Trust and the WebPKI

CS249i

Stanford University

censys

Certificates | 2423c7d936b4342eaa4b015614f6167a57871cb434e21545bf87b6d09dd9c9c9 | Search | ZD

zakird.com

Certificate | Trust | ZLint | PEM | Raw Data | Explore

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            03:00:dd:56:14:c0:63:85:ef:75:00:f4:08:da:9f:e5:a5:60
    Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            commonName                = PRINTABLESTRING:R3
            organizationName          = PRINTABLESTRING:Let's Encrypt
            countryName               = PRINTABLESTRING:US
        Validity
            Not Before: Jan  9 22:06:26 2023 GMT
            Not After : Apr  9 22:06:25 2023 GMT
        Subject:
            commonName                = PRINTABLESTRING:zakird.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
```

Symantec / DigiCert operated root c38dcb389593…

```
commonName          = UTN-USERFirst-NetworkApplications
orgUnitName         = http://www.usertrust.com
orgName             = The USERTRUST Network
localityName        = Salt Lake City
stateOrProvinceName = UT
countryName         = US
```

Comodo / Sectigo operated root 43f257412d44…

```
commonName          = UTN-USERFirst-Client Authentication and Email
orgUnitName         = http://www.usertrust.com
orgName             = The USERTRUST Network
localityName        = Salt Lake City
stateOrProvinceName = UT
countryName         = US
```

Figure 2: **Misleading Names**—The Subject fields of two roots previously operated by Symantec/DigiCert and Comodo/Sectigo illustrate that 1) the names in CA certificates do not reflect their operators, and 2) similar certificate names have no bearing on shared control.

	Organization	#	Symantec Affiliation
Blacklisted Roots	Symantec	10	–
	VeriSign	14	Acquired by Symantec (2010) [3]
	TC TrustCenter	10	Acquired by Symantec (2010) [75]
	GeoTrust	8	Acquired by VeriSign (2006) [53]
	Equifax	4	Acquired by GeoTrust (2001) [2]
	UserTrust	1	GeoTrust partnership (2001) [76]
	Thawte	10	Acquired by VeriSign (1999) [34]
	RSA Data Sec.	1	Spun out VeriSign (1995) [33]
Whitelisted	Apple	6	Sub-CA intermediates
	Google	1	Sub-CA intermediates
	DigiCert	2	Cross-signed DigiCert roots
	DigiCert	2	Transition intermediates

Table 1: **Symantec Distrust**—Blacklisting of Symantec-controlled roots involved 58 root certificates [4] with 8 separate orgs. in their X.509 Subject field. These orgs. are linked through a scattered history of corporate spin-offs and acquisitions.

[What's in a Name? Exploring CA Certificate Control]

- Zane Ma, Joshua Mason, Manos Antonakakis, Zakir Durumeric, and Michael Bailey
- *USENIX Security Symposium*, August 2021

Common CA Database

Home | Policy | For CAs | For Root Stores | Resources

m CCADB

Common CA Database

The Common CA Database (CCADB) is a repository of information about externally operated Certificate Authorities (CAs) whose root and intermediate certificates are included within the products and services of CCADB root store members. Root store operators participate in the CCADB to improve security, transparency, and interoperability. The CCADB is used by a number of different root store operators to manage their root stores, but it is run by Mozilla.

moz://a Microsoft Google

Root Stores

Let's Start at the Top — Root Stores

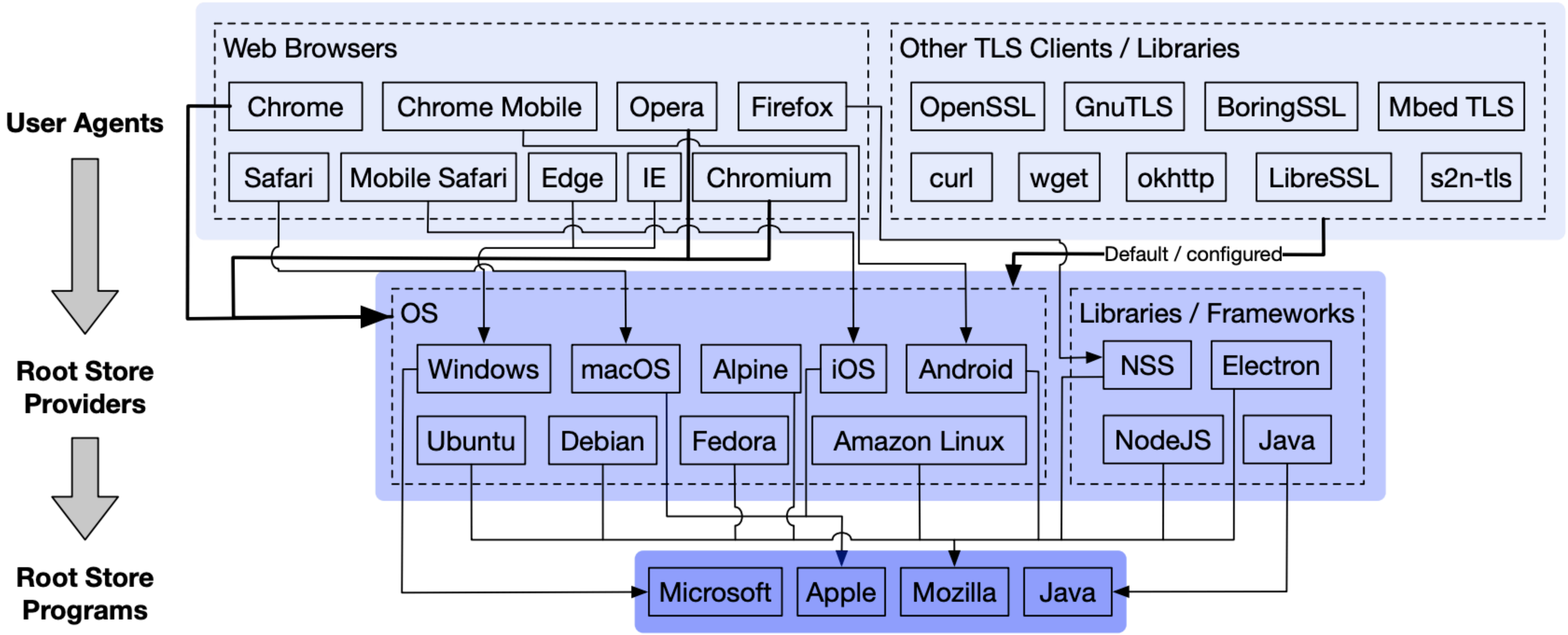


Figure 2: Root Store Ecosystem—The TLS root store ecosystem is an inverted pyramid, with a majority of clients trusting one of four root families.

[Tracing Your Roots: Exploring the TLS Trust Anchor Ecosystem]

- Zane Ma, James Austgen, Joshua Mason, Zakir Durumeric, and Michael Bailey
- *ACM Internet Measurement Conference* (IMC), November 2021

Change Over Time

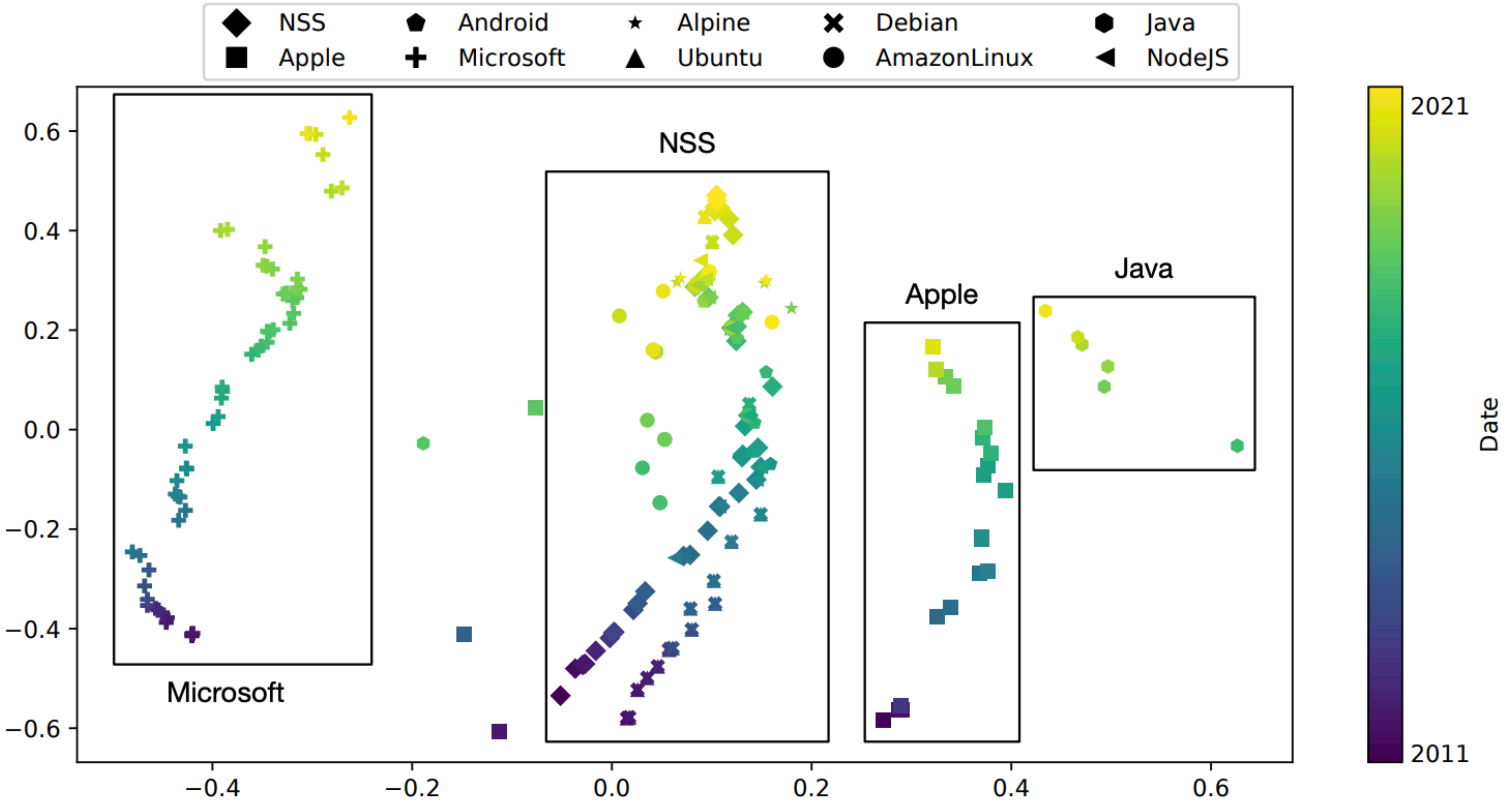


Figure 1: Root Store Similarity—Performing MDS on the Jaccard distance between root store providers from 2011–2021 illustrates four distinct clusters of roots. From left to right: Microsoft, NSS-like, Apple, Java.

🖨️ Certificate Issuance and Some History

Historical Validation and Issuance

Goals:

1) verify that a network identifier (i.e., IP address or DNS Name) controls some cryptographic public key

2) generate a certificate that attests to this linkage.

How to verify? What does “control” mean?

Historically... (around 2012...)

Confirming the Applicant as the Domain Name Registrant directly with the Domain Name Registrar; Communicating directly with Registrant via address, email, or telephone number provided by the Registrar;

Communicating directly with the Registrant using the contact information listed in the WHOIS record's "registrant", "technical", or "administrative" field;

Communicating with the Domain's administrator using an email address created by pre-pending 'admin', 'administrator', 'webmaster', 'hostmaster', or 'postmaster' followed by the Domain Name;

Relying upon a Domain Authorization Document;

Having the Applicant demonstrate practical control over the FQDN by making an agreed-upon change to information found on an online Web page identified by a uniform resource identifier containing the FQDN;

Using any other method of confirmation, provided that the CA maintains documented evidence that it establishes that the Applicant is the Registrant or has control over the FQDN to at least the same level of assurance as those methods previously described.

Since then...

(If CA == DNS Registrar) Confirming the Applicant as the Domain Name Registrant directly with the Domain Name Registrar;

Communicating directly with Registrant via address, email, or telephone number provided by the Registrar; Communicating directly with the Registrant using the contact information listed in the WHOIS record's "registrant", "technical", or "administrative" field; **(based on WHOIS info)**

Communicating with the Domain's administrator using an email address created by pre-pending 'admin', 'administrator', 'webmaster', 'hostmaster', or 'postmaster' followed by the Domain Name;

~~Relying upon a Domain Authorization Document;~~ **Removed**

Having the Applicant demonstrate practical control over the FQDN by making an agreed-upon change to information found on an online Web page identified by a uniform resource identifier containing the FQDN;

~~Using any other method of confirmation, provided that the CA maintains documented evidence that it establishes that the Applicant is the Registrant or has control over the FQDN to at least the same level of assurance as those methods previously described.~~

Certificates were not free!

Certificate Authority	Mid-2015 Prices		Mid-2019 Prices	
	Single	Wildcard	Single	Wildcard
GoDaddy [45, 46]	$69	$332	$79	$369
Comodo/Sectigo [31, 85]	$76	$404	$92	$422
GeoTrust [43, 44]	$149	$499	$149	$688
DigiCert [32, 33]	$195	$595	$207	$653
Symantec [92, 93]	$399	$1999	$399	$1999

Table 1: Prices for a one-year certificate for non-free CAs with the largest market shares. Single domain offerings are domain-validated; wildcard offerings sometimes require organization validation. The 2015 prices are from shortly before Let's Encrypt began offering service to the public.

2011: DigiNotar Certificate Authority

Dutch Certificate Authority

Compromised in September 2011 — issued fraudulent certificates

Dutch Government took over operational control. Declared bankruptcy within three weeks — after distrusted by major browsers

DigiNotar

Internet Trust Services

2011: TurkTrust Distrust

Turkish CA issued fraudulent certificate for google.com

“*Microsoft is aware of active attacks using one fraudulent digital certificate issued by TURKTRUST Inc., which is a CA present in the Trusted Root Certification Authorities Store,*” an advisory from Microsoft noted.

Mozilla distrusted CA.

2013: CAs Discovered through Internet Scanning

Identified 1,800 CA certificates belonging to 683 organizations

- Including religious institutions, libraries, non-profits, financial institutions, governments, and hospitals

- More than 80% of organizations controlling a CA certificate aren't commercial certificate authorities

More than half of the certificates were provided by the German National Research and Education Network (DFN)

All major browser roots are selling intermediates to third-party organizations without any constraints

Analysis of the HTTPS Certificate Ecosystem
Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman
ACM Internet Measurement Conference (IMC), October 2013

2013: CAs Discovered through Internet Scanning

The largest commercial provider of intermediate certificates is GTE CyberTrust Solutions, Inc., a subsidiary of Verizon Business, which has provided intermediate signing certificates to 49 third-party organizations ranging from Dell Inc. to Louisiana State University. Comodo (under the name *The USERTRUST Network*) provided intermediates to 42 organizations and GlobalSign to 20. We also saw a number of commercial authorities that provided a smaller number of certificates to seemingly unrelated entities. For example, VeriSign, Inc. provided intermediates for Oracle, Symantec, and the U.S. Government; SwissSign AG provided certificates for Nestle, Trend Micro, and other Swiss companies; StartCom Ltd.

Let's Encrypt

A nonprofit Certificate Authority providing TLS certificates to **300 million** websites.

Read all about our nonprofit work this year in our 2022 Annual Report.

Get Started | Sponsor

FROM OUR BLOG

Jan 19, 2023

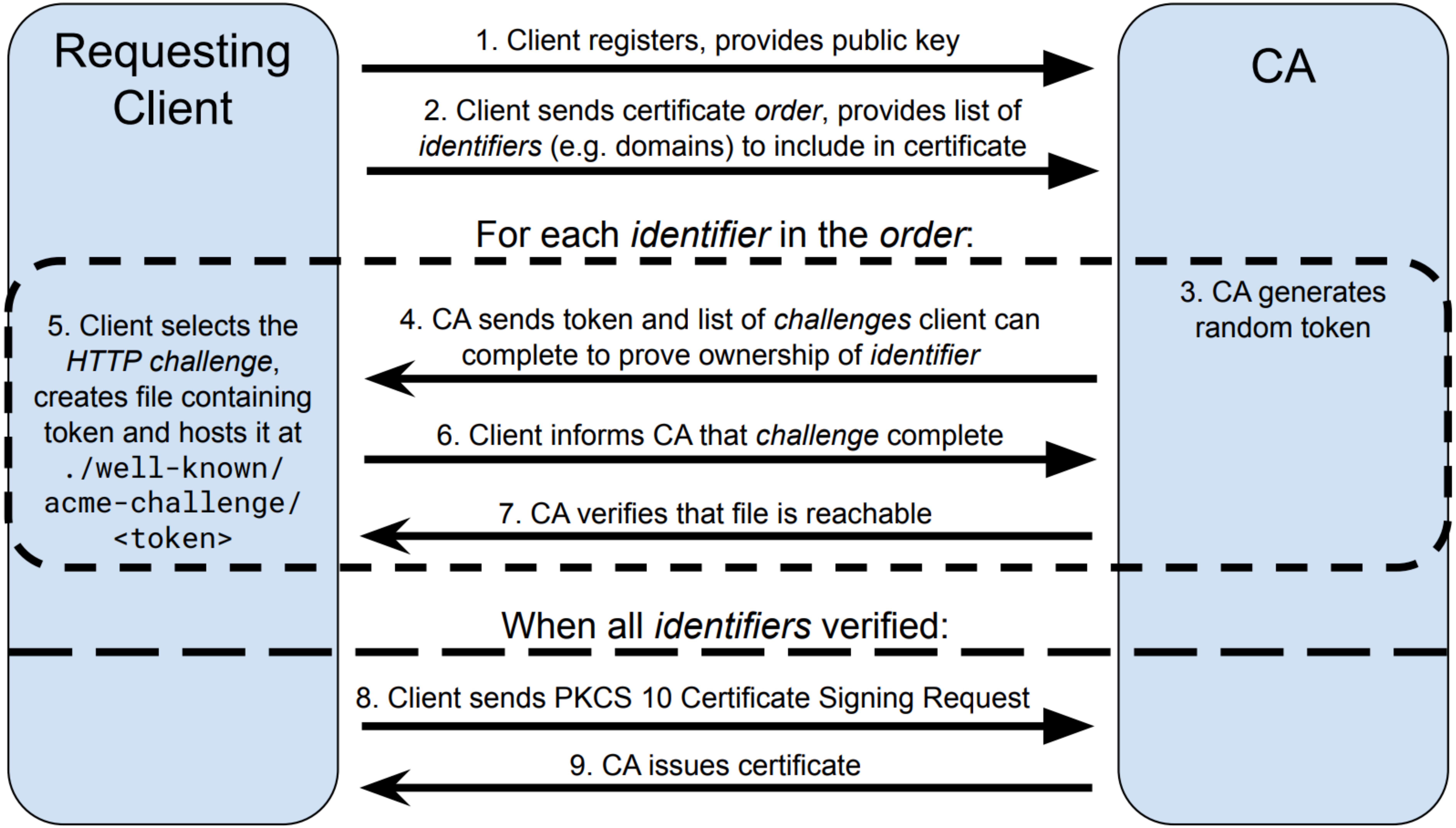
Thank you to our 2023 renewing sponsors

Let's Encrypt is a nonprofit service and our longtime and renewing sponsors play a major role in making that possible.

Read more

MAJOR SPONSORS AND FUNDERS

ACME Protocol



Validation Methods

There are currently three specified challenge types, all of which are supported by Let's Encrypt:

(1) The **HTTP** challenge requires the applicant to serve an object containing a CA-provided random value at a specific HTTP URL at the domain. The CA makes GET requests for the URL and verifies that the correct object is returned.

(2) The **DNS** challenge requires the applicant to provision a DNS record at `_acme-challenge.<domain>` containing a CA-provided random value. The CA fetches this record and verifies that its content is correct.

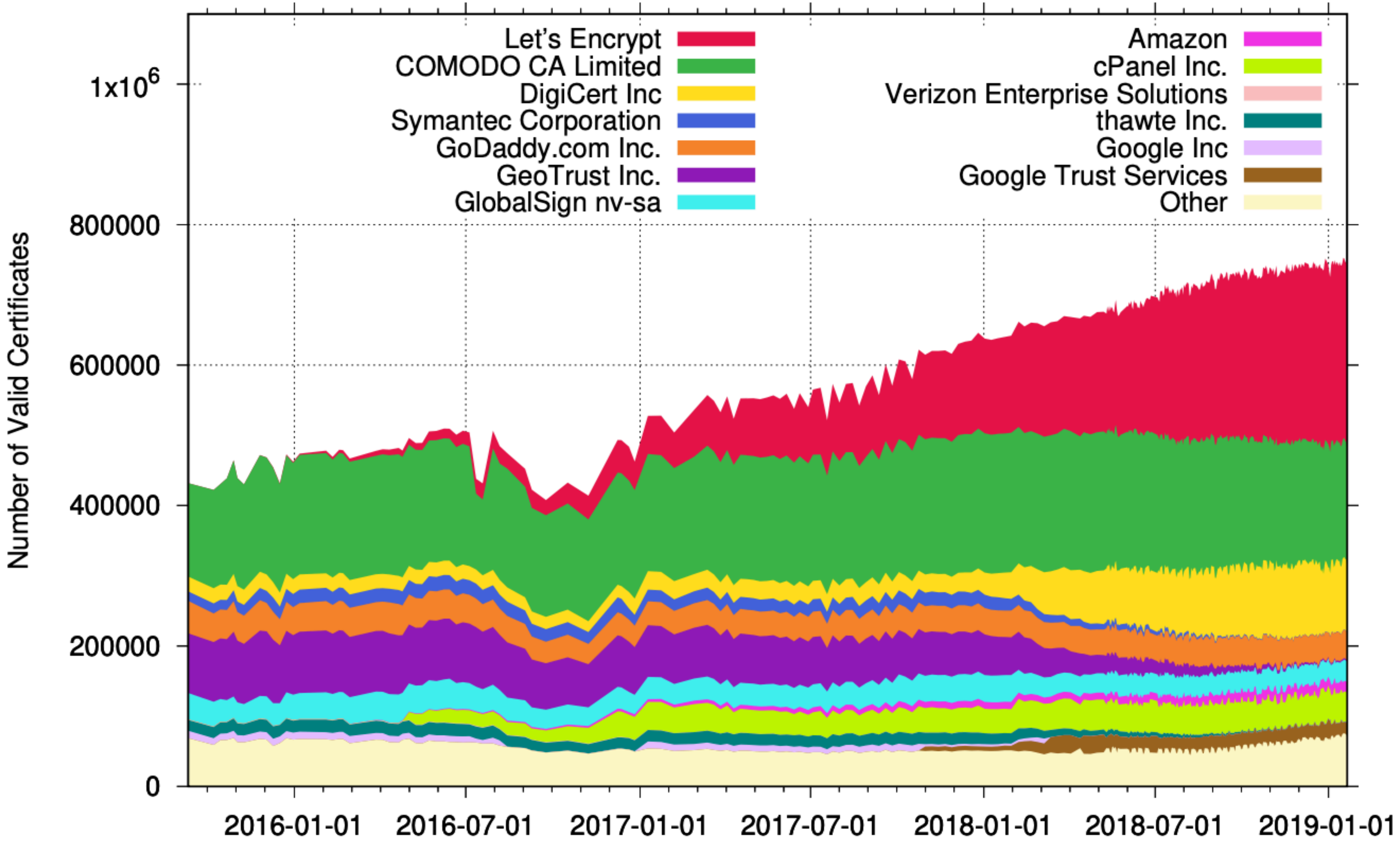
(3) The **TLS-ALPN** challenge requires the applicant to configure a TLS server to respond to a TLS ClientHello message containing a specific ALPN value and an ACME-specific TLS extension [42, 87]. The TLS server must then present a self-signed certificate containing a CA-provided random value and correctly complete the TLS handshake.

Since then... automated issuance

	Contact Method		Contact Info Source
A. Manual Contact	Phone Fax Postal Mail	SMS Email	WHOIS DNS (TXT, CAA) ‘admin’-like email

Automatable

B. Random Token	Step 1. CA sends token to client	Step 2. CA retrieves token from: DNS server HTTP / TLS server



(a) CA market share

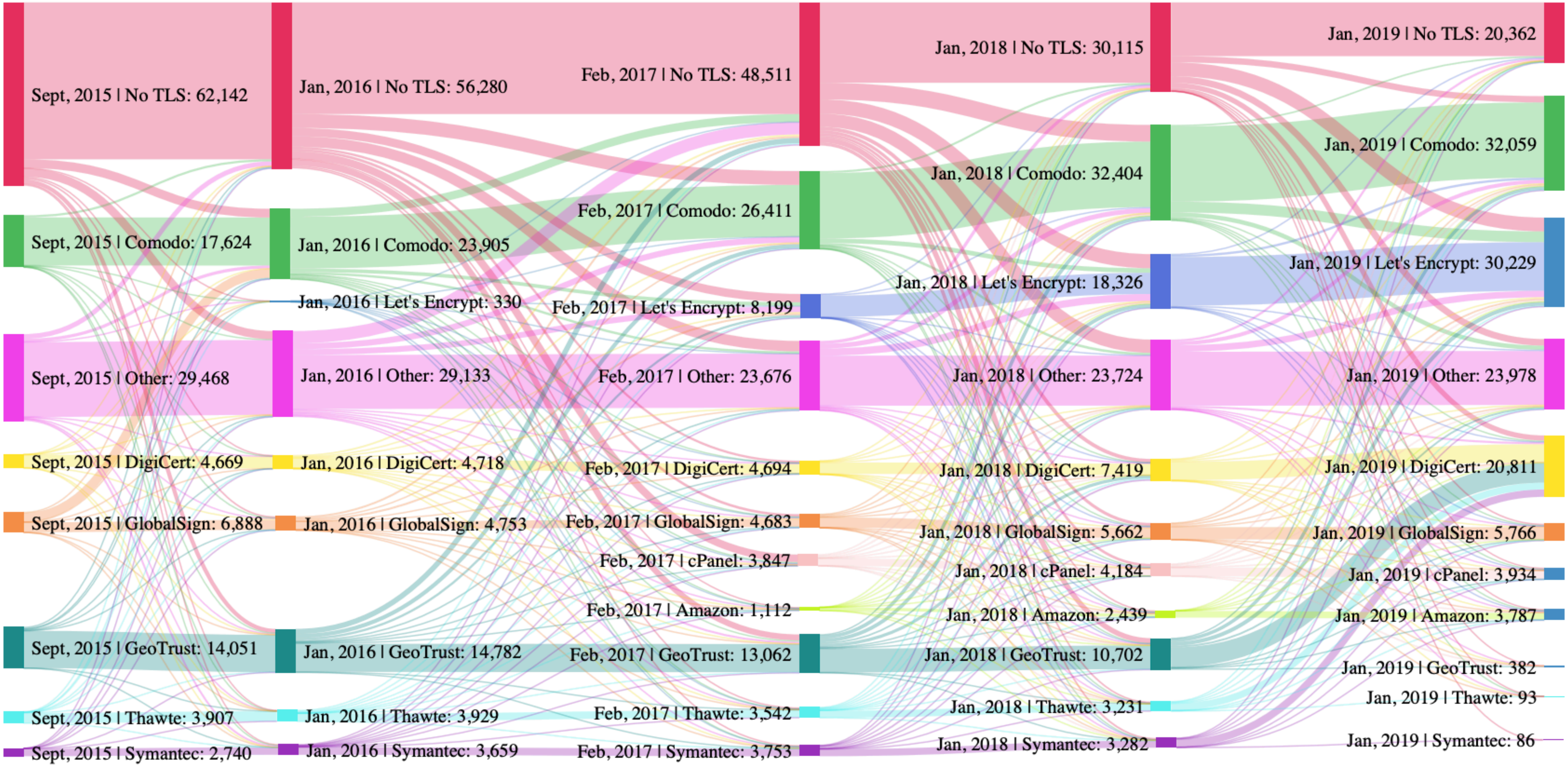


Figure 8: Certificate authority flow among stable, popular sites. We track CA choice for 141K domains over five snapshots, from 7/2015 to 1/2019. The included sites are those that were ranked in the Alexa Top Million at every snapshot, and so are likely more popular and long-lived than the top million overall.

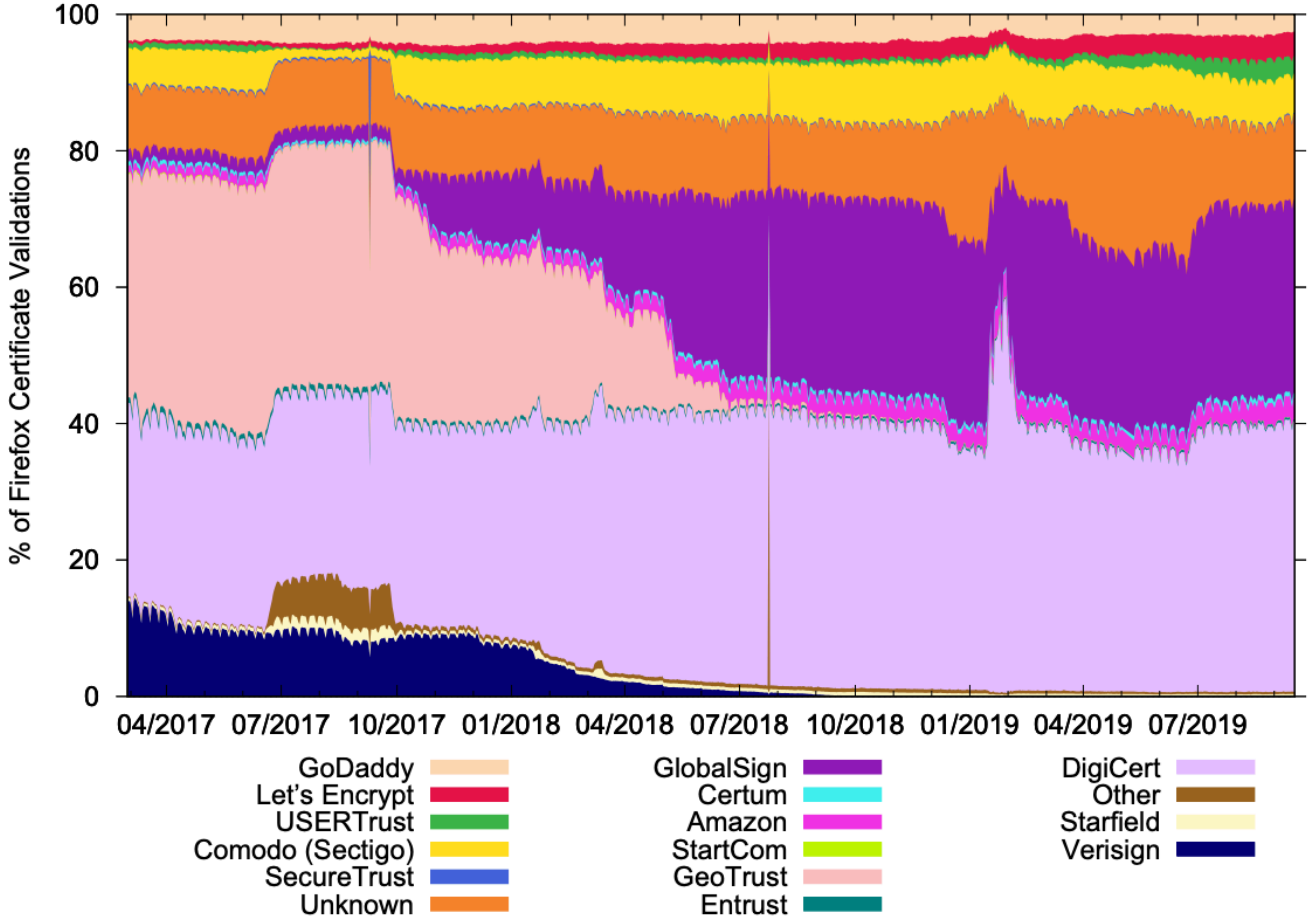
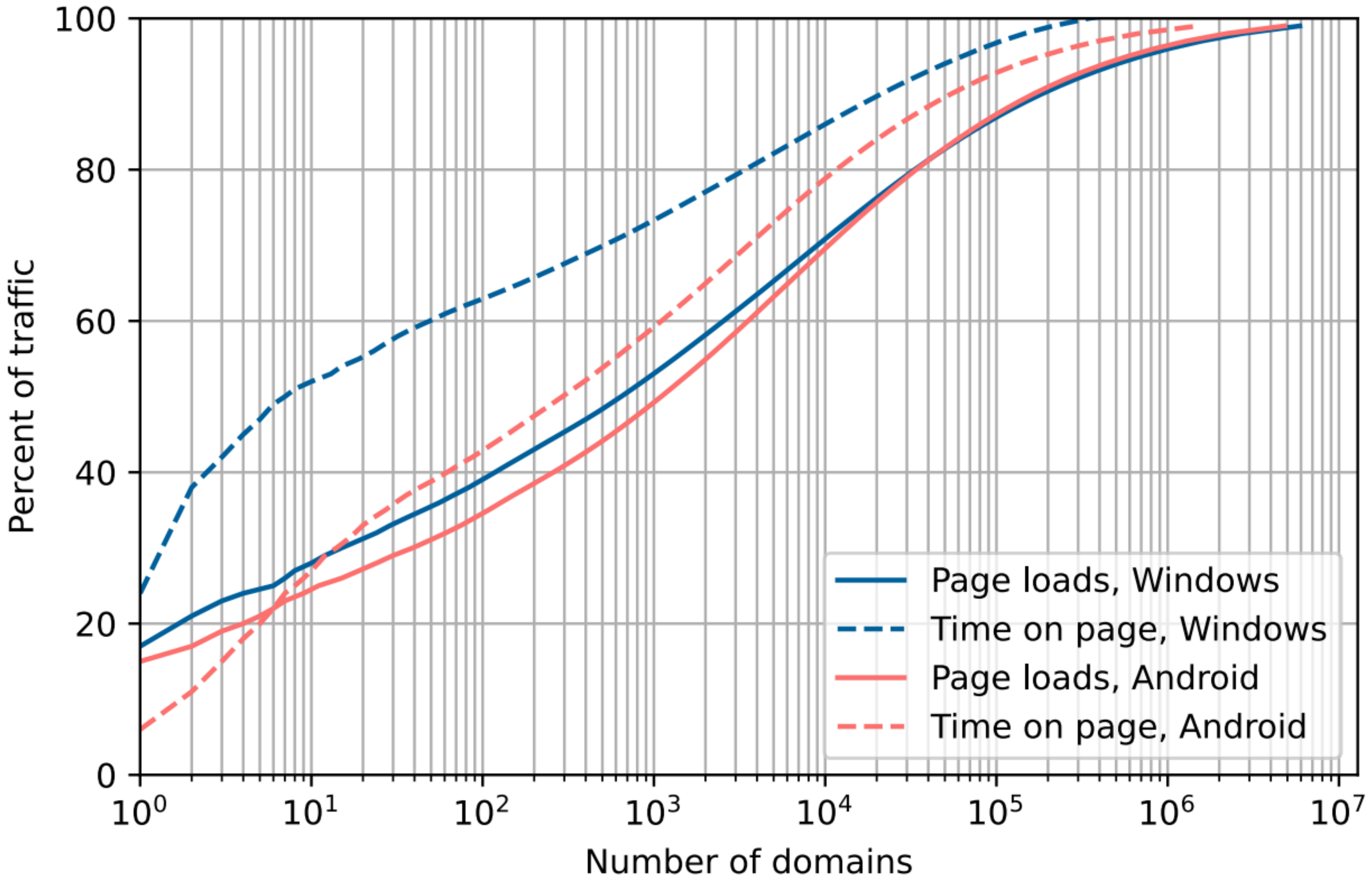


Figure 5: Firefox HTTPS connections by trust anchor. We show the trust anchors responsible for authenticating full TLS handshakes by Firefox Beta users. Let's Encrypt has become the fourth largest known CA.



A World Wide View of Browsing the World Wide Web
Kimberly Ruth, Aurore Fass, Jonathan Azose, Mark Pearson, Emma Thomas, Caitlin Sadowski, and Zakir Durumeric
ACM Internet Measurement Conference (IMC), October 2022

parsed.issuer_dn.raw	Certificates	
C=US, O=Let's Encrypt, CN=R3	386,737,770	51.18%
C=US, O=Cloudflare\, Inc., CN=Cloudflare Inc ECC CA-3	63,758,897	8.44%
C=US, O=Google Trust Services LLC, CN=GTS CA 1P5	39,029,449	5.16%
C=US, O=Amazon, OU=Server CA 1B, CN=Amazon	30,646,721	4.06%
C=US, O=Let's Encrypt, CN=E1	29,673,776	3.93%
C=US, ST=TX, L=Houston, O=cPanel\, Inc., CN=cPanel\, Inc. Certification Authority	28,955,770	3.83%
C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited, CN=Sectigo RSA Domain Validation Secure Server CA	27,686,231	3.66%
C=US, O=DigiCert Inc, OU=www.digicert.com, CN=Encryption Everywhere DV TLS CA - G1	20,653,171	2.73%
C=US, O=Cloudflare\, Inc., CN=Cloudflare Inc RSA CA-2	16,094,882	2.13%
C=GB, ST=Greater Manchester, L=Salford, O=Sectigo Limited, CN=Sectigo ECC Domain Validation Secure Server CA	11,421,489	1.51%
C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA	11,120,657	1.47%
C=US, ST=Arizona, L=Scottsdale, O=GoDaddy.com\, Inc., OU=http://certs.godaddy.com/repository/, CN=Go Daddy Secure Certificate Authority - G2	9,358,404	1.24%

Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web

Josh Aas*
Let's Encrypt

Richard Barnes*
Cisco

Benton Case
Stanford University

Zakir Durumeric
Stanford University

Peter Eckersley*
Electronic Frontier Foundation

Alan Flores-López
Stanford University

J. Alex Halderman*†
University of Michigan

Jacob Hoffman-Andrews*
Electronic Frontier Foundation

James Kasten*
University of Michigan

Eric Rescorla*
Mozilla

Seth Schoen*
Electronic Frontier Foundation

Brad Warren*
Electronic Frontier Foundation

ABSTRACT

Let's Encrypt is a free, open, and automated HTTPS certificate authority (CA) created to advance HTTPS adoption to the entire Web. Since its launch in late 2015, Let's Encrypt has grown to become the world's largest HTTPS CA, accounting for more currently valid certificates than all other browser-trusted CAs combined. By January 2019, it had issued over 538 million certificates for 223 million domain names. We describe how we built Let's Encrypt, including the architecture of the CA software system (Boulder) and the structure of the organization that operates it (ISRG), and we discuss lessons learned from the experience. We also describe the design of ACME, the IETF-standard protocol we created to automate CA–server interactions and certificate issuance, and survey the diverse ecosystem of ACME clients, including Certbot, a software agent we created to automate HTTPS deployment. Finally, we measure Let's Encrypt's impact on the Web and the CA ecosystem. We hope that the success of Let's Encrypt can provide a model for further enhancements to the Web PKI and for future Internet security infrastructure.

CCS CONCEPTS

• **Networks** → *Web protocol security*; • **Security and privacy** →

1 INTRODUCTION

HTTPS [78] is the cryptographic foundation of the Web, providing an encrypted and authenticated form of HTTP over the TLS transport [79]. When HTTPS was introduced by Netscape twenty-five years ago [51], the primary use cases were protecting financial transactions and login credentials, but users today face a growing range of threats from hostile networks—including mass surveillance and censorship by governments [99, 106], consumer profiling and ad injection by ISPs [30, 95], and insertion of malicious code by network devices [68]—which make HTTPS important for practically every Web request. Many cryptographic flaws in TLS have been discovered and mitigated (e.g., [11, 13, 17, 23, 37, 69]), but low adoption of HTTPS posed an even starker risk: as recently as 2015, 55–70% of browser page loads used *plaintext* HTTP [47].

A major barrier to wider HTTPS adoption was that deploying it was complicated, expensive, and error-prone for server operators [22, 57]. Most of the difficulty involved interactions with Certificate Authorities (CAs), entities trusted by Web browsers to validate a server's identity and issue a digitally signed certificate binding the identity to the server's public key. (Modern TLS implementations have negligible performance overhead in typical applications [48,

Certificate Transparency and Ecosystem Health

Certificate Transparency...
What if we put every certificate in a merkle tree?

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

Obsoleted by: 9162 EXPERIMENTAL
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 6962
Category: Experimental
ISSN: 2070-1721

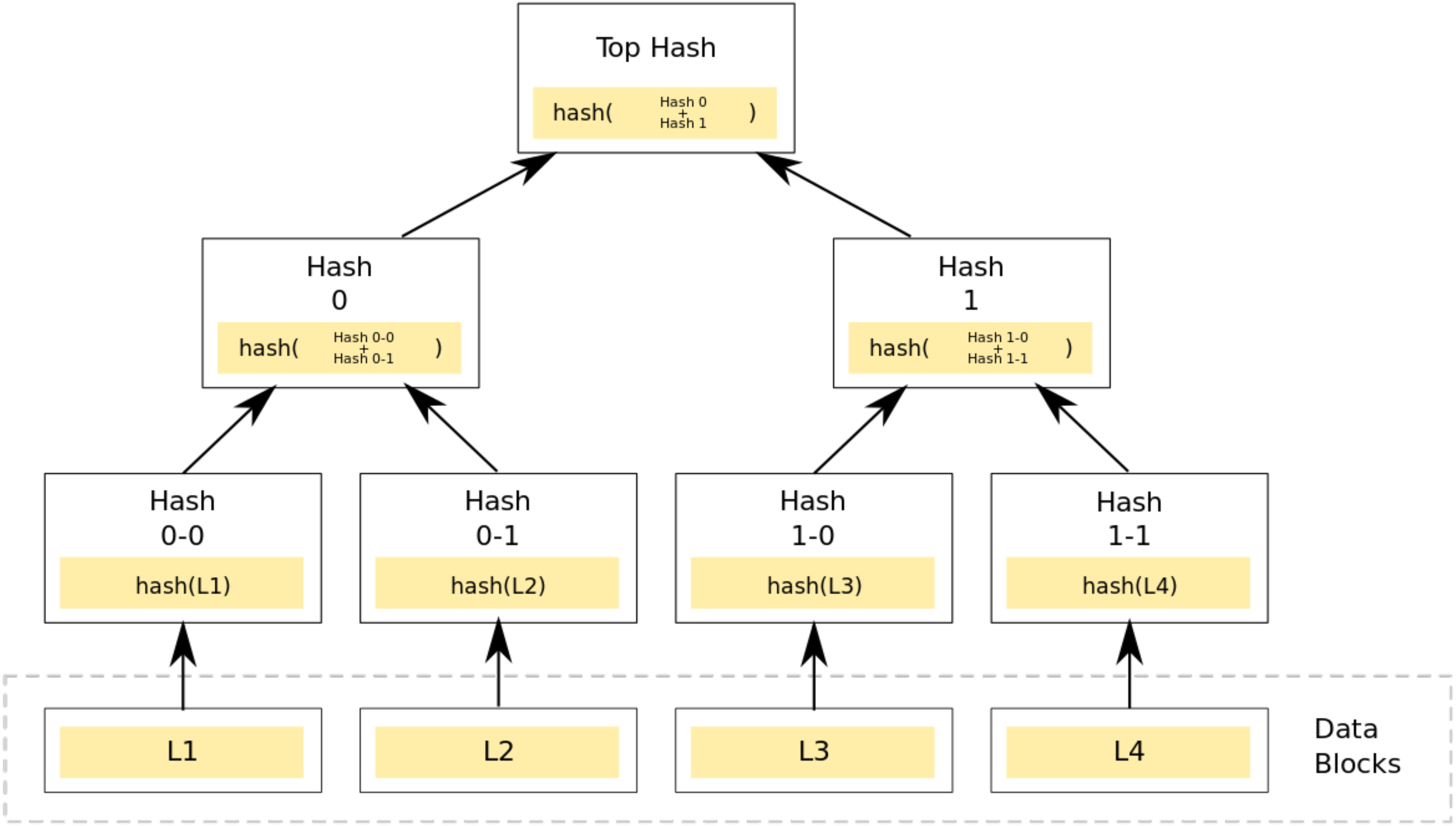
B. Laurie
A. Langley
E. Kasper
Google
June 2013

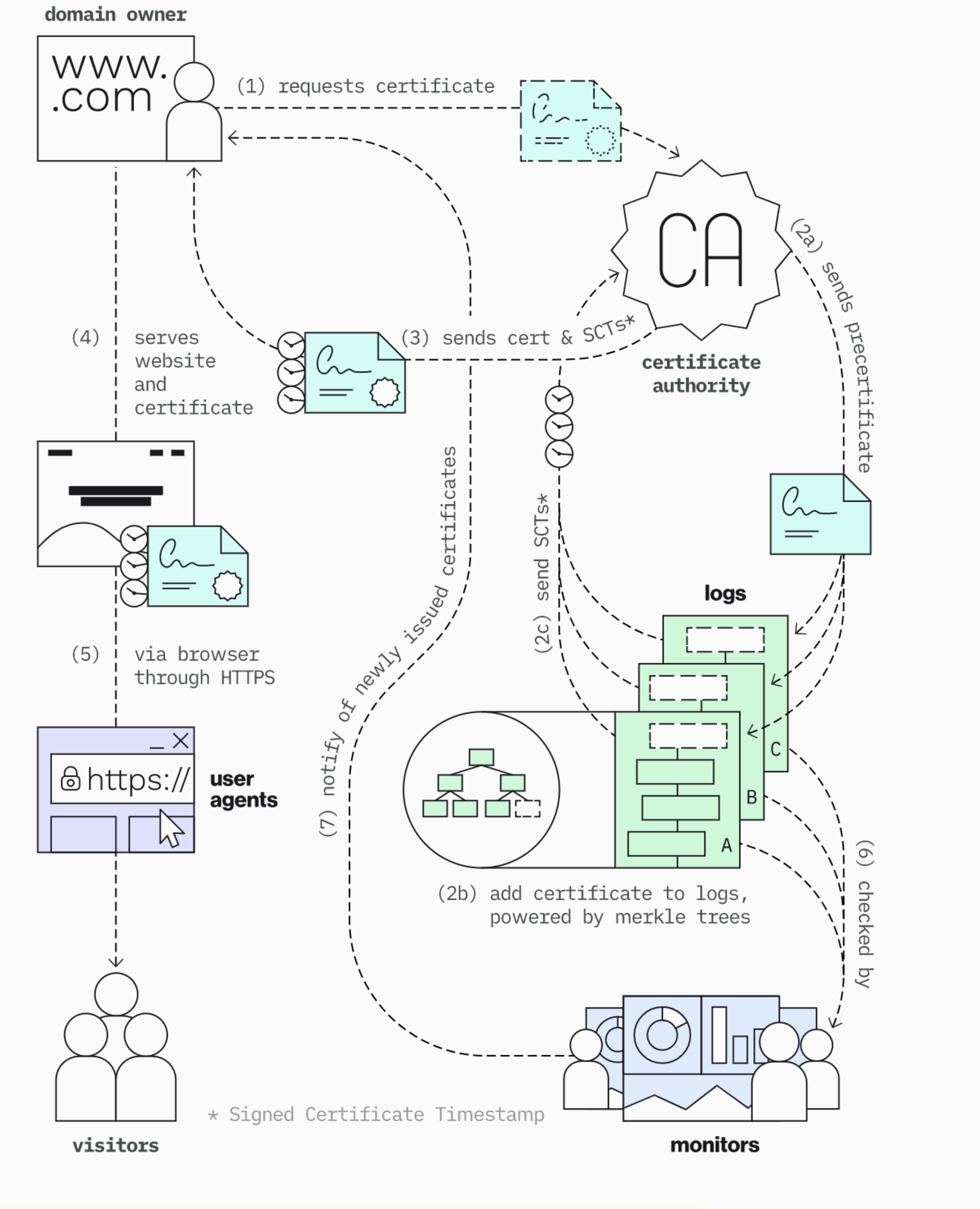
Certificate Transparency

Abstract

This document describes an experimental protocol for publicly logging the existence of Transport Layer Security (TLS) certificates as they are issued or observed, in a manner that allows anyone to audit certificate authority (CA) activity and notice the issuance of suspect certificates as well as to audit the certificate logs themselves. The intent is that eventually clients would refuse to honor certificates that do not appear in a log, effectively forcing CAs to add all issued certificates to the logs.

Logs are network services that implement the protocol operations for submissions and queries that are defined in this document.


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks


domain owner
www.
.com
(1) requests certificate
CA
certificate
authority
(2a) sends precertificate
(3) sends cert & SCTs*
(4) serves
website
and
certificate
(2c) send SCTs*
(7) notify of newly issued certificates
logs
C
B
A
(5) via browser
through HTTPS
https://
user
agents
(2b) add certificate to logs,
powered by merkle trees
(6) checked by
visitors
monitors
* Signed Certificate Timestamp

```go
func (l *InvalidCertificateVersion) Execute(
        cert *x509.Certificate) *LintResult {
    if cert.Version != 3 {
        return &LintResult{Status: Error}
    }
    return &LintResult{Status: Pass}
}

func init() {
    RegisterLint(&Lint{
        Name: "e_invalid_certificate_version",
        Description: "Certificates MUST be of
            type X.509 v3",
        Source: CABFBaselineRequirements,
        Citation: "CABF BR 7.1.1",
        EffectiveDate: util.CABV130Date,
        Lint: &InvalidCertificateVersion{},
    })
}
```

Code Block 1: **Example Lint**—Lints are self-contained Go functions that check for adherence with technical standards. This lint checks that a certificate uses the correct X.509 version.

Tracking Certificate Misissuance in the Wild

Deepak Kumar*, Zhengping Wang*, Matthew Hyder*, Joseph Dickinson*, Gabrielle Beck†, David Adrian†, Joshua Mason*, Zakir Durumeric*†‡, J. Alex Halderman†, Michael Bailey*

* University of Illinois Urbana-Champaign † University of Michigan ‡ Stanford University

Abstract—**Certificate Authorities (CAs) regularly make mechanical errors when issuing certificates. To quantify these errors, we introduce ZLint, a certificate linter that codifies the policies set forth by the CA/Browser Forum Baseline Requirements and RFC 5280 that can be tested in isolation. We run ZLint on browser-trusted certificates in Censys and systematically analyze how well CAs construct certificates. We find that the number errors has drastically reduced since 2012. In 2017, only 0.02% of certificates have errors. However, this is largely due to a handful of large authorities that consistently issue correct certificates. There remains a long tail of small authorities that regularly issue non-conformant certificates. We further find that issuing certificates with errors is correlated with other types of mismanagement and for large authorities, browser action. Drawing on our analysis, we conclude with a discussion on how the community can best use lint data to identify authorities with worrisome organizational practices and ensure long-term health of the Web PKI.**

I. Introduction

HTTPS depends on a supporting public key infrastructure (PKI) composed of hundreds of certificate authorities (CAs) that verify the identities of websites and issue digital certificates. To ensure compatibility between browsers and HTTPS-enabled websites, standards bodies like the IETF and CA/Browser Forum have developed policies that govern the digital certificates that CAs provide. Unfortunately, there is a long history of certificate authorities failing to adhere to accepted standards, due to both implementation errors and indifference. In this paper, we systematically analyze the errors that authorities make when constructing certificates and consider whether these errors can be used to predict more serious problems.

We begin by dissecting the policies set forth by RFC

in aggregate. Only 0.02% of certificates violate one of the two standards in 2017; 3.3% do not adhere to community best practices. This is a significant improvement from 2012 when more than 12% of certificates contained errors and nearly one third violated community recommendations. However, while the global misissuance rate is low, this is predominantly due to a handful of large authorities that consistently issue certificates without error. The three largest CAs by organization—Let's Encrypt, Comodo, and cPanel—signed 80% of the certificates in our dataset and have near-zero misissuance rates. Let's Encrypt, the largest CA by number of certificates issued, has a particularly stellar incident rate. Of the 37 million certificates the CA has signed, only 13 contain errors. None have warnings.

The bulk of misissuance is due to two classes of authorities. The first class is mid-sized authorities that make a variety of errors in a small percentage of their certificates. The second class is a long tail of small authorities that make the same errors in every issued certificate. Nearly half of the organizations in our dataset misissue more than 10% of certificates, and seventeen have made errors in every certificate. More than half of the errors and warnings in ZLint are triggered at least once. Most often, authorities fail to fully populate the Subject Alternative Names extension, encode the wrong type of data in the extension, or include invalid DNS names. Beyond individual certificates, we find that many organizations struggle to properly maintain OCSP/CRL responders. During our three week test period, the OCSP responders for 73 organizations (10%) failed every health check.

Next, in order to determine whether Lint data can be used to predict more serious issues, we investigate the correlation between the organizations that issue certificates containing errors, OCSP/CRL endpoint uptime, and browser removal. We

WELCOME TO THE CA/BROWSER FORUM

Information for the Public

Organized in 2005, we are a voluntary group of certification authorities (CAs), vendors of Internet browser software, and suppliers of other applications that use X.509 v.3 digital certificates for SSL/TLS, code signing, and S/MIME.

>read more

Information for Site Owners and Administrators

The CA/Browser Forum began in 2005 as part of an effort among certification authorities and browser software vendors to provide greater assurance to Internet users about the websites they visit by leveraging the capabilities of SSL/TLS certificates. In June 2007, the CA/Browser Forum adopted version 1.0 of the Extended Validation (EV) Guidelines. EV certificates are issued after extended steps to verify the identity of the entity behind the domain receiving the certificate. Internet browser software displays enhanced indication of that identity by changing the appearance of its display (i.e. colors, icons, animation, and/or additional website information).

Search

Type here

Search

- Ballot SC60: Membership of ZT Browser
- 2023-01-19 Minutes of the Server Certificate Working Group
- 2023-01-19 Minutes of the CA/Browser Forum Teleconference
- 2023-01-18 Minutes of the S/MIME Certificate Working Group
- 2023-01-12 Minutes of the Code Signing Certificate Working Group

Code Signing Forum
Network Security s S/MIME
Server Certificates

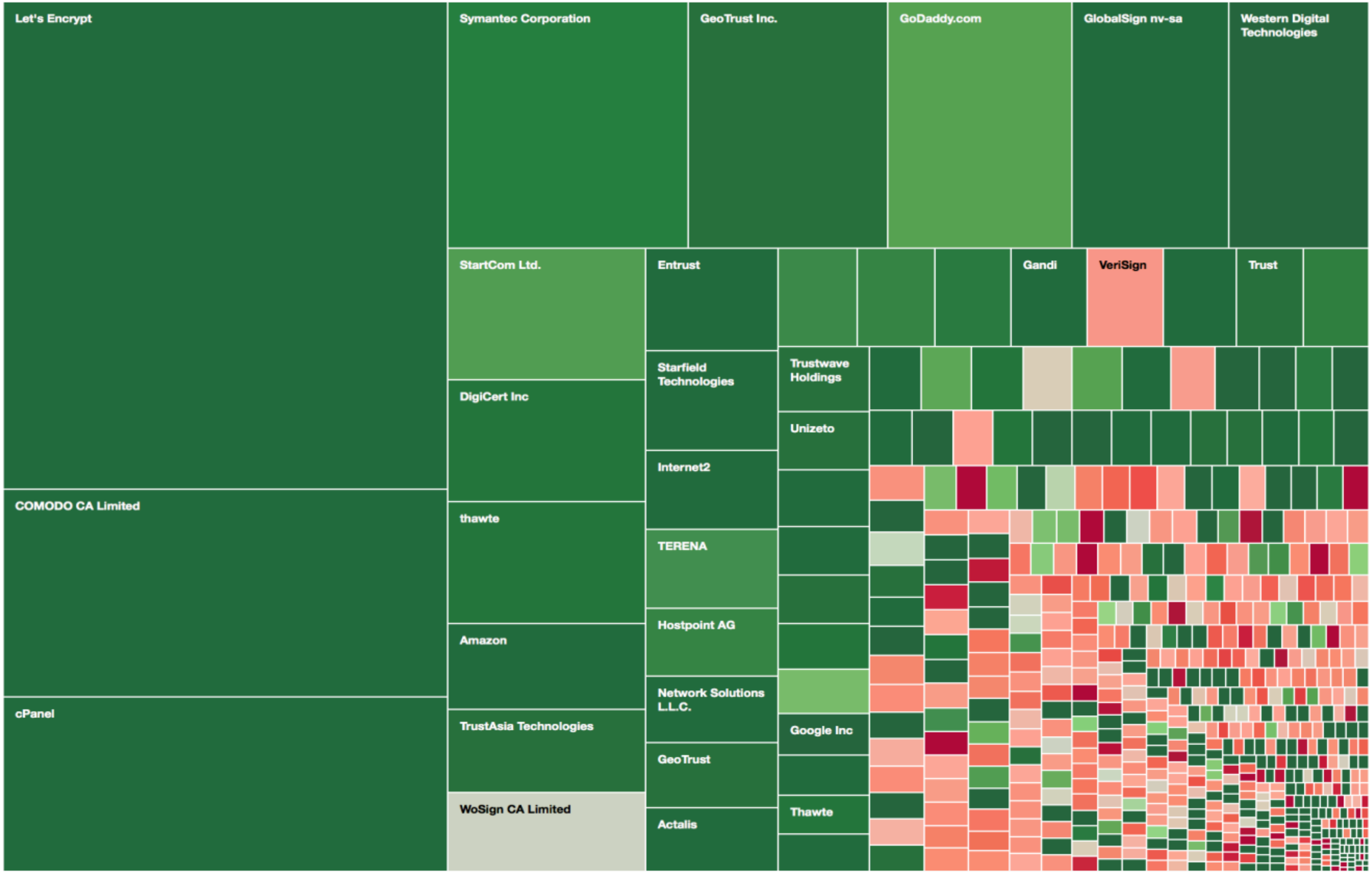


Fig. 5: **Percent ZLint Errors by Total Certificates Issued**—Large certificate authorities generally issue certificates with fewer ZLint errors than smaller authorities.

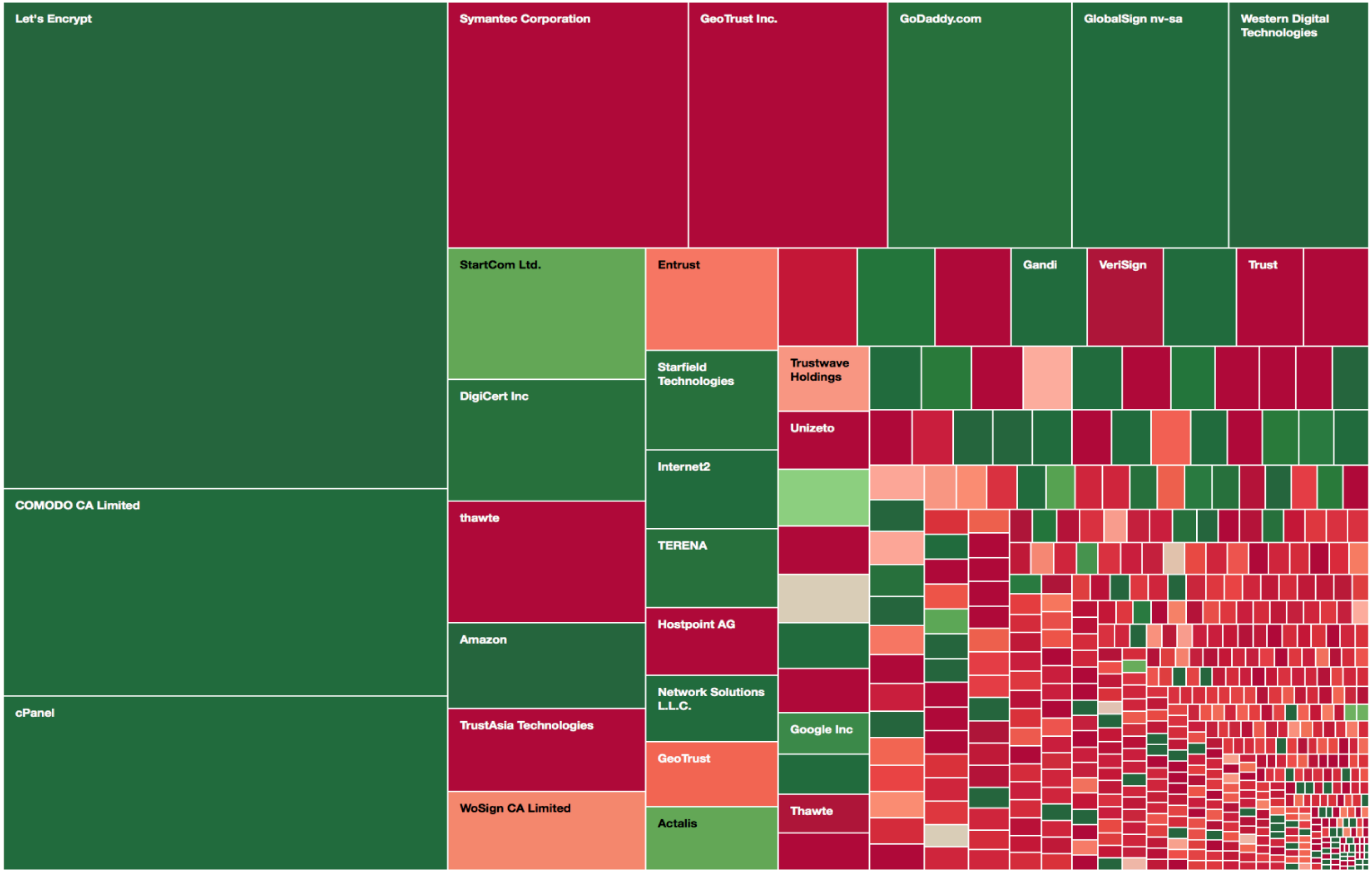


Fig. 6: **Percent ZLint Warnings by Total Certificates Issued**—ZLint warnings are more dispersed throughout the ecosystem, affecting both large and small players. A handful of large players, Symantec, GeoTrust, thawte, and TrustAsia, all issue more than 95% of their certificates with ZLint warnings.

Symantec Distrust

"Over a period of several years, Symantec willfully issues over 100 test certificates for 76 different domains without the authorization of the domain owners. This is discovered when Google's Certificate Transparency log monitor detects an unauthorized certificate for google.com in Certificate Transparency logs."

<INSERT TREMENDOUS DRAMA>

Symantec is distrusted by all major platforms due to general malfeasance.

And it just keeps going...

2016 - StartCom

Thijs Alkemade discovers that StartCom's brand new automated issuance API suffers from numerous flaws, including flaws that had previously been discovered and fixed by other CAs, that would allow attackers to obtain certificates for domains they don't control.

Cause: StartCom ignored developments in the standards community and instead chose to design their own, insecure automated issuance API.

During the ensuing investigation, it is revealed that StartCom had concealed their purchase by WoSign, another incompetent certificate authority.

Initially, StartCom announces that all certificates they issue will be logged to Certificate Transparency logs, but they are ultimately distrusted by all major platforms due to their malfeasance.

Root Store Lag

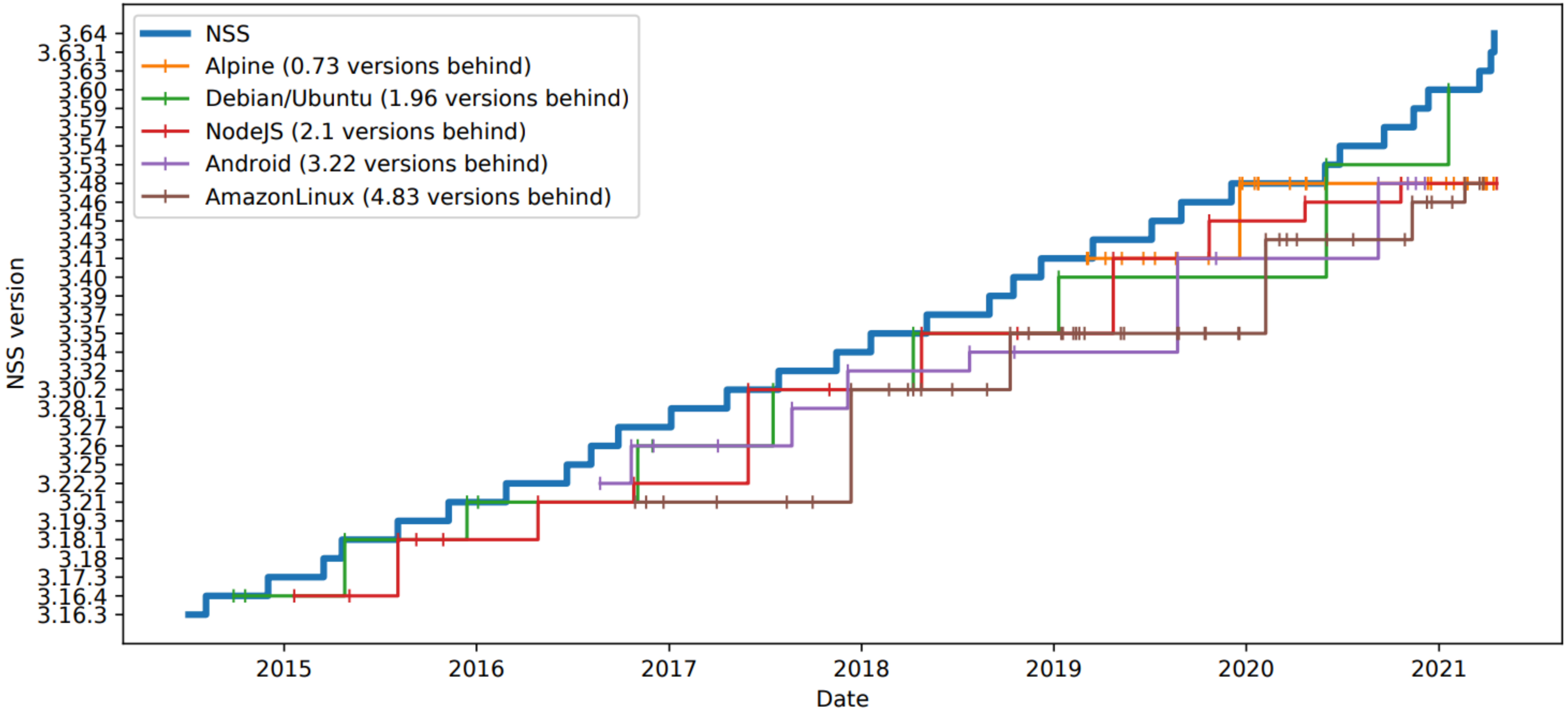


Figure 3: NSS derivative staleness—No derivative root stores match NSS's update regularity. Alpine Linux maintains closest parity to NSS, while AmazonLinux, on average, lags more than four substantial versions behind.

Root Store Lag

Root store	# Certs	Trusted until	Lag (days)
DigiNotar [101]		**2011-10-06**	
Microsoft	1	2011-08-30	-37
Apple	1	2011-10-12	6
Debian/Ubuntu	1	2011-10-22	16
CNNIC [78]		**2017-07-27**	
Apple	2	2015-06-30	-758
Android	1	2017-12-05	131
Debian/Ubuntu	2	2018-04-09	256
NodeJS	2	2018-04-24	271
AmazonLinux	2	2019-02-18	571
Microsoft	2	2020-02-26	944
StartCom [113]		**2017-11-14**	
Debian/Ubuntu	3	2017-07-17	-120
Microsoft	2	2017-09-22	-53
Android	3	2017-12-05	21
NodeJS	3	2018-04-24	161
AmazonLinux	3	2019-02-18	461
Apple	3	1 root still trusted	1,175+

Root store	# Certs	Trusted until	Lag (days)
WoSign [113]		**2017-11-14**	
Debian/Ubuntu	4	2017-07-17	-120
Microsoft	4	2017-09-22	-53
Android	4	2017-12-05	21
NodeJS	4	2018-04-24	161
AmazonLinux	4	2019-02-18	461
PSPProcert [38]		**2017-11-14**	
Debian/Ubuntu	1	2018-04-09	146
NodeJS	1	2018-04-24	161
AmazonLinux	1	2019-02-18	461
Certinomis [37]		**2019-07-05**	
NodeJS	1	2019-10-22	109
Alpine	1	2020-03-23	262
Debian/Ubuntu	1	2020-06-01	332
Android	1	2020-09-07	430
AmazonLinux	1	2021-03-26	630
Apple	1	2021-01-01*	577
Microsoft	1	Still trusted	607+

**Revoked via valid.apple.com at unknown date.*

Table 4: High severity removals—Comparison of root store responses to high severity NSS removals.

Announcing the Launch of the Chrome Root Program

Monday, September 19, 2022

In 2020, we announced we were in the early phases of establishing the Chrome Root Program and launching the Chrome Root Store.

The Chrome Root Program ultimately determines which website certificates are trusted by default in Chrome, and enables more consistent and reliable website certificate validation across platforms.

This post shares an update on our progress and how these changes help us better protect Chrome's users.

What's a root store or root program, anyway?

Chrome uses digital certificates (often referred to as "certificates," "HTTPS certificates," or "server authentication certificates") to ensure the connections it makes on behalf of its users are secure and private. Certificates are responsible for binding a domain name to a public key, which Chrome uses to encrypt data sent to and from the corresponding website.

As part of establishing a secure connection to a website, Chrome verifies that a recognized entity known as a "Certification Authority" (CA) issued its certificate. Certificates issued by a CA not recognized by Chrome or a user's local settings can cause users to see warnings and error pages.