75:e2:27:c5:e1:8e:a2:8b:5b:25:cd:60:26:24:8f:ed:ab:fd:
79:1b:73:68:11:29:df:a1:c4:d5:79:ed:44:ea:00:fb:1d:63:
68:e6:8d:05
Check that the request matches the signature
Signature ok
Everything appears to be ok, creating and signing the certificate
Successfully added extensions from config
The subject name appears to be ok, checking data base for clashes
Certificate Details:
  Serial Number: 4096 (0x1000)
  Validity
    Not Before: Mar 31 06:21:03 2024 GMT
    Not After : Apr 10 06:21:03 2025 GMT
  Subject:
    stateOrProvinceName      = NSW
    countryName               = AU
    organizationName          = UTS
    organizationalUnitName    = FEIT
    localityName              = SYD
    commonName                = utscrypto.com.au
    emailAddress              = root@utscrypto.com.au
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Cert Type:
    SSL Server
  Netscape Comment:
    OpenSSL Generated Server Certificate
  X509v3 Subject Key Identifier:
    69:F8:B8:EB:F9:9C:CE:6E:0E:04:9E:ED:CC:12:27:C5:2F:D4:A0:07
  X509v3 Authority Key Identifier:
    keyid:A3:9A:F6:14:02:DB:C9:04:08:07:D3:B6:8F:6D:63:35:8F:CF:C1:1F
    DirName:/C=AU/ST=NSW/L=SYD/O=UTS/OU=FEIT/CN=utscrypto.com.au/emailAddress=root@utscrypto.com.au
    serial:28:AA:0D:60:64:16:34:46:1E:F8:78:50:7F:AC:35:2C:B8:6C:98:EE
  X509v3 Key Usage: critical
    Digital Signature, Key Encipherment
  X509v3 Extended Key Usage:
    TLS Web Server Authentication
Certificate is to be certified until Apr 10 06:21:03 2025 GMT (375 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
writing new certificates
writing /home/albert/git/albert/assignment1/newcerts/1000.pem
Data Base Updated
~/git/albert/assignment1 master ?3 > chmod 444 ./albertferguson.cert
~/git/albert/assignment1 master ?3 > 
```

11s 17:21:11  
17:21:46

Verifying this all,

```
~/git/albert/assignment1 master ?3 > openssl x509 -in ./albertferguson.cert -text --noout          17:22:45
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number: 4096 (0x1000)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = AU, ST = NSW, L = SYD, O = UTS, OU = FEIT, CN = utscrypto.com.au, emailAddress = root@utscrypto.com.au
    Validity
        Not Before: Mar 31 06:21:03 2024 GMT
        Not After : Apr 10 06:21:03 2025 GMT
    Subject: ST = NSW, C = AU, O = UTS, OU = FEIT, L = SYD, CN = utscrypto.com.au, emailAddress = root@utscrypto.com.au
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
            Modulus:
```

The CA is also aware of the cert,

```
~/git/albert/assignment1 master ?3 > cat index.txt          17:23:49
V      250410062103Z      1000      unknown /ST=NSW/C=AU/O=UTS/OU=FEIT/L=SYD/CN=utscrypto.com.au/emailAddress=root@utscrypt
to.com.au
~/git/albert/assignment1 master ?3 >          17:23:55
```

## Part IV: Running a secure webserver

To run the webserver, first the `.cert` and `.key` files need to be merged into the PEM format,

```
cat albertferguson.key >> server.pem
cat albertferguson.cert >> server.pem
chmod 400 server.pem
```

Then I append a local domain in my host file that matches the cert (`utscrypto.com.au`) using `openssl` to launch a server with the created PEM file,

phone tabs | Projects | Libraries | A

**Certificate Viewer: utscrypto.com.au**

**General** Details

Issued To

Common Name (CN)	utscrypto.com.au
Organisation (O)	UTS
Organisational Unit (OU)	FEIT

Issued By

Common Name (CN)	utscrypto.com.au
Organisation (O)	UTS
Organisational Unit (OU)	FEIT

Validity Period

Issued On	Sunday, 31 March 2024 at 17:21:03
Expires On	Thursday, 10 April 2025 at 16:21:03

SHA-256 Fingerprints

Certificate	7663251fda00179e3ec704dd76e39fc20a95bde9f7a5ca2eac703235 906213a1
Public key	9857f13b26a96d45ae796203414aeee84a29f56f8fa93b0fb0c1fb9f5 27a160d

```
s_server -cert server.pem -www
Secure Renegotiation IS NOT supported
Ciphers supported in s_server binary
TLSv1.3 :TLS_AES_256_GCM_SHA384 TLSv1.3
TLSv1.3 :TLS_AES_128_GCM_SHA256 TLSv1.2
TLSv1.2 :ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2
TLSv1.2 :ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2
TLSv1.2 :DHE-RSA-CHACHA20-POLY1305 TLSv1.2
TLSv1.2 :ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2
TLSv1.2 :ECDHE-ECDSA-AES256-SHA384 TLSv1.2
TLSv1.2 :DHE-RSA-AES256-SHA256 TLSv1.2
TLSv1.2 :ECDHE-RSA-AES128-SHA256 TLSv1.2
TLSv1.0 :ECDHE-ECDSA-AES256-SHA TLSv1.0
SSLv3 :DHE-RSA-AES256-SHA TLSv1.0
TLSv1.0 :ECDHE-RSA-AES128-SHA SSLv3
TLSv1.2 :RSA-PSK-AES256-GCM-SHA384 TLSv1.2
TLSv1.2 :RSA-PSK-CHACHA20-POLY1305 TLSv1.2
TLSv1.2 :ECDHE-PSK-CHACHA20-POLY1305 TLSv1.2
TLSv1.2 :PSK-AES256-GCM-SHA384 TLSv1.2
TLSv1.2 :RSA-PSK-AES128-GCM-SHA256 TLSv1.2
TLSv1.2 :AES128-GCM-SHA256 TLSv1.2
TLSv1.2 :AES256-SHA256 TLSv1.2
TLSv1.0 :ECDHE-PSK-AES256-CBC-SHA384 TLSv1.0
SSLv3 :SRP-RSA-AES-256-CBC-SHA SSLv3
TLSv1.0 :RSA-PSK-AES256-CBC-SHA384 TLSv1.0
SSLv3 :RSA-PSK-AES256-CBC-SHA SSLv3
SSLv3 :AES256-SHA TLSv1.0
SSLv3 :PSK-AES256-CBC-SHA TLSv1.0
TLSv1.0 :ECDHE-PSK-AES128-CBC-SHA SSLv3
SSLv3 :SRP-AES-128-CBC-SHA TLSv1.0
TLSv1.0 :DHE-PSK-AES128-CBC-SHA256 SSLv3
SSLv3 :DHE-PSK-AES128-CBC-SHA SSLv3
TLSv1.0 :PSK-AES128-CBC-SHA256 SSLv3
---
Ciphers common between both SSL end points:
TLS_AES_128_GCM_SHA256 TLS_AES_256_GCM_SHA256
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-ECDSA-CHACHA20
ECDHE-RSA-AES128-SHA ECDHE-RSA-AES256-SHA
AES256-GCM-SHA384 AES128-SHA
Signature Algorithms: ECDSA+SHA256:RSA-PSS+SHA256
Shared Signature Algorithms: ECDSA+SHA256:RSA-PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA-PSS+SHA384:RSA+SHA384:RSA-PSS+SHA384
Supported groups: :x25519:secp256r1:secp384r1
Shared groups: x25519:secp256r1:secp384r1
---
New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
SSL-Session:
Protocol : TLSv1.3
Cipher : TLS_AES_128_GCM_SHA256
Session-ID: 7BBA8C1904081C6A70916BAE40C13DB6EE39794B37EB0E2A5BFEF7C5548A4D7D
Session-ID-ctx: 01000000
Resumption PSK: 0278C2218495AF25A2936088DAF7F48DE29F2AD39C941E5E7EC4541EBC848E24
PSK identity: None
PSK identity hint: None
SRP username: None
Start Time: 1711867357
Timeout : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
---
0 items in the session cache
0 client connects (SSL_connect())
0 client renegotiates (SSL_connect())
0 client connects that finished
16 server accepts (SSL_accept())
0 server renegotiates (SSL_accept())
1 server accepts that finished
0 session cache hits
0 session cache misses
0 session cache timeouts
0 callback cache hits
0 cache full overflows (128 allowed)
---
no client certificate available
```

The last part here would be to import the CA's certificate so that I can trust the site. Importing the root CA cert for Chrome under,

Settings > Privacy and Security > Manage Certificates > Authorities

I expected this to work. However, I am missing Subject Alternative Name (SAN) configuration,

```

cert server.pem www
negotiation IS NOT supported
supported in 5 server binary
[TLS_AES_256_GCM_SHA384] TLSv1.3 :TLS_GCM_CKEDH_Poly1305_SHA256
[TLS_AES_256_GCM_SHA384] TLSv1.2 :ECDHE-ECDSA-AES256-GCM-SHA384
:ECDHE-RSA-AES256-GCM-SHA384
[ECDHE-ECDSA-AES256-GCM-SHA384] TLSv1.2 :ECDHE-RSA-AES256-GCM-SHA384
[ECDHE-ECDSA-CHACHA20-POLY1305] TLSv1.2 :ECDHE-RSA-CHACHA20-POLY1305
[DHE-RSA-CHACHA20-POLY1305] TLSv1.2 :ECDHE-ECDSA-AES128-GCM-SHA256
[ECDHE-ECDSA-AES128-GCM-SHA256] TLSv1.2 :ECDHE-RSA-AES128-GCM-SHA256
[ECDHE-ECDSA-AES256-SHA384] TLSv1.2 :ECDHE-RSA-AES256-SHA384
[DHE-RSA-AES256-SHA256] TLSv1.2 :DHE-RSA-AES128-SHA256
[ECDHE-ECDSA-AES256-SHA] TLSv1.0 :ECDHE-RSA-AES256-SHA
[DHE-RSA-AES256-SHA] TLSv1.0 :ECDHE-ECDSA-AES128-SHA
[ECDHE-RSA-AES128-SHA] SSLv3 :DHE-RSA-AES128-SHA
[RSASSA-PKCS1-v1_5] AES256-GCM-SHA384 TLSv1.2 :DHE-PSK-AES256-GCM-SHA384

```

As of 2017, Chrome 58 removed support for common names and preferred SAN [1]. This missing config could be rectified by updating my root CA's config to include the [ alt\_names ] extension. However, I think this is fine for the purpose of this assignment.

## Bibliography

[1] "Deprecations and Removals in Chrome 58 | Blog," Chrome for Developers. Accessed: Mar. 31, 2024. [Online]. Available: <https://developer.chrome.com/blog/chrome-58-deprecations>

## Appendix Root CA Config

```
[ ca ]
default_ca = CA_default

[ CA_default ]
prompt = no
dir = /home/albert/git/albert/assignment1
certs = $dir/certs
crl_dir = $dir/crl
new_certs_dir = $dir/newcerts
database = $dir/index.txt
serial = $dir/serial
RANDFILE = $dir/private/.rand

# root key and cert
private_key = $dir/private/ca.key.pem
certificate = $dir/certs/ca.cert.pem

# for cert revocation lists
crlnumber = $dir/crlnum
crl = $dir/crl/mycrl.pem
default_crl_days = 30

default_md = sha256

# recommended from docs/sample file
name_opt = ca_default
cert_opt = ca_default

preserve = no
# typically this would be used to create intermediate CAs (strict policy)
# but we're skipping a step here
policy = policy
default_days = 365

[ policy ]
stateOrProvinceName = match
countryName = match
organizationName = match
organizationalUnitName = match
localityName = match
commonName = supplied
emailAddress = supplied

[ req ]
```

```

default_bits          = 2048
distinguished_name   = req_distinguished_name
string_mask          = utf8only
default_md           = sha256
attributes           = req_attributes

# Extension to add when the -x509 option is used.
x509_extensions      = v3_ca

[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate\_signing\_request>.
countryName           = Country Name (2 letter code)
stateOrProvinceName   = State or Province Name (full name) [Some-
State]
localityName          = Locality Name (eg, city)
0.organizationName     = Organization Name (eg, company)
organizationalUnitName = Organizational Unit Name (eg, section)
commonName            = Common Name (e.g. server FQDN or YOUR
name)
emailAddress          = Email Address

# specify defaults for this example
countryName_default    = AU
stateOrProvinceName_default = NSW
localityName_default    = SYD
0.organizationName_default = UTS
organizationalUnitName_default = FEIT
commonName_default       = utscrypto.com.au
emailAddress_default     = root@utscrypto.com.au

[ req_attributes ]
challengePassword      = A challenge password
challengePassword_min   = 4
challengePassword_max   = 20
# an optional company name as requested in the assignment details
unstructuredName        = An optional company name

[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier    = hash
authorityKeyIdentifier   = keyid:always,issuer
basicConstraints         = critical, CA:true
keyUsage                 = critical, digitalSignature, cRLSign,
keyCertSign

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints         = CA:FALSE
nsCertType                = server
nsComment                 = "OpenSSL Generated Server
Certificate"
subjectKeyIdentifier      = hash
authorityKeyIdentifier    = keyid,issuer:always
keyUsage                  = critical, digitalSignature,

```

```
keyEncipherment  
extendedKeyUsage = serverAuth
```