Network Security and Measurement

- Transport Security and Certification -

Prof. Dr. Thomas Schmidt

http://inet.haw-hamburg.de | t.schmidt@haw-hamburg.de

Agenda

Motivation and Idea

Transport Layer Security

Perfect Forward Secrecy

TLS 1.3

Certification: DANE

Certificate Transparency

Why Security on Transport?

MOTIVATION AND IDEA

Security on the Transport Layer

Initial concept developed by Netscape to build HTTPS

Authentication and encryption between applications

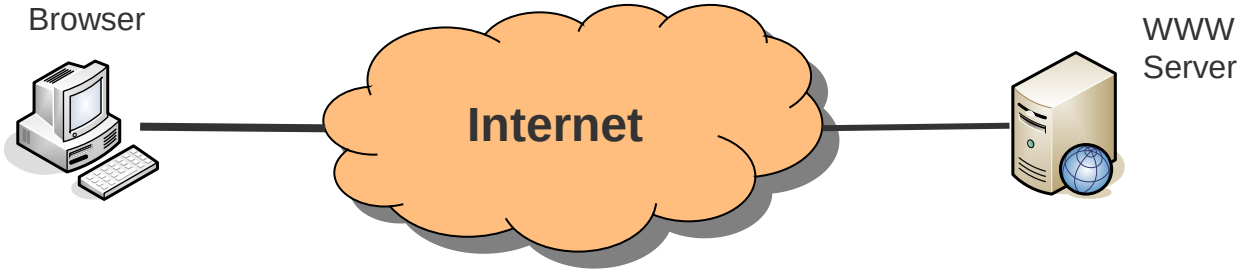
- Transport operates end-to-end

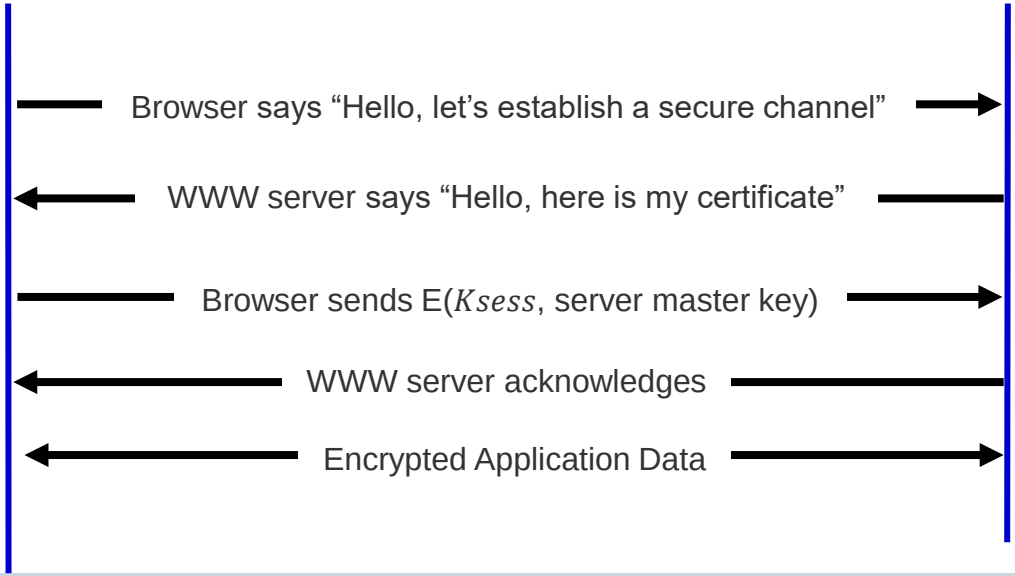
Establish a secure communication channel between unknown client and known server

- No pre-established keys nor trust

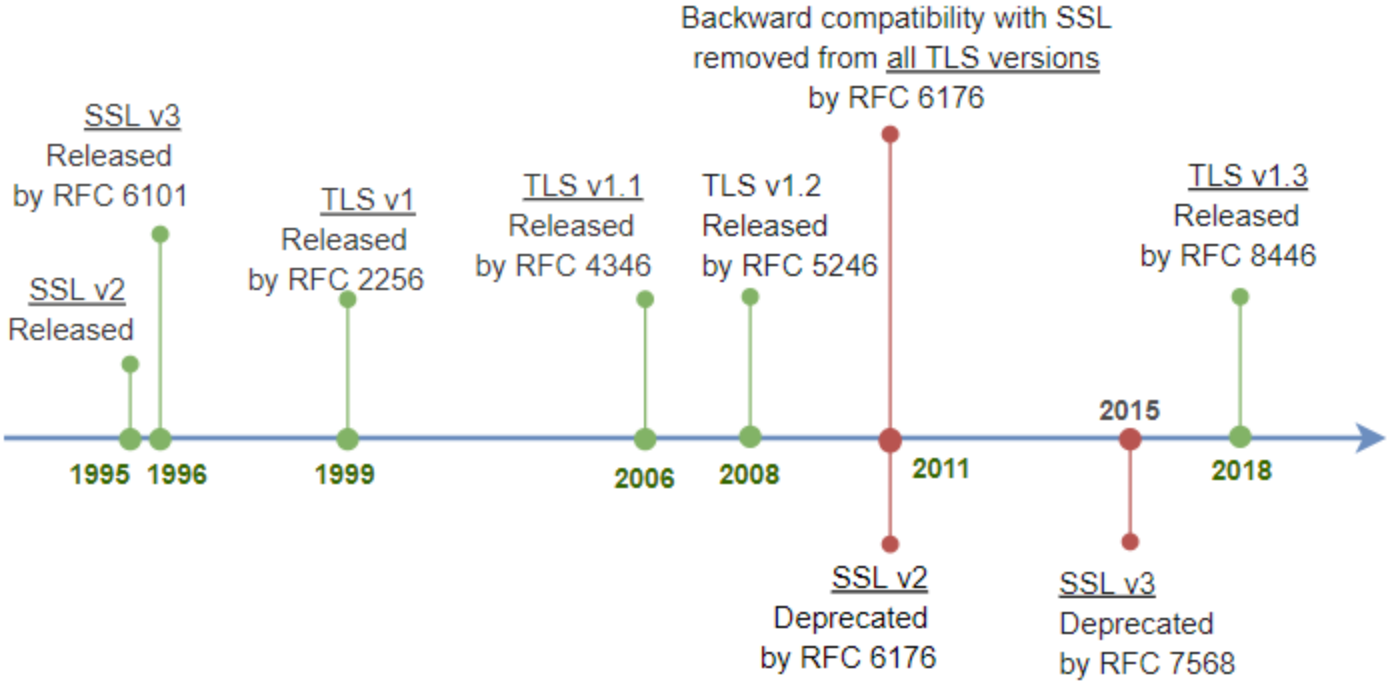
Trust infrastructure: DNS, Certificate Authorities (CAs)

Basic Idea


Browser
Internet
WWW
Server


Browser says “Hello, let’s establish a secure channel”
WWW server says “Hello, here is my certificate”
Browser sends E($Ksess$, server master key)
WWW server acknowledges
Encrypted Application Data

SSL/TLS Timeline



Source: „Ravi“: Making Sense of SSL/TLS

HAW HAMBURG

Key Concepts

TRANSPORT LAYER SECURITY

TLS Key Functions

Clients connect to a known server

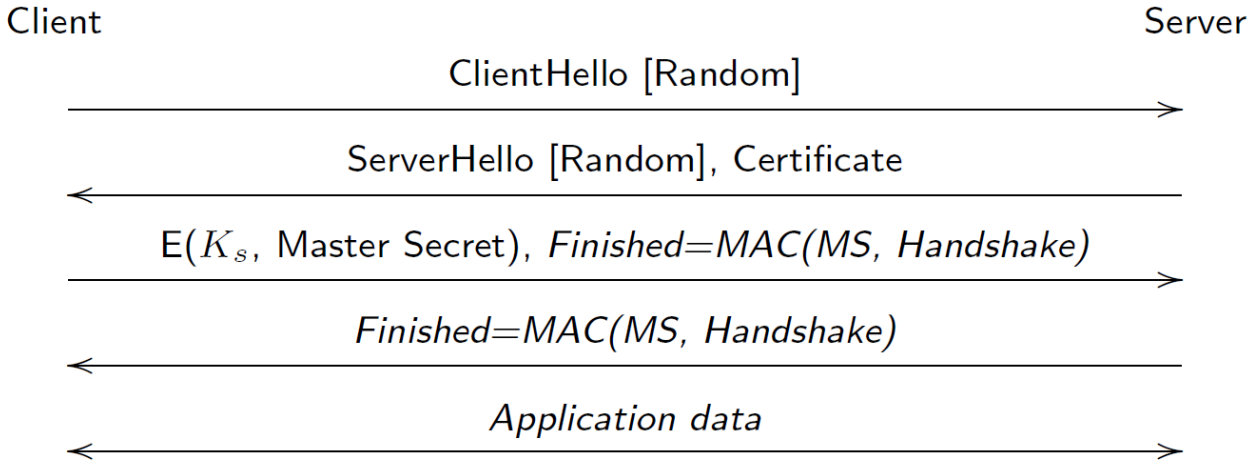
Server is authenticated by TLS via a certificate

Client may or may not be authenticated by TLS

- Clients can authenticate via the application

After channel set-up, data is encrypted and authenticated

TLS 1.2 Base Handshake



Source of Diagram: Eric Rescorla: TLS1.3 (Stanford)

Trust Derived from Public Key Infrastructure

Authentication in TLS relies on certificates

- Issued by a Certification Authority (CA)
- CA authorized in a trust chain w/ trusted root

Certificates are signed by the CA and contain

- ID of the issuer (the CA)
- ID of the certified subject
- Public key of the subject
- Further meta-information

Trust Derived from Public Key Infrastructure

Digital Certificates are generally defined in the ITU X.509 standard

Profiles for use as Internet PKI are specified in RFC 5280 + updates

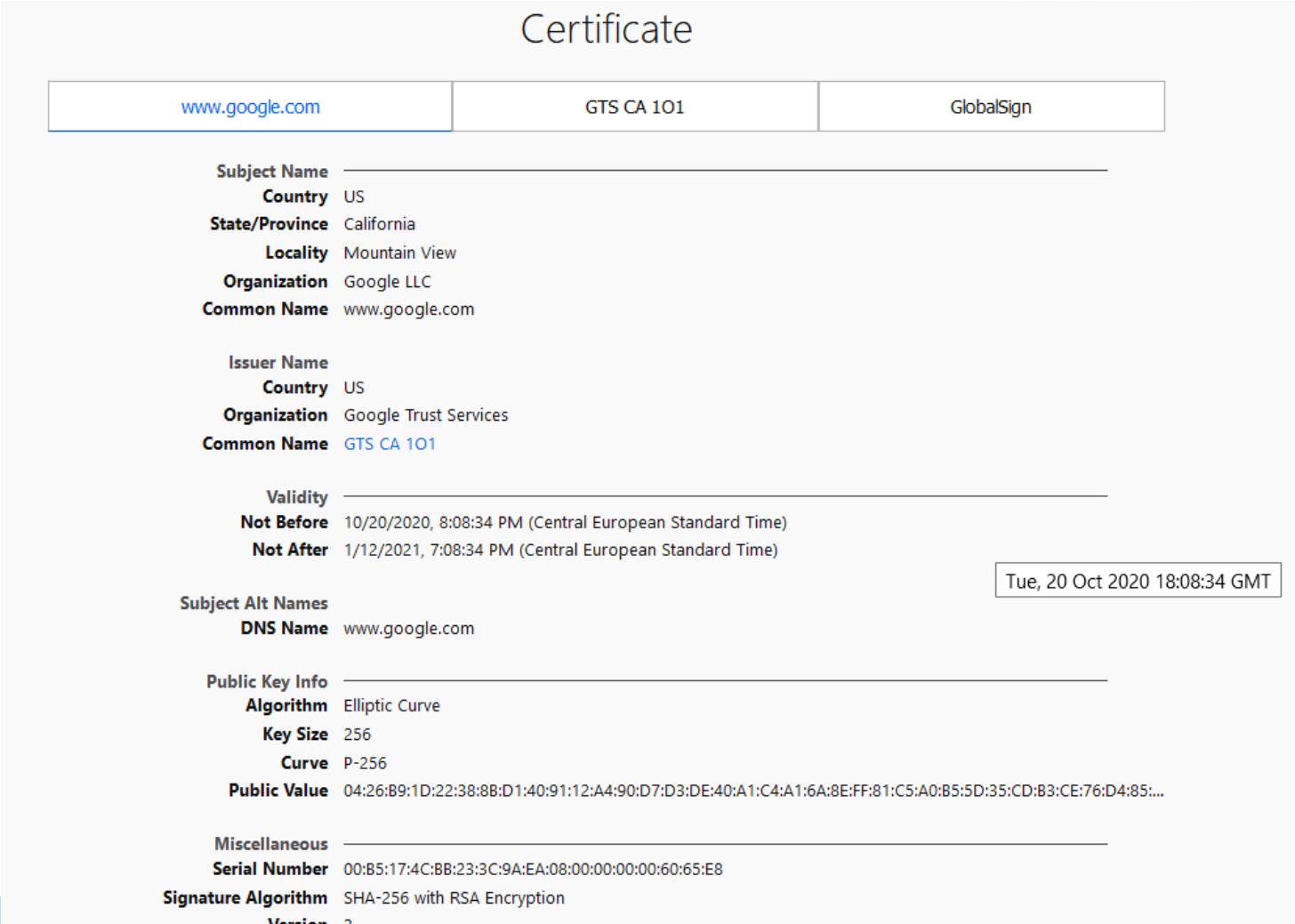
Authentication in TLS relies on certificates

- Issued by a Certification Authority (CA)
- CA authorized in a trust chain w/ trusted root

Certificates are signed by the CA and contain

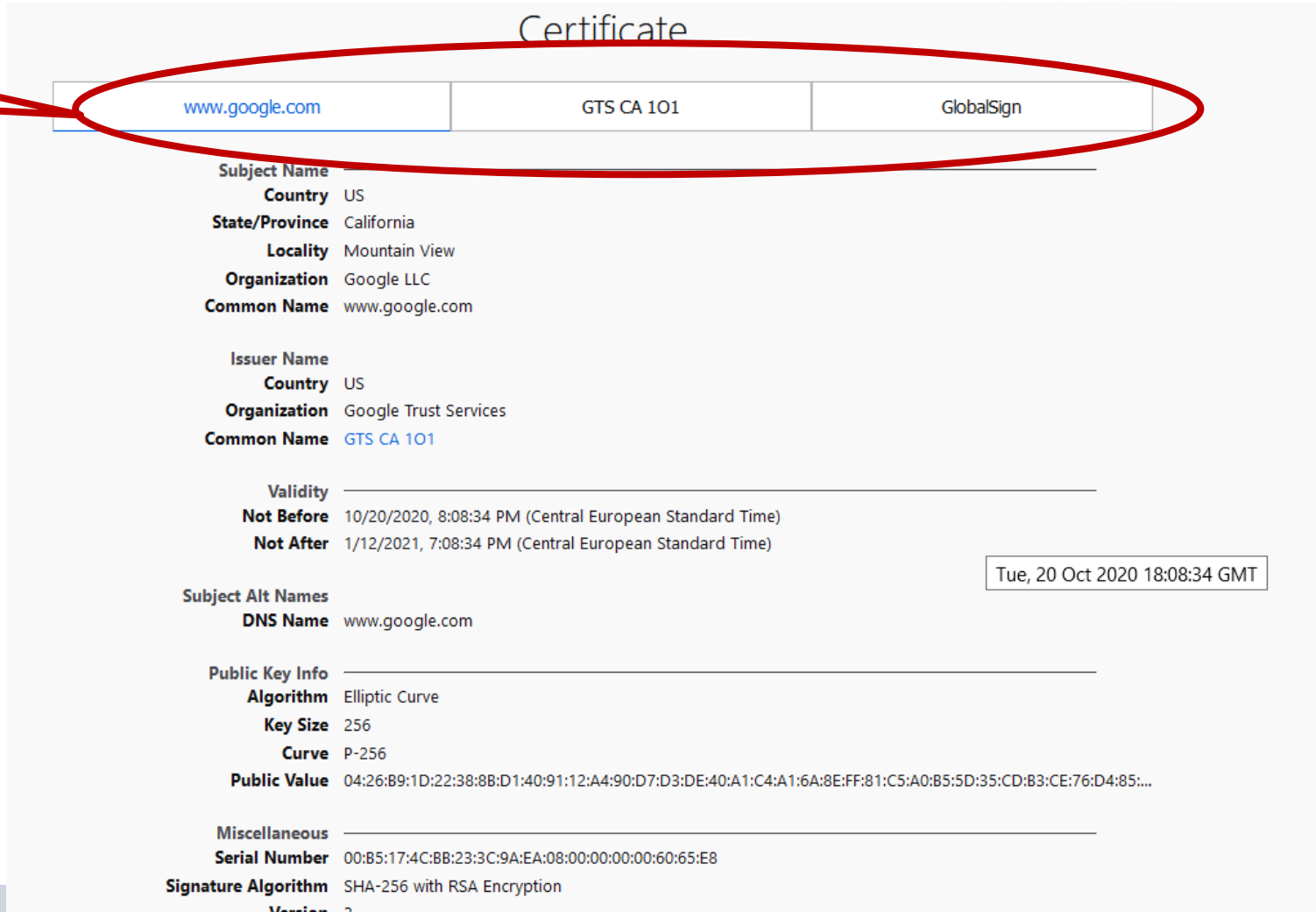
- ID of the issuer (the CA)
- ID of the certified subject
- Public key of the subject
- Further meta-information

Certificate from the Browser



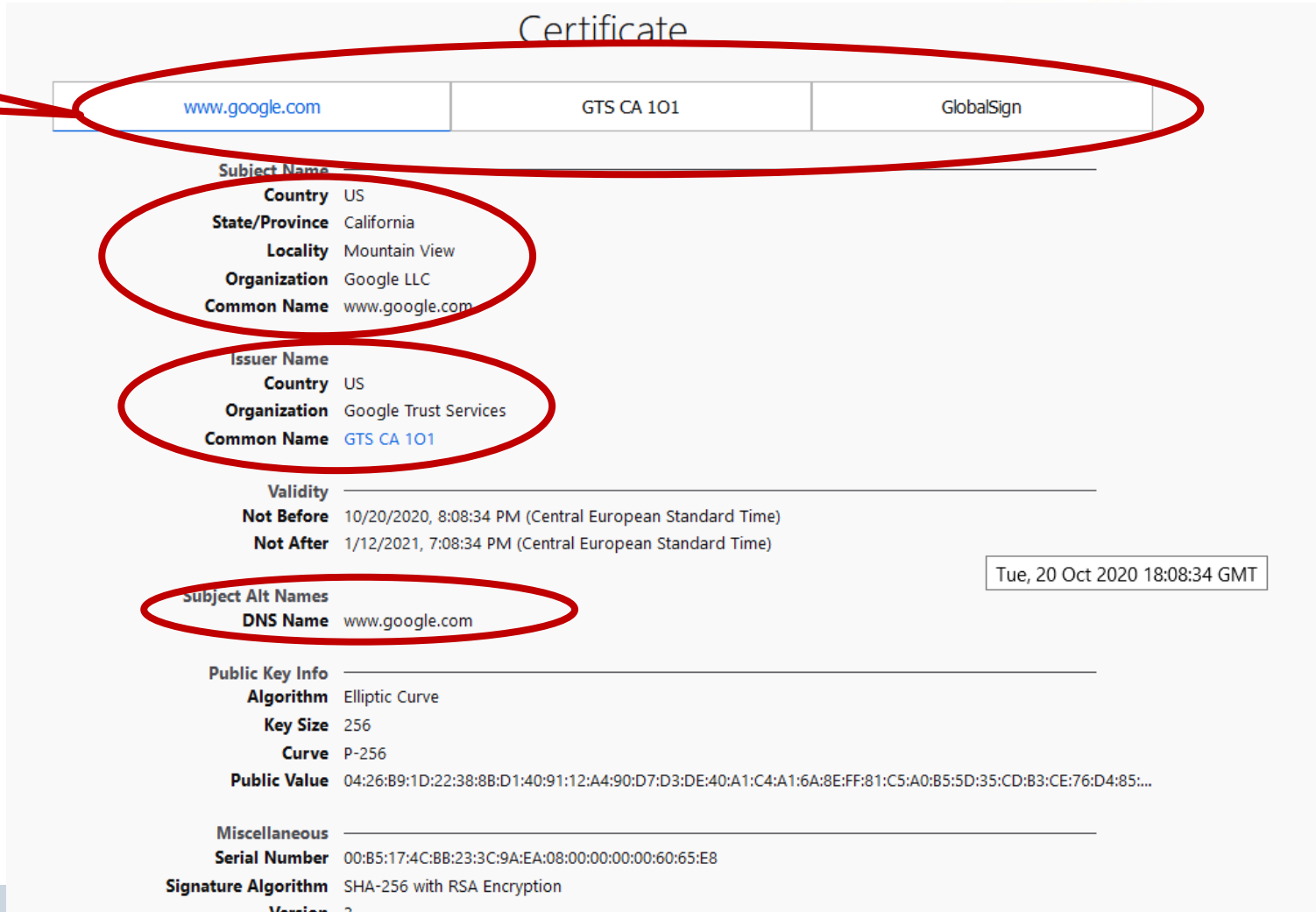
Certificate from the Browser

Trust Chain



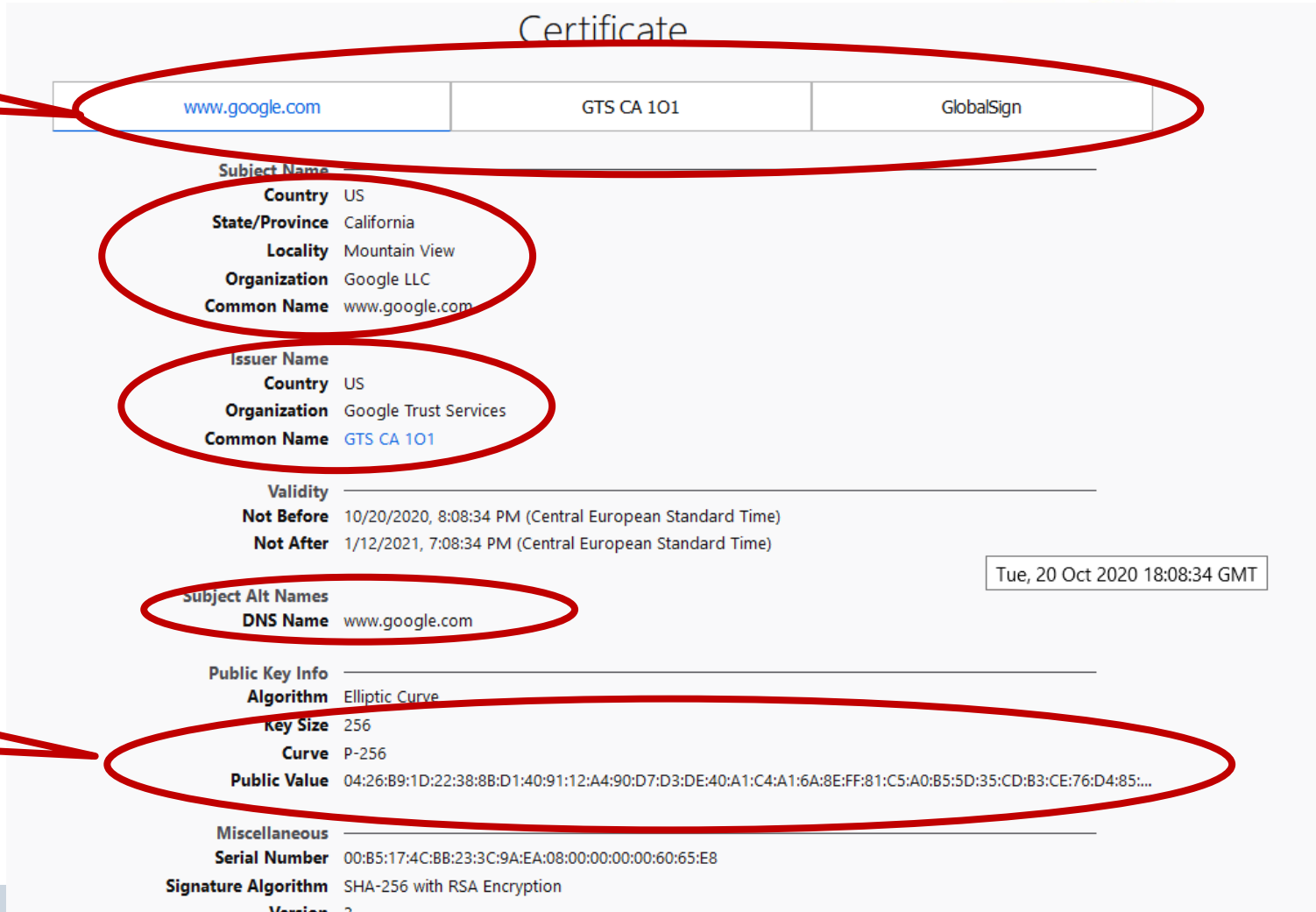
Certificate from the Browser

Trust Chain



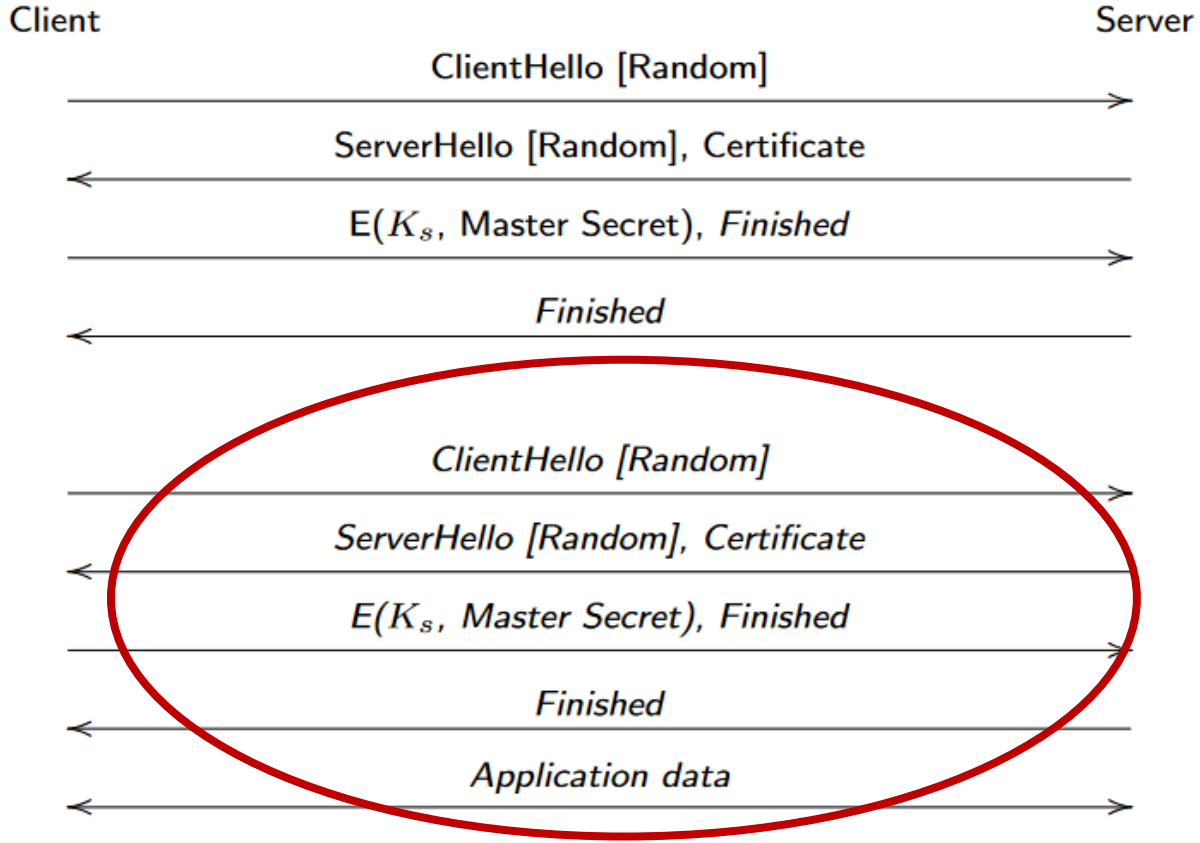
Certificate from the Browser

Trust Chain



Public Key

TLS 1.2 Base Renegotiation



DTLS

Transfers TLS (1.2) to UDP Transport

Adds stateful security contexts to channels

Defines a reliable security handshake incl. retransmissions

Bans stream cyphers to allow decryption of individual packets, adds sequence numbers

Provides replay detection by bitmap window

TLS 1.2 Shortcomings

High negotiation overheads (2 RTTs)

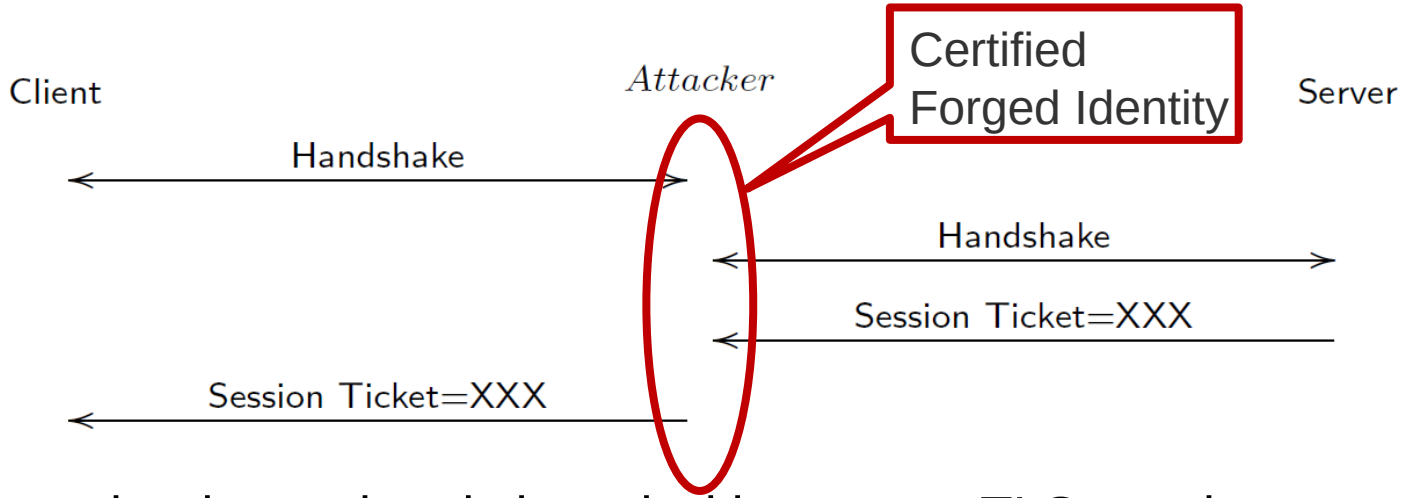
Supports insecure and outdated cyphers

Allows recovery of data after key compromise

Susceptible to Man-in-the-Middle attacks

Simple Man-in-the-Middle Attack

An attacker, who can present a ‘valid’ certificate to the client, can silently intercept a TLS session



If a static cypher is used and shared with a proxy, TLS sessions can be silently intercepted by this middlebox

Enhancing Robustness

PERFECT FORWARD SECRECY

What If a Server Key Gets Compromised?

An Attacker, who has captured the communication flows, can decipher all data after server key compromise

Servers persist a permanent private signing key
- Key renewal requires CA attestation

Server key is used for authentication
- Authentication remains valid until keys get unsealed

Server keys have been used for key exchange
- New session key encrypted with server key
- Session keys are disclosed after server key gets unsealed

Diffie-Hellman Key Agreement

Diffie, W., Hellman, M.: "*New Directions in Cryptography*"

Transactions on Information Theory (1976)

Problem: Two mutually unknown parties (A & B) want to exchange an encryption key via a public data channel

Approach: Public key cryptography applied to establishing a shared secret key

Potential: Key establishment is spontaneous – independent of any previous secret

Limitation: Mutual authentication left open - to public key infrastructure or off-channel solution

Diffie-Hellman Algorithm

Let p be a sufficiently large prime,

$$g : g^n \bmod p = p \text{ for some } n,$$

p and g publicly available.

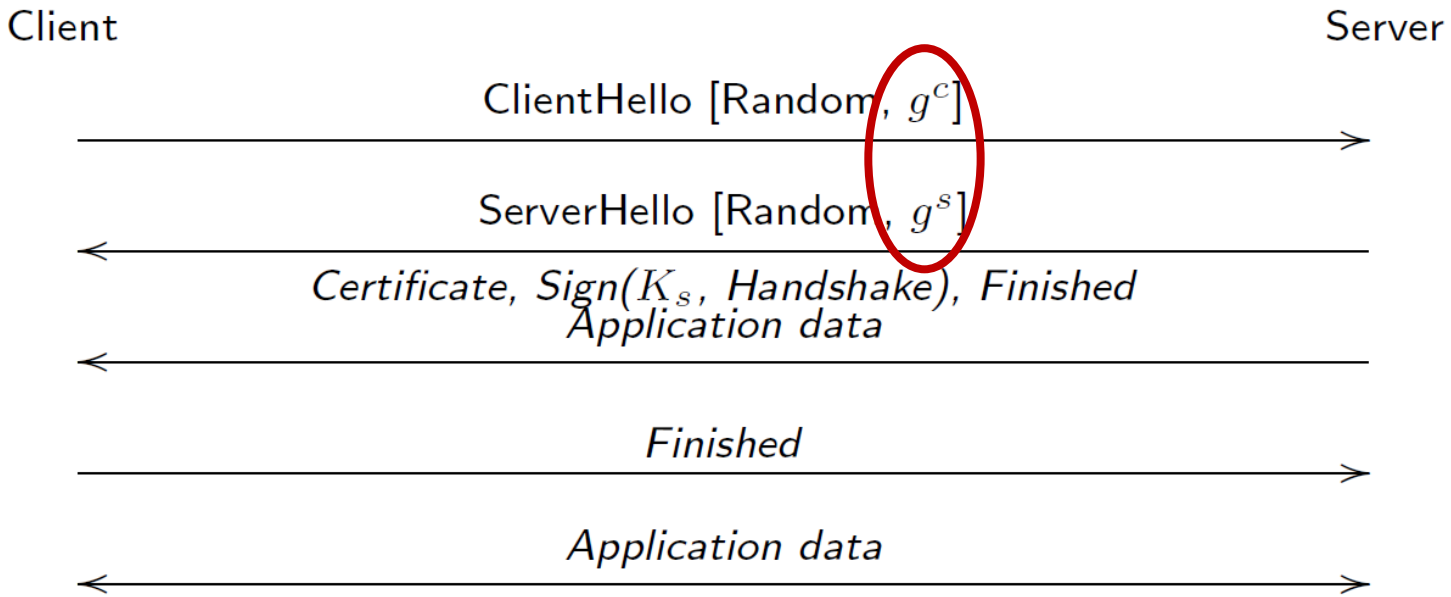
Then:

1. A chooses $0 \leq a \leq p-2$ at random and sends $c := g^a$ to B
2. B chooses $0 \leq b \leq p-2$ at random and sends $d := g^b$ to A
3. A computes the shared key $k = d^a = (g^b)^a$
4. B computes the shared key $k = c^b = (g^a)^b$

The strength of the algorithm relies on the secrets a and b.

a and b are discrete logarithms $\bmod p$

TLS 1.3 with Diffie-Hellman Key Exchange



TLS with Perfect Forward Secrecy

Session key exchange by ephemeral Diffie-Hellman key establishment (EDHE)

Assures that session keys remain secret even if long-term server keys are compromised

Same for key renegotiation

Server private signing key only used for authentication

Refurbished Transport Layer Security

TLS 1.3

The 1.3 Race for Redesigning TLS

TLS 1.3 efforts started
in 2013 and
ended in Aug. 2018
with RFC 8446

Clean up and discard insecure elements

Improve performance

Improve security by state-of-the art techniques

Implement perfect forward secrecy

Encrypt more of the protocol for privacy

Make a clear case against interception

Session Keys

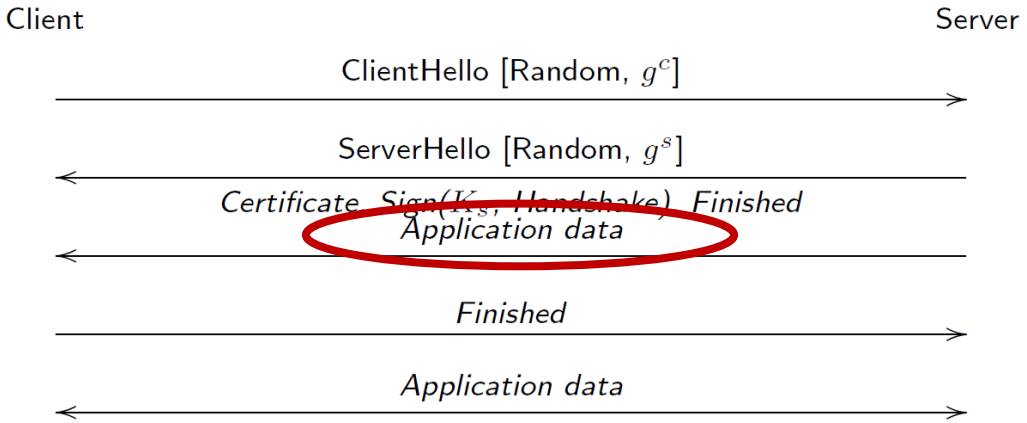
European standards body ETSI created eTLS – a counter approach that supports static keys for preconfigured proxies

TLS 1.3 restricts session key agreement to ephemeral Diffie-Hellman

- Perfect Forward Secrecy
- A small set of ‘safe’ DHE parameters: “Named Groups”
- No option of static keys (for sharing)
- No preconfigured TLS proxy (without certificate forgery)

TLS 1.3 Optimization

- Narrow options to a limited set of named groups for elliptic curve DHE
- Clients can make good guesses on server support
- If successful, server can send data immediately
- Client can send data after one roundtrip



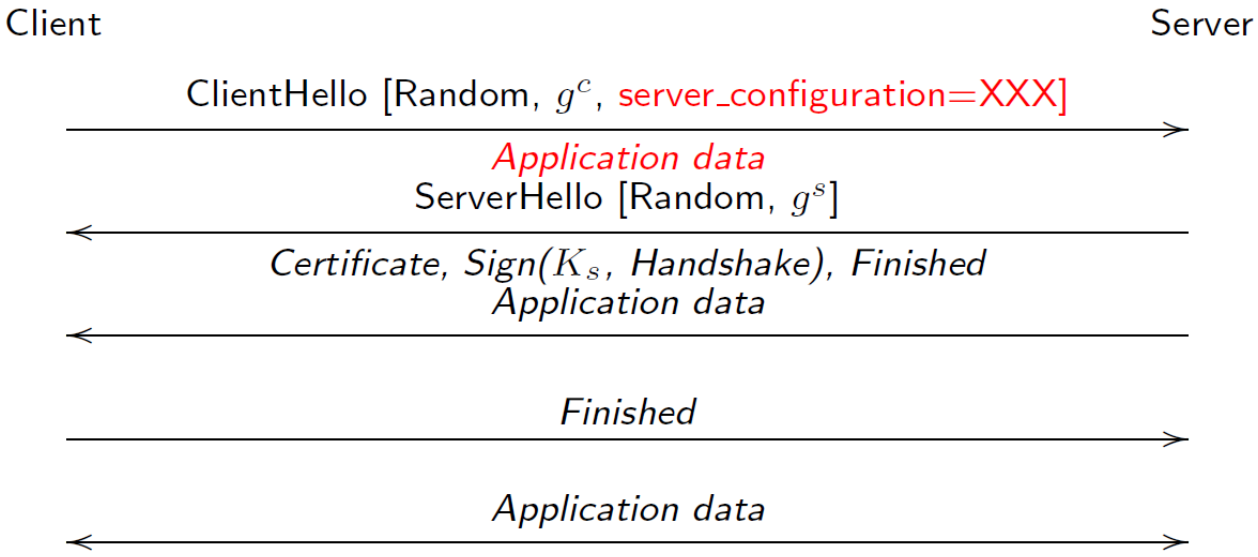
0-RTT Session Resumption

Cached pre-shared keys allow for ultra-fast session resumption

Often client and server re-establish a session after previous communication. In such cases, the client may use the previous session credentials as pre-shared keys:

+ Clients can cache server parameters from previous handshakes
+ Client can thereby authenticate and encrypt data immediately
- Data is not forward secret
- No replay protection is given

0-RTT Handshake



Securing Application Endpoints

DANE

E2E Application Security

Application transport today provides encryption, integrity protection, privacy, +++

- Examples are TLS, DTLS, IPSec, S/MIME, SSH, …

Secure channels require bootstrapping

- Built from CA hierarchies
- Relies on (a) trust of root CAs, and (b) integrity of trust delegation
- One compromise invalidates the complete chain of trust

Threats & Flaws of the CA Approach

CAs are universal & vulnerable
- No namespace constraints - any CA can issue certificates for any entity on the Internet
- July 10,2011 an attacker created a wildcard certificate for Google (DigiNotar)

Tolerance & delegation may lead to unexpected endpoints
- Often self-signed or expired certificates
- CDNs officially terminate TLS sessions

We learn CA keys out of band
- Local misuse by configuration („TLS-proxies“)

Key revocation problem
- Revocation lists slow, not scalable
- After compromise, everybody wants to revoke ➦Heartbleed!

DNS Based Authentication by Named Entities (DANE)

Move trust from CAs to DNSSEC Infrastructure

Built on top of DNSSEC: Defines new TLSA DNS record (RFC 6698)

- May constrain the CA, or
- Deliver certificate directly from DNS

TLSA Records in DNSSEC

The TLSA record ties a certificate to a named service

DNS record type to authenticate remote endpoints in transport: SSL/TLS (web, mail, …)

TLSA key: _port._proto.domain.tld –

```
_443._tcp.good.dane.verisignlabs.com
```

TLSA value: Meta-data + Certificate Association Data (raw cert data in hex) –

```
(0 0 1
d2abde240d7cd3ee6b4b28c54df034b9
7983a1d16e8a410e4561cb106618e971)
```

DANE verification process

DNS zones have TLSA record(s) that uniquely authorize certificates used by servers

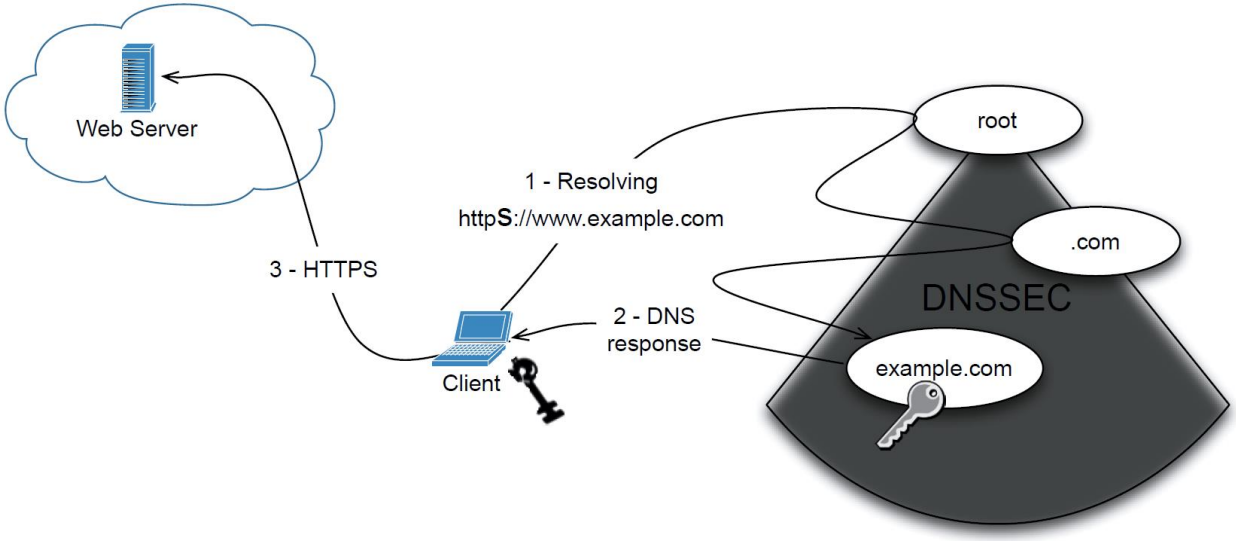


Image Source: Eric Osterweil, Verisign Labs

DANE Résumé

Promise:

Providing security between authorized transport endpoints (Web, Mail, …)

Reality:

Server-centric security toolset – mainly inter-SMTP mail security

Emerging building blocks for ‘Secure Email’ with clients (➦ Thunderbird)

Internet Society (ISOC) has a deployment program called Deploy 360:
`http://www.internetsociety.org/deploy360/resources/dane/`

Enhancing Visibility of CA Activities

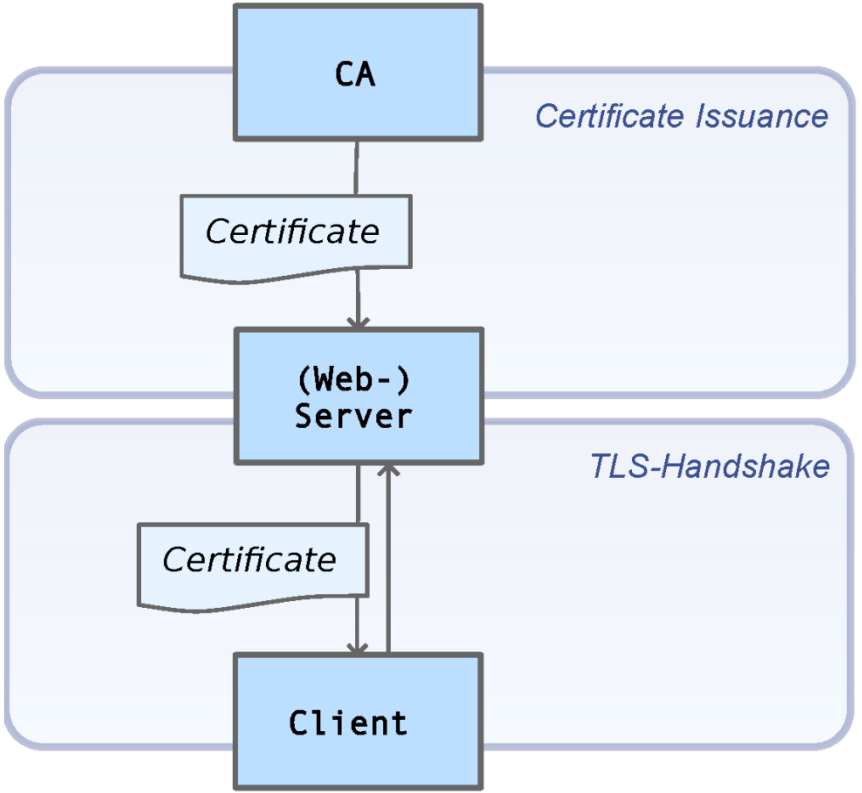
CERTIFICATE TRANSPARENCY

Where the CA Approach Falls Short

A CA in the trust chain can
- Certify any resource
- Remain in secrecy
- Lie about time of issuing

A Client alone cannot
- Verify correctness of the CA

Public trust anchors can help
- DANE per name
- CT per certificate

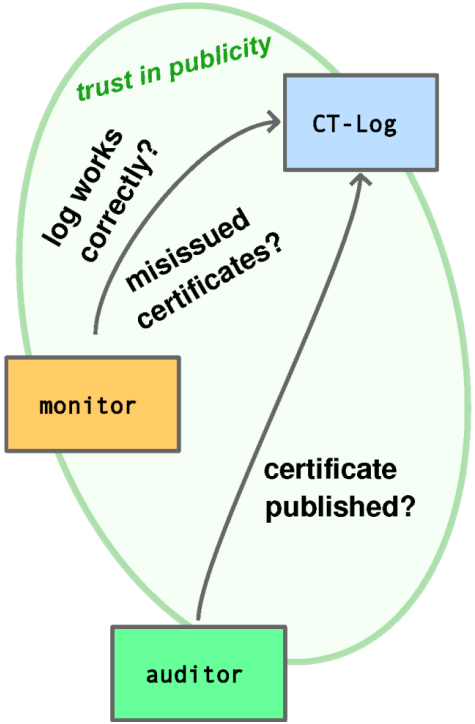


CT: Replicate Certificates in Public – RFC 6962

Publish certificates to independent CT-Logs
- Purpose of “monitoring”
- Requires valid trust chain

Logs promise to
- Provision certificate history online
- Maintain immutable entries
- Hold correct time-stamps: Returns Signed Certificate Timestamps (SCT)

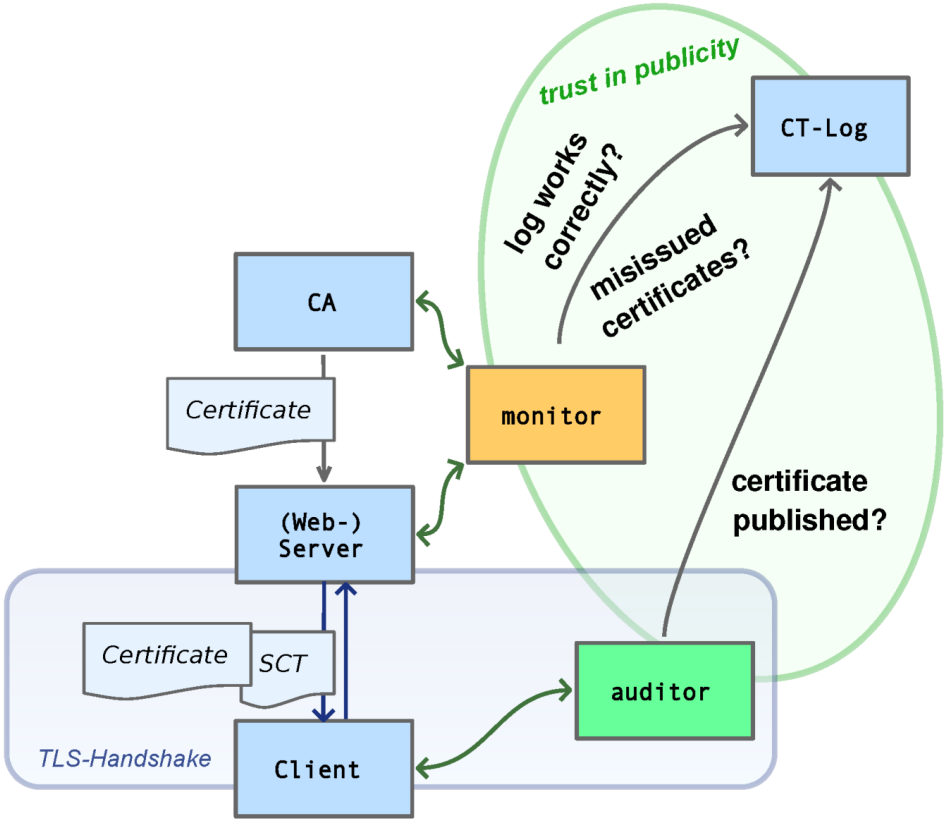
Clients check logs
- Purpose of auditing
- SCT serves as log promise
- Refuse unpublished or incorrect certificates



CT Enforces Visibility

Publication/Monitoring
- CAs
- Resource Owners
- 3rd Parties

Verification/Auditing
- Clients based on Signed Certificate Timestamp (SCT)



Case Study: CT Deployment

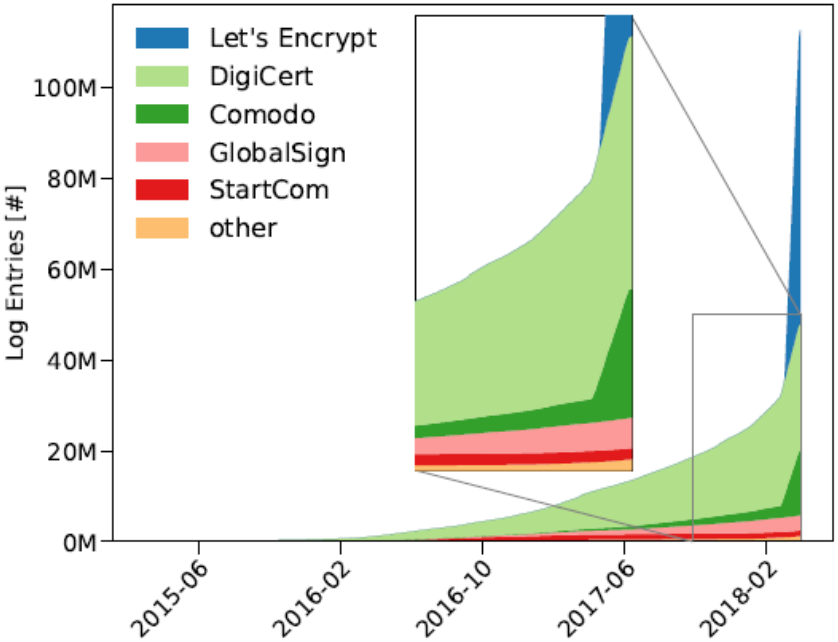
Initial certificate deployment in Logs remained low

Google announced in October 2016, they would only regard certificates trusted if published in Logs – with little impact

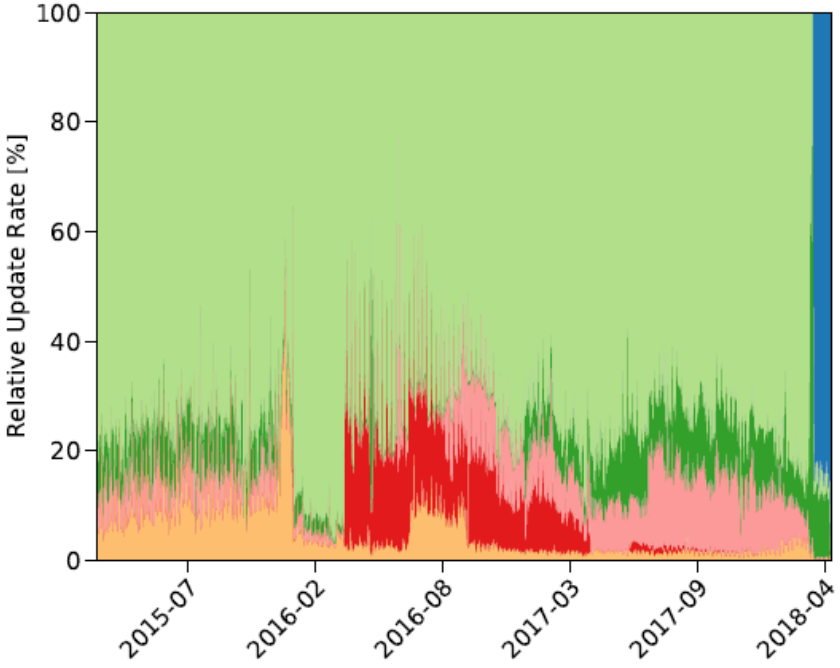
Google announced and implemented this policy in Chrome as of April 18th, 2018

This led to an explosion of deployment – and a sharp monopolization of CT logs

Logging of Precertificates
- While Approaching the Chrome Deadline

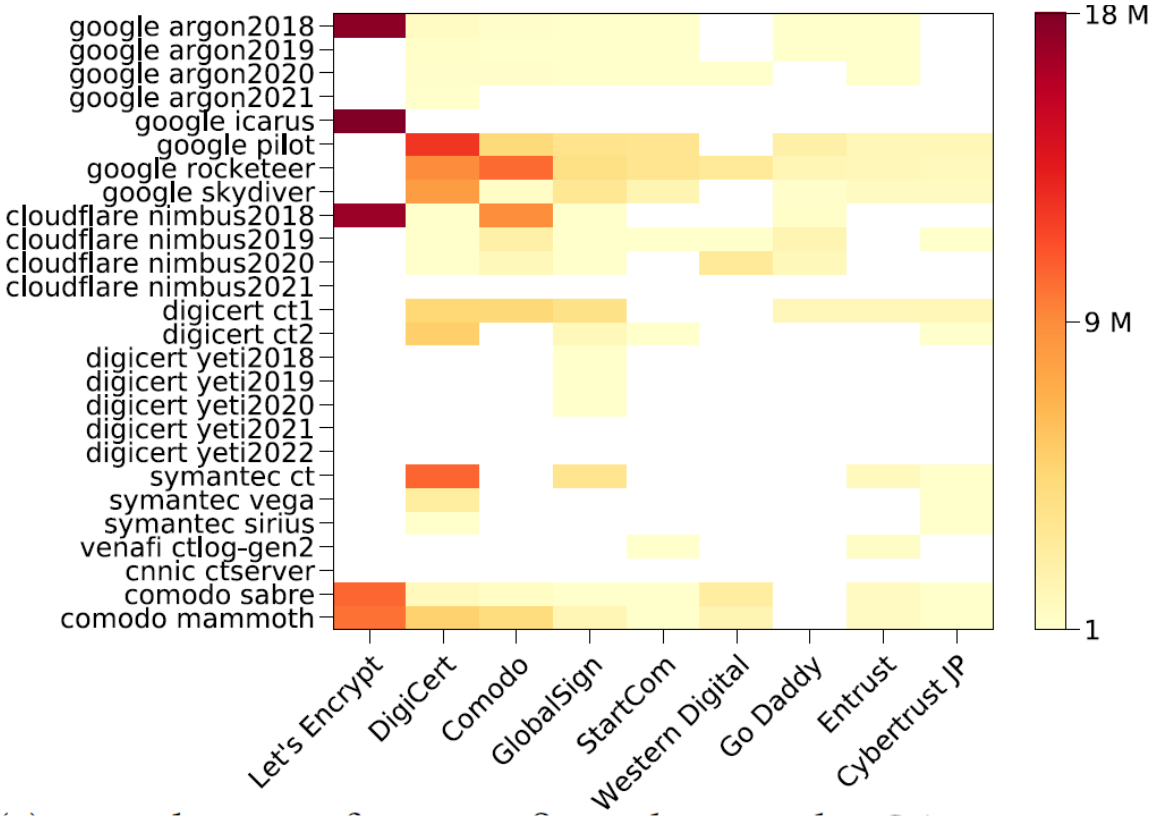


(a) Cumulative growth of logged precertificates by Certification Authority (CA).



(b) Relative update rate per CA and day. Let's Encrypt dominates after starting to log.

Distribution of Precertificate Logging: CAs versus CT-Logs



Leakage of DNS Subdomains

New Measurement Technique:
CT Honeypots

New attack vector brought by CT:
Publication of (future) subdomain names

- FQDNs of services accessible in CT-Logs

Subdomain enumeration – as prevented by DNSSec – is a common attack preparation

Measurement: CT Honeypot

- Inject hashed subdomain names
- Measure DNS queries
- Result: multiple queries within seconds

Résumé on CT

CT makes the use of TLS certificates transparent

- CAs and resource owners can publish
- Clients should check/enforce publication
- Integrity should be monitored, forgery becomes visible

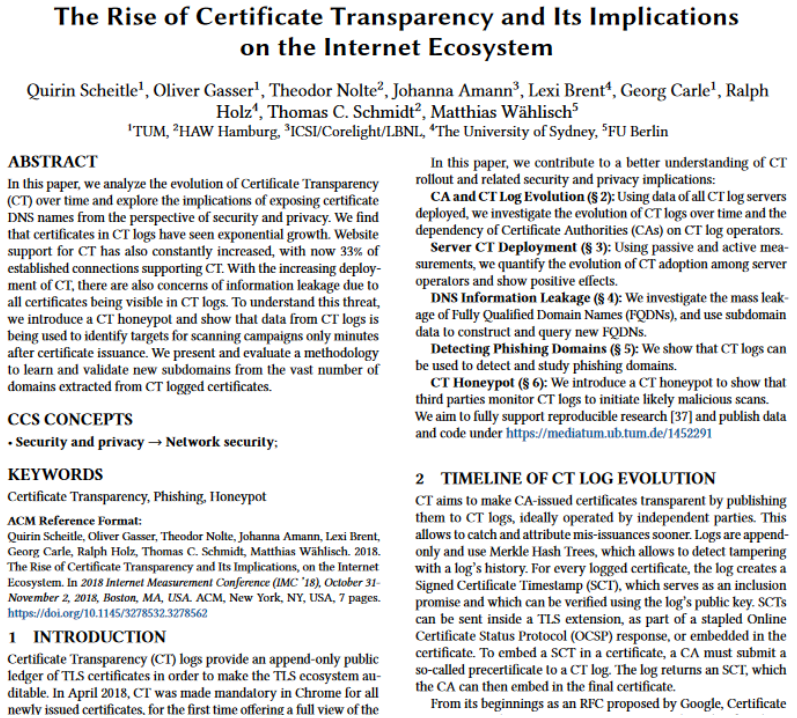
Technically issuing of illegitimate certificates remains unhindered

Privacy issue of CT

- Logs see certificate queries
- Leak subdomains
- But reveal potential pishing domains:
  `appleid.apple.com-7etr6eti.gq`

Literature

Q. Scheitle, O. Gasser, T. Nolte, J. Amman, L. Brent, G. Carle, R. Holz, T. C. Schmidt, M. Wählisch,
The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem,
In: *Proc. of ACM Internet Measurement Conference (IMC 2018), ACM Digital Library*.
DOI: https://doi.org/10.1145/3278532.3278562



The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem

Quirin Scheitle[1], Oliver Gasser[1], Theodor Nolte[2], Johanna Amann[3], Lexi Brent[4], Georg Carle[1], Ralph Holz[4], Thomas C. Schmidt[2], Matthias Wählisch[5]
[1]TUM, [2]HAW Hamburg, [3]ICSI/Corelight/LBNL, [4]The University of Sydney, [5]FU Berlin

ABSTRACT

In this paper, we analyze the evolution of Certificate Transparency (CT) over time and explore the implications of exposing certificate DNS names from the perspective of security and privacy. We find that certificates in CT logs have seen exponential growth. Website support for CT has also constantly increased, with now 33% of established connections supporting CT. With the increasing deployment of CT, there are also concerns of information leakage due to all certificates being visible in CT logs. To understand this threat, we introduce a CT honeypot and show that data from CT logs is being used to identify targets for scanning campaigns only minutes after certificate issuance. We present and evaluate a methodology to learn and validate new subdomains from the vast number of domains extracted from CT logged certificates.

CCS CONCEPTS

• **Security and privacy → Network security**;

KEYWORDS

Certificate Transparency, Phishing, Honeypot

ACM Reference Format:
Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, Matthias Wählisch. 2018. The Rise of Certificate Transparency and Its Implications, on the Internet Ecosystem. In *2018 Internet Measurement Conference (IMC '18), October 31-November 2, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3278532.3278562

1 INTRODUCTION

Certificate Transparency (CT) logs provide an append-only public ledger of TLS certificates in order to make the TLS ecosystem auditable. In April 2018, CT was made mandatory in Chrome for all newly issued certificates, for the first time offering a full view of the

In this paper, we contribute to a better understanding of CT rollout and related security and privacy implications:

CA and CT Log Evolution (§ 2): Using data of all CT log servers deployed, we investigate the evolution of CT logs over time and the dependency of Certificate Authorities (CAs) on CT log operators.

Server CT Deployment (§ 3): Using passive and active measurements, we quantify the evolution of CT adoption among server operators and show positive effects.

DNS Information Leakage (§ 4): We investigate the mass leakage of Fully Qualified Domain Names (FQDNs), and use subdomain data to construct and query new FQDNs.

Detecting Phishing Domains (§ 5): We show that CT logs can be used to detect and study phishing domains.

CT Honeypot (§ 6): We introduce a CT honeypot to show that third parties monitor CT logs to initiate likely malicious scans. We aim to fully support reproducible research [37] and publish data and code under https://mediatum.ub.tum.de/1452291

2 TIMELINE OF CT LOG EVOLUTION

CT aims to make CA-issued certificates transparent by publishing them to CT logs, ideally operated by independent parties. This allows to catch and attribute mis-issuances sooner. Logs are append-only and use Merkle Hash Trees, which allows to detect tampering with a log's history. For every logged certificate, the log creates a Signed Certificate Timestamp (SCT), which serves as an inclusion promise and which can be verified using the log's public key. SCTs can be sent inside a TLS extension, as part of a stapled Online Certificate Status Protocol (OCSP) response, or embedded in the certificate. To embed a SCT in a certificate, a CA must submit a so-called precertificate to a CT log. The log returns an SCT, which the CA can then embed in the final certificate.

From its beginnings as an RFC proposed by Google, Certificate