Certificates

Daniel Zappala

CS 465 · Introduction to Security and Privacy · Fall 2026

BYU

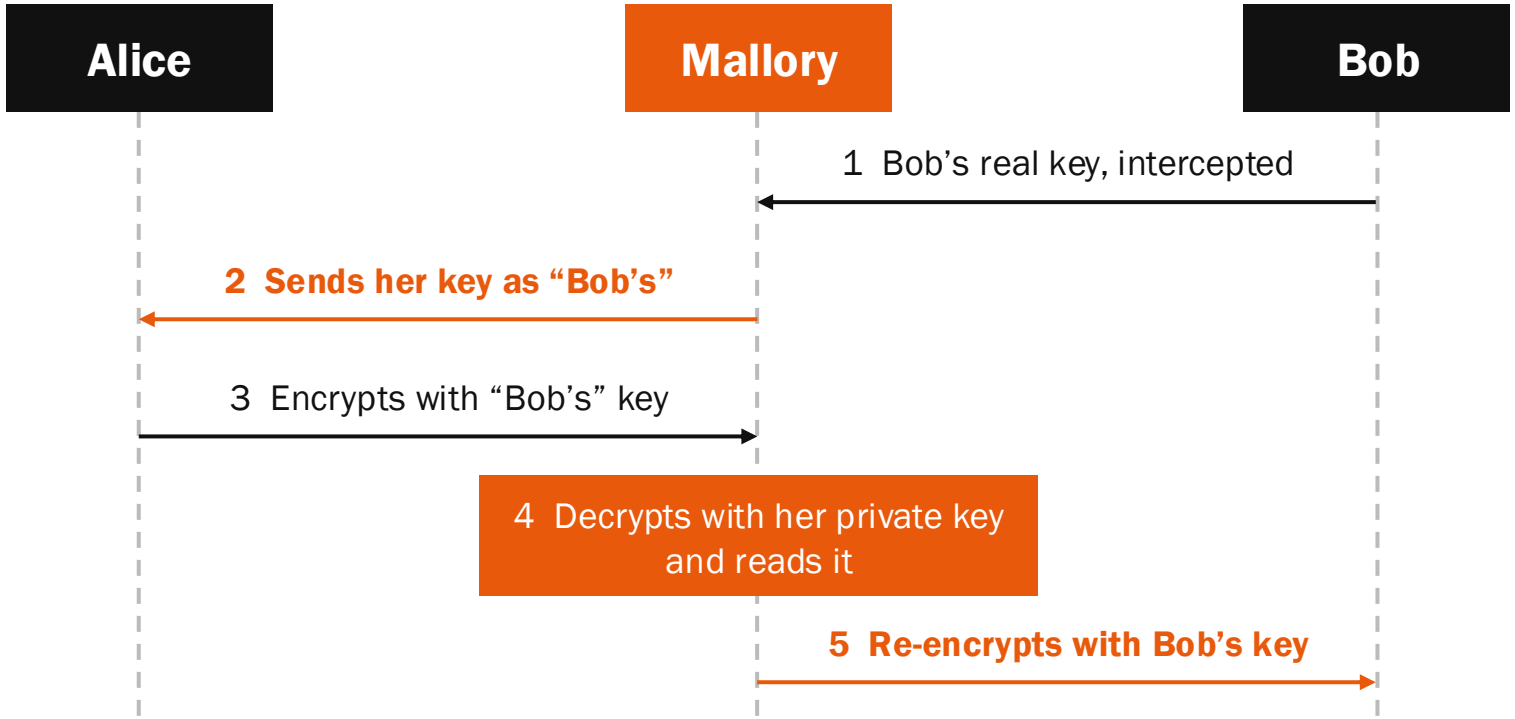

DRACO DORMIENS NUNQUAM TITILLANDUS
Hogwarts School
of Witchcraft and Wizardry
Headmaster: Albus Dumbledore
Order of Merlin, First Class, Grand Sorc., Chf. Warlock,
Supreme Mugwump, International Confed. of Wizards
Dear Mr. Potter,
We are pleased to inform you that you have been accepted at Hogwarts School of
Witchcraft and Wizardry.
ORDOR
KHAND
NURN
NAMHARAD

01

Public key certificates

A man-in-the-middle attack on public keys lets an attacker read messages meant for you

Public keys can be shared openly, but you must know who owns them



Bob decrypts normally, **so neither Alice nor Bob notices**

A certificate lets Alice verify that the key really belongs to Bob

Adapted from P. C. van Oorschot, Computer Security and the Internet, §8.1

A certificate is a CA's signature over a subject's name and public key

Subject

www.byu.edu

Public key

```
04 8b 2e 51 c7 9a 03 f4
1d 6c e0 92 7b 3a 5f c8 …
```

CA's signature
over every field above

Subject

The owner of the key: a hostname, an email address or an organization

Certification Authority (CA)

Signs to vouch that the public key belongs to the subject

Relying party

Trusts the CA and checks its signature with the CA's public key

The CA vouches for the name-to-key binding, not for the owner's honesty

Adapted from P. C. van Oorschot, Computer Security and the Internet, §8.1

01

Certificates

X.509 certificates contain the key, two names, a validity period and extensions

Field	Contents
Version	X.509 v3
Serial number	Unique ID for this certificate, used when revoking it
Issuer	Name of the CA that signed it
Validity period	Not-Before and Not-After dates
Subject	Name of the key's owner
Public key info	Algorithm and key value
Extensions	SAN list, basic constraints, key usage, CRL distribution points
Signature algorithm	Algorithm ID and parameters
Digital signature	Issuer's signature over all the fields above

Source: P. C. van Oorschot, Computer Security and the Internet, Table 8.1. Blue rows form the binding the CA signs

02

Issuing a certificate

02

Issuing

CAs certify keys for web servers, email addresses and organizations



Web server

Binds the key to a hostname

Your browser confirms it reached www.byu.edu, not an impostor



Email

Binds the key to an email address

Your mail client encrypts to, and verifies, the right person



Code signing

Binds the key to an organization: name, location, company officer

Your OS checks who published the software it runs

02

Issuing

Before issuing, a CA checks three kinds of evidence

1

Key ownership

The requester proves it holds the private key

Example

The CA sends a fresh challenge and checks the signed response

2

Control of the subject

The requester proves it controls the name in the certificate

Example

Serve a CA-chosen file on the domain, or answer mail sent to the address

3

Legal affiliation

For organization names, the requester belongs to the named legal entity

Example

Business registry lookup, call-back to a listed phone number

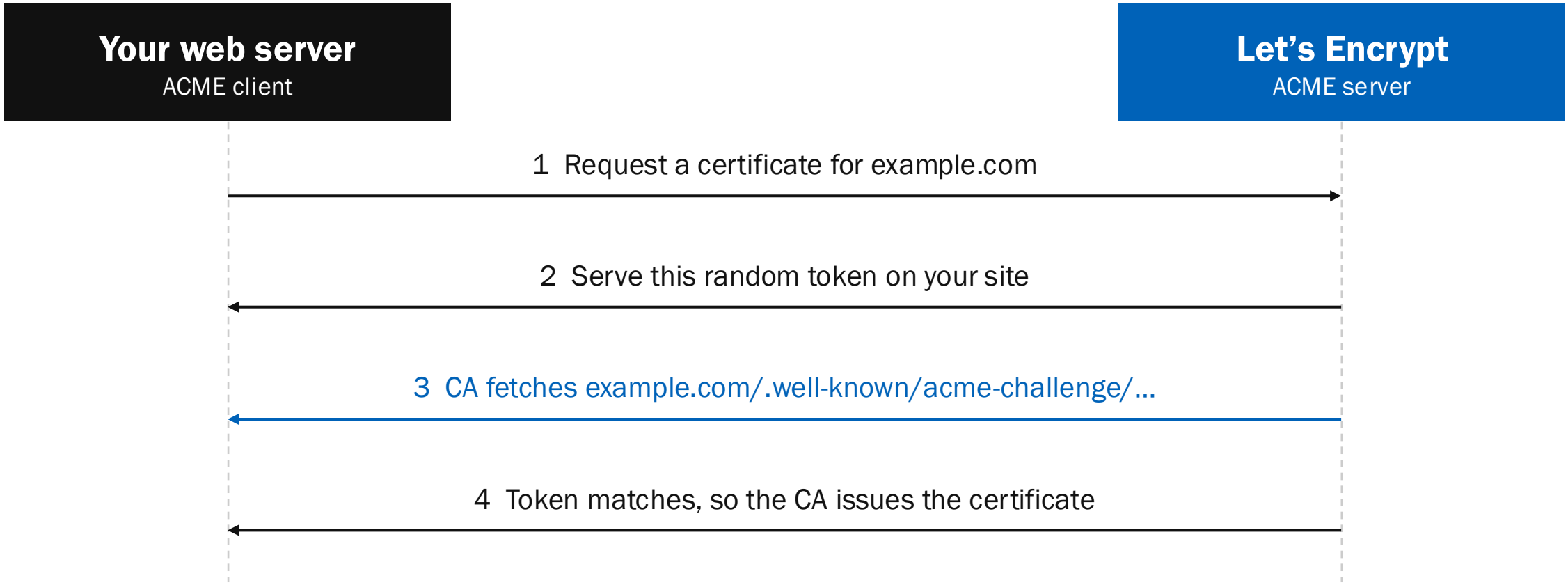
The requester generates its own key pair, so the CA never sees the private key

Adapted from P. C. van Oorschot, Computer Security and the Internet, §8.1

02

Issuing

Let’s Encrypt issues free certificates by automatically checking domain control



No cost and no human review: servers renew short-lived certificates automatically

Source: letsencrypt.org, ACME protocol (RFC 8555), HTTP-01 challenge

02

Issuing

DV, OV and EV certificates differ in how much the CA checked

Grade	What the CA checks	Issuance	What users see
Domain Validated (DV)	Control of the domain only	Free or cheap, fully automated	A lock icon, the same one a phishing site with DV gets
Organization Validated (OV)	Domain, plus business name and address	Manual checks	Organization details, if they open the certificate
Extended Validation (EV)	Legal existence, registration, location, requester's authority, domain	Manual and most expensive	Browsers once showed the company name; few users noticed

All three grades get the same lock icon in the browser, so no real advantage to an OV or EV certificate

Adapted from P. C. van Oorschot, Computer Security and the Internet, §8.5

02

Issuing

Browsers stopped showing EV in 2019, and free DV certificates now dominate

68%

of websites with an identifiable CA use Let's Encrypt[1]

Free, automated, domain-validated only

Live stats: letsencrypt.org/stats

2019: browsers dropped the EV display

Chrome 77, Firefox 70 and Safari removed the company name from the address bar; all three grades now get the same lock[2]

Why: users didn't notice

Google and UC Berkeley found the indicator didn't change user behavior, and phishing sites could get EV certificates too

Where OV and EV remain

Regulated industries, contracts or corporate policy that require them, and code signing

1. W3Techs, August 2026, via DreamHost "EV SSL Certificates: Why You Don't Need One in 2026"
2. Wikipedia, "Extended Validation Certificate"; Google/UC Berkeley study via Wikipedia, "Qualified website authentication certificate"

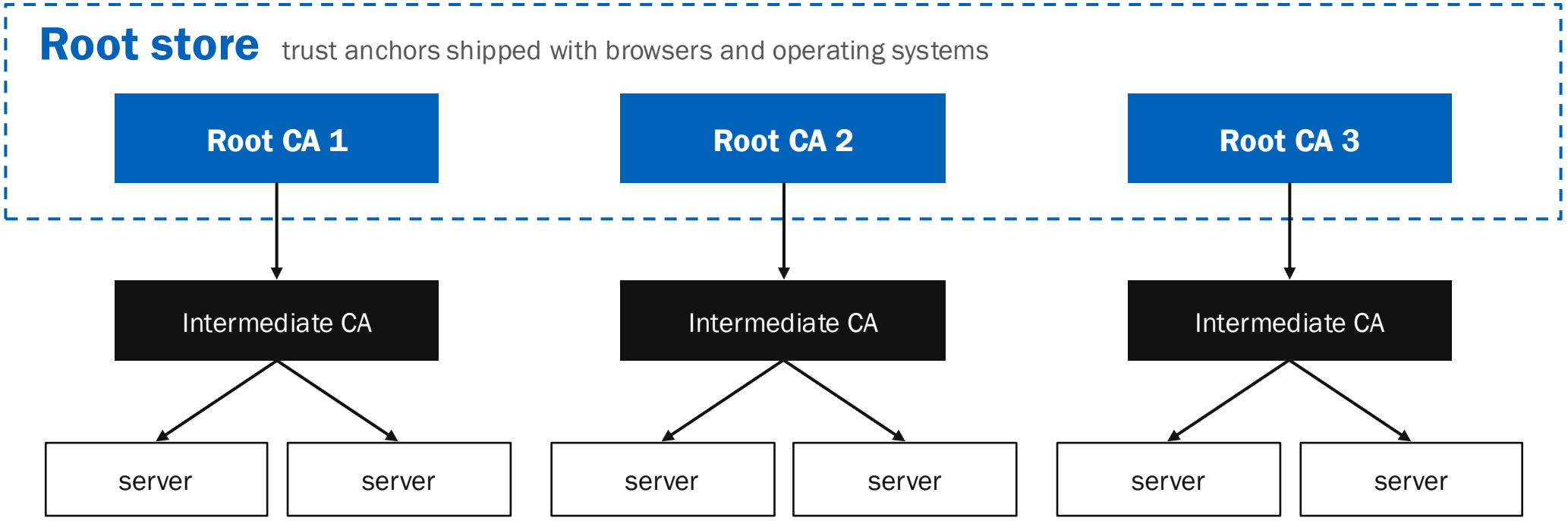
03

Verifying a certificate

Anyone can sign their own certificate, called a self-signed certificate

But it won’t necessarily be trusted!

Browsers and operating systems ship a root store of trusted CAs

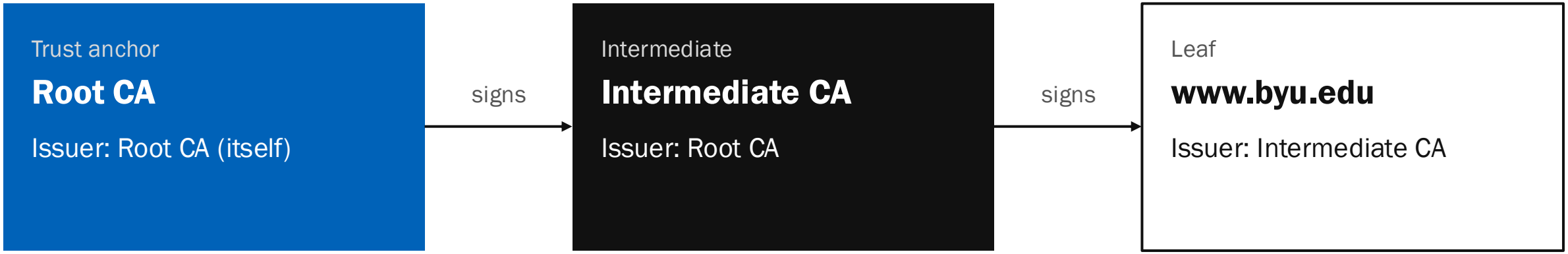


Arrows point from the signer to the certificate it signs

- **Each root CA anchors its own hierarchy;** there are no cross-certificates between them
- **A self-signed certificate that isn't in the store** triggers a browser warning

Adapted from P. C. van Oorschot, Computer Security and the Internet, Figure 8.7

Validation follows a chain of signatures from the site back to a trusted root



Check each signature with the public key of the certificate to its left, starting from a key you already trust

- **Servers send the leaf and usually the intermediates;** the root is already in your store
- **A link is missing?** Fetch the issuer's certificate and keep tracing toward a root
- **Every certificate in the chain must pass the checks,** not just the leaf

Adapted from P. C. van Oorschot, Computer Security and the Internet, Figure 8.2

03

Verifying

Every certificate in the chain must pass six checks

1. **Not expired**
   Today falls between Not-Before and Not-After
2. **Not revoked**
   The CA hasn’t withdrawn it early (section 04)
3. **Signature verifies**
   Using the signing CA’s public key
4. **Use fits the constraints**
   Basic constraints, key usage, name constraints
5. **Subject matches the use**
   The hostname you’re visiting, or the email address you’re writing to
6. **Signer is trusted**
   It’s a trust anchor, or repeat these checks one level up

Extensions limit what each certificate can be used for

Extension	What it limits	Example
Basic constraints	Whether the key belongs to a CA, and how long a chain it may sign below it	Path length 0: the CA may sign only leaf certificates
Key usage	What the key may be used for	Signatures, encryption, key agreement, CRL signing, code signing
Subject alternative name	Other names the certificate covers	byu.edu and www.byu.edu, an IP address, other email addresses
Name constraints	Which names a CA may certify	Only names under byu.edu, or only .com domains

- **The CA marks each extension** critical or non-critical when it issues the certificate
- **Non-critical:** software that doesn't understand it ignores it and processes the rest
- **Critical:** software that can't process it must reject the whole certificate
- **Why it matters:** older software can't silently skip a constraint the CA set

03

Exercise

Class exercise: inspect website certificates in Chrome

- **Open a site's certificate:** click the icon left of the address bar, then the certificate details
- **Find the fields:** subject, issuer, validity period, SAN list, key usage
- **Walk the chain:** which root anchors it, and is that root in your system's root store?
- **Compare sites:** a Let's Encrypt site against a large company's site

03

Verifying

Any CA can sign for any domain, so browser and OS vendors police CAs by distrusting them

Any trusted CA can issue a certificate for any domain: one rogue or careless CA weakens the whole web

How vendors respond, mildest to harshest

1. **Investigate**
   Incident reports expose bad issuance
2. **Demand fixes**
   Revoke bad certificates, fix the process
3. **Distrust new certificates**
   New certificates stop being trusted
4. **Remove the root**
   An update drops it from the root store

CAs that lost browser trust

DigiNotar, 2011
Hacked; fake Google cert used in Iran

WoSign and StartCom, 2016
Backdated certificates

Symantec, 2017–18
30,000+ misissued; sold to DigiCert

Entrust, 2024
Compliance failures; sold to Sectigo

Sources: P. C. van Oorschot, Computer Security and the Internet, §8.5; Zeltser, “The Past, Present, and Future of the Web’s Trust Model”, 2026; Encryption Consulting, “Entrust Certificate Distrust”, 2026

03

Verifying

Certificate transparency makes every certificate public, so anyone can spot misissuance

Before: only the CA knows what it issued

DigiNotar's fake *.google.com certificate was caught only because Chrome had Google's keys built in and a user in Iran reported the warning

Nobody else could see it: a rogue certificate aimed at one target could go unnoticed

With certificate transparency (CT)

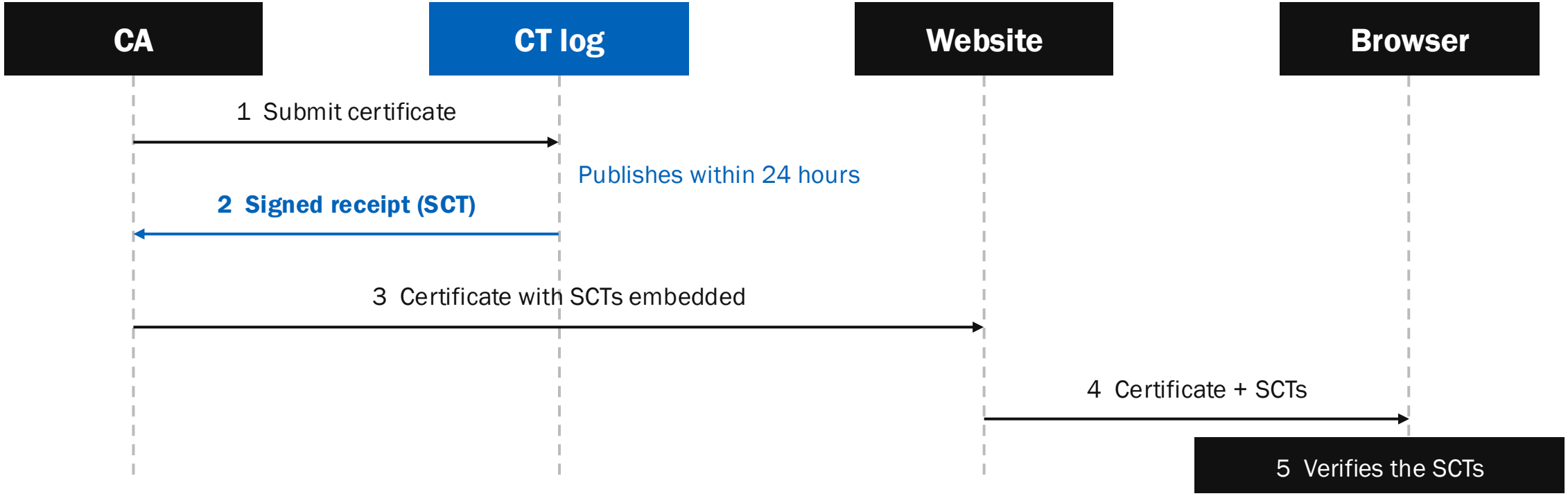
- **Every certificate goes in a public log** run by independent operators
- **Anyone can search the logs,** so a domain owner sees every certificate issued for their name
- **CT doesn't prevent misissuance;** it makes misissuance visible so vendors can act

Certificate Transparency, RFC 6962 (2013), updated by RFC 9162 (2021)

03

Verifying

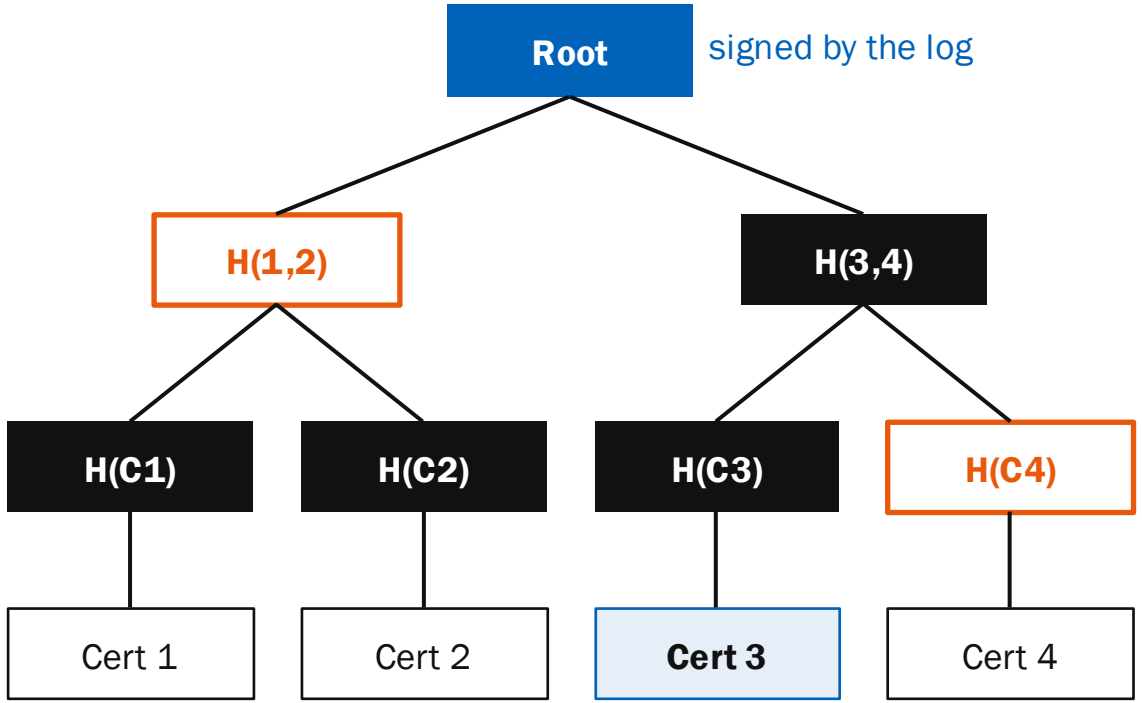
CAs submit each certificate to a public log and include the log's signed receipt



A signed certificate timestamp (SCT) is the log's signed promise to publish the certificate. If a log takes the SCT but never publishes, **the signed SCT proves it misbehaved**

Certificate Transparency, RFC 6962; Chrome Certificate Transparency policy

Logs are append-only hash trees, so no one can quietly change or delete entries



To prove Cert 3 is logged, the log sends only the orange hashes; anyone can recompute the root and compare

Append-only

New certificates become new leaves; old entries never change

Inclusion proof

A handful of hashes proves a certificate is in the log, without downloading the whole log

Consistency proof

Proves today's tree extends yesterday's, so nothing was removed

Signed root

Two different signed roots for the same log size prove the log lied

Certificate Transparency, RFC 6962 §2; hash trees build on the hashing lecture

03

Verifying

Browsers reject certificates without log receipts, and site owners watch the logs

Browsers enforce it

- **Chrome requires SCTs** on every certificate issued since April 30, 2018; Safari and Firefox followed
- **No SCTs, no lock:** the browser rejects the certificate

Site owners watch the logs

- **Search tools like crt.sh** list every logged certificate for a domain
- **Google found Symantec's** unauthorized google.com certificates this way in 2015

CT detects, it doesn't block

- **A rogue certificate still works** until someone notices and it is revoked or the CA distrusted

Chrome CT policy; SSL.com, "Questions about Certificate Transparency"; InfoQ, "Beyond the Padlock", 2025

03

Exercise

Class exercise: search the certificate transparency logs

- **Search crt.sh:** look up byu.edu and count the certificates logged for it
- **Find the issuers:** which CAs have issued certificates for byu.edu?
- **Spot a surprise:** any subdomain you didn’t expect, or a CA you don’t recognize?
- **Think like an owner:** how would BYU notice a certificate it never requested?

04

Revoking a certificate

04

Revocation

A certificate must be revoked early when its key or owner changes

Most serious

Private key stolen

An attacker can impersonate the owner until relying parties learn of it

Private key lost

The owner can no longer use the key, so it should stop being trusted

Owner details changed

The owner's name, role or organization in the certificate is out of date

Key replaced

The owner moved to a new key pair before the old one expired

Every certificate expires at Not-After; revocation ends trust before that date, and the CA that issued it is responsible for publishing it

04

Revocation

CRLs list every revoked certificate, but grow too large for phones

Certificate Revocation List signed by Example CA

`3a:91:f2:07:c4`	2026-09-02	key compromise
`7e:05:bd:11:9a`	2026-09-14	superseded
`c2:48:6f:e3:50`	2026-10-01	affiliation changed

How it works

The CA publishes a signed list of every unexpired but revoked certificate, and relying parties fetch it periodically

Drawback

The list grows with every revocation, a heavy download for phones

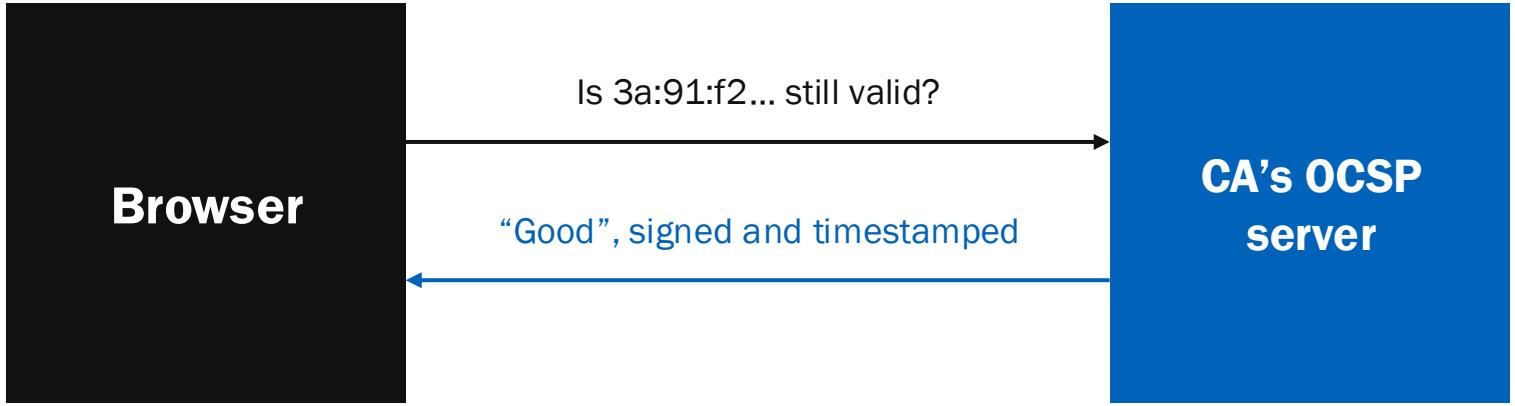
Workarounds

Split lists by serial range or send only changes (delta CRLs); shorter lifetimes let expired entries drop off

04

Revocation

OCSP checks one certificate at a time, but leaks browsing history and adds delay



Privacy

The CA learns every site you visit (your ISP already can)

Availability

The server must answer every check, or the browser must decide without it

Delay

An extra round trip before the page can load

Browsers fail open, so an attacker who blocks OCSP can use a revoked certificate



Fail closed

Treat the certificate as revoked

Safe, but every OCSP outage breaks sites for users

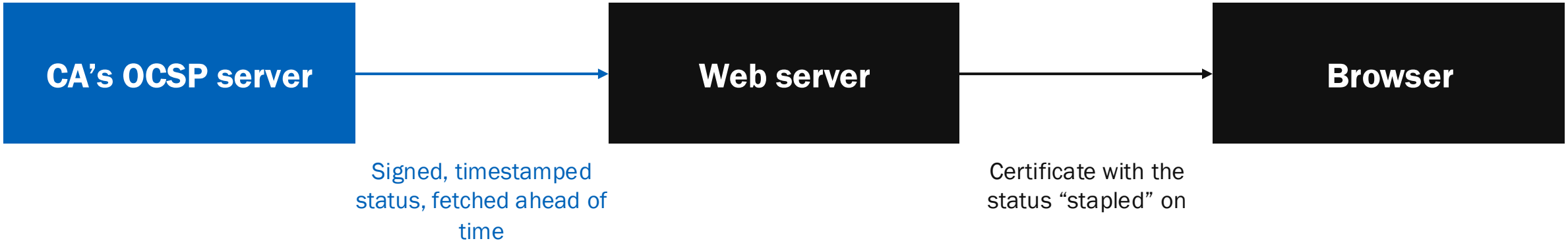
Fail open: what browsers do

Assume the certificate is fine

Users stay happy, but an attacker who blocks OCSP gets a revoked certificate accepted

Failing open breaks the safe-defaults principle: when in doubt, the browser trusts

OCSP stapling removes the delay, but an attacker can still strip the staple



Stapling fixes

No extra round trip: the browser gets the status with the certificate

But

An attacker can strip the staple, and the browser fails open again

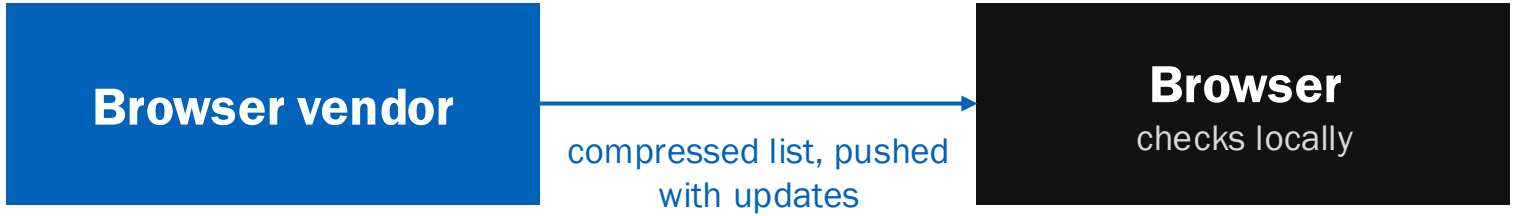
Must-staple

The certificate says a missing staple is an error; now stripping it causes denial of service, and a site that misconfigures stapling goes down

04

Revocation

Browsers now push compressed revocation lists instead



No lookup per site, so no delay and no browsing history sent to a CA

CRLSets: Google Chrome

Google gathers revocations it considers important and ships a small list with Chrome

CRLite: Mozilla Firefox

Compresses every revocation in the web PKI into a compact filter Firefox downloads

Mobile devices: uneven

Firefox and Apple push revocation data to phones; Chrome and Android get only Google’s emergency blocks

Mozilla CRLite paper, SIGCOMM 2026; Chromium security updates; eMudhra incident reports, Oct 2026

05

Public key infrastructure

PKI covers a key's whole life, from generation to revocation

Public key infrastructure (PKI): the technology and processes for managing public keys, private keys and their use



Create a key pair; the private key never leaves the owner

Prove ownership to a CA, which issues a certificate

Sign, encrypt and set up session keys

Renew the certificate or move to a new key

Revoke early, or let it expire

Supporting services

Private key backup and recovery · certificate directories · revocation status

Simplified from P. C. van Oorschot, Computer Security and the Internet, Figure 8.1

Organizations have made progress on PKI's hard problems; people now face them too

Organizations

PROGRESS **Revocation**

Firefox's CRLite checks every revocation, and 47-day certificates (2029) make stolen keys expire sooner

PROGRESS **Finding bad certificates**

Certificate transparency logs let anyone spot a certificate issued for their domain

STILL HARD **Storing keys**

CAs keep root keys offline, but web servers need keys online; hardware security modules help, yet one stolen key is costly

People

THE SHIFT **People increasingly own private keys**

Messaging apps, passkeys and crypto wallets all put a private key on your devices

STILL HARD **Storing and recovering keys**

Keys must survive a lost phone without becoming easy to steal

STILL HARD **Checking someone else's key**

With no CA hierarchy for people, how do you know a key is really theirs?

Coming up: email and PGP, SSH, Signal and other messengers, crypto wallets, and a first look at passkeys

06

Managing public and private keys for people

We'll examine managing keys for people in a variety of apps

1 Email

THE QUESTION

How do you get a stranger's key to send them encrypted email?

2 SSH

THE QUESTION

How do you trust a server, and manage keys for thousands of users?

3 Messaging

THE QUESTION

How do billions of people get each other's keys for secure messaging?

4 Crypto wallets

THE QUESTION

What happens when the private key protects a valuable asset?

Later lecture: passkeys bring the same idea to web logins

06

Email

Encrypted email needs every sender to find and trust each recipient's public key

To send Bob encrypted email, Alice needs Bob's public key and a reason to believe it is really his. **Two standards answer that differently.**

S/MIME

A CA vouches for the key

A CA certifies that the key belongs to Bob's email address, like a website certificate

- Built into Outlook and Apple Mail
- Used mostly by organizations with their own CA
- **Private key:** in your mail client; the organization often keeps a copy

PGP / OpenPGP

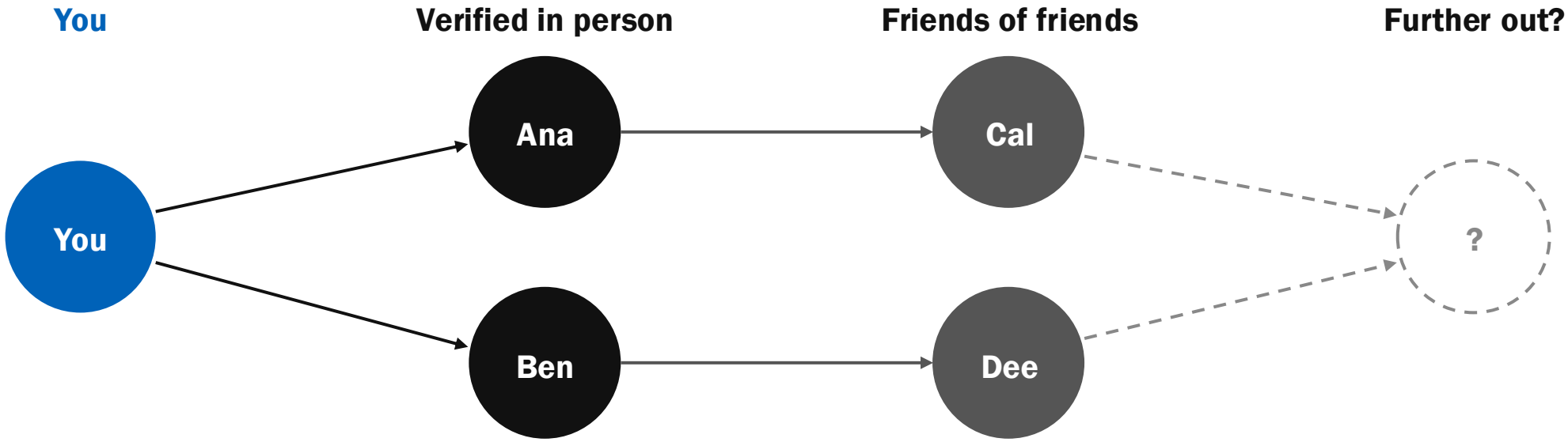
Other people vouch for the key

Bob publishes his own key; people who checked it sign it, and Alice decides whom to trust

- No CA needed, and free
- Users find, check and manage keys themselves
- **Private key:** on your computer, under a passphrase; lose it and old mail is unreadable

Both encrypt only the message body: sender, recipients and subject line travel in the clear

PGP's web of trust asked users to decide whom to trust, and mostly failed



An arrow means "signed this person's key after checking it"

- **Meet, verify and sign each other's keys,** for example at key-signing parties, then share the signatures
- **Each user decides how far trust reaches:** only people they checked, friends of friends, or further
- **Largely failed** in practice

06

Email

PGP largely failed because it asks too much of users and protects too little

Doesn’t scale beyond small groups

“designed to handle small personal keyrings”

Every new contact means finding a key and making a manual trust decision

No real revocation

“spread the word and hope everyone hears about it”

Revocation exists, but nothing reliably delivers it to others

Too hard for non-experts

“for people who prefer to pack their own parachutes”

Lose your key or passphrase and there is no recovery

Wrong trust model for organizations

“organic decentralized non-institutional approach”

Fits personal relationships, not how companies and governments decide trust

Also: both sides must opt in, metadata stays exposed, and there is no forward secrecy

Clark et al., “SoK: Securing Email—A Stakeholder-Based Analysis,” 2021, §PGP; blue quotes are Zimmermann, from the PGP user’s guide

06

SSH

SSH trusts a server's key on first use and warns if it changes

```
$ ssh cs.byu.edu
The authenticity of host 'cs.byu.edu' can't be established.
ED25519 key fingerprint is SHA256:q3Vx…Lw8.
Are you sure you want to continue connecting (yes/no)?
```

First visit	Accept the key without checking, and store it
Same key later	Connect silently: key continuity
Key changed	Warn: an attacker, or just a new key?

- **Risk:** an attacker present at first use wins, a "leap of faith"
- **Main critique:** users can't tell a legitimate key change from an attack, so they usually accept either
- **Used by** SSH; since 2016, Autocrypt, pEp and LEAP bring it to PGP email
- **Check the fingerprint once** before first use to remove the risk

Adapted from P. C. van Oorschot, Computer Security and the Internet, §8.2; Clark et al., SoK: Securing Email, 2021

At scale, companies replace users' SSH key files with short-lived certificates

Default: key files

- **Each user's public key** is copied into a list on every server
- **The private key sits on a laptop** for years, often with no passphrase
- **Removing access** when someone leaves means editing every server

Meta Engineering, "Scalable and secure access with SSH," 2016; Netflix, BLESS (Bastion's Lambda Ephemeral SSH Service)

At scale: an SSH certificate authority

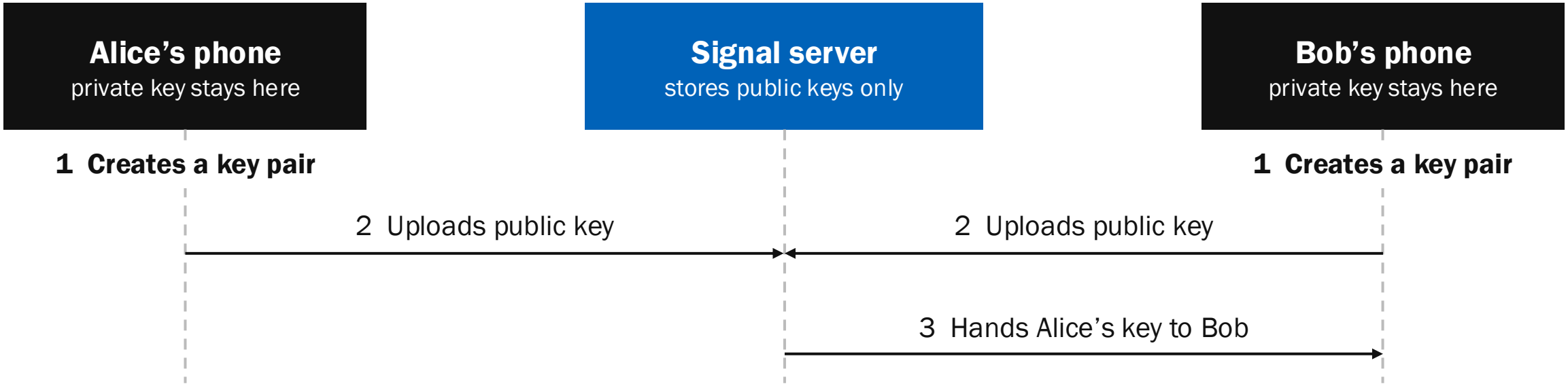
1. Engineer signs in with company single sign-on
2. SSH CA signs their key for minutes to a day
3. Servers trust the CA, not a list of keys

Expiry replaces revocation: a stolen certificate soon stops working. Servers get host certificates too, which removes trust on first use

Real products: Teleport, HashiCorp Vault and Smallstep tie certificates to company accounts and roles, so admins control which users can log in to which servers. Facebook and Netflix (BLESS) built their own

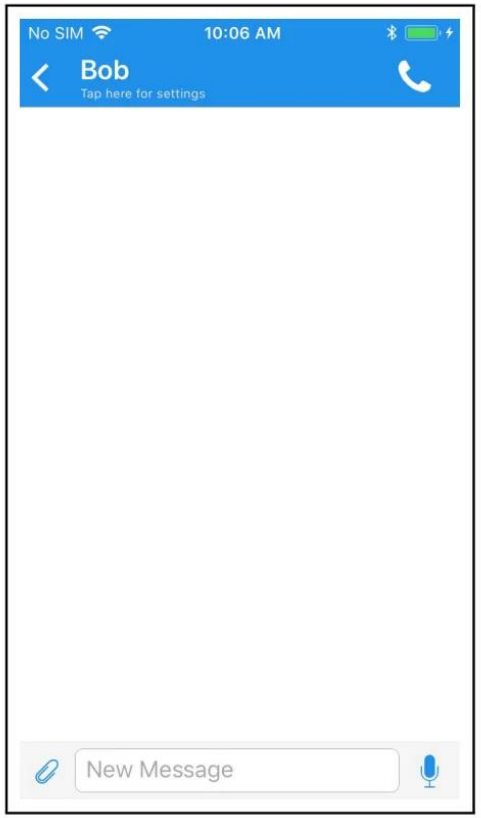
Signal keeps each private key on the phone and uses its server only to hand out public keys

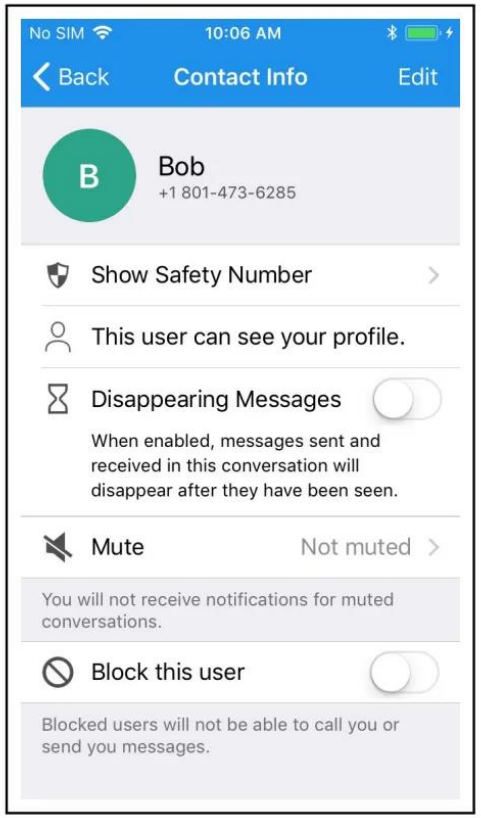
Signal: a free, nonprofit, end-to-end encrypted messenger whose protocol now protects billions of chats



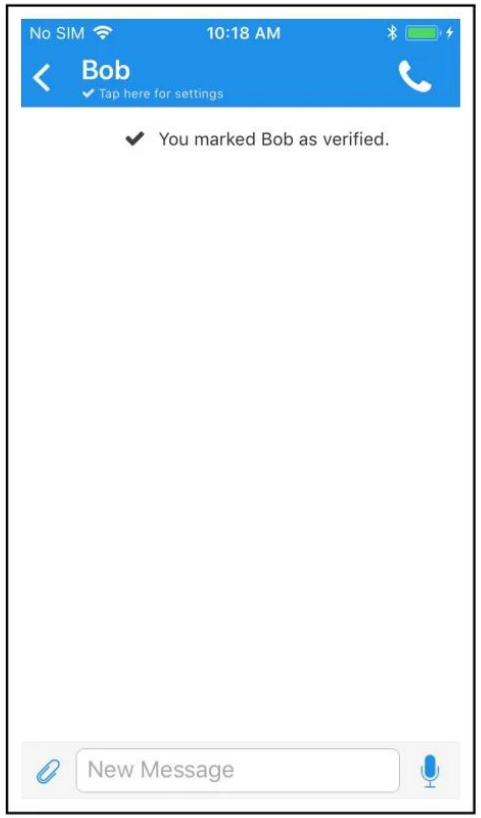
- **The server never sees a private key,** so it cannot read messages
- **But you are trusting the server** to hand out the right public key

Signal hands out keys automatically, and safety numbers let users check them









Compare safety numbers in person, or scan each other's QR code

A match means no one swapped the keys in between

Signal warns when a number changes; few users compare

06

Messaging

Few users complete the authentication ceremony, even when they know they should

Authentication ceremony: steps two users take to confirm each other's real keys, such as comparing safety numbers. **Skip it, and you are trusting the server to hand out the right keys**

WhatsApp, Viber, Messenger, 2017[1]

14% → **79%**

Told about threats only → Also told about the ceremony

Signal, original vs. redesigned, 2018[2]

30% → **90%**

Original ceremony → Redesigned to be easy to find

Share of participant pairs who found and completed the ceremony

Why it's rare: the ceremony is hidden several taps deep in every major app, and users don't see a reason to do it until something goes wrong

1. Vaziripour et al., "Is that you, Alice? A Usability Study of the Authentication Ceremony of Secure Messaging Applications," SOUPS 2017
2. Vaziripour et al., "Action Needed! Helping Users Find and Complete the Authentication Ceremony in Signal," SOUPS 2018

06

Messaging

WhatsApp, Messenger and Google Messages use the Signal Protocol and follow its model

Same model as Signal: private keys never leave each phone; the server hands out public keys, and the app warns when a contact's key changes, usually just a new phone

WhatsApp

Signal Protocol since 2016

HOW USERS CHECK KEYS

Compare a 60-digit security code or scan a QR code

Facebook Messenger

Signal Protocol

HOW USERS CHECK KEYS

Key comparison is buried in chat settings; 63% completed it when told how[1]

Google Messages (RCS)

Signal Protocol

HOW USERS CHECK KEYS

Compare a verification code, or use Key Verifier[2]

Next step, automation: WhatsApp began rolling out key transparency in 2023, a public log of user keys that apps can check automatically, much like certificate transparency for websites

1. Vaziripour et al., SOUPS 2017. 2. Google Messages Help, "Use end-to-end encryption in Google Messages." WhatsApp: Meta Engineering, "Deploying key transparency at WhatsApp," 2023

06

Crypto wallets

Crypto wallets show what happens when a stolen key can’t be revoked

$3.4B

in crypto stolen in 2025[1]

$1.5B from one exchange, Bybit

No CA and no revocation:

whoever holds the private key owns the funds, and a stolen or lost key can’t be undone

Services: hot vs. cold wallets

Exchanges keep most funds in offline “cold” storage and a little online for withdrawals; one stolen key can still dominate a year’s losses

Individuals: self-custody

You back up a seed phrase, 12–24 words that regenerate all your keys; lose it and the funds are gone, leak it and they are stolen

Individuals: let an exchange hold keys

Easier, but you are trusting a third party again, much like trusting a CA

1. Chainalysis, “2025 Crypto Theft Reaches $3.4 Billion,” 2026 Crypto Crime Report (January to early December 2025)

06

Summary

From the web to crypto wallets, each approach puts trust in a different place

Model	Who vouches for a key	Where the private key lives	User effort	Main weakness
CA model (web)	Root CAs in the browser's store	Web server, often in hardware; CA roots offline	None	Any CA can sign any domain; weak revocation
Web of trust (email, PGP)	People you or friends verified	Your computer, under a passphrase	Meet, sign keys, decide whom to trust	Too much work; largely failed
Trust on first use (SSH)	Your first connection	Key files on servers and laptops	Accept once	Attacker at first use wins; users accept changes
Safety numbers (messaging)	The app; you, if you compare	Your phone; a new phone means a new key	Compare numbers in person	Few users compare
Crypto wallet	No one: the key is the funds	Wallet app, hardware wallet or an exchange	Back up a seed phrase	Lost or stolen keys can't be undone