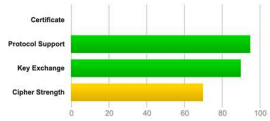
Lab 4: Trust and Digital Certificates

Objective: Digital certificates are used to define a trust infrastructure within PKI (Public Key Infrastructure). A certificate can hold a key pair, while a distributable certificate will only contain the public key. In this lab we will read-in digital certificates and analyse them.

📖 **Lab demo:** https://youtu.be/-uNQFv0GTZc

A Introduction

No	Description	Result
A.1	From: 📖 **Web link (Digital Certificate):** `http://asecuritysite.com/encryption/digitalcert` Open up Certificate 1 and identify the following:	Serial number: Effective date: Name: Issuer: What is CN used for: What is ON used for: What is O used for: What is L used for:
A.2	Now open-up the ZIP file for the certificate (Certificate 3), and view the DER file.	What other information can you gain from the certificate: What is the size of the public key: Which hashing method has been used: Is the certificate trusted on your system: [Yes][No]
A.3	Make a connection to the **www.live.com** Web site: `openssl s_client -connect www.live.com:443`	Can you identify the certificate chain? What is the subject on the certificate? Who is the issuer on the certificate?
A.4	Google moved in July 2018 to mark sites as being insecure if they did not have a match between their digital certificate and the site. A scan, at the time, on health and social care sites	Outline three sites that still have problems with their digital certificate, and the reason for the problem (you perhaps should try Chrome to assess):

	from the following page showed problems in digital certificates: https://bit.ly/2EkUvX0	Pick two sites that you feel are not setup properly for their digital certificate, and then run a scan from SSLLabs (www.ssllabs.com). Identify the problems that they have with their digital certificate: What are their SSLLabs rating? Can you find a site with an "T" rating?  

A.5 Which the certificates in A.2, for Example 2 to Example 6. Complete the following table:

Cert	Organisation (Issued to)	Date range when valid	Size of public key	Issuer	Root CA	Hash method	Is it trusted?
2							
3							
4							
5							

6							

A.6 Now download the DER files from:

🕮 **Web link (Digital Certificate):** http://asecuritysite.com/der.zip

Now use openssl to read the certificates:

```
openssl x509 -inform der -in [certname] -noout -text
```

B Creating certificates

Now we will create our own self-signed certificates.

Step 1: Create a Private Key for the CA
Step 2: Create a Self-Signed Certificate for the CA (MegaCorp)
Step 3: Create a Subordinate (Intermediate) CA
Step 4: Generate a CSR for the Subordinate CA
Step 5: Inspect the CSR File
Step 6: Root CA Signs the Intermediate CA's CSR
Step 7: Create a PKCS#12 File for Signing
Step 8: Convert Between Formats

No	Description	Result
B.1	Create a Private Key for CA. Can do this using OpenSSL *openssl genrsa -out ca.key 2048* – genrsa → generates an RSA private key – -out ca.key → saves the key to a file named ca.key 2048 → key length in bits (can be 4096 for higher security)	
B.2	You can generate the certificate using **OpenSSL**: *openssl req -new -x509 -days 1826 -key ca.key -out ca.crt* **Explanation:** – req → create a certificate request – -new -x509 → generate a self-signed X.509 certificate – -days 1826 → certificate validity (5 years) – -key ca.key → use the private key created earlier -out ca.crt → output file (the new CA certificate)	
B.3	You can generate a private key using **OpenSSL**: *openssl genrsa -out ia.key 2048* **Explanation:**	

	– genrsa → generates an RSA private key – -out ia.key → saves the key to a file named *ia.key* – 2048 → key length in bits (can be 4096 for higher security)	
B.4	We now create a **Certificate Signing Request (CSR)** — a file containing identity details and the public key. *openssl req -new -key ia.key -out ia.csr* **Explanation:** • req -new → creates a new certificate request • -key ia.key → uses the subordinate CA's private key • -out ia.csr → saves the request in *ia.csr*	
B.5	After generating the CSR, you can open it to view its encoded content: *cat ia.csr* **Explanation:** • The CSR begins with -----BEGIN CERTIFICATE REQUEST----- and ends with -----END CERTIFICATE REQUEST----- *openssl req -in ia.csr -noout -text* **Explanation:** Openssl: The OpenSSL command-line tool (used for cryptography, keys, and certificates). Req: This subcommand handles certificate requests (CSRs). It can both create and inspect them. -in ia.csr: Tells OpenSSL to read input from the file ia.csr — your CSR file. -noout: Means "don't output the encoded base64 content again." Without this, OpenSSL would print both the base64 data and the decoded info. -text: Displays the decoded details of the CSR in a human-readable format.	
B.6	Use the **Root CA** to issue a certificate for the **Intermediate CA**: *openssl x509 -req -days 730 -in ia.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out ia.crt* **Explanation:** – -req → process a certificate signing request – -days 730 → validity period (2 years)	

	– -CA & -CAkey → specify the Root CA's certificate and private key – -out ia.crt → output subordinate certificate	
B.7	To combine the key, certificate, and chain into a single file: *openssl pkcs12 -export -out ia.p12 -inkey ia.key -in ia.crt -chain -CAfile ca.crt* • Used for **digital signing** and **verification** (e.g., code signing). – ca.key – **ca.crt** – ia.key – ia.crt	
B.8	Convert binary .crt to Base64 .cer for email or web use: *openssl x509 -inform pem -outform pem -in ca.crt -out ca.cer* *openssl x509 -inform pem -outform pem -in ia.crt -out ia.cer*	

What I should have learnt from this lab?

The key things learnt:

- Understand how digital certificates are generated and ported onto systems.
- Identifying problems with digital certificates on sites.