

Manolis (Emmanouil Vasilomanolakis)

network security: TLS & Certificates

Course plan

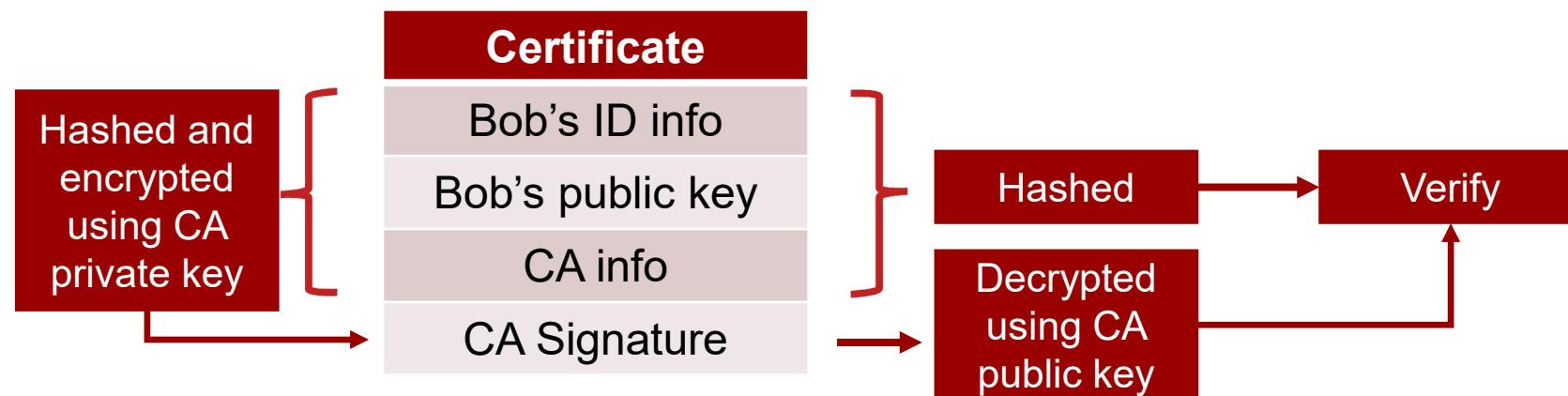
- Lecture 1: Intro (**Manolis, Carsten**) 30.01
- Lecture 2: **Crypto essentials** (**Carsten**) 06.02
- Lecture 3: **Authentication** (**Manolis**), lab bootcamp (TAs) 13.02
- Lecture 4: **TLS** (**Manolis**) 20.02
- Lecture 5: **Threat detection** (**Manolis**) 27.02
- Lecture 6: Hacking Lab day (**TAs**) – blue team 05.03
- Lecture 7: **IoT security** (**Manolis**) 12.03
- Lecture 8: **WIFI security** (**Manolis**) 19.03
- Lecture 9: **Private communication** (**Carsten**) 02.04
- Lecture 10: **When everything fails** (**Manolis**) 09.04
- Lecture 11: Hacking Lab day (**TAs**) – red team 16.04
- Lecture 12: Guest lecture (OT security, **Ludwig**) 23.04
- Lecture 13: **Exam preps** (**Carsten, Manolis**) 30.04

Outline

- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercises**

Certificates

- A **certificate** is a document that links someone's identity to their public key
 - Certificates are issued and signed by trusted **Certification Authorities** (CA)
 - A CA is a third party that **the community trusts** (e.g. government agency)
 - **X.509** is the universally accepted standardised format for certificates

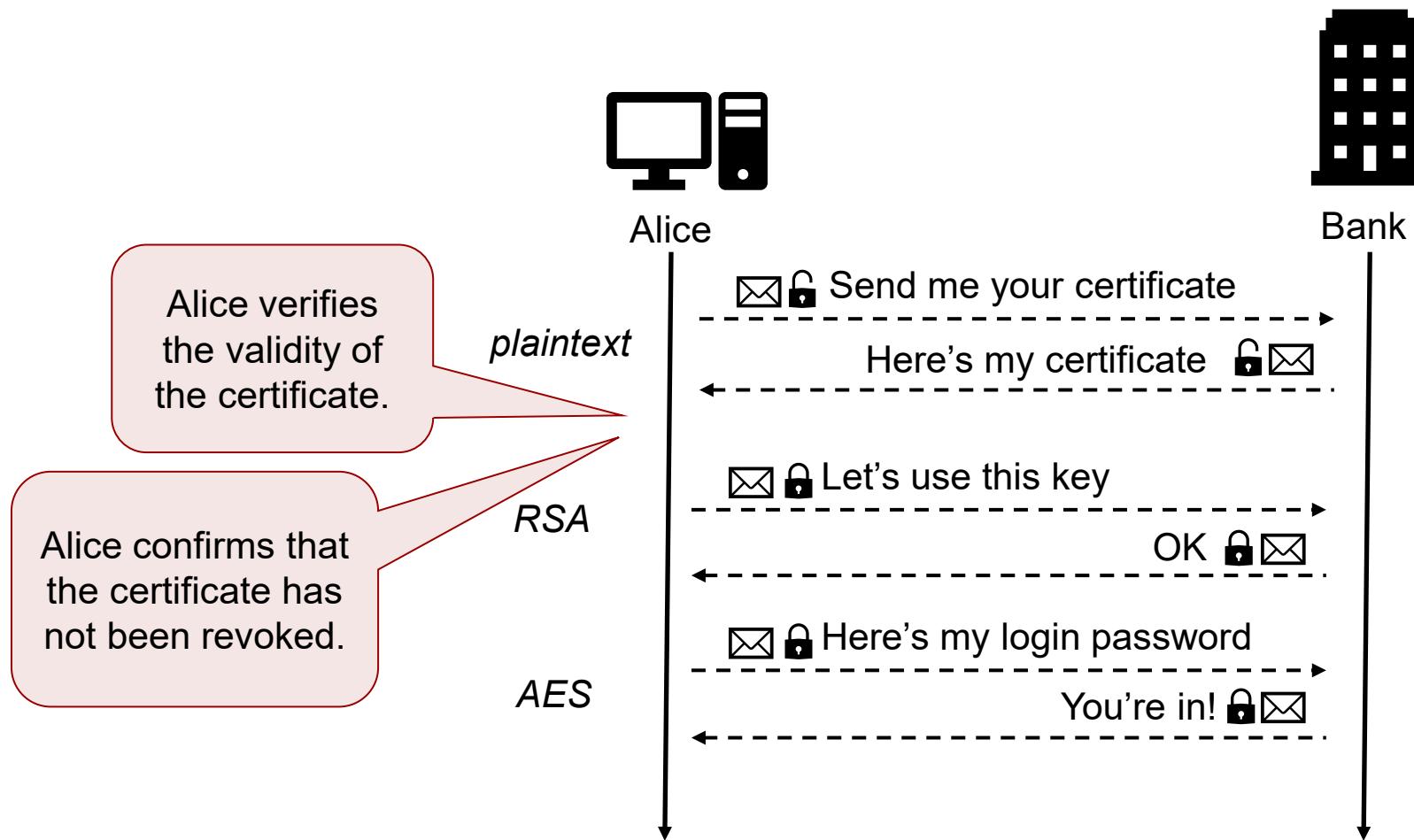


- Knowing the **public key of the CA** is sufficient to verify all the certificates that it issues
- No party except for the CA can **modify** the certificate without being detected

Certificate Lifetime

- Certificates have a **period of validity** (like credit cards)
 - “Not before” and “not after” fields
 - If expired then the certificate is invalid
 - A new certificate is issued with new public key
- Revocation of Certificates
 - Revocation of non-expired certificates may be needed if...
 - Private key gets compromised
 - Vital user info changes
 - The CA gets compromised
 - A CA keeps a **Certificate Revocation List (CRL)**
 - Signed with the CA’s private key

Public-Key Certificates



Certificates

- A piece of information that proves the identity of a public-key's owner.
 - Signed and delivered securely by a trusted third party entity called a Certificate Authority (CA)
- A Certificate contains
 - CA's identity
 - Owner's identity
 - Owner's **public-key**
 - Certificate expiry date
 - CA's signature of that certificate
 - ...
- With a certificate instead of a public key
 - A recipient can verify a few things about the issuer

x.509 certificates

- A standard for defining the format of a public key certificate
- The structure of an X.509 v3 digital certificate is as follows:
 - Certificate
 - Version Number
 - Serial Number
 - Signature Algorithm ID
 - Issuer Name
 - Validity period
 - Not Before
 - Not After
 - Subject name
 - Subject Public Key Info
 - Public Key Algorithm
 - Subject Public Key
 - Issuer Unique Identifier (optional)
 - Subject Unique Identifier (optional)
 - Extensions (optional)
 - ...
 - Certificate Signature Algorithm
 - Certificate Signature

Certificate Viewer: *.wikipedia.org

General Details

Issued To

Common Name (CN)	*.wikipedia.org
Organization (O)	Wikimedia Foundation, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	DigiCert TLS Hybrid ECC SHA384 2020 CA1
Organization (O)	DigiCert Inc
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Thursday, October 27, 2022 at 2:00:00 AM
Expires On	Saturday, November 18, 2023 at 12:59:59 AM

Fingerprints

SHA-256 Fingerprint	95 A6 25 3C F5 BA 9E 9C 79 C9 E1 66 74 AE 68 DA 28 99 75 43 93 FF 3F AA 5C 4B D5 10 B3 8D 95 A7
SHA-1 Fingerprint	91 D4 DD DD 2F F9 18 E0 19 07 D8 6B C7 54 54 F1 1A 8F 2C DC

Certificate Viewer: *.wikipedia.org

General Details

This certificate has been verified for the following uses:

- SSL Client Certificate
- SSL Server Certificate

Issued To

Common Name (CN)	*.wikipedia.org
Organisation (O)	Wikimedia Foundation, Inc.
Organisational Unit (OU)	

Serial Number 16:40:C5:D4:5D:2E:C4:D9:4C:7D:7C:6A

Issued By

Common Name (CN)	GlobalSign Organization Validation CA - SHA256 - G2
Organisation (O)	GlobalSign nv-sa
Organisational Unit (OU)	

Period of Validity

Begins On	9 November 2018
Expires On	22 November 2019

Fingerprints

SHA-256 Fingerprint	8D:CB:FD:60:E9:6C:79:CF:F0:5C:7F:17:52:CF:2B:25: 9D:88:41:F9:4A:22:1D:2D:89:09:D6:98:0E:E6:0F
SHA1 Fingerprint	06:DE:14:B2:A9:22:EF:92:F6:6B:80:81:14:72:60:23:F8:43:81:99

[Close](#)

What is this website?

- How do you know?

The screenshot shows the homepage of the DTU website. At the top, there is a navigation bar with the DTU logo, a search bar, and links for "Uddannelse", "Efteruddannelse", "Forskning", "Innovation", "Samarbejde", "Om DTU", and "Nyheder". Below the navigation bar is a large banner image featuring several young people in a workshop or laboratory setting, looking at something off-camera. Overlaid on this image is the text "ÅBENT HUS" in large white letters. Below the banner, there is a call-to-action button labeled "SE PROGRAM FOR ÅBENT HUS". Further down the page, there is a section titled "Hvad vil du læse?" with four categories: "DIPLOMINGENIØR", "BACHELOR", "KANDIDAT", and "PH.D.". Each category has a list of links below it.

Danmarks Tekniske Universitet

ÅBENT HUS

Kom til Åbent Hus den 2. marts og få svar på dine spørgsmål om DTU's uddannelser og studiemiljø.

SE PROGRAM FOR ÅBENT HUS

Hvad vil du læse?

DIPLOMINGENIØR

- Uddannelsesretninger
- Om uddannelsen

BACHELOR

- Uddannelsesretninger
- Om uddannelsen

KANDIDAT

- Uddannelsesretninger
- Om uddannelsen

PH.D.

- Ledige ph.d.-stillinger
- Om uddannelsen

DTU Course

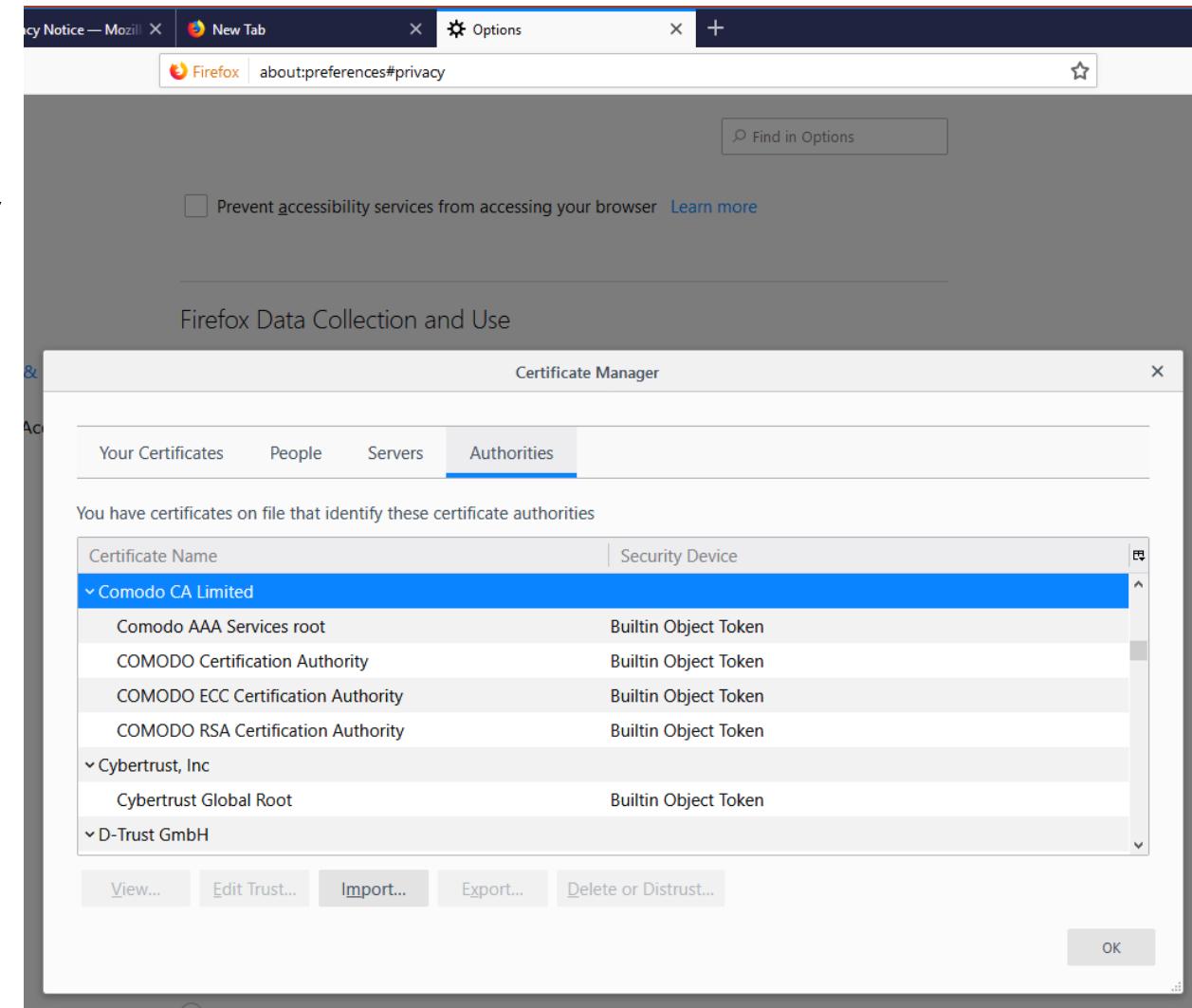
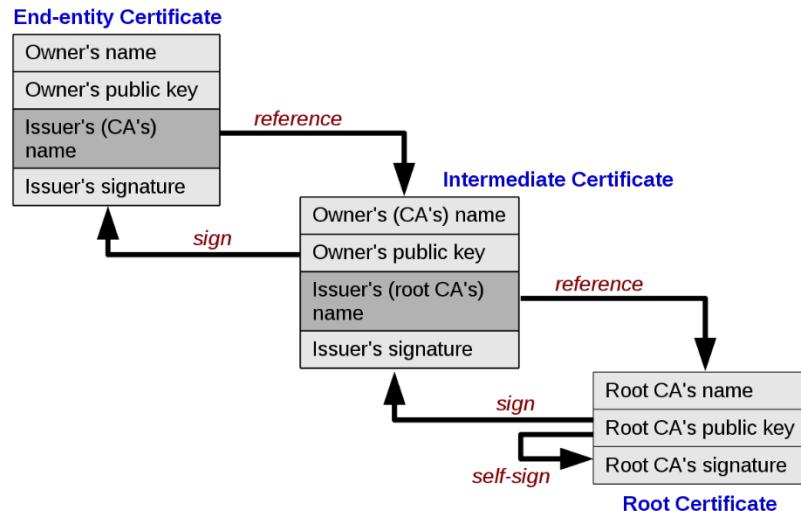
20 February 2024

course

9

Root certificates

- Root/trusted certificates are scary



Dtu.dk certificate

Certificate

[www.dtu.dk](#)[GEANT OV RSA CA 4](#)

USERTrust RSA Certification Authority

Subject Name

Country NL
Organization GEANT Vereniging
Common Name GEANT OV RSA CA 4

Subject Name

Country DK
State/Province Hovedstaden
Organization Danmarks Tekniske Universitet
Common Name www.dtu.dk

Issuer Name

Country NL
Organization GEANT Vereniging
Common Name GEANT OV RSA CA 4

Validity

Not Before Thu, 17 Nov 2022 00:00:00 GMT
Not After Fri, 17 Nov 2023 23:59:59 GMT

Issuer Name

Country US
State/Province New Jersey
Locality Jersey City
Organization The USERTRUST Network
Common Name [USERTrust RSA Certification Authority](#)

Subject Alt Names

DNS Name www.dtu.dk
DNS Name ait-psc05cd01.win.dtu.dk
DNS Name ait-psc05cd02.win.dtu.dk
DNS Name ait-psc05cd03.win.dtu.dk
DNS Name ait-psc05cd04.win.dtu.dk
DNS Name ait-psc05cd05.win.dtu.dk
DNS Name ait-psc05cd11.win.dtu.dk
DNS Name www.adgangskursus.dtu.dk
DNS Name www.alumni.dtu.dk
DNS Name www.aqua.dtu.dk
DNS Name www.bibliotek.dtu.dk
DNS Name www.bioengineering.dtu.dk

[View...](#)[Edit Trust...](#)[Import...](#)[Export...](#)[Delete or Distrust...](#)[OK](#)

Certificates and expiration...

- Expiring certificates are a pain
- It is very tempting to create **certificates** that will **last forever**
- **BAD** practice!



An example of bad certificate practice

- We have been running an experiment:
 - Scan the whole Internet (IPv4 address space)
 - Focus on an industrial automation protocol: OPC UA (Unified Architecture)
 - Port 4840
 - Commonly used for device-to-device communication



Screenshot of the Pro-face GP SERIES HMI software interface. The top navigation bar includes links for Japanese, English, Remote maintenance, Equipment Information, Viewer, Device View, Alarm Information, File Transfer, Log-Off, and Device View (highlighted with a green circle). The main content area shows configuration fields for a device named '#MEMLINK' with access points '#INTERNAL' and '#ME-MLINK'. It also shows a dropdown for Data Type set to '16 bit signed' and a 'Start Monitoring' button. Below this is a table with several empty rows.

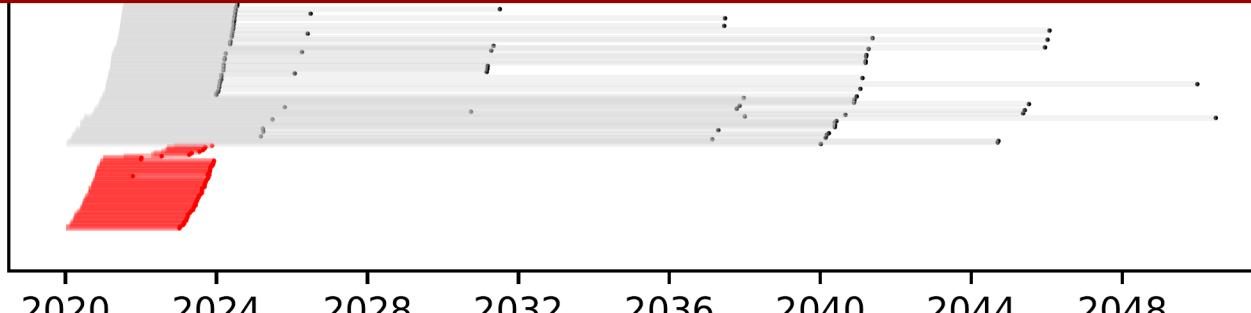
Certificates and expiration...

OPC UA certificate validity

Expired
Valid

And the **winner** is:

<X509Name object '/C=AT/ST=Burgenland/L=Eisenstadt/O=ETM professional control/OU=Development/CN=WinCC OA OPC UA Server'> <X509Name object '/C=AT/ST=Burgenland/L=Eisenstadt/O=ETM professional control/OU=Development/CN=WinCC OA OPC UA Server'> ##### ###### **10950 days**



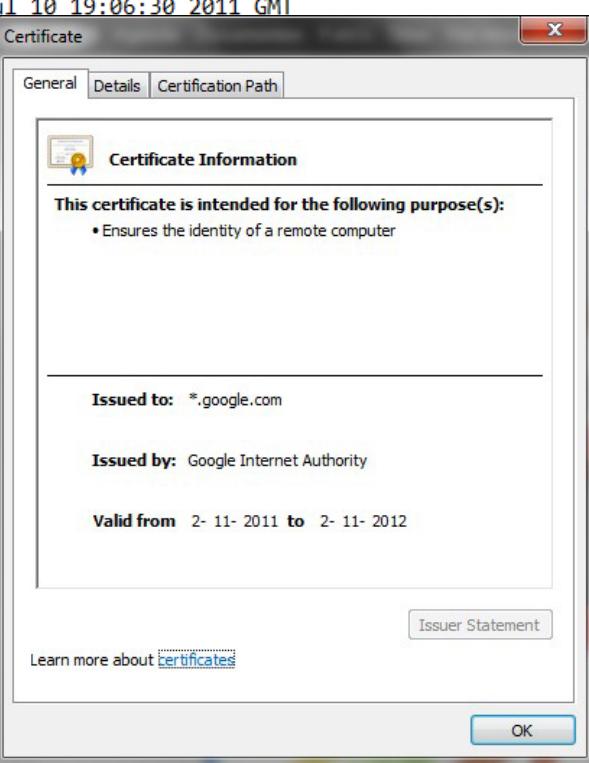
CA attacks

- There is no cryptography to protect us from a CA compromise
- Only option: stringent auditing and scrupulous OpSec
- Not a theoretical threat
- Nightmare scenario: CA compromise that remains not publicly known.
- Example: DigiNotar
 - Summer 2011: Hacker gains access to DigiNotar's network
 - July 10: 531 rogue certificates are issued
 - July 19: DigiNotar detects the breach, but doesn't go public
 - End of August: Google notices a man-in-the-middle attack in Iran with one of the rogue certificates. Trust in DigiNotar certificates is revoked by all major browsers
 - Further info: [podcast "Darknet Diaries", Episode 3](#)

DigiNotar attack (2011)

- Wildcard certificate for Google
 - Mitm attacks in Iran
 - Searches and email communication
 - Claims that people even died because of this
- Detected from Chrome users
 - Public key pinning: Google hard-coded the fingerprints for its own sites' encryption keys into Chrome, and told the browser to ignore contrary information from certificate authorities

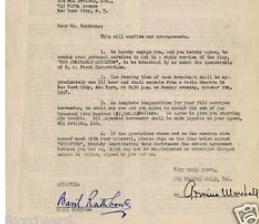
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
05:e2:e6:a4:cd:09:ea:54:d6:65:b0:75:fe:22:a2:56
Signature Algorithm: sha1WithRSAEncryption
Issuer:
emailAddress = info@diginotar.nl
commonName = DigiNotar Public CA 2025
organizationName = DigiNotar
countryName = NL
Validity
Not Before: Jul 10 19:06:30 2011 GMT
Not After : Jul 10 19:06:30 2012 GMT
Subject:
commonName
serialNumber
localityName
organizationName
countryName
Subject Public Key
Public Key Alg
RSA Public Key
Modulus (2048 bits)
00:cd:
24:3f:
0d:ec:
79:e2:
9e:90:
cc:d1:
88:b6:
6e:f7:
2d:1f:
a1:28:
26:ec:
aa:64:
2a:79:
9f:2d:
f7:c8:a7:70:89:43:9a:b8:d8:ce:5a:29:3d:c3:0f:
93:de:57:37:f8:ad:f2:4a:40:d8:02:4d:68:88:05:
cf:57:71:61:14:ba:cc:f0:02:c9:e6:83:b7:b6:10:
94:5d
Exponent: 65537 (0x10001)
X509v3 extensions:
Authority Information Access:
OCSP - URI:<http://validation.diginotar.nl>



An attack on a CA

- Comodo 2016 Optical Character Recognition (OCR) attack

1. Register "altelekom.at"



5. Here is your TLS certificate for "a1telekom.at"!

That's Telekom Austria :\$
one of the largest telecom providers in Austria



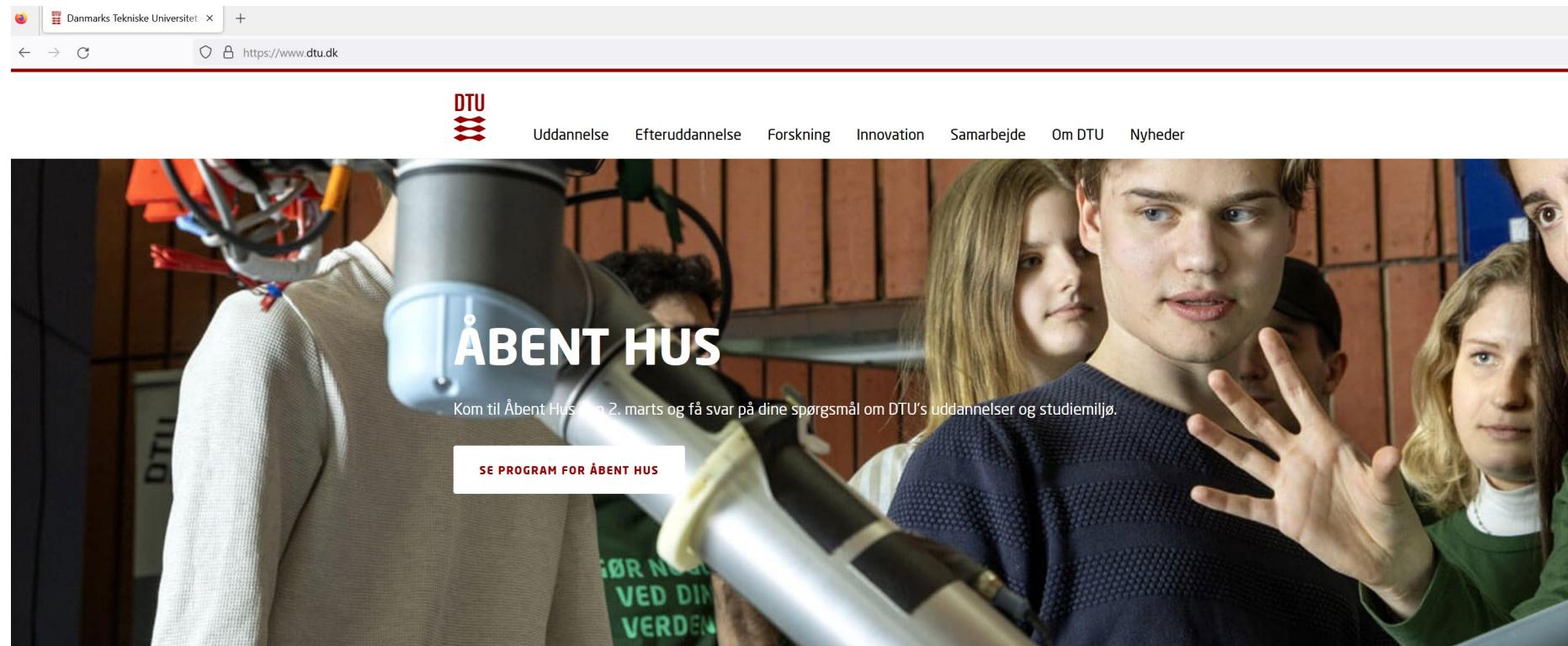
3. Comodo uses OCR to read received documents



4. Shit happens 😊

OCR reads "a1telekom.at"

And why do we need the s in https?



The screenshot shows the official website of Danmarks Tekniske Universitet (DTU) at <https://www.dtu.dk>. The page features a large banner image of students in a laboratory setting. Overlaid on the banner is the text "ÅBENT HUS" and a button labeled "SE PROGRAM FOR ÅBENT HUS". The DTU logo is visible in the top left corner of the page. The navigation menu includes links for Uddannelse, Efteruddannelse, Forskning, Innovation, Samarbejde, Om DTU, and Nyheder.

ÅBENT HUS

Kom til Åbent Hus den 2. marts og få svar på dine spørgsmål om DTU's uddannelser og studiemiljø.

SE PROGRAM FOR ÅBENT HUS

Hvad vil du læse?

DIPLOMINGENIØR	BACHELOR	KANDIDAT	PH.D.
→ Uddannelsesretninger → Om uddannelsen	→ Uddannelsesretninger → Om uddannelsen	→ Uddannelsesretninger → Om uddannelsen	→ Ledige ph.d.-stillinger → Om uddannelsen

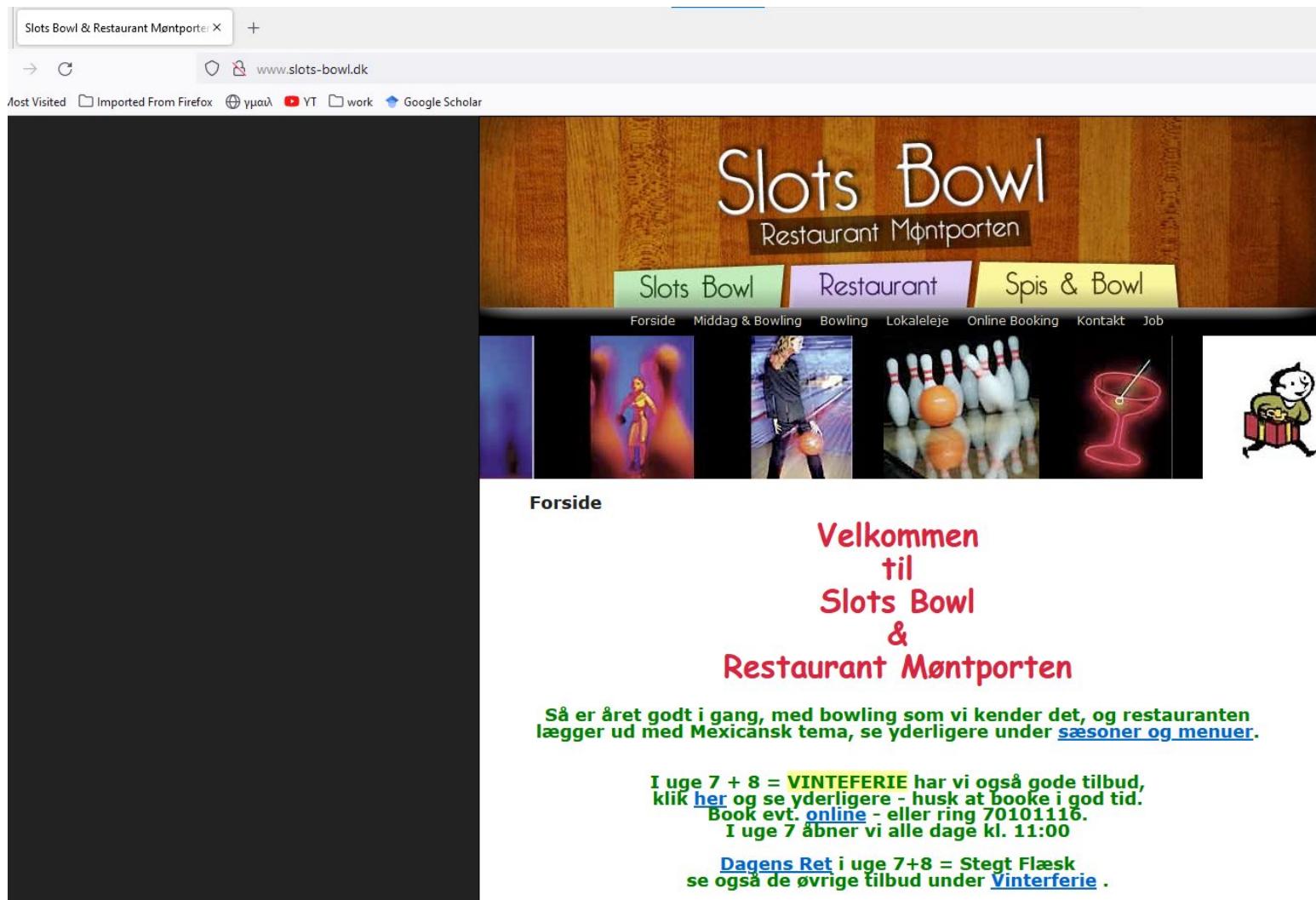
Why is it always bowling websites?

Video: [anatomy of an IoT attack](#)

Among others the hacker exploited a (very old) bowling website

Let's go back in time

- Meet Slots bowl
- Bowling place in Hillerød
- Website is a time machine
 - 90s aesthetics
 - But also, no TLS



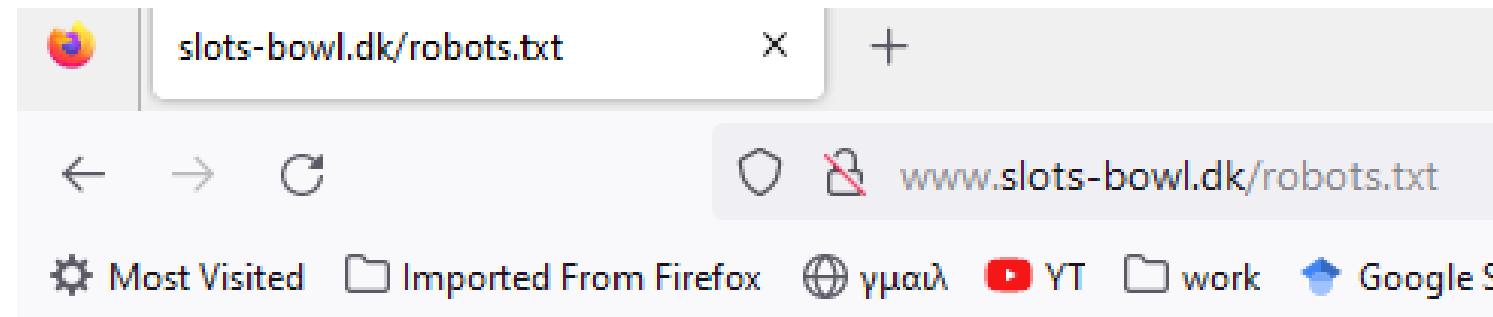
Let's go back in time

- Booking is done via TLS (that's good)
- BUT how about the rest of the website?



Let's go back in time (don't try this at home)

- Booking is done via TLS (that's good)
- BUT how about the rest of the website?



Let's go back in time (don't try this at home)

- Booking is done via TLS (that's good)
- BUT how about the rest of the website?



Let's go back in time (don't try this at home)

The screenshot shows a NetworkMiner tool interface with an 'http' tab selected. The main pane displays a list of network traffic entries:

No.	Time	Source	Destination	Protocol	Length	Info
29	23.556450	192.168.0.6	185.93.195.155	HTTP	704	POST /admin/login.php HTTP/1.1 (application/x-www-form-urlencoded)
32	23.710144	185.93.195.155	192.168.0.6	HTTP/X...	1385	HTTP/1.1 200 OK
34	23.803670	192.168.0.6	185.93.195.155	HTTP	459	GET /admin/loginstyle.php HTTP/1.1
38	23.833640	185.93.195.155	192.168.0.6	HTTP	427	HTTP/1.1 200 OK (text/css)

The details pane shows the raw request and response data. The request is a POST to /admin/login.php with the following headers:

```
Host: www.slots-bowl.dk\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8Accept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nContent-Type: application/x-www-form-urlencoded\r\nContent-Length: 49\r\nOrigin: http://www.slots-bowl.dk\r\nDNT: 1\r\nConnection: keep-alive\r\nReferer: http://www.slots-bowl.dk/admin/login.php\r\nCookie: CMSSESSID486eed53=7fc99d227461627c4ed0406b292733f1\r\nUpgrade-Insecure-Requests: 1\r\nSec-GPC: 1\r\n\r\n[Full request URI: http://www.slots-bowl.dk/admin/login.php]
```

The response is a 200 OK with the following headers:

```
refox/11.0.0 Accpt: tex t/html,appli cati on/xhtml+xml,app lication/xml;q=0.9,image/avif,image/webp,*/*;q=0.8..Accept-Language: en-US,en;q=0.5..Accept-Encoding: gzip,deflate..Content-Type: application/x-www-form-urlencoded..Content-Length: 49..Origin: http://www.slots-bowl.dk..DNT: 1..Connection: keep-alive..Referer: http://www.slots-bowl.dk/ad min/login.php..Cookie: CMSSESSID486eed53=7fc99d227461627c4ed0406b292733f1..Upgrade-Insecure-Requests: 1..Sec-GPC: 1..username=bowling&password=123456..loginsubmit=Send
```

The browser screenshot shows a Firefox window with the URL `http://www.slots-bowl.dk/admin/login.php`. The page displays a login form with fields for 'username' (set to 'bowling'), 'password' (set to '123456'), and 'loginsubmit' (set to 'Send').

Outline

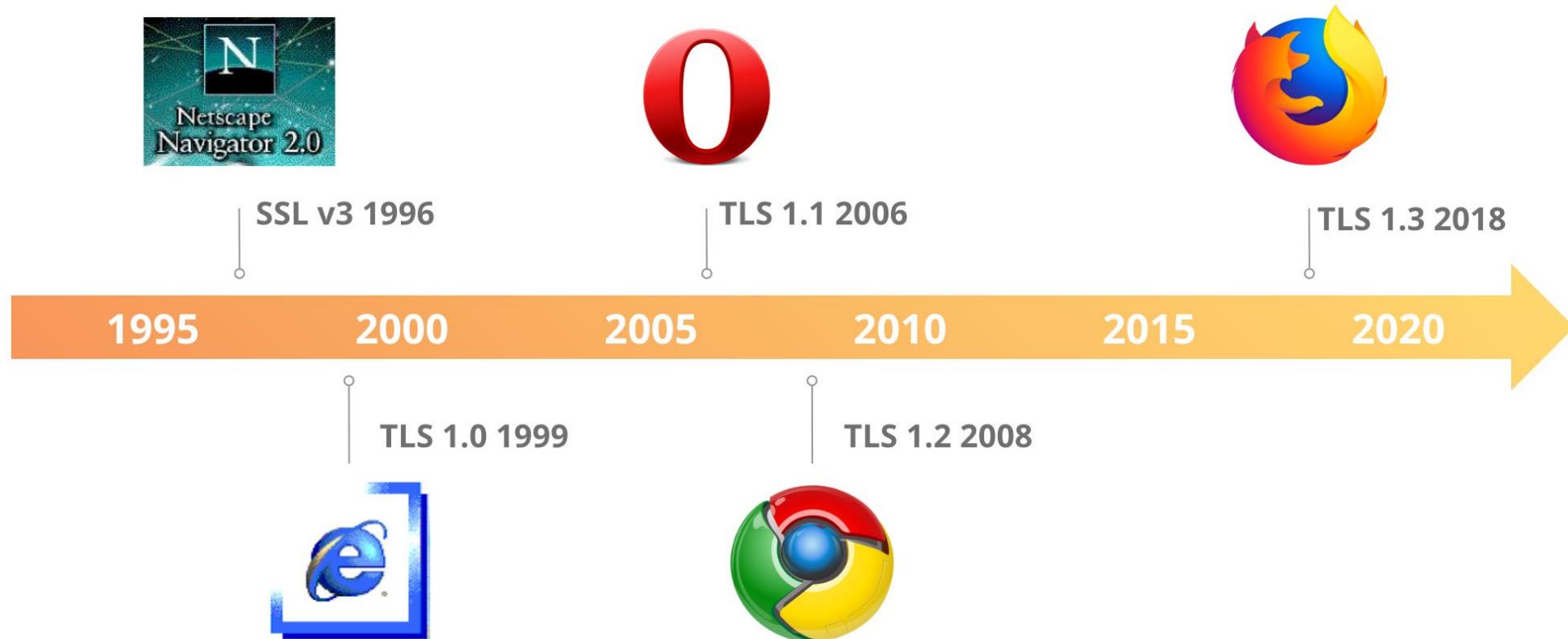
- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercise**

On the ambiguity of names

- SSL: Secure Sockets Layer
- TLS: Transport Layer Security
- TLS is a **newer version** of SSL
 - For political/historical reasons SSL is still used/mentioned

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecation planned in 2020 ^[11]
TLS 1.1	2006	Deprecation planned in 2020 ^[11]
TLS 1.2	2008	
TLS 1.3	2018	

On the ambiguity of names



What does it do?

- **Confidentiality**
 - Encryption of a session (e.g., via DH-RSA)
- **Authentication (via certificates)**
 - Is valid for the same domain as the one being accessed
 - Has been issued by a trusted CA (Certificate Authority)
 - Is valid and not passed its expiration date
- **Data integrity**
- **Forward secrecy**
 - (most of the time; depends on protocol version and settings)

What it doesn't do

- **Privacy/Anonymity**
 - Everyone knows that A communicates with B
 - They don't know the content of the communication
- It doesn't make a website "**safe**"
 - Anyone can create an SSL/TLS certificate
- It doesn't mean that the data saved on the server is encrypted too
- (also sometimes protocol ≠ implementation ;)

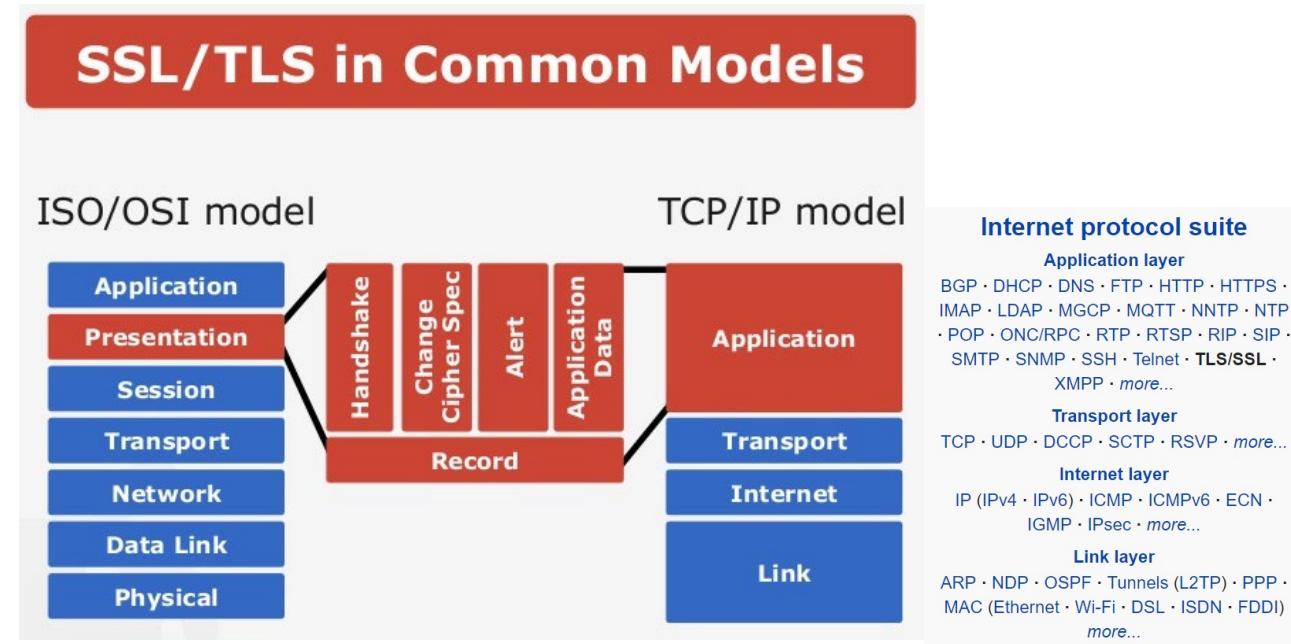


Outline

- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercise**

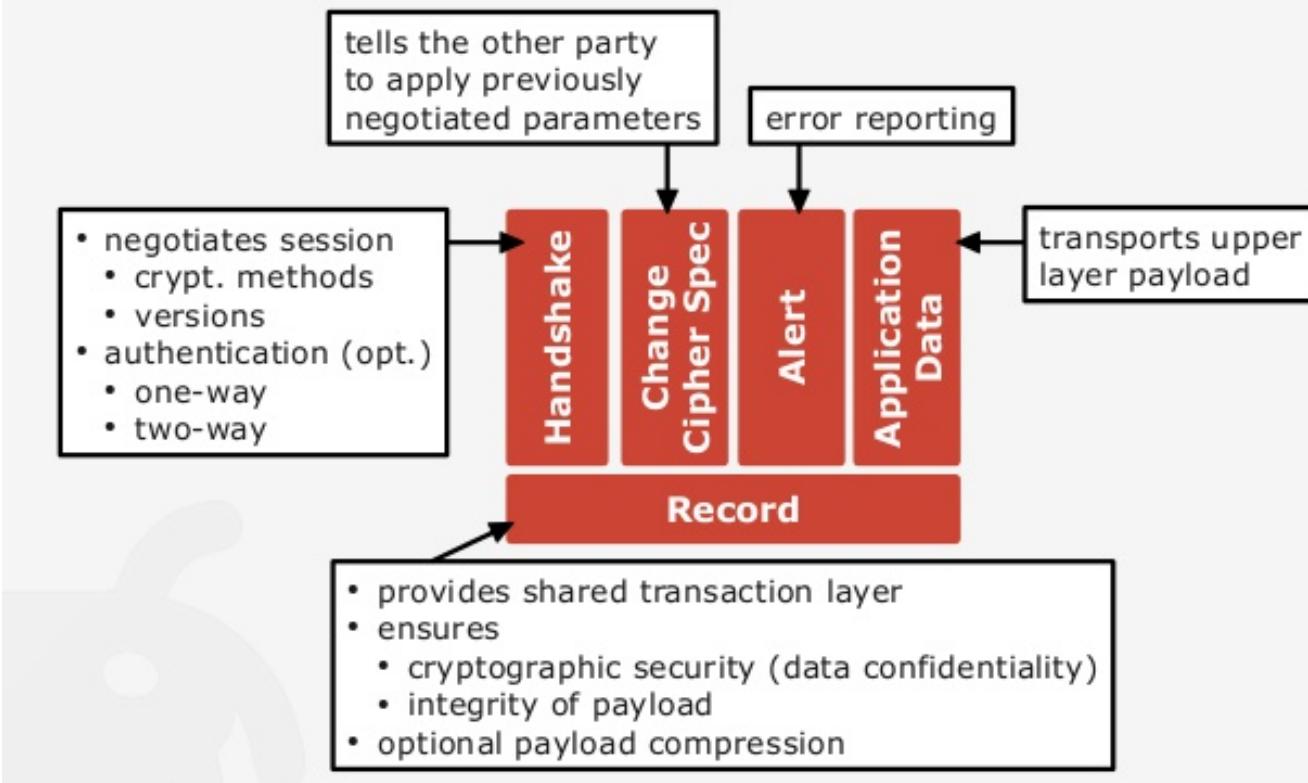
TLS at a glance

- A protocol offering various security properties



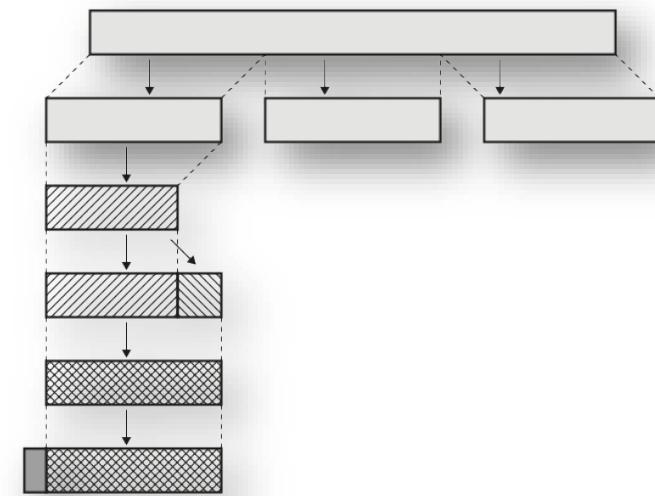
TLS at a glance

SSL/TLS Protocol Stack

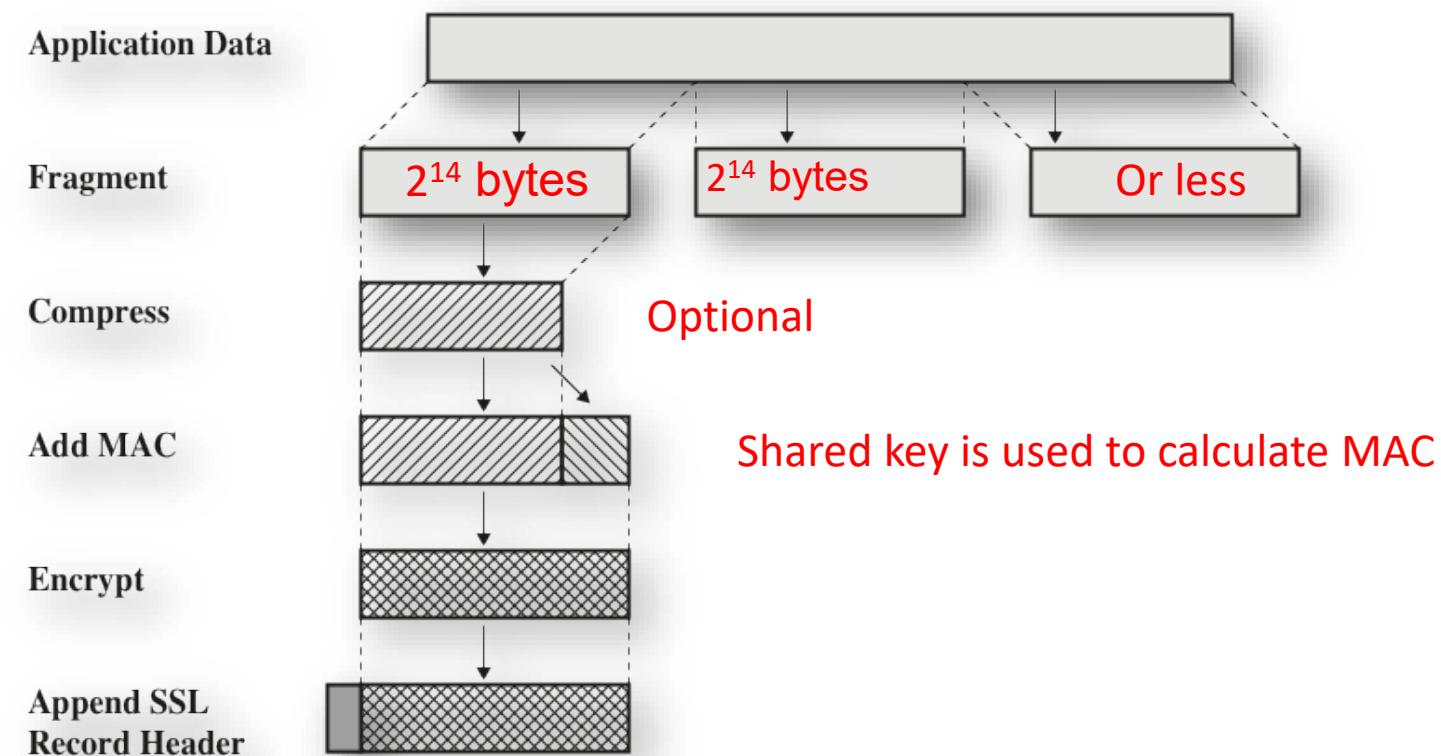


TLS Record Protocol

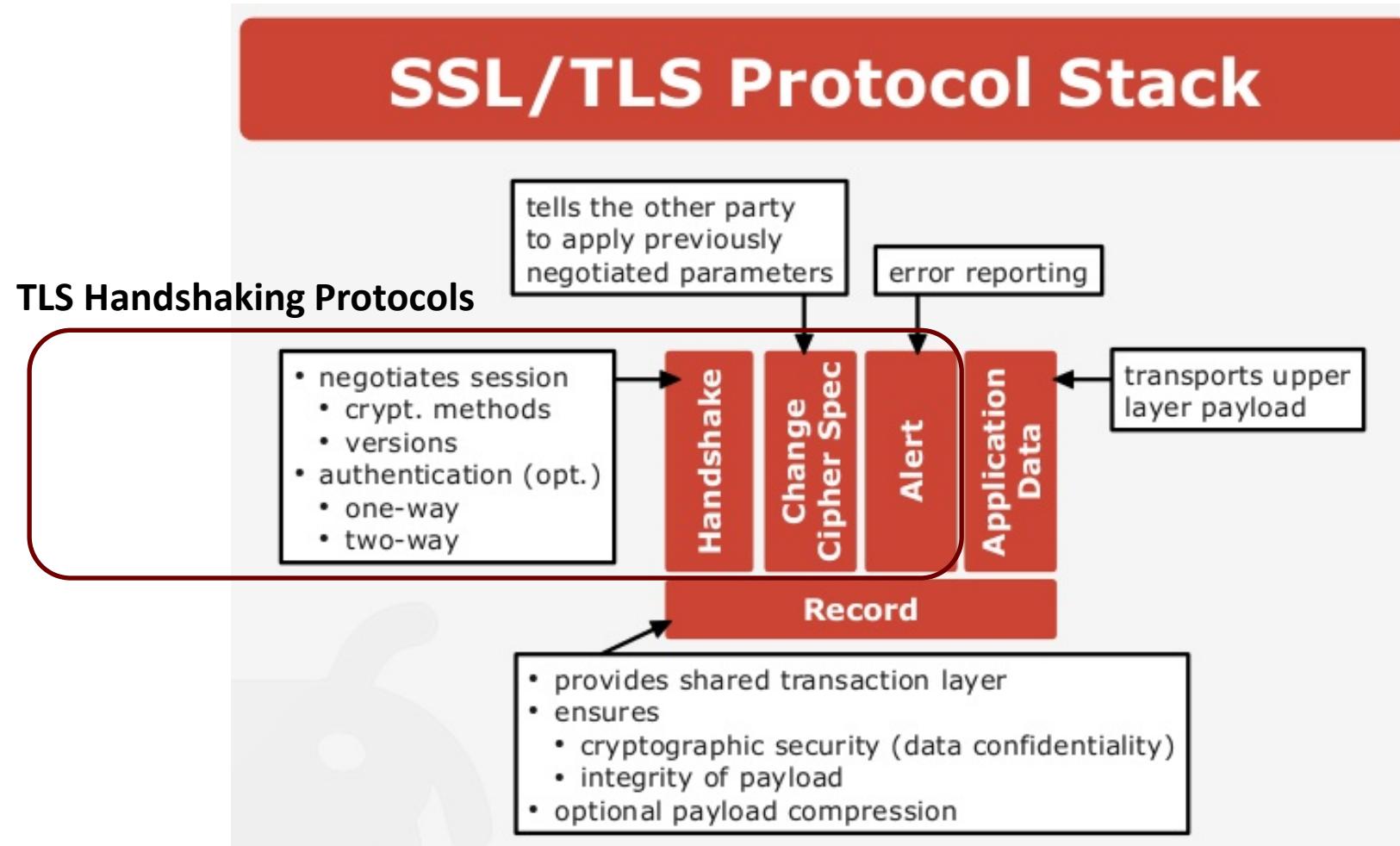
- The Record Protocol **takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result**
- (For the receiver's end: received data is verified, decrypted, reassembled, and then delivered to higher-level clients)



TLS Record Layer

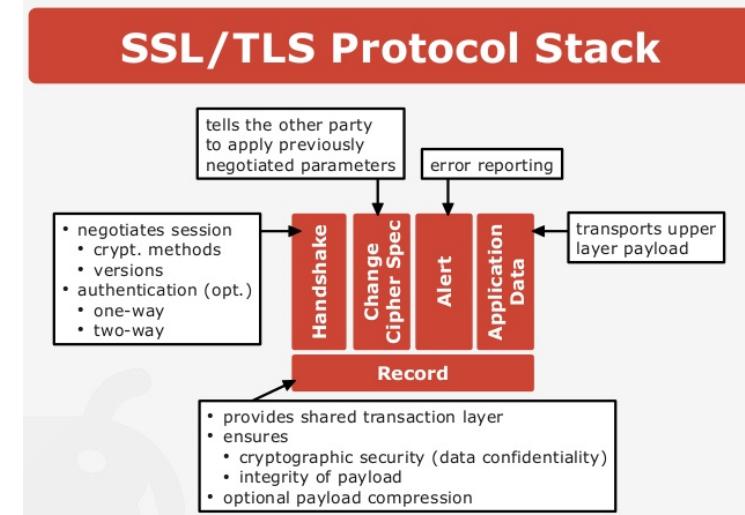


TLS at a glance



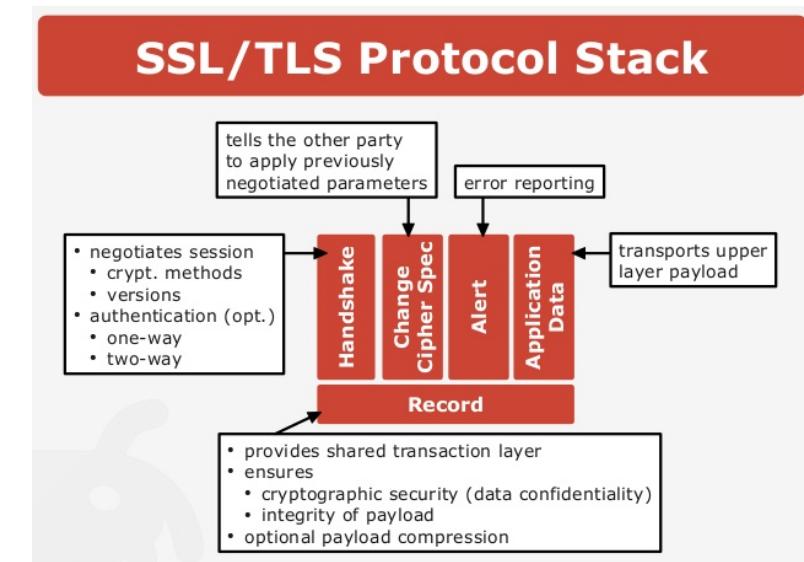
TLS Higher Layer sub-protocol

- **Change Cipher Spec**
 - Notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys
 - Only one byte long, and signals the change in communications protocol by having a value of ‘1’
 - Exists only in TLS versions <=1.2
- **Alert** –Sends errors, problems or warnings about the connection. This layer is formed with two fields:
 - **Severity Level:** value 1 (warning) or 2 (fatal error; discontinue session)
 - **Alert Description:** indicates the specific error that caused the Alert Message to be sent from a party. This field is one byte, mapped to one of twelve specific numbers (e.g., handshakeFailure, BadCertificate, CertificateRevoked, etc.)



TLS Handshake

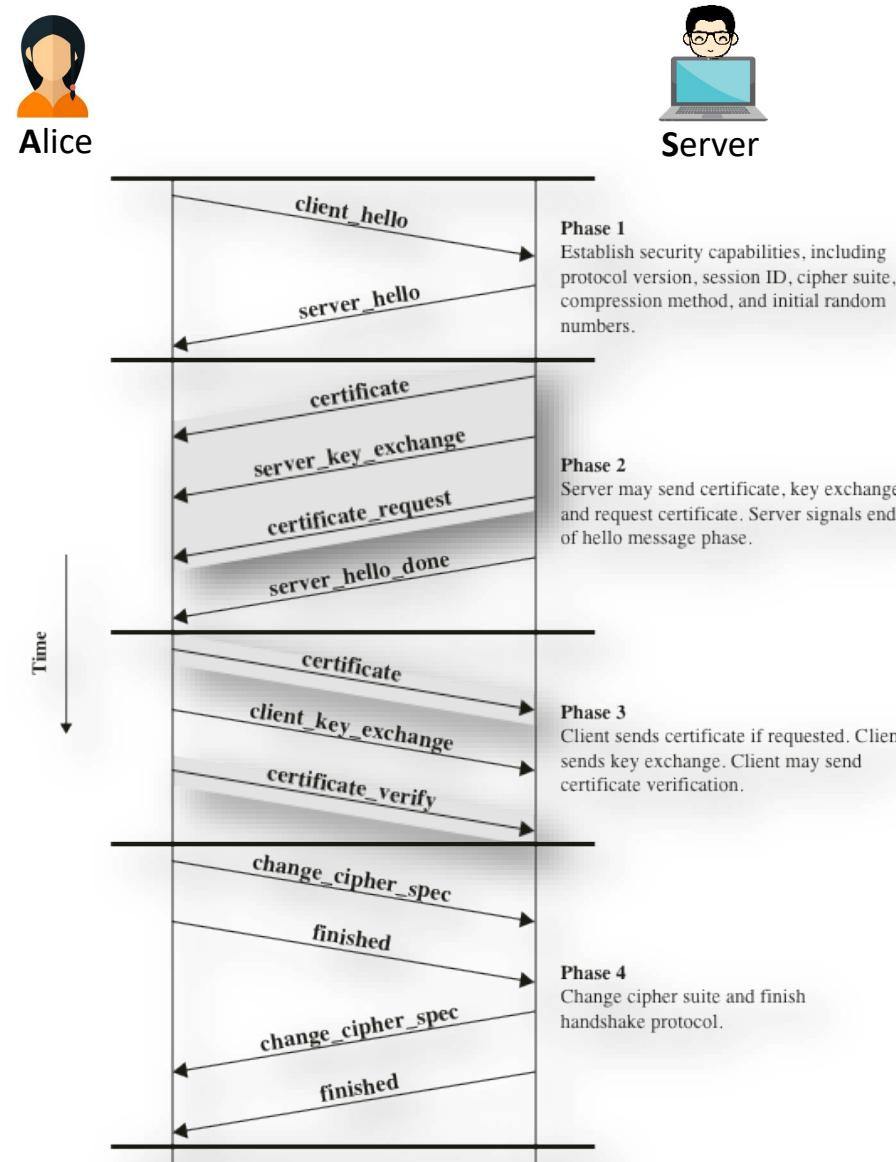
- Allows server/client to authenticate each other and negotiate an encryption and MAC algorithm and cryptographic keys
- Used before any application data is transmitted
- Each message has three fields
 - Type (1 byte) – Indicates one of the 10 messages (see next slide)
 - Length (3 bytes) – the length of the message in bytes
 - Content (> 0 bytes) – the parameters associated with this message



TLS Handshake Protocol Message Types

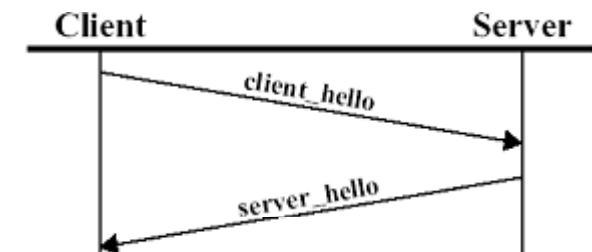
Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Handshake Protocol Action (TLS <=1.2)



Phase 1 – Establish Security Capabilities

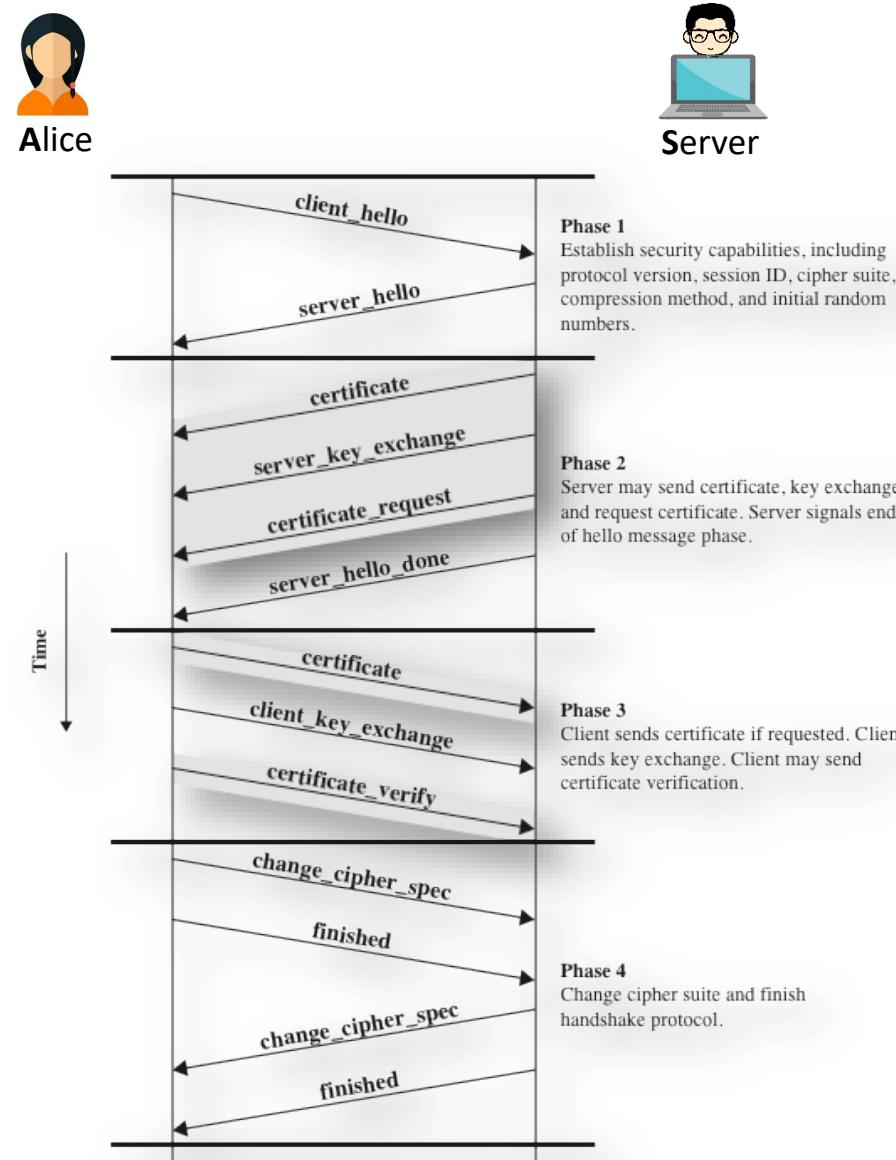
- Used to initiate a logical connection and to establish the security capabilities that will be associated with it
 - Exchange is initiated by the client with following parameters:
 - Version
 - Random
 - Session-ID
 - CipherSuite
 - Compression Method
- First element in the CipherSuite is Key exchange method
 - RSA, Fixed Diffie-Hellman, Ephemeral Diffie-Hellman, Anonymous Diffie-Hellman, Fortezza



Note on DH versions

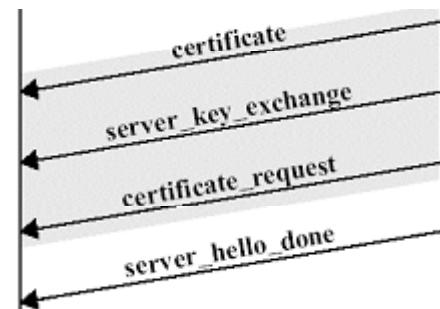
- **Anonymous Diffie-Hellman:** classic Diffie-Hellman, i.e., without authentication
 - Because the keys used in the exchange are not authenticated, the protocol is susceptible to Man-in-the-Middle attacks.
- **Fixed Diffie-Hellman** embeds the server's public parameter in the certificate, and the CA then signs the certificate. That is, the certificate contains the Diffie-Hellman public-key parameters, and those parameters never change.
- **Ephemeral Diffie-Hellman** uses temporary, public keys. Each instance or run of the protocol uses a different public key. The authenticity of the server's temporary key can be verified by checking the signature on the key.
 - Because the public keys are temporary, a compromise of the server's long-term signing key does not jeopardize the privacy of past sessions (**Perfect Forward Secrecy**)

Handshake Protocol Action (TLS <=1.2)

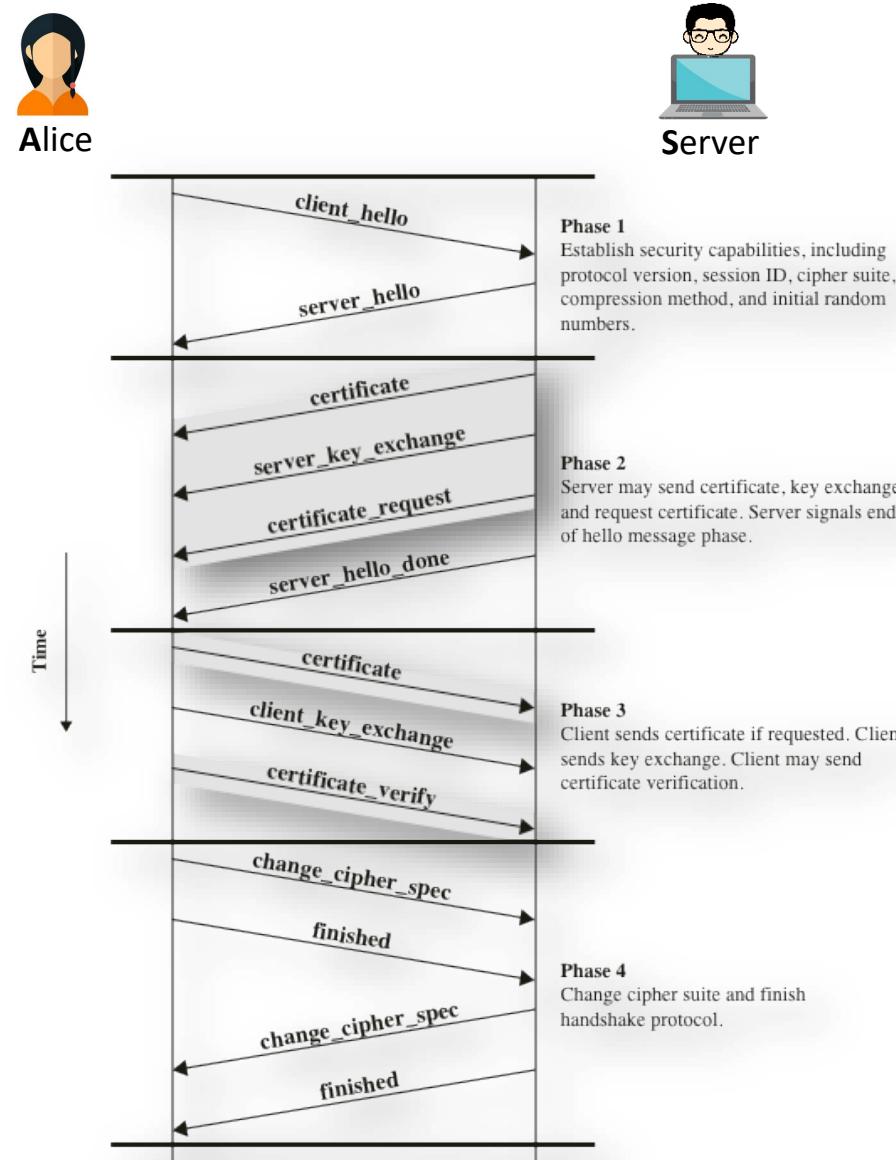


Phase 2– Server Authentication and Key Exchange

- Server begins this phase by sending its **certificate** if it needs to be authenticated
 - Message contains one chain of certificates
- Next Message “**server_key_exchange** message”
 - The message is not required if the server has sent a certificate with fixed Diffie-Hellman Parameters or RSA key exchange is to be used
- **Certificate_request** message
 - Includes two parameters – `certificate_type` and `certificateAuthorities`
- Final message “**server_done** message”
 - Sent by the server to indicate the end of the server hello and associated messages
- Server waits for a client response

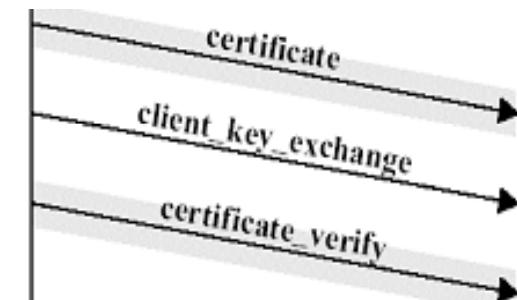


Handshake Protocol Action (TLS <=1.2)

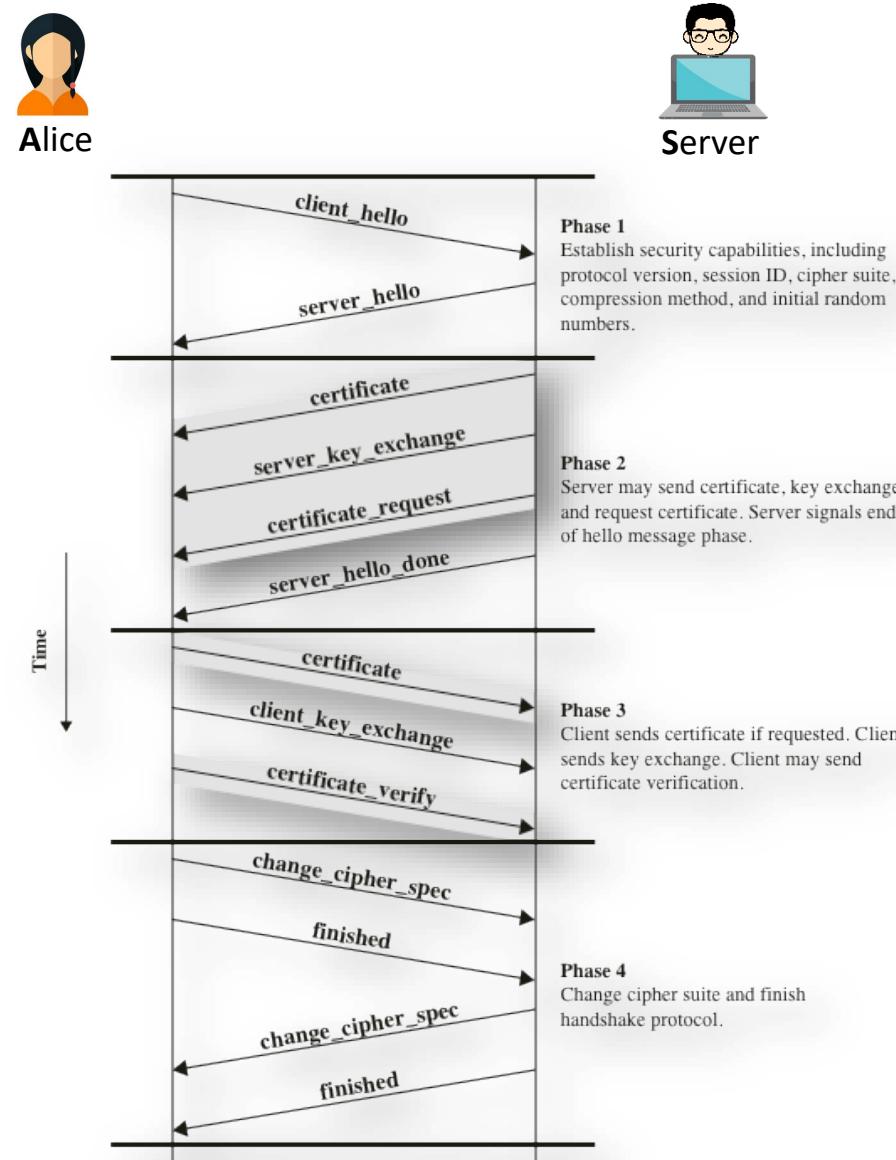


Phase 3– Client Authentication and Key Exchange

- Client verifies that the server provided a valid certificate
- Checks that the parameters are acceptable
- If requested by the server client sends its own ***certificate***
- Next message “*client_key_exchange* message”
 - Contents of message depends on the type of the key exchange
- Final message “*certificate_verify* message”
 - Provide verification of a client certificate

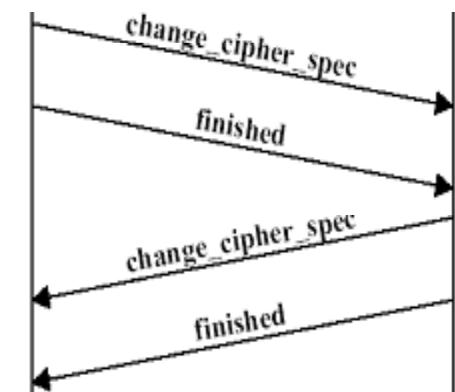


Handshake Protocol Action (TLS <=1.2)



Phase 4—Finish

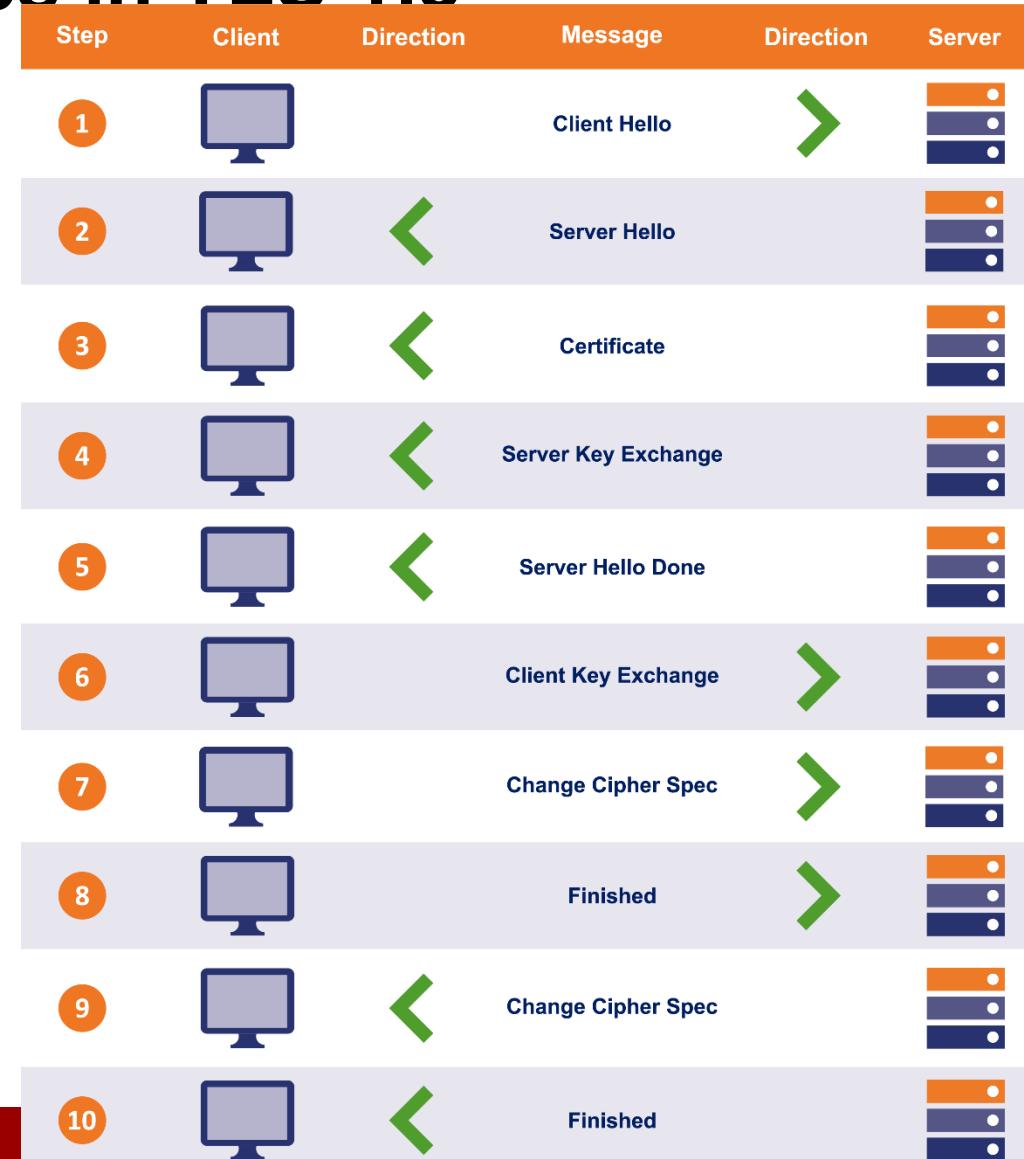
- This phase completes the set up
- Client send a `change_cipher_spec` message
 - Copies the pending CipherSpec into the current CipherSpec
- Client sends the finished message under the new algorithms, keys, and secrets
 - Verifies that the key exchange and authentication process was successful
- Similar response from the server side
 - The client and the server can begin to exchange application-layer data



Heartbeat Protocol

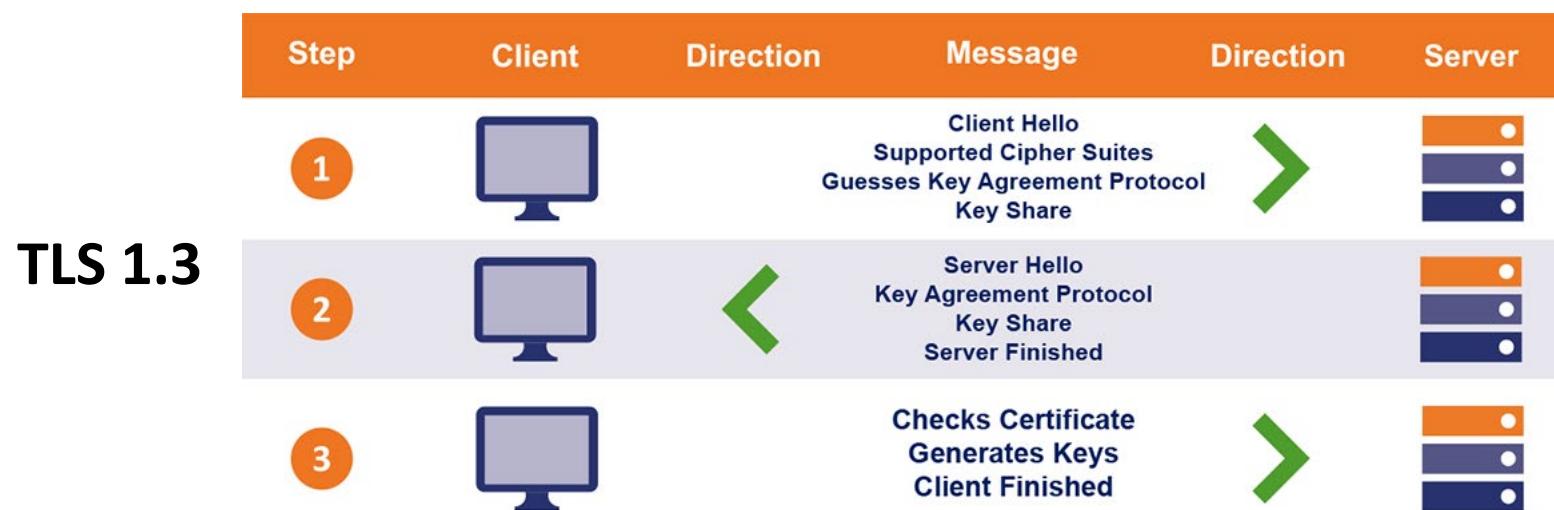
- Periodic message exchange to **keep connection alive**
 - Used in popular TLS library implementations
 - problem: no bound checks on heartbeat messages

Differences in TLS 1.3



TLS 1.2 (and earlier versions)

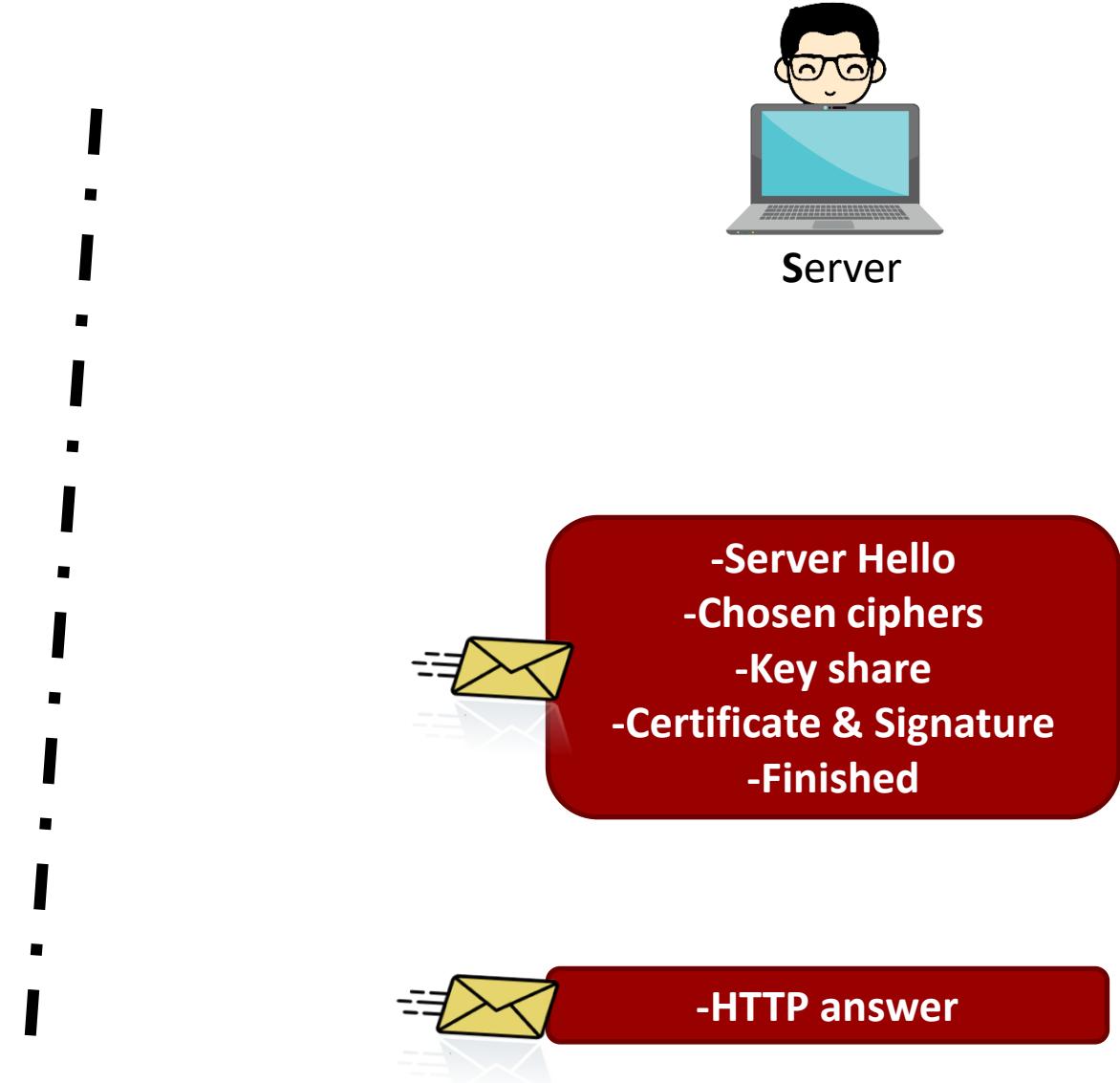
Differences in TLS 1.3





Alice

TLS 1.3 Handshake



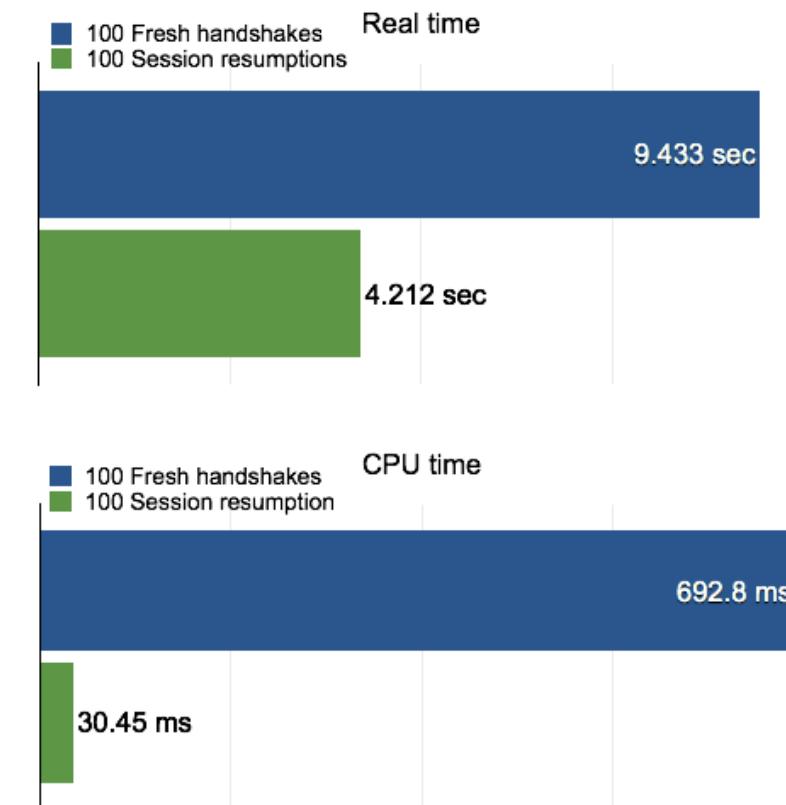
Outline

- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercise**

TLS resumption

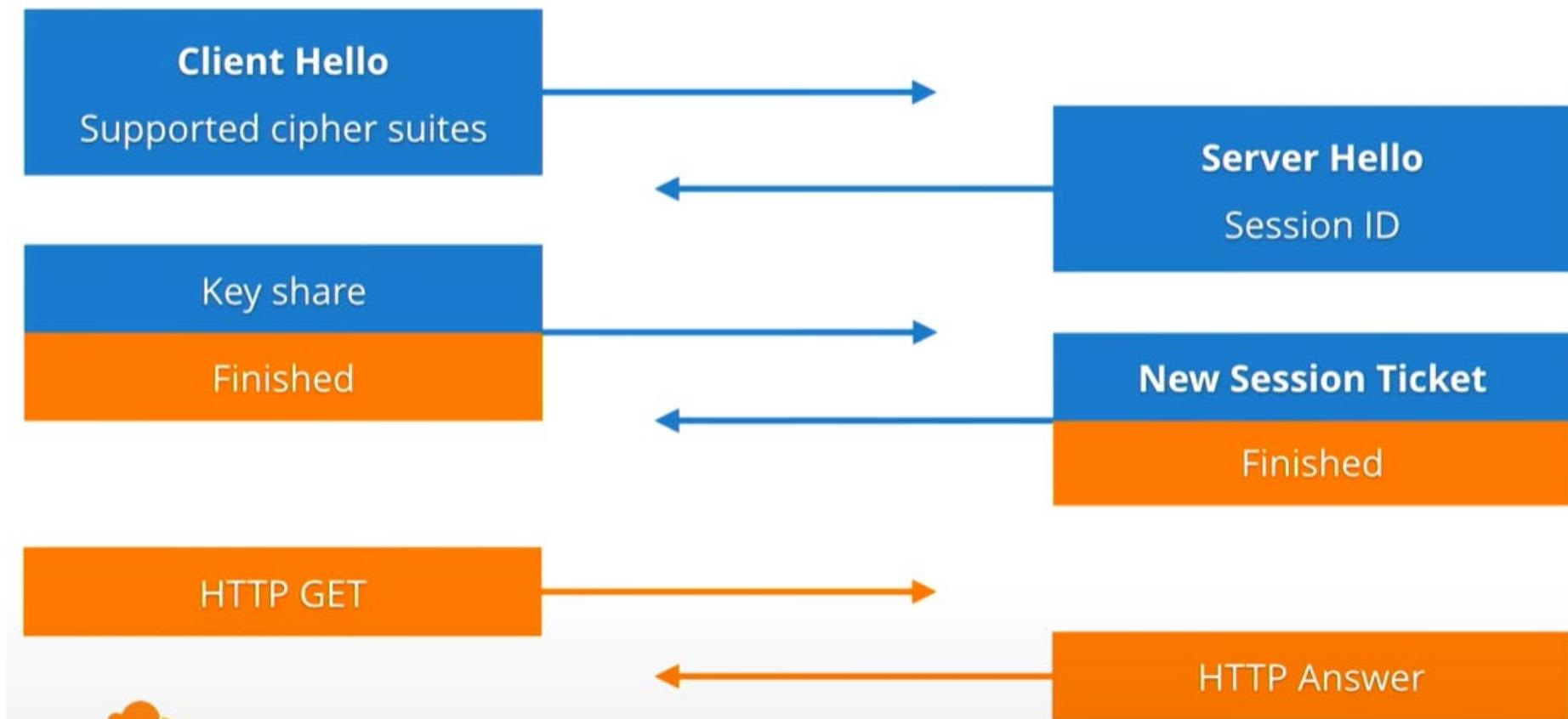
Most TLS connections are not new

- If a client has visited a server before then a TLS resumption will occur
- Much faster
- TLS 1.2 and 1.3 do things differently



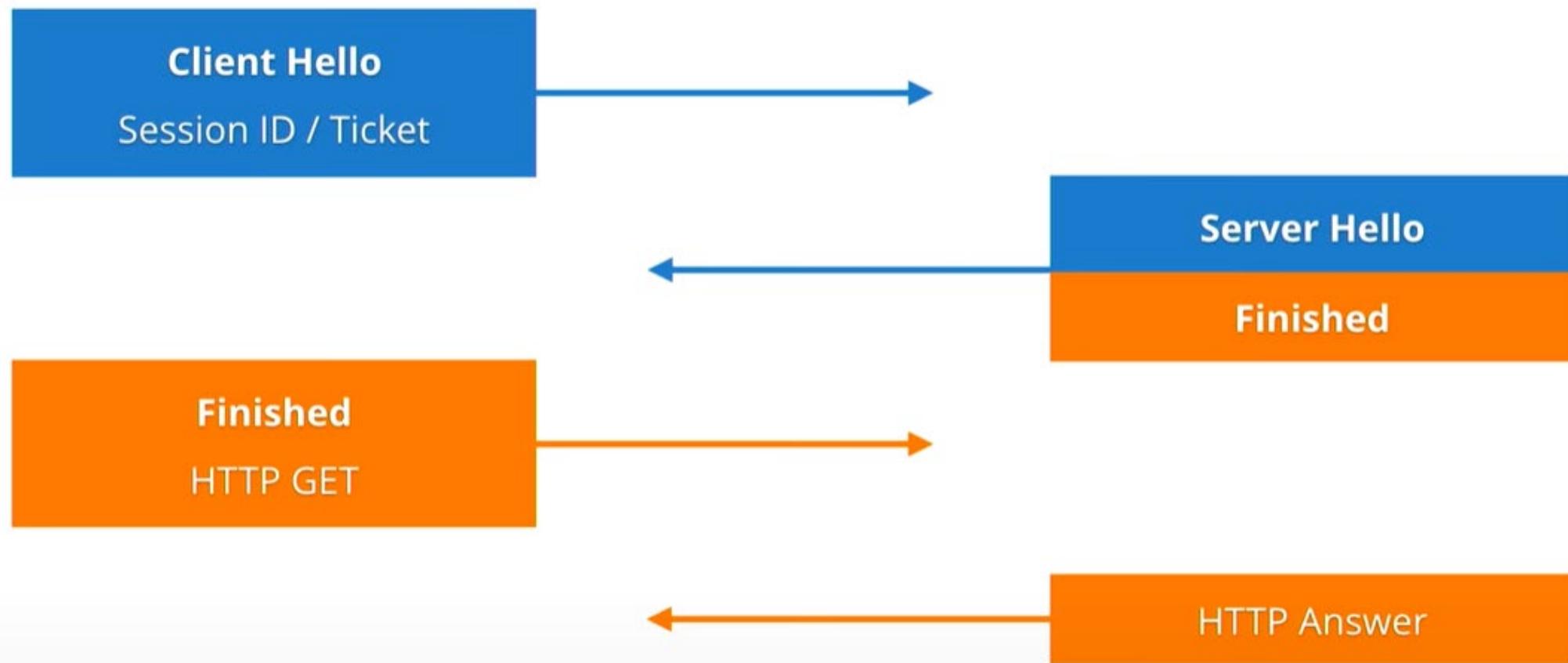
TLS 1.2 resumption

- Reminder: full TLS 1.2 (and introducing session ticket)



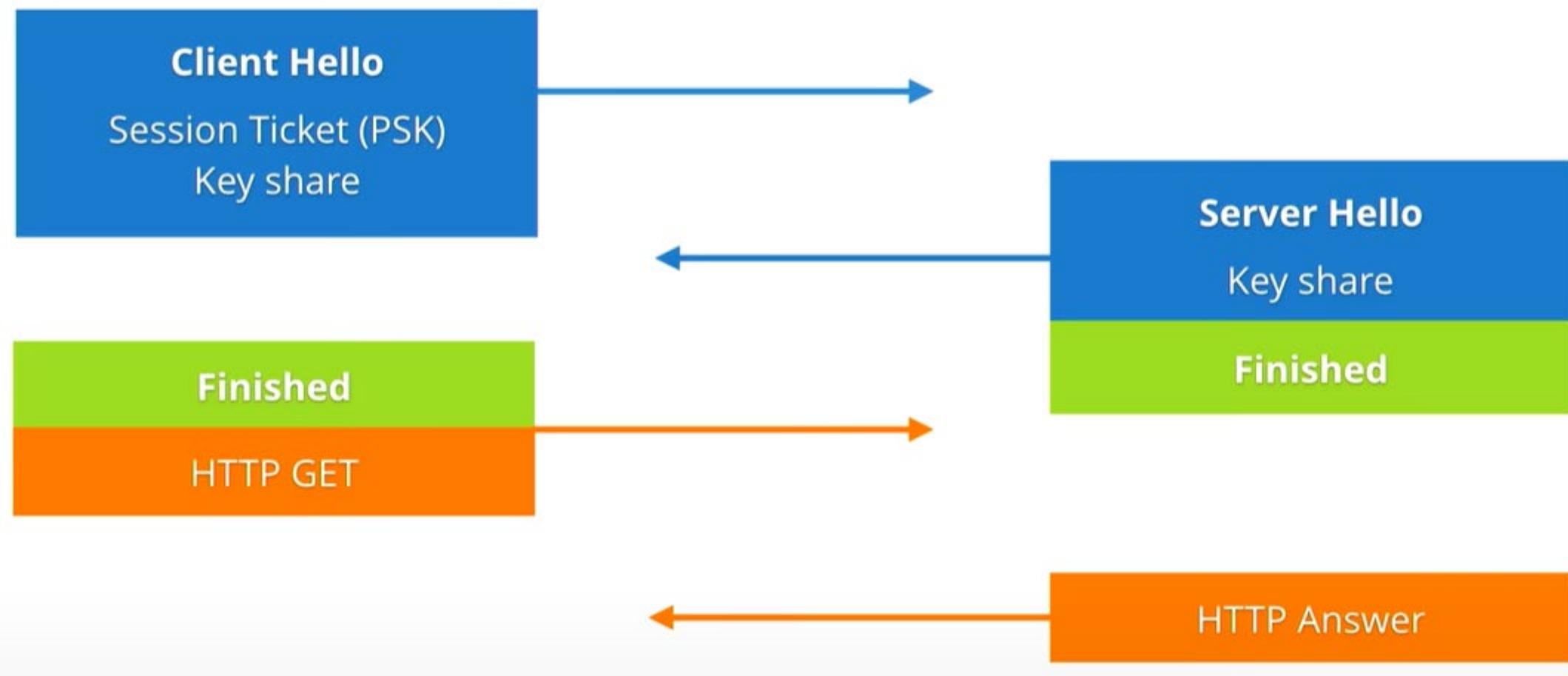
TLS 1.2 resumption

- TLS 1.2 with resumption is 1-RTT

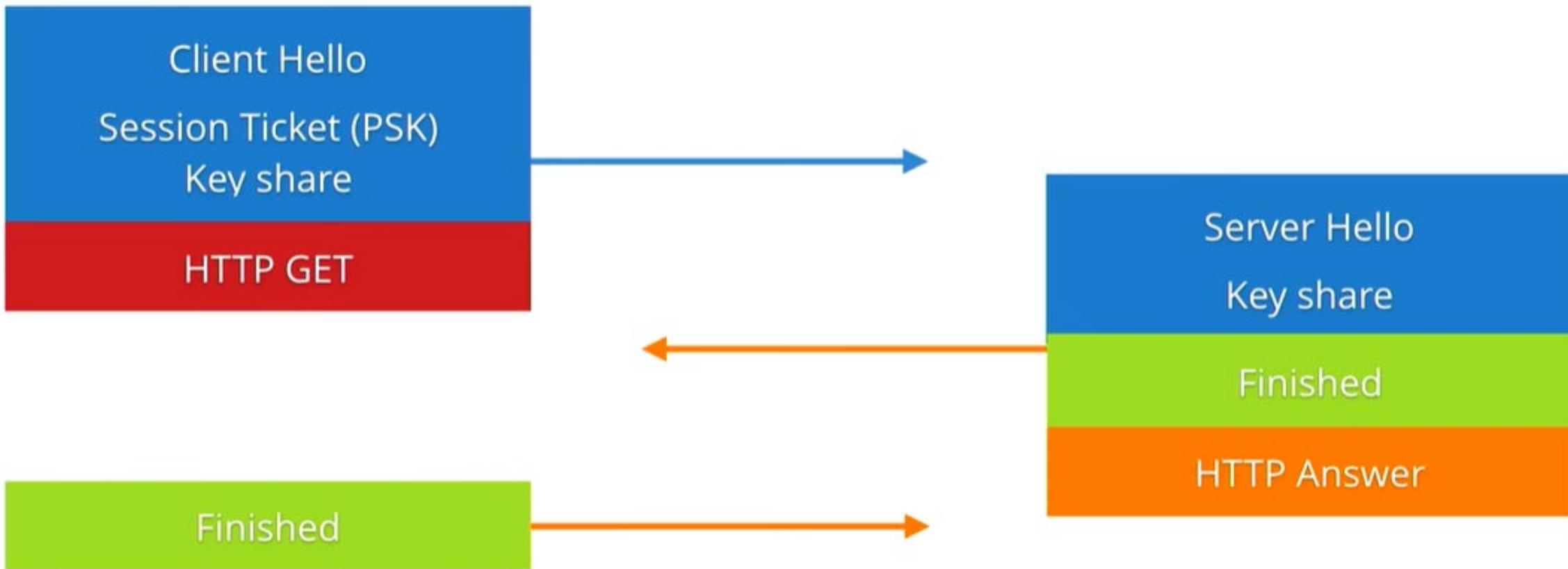


TLS 1.3 resumption 0-RTT

- Reminder of TLS 1.3 (and introducing PSK)

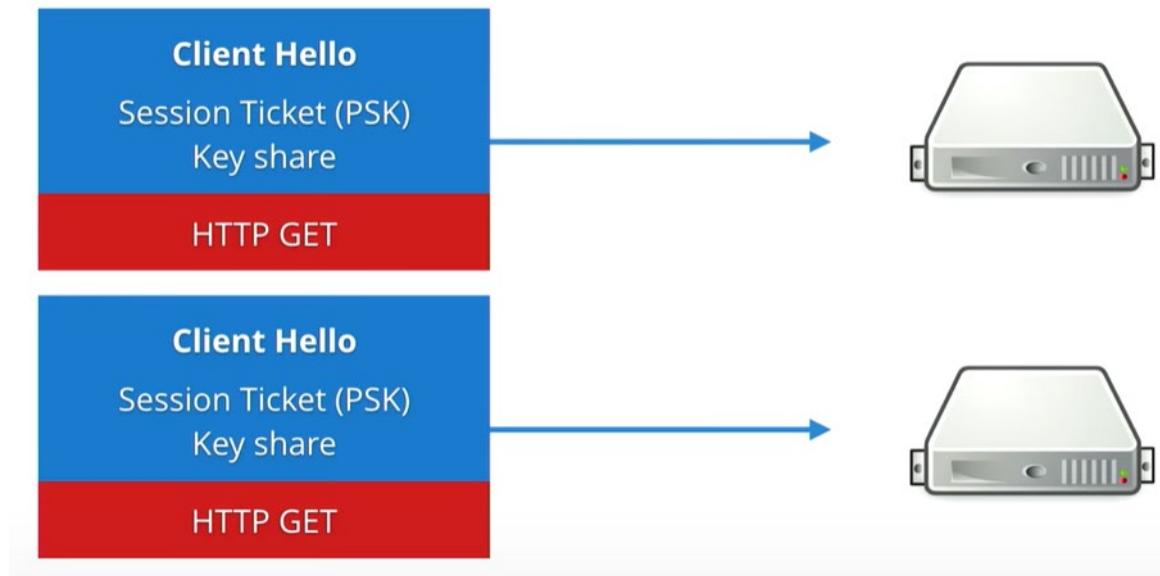


TLS 1.3 resumption 0-RTT



0-RTT limitations

- No forward secrecy on early data
- Early data replay attack on 0-RTT



Forward secrecy comparison

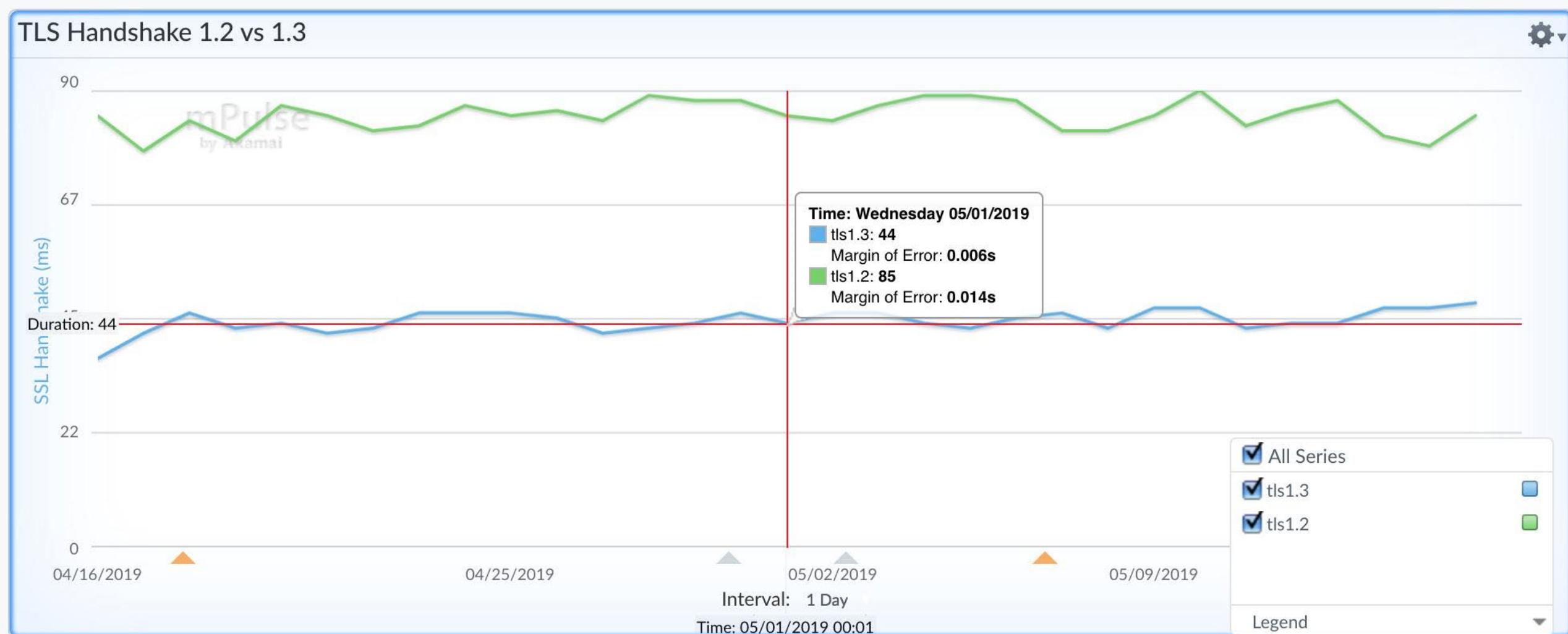
- **TLS 1.2**
 - Certificate compromise: **always but only with ECDHE mode**
 - Ticket key compromise : **never**
- **TLS 1.3**
 - Certificate compromise : **always**
 - Ticket key compromise : **always except 0-RTT early data**

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 ^[54]
ECDH-ECDSA	No	No	Yes	Yes	Yes	No
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes
ECDH-EdDSA	No	No	Yes	Yes	Yes	No
ECDHE-EdDSA (forward secrecy)^[55]	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	Yes
ECDHE-PSK (forward secrecy)	No	No	Yes	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^[56]	No	No	Yes	Yes	Yes	

TLS 1.2 vs TLS 1.3



- Much faster!
- 0-RTT via resumption
- Old ciphers removed
- AEAD ciphers
- Version negotiation removed
- Forward Secrecy



Permitted encryption algorithms

Cipher security against publicly known feasible attacks

Cipher			Protocol version						Status
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 ^{[46][n 5]}	256, 128	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	AES CCM ^{[47][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 ^{[48][n 5]}	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	Camellia CBC ^{[49][n 6]}		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	ARIA GCM ^{[50][n 5]}	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	ARIA CBC ^{[50][n 6]}		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	SEED CBC ^{[51][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 ^{[45][n 7]}	256	N/A	N/A	Insecure	Insecure	Insecure	N/A	Defined in RFC 4357 ^[52]
	IDEA CBC ^{[n 6][n 7][n 9]}	128	Insecure	Insecure	Insecure	Insecure	N/A	N/A	Removed from TLS 1.2
	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	Forbidden in TLS 1.1 and later
	RC2 CBC ^{[n 6][n 7]}	40 ^[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
Stream cipher	ChaCha20-Poly1305 ^{[56][n 5]}	256	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	RC4 ^[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	Prohibited in all versions of TLS by RFC 7465 ^[53]
		40 ^[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
None	Null ^[n 12]	–	N/A	Insecure	Insecure	Insecure	Insecure	N/A	Defined for TLS 1.2 in RFCs

Permitted key-exchange & authentication algorithms

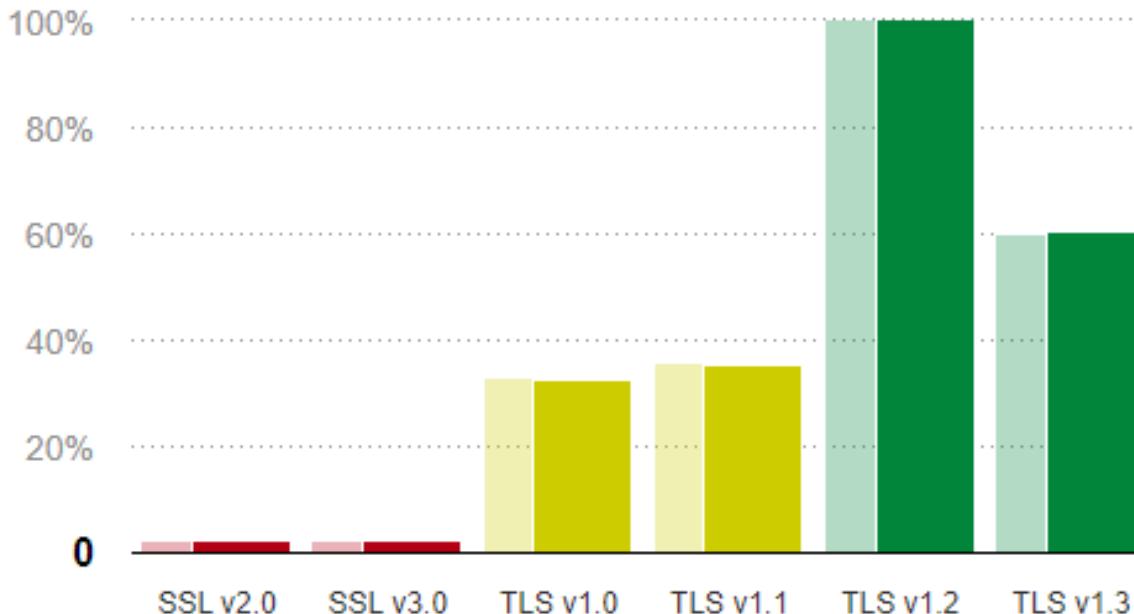
Key exchange/agreement and authentication							
Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
RSA	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
DH-RSA	No	Yes	Yes	Yes	Yes	No	
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes	
ECDH-RSA	No	No	Yes	Yes	Yes	No	
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
DH-DSS	No	Yes	Yes	Yes	Yes	No	
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No ^[44]	
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 ^[45]	No	No	Yes	Yes	Yes		Proposed in RFC drafts

Permitted integrity algorithms

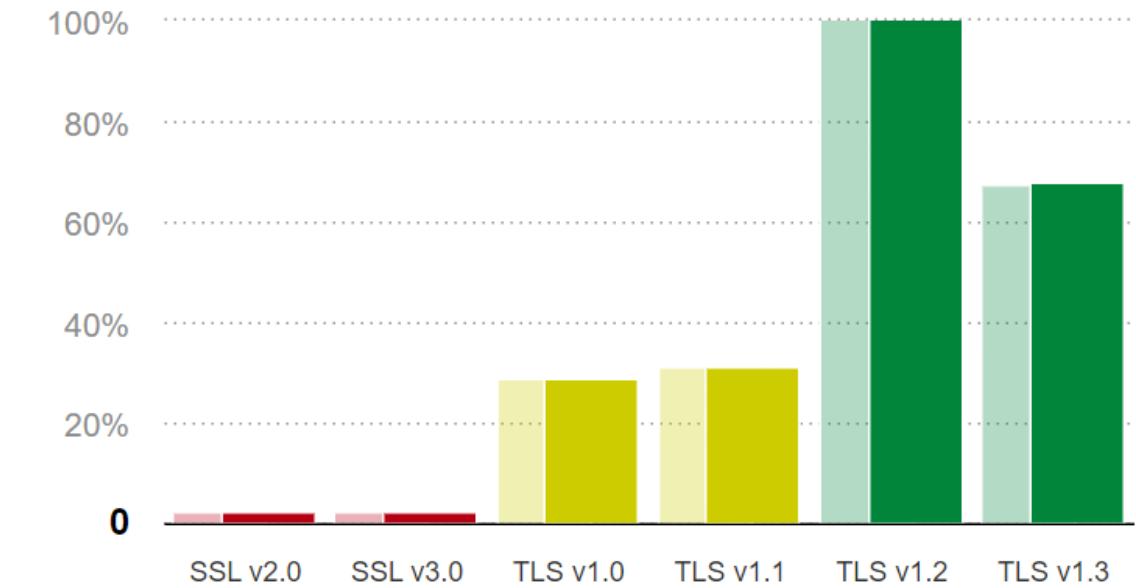
Algorithm	Data integrity						Status
	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	
HMAC-MD5	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
HMAC-SHA1	No	Yes	Yes	Yes	Yes	No	
HMAC-SHA256/384	No	No	No	No	Yes	No	
AEAD	No	No	No	No	Yes	Yes	Proposed in RFC drafts
GOST 28147-89 IMIT ^[45]	No	No	Yes	Yes	Yes		
GOST R 34.11-94 ^[45]	No	No	Yes	Yes	Yes		

Protocol support

Protocol Support



Protocol Support



Source: <https://www.ssllabs.com/ssl-pulse/>

Outline

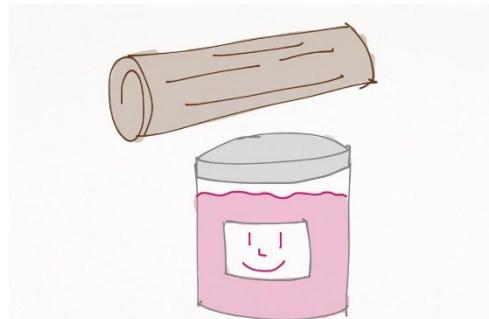
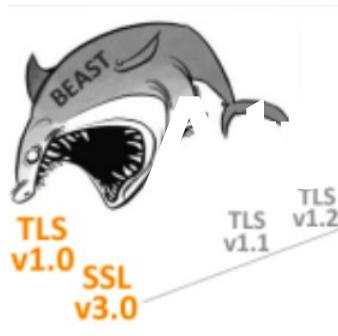
- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercise**

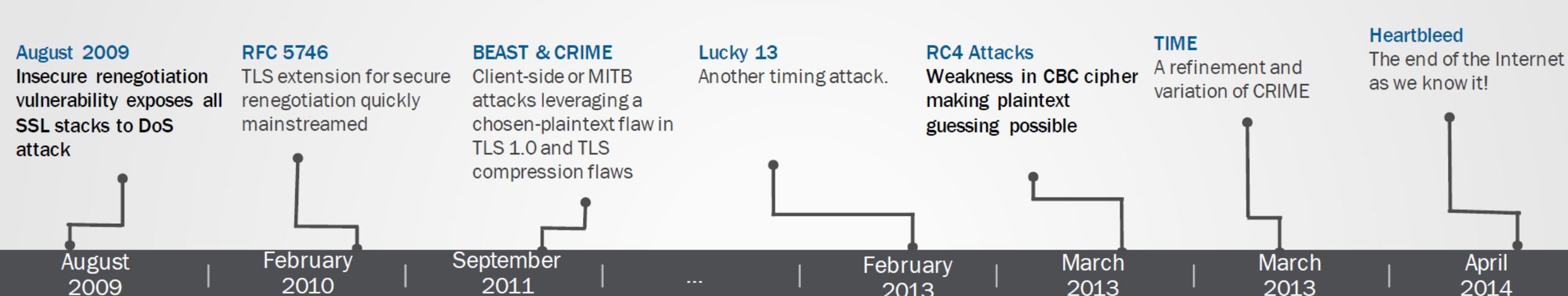
Other TLS based Applications

- DTLS: Datagram Transport Layer Security (DTLS)
 - Provides security for **UDP datagrams**
- SMTPS: Simple Mail Transfer Protocol (SMTP) over TLS
- IMAPS: Internet Message Access Protocol (IMAP) over TLS
- POP3S: Post Office Protocol 3 (POP3) over TLS
- DoT: Domain Name System (DNS) over TLS
- FTPS: File Transfer Protocol (FTP) over TLS
- IRCS: Internet Relay Chat (IRC) over TLS
- ...

Outline

- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercise**





Attacks on TLS

Not many that work against TLS 1.2 or 1.3:

- Cipher-specific attacks (e.g., RC4)
- Compression attacks (e.g., CRIME, TIME, BREACH)
- Downgrade attacks (e.g., POODLE, FREAK)
- Padding Oracle attacks (e.g., Lucky13)
- Implementation attacks (e.g., Heartbleed)

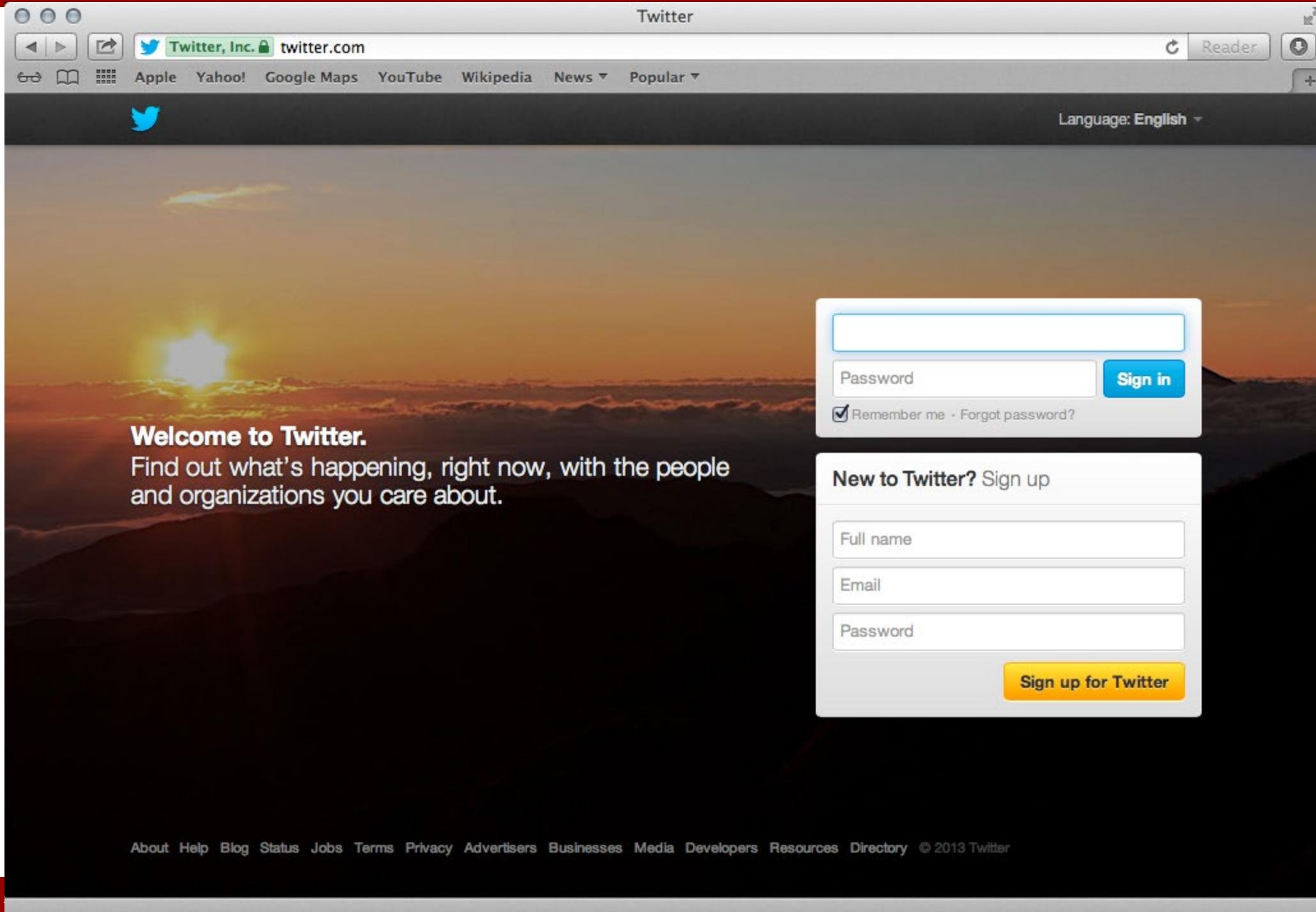
The most used practical attack: sslstrip

- Original presentation: Moxie Marlinspike, Black Hat 2009
<https://www.youtube.com/watch?v=MFoI6IMbZ7Y>
- **sslstrip** is a tool that
 - transparently hijacks HTTP traffic on a network
 - watch for HTTPS links and redirects, and then map those links into look-alike HTTP links or homograph-similar HTTPS links
 - also supports modes for supplying a favicon which looks like a lock icon, selective logging, and session denial.

The most used (practical) attack

- Original presentation: <https://www.youtube.com/watch?v=MFoI6IMbZ7Y>



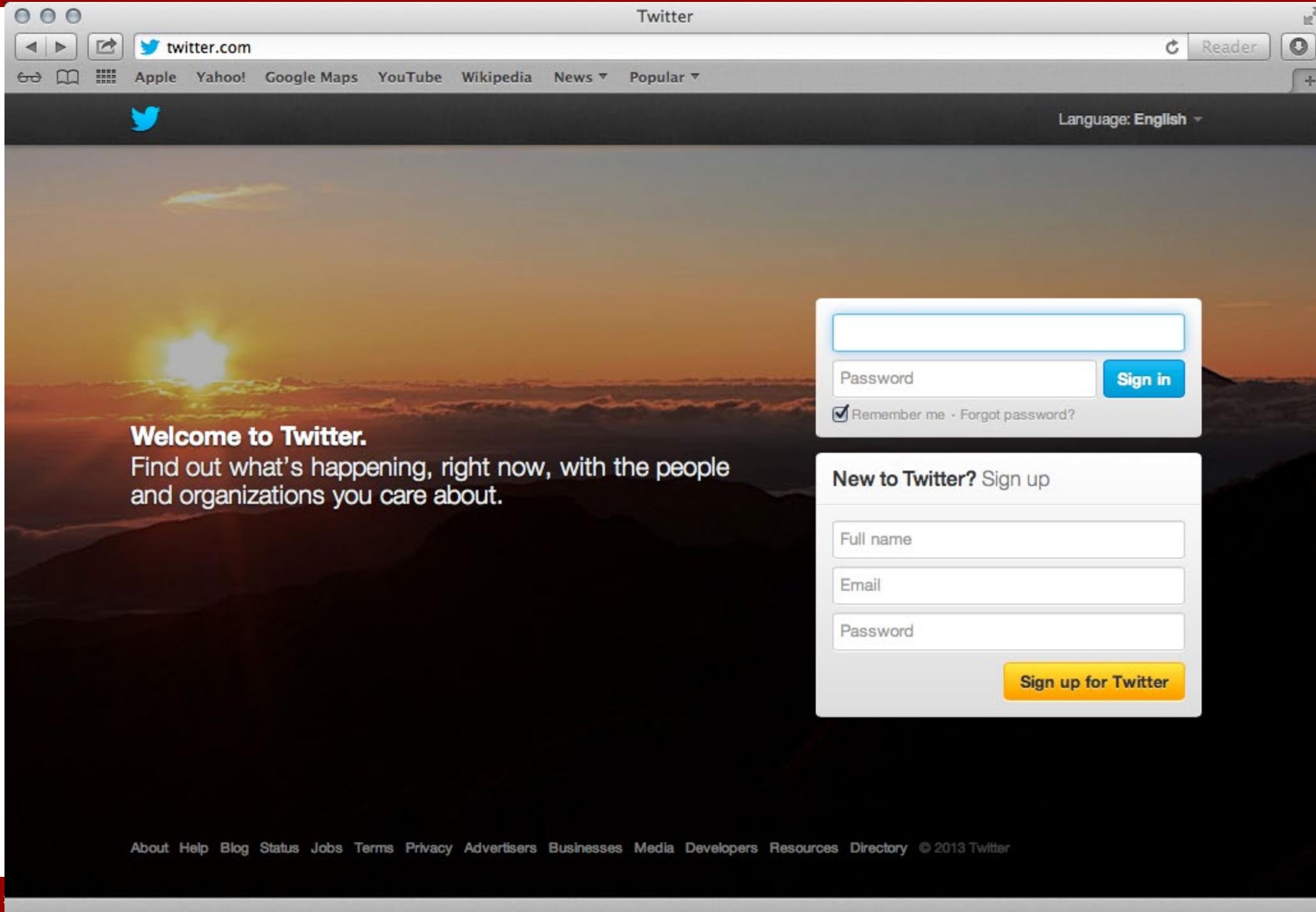


The image shows the Twitter login page. The background features a scenic sunset over mountains. At the top, there's a navigation bar with links like 'Twitter, Inc.', 'twitter.com', 'Apple', 'Yahoo!', 'Google Maps', 'YouTube', 'Wikipedia', 'News', and 'Popular'. On the right side of the header, there are icons for 'Reader' and 'Download'. Below the header, there's a language selection dropdown set to 'English'. A small Twitter logo is located in the top-left corner of the main content area. The central part of the page has two large, semi-transparent callout boxes. The top box is for logging in, containing fields for 'Email' (empty), 'Password', and a 'Sign in' button. It also includes a 'Remember me' checkbox and a 'Forgot password?' link. The bottom box is for sign-up, titled 'New to Twitter? Sign up', and contains fields for 'Full name', 'Email', and 'Password', followed by a large yellow 'Sign up for Twitter' button.

Welcome to Twitter.

Find out what's happening, right now, with the people and organizations you care about.

About Help Blog Status Jobs Terms Privacy Advertisers Businesses Media Developers Resources Directory © 2013 Twitter



The image shows the Twitter login page. The background features a scenic sunset over mountains. At the top, there's a navigation bar with links to Apple, Yahoo!, Google Maps, YouTube, Wikipedia, News, and Popular. On the right side of the header, there are icons for Reader, Download, and a plus sign. Below the header, the Twitter logo is on the left, and the language setting "Language: English" is on the right. The main content area has two large, semi-transparent callout boxes. The top box is for logging in, containing fields for "Email" (empty), "Password", and a "Sign in" button. It also includes a "Remember me" checkbox and a "Forgot password?" link. The bottom box is for new users, titled "New to Twitter? Sign up". It contains fields for "Full name", "Email", and "Password", followed by a large yellow "Sign up for Twitter" button.

Welcome to Twitter.

Find out what's happening, right now, with the people and organizations you care about.

About Help Blog Status Jobs Terms Privacy Advertisers Businesses Media Developers Resources Directory © 2013 Twitter

Is it still possible? (usually not)

- Depends on
 - Website/server
 - User
 - Browser
- Mitigation: **HTTP Strict Transport Security**
 - Policy, RFC 6797
 - Web-servers demand browsers to HTTPS only

HTTP Strict Transport Security

- Limitations
 - Initial request still vulnerable
 - The protection only applies after a user has visited the site at least once
 - TLS attacks: e.g., Beast, Crime, etc. not mitigated
 - DNS spoofing can circumvent the policy

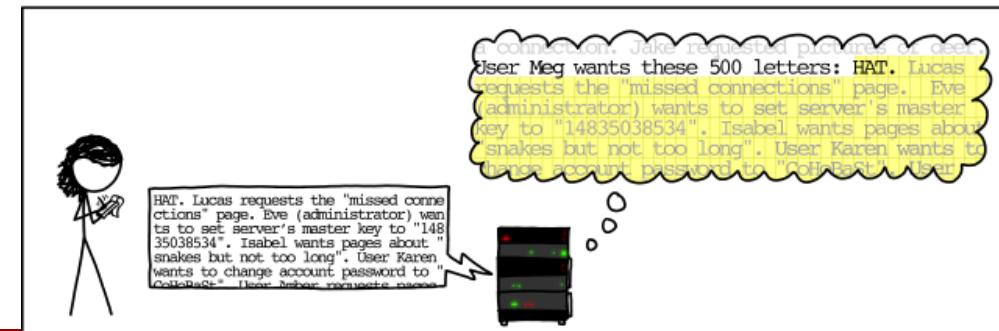
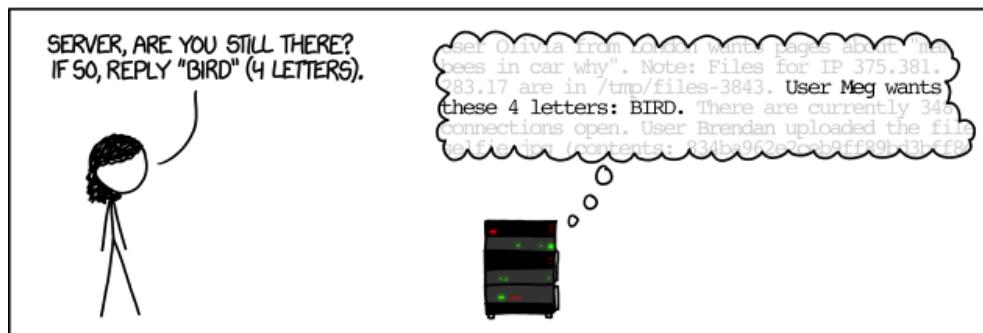
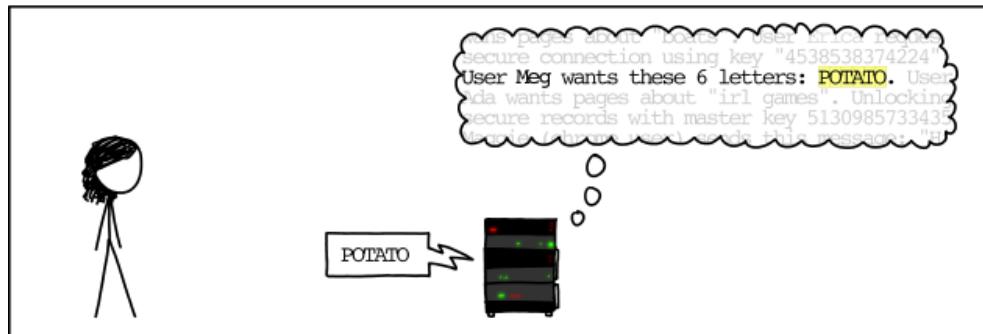
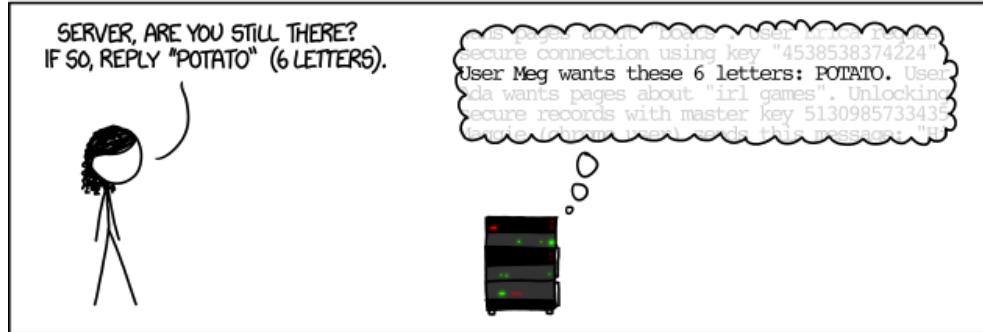


Heartbleed at a glance

- 01.04.14
- Implementation-specific bug of **OpenSSL**
- **Buffer-over-read** type of vulnerability
- Improper input validation (due to a missing bounds check) in the implementation of the TLS heartbeat extension
- **Huge** impact (amazon, Github, Wikipedia, Tumblr, Reddit, etc.)

xkdc Heartbleed Explanation

HOW THE HEARTBLEED BUG WORKS:



Heartbleed at a glance



Some more attacks

- **Poodle attack** (downgrade attack):

<https://www.youtube.com/watch?v=C8ks8WLoZto>

- Blackhat Asia 15, **Bar-Mitzva Attack** (RC4-based)

https://www.youtube.com/watch?v=AHkUCD_EYok

- **FREAK attack** (RSA downgrade attack)

<https://en.wikipedia.org/wiki/FREAK>

Some more attacks

**The 9 Lives of Bleichenbacher's
CAT: New Cache ATtacks on TLS
Implementations**

See: <https://eyalro.net/project/cat.html>

Works against TLS1.3
implementations too (by downgrading
and attacking RSA)



Testing TLS implementations

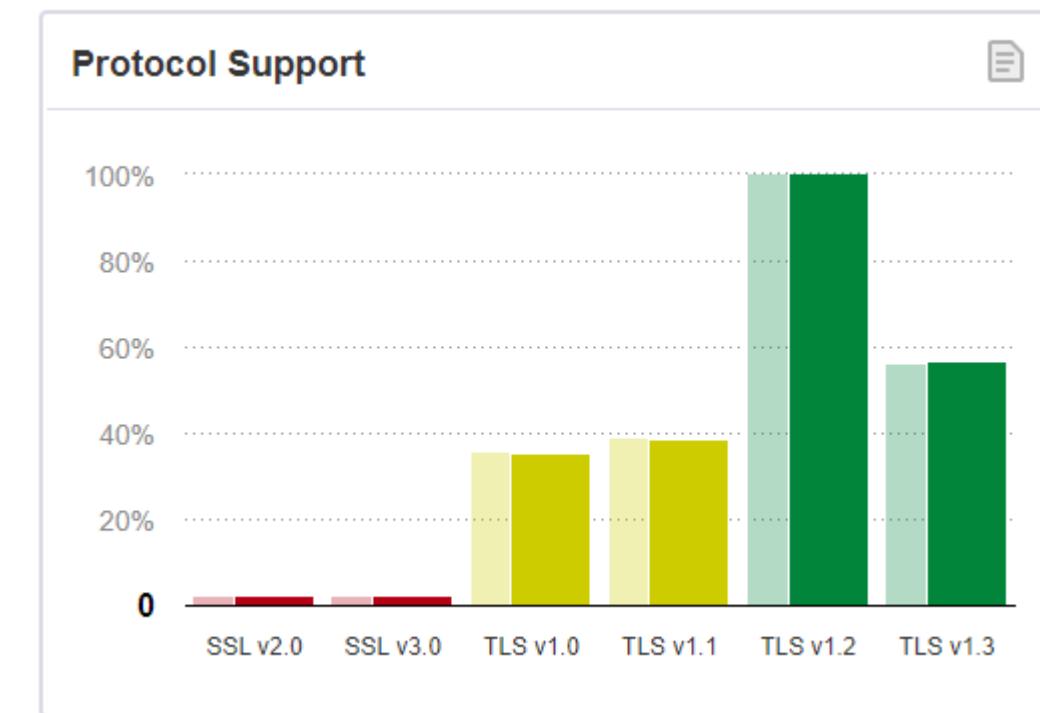
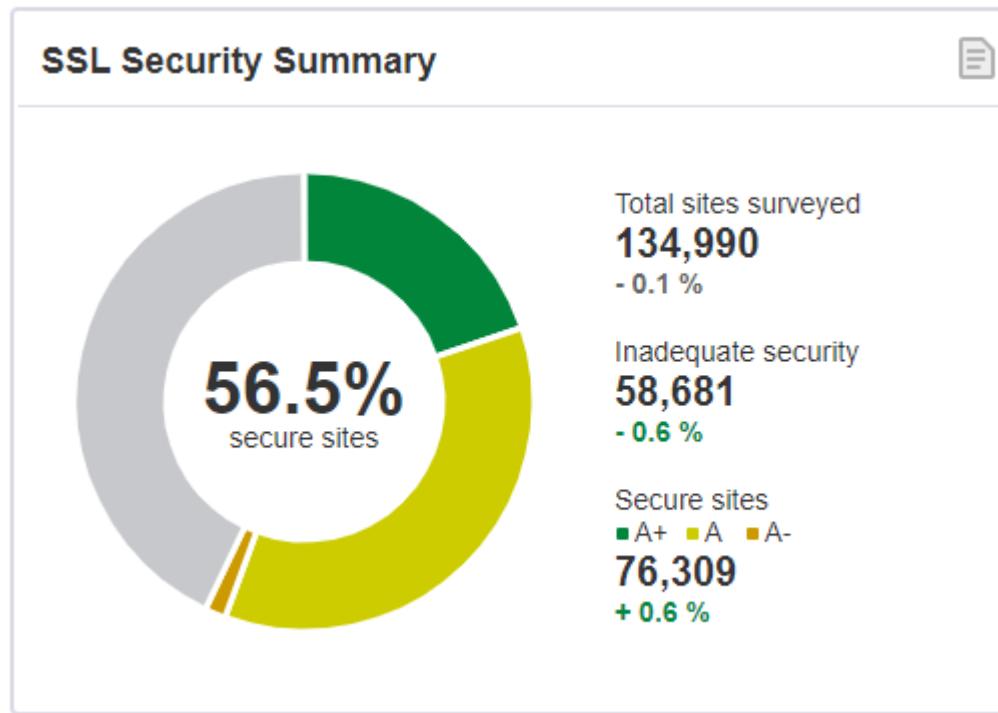
- <https://www.ssllabs.com> offers a very nice scanning tool
- Also <https://www.ssllabs.com/ssl-pulse/> provides some global statistics about TLS usage worldwide

The screenshot shows the Qualys SSL Labs SSL Report for the domain dtu.dk. The report has an overall rating of B. The summary section includes a chart comparing Certificate, Protocol Support, Key Exchange, and Cipher Strength. Below the chart, there are links to documentation and information about supported TLS versions. The detailed protocols section lists support for TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0, SSL 3, and SSL 2, with TLS 1.2, 1.1, and 1.0 marked as Yes. The revocation information section shows that the certificate is not revoked.

Protocol	Status
TLS 1.3	No
TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

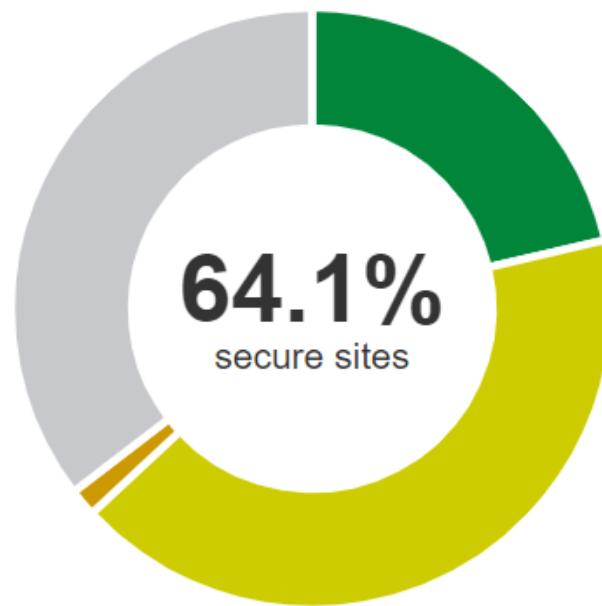
Revocation information: CRL: http://GEANT.crl.sectigo.com/GEANTOVRSACA4.crl, OCSP: http://GEANT.ocsp.sectigo.com
Revocation status: Good (not revoked)

From (2023) <https://www.ssllabs.com/ssl-pulse>



From (2024) <https://www.ssllabs.com/ssl-pulse>

SSL Security Summary



Total sites surveyed

134,595

- 0.4 %

Inadequate security

48,330

- 0.6 %

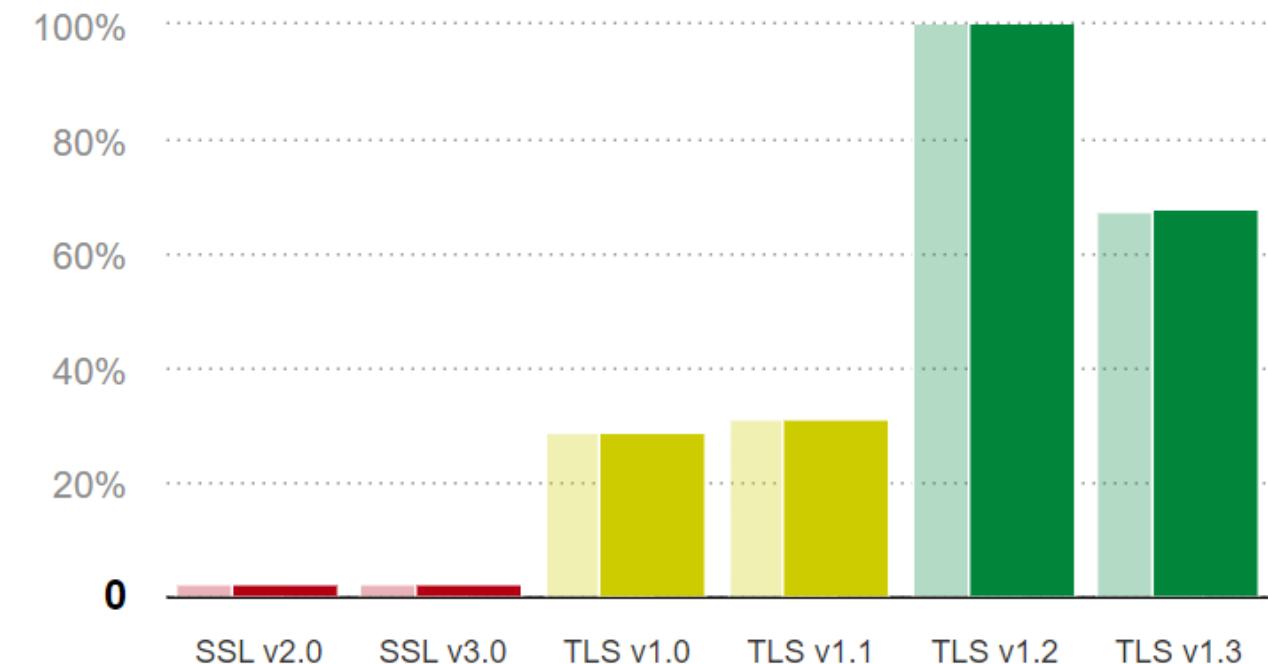
Secure sites

■ A+ ■ A ■ A-

86,265

+ 0.6 %

Protocol Support



Outline

- **Introduction**
- **Certificates**
- **TLS**
 - What it can offer
 - TLS protocol
 - TLS resumption
- **Beyond HTTP**
- **Attacks on TLS**
- **Lab exercise**

Conclusion

- **Transport Layer Security**
 - One of THE most important security protocols in use
 - Various layers/sub-protocols
 - Handshake is the most complicated one
 - Remember that most connections are TLS Resumptions
 - Used for many, many different things
 - HTTPS, email (kinda), TOR, voip, etc.

Further reading

- **Deploying TLS 1.3: the great, the good and the bad (33c3) from Cloudflare**
 - <https://www.youtube.com/watch?v=0opakLwtPWk>
- **Certificate transparency**
 - <https://certificate.transparency.dev/>



Lab exercises

- Get hands-on experience with **self-signed certificates** using openssl
- Get hands-on experience with **setting up a test HTTPS** Web server using openssl
- Replicate an example of **HTTP** and **Wireshark**
- Play with sslstrip (optional)