# Cryptography and the Internet

Ali Aydın Selçuk
TOBB-ETÜ

LKD 2019 Yaz Kampı
22.7.2019

# Early Internet

Started as an academic network for info sharing:

- FTP, e-mail (since 1970s)
- Usenet news (1980s)
- WAIS, Gopher (~1990)
- WWW (since 1991)
  - SSL encryption (1995)

# Modern Internet

Now all sorts of networks have been merged into the Internet:

- Data, voice, TV, entertainment...
- Banking and finance
- Government services
- ...

Internet has become a part of the critical infrastructure, just like water and electricity, always assumed to be there, carrying critical traffic.

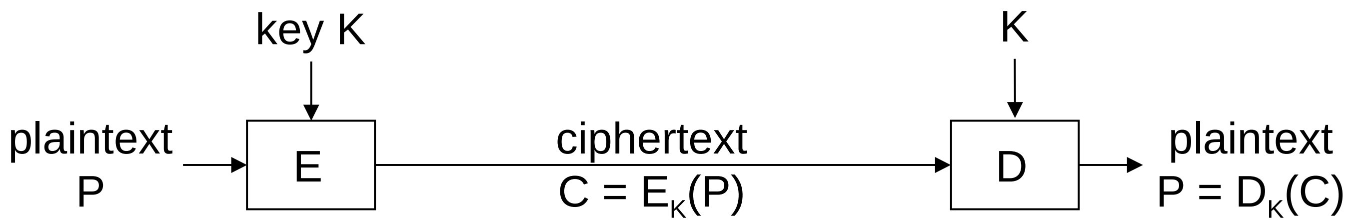
# Success of the Internet

- Design philosophy: dump core, smart endpoints
  - Simple & flexible (compare to the old telephone network)
  - 1970 -> 2020, from kilobits to terabits  (Internet connection of Turkey was 128 Kb/s in 1993).
  - Now TCP/IP is supporting a totally different world than it was designed for. Remarkable!

- Timely arrival of cryptography (*)
  - Without that, the WWW would be just an improved version of Gopher and WAIS.

# Cryptographic Fundamentals

- Basic encryption:




Key: An easy-to-change, variable parameter of the encryption algorithm.

- <u>Kerckhoffs' principle (1883):</u>
  Security should not rely on the secrecy of the algorithm; everything may be known but the key.

# Some Historical Examples

- Shift Cipher:
  - For an n-letter alphabet, $P,C,K \in Z_n$

    $E_K(P) = P + K \bmod n$

    $D_K(C) = C - K \bmod n.$
  - Cryptanalysis: exhaustive key search
  - Solution: increase the key size
- Substitution Cipher:
  - $P,C \in Z_n$; K is a bijection, f, over $Z_n$

    $E_K(P) = f(P)$

    $D_K(C) = f^{-1}(C).$
  - Cryptanalysis: frequency analysis
  - Solution: increase the input domain size

# Some Historical Examples

- One-Time Pad:
  - $P, C, K \in \{0,1\}^n$, for some $n \geq 1$.
    $E_K(P) = P \oplus K$
    $D_K(C) = C \oplus K$
  - Problem: Key needs to be transmitted, which is as long as the message.
  - Used for top-secret applications (E.g., Washington-Moscow red line)

# Modern Ciphers

Shortcomings of historical systems:

- **Substitution cipher:** Small size of the input domain, which enables frequency analysis.
- **One-time pad:** Unlimited key size, which makes key generation and exchange a problem.

Modern ciphers:

- **Block ciphers:** Increasing the size of the input chunks (i.e. blocks) for substitution  (DES, AES)
- **Stream ciphers:** Using a PRNG for generating the key stream  (A5/1, A5/2, RC4)

# Speed Comparisons

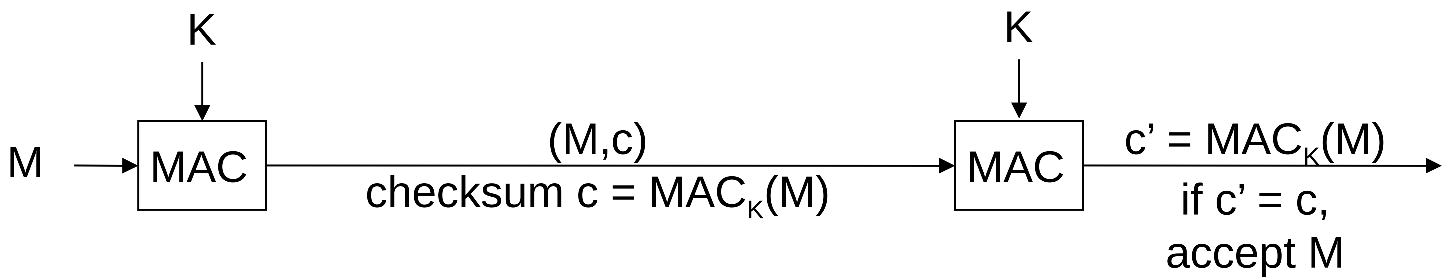
(Crypto++ 5.6 benchmarks, 2.2 GHz AMD Opteron 8354.)

| Algorithm | Speed (MiByte/s.) |
|---|---|
| 3DES (block) | 17 |
| AES-128 (block) | 198 |
| RC4 | 124 |
| SALSA20 | 953 |

(Note: With the new AES-NI instructions, now AES is about 10x faster.)

# Message Authentication

- MAC: “message authentication code”



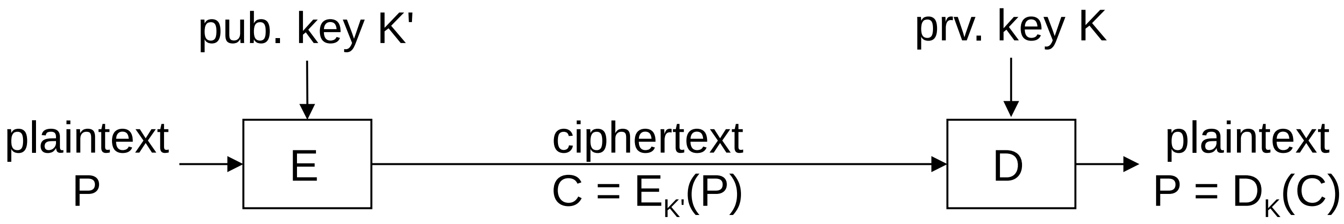

- A checksum (MAC) is computed over the message using the secret key & is transmitted.
- Message is accepted as authentic if the receiver also obtains the same checksum value.

# Message Authentication

- MAC (by symmetric key)
  - Requires that both the sender & the receiver have the same key.
  - E.g., routers under the same administration may share a message authentication key.
- Digital signature (by public key)
  - Uses "asymmetric cryptography"
  - Only one party can sign (with K)
  - But anybody can verify (with K')
  - Very useful in many real-life settings (e.g., authenticating Microsoft patches with a public key)

# Public Key Cryptography

- The single most important idea in modern cryptography.
- Proposed by Diffie & Hellman, 1976 (won the 2015 Turing Award!)
- Asymmetric key cryptography:




- It shouldn’t be possible to obtain K from K'.

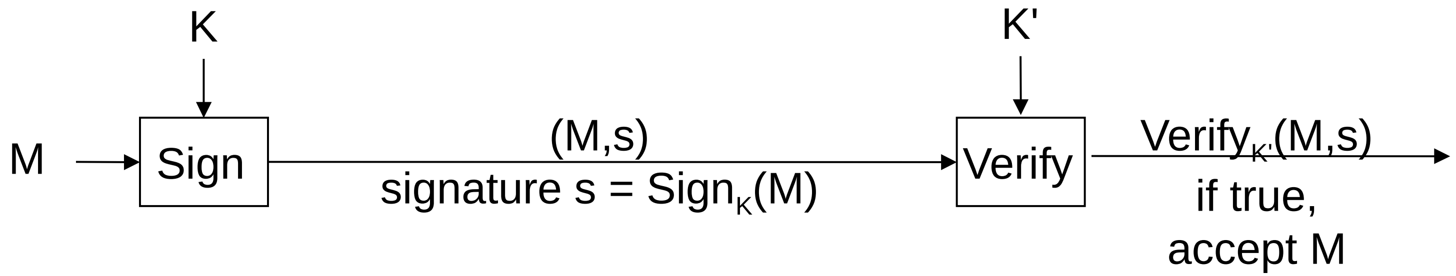
# Public Key Cryptography

- PKC solves the classical “key distribution problem”:
  - If there is no secure channel, how can A & B share the key securely?
- PKC solution:
  - Alice makes her encryption key K' public
  - Everyone can send her an encrypted message:
    $C = E_{K'}(P)$
  - Only Alice can decrypt it with the private key K:
    $P = D_K(C)$

# Digital Signatures

PKC also solves the message source auth. ("digital signature") problem:

- Only Alice can "sign" a message, using K.
- Anyone can verify the signature, using K'.




Only if such a function could be found...

# Discrete Logarithm Problem

- DLP: Given g and $y = g^x$, what is x?

- Easy over Z.
  E.g., if $2^x = 4096$, $x = 12$.

- Hard over $Z_p$.
  E.g., if $2^x = 28 \pmod{113}$, $x = ?$

# Diffie-Hellman Key Exchange

- Public: prime p, generator g  (2048-bit or larger integers)
- Alice chooses random a (secret);
  Bob chooses random b (secret).

Alice | Bob

$$\text{Alice} \xrightarrow{g^a \bmod p} \text{Bob}$$

$$\text{Alice} \xleftarrow{g^b \bmod p} \text{Bob}$$

$$\text{Alice} \longleftrightarrow \text{Bob}$$

$K = g^{ab} \bmod p$

$K = (g^b)^a \bmod p$ $\qquad$ $K = (g^a)^b \bmod p$

# Number Theory Review

Def: $m, n \in Z$ are *relatively prime* if $\gcd(m,n) = 1$.

Def: $Z_n^*$: the numbers in $Z_n$ relatively prime to n.

e.g., $Z_6^* = \{1, 5\}$, $Z_7^* = \{1, 2, 3, 4, 5, 6\}$.

Def: $\varphi(n) = |Z_n^*|$.

e.g., $\varphi(6) = 2$, $\varphi(7) = 6$.

Theorem (Euler): For all $m \in Z_n^*$, we have

$$m^{\varphi(n)} \equiv 1 \pmod{n}.$$

E.g. For $n = 6$, $\varphi(n) = 2$; $x = 5$:

$$x^2 = 25 \equiv 1 \pmod{6}$$

# RSA Cryptosystem

- The first successful pub. key algo. by Rivest, Shamir, Adleman, 1977 (won the 2002 Turing Award!)

- RSA:
  - Alice chooses large primes p, q; $n = pq$.
  - e, such that $\gcd(e, \varphi(n)) = 1$.
  - $d = e^{-1} \bmod \varphi(n)$
  - n, e public. d is the private key.
  - Encryption: $E(x) = x^e \bmod n$
    Decryption: $D(x) = x^d \bmod n$

# RSA Cryptosystem (cont.)

- Enc: $y = x^e \bmod n$
  Dec: $x = y^d \bmod n$

- Correctness: The decrypted text is,

$$
\begin{aligned}
y^d = (x^e)^d &= x^{\varphi(n) \cdot c + 1} \mod n \\
&= (x^{\varphi(n)})^c \, x \mod n \\
&= x
\end{aligned}
$$

# RSA Cryptosystem (cont.)

- Security: Relies on difficulty of factoring n.
  - If n= p·q is known, then so is $\varphi$(n), and d.
  - Conversely, if we can find d, we can factor n.
  - Hence, finding d ≡ factoring n.
- Any other ways to obtain x from e, n, y? Probably not.
- Suggested key lengths:
  - short term: 2048 bits
  - longer term: 4096 bits

# Speed Comparisons

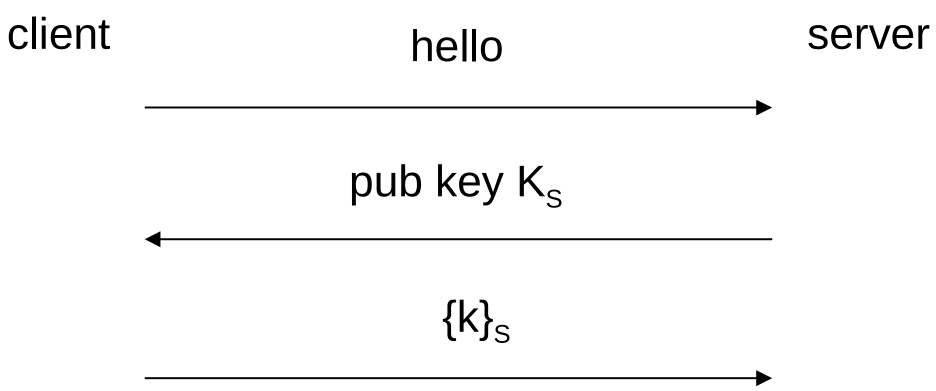
(Crypto++ 5.6 benchmarks, 2.2 GHz AMD Opteron 8354.)

| Algorithm | enc. time (ms/op.) | dec. time (ms/op.) |
|---|---|---|
| AES-128 (block) | 0.00008 | 0.00008 |
| RSA-2048 | 0.08 | 2.90 |

- Public key operations are much slower than symmetric key operations.
- Typically, PKC is used for the initial session key exchange, and then the symmetric key is used for the rest of the session.

# A Simple Protocol

~ SSL key exchange protocol:




k is the session key

# Active Attacks & Certificates

- Simple public key encryption solves the key distribution problem against passive attackers (i.e., an attacker that just eavesdrops).
- Active attackers can send a fake public key & become a "man in the middle" (MitM).

Notation:

- $[M]_X$ : message M signed with the prv. key of X
- $\{M\}_X$ : message M enc. with the pub. key of X

# MitM Attack

Normal op:

Alice

pub key $K_A$

Bob

$\{k\}_A$

k is the session key

MitM attack:

Trudy

Alice

$K_A$

$K_{A'}$

Bob

$\{k\}_A$

$\{k\}_{A'}$

key k

key k

# MitM Attack against DH

Normal op:

$$\text{Alice} \xrightarrow{g^a \bmod p} \text{Bob}$$

$$\text{Alice} \xleftarrow{g^b \bmod p} \text{Bob}$$

$$k = g^{ab} \bmod p$$

MitM attack:

Trudy

$$\text{Alice} \xrightarrow{g^a \bmod p} \text{Trudy} \xrightarrow{g^{a'} \bmod p} \text{Bob}$$

$$\text{Alice} \xleftarrow{g^{b'} \bmod p} \text{Trudy} \xleftarrow{g^b \bmod p} \text{Bob}$$

$$k' = g^{ab'} \bmod p \qquad k'' = g^{a'b} \bmod p$$

# Certificates

- These attacks are possible because a receiver cannot distinguish fake and real public keys.

- We need to “bind” pub.keys with user identities.

- Certificates: IDs and public keys are signed by a trusted authority (“certification authority”).

- E.g., $cert_A = [ID_A, PK_A, \text{exp.date}, \ldots]_{CA}$

# Certification Authorities

- CA’s public key should have been distributed in a trusted way to all the parties in the system.
- For instance, in SSL:
  - CAs are accredited by browser makers.
  - Accredited CAs’ public keys are embedded in the browser code & distributed to the users.
  - https://wiki.mozilla.org/CA:How_to_apply
- PKI: Public key infrastructure
  - A hierarchy of CAs, with one or more trusted roots, that issue certificates to a given domain of end users.

# Key Exchange with Certificates

~ SSL key exchange protocol:

client — server

client $\xrightarrow{\text{hello}}$ server

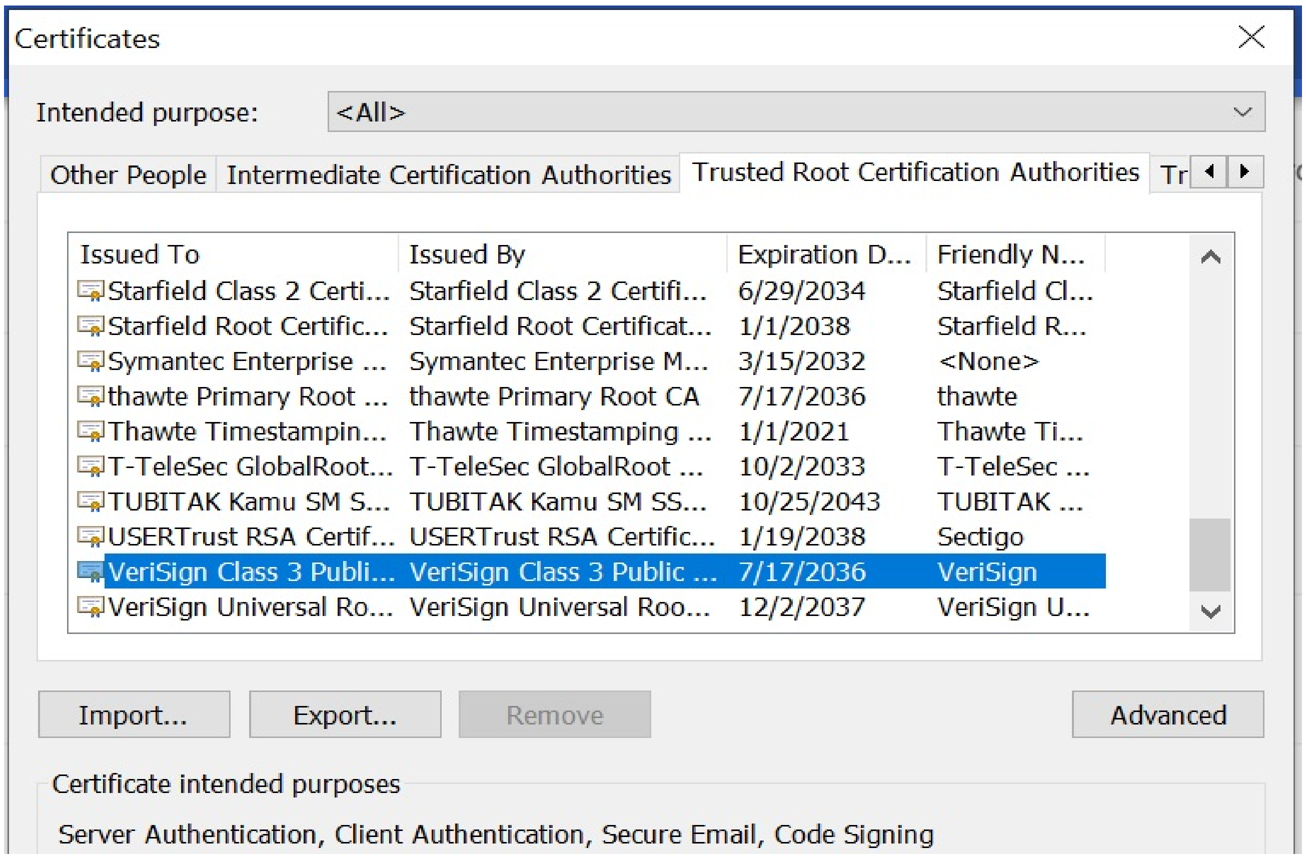
client $\xleftarrow{cert_S = [ID_S,\ PK_S,\ \ldots]_{CA}}$ server

client $\xrightarrow{\{k\}_S}$ server

k is the session key

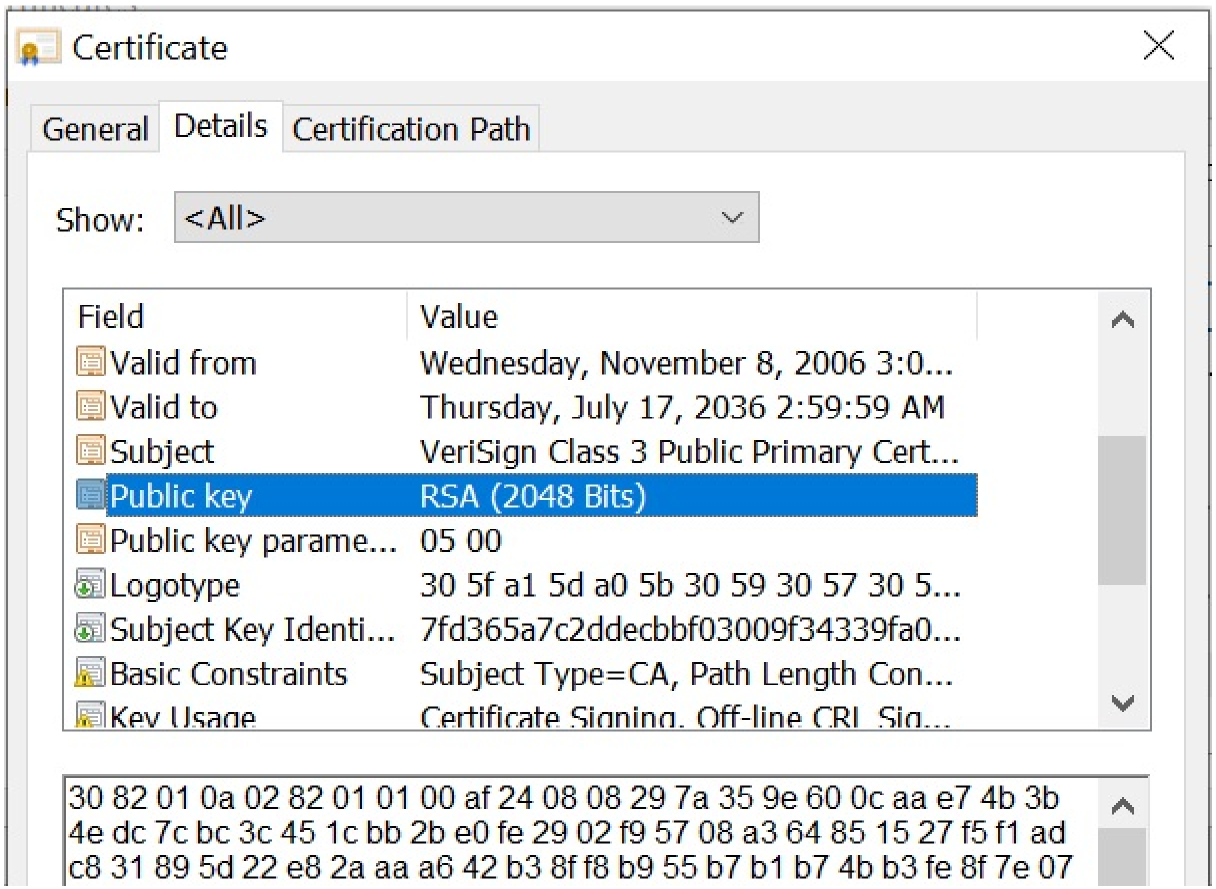
# Example: Chrome (on Win10)

- Settings > Advanced > Manage certificates
- Trusted root CAs:



# Example: Chrome (on Win10)

- E.g., VeriSign root certificate:



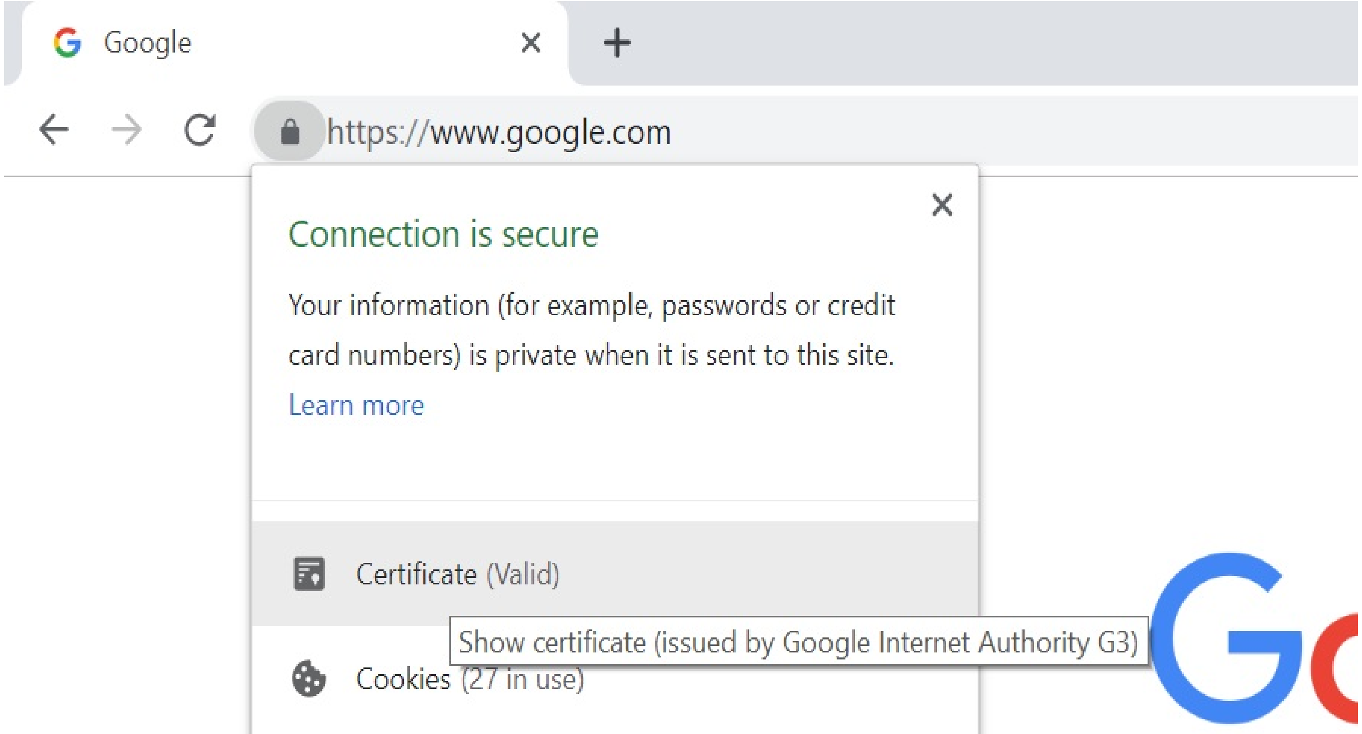
# Certificates & Validation

- Valid SSL/TLS certificates are issued to web servers by root or intermediate CAs.
  - E.g., Google's certificate: GeoTrust (root) → Google Internet Authority → accounts.google.com
- Client (browser) authenticates this chain of certificates beginning from the root CA.

  http://en.wikipedia.org/wiki/Certification_path_validation_algorithm

# Example Client Certificate

- E.g., google.com

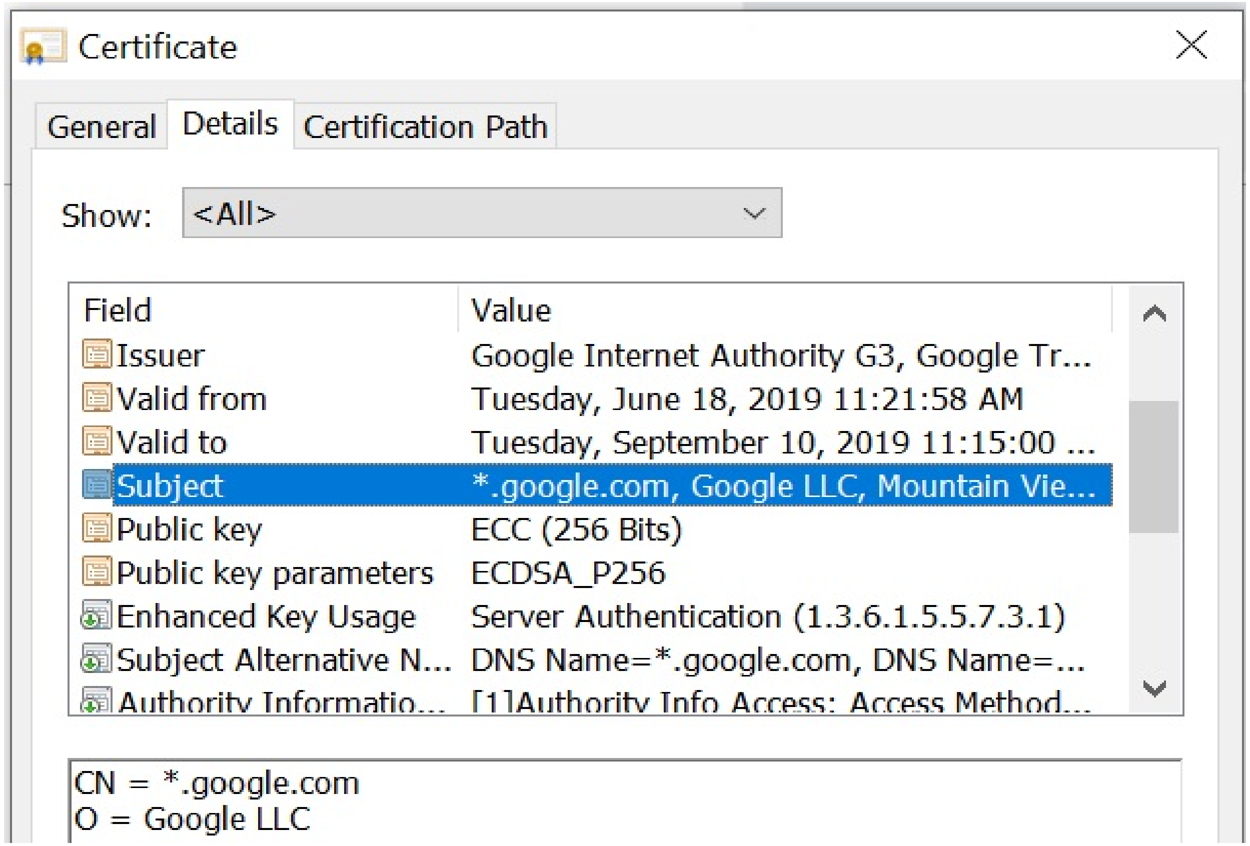


# Example Client Certificate

- E.g., google.com

# Example Client Certificate

- E.g., gmail.com   (or, accounts.google.com)



# Key Exchange by Encryption

~ SSL key exchange protocol:

client → server: hello

server → client: $cert_S = [ID_S, PK_S, \ldots]_{CA}$

client → server: $\{k\}_S$

k is the session key

# Certified DH Key Exchange

~ IPsec key exchange protocol:

$$\text{Alice} \qquad \qquad \text{Bob}$$

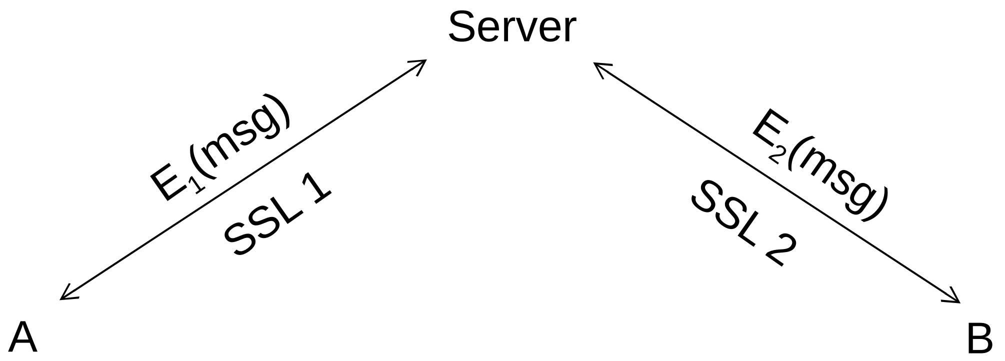
$$\xrightarrow{[g^a \text{ mod } p]_A,\ \text{cert}_A}$$

$$\xleftarrow{[g^a \text{ mod } p]_B,\ \text{cert}_B}$$

$$k = g^{ab} \text{ mod } p$$

# Case: WhatsApp Encryption

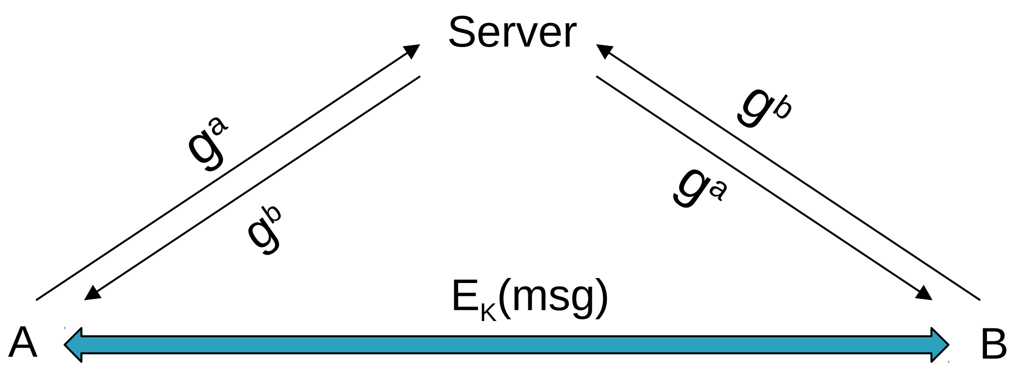
The old way: security by SSL encryption




- Protects against outsiders.
- But server has access to chat messages.

# Case: WhatsApp Encryption

The new way: end-to-end encryption after DH key exchange:




- Public keys (DH) are exchanged via the server.
- Server has no access to chat messages.
- Session key K can be verified by QR code.

# Internet & Cryptography

- It is not just the WWW and SSL; crypto is used everywhere: VPNs, app-layer security, wireless security, routing and DNS security…

- A perfect match:
  - By the Internet, cryptography and secure communication have been brought to the masses.
  - By cryptography, the Internet has become the critical network of the world.