Analysis of Encrypted Communication

Blake Anderson, Andrew Chi, Scott Dunlop, and David McGrew. 2019. Limitless HTTP in an HTTPS World: Inferring the Semantics of the HTTPS Protocol without Decryption. In Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy (CODASPY '19). ACM, New York, NY, USA, 267-278. DOI: https://doi.org/10.1145/3292006.3300025

Outline

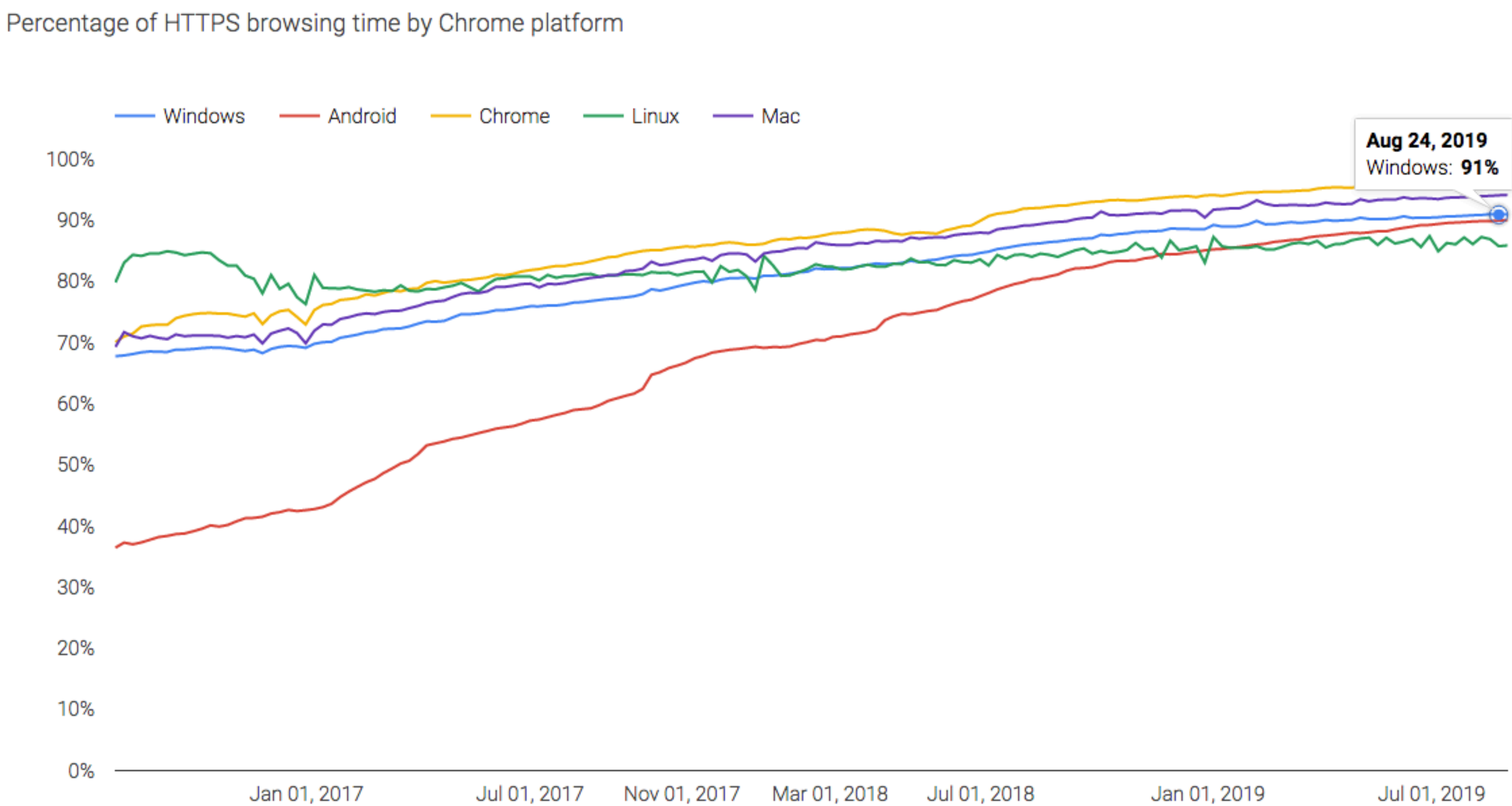
- The Problem
- TLS Evolution
- Inferring HTTPS semantics
- Other possibilities?

Motivation

- Security has become taken seriously these days. Most of the communication on the Internet is protected by TLS.
- This reduces the possibility to apply the usual network forensics approach.
- Is TLS Interception technique the only possibility we have?
- Encrypted communication often uses TLS protocol. Currently, most used is TLS 1.2, but some time we also have version 1.3, which improved over the previous version significantly.

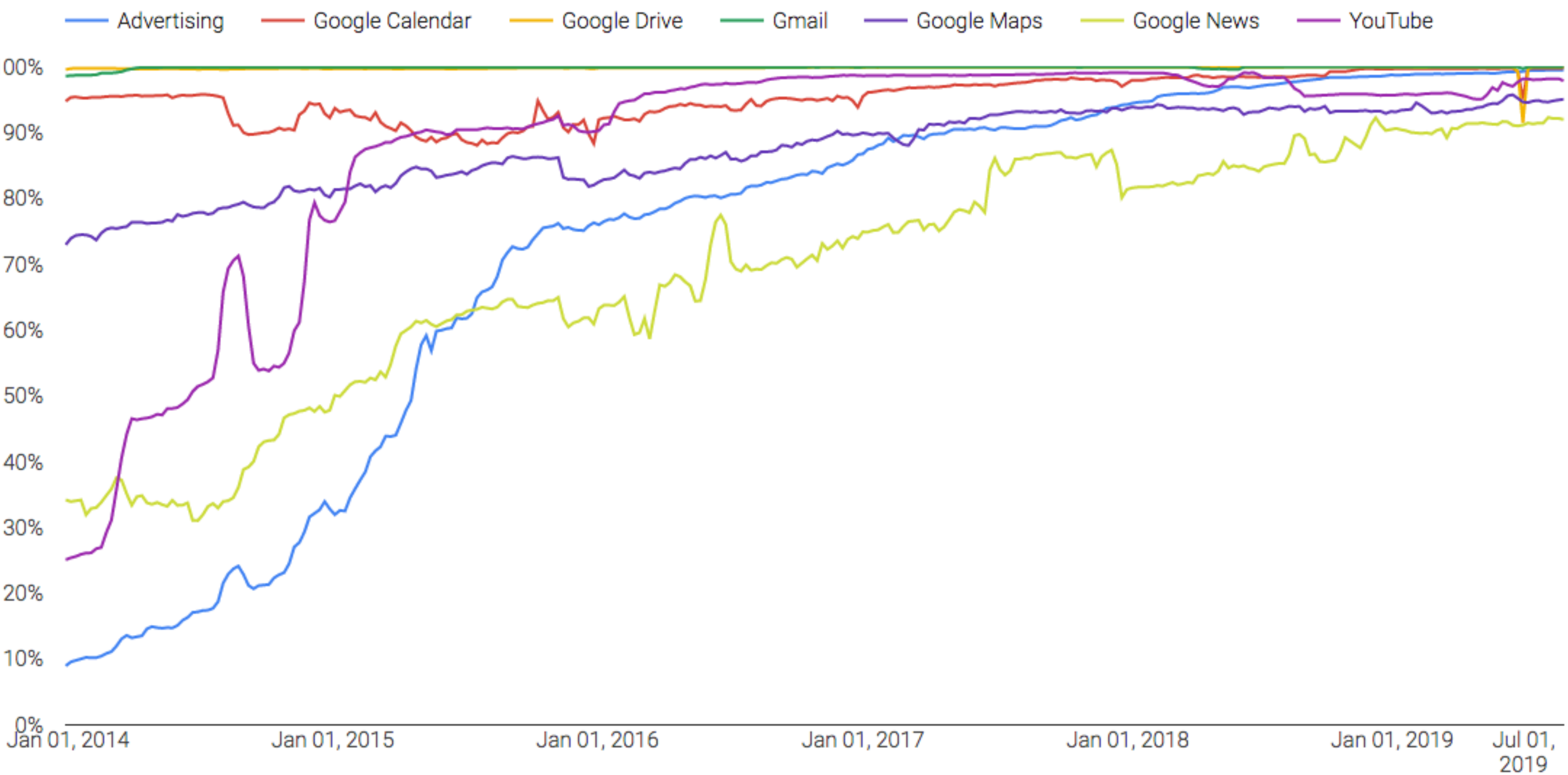
Current situation

- HTTPS protects communication with web servers using encryption technology—SSL or TLS—to secure these connections.
- One of the Google's Transparency reports provides information about the use of encryption in the Internet:
  - Web traffic ~ 96% sites support HTTPS https://transparencyreport.google.com/https/overview
  - Email encryption https://transparencyreport.google.com/safer-email/overview

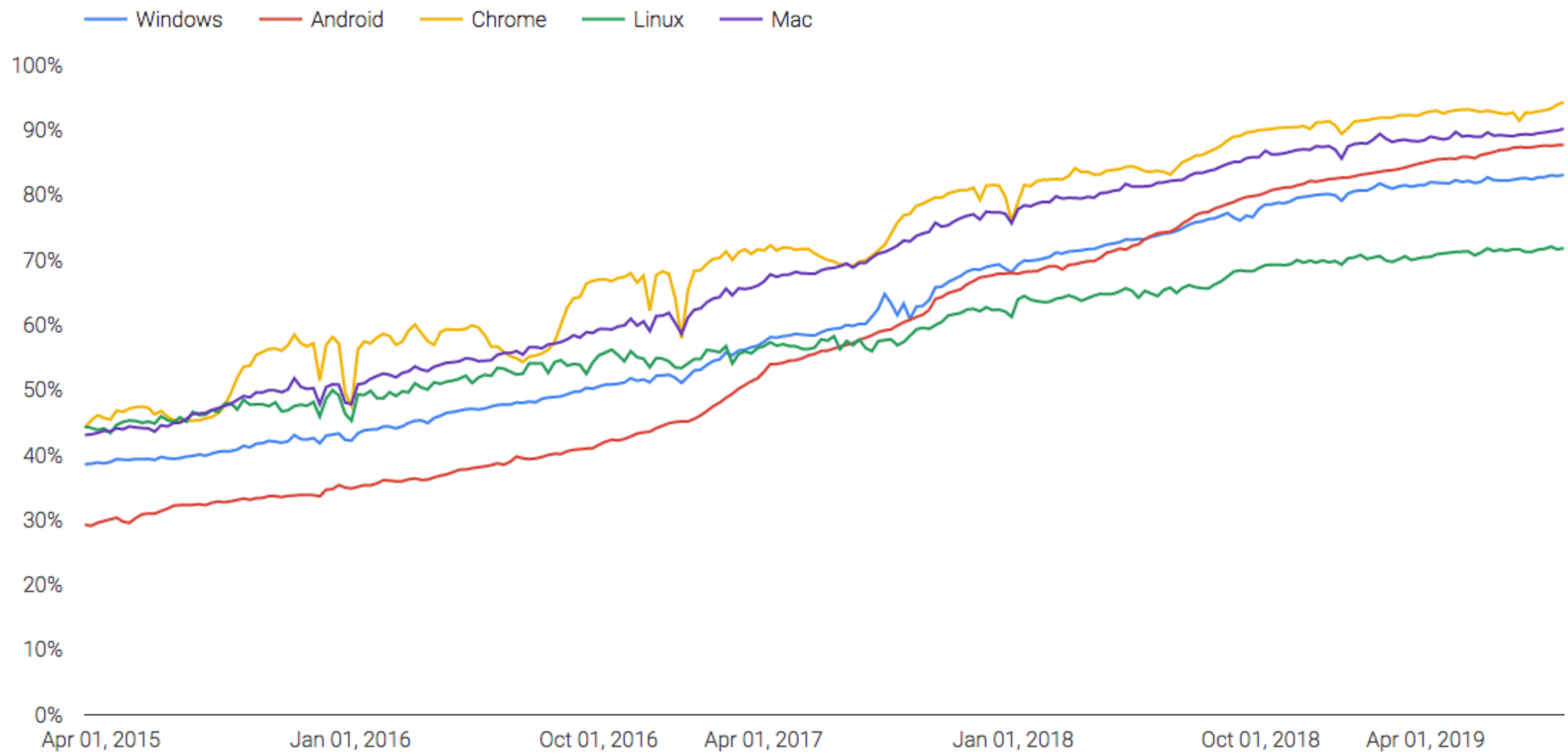

Percentage of HTTPS browsing time by Chrome platform
Windows
Android
Chrome
Linux
Mac
Aug 24, 2019
Windows: 91%
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Jan 01, 2017
Jul 01, 2017
Nov 01, 2017
Mar 01, 2018
Jul 01, 2018
Jan 01, 2019
Jul 01, 2019

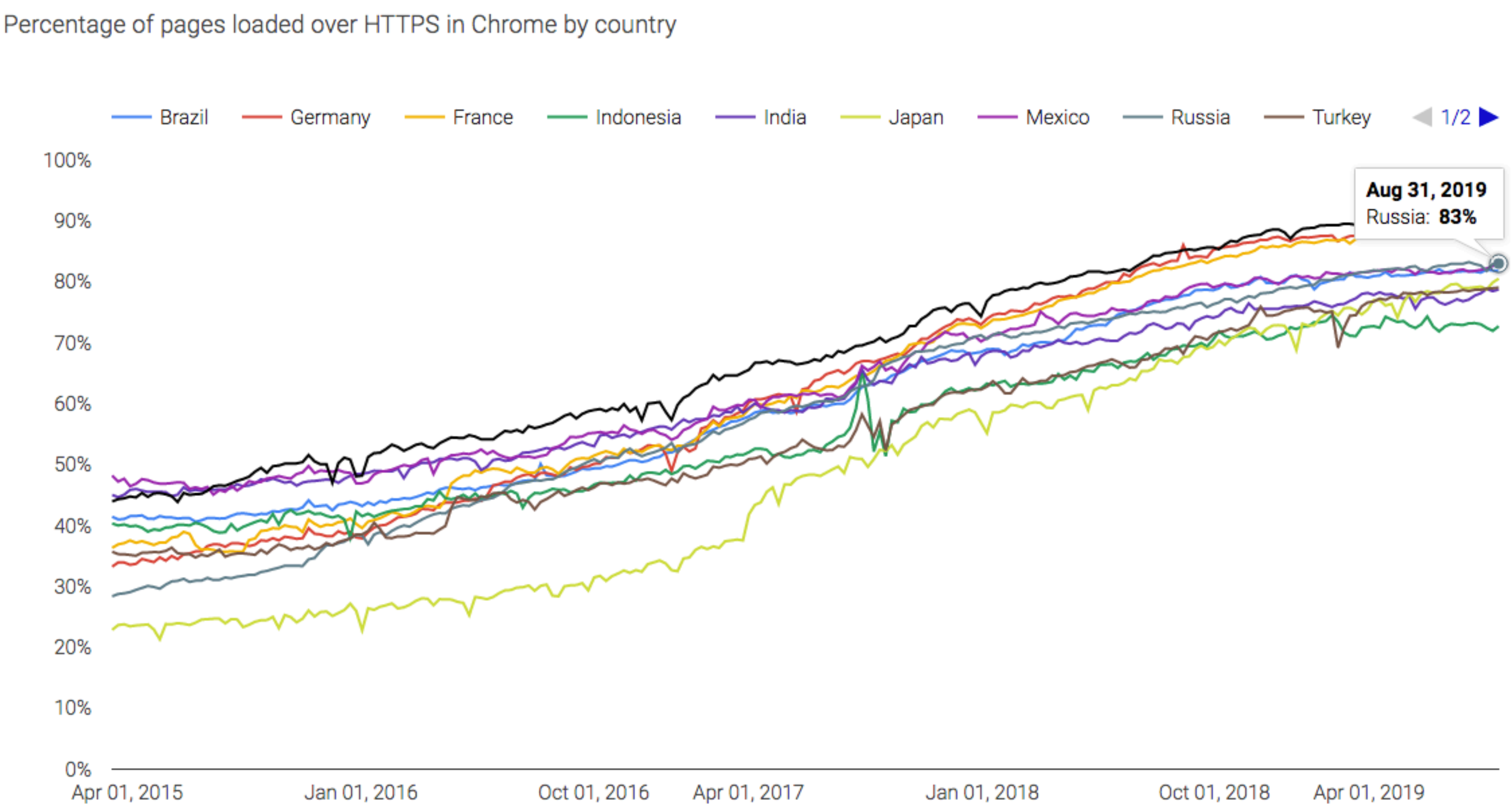
Encryption by product at Google

This chart provides a snapshot of encrypted traffic for several products. Numbers are based on the majority of Google traffic for a given product. We continue to work through the technical barriers that make it difficult to support encryption on some of our products. This chart will change over time to reflect product developments.

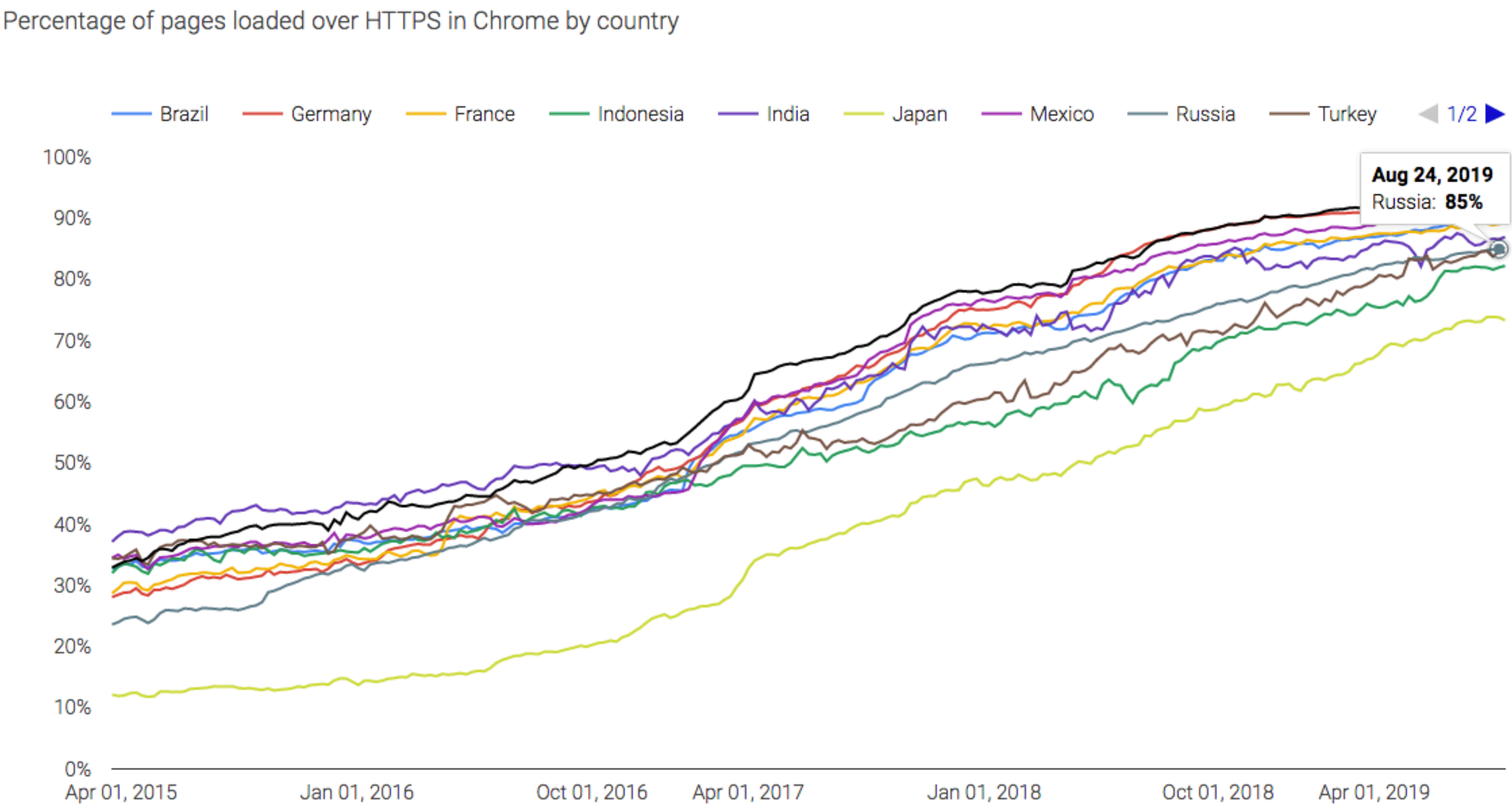


Percentage of pages loaded over HTTPS in Chrome by platform

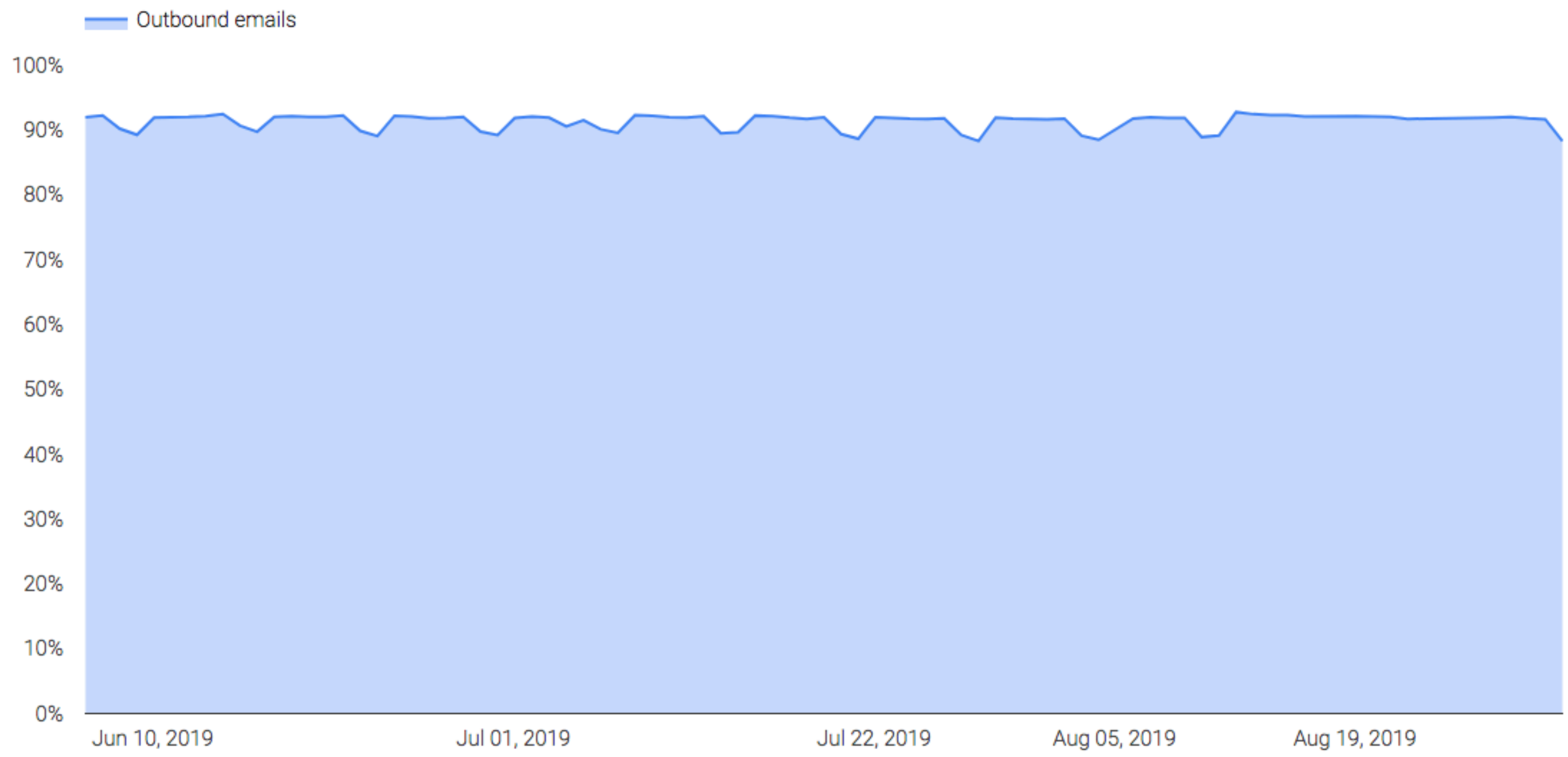


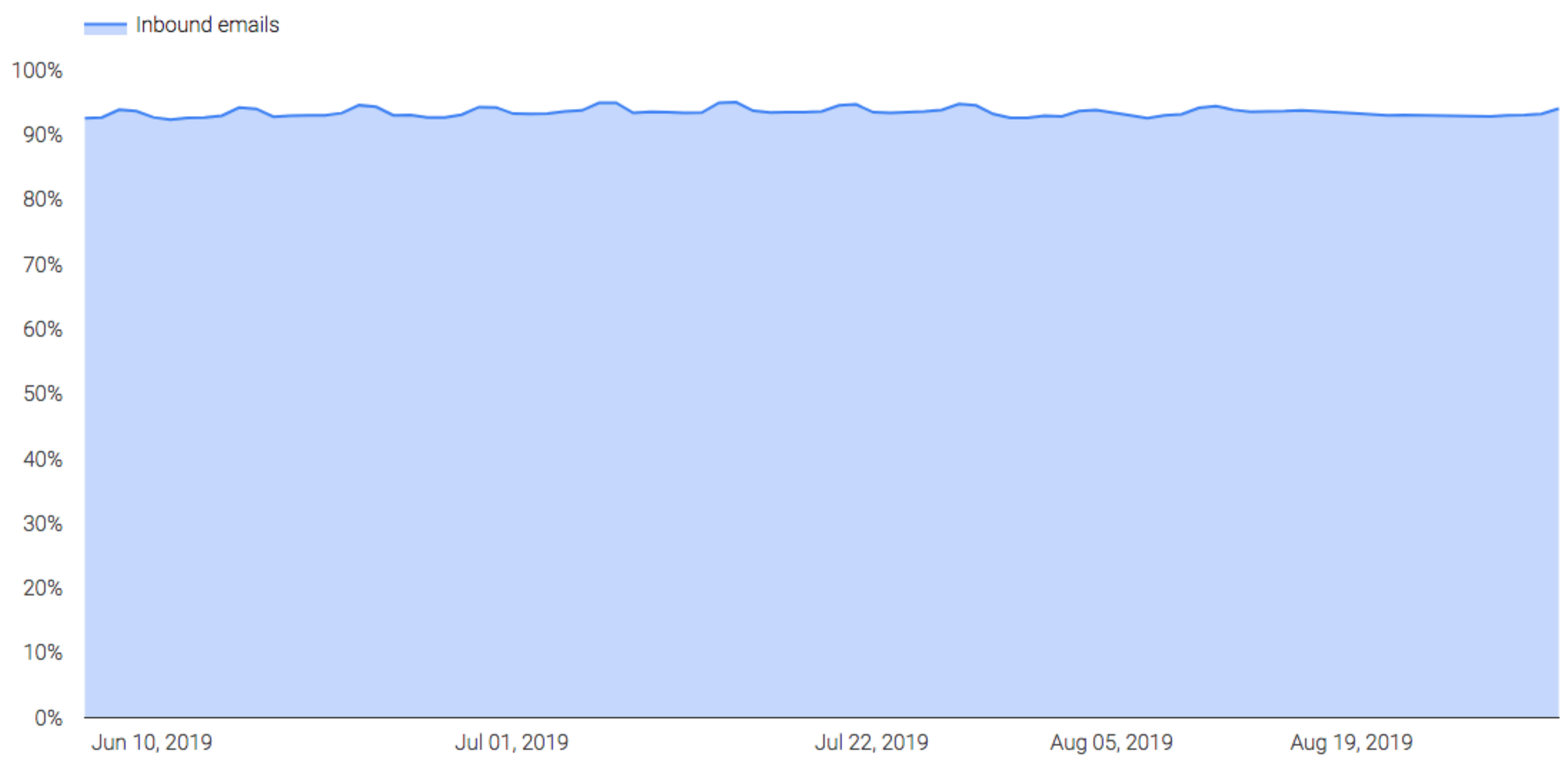

Percentage of pages loaded over HTTPS in Chrome by country
Brazil
Germany
France
Indonesia
India
Japan
Mexico
Russia
Turkey
1/2
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Apr 01, 2015
Jan 01, 2016
Oct 01, 2016
Apr 01, 2017
Jan 01, 2018
Oct 01, 2018
Apr 01, 2019
Aug 31, 2019
Russia: 83%

WINDOWS


Percentage of pages loaded over HTTPS in Chrome by country
Brazil
Germany
France
Indonesia
India
Japan
Mexico
Russia
Turkey
1/2
Aug 24, 2019
Russia: 85%
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Apr 01, 2015
Jan 01, 2016
Oct 01, 2016
Apr 01, 2017
Jan 01, 2018
Oct 01, 2018
Apr 01, 2019

ANDROID

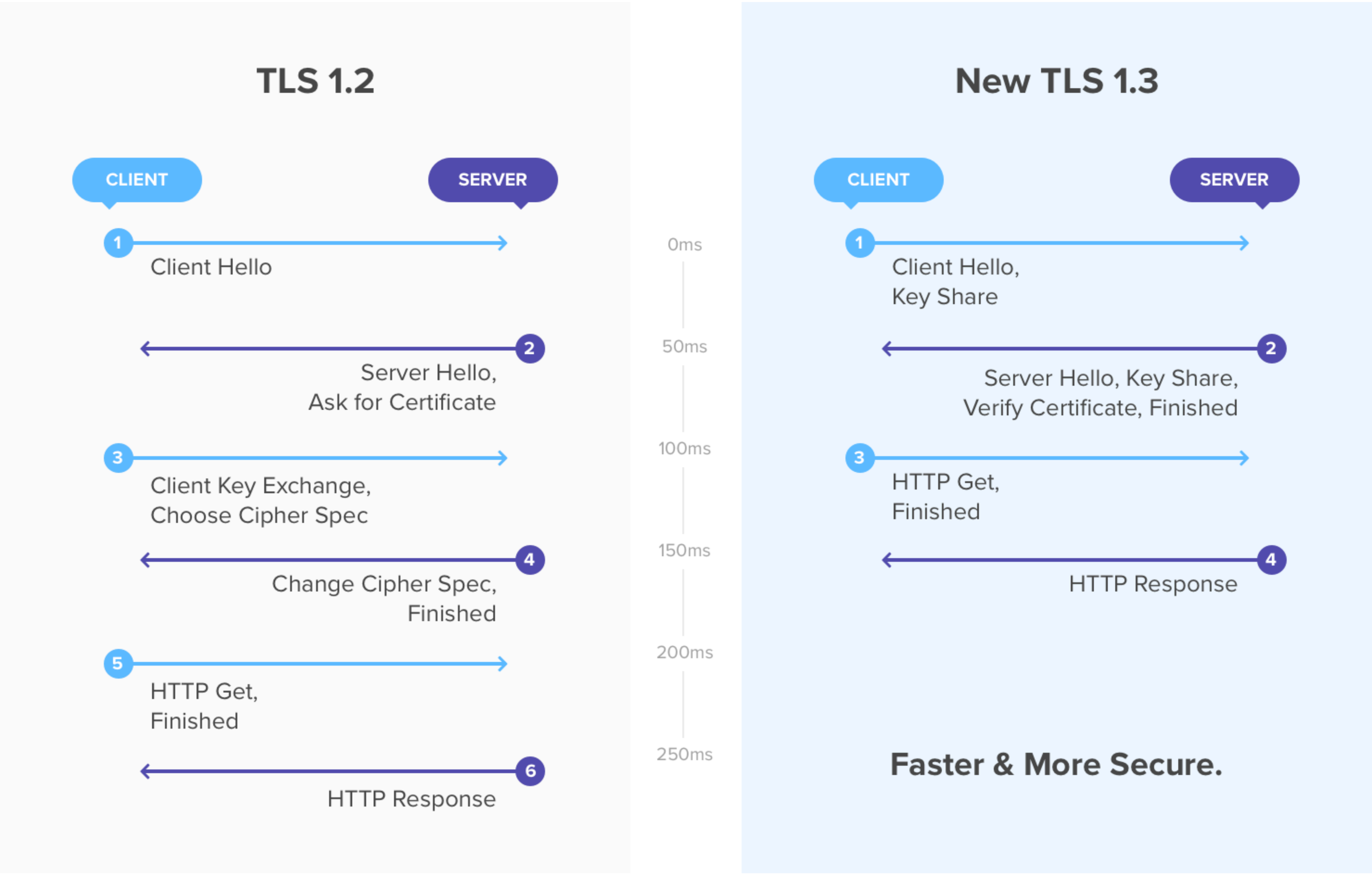

Outbound emails
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Jun 10, 2019
Jul 01, 2019
Jul 22, 2019
Aug 05, 2019
Aug 19, 2019


Inbound emails
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Jun 10, 2019
Jul 01, 2019
Jul 22, 2019
Aug 05, 2019
Aug 19, 2019

TLS

Overview of TLS

- TLS stands for Transport Layer Security and is the successor to SSL (Secure Sockets Layer).
- TLS provides **secure communication** between web browsers and servers. The connection itself is secure because **symmetric cryptography** is used to encrypt the data transmitted.
- The **keys are uniquely generated for each connection** and are based on a shared secret negotiated at the beginning of the session, also known as a **TLS handshake**.
- Many IP-based protocols, such as HTTPS, SMTP, POP3, FTP support TLS to encrypt data.

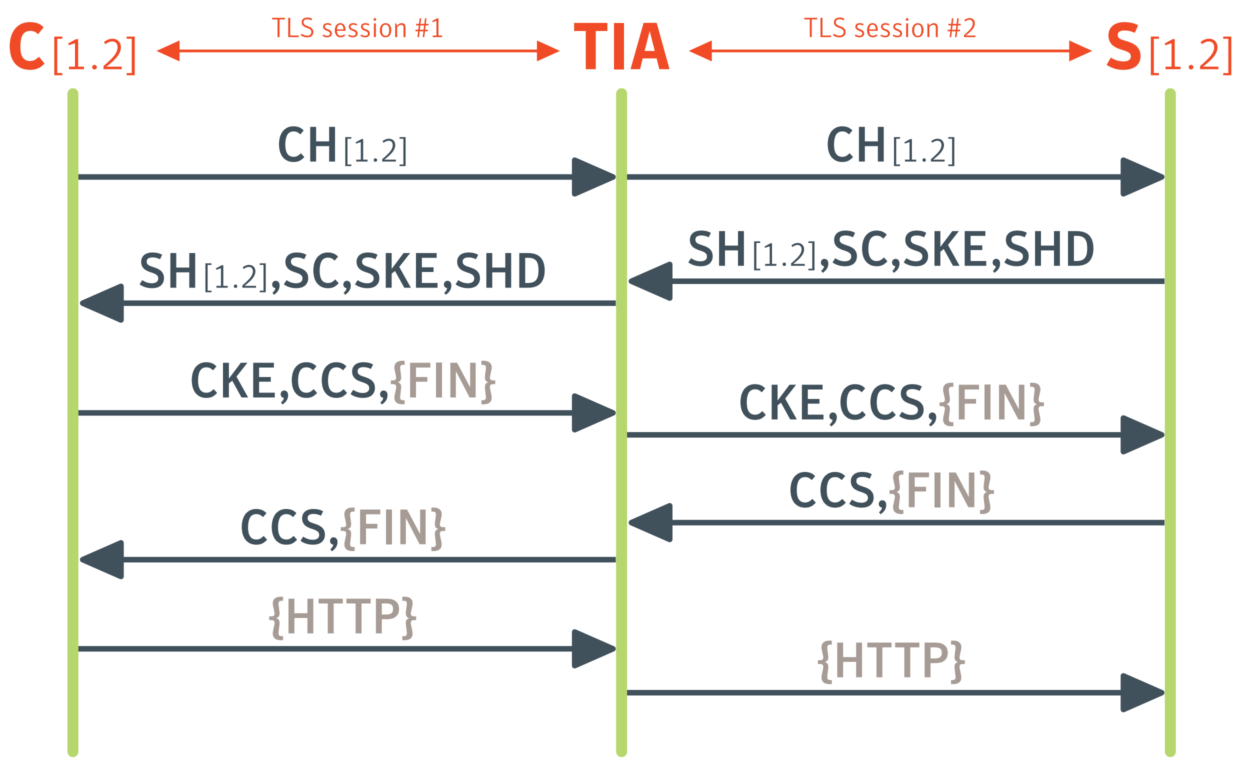

TLS 1.2
CLIENT
SERVER
1
Client Hello
2
Server Hello,
Ask for Certificate
3
Client Key Exchange,
Choose Cipher Spec
4
Change Cipher Spec,
Finished
5
HTTP Get,
Finished
6
HTTP Response
0ms
50ms
100ms
150ms
200ms
250ms
New TLS 1.3
CLIENT
SERVER
1
Client Hello,
Key Share
2
Server Hello, Key Share,
Verify Certificate, Finished
3
HTTP Get,
Finished
4
HTTP Response
Faster & More Secure.

TLS 1.3 Improvements

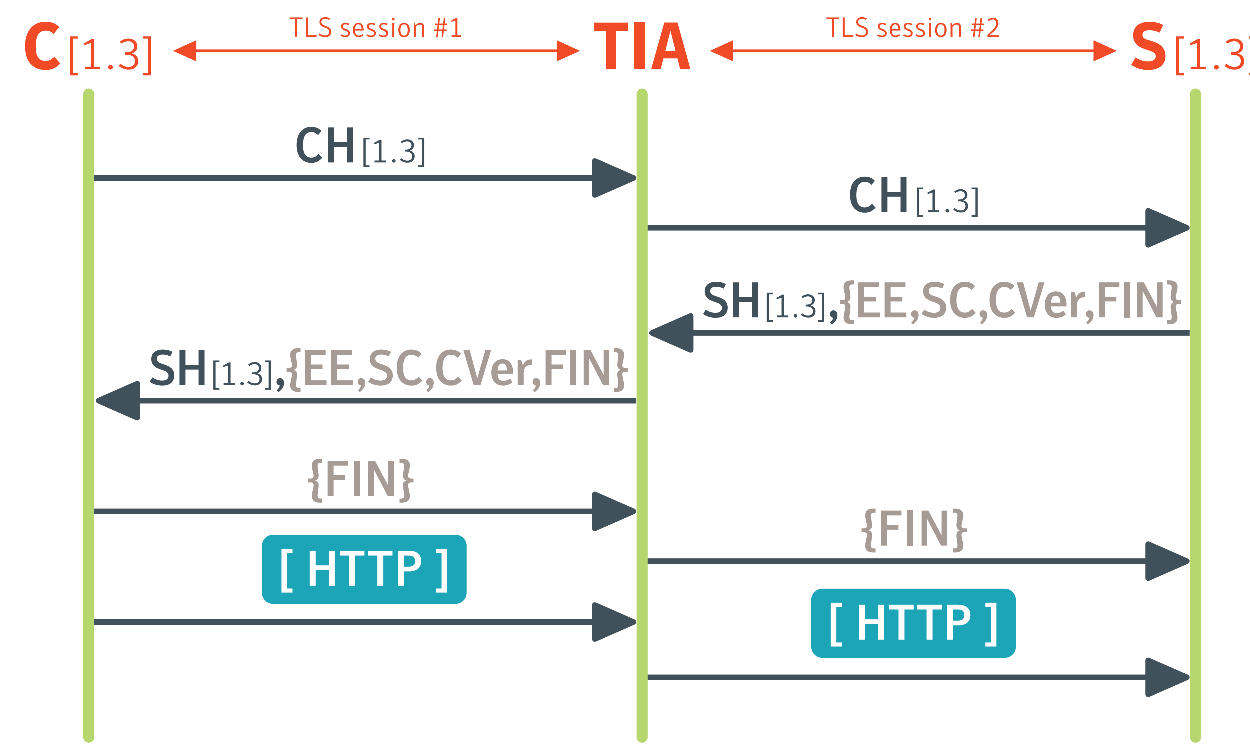
- All cipher suites that do not provide forward secrecy have been eliminated from TLS 1.3.
- TLS 1.3 features a new version downgrade protection to guard against vulnerabilities like POODLE.
- In TLS 1.3, the certificate is encrypted.
- Intercepting TLS 1.3 is more difficult than in TLS 1.2:
  - Passive interception - it is necessary to know the session key (server key is not enough because perfect forward secrecy and deprecation of RSA)
  - Active interception - certificate issues, downgrade detection

TLS Interception

TLS 1.2



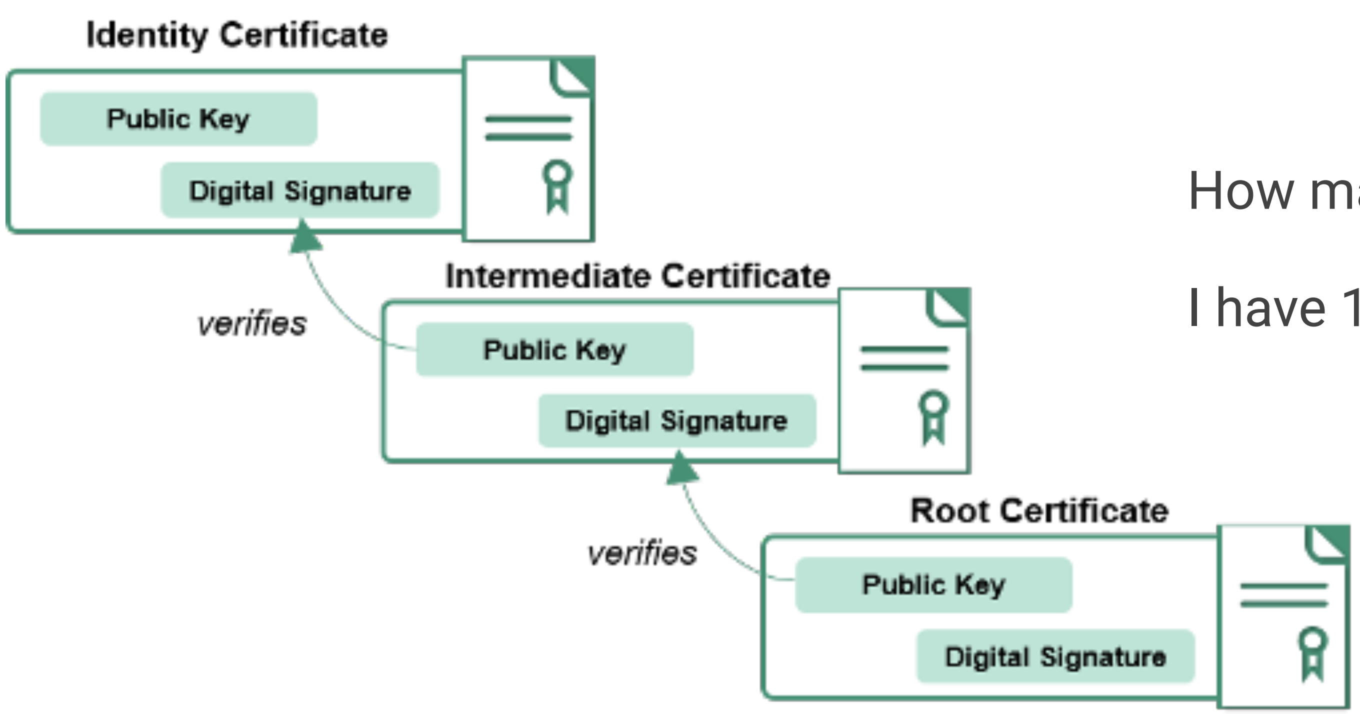
TLS 1.3



https://www.symantec.com/content/dam/symantec/docs/other-resources/responsibly-intercepting-tls-and-the-impact-of-tls-1.3-en.pdf

Certificate Validation (PKI)

- Certificates are digital documents that bind a public key to an individual subject.
- The binding is asserted by having a trusted Certification Authority (CA) verify the identity of prospective certificate owners.
- This trust relationship means that web user security is not absolute; rather, it requires users to trust browsers and CAs to protect their security.
- The security of any CA-based system is based on many links and they're not all cryptographic. **People are involved.**
- What if people are less involved, e.g., Let's Encrypt?

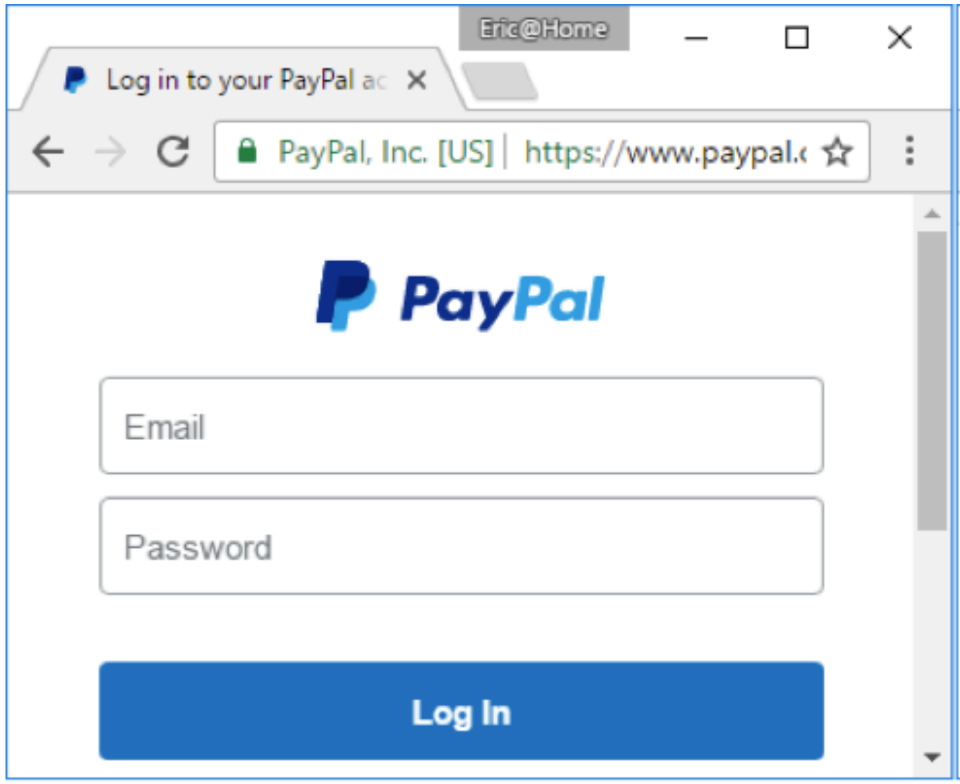


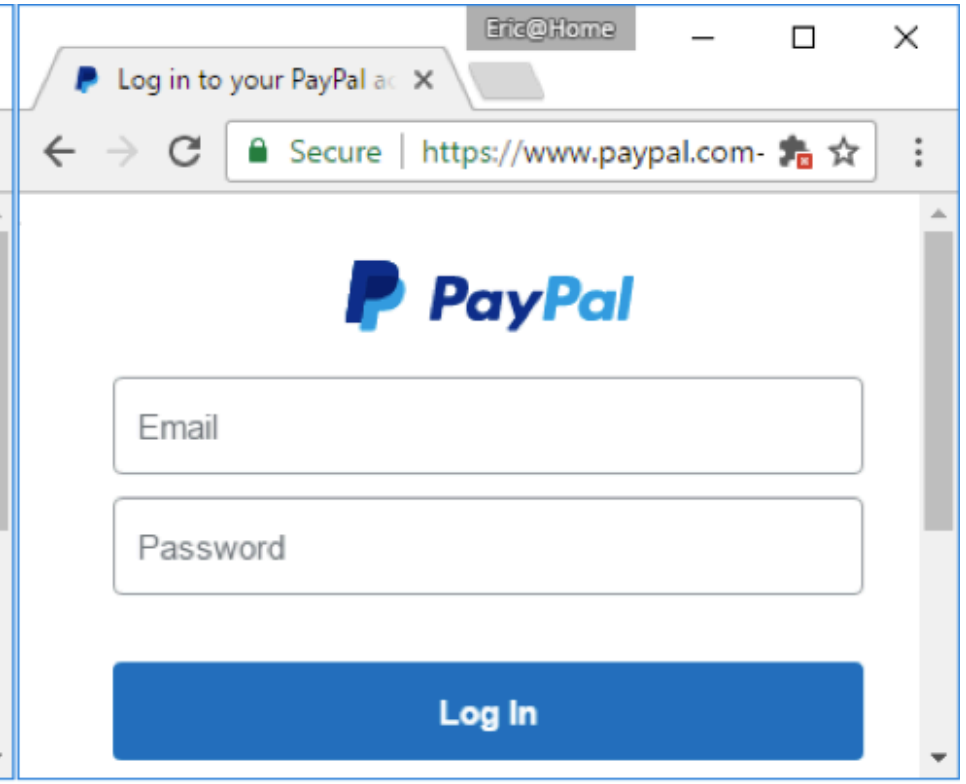
How many certificates do you have installed in your OS?

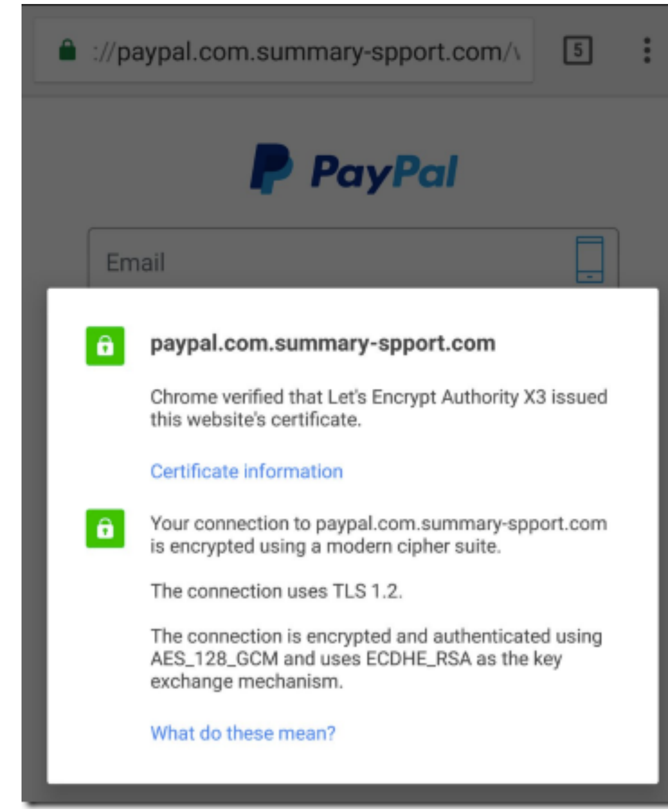
I have 162 "system" CA certificates.

https://www.ssl.com/article/browsers-and-certificate-validation/

https://www.schneier.com/academic/paperfiles/paper-pki.pdf







https://transparencyreport.google.com/https/certificates/bLEAIT34F5LrzjsqdVJPr3ewyF0coHU5shcIEyHK330%3D

By December 8, 2016, LetsEncrypt had issued 409 certificates containing "Paypal" in the hostname

Current counter measures

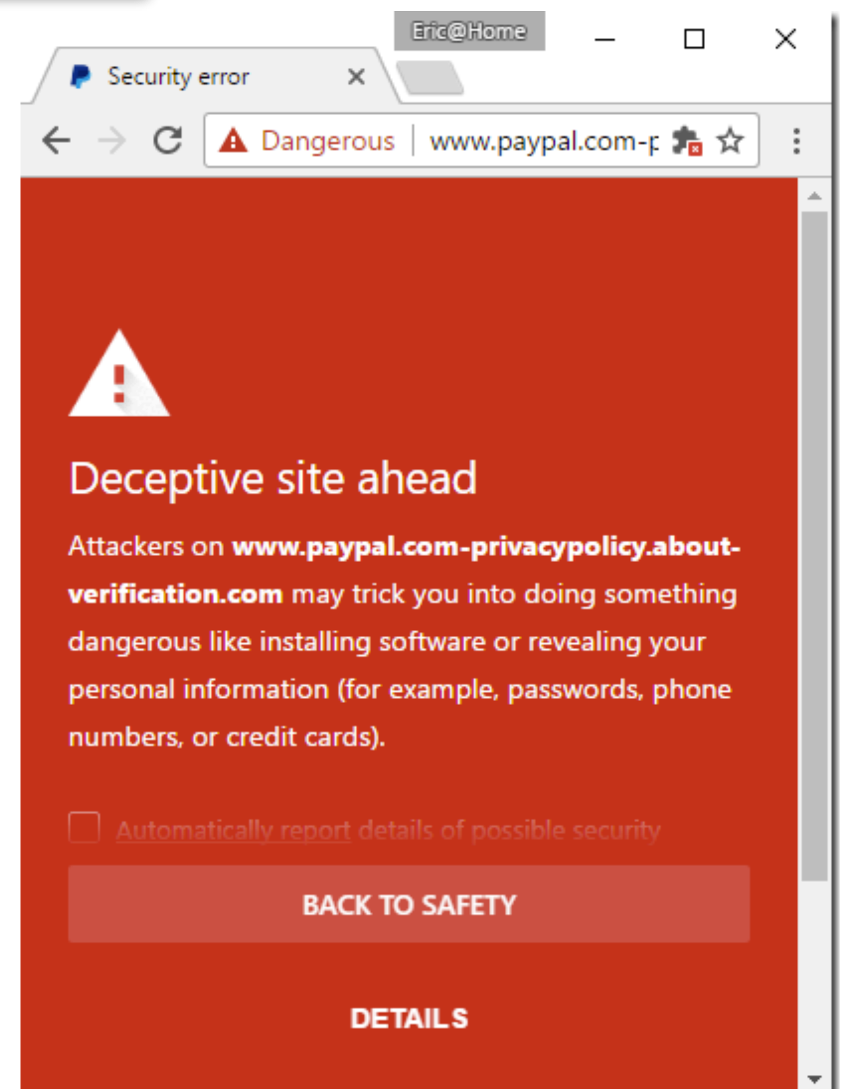
In browsers, we have a well-established concept of bad reputation (your site or download appears on a block list)

https://transparencyreport.google.com/safe-browsing/overview

https://transparencyreport.google.com/https/certificates

https://textslashplain.com/2017/01/16/certified-malice/

https://securityboulevard.com/2019/01/lets-encrypt-are-enabling-the-bad-guys-and-why-they-should/



What is SSL pinning?

When mobile apps communicate with a server, they typically use SSL to protect the transmitted data against eavesdropping and tampering. By default, SSL implementations used in apps trust any server with certificate trusted by the operating system's trust store. This store is a list of certificate authorities that is shipped with the operating system.

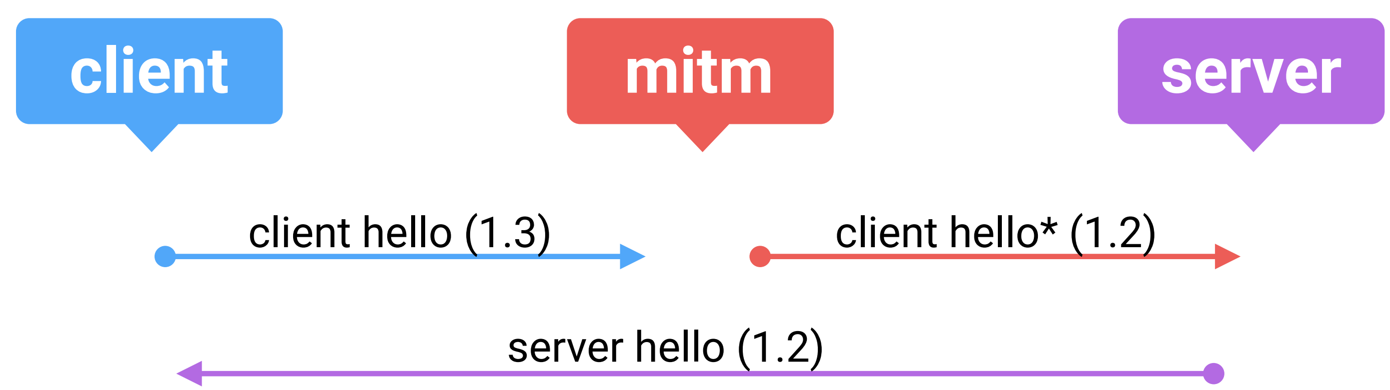
With SSL pinning, however, the application is configured to reject all but one or a few predefined certificates. Whenever the application connects to a server, it compares the server certificate with the pinned certificate(s). If and only if they match, the server is trusted and the SSL connection is established.

https://labs.nettitude.com/tutorials/tls-certificate-pinning-101/

TLS Downgrade Protection

- The ClientHello message includes a list of supported protocol versions.
- TLS 1.3 changes the way in which version negotiation is performed as a protection against downgrade attacks.

▼ Extension: supported_versions (len=11)
Type: supported_versions (43)
Length: 11
Supported Versions length: 10
Supported Version: Unknown (0xcaca)
~~Supported Version: TLS 1.3 (0x0304)~~
Supported Version: TLS 1.2 (0x0303)
Supported Version: TLS 1.1 (0x0302)
Supported Version: TLS 1.0 (0x0301)



`ServerHello.random`

`RANDOM value + 44 4F 57 4E 47 52 44 01`

https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2019/february/downgrade-attack-on-tls-1.3-and-vulnerabilities-in-major-tls-libraries/

Summary

- TLS 1.3 removes some problematic cipher suites
- TLS 1.3 does not use RSA thus it is not enough to obtain server private key to decrypt any communication.
  - Passive MITM limited - need to obtain session key.
  - Active MITM - similar as in TLS 1.2
- TLS 1.3 hides more information useful for identification of the connection (certificate)
- Certificate pinning as a method to avoid active MITM.
  - Not relying on PKI, certificates hardwired in applications, not suitable for every application.

Inferring HTTPS Semantics