**HTTP and HTTPS**

HTTPS is HTTP run over SSL or TLS

* Provides security services of TLS
* Optional client or browser authentication

When to access a website with https, the webserver will start the task to invoke ssl and protect the communication

* Server send message back to client, indicating secure session is established
* The client then sends its security parameters
  + The key exchange, algorithm, certificate etc.
  + Compares the clients to its own until it finds a match which is called **handshaking**
* The server authenticates the client by sending it a digital certificate
* The server will then generate a symmetric session key (AES) and encrypts it with the server’s public key
* This key is then sent to the webserver where both the client and the webserver will use the symmetric key to encrypt the data they send back and forth

Bob needs to be authenticated as bob

* If a man is sat in the middle, then he could send a fake digital key pretending to be bob
* Hashes and digital signatures are used within digital certificates as a method of authentication

The same as when HTTPS is used

* The server will have a public key which is used to exchange session keys to start encryption
* However, that public key needs to be authenticated
  + To make sure its legitimate
  + One solution is to **use digital certificates** which are digitally signed by a chain of trust

X.509 is the standard used for digital certificates

* Digital document that contains information about the owner of the certificate or website being certified

The public key and digital signature are validated by a trusted authority

**i.e., DigiCert**

SHA fingerprint is just a unique hash or number for the specific certificate

the public keys can be found in the certificate details (e.g., https://Mozilla.com)

* If anything is encrypted with this private key under the same encryption protocol/algorithm e.g., RSA, then only Mozilla’s private key can decrypt it

**A digital signature is a value hash that has been encrypted with the issuers private key** (DigiCert)

* In order to confirm that certificates are trustworthy, then the fingerprint must be decoded using the public key of the issuer and the decryption must match what’s on the issuers certificate

HTTPS is only as strong as its weakest link

* Issuance of fake certificates that your browser trusts
* This means your HTTPS could be intercepted and read
* The traffic will be sent and encrypted as normal
* The padlock will still be there, and everything will look fine
* But whoever issued the fake certificate can decrypt the traffic as they have the private key

**Certificate authorities always screw up, so it is necessary to do your own checks** (e.g. Symantic)

The tree of trust for certificate authorities

* Over 1400 authorities trusted my Microsoft, Mozilla etc
* Even the Honk Kong post office, the US dept. of homeland security, US defence contractors etc
* **Nation States will have influence of CAs, if not issue out certificates themselves and say they are whoever and your browser would trust that certificate**
* **The 14 eyes are all likely to issue certificates that your browser would trust**

There can vulnerabilities in the process of getting certificates

* Null byte poisoning, where you’re able to get certificates for domains that you don’t own
* Nation States will defo be working on new ways to subvert the process of getting certificates and getting false ones

SSLsniff

Certificate patrol

* An add-on that shows when certificates are updated so you can ensure it was a legitimate change

Pinning

* You can associate a host with their expected X.509 certificate or public key
* This will make the host only accept one public key so that if it is changed, then no data will be exchanged
* This is a security measure used in banking apps as they only need to connect to a couple of websites so they “lock out” any other sites or public keys