

Public Key Infrastructure (PKI) Certificates Using OpenSSL

Documentation: <http://www.openssl.org/docs/apps/pkcs8.html>

➤ Certificate Requests Commands:

- *To generate a certificate request and sign it from CA.*

In the following: replace <student> with your <login name>

```
% mkdir <student>_cert_request  
% cd <student>_cert_request  
% cp /home/cs772/public_html/PKI/shell_scripts/students/* .
```

This copies all the shell scripts and files needed.

```
% gencertreq-sh <student>
```

That will create two files:

```
<student>_certreq.pem & <student>_privatekey.pem
```

```
% printcertreq-sh <student>
```

This prints <student>_certreq.pem

```
% submit cs772
```

Submit file: <student>_certreq.pem

After the CA has signed your certificate and notifies you via email do the following:

```
% cp  
/home/cs772/public_html/PKI/certificates/students/<student>_cert.pem .  
% cp /home/cs772/public_html/PKI/certificates/students/ca_cert.pem .
```

```
% printcert.sh <student>
```

This prints <student>_cert.pem

- **To renew a certificate and re-sign it from CA.**

```
% certoreq.sh <student>
```

*This generates <student>_certreq.pem using the your **original** public/private keys. Submit this request for the CA to sign as you did for the original request.*

➤ **Certificate Authority (CA) Commands**

To setup the necessary environment to create CA keys and root certificate, and to sign and resign the students certificate requests.

- **To create CA keys and root certificate**

```
% cp /home/cs772/public_html/PKI/shell_scripts/ca/* .
```

*Copy the ca shell scripts in any directory you choose.
Edit file **myopenssl.cnf** to replace **cs772** with your <login name>*

```
% setup.sh
```

Create the needed files and directories.

```
% genca.sh
```

Create CA keys (ca_key.pem) & certificate (ca_cert.pem).

% [printcert.sh](#) ca

print certificate ca_cert.pem

▪ [To sign certificate requests](#)

% [issuercert.sh](#) <student>

This signs a request from student under:

submitted_requests/<student>_cerreq.pem

And produces:

signed_requests/<student>_cert.pem & issued_certs/xx.pem

where <xx> is serial number of cert.

To sign more than one certificate put the names in a file called: *list* and then use:

% [batch issuercert-sh](#)

▪ [To re-sign certificate requests](#)

Follow the process of signing certificates after removing the old certificate entry from file: [index.txt](#)

Application of PKI:

*Secure Multipurpose Internet Mail Exchange
([SMIME](#))*

Use unix [Mail](#) to send and read your mail.

In reading a mail message: write the message to a file (e.g., [w file](#))

```
% cd <student>_cert_request
```

```
% cp /home/cs772/public_html/PKI/shell_scripts/securemail/* .
```

This copies all the shell scripts and files needed.

- **Encrypted mail**

Send: % sendencmail-sh <receiver> <file>

The sender should have: <receiver>_cert.pem

This encrypts mail for the given recipient certificates.

Input <file> is the message to be encrypted.

The output <file>.enc is the encrypted mail in MIME format.

The file is encrypted with a secret key.

The secret key is encrypted with the public key in <receiver>_cert.pem

Read: First read your mail and save the msg in <file> then use the following command:

```
% readencmail-sh <receiver> <file>
```

The recipient should have:

<receiver>_cert.pem & <receiver>_privatekey.pem

Decrypt mail using the supplied certificate and private key.

It expects an encrypted mail message in MIME format for the input <file>.

The secret key is decrypted with <receiver>_privatekey.pem

The file is decrypted with a secret key.

- **Signed mail**

Send: % sendsignmail-sh <receiver> <file> <sender>

The sender should have:

<sender>_cert.pem & <sender>_privatekey.pem

This signs mail using the supplied certificate and private key.

Input <file> is the message to be signed.

The signed message in MIME format is written to the output <file>.sig

Read: First read your mail and save the msg in `<file>` then use the following command:

```
% readsignmail-sh <file>
```

The recipient should have: `ca_cert.pem`

This verifies signed mail.

Expects a signed mail message on input and outputs the signed data.

- **Signed + Encrypted mail**

Send: % `sendsign_email-sh` `<receiver>` `<file>` `<sender>`

The sender should have: `<receiver>_cert.pem`,
`<sender>_cert.pem` & `<sender>_privatekey.pem`

Read: First read your mail and save the msg in `<file>` then use the following command:

```
% readsign_email-sh <receiver> <file>
```

The recipient should have: `<receiver>_cert.pem`,
`<receiver>_privatekey.pem` & `ca_cert.pem`

Sign then Encrypt Versus **Encrypt then Sign**?

Sign then Encrypt:

A sends B message `M`,

B decrypt, reads `M` and verify it is from A.

if B wants to forwards the message to C,

B can encrypt it and forwards the signed message to C,

C can decrypt, reads the message and verify that A is the author.

Encrypt then Sign:

if B wants to forwards the message to C,

C can not decrypt it since he does not have the private key of B.

