Transport Layer Security: TLS/SSL and Certificates (CS 642)

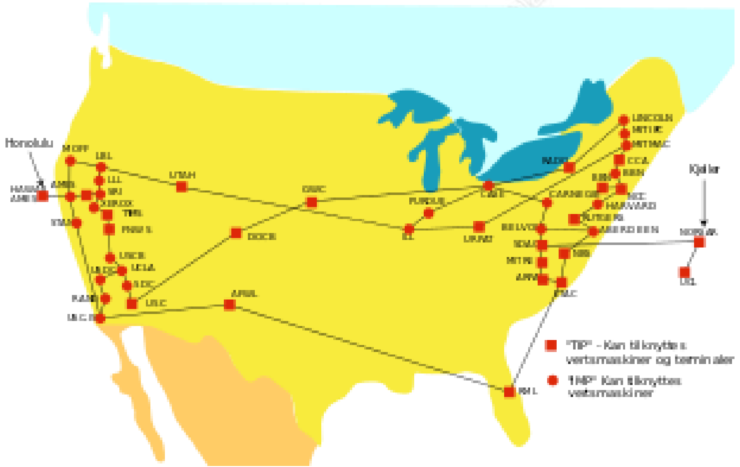
Earlence Fernandes
earlence@cs.wisc.edu

* Some slides are borrowed from Clarkson, Shmatikov, Jana

Internet: The network of computers

History

- Started as (D)ARPANET in late 1960s
- Initially there were small networks of computers
- 1972 email was invented
- 1981 IBM created Bit-Net
- 1982 First "Internet" was used to connect different isolated networks
- 1984 Domain Name System (DNS)
- 1989 100,000 computers connected, starting of the Web
- 1994 SSL, 1999 TLS

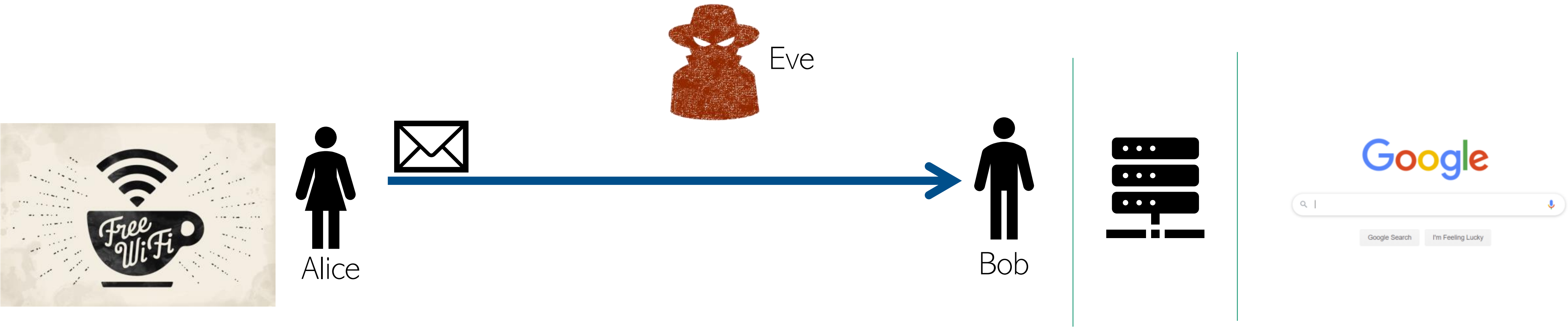


BIT-NET internet services

Source: https://www.internetsociety.org/internet/history-internet/brief-history-internet/

Trust in the untrusted Internet

The problem



- Should be able to "surf the Internet" no matter where you are
- Threat model
  - Network adversary – Attacker completely owns the network: controls Wi-Fi, DNS, routers, his own websites, can listen to any packet, modify packets in transit, inject his own packets into the network
  - Goal – Learn the communicated messages? And?

Didn't public key crypto solve it already?

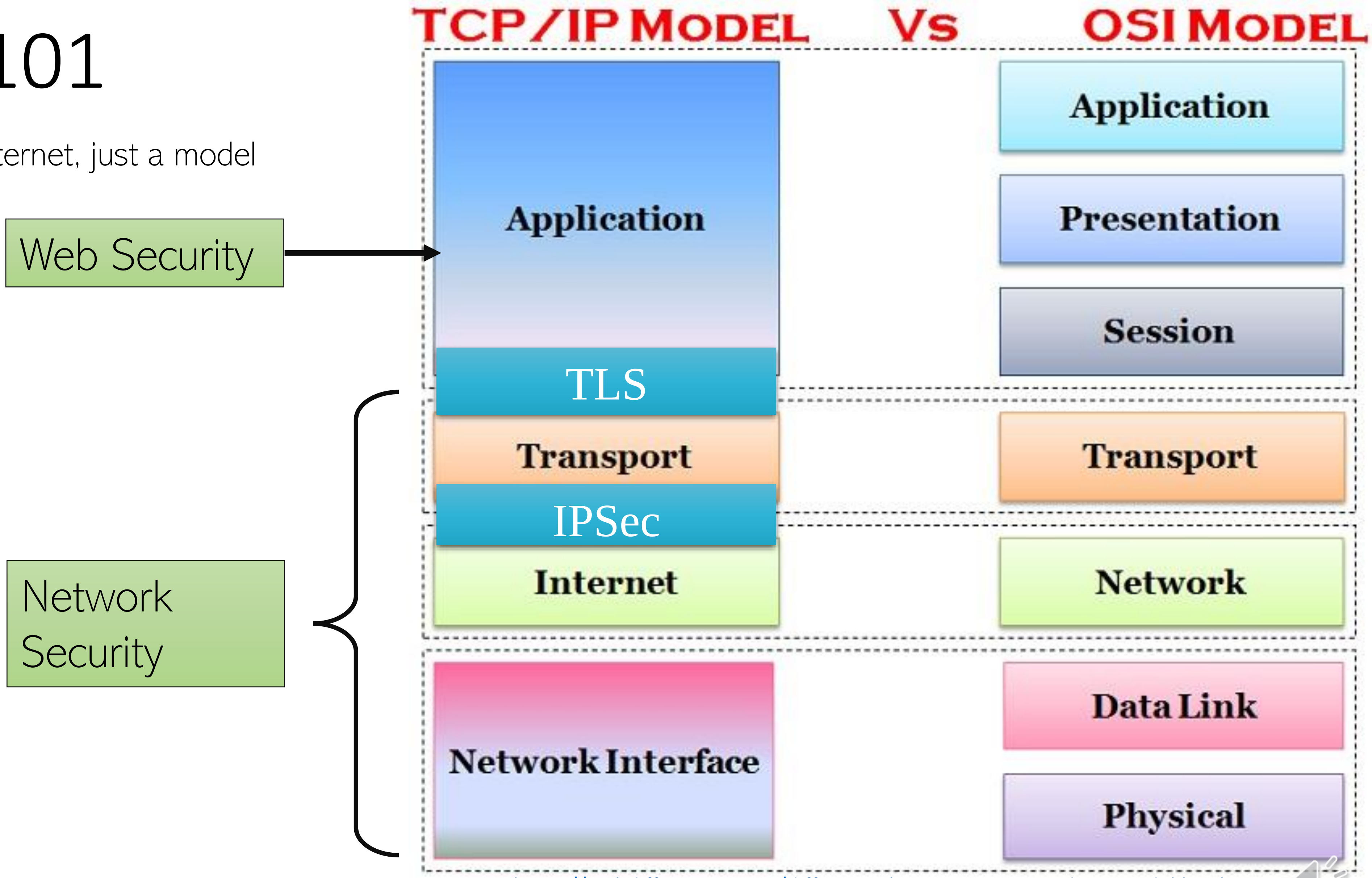
- Well NO!
  - It gives us the building blocks, but still lot to build

- How does Alice know the public key of Bob?
- How does Alice know if the key is indeed of Bob?
- How to decide what to encrypt and what not?
- How is the "secure" connection initiated? What is the protocol?

Network 101

OSI – Open Systems Internet, just a model

- Just a model



https://techdifferences.com/difference-between-tcp-ip-and-osi-model.html

Transport layer security (TLS)

- What is SSL then?
  - Secure Socket Layer
  - SSL 1.0 – internal Netscape design, early 1994(?) Lost in the mists of time
  - SSL 2.0 – Netscape, Nov 1994
    - Several weaknesses
  - SSL 3.0 – Netscape and Paul Kocher, Nov 1996
- TLS 1.0 – Internet standard, Jan 1999
  - Based on SSL 3.0, but not interoperable (uses different cryptographic algorithms)
- TLS 1.1 – Apr 2006
- TLS 1.2 – Aug 2008 (most widely used)
- TLS 1.3 – Aug 2018 (published)

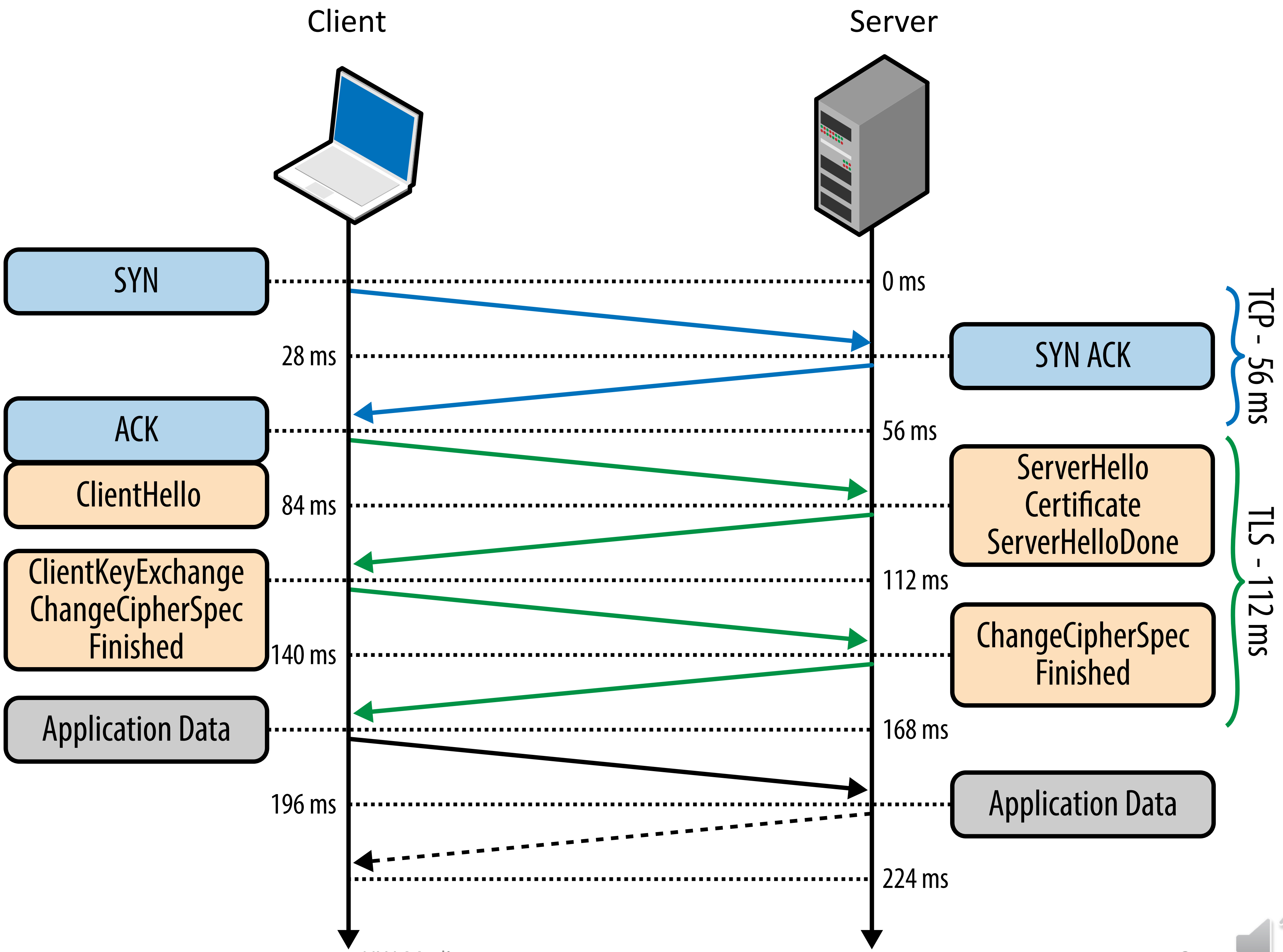
TLS 1.3

Transport layer security (TLS)

TLS consists of **two** protocols

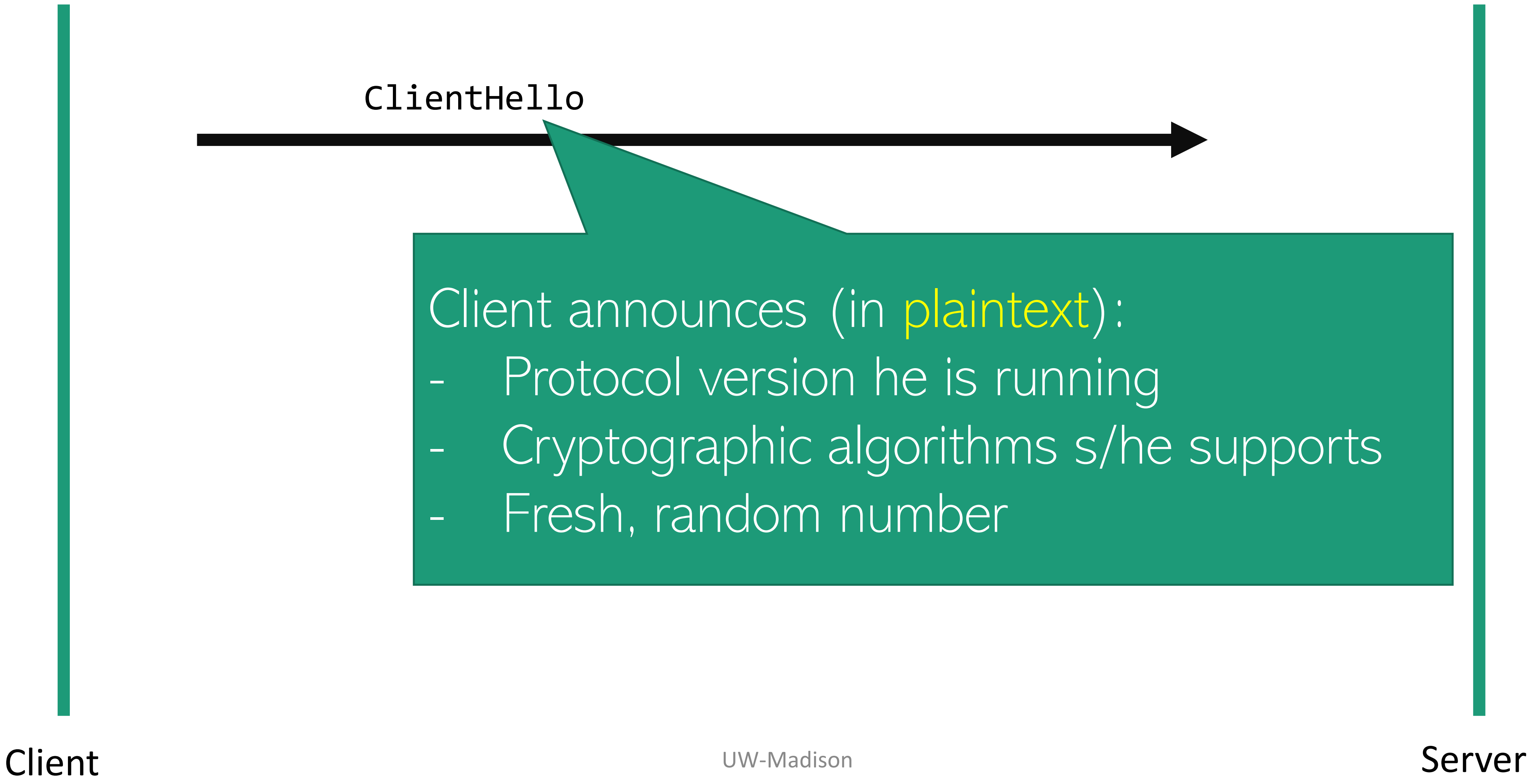
- Handshake protocol
  - Key agreement
  - Uses public-key cryptography to establish several shared secret keys between the client and the server
- Record layer protocol
  - How to encrypt
  - Uses the secret keys established in the handshake protocol to protect confidentiality, integrity, and authenticity of data exchange between the client and the server

TLS handshake



Source: https://hpbn.co/transport-layer-security-tls/

ClientHello



ClientHello (RFC 5246, TLSv1.2)

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present) {
       case false: struct {};
       case true: Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

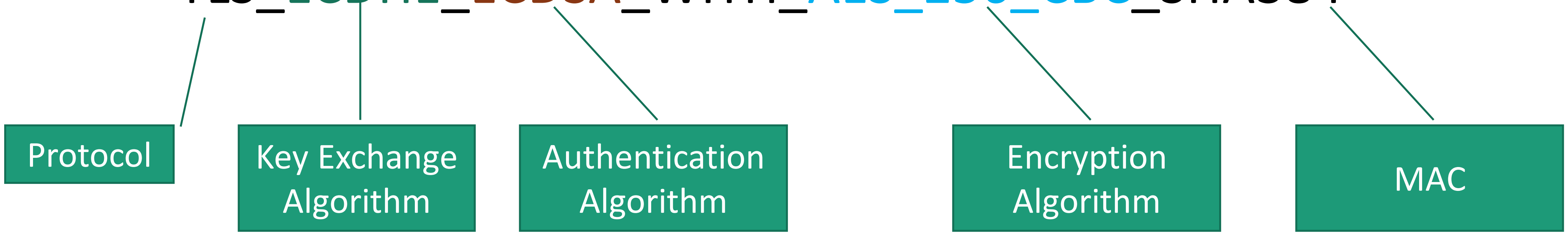
Session id (if the client wants to resume an old session)

Cipher Suites

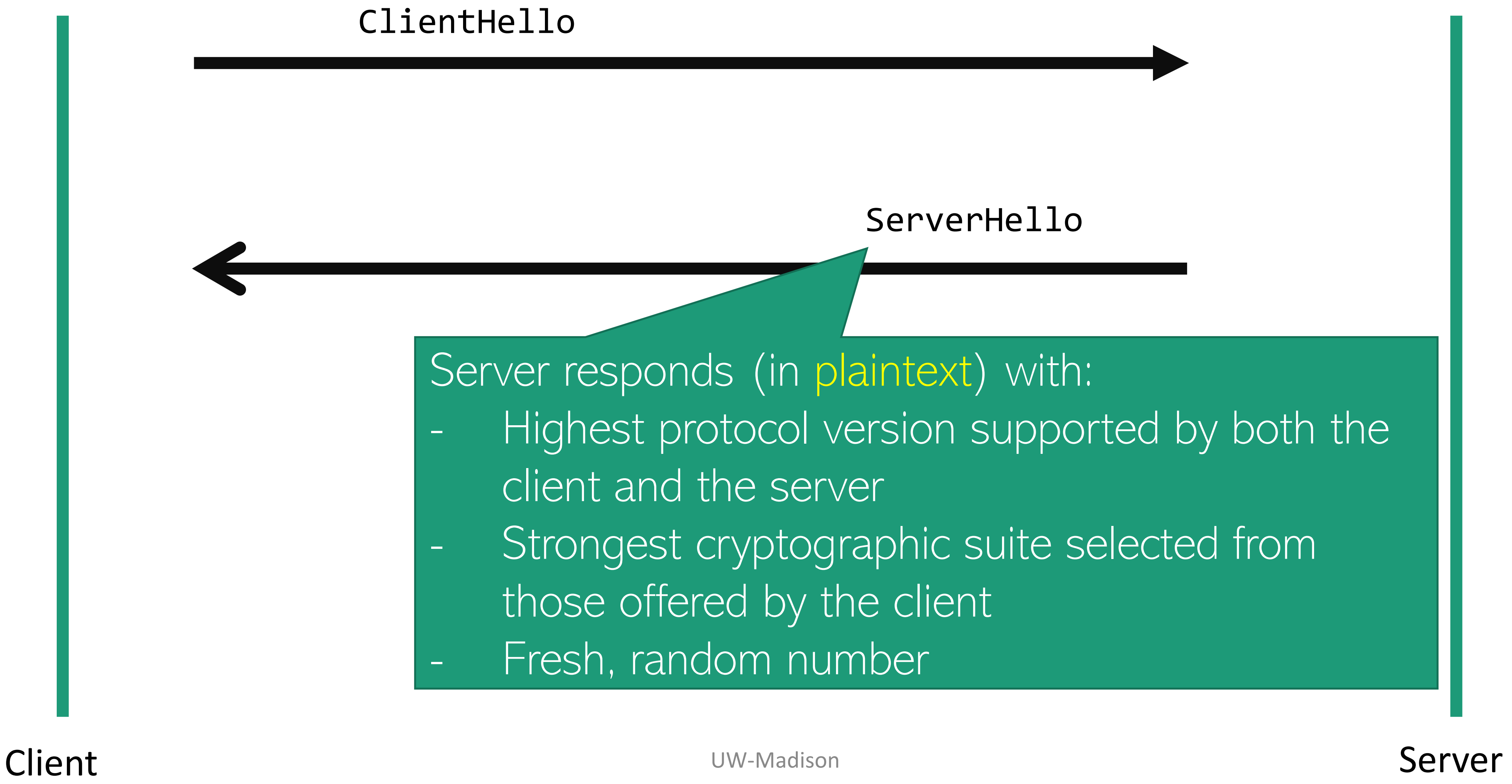
- Set of algorithms supported by the client / server

- Example:

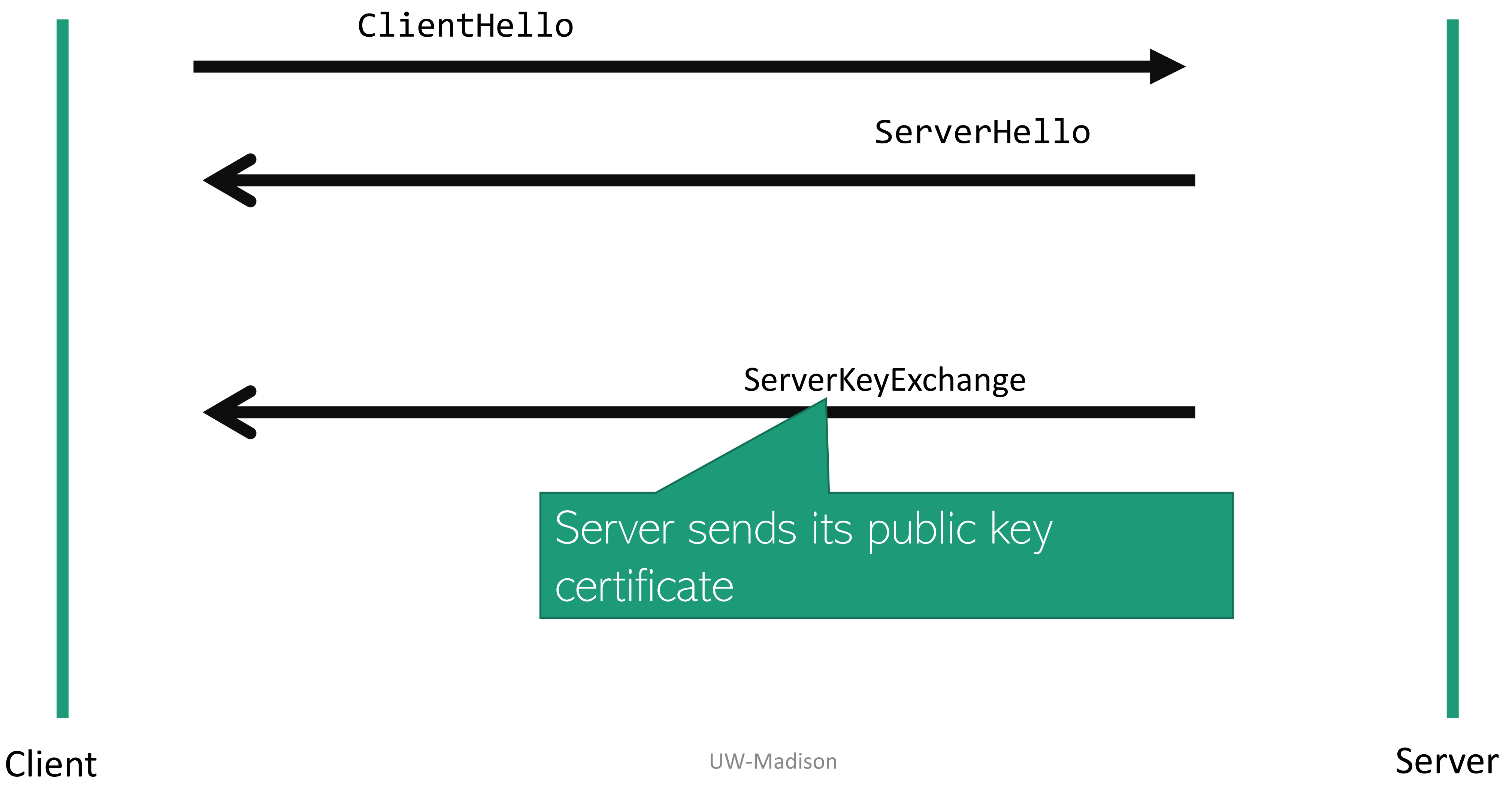
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384



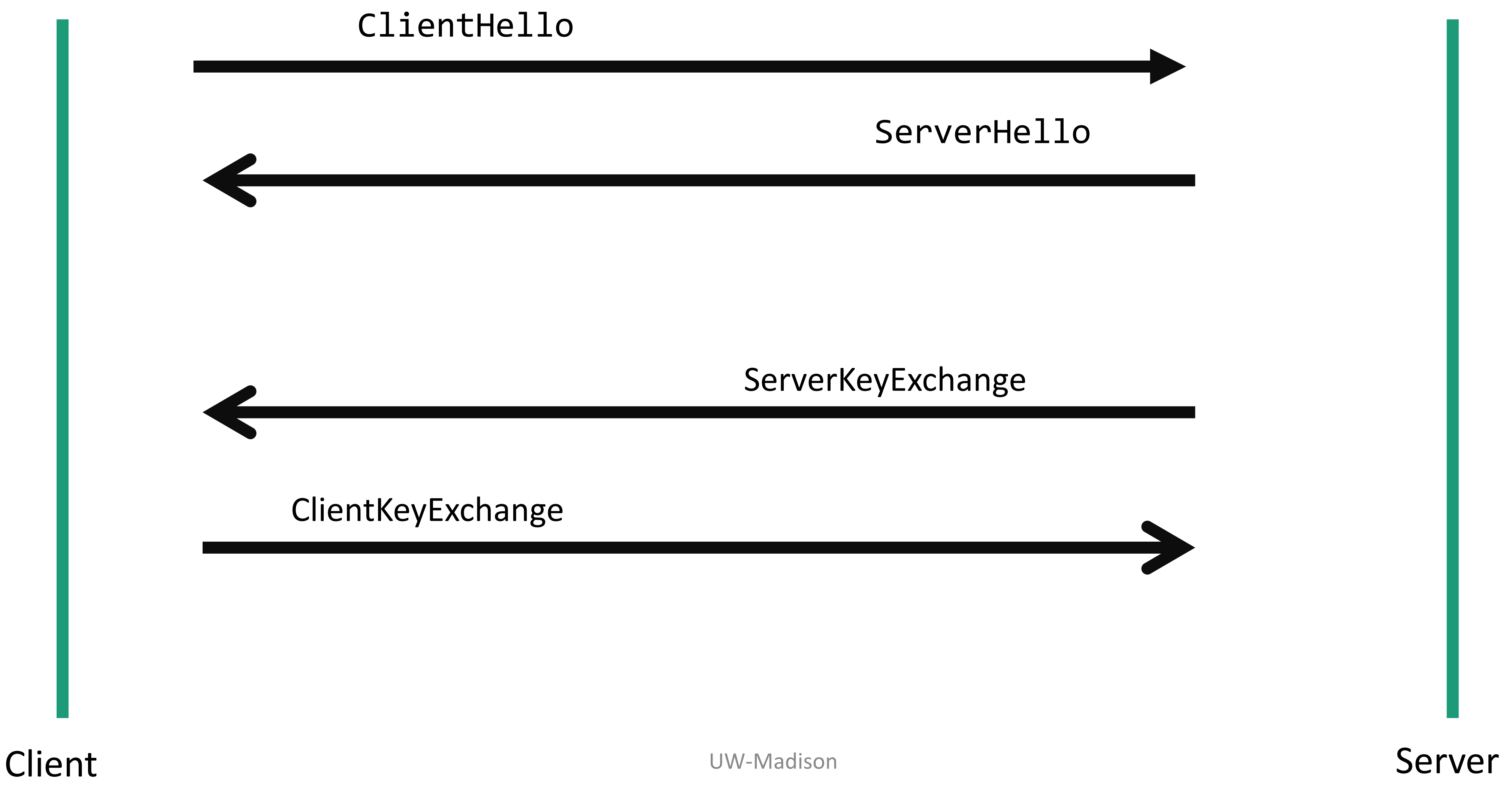
ServerHello



ServerKeyExchange



ClientKeyExchange



ClientKeyExchange (RFC)

```
struct {
    select (KeyExchangeAlgorithm) {
        case rsa: EncryptedPreMasterSecret;
        case diffie_hellman: ClientDiffieHellmanPublic;
    } exchange_keys
} ClientKeyExchange;
```

Where does randomness come from?

```
struct {
    ProtocolVersion client_version;
    opaque random[46];
} PreMasterSecret
```

Random bits from which symmetric keys will be derived (by hashing them with nonces)

Debian Linux (2006-08)

```
#ifndef PURIFY
        MD_Update(&m,buf,j); /* purify complains */
#endif
```

Without this line, the seed for the pseudo-random generator is derived only from process ID

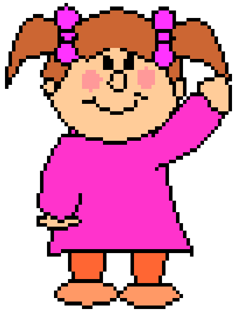
- Default maximum on Linux = 32768

Result: all keys generated using Debian-based OpenSSL package in 2006-08 are predictable

Key Agreement: Diffie-Hellman Protocol

Key agreement protocol, both A and B contribute to the key

Setup: p prime and g generator of Z_p^*, p and g public.



$g^a \bmod p$ →

← $g^b \bmod p$

Pick random, secret (a)
Compute and send $g^a \bmod p$

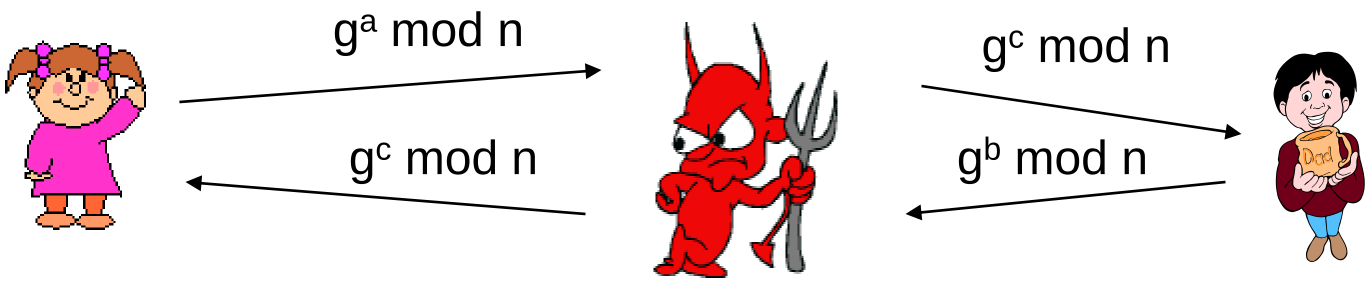
$K = (g^b \bmod p)^a = g^{ab} \bmod p$

Pick random, secret (b)
Compute and send $g^b \bmod p$

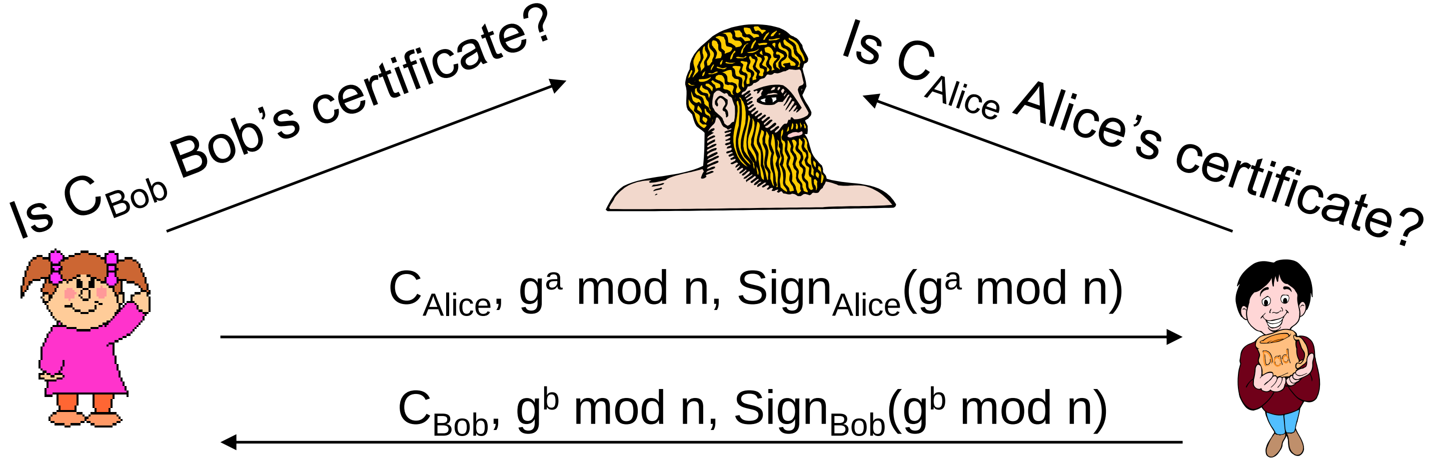
$K = (g^a \bmod p)^b = g^{ab} \bmod p$

* From Clarkson

Authenticated Diffie-Hellman

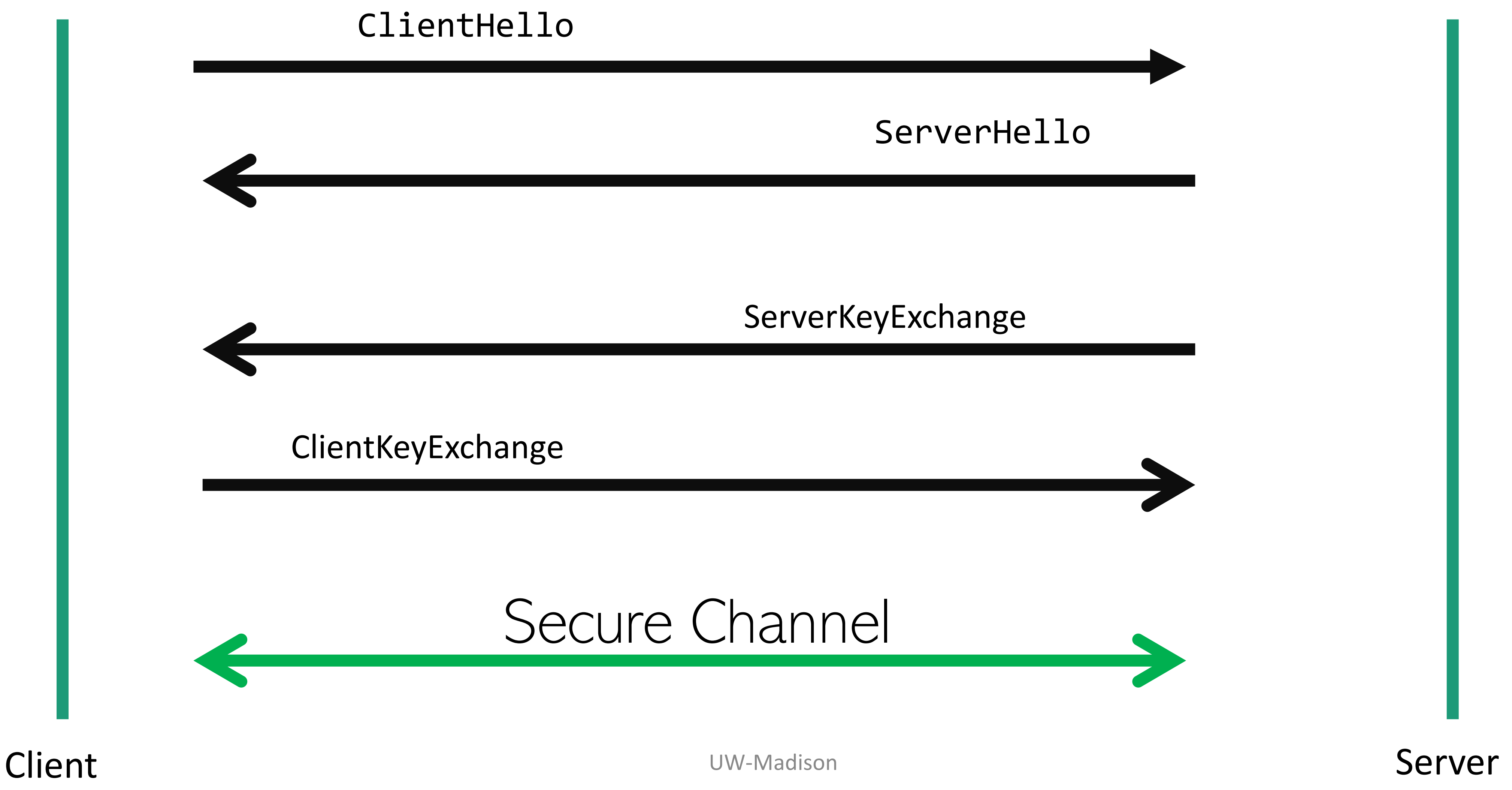


Alice computes g^{ac} mod n and Bob computes g^{bc} mod n !!!



* From Clarkson

Handshake Finished, secure channel established, or handshake aborted

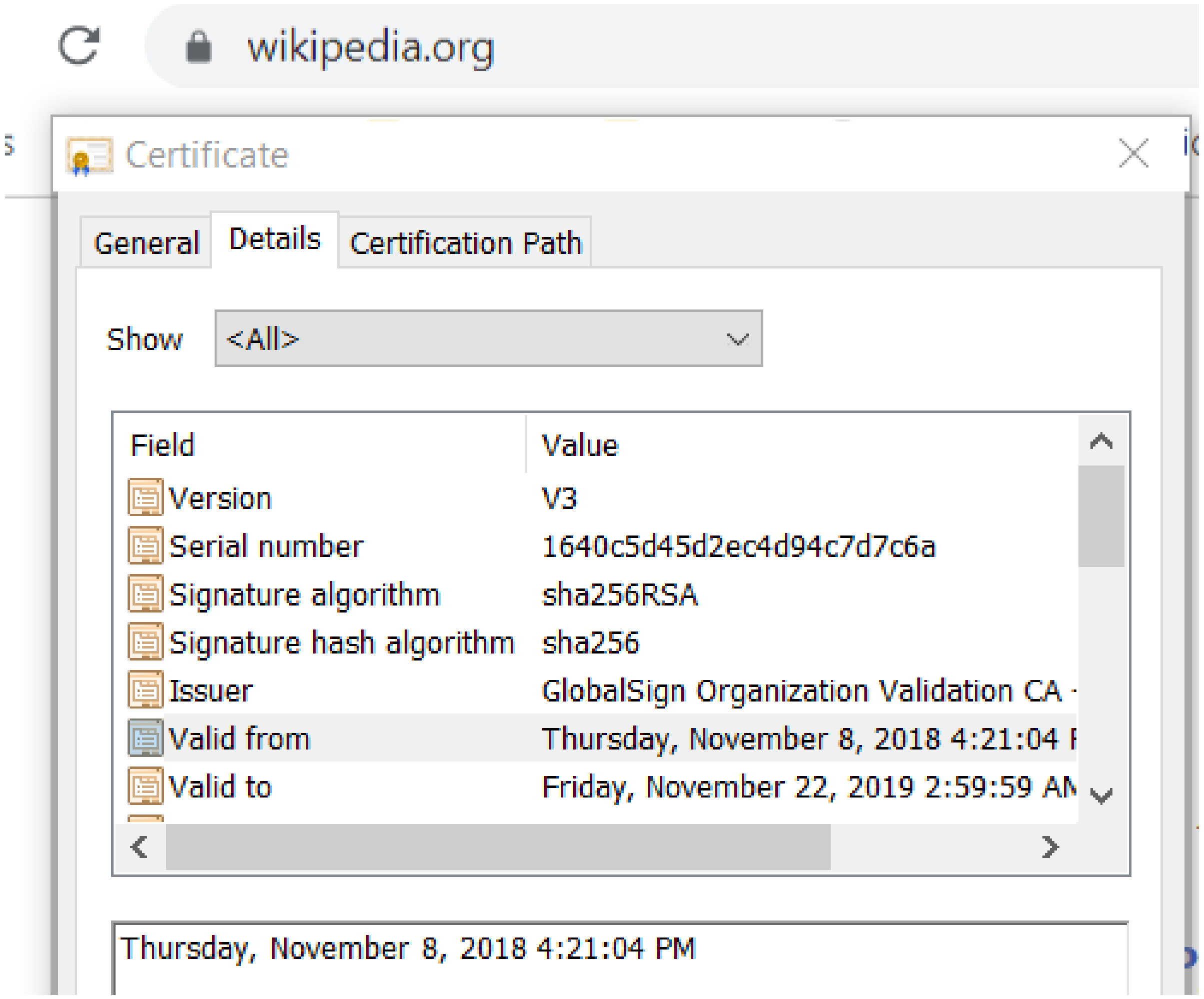


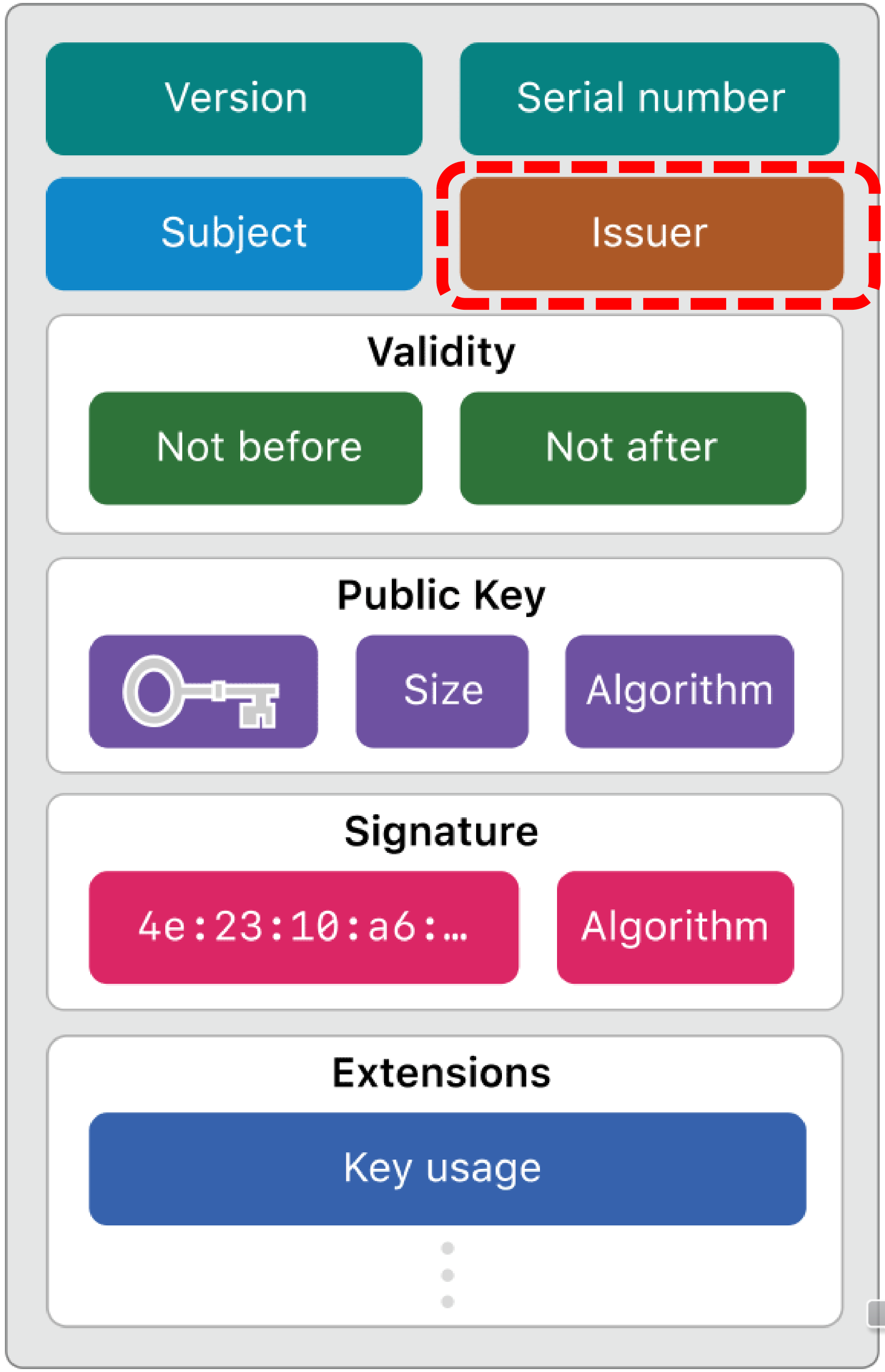
TLS

- Provide confidentiality and integrity above the transport layer
- Authenticity?
  - Certificates

Certificates

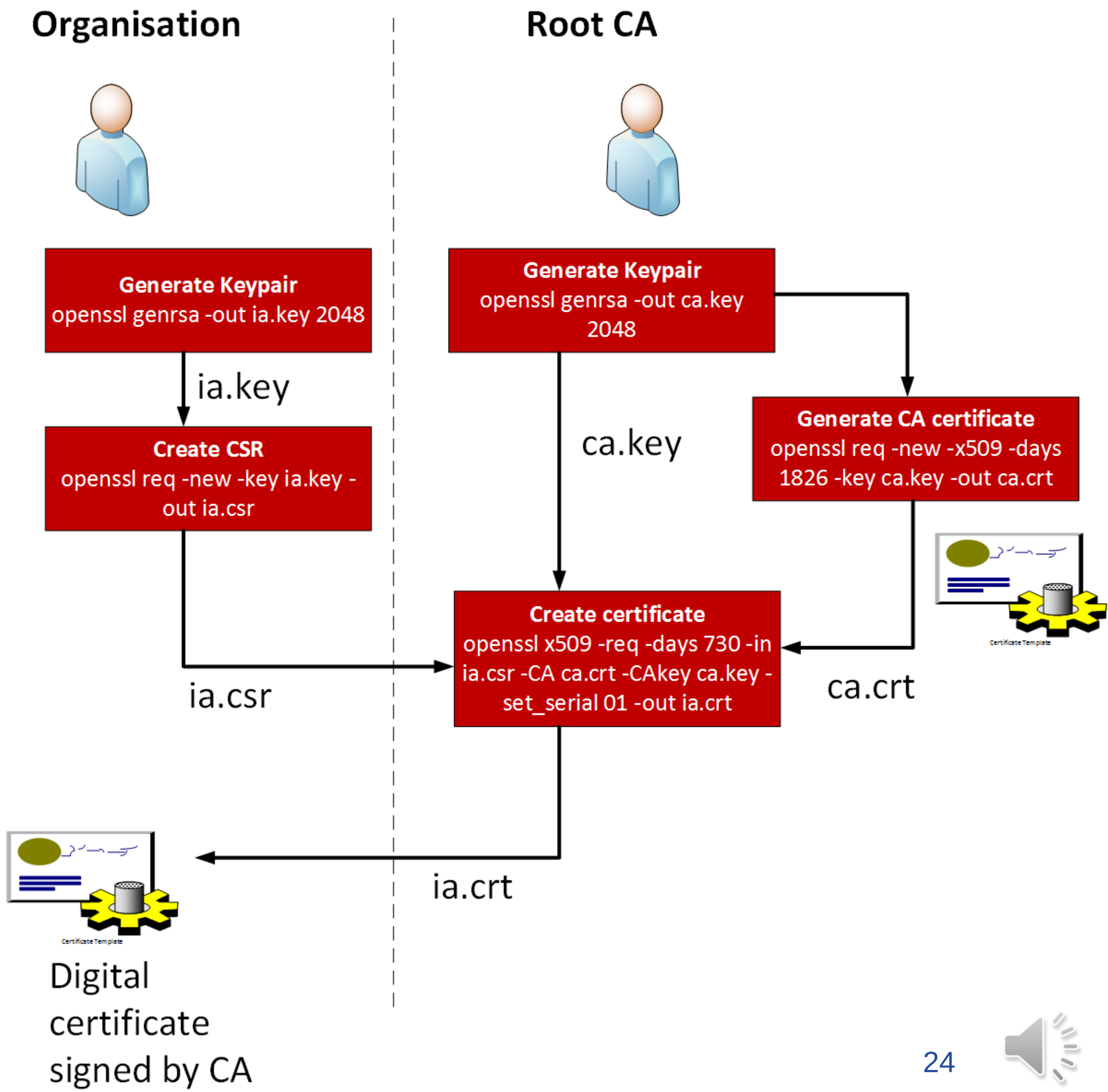
X.509 Certificate format





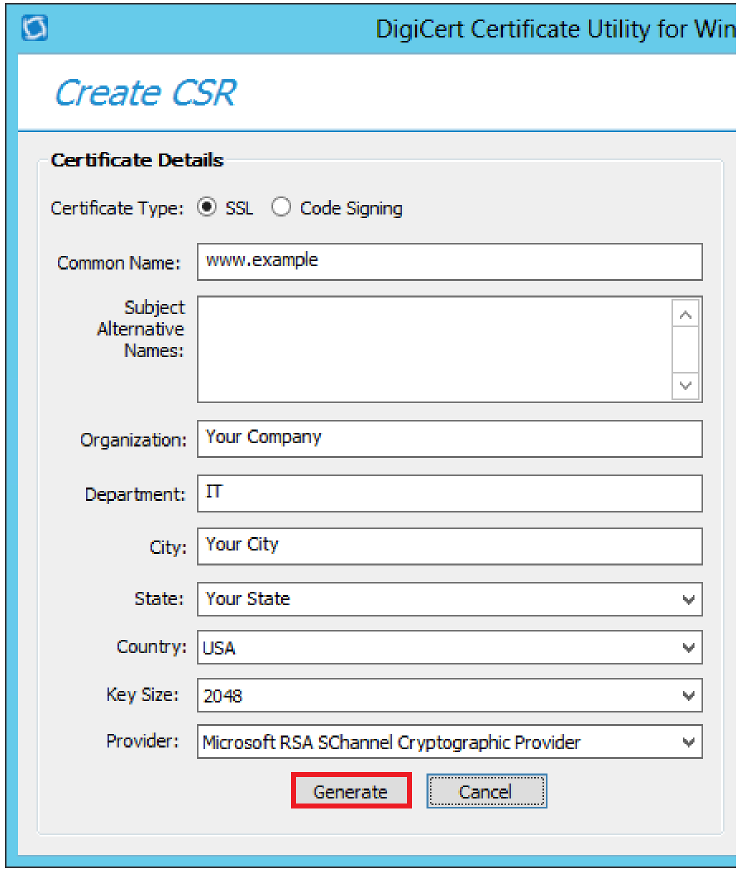
How to obtain a Certificate?

- Define your own CA (use openssl or Java Keytool)
  - Certificates unlikely to be accepted by others
- Obtain certificates from one of the vendors: VeriSign, Thawte, and **many many** others



Certificate Signing Request

```
$ openssl req -new
      -newkey rsa:2048
      -nodes -keyout server.key
      -out server.csr
```



Asks a bunch of details, including organization, city, state, country, etc. Most interesting one is
Common Name

Can be:
`www.google.com, secure.website.org, *.domain.net, etc.`

CAs and Trust

- Certificates are trusted if signature of CA verifies
- Chain of CA's can be formed, head CA is called root CA
- In order to verify the signature, the public key of the root CA should be obtained.
- TRUST is centralized (to root CA's) and hierarchical
- What bad things can happen if the root CA system is compromised?
- Who Signs CA's certificates?

Comodo certificate hack—it gets worse

The big news that didn't make the news is back again, and yeah it's gotten worse. Last week I wrote...

TECHNOLOGY

Trustico revokes 23,000 SSL certificates due to compromise

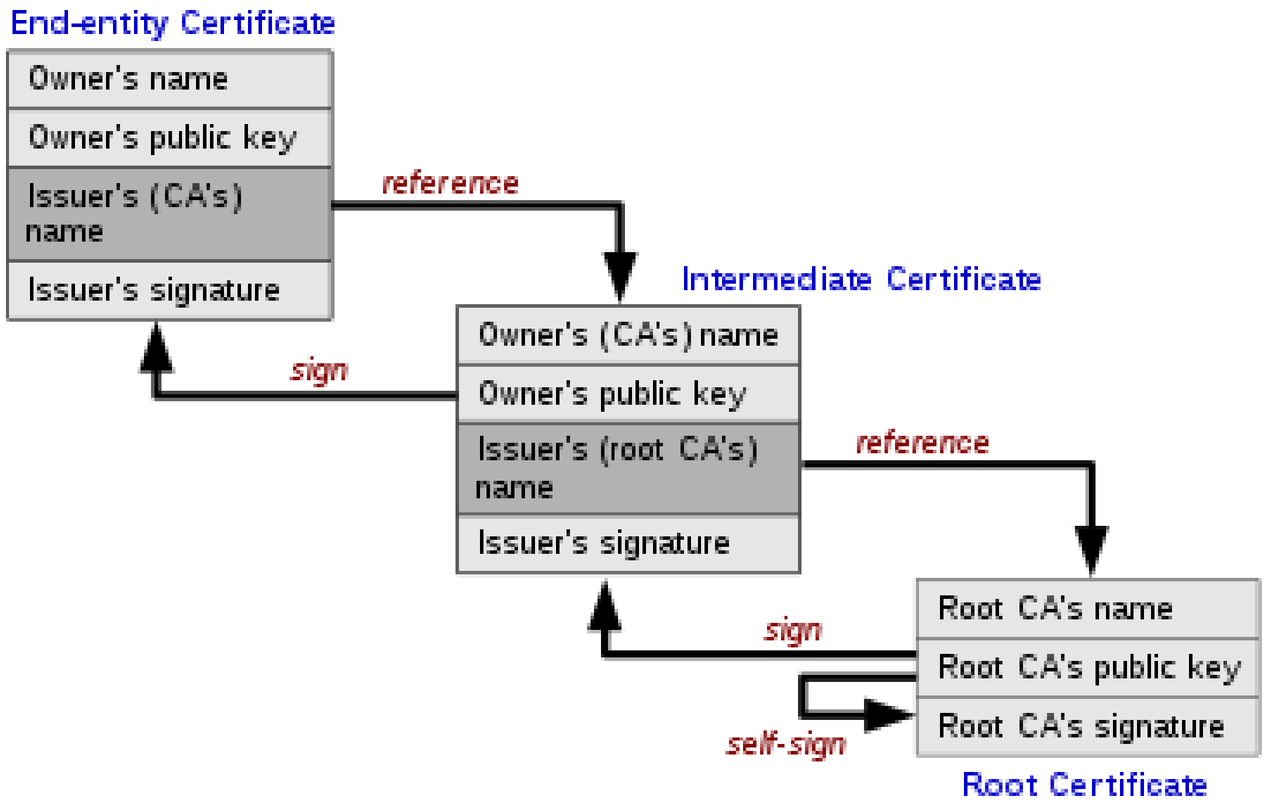
SECURITY

Comodohacker returns in DigiNotar incident

Claiming credit for the cyberattack against Dutch certificate company DigiNotar, Comodohacker is threatening to release other fake certificates.

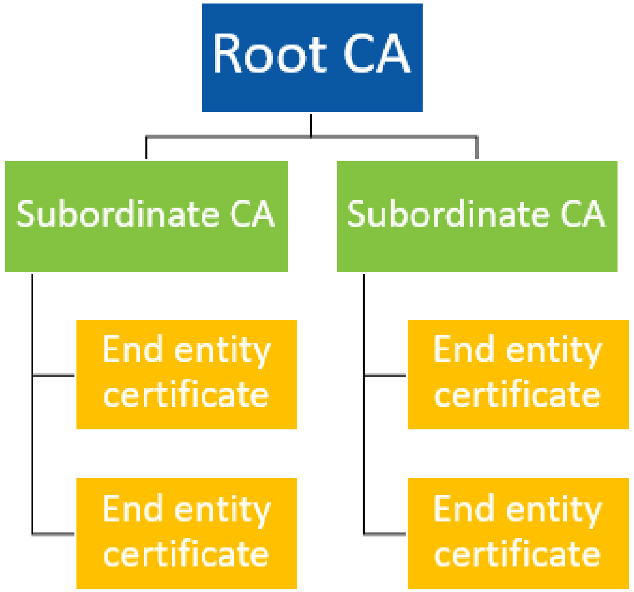
BY LANCE WHITNEY | SEPTEMBER 6, 2011 8:35 AM PDT

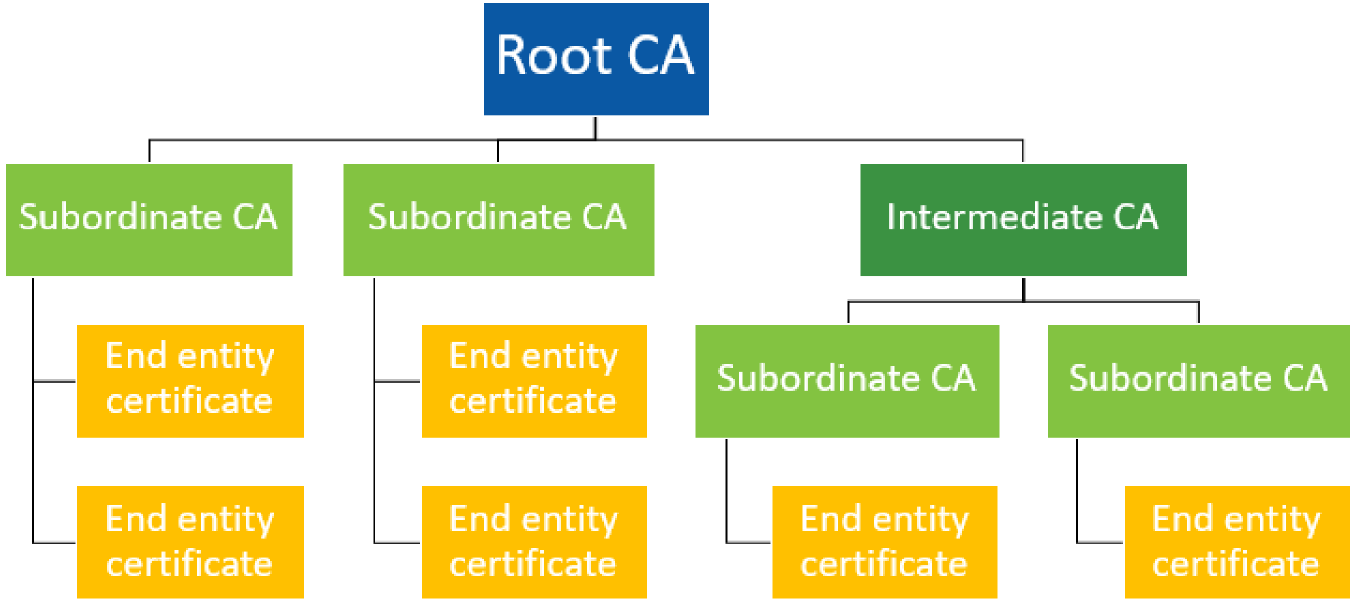
Root CA



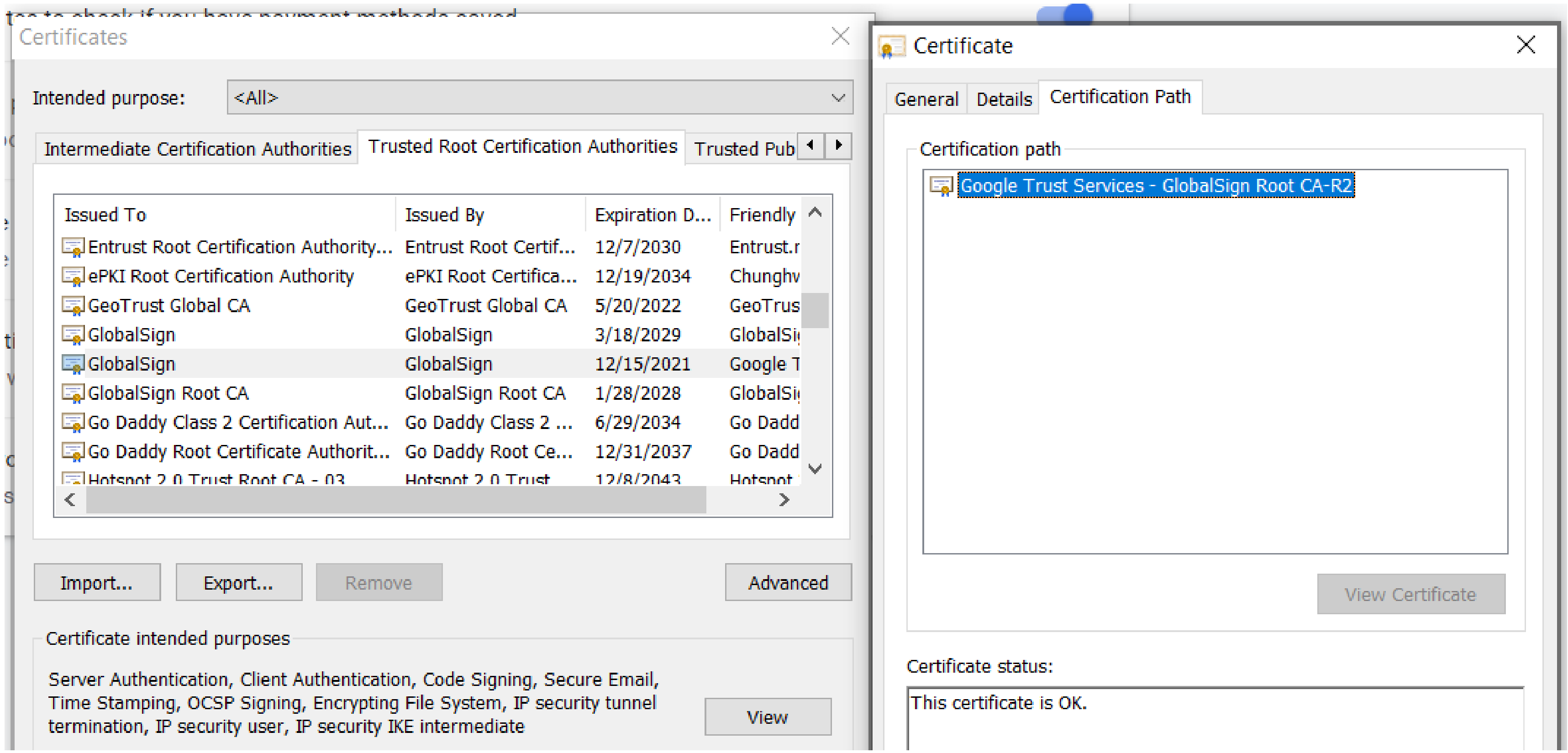
- Verisign, DigiCert are root CAs
- Apple, Microsoft, Google, has their root Cas

Subordinate CA





Trusted CAs

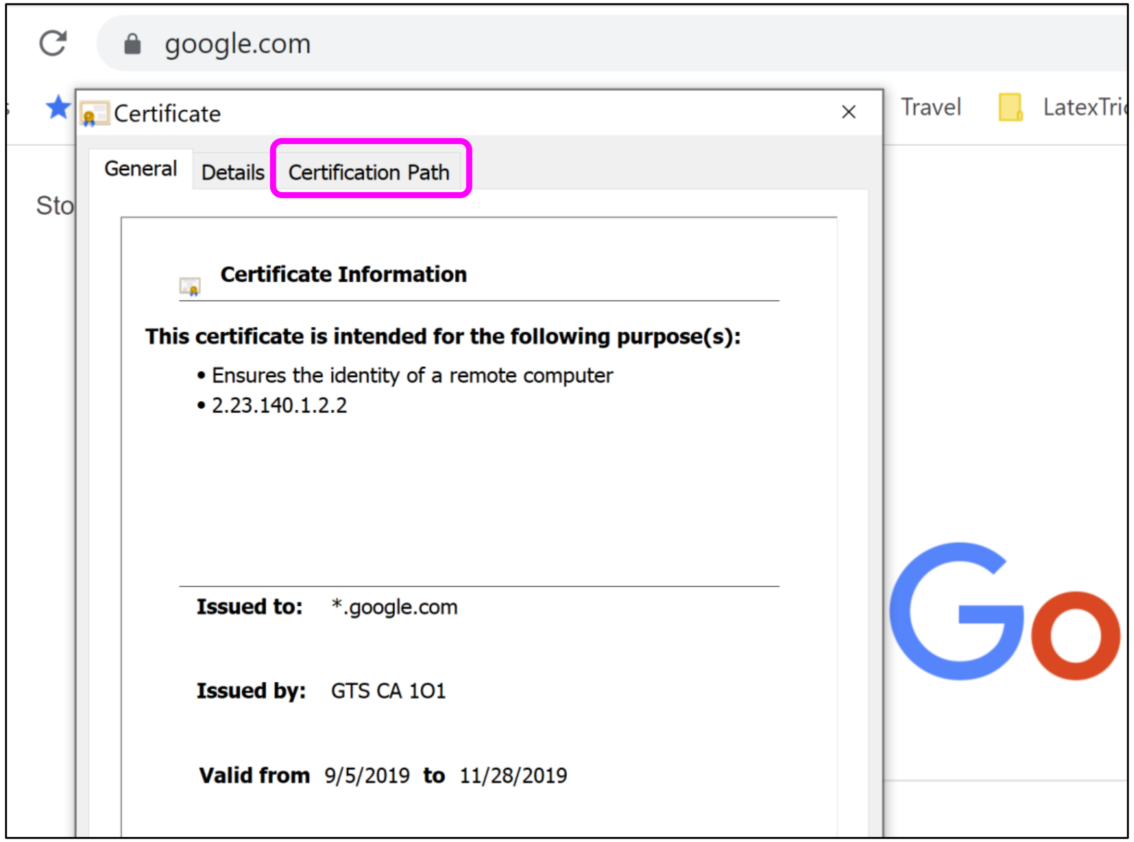

Certificates
Intended purpose: <All>
Intermediate Certification Authorities
Trusted Root Certification Authorities
Trusted Pub
Issued To
Issued By
Expiration D...
Friendly
Entrust Root Certification Authority... Entrust Root Certif... 12/7/2030
ePKI Root Certification Authority ePKI Root Certifica... 12/19/2034
GeoTrust Global CA GeoTrust Global CA 5/20/2022
GlobalSign GlobalSign 3/18/2029
GlobalSign GlobalSign 12/15/2021
GlobalSign Root CA GlobalSign Root CA 1/28/2028
Go Daddy Class 2 Certification Aut... Go Daddy Class 2 ... 6/29/2034
Go Daddy Root Certificate Authorit... Go Daddy Root Ce... 12/31/2037
Import...
Export...
Remove
Advanced
Certificate intended purposes
Server Authentication, Client Authentication, Code Signing, Secure Email, Time Stamping, OCSP Signing, Encrypting File System, IP security tunnel termination, IP security user, IP security IKE intermediate
View
Certificate
General
Details
Certification Path
Certification path
Google Trust Services - GlobalSign Root CA-R2
View Certificate
Certificate status:
This certificate is OK.

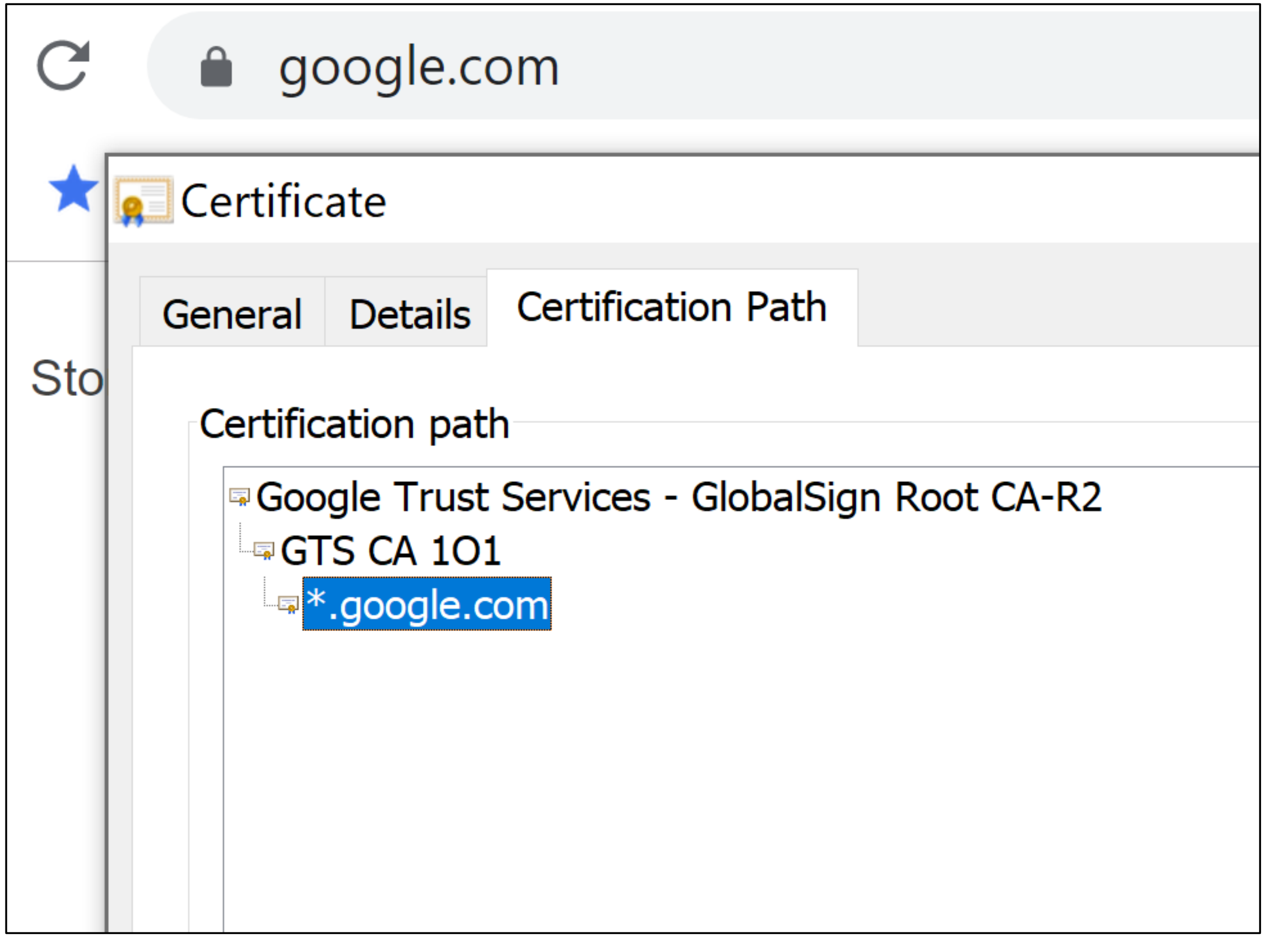
TLS + HTTP => HTTPS

HTTPS Lock: What does it guarantee?

1. Source authentication: The source of the rendered content of the website is indeed from “en.wikipedia.org”
2. Content integrity: The content of the website is not tampered in transit.

Certificate chain (of trust)





Certification revocation

Why?

- unspecified (0)
- keyCompromise (1)
- cACompromise (2)
- affiliationChanged (3)
- superseded (4)
- cessationOfOperation (5)
- certificateHold (6)
- removeFromCRL (8)
- privilegeWithdrawn (9)
- aACompromise (10)

How

- Certificate revocation list (CRL)
  - Can be too long
- Online Certificate Status Protocol (OCSP)
  - Over burdens the CAs
  - Privacy concern
- OCSP Stapling
  - TLS Certificate Status Request

Recap

- Transport Layer Security
  - Above Transport Layer under Application layer
  - Main challenge:
  1. Protocol
  2. Trust of the public key
- Certificate
  - Format X.509
  - Chain of trust beginning at Certificate Authorities
  - Revocation