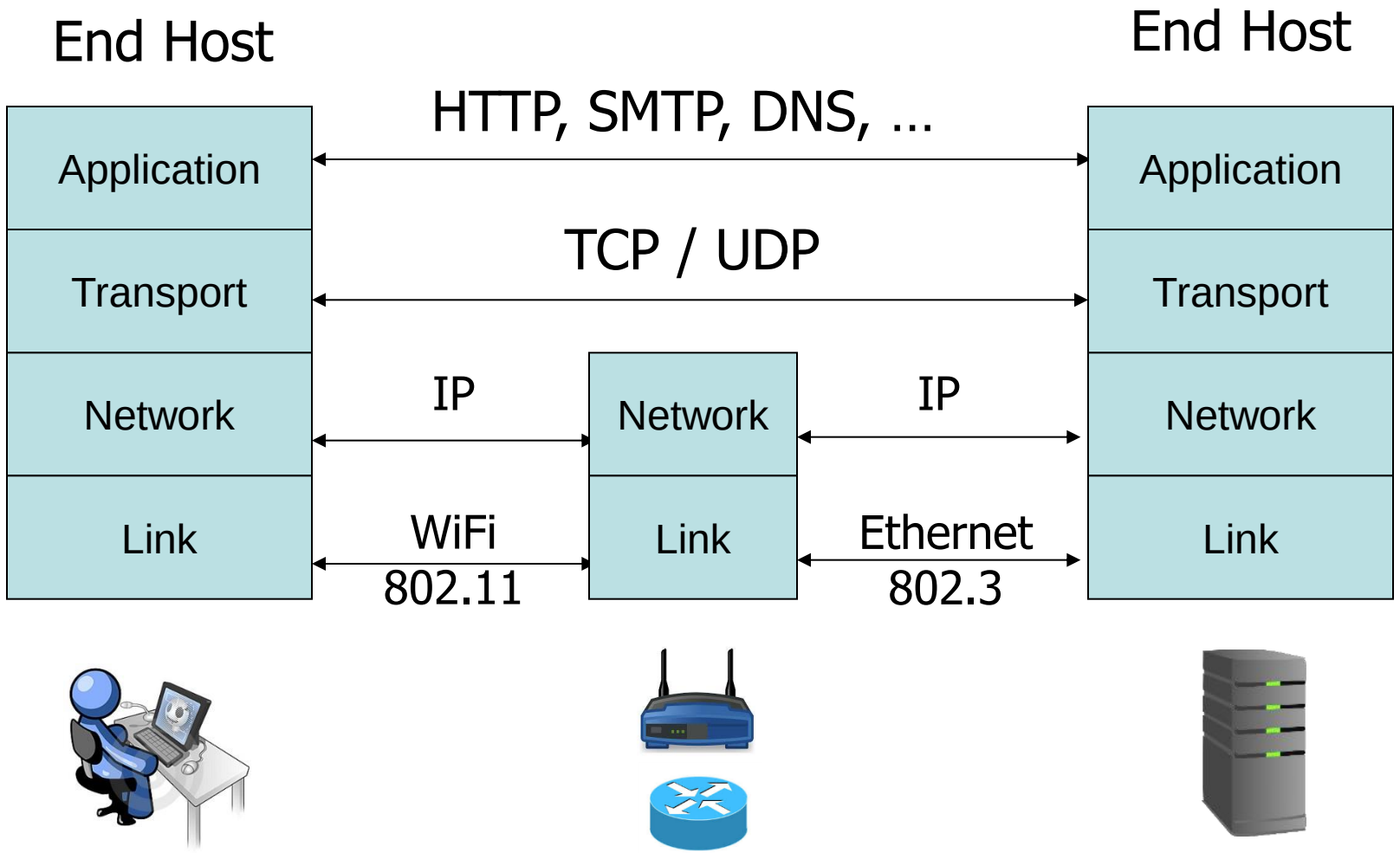
institute imdea software

www.software.imdea.org

HTTPS, TLS, SSL

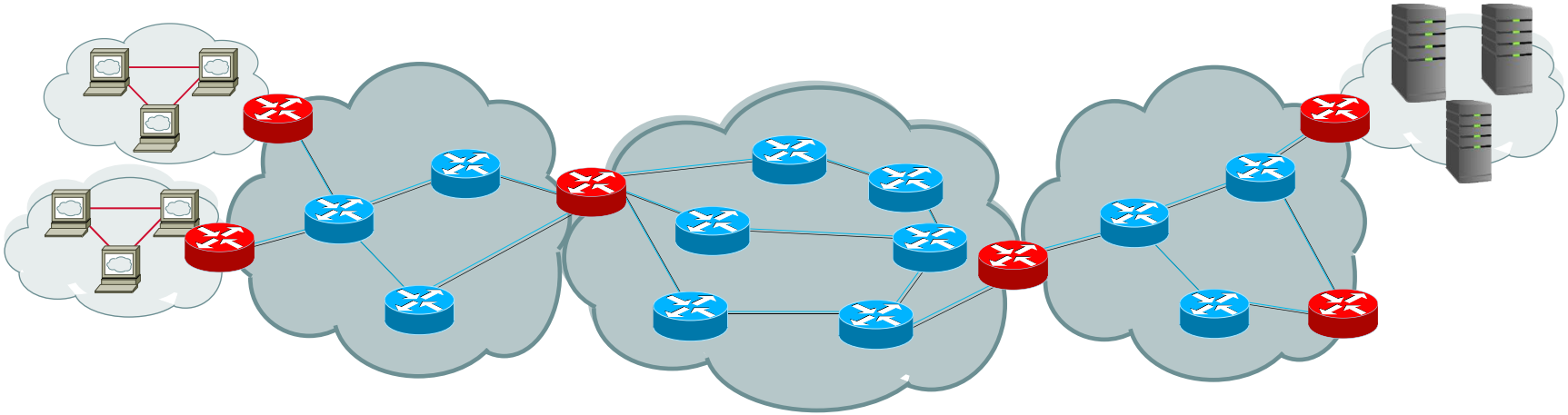
TCP/IP Protocol Stack



(Network) Security Properties

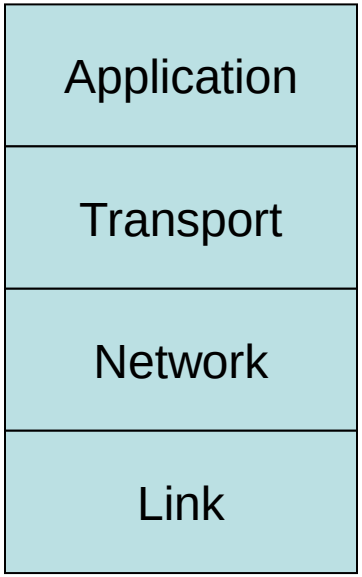
- Confidentiality
  - Information kept secret
- Integrity
  - Communication not modified
- Authentication
  - Communication between intended parties
- Availability
  - Communication happens in reasonable time

Network Attacker



- Most common threat model in network security
- Controls network traffic
  - Can eavesdrop, inject, drop, modify traffic
- Examples
  - Untrusted access network (e.g., WiFi Router, Hotel, Enterprise)
  - Rogue ISP
  - State-level: NSA, Great Firewall of China

Network Security Protocols



HTTPS, PGP, Kerberos, DNSSec, …

TLS/SSL, SSH

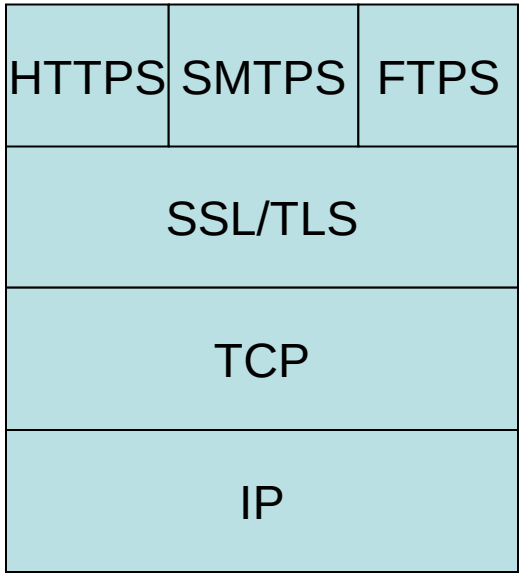
IPSec

WPA3, WPA2, WEP, 802.11x, …

Lecture Outline

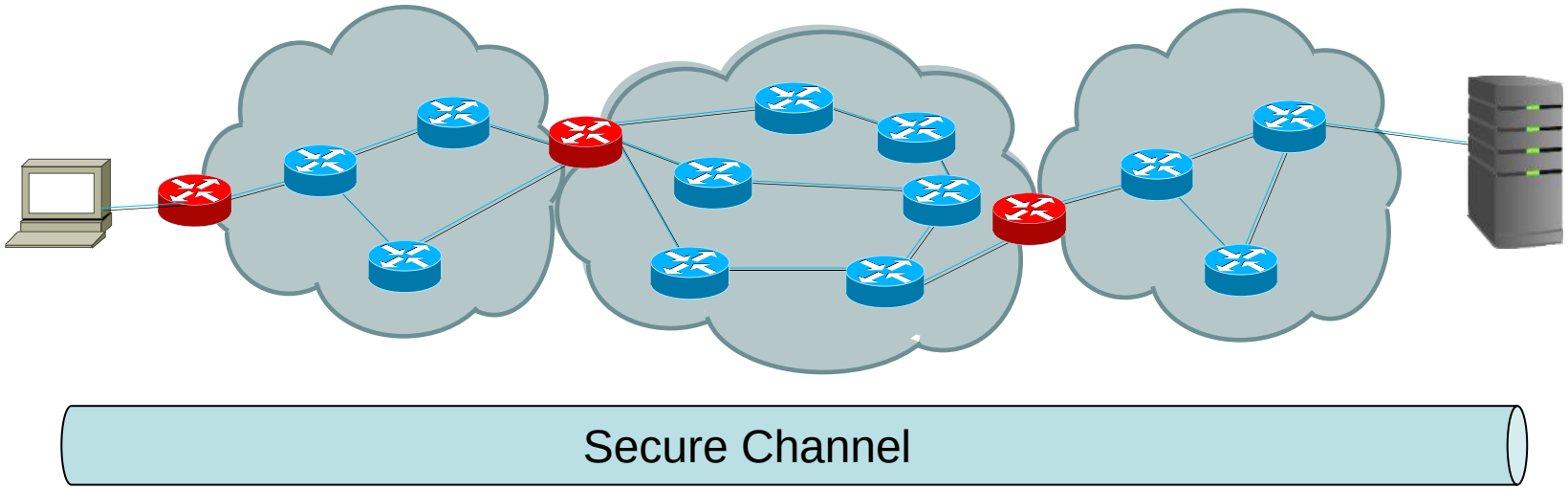
- How TLS/SSL works?
- How HTTPS builds on TLS/SSL?
- HTTPS issues

HTTPS, TLS, SSL



- TLS = Transport Layer Security
- SSL = Secure Sockets Layer
- HTTPS = HTTP over SSL/TLS

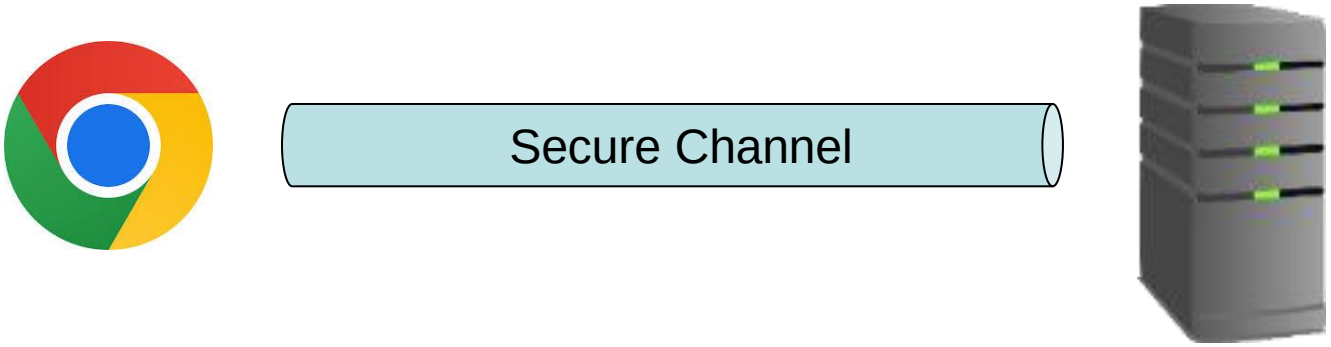
TLS in a Nutshell



- Two hosts establish a secure communication channel
- Secure communication over insecure network
- End-to-end security

HTTPS in a Nutshell

https://software.imdea.org



- A client and a Web server can have a **secure conversation over an insecure** network having never met before

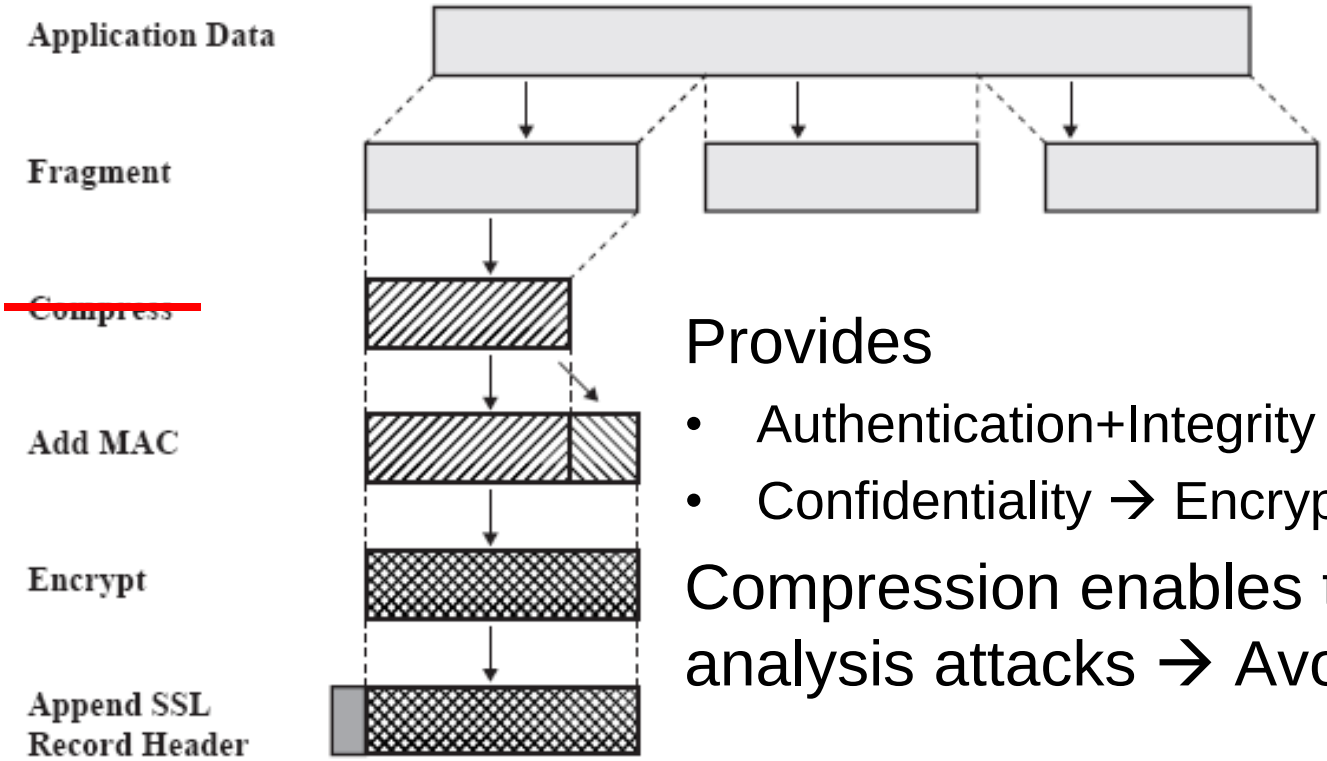
SSL / TLS Properties

- Confidentiality
  - Symmetric cryptography used to encrypt the data
  - Keys generated uniquely for each connection
  - Keys negotiated at the start of the session
- Integrity
  - Each message transmitted includes HMAC
- Authentication
  - Identity of the parties can be authenticated using public-key cryptography.
  - Each message transmitted includes HMAC
  - Typically, only client authenticates server

SSL / TLS History

- ~~SSL v1.0 (Sep 1994)~~ **Never deployed**
  - Initial protocol design by Netscape
- ~~SSL v2.0 (Dec 1994)~~ **Deprecated in 2011**
  - Several independent implementations
  - Public review
- ~~SSL v3.0 (Dec 1995)~~ **Deprecated in 2015**
  - Fixed security flaws found during public review
  - Very similar to later TLS standard
- ~~TLS 1.0 (RFC 2246, Jan. 1999)~~ **Deprecated by PCI DSS 06/2018**
  - First standardization by IETF
- ~~TLS 1.1 (RFC 4346, Apr 2006)~~ **Deprecated in 2020**
  - Added protection against cipher-block chaining (CBC) attacks
- TLS 1.2 (RFC 5246, Aug 2008)
  - SHA256 support added
  - Client and server can specify hash, signature algorithms they accept.
- TLS 1.3 (RFC 8446, Aug 2018)
  - Dropping support for many insecure features → Few supported ciphers, hashes, …
  - Fast handshake: 1-RTT
  - Encrypted certificate

SSL Record Protocol



Provides

- Authentication+Integrity → MAC
- Confidentiality → Encryption

Compression enables traffic analysis attacks → Avoid using it

Cipher Suites

- Defines:
  - key exchange method (e.g., RSA)
  - block/stream cipher (e.g., AES)
  - hashing algorithm (e.g., SHA-256)
- Examples:
  - TLS_RSA_WITH_AES_128_CBC_SHA (TLS 1.2)
  - TLS_AES_256_GCM_SHA384 (TLS 1.3)
- TLS 1.3 separates key agreement and authentication alg. from cipher suites

Cipher Suite: Key Exchange

- TLS supports multiple key exchange / auth. protocols

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes
DH-DSS	No	Yes	Yes	Yes	Yes	No
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No[45]
ECDH-ECDSA	No	No	Yes	Yes	Yes	No
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes
PSK	No	No	Yes	Yes	Yes	
PSK-RSA	No	No	Yes	Yes	Yes	
DHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes	
ECDHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes	
SRP	No	No	Yes	Yes	Yes	
SRP-DSS	No	No	Yes	Yes	Yes	
SRP-RSA	No	No	Yes	Yes	Yes	
Kerberos	No	No	Yes	Yes	Yes	
DH-ANON (insecure)	No	Yes	Yes	Yes	Yes	
ECDH-ANON (insecure)	No	No	Yes	Yes	Yes	
GOST R 34.10-94 / 34.10-2001[46]	No	No	Yes	Yes	Yes	

Forward Secrecy

- A feature of key exchange protocols
- Protects past communication
- Unique session key for every session → Compromise of session key does not affect other sessions
- Compromise of server private key does not affect past (before compromise) session keys
- Achieved with Ephemeral Diffie-Hellman

Cipher Suite: Ciphers

- TLS supports multiple block and stream ciphers

Cipher			Protocol version					
Type	Algorithm	Nominal strength (bits)	SSL 2.0	SSL 3.0 [n 1][n 2][n 3][n 4]	TLS 1.0 [n 1][n 3]	TLS 1.1 [n 1]	TLS 1.2 [n 1]	TLS 1.3
Block cipher with mode of operation	AES GCM[47][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	Secure
	AES CCM[48][n 5]		N/A	N/A	N/A	N/A	Secure	Secure
	AES CBC[n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	Camellia GCM[49][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A
	Camellia CBC[50][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	ARIA GCM[51][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A
	ARIA CBC[51][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	SEED CBC[52][n 6]	128	N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	3DES EDE CBC[n 6][n 7]	112[n 8]	Insecure	Insecure	Insecure	Insecure	Insecure	N/A
	GOST 28147-89 CNT[46][n 7]	256	N/A	N/A	Insecure	Insecure	Insecure	N/A
	IDEA CBC[n 6][n 7][n 9]	128	Insecure	Insecure	Insecure	Insecure	N/A	N/A
	DES CBC[n 6][n 7][n 9]	56	Insecure	Insecure	Insecure	Insecure	N/A	N/A
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A
	RC2 CBC[n 6][n 7]	40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A
Stream cipher	ChaCha20-Poly1305[57][n 5]	256	N/A	N/A	N/A	N/A	Secure	Secure
	RC4[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	N/A
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A
None	Null[n 12]	–	N/A	Insecure	Insecure	Insecure	Insecure	N/A

Cipher Suite: Hashes

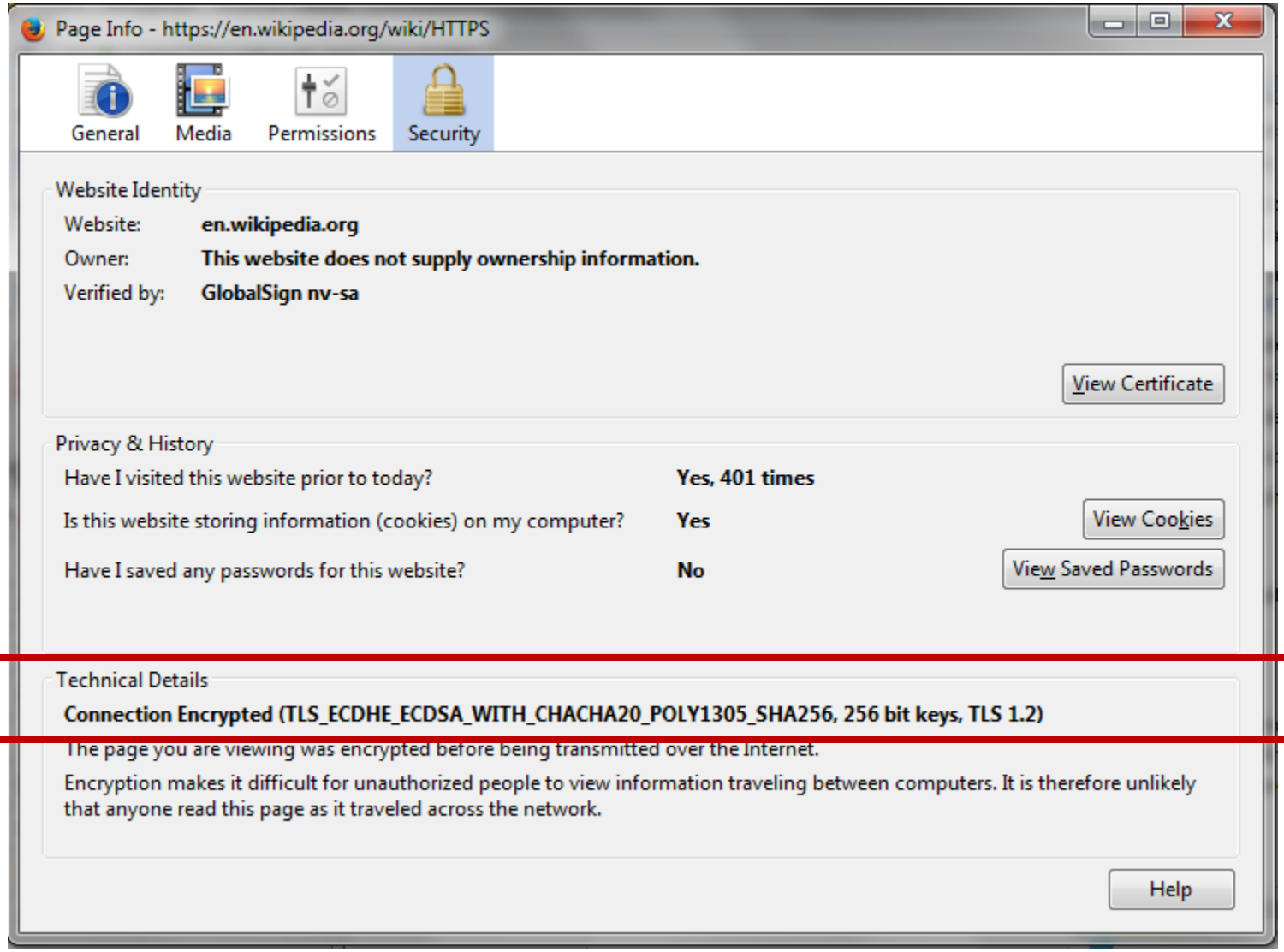
- ~~MD5 (16 bytes, 128 bit)~~
  - Collisions have been achieved
  - In 2008, Sotirov et al. created fake CA cert
- ~~SHA1 (20 bytes, 160 bit)~~
  - Stevens et al. show collisions can be achieved with reasonable budget ($75K–$120K)
  - NIST deprecates it for government use in 2010
  - CA/Browser forum deprecates it for Internet use in 2011
  - Chosen-prefix collision (Leurent and Peyrin, 2020)
- SHA256 (32 bytes, 256 bit)
- SHA384 (48 bytes, 384 bit)

Cipher Suite: Integrity

Data integrity

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
HMAC-MD5	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
HMAC-SHA1	No	Yes	Yes	Yes	Yes	No	
HMAC-SHA256/384	No	No	No	No	Yes	No	
AEAD	No	No	No	No	Yes	Yes	
GOST 28147-89 IMIT[53]	No	No	Yes	Yes	Yes		Proposed in RFC drafts
GOST R 34.11-94[53]	No	No	Yes	Yes	Yes		

Cipher Suite Example

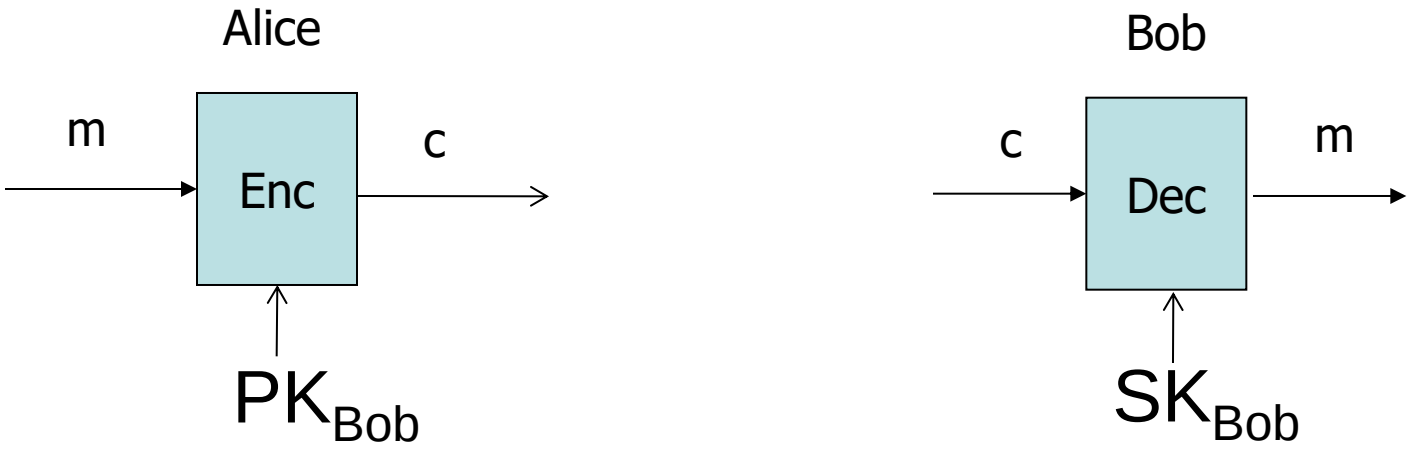


SSL Handshake Protocol

- Establishes keys and other needed data (nonces, initialization vectors, …)
- Authenticates parties
  - Server authentication
  - Optional client authentication
- Uses public-key cryptography
- Certificates used to authenticate public keys
- After SSL handshake completes, all communication is encrypted and signed

Public-Key Encryption

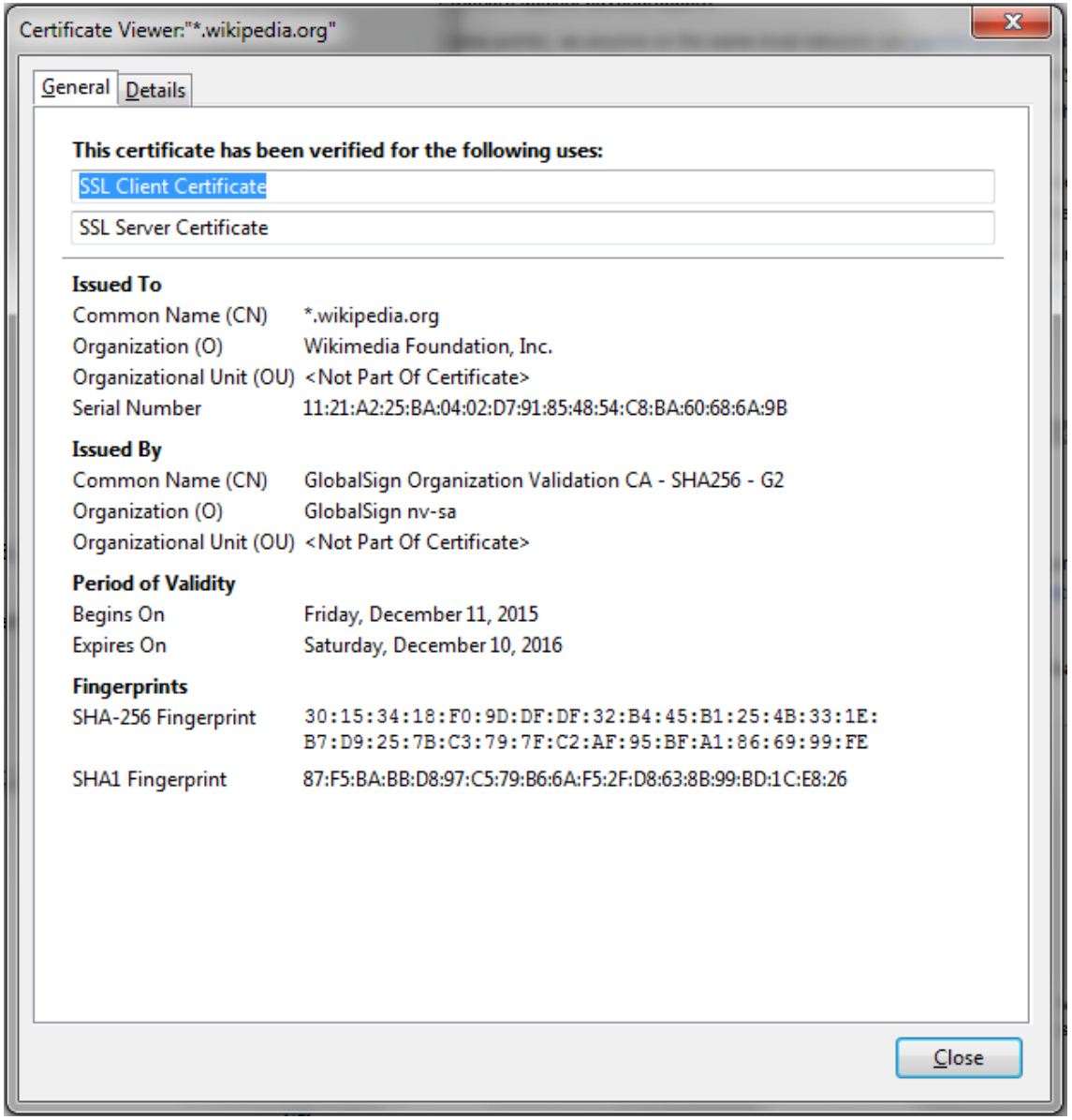
- Each party has two keys: public and private
  - Public key (PK) can/should be shared
  - Private key (SK) is secret
- Alice encrypts using Bob's public key
- Only Bob can decrypt using its private key



Certificates

- Alice needs to obtain Bob's public key
- How does Alice trust the provided key is really Bob's public key?
- Trusted Certification Authority (CA) emits certificate for Bob's public key
  - Digitally signed by CA
  - Anyone with CA's public key can validate signature
- Certificate links Bob's identity to his PK
- Certificate has limited validity period
  - The longer the validity, the more expensive

X.509 Certificates



Subject

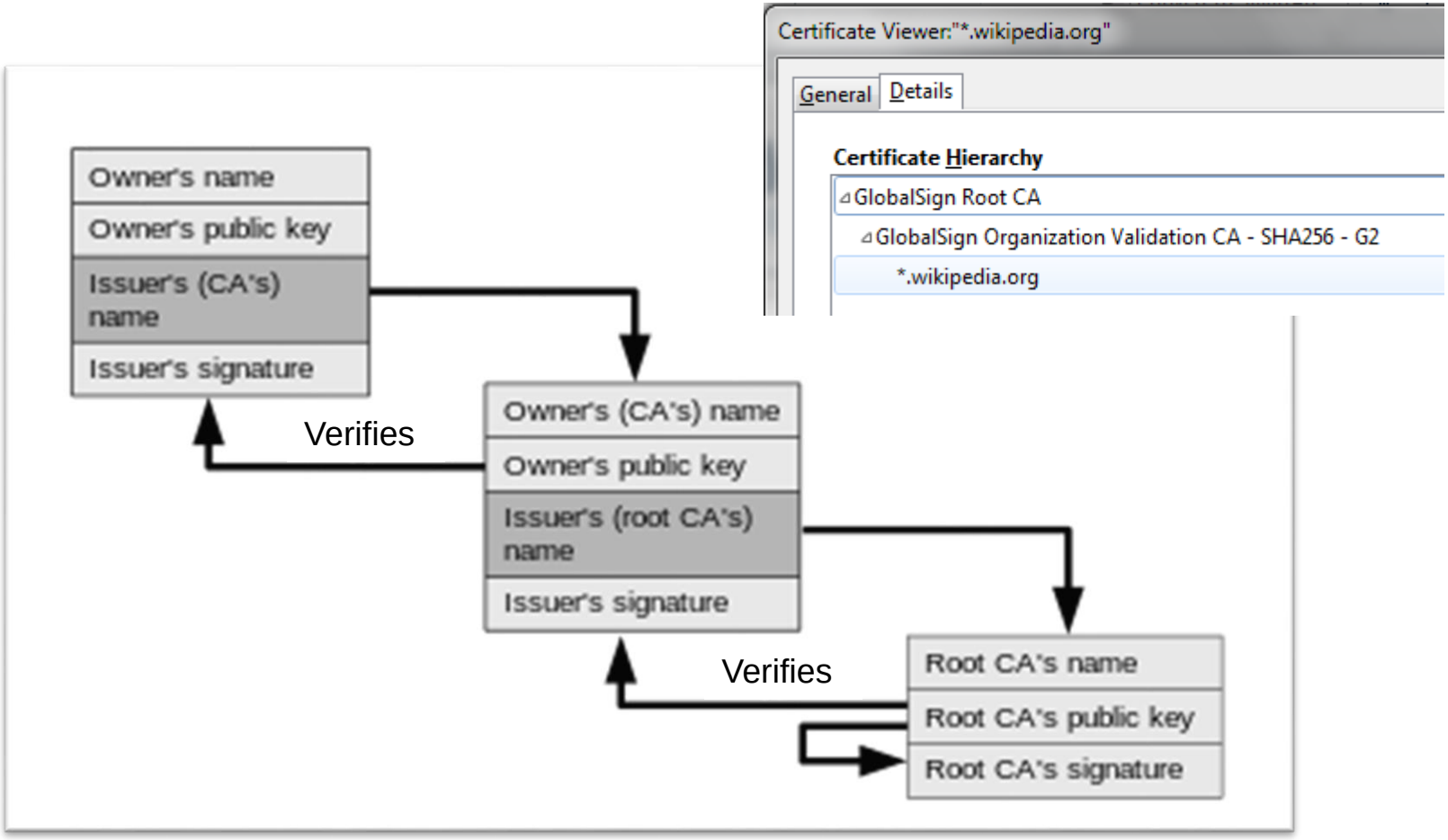
Issuer

Validity

Hash of digital signature

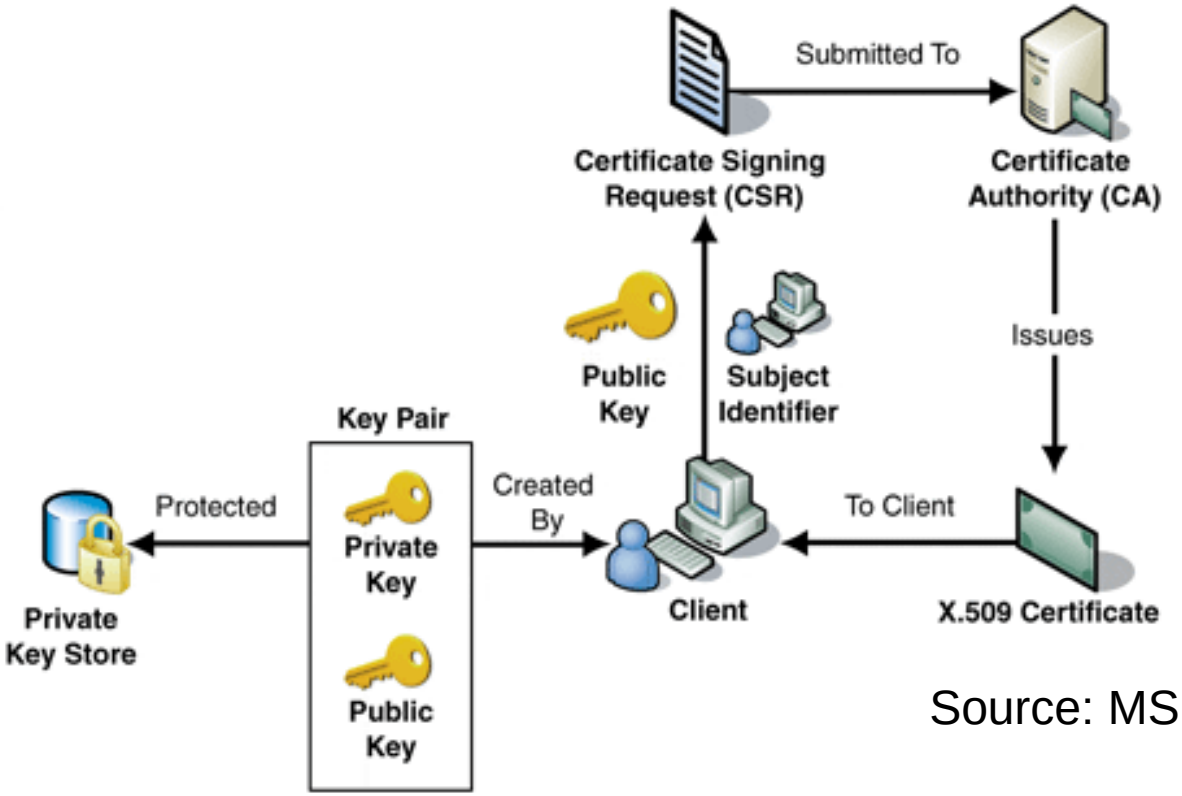
Public Key info
Extensions
…

Certificate Chain



Invalid chains (e.g., self-signed certificates) enable TLS/SSL interception

Requesting a Certificate



Source: MSDN

Automated mechanisms for requesting a certificate can also be used
ACME (Automated Certificate Management Environment) RFC 8555

Certificate Revocation

- Certificates may need to be revoked before expiration
  - Private key compromised
  - System administrator leaving company

- Two revocation methods:

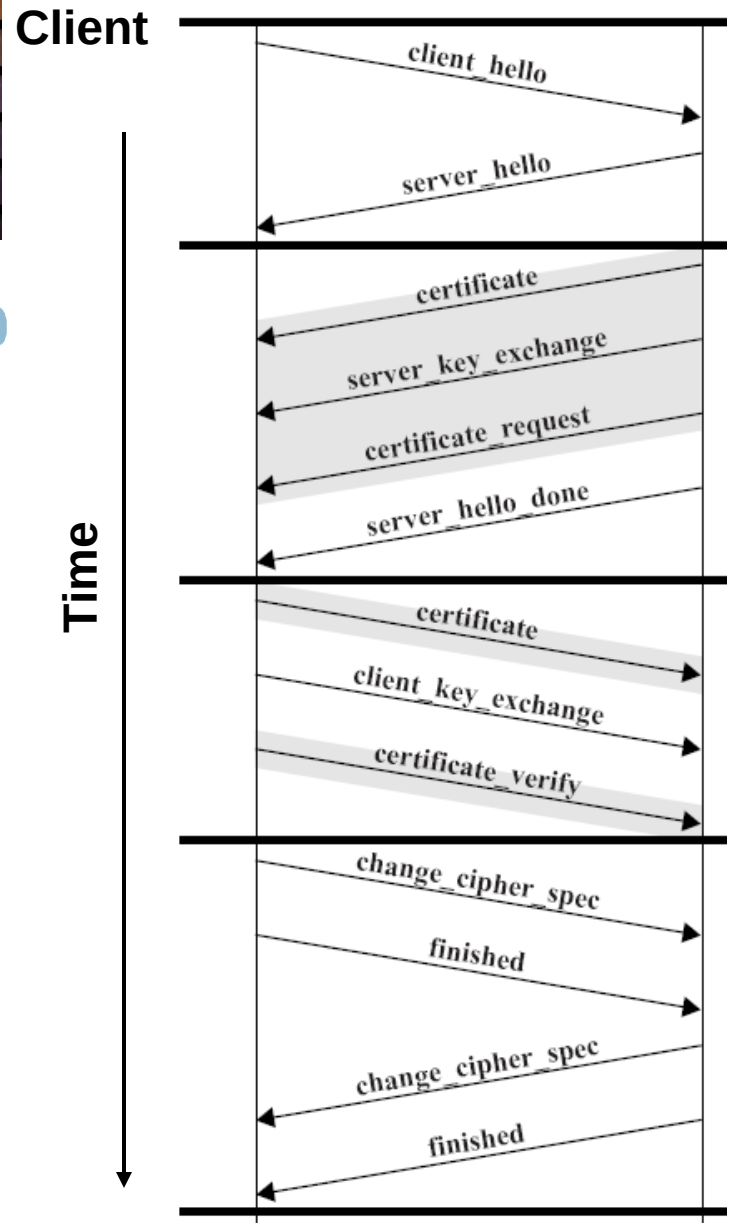
1. Certificate Revocation Lists (CRL)
   - List of revoked certificates periodically published by CA
   - Large size, constantly increasing

2. Online Certificate Status Protocol (OCSP)
   - If online authority is unavailable, verification fails
   - OCSP Stapling → TLS extension to insert ("staple") a time-stamped OCSP response signed by the CA in TLS handshake

Short-Term Certificates

- An alternative to revocation
- Issue a sequence of short-validity certificates (e.g., 1-3 days) terminated upon compromise.
- Short-term, automatically renewed (STAR) certificates using ACME
- CA is responsible for publishing the next certificate at an agreed upon URL before previous one expires.
- RFC 8739 (March 2020)

SSL Handshake Protocol



Phase 1: Establish security capabilities

version, cipher suite, session id, nonces

Phase 2: Server

send server certificate chain,
key exchange,
request client cert chain

Phase 3: Client

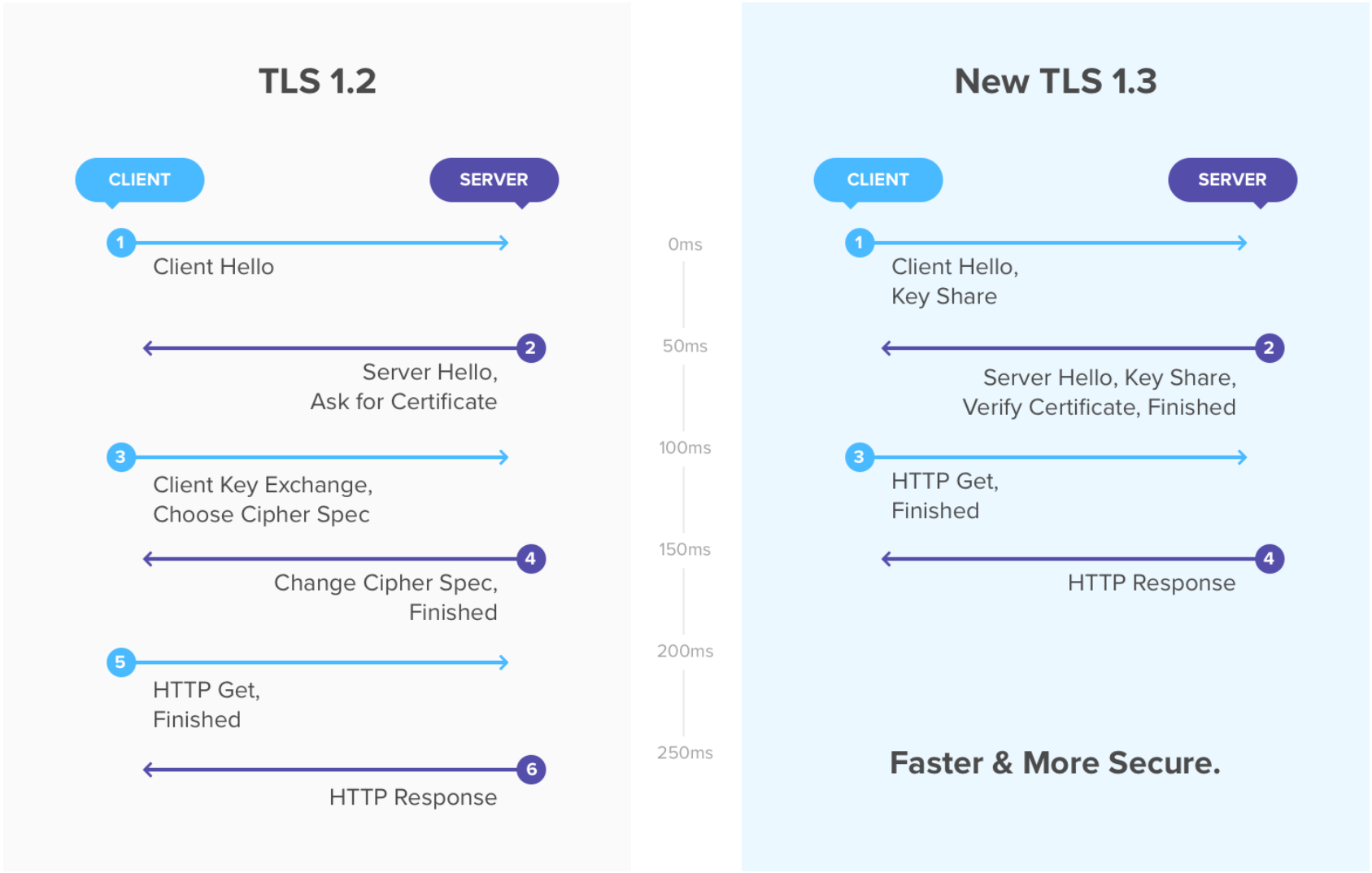
verify server certificate chain
(optional) send client certificate chain,
key exchange

Phase 4: Change Suite and finish

(optional) verify client certificate chain
change cipher suite and finish handshake

Shaded transfers are present for some cipher suites only

TLS 1.3 1-RTT Handshake



institute imdea software

Lecture Outline

- How TLS/SSL works?
- How HTTPS builds on TLS/SSL?
- HTTPS issues

HTTPS Certificates

- Subject's Common Name contains website's domain name (or IP address)
- Certificate can be valid for multiple domains using Subject Alternative Names (SAN) extension
- Two options
  - Explicit name, e.g., software.imdea.org
  - Wildcard, e.g., *.imdea.org
- Wildcard matching rule (RFC 2818)

  * must be in leftmost dot-separated component, e.g., *.imdea.org matches software.imdea.org, but not malicialab.software.imdea.org
- Browser checks certificate's Subject CN or SAN matches visited domain/IP in URL

Trusted CAs

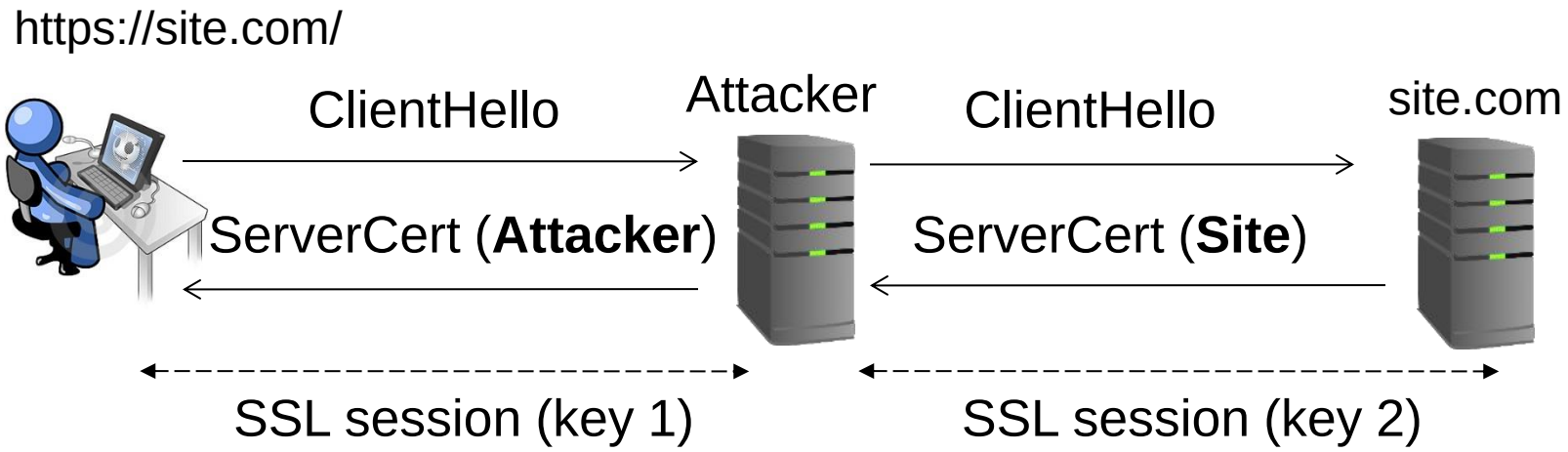
- Trusted/Root CAs saved in certificate store
- Certificate store

Browser certificate store

OS certificate store

- Mozilla CA Store (December 8th, 2021)
  - 53 CAs
  - Should we trust them all?

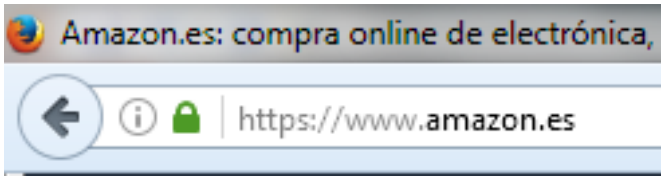
HTTPS Man-in-the-Middle Attack



- Attacker proxies traffic between user and target site
- Attacker can observe and modify traffic
- Attacker's certificate can be:
  - Invalid, e.g., self-signed → Won't validate
  - Valid → Rogue trusted CA or cert installed locally

HTTPS Lock Icon

- Positive security indicator
  - Indicates user page contents secure against network attacker
  - Provides page origin identity



- Displayed when:
  - All page elements fetched using HTTPS
  - Site's certificate chain valid (e.g., not expired and rooted at CA trusted by browser)
  - Domain in URL matches leaf certificate Common Name or Subject Alternative Name

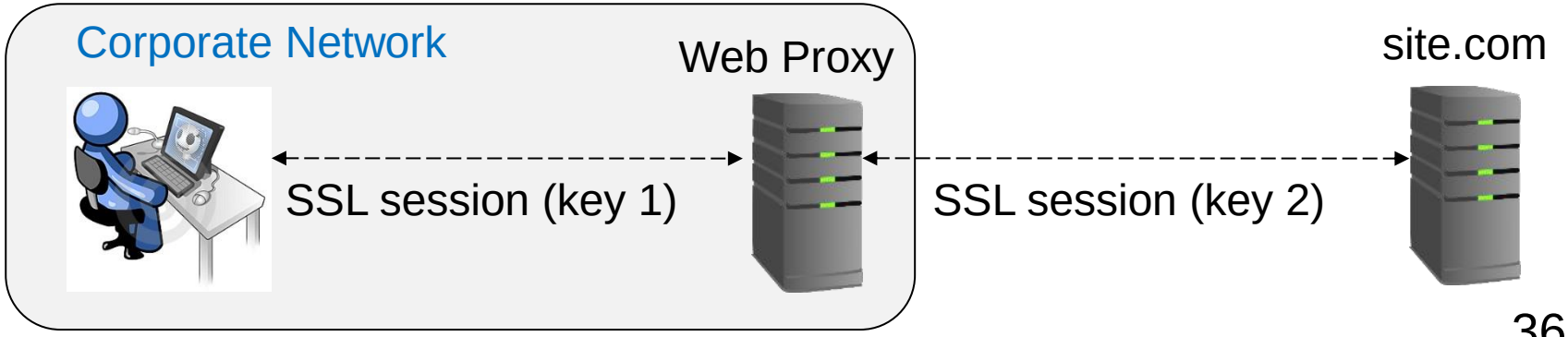
Positive security indicators do not work well.
Research shows that users do not notice their absence.

SNI header

- TLS client hello extension
- Enables HTTPS virtual hosting
- Allows a server to present multiple certificates on the same IP address and TCP port

HTTPS and Web Proxies

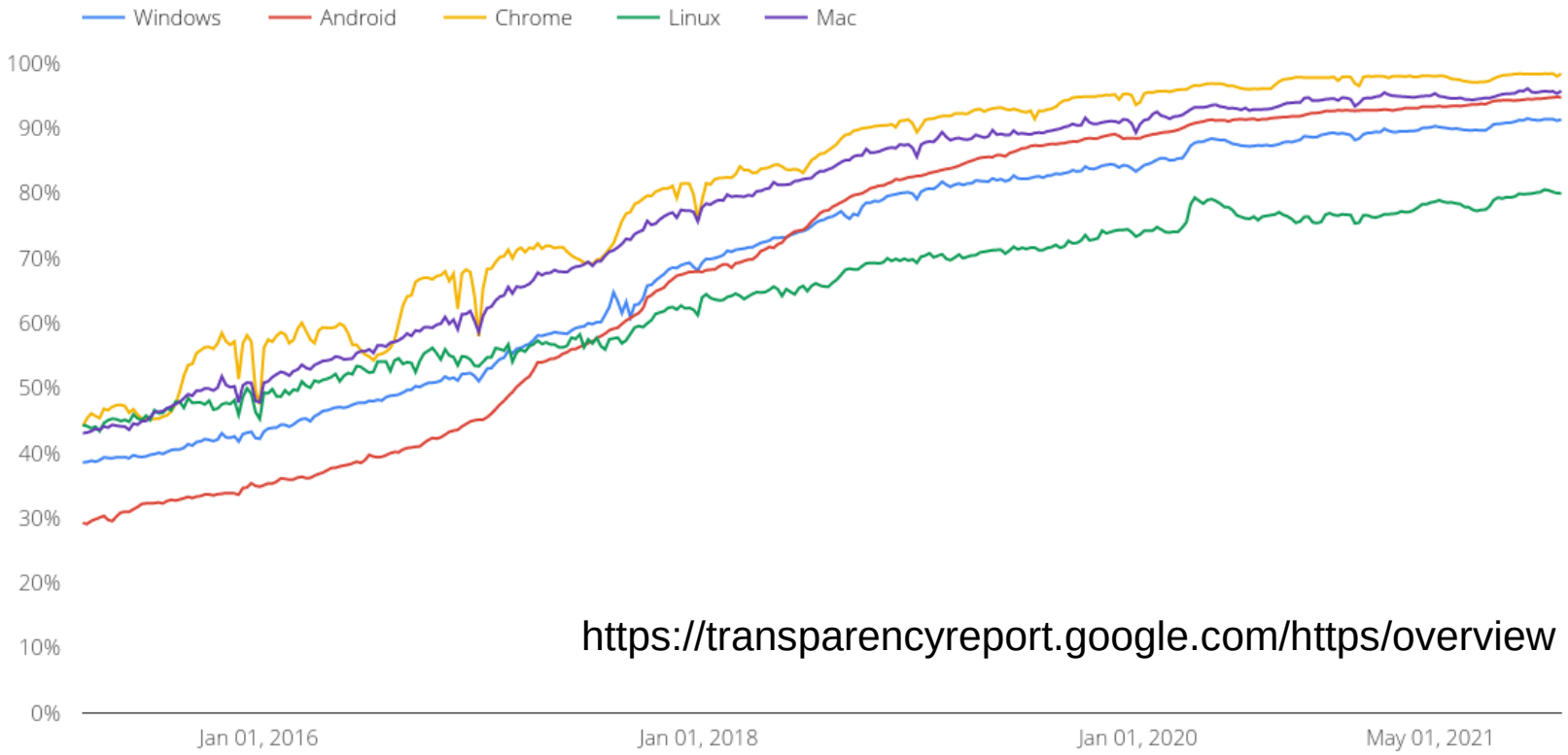
- Option 1: HTTP Tunnel
  - Browser sends HTTP “CONNECT domain” to proxy
  - Proxy transparently forwards TCP packets to destination
  - Proxy does not observe plaintext
- Option 2: Man-in-the-Middle interception
  - Proxy establishes two SSL connections
  - Company forces user to install Trusted CA certificate
  - Proxy observes plaintext



Lecture Outline

- How TLS/SSL works?
- How HTTPS builds on TLS/SSL?
- HTTPS issues
  1. Why HTTPS not used everywhere?
  2. Do you trust your Trusted CAs?
  3. Mixing of HTTP and HTTPS content

HTTPS Usage



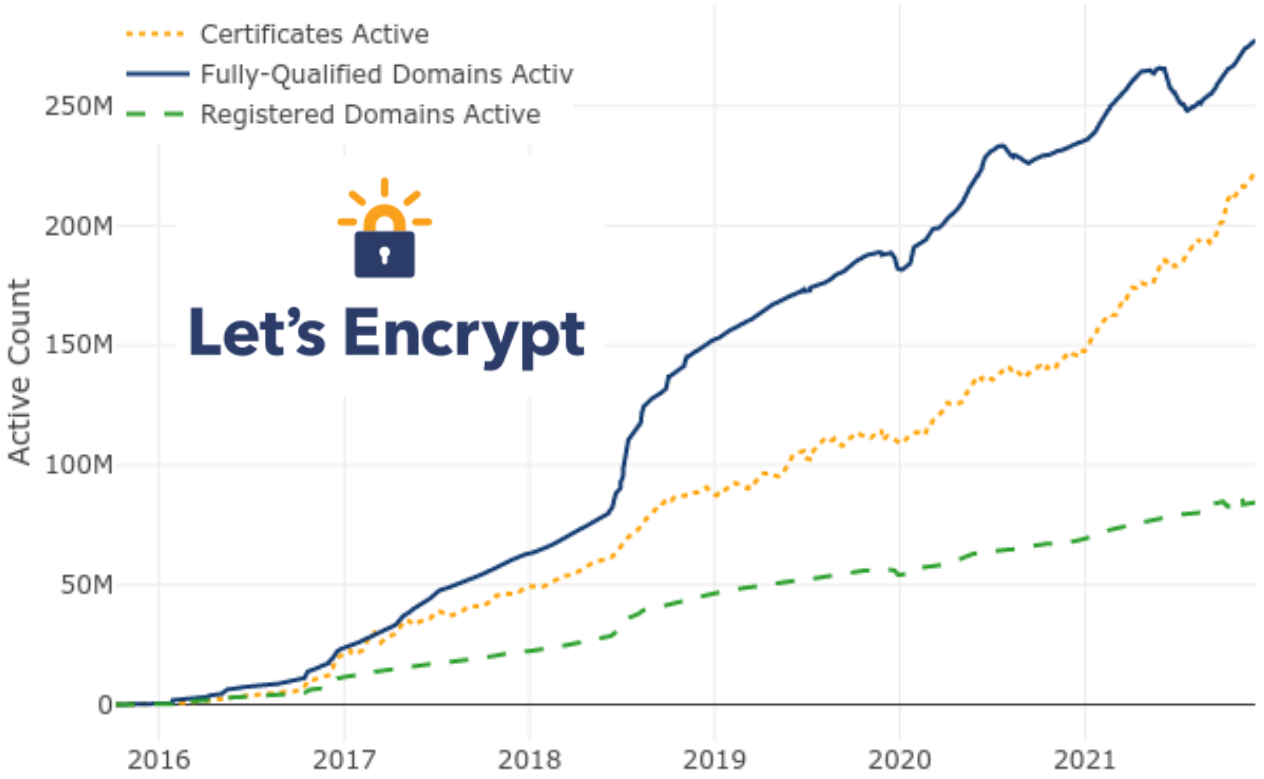
https://transparencyreport.google.com/https/overview

- 98% of Chrome page loads use HTTPS (Dec. 2021)
  - Measured by Google Telemetry
  - Increasing usage over time
- 77% of sites use HTTPS as default (Dec. 2021)
  - https://w3techs.com/technologies/details/ce-httpsdefault

Why HTTPS not everywhere?

- HTTPS slows down servers
  - But not much, check https://istlsfastyet.com/
- Certificates cost money
  - Let's Encrypt (next slides)
- Some Ad-networks do not support HTTPS
  - Reduces website revenues
- Old Browsers (e.g., IE 6)
  - No SNI support → problems with virtual hosting
  - <1% browsers

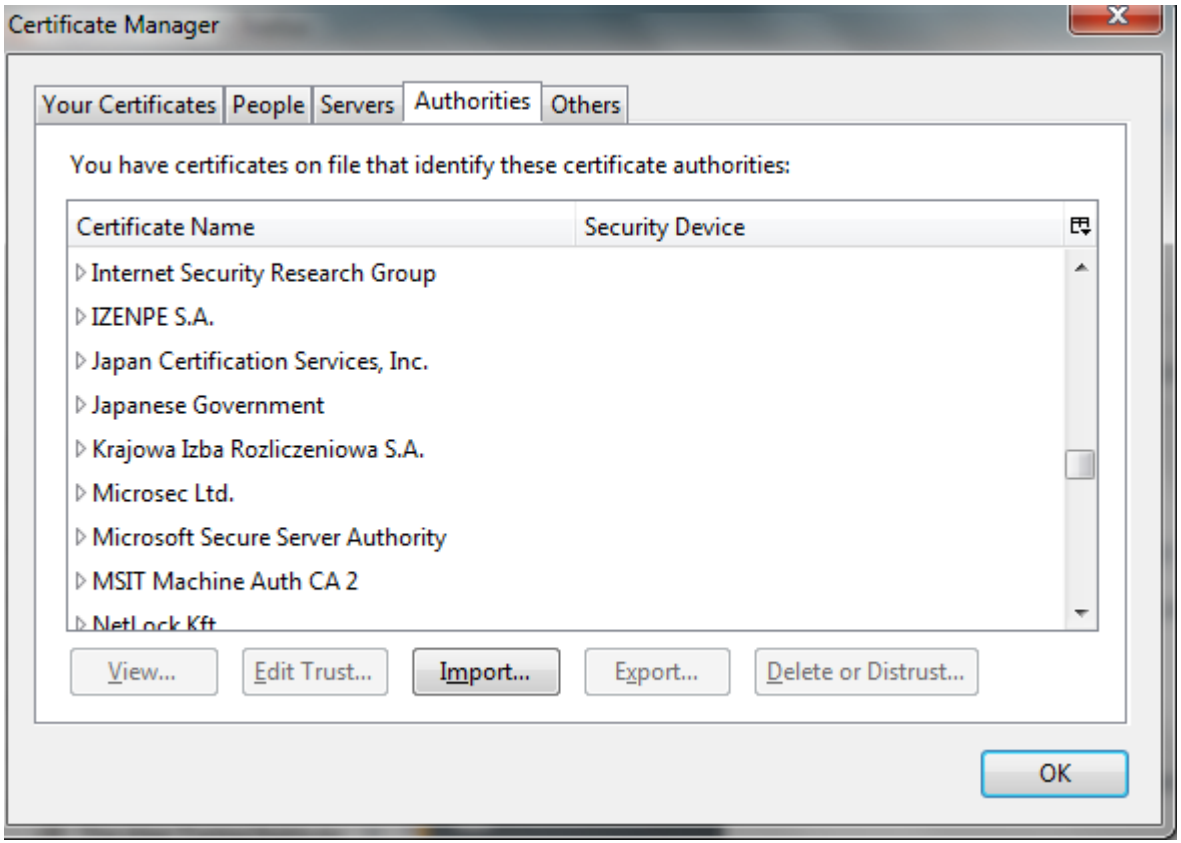
Solution: Let's Encrypt



- Free, Automated, Open CA
- Issues **free domain certificates**
- Goal is to get 100% of Web using HTTPS
- Issues between 2M and 3M certificates per day

Do you trust your Trusted CAs?

- Any trusted CA can generate certificate for any domain
- A single CA compromise brings down HTTPS



Wrongly Issued Certificates

2011: **Comodo/DigiNotar** CAs hacked, issue certificates for Gmail, Yahoo! Mail, …

- Suspected interception by Iranian government on citizens

2013: **TurkTrust** issued certificate for gmail.com

2014: **Indian NIC** (intermediate CA issued by **IndiaCCA** root CA) issue certs for Google and Yahoo! Domains

- India CCA revoked NIC's intermediate certificate
- Chrome restricts India CCA root to 7 Indian domains

2015: **MCS** (intermediate CA cert issued by **CNNIC**) issues certs for Google domains

- CNNIC root removed from Chrome

2016: Startcom/WoSign issued base domain certs if requestor proved control of a subdomain

- A person got cert over github.com by proving control of x.github.com

Solution: Public Key Pinning

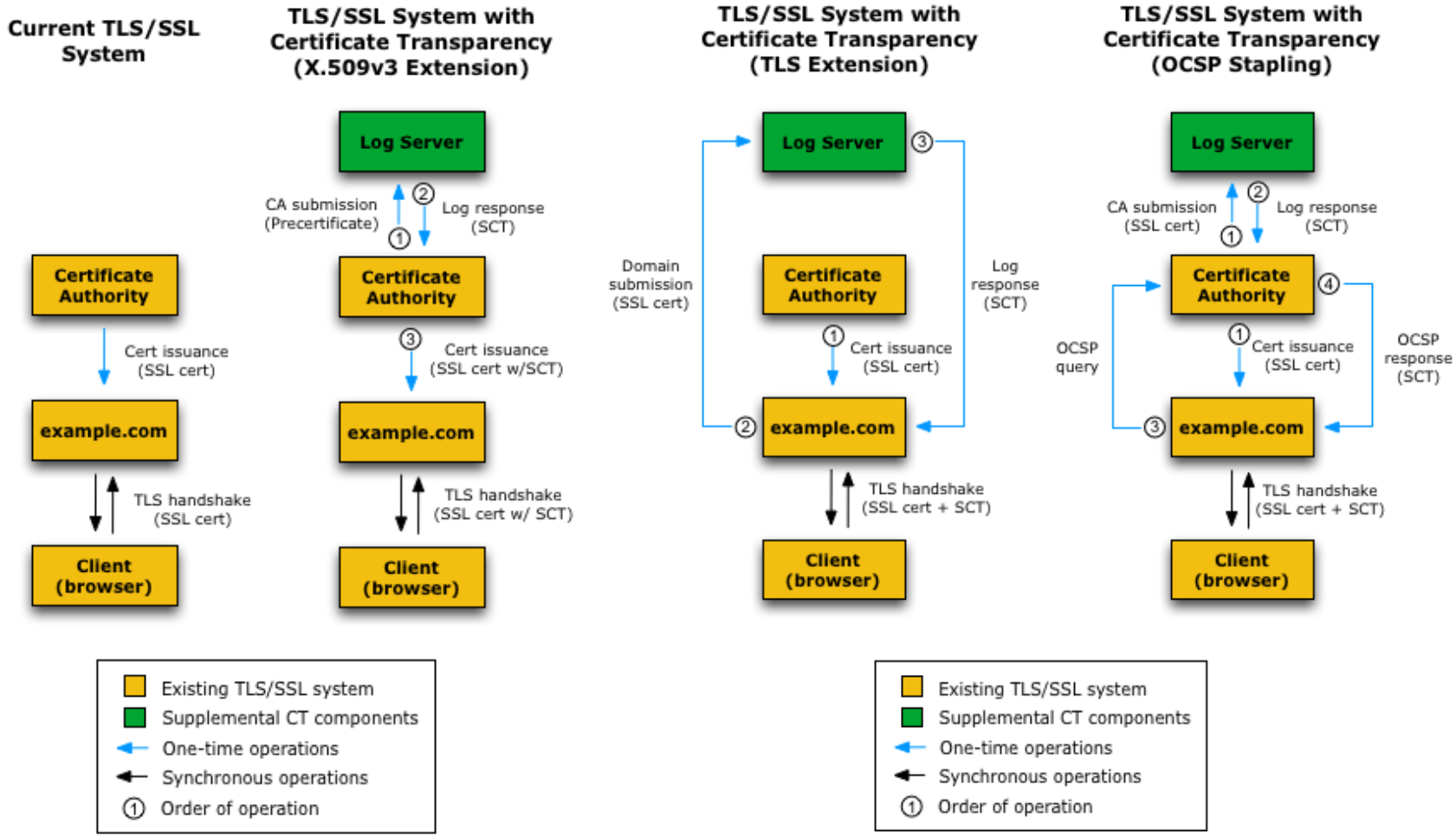
- Compare PK in certificate chain with known PK hash
  - Chrome used it to detect DigiNotar 2011 incident
- HPKP: HTTP Public Key Pinning (RFC 7469)
  - HTTP header that specifies PK hash of one certificate in website's certificate chain (and one backup key)
  - Cached by browser for max-age period
  - Dynamic → Avoids hardcoding PK hash in application
- Can be abused (HPKP Suicide, Ransom PKP)
- Deprecated in 2017 by Chrome, Firefox, Opera

```
Public-Key-Pins: max-age=2592000;
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    report-uri="https://example.net/pkp-report"
```

Solution: Certificate Transparency

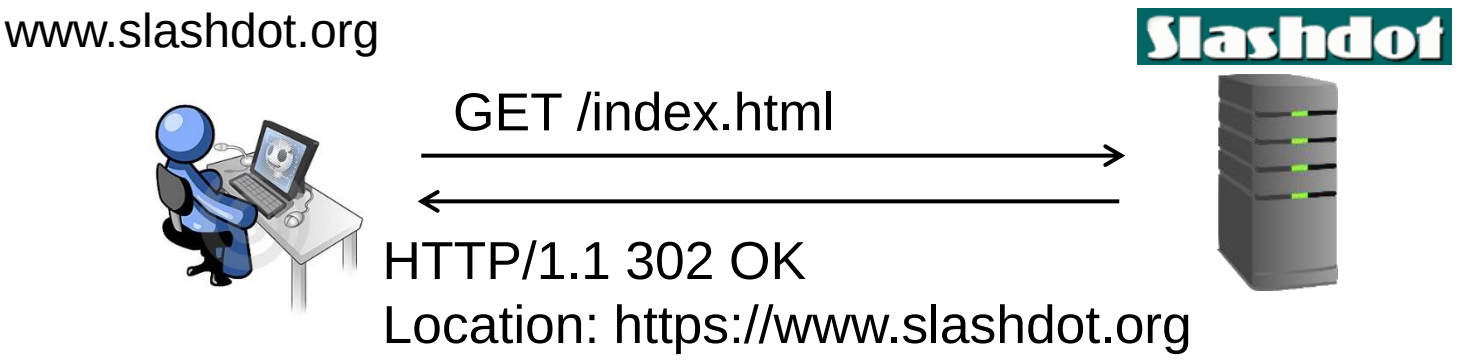
- https://www.certificate-transparency.org
- CAs advertise log of **all** certificates they issue
  - Cryptographically signed, append-only
  - Log returns signed certificate timestamp (SCT)
- Monitors scan logs for invalid certificates
- Auditors check all certificates exist in log

Certificate Transparency



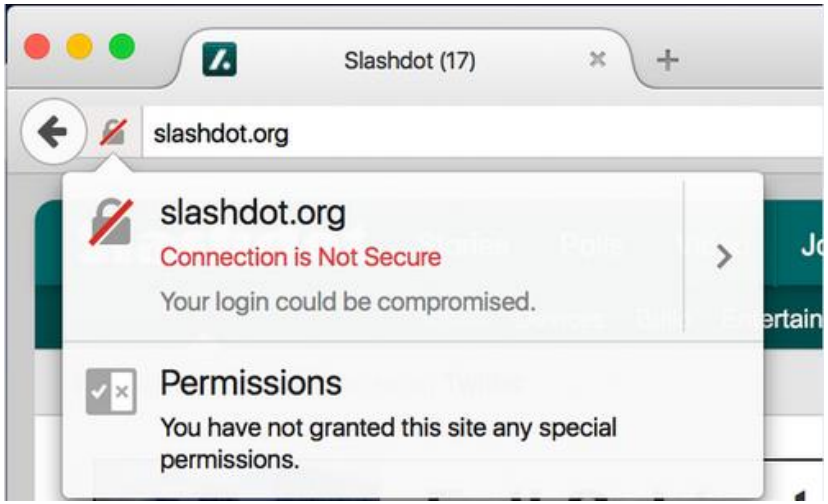
Mixed HTTPS and HTTP

- Users often type domain without https://
- User lands on http:// page
- Active network attacker can hijack session
- User should be redirected to https:// page



Mixed HTTPS and HTTP: Login

- HTTP page may host login form that submits login info to HTTPS page
- Attacker can inject code to the HTTP page and steal info before it is securely submitted
- Used to be very common even in large sites
- Browsers warn now → situation has improved



<form method="post"

action="**https**://login.slashdot.org/"

Added in Firefox 44 (Oct 2015)

HTTP in HTTPS page

- HTTPS page may contain HTTP content
  - HTTP portion can be read or modified
- Avoid: <script src="http://.../script.js">
- Use: <script src="https://.../script.js">
- Browsers now warn, e.g., Firefox icons:

All page elements transmitted over HTTPS

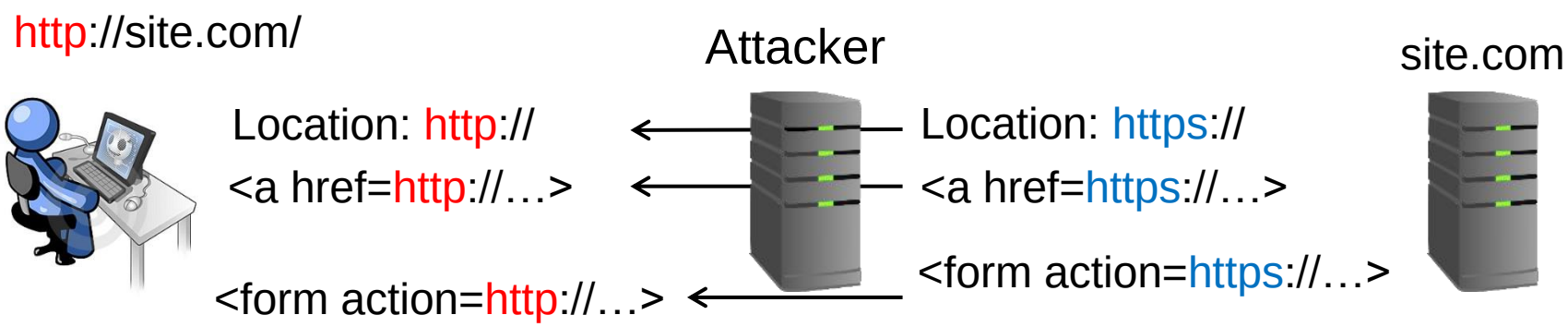
Firefox is blocking insecure **active** content, but not blocking insecure **passive** content

External images are common: <img src="http://site.com/img.jpg">

Firefox is not blocking insecure elements. Only seen if user disables mixed content blocking.

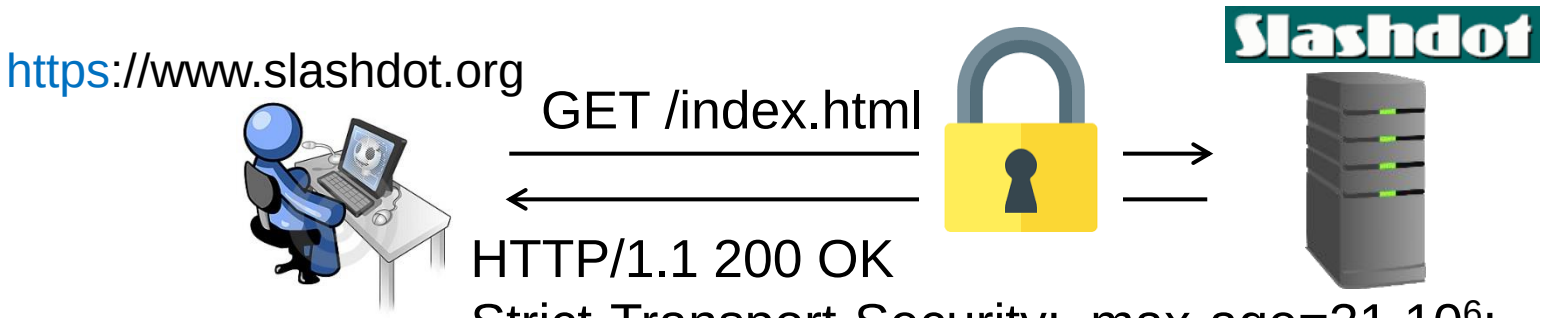
SSL Strip Attack [Moxie, 2009]

- Many sites browsed over HTTP, move to HTTPS for sensitive action, e.g., Login or Checkout
- SSL strip = MITM downgrade attack HTTPS → HTTP



Solution: Strict Transport Security

- HSTS, RFC 6797, November 2012
- HTTP Header sent by site to force browser use HTTPS
  - Must be received over HTTPS, ignored otherwise
  - Cached by browser first time received from site
  - Subsequent visits must be over HTTPS
  - Browser converts any http links to https
  - Browser refuses to connect over HTTP
- Requires **entire** site be served over HTTPS



Some Attacks on SSL/TLS

- RFC 7457 summarizes Attacks on TLS
- Beast (September 2011)
  - Attack on CBC ciphers in TLS 1.0
- Crime / Breach (September 2012)
  - Attacks on TLS compression
- Lucky Thirteen (February 2013)
  - Timing attack on MAC checks
- Poodle (October 2014)
  - Downgrade attack on SSL 3.0
- Freak (March 2015)
  - Downgrade attack on TLS to export-grade ciphers
- Logjam (May 2015)
  - Downgrade attack on TLS using Diffie-Hellman ciphersuites
- Drown (March 2016)
  - Attack on SSLv2, still in use by over 5M Web Servers

Conclusion: Secure HTTPS

- CA assumptions
  - No CA got their root key compromised
  - No employee of any CA can issue a bogus certificate
  - All CAs thoroughly check that only owner of domain can obtain certificate
- Crypto assumptions
  - All crypto algorithms are secure: ciphers, hash
- Browser assumptions
  - No bogus trusted certificate installed in browser
  - Browser has no remotely exploitable vulnerabilities
- User assumptions
  - User checks for lock icon
  - User cannot be tricked into bogus URL

- If one assumption fails, HTTPS not secure!

institute IMdea software

www.software.imdea.org

Questions?