Imperial College London

Network and Web Security

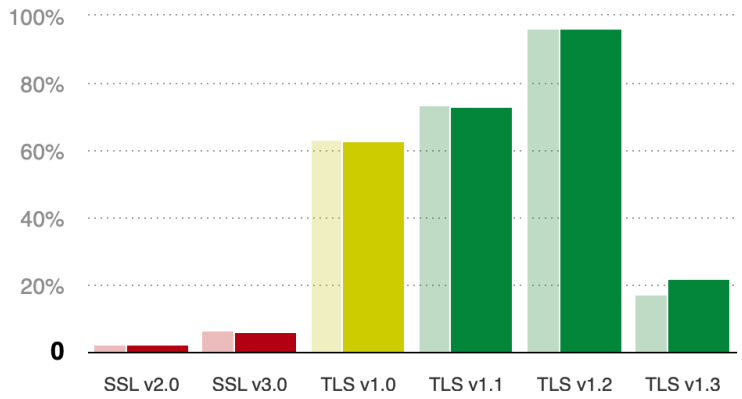
TLS

Dr Sergio Maffeis
Department of Computing
Course web page: https://331.cybersec.fun

Transport Layer Security (TLS)

Imperial College London

- A cryptographic protocol to provide confidentiality and integrity of network communications at the *session* layer
  - Eavesdropper sees unreadable ciphertexts
  - MITM tampering leads to integrity checks failure
- Used mainly to protect web (HTTPS) and email (STARTTLS, SMTPS)
- TLS should be sent over a “reliable medium”
  - Normally that is TCP/IP
    - DTLS (TLS over UDP) exists but it’s not widely used
  - TLS protects only TCP payload data
    - **IP** and **port** are **not protected**
- Latest version of TLS is 1.3, most adopted is 1.2
  - TLS 1.0 and 1.1 are insecure and deprecated
    - Chrome, Firefox, Safari, Edge announced to drop support in 2020
  - Before TLS there was SSL (nowadays the word SSL is mostly misused to mean TLS)



Alexa top 150k, 2020
Data by ssllabs.com

TLS certificates

Imperial College London

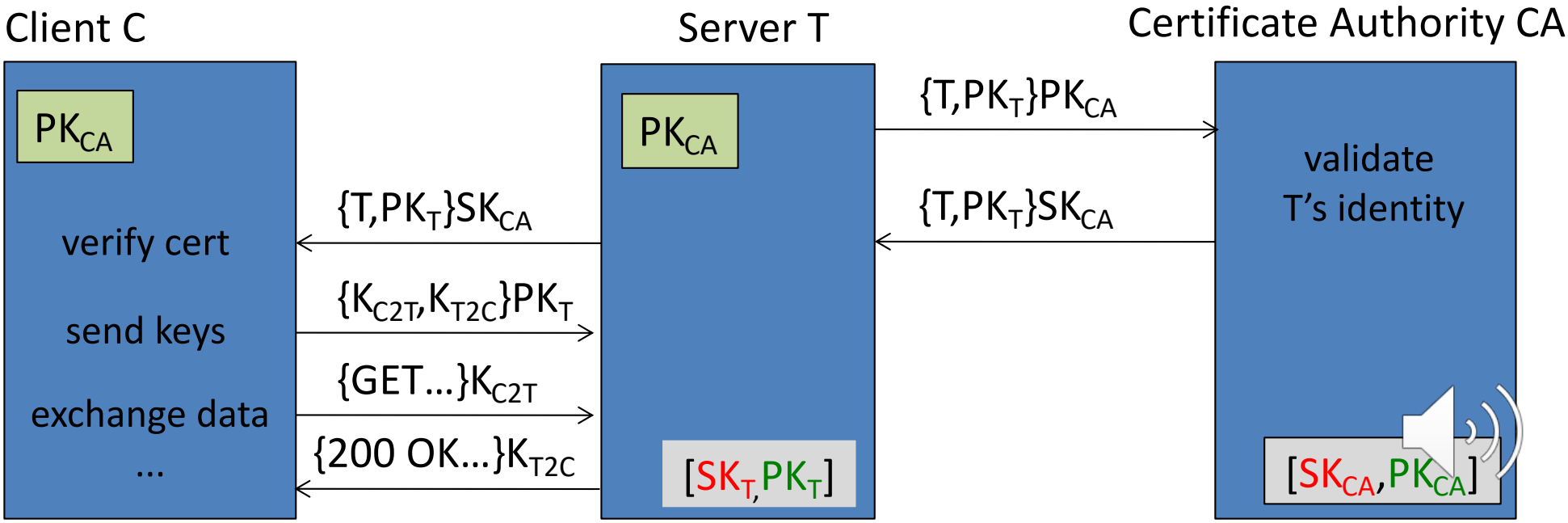
- A TLS server needs a certificate stating its identity and public key
  - X.509 Public Key Infrastructure Certificate, specified by IETF RFC 5280
- Main attributes
  - *Issuer*, typically a Certificate Authority (CA)
  - *Validity*, with start and end dates
  - *Subject*, identifying the owner
    - Typically an explicit domain name: `imperial.ac.uk`
  - *Subject Alternative Names*
    - Other domains covered by the certificate
    - For example `*.ic.ac.uk`
      - Matches `doc.ic.ac.uk`, but not `cate.doc.ic.ac.uk`
  - *Subject Public Key Info*, containing the actual public key
  - *Certificate Signature Value*, where the issuer signs the certificate body
- Certificate chains
  - TLS client normally trusts a number of widely known CAs
  - TLS server may send a certificate chain so client can verify ownership of public key

TLS: the main idea

Imperial College London

Public key cryptography refresher

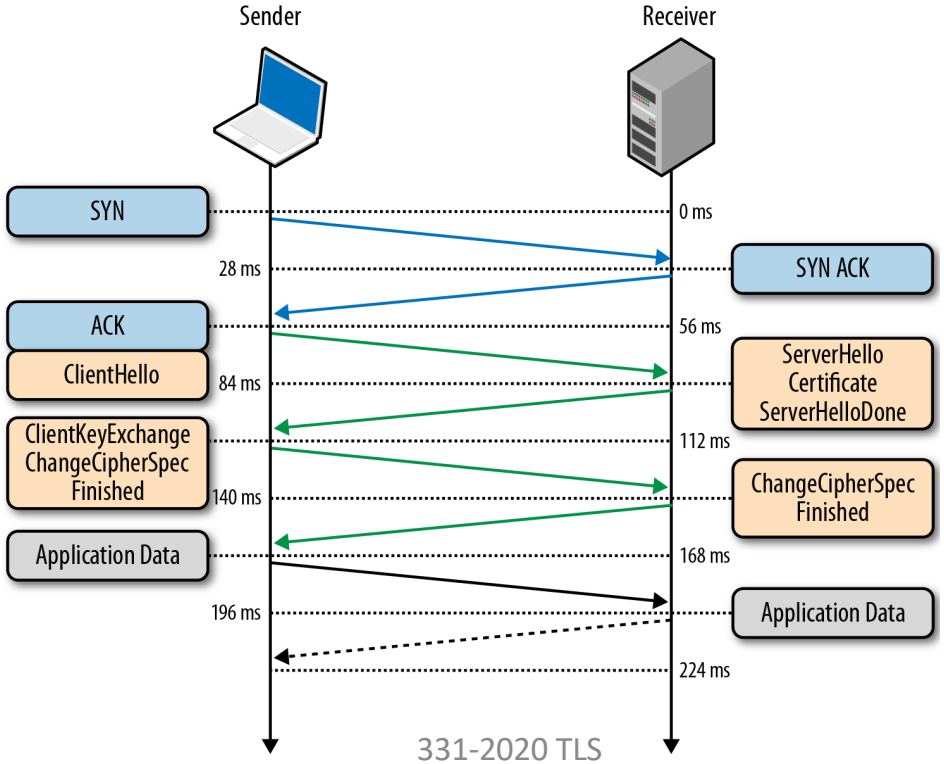
- $[SK_A,PK_A]$ is a *keypair* of cryptographically related keys
  - SK_A is secret (or *signing*) key of principal A, kept secret by A
  - PK_A is public (or *verification*) key, and can be revealed to other participants
- $\{msg\}PK_A$ denotes asymmetric **encryption** using the public key PK_A (only A can decrypt)
- $\{msg\}SK_A$ denotes a **signature** using the secret key SK_A (only A can generate)
- $Decode(K_1, \{msg\}K_2) = msg$ if and only if $[K_1,K_2]$ or $[K_2,K_1]$ is a valid keypair
- $\{msg\}K_L$ denotes symmetric (= fast) **encryption** using symmetric K labelled L



TLS 1.2 handshake

Imperial College London

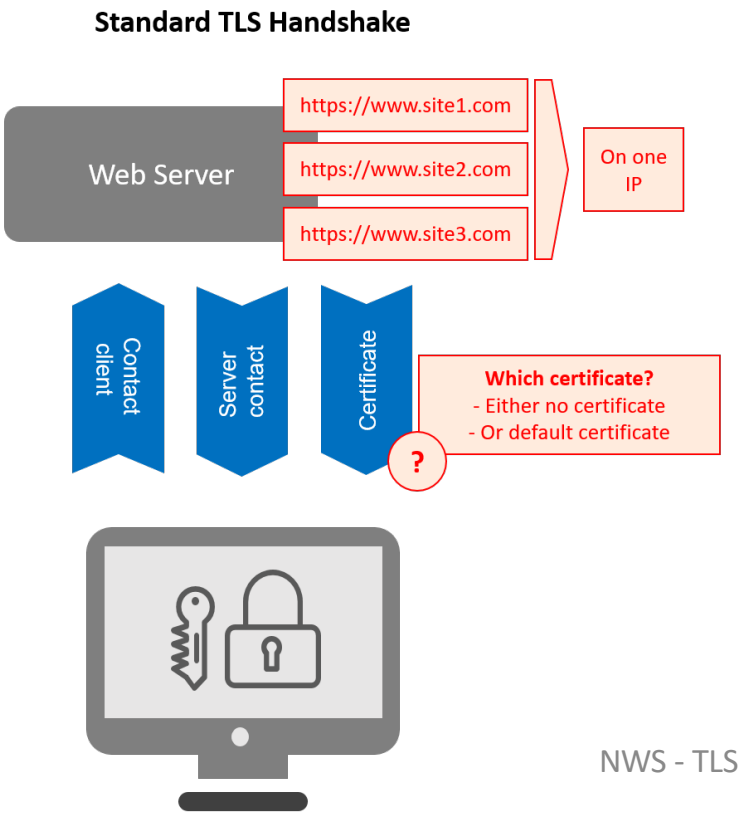
- Main messages of the TLS handshake
  - *ClientHello*: supported ciphersuites, compression methods, and protocol extensions
  - *ServerHello*: ciphersuite and compression method chosen by server
  - *Certificate*: chain of TLS certificates
  - *ClientKeyExchange*: information to allow server compute symmetric session key
  - *ChangeCipherSpec*: further messages will be encrypted with session key
- Note: none of these messages is encrypted
  - Although the session key itself is encrypted, see previous slide

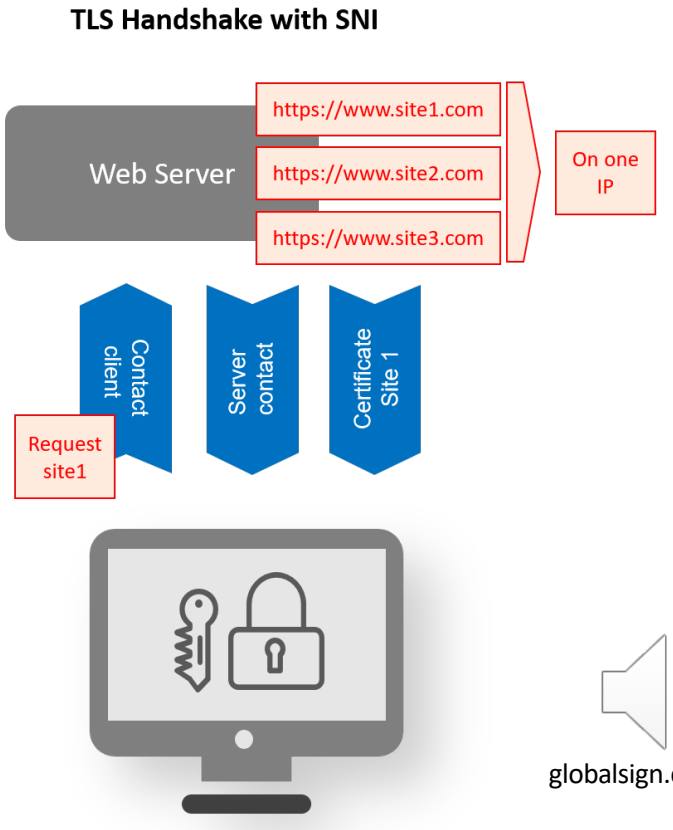


Server Name Indication

Imperial College London

- SNI is a common TLS extension to specify what site is requested from a virtual-hosting server or CDN
  - Careful clients should check that certificate name matches SNI
- Inferior alternative
  - CDN uses certificate covering multiple unrelated domains
  - Needs to be revoked any time a domain is added/removed
  - Leads to large certificates
  - Problem of mutual trust between domains



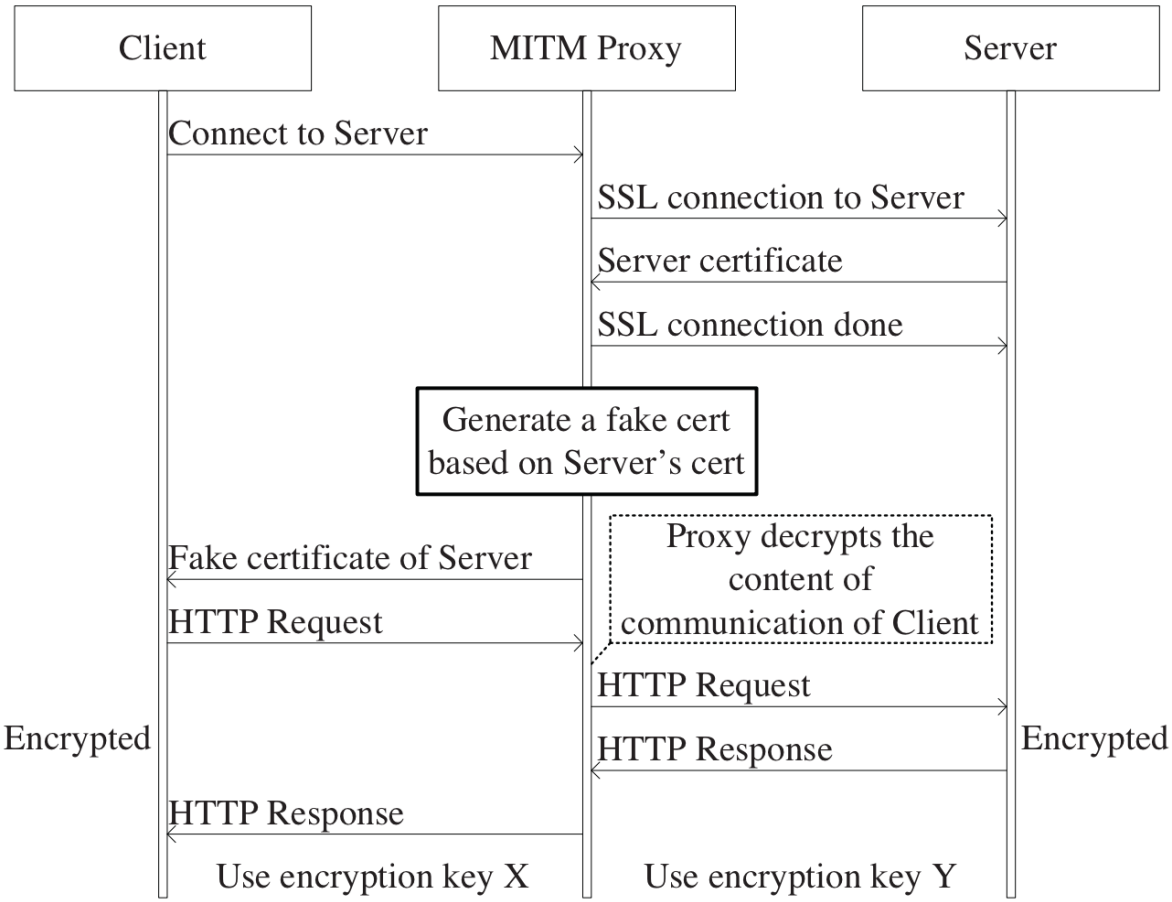


globalsign.com

Certificate trust

Imperial College London

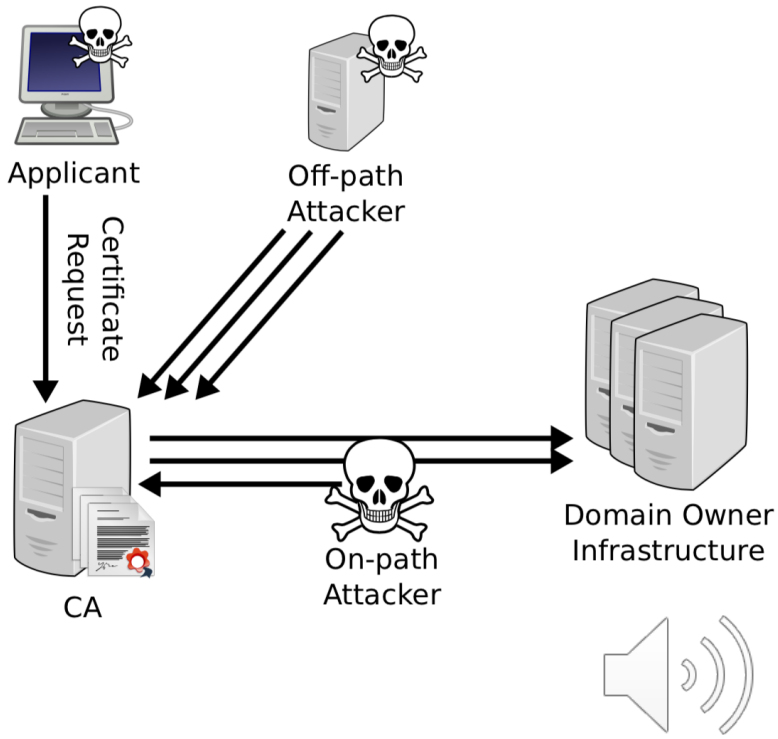
- Certificates are normally signed by a known Certificate Authority (CA)
  - By default TLS clients trust a number of reputable CAs
  - Buying and renewing certificates from CAs can be expensive
  - https://letsencrypt.org initiative provides free certificates, and automated renewal scripts
- Self-signed certificates (SSC)
  - Client must trust that self-signing key is appropriate for domain covered by SSC
  - Clients can trust a custom CA as a whole
    - Used to enable proxy to inspect TLS traffic in the clear (we do that in Kali)
- TLS protections against spoofing and MITM are ineffective when certificates cannot be trusted
  - Stolen private keys of certificates (Sony Hack, 2014)
  - Compromised CAs can sign spoofed certificates (DigiNotar 2011, Symantec 2015)



Certificate validation

Imperial College London

- In order to issue a new certificate, its parameters need to be verified
- *Extended Validation (EV)*
  - Used to verify the identity of the owner, or of an organisation
  - Offline-channels are common: by phone, in person, via lawyer
  - Different companies with same name may independently obtain EV for the same name
    - EV is now deprecated by browsers
- *Domain Validation (DV)* is the common case
  - Domain owner must prove control over domain when asking for certificate
  - In practice, DV is mostly Internet-based
  - CA generates a random token, then one of
    - Owner places token in DNS record for domain
    - Owner serves token at specific URL from domain
    - Owner includes token in fresh TLS certificate served from domain
    - CA emails token to owner, who submits challenge to CA online
- Attackers can compromise DV process and obtain rogue certificates
  - IP spoofing: fake DNS reply
  - DNS hijacking: serve challenge from malicious IP
  - Email snooping: read challenge from cleartext email
  - Main mitigations: use DNSSEC, encrypted emails
    - Example of “threat transfer”



Mitigating rogue certificates

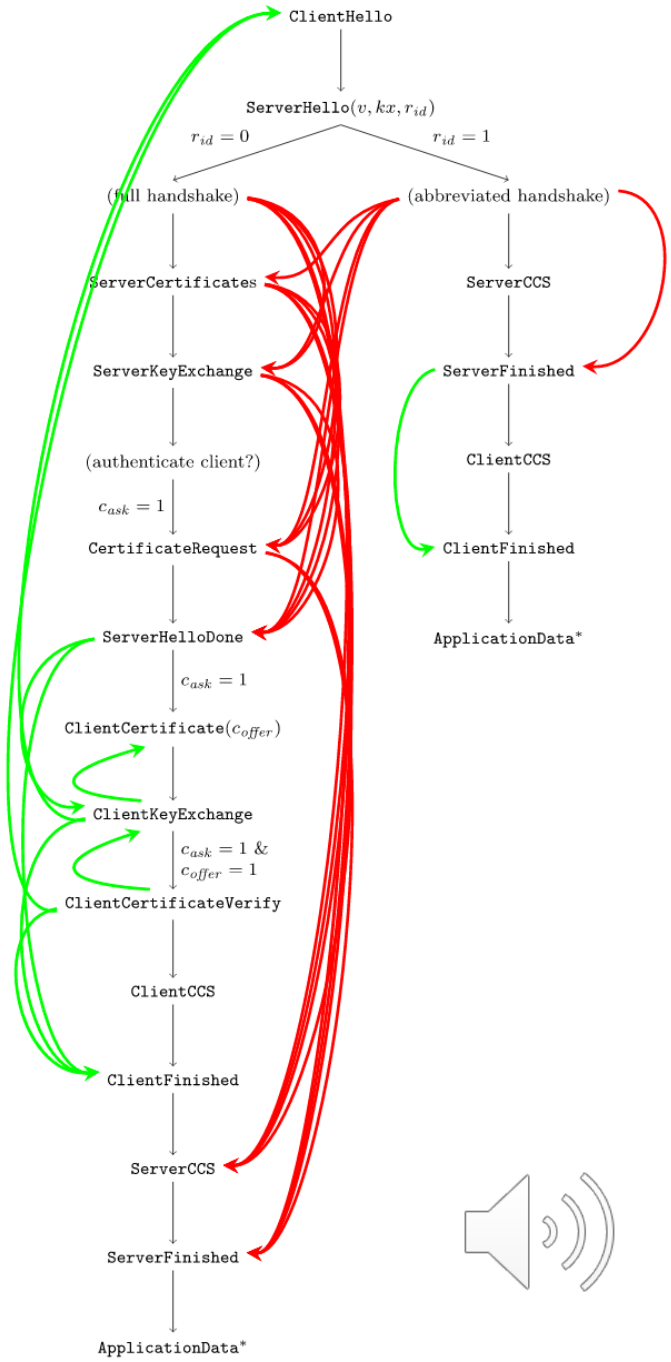
Imperial College London

- Compromised and rogue CAs undermine trust in TLS
  - Can issue rogue certificates for legitimate domains
- Certificate transparency
  - All certificates created by complying CAs should be reported in public logs
    - Merkle hash trees (key idea of blockchain)
  - Domain owners can monitor logs, detect rogue certificates, and have them revoked
- DANE
  - Rely on DNSSEC to control trust in TLS certificates
  - Domain owner can deploy trusted self-signed certificates
  - Possible to restrict acceptable CA or certificate for a domain
  - Trust moves from CAs to DNS operators
  - TLSA DNS records
    - 0 – CA specification: trust this public CA
    - 1 – Specific TLS certificate: trust, but verify, this certificate
    - 2 – Trust anchor assertion: trust this new CA
    - 3 – Domain-issued certificate: trust this self-signed certificate

TLS security issues

Imperial College London

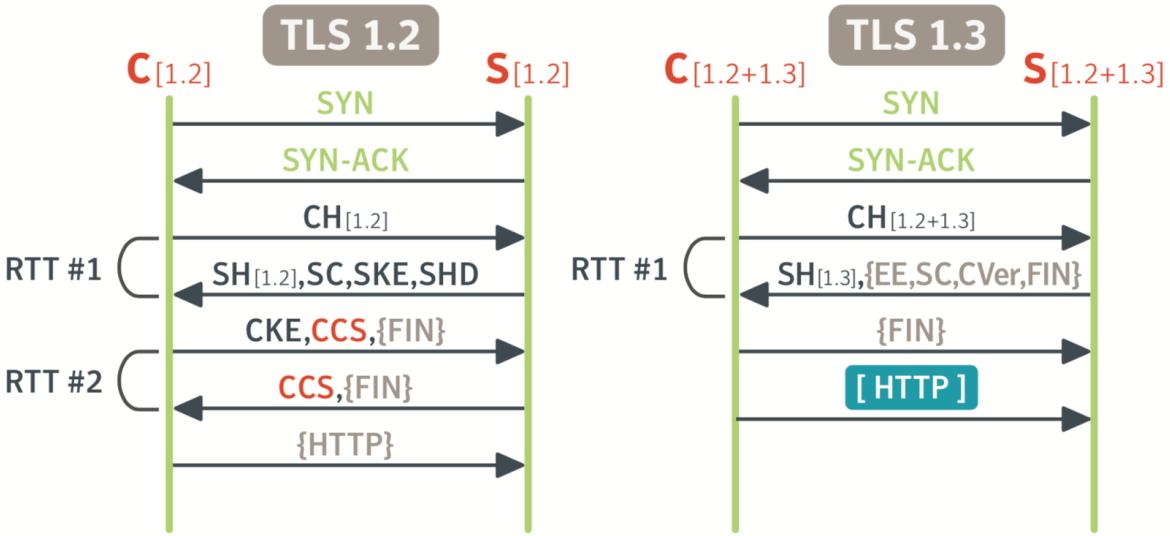
- TLS leaks information via traffic analysis
  - **BEAST** CVE-2011-3389: compromises TLS 1.0 via RC4 leakage
  - **CRIME** CVE-2012-4929: compromises SPDY via compression ratio
- OpenSSL implementation bugs
  - **HEARTBLEED** CVE-2014-0160: data disclosure due to buffer overrun
- Formal analysis of TLS state machine uncovered vulns
  - See https://mitls.org/pages/attacks/SMACK
  - **FREAK** CVE-2015-0204, CVE-2015-1637: force TLS client to use a weak ciphersuite
- And more have followed…
  - **SLOTH** CVE-2015-7575, **Sweet32** CVE-2016-2183, CVE-2016-6329, **DROWN** CVE-2016-0800, **ROBOT** (8 CVEs in 2017)
- Implementation issues keep surfacing in TLS clients
  - 30 TLS-related CVEs between Nov 2020 and Jan 2021



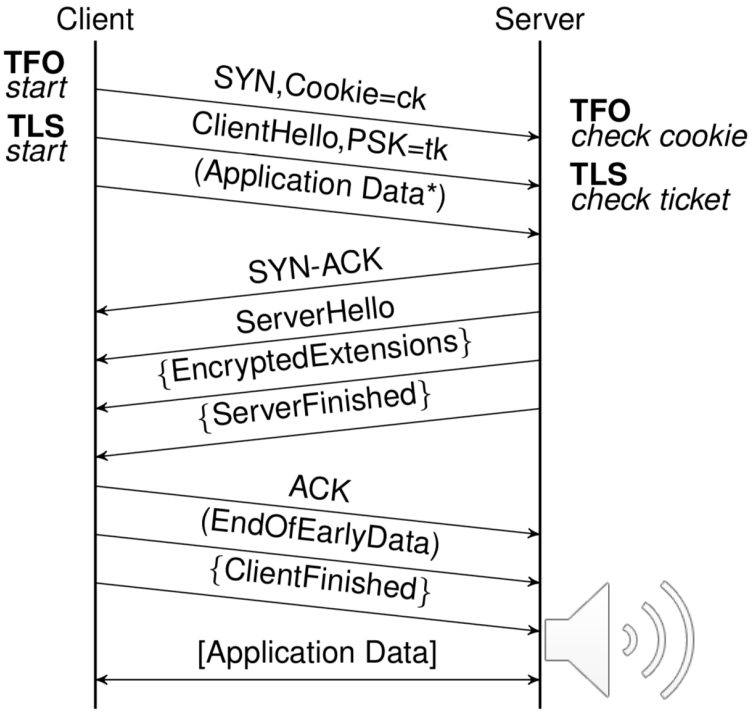
From TLS 1.2 to 1.3

Imperial College London

- TLS 1.2 connection is not efficient
  - 3 round trips (RTs) to establish session
  - 2 RTs to resume from accidental interruption
- TLS 1.3 uses one less RT in each case, over TCP
  - With TCP Fast Open (TFO), 0 RTs for resumption!
  - The trick is to save state ("cookies") when connection is established and send the state on resumption



Symantec



TLS 1.3 security

Imperial College London

- All recent TLS vulnerabilities are addressed, and in particular
  - Weak crypto suites are not allowed
    - MD5, SHA-1, DES, RC4, CBC encryption mode
  - TLS-level compression is not allowed
    - Protection from CRIME-like privacy leaks
  - Downgrade attacks can be detected (and stopped)
- Encrypted handshake
  - ClientHello include client public key
  - The rest of the handshake is encrypted
    - Including server certificate
- ESNI extension makes it possible to encrypt SNI
- ESNI and encrypted server certificate prevent MITM learning target domain
  - Stronger privacy protections
- TLS 1.3 interception in the enterprise
  - TLS 1.3 disallows RSA key exchange
    - In TLS 1.2, with server private key, IDS could decrypt traffic as eavesdropper
    - In TLS 1.3 client would have to keep sending session keys to IDS
  - MITM IDS could filter TLS 1.2 traffic based on black/whitelist policies
    - Block Facebook, don't intercept personal banking
  - With ENSI and encrypted certificates filtering is much harder
    - Filter based on DNS queries and/or IPs

Malicious TLS

Imperial College London

- Malware uses TLS (Trickbot, Emotet, Locky, Petya, Dridex, …)
  - Hide infection and command & control (C2) traffic
    - IDS cannot check for malicious signatures
  - Data exfiltration
    - DLP system cannot check outbound data for corporate secrets
    - Side channels: data exfiltrated by encoding them as certificates
- Traditional defenses
  - TLS traffic interception
  - Certificate blacklisting
- TLS fingerprinting
  - Salesforce's JA3: compute MD5 of some ClientHello parameters
    - TLSVersion, Cipher, TLSExtension, EllipticCurve, EllipticCurvePointFormat depend on SSL library and application
  - Turns out that it's a good fingerprint to classify what application started the TLS session
    - Also to detect malware: C2 traffic often uses custom parameters, with unique fingerprints
  - JA3S adds ServerHello parameters to improve classification
  - Limitations: not hard to spoof the fingerprint, plus usual problems of blacklisting solutions