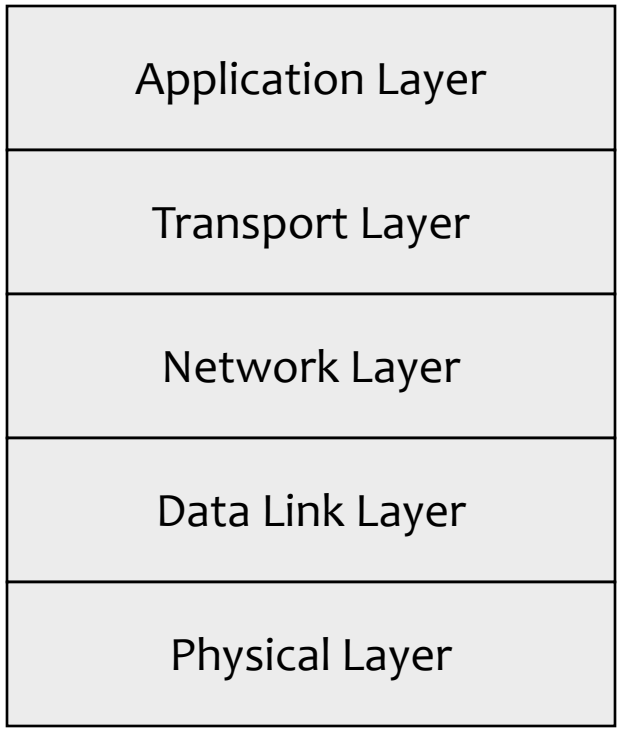
Introduction to Information Security
14-741/18-631 Fall 2020
Unit 5, Lecture 1-2 Addendum:
SSL/TLS overview

Limin Jia liminjia@andrew

SSL/TLS overview

- **Perform secure communications (e-commerce) across Internet**
  - Secure bank transactions
  - Secure online purchases
  - Secure web login (e.g., canvas)
- **Security requirements**
  - Secrecy to prevent eavesdroppers to learn sensitive information
  - Entity and message authentication to prevent message alteration / injection

Position of security in the network stack



SSH, PGP, …

SSL, TLS

IPsec

SSL/TLS history

- **SSL: Secure Sockets Layer protocol**
- **SSL v1: Designed by Netscape, never deployed**
- **SSL v2: Deployed in Netscape Navigator 1.1 in 1995**
- **SSL v3: Substantial overhaul, fixing security flaws, publicly reviewed**
- **TLS: Transport Layer Security protocol**
- **TLS v1: IETF standard improving on v3**
- **TLS 1.2 (2008)**
  - MD5/SHA1 replaced with SHA-256
- **TLS 1.3 (March 2018)**
  - Removed weak elliptic and MD5/SHA-224, DES, 3-DES, AES-CBC
  - Added ChaCha20 cipher, Ed25519 digital signature

SSL/TLS protocol

- C → S: client_hello
- S → C: server_hello

Phase 1

optional message

- S → C: certificate
- S → C: server_key_exchange
- S → C: certificate_request
- S → C: server_hello_done

Phase 2

- C → S: certificate
- C → S: client_key_exchange
- C → S: certificate_verify

Phase 3

- C → S: change_cipher_spec
- C → S: finished
- S → C: change_cipher_spec
- S → C: finished

Phase 4

SSL/TLS phase 1

- **Phase 1: Establish security capabilities**
- **{Client,Server}_hello_message**
  - Highest supported version
  - Random = 32 bit timestamp || 28 bytes random
  - Session id
  - client_hello
    - Supported cipher suite
    - Ciphers are listed in decreasing order of preference
  - server_hello
    - chosen cipher

Cipher suite

- **Cipher suite = key exchange, cipher spec**
- **Key exchange methods**
  - RSA, encrypt key with receiver's public key
  - Fixed Diffie-Hellman, public key certificate contains public DH key
  - Ephemeral Diffie-Hellman, public key is used to sign temporary DH key
  - Anonymous Diffie-Hellman, DH without authentication
- **Cipher spec**
  - Cipher Algorithm (RC4, RC2, DES, 3DES, DES40, IDEA, AES)
  - MAC Algorithm (MD5, SHA-1, SHA-256)
  - Cipher Type (stream or block)
  - Is Exportable (true or false)
  - Hash size (0 or 16 for MD5, 20 for SHA-1)
  - …

SSL/TLS phase 2

- **After Phase 1, both parties know which key exchange mechanism and which cipher to use**
- **Phase 2: Server authentication and key exchange**
- **S → C: certificate**
- **S → C: server_key_exchange**
  - Anonymous DH, Ephemeral DH, …
- **S → C: certificate_request**
  - Cert_type (RSA or DSS for key exchange)
  - List of acceptable certificate authorities
- **S → C: server_hello_done**

SSL/TLS phase 3

- **After phase 2, client has all required values to generate the session key**
- **Phase 3: Client authentication and key exchange**
- **C → S: certificate**
  - If server requested a certificate
- **C → S: client_key_exchange**
  - RSA: client generates 48 byte pre-master secret, encrypts it with server's public key
  - Eph or anon DH: client public DH value
  - …
- **C → S: certificate_verify**
  - Only used if client sent signing certificate
  - CertificateVerify.signature.SHA-256_hash =
    SHA-256(master secret || pad2 || SHA-256(handshake messages || master secret || pad1))

SSL/TLS phase 4

- **After phase 3, client and server share master secret and authenticated each other**
- **Phase 4: Finish**
- **C → S: change_cipher_spec**
  - Establish set up cipher and keys
- **C → S: finished**
  - SHA-256(master_secret|| pad2 || SHA-256(handshake messages || Sender || master_secret|| pad1)) || SHA-256(master_secret|| pad2 || SHA-256(handshake messages || Sender || master_secret|| pad1))
  - Handshake messages contains all messages up to now
- **S → C: change_cipher_spec**
- **S → C: finished**

Cryptographic computations

- **Master secret (MS) creation from pre-master secret (PS), Client random (CR), Server random (SR)**
  - MS = SHA-256(PS || SHA-1(`A' || PS || CR || SR)) ||
    SHA-256(PS || SHA-1(`BB' || PS || CR || SR)) ||
    SHA-256(PS || SHA-1(`CCC' || PS || CR || SR))

- **ChangeCipherSpec requires client & server MAC key, client & server encryption key, client & server IV, generated from MS:**
  - SHA-256(MS || SHA-1(`A' || MS || SR || CR)) ||
    SHA-256(MS || SHA-1(`BB' || MS || SR || CR)) ||
    SHA-256(MS || SHA-1(`CCC' || MS || SR || CR)) ||
    SHA-256(MS || SHA-1(`DDDD' || MS || SR || CR)) || ...

SSL record format

- **SSL Record =**
  Content type || Major version || Minor version || Length ||
  { Data || MAC(K', Data) } $_K$

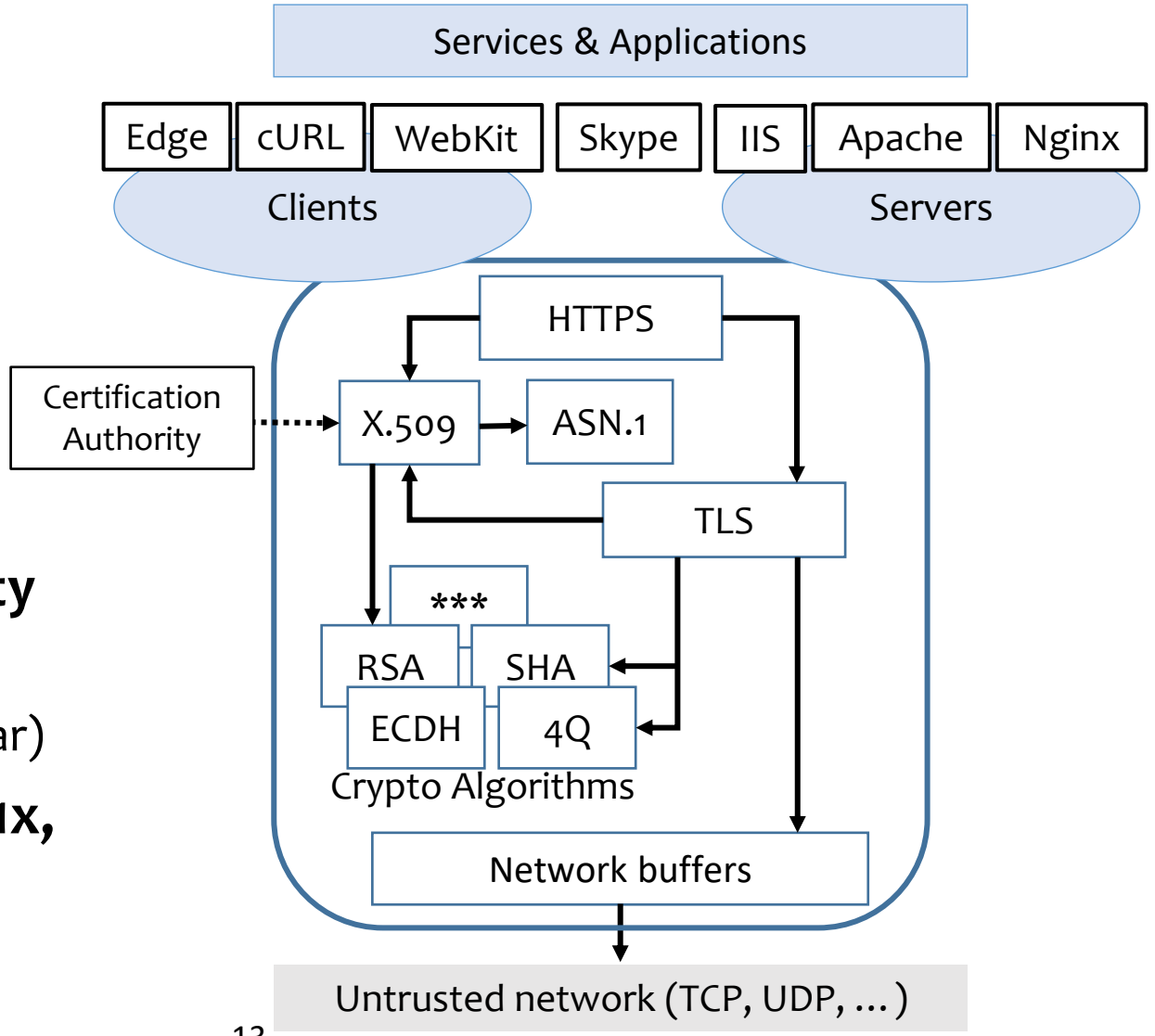
- **Different keys for each direction, encryption or MAC**

Sample TLS session

- **Client has no certificate, only server authenticated**
- **C → S: client_hello**
- **S → C: server_hello**
  - Ephemeral DH key exchange, AES encryption, SHA-384-based MAC
- **S → C: Server certificate, containing RSA public key**
  - Client checks validity + verifies URL matches certificate!
- **S → C: server_key_exchange: g, p, gs, $\{H(g, p, gs)\}_{K_S^{-1}}$**
- **S → C: server_hello_done**
- **C → S: client_key_exchange: gc**
- **C → S: change_cipher_spec**
- **C → S: finished**
- **S → C: change_cipher_spec**
- **S → C: finished**

The HTTPS Ecosystem

Slide ack: Bryan Parno



- **Most widely deployed security protocol?**
  - 40% all Internet traffic (+40%/year)
- **Web, cloud, email, VoIP, 802.1x, VPNs, …**

The HTTPS Ecosystem is buggy

Slide ack: Bryan Parno

- **20 years of attacks & fixes**
  - Buffer overflows
  - Memory management
  - Incorrect state machines
  - Lax certificate parsing
  - Weakly or badly implemented crypto
  - Side channels
  - Error-inducing APIs
  - Flawed standards
  - …
- **Many implementations**
  - SChannel, OpenSSL, NSS, …
  - ***Still patched every month!***