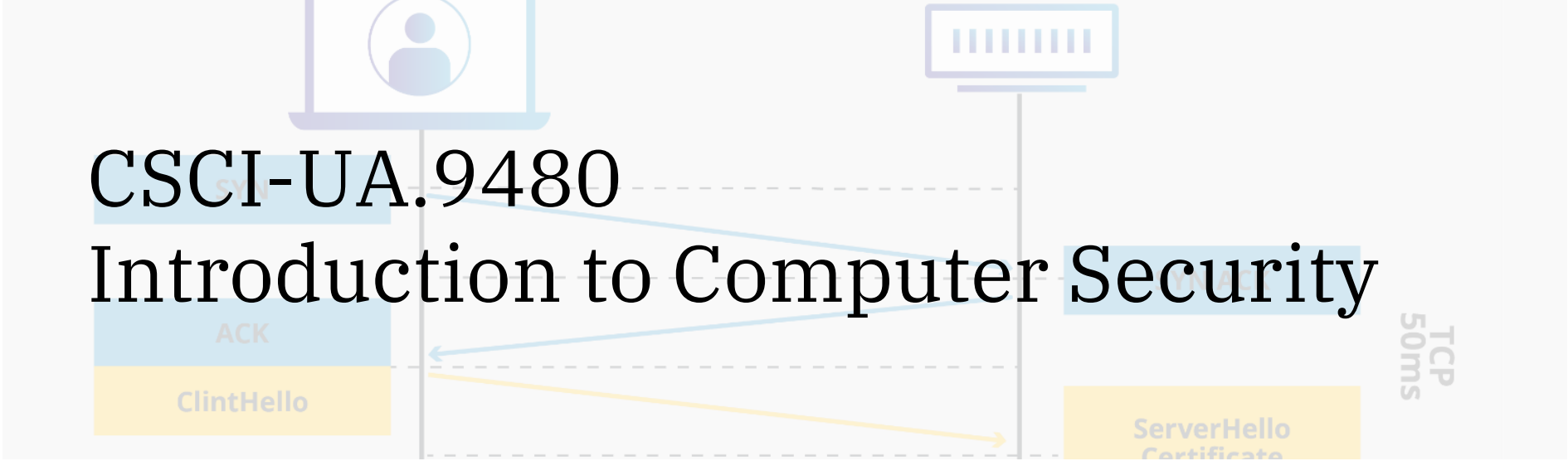


CSCI-UA.9480
Introduction to Computer Security

NYU

Session 1.4
Transport Layer Security

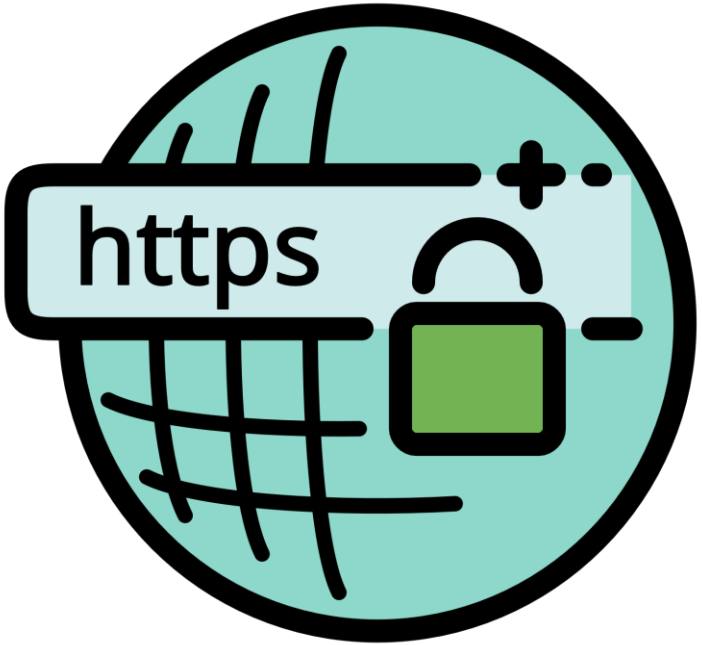
Prof. Nadim Kobeissi

HTTPS and TLS

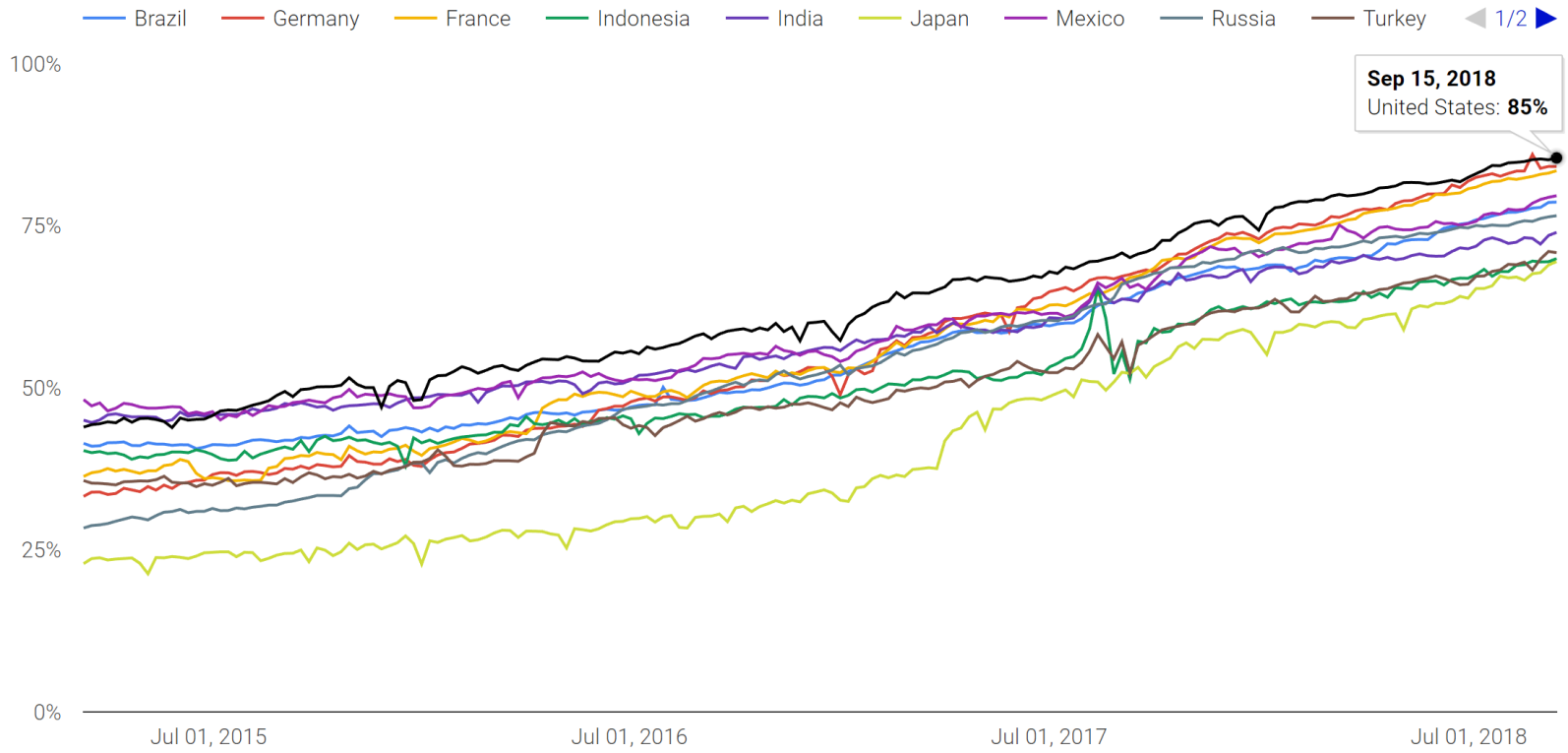
1.4a

What is TLS?

- The **S** in HTTP**S**.
- Most likely the most relevant web encryption protocol.
- Built on all the technologies we've seen so far:
    - Public key cryptography.
    - Symmetric encryption.
    - Hashing.

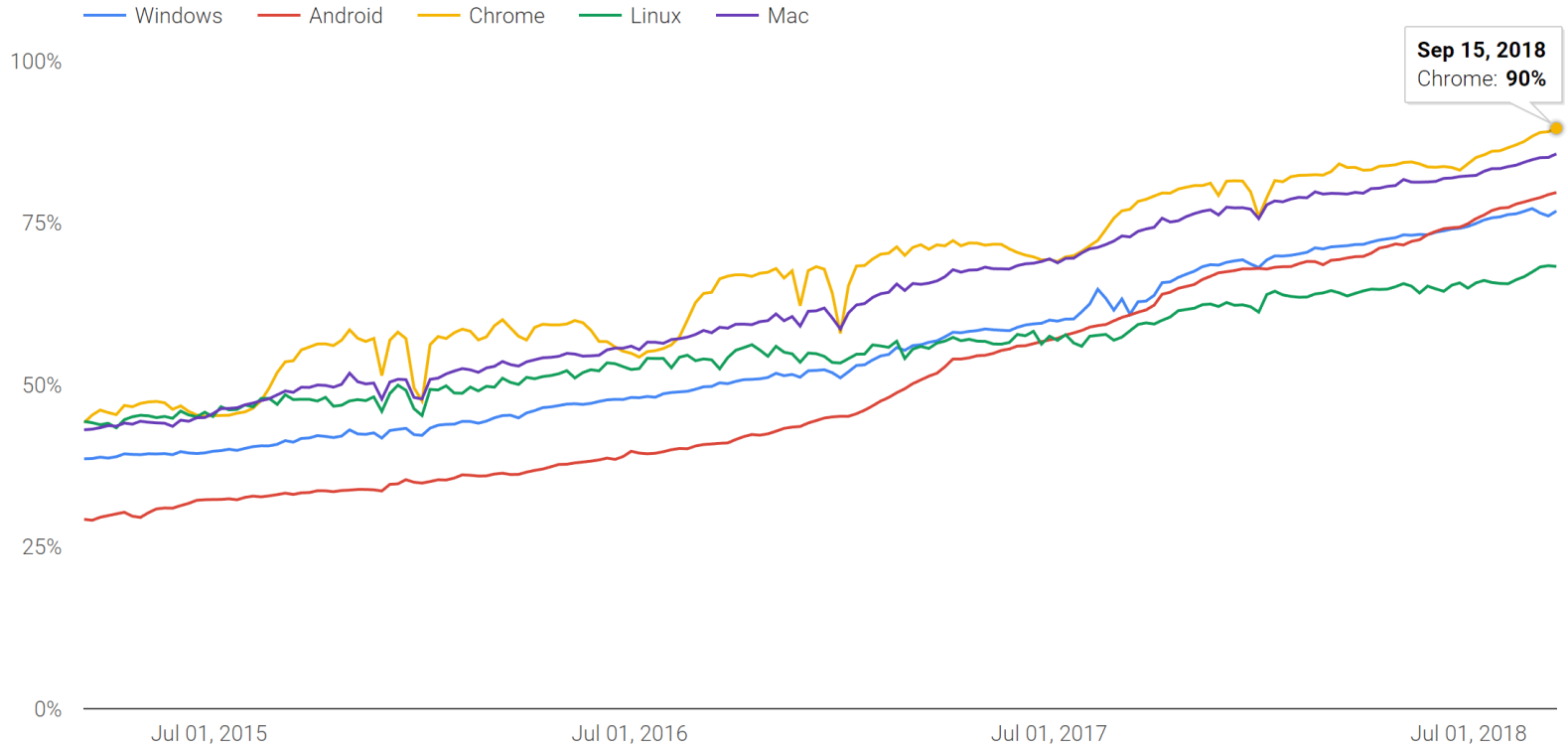


HTTPS Pages by Country (Chrome)



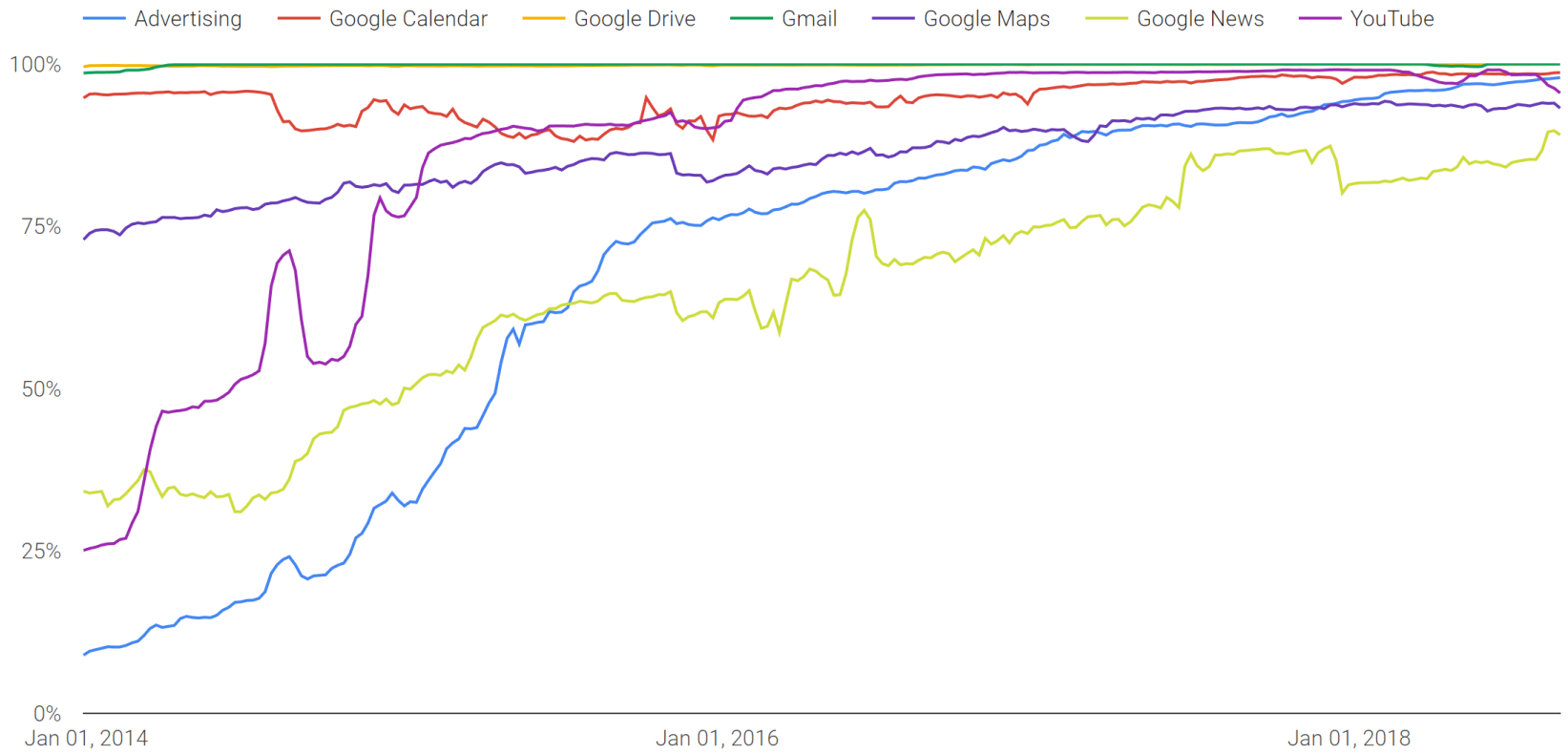
Source: https://transparencyreport.google.com/https/

HTTPS Pages by Platform (Chrome)



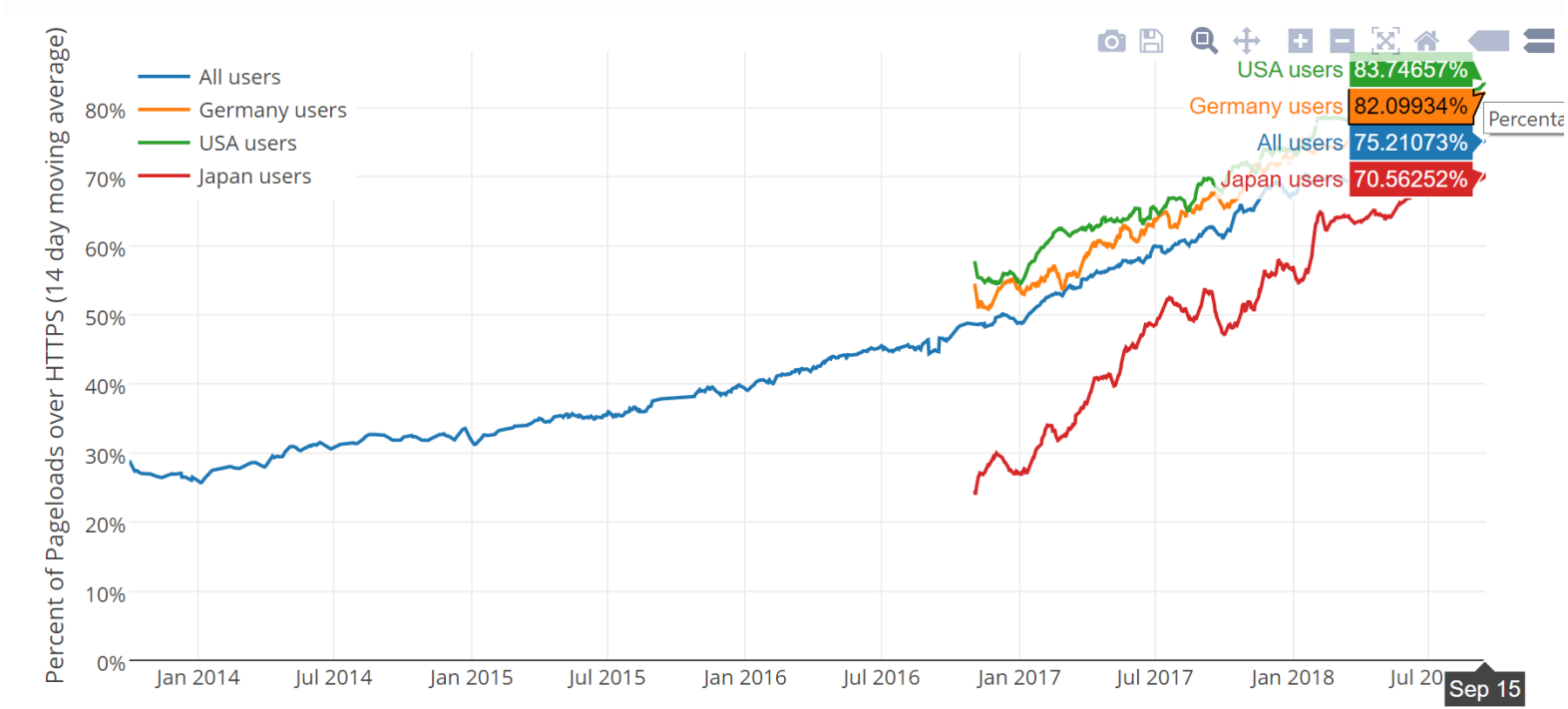
Source: https://transparencyreport.google.com/https/

HTTPS Pages by Google Service



Source: https://transparencyreport.google.com/https/

HTTPS Pages by Country (Firefox)



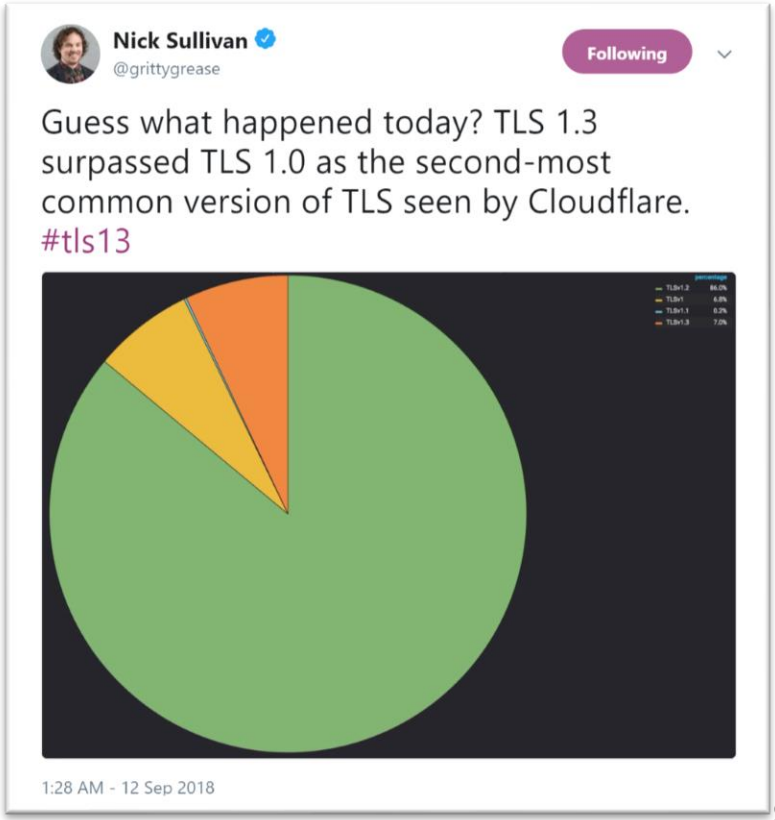
Source: https://letsencrypt.org/stats/

Did you know?

SSL 1.0 was never released due to critical security flaws. SSL 2.0 barely lasted one year before being replaced.

History of TLS

- SSL (Secure Socket Layer) 1.0 was never released. SSL 2.0 lasted a year. SSL 3.0 released in 1996.
- TLS 1.0 released in 1999.
- TLS 1.1 released in 2006.
- TLS 1.2 released in 2008.
- TLS 1.3 released in 2018.



As discussed last time: protocols.

In *protocols*, we reason about:

- Principals: Alice, Bob.
- Security goals: confidentiality, authenticity, forward secrecy...
- Use cases and constraints.
- Attacker model.
- Threat model.

Protocols need to do things.

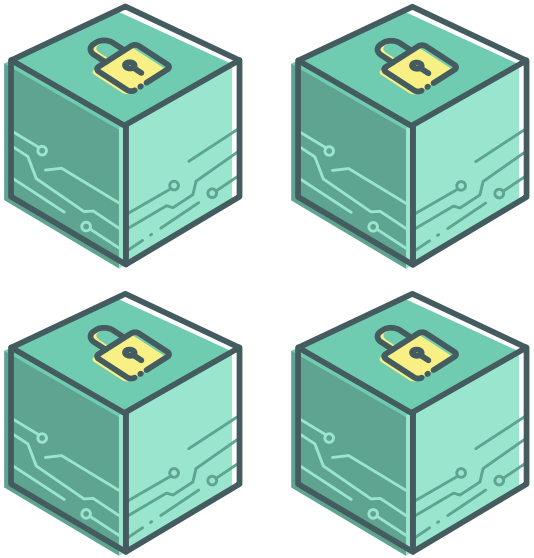
Protocols are frequently entrusted with:

- Communicating secret data without a malicious party being able to read it: *confidentiality*.
- Ensuring that any data Bob receives that appears to be from Alice is indeed from Alice: *authenticity*.
- Limiting the damage that can be caused by device compromise or theft: *post-compromise security.*

Protocols need to do things.

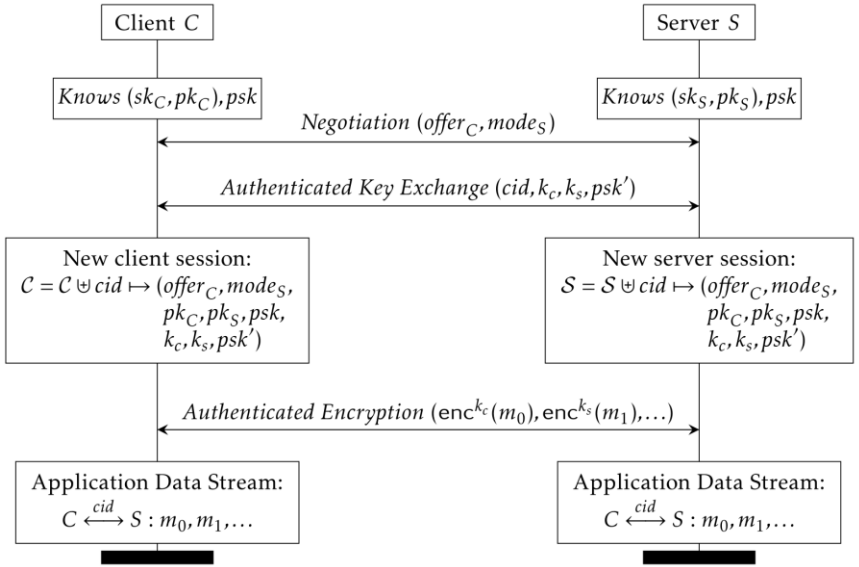
Protocols have building blocks:

- *Public key agreement*: Client and server agree on some shared secret key over an insecure channel.
- *Symmetric encryption*: Encrypting and decrypting data with a shared secret key.
- *Hashing and signatures*: Providing integrity and authenticity of communicated data.



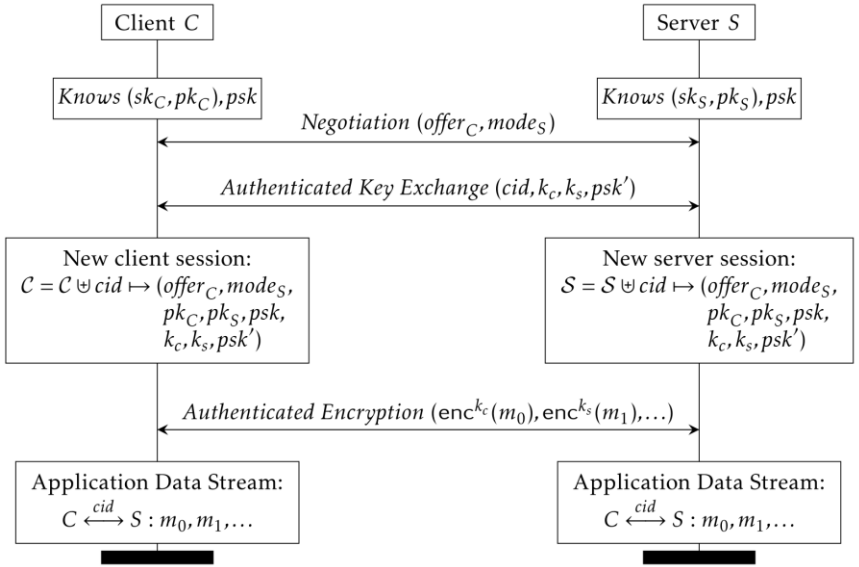
TLS is a secure channel protocol.

- *Authenticated key exchange phase*: Exchange public keys, establish shared secrets and start a session.
- *Application data/messaging stage*: Send encrypted, authenticated data (websites, messages, files, videos...)



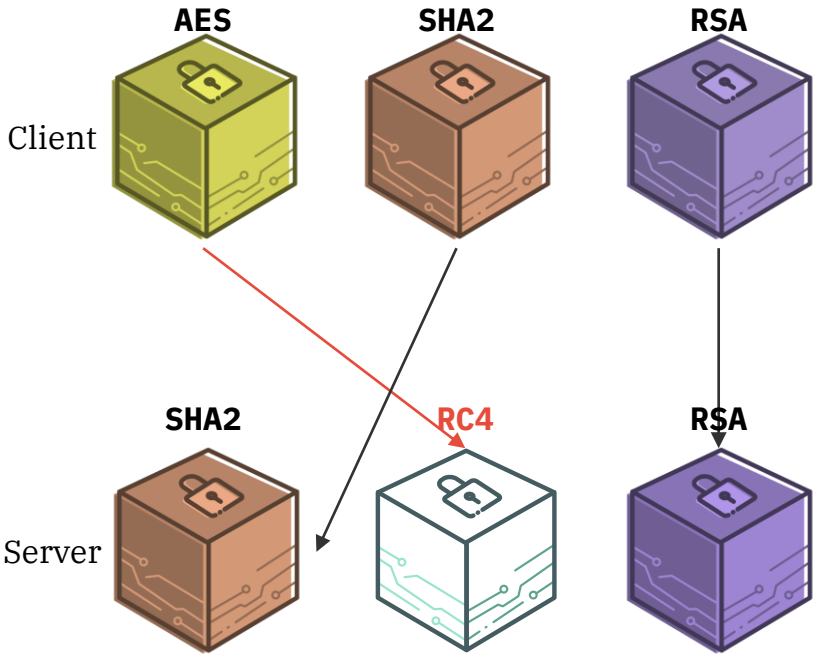
TLS is a secure channel protocol.

- *Client's local state:* server certificate, accepted cipher configurations, ephemeral public key pair, pre-shared secret for session resumption…
- *Server's local state:* long-term keys, accepted cipher configurations, pre-shared secret for session resumption…



Cipher suites?

- Set of supported cryptographic primitives by the client and server.
- What if the server advertises a bad cipher suite?
    - FREAK, POODLE, LOGJAM...



Evaluating HTTPS overall security.

SSL Report: nyu.edu (216.165.47.10)

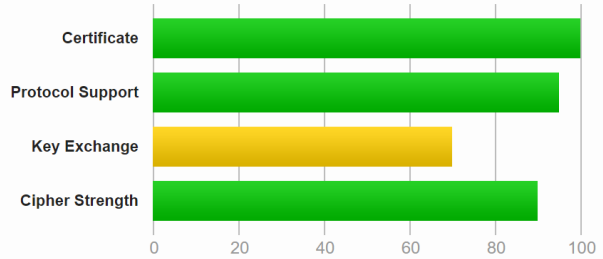
Assessed on: Mon, 17 Sep 2018 12:50:49 UTC | Hide | Clear cache

Scan Another »

Summary

Overall Rating

B



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports weak Diffie-Hellman (DH) key exchange parameters. Grade capped to B. MORE INFO »

This server does not support Forward Secrecy with the reference browsers. Grade capped to B. MORE INFO »

NYU.edu: Supported protocols.

Protocols

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

For TLS 1.3 tests, we currently support draft version 28.

NYU.edu: Supported cipher suites.

Cipher Suites

 # TLS 1.2 (suites in server-preferred order)

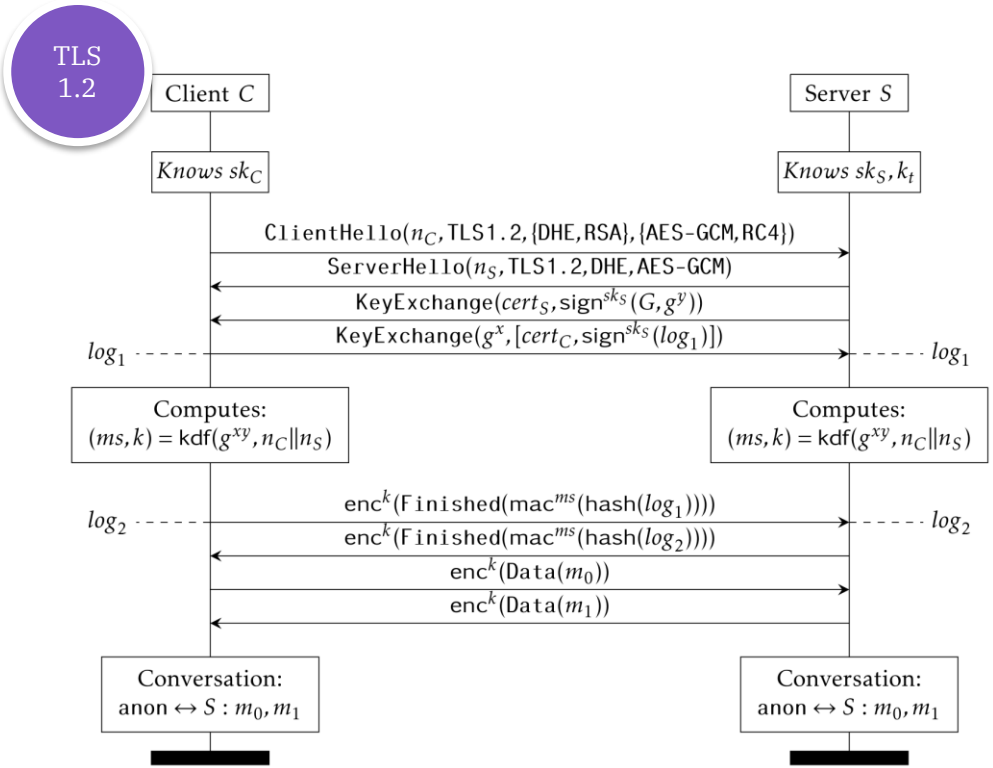
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f)	DH 1024 bits FS **WEAK**	256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	DH 1024 bits FS **WEAK**	128
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b)	DH 1024 bits FS **WEAK**	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39)	DH 1024 bits FS **WEAK**	256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67)	DH 1024 bits FS **WEAK**	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)	DH 1024 bits FS **WEAK**	128
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16)	DH 1024 bits FS **WEAK**	112
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	**WEAK**	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	**WEAK**	128
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	**WEAK**	256
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	**WEAK**	256
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	**WEAK**	128
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	**WEAK**	128
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	**WEAK**	112
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp384r1 (eq. 7680 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH secp384r1 (eq. 7680 bits RSA) FS	128

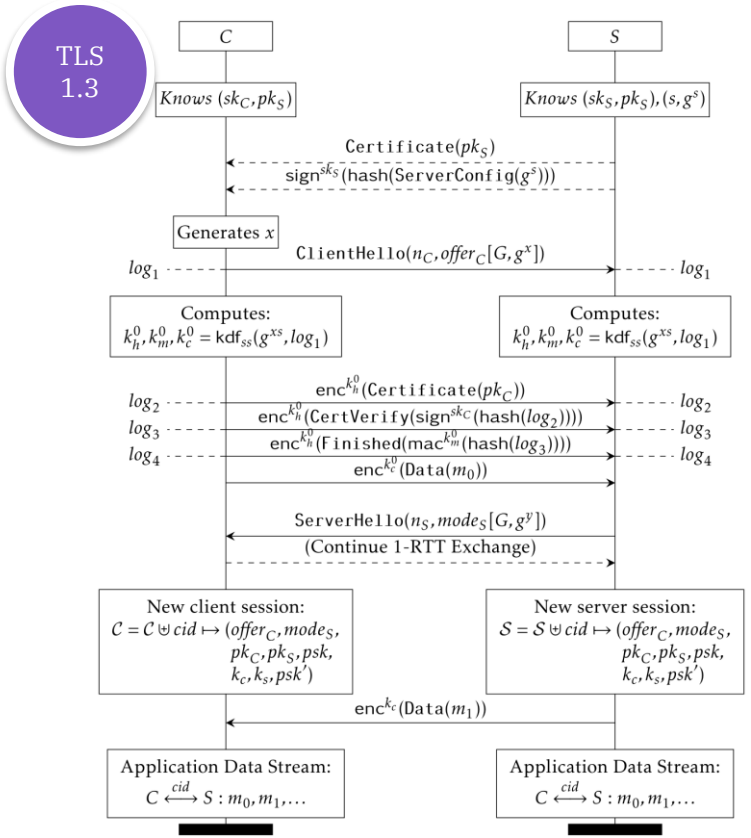
NYU.edu: Supported devices.

Handshake Simulation

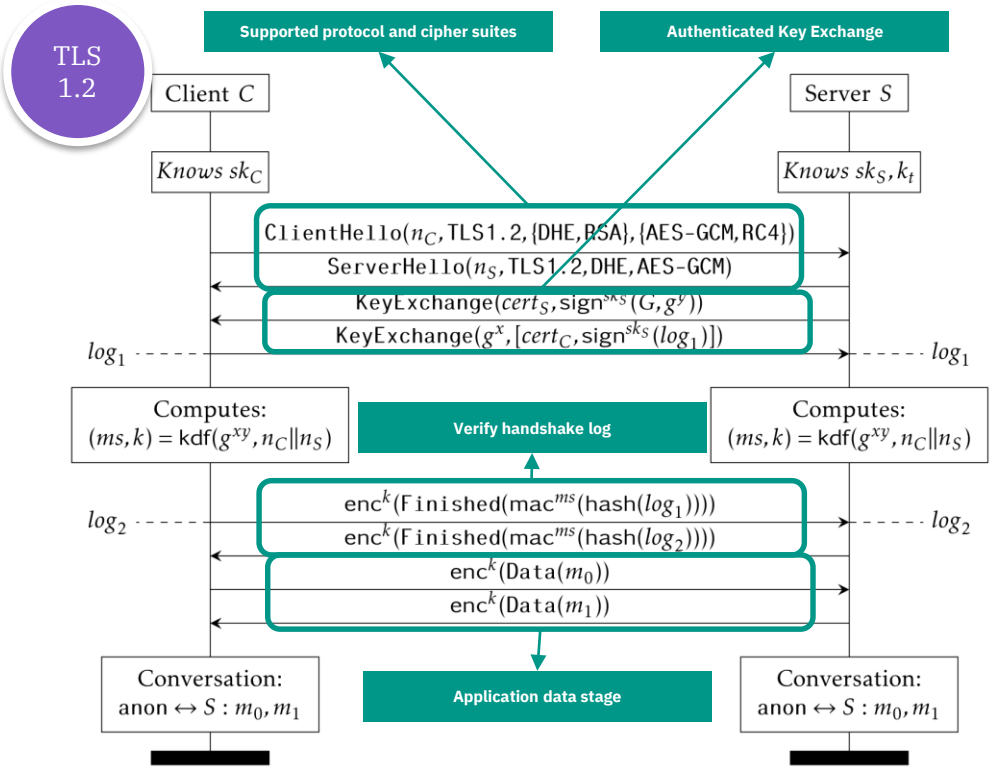
Android 2.3.7 No SNI [2]	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_128_CBC_SHA DH 1024 FS
Android 4.0.4	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Android 4.1.1	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Android 4.2.2	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Android 4.3	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Android 4.4.2	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 DH 1024 FS
Android 5.0.0	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 DH 1024 FS
Android 6.0	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 DH 1024 FS
Android 7.0	RSA 2048 (SHA256)	TLS 1.2	TLS_RSA_WITH_AES_256_GCM_SHA384 No FS
Baidu Jan 2015	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
BingPreview Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 DH 1024 FS
Chrome 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2	TLS_RSA_WITH_AES_128_GCM_SHA256 No FS
Chrome 69 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_RSA_WITH_AES_256_GCM_SHA384 No FS
Firefox 31.3.0 ESR / Win 7	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Firefox 47 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Firefox 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 1024 FS
Firefox 62 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_RSA_WITH_AES_256_CBC_SHA No FS
Googlebot Feb 2018	RSA 2048 (SHA256)	TLS 1.2	TLS_RSA_WITH_AES_256_GCM_SHA384 No FS
IE 7 / Vista	RSA 2048 (SHA256)	TLS 1.0	TLS_RSA_WITH_AES_256_CBC_SHA No FS

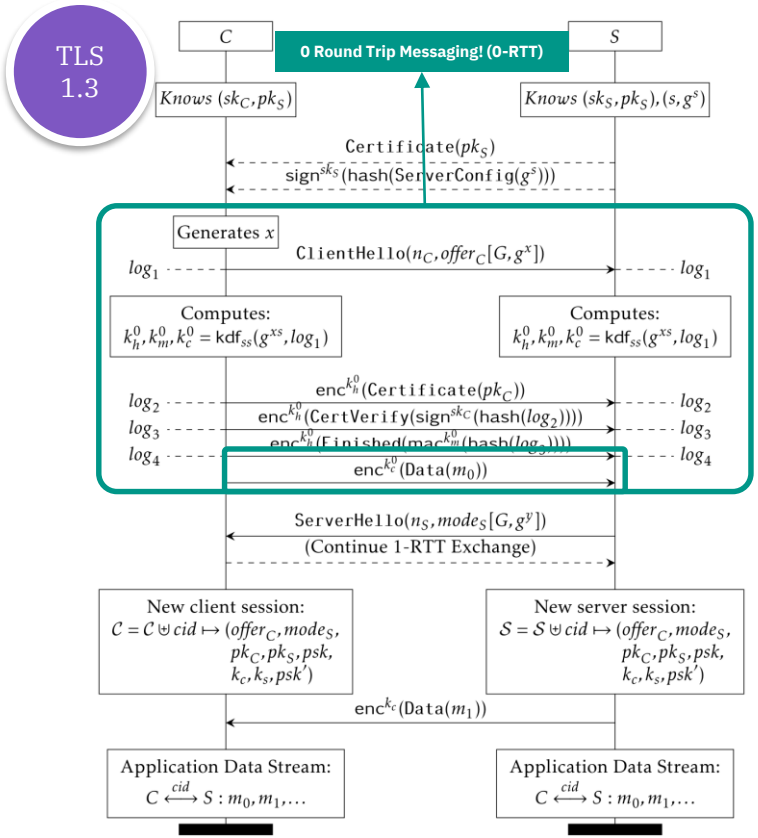
TLS 1.2 and TLS 1.3: How Protocols Evolve





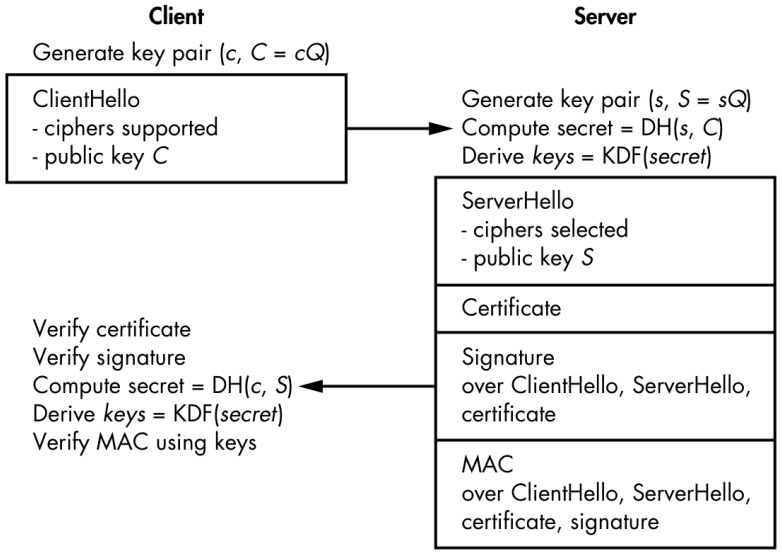
TLS 1.2 and TLS 1.3: How Protocols Evolve





TLS 1.3: A Simpler Overview

- By employing the primitives introduced in earlier sessions, we obtain all of our security guarantees.

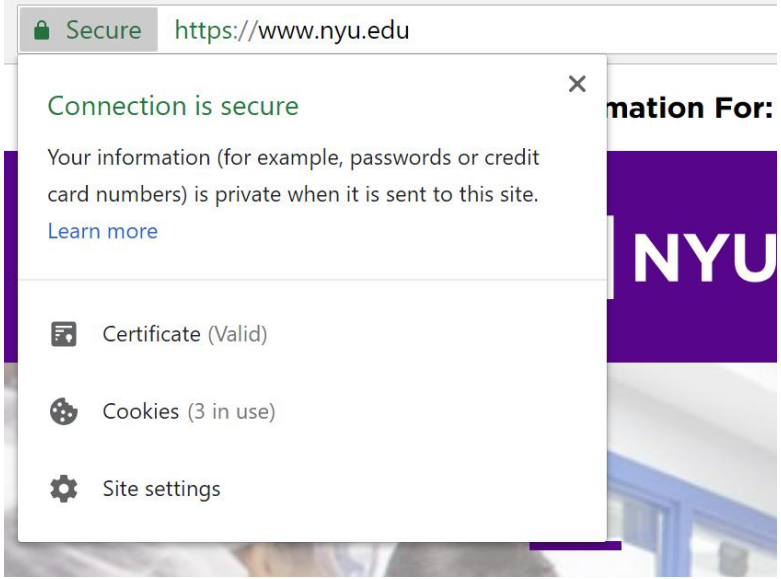


Public Key Infrastructure

1.4b

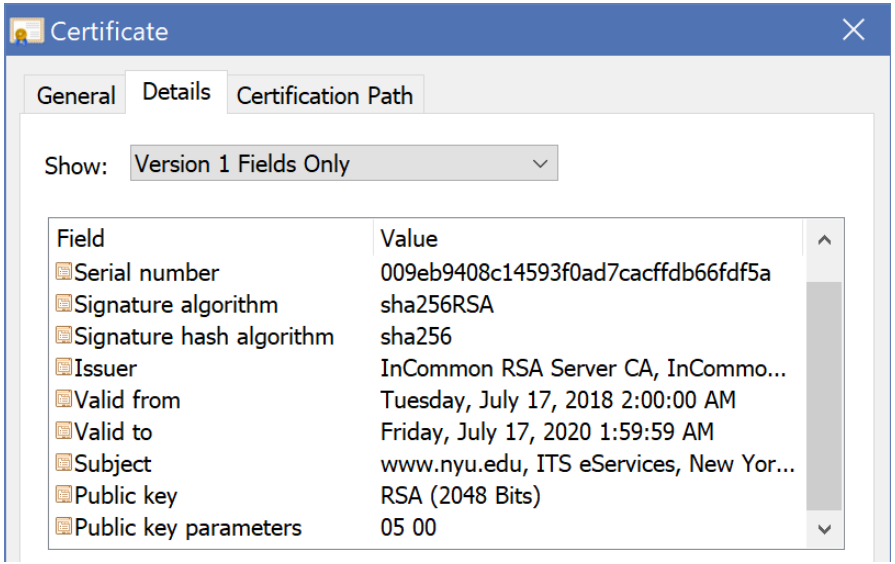
Why do certificates matter?

- Certificates *authenticate* a set of claims that a server is making about its authority and ownership over some website.



Why do certificates matter?

- Certificates *authenticate* a set of claims that a server is making about its authority and ownership over some website.
    - Long-term public keys (identity keys.)
    - Entity operating the website.
- But who vouches for these claims? Certificate authorities.
- Public signing keys of certificate authorities shipped hardcoded into consumer devices.



Certificate Authorities: a complete mess.

Certificate authorities are a scam that benefits nobody.

- They contribute almost nothing to online security, cost a lot of money, are a barrier to deploying secure websites.
- If one of them gets compromised, the entire Web's endpoint authentication is put at risk.

Certificate Authorities: a complete mess.

NEWS

Microsoft blacklists latest rogue SSL certificates

By Lucian Constantin

Romania Correspondent, IDG News Service | MAR 25, 2015 8:04 AM PT

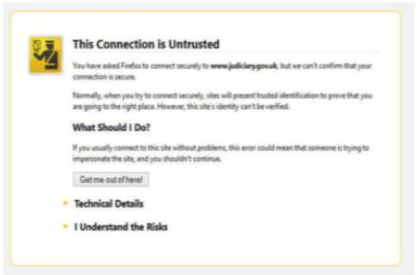
Microsoft has blacklisted a subordinate CA certificate that was wrongfully used to issue SSL certificates for several Google websites. The action will prevent those certificates from being used in Google website spoofing attacks against Internet Explorer users.

Security

Google Chrome's HTTPS ban-hammer drops on WoSign, StartCom in two months

Substandard certs, already in partial exile, soon to be shunned completely

By Thomas Claburn in San Francisco 7 Jul 2017 at 22:27 27



Update Google in two months will conclude its prolonged excommunication of misbehaving SSL/TLS certificate authorities WoSign and subsidiary StartCom, a punishment announced last October.

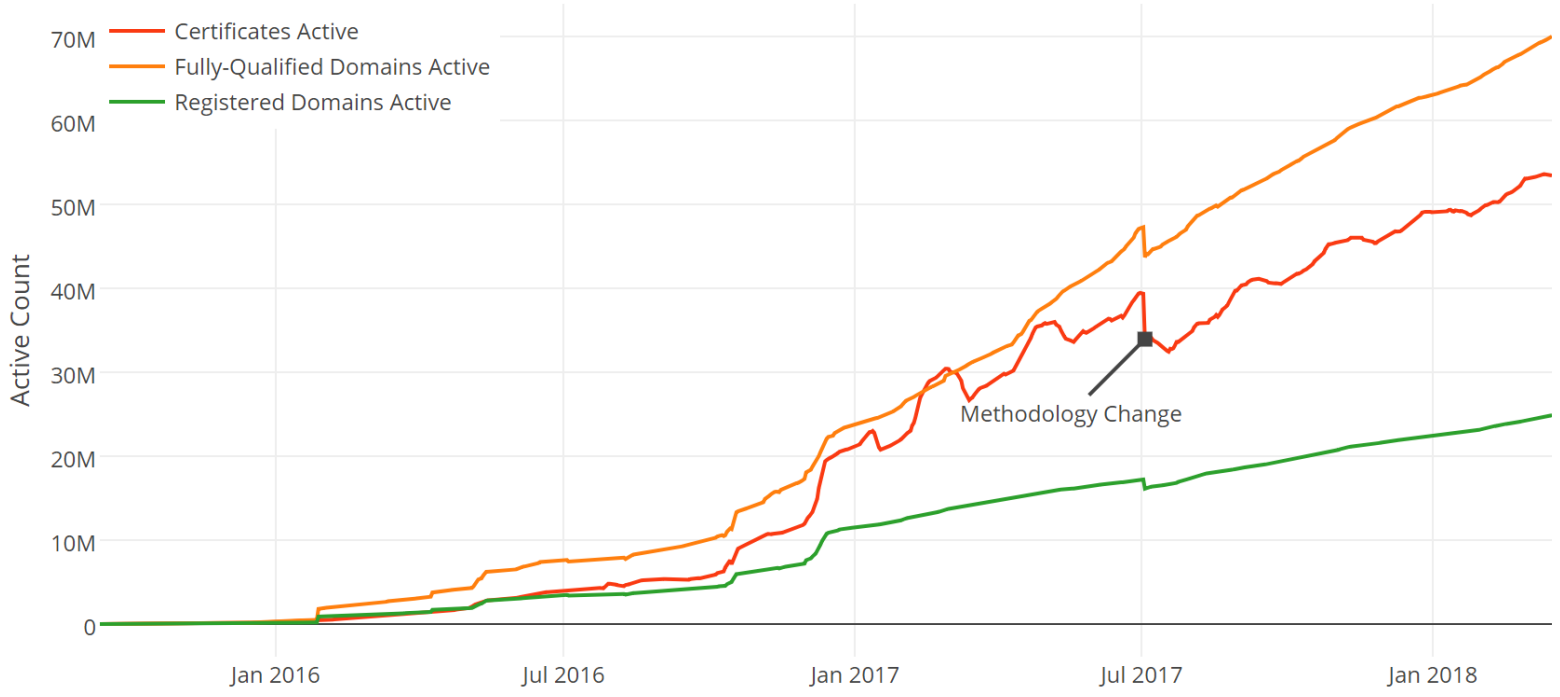
Symantec has fallen out of the good graces of the InfoSec community, and the larger companies in Silicon Valley are taking action. As Bleeping Computer reports, Mozilla's Firefox Nightly will release a beta version in early September that recognizes Symantec TLS certs as a security risk. When a user accesses websites with Symantec certificates, they will be met with a message informing that their connection isn't private. Additionally, Google has set up its September beta release of Chrome 70 Canary to give a similar warning to its users who land on Symantec TLS encrypted pages.

The move comes after a July investigation conducted by Google and Mozilla engineers showed that Symantec did not consistently follow the regulations for TLS issuing. As Bleeping Computer notes, this set of actions on the part of Google and Mozilla is the final step in fully legitimizing Symantec certificates, with the first step being Symantec "demoting itself from the position of Root Certificate Authority to that of a Subordinate Certificate Authority that abides by the rules of a different party."

Let’s Encrypt: a new hope?

- Free certificates.
- Automated certificate issuance protocol (ACME) – the first of its kind!
    - Formally verified recently.
- Free secure websites for everyone.

Let's Encrypt Growth



Source: https://letsencrypt.org/stats/

Certificate Authority Market Share

	2017 1 Sep	2017 1 Oct	2017 1 Nov	2017 1 Dec	2018 1 Jan	2018 1 Feb	2018 1 Mar	2018 1 Apr	2018 1 May	2018 1 Jun	2018 1 Jul	2018 1 Aug	2018 1 Sep	2018 17 Sep
IdenTrust	29.6%	30.5%	31.5%	32.5%	32.8%	33.5%	35.5%	36.9%	38.3%	39.6%	41.0%	44.0%	45.4%	45.9%
Comodo	39.8%	39.4%	38.7%	38.2%	38.0%	37.6%	36.7%	36.2%	35.8%	35.1%	34.0%	32.3%	31.4%	31.0%
DigiCert Group	2.2%	2.2%	2.2%	15.0%	14.8%	14.5%	13.8%	13.2%	12.7%	12.3%	12.1%	11.4%	11.0%	10.8%
GoDaddy Group	7.6%	7.5%	7.5%	7.5%	7.5%	7.5%	7.4%	7.4%	7.3%	7.2%	7.2%	6.9%	6.9%	6.9%
GlobalSign	4.6%	4.5%	4.5%	4.4%	4.4%	4.3%	4.2%	3.9%	3.7%	3.5%	3.5%	3.3%	3.1%	3.1%
Certum	0.6%	0.6%	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.8%	0.8%
Actalis	0.2%	0.2%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.4%	0.4%
Entrust	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%
Secom Trust	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%	0.3%
Let's Encrypt	0.1%	0.2%	0.2%	0.2%	0.1%	0.2%	0.2%	0.2%	0.1%	0.2%	0.2%	0.2%	0.2%	0.2%
Trustwave	0.3%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
WISeKey Group	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
StartCom	0.2%	0.1%	0.2%	0.2%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	<0.1%	<0.1%
Symantec Group	13.8%	13.4%	13.1%											

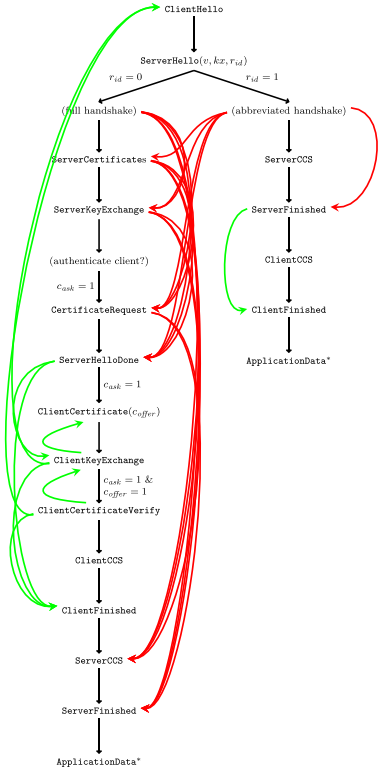
Source: https://w3techs.com/technologies/history_overview/ssl_certificate

Attacks on TLS

1.4c

Attacks on TLS: SMACK and FREAK

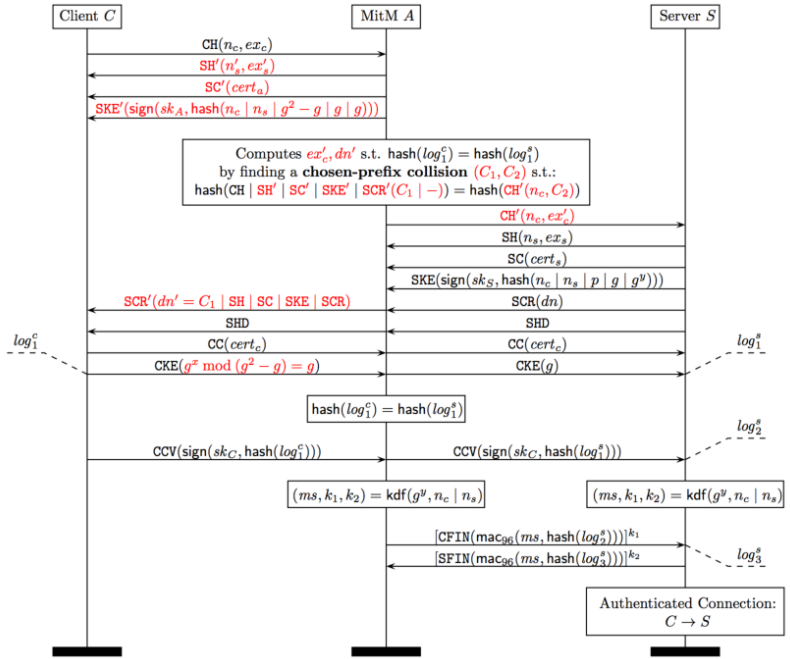
- *SMACK:* Can't get past key exchange or authentication? Just skip the messages!
- *FREAK:* In the 1990s, NSA mandated weak cipher suites for HTTPS so that foreign and civilian communications could be decrypted.
    - Thanks to insecure state transition logic, we can force these cipher suites to be used even in 2015.
    - Expanded with *Logjam*.



Attacks on TLS: Sloth

- RSA-MD5 couples the public key primitive RSA with the outdated hash function MD5, which can now have pre-images obtained with 2^{39} calculations.
- By obtaining targeted pre-images, client authentication can be broken.

Many more attacks on TLS exist: Sweet32, Triple Handshake...



“SLOTH is also a not-so-subtle reference to laziness in the protocol design community with regard to removing legacy cryptographic constructions.”
– *SLOTH paper authors.*

Next time: Usability and Secure Messaging.

1.5