TLS

A dive into what makes TLS work

What is TLS?

- TLS is a protocol to encrypt and authenticate application layer traffic.
- When implemented correctly
  - TLS guarantees the identity of a server
  - May guarantee the identity of a client
  - Traffic between the server and client cannot be read by a third party
- TLS (Transport Layer Security) is the successor to SSL (Secure Sockets Layer)
- This talk will focus on TLS 1.2 and TLS 1.3

Where is TLS used?

- HTTPS
- OpenVPN
- PEAP (used for WiFi authentication)
- Stunnel (allows tunneling any TCP traffic over TLS)
- A variant of TLS called DTLS is available for UDP
- QUIC uses TLS for reliably encrypted connections over UDP

Let’s review some crypto

Asymmetric Cryptography

- 2 keys
- Public keys
  - Publicly distributed
  - May be used to encrypt (depending on protocol)
  - May be used to validate signatures (depending on protocol)
- Private keys are kept secret
  - Secret (DON’T SHARE IT (not even with the certificate authority (we’ll get to that)))
  - May be used to encrypt (depending on protocol)
  - May be used to validate signatures (depending on protocol)
- Slower than symmetric crypto methods
- Usually limited message size
- Multiple algorithms
  - RSA (encryption and signatures)
  - ECDSA/DSA (signatures only)

Symmetric Cryptography

- 1 key
- Only trusted parties should have the key
- Fast (often hardware accelerated)
- Two types: block ciphers and stream ciphers
- Block ciphers have modes of operation: (ECB, OFB, CBC, GCM, CCM, CTR, etc.)
- USE AEAD algorithms (they authenticate messages)
- Signatures can be achieved via additional data in an AEAD cipher or with an HMAC
- Multiple popular algorithms
    - AES-CBC with HMAC
    - AES-CCM
    - AES-GCM
    - Chacha20-Poly1305

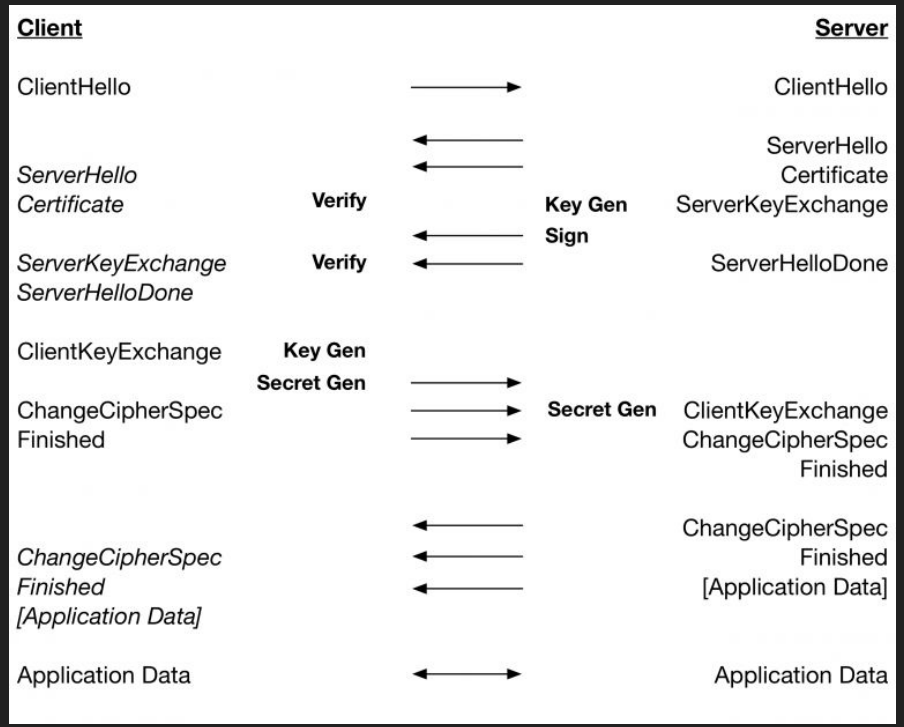
Key Exchange

- RSA encrypted key exchange
  - Alice generates a key, encrypts it with Bob’s public key and sends it
- Diffie-Hellman
  - Alice and Bob select a generator g
  - Alice selects secret A; Bob selects secret B
  - Alice sends Bob g^B; Bob sends Alice g^B
  - Alice computes g^B^A; Bob computes g^A^B
  - Alice and Bob hash this shared value
- When possible, use Diffie-Hellman for exchanging keys
- At least one of the following needs to happen
  - Alice needs to sign g^A with their certificate
  - Bob needs to sign g^B with their certificate

TLS handshake

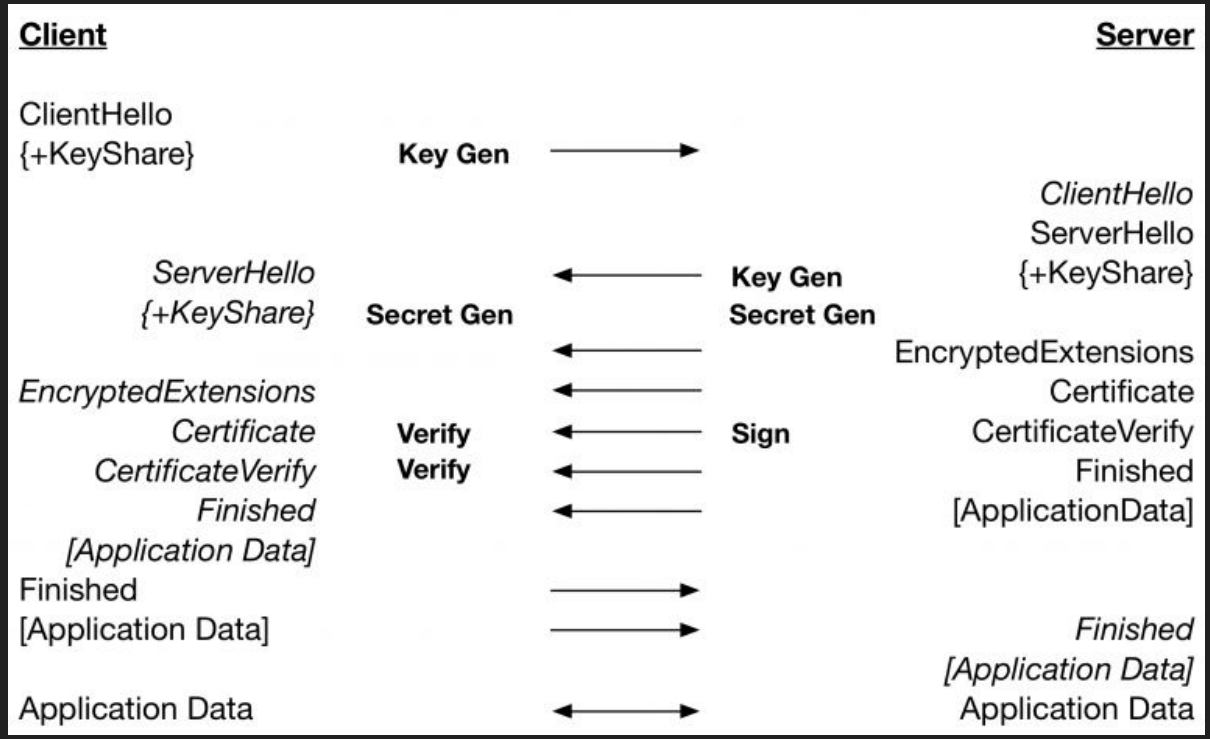
TLS 1.2 Handshake

- Needs TCP connection (1 round trip)
- 2 round trips
- Many cipher suites available
- Some are vulnerable, so only AEAD ciphers should be allowed
- DH should always be used for key exchange but it is possible not to



TLS 1.3 Handshake

- Needs TCP connection (1 round trip)
- 1 round trip
- Only AEAD ciphers
- Always uses DH for key exchange
- When resuming a connection 0 round trips can be used, but this is vulnerable to replay attacks and may limit forward secrecy protections



Handshake Cryptography

Client says hi

Server sends public key

Client generates secret, encrypts it with the server public key and sends it to the server

Both the client and server have a shared secret, and they start using symmetric cryptography

Forward Secrecy

The problem with the prior approach is that if an attacker is recording all traffic between the client and server, and they ever acquire the private key of the server (it’s a long term key), they can decrypt all the recorded traffic. In other words, key compromise allows a passive attacker to read all traffic.

Instead, the key exchange should take place via diffie hellman. The client and server exchange one time random values and derive a shared secret using diffie hellman. To prevent active attackers from tampering with the diffie hellman public keys, they should be signed by the long term keys.

How does an entity trust another’s public key?

Key Signing Parties

- Just ask for some ID
- No more MITM
- Problem solved!

Certificates

- Certificates are public keys attached to extra information
- Extra information allows someone to verify what the certificate holder should be allowed to do
- The public key authenticates an identity
- The certificate data authorizes an identity
- The owner of the certificate signs the data to prevent tampering
- A certificate authority attests to the validity of the data by signing the certificate

X.509

- X.509 is the standard for public key certificates
- Used for email (S/MIME), HTTPS (TLS), and more
- Common fields in a certificate:
    - Serial number
    - Crypto info
    - Expiration date
    - Not valid before date
    - Subject
        - Country
        - State
        - Locality
        - Organization
        - Organization Unit
        - Common Name (usually the domain name goes here)
    - Extensions
        - Key usage
        - Extended key usage
        - Basic Constraints
        - CRL Distribution Points
- The list is far from exhaustive

A certificate authority signs a certificate

How do you know to trust the certificate authority?

Sign the Certificate Authority certificate with another Certificate Authority

EZ, problem solved

Wait...

Certificate Validation

- The issuer of a certificate is a certificate authority
- Either the authority is a root authority in a trust store (it is trusted by the device)
- Or it is signed by another certificate authority (which you must also validate)
- The series of certificates to the root certificate is called a certificate chain

Trust Stores

- Trust stores contain certificates for certificate authorities
- These certificate authorities are trusted to issue end entity certificates and/or certificates for certificate authorities
- The root of a certificate chain must reside in a trust store in order for the end entity certificate to be accepted

Aside: Superfish

Case study in a compromised trust store

Superfish was an advertising company, and Lenovo bundled their certificates with laptops. Superfish software would intercept traffic locally on the computer to place ads on encrypted pages.

What’s the problem with this?

Since the superfish CA was trusted, and the same certificate was stored on multiple laptops, and the interception happened locally, every such laptop had the key to the root CA and could intercept traffic from other laptops trusting the CA.

Okay, so you just check that there’s a chain of certificates all the way to the root, right?

Nonononononono

Validation Oopsies Part 1

Let’s say, you’ve been issued a certificate for your website.

And now you decide to create another certificate.

And you sign this new certificate with the certificate you were issued.

Root -> Website -> New certificate

What if this new certificate is for google.com?

Is this a valid chain?

Validation Oopsies Part 1

NO! But how can someone tell?

In the X.509 extensions, there's one called Basic Constraints and one of the basic constraints is a flag that establishes if the certificate is a CA certificate.

Your website certificate should have that flag set to false. When validating a certificate chain, all signing certificates must be CAs. If that flag is false, the certificate cannot sign another certificate.

Is this easy to miss when writing your own validator? Has it happened in the past?

Yes and yes! DON'T ROLL YOUR OWN CRYPTO!!!

Okay, so just check that there’s a chain of CA certificates all the way to the root, right?

Nonononononono

Validation Oopsies Part 2

By validating the root, all you've done is assure yourself that the information in the certificate is accurate. You still need to check that information!

What does that mean? For browsers, it means checking that the domain name matches the common name of the certificate. Usually, the identifying information is in the subject name. For email, it means checking that the email address on the certificate matches the email address that you're receiving mail from.

This is also easy to miss when writing software. In fact, for WiFi authentication (WPA2 with PEAP), many clients miss this step.

How does one get a certificate?

Certificate Issuance

- Do NOT give the CA your private key
- Generate a private key (or use an existing one)
- Create a certificate signing request (CSR) including
  - the information you want included with the certificate
  - a signature
- The CA takes the CSR and creates a certificate if they agree that the information in the CSR represents you
- For websites, this process is automated via the ACMEv2 protocol
- Let’s Encrypt provides free DV (domain validated) certificates

What happens if the certificate is compromised?

Revocation: CRL

- Browsers should check if a certificate is revoked, but revocation comes with many challenges
- A CRL (Certificate Revocation List) is a list of certificates that have been revoked by a CA
- This is can be a massive list and can be problematic to download every time you want to check if a certificate is valid
- Intermediate CAs might be found on revocation lists
- A different protocol is used for end entity certificates

Revocation: OCSP

- End entity certificates are checked via OCSP (Online Certificate Status Protocol)
- The CA is queried to check if a particular certificate is valid
- Unfortunately, this means that the CA knows when an individual is validating a certificate, which could leak which website the user is visiting to the CA. This is a privacy concern
- This also puts a lot of load on the CA
- A bigger issue is that the CA must be online for the certificate check to work
- If the check fails, a browser will terminate the connection
- If a check times out, a browser will continue with the connection
- An attacker can block requests to the CA and trick the browser into accepting a revoked certificate

Does this mean the certificate revocation is hopeless?

Probably

But there is more...

Revocation: OCSP Stapling and Must-Staple

- Every now and then, the server queries the CA for proof that the certificate is valid
- The server then attaches this OCSP response to the certificate when sending the Server Hello and certificate
- This is called OCSP stapling
- Now clients don’t have to rely on the CA
- But an attacker can just use an expired certificate without attaching the OCSP response
- Browsers can be asked to reject certificates without a stapled OCSP response by configuring a flag in the certificate itself (when getting a certificate from the CA)
- This flag cannot be tampered with, since it is set by the CA
- Do browsers actually obey this? It depends on the browser
    - Chrome: No
    - Firefox: Yes

So what if a CA missisues a certificate?

There’s no hope and everybody freaks out for a bit if they ever detect it

Really?

Ish...

Certificate Transparency

Certificates issued for websites are logged in public append only logs. This allows people to monitor for misissued certificates. Browsers will reject certificates that are not in logs. The implementation involves Merkle trees.

CAA Records

CAA records are DNS records that prohibit certificate authorities from issuing certificates for a domain. CAs are supposed to check for CAA records before issuing a certificate.

What happens if my information changes?

Surely the CA should have to revalidate my identity every now and then?

Yup. Certificates are only valid between two dates.

Sounds great, right?

But how do you know what time it is?

NTP

Time is managed via the Network Time Protocol. Most uses of NTP are unauthenticated. Attackers can spoof the time. If the time on your computer is controlled by an attacker, checking certificate expiration is faulty. Browsers can check the time with an authenticated source and display an error if the computer clock is out of sync. A better authenticated time protocol is currently being drafted.

Let’s talk about key generation...

CSPRNGs

- RSA certificates depend on generating a random private key
- If there's not enough entropy during key generation, keys that have been used elsewhere may be generated
- Be wary of key generation
- https://www.cs.umd.edu/class/fall2018/cmsc818O/papers/ps-and-qs.pdf

So as long as website supports HTTPS, I’m secure right?

...

No

But seriously, let’s talk about HTTP...

SSL Stripping and HSTS

- SSL Strip intercepts the connection between a client and a server
- The client sees an HTTP connection
- The server sees an HTTPS connection
- The client makes an initial connection over HTTP
- HSTS headers tell the client to make the initial connection over HTTPS
- The first connection is still over HTTP
- There’s an HSTS preload list that’s hard coded into browsers. It has a list of domains that should only be connected to via HTTPS

Is there more?

Yes. A lot more.

But I can only talk for so long.

SSL Labs (https://ssllabs.com) will test your server for common configuration mistakes