Steps to generate Keys and Self-signed Certificate using OpenSSL

To run the emulator with a session, a server.crt and server.key files are required to be present at the same level as emulator.py in the directory. To generate a self-signed certificate and private key go through the following steps:

1. Check if you have OpenSSL installed in your system.

To download OpenSSL: [https://www.openssl.org/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbDVZNmcxQ2w5Y1BHYWlXLXgwS0RHU3ZWY1l0QXxBQ3Jtc0trT0xvQ1RMbF9wOUZOc3Y5TDNCUDJrajlCdVMyUjZsSFNwdWpacFlDS1RxT3hHM3pxRTJOeXU0ZndSMDUwSDFXWDR2SjhNSWs2OGZueFZUX0VlNEtWaV9VcUZaX3drSGdiYVRObDEtbUkwMmhveFZBRQ&q=https%3A%2F%2Fwww.openssl.org%2F&v=wzbf9ldvBjM)

Once installed, to check the success, try “openssl version -a” command in the command prompt

1. Now create key pair using the RSA algorithm and the appropriate key length. This is a private key file.

Go to the Swordfish emulator folder where you want to create keys and certificates.

Run: openssl genrsa -out server.key 2048

1. Now extract the public key from the previously generated key pair file.

Run: openssl rsa -in server.key -pubout -out server\_public.key

1. Now create a certificate signing request (.csr file) before generating the certificate file.

Run: openssl req -new -key server.key -out server.csr

Here, answer the questions which are required to generate .csr file.

For example, country name, state name, common name, email id, etc.

1. To verify the csr file before signing it run the following command.

Run: openssl req -text -in server.csr -noout -verify

1. Once verified, now generate a self-signed certificate with a validation period.

Run: openssl x509 -in server.csr -out server.crt -req -signkey server.key -days 365

When you get the “Signature ok” message, your certificate is signed successfully.