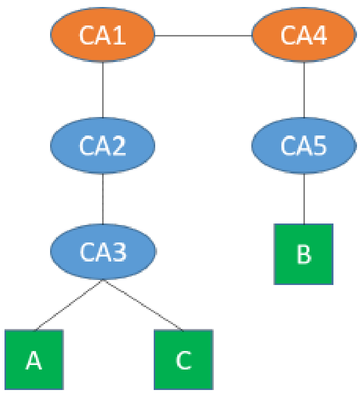
There are certificate authorities (CA) and end entities that follow the X.509 hierarchy as follows.



- CA1 and CA4 are the root certificate authorities.
- End-entity A is a subscriber of CA3 and it trusts CA1.
- End-entity B is a subscriber of CA5 and it trusts CA4.
- End-entity C is a subscriber of CA3 and it trusts CA1.
- Suppose that the certificate authorities and the end-entities have created their 2048-bit RSA public and private key files as follows.

Entity	Public Key File	Private Key File
CA1	pk1.pem	sk1.pem
CA2	pk2.pem	sk2.pem
CA3	pk3.pem	sk3.pem
CA4	pk3.pem	sk4.pem
CA5	pk5.pem	sk5.pem
A	pka.pem	ska.pem
B	pkb.pem	skb.pem
C	pkc.pem	skc.pem

Note. All files can be found in /step1, /step2, /step3, and /step4 folders. All of the public keys, private keys can be found in /step2. Cross-certifications are found in /step3 and /step4 respectively.

[6 marks] Create certificates for the root certificate authorities, CA1 and CA4, respectively

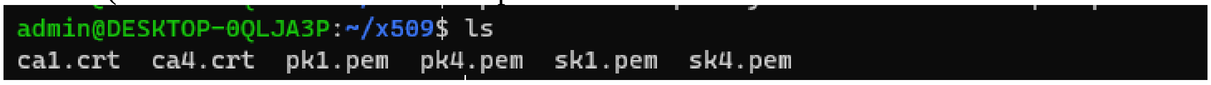
1. Create CA1's Self Signed Certificate:
   `'openssl req -x509 -sha256 -nodes -days 3650 -newkey rsa:2048 -keyout sk1.pem -out ca1.crt'`

2. Create CA4's Self Signed Certificate:
   `'openssl req -x509 -sha256 -nodes -days 3650 -newkey rsa:2048 -keyout sk4.pem -out ca4.crt'`

3. Extract CA1's and CA4's public key from the certificates respectively:

```
`openssl x509 -pubkey -in ca1.crt -out pk1.pem`
`openssl x509 -pubkey -in ca4.crt -out pk4.pem`
```

4. Results (Note this can be found in /Step1 folder



```
admin@DESKTOP-0QLJA3P:~/x509$ ls
ca1.crt  ca4.crt  pk1.pem  pk4.pem  sk1.pem  sk4.pem
```

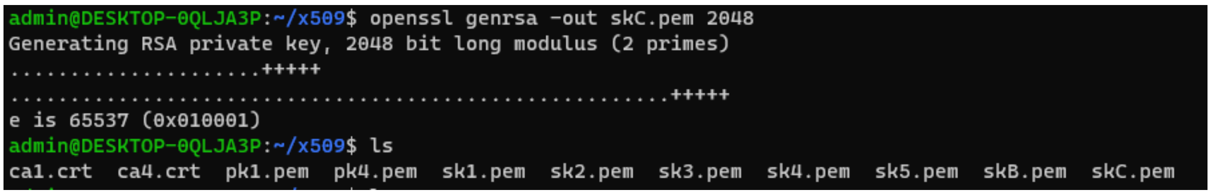
2. [12 marks] Create certificates for B and C, respectively.

Note due to the X.509 hierarchy this will require creating CA3, CA2 and CA5 respectively as CA3 is required to sign C's certificate and CA5 is required to sign B's certificate

1. Create CA'2, CA3's, Ca5's, B's, C's and A's public and private keys respectively

```
‘openssl genrsa -out sk2.pem 2048’
‘openssl genrsa -out sk3.pem 2048’
‘openssl genrsa -out sk5.pem 2048’
‘openssl genrsa -out skb.pem 2048’
‘openssl genrsa -out skc.pem 2048’
‘openssl genrsa -out ska.pem 2048’
```

Note e is always 65537



```
admin@DESKTOP-0QLJA3P:~/x509$ openssl genrsa -out skC.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................+++++
.......................................................+++++
e is 65537 (0x010001)
admin@DESKTOP-0QLJA3P:~/x509$ ls
ca1.crt  ca4.crt  pk1.pem  pk4.pem  sk1.pem  sk2.pem  sk3.pem  sk4.pem  sk5.pem  skB.pem  skC.pem
```

2. Generate certificate signing requests (CRL) files for CA2 to CA1 and have CA1 sign the request

```
`openssl req -new -key sk2.pem -out ca2.csr`
`openssl x509 -req -days 365 -in ca2.csr -CA ca1.crt -CAkey sk1.pem
-sha256 -CAcreateserial -out ca2.crt -extfile ca.ext`
```

Contents of ca.ext. This file is created to ensure that the intermediate CA's created are NOT leaf nodes.

```
`authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:TRUE`
```

3. Generate certificate signing request (CRL) files for CA3 to CA2 and have CA2 sign the request

```
‘openssl req -new -key sk3.pem -out ca3.csr’
```

```
   ‘openssl x509 -req -days 365 -in ca3.csr -CA ca2.crt -CAkey
sk2.pem -sha256 -CAcreateserial -out ca3.crt -extfile ca.ext`
```

4. Generate certificate signing request (CRL) files for CA5 to CA4 and have CA4 sign the request

```
   ‘openssl req -new -key sk5.pem -out ca5.csr’
   ‘openssl x509 -req -days 365 -in ca5.csr -CA ca4.crt -CAkey
sk4.pem -sha256 -CAcreateserial -out ca5.crt -extfile ca.ext`
```

5. Generate certificate signing request (CRL) files for C to CA3 and have CA3 sign the request

```
‘openssl req -new -key skc.pem -out c.csr’
‘openssl x509 -req -days 365 -in c.csr -CA ca3.crt -CAkey sk3.pem -
sha256 -CAcreateserial -out c.crt’
```

6. Generate certificate signing request (CRL) files for B to CA5 and have CA5 sign the request

```
‘openssl req -new -key skb.pem -out b.csr’
‘openssl x509 -req -days 365 -in b.csr -CA ca5.crt -CAkey sk5.pem -
sha256 -CAcreateserial -out b.crt’
```

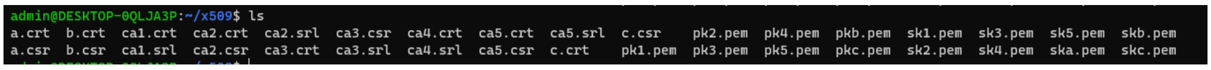
7. Generate certificate signing request (CRL) files for A to CA3 and have CA3 sign the request

```
‘openssl req -new -key ska.pem -out a.csr’
‘openssl x509 -req -days 365 -in a.csr -CA ca3.crt -CAkey sk3.pem -
sha256 -CAcreateserial -out a.crt’
```

8. Extract CA2, CA3, CA4, C and B’s public key from the certificates respectively:

```
`openssl x509 -pubkey -in ca2.crt -out pk2.pem`
`openssl x509 -pubkey -in ca3.crt -out pk3.pem`
`openssl x509 -pubkey -in ca5.crt -out pk5.pem`
`openssl x509 -pubkey -in c.crt -out pkc.pem`
`openssl x509 -pubkey -in b.crt -out pkb.pem`
`openssl x509 -pubkey -in a.crt -out pka.pem`
```

9. Results (Note this can be found in /Step2 folder, all CSR and SRL have been moved to /step/CSR:



```
admin@DESKTOP-0QLJA3P:~/x509$ ls
a.crt  b.crt  ca1.crt  ca2.crt  ca2.srl  ca3.csr  ca4.crt  ca5.crt  ca5.srl  c.csr    pk2.pem  pk4.pem  pkb.pem  sk1.pem  sk3.pem  sk5.pem  skb.pem
a.csr  b.csr  ca1.srl  ca2.csr  ca3.crt  ca3.srl  ca4.srl  ca5.csr  c.crt    pk1.pem  pk3.pem  pk5.pem  pkc.pem  sk2.pem  sk4.pem  ska.pem  skc.pem
```

3. [6 marks] Demonstrate how to verify C’s certificate by B.

Important note! I'm not sure if there is a mistake in the question here! It seems as if you want us to verify C's certificate by A. I suspect this because the ordering of the questions and mark allocations seem weird considering both question 3 and question 5 need cross certification from 4-1 and 1-4 respectively and question 3 only gets 6 marks while question 5 gets 7 and an intermediate question!

1. Generate Cross Certification from CA4 to CA1

```
`openssl req -new -key sk1.pem -out ca1.csr`
`openssl x509 -req -days 365 -in ca1.csr -CA ca4.crt -CAkey sk4.pem
-sha256 -CAcreateserial -out ca4-1.crt`
```

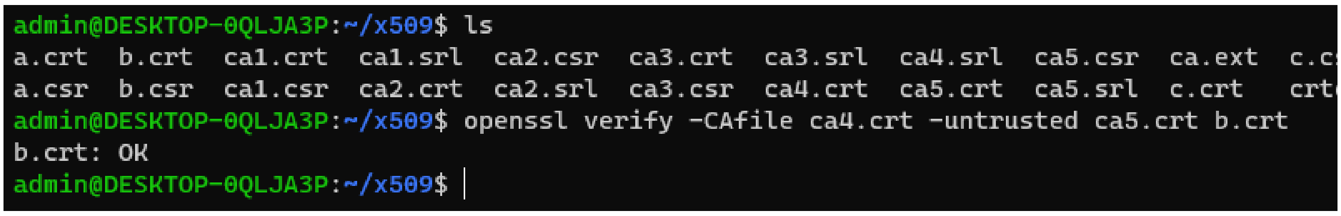
2. Verifying C's Certificate by B

We would expect the following verification chain B <<CA5>>, CA5 <<CA4>>, CA4 <<CA1>>, CA1 <<CA2>>, CA2 <<CA3>>, CA3 <<C>>

a. Check if the root authority is trusted directly

Note, `-untrusted` must always be used when verifying through intermediate authorities which is why it has been included in the command!

```
`openssl verify -CAfile ca4.crt -untrusted ca5.crt b.crt`
```



```
admin@DESKTOP-0QLJA3P:~/x509$ ls
a.crt  b.crt  ca1.crt  ca1.srl  ca2.csr  ca3.crt  ca3.srl  ca4.srl  ca5.csr  ca.ext  c.c
a.csr  b.csr  ca1.csr  ca2.crt  ca2.srl  ca3.csr  ca4.crt  ca5.crt  ca5.srl  c.crt   crt
admin@DESKTOP-0QLJA3P:~/x509$ openssl verify -CAfile ca4.crt -untrusted ca5.crt b.crt
b.crt: OK
admin@DESKTOP-0QLJA3P:~/x509$
```

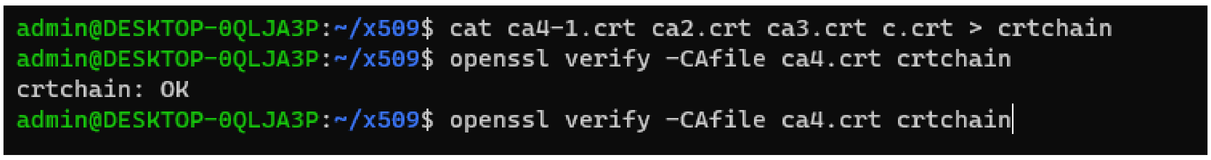
b. Use cross certification to check if the user trusts C

Combine all the certificates in a certification path

```
`cat ca4-1.crt ca2.crt ca3.crt c.crt > crtchain`
```

Verify C's certificate using crtchain

```
`openssl verify -CAfile ca4.crt crtchain`
```



```
admin@DESKTOP-0QLJA3P:~/x509$ cat ca4-1.crt ca2.crt ca3.crt c.crt > crtchain
admin@DESKTOP-0QLJA3P:~/x509$ openssl verify -CAfile ca4.crt crtchain
crtchain: OK
admin@DESKTOP-0QLJA3P:~/x509$ openssl verify -CAfile ca4.crt crtchain
```

4. [5 marks] Create certificate(s) that are needed to verify B's certificate by A.

1. Generate Cross Certification from CA1 to CA4

```
`openssl req -new -key sk4.pem -out ca4.csr`
`openssl x509 -req -days 365 -in ca4.csr -CA ca1.crt -CAkey sk1.pem
-sha256 -CAcreateserial -out ca1-4.crt`
```

2. All other documents were created in question 2, i.e., the intermediate certification chain.

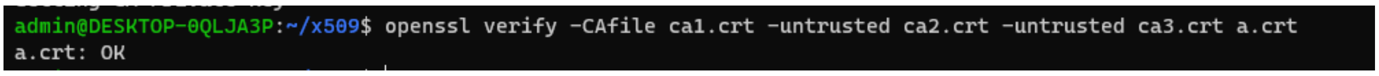
5. [7 marks] Demonstrate how to verify B's certificate by A.

We would expect the following verification chain A <<CA3>>, CA3 <<CA2>>, CA2 <<CA1>>, CA1 <<CA4>>, CA4 <<CA5>>, CA5 <<B>>

1. Check if the root authority is trusted directly

Note, `-untrusted` must always be used when verifying through intermediate authorities which is why it has been included in the command!

```
`openssl verify -CAfile ca1.crt -untrusted ca2.crt -untrusted ca3.crt
a.crt`
```



```
admin@DESKTOP-0QLJA3P:~/x509$ openssl verify -CAfile ca1.crt -untrusted ca2.crt -untrusted ca3.crt a.crt
a.crt: OK
```

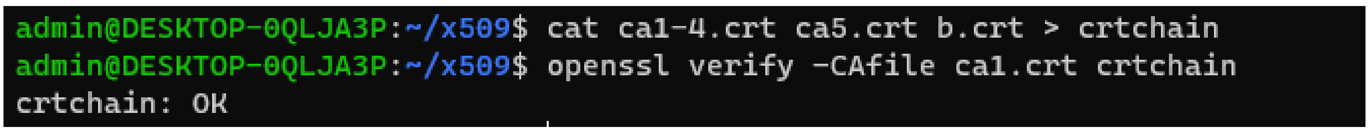
2. Use cross certification to check if the user trusts B

   a. Combine all the certificates in a certification path

```
`cat ca1-4.crt ca5.crt b.crt > crtchain`
```

   b. Verify C's certificate using crtchain

```
`openssl verify -CAfile ca1.crt crtchain`
```



```
admin@DESKTOP-0QLJA3P:~/x509$ cat ca1-4.crt ca5.crt b.crt > crtchain
admin@DESKTOP-0QLJA3P:~/x509$ openssl verify -CAfile ca1.crt crtchain
crtchain: OK
```