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: `702958`
		
		Effective date: `4/24/2008 8:18:42 PM`
		
		Name: `CN=Fred Smith, OU=None, E=fred@home, O=Nowhere, L=Edinburgh, S=Lothian, C=GB`
		
		Issuer: `CN=Fred Smith, OU=None, E=fred@home, O=Nowhere, L=Edinburgh, S=Lothian, C=GB`
		
		What is CN used for:
		`The Common Name – the primary name of the certificate holder, typically the domain name or person's name`
		What is ON used for:
		`Organizational Unit – identifies the department or group within the organization`
		What is O used for:
		`Organization – the company or legal entity owning the certificate`
		What is L used for:
		`Locality – the city where the organization or certificate owner is located`

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:
		`Signature Algorithm: sha256WithRSAEncryption`
		`Issuer: Google Trust Services LLC (GTS CA 1C3)`
		`Key Usage: Digital Signature`
		`Extended Key Usage: TLS Web Server Authentication`
		
		What is the size of the public key: `256 bit`
		
		Which hashing method has been used: `SHA-256`
		
		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? `Yes`
		
		What is the subject on the certificate?
		`subject=C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=outlook.live.com`
		
		Who is the issuer on the certificate?
		`issuer=C=US, O=DigiCert Inc, CN=DigiCert Cloud Services CA-1`

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

https://www.capability-scotland.org.uk
Expired: Sunday 13 November 2022 at 23:59:59

https://www.heartstroketayside.org.uk
localhost (is not trusted)
Root certificate authority

https://www.travax.scot.nhs.uk
Expired: Saturday 1 November 2025 at 14:57:22

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:

https://www.capability-scotland.org.uk
Sun, 13 Nov 2022 23:59:59 UTC (expired 2 years and 11 months ago) EXPIRED

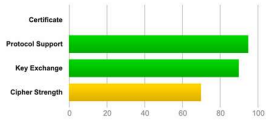
https://www.heartstroketayside.org.uk
Alternative names – INVALID

What are their SSLLabs rating?

https://www.capability-scotland.org.uk
T Rating

https://www.heartstroketayside.org.uk
T Rating

Can you find a site with an "T" rating? Yes



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	No One / Nowhere Ltd (CN=No One)	Oct 29 2011 – Oct 28 2013	1024-bit RSA	No One (self-signed)	None (self-signed)	sha1WithRSAEncryptio	No
3	Google (CN=*.google.com)	Feb 08 2023 – May 03 2023	256-bit EC (P-256)	GTS CA 1C3	Google Trust Services Root	sha256WithRSAEncryptio n	No
4	Cisco Systems (CN=www.cisco.com)	Jul 10 2012 – Jul 11 2013	1024-bit RSA	VeriSign Class 3 Secure Server CA – G3	VeriSign Root	sha1WithRSAEncryption	No
5	Microsoft Corporation (CN=microsoft.com)	Jan 13 2023 – Jan 08 2024	2048-bit RSA	Microsoft Azure TLS Issuing CA 05	Microsoft Root CA	sha384WithRSAEncryption	No

6	Oracle Corporation (CN=oracle.com)	Feb 14 2023 – Feb 26 2024	2048-bit RSA	DigiCert TLS RSA SHA256 2020 CA1	DigiCert Root	sha256WithRSAEncryption	No

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)	A 2048-bit RSA private key stored in ca.key This key will be used to sign other certificates (it is the Root CA's private key) This command generates a new RSA private key for the Certificate Authority (CA). genrsa creates an unencrypted key, and -out ca.key saves it as a file
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)	Generated a self-signed CA certificate valid for 1826 days Output file: ca.crt This certificate includes the public key + issuer information This creates the Root Certificate. Because no external CA signs it, -x509 makes it self-signed, meaning it signs itself using ca.key.
B.3	You can generate a private key using **OpenSSL**: *openssl genrsa -out ia.key 2048* **Explanation:**	Created a 2048-bit RSA private key for the intermediate authority File ia.key will be used to create a CSR next This is the private key for the subordinate CA, also called "Intermediate CA."

	– 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*	Output file: ia.csr Contains: Intermediate CA's public key Identity information Intended certificate fields A CSR is like an "application form" sent to a CA to request a certificate. It contains the public key and subject (organisation) information, but not a signature yet. A CSR is like an "application form" sent to a CA to request a certificate. It contains the public key and subject (organisation) information, but not a signature yet.
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.	Shows: Subject information (organisation, common name) Public Key (2048-bit RSA) Signature (CSR signature) Algorithm used Proof the CSR was signed using ia.key -text makes the CSR human-readable. This confirms what information will appear in the issued certificate.
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)	Output: ia.crt (Intermediate CA certificate) Valid for 730 days (2 years) Signed by the Root CA (ca.crt + ca.key) The certificate now has: Issuer: Root CA Subject: Intermediate CA This step turns the CSR into a real certificate. It is signed by the Root CA, establishing a trust chain.

	– -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	Created a file: ia.p12 Contains: Intermediate CA private key Intermediate CA certificate Root CA certificate (the chain) PKCS#12 files (.p12 or .pfx) are used for: Importing keys into browsers or servers Code signing TLS authentication This bundles everything into one password-protected file.
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*	ca.cer and ia.cer created in Base64 (PEM) format These are easier to distribute or email Same certificate data, different encoding PEM format (.cer) is Base64 encoded and includes the header lines: -----BEGIN CERTIFICATE----- -----END CERTIFICATE----- This makes the certificate readable and portable.

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.