Google

Didn't Chrome Already Have a Root Store?

Chrome Security

@davidcadrian | @dadrian.io

Sep 2023

“Starting in Chrome 105, to improve user security and provide a consistent experience across different platforms, Chrome maintains its own default root store and built-in certificate verifier.”

Chrome Release Notes
Chrome 105, August 2022

HTTPS is the foundation of web security

Diffie-Hellman
KEMs

We can derive a shared secret and use it to create a secure channel.

Encryption
HMAC / AEADs

Confidentiality

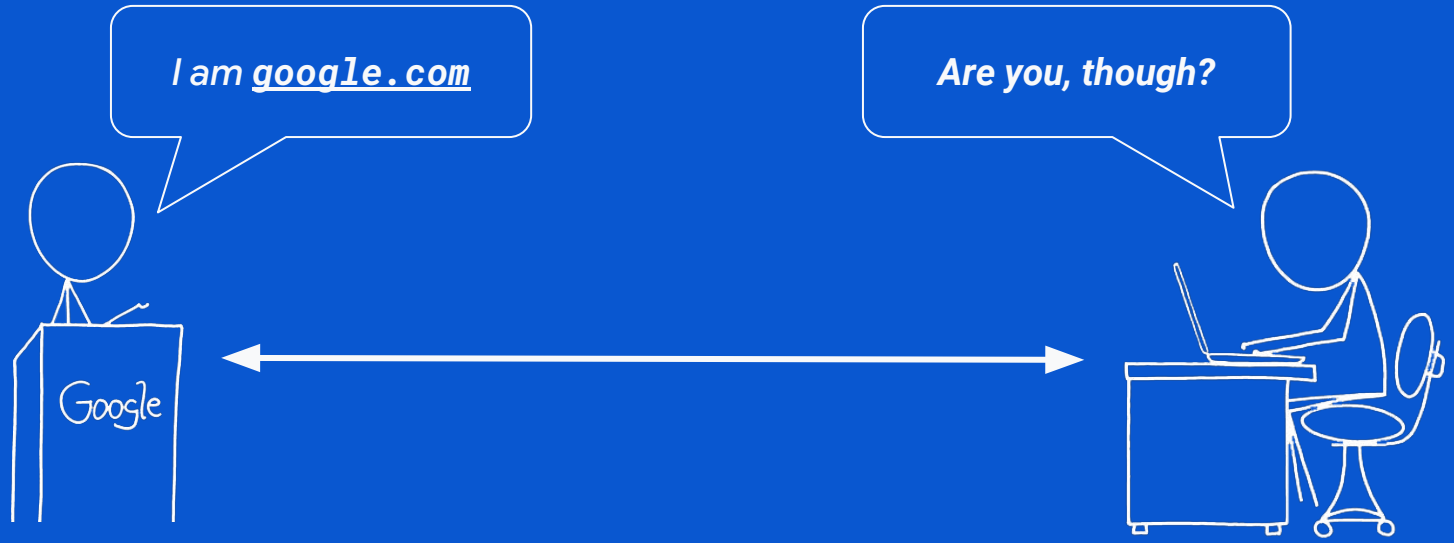
Integrity

Authentication

Confidentiality

Integrity

Authentication

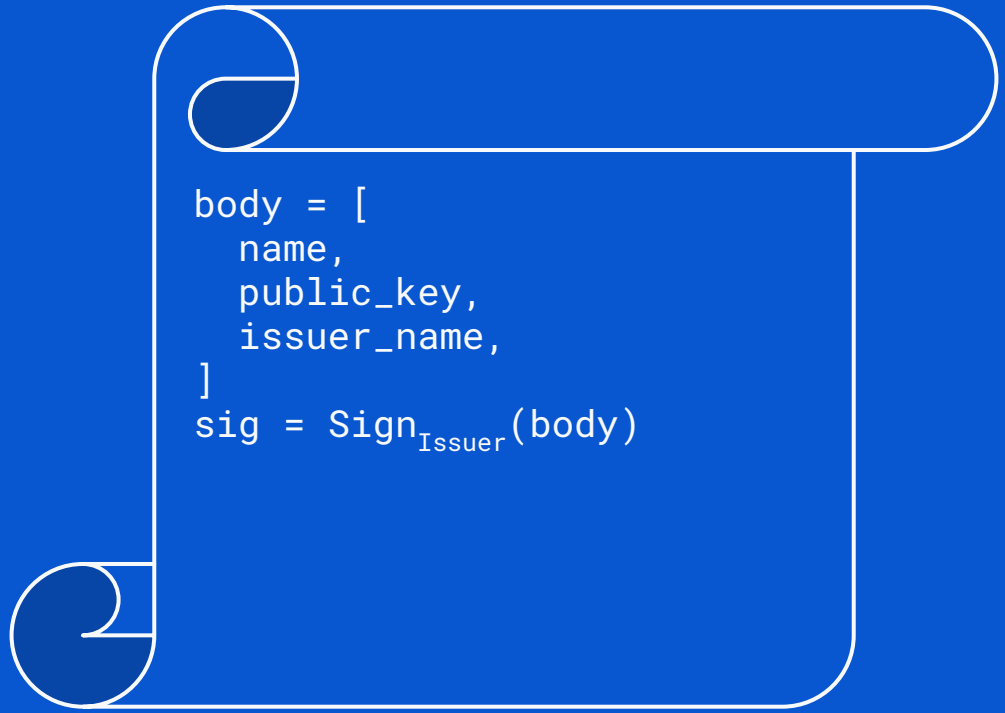

I am google.com
Are you, though?
Google

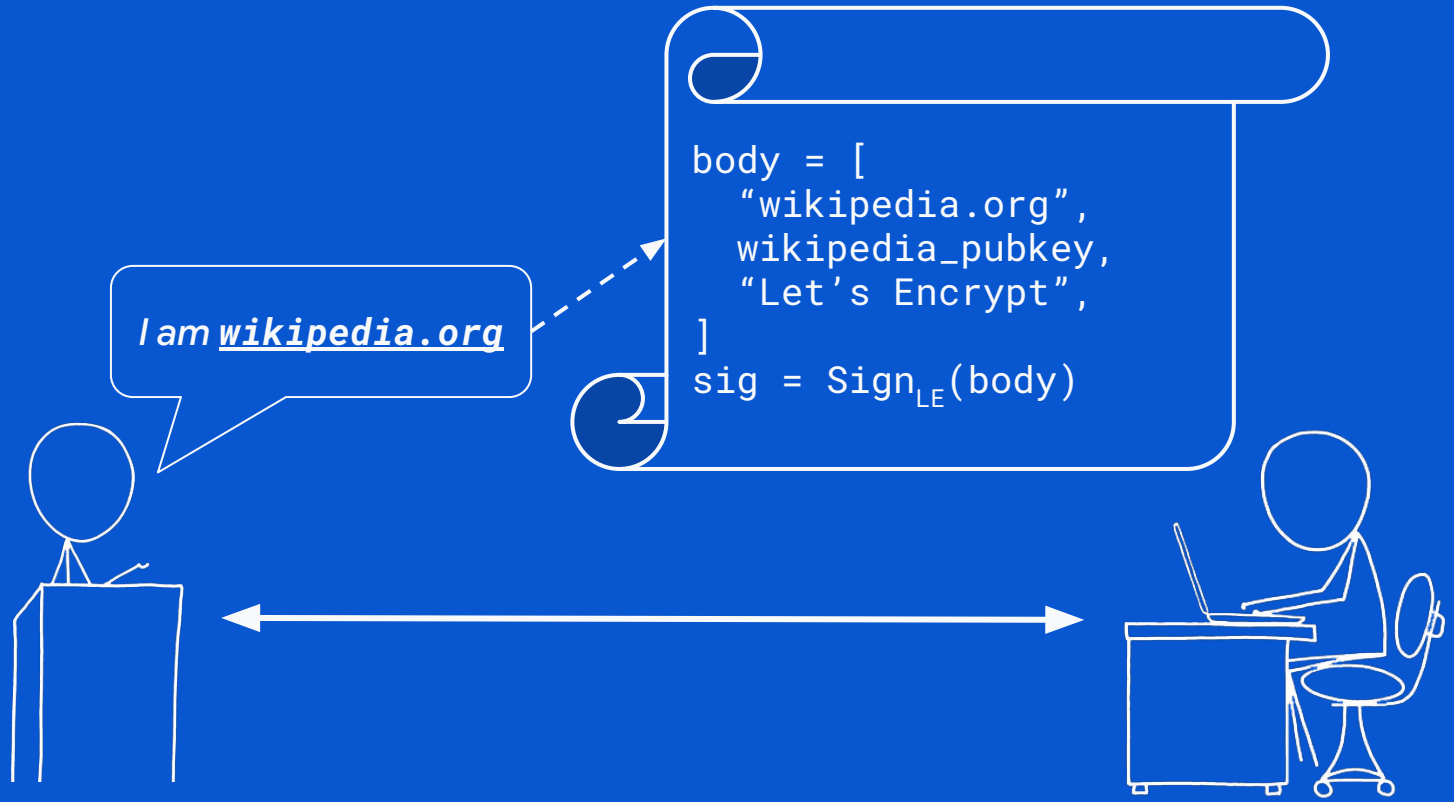
Digital signatures enable us to authenticate someone if we know their **public key**.

```
(public_key, private_key) = GenSigningKey()
sig = Sign(private_key, data)
ok = Verify(public_key, sig, data)
```

Digital signatures enable us to authenticate someone if we know their **public key**.

But how do we know the public key?


body = [
name,
public_key,
issuer_name,
]
sig = Sign_Issuer(body)


I am wikipedia.org
body = [
“wikipedia.org”,
wikipedia_pubkey,
“Let’s Encrypt”,
]
sig = Sign$_{LE}$(body)

Where do issuers come from?

Where do issuers come from?

Browser and platform vendors!

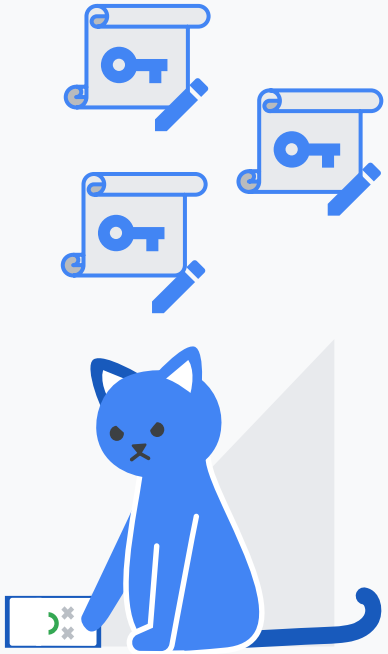
Vendors trust a set of **certification authorities** to **validate domain ownership** and **issue** certificates that attest that a key is associated with a set of names.

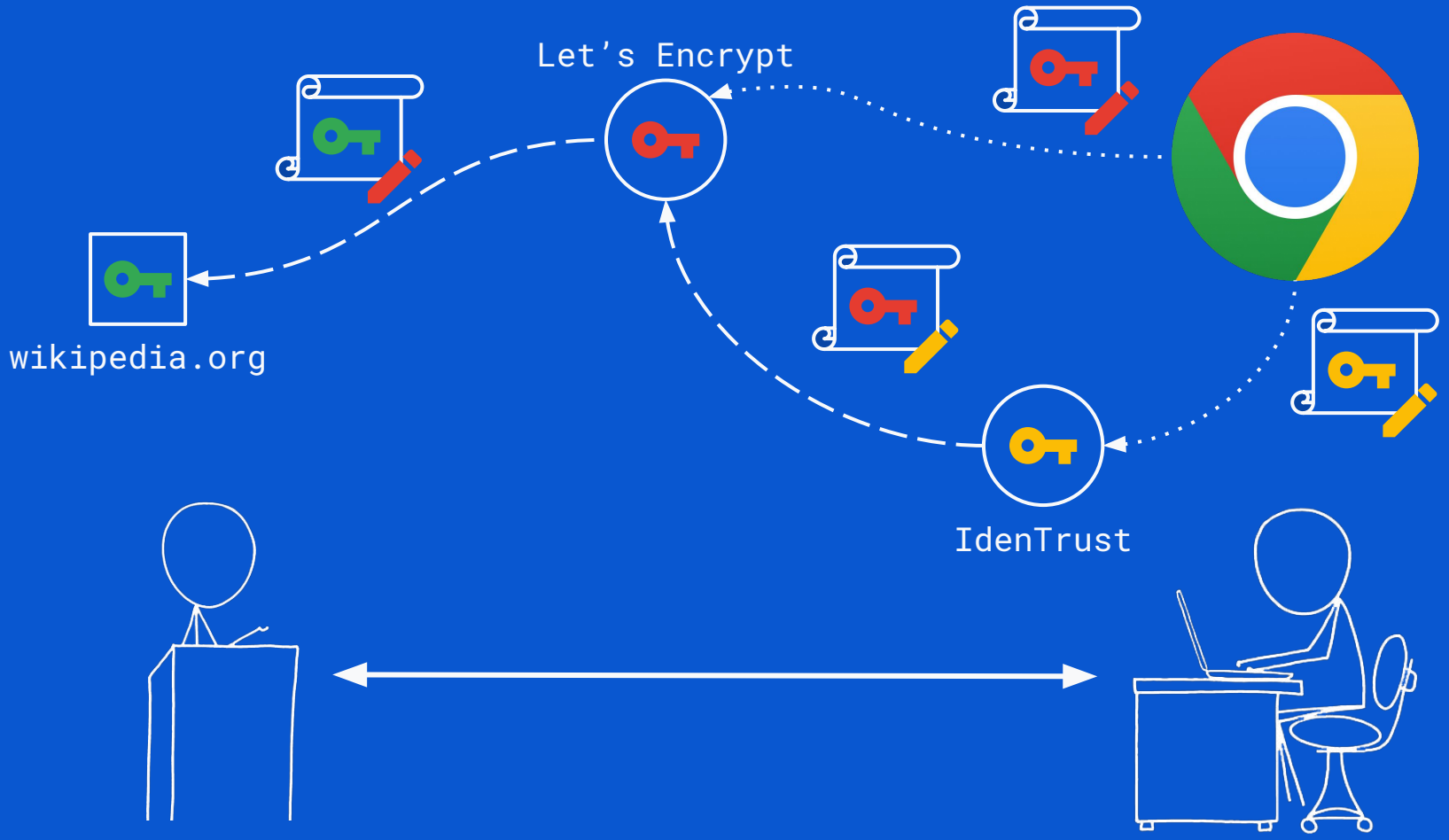
Root Stores and Root Programs

Browser and platform vendors maintain **root stores**, which contain the set of **root certificates** representing trusted issuers.

The policies and requirements around what root certificates are included a root store is known as a **root program.**

- Mozilla Root Program
- Apple Root Program
- Microsoft Root Program
- Chrome Root Program




Let’s Encrypt
wikipedia.org
IdenTrust

Ensuring the requestor operates the domain

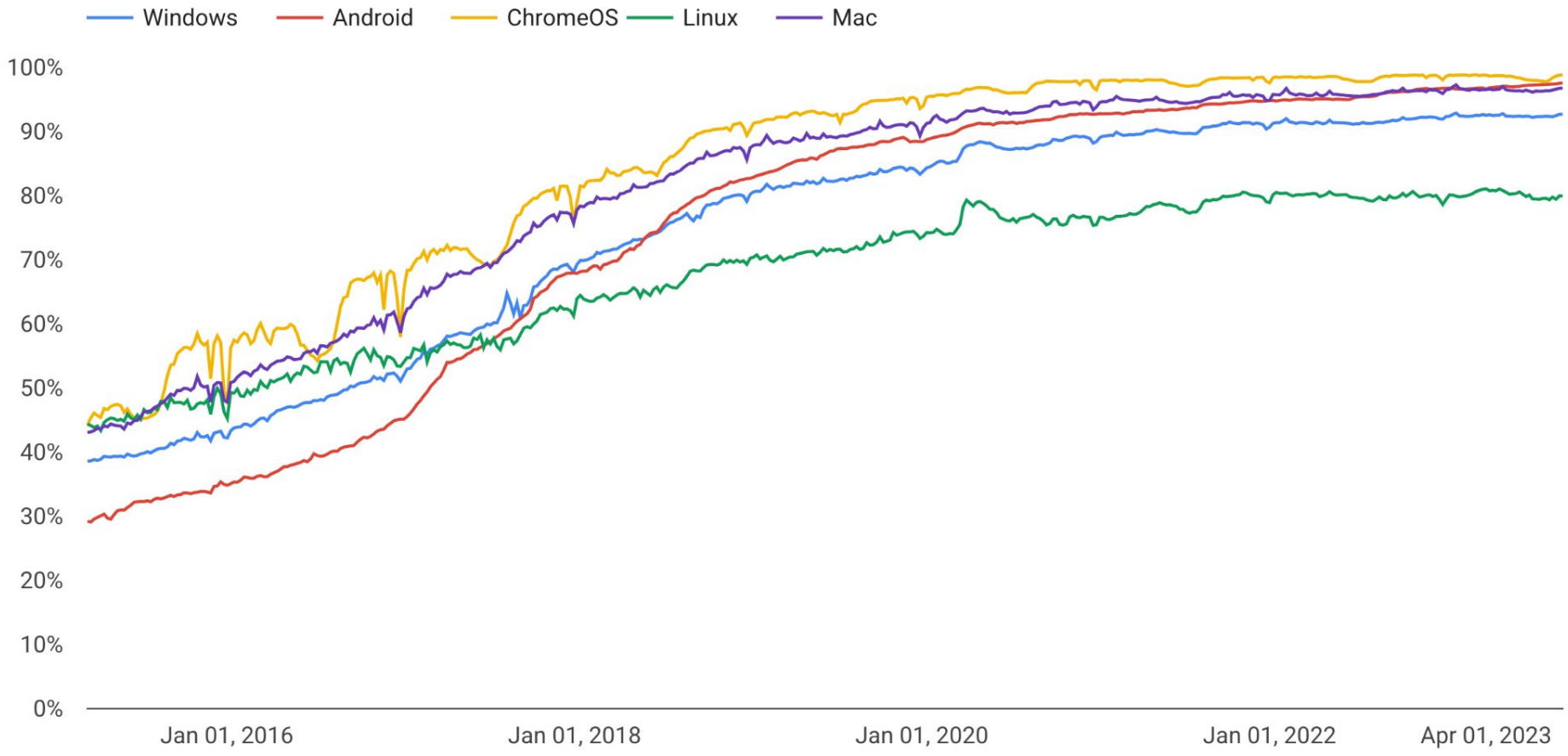
CAs are responsible for domain validation, issuance, and conveying status.

Signing certificates

Is a specific certificate revoked?

A **single** **compromised** CA can **negatively impact** any number of domains on the Internet.

Trust.



Source: HTTPS Transparency Report

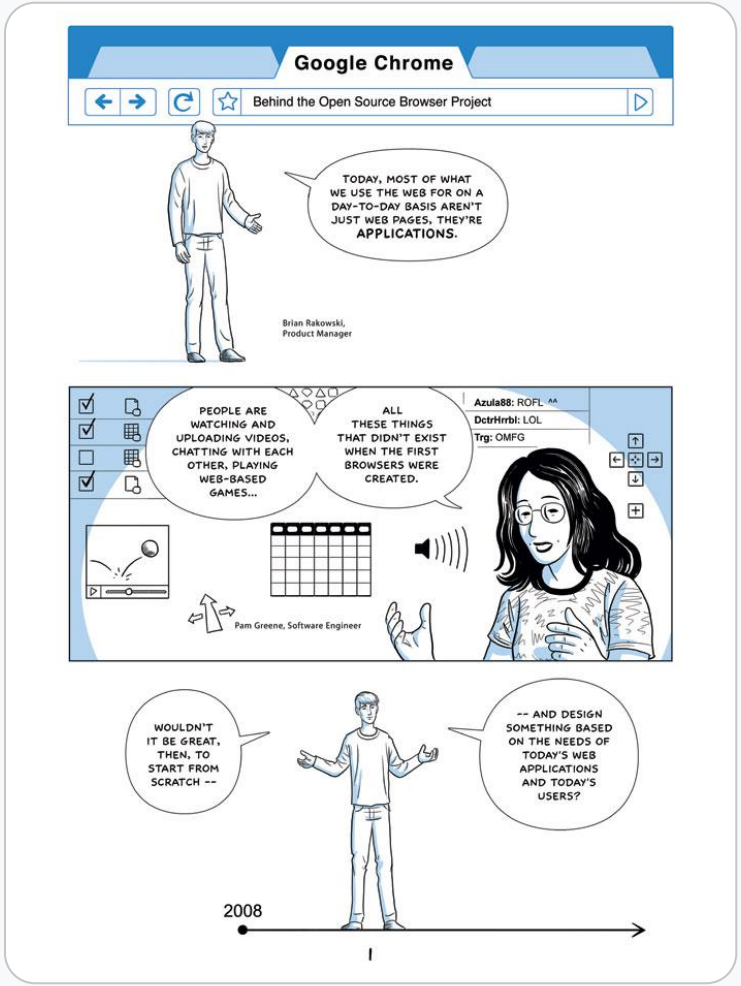
Chrome

The Old Days

Chrome launched in 2008 (alongside a comic drawn by Scott McCloud).

Chrome used the platform-provided root stores.

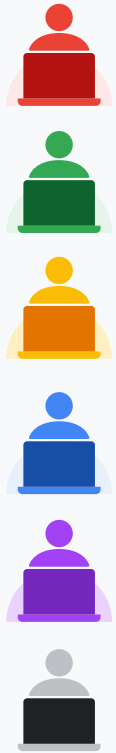
- Apple Root Store
- Microsoft Root Store
- Mozilla Root Store



Browsers need to offer consistent capabilities on a diverse set of devices

Over time, Chrome started providing its own implementations for more components of the browser (HTTP, Graphics, TLS).

- Performance
- Patching
- Prioritization



Platform Certificate Verifiers

✅ Good citizen of the platform

✅ Automatically pick up locally installed roots

✅ Automatically pick up device configuration changes

❌ Limited by the update cadence of the platform and device

❌ Often coupled with platform root stores

❌ May not meet the needs of a browser that needs to handle trust incidents on behalf of its users

A Few Trust Incidents...

DigiNotar
2011

- Attacker compromises DigiNotar CA and issues rogue certificates
- All platforms immediately distrust the CA

TurkTrust
2011

- Accidentally issued two intermediate CA certificates to subscribers, one of which was used for impersonation.
- Chrome blocked the misissued certificates, platforms continued to trust the CA

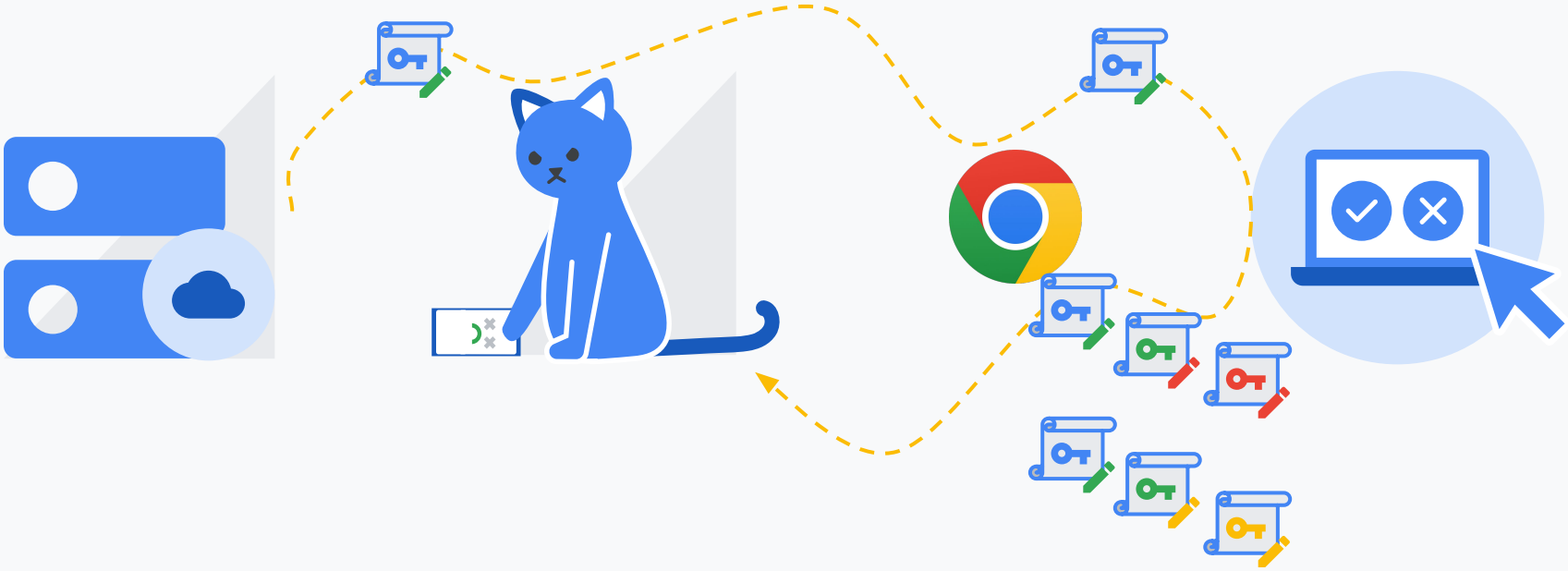
Wosign
2016

- Misissued a certificate for Github and backdated certificates to avoid security requirements
- Staged distrust by Chrome, eventually distrusted by all platforms.

Symantec
2017

- Repeated failure to validate domain ownership before issuance
- Staged distrust by Chrome, eventually distrusted by all platforms

Platform Certificate Verifiers



Chrome Certificate Verifier and Chrome Root Store

Chrome Certificate Verifier

- A common certificate verification process across Windows, macOS, Chrome OS, Linux, and Android
- Implements additional verification required by Chrome (Certificate Transparency, etc)
- Consistency on the failure cases!

Chrome Root Store

- Root store operated by the Chrome Root Program shipped directly with Chrome on all Blink platforms
- Updated with Chrome
- Chrome Root Program can more directly represent Chrome in the Web PKI

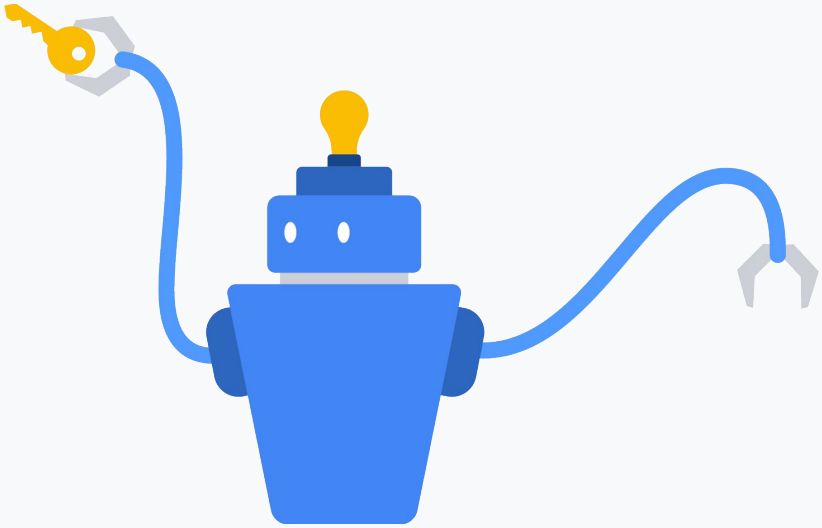
Launch

Goals

- Don’t break anything.
- Maintain a high bar for certificate verification

Approach

- Platform-by-platform
- Import local trust decisions
- Flag and enterprise policy to revert to old behavior



Launch

Windows

- 13 unique location for trust and distrust information
- *Trusted People* store

Mac

- Load local trust anchors added to the Default and System keychains
- Cache at startup to work around keychain becoming non-responsive

Android

- Load user-installed certificates with Android Java SDK
- Cross-language callback and synchronization

Linux
ChromeOS

- Very uneventful
- Had been using CCV since ~2019

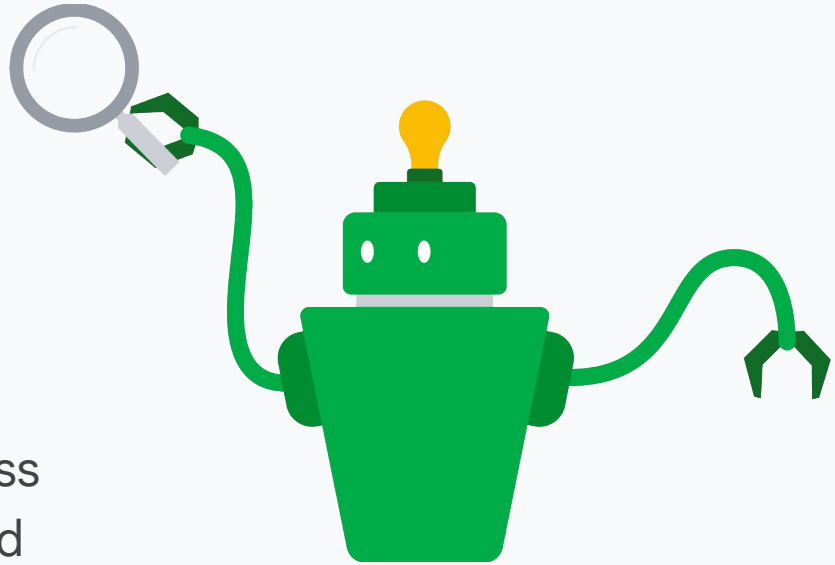
Launch Validation

Dual-Verifier Trial

- Run the platform verifier and CCV+CRS for a subset of users
- Look for differences in results

A/B Test

- Standard part of Chrome launch process
- Compare CRS+CCV enabled vs disabled



Despite **stricter requirements**, launching the Chrome Root Store and Chrome Certificate Verifier **decreased certificate errors** across the board.

Metric Improvements

More accurate verification

40% decrease in certificate interstitials on Windows

Faster certificate verification

30-85% decrease in time to verify a certificate

Faster page loads

~1% decrease in time to first contentful paint

Improvement in Core Web Vitals

~0.1% increase in pass rates

Why launch a root program?

Improve security

Improve experience

Improve the Web PKI

Moving Forward, Together

Increase Security through Agility and Simplicity

✨ **Automation** ✨

CAs

How to be a CA

Bootstrapping

1. Audited key generation with hardware-backed key material (HSM)
2. Root certificate compliant with the CA/Browser Forum (CABF) Baseline Requirements (BRs)
3. Apply to root programs

CA Audits

Brittle: CA's trustworthiness assessed by paperwork-like audits

- The CA chooses the auditor
- Audit output is 2-3 pages of mostly boilerplate
- Audit quality varies

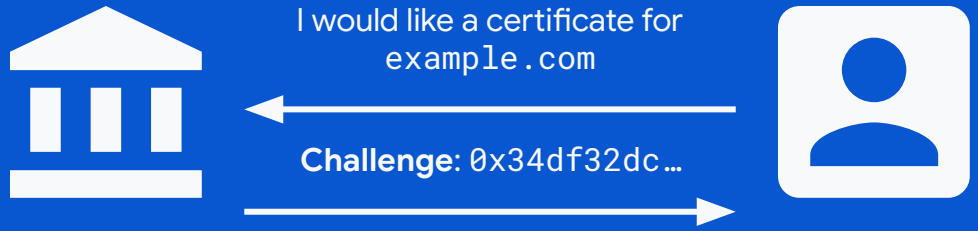
Compliance, not security.

How to be a CA

Operations

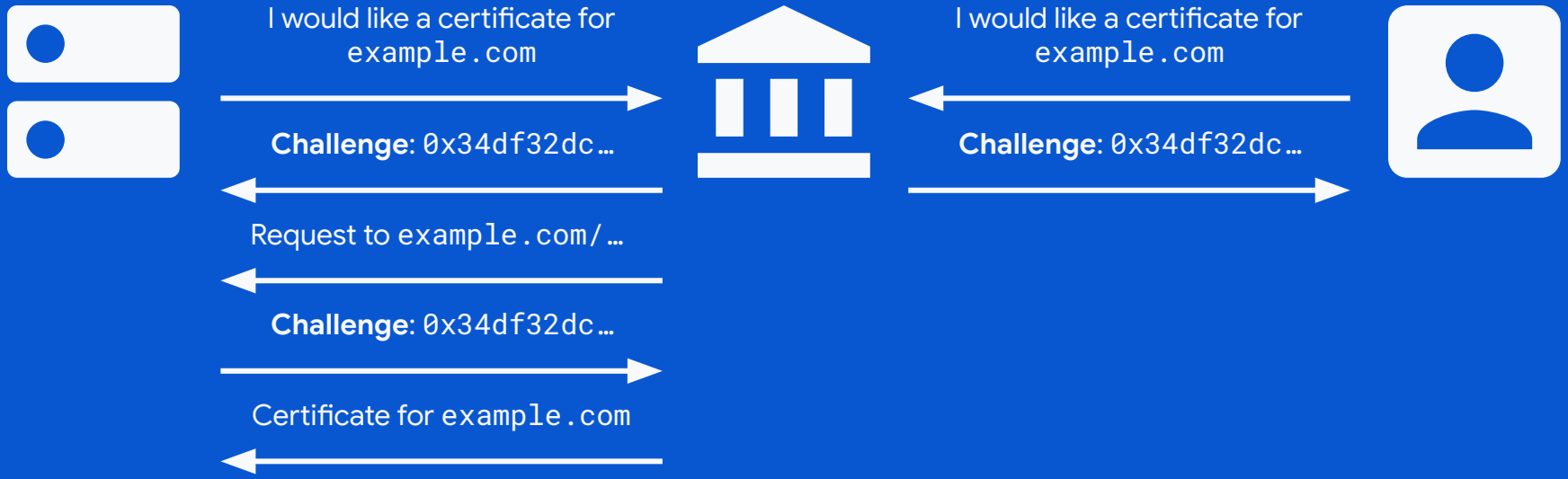
1. **Domain Validation**
2. Issuance
3. Certificate Status

Domain Validation



Manual Issuance

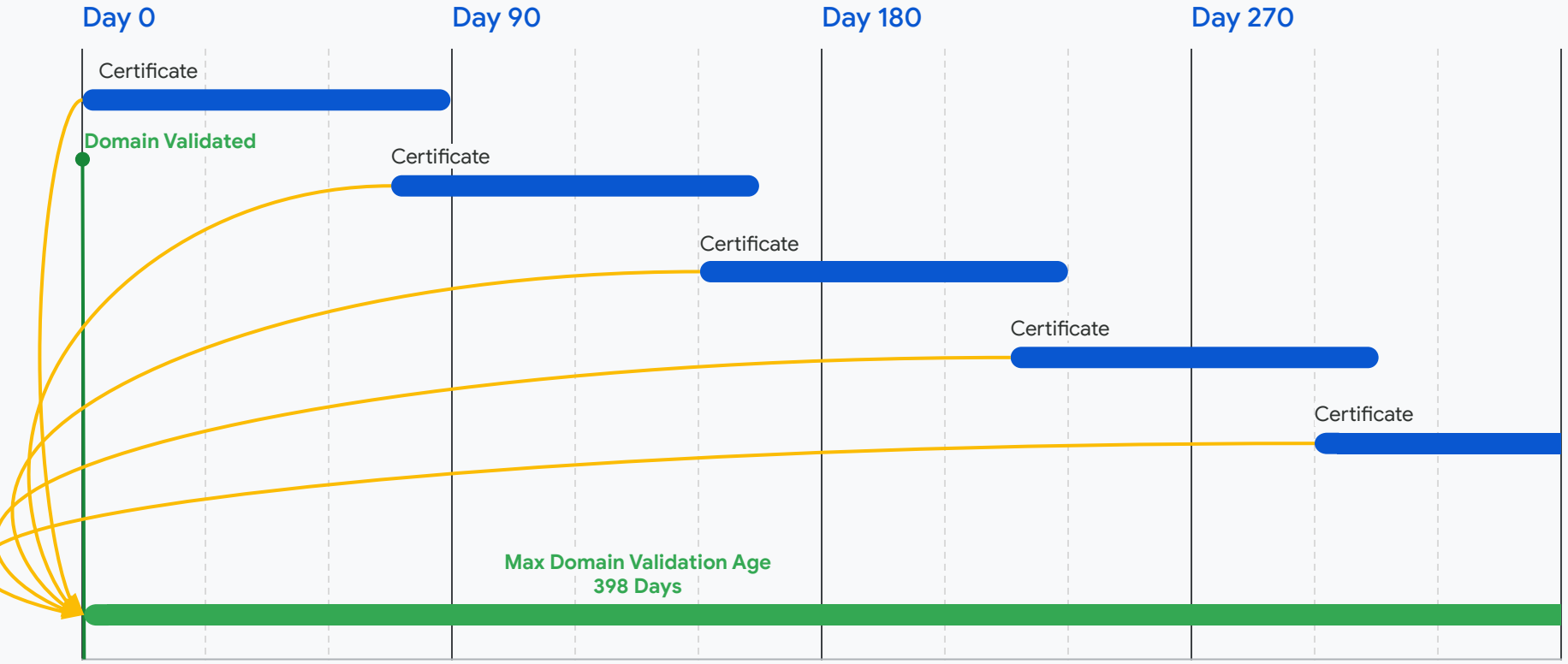
Domain Validation



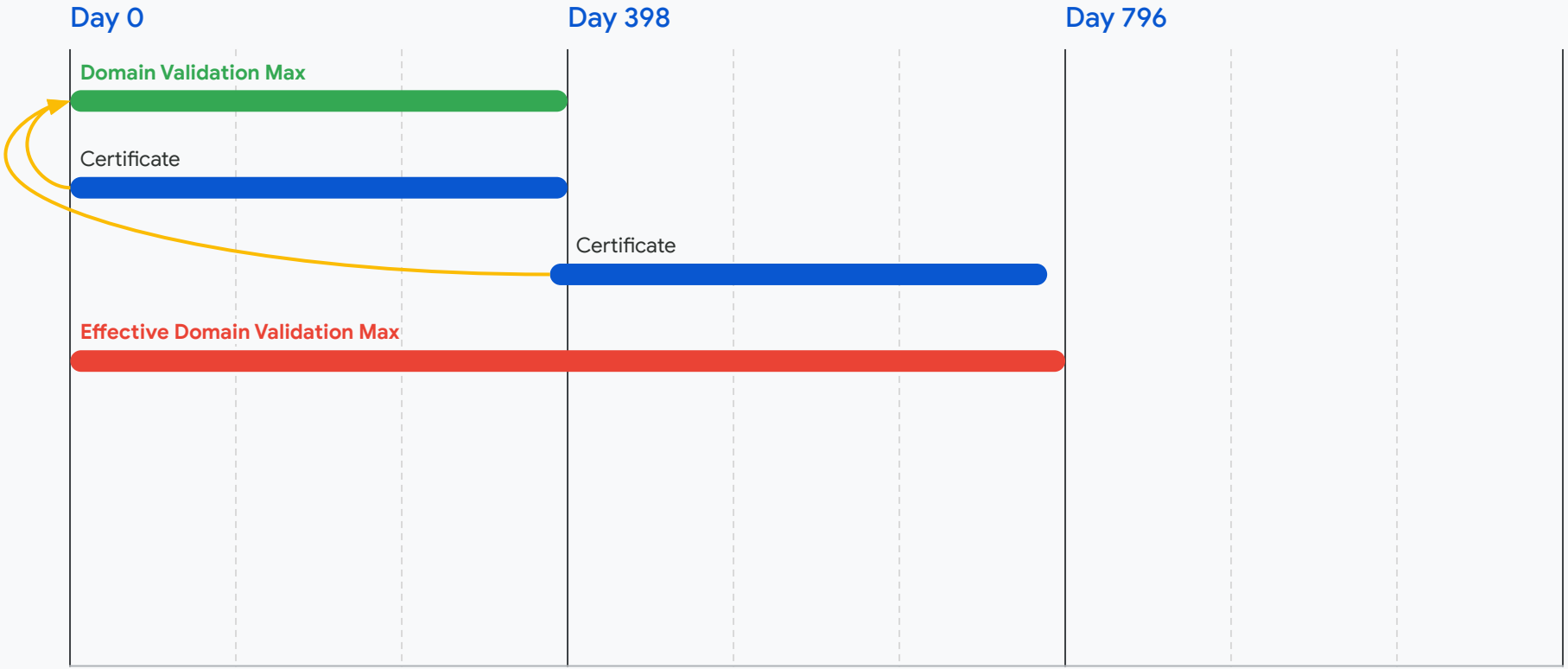
Domain Validation

Agile: ✨ Automation ✨

Domain Validation Reuse



Domain Validation Reuse



Domain Validation Reuse

Risk: Point in time validation spread across lifetime of certificate

More secure: Reduce or eliminate domain validation reuse

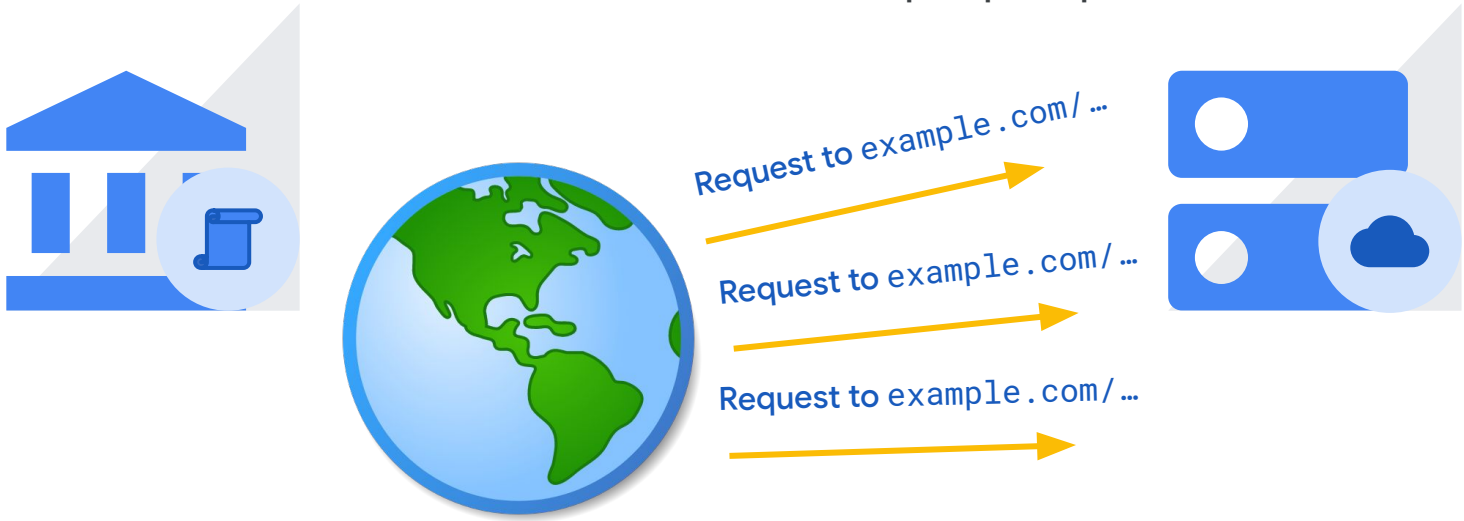
More secure: Eventual reduction of maximum certificate lifetime

Multi-Perspective Domain Validation

Risk: BGP hijacking, DNS spoofing, etc.

Reality: Web PKI is distributed and delegated TOFU

More secure: Domain validation from multiple perspectives



How to be a CA

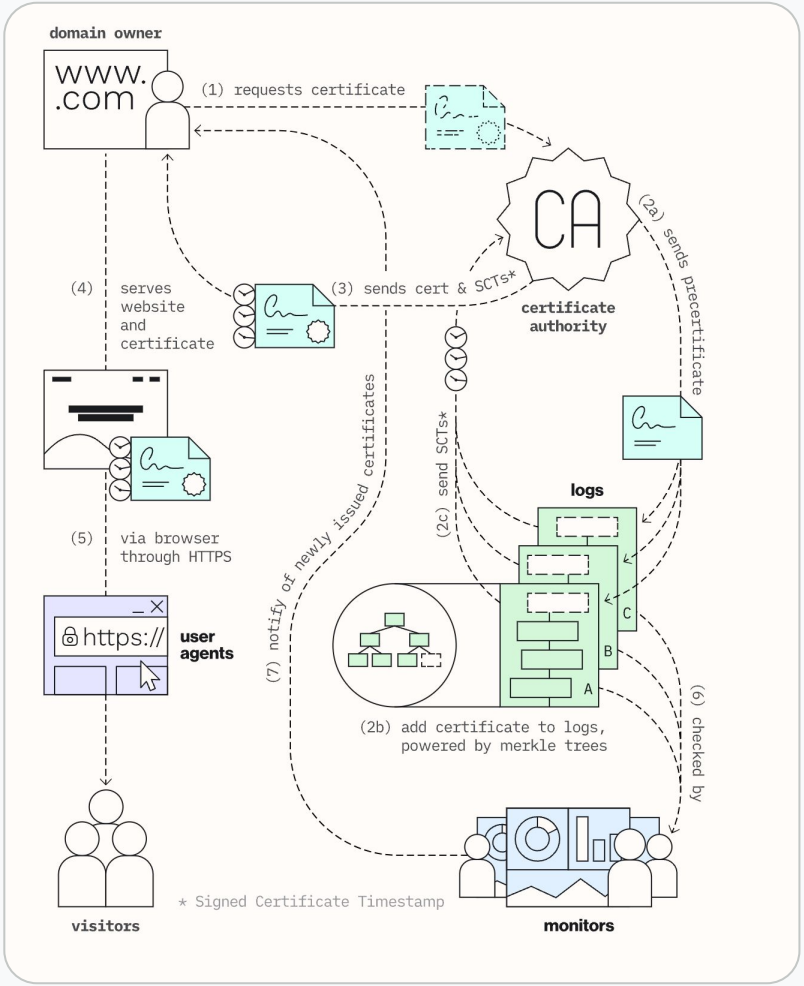
Operations

1. Domain Validation
2. **Issuance**
3. Certificate Status

How do site operators know what certificates exist for their site?

Certificate Transparency

More secure: Reveal targeted attacks by requiring trusted certificates to be publicly logged



How to be a CA

Operations

1. Domain Validation
2. Issuance
3. **Certificate Status**

Revocation.
How hard could it be?

Certificate Revocation Lists (CRLs)

Big lists of certificates that have been revoked

- Key compromise
- Administrative reasons
- Change of ownership

Certificate Revocation Lists (CRLs)

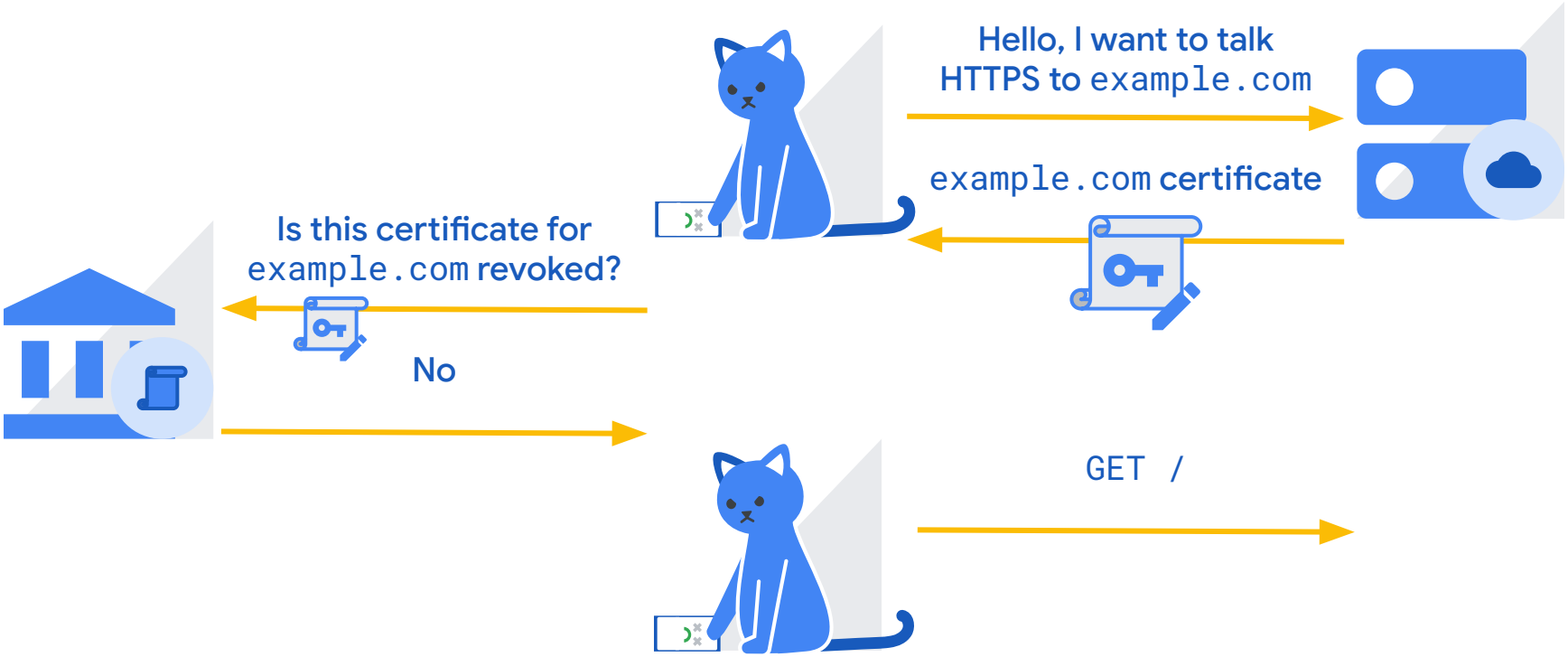
Too Big for Clients: Tens or hundreds of megabytes of revocation data

Expensive to Host: $400K-$1M / month to host in the wake of Heartbleed

Not Really Required: CRLs were optional in the BRs

Source:: Hard Costs of Heartbleed

Online Certificate Status Protocol (OCSP)



Online Certificate Status Protocol (OCSP)

Privacy Leak: Reveal browsing habits to CA

Fail Open: Too slow and unreliable to be checked for every connection

Overcomplicated: Onerous requirement that provide little security value

Expensive: Let's Encrypt receives more OCSP requests than for all ACME endpoints by an order of magnitude.

Source: ServerCert WG Archive

Simplified Revocation

1. **Get rid of OCSP**

   Don't waste time enforcing requirements that don't add security value.

   Reduce costs and spend that effort in higher-impact areas.

2. **Mandate CRLs**

   Browser's can distribute revocation information consumed from crawling CRLs (CRLite)

3. **Reduce the need for revocation by reducing certificate liftimes**

   ✨***Automation***✨ *unlocks short-lived certs*

Simplify requirements

Ballot SC-063: Make OCSP optional, require CRLs, and incentivize automation.

Enable browser-mediated revocation

Achieve ***security*** via ***agility***

Note: Ballot passed on July 13th, 2023

From **overcomplicated** to **simple**

...leveraging ✨***Automation***✨ to go from **glacial** to **agile**

...helps us make the Web PKI **more secure**

Improve security

Improve experience

Improve the Web PKI

public@ccadb.org

Google

Didn't Chrome Already Have a Root Store?

Chrome Security

@davidcadrian | @dadrian.io

Sep 2023