Secure Channel

Security Protocol:

A distributed algo that sees parties exchange cryptographic messages

Assumption:

Cryptography is not broken

Attacker cannot break cryptography

Dolev-Yao Attacker

- Single powerful attacker who fully controls the network but cannot break cryptography
- Can do:
  - Drop any message
  - Eavesdrop any message
  - Derive info from any saved message
  - Inject new messages & replay saved messages
- Cannot break crypto.

Problems with key establishment

- Recall Diffie Hellman:
  - A generates x & builds $y_a = g^x \bmod p$
  - B generates y & builds $y_b = g^y \bmod p$
  - A sends B : y_a & viceversa
  - B computes $K_{ab} = (y_a)^y = (g^x)^y = g^{xy}$
  - A computes $K_{ab} = (y_b)^x = (g^y)^x = g^{xy}$

Now, is K_{ab} secret to the attacker?

- technically yes, but...
- What if M generates 2 & builds $y_m = g^2 \bmod p$
- It then injects it, and drops other
- Man-in-the-middle attack!

- Diffie Hellman fails at confidentiality!
- To fix the attack we would need Authenticated Diffie-Hellman.

Alice

- Gen $y_a = g^x \bmod p$
- Has a pair (sk_a, vk_a)

Bob

- Gen $y_b = g^y \bmod p$
- Has a pair (sk_b, vk_b)

$$\xrightarrow{\{A, B, y_a\}_{sk_a, vk_a}}$$

- Check signature using vk_a

Check sign using vk_b

$$\xleftarrow{\{B, A, y_b\}_{sk_b, vk_b}}$$

Digital Certificate

- Assume Bob stores vk_a, Would be a fix!
- Bob would need to store ALL verification keys, problem

- Better fix:

  Store one main vk that certifies others vk:

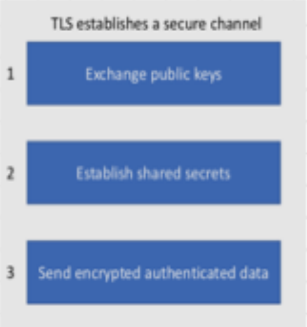
- Goal: Link a public key to its owner
- Anyone should be able to get certificate for the public key.
- Through digital signatures
- Standard format = X.509
- Every CA is certified by another (higher one)
- Top level is root certification authority (RCA)
- Who certifies the RCA? self-signes its certificate
- RCA public key:
  - Stored in the client machine
  - Installing new RCA PK should be a super guarded process!
- Issuing requirements:
  - Domain validated certificate
    - CA requires to prove control over some domain name
    - It is a cheap and automated process.
    - No legal entity bound to certificate
  - Extended Validation certificate
    - CA requires info from external sources.
    - Manual checks means more expensive
    - Legal entity bound
- Invalid certificates are the one that:
  - Have a wrong name
  - Are self-signed
  - Are expired
  - Have a weak cipher

  ⋮
- Certificate Revocation List:

  CRL's are lists of certificates not to be used
  Signed by same CA who issued certificate.
  Browser periodically access CA servers to fetch recent CRLs
- Online certificate status protocol (OCSP)
  - OCSP requests are sent to CA to know whether certificates has been revoked.
  - OCSP stapling → addended time-stamped OCSP response signed by CA
- The CA itself can be compromised!
- Goal of Certificate Transparency is to identify CA that maliciously issues certificates.

Transport Layer Security (TLS)

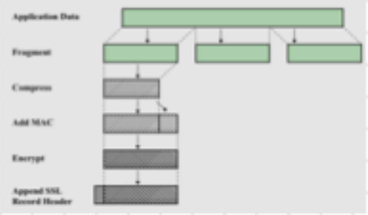
- It is the internet security protocol
- The S in HTTPS
- Goal: Provide confidentiality & data integrity between two parties!
  - Optionally provides authenticity
- Confidentiality is given by symmetric encryption
- Data integrity through MAC
- Authentication through digital signatures!

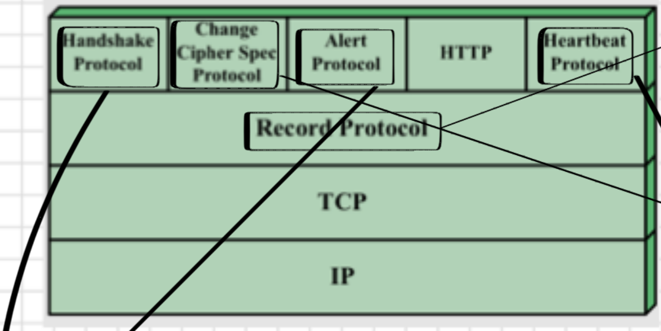

TLS establishes a secure channel

1 Exchange public keys

2 Establish shared secrets

3 Send encrypted authenticated data





Used to signal that the communication shifted from unencrypted to encrypted!

Switch to symmetric encryption in handshake msg with a single byte - 1

Used to signal TLS-related alerts

Fatal alerts: TLS conn immediately terminated

Warning alerts: TLS session may not be terminated

Used to check peer still alive

Gen activity during idle periods, avoid close

Not to be confused with Heartbleed.

Handshake Protocol

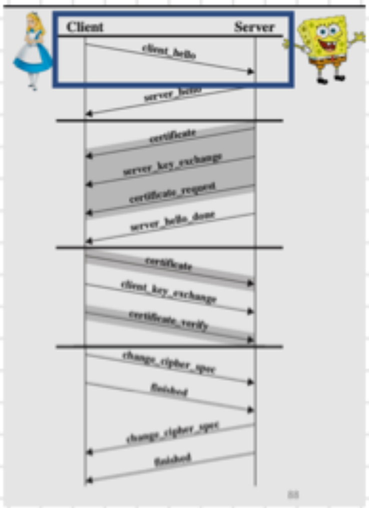
- Most complex, actual security protocol
- Establishes a master secret & derives secrets from it.
- Runs prior to any application data transmission.
- TLS 1.2 → 4 phases, 2 round trip
- TLS 1.3 → 2 phases, 1 round trip

TLS 1.2

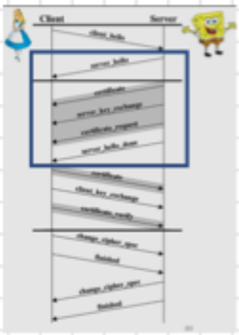
- Phase 1, RT:1
  - Establishes cipher suite
    - Client offers ciphers she supports

      It is a combination of:

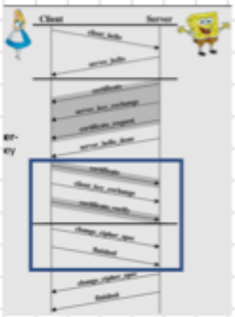
      * PK algos (RSA, DH_DSS, DH_RSA)
      * Symmetric-key algos (RC4, 3DES, AES)
      * HMAC algos (MD5, SHA-1, SHA-256)



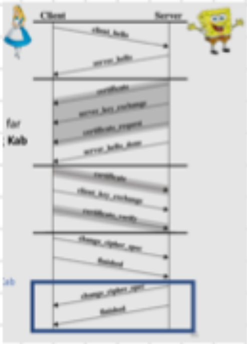
- Phase 2, RH: 1
    - Server chooses ciphers (RSA, AES, SHA-256)
    - Server may send its certificate
    - Server may request client's certificate

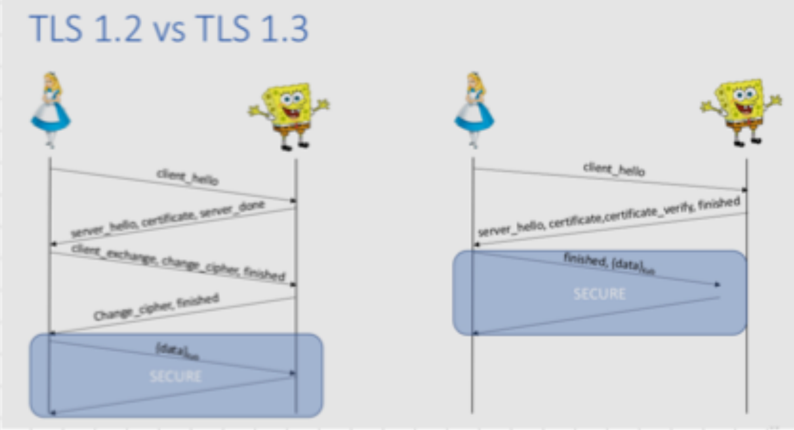


- Phase 3, RH: 2
    - client may send its certificate
    - client_key_exchange contains pre-master-secret
    - it is encrypted with server's PK



- Phase 4, RH: 2
    - finished contains hash of full traffic seen so far
    - From now on, everything encrypted,
    - Kab derived from the pre-master-secret.




TLS 1.2 vs TLS 1.3
client_hello
server_hello, certificate, server_done
client_exchange, change_cipher, finished
Change_cipher, finished
{data}Kab
SECURE
client_hello
server_hello, certificate,certificate_verify, finished
finished, {data}Kab
SECURE

- 1.3 comprises last parts
- faster & more secure!!!
- it removes many ciphers:

  SHA1, ~~RC4, DES, 3DES~~, AES-CBC, MD5

Attacks on TLS

- Beast → chosen plaintext attack
- Crime → cookie hijacking
- Breach → confidentiality attack
- Heartbleed → server memory overread
- Poodle → downgrade attack
- SMACK → message skipping attack
- FREAK → weak cipher export

🟥 Attack on TLS design

🟪 Attack on TLS implementations