The Hong Kong Polytechnic University

COMP 3334 – Computer Systems Security (Semester 2, Winter 2024)

Lab 2 – Part 2 – X.509 certificates

In this second part of the lab, you will explore web certificates and how to generate certificates.

Preliminaries

For this lab, you will need to download the openssl tool.

1. Download OpenSSL v3 from Blackboard:
   https://learn.polyu.edu.hk/bbcswebdav/xid-55258061_1
   It contains a recent 64-bit binary of OpenSSL for Windows. If your browser warns you about a the fact that the file looks suspicious, ignore the warning and keep downloading the file. This binary is taken from Git SCM.

Extract the archive at a convenient location (e.g., Desktop). Next, launch a command prompt, and change the currently working directory to that location (e.g., `cd Desktop`). You should be able to run OpenSSL with "`bin\openssl version`", and it should display "OpenSSL 3.2.1".

Exercises

Exercise 2: Web certificates [45min]

How to make sure that you are visiting `google.com` and not another website? The TLS protocol ensures confidentiality of communications as well as authenticates the server (and possibly also the client) using certificates. Certificate are a set of public key and identification information about the bearer of the public key. A trusted third-party needs to validate that the identity in the certificate is correct and *sign* the association between the identity and the public key. When a certificate is distributed, someone can verify the signature of the trusted third-party and get the assurance that the public key is that of the identified entity.

Trusted third-parties are called Certificate Authorities. Their job, on the web, is to validate the ownership of domain names. If somebody can prove they have control over a domain name, e.g., by hosting a file on the web server or changing DNS entries, then they must be the owner of that domain and therefore they are able to obtain a certificate for that domain name that binds the domain name to a public key of their choice.

Your job

Complete the following tasks.

Task 1. For the provided certificate `polyu.edu.hk.crt`, parse the content using openssl. On a command line, use:

```
openssl.exe x509 -in certificatefile -noout -text
```

certificatefile is the certificate file path.

The output represents what's encoded in the certificate. The subject is "who this certificate has been issued to". It contains various information about the organization, and the main domain name this certificate is bound to. The issuer is the organization that issued the certificate after verifying that the requester owns the domain name. The certificate then includes a public key, in our case a 2048-bit RSA public key. At the bottom of the output, you will see the signature of the issuer on this certificate.

Answer the following questions:

1. Until when this certificate is valid? Copy the output of the corresponding line on the answer sheet.

Task 2. We now would like to generate our own certificate. Since we cannot bother a real Certificate Authority (and given that you do not own any domain name for the purpose of this course), we will first create our own CA.

The CA possesses its own certificate. In this case, nobody else vouches for them, therefore their certificate is *self-signed* (signed by themselves). CA certificates are provisioned in browsers and operating systems to bootstrap the trust chain.

Use the following command line to generate an elliptic curve public/private key pair. Elliptic curves keys are preferred over RSA keys due to various reasons, including a smaller key length and (so far) fewer attacks against implementation issues.

Use the following command to create an elliptic curve (EC) private key file:

```
openssl ecparam -out <outfile> -name prime256v1 -genkey
```

You will need to adapt the outfile, which is the private key file generated.

Create a Certificate Signing Request (CSR), which is a document where you bind the public key and your identity information.

```
openssl req -new -key <keyfile> -out <CSR> -config openssl.cnf
```

Next, use the following command to create a self-signed certificate using the provided private key. Note that the CA certificate is self-signed, i.e., it is signed by its own private key. We use `-config openssl.cnf` to point to a configuration file for OpenSSL, which is given to you.

```
openssl.exe req -x509 -key <keyfile> -in <CSR> -out <certificate>
-days 365 -config openssl.cnf
```

Adapt the keyfile to point to the private key generated above, as well as the output file. A self-signed certificate will be output and will be valid for a year.

Then, generate a new private key for your domain certificate:

```
openssl ecparam -out <outfile> -name prime256v1 -genkey
```

Create a CSR for your domain certificate, which will be submitted to the CA.

```
openssl req -new -key <keyfile> -out <CSR> -config openssl.cnf
```

The CSR needs to be signed by your private key, which you should specify as a parameter.

You will be prompted for a few things, including your organization name, and a common name. The Common Name is what will appear in the Subject line. It usually refers to the domain name you want to issue the certificate for. **Choose the following domain name:** YOURSTUDENTID.polyu.edu.hk

Finally, as a CA, you will now be able to sign the CSR by the CA's private key and create a (valid) certificate. Run the following command:

```
openssl x509 -req -in <CSR> -CA <CA> -CAkey <keyfile> -CAcreateserial
-out <CRT> -days 365
```

The command takes as input the CSR, the CA certificate, the CA private key, and outputs a X.509 certificate.

Upload your CA certificate and domain certificate to Blackboard!

Need help?

Teaching Assistants you can contact about this lab:

1. Xuyuan CAI, xuyuan.cai@connect.polyu.hk, exercise 1