# INFA723 Cryptography and Network Security

# Lab 10 Use OpenSSL to Set up a Simple SSL/TLS Server and Test a Remote Host Using SSL/TLS

# 1 Introduction

OpenSSL toolkit provides a command line tool and a crypto library used for a wide range of crypto operations. In this exercise, we are going to learn how to use OpenSSL to set up a simple SSL/TLS server and test a remote host using SSL/TLS. Note that for all the functions implemented in the lab, you can find a way to implement them using the OpenSSL crypto library.

A lab package has been created for the lab Labs9-10.zip. Go to the class website and download the lab package. Assume your Cygwin installation folder is c:\cygwin. Unzip the package to your Cygwin home folder, e.g., C:\cygwin\home\user name.

OpenSSL command line provides utilities to set up a simple SSL/TLS server and also a client to connect to a remote server using SSL/TLS. The tool enables you to check the interactions between the client and the server. From each SSL/TLS session, you will be able find out the protocol, the cipher, and the key used for each session.

Need to finish Lab 9 first. Lab 10 utilizes the private key and self-signed certificated created in Lab 9.

# 2 Objectives

* Learn and understand how SSL/TLS provides communication security over the Internet
* Use OpenSSL to set up a simple SSL/TLS server
* Test a remote host using SSL/TLS

# 3 Use OpenSSL to Set up a Simple SSL/TLS Server and Test a Remote Host Using SSL/TLS

1. Goto Lab10 folder.
2. Copy private.key and cacert.cer from Lab9 folder to Lab10 folder.

**$ cp ../lab9/cacert.cer ../lab9/private.key .**

1. Start a SSL/TLS server using OpenSSL:

**$ openssl s\_server -cert cacert.cer -key private.key -accept 4443 -WWW**

**Using default temp DH parameters**

**Using default temp ECDH parameters**

**ACCEPT**

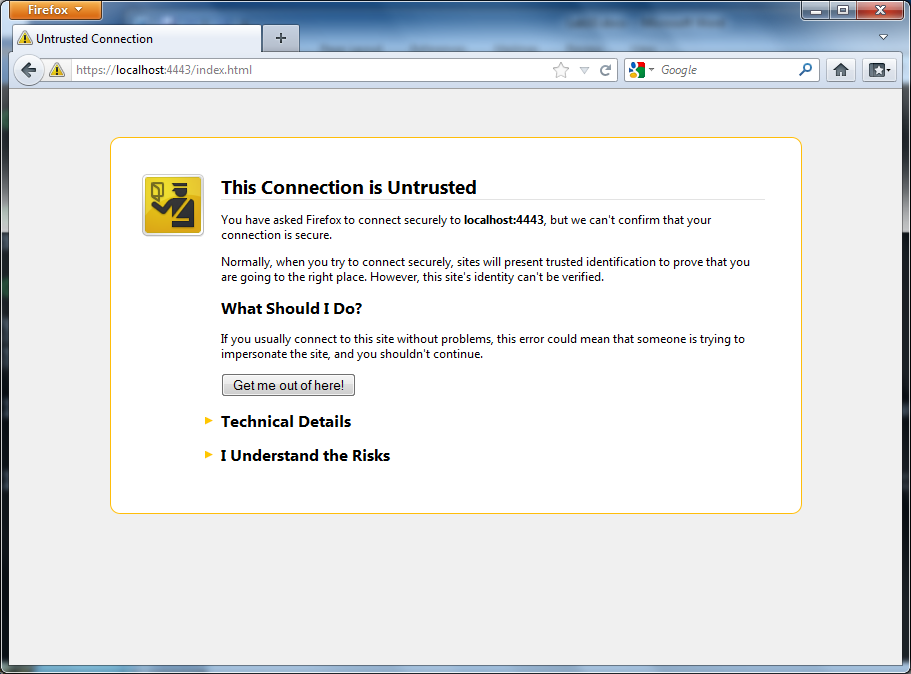
The server is listening on port 4443 to accept any SSL/TLS connection. You can change the port to another port number. The server has a certificate cacert.cer (which includes the public key of the server) and use the private.key for encryption and digital signature. The option –WWW emulates a simple web server. Pages will be resolved relative to the current directory, for example, if the URL <https://localhost:4443/index.html> is requred, the file ./index.html will be loaded.

Windows firewall may pop up a window to ask if you want to access the server. Click “Allow Access” to enable the server.

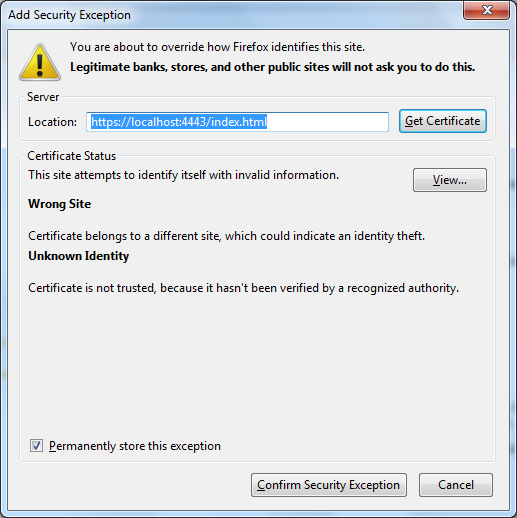
1. Open an internet explorer and enter the URL address <https://localhost:4443/index.html>.

If you use another port number, be sure to change the port number in the URL too.

1. Since the server’s certificate is self-signed and cannot be verified. The explorer will confirm if you want to make an exception.



Click “Add Exception …” button to allow the access.



Click “Confirm Security Exception” button.

You can try to create another html page and see if you can load it in your internet explorer.

1. Check SSL/TLS session parameters. Stop the server by CTRL-C. Restart the server with –www option.

**$ openssl s\_server -cert cacert.cer -key private.key -accept 4443 –www**

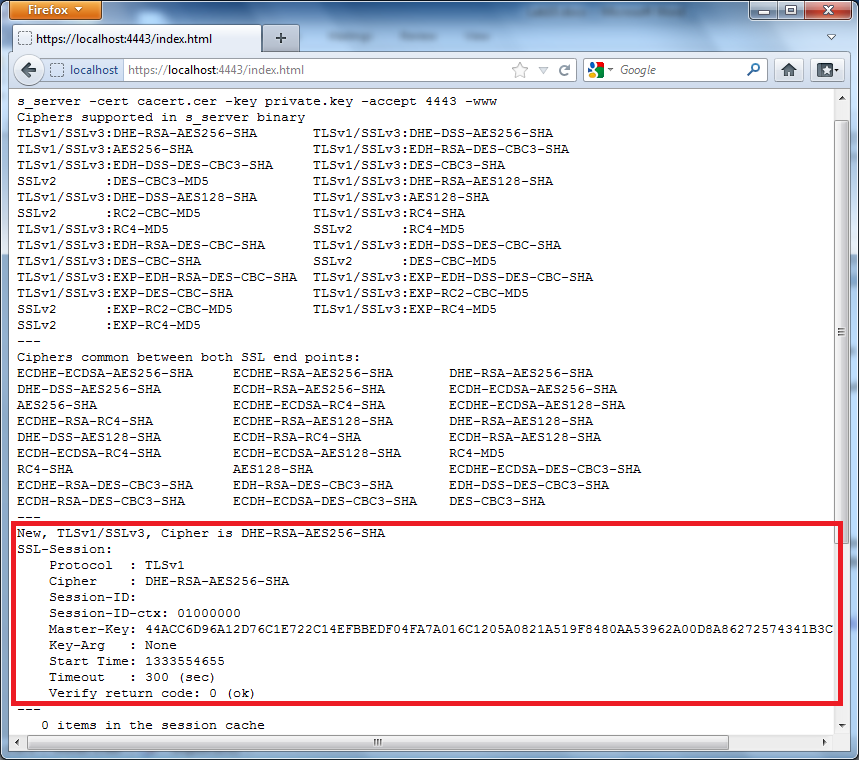
**Using default temp DH parameters**

**Using default temp ECDH parameters**

**ACCEPT**

-www sends a status message back to the client when it connects. This includes lots of information about the ciphers used and various session parameters. The output is in HTML format so this option will normally be used with a web browser.

1. Open an internet explorer and enter the URL address <https://localhost:4443/index.html>.



Take a look at the section in the red box. The red box identified the SSL/TLS parameters used in the protocol. For example, the protocol is TLSv1. The cipher between the client and server is DHE-RSA-AES256-SHA. The Masker-Key used is also listed in the red box.

1. Connect to the SSl/TLS server using OpenSSL s\_client function. Open another Cygwin shell and enter the following command.

**$ openssl s\_client -connect localhost:4443**

**CONNECTED(00000003)**

**depth=0 /C=US/ST=SD/L=Madison/O=DSU/OU=BIS/CN=INFA723/emailAddress=yong.wang@dsu .edu**

**verify error:num=18:self signed certificate**

**verify return:1**

**depth=0 /C=US/ST=SD/L=Madison/O=DSU/OU=BIS/CN=INFA723/emailAddress=yong.wang@dsu .edu**

**verify return:1**

**---**

**Certificate chain**

**0 s:/C=US/ST=SD/L=Madison/O=DSU/OU=BIS/CN=INFA723/emailAddress=yong.wang@dsu.ed u**

**i:/C=US/ST=SD/L=Madison/O=DSU/OU=BIS/CN=INFA723/emailAddress=yong.wang@dsu.ed u**

**---**

**Server certificate**

**-----BEGIN CERTIFICATE-----**

**MIIDWTCCAsKgAwIBAgIJAPcxcqEGrjUzMA0GCSqGSIb3DQEBBQUAMHwxCzAJBgNV**

**BAYTAlVTMQswCQYDVQQIEwJTRDEQMA4GA1UEBxMHTWFkaXNvbjEMMAoGA1UEChMD**

**RFNVMQwwCgYDVQQLEwNCSVMxEDAOBgNVBAMTB0lORkE3MjMxIDAeBgkqhkiG9w0B**

**CQEWEXlvbmcud2FuZ0Bkc3UuZWR1MB4XDTEyMDQwMjIxNDEwMFoXDTE1MDQwMjIx**

**NDEwMFowfDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAlNEMRAwDgYDVQQHEwdNYWRp**

**c29uMQwwCgYDVQQKEwNEU1UxDDAKBgNVBAsTA0JJUzEQMA4GA1UEAxMHSU5GQTcy**

**MzEgMB4GCSqGSIb3DQEJARYReW9uZy53YW5nQGRzdS5lZHUwgZ8wDQYJKoZIhvcN**

**AQEBBQADgY0AMIGJAoGBAOwlJfHAzzUccLdKFjfIZaltMUPNOENieo2KPdwz6NVK**

**3xbfKzVX8iPB5EezdXBz2HzcFT9SHuKBeIzbh1seEPVLfiUN+ygBaBhLroXA8AbD**

**udkullDOck7FJQ0dB0odYNQHzSXCuatTa+0/L0L7d/lyuG3VIUFWxEX6FxMmUmHZ**

**AgMBAAGjgeIwgd8wHQYDVR0OBBYEFFrChQ7Iv2Wl9aQjSTcTLSpDB9UQMIGvBgNV**

**HSMEgacwgaSAFFrChQ7Iv2Wl9aQjSTcTLSpDB9UQoYGApH4wfDELMAkGA1UEBhMC**

**VVMxCzAJBgNVBAgTAlNEMRAwDgYDVQQHEwdNYWRpc29uMQwwCgYDVQQKEwNEU1Ux**

**DDAKBgNVBAsTA0JJUzEQMA4GA1UEAxMHSU5GQTcyMzEgMB4GCSqGSIb3DQEJARYR**

**eW9uZy53YW5nQGRzdS5lZHWCCQD3MXKhBq41MzAMBgNVHRMEBTADAQH/MA0GCSqG**

**SIb3DQEBBQUAA4GBADwU7F+3XZu1AO88pKyjW+4f1JXn2XEuJEDV+aU3SwdkF5zu**

**OAGBEJnCwilmg0BvXhSuCkqE+IYyDZsFWd4Q3jMzcsCY4nvvVUEdAO1ny1hOEYUn**

**P5ppYQm6e9+pzXJqzRQ3jEaQyiTN8bwVTy7h2WA9wIRnZKvnZ5Jom6q6cE6B**

**-----END CERTIFICATE-----**

**subject=/C=US/ST=SD/L=Madison/O=DSU/OU=BIS/CN=INFA723/emailAddress=yong.wang@dsu .edu**

**issuer=/C=US/ST=SD/L=Madison/O=DSU/OU=BIS/CN=INFA723/emailAddress=yong.wang@dsu. edu**

**---**

**No client certificate CA names sent**

**---**

**SSL handshake has read 1304 bytes and written 255 bytes**

**---**

**New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA**

**Server public key is 1024 bit**

**Secure Renegotiation IS supported**

**Compression: NONE**

**Expansion: NONE**

**SSL-Session:**

**Protocol : TLSv1**

**Cipher : DHE-RSA-AES256-SHA**

**Session-ID: 32B199A02E5FEF60FAB0E9D3319160F861FCE44CCD9A52CBF6878F41E22F9993**

**Session-ID-ctx:**

**Master-Key: 6A72528311A9B7BEAA1B4E45D65294732A81F810192995CDE9609D062D9698C0 4FBDC0729276325B5941679DF5F2BE07**

**Key-Arg : None**

**Start Time: 1333555358**

**Timeout : 300 (sec)**

**Verify return code: 18 (self signed certificate)**

**---**

Once the session is up, if you are familiar with the protocols between the client and server, you can manually issue the command and check the response. For example, you can try HTTP request, such as “GET /” or “GET /index.html”. Type the command “GET /” and check what is the response. (Think about why?)

1. Start a new session and connect to gmail.com using OpenSSL s\_client.

**$ openssl s\_client -connect gmail.com:443**

and type the command “GET /” after the session is initialized. Check the response from gmail server and think about why.

# 4 Question (30points)

1. (10 points) Find the certificate of the simple server you set up in the lab in internet explorer. Make a screen snapshot and include it in your lab assignment. Check the fields of the certificate.
2. (10 points) Many attacks have been reported targeting SSL/TLS. Conduct your research on the following two attacks and briefly explain how these attacks work and how to prevent these attacks:
   1. The Raccoon Attack
   2. POODLE Attack (Padding Oracle On Downgraded Legacy Encryption)
3. (10 points) What is public key infrastructure (PKI)? Are there any issues when using PKI for authentication for large-scale networks such as Internet of Things?