

# Network and System Security ET2595

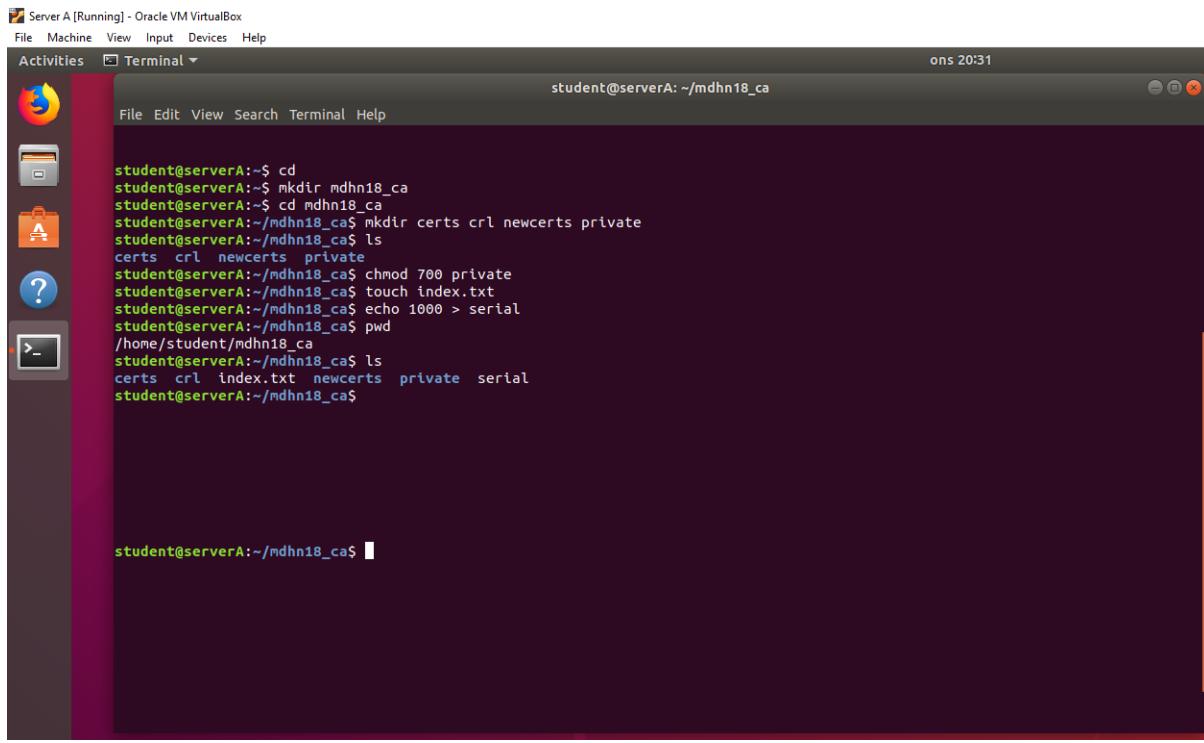
## Lab 2 Report

Name: Md Rabiul Ahamed Bin Hanif

P. no. 19920212-2637

Email: [mdhn18@student.bth.se](mailto:mdhn18@student.bth.se)

### Environments create:



```
student@serverA:~$ cd
student@serverA:~$ mkdir mdhn18_ca
student@serverA:~$ cd mdhn18_ca
student@serverA:~/mdhn18_ca$ mkdir certs crl newcerts private
student@serverA:~/mdhn18_ca$ ls
certs crl newcerts private
student@serverA:~/mdhn18_ca$ chmod 700 private
student@serverA:~/mdhn18_ca$ touch index.txt
student@serverA:~/mdhn18_ca$ echo 1000 > serial
student@serverA:~/mdhn18_ca$ pwd
/home/student/mdhn18_ca
student@serverA:~/mdhn18_ca$ ls
certs crl index.txt newcerts private serial
student@serverA:~/mdhn18_ca$
```



```
student@serverA:~$ cp /etc/ssl/openssl.cnf /home/student/mdhn18_ca
student@serverA:~$
```

- openssl.cnf file was changed according to the instruction. All screen short are given below.

```

# We can add new OIDs in here for use by 'ca', 'req' and 'ts'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# or use config file substitution like this:
# testoid2=$(testoid1).5.6

# Policies used by the TSA examples.
tsa_policy1 = 1.2.3.4.1
tsa_policy2 = 1.2.3.4.5.6
tsa_policy3 = 1.2.3.4.5.7

#####
[ ca ]
default_ca = CA_default      # The default ca section

#####
[ CA_default ]

dir      = /home/student/mdhn18_ca    # Where everything is kept
certs   = $dir/certs        # Where the issued certs are kept
crl_dir = $dir/crl          # Where the issued crl are kept
database = $dir/index.txt   # database index file.
unique_subject = no          # Set to 'no' to allow creation of
                             # several certs with same subject.
new_certs_dir = $dir/newcerts # default place for new certs.

certificate = $dir/certs/root.cert.pem # The CA certificate
serial    = $dir/serial        # The current serial number
crlnumber = $dir/crlnumber    # the current crl number
crl      = $dir/crl.pem       # The current CRL
private_key = $dir/private/root.key.pem # The private key
RANDFILE  = $dir/private/.rand # private random number file

# x509_extensions = usr_cert      # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt = ca_default         # Subject Name options
cert_opt = ca_default          # Certificate field options

# Extension copying options: use with caution.
# copy_extensions = copy

```

## Task 1: [ v3\_ca ]

```

[ v3_ca ]

# Extensions for a typical CA

# PKIX recommendation.

subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = critical,CA:true

# Key usage: this is typical for a CA certificate. However since it will
# prevent it being used as an test self-signed certificate it is best
# left out by default.
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX recommendation
# subjectAltName=DNS:copy
# Copy issuer details
# issuerAltName=issuerCopy

```

**subjectKeyIdentifier = hash**

Identifies certificates which contain a specific public key.

**authorityKeyIdentifier = keyid:always,issuer**

Identifies the public key corresponding to the private key which was used to sign a certificate. For the **keyid** option, if it is present an attempt is made to copy the subject key identifier from the parent certificate.

For the **issuer** option, it copies the issuer and serial number from the issuer certificate. This is applicable when the **keyid** option fails or is not included unless the "always" flag will always include the value.

**basicConstraints = CA:true**

This is a multi valued extension which indicates whether a certificate is a CA certificate. The first (mandatory) name is CA followed by TRUE or FALSE. If CA is TRUE then an optional "pathlen" is used followed by an non-negative integer value can be included.

`keyUsage = critical, digitalSignature, cRLSign, keyCertSign`

Key usage is a multi valued extension consisting of a list of names of the permitted key usages. The digitalSignature bit is used when the subject public key is used for verifying digital signatures, other than signatures on certificates (bit 5) and CRLs (bit 6), such as those used in an entity authentication service, a data origin authentication service, and/or an integrity service.

The cRLSign bit is used when the subject public key is used for verifying signatures on certificate revocation lists (for example, CRLs, delta CRLs, or ARLs).

The keyCertSign bit is used when the subject public key is used for verifying signatures on public key certificates.

## Task 2: [ [ v3\_intermediate\_ca ] ]

```
[ v3_intermediate_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

`subjectKeyIdentifier=hash`

Identifies certificates which contain a specific public key.

`authorityKeyIdentifier = keyid:always,issuer`

Identifies the public key corresponding to the private key which was used to sign a certificate.

For the **keyid** option, if it is present an attempt is made to copy the subject key identifier from the parent certificate.

For the **issuer** option, it copies the issuer and serial number from the issuer certificate. This is applicable when the keyid option fails or is not included unless the "always" flag will always include the value.

`basicConstraints = critical, CA:true, pathlen:0`

Indicates whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate.

If CA is TRUE then an optional pathlen name followed by an non-negative integer value can be included.

The pathlen parameter indicates the maximum number of CAs that can appear below this one in a chain. So, a CA with a pathlen of zero can only be used to sign end user certificates and not further CAs.

`keyUsage = critical, digitalSignature, cRLSign, keyCertSign`

Key usage is a multi valued extension consisting of a list of names of the permitted key usages. The digitalSignature bit is used when the subject public key is used for verifying digital signatures, other than signatures on certificates (bit 5) and CRLs (bit 6), such as those used in an entity authentication service, a data origin authentication service, and/or an integrity service.

The cRLSign bit is used when the subject public key is used for verifying signatures on certificate revocation lists (for example, CRLs, delta CRLs, or ARLs).

The keyCertSign bit is used when the subject public key is used for verifying signatures on public key certificates.

Difference between [ v3\_ca ] and [ v3\_intermediate\_ca ] is the values of basicConstraints. The values are **CA:true** and **critical, CA:true, pathlen:0** respectively.

### Task 3: [ usr\_cert ]

```
[ usr_cert ]
?
# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType          = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

# This will be displayed in Netscape's comment listbox.
nsComment          = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
```

**subjectKeyIdentifier=hash**

Identifies certificates which contain a specific public key.

**authorityKeyIdentifier = keyid,issuer**

Identifies the public key corresponding to the private key which was used to sign a certificate. For the **keyid** option, if it is present an attempt is made to copy the subject key identifier from the parent certificate.

For the **issuer** option, it copies the issuer and serial number from the issuer certificate. This will only be done if the keyid option fails or is not included unless the "always" flag will always include the value.

**basicConstraints=CA:FALSE**

An end user certificate must either set CA to FALSE or exclude the extension entirely. Some software may require the inclusion of basic Constraints with CA to be set to FALSE for end entity certificates.

**nsComment = "OpenSSL Generated Certificate"**

This message will be displayed in Netscape's comment listbox.

keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment

The **nonRepudiation** bit is used when the subject public key is used to verify digital signatures, other than signatures on certificates (bit 5) and CRLs (bit 6), used to provide a non-repudiation service that protects against the signing entity falsely denying some action.

The **digitalSignature** bit is used when the subject public key is used for verifying digital signatures, other than signatures on certificates (bit 5) and CRLs (bit 6), such as those used in an entity authentication service, a data origin authentication service, and/or an integrity service.

The **keyEncipherment** bit is used when the subject public key is used for enciphering private or secret keys, i.e., for key transport.

#### **extendedKeyUsage = clientAuth, emailProtection**

This extension consists of a list of usages indicating purposes of which the certificate public key could be used for.

The value **clientAuth** represents SSL/TLS Web Client Authentication.

The value **emailProtection** represents E-mail Protection (S/MIME).

### **Task 4: [ server\_cert ]**

```
[ server_cert ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

#### **basicConstraints=CA:FALSE**

An end user certificate must either set CA to FALSE or exclude the extension entirely. Some software may require the inclusion of basic Constraints with CA to be set to FALSE for end entity certificates.

#### **subjectKeyIdentifier=hash**

Identifies certificates which contain a specific public key.

#### **authorityKeyIdentifier = keyid,issuer:always**

Identifies the public key corresponding to the private key which was used to sign a certificate. For the **keyid** option, if it is present an attempt is made to copy the subject key identifier from the parent certificate.

For the issuer option, it copies the issuer and serial number from the issuer certificate. This will only be done if the keyid option fails or is not included unless the "always" flag will always include the value.

#### **keyUsage = critical, digitalSignature, keyEncipherment**

The digitalSignature bit is used when the subject public key is used for verifying digital signatures, other than signatures on certificates (bit 5) and CRLs (bit 6), such as those used in an entity authentication service, a data origin authentication service, and/or an integrity service. The keyEncipherment bit is asserted when the subject public key is used for enciphering private or secret keys, i.e., for key transport.

#### **extendedKeyUsage = serverAuth**

This extension consists of a list of usages indicating purposes of which the certificate public key could be used for.

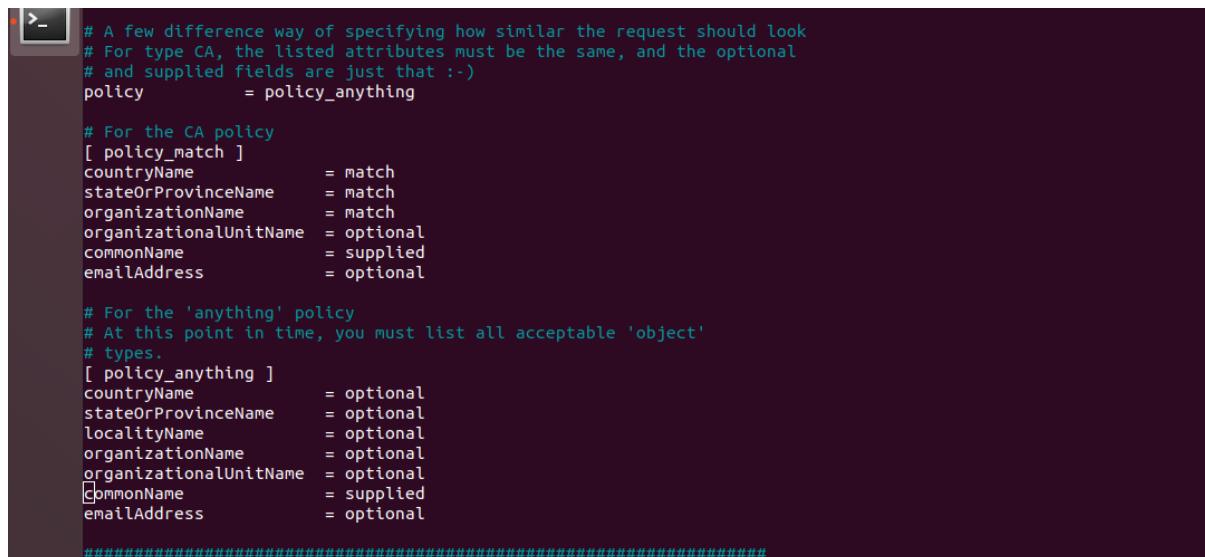
The value **serverAuth** represents SSL/TLS Web Server Authentication

## Task 5: Policies

- Before checking the policy, I create ca1 directory which is shown I below screen short.



```
student@serverA:~/mdhn18_ca$ mkdir ca1
student@serverA:~/mdhn18_ca$ cd ca1
student@serverA:~/mdhn18_ca/ca1$ ls
certs crl newcerts private csr
student@serverA:~/mdhn18_ca/ca1$ ls
certs crl csr newcerts private
student@serverA:~/mdhn18_ca/ca1$ chmod 700 private
student@serverA:~/mdhn18_ca/ca1$ touch index.txt echo 2000 > serial
student@serverA:~/mdhn18_ca/ca1$ echo 2000 > crlnumber
student@serverA:~/mdhn18_ca/ca1$ cp ./openssl.cnf .
student@serverA:~/mdhn18_ca/ca1$
```



```
# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy      = policyAnything

# For the CA policy
[policyMatch]
countryName      = match
stateOrProvinceName = match
organizationName   = match
organizationalUnitName = optional
commonName        = supplied
emailAddress       = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[policyAnything]
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName   = optional
organizationalUnitName = optional
commonName        = supplied
emailAddress       = optional
```

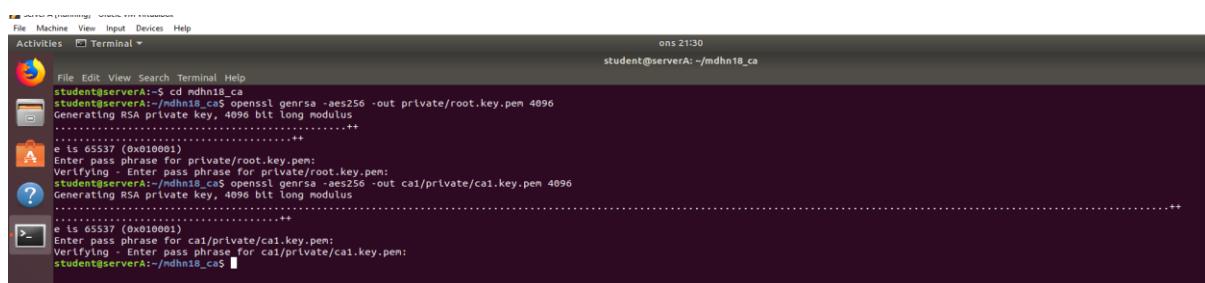
If the value of an attribute is "match", then it should be matched with the CA's DN. All fields listed as "supplied" must be present. If a field is listed as "optional", then it is allowed and not even required to be there.

In **policy\_match** section, the values of countryName, stateOrProvinceName, organizationName must be present as the CA for all certificates it signs and commonName will be supplied by the issuer.

In **policy\_anything** section, only commonName needs to be supplied and all the other values are optional.

## Task 6: Options for the root certificate

- create the private RSA key for *root* and *ca1*, respectively.



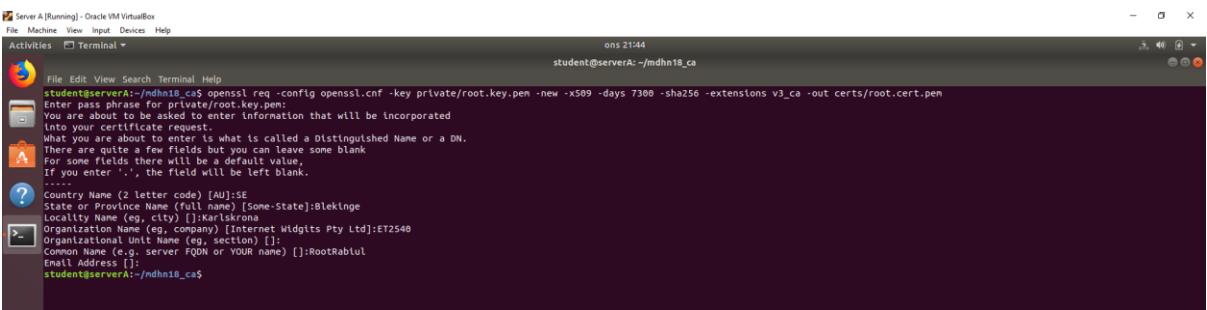
```
student@serverA:~/mdhn18_ca$ cd mdhn18_ca
student@serverA:~/mdhn18_ca$ openssl genrsa -aes256 -out private/root.key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....+
e Ls 65537 (0x100001)
Enter pass phrase for private/root.key.pem:
Verifying - Enter pass phrase for private/root.key.pem:
student@serverA:~/mdhn18_ca$ openssl genrsa -aes256 -out ca1/private/ca1.key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....+
e Ls 65537 (0x100001)
Enter pass phrase for ca1/private/ca1.key.pem:
Verifying - Enter pass phrase for ca1/private/ca1.key.pem:
student@serverA:~/mdhn18_ca$
```

- To obtain the public key (e.g., for root.ca.pem) using OpenSSH you can use the following command.
- For additional protection, the filesystem access rights to private keys should be restricted to the owner.



```
Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal 21:38
student@serverA: ~/mdhn18_ca
student@serverA:~$ cd mdhn18_ca
student@serverA:~/mdhn18_ca$ openssl rsa -in private/root.key.pem -pubout -out root.pub.pem
writing RSA key
student@serverA:~/mdhn18_ca$ chmod 400 private/root.key.pem
student@serverA:~/mdhn18_ca$ chmod 400 ca1.key.pem
student@serverA:~/mdhn18_ca$
```

- To generate the self-signed certificate entering the following command.



```
Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal 21:44
student@serverA: ~/mdhn18_ca
student@serverA:~/mdhn18_ca$ openssl req -config openssl.cnf -key private/root.key.pem -new -x509 -days 7300 -sha256 -extensions v3_ca -out certs/root.cert.pem
Enter pass phrase for private/root.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 about to be 6 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]:SE
State or Province Name (full name) [Some-State]:Blekinge
Locality Name (eg, city) []:Karlskrona
Organization Name (eg, company) [Internet Widgets Pty Ltd]:ET2540
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
student@serverA:~/mdhn18_ca$
```

## Command

***openssl req -config openssl.cnf -key private/root.key.pem -new -x509 -days 7300 -sha256 -extensions v3\_ca -out certs/root.cert.pem***

### Description

#### **-config filename**

this allows an alternative configuration file to be specified, this overrides the compile time filename or any specified in the OPENSSL\_CONF environment variable.

#### **-key filename**

This specifies the file to read the private key from. It also accepts PKCS#8 format private keys for PEM format files.

#### **-new**

this option generates a new certificate request. It will prompt the user for the relevant field values. The actual fields prompted for and their maximum and minimum sizes are specified in the configuration file and any requested extensions. If the -key option is not used it will generate a new RSA private key using information specified in the configuration file.

#### **-x509**

this option outputs a self signed certificate instead of a certificate request. This is typically used to generate a test certificate or a self signed root CA. The extensions added to the certificate (if any) are specified in the configuration file. Unless specified using the set\_serial option, a large random number will be used for the serial number.

#### **-days n**

when the -x509 option is being used this specifies the number of days to certify the certificate for. The default is 30 days.

### -extensions section

this option specifies alternative sections to include certificate extensions (if the -x509 option is present) or certificate request extensions. This allows several different sections to be used in the same configuration file to specify requests for a variety of purposes.

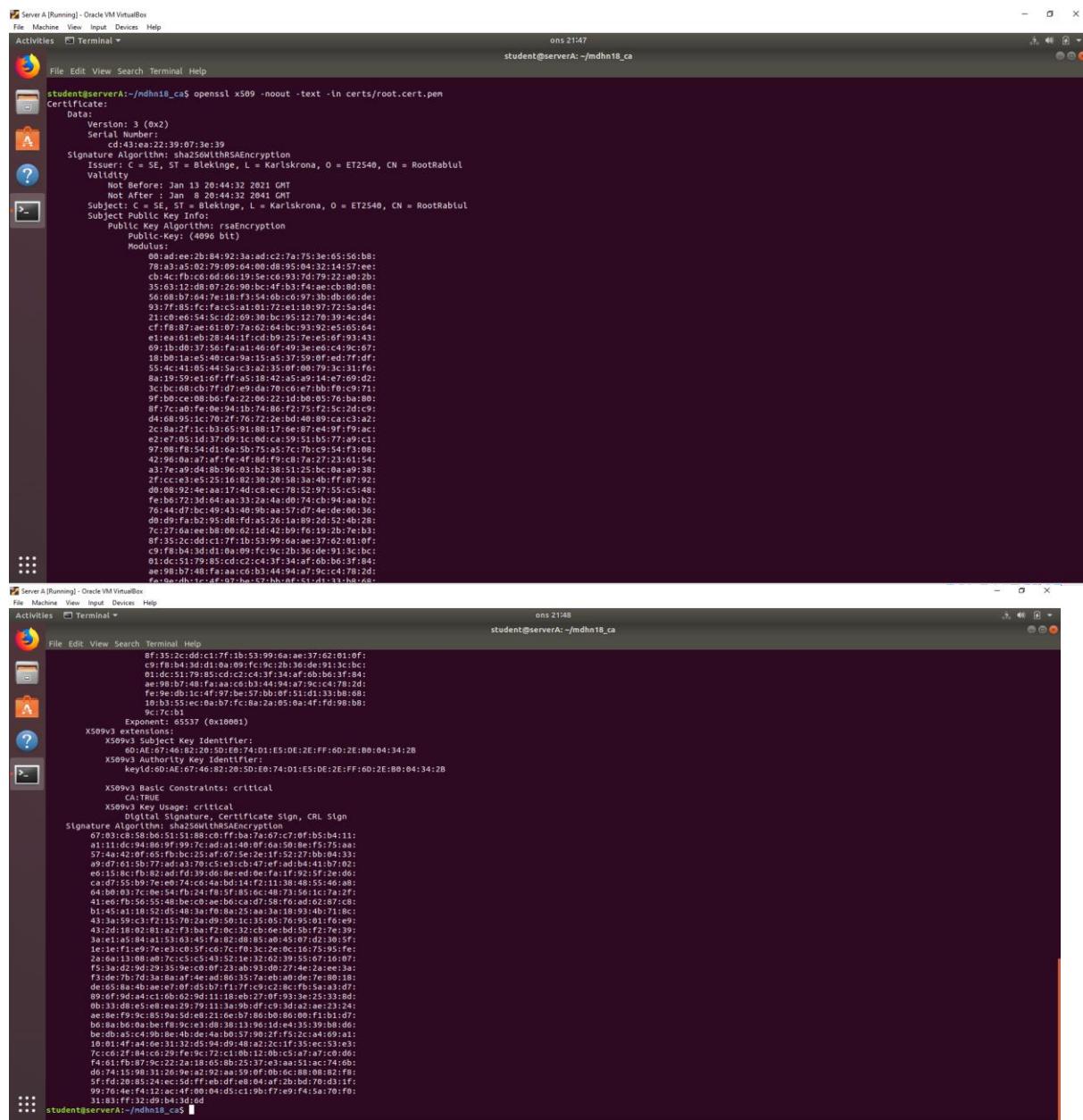
### -out filename

This specifies the output filename to write to or standard output by default.

## Task 7: Verify the root certificate

### Command:

```
openssl x509 -noout -text -in certs/root.cert.pem
```



The image shows two terminal windows side-by-side, both titled "Server A [Running] - Oracle VM VirtualBox". The top terminal window shows the output of the command "openssl x509 -noout -text -in certs/root.cert.pem". The bottom terminal window shows the output of the command "openssl x509 -noout -text -in certs/root.cert.pem" followed by "student@serverA:~/mdhn18\_ca\$". Both terminals are running on a Linux desktop environment with a dark theme.

```
student@serverA:~/mdhn18_ca$ openssl x509 -noout -text -in certs/root.cert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            cd:43:ee:22:39:07:3e:99
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = RootRabil
        Validity:
            Not Before: Jan 13 20:44:32 2021 GMT
            Not After : Jan 8 20:44:32 2041 GMT
        Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = RootRabil
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
                Modulus:
                    00:ad:ee:2b:84:92:3a:ad:c2:7a:75:3e:65:56:b8:
                    78:1a:3a:05:02:79:09:64:00:08:95:04:32:14:57:ee:
                    c5:81:31:0c:09:24:0f:40:1e:00:3d:53:14:38:65:
                    35:63:12:08:07:26:90:bc:f4:b3:f4:aec:cb:8d:08:
                    56:08:b7:04:7e:10:f1:51:0b:co:97:3b:0b:66:de:
                    93:7f:85:fcc:f1:c5:ca:01:72:e1:08:97:72:5a:d4:
                    21:c0:ed:54:01:d2:d9:03:0c:95:12:76:19:4c:d4:
                    cf:9c:09:31:03:24:05:11:0e:5d:91:5a:3d:03:1c:05:
                    e1:k8:61:6b:78:44:1f:cd:b9:25:7e:ee:5f:93:43:
                    69:1b:dd:03:56:fa:aa:14:00:0f:49:3e:ee:0c:49:07:
                    18:bb:1a:5:40:ca:9a:15:5:37:59:0f:fe:df:7f:df:
                    55:4c:41:05:44:5a:c1:a1:23:5:0f:00:79:c1:31:f6:
                    51:30:4d:15:1b:0b:0b:00:12:14:00:00:00:00:00:00:
                    3c:bc:0d:cb:7f:1d:0f:da:70:c6:07:bb:01:00:93:73:
                    9f:bb:ce:08:bb:6b:fa:22:0d:22:1d:b6:b0:05:76:ba:80:
                    8f:7c:a0:rf:fe:0e:94:1b:74:86:f2:75:f2:c1:2d:c9:
                    d4:18:95:1c:70:2f:76:72:2e:bd:40:89:ca:c3:a2:
                    2c:18:a2:2f:c1:0c:91:88:01:0e:01:ef:0f:1c:a2:c1:
                    55:4e:03:57:09:18:03:0c:3a:54:56:57:09:01:11:
                    57:98:fb:54:d1:6a:b5:7b:5a:57:c7:7b:c9:54:f3:08:
                    42:96:0a:aa:7a:fe:af:bf:0d:79:c8:7a:27:23:01:54:
                    a3:7e:a9:d4:bb:96:03:bb:23:8:51:25:bc:0a:9:38:
                    2f:cc:ce:0e:53:16:82:00:26:58:01:40:ff:0f:37:92:
                    0d:9b:0e:21:0b:00:00:00:00:00:00:00:00:00:00:00:
                    fe:9b:72:3d:64:aa:31:7a:4x:00:74:cb:94:aa:b2:
                    76:44:0d:7b:c4:99:43:40:9a:aa:57:d7:4e:de:06:36:
                    d0:d9:fa:b2:95:d8:fd:a5:26:1a:89:24:52:4b:28:
                    7c:27:0a:ae:eb:00:02:id:42:b9:fo:19:b:b7:b3:
                    ff:78:52:2c:22:33:00:00:00:00:00:00:00:00:00:00:
                    c9:fb:84:5d:01:0a:00:f1:c9:2b:36:0d:9d:3c:bc:
                    01:dc:51:79:85:cd:c2:c4:3f:34:af:0b:b6:3f:84:
                    ae:9b:7b:40:fa:aa:cb:14:44:94:07:9c:c4:78:2d:
                    fa:0c:dh:1c:47:0f:07:ba:57:bb:af:51:d3:13:bb:68:
                    student@serverA:~/mdhn18_ca$
```

## Task 8: Verify the CSR

- Following command below to create the CSR

```
student@serverA:/mdhn18_ca$ openssl req -config ca1/openssl.cnf -new -sha256 -key ca1/private/ca1.key.pem -out ca1/csr/ca1.csr.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]:SE
State or Province Name (full name) [Some-State]:Blekinge
Locality Name (eg, city) [Karlskrona]:Karlskrona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ET2540
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:CA1Rabtl
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
student@serverA:/mdhn18_ca$
```

- command below to verify the CSR

```
student@serverA:/mdhn18_ca$ openssl req -text -noout -verify -in ca1/csr/ca1.csr.pem
verify OK
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = CA1Rabtl
SubjectKeyId: 00:01:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
Public-Key Algorithm: rsaEncryption
-----  
Modulus:
00:ea:30:c1:a7:2e:13:8c:de:d6:5d:02:0e:31:44:  
20:9c:4d:dd:2b:5a:ce:25:ba:4f:6:52:c1:a1:29:c8:  
95:4e:f9:03:b6:ab:52:50:1e:81:00:60:00:00:00:00:  
73:4a:70:3a:31:70:2e:00:00:00:00:00:00:00:00:00:  
cb:8d:df:52:07:46:7d:43:57:20:56:ds:ci:s1:bb:  
93:a8:7a:z7:84:09:1f:88:55:84:77:06:11:7c:7b:  
c8:93:0c:93:2a:1c:3e:44:6c:4a:71:93:3d:31:47:  
5e:c5:85:05:11:c3:35:c5:bb:bb:2a:10:ci:ee:89:  
2d:49:der:01:f0:3e:65:34:15:49:be:f7:ef:c8:a8:  
16:53:01:02:04:02:03:01:01:00:00:00:00:00:00:  
95:8a:ef:30:0c:ca:b8:62:39:ad:46:93:24:01:38:39:  
15:00:0f:f8:bf:0b:49:de:97:23:b3:0c:df:f1:c8:0a:  
23:b4:1e:11:00:00:00:00:00:00:00:00:00:00:00:  
c0:0b:33:ec:c1:71:a7:94:96:e3:a9:77:be:c1:c9:4e:  
78:b1:dc:ae:a5:b0:00:09:00:62:14:13:30:01:c5:  
09:21:0a:00:00:00:00:00:00:00:00:00:00:00:00:  
5f:28:a7:05:de:55:2b:29:50:0d:f6:cf:3a:c0:c0:f4:  
b2:c5:c0:c1:c1:a7:04:f7:42:11:14:03:29:00:b8:a1:  
20:79:0b:0e:40:00:00:00:00:00:00:00:00:00:00:  
27:2d:e1:a5:53:41:a3:e4:02:7b:00:dc:74:f0:91:ad:  
8f:6e:40:08:09:2d:a5:6e:a0:2f:de:de:ef:7b:7:46:  
49:0a:3d:1e:65:01:00:00:00:00:00:00:00:00:00:  
b1:5d:01:a4:d3:aa:97:de:80:00:c0:93:51:c1:  
f1:25:b2:dc:34:8b:1b:39:db:84:24:cd:09:39:30:  
33:0e:6b:83:43:3d:40:00:00:00:00:00:00:00:  
ae:0f:c1:5d:cc:0f:fd:13:fc:f1:ad:ca:5b:2c:d2:  
6c:61:ab:98:0d:bf:ff:ab:44:i1:2:56:ab:58:0b:fe:24:  
4f:79:30:f1:a4:b9:98:01:91:6e:0e:7c:6e:50:7e:  
b4:f6:ef:0c:1e:f5:0e:c1:5b:c7:c0:fd:40:0b:db:  
ea:f0:84:38:05:11:63:53:46:7e:27:31:75:5:f1:  
45:03:01:00:00:00:00:00:00:00:00:00:00:00:  
70:02:fd:97:b4:64:84:2d:09:17:75:77:f3:d0:4d:  
d9:fd:cc:0b:7:6:7b:5c:0d:0:21:21:e9:07:10:30:8d:  
-----  
Exponent: 65537 (0x10001)
```

Attributes:  
-----  
Signature Algorithm: sha256WithRSAEncryption  
-----

```
student@serverA: ~/mdhn18_ca
```

## Task 9: Options for intermediate CA certificate

- Create the certificate for CA1 using the CSR and the root CA is signing the certificate it is important to use *root's* OpenSSL configuration file.

```
student@serverA: ~/mdhn18_ca$ openssl ca -config openssl.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha256 -in ca1/csr/ca1.csr.pem -out ca1/certs/ca1.cert.pem
Using configuration from openssl.cnf
Enter pass phrase for /home/student/ndhn18_ca/private/root.key.pem:
Can't open '/home/student/ndhn18_ca/index.txt.attr' for reading, No such file or directory
139666560471488:error:20080080:BIO routines:IO_newfile:no such file:../crypto/bio/bss_file.c:74:fopen('/home/student/ndhn18_ca/index.txt.attr','r')
139666560471488:error:02000080:BIO routines:IO_newfile:No such file:../crypto/bio/bss_file.c:87:
Signature ok
Certificate Details:
    Serial Number: 4096 (0x1000)
    Validity
        Not Before: Jan 13 21:05:46 2021 GMT
        Not After : Jan 11 21:05:46 2031 GMT
    Subject:
        countryName            = SE
        stateOrProvinceName   = Stockholm
        organizationName      = ESFIA
        commonName             = CA1Rabilul
X509v3 extensions:
    X509v3 Subject Key Identifier:
        A2:32:FA:60:A6:41:01:17:E3:90:91:79:92:5B:71:A4:E8:6E:59:97
    X509v3 Authority Key Identifier:
        keyid:00:A6:07:46:B2:20:50:E0:74:D1:E5:DE:2E:FF:60:2E:B0:04:34:28
    X509v3 Basic Constraints: critical
        CA:TRUE, pathlen:0
    X509v3 Certificate Policies:
        1.3.6.1.4.1.4033.1.1.4.1.1
        Digital Signature, Certificate Sign, CRL Sign
Certificate is to be certified until Jan 11 21:05:46 2031 GMT (3650 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
Data Base Updated
student@serverA: ~/ndhn18_ca$
```

Command

```
openssl ca -config openssl.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha256 -in ca1/csr/ca1.csr.pem -out ca1/certs/ca1.cert.pem
```

## Description

### **-config openssl.cnf**

specifies the configuration file to use.

### **-extensions v3\_intermediate\_ca**

the section of the configuration file containing certificate extensions to be added when a certificate is issued (defaults to x509\_extensions unless the -extfile option is used). If no extension section is present then, a V1 certificate is created. If the extension section is present (even if it is empty), then a V3 certificate is created.

### **-days 3650**

the number of days to certify the certificate for.

### **-notext**

don't output the text form of a certificate to the output file.

### **-md sha256**

the message digest to use. Possible values include md5, sha1 and mdc2. This option also applies to CRLs.

### **-in ca1/csr/ca1.csr.pem**

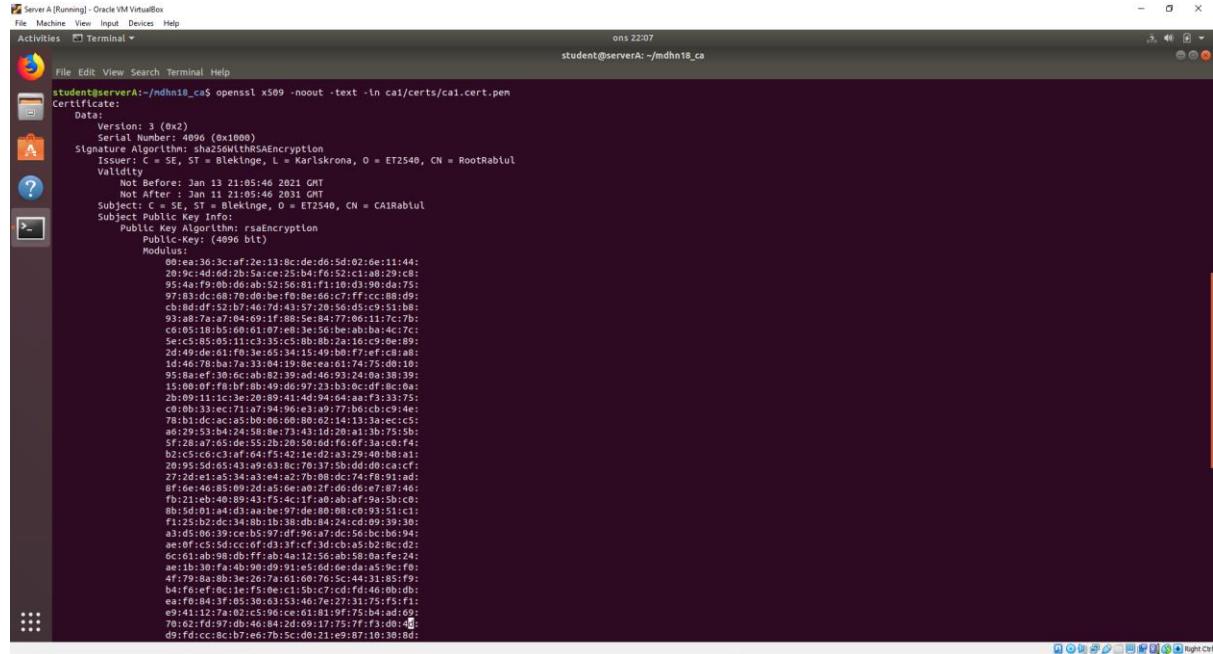
an input filename containing a single certificate request to be signed by the CA.

### **-out ca1/certs/ca1.cert.pem**

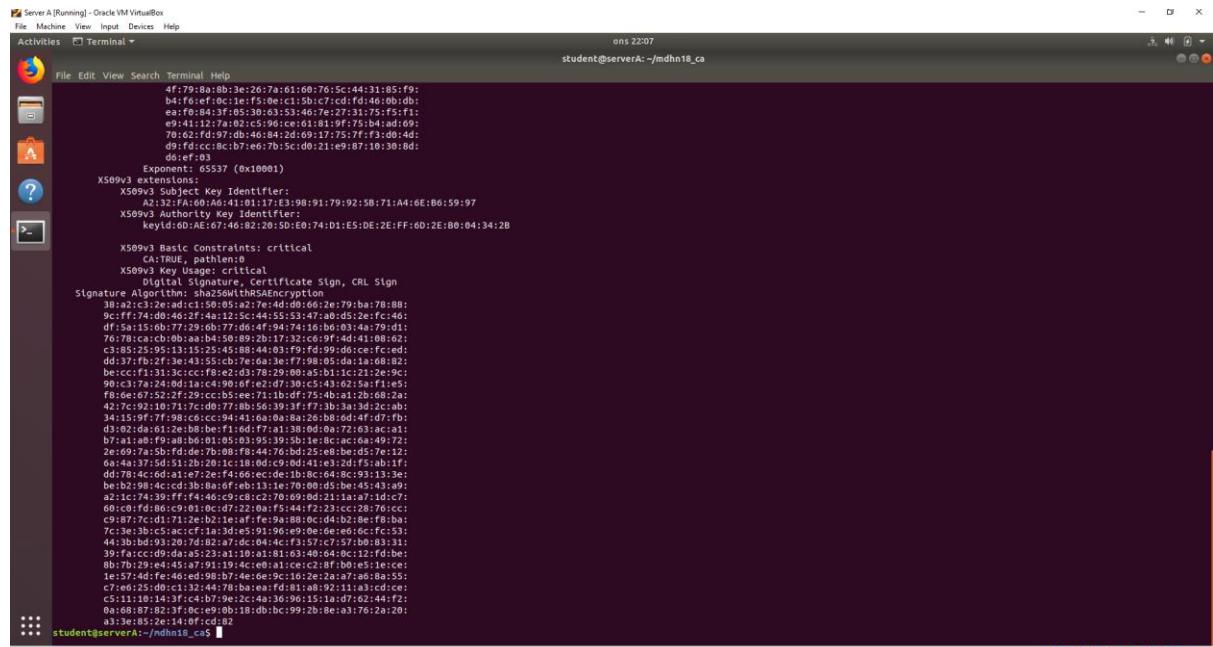
the output file to output certificates to. The default is standard output. The certificate details will also be printed out to this file in PEM format (except that -spkac outputs DER format).

## Task 10: Verify the certificate for CA1

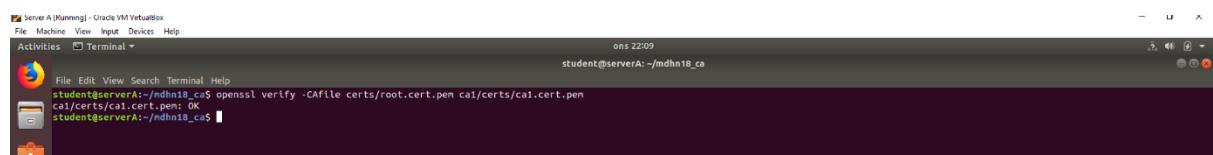
- Verify the certificate for CA1, first command prints the cert and the second verifies the CA1's certificate authenticity against the root certificate.



```
student@serverA:~/mdhn18_ca5 openssl x509 -noout -text -in ca1/certs/ca1.cert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = RootRabilul
        Validity
            Not Before: Jan 13 21:05:46 2021 GMT
            Not After : Jan 11 21:05:46 2031 GMT
        Subject: C = SE, ST = Blekinge, O = ET2540, CN = CA1Rabilul
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                    Modulus:
                        00:ea:3b:c4:ff:2e:1b:8c:d1:4d:5d:09:6e:11:45:
                        a9:9d:4d:ed:2b:1c:25:b4:f0:52:c2:1e:29:a5:75:
                        95:4a:f9:b6:ab:52:5a:b1:f1:10:d1:9a:8a:75:
                        97:83:dc:68:70:7d:be:f0:66:c7:ff:cc:08:09:
                    cb:8d:df:52:7d:43:57:20:56:5d:cb:91:51:b3:
                    93:a8:7a:7d:7a:04:9f:1f:88:5e:84:77:06:11:7c:7b:
                    e0:96:41:1f:21:3a:43:4a:82:40:30:1e:08:08:
                    5e:c5:85:b5:11:c3:35:c5:50:b2:2a:16:c9:08:89:
                    2d:49:de:61:f0:3e:65:34:15:49:bf:fe:f1:c8:a8:
                    1d:46:78:be:a7:33:04:19:88:e1:74:75:08:10:
                    95:8a:ef:30:0c:ab:82:39:ad:46:93:24:01:38:39:
                    15:08:0f:f8:5b:0b:8d:07:34:30:9d:0f:8c:08:
                    20:3b:54:7d:5b:50:80:41:94:64:f5:f3:71:55:
                    c0:0b:33:ec:c7:1a:79:4a:96:3a:a9:77:bc:0c:c9:4e:
                    78:b1:dc:ac:a5:b6:06:66:02:14:13:3a:ec:c5:
                    a6:29:53:bd:12:4:58:8e:73:43:1d:20:a1:3b:75:5b:
                    37:03:62:38:43:45:51:53:1a:3f:5b:1d:2b:f4:
                    b1:c5:1c:ec:af:64:f5:21:1e:42:43:29:40:48:11:
                    20:95:5d:65:43:a9:e3:bc:70:37:5b:dd:0c:a1:c1:
                    27:2d:ea:1a:53:a3:a3:ed:ea:27:b8:0b:dc:74:fb:91:ad:
                    bf:6e:40:85:09:2d:a5:0e:02:2f:0d:de:ea:78:7:46:
                    fb:21:eb:a8:89:43:f5:4c:1f:f8:ab:a8:9a:5b:c8:
                    a0:3c:19:25:5d:03:36:1d:21:7a:1c:51:20:11:
                    f1:25:b2:2d:c3:34:8b:1b:38:0b:84:24:cd:c9:39:30:
                    a3:d5:06:39:cce:b5:97:df:96:7a:dc:56:bc:b6:94:
                    ae:0f:cc:5d:cc:0f:d3:37:cfc:3d:ca:b5:b2:c1:cd:
                    cc:61:ab:10:0b:ff:ab:4a:02:12:0f:5b:08:0e:24:
                    ac:63:0d:08:0c:0e:1d:3e:0d:66:0d:0d:60:0e:40:
                    4f:79:8a:0b:3e:26:74:a1:60:76:5c:44:31:85:f9:
                    b4:f6:ef:0c:1e:f5:0e:c1:c7:cdf:fd:40:0b:db:
                    ea:04:84:3f:05:30:03:53:46:7e:27:31:75:f5:f1:
                    e9:41:12:7a:a0:c5:96:ce:a1:81:9f:75:1b:42:ad:69:
                    f0:02:10:57:00:40:64:20:09:17:73:01:3:80:1q:
                    d9:fd:cc:8c:b7:ea:7b:5c:00:21:e9:87:10:38:6d:
```



```
student@serverA:~/mdhn18_ca5 openssl verify -CAfile certs/root.cert.pem ca1/certs/ca1.cert.pem
ca1/certs/ca1.cert.pem: OK
student@serverA:~/mdhn18_ca5
```

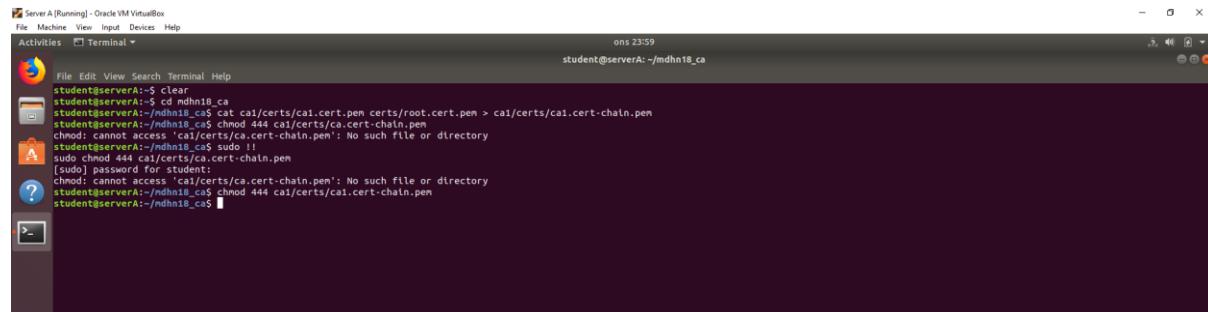


```
student@serverA:~/mdhn18_ca5 openssl verify -CAfile certs/root.cert.pem ca1/certs/ca1.cert.pem
ca1/certs/ca1.cert.pem: OK
student@serverA:~/mdhn18_ca5
```

## Task 11: Create server certificate

Create a certificate chain.

Ensure the certificate chain is accessible to all users on Server A



```
student@serverA:~/mdhn18_ca$ clear
student@serverA:~/mdhn18_ca$ cd mdhn18/ca1
student@serverA:~/mdhn18/ca1$ cat ca1.cert.pem root.cert.pem > ca1/certs/ca1.cert-chain.pem
student@serverA:~/mdhn18/ca1$ chmod 444 ca1/certs/ca1.cert-chain.pem
chmod: cannot access 'ca1/certs/ca1.cert-chain.pem': No such file or directory
student@serverA:~/mdhn18/ca1$ sudo !!
sudo chmod 444 ca1/certs/ca1.cert-chain.pem
[sudo] password for student:
sudo: unable to access '/ca1/certs/ca1.cert-chain.pem': No such file or directory
student@serverA:~/mdhn18/ca1$ chmod 444 ca1/certs/ca1.cert-chain.pem
student@serverA:~/mdhn18/ca1$
```

### Content of the CSR

To solve this task, I follow those steps. All screen shorts are the evidence to show, did this task successfully.

#### Steps

1. Create an RSA private key for the server.

```
openssl genrsa -out ca1/private/ca1server.key.pem 2048
```

2. Generate a CSR using the RSA private key from the previous step.

```
openssl req -config ca1/openssl.cnf -new -key ca1/private/ca1server.key.pem -out
ca1/csr/ca1server.csr.pem
```

3. Content of the CSR

```
openssl req -noout -text -in ca1/csr/ca1server.csr.pem
```

4. Use CA1's private key to sign the CSR and create a certificate for the server.

```
openssl ca -config ca1/openssl.cnf -extensions server_cert -days 3650 -notext -in
ca1/csr/ca1server.csr.pem -out ca1/certs/ca1server.cert.pem
```

5. Content of the signed certificate.

```
openssl x509 -noout -text -in ca1/certs/ca1server.cert.pem
```

6. Verify the server certificate against certificate chain.

```
openssl verify -CAfile ca1/certs/ca1.cert-chain.pem ca1/certs/ca1server.cert.pem
```

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 00:17
student@serverA: ~/mdhn18_ca
student@serverA:~/mdhn18_ca$ openssl genrsa -out ca1/private/caiserver.key.pem 2048
Generating RSA private key, 2048 bit long modulus
....+
e is 65537 (0x10001)
student@serverA:~/mdhn18_ca$ openssl req -config ca1/openssl.cnf -new -key ca1/private/caiserver.key.pem -out ca1/csr/caiserver.csr.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]:SE
State or Province Name (full name) [Some-State]:Blekinge
Locality Name (eg, city) [Karlskrona]:Karlskrona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ET2540
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
student@serverA:~/mdhn18_ca$ openssl req -nouot -text -in ca1/csr/caiserver.csr.pem
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = localhost
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:d0:02:d9:9e:3e:91:d6:5a:17:cd:0e:d7:5f:6d:
d2:90:eb:s1:97:14:e8:9a:a7:c8:8a:8e:31:14:3:e5:
e3:1f:4e:34:60:33:fb:d2:a1:f:a8:ee:81:3:e:ba:
be:f3:db:a4:60:c7:74:24:27:06:c3:46:0e:08:b1:93:2e:
84:6a:27:5f:ad:f6:26:47:7f:66:9d:dc:11:81:de:f4:
02:86:53:dc:c0:b4:1a:bf:df:f3:2a:41:81:1:6a:
70:9f:fb:40:1b:5c:56:f7:45:04:7c:0e:35:54:62:
3d:59:03:ca:73:1b:5d:85:5d:21:89:12:70:56:35:
73:4c:92:f7:35:c1:8b:dd:fc:9d:86:06:d9:03:18:
0e:2b:9b:99:01:3c:25:33:cc:33:47:70:17:ff:bf:
0f:2b:0f:45:a1:b6:9d:de:95:35:50:b1:ee:89:12:
f0:3d:d9:1c:33:52:65:ds:30:98:6f:51:ee:43:2:a3:
e7:48:7f:ca:8a:7d:0b:9f:1c:4b:ea:ef:4f:2:ha:
ca:31
Exponent: 65537 (0x10001)
Attributes:
a1:00
Signature Algorithm: sha256WithRSAEncryption

```

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 00:17
student@serverA: ~/mdhn18_ca
student@serverA:~/mdhn18_ca$ openssl req -nouot -text -in ca1/csr/caiserver.csr.pem
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = localhost
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:d0:02:d9:9e:3e:91:d6:5a:17:cd:0e:d7:5f:6d:
d2:90:eb:s1:97:14:e8:9a:a7:c8:8a:8e:31:14:3:e5:
e3:1f:4e:34:60:33:fb:d2:a1:f:a8:ee:81:3:e:ba:
be:f3:db:a4:60:c7:74:24:27:06:c3:46:0e:08:b1:93:2e:
84:6a:27:5f:ad:f6:26:47:7f:66:9d:dc:11:81:de:f4:
02:86:53:dc:c0:b4:1a:bf:df:f3:2a:41:81:1:6a:
70:9f:fb:40:1b:5c:56:f7:45:04:7c:0e:35:54:62:
3d:59:03:ca:73:1b:5d:85:5d:21:89:12:70:56:35:
73:4c:92:f7:35:c1:8b:dd:fc:9d:86:06:d9:03:18:
0e:2b:9b:99:01:3c:25:33:cc:33:47:70:17:ff:bf:
0f:2b:0f:45:a1:b6:9d:de:95:35:50:b1:ee:89:12:
f0:3d:d9:1c:33:52:65:ds:30:98:6f:51:ee:43:2:a3:
e7:48:7f:ca:8a:7d:0b:9f:1c:4b:ea:ef:4f:2:ha:
ca:31
Exponent: 65537 (0x10001)
Attributes:
a1:00
Signature Algorithm: sha256WithRSAEncryption

```

```
Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 00:17
student@serverA: ~/mdhn18_ca5
student@serverA:~/mdhn18_ca5$ openssl req -nouot -text -in ca1/csr/caiserver.csr.pem
Certificate Request
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = localhost
Subject Public Key Info:
    Public-KeyAlgorithm: rsaEncryption
    Public-Key: (2048 bit)
        Modulus:
            00:d0:02:d9:9e:3e:91:d6:5d:17:cd:0e:d7:5f:6d:
            d2:90:eb:s1:97:14:e8:9a:a7:c0:8a:8e:31:14:3e:51:
            58:3c:08:6f:8d:39:85:97:14:56:0c:04:2c:0e:35:34:62:
            c9:0e:37:79:b8:97:f3:82:72:5f:41:32:c2:64:ab:
        bits: 2048
        exponent: 65537 (0x10001)
Attributes:
a0:00
Signature Algorithm: sha256WithRSAEncryption
-----BEGIN CERTIFICATE REQUEST-----
MIGfMA0GCSqGSIb3DQEBCwQAEgCzB48Iw0C8cGz23:cb:Kc:31:
99:84:9b:17:7f:77:85:D9:73:b1:c6:78:80:b1:b3:a7:9e:86:
B5:37:0F:ee:76:b2:a5:47:5a:71:ab:3b:7f:1b:81:66:08:55:
63:ea:a4:f4:97:8:c3:31:52:ed:eb:a2:a:f:b:c9:98:37:fb:c0:1d:
f4:b3:a8:60:39:ea:14:04:d6:8:a:c7:7:cb:a4:dc:7d:76:c0:51:
01:00:9b:79:ca:93:da:01:81:f4:93:0a:7d:41:32:23:02:89:ad:
D0:35:6c:23:22:29:c1:31:58:50:39:34:41:32:19:66:25:9c:
08:id:76:40:33:hb:2:a6:3c:b3:3d:72:61:9e:da:0e:08:c0:8f:
a0:98:25:dc:14:63:a4:f4:45:1c:8a:25:f:22:59:09:46:45:f3:
13:88:02:18:7c:57:05:7f:ee:3d:bc:18:9c:f5:cd:fb:2a:52:
09:f2:41:10:9e:0d:4c:17:95:49:0b:fa:3a:72:e5:
5c:31:6c:64:09:c9:02:60:c0:8a:51:31:17:43:67:05:55:
ca:7f:e9:04:db:9b:e4:93:fd:9e:89:60:fd:24:2a:1c:f1:3b:
9a:03:13:54:64:50:65:8f:58:f4:e8:f2:75:be:ad:ce:0f:c9:
9b:a5:ee:4d
-----END CERTIFICATE REQUEST-----
student@serverA:~/mdhn18_ca5
```

```
Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 00:53
student@serverA: ~/mdhn18_ca5
student@serverA:~/mdhn18_ca5$ openssl ca -config ca1openssl.cnf -extensions server_cert -days 3650 -notext -in ca1/csr/caiserver.csr.pem -out ca1/certs/caiserver.cert.pem
Using configuration from /home/student/mdhn18_ca/ca1/openssl.cnf
Enter pass phrase for /home/student/mdhn18_ca/ca1/private/ca1.key.pem:
Can't open '/home/student/mdhn18_ca/ca1/index.txt.attr' for reading, No such file or directory
-----[REDACTED]-----
Check that the request matches the signature
Signature ok
-----[REDACTED]-----
Certificate Details:
    Serial Number: 8192 (0x2000)
    Validity
        Not Before: Jan 13 23:52:44 2021 GRT
        Not After : Jan 11 23:52:44 2031 GRT
    Subject:
        countryName          = SE
        stateOrProvinceName = Blekinge
        localityName        = Karlskrona
        organizationName   = ET2540
        commonName          = localhost
    X509v3 Extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Subject Key Identifier:
            E2:c4:45:85:D9:3B:53:70:1f:C0:A4:9A:AB:AB:A0:15:6F:97:C8:62
        X509v3 Authority Key Identifier:
            keyId:A2:32:FA:00:60:41:01:17:E3:98:91:79:93:9B:71:A4:66:80:59:97
            DirName:/etc/ssl/certs/localhost/C=SE/L=Karlskrona/O=ET2540/CN=RootRahul
            serial:10:00
        X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
        X509v3 Extended Key Usage:
            TLS Web Server Authentication
-----[REDACTED]-----
Certificate is new, not yet certified until Jan 11 23:52:44 2031 GRT (3650 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
Data Base Updated
student@serverA:~/mdhn18_ca5
```

```

student@serverA:~/mdhn18_ca$ openssl x509 -noout -text < ca1/certs/ca1server.cert.pem
-----  

Certificate:  

  Version: 3 (0x9)  

  Serial Number: 8192 (0x2000)  

  Signature Algorithm: sha256WithRSAEncryption  

  Issuer: C = SE, ST = Blekinge, O = ET2540, CN = CAIRabilul  

  Validity  

    Not Before: Jan 13 23:52:44 2021 GMT  

    Not After : Jan 11 23:52:44 2031 GMT  

  Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = localhost  

  Subject Public Key Info:  

    Public Key Algorithm: rsaEncryption  

      Public Key: (2048 bit)  

        Modulus:  

          00:d0:b2:02:id:91:9c:3c:91:0d:1f:cd:0c:id:f2:0f:cd:  

          42:80:31:55:3a:95:22:34:7d:0e:62:2a:9f:a8:ed:08:1:3:e:be:  

          e3:1f:4e:34:00:33:fb:d2:2a:9f:a8:ed:08:1:3:e:be:  

          c9:0e:37:79:08:97:f3:a2:72:5f:41:37:c2:04:ab:  

          9e:09:1c:07:3e:49:96:8d:2a:6d:01:9f:14:18:01:3:  

          b4:6a:27:5f:ad:f6:2d:47:7f:66:9d:dc:1:0:de:f6:  

          02:86:53:dc:0:b4:1a:b1:df:3f:2a:41:81:1:3:d:6a:  

          78:3d:99:5e:19:5e:2d:3f:2a:41:81:1:3:d:6a:  

          5e:05:4b:08:f4:0b:06:56:f7:05:94:2c:96:35:54:02:  

          73:4c:42:f7:35:c1:b8:dd:fc:9d:80:0e:ad:03:18:  

          9c:5b:42:28:23:71:3e:44:5e:11:3b:3c:11:3b:3c:  

          8f:0b:0f:45:1a:b0:9d:de:05:35:50:b3:re:9:09:12:  

          f0:3d:d9:1c:13:33:52:05:05:30:98:0f:51:e4:32:a3:  

          f0:7d:69:20:3b:3c:02:4b:1e:1c:01:41:c8:26:59:98:  

          b0:c7:79:9c:67:e2:51:1b:0e:41:c8:26:59:98:  

          6f:8a:99:bd:4:88:33:3b:71:13:c1:c6:60:94:40:ab:61:  

          0:19:f9:91:41:2e:55:3:a:b:dc:4e:45:ec:d7:07:  

          ca:31  

        Exponent: 05537 (0x10001)  

X509v3 extensions:  

  X509v3 Basic Constraints:  

    CA:FALSE  

  X509v3 Subject Key Identifier:  

    E2:4:45:85:D9:3B:53:7D:1F:C0:A4:9A:AB:AB:15:6F:97:C8:62  

  X509v3 Authority Key Identifier:  

    KeyId:02:32:F4:69:60:41:01:17:E3:98:91:79:92:58:71:A4:6EB6:59:97  

    DirName:/C=SE/ST=Blekinge/L=Karlskrona/O=ET2540/CN=RootRabilul  

  serial:10:00
-----
```

```

student@serverA:~/mdhn18_ca$ openssl verify -CAfile ca1/certs/ca1.cert-chain.pem ca1/certs/ca1server.cert.pem
ca1.cert: OK
student@serverA:~/mdhn18_ca$
```

## Task 12: Show your certificate in Firefox

Before that task start, I need.

Configure Apache web server to use the server certificate.

- Copy the certificate to the location, shown in a screen short.

```

student@serverA:~/mdhn18_ca$ sudo su
[sudo] password for student:
root@serverA:/home/student/mdhn18_ca# cp /home/student/mdhn18_ca/ca1/private/ca1server.key.pem /etc/ssl/private
root@serverA:/home/student/mdhn18_ca# cp /home/student/mdhn18_ca/ca1/certs/ca1server.cert.pem /etc/ssl/certs
root@serverA:/home/student/mdhn18_ca# cp /home/student/mdhn18_ca/ca1/certs/ca1.cert-chain.pem /etc/apache2
root@serverA:/home/student/mdhn18_ca# sudo gedit /etc/apache2/sites-enabled/default-ssl.conf
-----
** (gedit:2723): WARNING **: 01:12:19.510: Set document metadata failed: Setting attribute metadata::gedit-spell-language not supported
** (gedit:2723): WARNING **: 01:12:19.510: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported
** (gedit:2723): WARNING **: 01:12:23.712: Set document metadata failed: Setting attribute metadata::gedit-position not supported
root@serverA:/home/student/mdhn18_ca# sudo service apache2 restart
root@serverA:/home/student/mdhn18_ca#
```

- Edit the file /etc/apache2/sites-enabled/default-ssl.conf

```
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

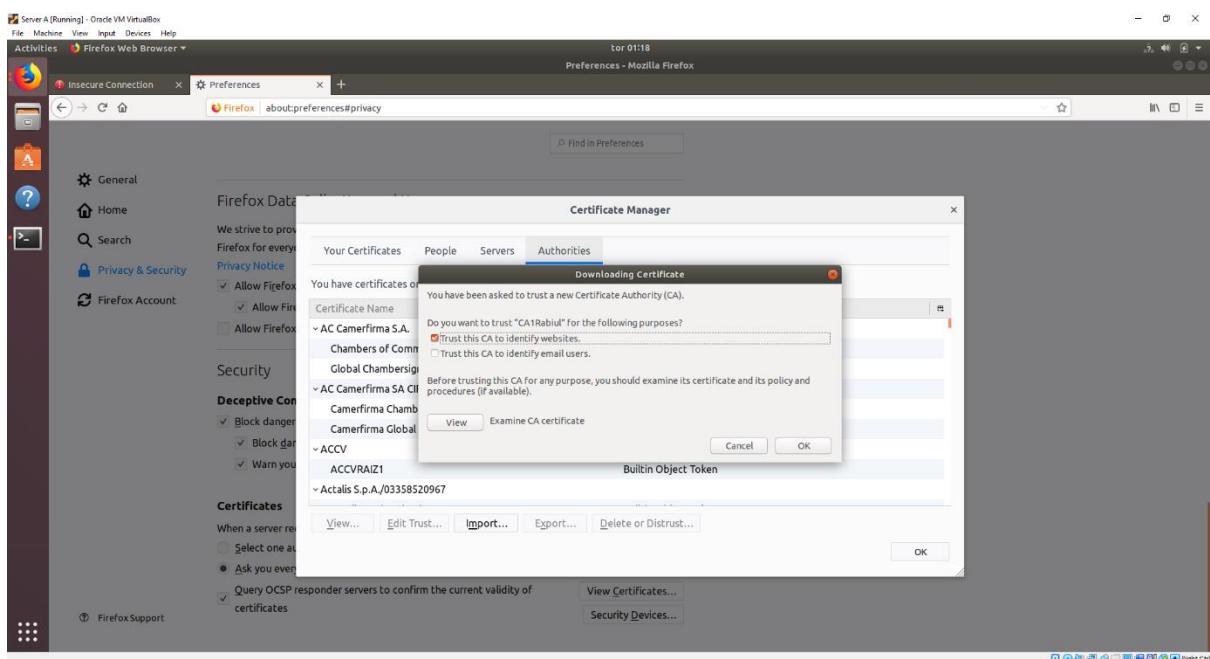
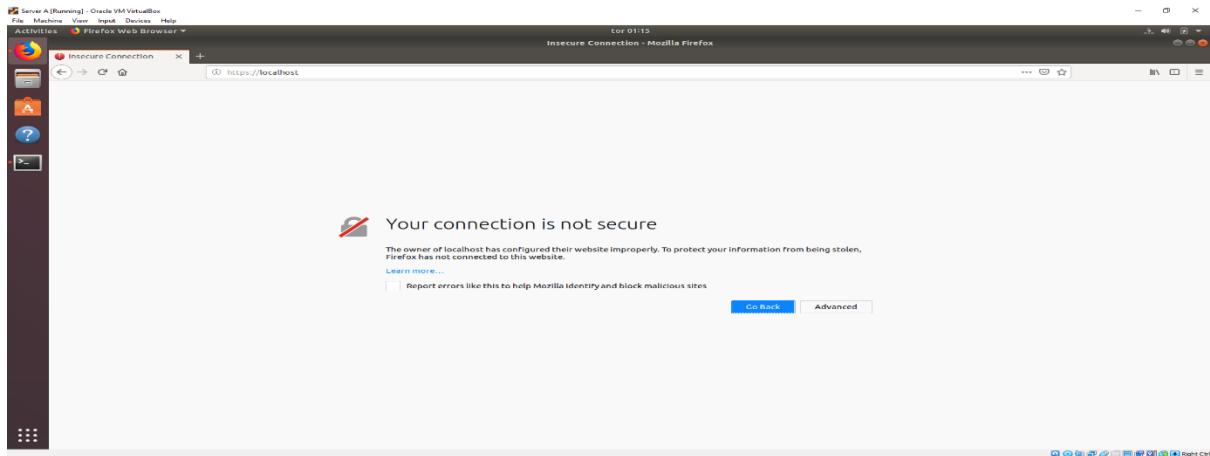
# SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

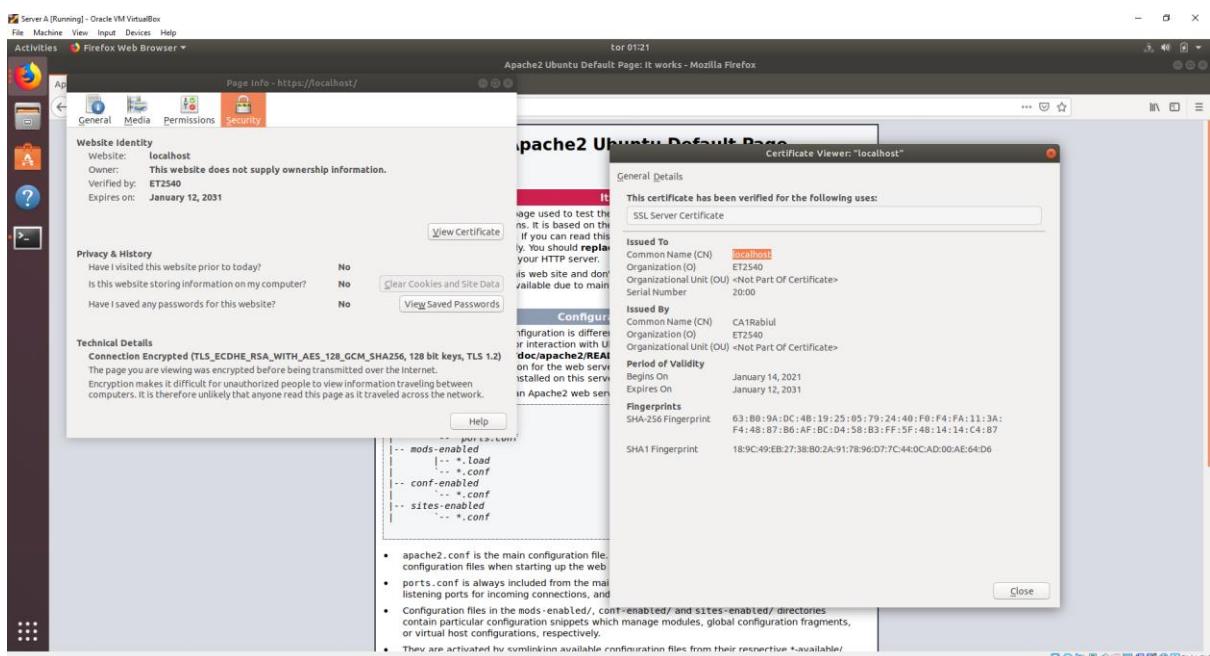
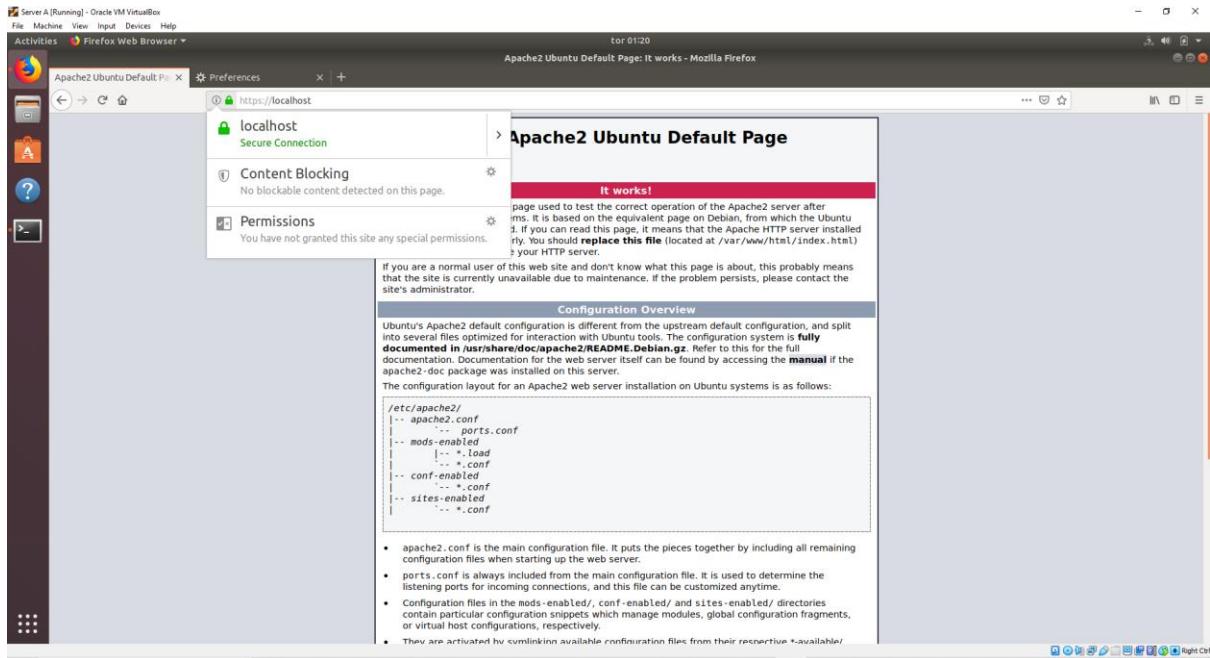
# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/ssl-cert/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
SSLCertificateFile /etc/ssl/certs/calserver.cert.pem
SSLCertificateKeyFile /etc/ssl/private/calserver.key.pem

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively the
# referenced file can be the same as SSLCertificateFile
# and the certificate file will be directly appended to the server
# certificate for convenience.
SSLCertificateChainFile /etc/apache2/cal.cert-chain.pem

# Certificate Authority (CA):
```

## Add certificate chain to Firefox web browser.





## Task 13: Create a CRL for CA1

Before that I edit the CA1 OpenSSL configuration file and add the following parameter to the serv\_cert section:

crlDistributionPoints = URI:https://localhost/ca1.crl.pem

- Following command to create a CRL for CA1  
**openssl ca -config ca1/openssl.cnf -gencrl -out ca1/crl/ca1.crl.pem**

- Dump the contents of the CRL using the following command and include the contents in the report:

```
openssl crl -in ca1/crl/ca1.crl.pem -noout -text
```

```
student@serverA:~/mdhn18.ca$ openssl ca -config ca1/openssl.cnf -gencrl -out ca1/crl/ca1.crl.pem
Using configuration from ca1/openssl.cnf
-----  

-----  

crl: Use -help for summary.  

student@serverA:~/mdhn18.ca$ openssl crl -in ca1/crl/ca1.crl.pem -noout -text  

-----  

-----  

Certificate Revocation List (CRL)  

-----  

Signature Algorithm: sha256WithRSAEncryption  

Issuer: /C=SE/ST=blekinge/O=ET2540/CN=CA1habitul  

Last Update: Jan 14 00:31:05 2021 GMT  

Next Update: Feb 13 00:31:05 2021 GMT  

CRL extensions:  

X509v3 CRL Number:  

8193  

No Revoked Certificates:  

-----  

Signature Algorithm: sha256WithRSAEncryption  

Issuer: /C=SE/ST=blekinge/O=ET2540/CN=CA1habitul  

Last Update: Jan 14 00:31:05 2021 GMT  

Next Update: Feb 13 00:31:05 2021 GMT  

CRL extensions:  

X509v3 CRL Number:  

8193  

-----  

student@serverA:~/mdhn18.ca$
```

## Task 14: Revoke a certificate

I follow those steps to complete this task, and all screen short provide as proof.

### Steps

- Create an RSA private key for the server.

```
openssl genrsa -out ca1/private/dragos.ilie@bth.se.key.pem 2048
```

- Generate a CSR using the RSA private key from the previous step.

```
openssl req -config ca1/openssl.cnf -new -key ca1/private/dragos.ilie@bth.se.key.pem -out ca1/csr/dragos.ilie@bth.se.csr.pem
```

- Content of the CSR.

```
openssl req -noout -text -in ca1/csr/dragos.ilie@bth.se.csr.pem
```

- Use CA1's private key to sign the CSR and create a certificate for the server.

```
openssl ca -config ca1/openssl.cnf -extensions usr_cert -days 3650 -notext -in ca1/csr/dragos.ilie@bth.se.csr.pem -out ca1/certs/dragos.ilie@bth.se.cert.pem
```

- Content of the signed certificate.

```
openssl x509 -noout -text -in ca1/certs/dragos.ilie@bth.se.cert.pem
```

- Verify the server certificate against certificate chain.

```
openssl verify -CAfile ca1/certs/ca1.cert-chain.pem ca1/certs/dragos.ilie@bth.se.cert.pem
```

student@serverA:~/mdhn18\_ca

```
File Edit View Search Terminal Help
Activities Terminal tor 01:46
student@serverA: ~/mdhn18_ca

student@serverA:~/mdhn18_ca$ openssl genrsa -out ca1/private/dragos.llie@bth.se.key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+
e is 65537 (0x100001)
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]:SE
State or Province Name (full name) [Some-State]:Blekinge
Locality Name (eg, city) [Karlskrona]:Karlskrona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ET2540
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:dragos.llie@bth.se
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
student@serverA:~/mdhn18_ca$ openssl req -newout -text -in ca1/csr/dragos.llie@bth.se.csr.pem
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = dragos.llie@bth.se
Subject Public Key Info:
Public-Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:d5:d1:0e:7d:0a:b1:d9:71:ea:b3:74:72:e2:a7:
07:20:c2:c5:57:9b:f3:3b:8b:12:c4:53:d1:48:65:
45:9c:6f:35:60:1a:45:38:13:1a:44:15:90:58:1a:49:
29:1c:dc:f7:be:f0:60:97:9b:49:0e:rd:41:bc:2a:
cc:bf:99:64:9b:05:0c:1a:52:52:63:0e:6e:25:15:
76:99:34:90:0f:0a:50:b1:76:87:e5:44:21:55:4b:
93:db:32:e0:4d:08:60:a3:a8:9d:f8:bf:0d:6e:4:53:
9d:c9:9d:9e:10:35:6a:f2:9e:63:81:1d:46:4a:48:
af:c9:60:a2:d1:03:93:f2:04:77:61:bd:f1:50:af:fd:
e1:b8:8c:8c:c7:59:6b:ice:43:c1:c1:26:1e:h5:95:
b6:1e:4a:f4:88:03:c6:a7:fb:2:95:38:c3:dc:b6:3b:
eb:d6:a5:b6:54:a5:F7:41:2d:iae:0d:10:34:c9:e8:
d1:28:f5:ce:4e:35:2f:7a:8d:ab:b3:c2:72:be:25:
d1:05:22:be:7e:10:55:ef:81:0d:24:2d:7b:8e:4a:
bd:77:9f:46:8d:e1:33:ice:87:f5:dc:8e:9c:9b:31:fc:
```

student@serverA:~/mdhn18\_ca

```
File Edit View Search Terminal Help
Activities Terminal tor 01:46
student@serverA: ~/mdhn18_ca

student@serverA:~/mdhn18_ca$ openssl genrsa -out ca1/private/ca1.key.pem 1024
Generating RSA private key, 1024 bit long modulus
.....+
e is 65537 (0x100001)
Attributes:
    a0:00
Signature Algorithm: sha256WithRSAEncryption
    0f:2f:8f:f4:51:b1:e0:ac:ad:16:c:f3:3b:51:35:bd:b3:0a:8b:
    88:81:6a:ed:68:83:91:91:6f:5e:50:81:81:81:81:81:81:81:81:
    4a:49:49:49:49:49:49:49:49:49:49:49:49:49:49:49:49:49:49:49:
    22:f0:57:d1:94:75:28:8d:74:9f:73:ec:c5:59:e7:bb:fc:7e:74:
    51:a6:2d:c2:07:de:a7:ed:e4:bd:13:09:a9:65:ff:53:3b:6d:
    25:e2:56:8d:80:c5:83:b2:72:d1:13:cfa:74:5:70:b7:
    be:3e:81:20:19:98:83:33:b2:96:83:c1:c0:7c:03:16:17:
    b9:03:8d:62:14:36:91:40:49:83:19:40:07:9f:08:21:11:f1:
    fb:15:ce:re:eb:40:eb:41:26:11:f1:86:2c:79:dd:12:5:2:5e:
    e3:ac:22:91:ca:95:03:24:a2:da:96:f4:aa:87:a8:11:09:f2:97:
    7:f5:cb:91:c8:cd:0f:f:fd:46:39:0d:75:87:24:5:f:dd:1e:1:67:67:
    f5:7a:3c:3c:7f:84:b3:77:5b:4c:3d:c3:2e:83:2b:ff:f0:1:7f:
    4e:2d:3d:2d:4d:2d:b1:f5:53:9b:05:0b:fc:73:67:0b:9f:28:79:
    30:ca:cd:30:59:00:24:51:as:3c:69:a7:2e:82:07:99:3e:c8:
    ec:7a:41:41:45

student@serverA:~/mdhn18_ca$ openssl ca -config ca1/openssl.cnf -extensions usr_cert -days 3650 -notext -in ca1/csr/dragos.llie@bth.se.csr.pem -out ca1/certs/dragos.llie@bth.se.cert.pem
Using configuration from ca1/openssl.cnf
Enter pass phrase for /home/student/mdhn18_ca/ca1/private/ca1.key.pem:
 aborted!
unable to load CA private key
140312339108288:error:PEM routines:PEM_do_header:bad password read:../crypto/pem/pem_lib.c:418:
student@serverA:~/mdhn18_ca$ openssl ca -config ca1/openssl.cnf -extensions usr_cert -days 3650 -notext -in ca1/csr/dragos.llie@bth.se.csr.pem -out ca1/certs/dragos.llie@bth.se.cert.pem
Enter pass phrase for /home/student/mdhn18_ca/ca1/private/ca1.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 8193 (0x2001)
    Validity
        Not Before: Jan 14 00:43:56 2021 GMT
        Not After : Jan 12 00:43:56 2031 GMT
    Subject:
        countryName      = SE
        stateOrProvinceName  = Blekinge
        localityName     = Karlskrona
        organizationName   = ET2540
        commonName        = dragos.llie@bth.se
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA=TRUE
```

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal ▾
student@serverA: ~/mdhn18_ca
student@serverA: ~/mdhn18_ca$ openssl x509 -noout -text -in ca1/certs/dragos.ilie@bth.se.cert.pem
Certificate:
    X509v3 Basic Constraints:
        CA:FALSE
    X509v3 Key Usage: critical
        Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Client Authentication, E-mail Protection
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        D1:19:0A:F0:A1:01:F1:15:21:AD:9B:8B:AB:F5:6F:2F:15:0F:35:87
    X509v3 Authority Key Identifier:
        keyid:D2:32:FA:60:A6:41:01:17:E3:9B:91:79:92:5B:71:A4:6E:B6:59:97
        Certificate is to be certified until Jan 12 00:43:56 2031 GMT (3650 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
Database successfully updated
student@serverA: ~/mdhn18_ca$ openssl x509 -noout -text -in ca1/certs/dragos.ilie@bth.se.cert.pem
Certificate:
    Version: 3 (0x2)
    Serial Number: 1193 (0x2001)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = SE, ST = Blekinge, O = ET2540, CN = Ca1Rabiu1
    Validity
        Not Before: Jan 14 00:43:56 2021 GMT
        Not After : Jan 13 2031 04:35:56 2031 GMT
    Subject: C = SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = dragos.ilie@bth.se
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
        [REDACTED]
        Exponent: 65537 (0x100001)
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Key Usage: critical
            Digital Signature, Non Repudiation, Key Encipherment
        X509v3 Extended Key Usage:
            TLS Web Client Authentication, E-mail Protection
        Netscape Comment:
            OpenSSL Generated certificate
        X509v3 Subject Key Identifier:
            D1:19:0A:F0:A1:01:F1:15:21:AD:9B:8B:AB:F5:6F:2F:15:0F:35:87
        X509v3 Authority Key Identifier:
            keyid:D2:32:FA:60:A6:41:01:17:E3:9B:91:79:92:5B:71:A4:6E:B6:59:97
        Signature Algorithm: sha256WithRSAEncryption

```

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal ▾
student@serverA: ~/mdhn18_ca
student@serverA: ~/mdhn18_ca$ 2>> /dev/null
student@serverA: ~/mdhn18_ca$ openssl x509 -noout -text -in ca1/certs/dragos.ilie@bth.se.cert.pem
Certificate:
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Key Usage: critical
            Digital Signature, Non Repudiation, Key Encipherment
        X509v3 Extended Key Usage:
            TLS Web Client Authentication, E-mail Protection
        Netscape Comment:
            OpenSSL Generated certificate
        X509v3 Subject Key Identifier:
            D1:19:0A:F0:A1:01:F1:15:21:AD:9B:8B:AB:F5:6F:2F:15:0F:35:87
        X509v3 Authority Key Identifier:
            keyid:D2:32:FA:60:A6:41:01:17:E3:9B:91:79:92:5B:71:A4:6E:B6:59:97
        Signature Algorithm: sha256WithRSAEncryption
student@serverA: ~/mdhn18_ca$ openssl verify -CAfile ca1/certs/ca1.cert-chain.pem ca1/certs/dragos.ilie@bth.se.cert.pem
ca1/certs/dragos.ilie@bth.se.cert.pem: OK
student@serverA: ~/mdhn18_ca$ 

```

- Now, revoke the newly created certificate by entering the following command:  
**openssl ca -config ca1/openssl.cnf -revoke ca1/certs/dragos.ilie@bth.se.cert.pem**

- Content of the file ca1/index.txt file

```
student@serverA:~$ openssl ca -config ca1/openssl.cnf -revoke ca1/certs/dragos.ilie@bth.se.cert.pem
Using configuration from /etc/ssl/certs/ca1/openssl.cnf
Enter pass phrase for /home/student/mdhn18_ca/ca1/private/ca1.key.pem:
Revoking Certificate 2001.
Data Base Updated.

student@serverA:~$ mdhn18_ca$ cat ca1/index.txt
-----[REDACTED]-----
Revoked Certificates:
Serial Number: 2001
    Revocation Date: Jan 14 01:04:02 2021
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=dragos.ilie@bth.se,CN=CA1RootCA,O=ET2540/CN=localhost
    Last Update: Jan 14 01:12:27 2021 GMT
    Next Update: Feb 13 01:12:27 2021 GMT
    CRL extensions:
        X509V3 CRL Number:
        [REDACTED]
    Revoked Certificates:
        Serial Number: 2001
        Revocation Date: Jan 14 01:04:02 2021
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=dragos.ilie@bth.se,CN=CA1RootCA,O=ET2540/CN=localhost
        Last Update: Jan 14 01:12:27 2021 GMT
        Next Update: Feb 13 01:12:27 2021 GMT
        CRL extensions:
            X509V3 CRL Number:
            [REDACTED]
[REDACTED]
```

## Task 15: Host-to-host transport mode VPN with PSK authentication

- For this task need to install by this command:  
**sudo apt-get install strongswan**

- The main configuration files for strongSwan are */etc/ipsec.conf* and */etc/ipsec.secrets*.

```
student@serverA:~$ sudo cat /etc/ipsec.conf
# ipsec.conf - strongSwan IPsec configuration file
#
# basic configuration
#
# config setup
#       strictcrlpolicy=yes
#       uniquid=no
#
# Add connections here.
#
# Sample VPN connections
#
#conn sample-self-signed
#  leftsubnet=192.168.0.0/16
#  leftcert=serverAselfcert.der
#  leftsendcert=never
#  right=192.168.0.2
#  rightsubnet=192.168.0/16
#  rightcert=peerCert.der
#  auto=start

#conn sample-with-ca-cert
#  leftsubnet=192.168.0.0/16
#  leftcert=serverAcert.pem
#  cert=peerCert.pem
#  right=192.168.0.2
#  rightsubnet=192.2.0.0/16
#  rightid="C=CH, O=Linux strongSwan CN=peer name"
#  auto=start

conn serverA-to-serverB
  auto=route
  authby=psk
  type=transport
  keyexchange=ikev2
  left=192.168.70.5
  right=192.168.70.6

student@serverA:~$ sudo cat /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

192.168.70.5 192.168.70.6 : PSK "student"

student@serverA:~$
```

```
student@serverB:~$ sudo cat /etc/ipsec.conf
# ipsec.conf - strongSwan IPsec configuration file
#
# basic configuration
#
# config setup
#       strictcrlpolicy=yes
#       uniquid=no
#
# Add connections here.
#
# Sample VPN connections
#
#conn sample-self-signed
#  leftsubnet=192.0.0/16
#  leftcert=serverBselfcert.der
#  leftsendcert=never
#  right=192.168.0.2
#  rightsubnet=192.2.0.0/16
#  rightcert=peerCert.der
#  auto=start

#conn sample-with-ca-cert
#  leftsubnet=192.0.0/16
#  leftcert=serverBcert.pem
#  cert=peerCert.pem
#  right=192.168.0.2
#  rightsubnet=192.2.0.0/16
#  rightid="C=CH, O=Linux strongSwan CN=peer name"
#  auto=start

conn serverB-to-serverA
  auto=route
  authby=psk
  type=transport
  keyexchange=ikev2
  left=192.168.70.6
  right=192.168.70.5

student@serverB:~$ sudo cat /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

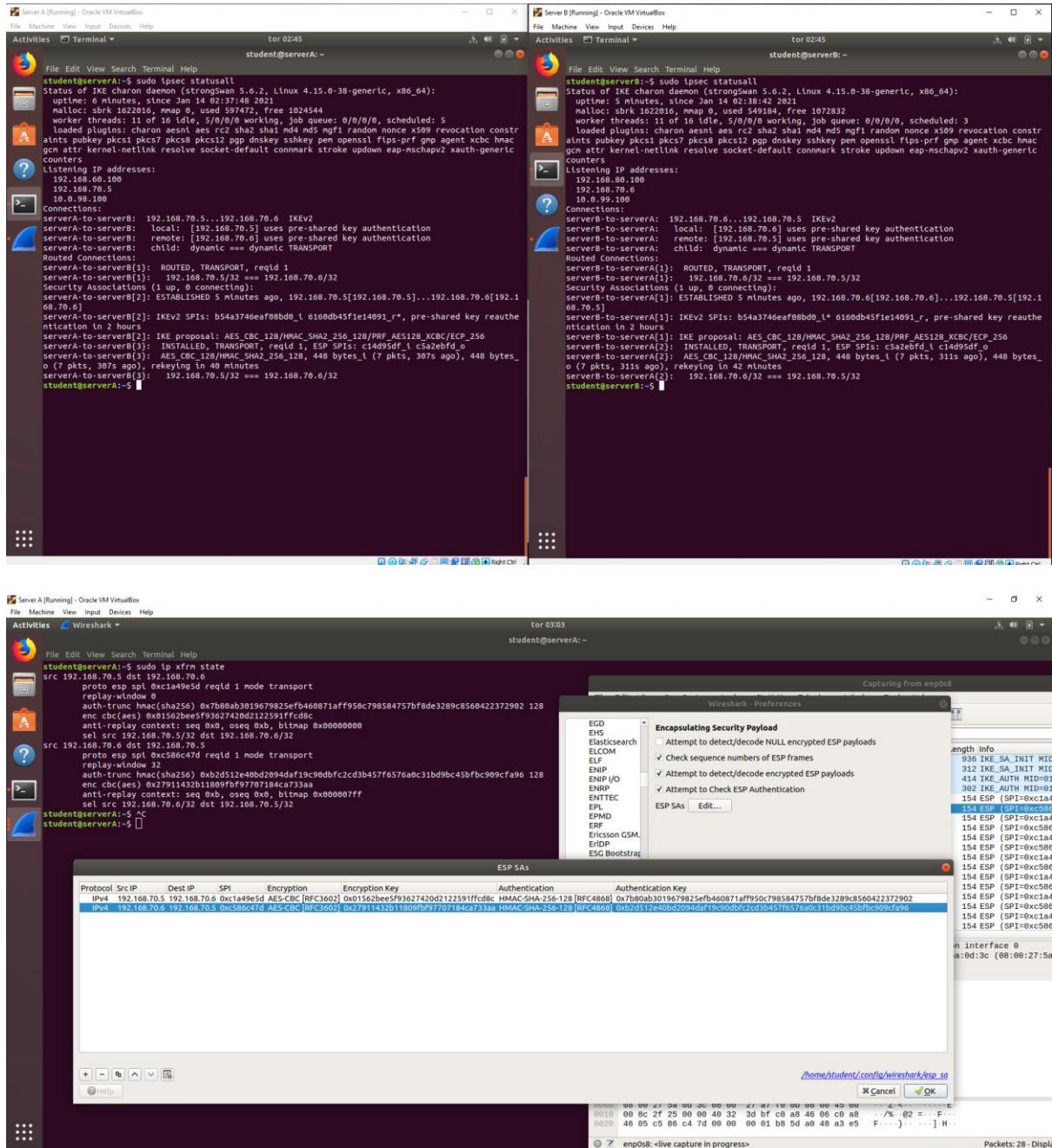
192.168.70.6 192.168.70.5 : PSK "student"

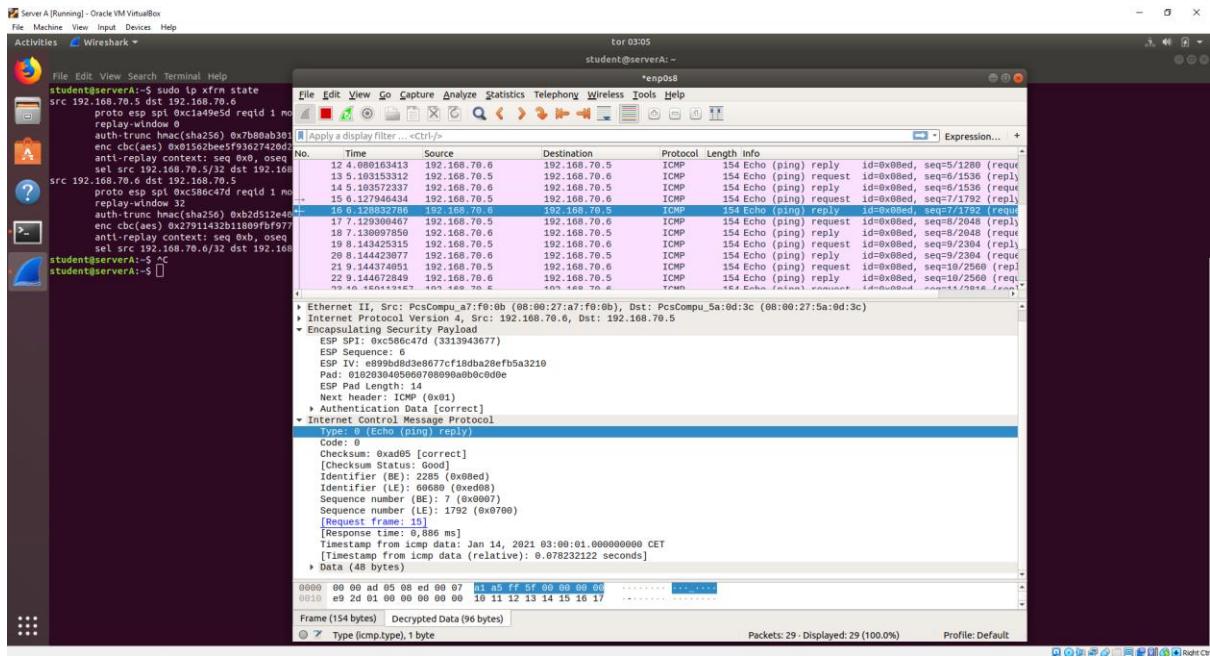
student@serverB:~$
```



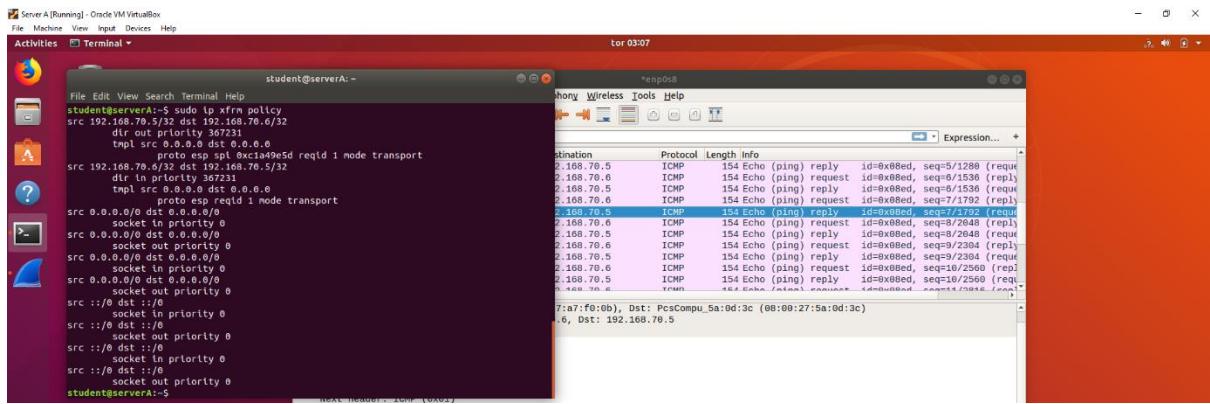
## Task 16: Decrypt traffic with Wireshark

- For this task, decrypt traffic with wireshark shown below screen shorts.





## Task 17: List the entries in the SPD



### Description in

It specifies the incoming SAs from ipsec.

### tmpl

It defines how ipsec should encapsulate matching packets.

### esp

It shows that the type of protocol was esp.

### reqid

It links the Security Policy Database (SPD) and the Security Association Database (SAD).

### transport

It shows that the mode of transport was transport.

## Task 18: Host-to-host transport mode VPN with cert authentication

- Before Host-to-host transport mode VPN with certificate authentication we need to create key for serverA and serverB.
- All screen short are shown those process step by step.
- Lastly verified all the certificates.

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 03:25
student@serverA: ~/mdhn18_ca
File Edit View Search Terminal Help
student@serverA: ~/mdhn18_ca$ openssl genrsa -out ca1/private/192.168.70.5.key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+
e is 65537 (0x100001)
student@serverA: ~/mdhn18_ca$ openssl req -config ca1/openssl.cnf -new -key ca1/private/192.168.70.5.key.pem -out ca1/csr/192.168.70.5.csr.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]:SE
State or Province Name (full name) [Some-State]:Blekinge
Locality Name (eg, city) []:Karlskrona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ET2540
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.70.5
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
X509 extensions []:
student@serverA: ~/mdhn18_ca$ openssl req -nouout -text -in ca1/csr/192.168.70.5.csr.pem
Certificate Request:
Data:
Version: 3 (0x80)
Subject: O = SE, ST = Blekinge, L = Karlskrona, C = ET2540, CN = 192.168.70.5
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
        .....
        Exponent: 65537 (0x100001)
Attributes:
    a0:00
Signature Algorithm: sha256WithRSAEncryption
7a:8d:12:12:a2:4e:d0:f8:1c:19:2a:5f:88:9f:9e:1b:b7:8c:
50:9d:12:12:a2:4e:d0:f8:1c:19:2a:5f:88:9f:9e:1b:b7:8c:
ad:ef:0f:bf:11:40:2f:f6:sb:dd:1e:10:8d:94:c2:d8:ba:de:
ca:sb:dd:09:10:3c:0a:40:b4:f4:c4:ce:7d:59:ca:16:5a:de:
7e:9d:0c:c4:9c:f8:1f:2d:ae:d2:04:53:85:87:10:21:fd:ba:
e:59:a1:4c:f7:1a:10:3c:0a:40:b4:f4:c4:ce:7d:59:ca:16:5a:de:
97:59:dc:bf:00:31:16:53:44:39:17:3a:37:8d:be:47:96:6d:
86:99:ea:8d:0f:1b:09:71:84:ca:1e:15:bb:45:2b:dd:cc:7d:70:
a2:9b:08:70
student@serverA: ~/mdhn18_ca$ openssl ca -config ca1/openssl.cnf -extensions server_cert -days 3650 -notext -in ca1/csr/192.168.70.5.csr.pem -out ca1/certs/192.168.70.5.cert.pem
Using configuration from /home/student/openssl.cnf
Enter pass phrase for /home/student/openssl.cnf:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 8194 (0x2002)
    Validity
        Not Before: Jan 14 02:22:08 2021 GMT
        Not After : Jan 12 02:22:08 2031 GMT
    Subject:
        countryName      = SE
        stateOrProvinceName = Blekinge
        localityName     = Karlskrona
        organizationName = ET2540
        commonName       = 192.168.70.5
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    X509v3 Subject Key Identifier:
        F1:C:E:7:C:AD:4D:1C:52:08:D9:72:3D:80:25:F7:56:6B:80:85:E3:F3
    X509v3 Authority Key Identifier:
        Keypair:13:32:64:4B:96:56:01:17:E5:09:01:70:03:0B:71:8A:6E:BA:49:07

```





```

student@serverA:~$ openssl verify -CAfile ca/certs/ca1.cert-chain.pem ca/certs/192.168.70.6.cert.pem
ca/certs/192.168.70.6.cert.pem: OK
student@serverA:~$ mdhn18_ca

```

- Distribute the certificate exact location in serverA and serverB.

```

student@serverA:~$ sudo ls /etc/ipsecd/private/
192.168.70.5.key.pem
student@serverA:~$ sudo ls /etc/ipsecd/certs/
192.168.70.5.cert.pem
student@serverA:~$ sudo ls /etc/ipsecd/cacerts/
ca1.cert.pem root.cert.pem
student@serverA:~$ 
student@serverB:~$ sudo ls /etc/ipsecd/private/
192.168.70.6.key.pem
student@serverB:~$ sudo ls /etc/ipsecd/certs/
192.168.70.6.cert.pem
student@serverB:~$ sudo ls /etc/ipsecd/cacerts/
ca1.cert.pem root.cert.pem
student@serverB:~$ 

```

- Edit the ipsec.conf and ipsec.secrets as shown in below:

```

# Sample VPN connections
#conn sample-self-signed
# leftsubnet=10.1.0.0/16
# leftcert=SelfCert.der
# leftscert=never
# rightsubnet=10.2.0.0/16
# rightcert=PeerCert.der
# auto=start
#
#conn sample-with-ca-cert
# leftsubnet=10.1.0.0/16
# leftcert=MyCert.pem
# right=192.168.0.2
# rightsubnet=10.2.0.0/16
# rightid="C=CH, O=Linux strongSwan CN=peer name"
# auto=start
#
#conn %default
#   ikelifetime=60m
#   keylife=20m
#   rekeymargin=3m
#   keytries=3
#   nobike=no
#   keyexchange=ikev2
#
#conn serverA-to-serverB-transport
# left=192.168.70.5
# leftcert=192.168.70.5.cert.pem
# leftid="C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5"
# leftscert=never
# right=192.168.70.6
# rightid="C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6"
# type=transport
# authby=rsa
# auto=route
#

```

```

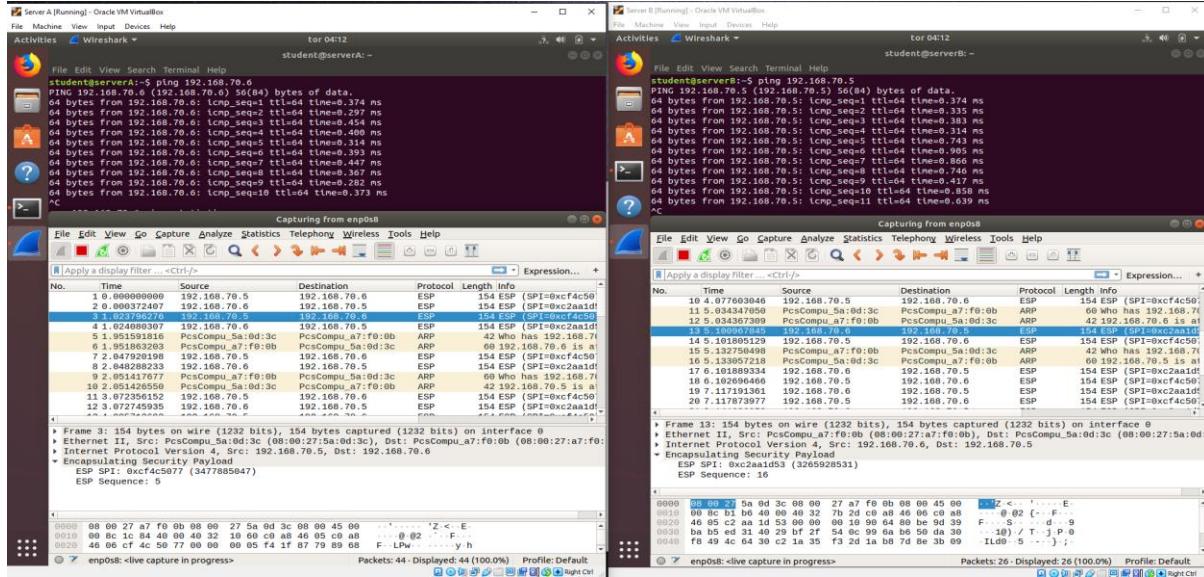
# This file holds shared secrets or RSA private keys for authentication.
# RSA private key for this host, authenticating it to any other host
# which knows the public part.
6 192.168.70.5 192.168.70.6 : PSK "student"
7 : RSA 192.168.70.5.key.pem

```



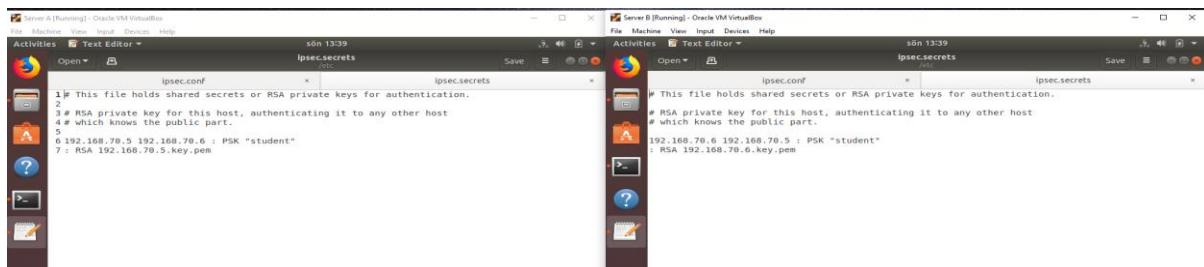
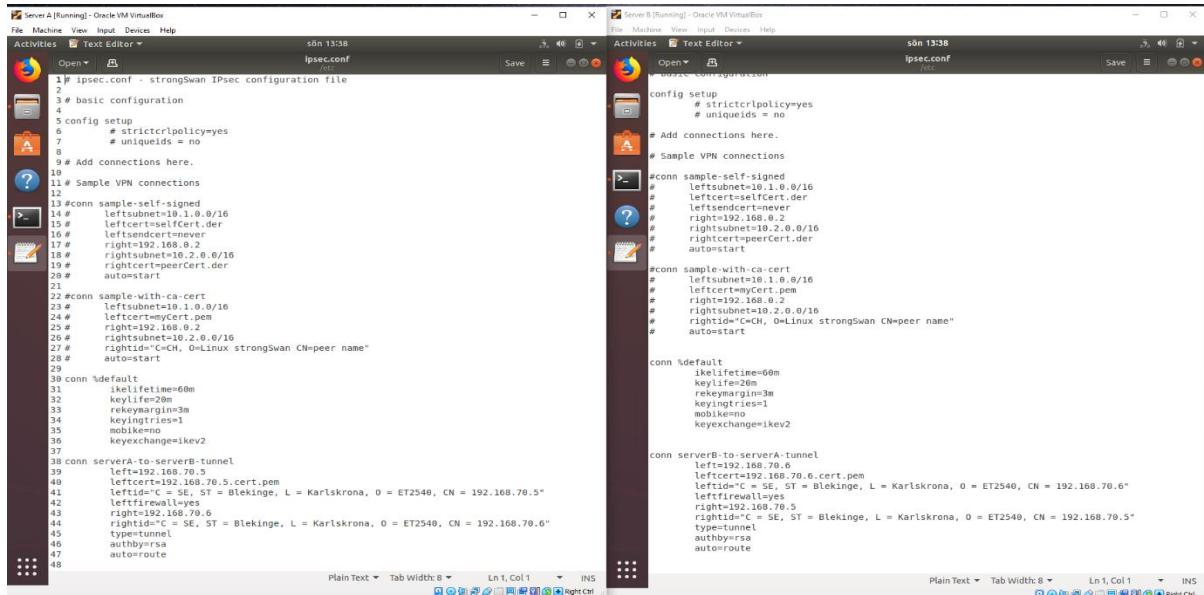


- Successfully ping serverA to server and server to serverA with wireshark traffic.



## Task 19: Tunnel mode VPN with cert authentication between Server A and Server B

- For this task I edit again ipsec.conf and ipsec.secrets.



- Tunnel mode VPN with cert authentication between Server A and Server B established successfully.

```

student@serverA:~$ sudo ipsec up serverA-to-serverB-tunnel
introducing IKE_SA serverA-to-serverB[1] to 192.168.70.6
generating CERT_AUTH_SA request 0 [ SA No TSL TsR ]
sending packet: from 192.168.70.5[500] to 192.168.70.6[500] (256 bytes)
received packet: from 192.168.70.6[500] to 192.168.70.5[500] (208 bytes)
parsed CERT_AUTH_SA response 0 [ SA No TSL TsR ]
IKE_SA serverA-to-serverB[1] established with SPIs ca4b6004_1 cb06f8df_o and TS 192.168.70.6/32
connection 'serverA-to-serverB-tunnel' established successfully
student@serverA:~$ 

student@serverB:~$ sudo ipsec up serverB-to-serverA-tunnel
establishing CHILD_SA serverB-to-serverA-tunnel[3]
generating CREATE_CHILD_SA request 0 [ SA No TSL TsR ]
sending packet: from 192.168.70.5[500] to 192.168.70.6[500] (256 bytes)
received packet: from 192.168.70.6[500] to 192.168.70.5[500] (208 bytes)
parsed CREATE_CHILD_SA response 0 [ SA No TSL TsR ]
CHILD_SA serverB-to-serverA-tunnel[3] established with SPIs ca4b6004_1 cb06f8df_o and TS 192.168.70.5/32
connection 'serverB-to-serverA-tunnel' established successfully
student@serverB:~$ 

```

```

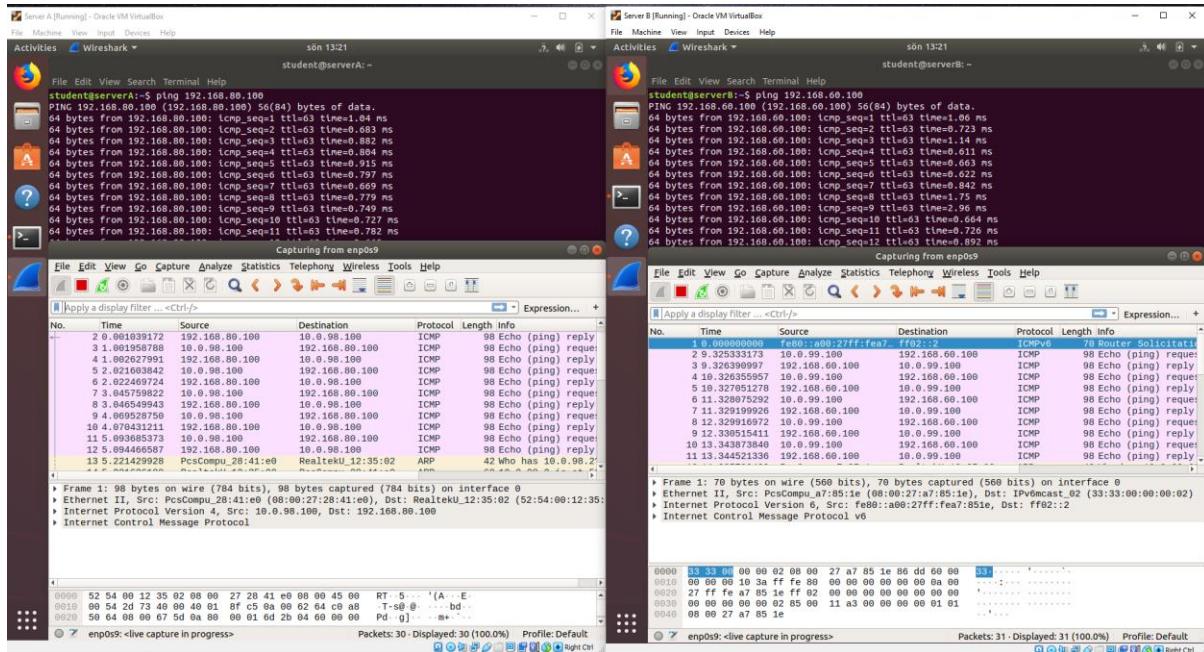
student@serverA:~$ sudo ipsec statusall
Status of IKE charon daemon (strongSwan 5.6.2, Linux 4.15.0-38-generic, x86_64):
    uptime: 7 minutes, since Jan 17 13:02:48 2021
    pid: 89184, uptime: 7 minutes
    worker threads: 11 of 16 idle, 5/0/0 working, job queue: 0/0/0, scheduled: 3
    loaded plugins: charon aesi nt5 aes rc2 sha2 sha1 md4 md5 md5f ngf1 random nonce x509 revocation constraints pubkey pkcs7 pkcs8 pkcs12 ppp dnskey sshkey pem openssl fips-prf gmp agent xcbc hmac gcm atri kernel-netlink resolve socket-default connmark stroke updown eap-mschapv2 xauth-generic counters:
    Listening IP addresses:
        192.168.68.100
        192.168.70.5
        10.0.99.100
    Connections:
    serverA-to-serverB-tunnel: 192.168.70.5...192.168.70.6 IKEV2
    serverA-to-serverB-tunnel: local: [C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5]
    uses public key authentication
    serverA-to-serverB-tunnel: cert: "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5"
    serverA-to-serverB-tunnel: remote: [C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6]
    serverA-to-serverB-tunnel: uses public key authentication
    serverA-to-serverB-tunnel: child: dynamic === dynamic TUNNEL
    Routed connections:
    serverA-to-serverB-tunnel(): ROUTED, TUNNEL, reqid 1
    serverA-to-serverB-tunnel(): 192.168.70.5/32 === 192.168.70.6/32
    serverA-to-serverB-tunnel(): 0 connected
    serverA-to-serverB-tunnel(): ESTABLISHED 6 minutes ago, 192.168.70.5[C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5]...192.168.70.6[C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6]
    serverA-to-serverB-tunnel(): IKEV2 SPIs: 32421880903c0238_* aa0794c4780e7Sea_r, public key reauthentication in 7 minutes
    serverA-to-serverB-tunnel(): IKE proposal: AES_CBC_128/HMAC_SHA2_256_128/PRF_AES128_XCB/ECP_256
    serverA-to-serverB-tunnel(): INSTALLED, TUNNEL, reqid 1, ESP SPIs: caa59e7_1 cbaf2fe2_o
    serverA-to-serverB-tunnel(): AES_CBC_128/HMAC_SHA2_256_128, 0 bytes_o, 0 bytes_o, rekeying in 9 minutes
    serverA-to-serverB-tunnel(): 192.168.70.5/32 == 192.168.70.6/32
    serverA-to-serverB-tunnel(): INSTALLED, TUNNEL, reqid 1, ESP SPIs: cb06f8df_i ca4b6004_o
    serverA-to-serverB-tunnel(): AES_CBC_128/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 8 minutes
    serverA-to-serverB-tunnel(): 192.168.70.5/32 == 192.168.70.6/32
student@serverA:~$ 

student@serverB:~$ sudo ipsec statusall
Status of IKE charon daemon (strongSwan 5.6.2, Linux 4.15.0-38-generic, x86_64):
    uptime: 7 minutes, since Jan 17 13:02:48 2021
    pid: 892736, uptime: 7 minutes
    worker threads: 11 of 16 idle, 5/0/0 working, job queue: 0/0/0, scheduled: 2
    loaded plugins: charon aesi nt5 aes rc2 sha2 sha1 md4 md5 md5f ngf1 random nonce x509 revocation constraints pubkey pkcs7 pkcs8 pkcs12 ppp dnskey sshkey pem openssl fips-prf gmp agent xcbc hmac gcm atri kernel-netlink resolve socket-default connmark stroke updown eap-mschapv2 xauth-generic
    counters:
    Listening IP addresses:
        192.168.88.100
        192.168.70.6
        10.0.99.100
    Connections:
    serverB-to-serverA-tunnel: 192.168.70.6...192.168.70.5 IKEV2
    serverB-to-serverA-tunnel: local: [C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6]
    uses public key authentication
    serverB-to-serverA-tunnel: cert: "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6"
    serverB-to-serverA-tunnel: remote: [C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5]
    serverB-to-serverA-tunnel: uses public key authentication
    serverB-to-serverA-tunnel: child: dynamic === dynamic TUNNEL
    Routed connections:
    serverB-to-serverA-tunnel(): ROUTED, TUNNEL, reqid 1
    serverB-to-serverA-tunnel(): 192.168.70.6/32 == 192.168.70.5/32
    security associations (1 up, 0 connecting):
    serverB-to-serverA-tunnel(): ESTABLISHED 6 minutes ago, 192.168.70.6[C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6]...192.168.70.5[C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5]
    serverB-to-serverA-tunnel(): IKEV2 SPIs: 32421880903c0238_* aa0794c4780e7Sea_r, public key reauthentication in 40 minutes
    serverB-to-serverA-tunnel(): 0 connected
    serverB-to-serverA-tunnel(): IKE proposal: AES_CBC_128/HMAC_SHA2_256_128/PRF_AES128_XCB/ECP_256
    serverB-to-serverA-tunnel(): INSTALLED, TUNNEL, reqid 1, ESP SPIs: cbaf2fe2_i caa59e7_o
    serverB-to-serverA-tunnel(): AES_CBC_128/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 7 minutes
    serverB-to-serverA-tunnel(): 192.168.70.6/32 == 192.168.70.5/32
    serverB-to-serverA-tunnel(): INSTALLED, TUNNEL, reqid 1, ESP SPIs: ca4b6004_i cb06f8df_o
    serverB-to-serverA-tunnel(): AES_CBC_128/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 7 minutes
    serverB-to-serverA-tunnel(): 192.168.70.6/32 == 192.168.70.5/32
student@serverB:~$ 

```



- Tested by pinging the private addresses of Server A (192.168.60.100) and Server B (192.168.80.100), respectively.



## Task 20: Tunnel mode VPN with IP forwarding for client A and client B

- For this task I edit ipsec.conf.

```

# /etc/ipsec.conf
# Sample VPN connections
conn sample-self-signed
  leftsubnet=10.1.0.0/16
  leftcert=selfcert.der
  rightcert=never
  right=192.168.0.2
  rightsubnet=10.2.0.0/16
  rightcert=peerCert.der
  auto=start

conn sample-with-ca-cert
  leftsubnet=10.1.0.0/16
  leftcert=myCert.pem
  right=192.168.0.2
  rightsubnet=10.2.0.0/16
  rightid="<CH, 0>Linux strongSwan CN=peer name"
  auto=start

conn default
  ikelifetime=60m
  keylife=20m
  rekeymargin=3m
  keyingtries=1
  mobike=no
  keyexchange=ikev2

conn serverA-to-serverB-tunnel
  left=192.168.70.5
  leftcert=192.168.70.5.cert.pem
  leftid="<SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = 192.168.70.5"
  leftsubnet=192.168.60.0/24
  leftfirewall=yes
  right=192.168.70.6
  rightid="<SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = 192.168.70.6"
  rightsubnet=192.168.80.0/24
  type=tunnel
  authby=rsa
  auto=route

conn serverB-to-serverA-tunnel
  left=192.168.70.6
  leftcert=192.168.70.6.cert.pem
  leftid="<SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = 192.168.70.6"
  leftsubnet=192.168.60.0/24
  leftfirewall=yes
  right=192.168.70.5
  rightid="<SE, ST = Blekinge, L = Karlskrona, O = ET2540, CN = 192.168.70.5"
  rightsubnet=192.168.80.0/24
  type=tunnel
  authby=rsa
  auto=route

```

- Main part of this task added firewall rules under the section “# Firewall rules for task 20”

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor tor 01:20
fw_lab2.sh
Save
24 #IPPT -t nat -j
25 # Delete any user-defined chains in NAT table
26 SIPT -t nat -X
27 # Flush all chains in MANGLE table
28 SIPT -t mangle -F
29 # Delete any user-defined chains in MANGLE table
30 SIPT -t mangle -X
31 # Flush all chains in RAW table
32 SIPT -t raw -F
33 # Delete any user-defined chains in RAW table
34 SIPT -t mangle -X
35
36 # Default policy is to send to a dropping chain
37 SIPT -t filter -P INPUT ACCEPT
38 SIPT -t filter -P OUTPUT ACCEPT
39 SIPT -t filter -P FORWARD ACCEPT
40
41
42
43 # Create logging chains
44 SIPT -t filter -N input_log
45 SIPT -t filter -N output_log
46 SIPT -t filter -N forward_log
47
48 # Set some logging targets for DROPPED packets
49 SIPT -t filter -A input_log -j LOG --log-level notice --log-prefix "input drop: "
50 SIPT -t filter -A output_log -j LOG --log-level notice --log-prefix "output drop: "
51 SIPT -t filter -A forward_log -j LOG --log-level notice --log-prefix "forward drop: "
52 echo "Added Logging"
53
54 # Return from the logging chain to the built-in chain
55 SIPT -t filter -A input_log -j RETURN
56 SIPT -t filter -A output_log -j RETURN
57 SIPT -t filter -A forward_log -j RETURN
58
59
60 # Firewall rules for task 20
61 SIPT -A INPUT -s 192.168.70.5 -d 192.168.70.6 -j ACCEPT
62 SIPT -A OUTPUT -s 192.168.70.6 -d 192.168.70.5 -j ACCEPT
63 SIPT -t filter -A FORWARD -l $HIF -j ACCEPT
64 SIPT -t filter -A FORWARD -l $INIF -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
65
66
67
68 # These rules must be inserted at the end of the built-in
69 # chain to log packets that will be dropped by the default
sh Tab Width: 8 Ln 60, Col 16 INS

```

```

Server B [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor tor 01:20
fw_lab2.sh
Save
24 #IPPT -t nat -j
25 # Delete any user-defined chains in NAT table
26 SIPT -t nat -X
27 # Flush all chains in MANGLE table
28 SIPT -t mangle -F
29 # Delete any user-defined chains in MANGLE table
30 SIPT -t mangle -X
31 # Flush all chains in RAW table
32 SIPT -t raw -F
33 # Delete any user-defined chains in RAW table
34 SIPT -t mangle -X
35
36 # Default policy is to send to a dropping chain
37 SIPT -t filter -P INPUT ACCEPT
38 SIPT -t filter -P OUTPUT ACCEPT
39 SIPT -t filter -P FORWARD ACCEPT
40
41
42
43 # Create logging chains
44 SIPT -t filter -N input_log
45 SIPT -t filter -N output_log
46 SIPT -t filter -N forward_log
47
48 # Set some logging targets for DROPPED packets
49 SIPT -t filter -A input_log -j LOG --log-level notice --log-prefix "input drop: "
50 SIPT -t filter -A output_log -j LOG --log-level notice --log-prefix "output drop: "
51 SIPT -t filter -A forward_log -j LOG --log-level notice --log-prefix "forward drop: "
52 echo "Added Logging"
53
54 # Return from the logging chain to the built-in chain
55 SIPT -t filter -A input_log -j RETURN
56 SIPT -t filter -A output_log -j RETURN
57 SIPT -t filter -A forward_log -j RETURN
58
59
60 # Firewall rules for task 20
61 SIPT -A INPUT -s 192.168.70.5 -d 192.168.70.6 -j ACCEPT
62 SIPT -A OUTPUT -s 192.168.70.6 -d 192.168.70.5 -j ACCEPT
63 SIPT -t filter -A FORWARD -l $HIF -j ACCEPT
64 SIPT -t filter -A FORWARD -l $INIF -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
65
66
67
68 # These rules must be inserted at the end of the built-in
69 # chain to log packets that will be dropped by the default
sh Tab Width: 8 Ln 60, Col 29 INS

```

```

Server A [Running] - Oracle VM VirtualBox
File Edit View Search Terminal Help
student@serverA:~$ sudo sysctl -w net.ipv4.ip_forward=1
student@serverA:~$ sudo ./fw_lab2.sh
Added Logging
student@serverA:~$ 

Server B [Running] - Oracle VM VirtualBox
File Edit View Search Terminal Help
student@serverB:~$ sudo sysctl -w net.ipv4.ip_forward=1
student@serverB:~$ sudo ./fw_lab2.sh
Added Logging
student@serverB:~$ 

```

```

Server A [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 01:22
student@serverA:~$ 
File Edit View Search Terminal Help
LG) N(MULT_AUTH)
received cert request for "C=SE, ST=Blekinge, O=ET2540, CN=CAIRabul"
received cert request for "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=RootRabul"
sending cert request for "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=RootRabul"
authentication of "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5" (myself) with RSA_EMSA_PKCS1_SHA2_256 successful
sending end entity cert "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5"
authorizing 'IKE AUTH request 1 [ IDI CERT SA TSL TSr N(MULT_AU
H) N(EAP ONLY) N(MSC_ID_SVNL_SUP) ]'
splitting IKE message with length of 2 bytes into 2 fragments
generating IKE_AUTH request 1 [ EF(1/2) ]
generating IKE_AUTH request 2 [ EF(2/2) ]
sending packet: from 192.168.70.5[500] to 192.168.70.6[500] (1252 bytes)
sending packet: from 192.168.70.6[500] to 192.168.70.5[500] (932 bytes)
received packet: from 192.168.70.6[500] to 192.168.70.5[500] (1252 bytes)
parsed IKE_AUTH response 1 [ EF(1/2) ]
received fragment #1 of 2, waiting for complete IKE message
received packet: from 192.168.70.6[500] to 192.168.70.5[500] (708 bytes)
parsed IKE_AUTH response 1 [ EF(2/2) ]
received Fragment #2 of 2, reassembling fragmented IKE message
parsed IKE_AUTH response 1 [ IDI CERT AUTH SA TSL TSr N(AUTH_LFT) ]
received end entity cert "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6"
using certificate "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6"
using certificate "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=CAIRabul"
checking certificate status of "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6"
fetching crl from 'https://localhost/ca1.crl.pem'
unable to fetch from https://localhost/ca1.crl.pem, no capable fetcher found
crl fetching failed
certificate status is not available
using self-signed certificate "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=RootRabul"
checking certificate status of "C=SE, ST=Blekinge, O=ET2540, CN=CAIRabul"
certificate status is not available
reached self-signed root ca with a path length of 1
authentication of "C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6" with RSA_EMSA_PKCS1_SHA2_256 successful
IKE_SA 'serverA-to-serverB-tunnel[1]' established between 192.168.70.5[C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.5]...192.168.70.6[C=SE, ST=Blekinge, L=Karlskrona, O=ET2540, CN=192.168.70.6]
scheduling reauthentication in 3344s
maximum IKE_SA lifetime 3524s
IKE_SA 'serverA-to-serverB-tunnel[2]' established with SPIs c55a261d_1 c916dd03_o and TS 192.168.66.0/24
connection 'serverA-to-serverB-tunnel' established successfully
student@serverA:~$ 

```

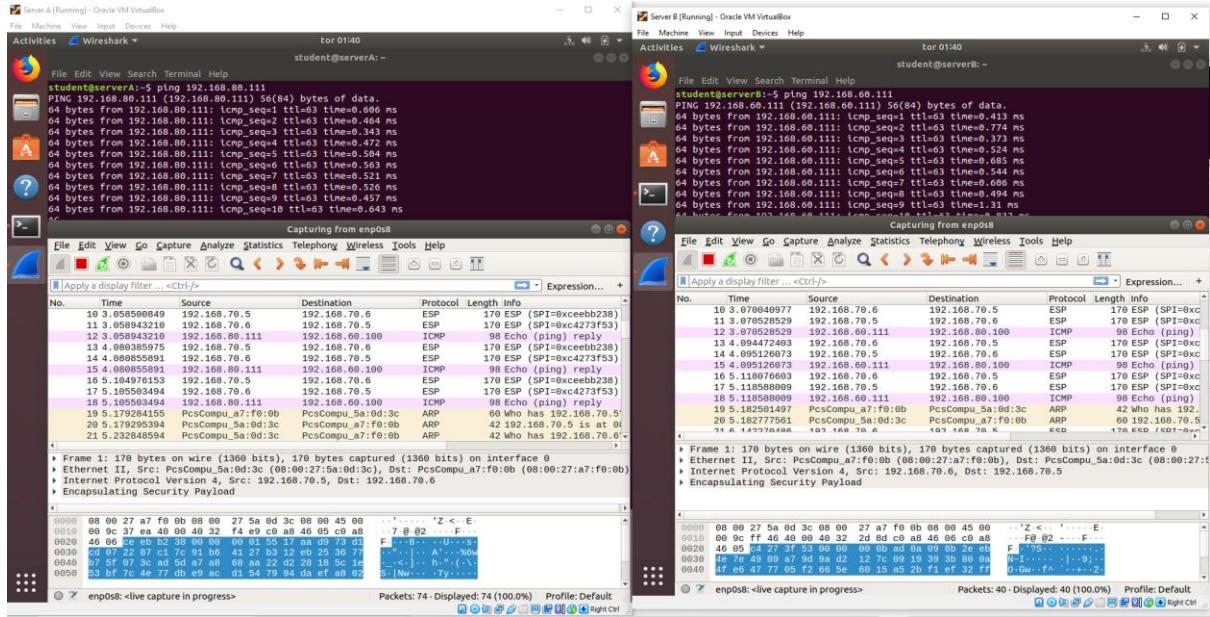
  

```

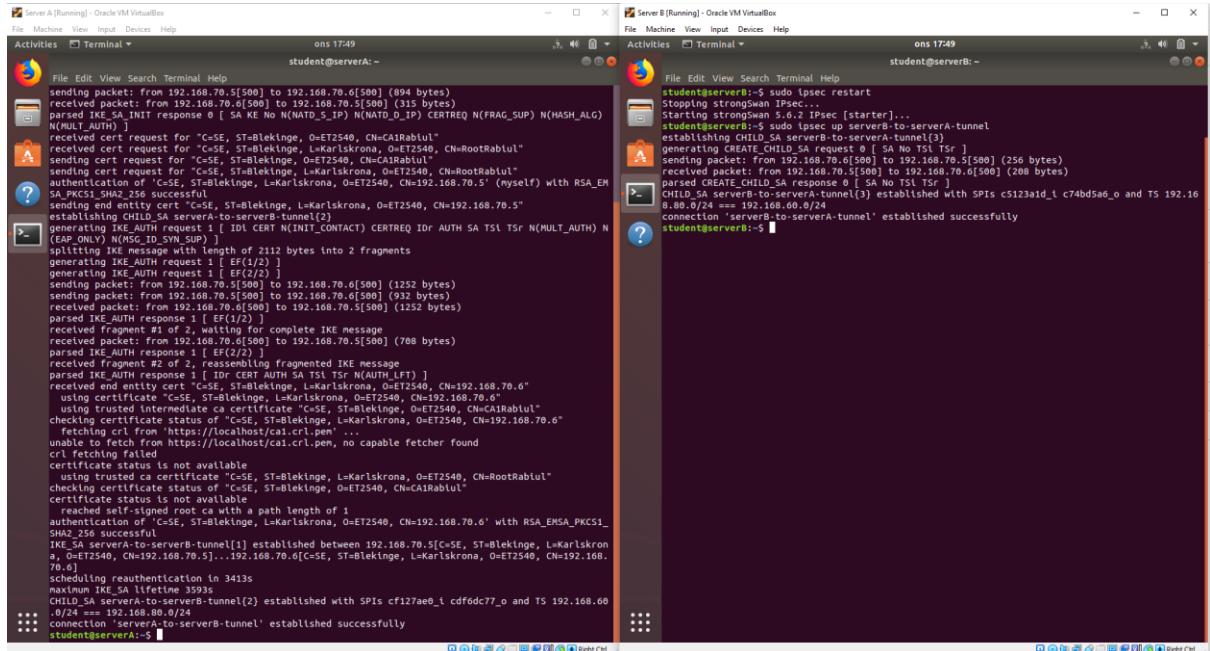
Server B [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal tor 01:22
student@serverB:~$ 
File Edit View Search Terminal Help
student@serverB:~$ sudo ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.6.2 IPsec [starter]...
student@serverB:~$ sudo ipsec up serverA-to-serverB-tunnel
establishing CHAP SA kernel>serverA-tunnel(3)
generating CREATE_CHILD_SA request 0 [ SA No TSL TsR ]
sending packet: from 192.168.70.6[500] to 192.168.70.5[500] (256 bytes)
received packet: from 192.168.70.5[500] to 192.168.70.6[500] (268 bytes)
parsed CREATE_CHILD_SA response 0 [ SA No TSL TsR ]
student@serverB:~$ ipsec up serverA-to-serverB-tunnel(3) established with SPIs c6c0a230_l cde6588d_o and TS 192.168.66.0/24
connection 'serverB-to-serverA-tunnel' established successfully
student@serverB:~$ 

```

- As a evidence for this task ping from serverA to clientB and from serverB to clientA which is shown below screen short with wireshark traffic.



## Task 21: Site A to Site B VPN with default DROP firewall rules



- I did not make any changes to ipsec.conf files. I just restarted strongSwan and enabled tunnel mode. Then I set the default firewall policies to DROP for both Server A and Server B. After that, I added the firewall rules for Server A and Server B mentioned in the screenshot below:

```

33 # Delete any user-defined chains in KMR table
34 #IPT -t nftangle -X
35
36 # Default firewall policy to DROP
37 #IPT -t filter -P INPUT DROP
38 #IPT -t filter -P OUTPUT DROP
39 #IPT -t filter -P FORWARD DROP
40
41
42 # Create logging chains
43 #IPT -t filter -N input_log
44 #IPT -t filter -N output_log
45 #IPT -t filter -N forward_log
46
47 # Set some logging targets for DROPPED packets
48 #IPT -t filter -A input_log -j LOG --log-level notice --log-prefix "input drop: "
49 #IPT -t filter -A output_log -j LOG --log-level notice --log-prefix "output drop: "
50 #IPT -t filter -A forward_log -j LOG --log-level notice --log-prefix "forward drop: "
51 echo "Added logging"
52
53 # Return from the logging chain to the built-in chain
54 #IPT -t filter -A input_log -j RETURN
55 #IPT -t filter -A output_log -j RETURN
56 #IPT -t filter -A forward_log -j RETURN
57
58 #Firewall rules for task 20
59 #IPT -A INPUT -s 192.168.70.5 -d 192.168.70.6 -j ACCEPT
60 #IPT -A OUTPUT -s 192.168.70.6 -d 192.168.70.5 -j ACCEPT
61 #IPT -t filter -A FORWARD -l SHIF -j ACCEPT
62 #IPT -t filter -A FORWARD -l SNIF -n conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
63
64 #task 21
65 #IPT -A FORWARD -s 192.168.80.0/24 -d 192.168.60.0/24 -j ACCEPT
66 #IPT -A FORWARD -s 192.168.60.0/24 -d 192.168.80.0/24 -j ACCEPT
67 #IPT -t filter -A FORWARD -l SHIF -j ACCEPT
68 #IPT -t filter -A FORWARD -l SNIF -n conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
69
70 #IPT -A INPUT -s 192.168.60.111 -d 192.168.70.5 -j ACCEPT
71 #IPT -A OUTPUT -s 192.168.70.5 -d 192.168.60.111 -j ACCEPT
72 #IPT -A INPUT -s 192.168.70.6 -d 192.168.70.5 -j ACCEPT
73 #IPT -A OUTPUT -s 192.168.70.5 -d 192.168.70.6 -j ACCEPT
74
75 #IPT -t nat -A POSTROUTING -j SNAT -o SNIF --to SNIP
76 # These rules must be inserted at the end of the built-in
77 # chain to log packets that will be dropped by the default
78 # DROP policy
79 #IPT -t filter -A INPUT -j input_log
80 #IPT -t filter -A OUTPUT -j output_log
81 #IPT -t filter -A FORWARD -j forward_log

```

- Then I ran the firewall scripts and enabled forwarding.

```

File Edit View Search Terminal Help
student@serverA:~$ sudo ./fw_lab2.sh
Added logging
student@serverA:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
student@serverA:~$ sudo sysctl -p
student@serverA:~$ 

File Edit View Search Terminal Help
student@serverB:~$ sudo ./fw_lab2.sh
Added logging
student@serverB:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
student@serverB:~$ sudo sysctl -p
student@serverB:~$ 

```

- The following is the iptables state after running the firewall script from server A and serverB.

```

File Edit View Search Terminal Help
student@serverA:~$ sudo iptables -L
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT all -- clientA vpnA
ACCEPT all -- vpnB vpnA

Chain FORWARD (policy DROP)
target prot opt source destination
ACCEPT all -- 192.168.80.0/24 192.168.80.0/24
ACCEPT all -- 192.168.60.0/24 192.168.60.0/24
ACCEPT all -- anywhere anywhere ctstate RELATED,ESTABLISHED
ACCEPT all -- anywhere anywhere

Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT all -- vpnA clientA
ACCEPT all -- vpnA vpnB

Chain forward_log (0 references)
target prot opt source destination
LOG all -- anywhere anywhere LOG level notice prefix "forward dro
p: "
RETURN all -- anywhere anywhere

Chain input_log (0 references)
target prot opt source destination
LOG all -- anywhere anywhere LOG level notice prefix "input dr
p: "
RETURN all -- anywhere anywhere

Chain output_log (0 references)
target prot opt source destination
LOG all -- anywhere anywhere LOG level notice prefix "output d
p: "
RETURN all -- anywhere anywhere
student@serverA:~$ 

File Edit View Search Terminal Help
student@serverB:~$ sudo iptables -L
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT all -- clientB vpnB
ACCEPT all -- vpnA vpnB

Chain FORWARD (policy DROP)
target prot opt source destination
ACCEPT all -- 192.168.80.0/24 192.168.80.0/24
ACCEPT all -- 192.168.60.0/24 192.168.60.0/24
ACCEPT all -- anywhere anywhere ctstate RELATED,ESTABLISHED
ACCEPT all -- anywhere anywhere

Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT all -- vpnB clientB
ACCEPT all -- vpnB vpnA

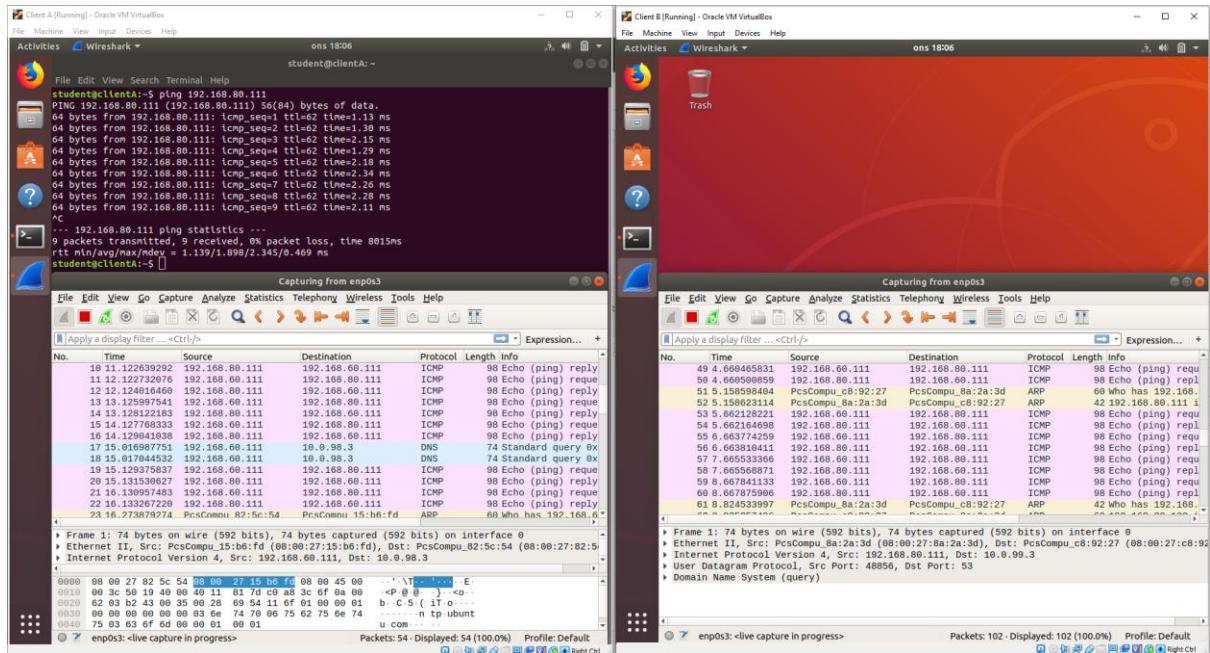
Chain forward_log (0 references)
target prot opt source destination
LOG all -- anywhere anywhere LOG level notice prefix "forward
RETURN all -- anywhere anywhere

Chain input_log (0 references)
target prot opt source destination
LOG all -- anywhere anywhere LOG level notice prefix "input dr
RETURN all -- anywhere anywhere

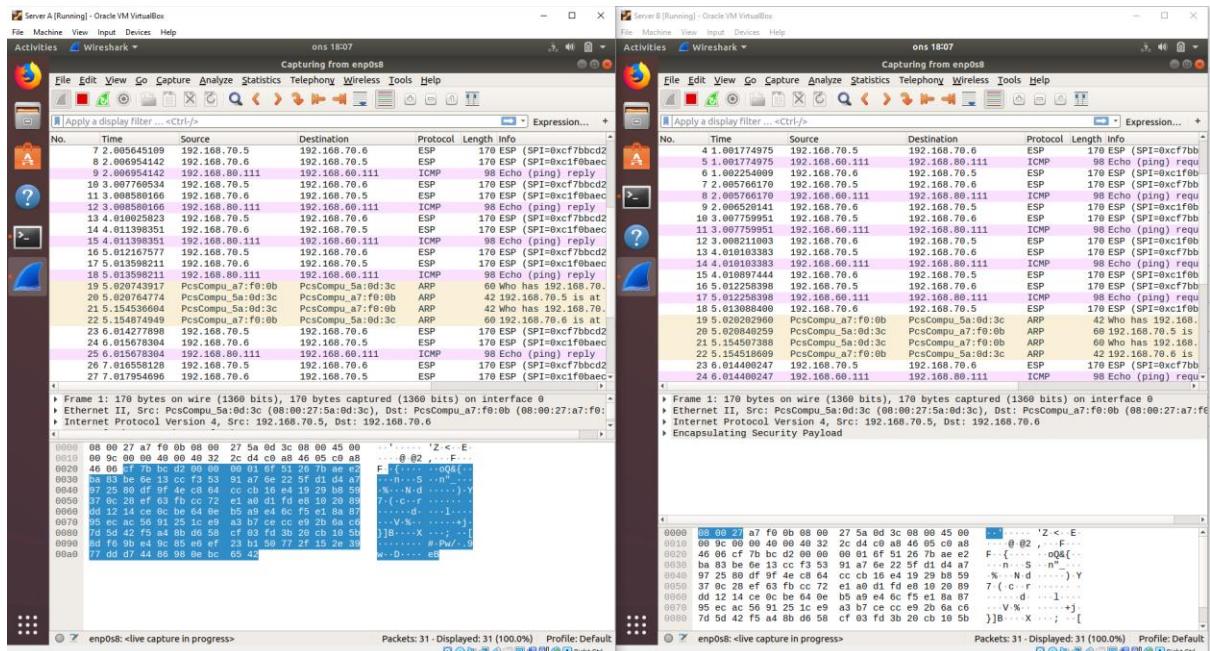
Chain output_log (0 references)
target prot opt source destination
LOG all -- anywhere anywhere LOG level notice prefix "output d
RETURN all -- anywhere anywhere
student@serverB:~$ 

```

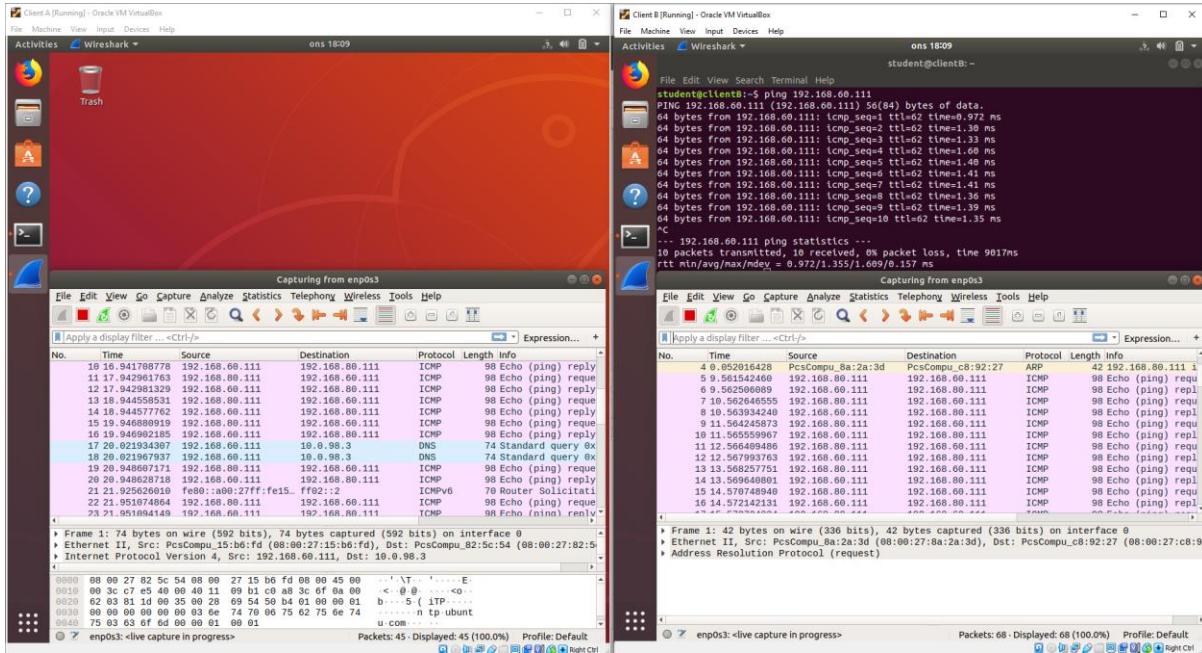
- Then I ping from Client A to Client B and it was working.



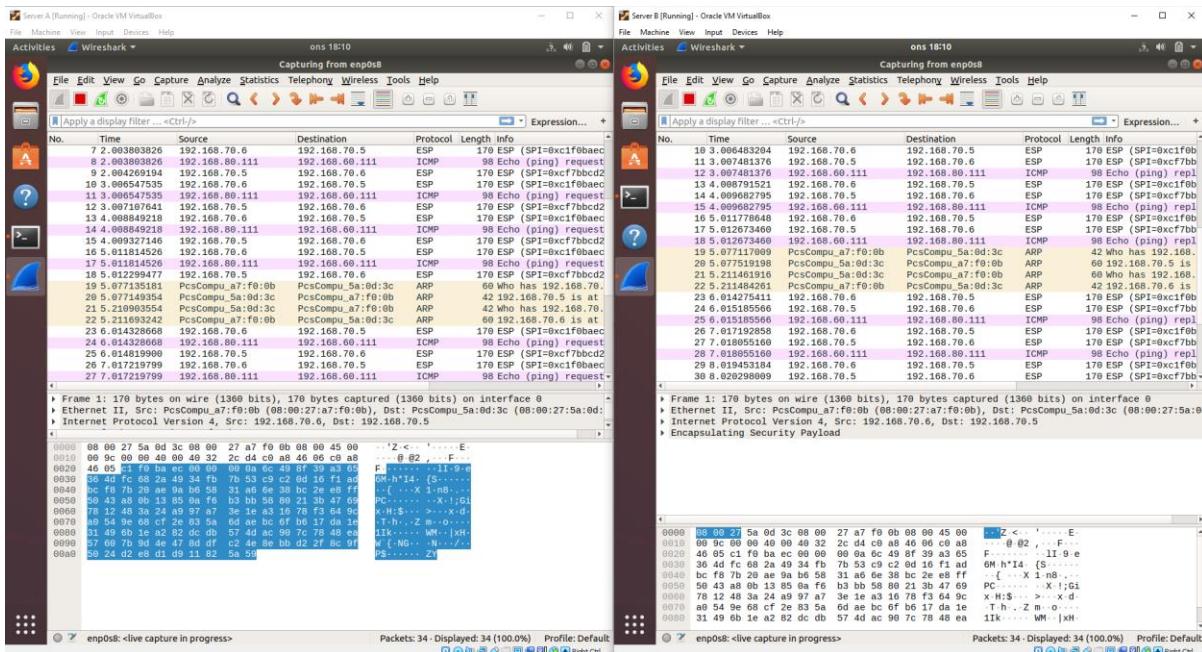
- At the same time, I observed the traffic in Server A and Server B. I found the traffic is going on over enp0s8 interface and it was clear that the traffic was going over the IPsec tunnel.



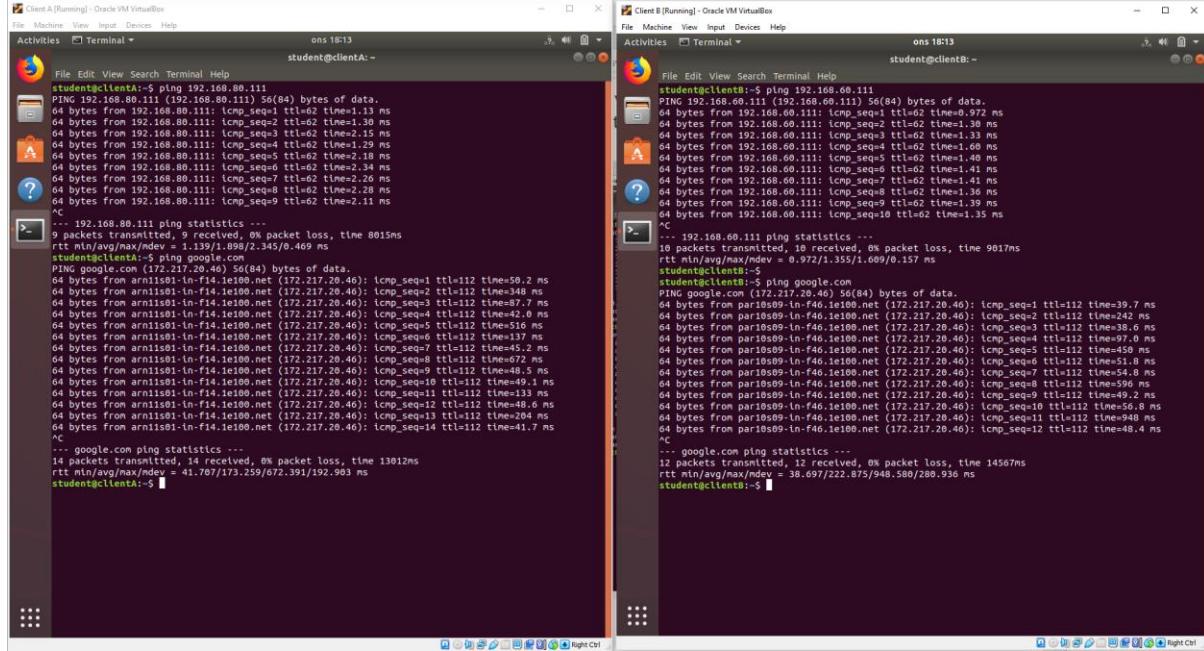
- Next, I ping from Client B to Client A and it was working.



- At the same time, I again observed the traffic in Server A and Server B. I found the traffic is going on over enp0s8 interface and it was clear that the traffic was going over the IPsec tunnel.



- Then I ping to Google from both Client A and Client B. It worked in both cases. Client A's traffic was forwarded by Server A to the NAT and this is how it accessed the internet. Same goes for Client B's to Server B.



## Reference:

1. Lab 1.2: Certificates and VPN instructions
2. [https://www.ibm.com/support/knowledgecenter/en/SSB23S\\_1.1.0.13/gtps7/cfgcert.html](https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.13/gtps7/cfgcert.html)
3. <http://rcardon.free.fr/websign/download/api-x509ext/be/cardon/asn1/x509/extensions/KeyUsage.html>
4. <https://www.phildev.net/ssl/opensslconf.html>
5. [https://www.ibm.com/support/knowledgecenter/sl/SSKTMJ\\_9.0.1/admin/conf\\_keyusageextensions\\_andextendedkeyusage\\_r.html](https://www.ibm.com/support/knowledgecenter/sl/SSKTMJ_9.0.1/admin/conf_keyusageextensions_andextendedkeyusage_r.html)
6. [http://javavadoc.iaik.tugraz.at/iaik\\_jce/old/iaik/x509/extensions/ExtendedKeyUsage.html](http://javavadoc.iaik.tugraz.at/iaik_jce/old/iaik/x509/extensions/ExtendedKeyUsage.html)
7. <https://security.stackexchange.com/questions/84816/what-exactly-is-meant-by-serverauthandclientauth>
8. <https://jamielinux.com/docs/openssl-certificate-authority/create-the-root-pair.html>
9. <https://www.phildev.net/ssl/opensslconf.html>
10. <https://linux.die.net/man/1/openssl>
11. <https://www.openssl.org/docs/manmaster/man1/smime.html>
13. <http://www.firewall.cx/networking-topics/protocols/870-ipsec-modes.html>
14. [https://www.gypthecat.com/ipsec-vpn-host-to-host-on-ubuntu-14-04-with-strongswan?fbclid=IwAR1ZUnRqyNBr6ggXWzGOzH%20UL\\_k5LWZyKSxqhhKGcW0eZN4zSL4INdCvK00](https://www.gypthecat.com/ipsec-vpn-host-to-host-on-ubuntu-14-04-with-strongswan?fbclid=IwAR1ZUnRqyNBr6ggXWzGOzH%20UL_k5LWZyKSxqhhKGcW0eZN4zSL4INdCvK00)