Imperial College London

CO331 – Network and Web Security

8. TLS

Dr Sergio Maffeis
Department of Computing
Course web page: http://www.doc.ic.ac.uk/~maffeis/331

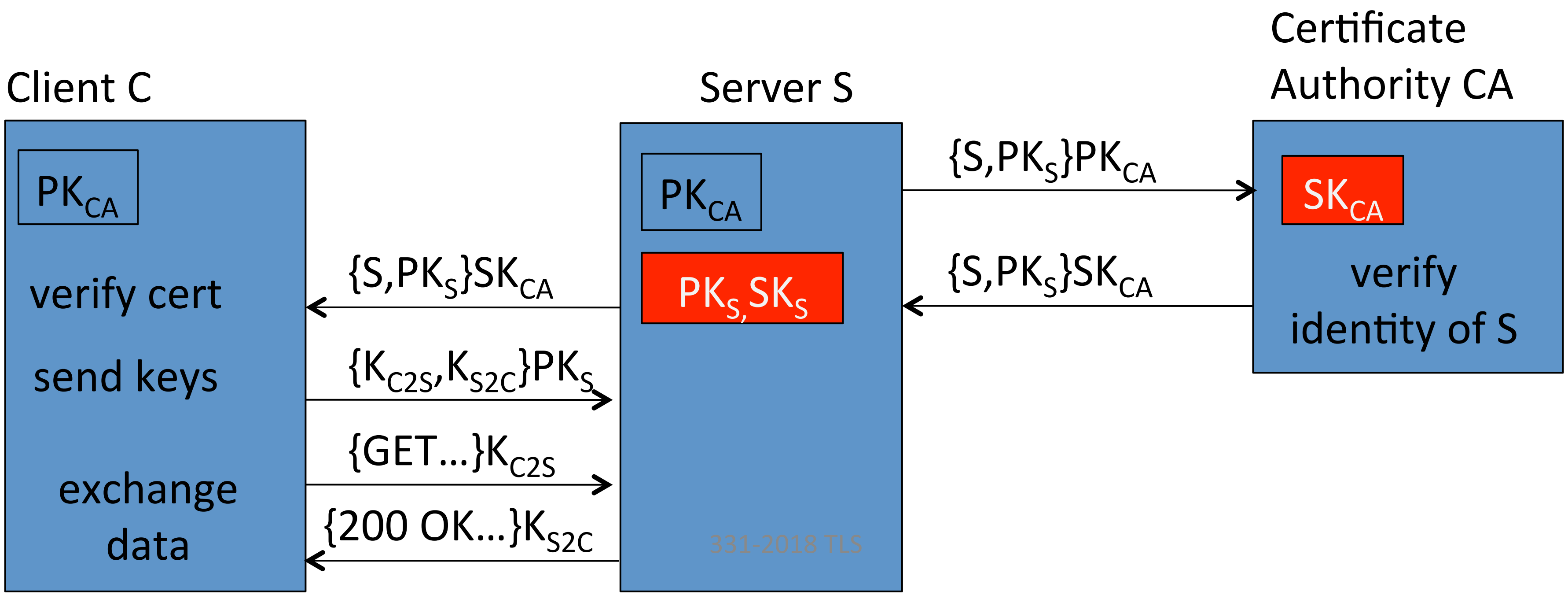
TLS in a nutshell

Imperial College London

- Transport Layer Security (TLS)
  - A cryptographic protocol to protect network connections
  - Provides both confidentiality and integrity
    - Eavesdropper sees unreadable ciphertexts
    - MITM injection leads to integrity checks failure
  - TLS is an improvement on Secure Socket Layer (SSL)
    - TLS 1.2 is considered secure, TLS 1.0 is deprecated
  - Based on notions of clients, servers and certificates
- Server needs an X.509 certificate stating its identity and public key
  - Certificates have limited validity in time
  - Certificates may identify
    - An explicit domain name: `imperial.ac.uk`
    - A set of hostnames: `*.ic.ac.uk` (matches `doc.ic.ac.uk`, not `cate.doc.ic.ac.uk`)
  - Most certificates are signed by a Certificate authority (CA) trusted by the client
  - Self-signed certificates are sometimes used but provide limited trust
    - Certificate parameters (including domain name) need to be verified offline
  - Structural weakness: TLS protection is effective only if certificates are trusted
    - Stolen certificates (Sony Hack, 2014)
    - Compromised CAs can sign spoofed certificates (DigiNotar, 2011)

TLS: the main idea

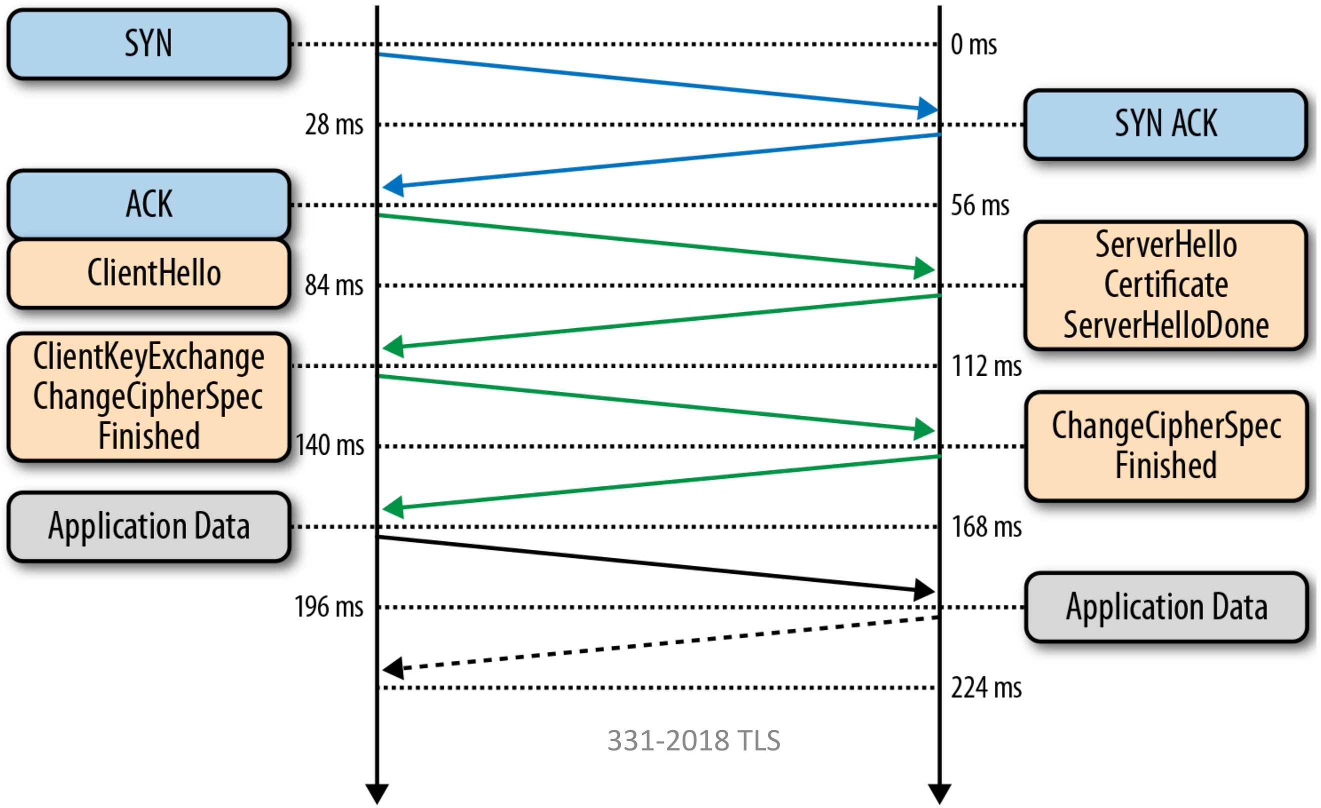
- Asymmetric encryption
  - KeyPair($PK_{A,}SK_A$) = public (or *verification*) and secret (or *signing*) key of principal A
    - SK_A is kept secret by A
    - PK_A can be revealed (for example, in a X.509 certificate)
  - ADecrypt(K_1,AEncrypt(K_2,msg))=msg if and only if KeyPair(K_1,K_2) or KeyPair(K_2,K_1)
  - {"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 encrypt)
  - {"msg"}K_L denotes symmetric (= fast) **encryption** using symmetric K labelled L
- TLS conceptual diagram:



TLS handshake

Imperial College London

- TLS should be sent over a "reliable medium"
  - Normally, that is TCP/IP
  - Hence payload data is protected but IP and port are not
- Client and server need to agree on what ciphersuite to use
  - Hash: MD5 vs SHA256; Encryption: AES vs DES; Key exchange: DH vs RSA; …
- Some choices are insecure, but servers now tend to insist on secure options



TLS security issues

- Recent vulnerabilities
  - 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
  - Not getting much better…
    - **SLOTH** CVE-2015-7575, **Sweet32** CVE-2016-2183, CVE-2016-6329, **DROWN** CVE-2016-0800, **ROBOT** (8 CVEs in 2017)
- TLS 1.3
  - Addresses all recent vulnerabilities
  - Disallows: weak crypto suites (RC4, MD5, SHA-1,…), CBC mode, TLS-level compression
  - Highly efficient: 1 less roundtrip on handshake, 0-RTT resumed connections
  - Already supported by Cloudflare, Chrome 63 and others