University of BRISTOL

Computer System- B Security

Introduction to Internet Security

SSL/TLS

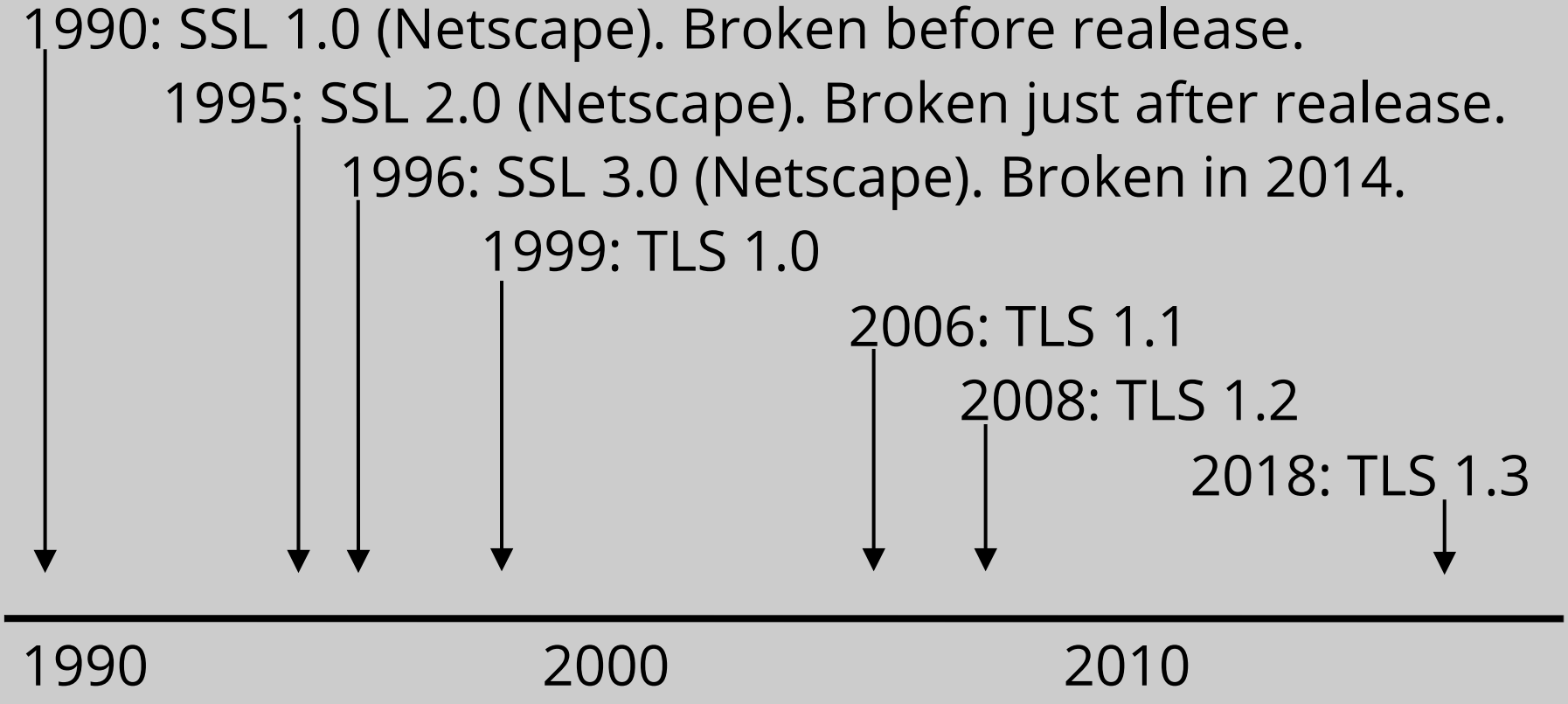
Sanjay Rawat

bristol.ac.uk

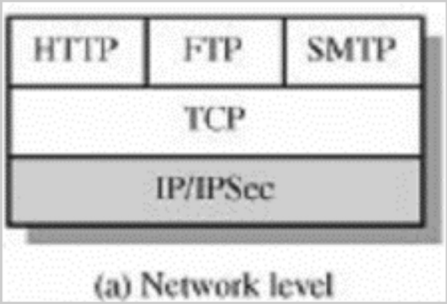
Recall… HTTPS and VPN across layers

- Transport layer
  - SSL/TSL
    - Provides encryption and authentication at application layer, which is the most common way to provide CIA security properties over the internet.

SSL / TLS timeline

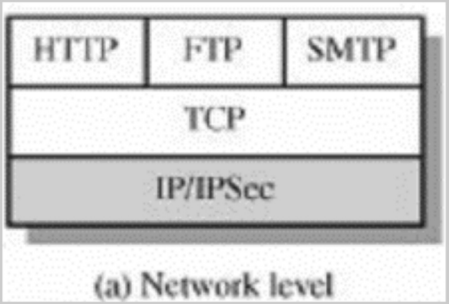


Security at layers

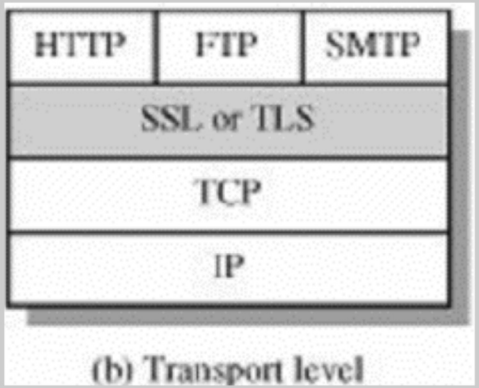


(a) Network level

Security at layers

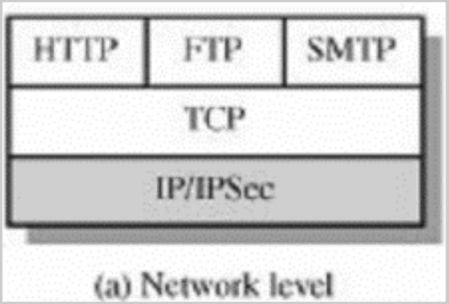


(a) Network level

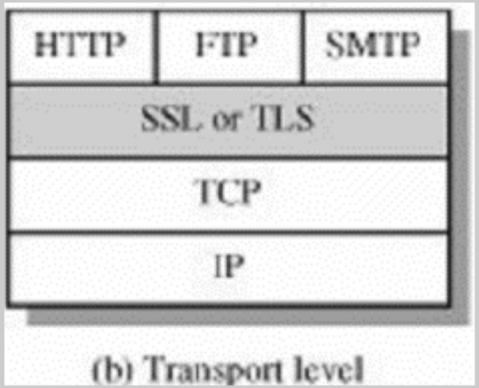


(b) Transport level

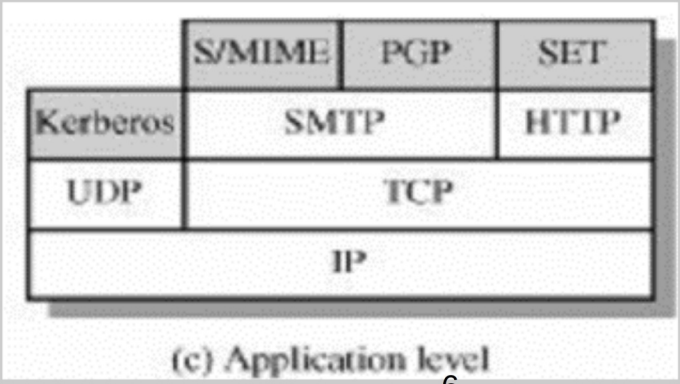
Security at layers



(a) Network level



(b) Transport level

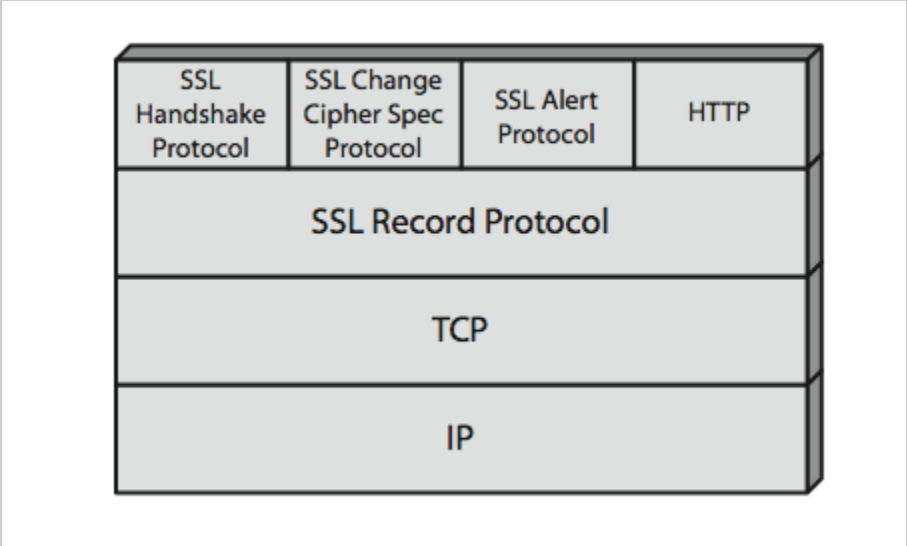


(c) Application level

SSL/TLS

- transport layer security service
- originally developed by Netscape
- uses TCP to provide a reliable end-to-end service
- Libraries (implementation)
  - OpenSSL, BoringSSL, LibreSSL, GnuTLS,...
- Has two layers of protocols
  - L1: SSL Record Protocol
  - L2: Handshake, change cipher, alert

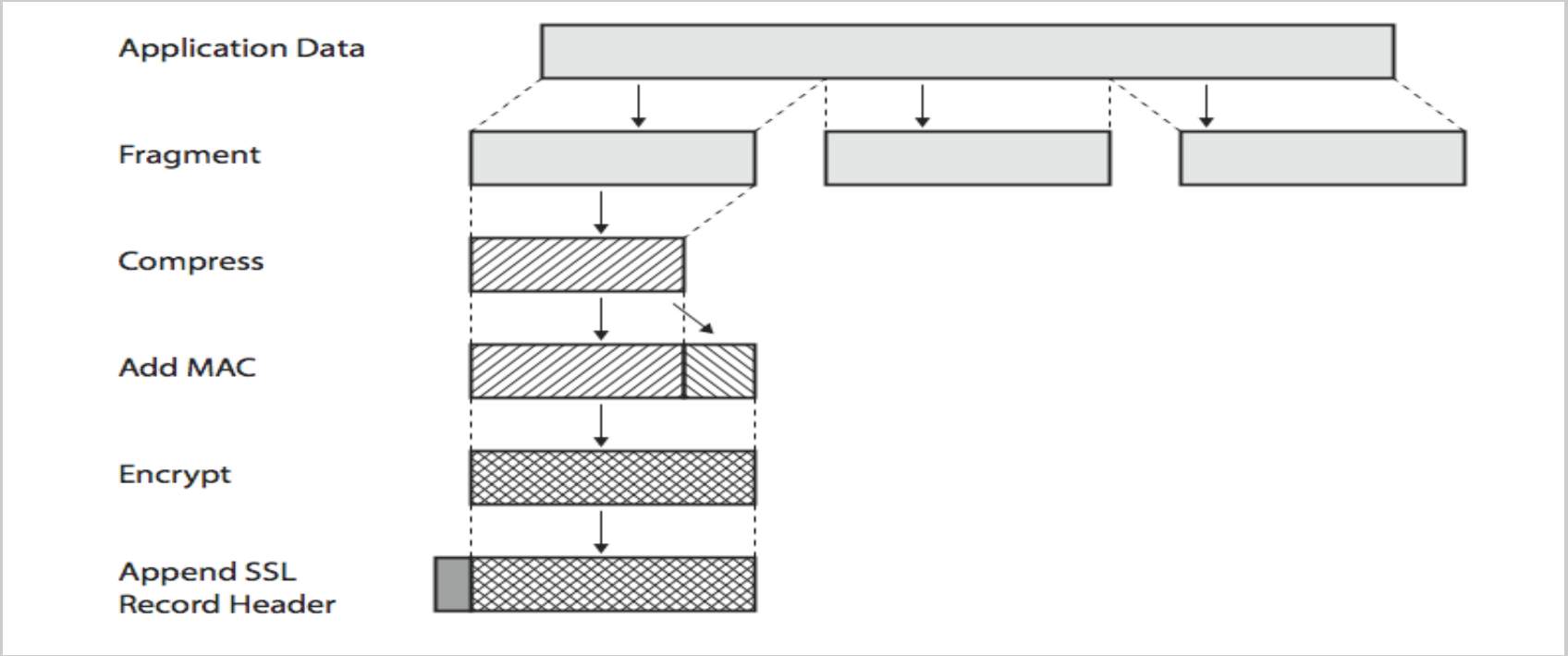
SSL/TLS Architecture



SSL Record Protocol Services

- **message integrity**
  - using a MAC with shared secret key
- **confidentiality**
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - message is compressed (optionally) before encryption

SSL Record Protocol Operation

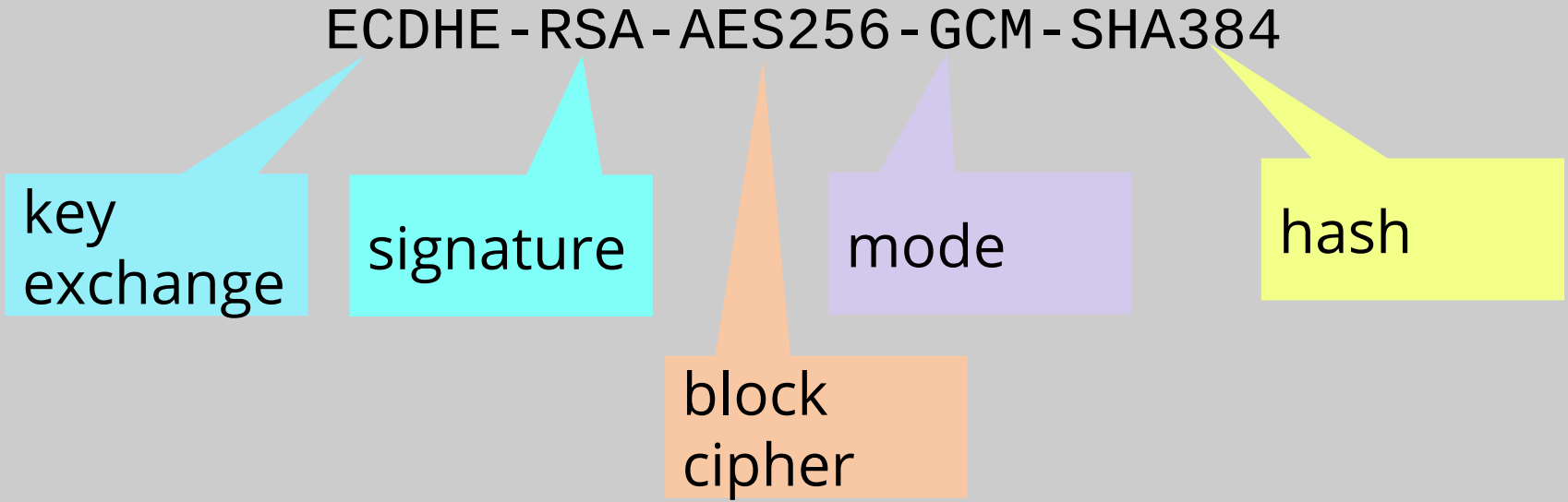


Alert Protocol

- conveys TLS-related alerts to peer entity
- Consists of 2 bytes
- Severity (1st byte)
    - warning or fatal
- specific alert (2nd byte)
    - fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
    - warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- compressed & encrypted like all TLS data

TLS cipher suites

TLS 1.2 using OpenSSL 1.0.1f: 80 cipher suites, e.g.



TLS cipher suites

Some of the main options:

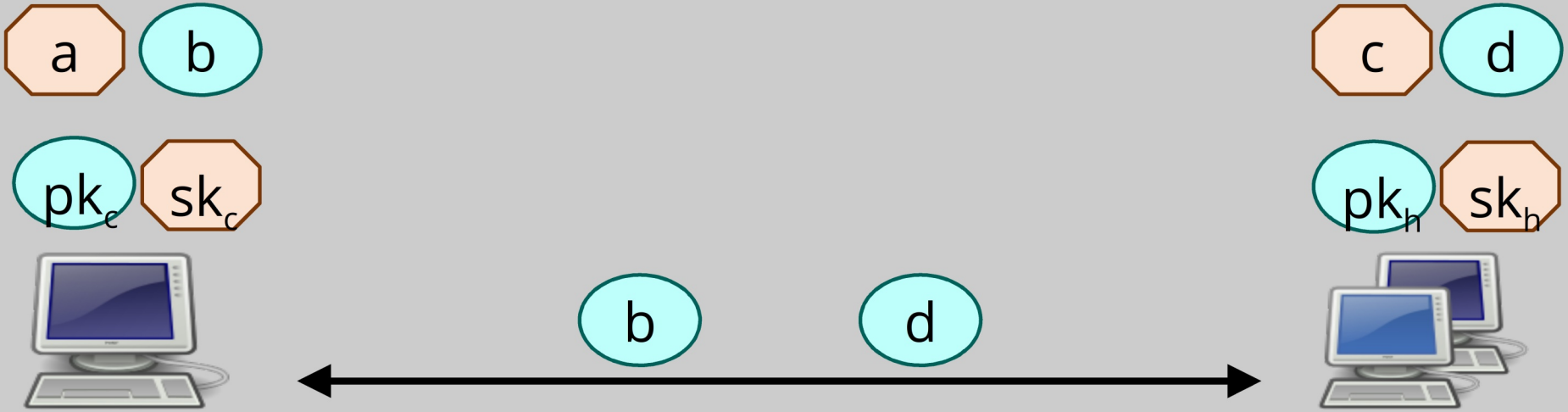
key ex	sig	cipher	hash
DH	RSA	AES256-GCM	SHA
DHE	DSS	AES256-CBC	SHA384
ECDH	ECDSA	CAMELLIA	SHA256
ECDHE		AES128-GCM	
		AES128-CBC	

If using PKC, why key Exchange?

- **Forward secrecy**
  - An interactive session has forward secrecy if compromise of the parties, after the session has ended (i.e. in future), does not reveal session's contents.
  - The usual way to achieve this is ephemeral key exchange, i.e. do a new key exchange for each session.

Ephemeral key exchange

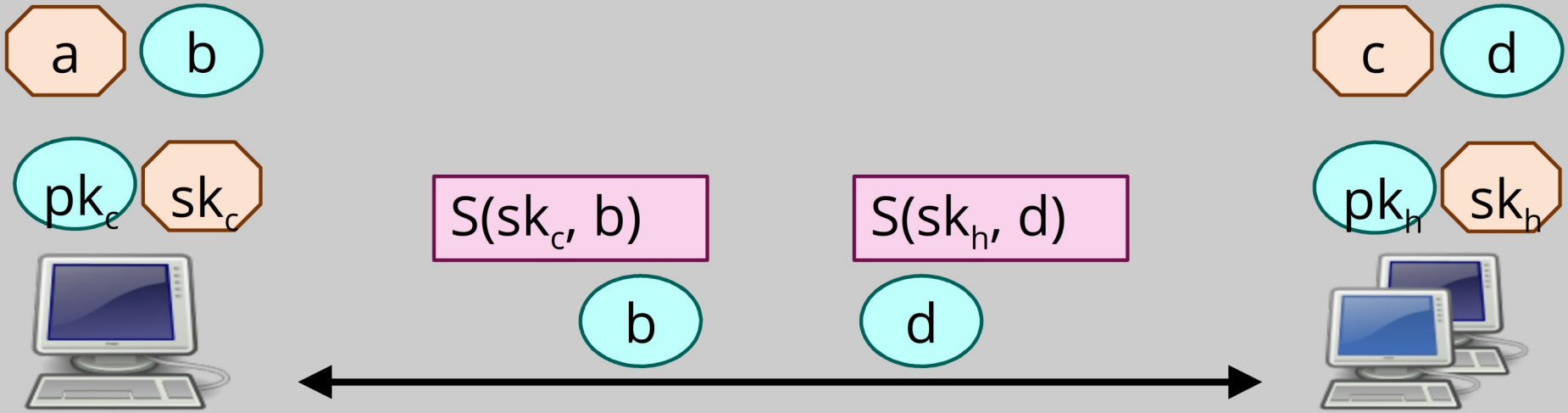
- Solution: create an *ephemeral* (=temporary) key pair for each session, do the key exchange with that. Delete the session keys as soon as the session ends.



Ephemeral key exchange

Why do we have long-term keys at all then?

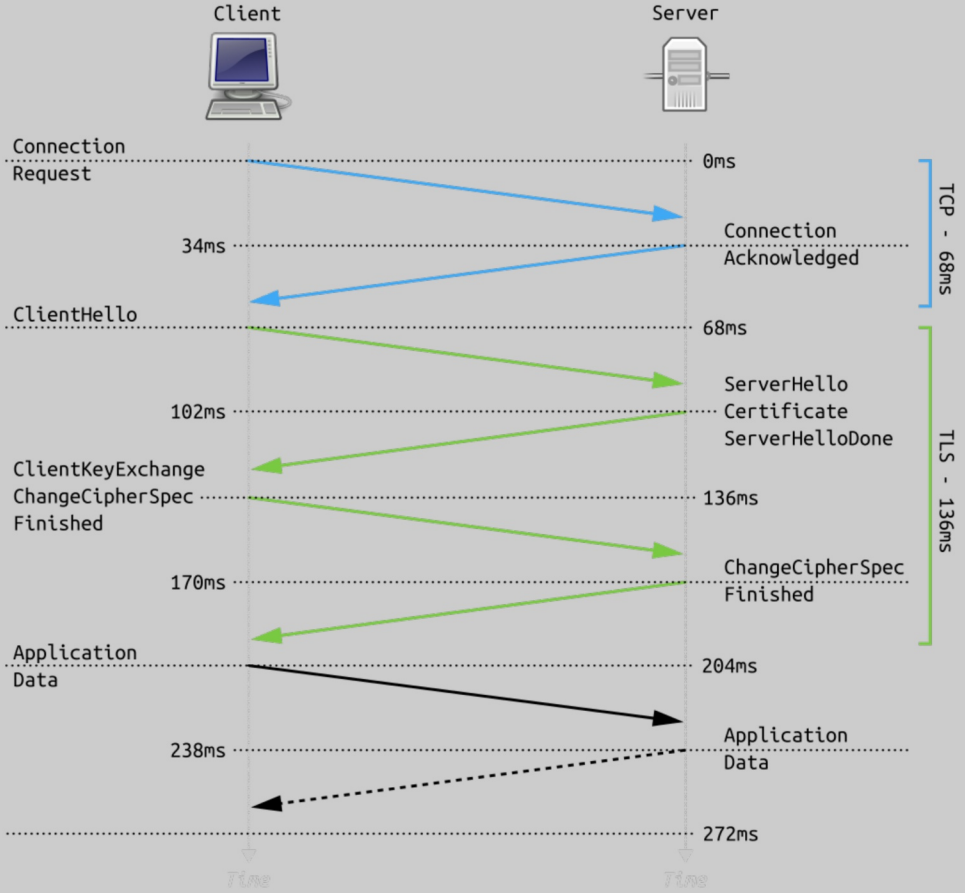
They are for authentication: parties *sign* the key exchange with their long-term keys.



SSL Handshake Protocol

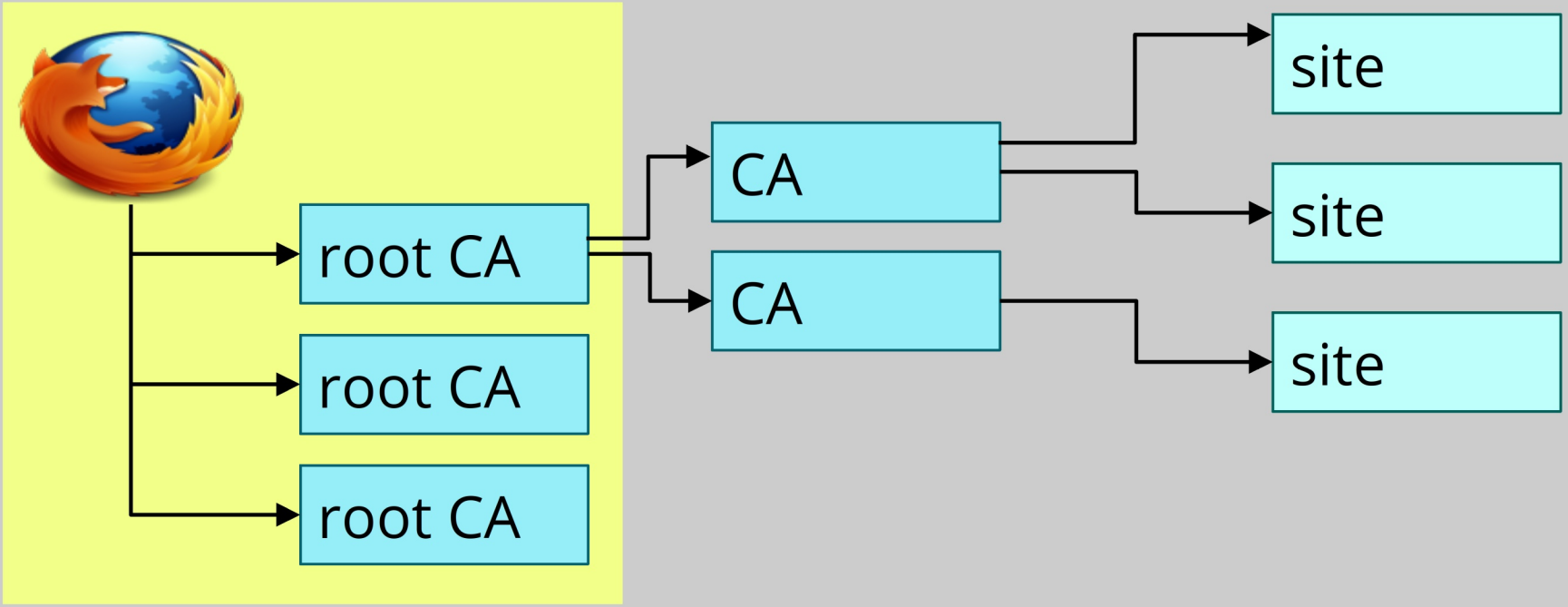
- allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
  1. Establish Security Capabilities
  2. Server Authentication and Key Exchange
  3. Client Authentication and Key Exchange
  4. Finish

TLS Handshake Protocol



By Fleshgrinder and The People from The Tango! Desktop Project.
https://commons.wikimedia.org/w/index.php?curid=40355523

TLS certificates



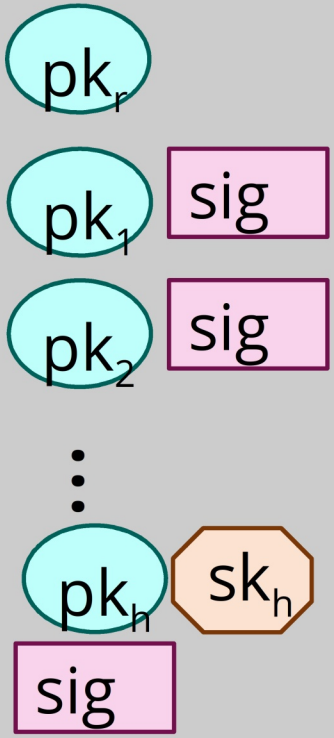
Root CA certificates are shipped with your browser.

TLS protocol layer

root of trust

intermediate CA keys

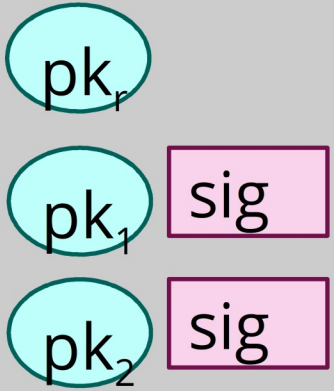
host key



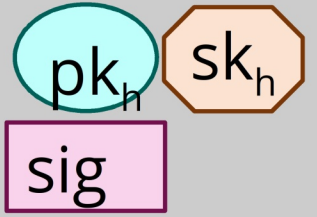
TLS protocol layer

root of trust

intermediate CA keys



host key



We can check such data from our browser!