**Project 1: Public Key, Root CA, Certificates, Hash, MAC and SSL VPNs**

[1 Lab Preparation 2](#_Toc379205712)

[2 Performance Comparison: RSA versus AES 2](#_Toc379205713)

[3 Become a root CA and sign a Certificate 3](#_Toc379205714)

[3.1 Create a new CA 3](#_Toc379205715)

[3.2 Generate a new certificate request 4](#_Toc379205716)

[3.3 Sign the certificate request 4](#_Toc379205717)

[4 Configure SSL certificates on the Apache server 5](#_Toc379205718)

[4.1 Modify the Apache server configuration 5](#_Toc379205719)

[4.2 Test an SSL/TLS connection 5](#_Toc379205720)

[5 Install the CA root certificate as a Trusted Root Certificate 6](#_Toc379205721)

[5.1 Install the root CA for Firefox on the Linux server 6](#_Toc379205722)

[5.2 Install the root CA for Firefox on a Linux client 7](#_Toc379205723)

[5.3 Install the root CA for IE on Windows 7](#_Toc379205724)

[5.4 Install the root CA for Firefox on Windows 7](#_Toc379205725)

[6 One-Way Hash Function and MAC 7](#_Toc379205726)

[6.1 One-way Hash Functions 7](#_Toc379205727)

[6.2 Keyed Hash and HMAC 8](#_Toc379205728)

[7 Setting SSL based VPNs for Linux 8](#_Toc379205729)

[7.1 Build Certificate Authority (CA) 8](#_Toc379205730)

[7.2 Certificate and key generation 9](#_Toc379205731)

[7.3 Server configuration 10](#_Toc379205732)

[7.4 Client configuration 11](#_Toc379205733)

[8 Configure client certificates for authentication (optional) 11](#_Toc379205734)

**Goal:** In this lab we will exercise the idea that digital certificates are verified using a chain of trust. The trust anchor for the digital certificates is the Root Certificate Authority (CA). In addition, we will practice using command-line procedures for setting up SSL/TLS certificates. First, we will generate a root certificate (to become the root CA) and other certificates. Then, we will sign the certificates with the root certificate. Next, we will place these certificates on an Apache web server. When the web server is visited by a client’s browser with the root CA installed, the server will be trusted automatically due the chain of trust.

**Major Reference for this lab:**

<http://www.reppep.com/~pepper/writing/tidbits/ssl-article/ssl2.text>

# Lab Preparation

On your FC Linux server, as root, type the following command to update your openssl and httpd utilities: (needs Internet access)

yum –y install openssl-\*

yum –y install httpd-\*

yum –y install mod\_ssl

# Performance Comparison: RSA versus AES

In this task, we will study the performance of public-key algorithms. Please prepare a file (message.txt) that contains a 16-byte message. Please also generate a 1024-bit RSA public/private key pair by using the following commands:

openssl genrsa -out private.pem 1024

This command creates a key file called private.pem that uses 1024 bits. This file actually has both the private and public keys, so you should extract the public one from this file:

openssl rsa -in private.pem -out public.pem -outform PEM –pubout

The commands of encryption and decryption using public key look like the following:

openssl rsautl -encrypt -inkey public.pem -pubin -in file.txt -out file.ssl

openssl rsautl -decrypt -inkey private.pem -in file.ssl -out decrypted.txt

The commands of encryption and decryption using AES like the following:

openssl enc –aes-128-cbc -in file.txt -out file.enc

openssl enc –d –aes-128-cbc -in file.enc -out file.txt

Next, do the following:

1. Encrypt message.txt using the public key; save the output in message enc.ssl.

2. Decrypt message enc.ssl using the private key.

3. Encrypt message.txt using a 128-bit AES key.

### Compare the time spent on each of the above operations, and describe your observations. If an operation is too fast, you may want to repeat it for many times, and then take an average.

After you finish the above exercise, you can now use the openssl speed commands to do such a benchmarking. The following command shows examples of using speed to benchmark RSA and AES:

openssl speed rsa

openssl speed aes

### Please describe whether your observations are similar to those from the outputs of the speed command.

# Become a root CA and sign a Certificate

Go to the /etc/pki/tls/misc/ directory. There is a file named *CA.pl*. We will use this file to create a root certificate and other certificates. Reference on how to use CA.pl is here: <http://www.openssl.org/docs/apps/CA.pl.html>.

## Create a new CA

./CA.pl -newca

 A certificate filename (or enter to create) ***<enter>***

 Making CA certificate ...

 Using configuration from openssl.cnf

 Generating a 1024 bit RSA private key

 ............++++++

 ......................++++++

 writing new private key to '../../CA'

 Enter PEM pass phrase: ***<secret passphrase here> write it down for future reference, mine is CMSC491***

 Verifying password - Enter PEM pass phrase: ***<secret passphrase again>***

 -----

 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]:***US***

 State or Province Name (full name) [Some-State]:***NYC***

 Locality Name (eg, city) []:***Holmdel***

 Organization Name (eg, company) []:***cu***

 Organizational Unit Name (eg, section) []:***CS***

 Common Name (eg, YOUR name) []: ***WeiC # I put WeiC here as the example common name. A more general practice is to put your sever’s DNS name.***

Email Address []:***yourname@cu.edu***

No need to add a challenge password

You will be asked to input your PEM key again in the end (before the accomplish of the certificate creation)

This creates a new root certificate in the directory */etc/pki/CA*. The new root certificate file is named as *cacert.pem*.

## Generate a new certificate request

./CA.pl -newreq

Using configuration from openssl.cnf

 Generating a 1024 bit RSA private key

 ..........++++++

 ..............++++++

 writing new private key to 'newreq.pem'

 Enter PEM pass phrase: ***<another secret passphrase here> mine is 491req***

 Verifying password - Enter PEM pass phrase: ***<another secret passphrase again>***

 -----

 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]:***US***

 State or Province Name (full name) [Some-State]:***VA***

 Locality Name (eg, city) []:***Richmond***

 Organization Name (eg, company) [Internet Widgits Pty Ltd]:***VCU***

 Organizational Unit Name (eg, section) []:***CStest***

 Common Name (eg, YOUR name) []:***192.168.0.10, which is the http server’s IP address. You can use your FC Linux server’s IP (i.e., the IP of the virtual machine). If the http server has a DNS name, please use the DNS name.***

 Email Address []:***yourname@vcu.edu***

 Please enter the following 'extra' attributes

 to be sent with your certificate request

 A challenge password []:***<enter>***

 An optional company name []:***<enter>***

 Request is in newreq.pem, private key is in newkey.pem

The generated certificate request and private key in are stored newreq.pem and newkey.pem, respectively.

## Sign the certificate request

./CA.pl -sign

You will be prompted for the root CA password, which is *cmsc491* in this particular case. If you received no errors, the certificate has been signed by your root certificate.

Now, under the */etc/pki/tls/misc/* directory, there will be the certificate, newcert.pem, and the private key - newreq.pem (encrypted) and newkey.pem (unencrypted). They are now ready to be used. You may wish to rename the files to more intuitive names. In addition, the root certificate file (named as *cacert.pem*) is located in the */etc/pki/CA* directory.

# Configure SSL certificates on the Apache server

## Modify the Apache server configuration

On the FC Linux server, go the /etc/httpd/conf.d directory. There is a file named ssl.conf. edit this file and modify the two places:

*SSLCertificateFile /etc/pki/tls/misc/newcert.pem*

*SSLCertificateKeyFile /etc/pki/tls/misc/newkey.pem*

Save the file and quit. Now, stop the httpd service if it is already running. Restart the httpd service in the debug mode by typing:

httpd -X

### Were asked to provide a key? Which key should you provide?

## Test an SSL/TLS connection

Open a browser, type the <https://server_IP>.

### Did you receive a warning message?

You may notice that you are required to enter the key when the http restarts. There is way to start the Apache server without requiring inputting the key. Here are the steps:

cd /etc/pki/tls/misc

cp newkey.pem newkey.pem.org

openssl rsa -in newkey.pem.org -out newkey.pem

Now, issue service *httpd-k restart* command to verify the setting is okay.

# Install the CA root certificate as a Trusted Root Certificate

Next, we will verify the chain of trust.

## Install the root CA for Firefox on the Linux server

Open the Firefox. Go to Edit -> Preferences -> Advanced -> Encryption ->View Certificates:



Next, import our Authorities (the *cacert.pem* in */etc/pki/CA*):





Test an SSL/TLS connection. If you aren't warned about the site certificate, the root certificate has been successfully installed.

### Explain why you did not receive the warning message?

## Install the root CA for Firefox on a Linux client

Go a FC Linux client virtual machine; repeat the same procedures to install the root CA.

Test an SSL/TLS connection

If you aren't warned about the site certificate, the root certificate has been successfully installed.

## Install the root CA for IE on Windows

On your FC Linux server, convert the Root Certificate from PEM to DER format (for Windows client use):

openssl x509 -in cacert.pem -inform PEM -out my-rootCA.der -outform DER

This will create a DER format certificate that can be easily installed on Windows XP.

Transfer the DER format certificate to a Windows machine. An easy way to transfer the certificate is to place the certificate on the Linux web server and download it from the Windows machine. Copy the certificate to */var/www/html*.

Once you have downloaded the certificate to the Windows machine, double click the certificate to install it. Follow the default option should be okay.

Test an SSL/TLS connection using IE.

If you aren't warned about the site certificate, the root certificate has been successfully installed.

Test an SSL/TLS connection using Firefox.

### Explain what happened. Why?

## Install the root CA for Firefox on Windows

The procedure to install the root CA for Firefox on Windows is similar to that of Linux.

# One-Way Hash Function and MAC

## One-way Hash Functions

In this task, we will play with various one-way hash algorithms. You can use the following commands to generate the hash value for a file. You can create the file by typing *echo “whatever contend in here” > filename*

*md5sum filename*

*sha256sum filename*

*sha512sum filename*

Output the hash values to a file for future reference.

Alternatively, you can use *openssl dgst* command to achieve the same goals. The command prototype looks like: *openssl dgst dgsttype filename*, where the dgsttype refers to a specific one-way hash algorithm, such as -md5, -sha256, etc.

To understand the randomness of the one-way function, we will do the following exercise for MD5, SHA256, and SHA512. Modify one letter of the original input file and regenerate the hash values for three one-way functions.

### Please observe whether the results are similar or not. Please describe your observations in the project report.

## Keyed Hash and HMAC

In this task, we would like to generate a keyed hash (i.e. MAC) for a file. We can use the *-hmac* option. The following example generates a keyed hash for a file using the HMAC-MD5 algorithm. The string following the -hmac option is the key.

*openssl dgst -md5 -hmac "abcdefg" filename*

Please generate a keyed hash using HMAC-MD5 and HMC-SHA512 for any file that you choose. Please try several keys with different length.

### Do we have to use a key with a fixed size in HMAC? If so, what is the key size? If not, why?

# Setting SSL based VPNs for Linux

The instructions below are adopted from the following source with some modifications: <http://openvpn.net/index.php/open-source/documentation/howto.html>

Determine your VPN server and client machines. On both FC server and FC client:

yum –y install openssl openvpn

yum –y install openpn easy-rsa

systemctl disable firewalld

systemctl stop firewalld

## Build Certificate Authority (CA)

On your FC server:

cp –r /usr/share/easy-rsa/ /etc/openvpn/

cd /etc/openvpn/easy-rsa/2.0

. ./vars

./clean-all

./build-ca

The final command (**build-ca**) will build the certificate authority (CA) certificate and key by invoking the interactive **openssl** command:

ai:easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
............++++++
...........++++++
writing new private key to 'ca.key'
-----
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) [KG]:
State or Province Name (full name) [NA]:
Locality Name (eg, city) [BISHKEK]:
Organization Name (eg, company) [OpenVPN-TEST]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:WeiFC
Email Address [me@myhost.mydomain]:

|  |
| --- |
| Note that in the above sequence; most queried parameters were defaulted to the values set in the **vars** files. The only parameter which must be explicitly entered is the **Common Name**. In the example above, I used "WeiFC". |

## Certificate and key generation

We first will generate a certificate and private key for the server:

*./build-key-server server*

As in the previous step, most parameters can be defaulted. When the **Common Name** is queried, enter "server". Two other queries require positive responses, "Sign the certificate? [y/n]" and "1 out of 1 certificate requests certified, commit? [y/n]".

Secondly, generate certificates & keys for a client. Generating client certificates is very similar to the previous step:

*./build-key client*

Remember that for each client, make sure to type the appropriate **Common Name** when prompted, i.e. "client”. Always use a unique common name for each client.

Thirdly, generate Diffie Hellman parameters. [Diffie Hellman](http://www.rsasecurity.com/rsalabs/node.asp?id=2248) parameters must be generated for the OpenVPN server:

*./build-dh*

The **final** step in the key generation process is to copy all the necessary files to the machines which need them. In our case, we need to copy “ca.crt”, “client.crt”, and “client.key” to the client machine using the command scp. Example commands look like this:

*scp ca.crt root@client\_IP\_address:/etc/openvpn*

## Server configuration

It's best to use the OpenVPN sample configuration files as a starting point for your own configuration. These files are located in ***/usr/share/doc/openvpn/sample/sample-config-files***

Now, copy the example server.conf file from the above directory to /etc/openvpn:

cp /usr/share/doc/openvpn/sample/sample-config-files/server.conf /etc/openvpn

vi /etc/openvpn/server.conf

Pay attention to the following 4 parameters:

ca ca.crt

cert server.crt

key server.key # This file should be kept secret

- and -

dh dh2048.pem

You need to add the right path information for each of the 4 parameters. These 4 files should be located in **/etc/openvpn/easy-rsa/2.0/keys**. Thus, it should look like the following:

ca /etc/openvpn/easy-rsa/2.0/keys/ca.crt

…….

Now, you can start the VPN server via:

*openvpn /etc/openvpn/server.conf*

|  |
| --- |
| A normal server startup should look like this (output will vary across platforms): |

Sun Feb 6 20:46:38 2005 OpenVPN 2.0\_rc12 i686-suse-linux [SSL] [LZO] [EPOLL] built on Feb 5 2005
Sun Feb 6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb 6 20:46:38 2005 TLS-Auth TCU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Feb 6 20:46:38 2005 TUN/TAP device tun1 opened
Sun Feb 6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 tcu 1500
Sun Feb 6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.8.0.2
Sun Feb 6 20:46:38 2005 Data Channel TCU parms [ L:1542 D:1450 EF:42 EB:23 ET:0 EL:0 AF:3/1 ]
Sun Feb 6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb 6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb 6 20:46:38 2005 MULTI: multi\_init called, r=256 v=256
Sun Feb 6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb 6 20:46:38 2005 IFCONFIG POOL LIST
Sun Feb 6 20:46:38 2005 Initialization Sequence Completed

## Client configuration

Go to your FC client machine. The client configuration steps are similar to the server’s.

Copy the client.conf file from the sample-config-files directory to /etc/openvpn:

cp /usr/share/doc/openvpn/sample/sample-config-files/client.conf /etc/openvpn

vi /etc/openvpn/client.conf

Modify the following 3parameters accordingly (these 3 files were copied from the server in the previous section 6.2):

ca ca.crt

cert client.crt

key client.key

You also need to modify the /etc/hosts file by adding one entry:

IP\_address\_of\_your \_FC\_server my-server-1

Now, you can start the VPN client via:

*openvpn /etc/openvpn/client.conf*

A normal client startup will look similar to the server output above, and should end with the **Initialization Sequence Completed** message. In addition, make sure that the VPN server is running on the FC server machine.

Now, try a ping across the VPN from the client:

 ping 10.8.0.1

If the ping succeeds, congratulations! You now have a functioning SSL based VPN.

Start the Wireshark to capture the ongoing ping packets.

### Check whether the ping packets are encrypted.

### In your words, explain what does this setup do?

# Configure client certificates for authentication (optional) (10 points)

Sometime, you may want to limit the access to certain content on a web site. One common approach is to set up user name and password pairs for access control. Another stronger approach is to deploy client certificates for authentication.

If you have some time, please follow the steps described in the following link to set up client certificates for authentication: <http://www.vanemery.com/Linux/Apache/apache-SSL.html>.