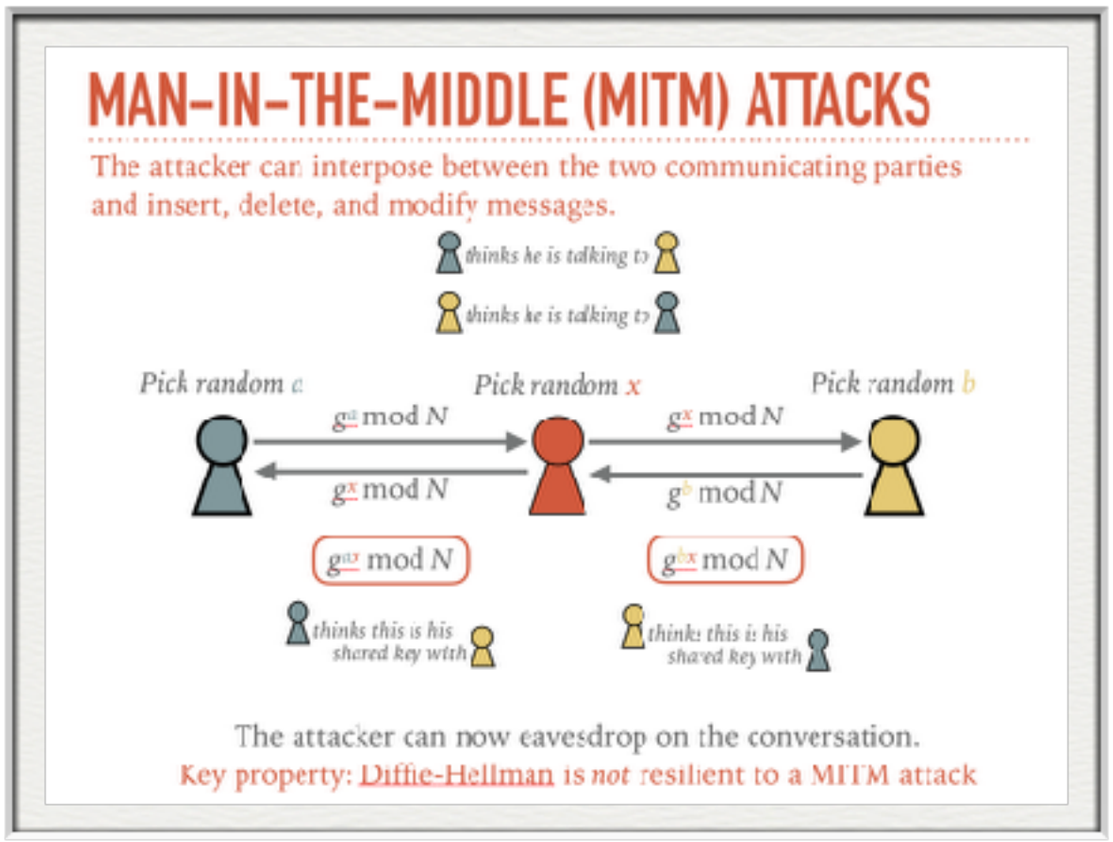
PROVING WHO YOU ARE

TLS & THE PKI

CMSC 414

MAR 29 2018

RECALL OUR PROBLEM WITH DIFFIE-HELLMAN



The two communicating parties thought, *but did not confirm*, that they were talking to one another.

Therefore, they were vulnerable to MITM attacks.

Certificates allow us to verify with whom we are communicating.

We will solve this by incorporating public key cryptography

Back to authentication

How can we know it was really  who posted PK?

Back to authentication

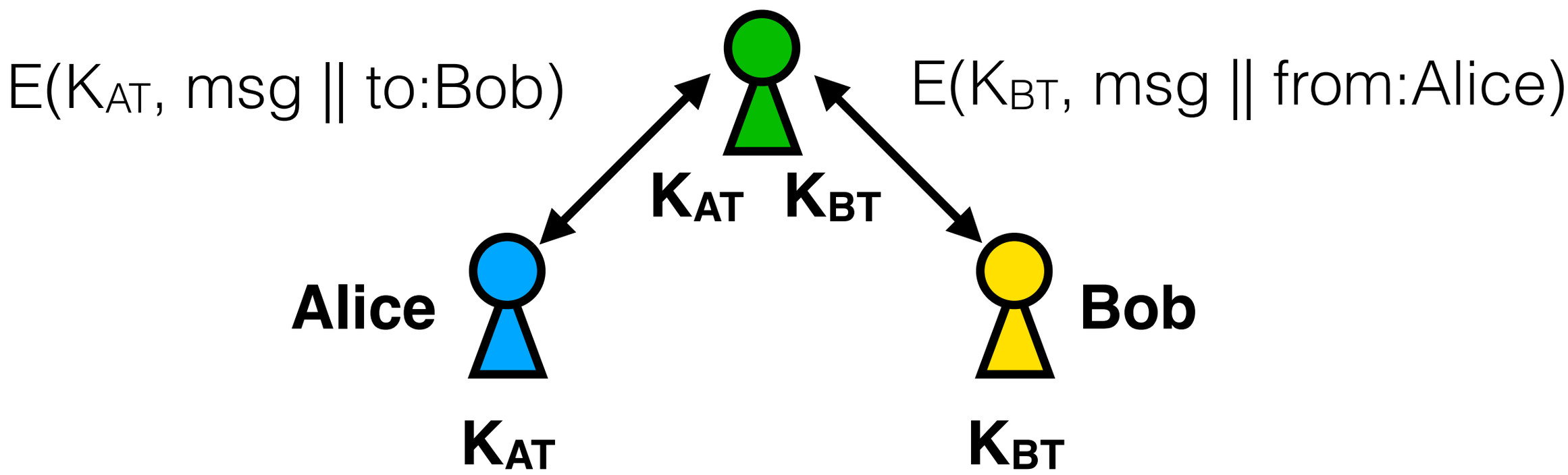
Generate public/private key pair (PK,SK); publicize PK

How can we know it was really who posted PK?

Back to authentication

Generate public/private key pair (PK,SK); publicize PK

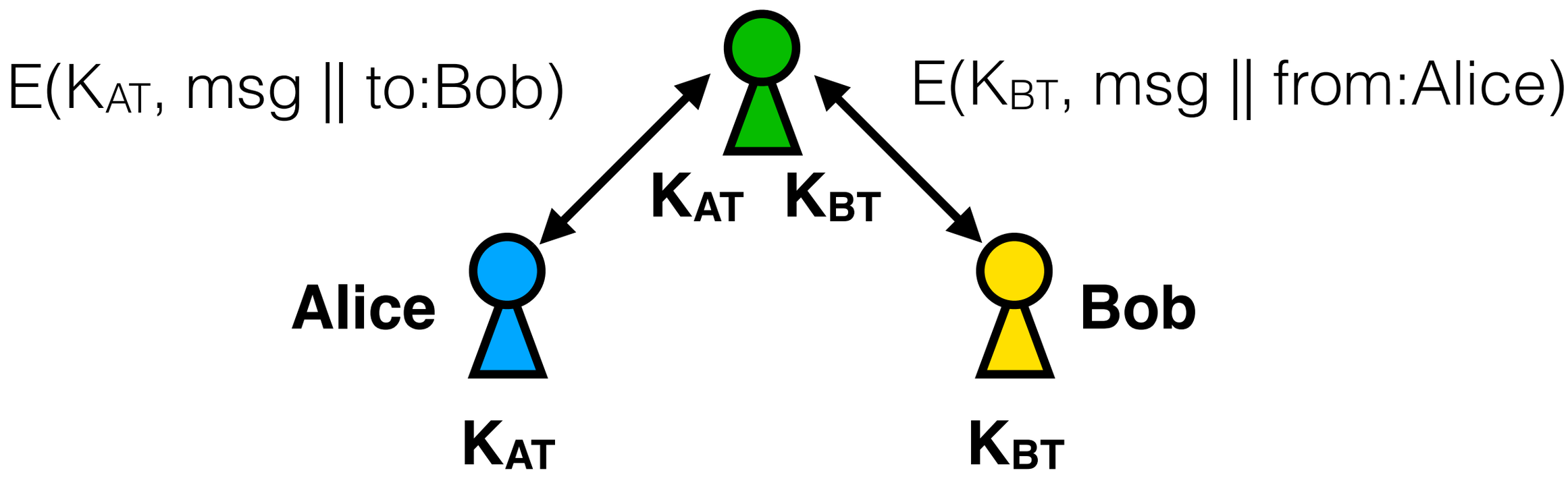
How can we know it was really who posted PK?



Back to authentication

Generate public/private key pair (PK,SK); publicize PK

How can we know it was really who posted PK?



Can we achieve authentication without Trent in the middle of *every message*?

Authentication with public keys

Trent  **(PK_T, SK_T)**

1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

Alice 

Bob 

Authentication with public keys

Trent  **(PK_T, SK_T)**

1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

Alice  **PK_T**

Bob  **PK_T**

Authentication with public keys

Trent  **(PK_T, SK_T)**

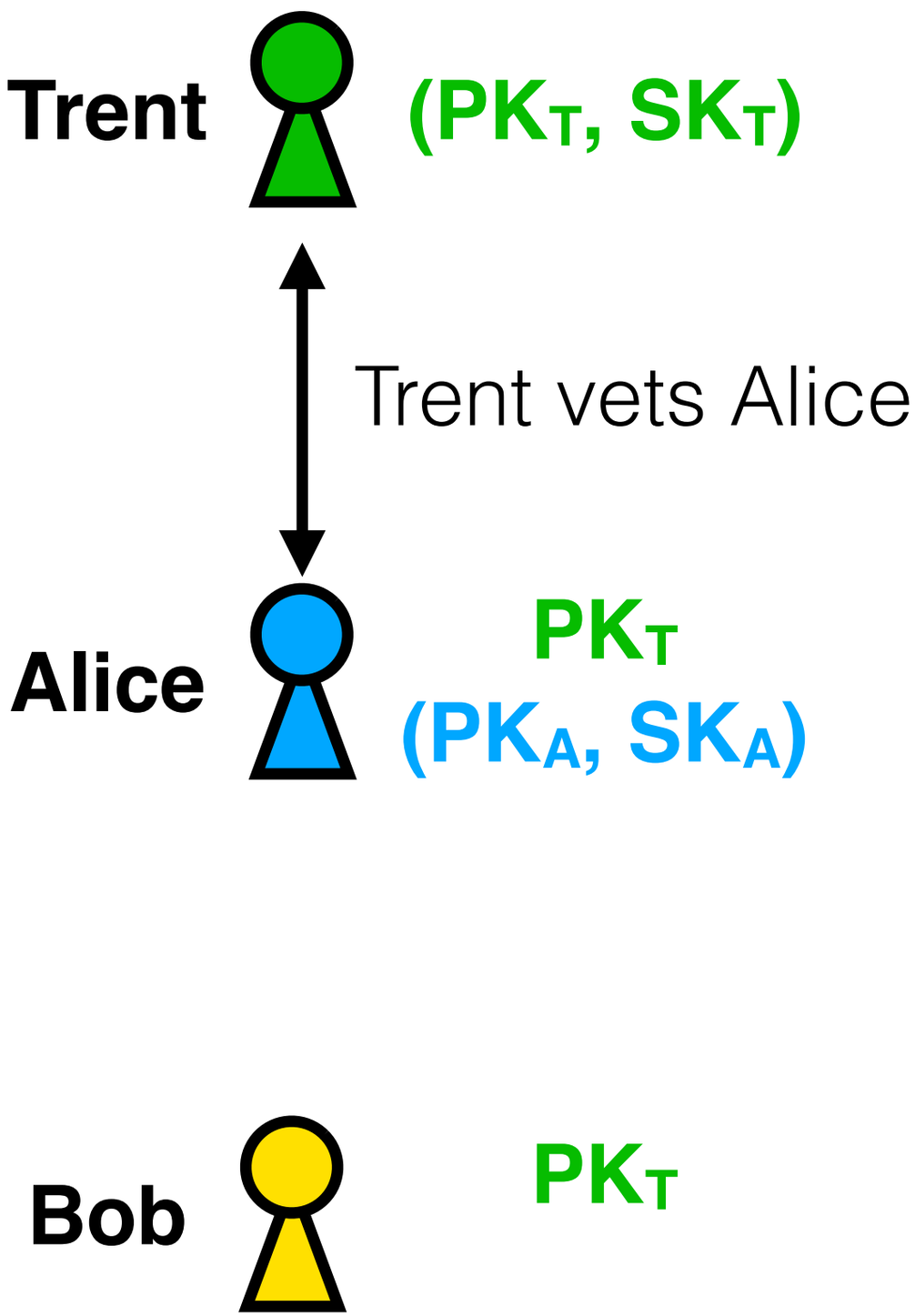
1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

Alice  **PK_T**
(PK_A, SK_A)

2. Alice generates a public/private key pair and asks Trent to bind her **PK_A** to her identity

Bob  **PK_T**

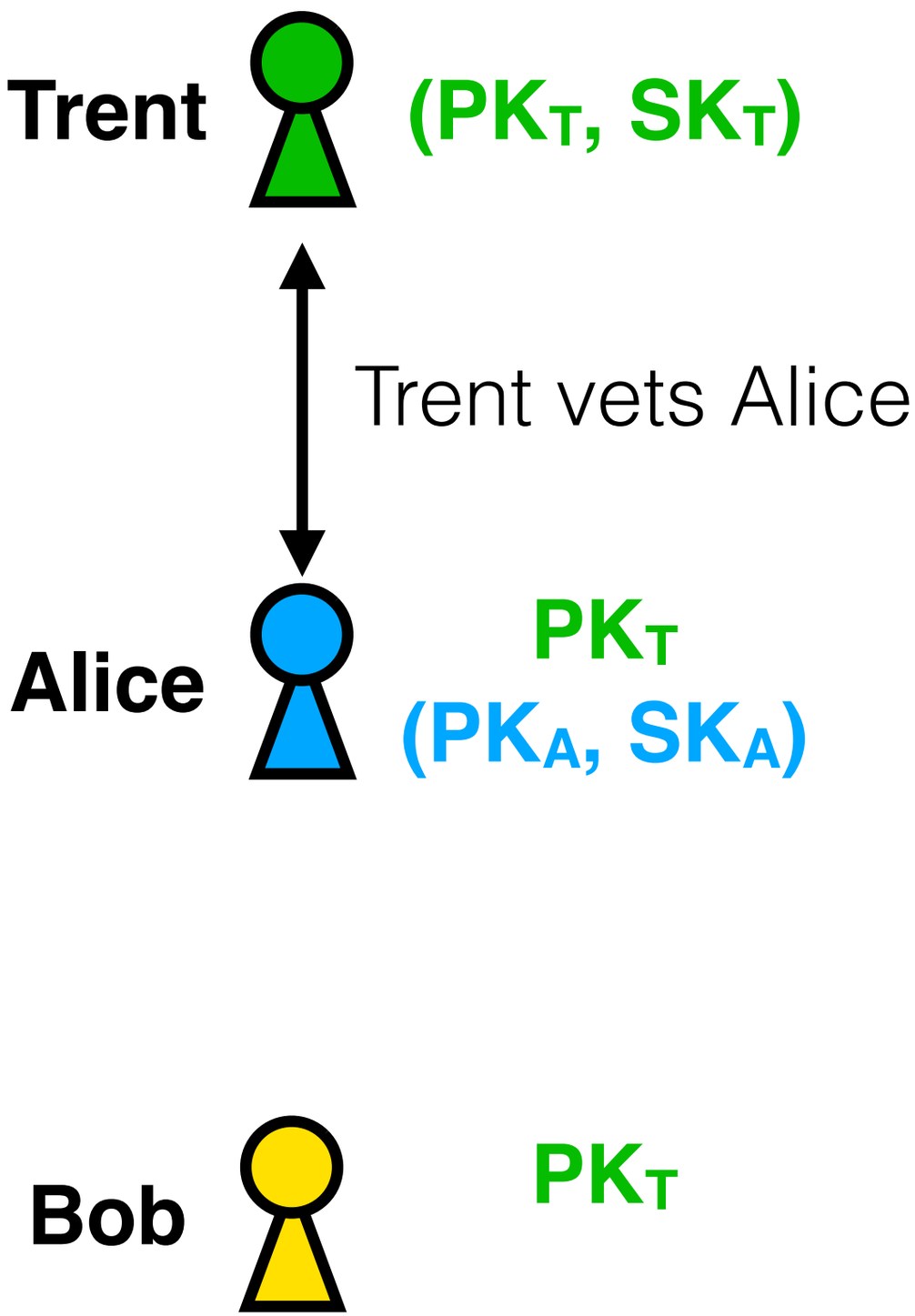
Authentication with public keys



1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

2. Alice generates a public/private key pair and asks Trent to bind her PK_A to her identity

Authentication with public keys



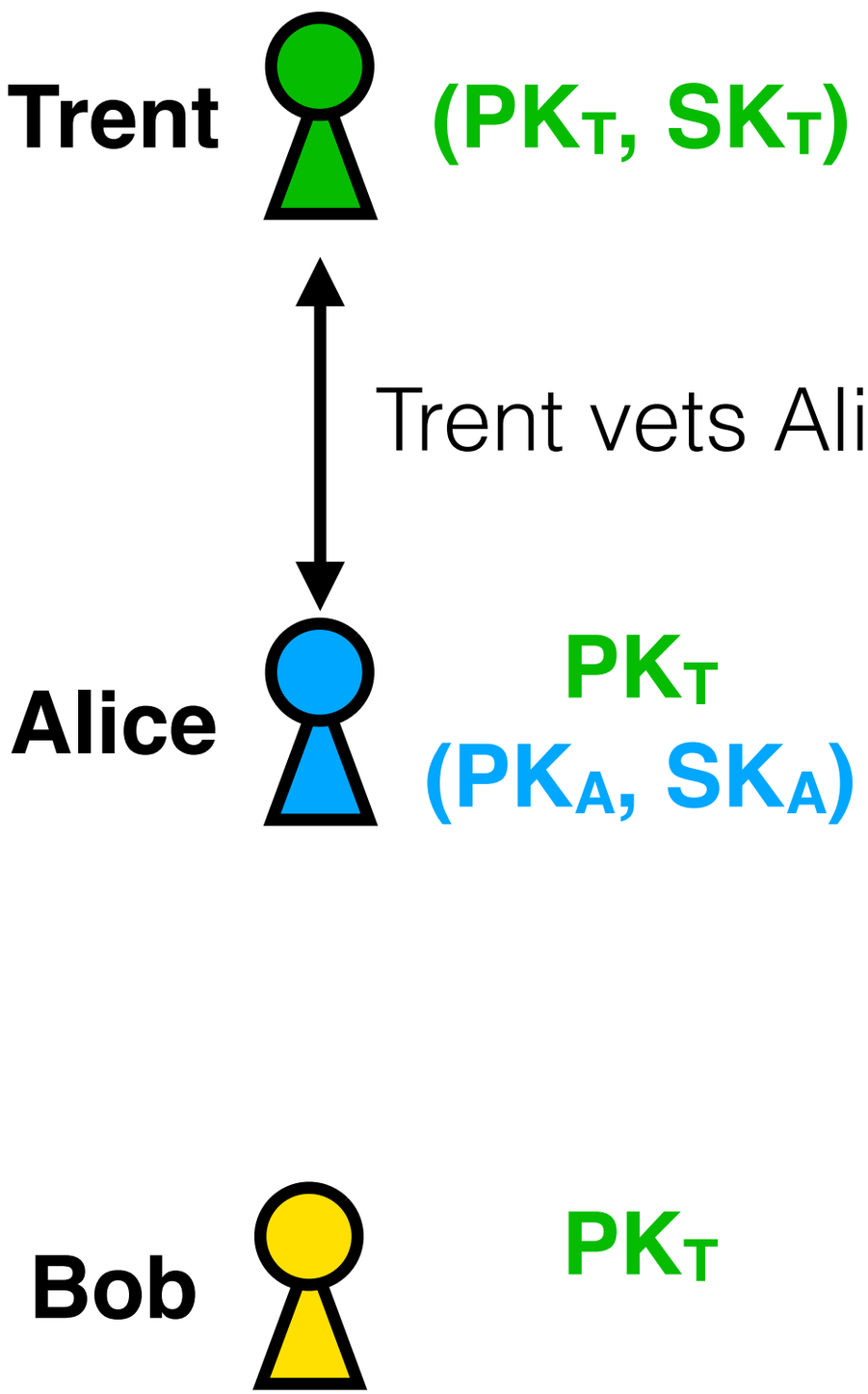
1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

2. Alice generates a public/private key pair and asks Trent to bind her $\mathbf{PK_A}$ to her identity

3. Trent *signs* a message (with $\mathbf{SK_T}$):

"The owner of the secret key corresponding to $\mathbf{PK_A}$ is Alice"

Authentication with public keys



1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

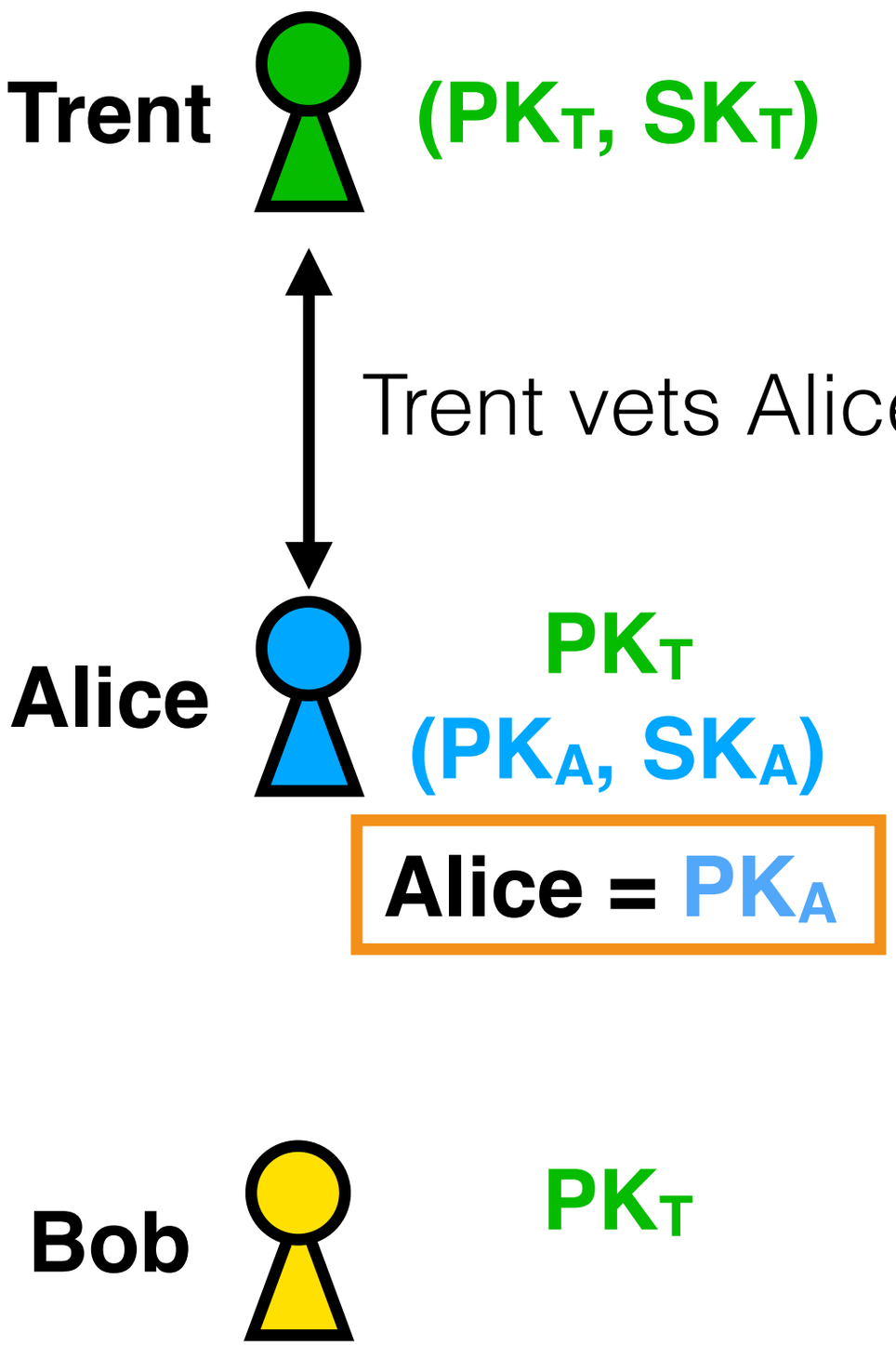
2. Alice generates a public/private key pair and asks Trent to bind her $\mathbf{PK_A}$ to her identity

3. Trent *signs* a message (with $\mathbf{SK_T}$):

> "The owner of the secret key corresponding to $\mathbf{PK_A}$ is Alice"

This message + sig = **Certificate**

Authentication with public keys



1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

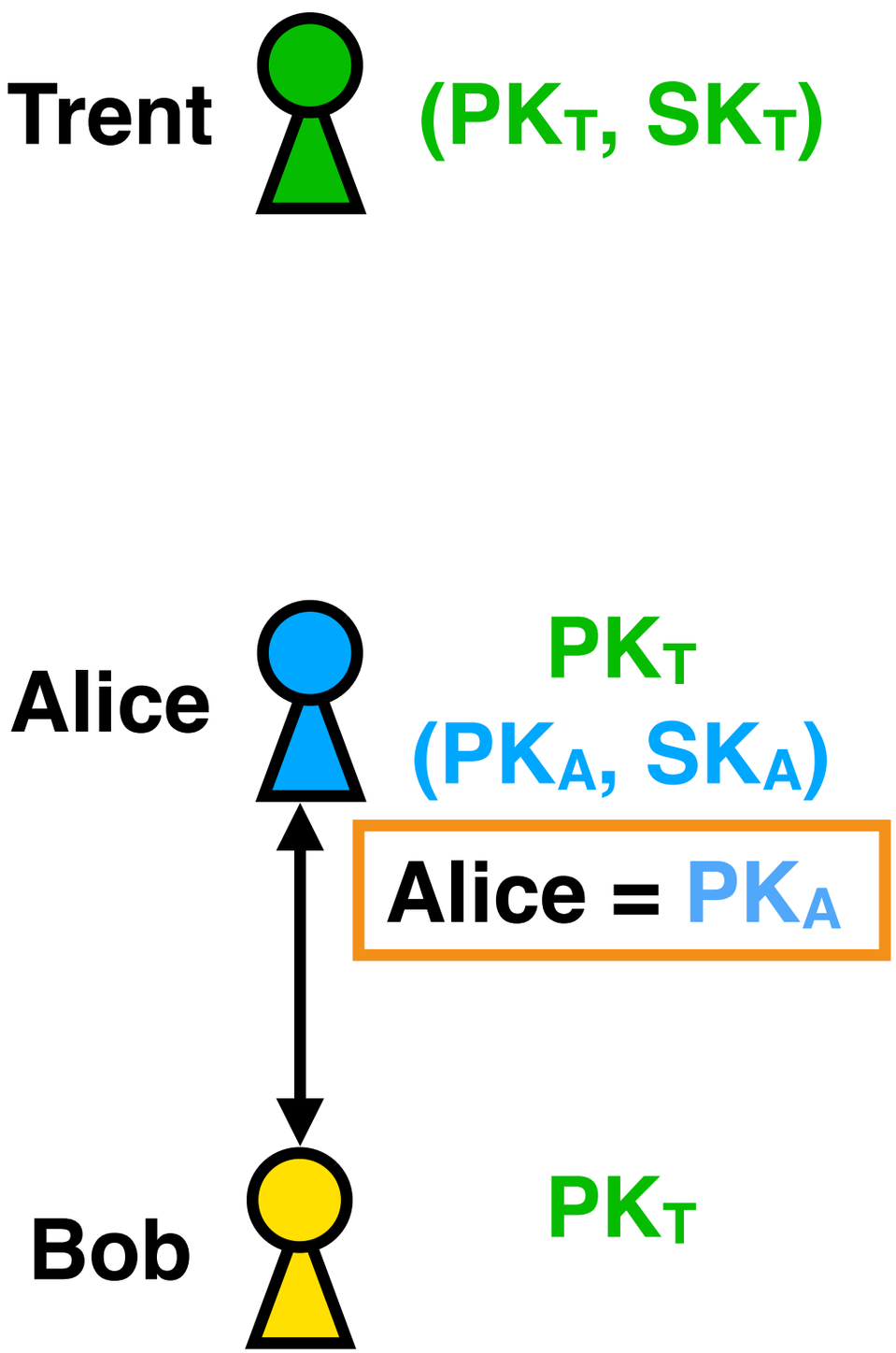
2. Alice generates a public/private key pair and asks Trent to bind her PK_A to her identity

3. Trent *signs* a message (with SK_T):

"The owner of the secret key corresponding to PK_A is Alice"

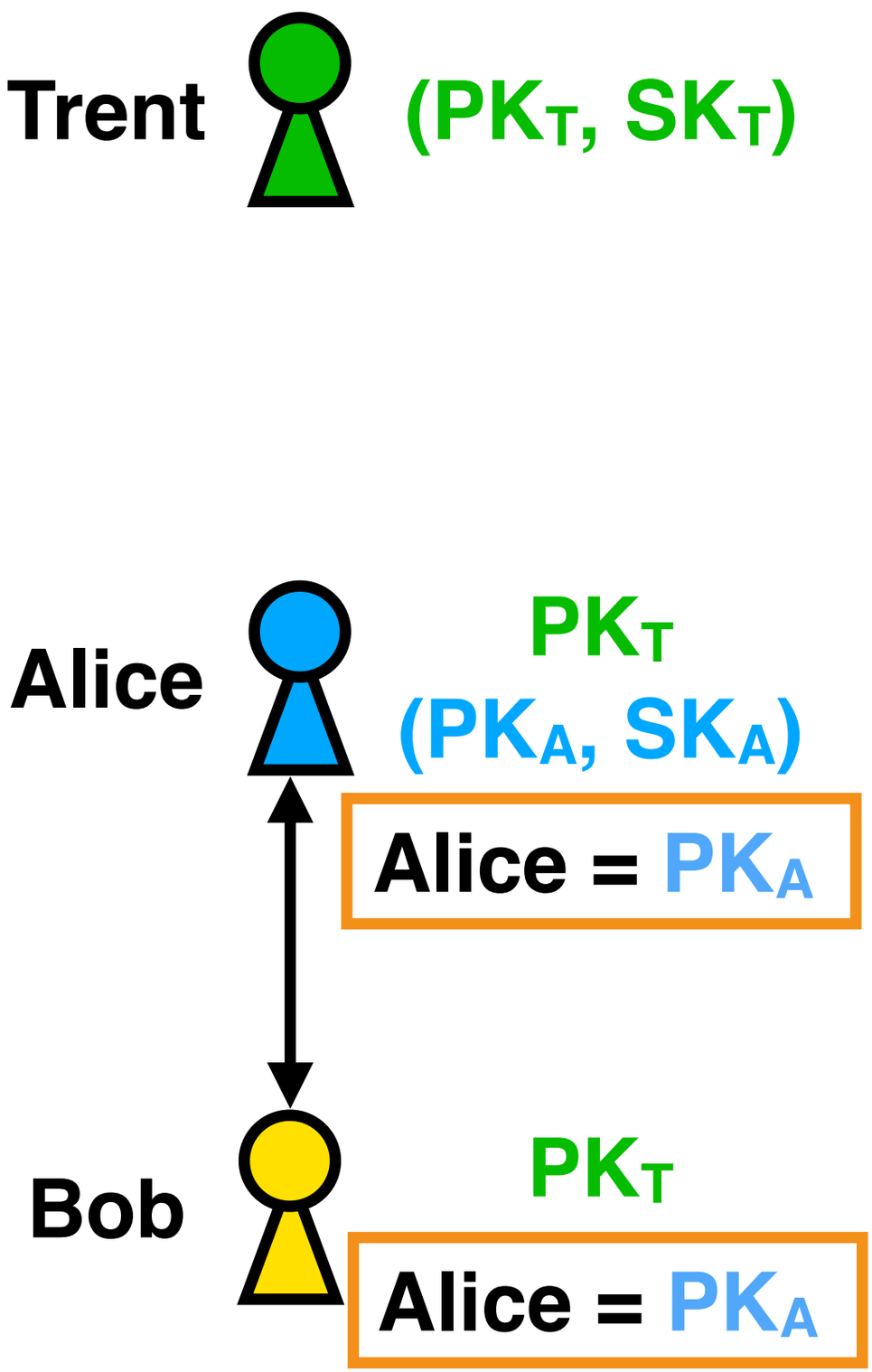
This message + sig = **Certificate**

Authentication with public keys



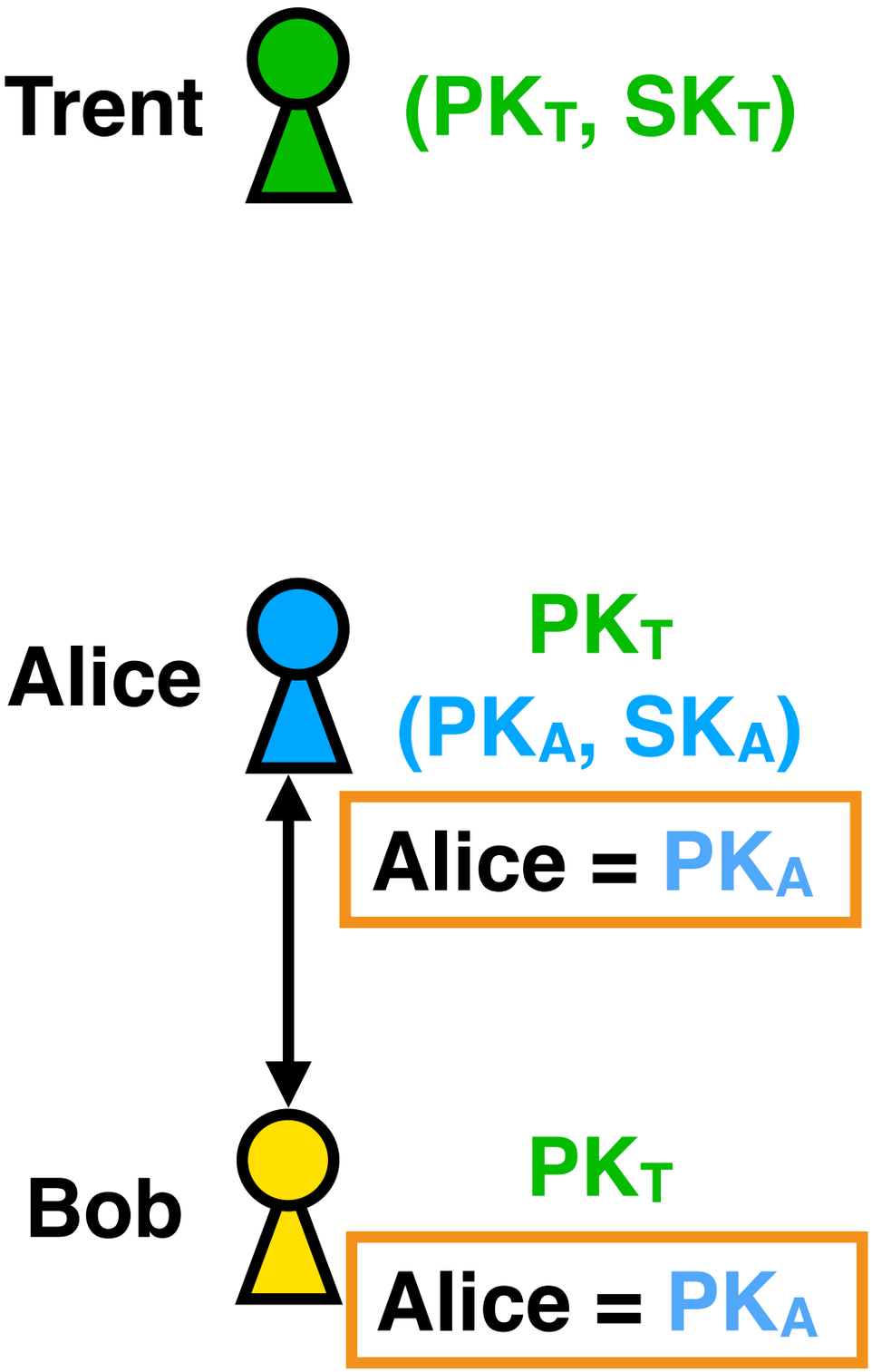
4. Alice makes her **certificate** publicly available (or Bob simply asks for it)

Authentication with public keys



4. Alice makes her **certificate** publicly available (or Bob simply asks for it)

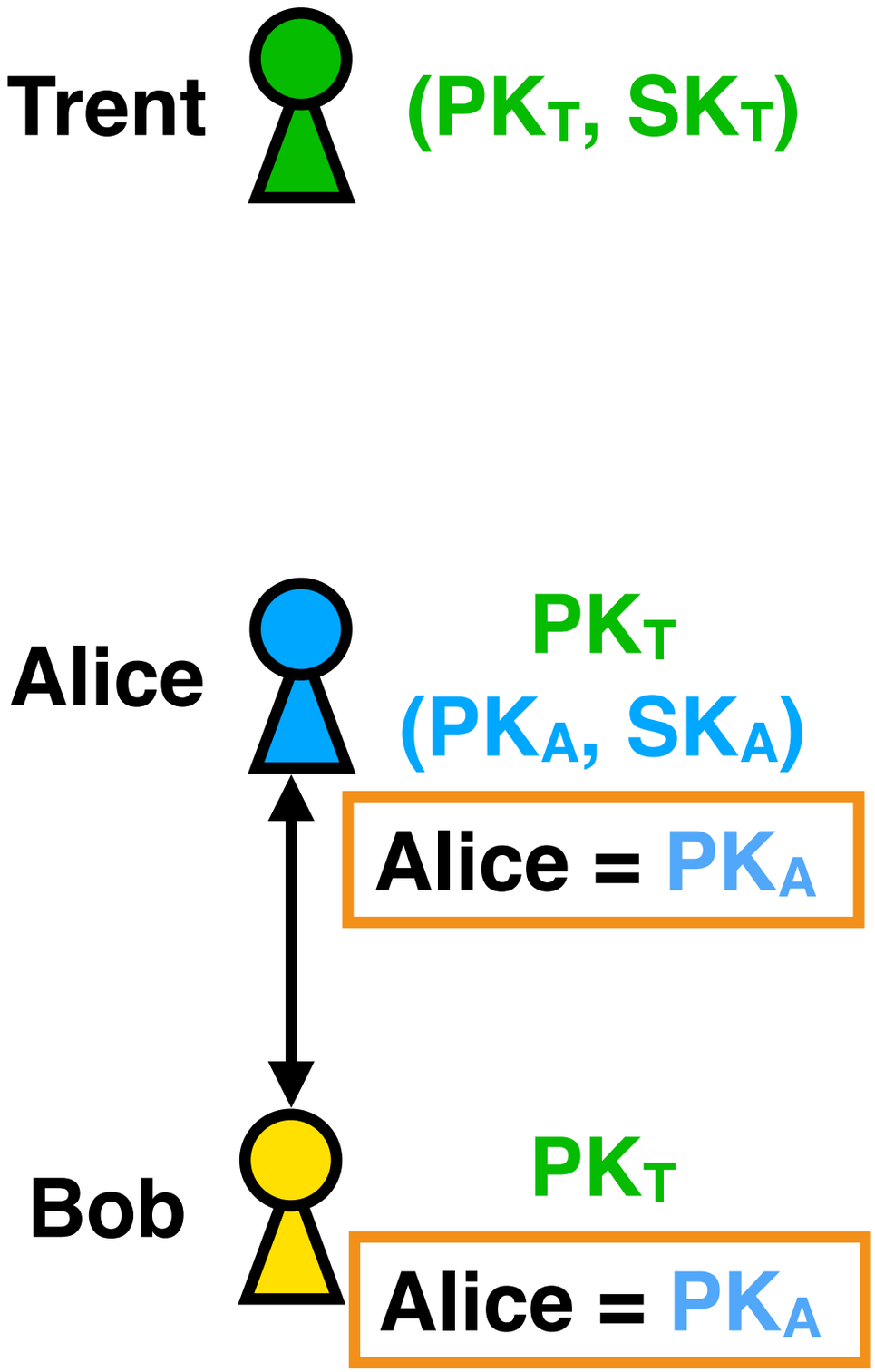
Authentication with public keys



4. Alice makes her **certificate** publicly available (or Bob simply asks for it)

5. Bob verifies the **certificate** using $\mathbf{PK_T}$

Authentication with public keys

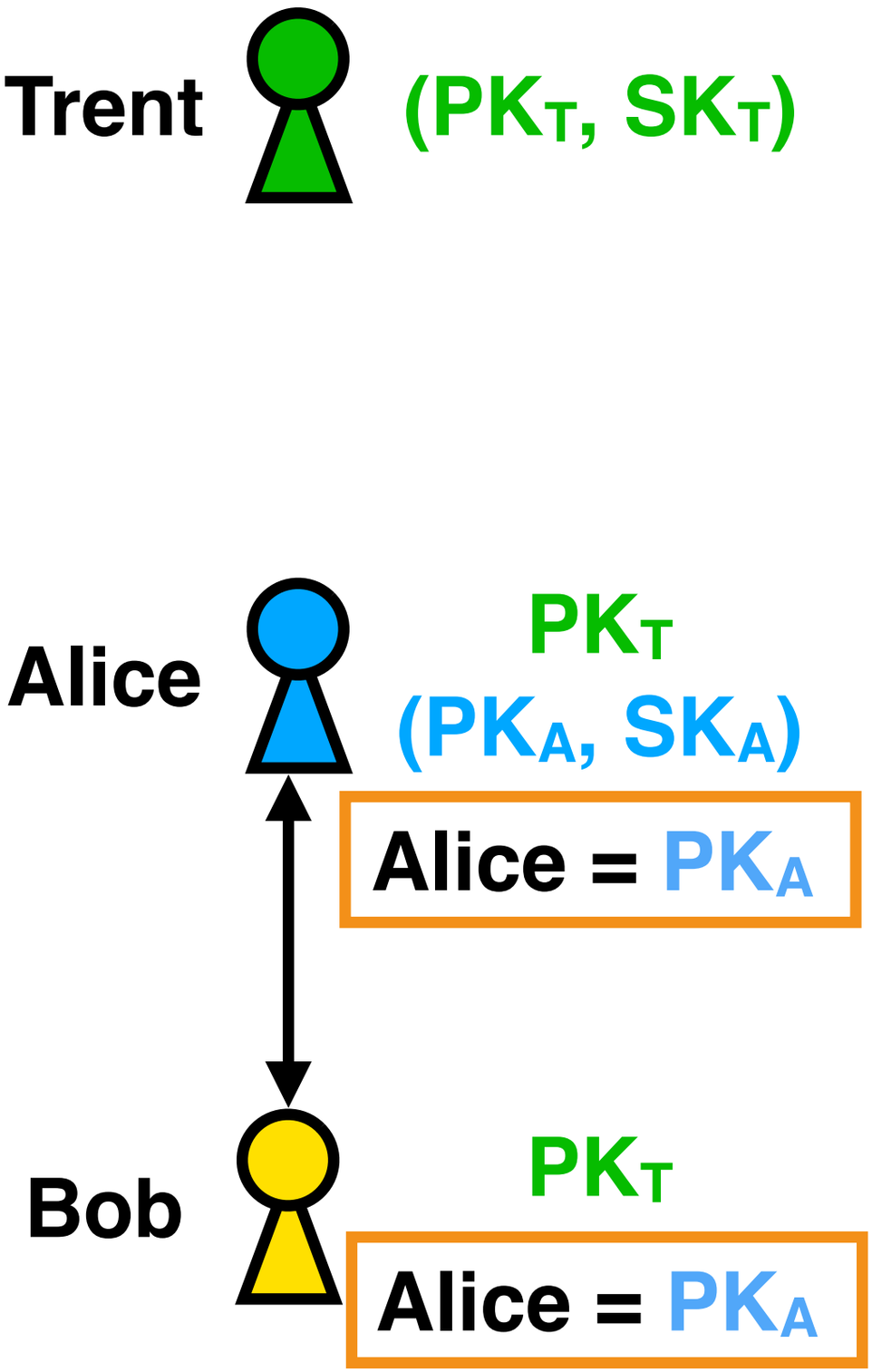


4. Alice makes her **certificate** publicly available (or Bob simply asks for it)

5. Bob verifies the **certificate** using PK_T

If Bob trusts Trent, then Bob trusts that he properly vetted Alice, and thus that her public key is PK_A

Authentication with public keys



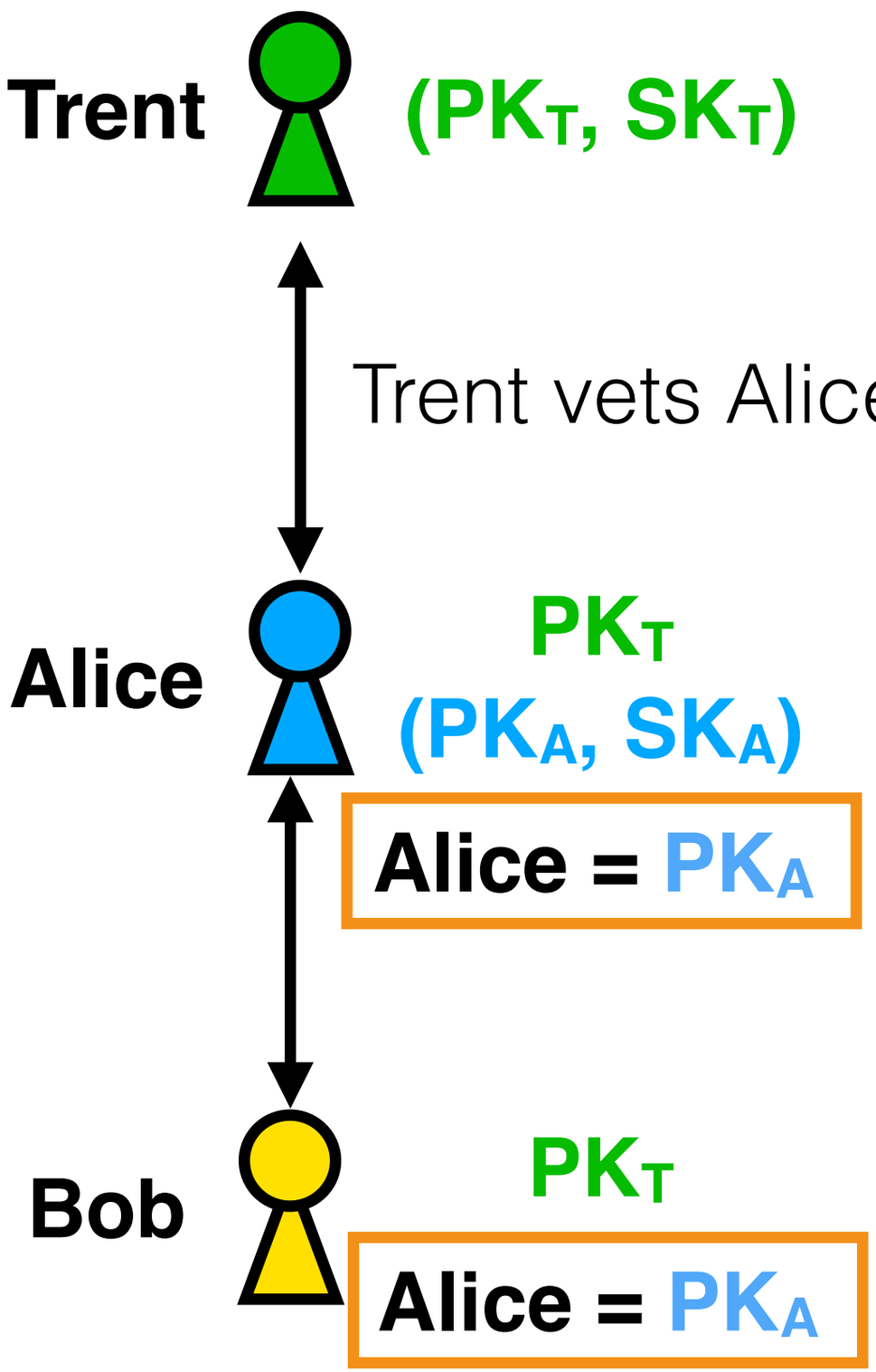
4. Alice makes her **certificate** publicly available (or Bob simply asks for it)

5. Bob verifies the **certificate** using $\mathbf{PK_T}$

If Bob trusts Trent, then Bob trusts that he properly vetted Alice, and thus that her public key is $\mathbf{PK_A}$

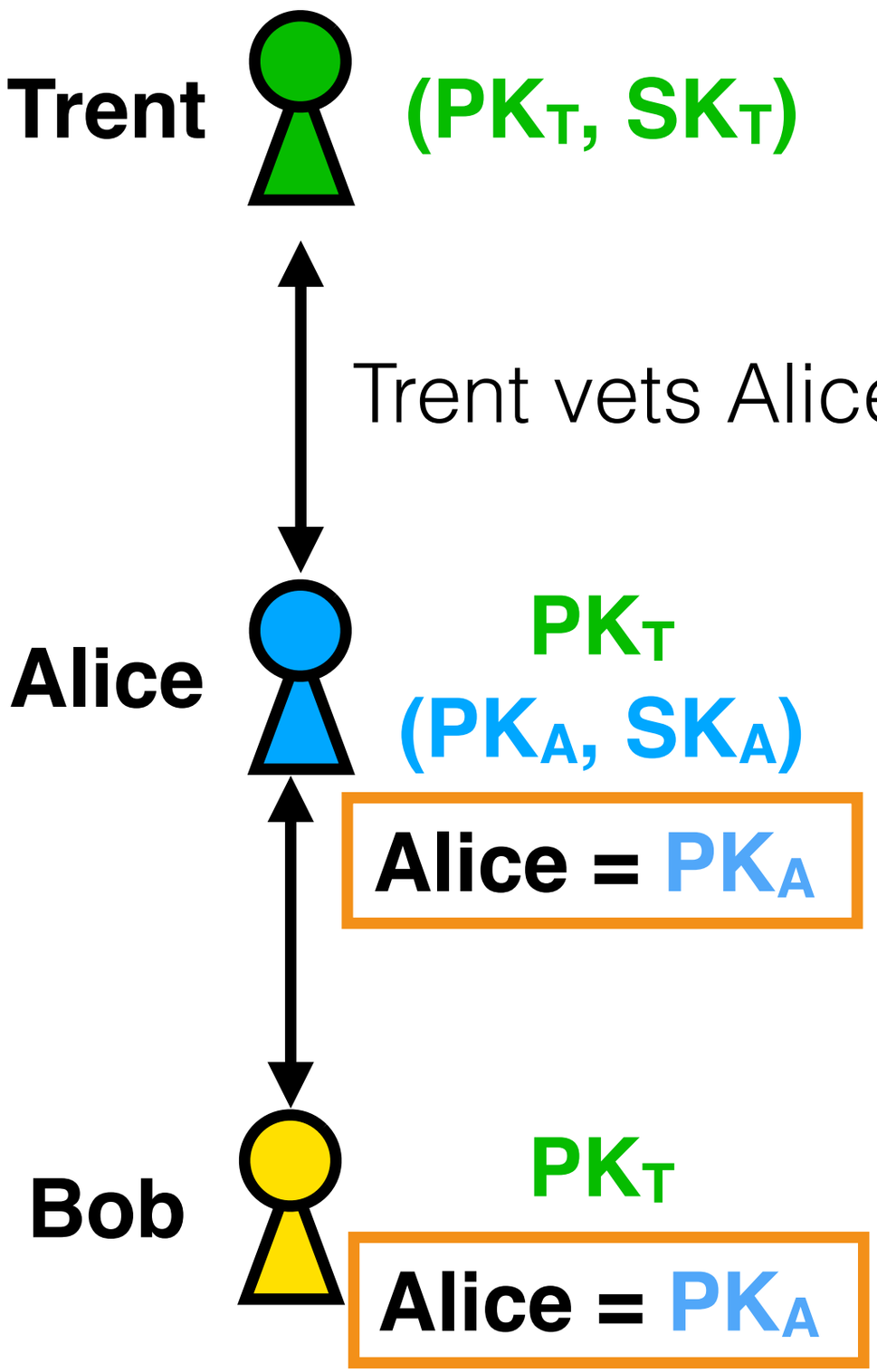
6. Bob (via hybrid encryption) sends a message to Alice using her public key $\mathbf{PK_A}$

Authentication with public keys



Properties

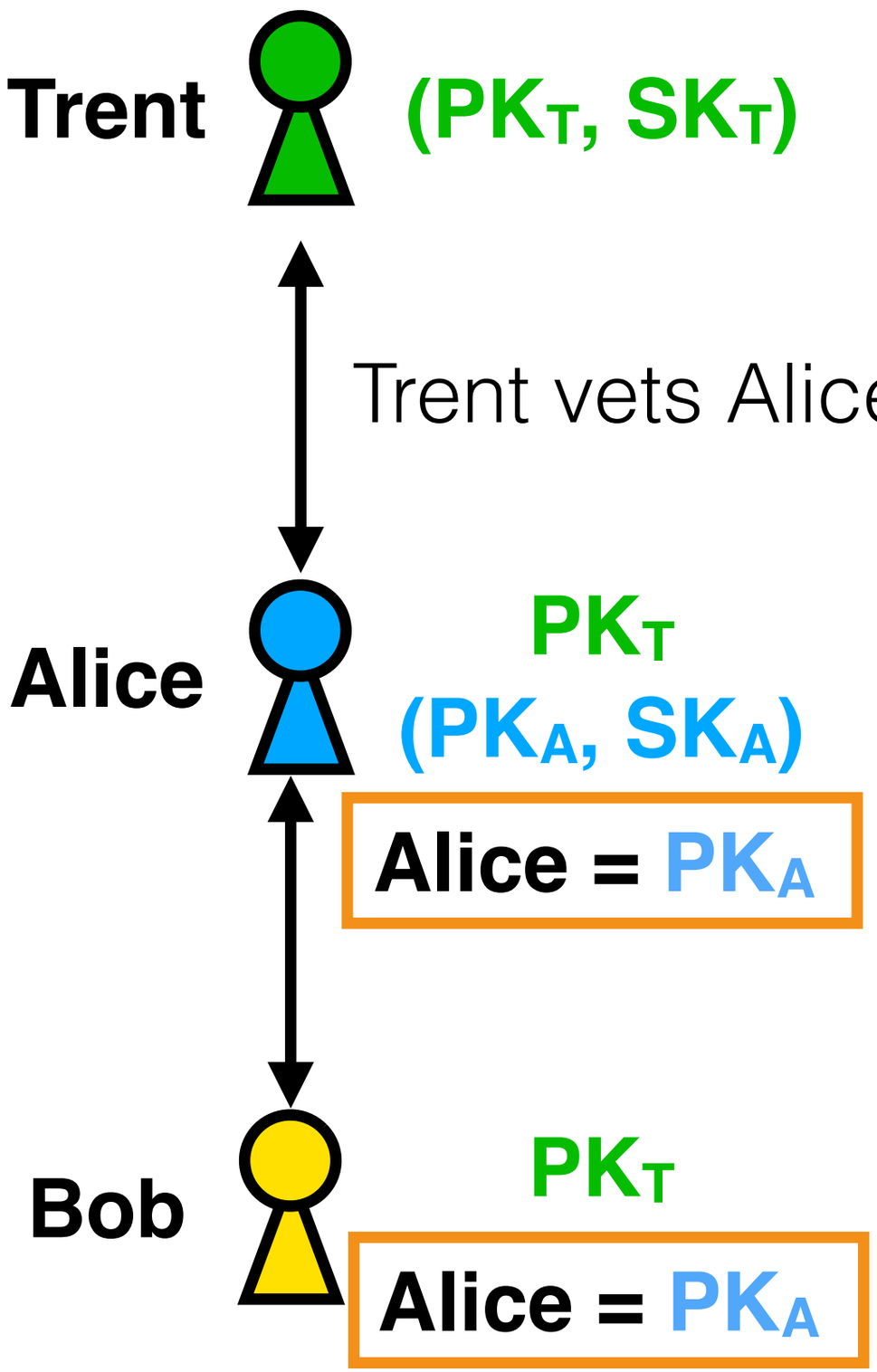
Authentication with public keys



<u>Properties</u>

Trent need be online only when giving out **certificates**, not any time users want to communicate with one another

Authentication with public keys

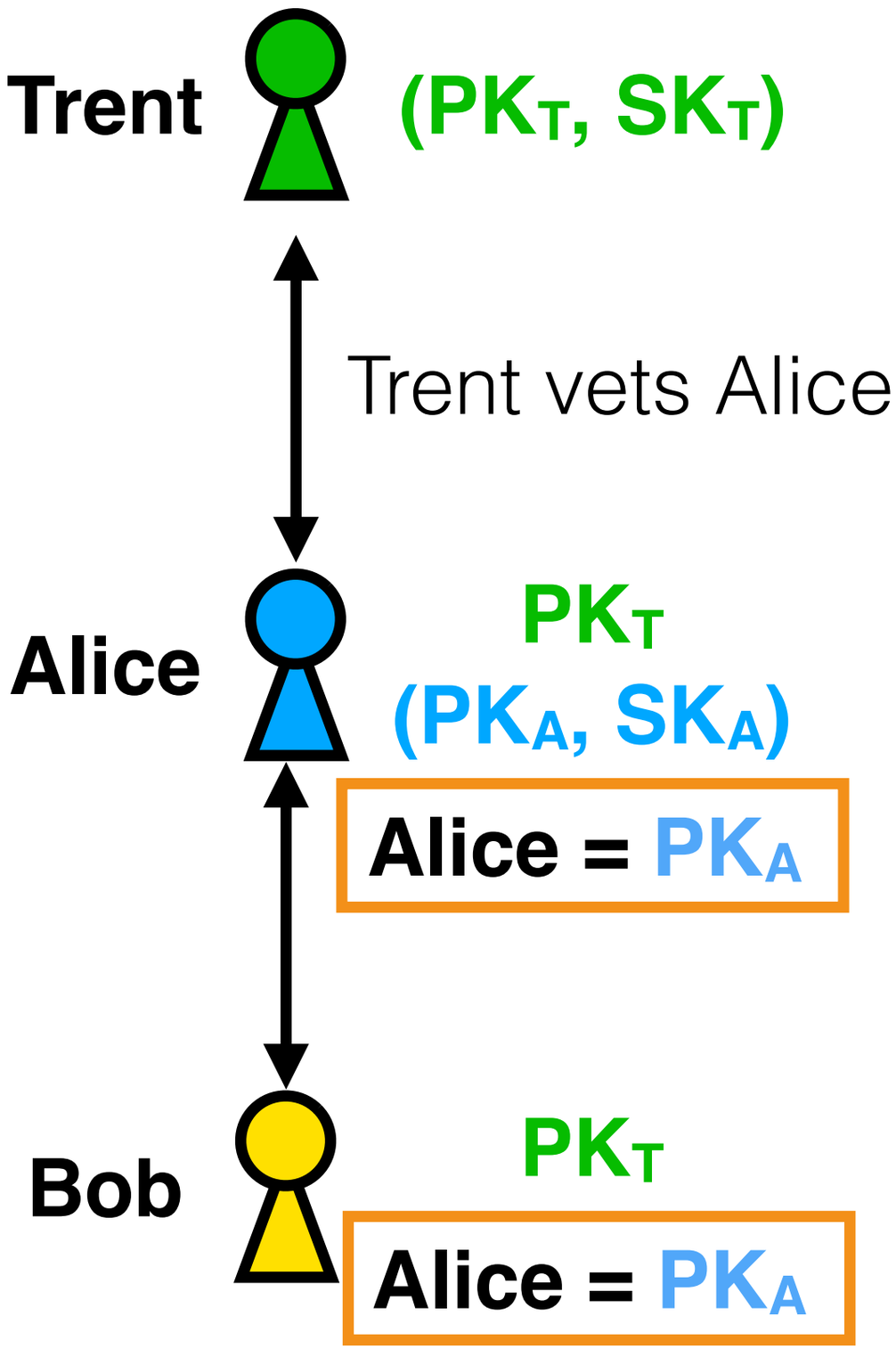


<u>Properties</u>

Trent need be online only when giving out **certificates**, not any time users want to communicate with one another

Alice and Bob can communicate in an authenticated manner without having to go through Trent

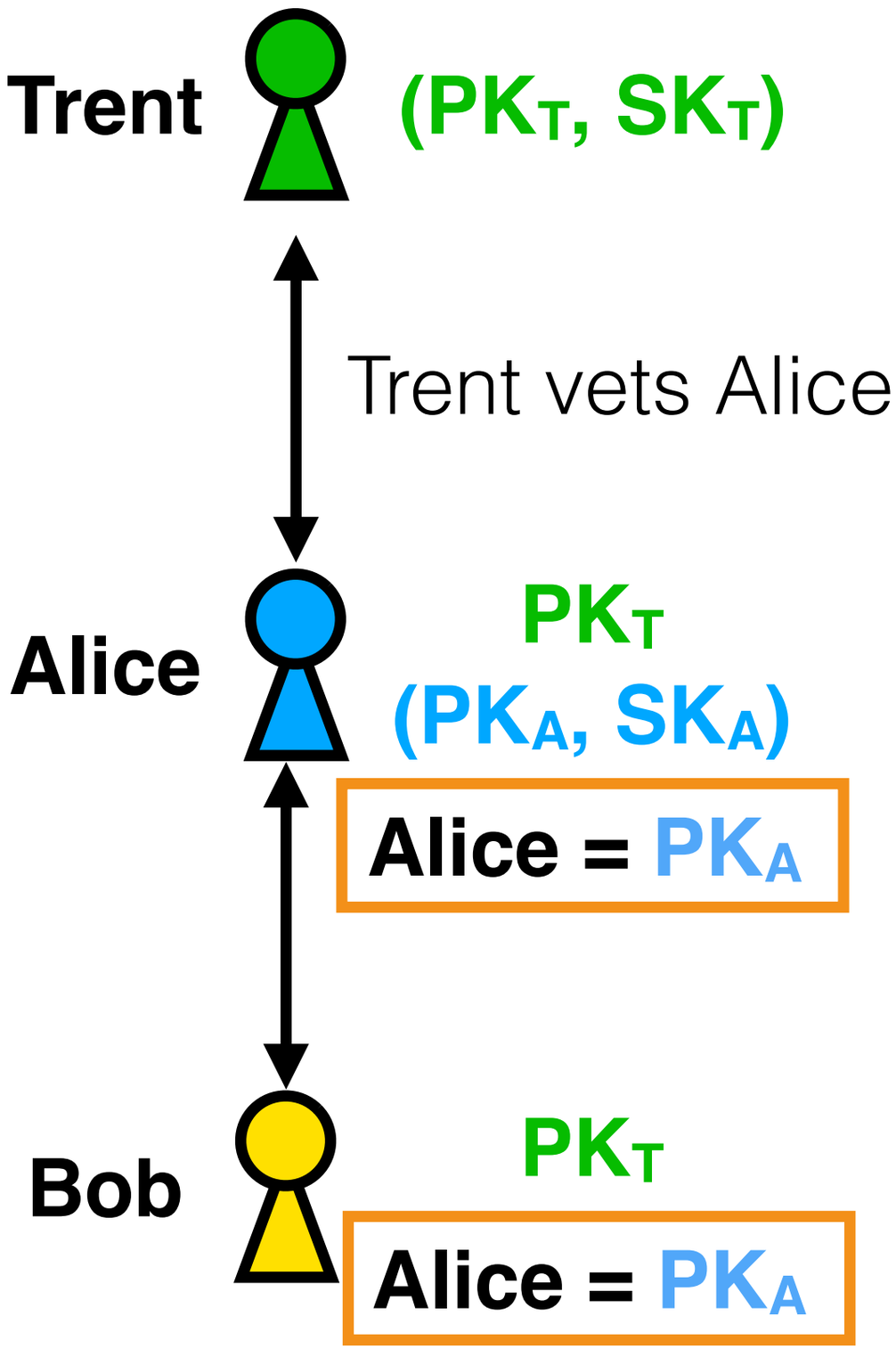
Authentication with public keys



Trust assumptions from our symmetric key protocol:

1. Do not *read* messages
2. Do not *alter* messages
3. Do not *forge* messages
4. Do not *go offline*

Authentication with public keys



Trust assumptions from our symmetric key protocol:

1. Do not *read* messages
2. Do not *alter* messages
3. Do not *forge* messages
4. Do not *go offline*

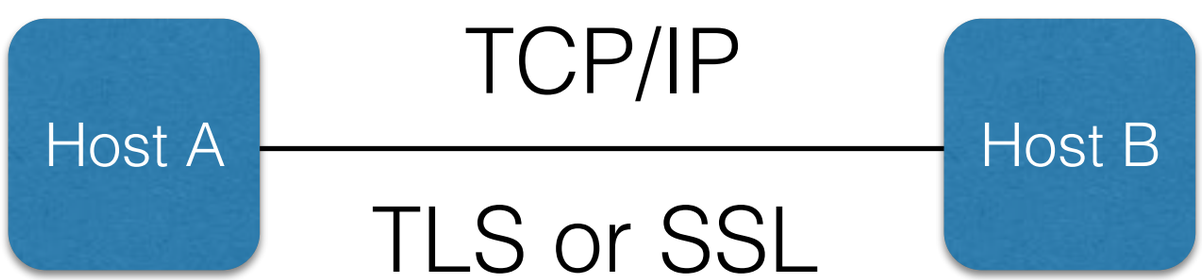
Trust assumptions in this public key protocol:

1. Correctly vet users

(Some more in practice…)

TLS/SSL

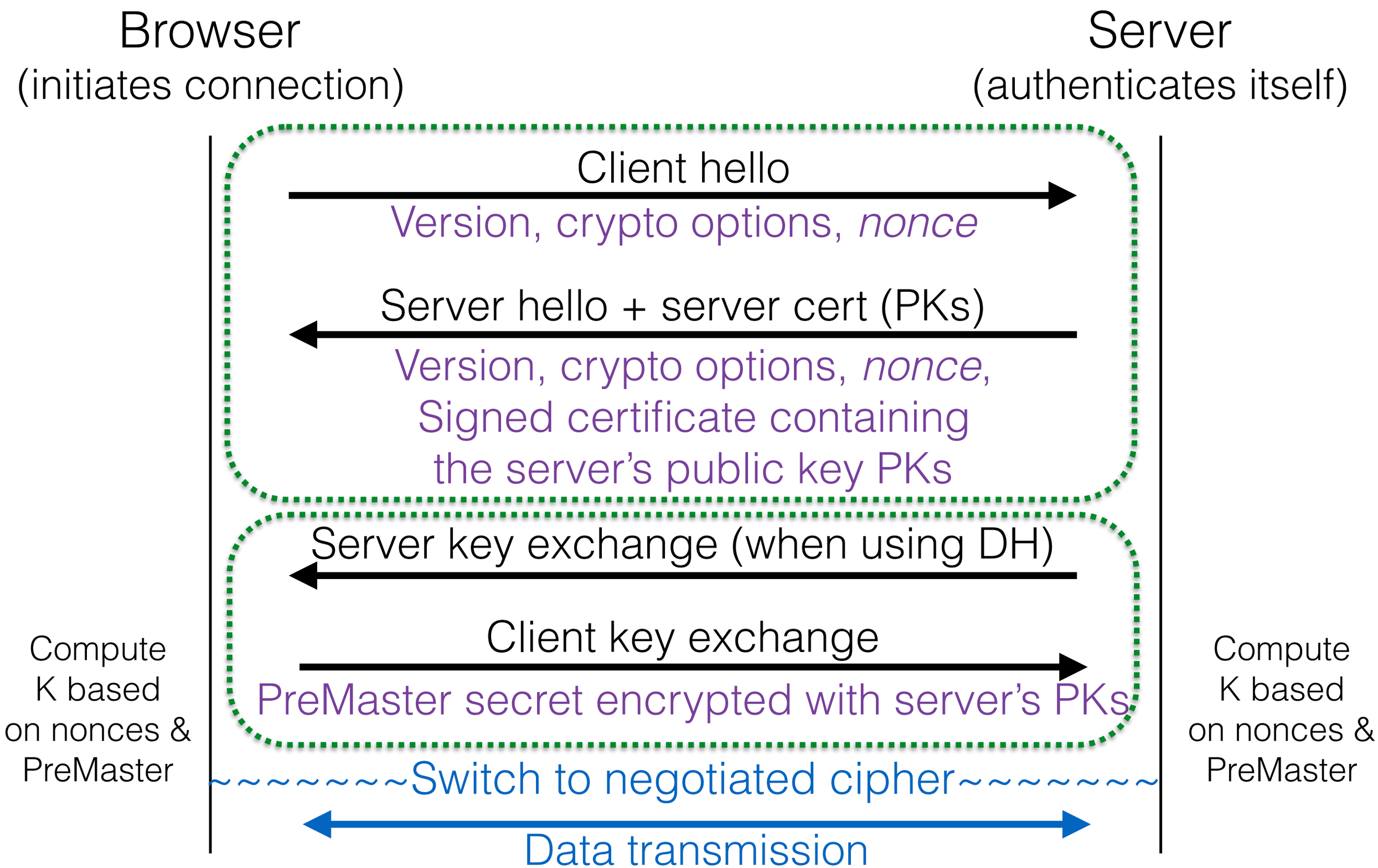
- TLS (Transport Layer Security)
  - A suite of protocols to provide secure communication
    - Confidentiality by applying block & stream ciphers
    - Integrity with MACs
    - Authenticity with certificates
  - Predecessor: SSL (secure sockets layer)
    - TLS was proposed as an upgrade
    - All versions of SSL are considered insecure (recently, the POODLE—padding oracle—attack)



TCP/IP: Host A and B can send packets to one another

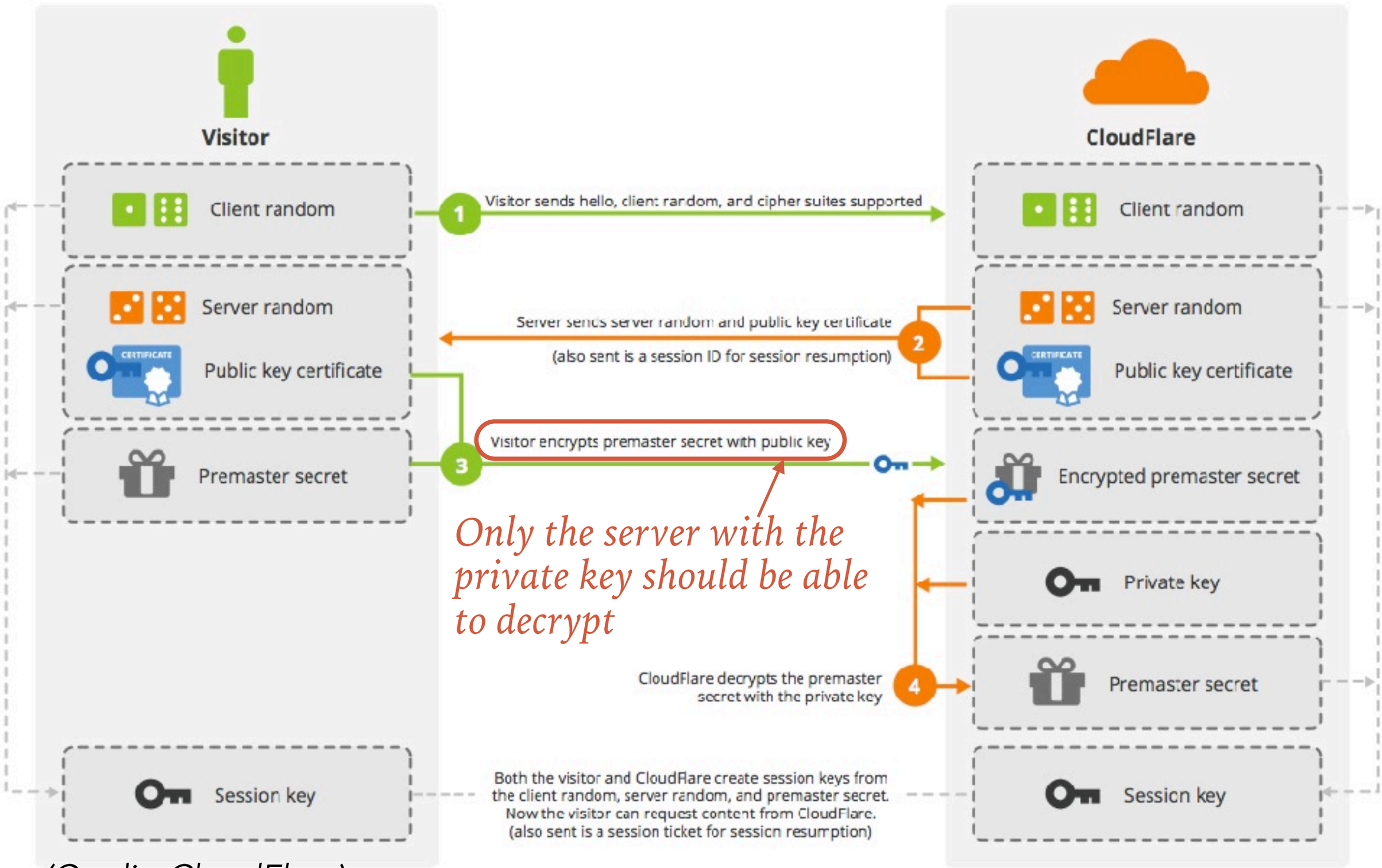
TLS/SSL: operate “over” TCP/IP to ensure security/authenticity

TLS/SSL protocol (high level)



SSL Handshake (RSA)

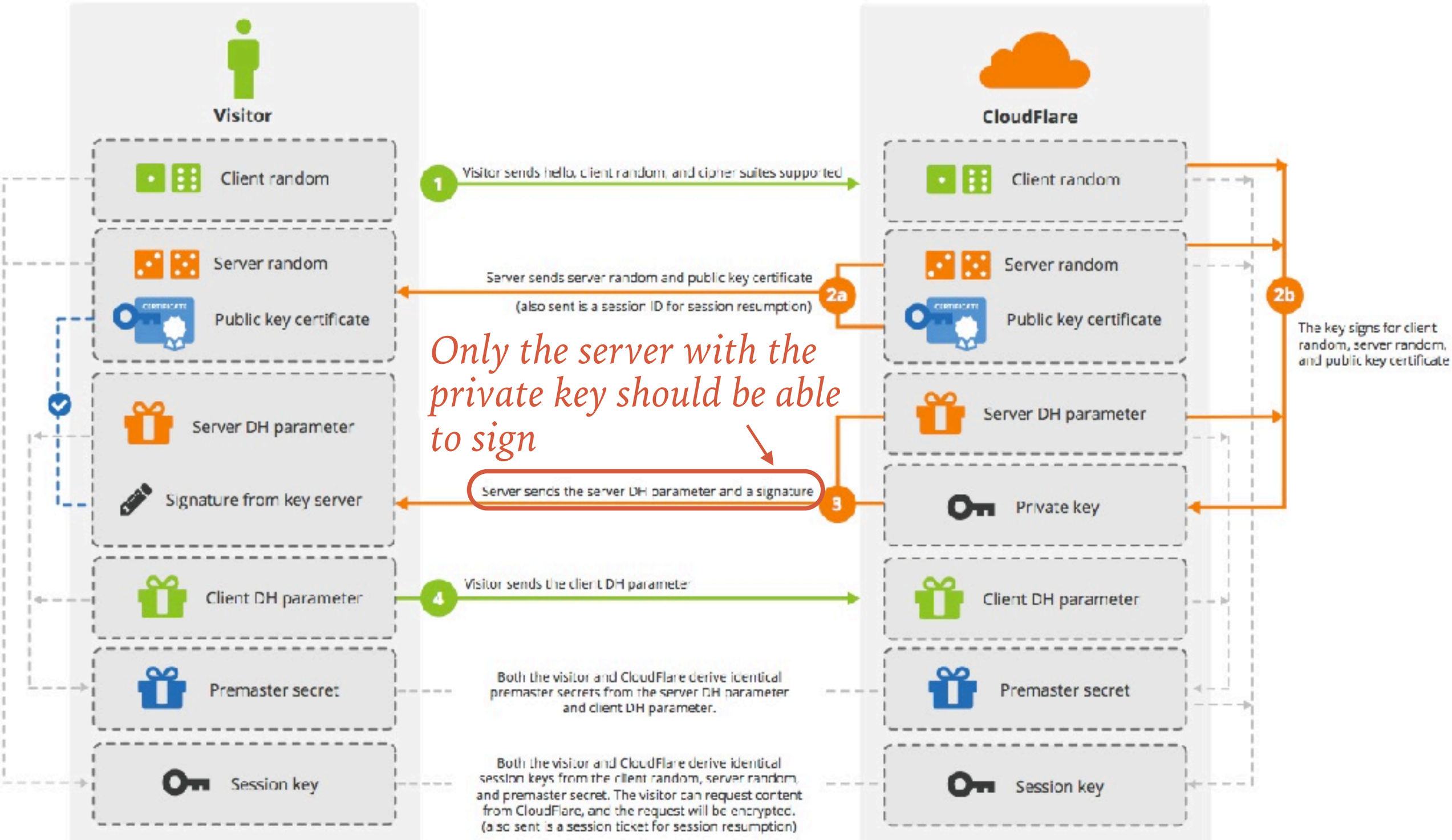
Handshake



(Credit: CloudFlare)

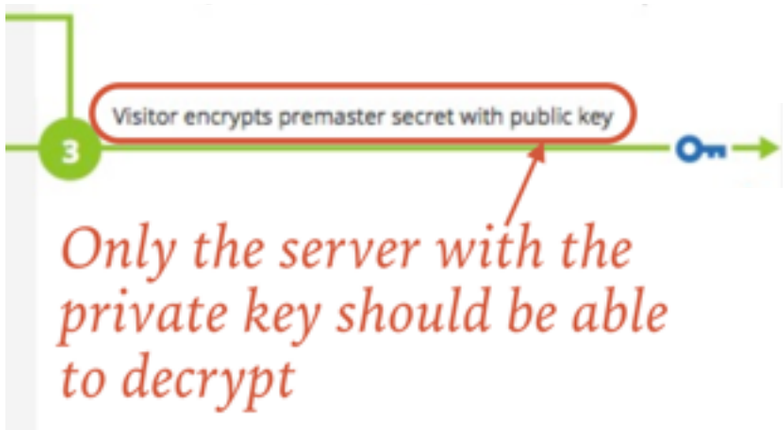
SSL Handshake (Diffie-Hellman)

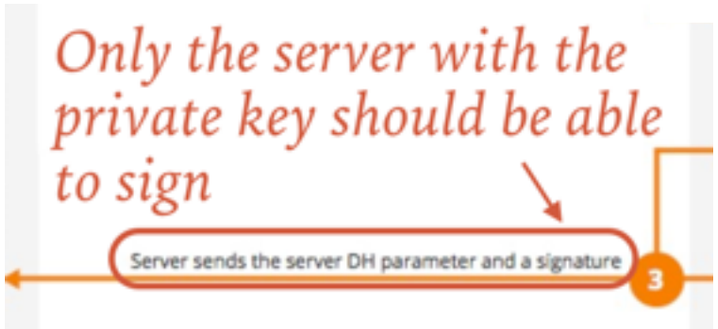
Handshake



(Credit: CloudFlare)

AUTHENTICATED DIFFIE-HELLMAN





Both of these serve as a “challenge/response” protocol:

The client is “challenging” the server to prove that it knows the secret key corresponding to the public key in the certificate

The server is providing a “zero-knowledge proof”:

The server proves that it knows the secret key
without having to reveal the secret key itself

The key property that makes this work:
The only person who knows the secret key is the entity in the certificate

Certificate revocation

3. Trent *signs* a message (with **SK_T**):

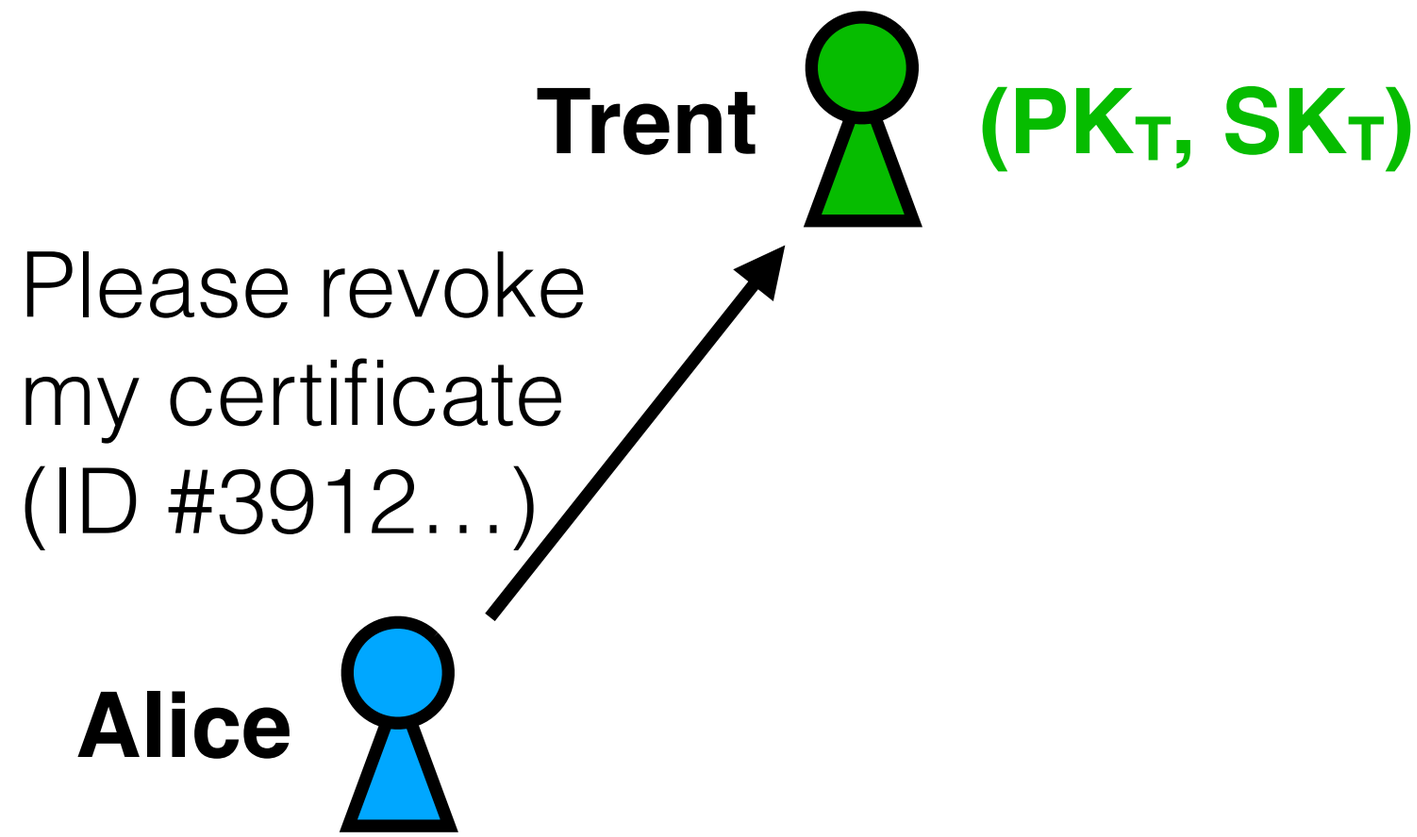
"The owner of the secret key corresponding to **PK_A** is Alice"

This message + sig = **Certificate**

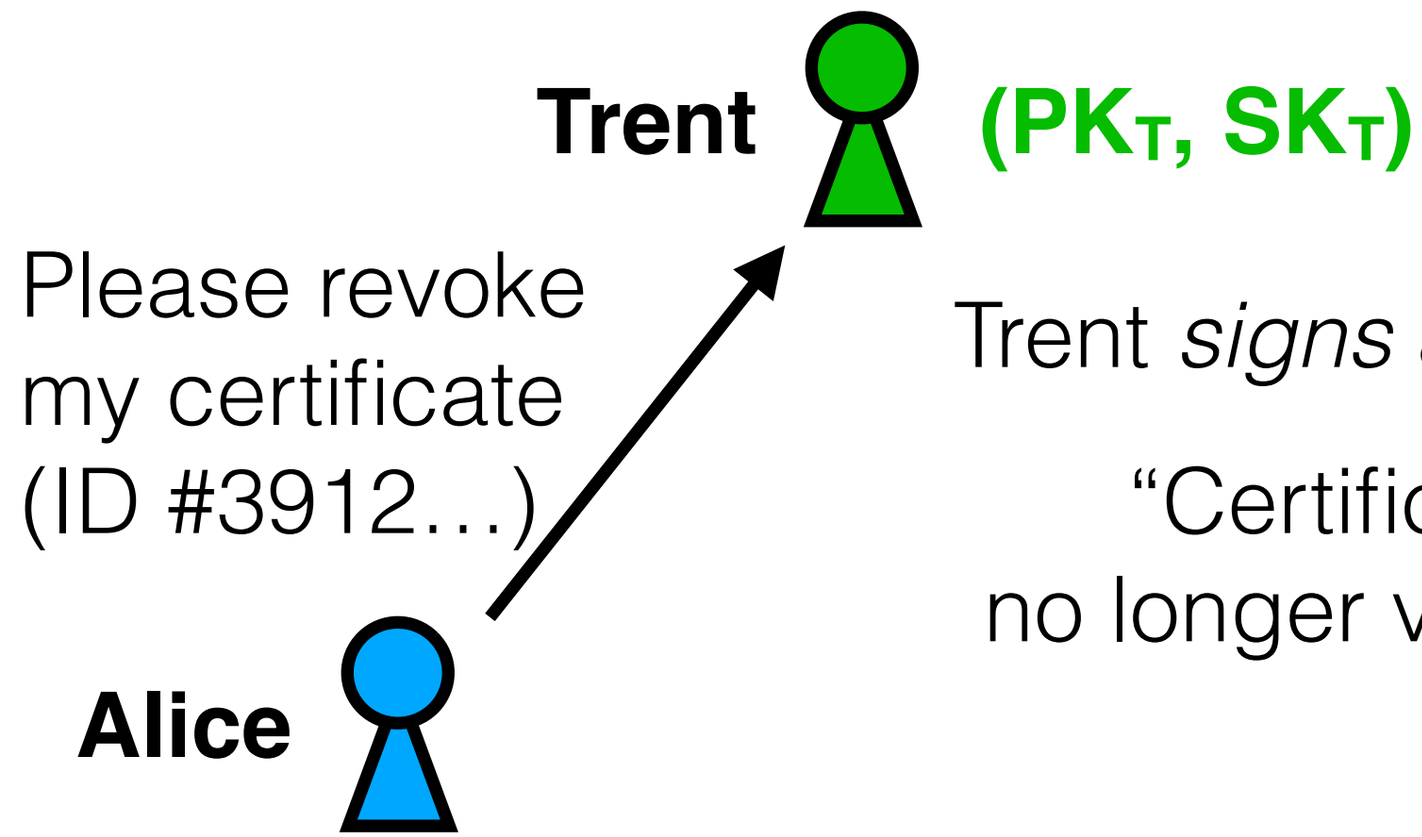
Put another way:
"The only person who knows **SK_A** is Alice"

What happens if Alice's key gets compromised?
(Stolen, accidentally revealed, …)

Certificate revocation



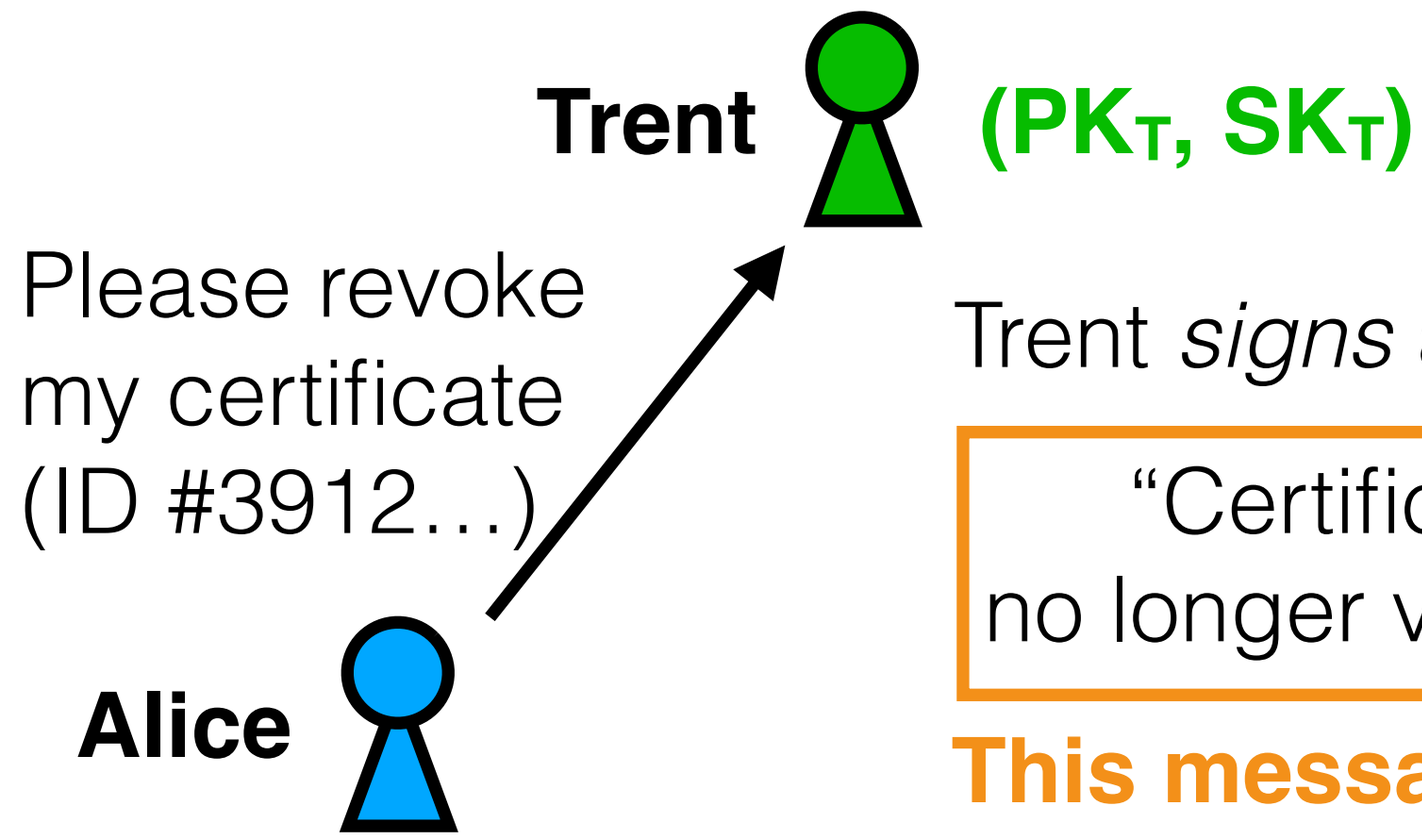
Certificate revocation



Trent *signs* a message (with **SK$_T$**):

"Certificate ID #3912… is no longer valid, as of April 5, …"

Certificate revocation

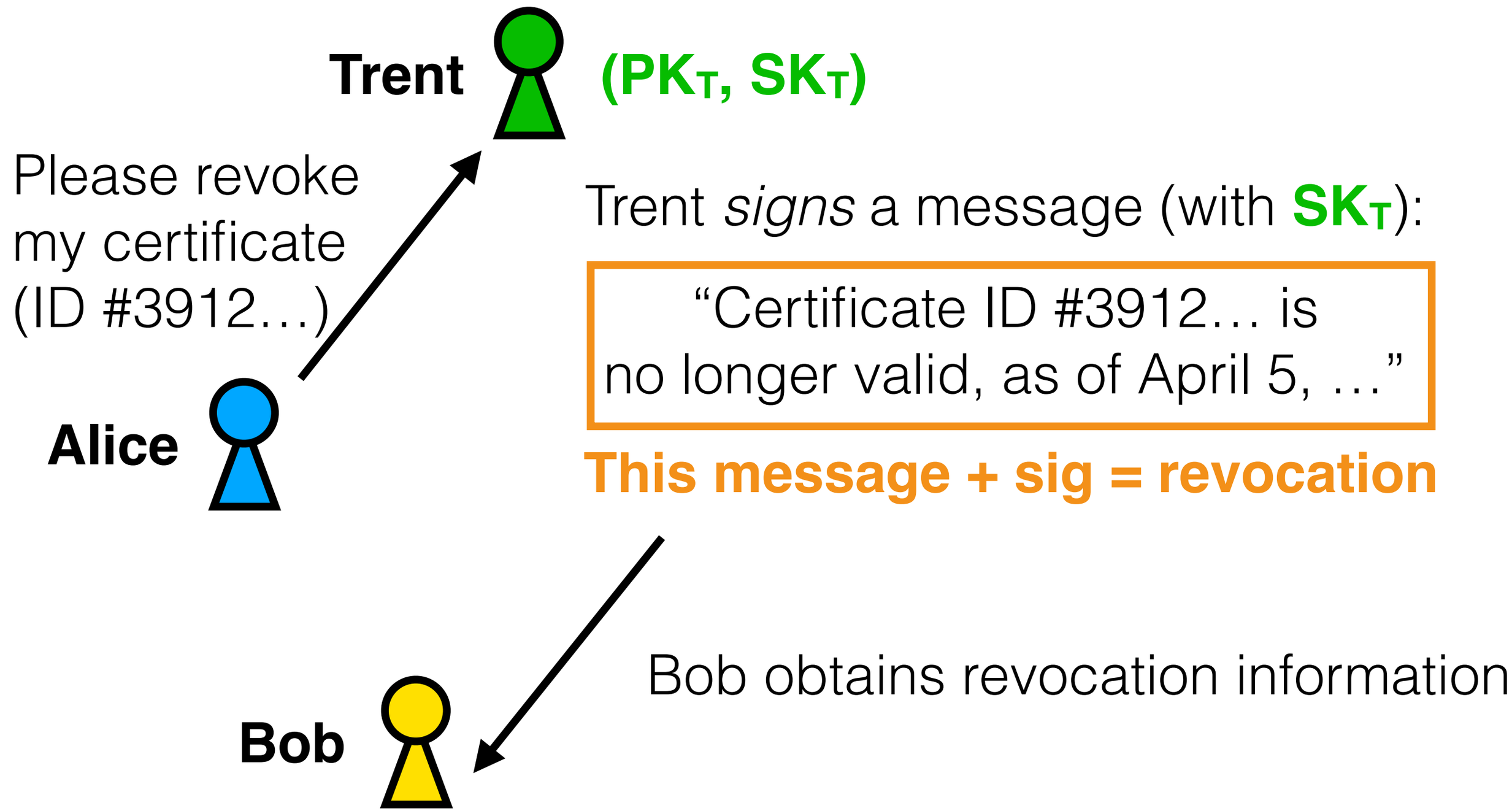


Trent *signs* a message (with **SK$_T$**):

> "Certificate ID #3912… is no longer valid, as of April 5, …"

This message + sig = revocation

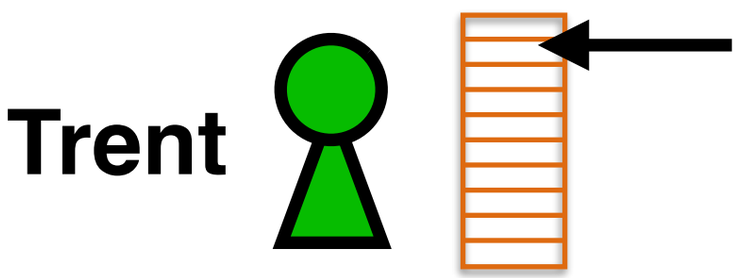
Certificate revocation



Obtaining revocation data

Certificate Revocation Lists (CRLs)

A (often large) signed list of revocations

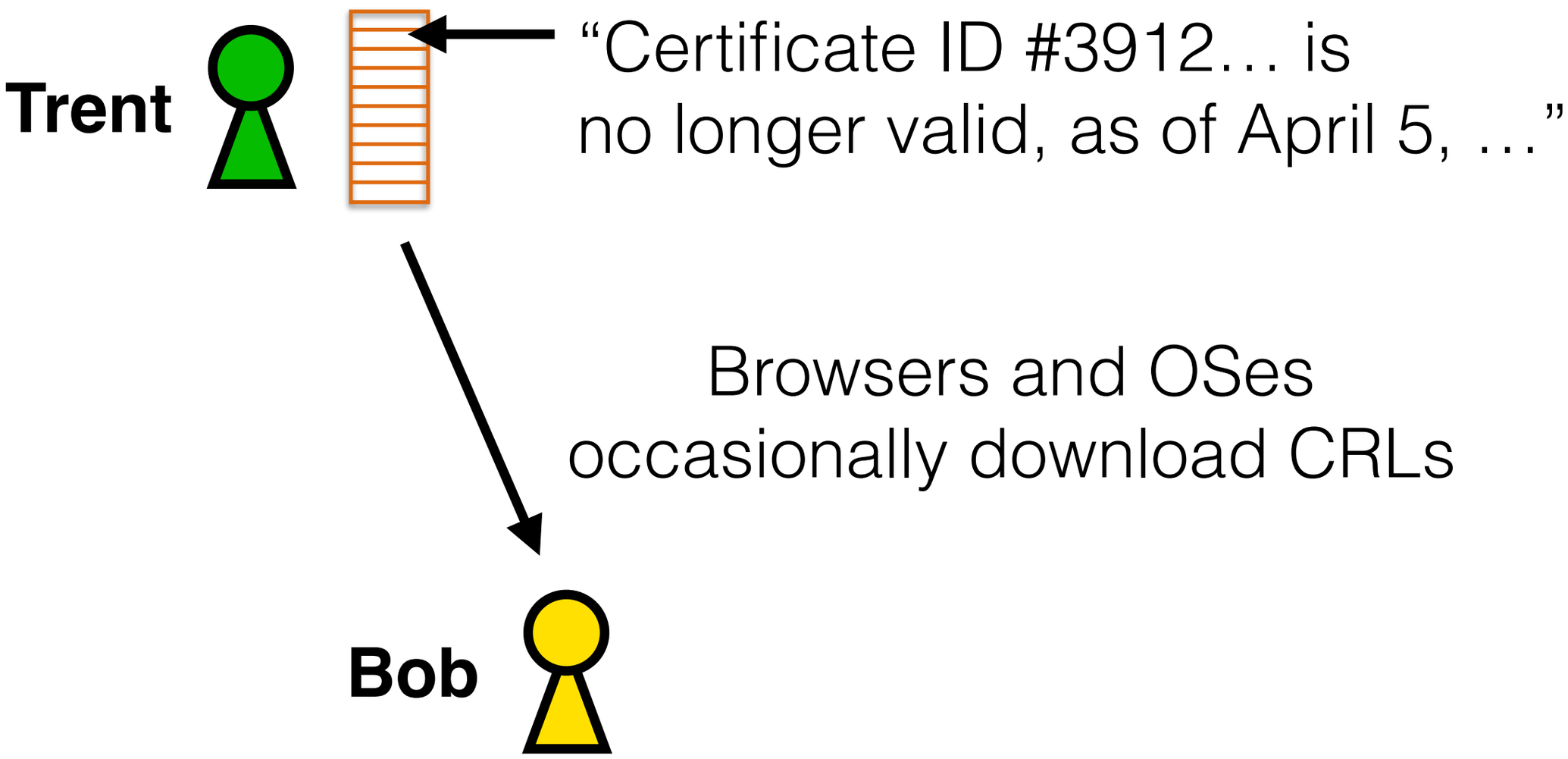


"Certificate ID #3912… is
no longer valid, as of April 5, …"

Obtaining revocation data

Certificate Revocation Lists (CRLs)

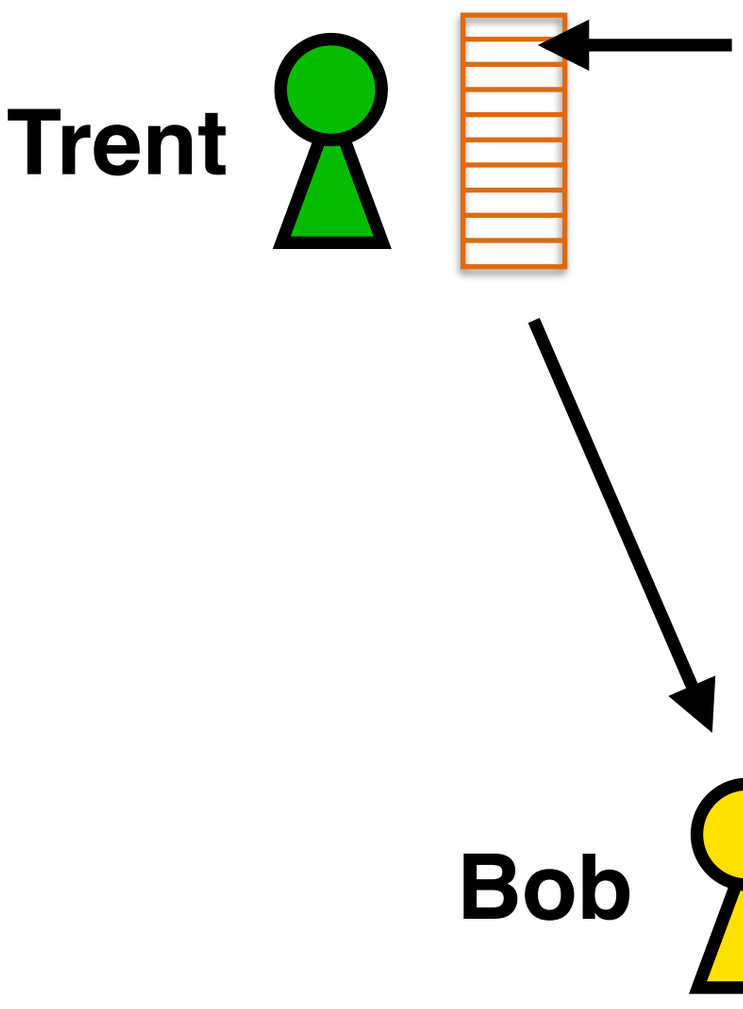
A (often large) signed list of revocations



Obtaining revocation data

Certificate Revocation Lists (CRLs)

A (often large) signed list of revocations



"Certificate ID #3912… is no longer valid, as of April 5, …"

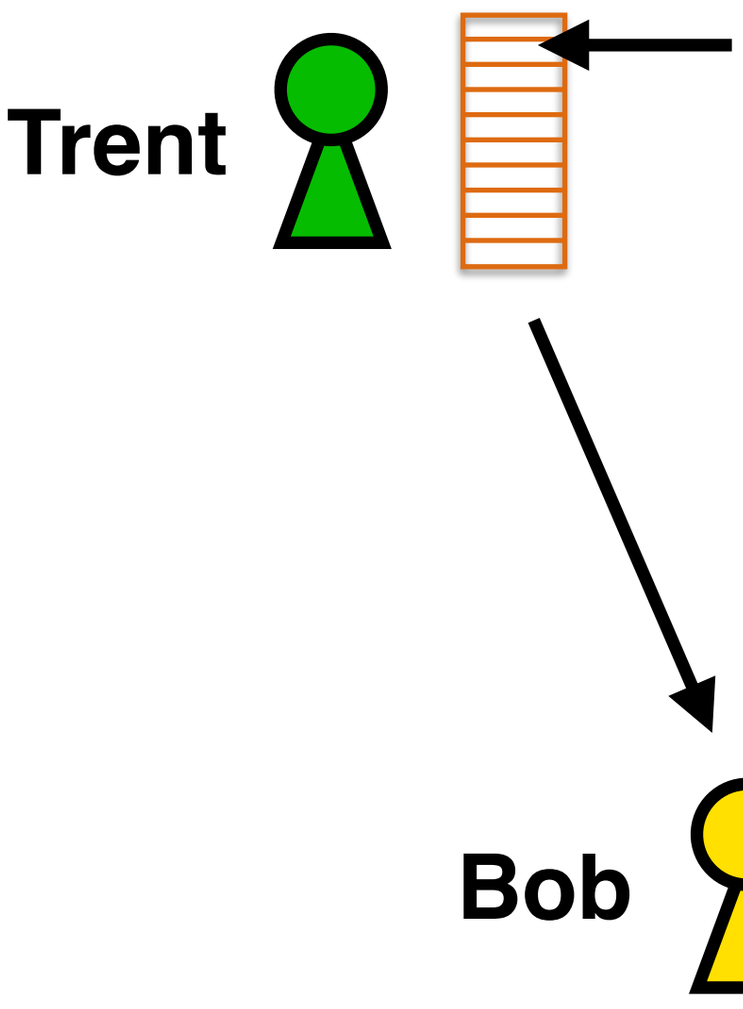
Browsers and OSes occasionally download CRLs

Disincentive: CRLs can be large, so it takes time & bandwidth

Obtaining revocation data

Certificate Revocation Lists (CRLs)

A (often large) signed list of revocations



"Certificate ID #3912… is no longer valid, as of April 5, …"

Browsers and OSes occasionally download CRLs

Disincentive: CRLs can be large, so it takes time & bandwidth

Result: delayed days/weeks/*forever*

Obtaining revocation data

Online Certificate Status Protocol (OCSP)

Browsers and OSes perform OCSP checks on-demand (when verifying the certificate)

Obtaining revocation data

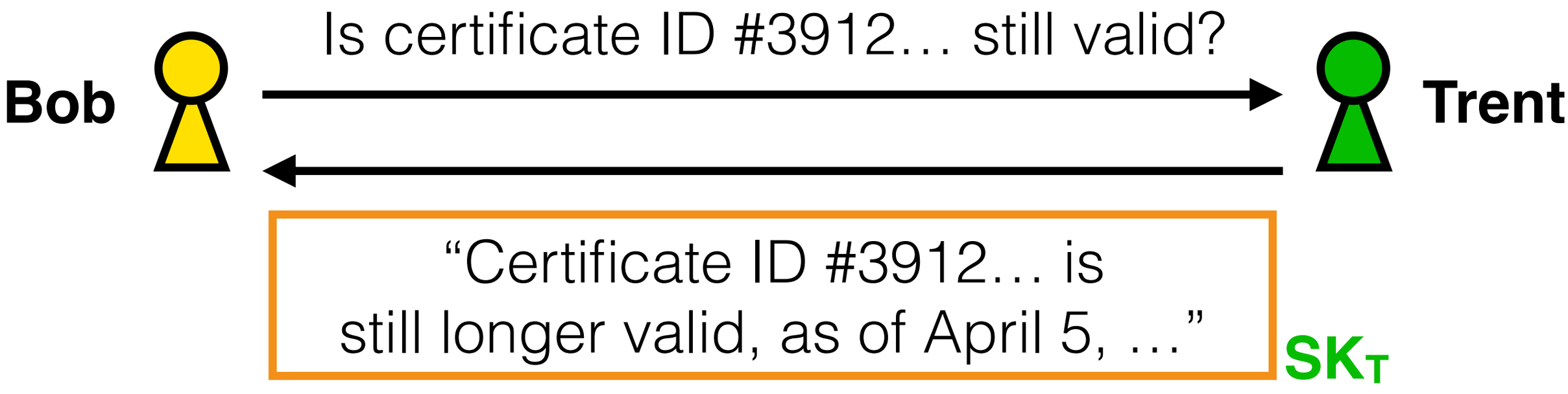
Online Certificate Status Protocol (OCSP)

Browsers and OSes perform OCSP checks on-demand (when verifying the certificate)

Obtaining revocation data

Online Certificate Status Protocol (OCSP)

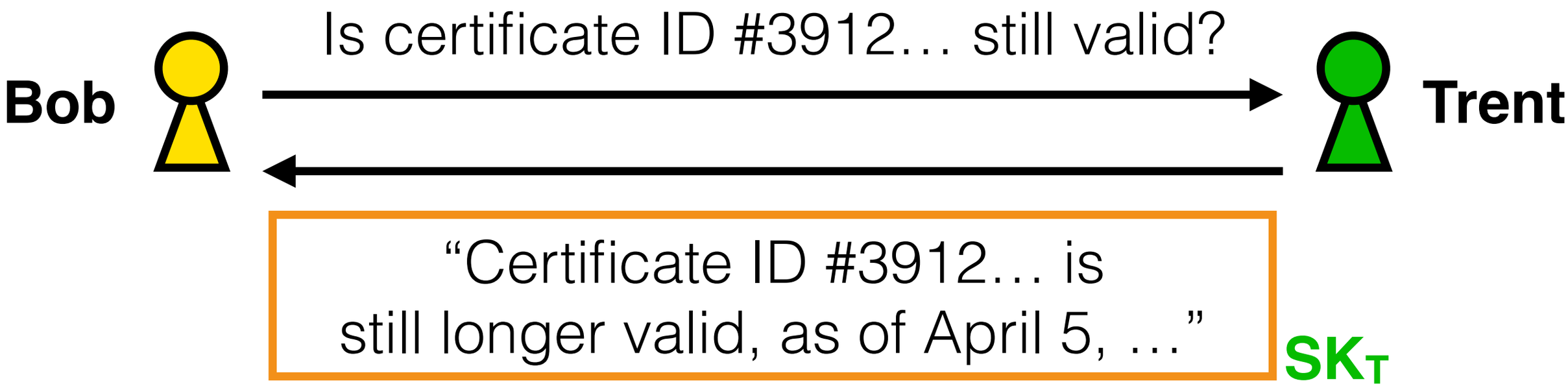
Browsers and OSes perform OCSP checks on-demand (when verifying the certificate)



Obtaining revocation data

Online Certificate Status Protocol (OCSP)

Browsers and OSes perform OCSP checks on-demand (when verifying the certificate)

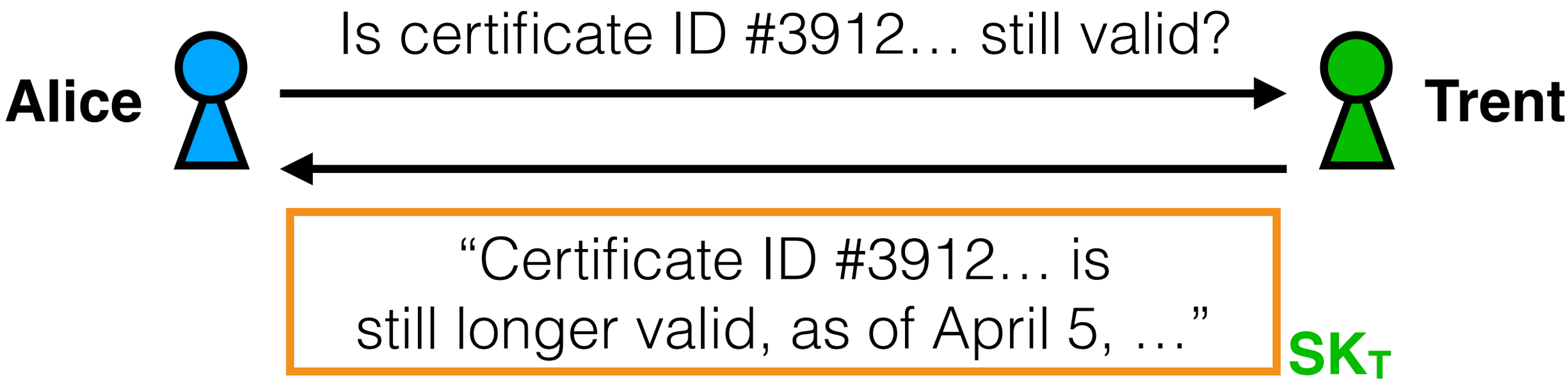


Disincentive: Still delays the initial validation of the certificate (can increase webpage load time)

Obtaining revocation data

OCSP Stapling

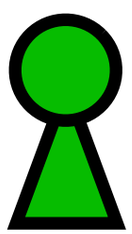
Websites issue OCSP requests,
include responses in initial handshake



Alice forwards this to Bob along with the certificate when they first start to communicate

Certificate revocation responsibilities

Alice's responsibility:
Request revocations

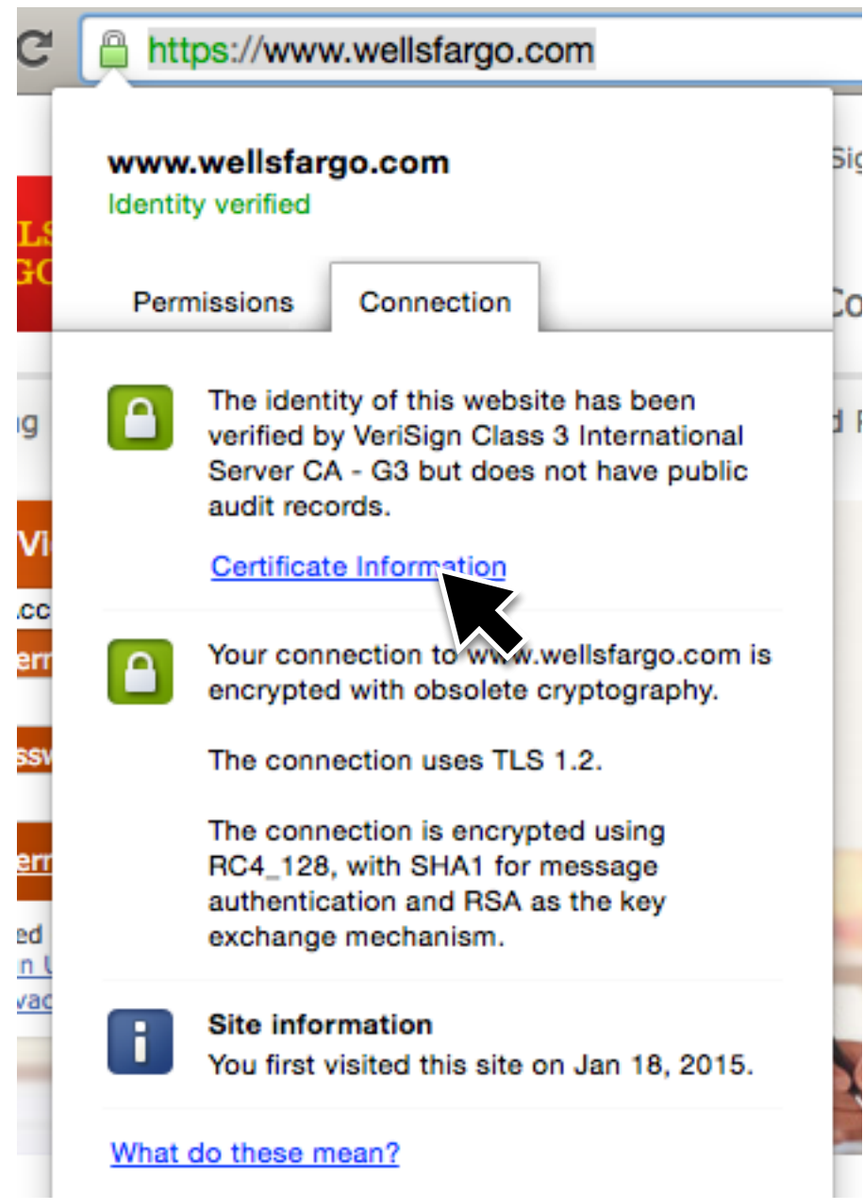


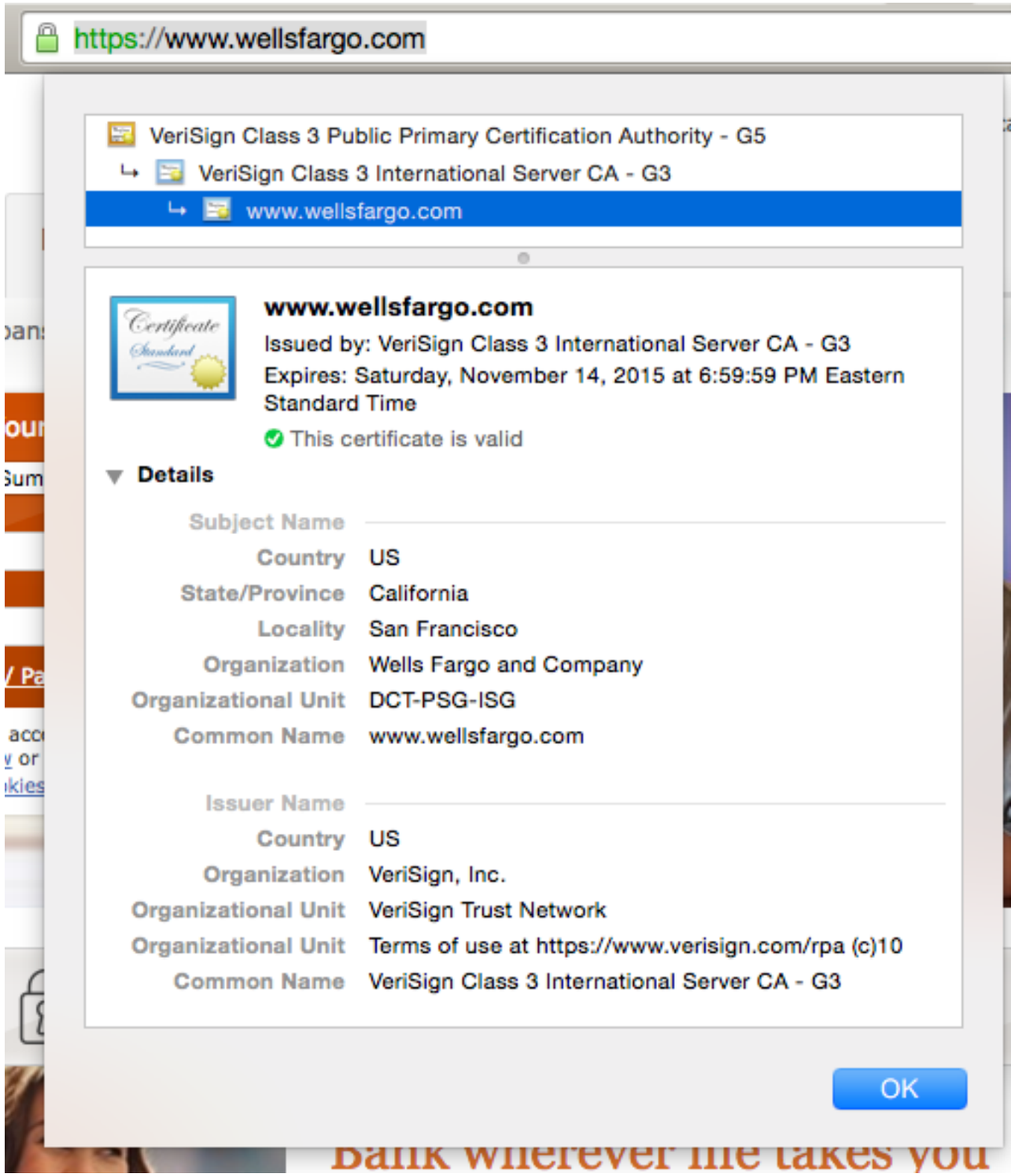
Trent's responsibility:
Make revocations publicly available

Bob's responsibility:
Check for revocations

Certificates in the wild

The lock icon indicates that the browser was able to authenticate the other end, i.e., validate its certificate





Certificate chain

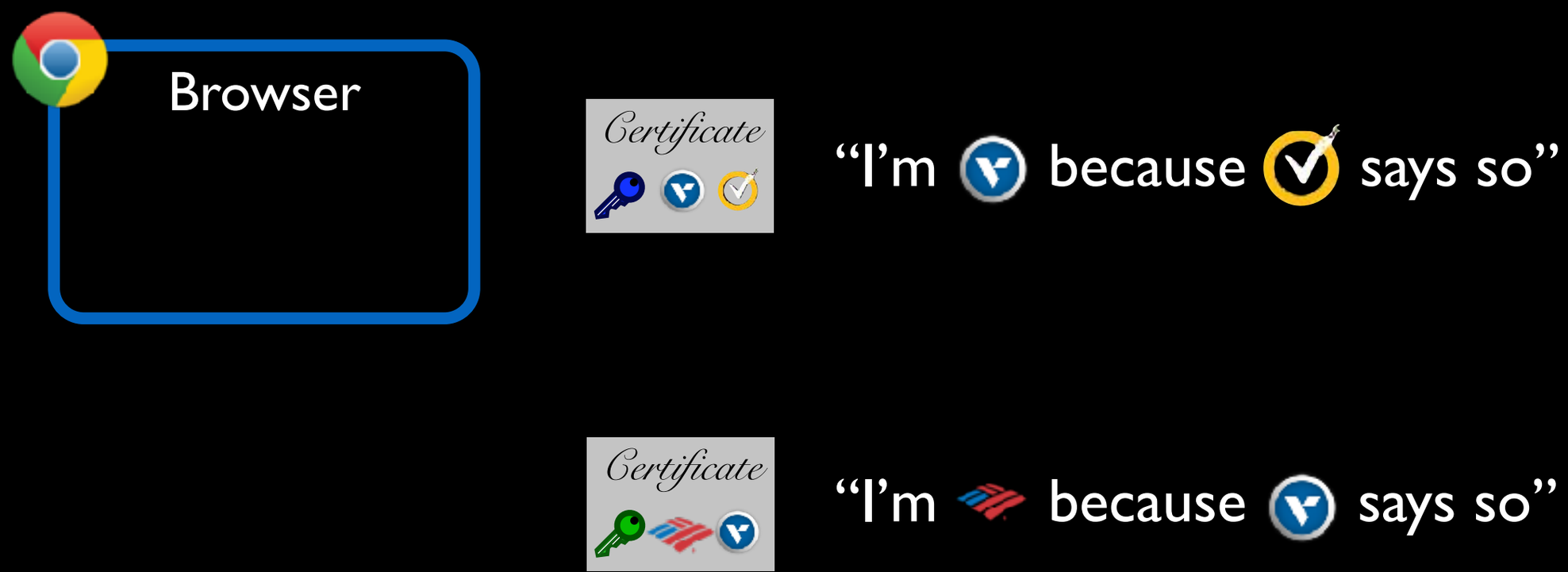
Subject (who owns the public key)

Common name: the URL of the subject

Issuer (who verified the identity and signed this certificate)

Verifying certificates

Verifying certificates



Verifying certificates

"I'm ✓ because I say so!"

"I'm because ✓ says so"

"I'm because ✓ says so"

Verifying certificates

Keychain Access

Click to unlock the System Roots keychain.

Search

Keychains
- login
- iCloud
- System
- System Roots

Category
- All Items
- Passwords
- Secure Notes
- My Certificates
- Keys
- Certificates

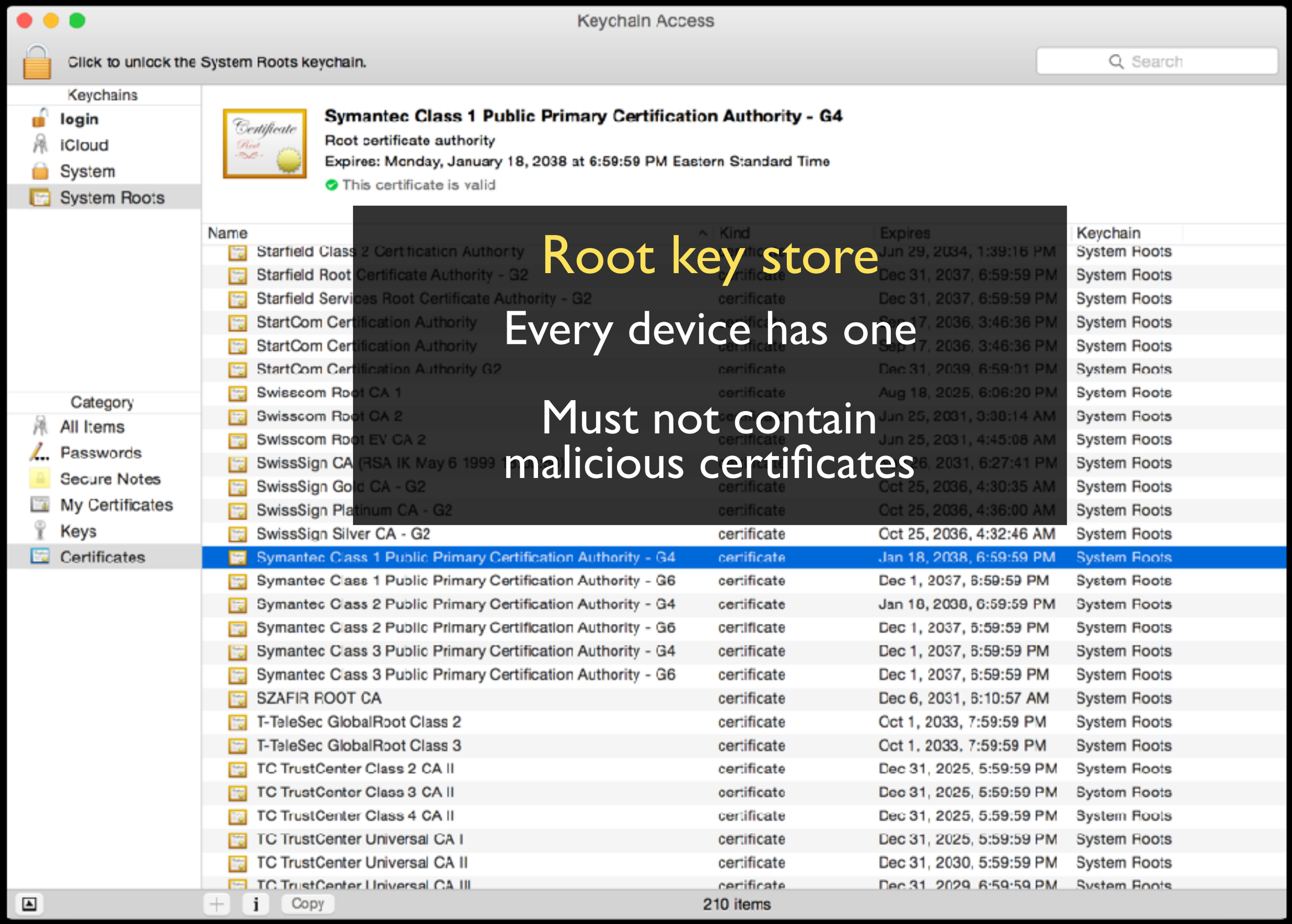
Symantec Class 1 Public Primary Certification Authority - G4
Root certificate authority
Expires: Monday, January 18, 2038 at 6:59:59 PM Eastern Standard Time
This certificate is valid

Name	Kind	Expires	Keychain
Starfield Class 2 Certification Authority	certificate	Jun 29, 2034, 1:39:16 PM	System Roots
Starfield Root Certificate Authority - G2	certificate	Dec 31, 2037, 6:59:59 PM	System Roots
Starfield Services Root Certificate Authority - G2	certificate	Dec 31, 2037, 6:59:59 PM	System Roots
StartCom Certification Authority	certificate	Sep 17, 2036, 3:46:36 PM	System Roots
StartCom Certification Authority	certificate	Sep 17, 2036, 3:46:36 PM	System Roots
StartCom Certification Authority G2	certificate	Dec 31, 2039, 6:59:01 PM	System Roots
Swisscom Root CA 1	certificate	Aug 18, 2025, 6:06:20 PM	System Roots
Swisscom Root CA 2	certificate	Jun 25, 2031, 3:38:14 AM	System Roots
Swisscom Root EV CA 2	certificate	Jun 25, 2031, 4:45:08 AM	System Roots
SwissSign CA (RSA IK May 6 1999 18:00:58)	certificate	Nov 26, 2031, 6:27:41 PM	System Roots
SwissSign Gold CA - G2	certificate	Oct 25, 2036, 4:30:35 AM	System Roots
SwissSign Platinum CA - G2	certificate	Oct 25, 2036, 4:36:00 AM	System Roots
SwissSign Silver CA - G2	certificate	Oct 25, 2036, 4:32:46 AM	System Roots
Symantec Class 1 Public Primary Certification Authority - G4	certificate	Jan 18, 2038, 6:59:59 PM	System Roots
Symantec Class 1 Public Primary Certification Authority - G6	certificate	Dec 1, 2037, 6:59:59 PM	System Roots
Symantec Class 2 Public Primary Certification Authority - G4	certificate	Jan 18, 2038, 6:59:59 PM	System Roots
Symantec Class 2 Public Primary Certification Authority - G6	certificate	Dec 1, 2037, 6:59:59 PM	System Roots
Symantec Class 3 Public Primary Certification Authority - G4	certificate	Dec 1, 2037, 6:59:59 PM	System Roots
Symantec Class 3 Public Primary Certification Authority - G6	certificate	Dec 1, 2037, 6:59:59 PM	System Roots
SZAFIR ROOT CA	certificate	Dec 6, 2031, 6:10:57 AM	System Roots
T-TeleSec GlobalRoot Class 2	certificate	Oct 1, 2033, 7:59:59 PM	System Roots
T-TeleSec GlobalRoot Class 3	certificate	Oct 1, 2033, 7:59:59 PM	System Roots
TC TrustCenter Class 2 CA II	certificate	Dec 31, 2025, 5:59:59 PM	System Roots
TC TrustCenter Class 3 CA II	certificate	Dec 31, 2025, 5:59:59 PM	System Roots
TC TrustCenter Class 4 CA II	certificate	Dec 31, 2025, 5:59:59 PM	System Roots
TC TrustCenter Universal CA I	certificate	Dec 31, 2025, 5:59:59 PM	System Roots
TC TrustCenter Universal CA II	certificate	Dec 31, 2030, 5:59:59 PM	System Roots
TC TrustCenter Universal CA III	certificate	Dec 31, 2029, 6:59:59 PM	System Roots

Copy

210 items

Verifying certificates



Verifying certificates

"I'm ✓ because I say so!"

"I'm because ✓ says so"

"I'm because ✓ says so"

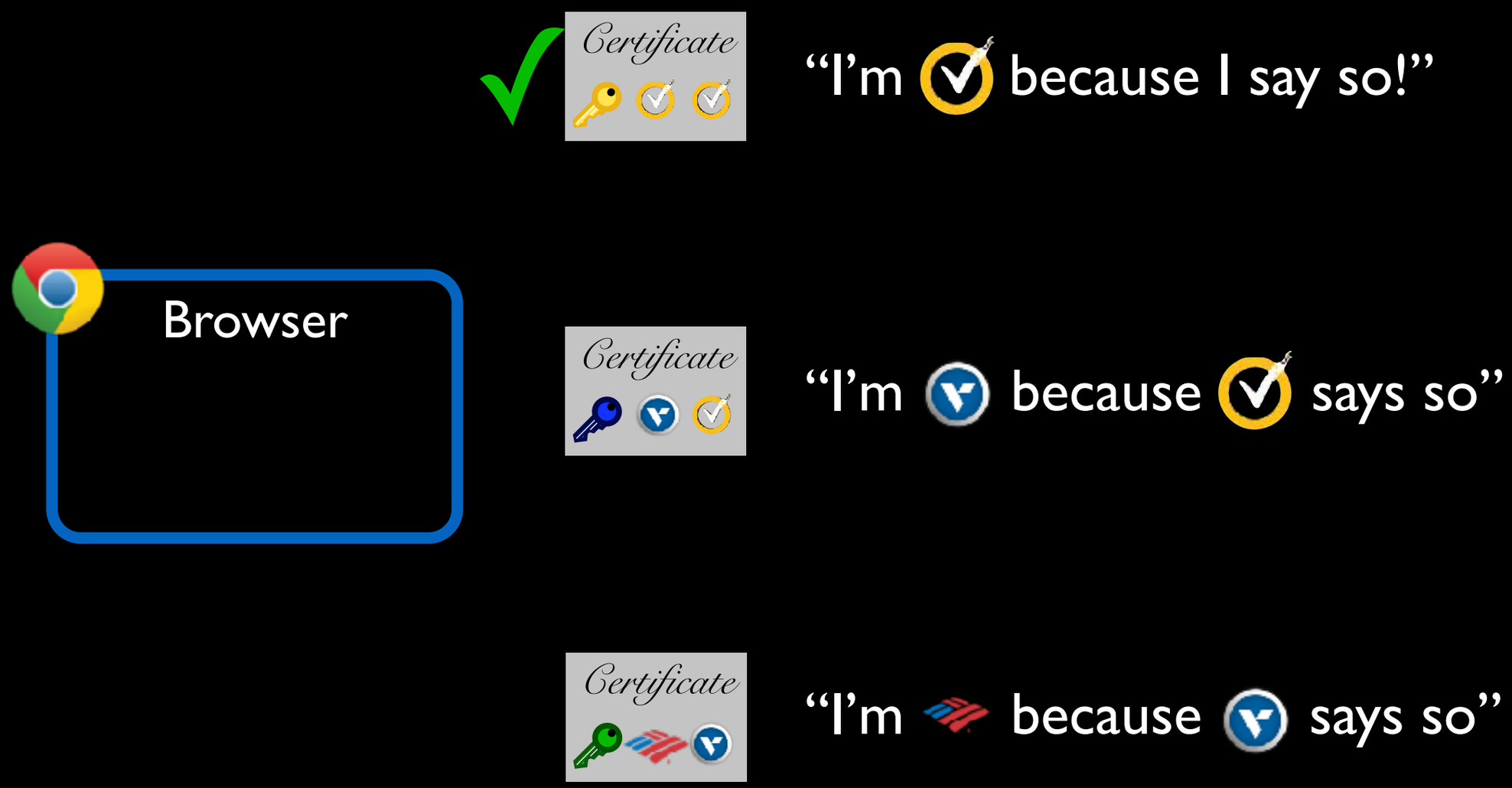
Verifying certificates

"I'm ✓ because I say so!"

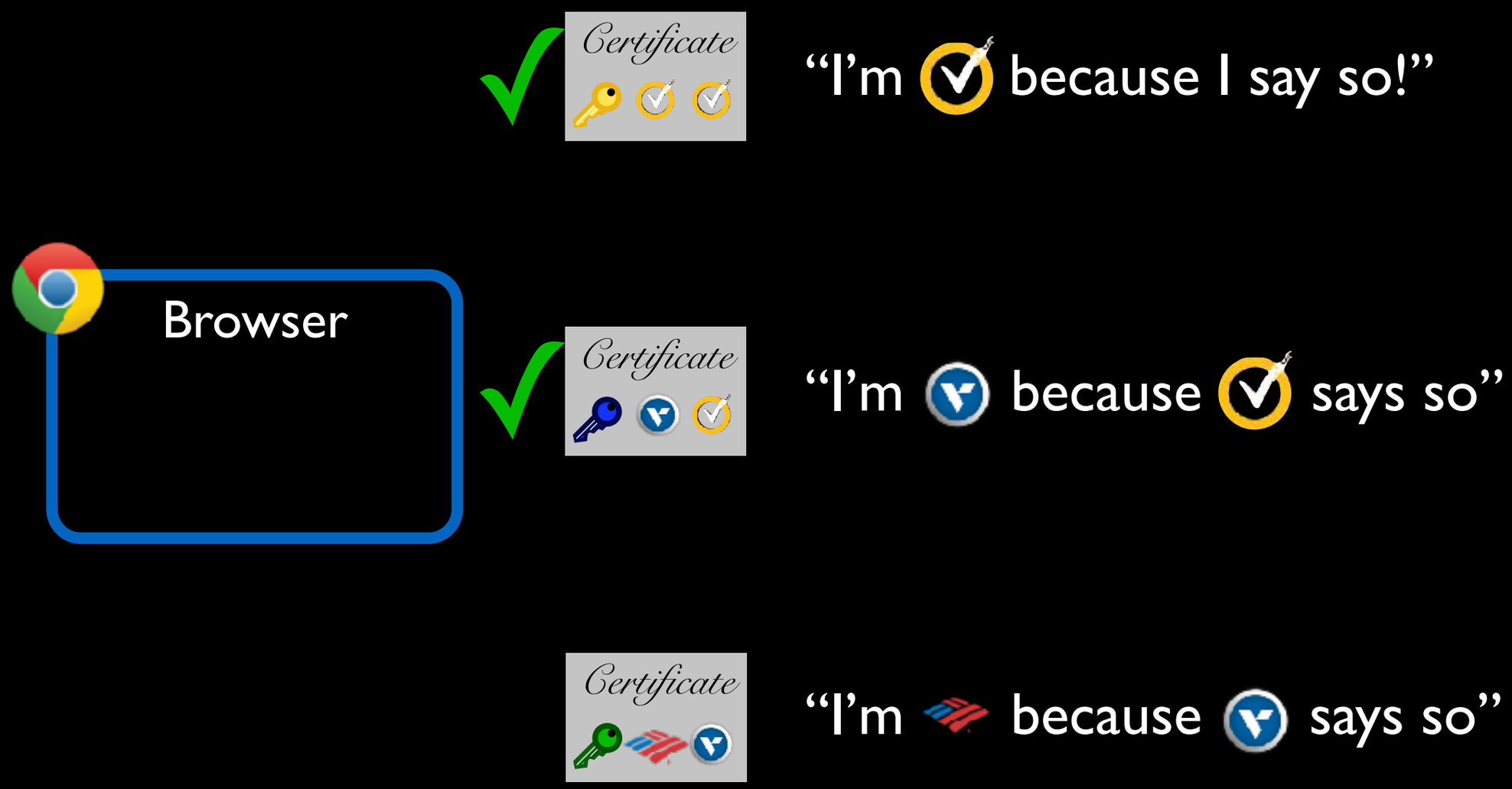
"I'm because ✓ says so"

"I'm because ✓ says so"

Verifying certificates



Verifying certificates



Verifying certificates

"I'm because I say so!"

"I'm because says so"

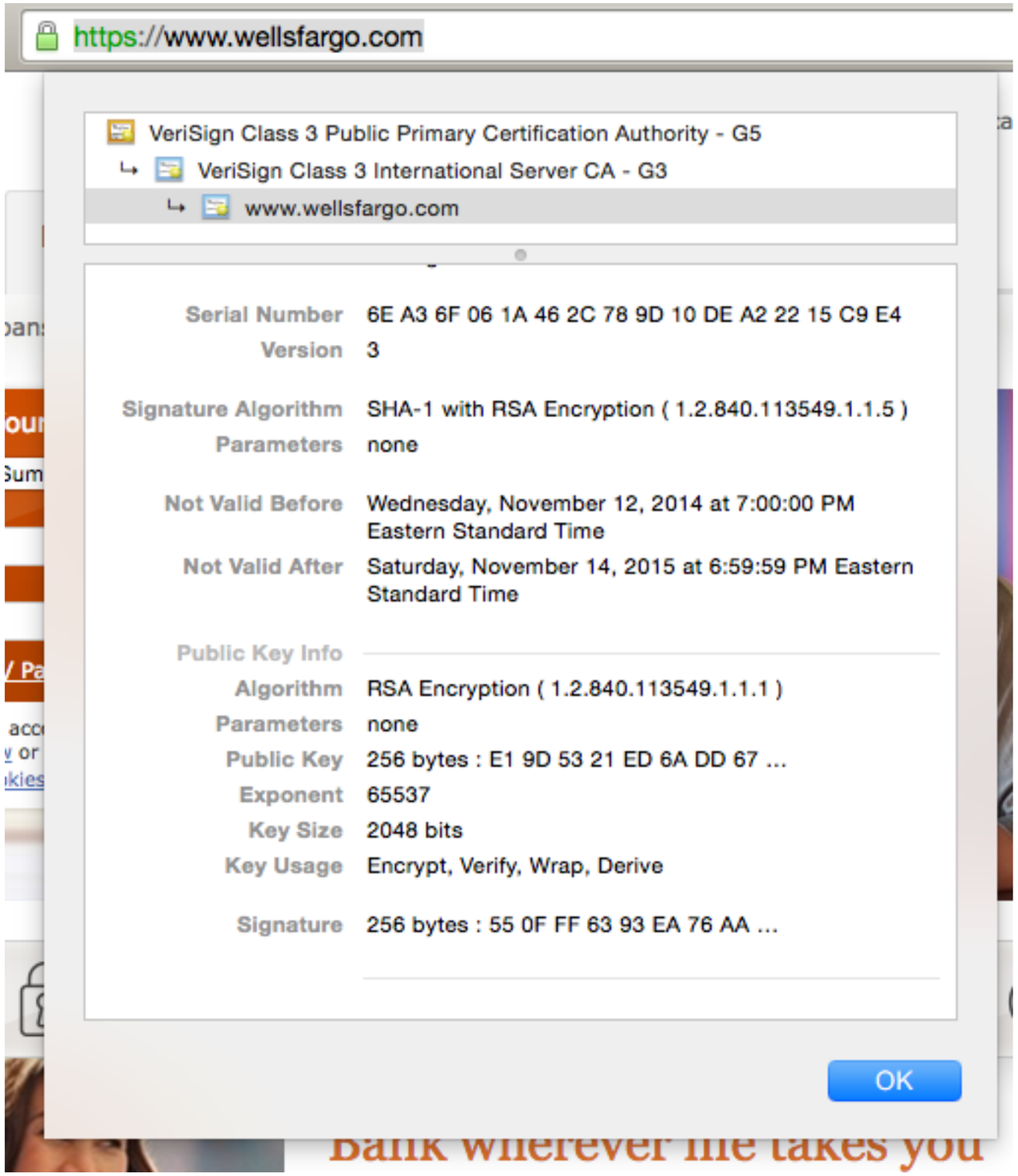
"I'm because says so"

Verifying certificates

"I'm ✓ because I say so!"

"I'm because ✓ says so"

"I'm because says so"

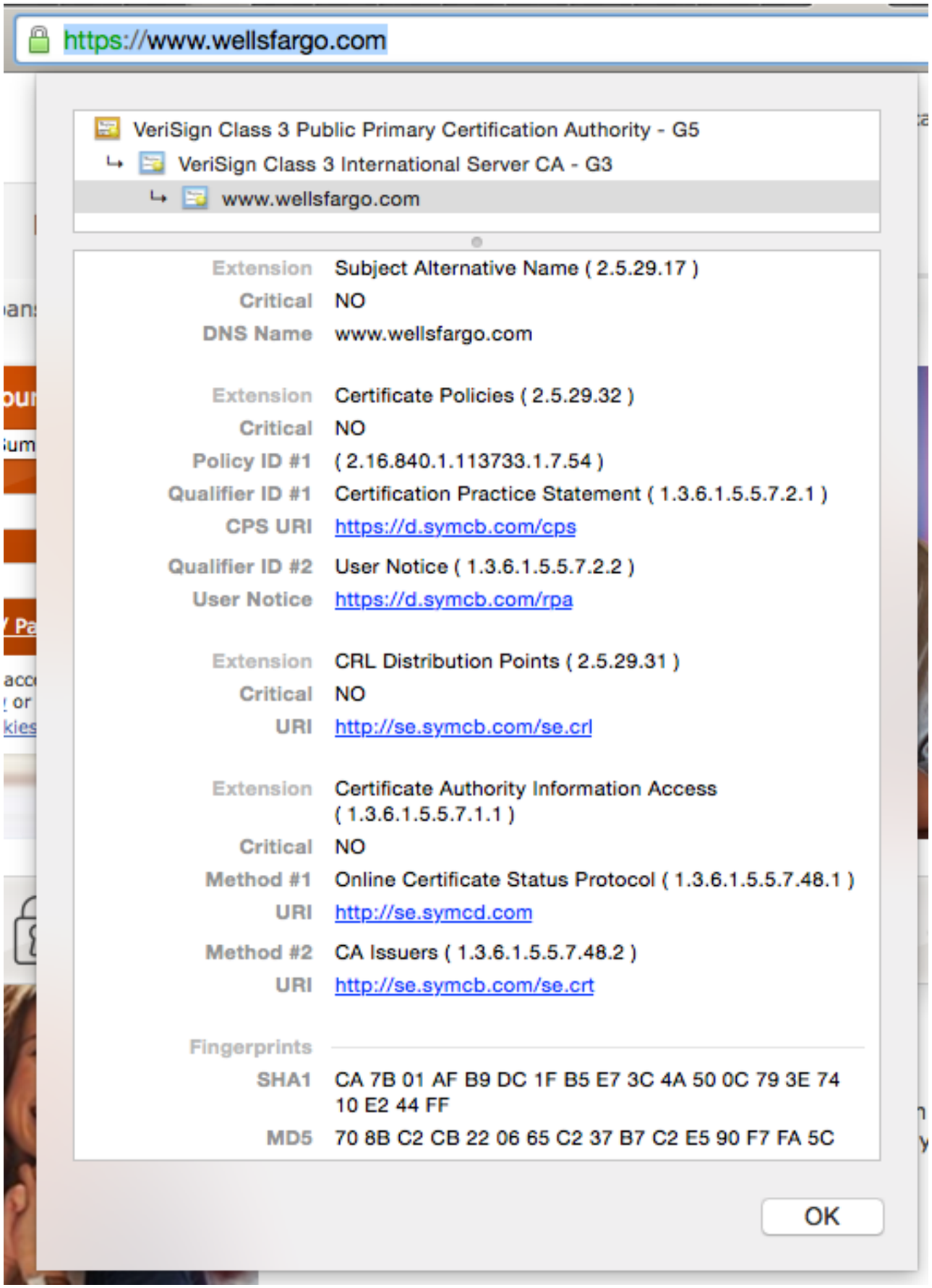


Serial number: Uniquely identifies this cert with respect to the issuer (look for this in CRLs)

Signature algorithm: How the issuer will sign parts of the cert

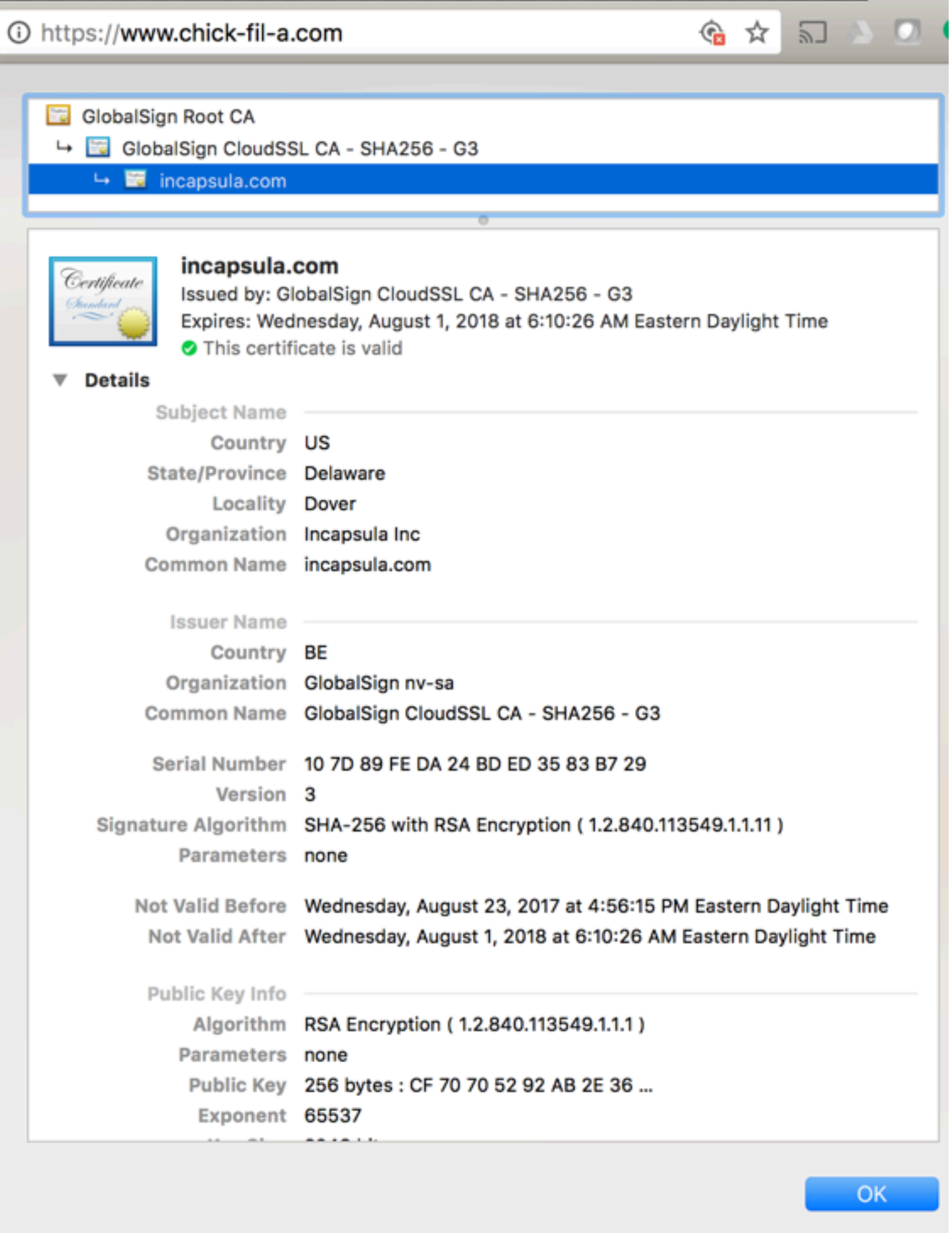
Not valid before/after: When to start and stop believing this cert (start & expiration dates)

The public key: And the issuer's signature of the public key



Subject Alternate Names:
Other URLs for which this cert should be considered valid. (wellsfargo.com is not the same as www.wellsfargo.com)

Can include wildcards, e.g., *.google.com



Subject Alternate Names:
The *spirit* is that it represents different domain names of the same entity
(google.com, google.co.uk, youtube.com, …)

The *letter* of the rule doesn't say that they need to be the same company—or really have anything in common

https://www.chick-fil-a.com

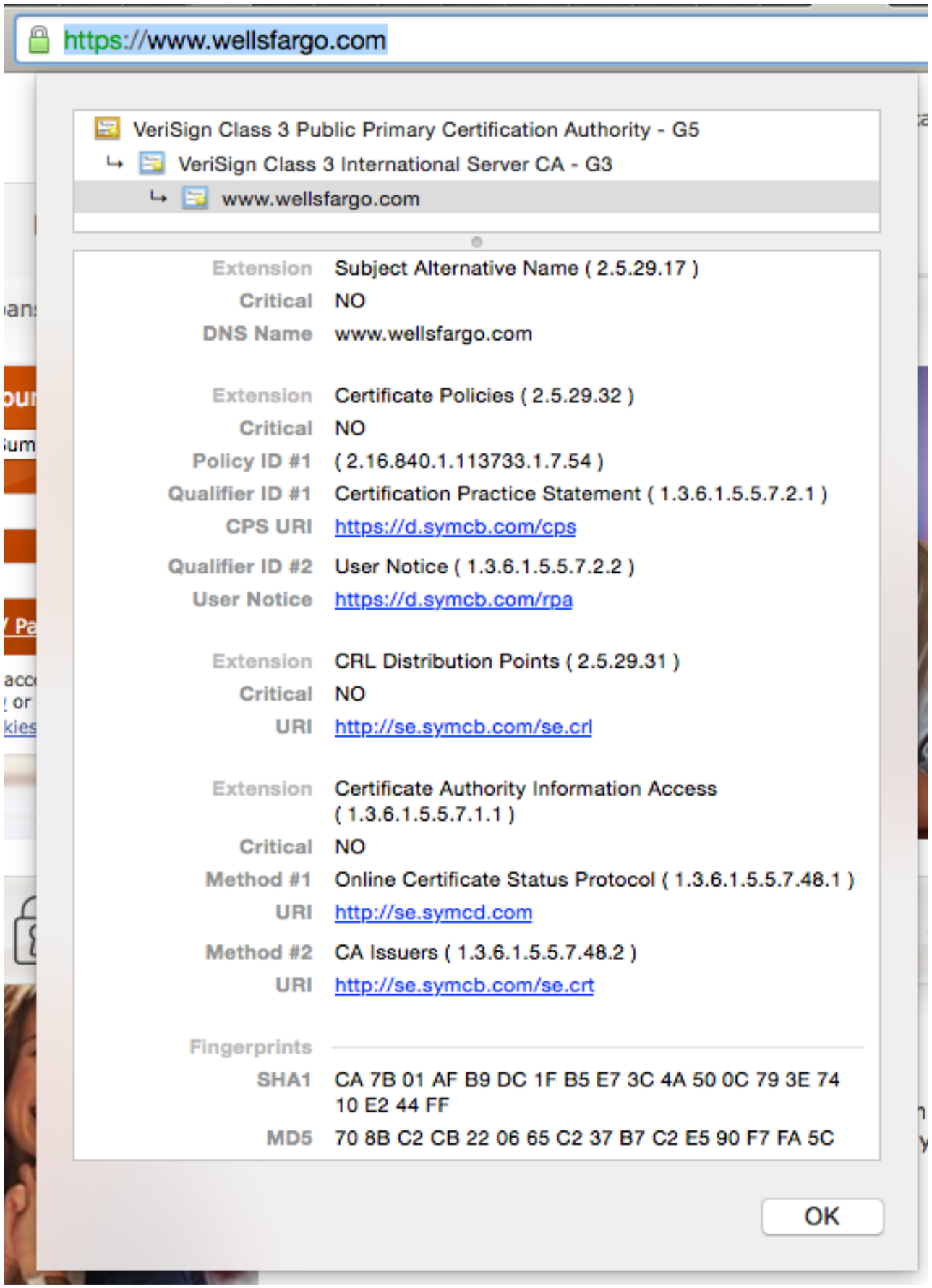
GlobalSign Root CA
↳ GlobalSign CloudSSL CA - SHA256 - G3
↳ incapsula.com

Extension	Subject Alternative Name (2.5.29.17)
Critical	NO
DNS Name	incapsula.com
DNS Name	*.70trades.net
DNS Name	*.acquapanna.com
DNS Name	*.afrique360.com
DNS Name	*.alghanim.com
DNS Name	*.ashbree.com.au
DNS Name	*.atacadao.com.br
DNS Name	*.beautysolutions.com.au
DNS Name	*.biglegalminds.com
DNS Name	*.borsapro.com
DNS Name	*.bsp2012.iata.or.kr
DNS Name	*.cescomarketing.com
DNS Name	*.cgt.it
DNS Name	*.chick-fil-a.com
DNS Name	*.coleccion.caixaforum.com
DNS Name	*.corporateaffairs.tv
DNS Name	*.critical-intelligence.com
DNS Name	*.cultivatenext.org
DNS Name	*.dogchow.ca
DNS Name	*.electronicthecorporatecounsel.com
DNS Name	*.enrich.malaysiaairlines.com
DNS Name	*.exporters.mod.gov.il
DNS Name	*.financikatrade.ae
DNS Name	*.financikatrade.net
DNS Name	*.girona.cat
DNS Name	*.greennghetto.org
DNS Name	*.hatio.com
DNS Name	*.meettherealme.org

OK

Subject Alternate Names:
The *spirit* is that it represents different domain names of the same entity
(google.com, google.co.uk, youtube.com, …)

The *letter* of the rule doesn't say that they need to be the same company—or really have anything in common



Subject Alternate Names:
Other URLs for which this cert should be considered valid. (wellsfargo.com is not the same as www.wellsfargo.com)

Can include wildcards, e.g., *.google.com

CRL & OCSP:
Where to go to check if this certificate has been revoked

Non-cryptographic checksums

Certificate types

Certificates can be classified in two broad ways

What the certificate can be used for

Signing (root and intermediate certs)

Encrypting (leaf certs)

The type of vetting process used

DV (Domain validation)
Prove administrative access to the domain, e.g., by uploading a file

OV (Organization validation)
Prove ownership of the organization that owns the domain

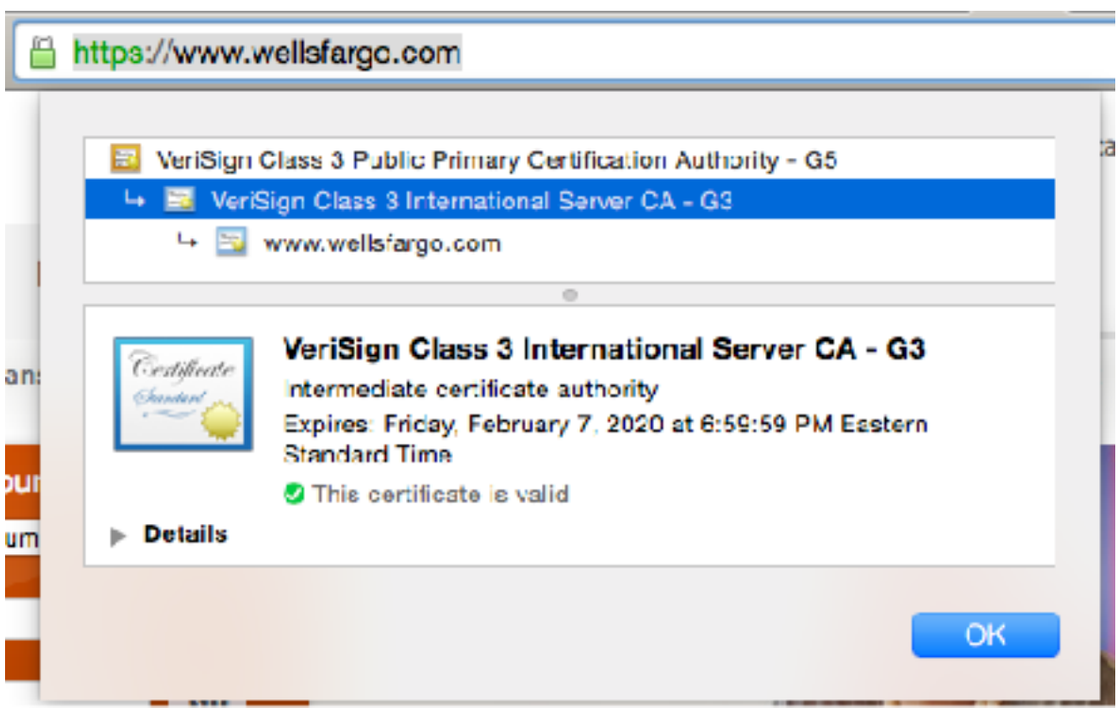
EV (Extended validation)
More extensive validation ($$)

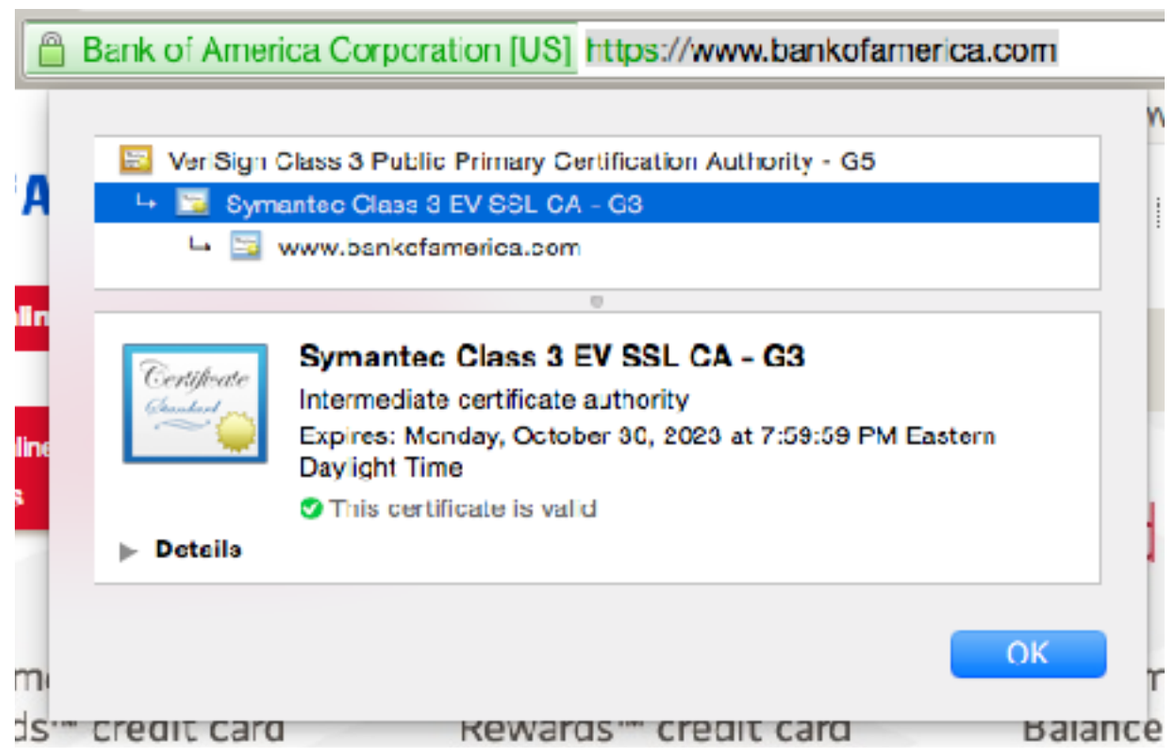
Certificate types

Why are these different?

Certificate types

Why are these different?





This is an EV (extended validation) certificate; browsers show the full name for these kinds of certs

Proper reaction to Heartbleed

1. Patch the software

2. “Reissue” a new key (get a new one and load it onto your servers)

3. Revoke the old key

Proper reaction to Heartbleed

1. Patch the software

2. “Reissue” a new key (get a new one and load it onto your servers)

3. Revoke the old key

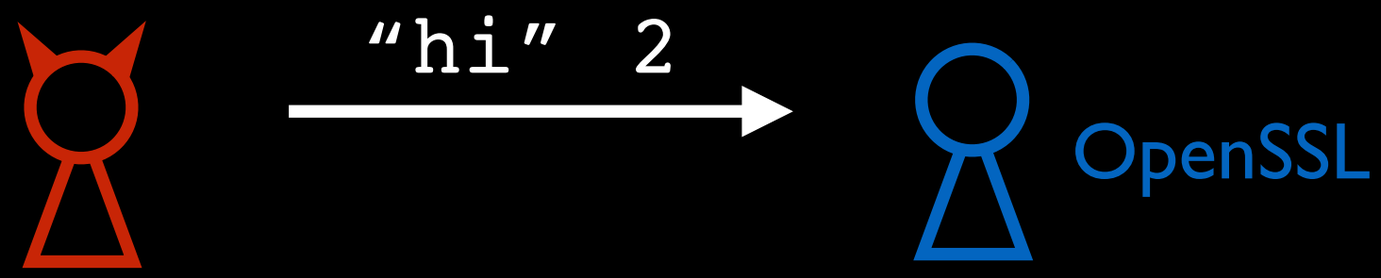
<u>Order matters!</u>
If we reissued and then patched,
then our new key would be compromised, too.

If we revoked first, we’d be offline.

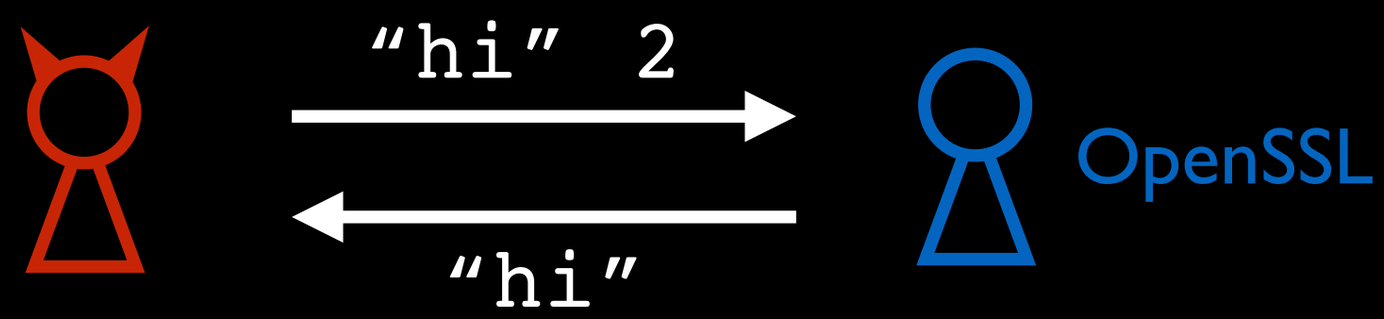
Heartbleed

OpenSSL

Heartbleed



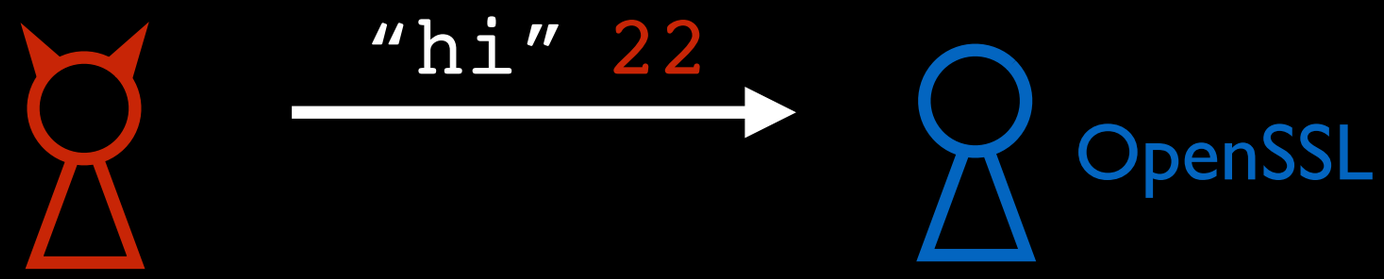
Heartbleed



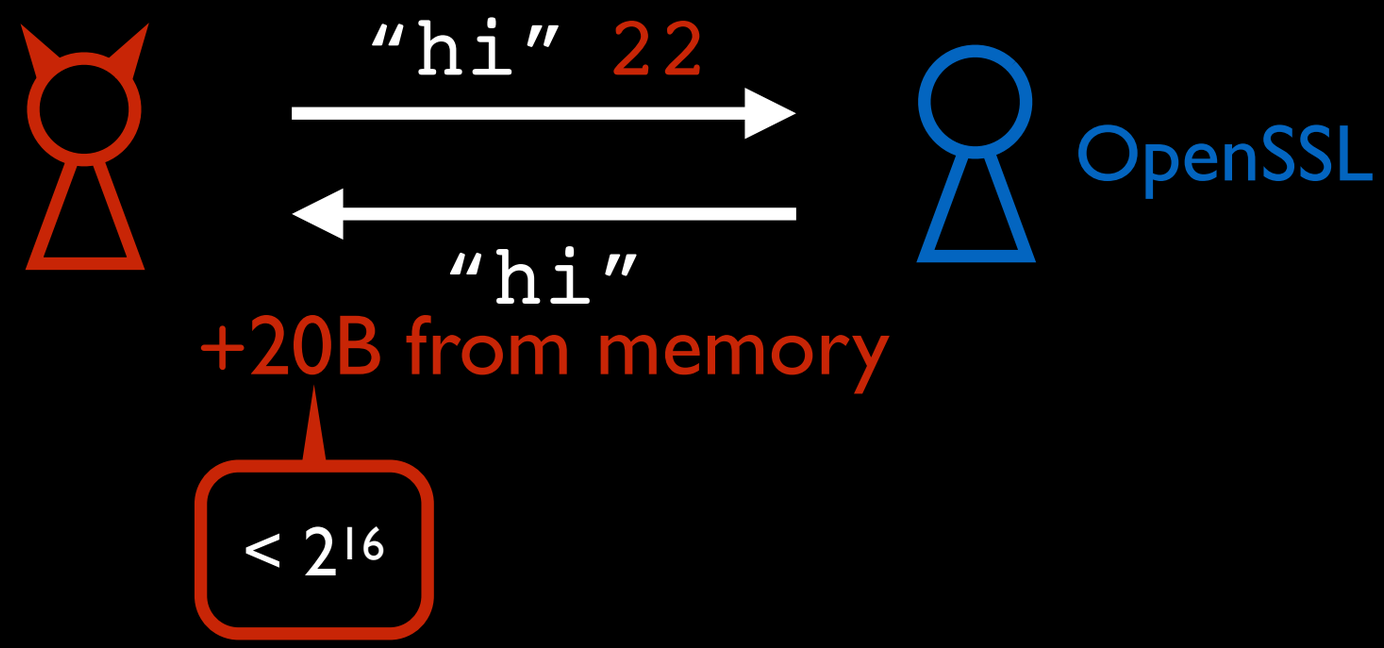
Heartbleed

OpenSSL

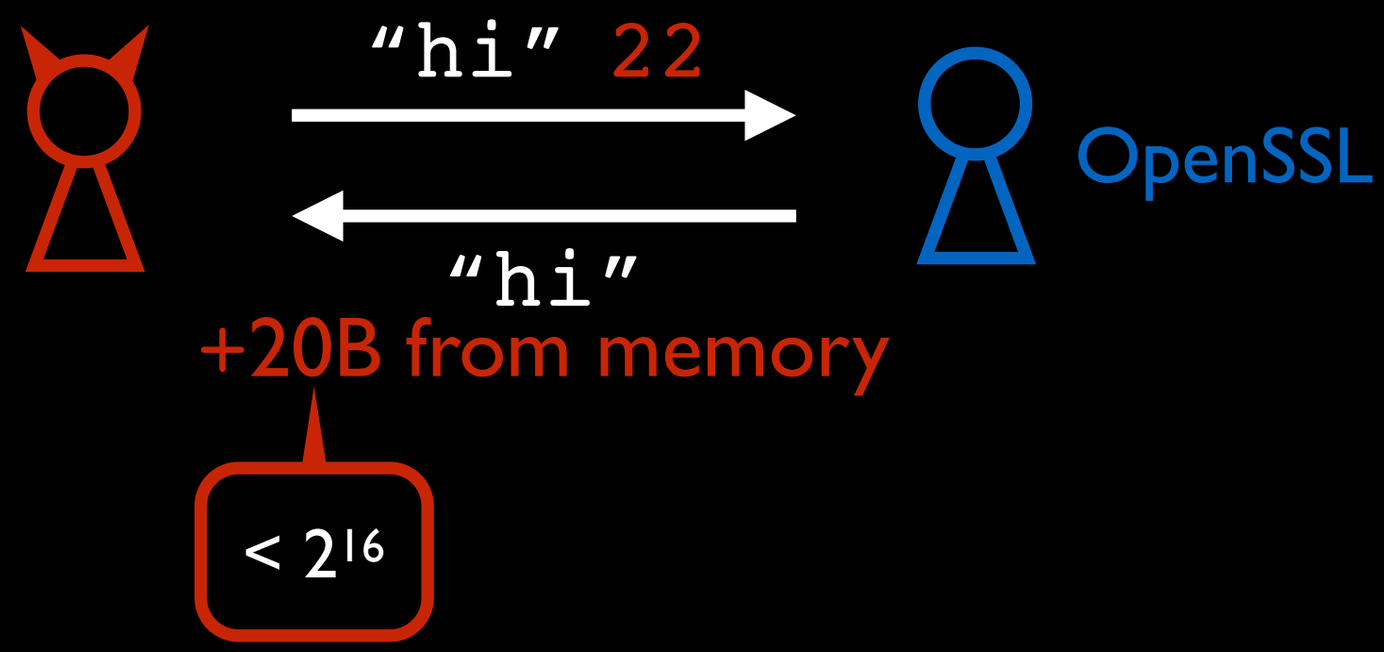
Heartbleed



Heartbleed



Heartbleed

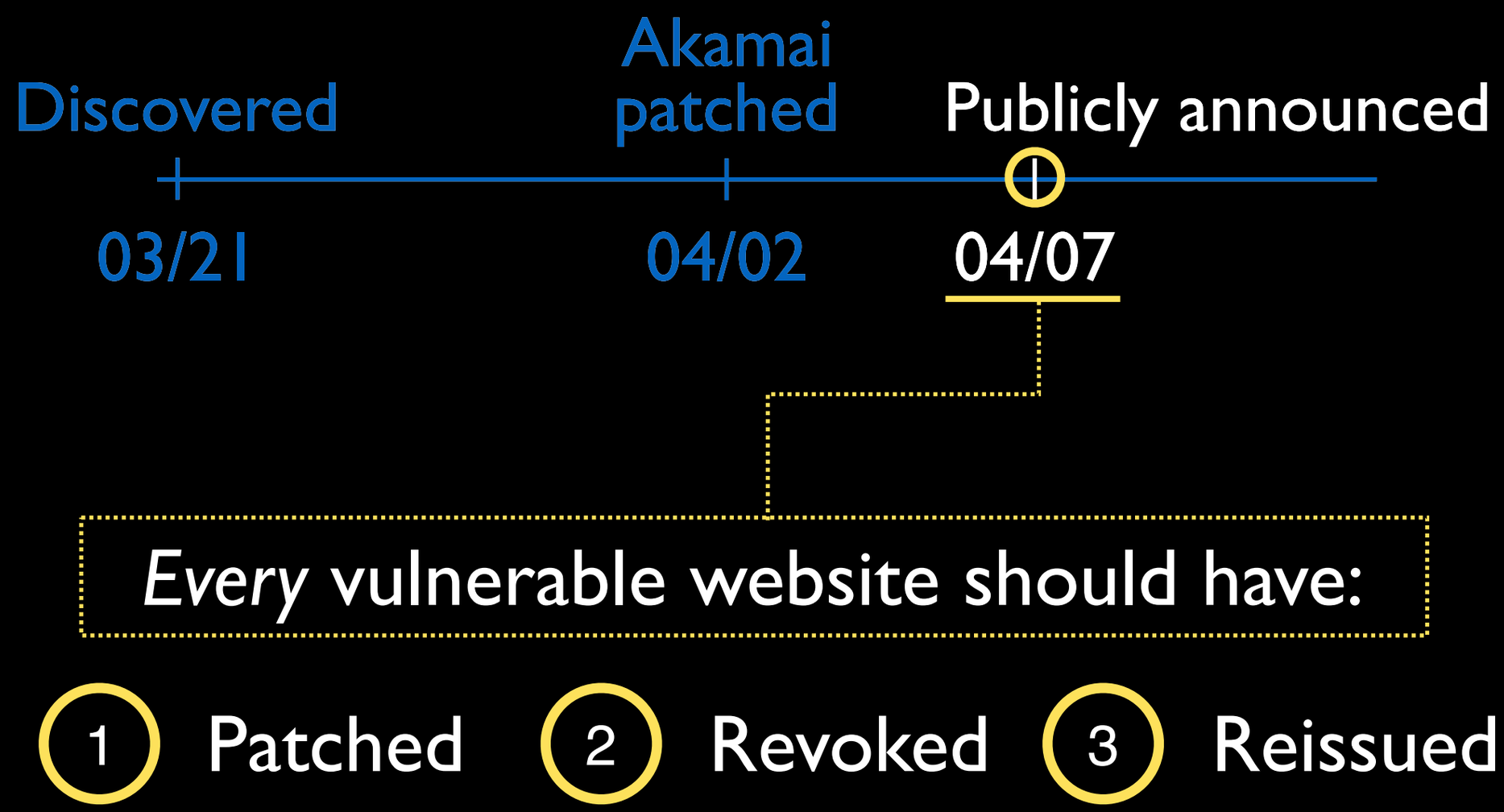


Potentially reveals user data and private keys

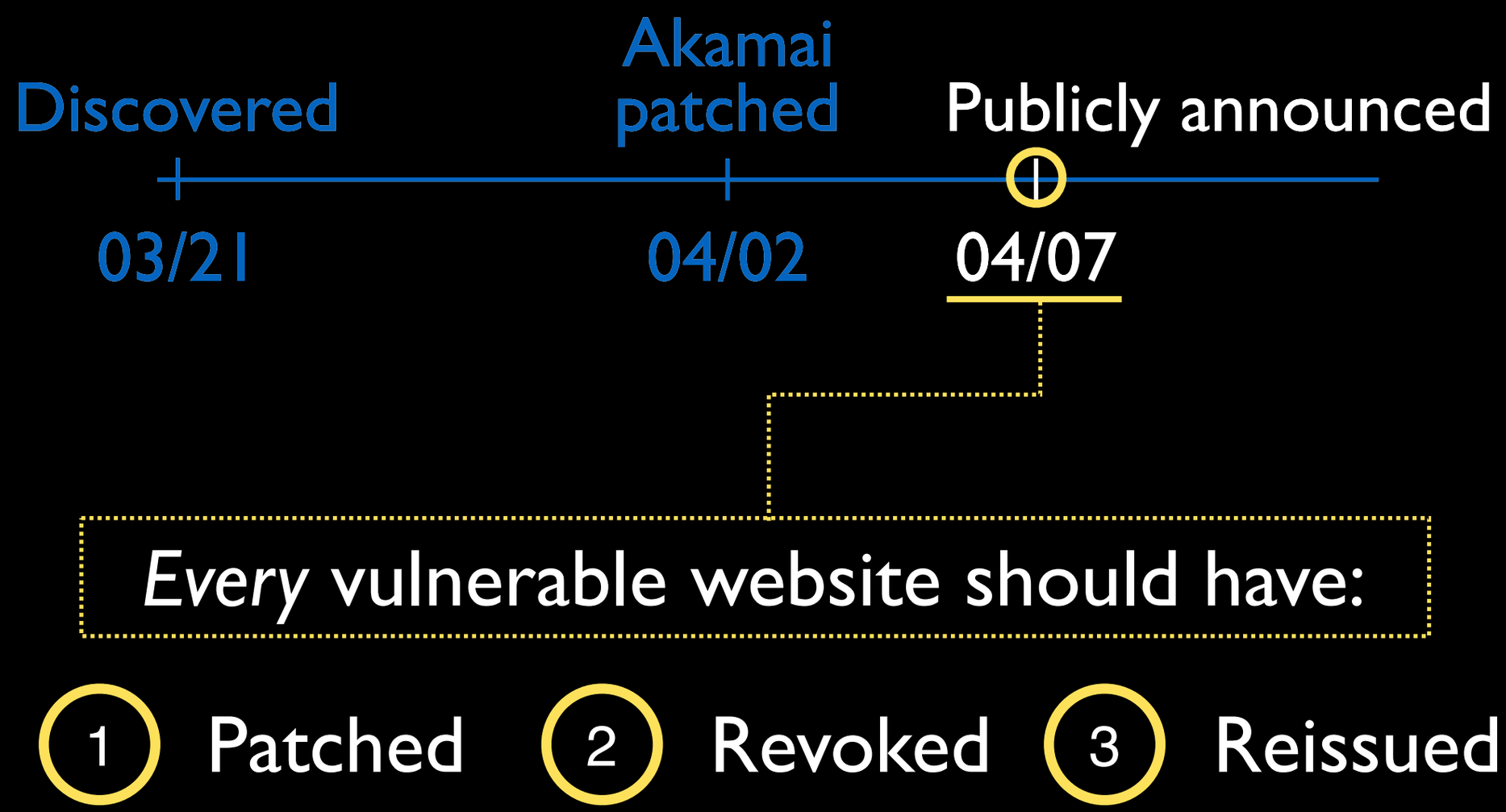
Heartbleed exploits were undetectable

Why study Heartbleed?

Why study Heartbleed?



Why study Heartbleed?



Heartbleed is a natural experiment:

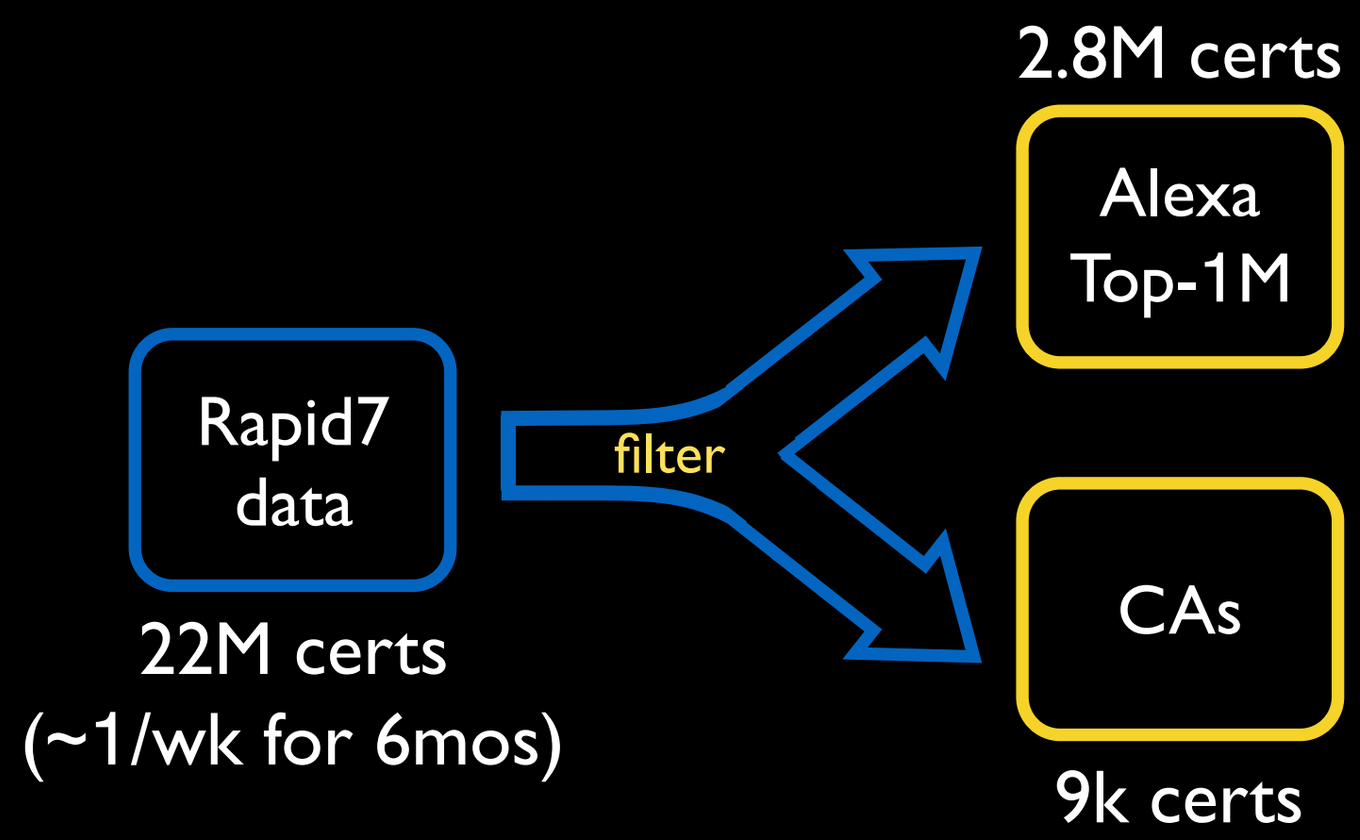
How quickly and thoroughly do administrators act?

Dataset

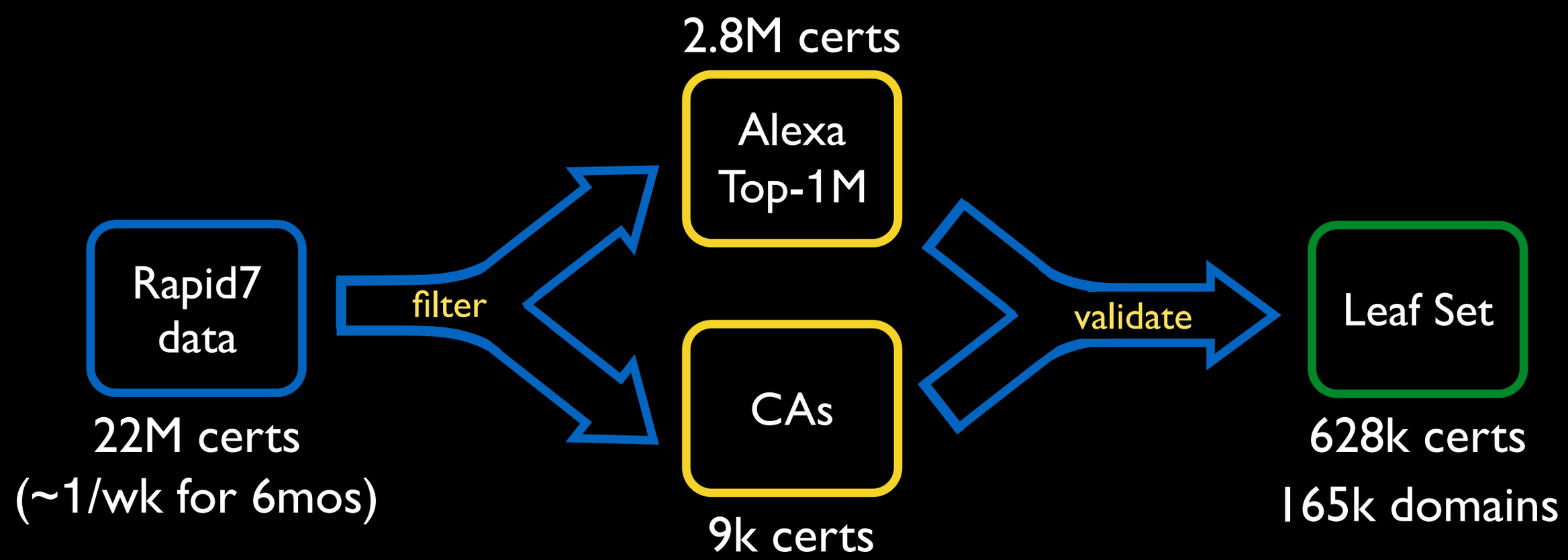
Rapid7
data

22M certs
(~1/wk for 6mos)

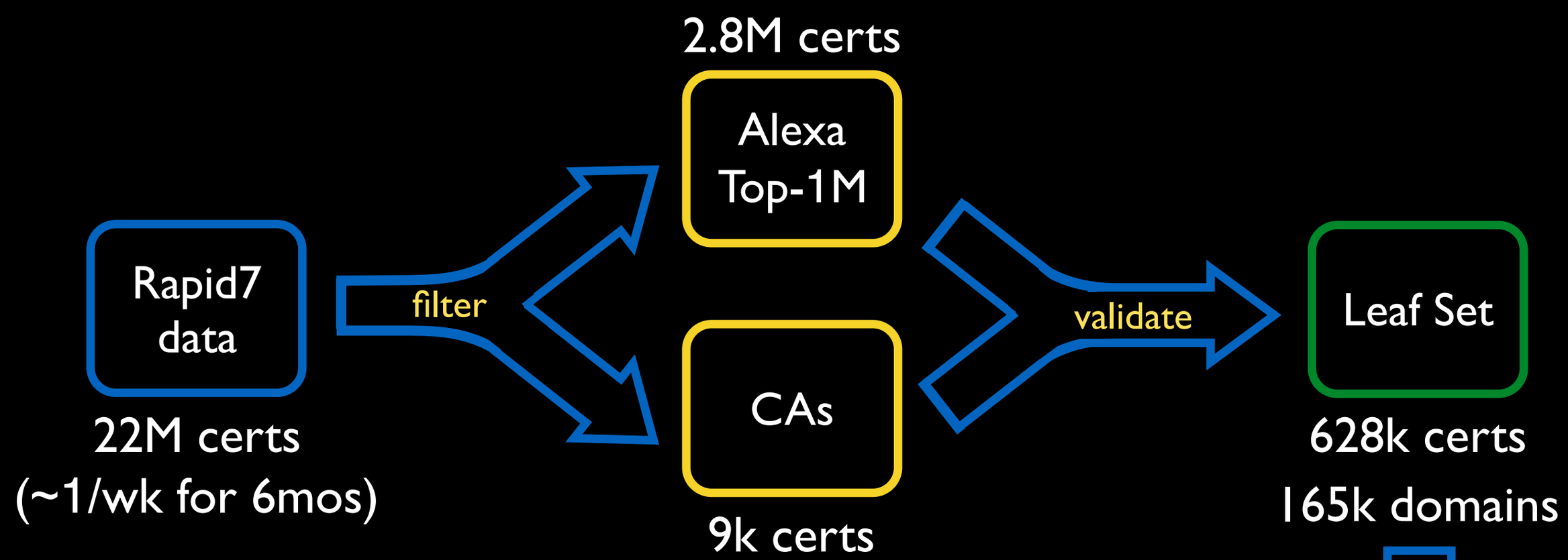
Dataset


2.8M certs
Alexa
Top-1M
Rapid7
data
filter
CAs
22M certs
(~1/wk for 6mos)
9k certs

Dataset

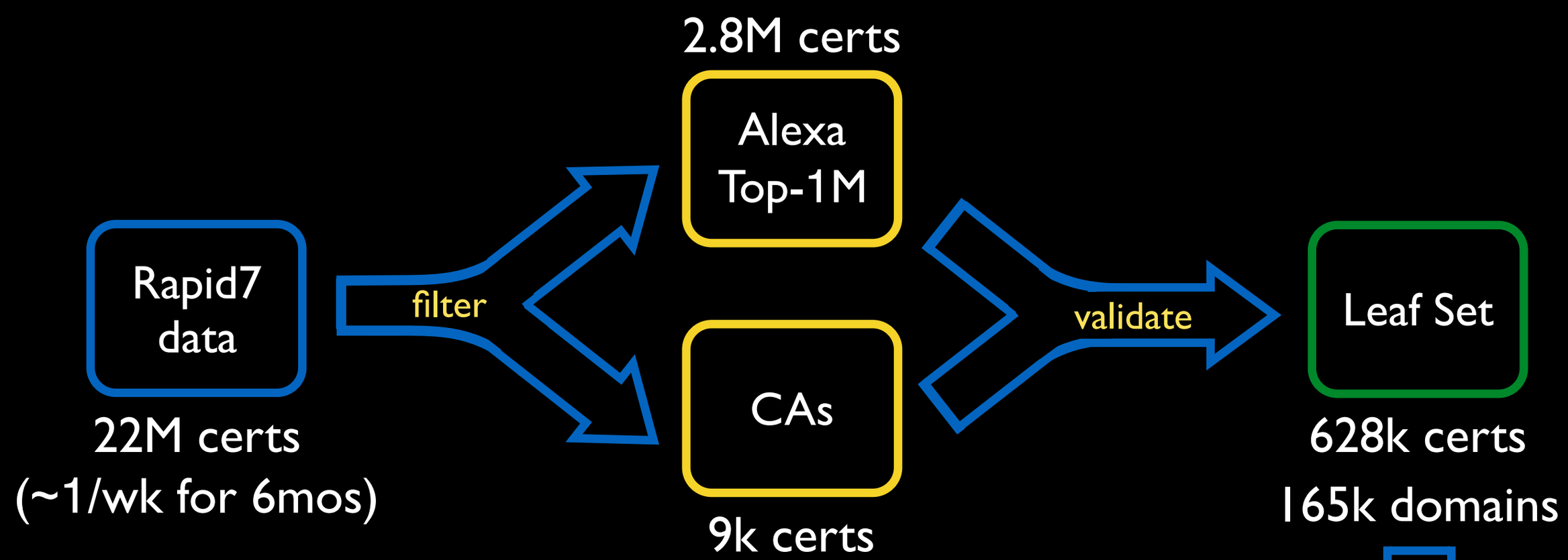

2.8M certs
Alexa
Top-1M
Rapid7
data
filter
CAs
validate
Leaf Set
22M certs
(~1/wk for 6mos)
9k certs
628k certs
165k domains

Dataset



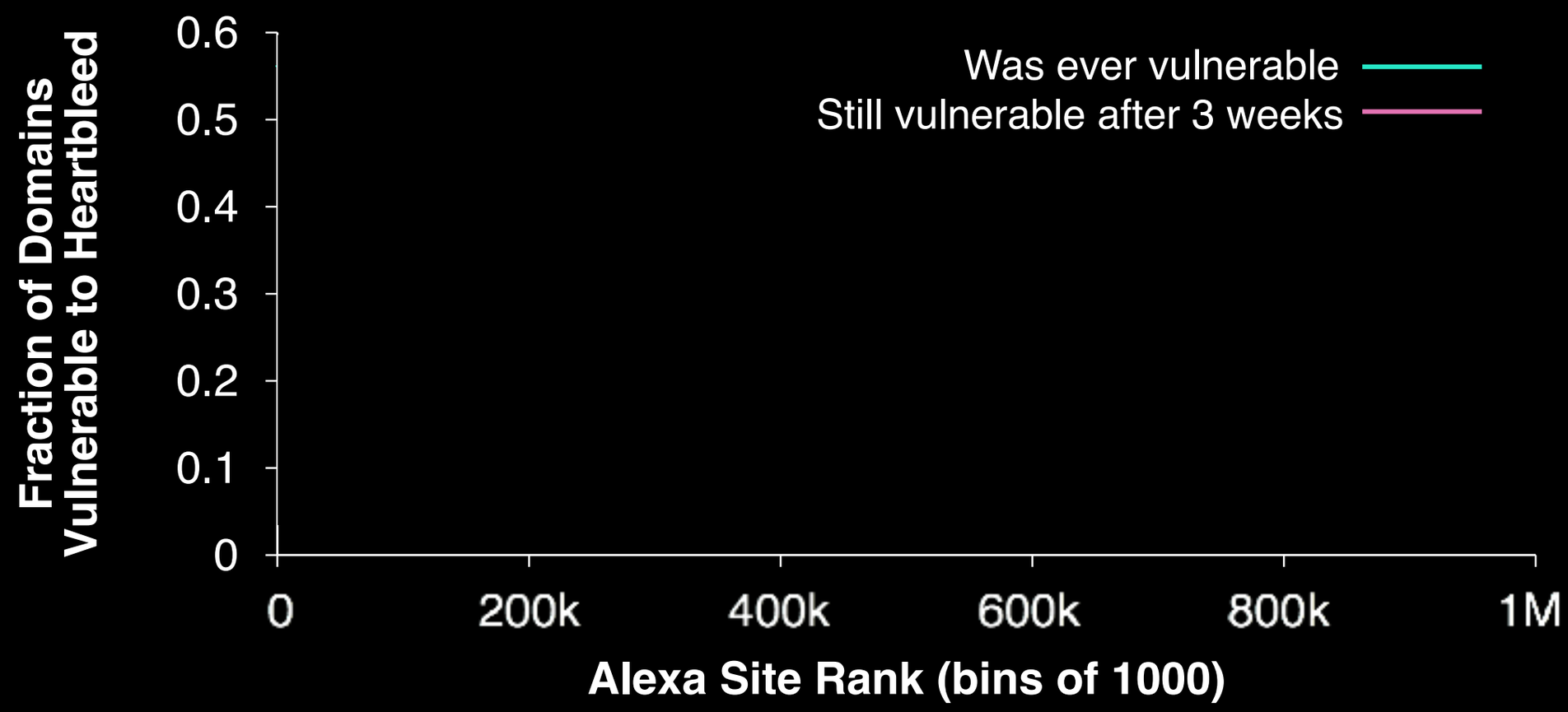
- Download CRLs
- Detect vulnerability
- Identify *Heartbleed-induced* reissues & revocations

Dataset

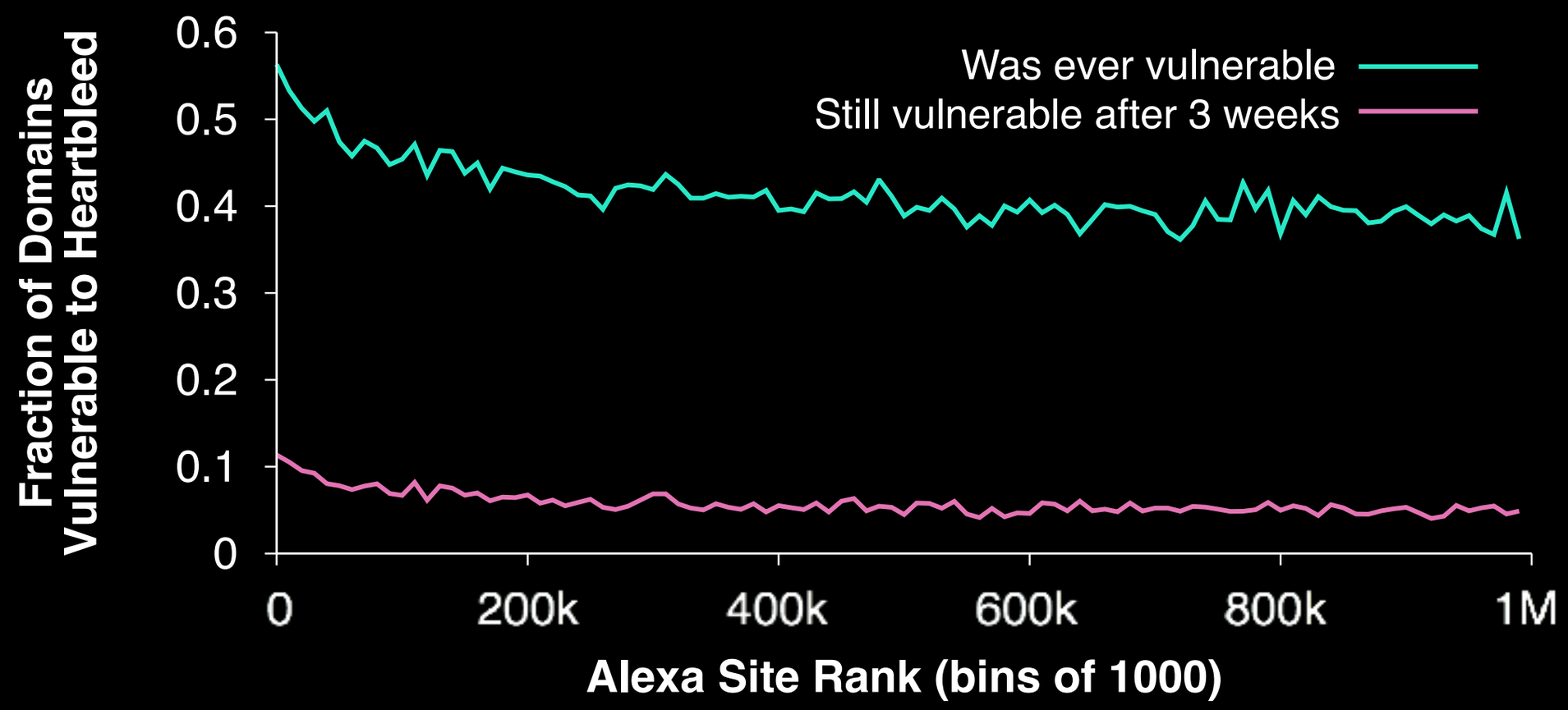


- Download CRLs
- Detect vulnerability
- Identify *Heartbleed-induced* reissues & revocations

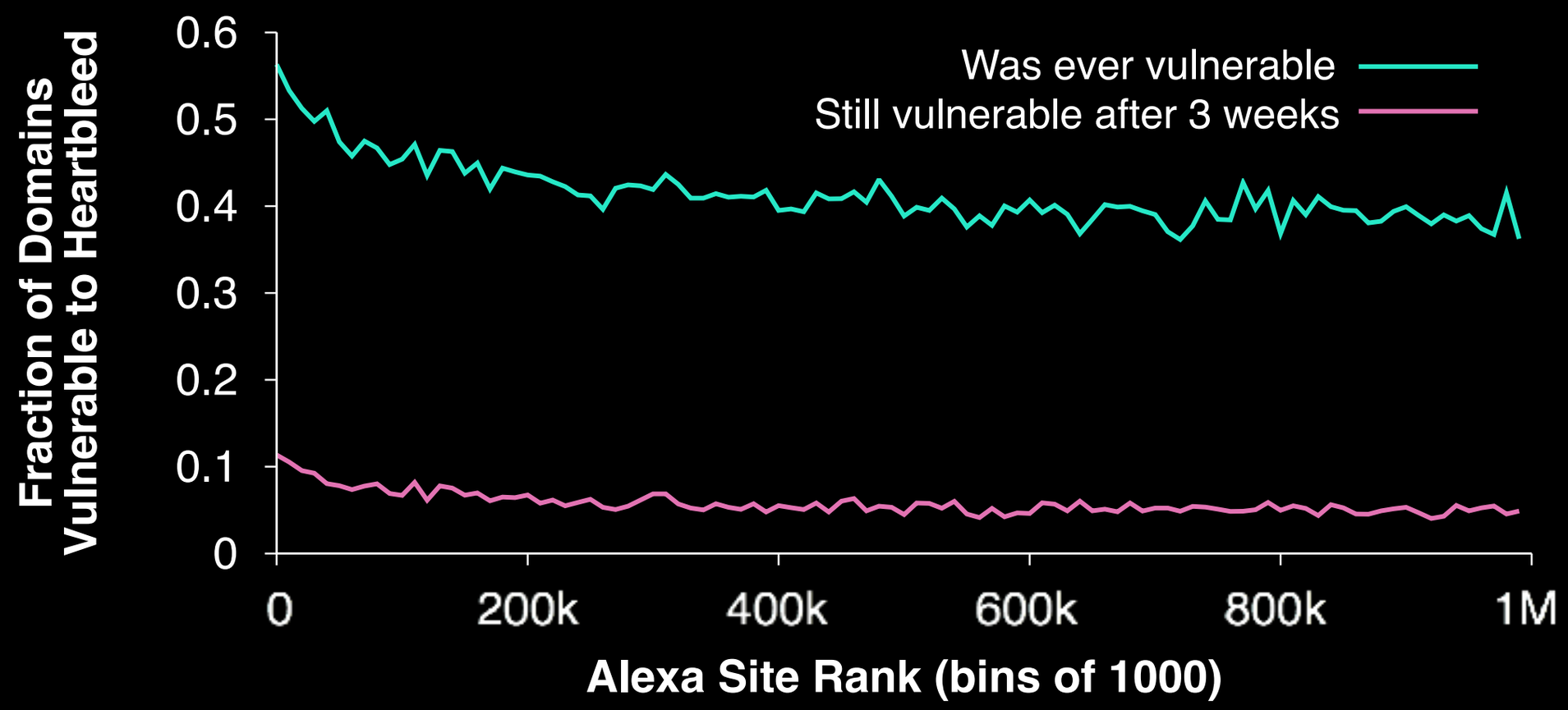
Prevalence and patch rates



Prevalence and patch rates



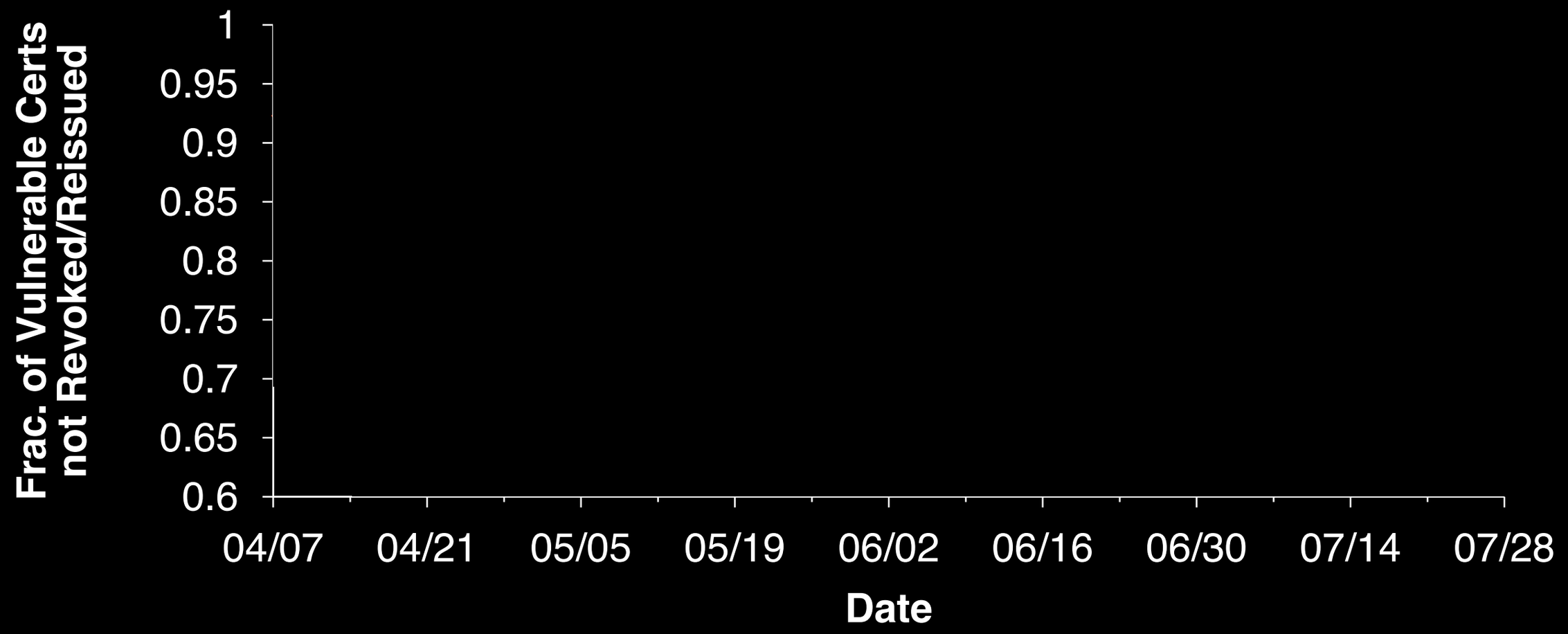
Prevalence and patch rates



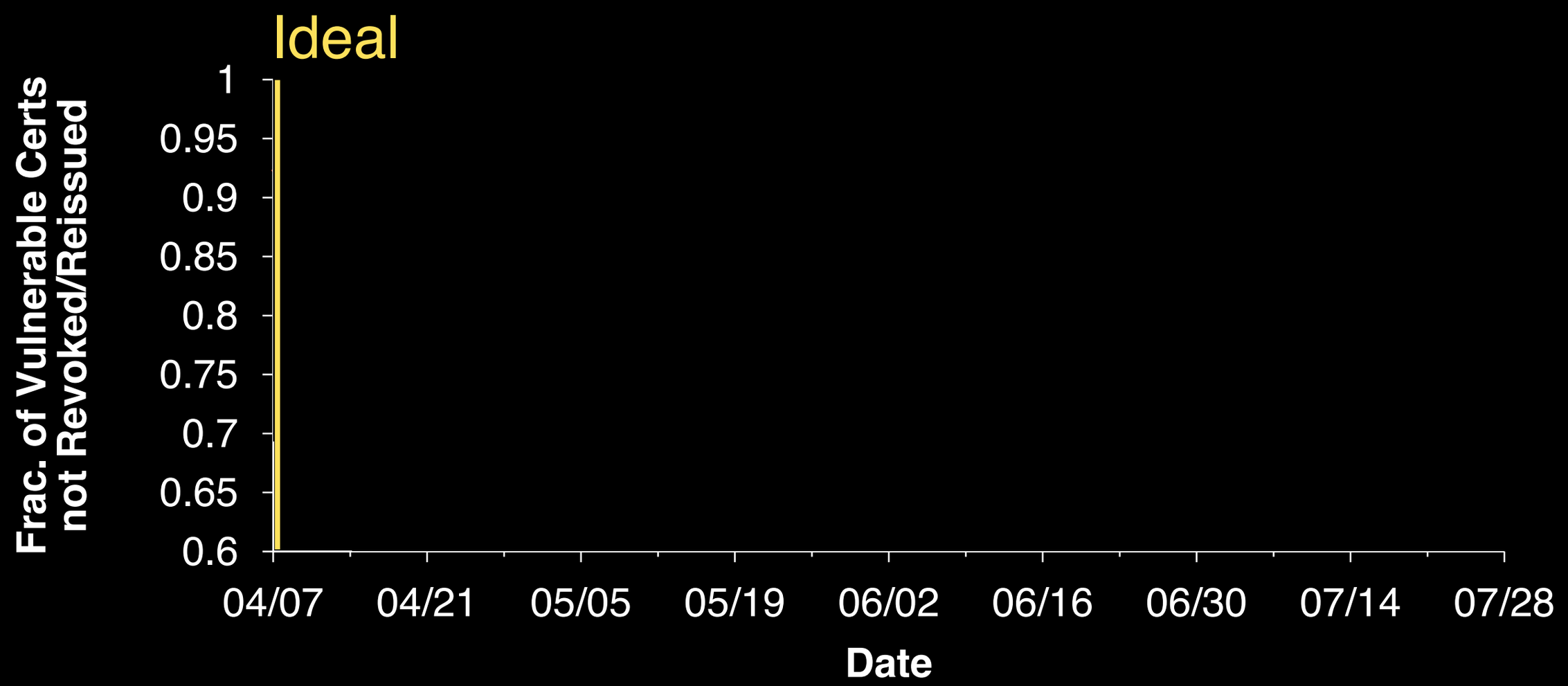
Patching rates are mostly positive

Only ~7% had not patched within 3 weeks

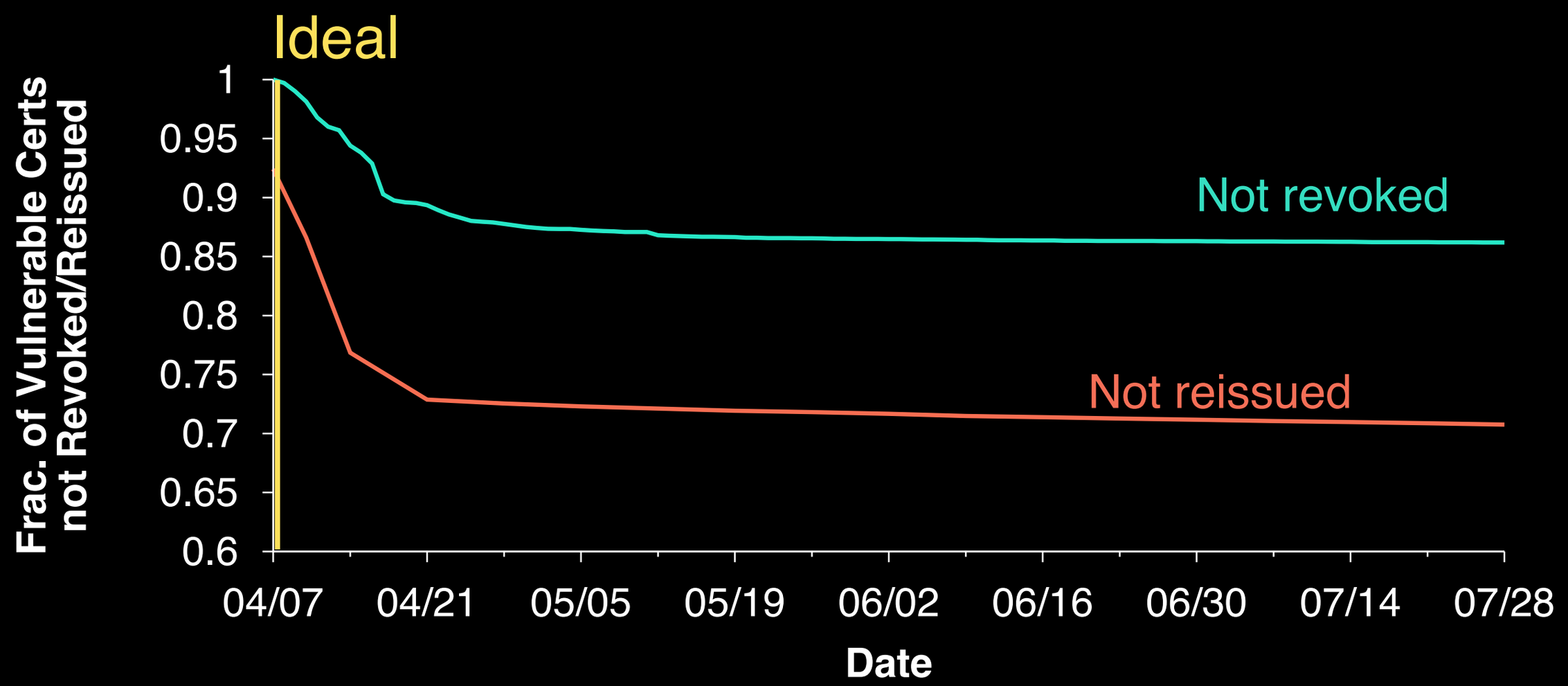
Certificate update rates



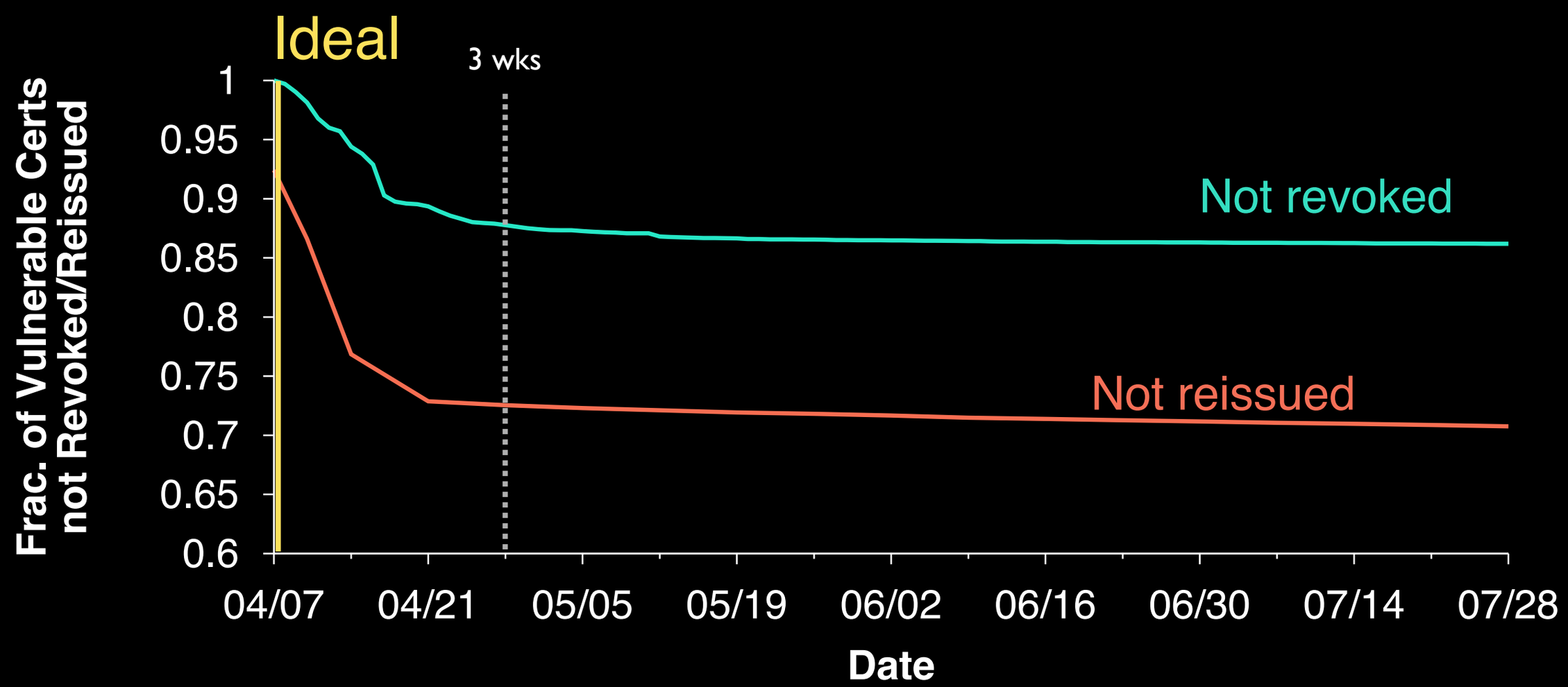
Certificate update rates



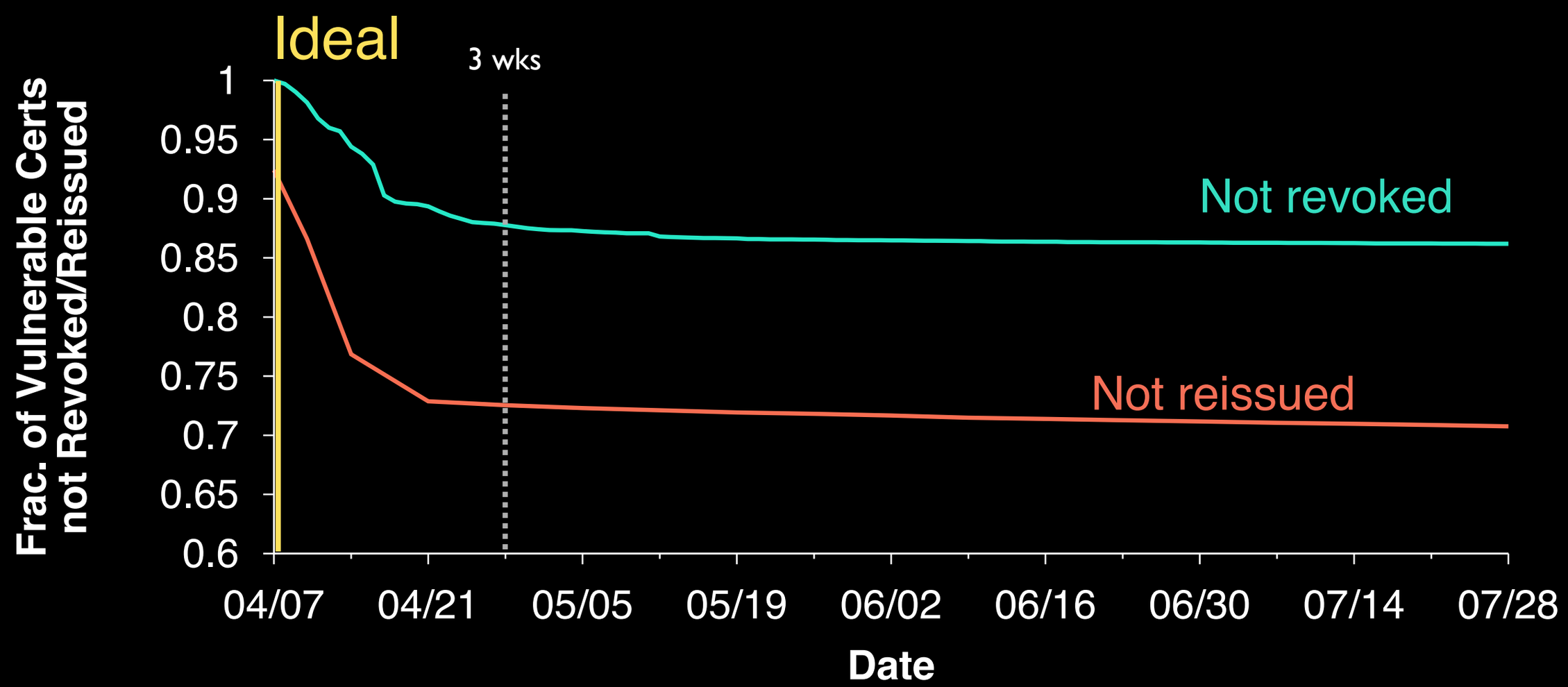
Certificate update rates



Certificate update rates



Certificate update rates

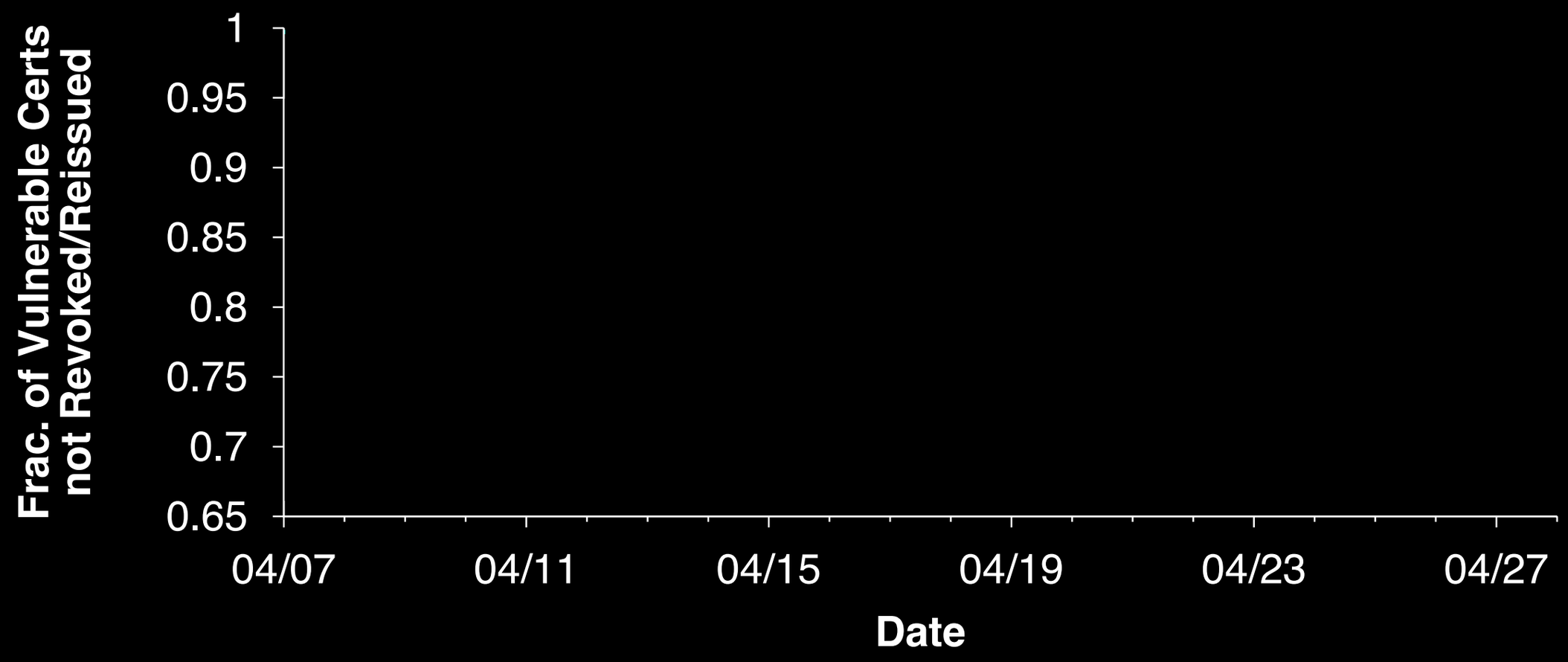


Similar pattern to patches:

Exponential drop-off, then levels out

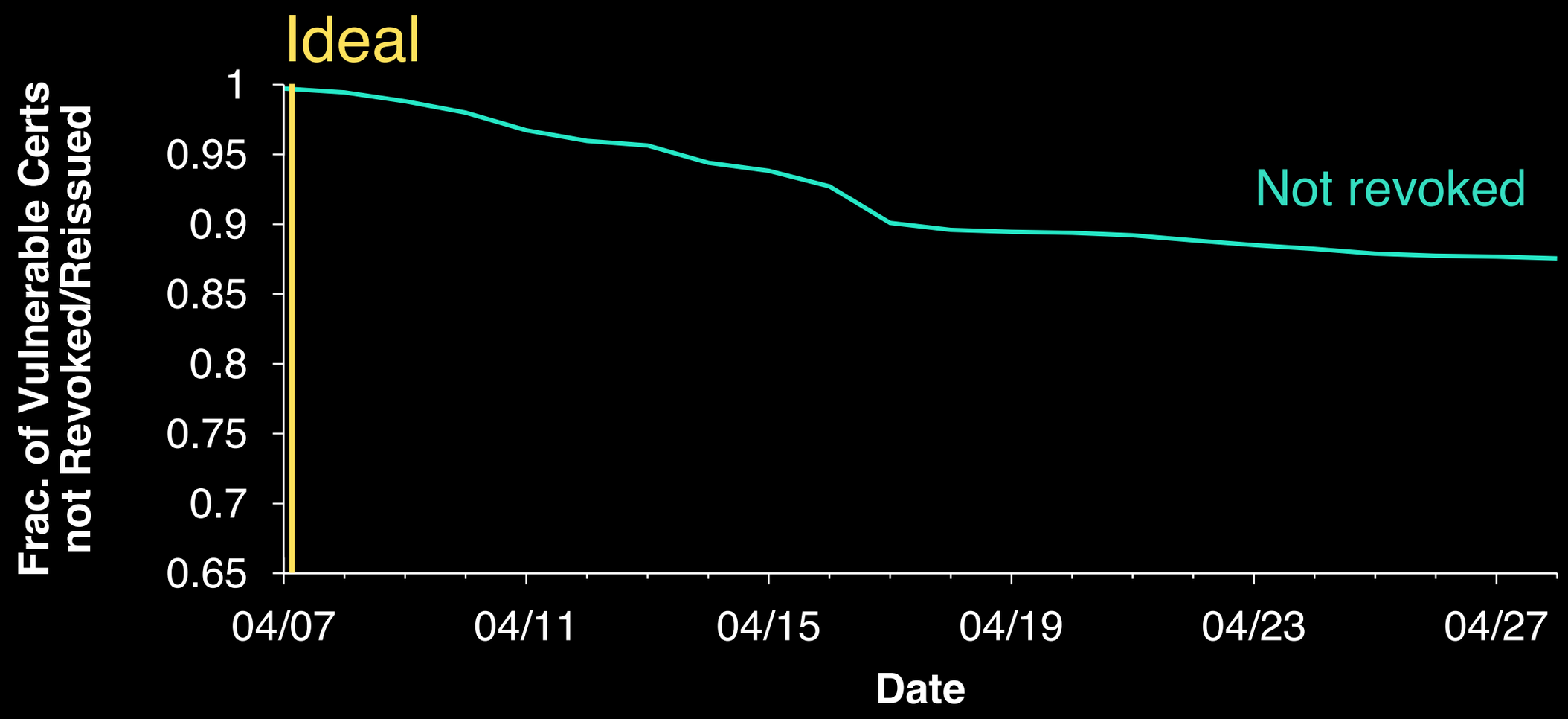
After 3 weeks: 13% Revoked

Certificate update rates

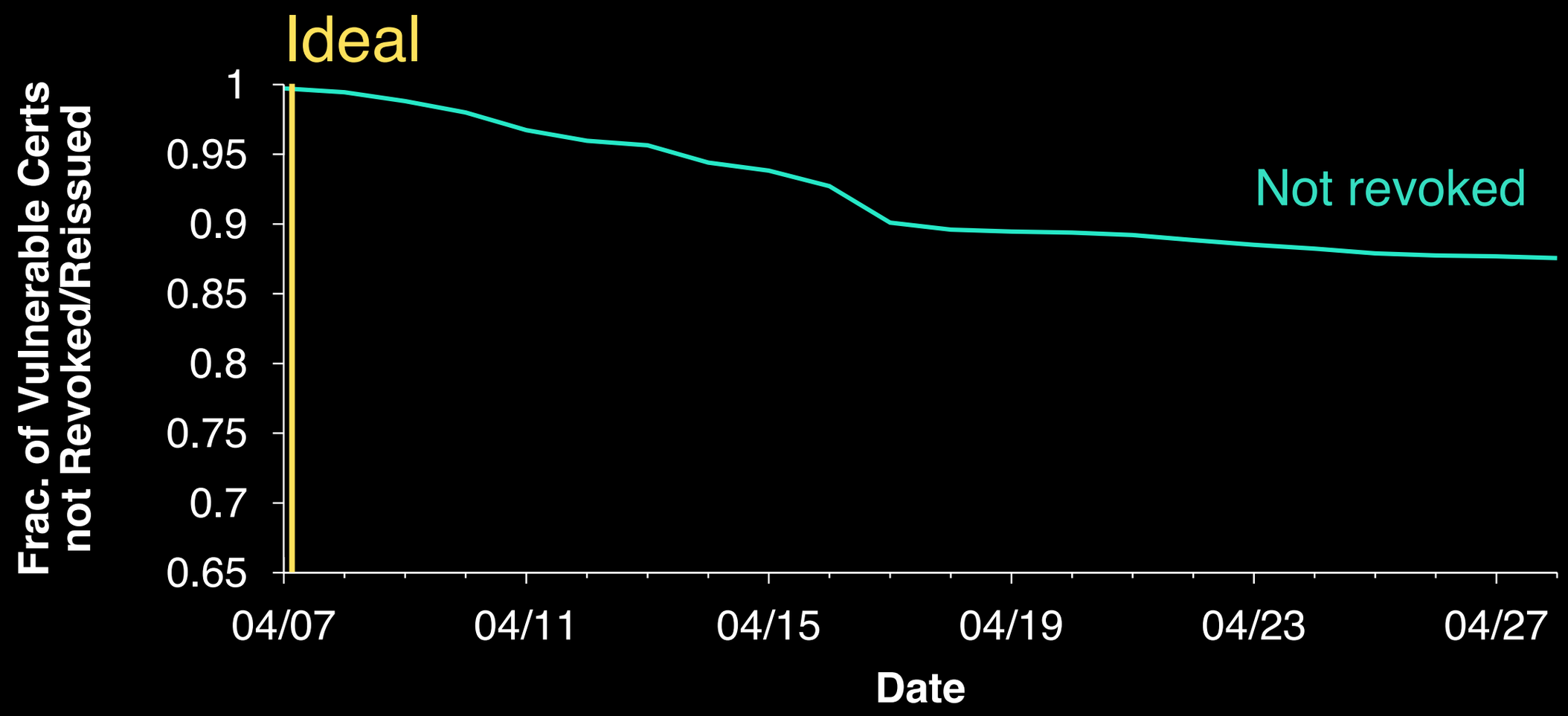


Certificate update rates

Certificate update rates



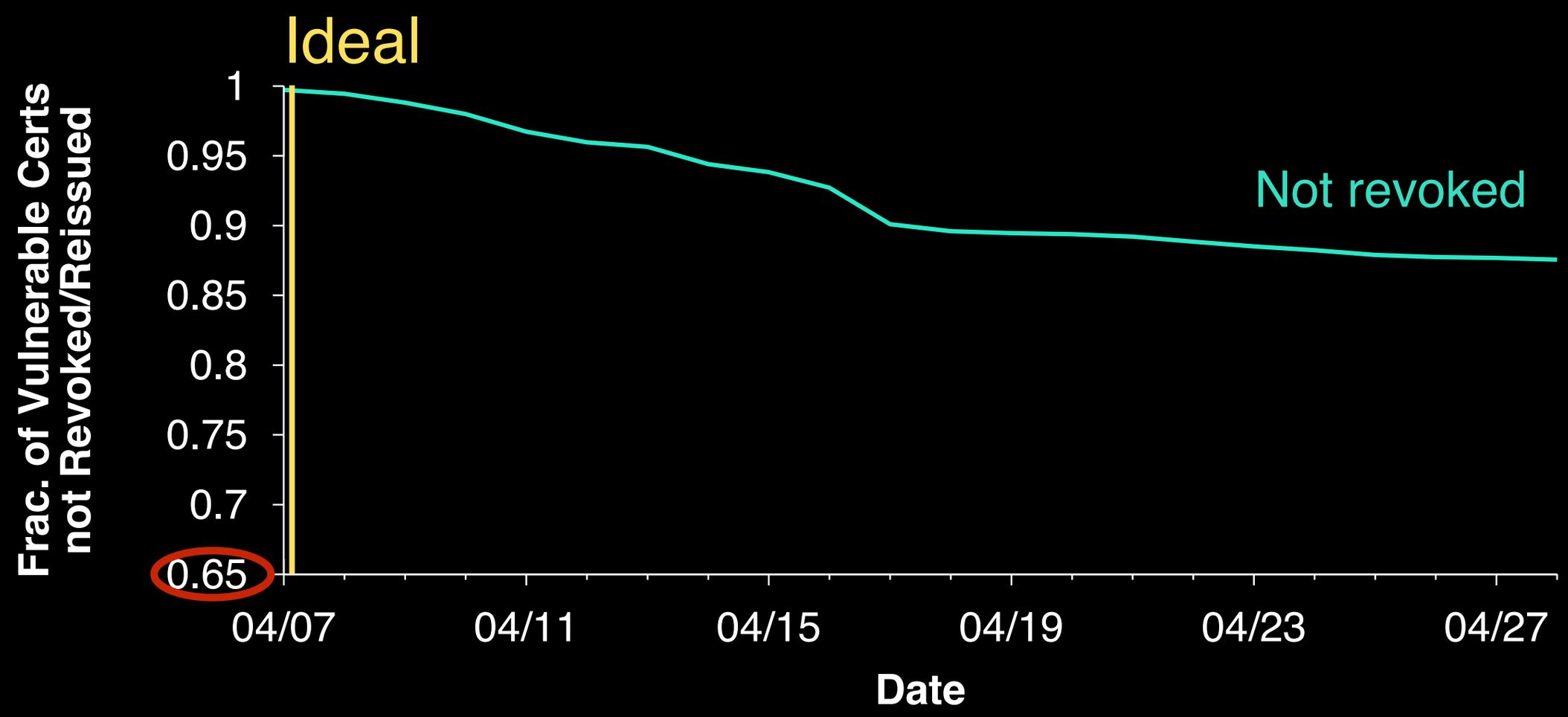
Certificate update rates



Similar pattern to patches:

Exponential drop-off, then levels out

Certificate update rates

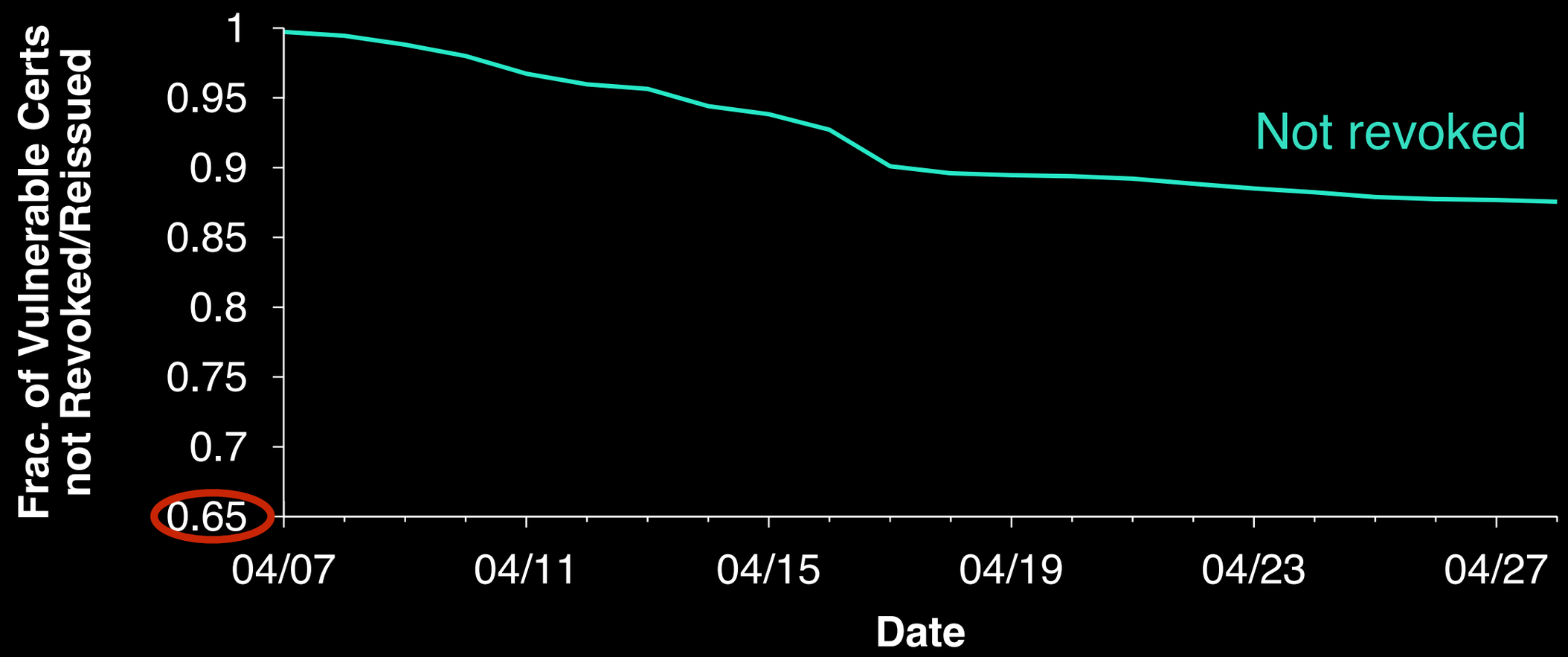


Similar pattern to patches:

Exponential drop-off, then levels out

After 3 weeks: 13% Revoked

Certificate update rates

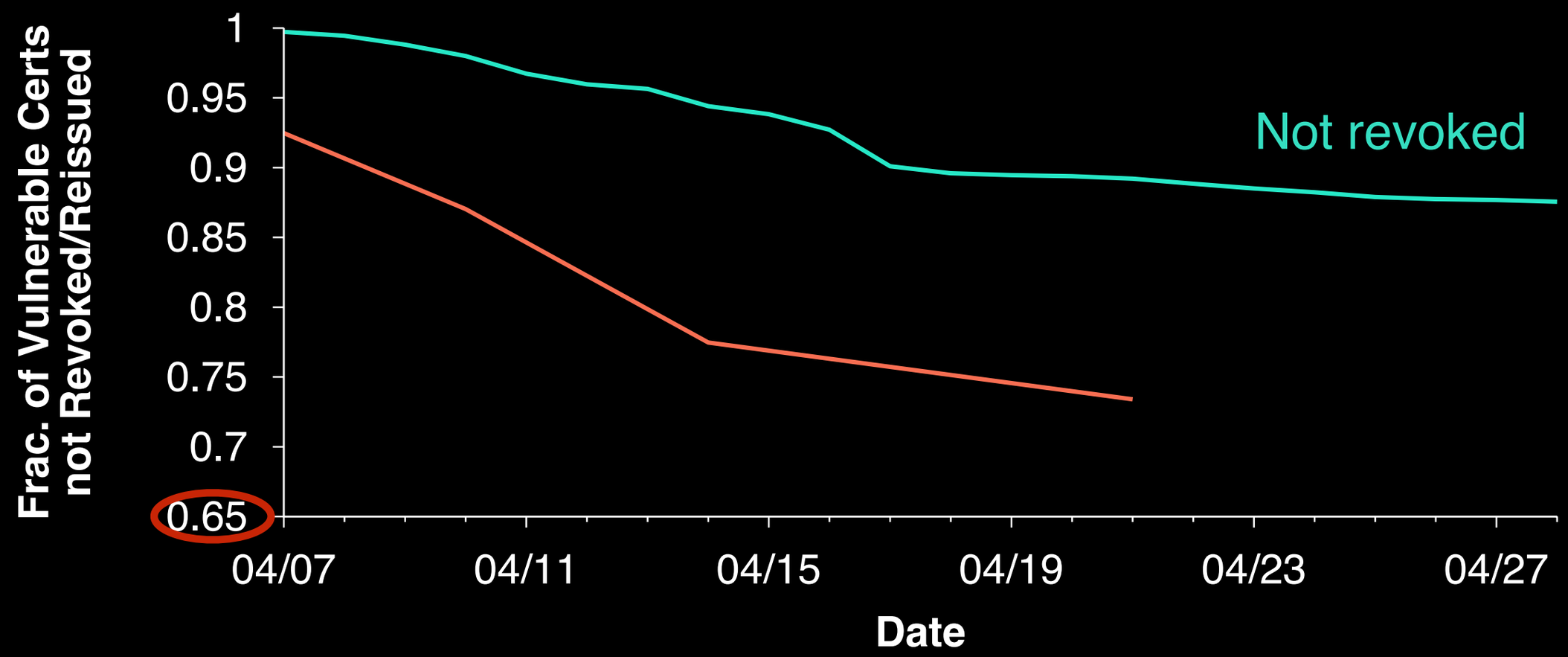


Similar pattern to patches:

Exponential drop-off, then levels out

After 3 weeks: 13% Revoked

Certificate update rates

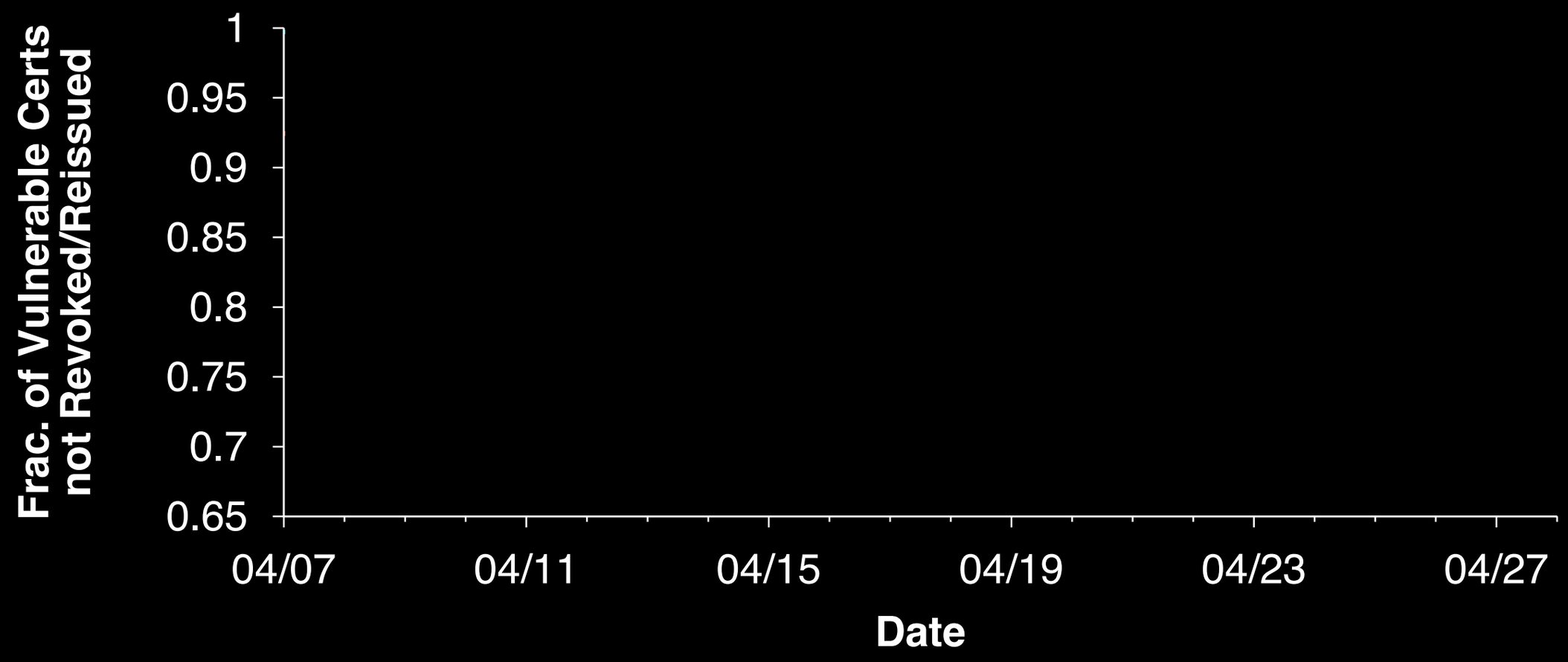


Similar pattern to patches:

Exponential drop-off, then levels out

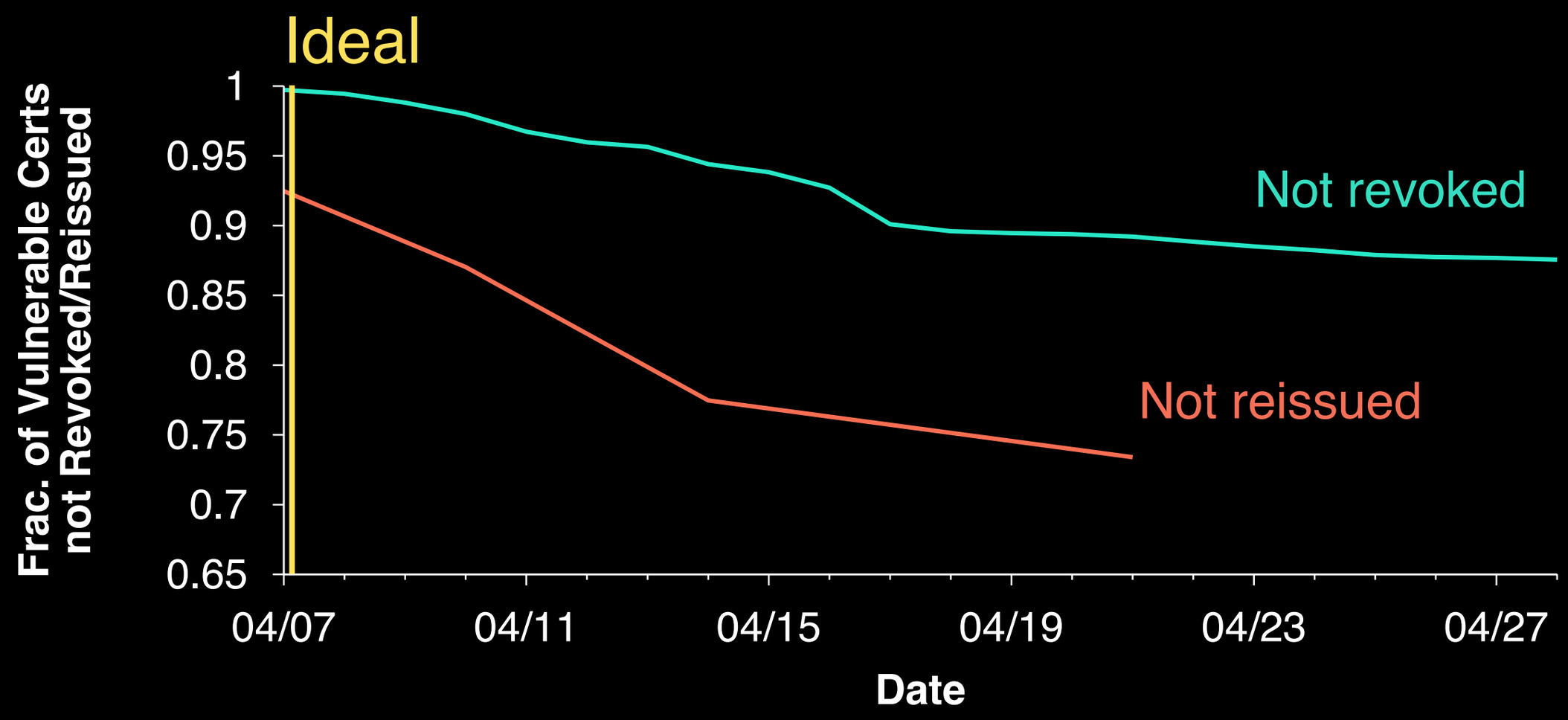
After 3 weeks: 13% Revoked

Certificate update rates

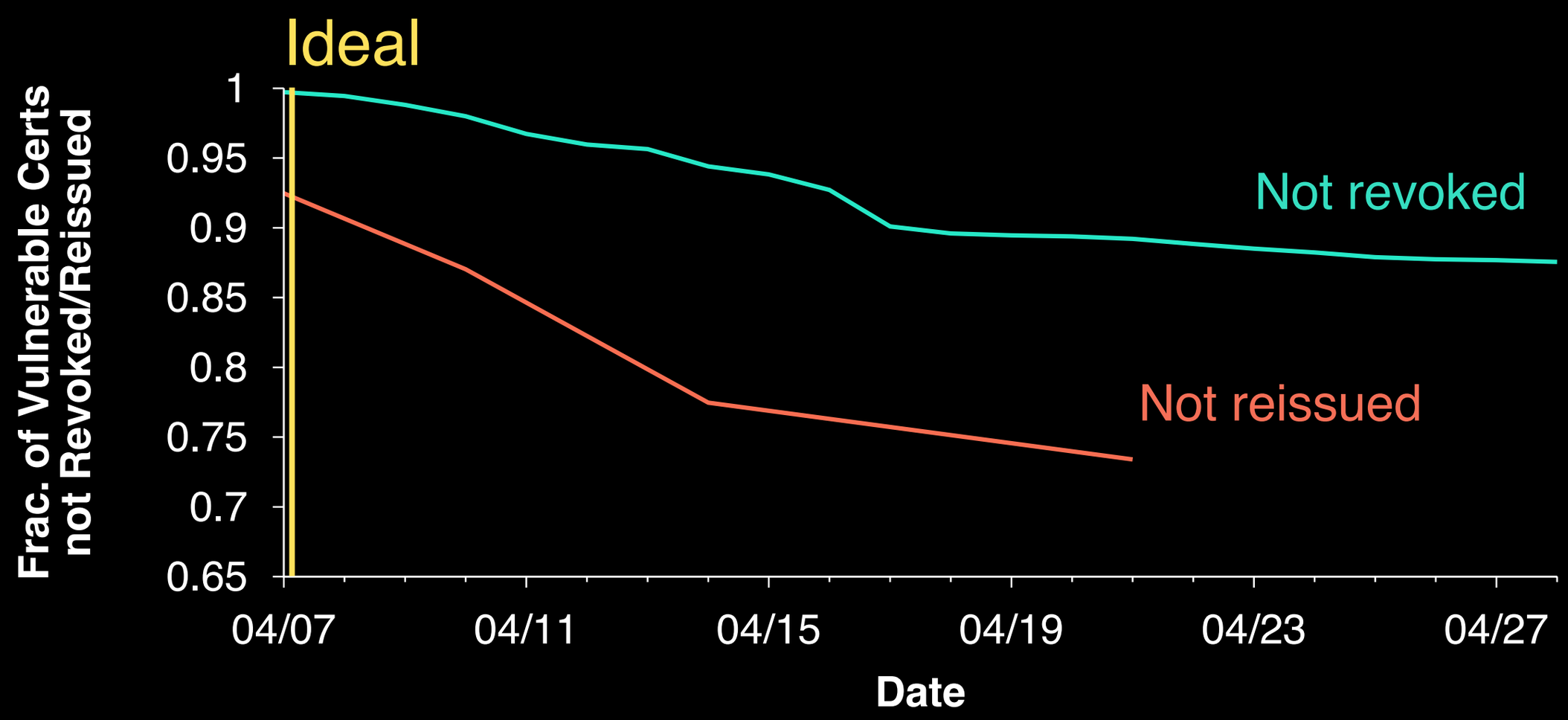


Certificate update rates

Certificate update rates



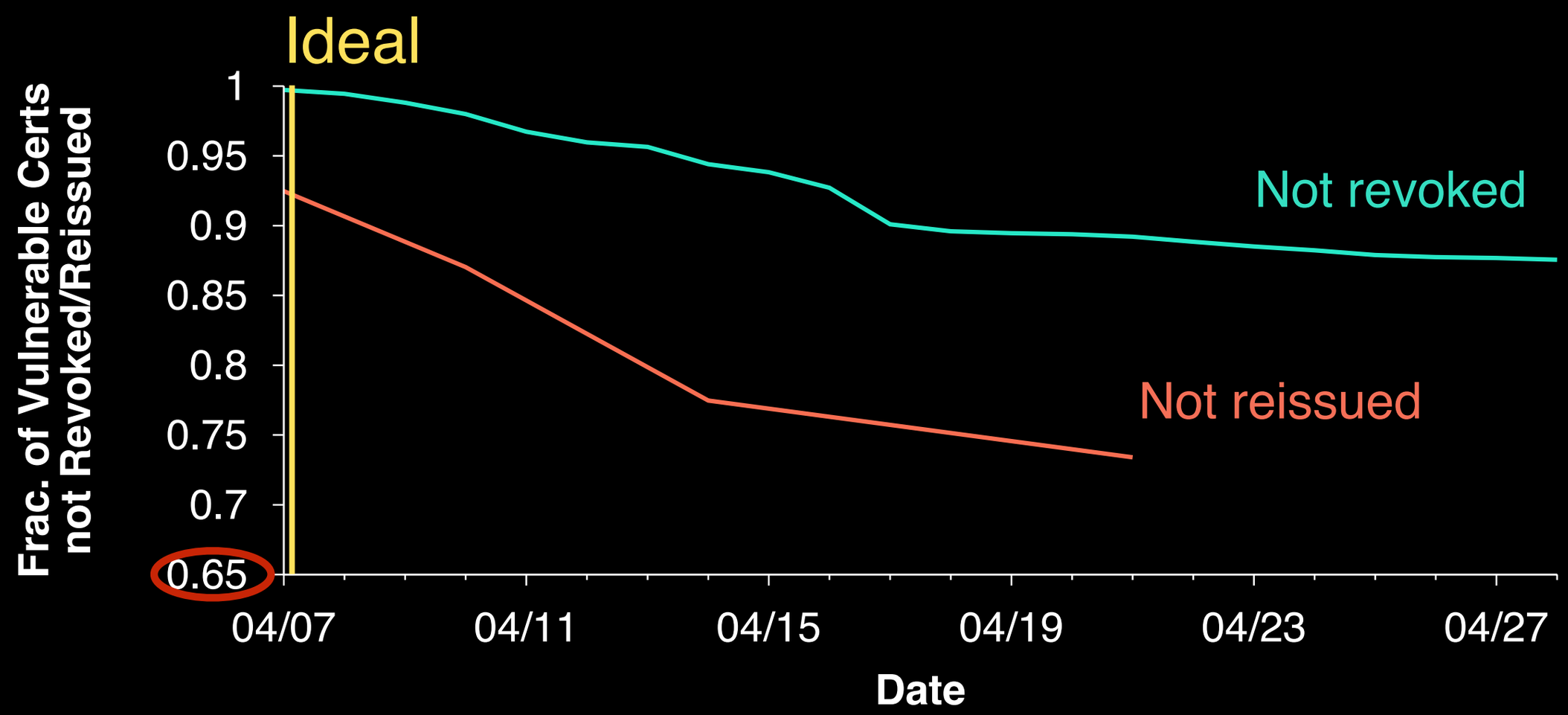
Certificate update rates



Similar pattern to patches:

Exponential drop-off, then levels out

Certificate update rates

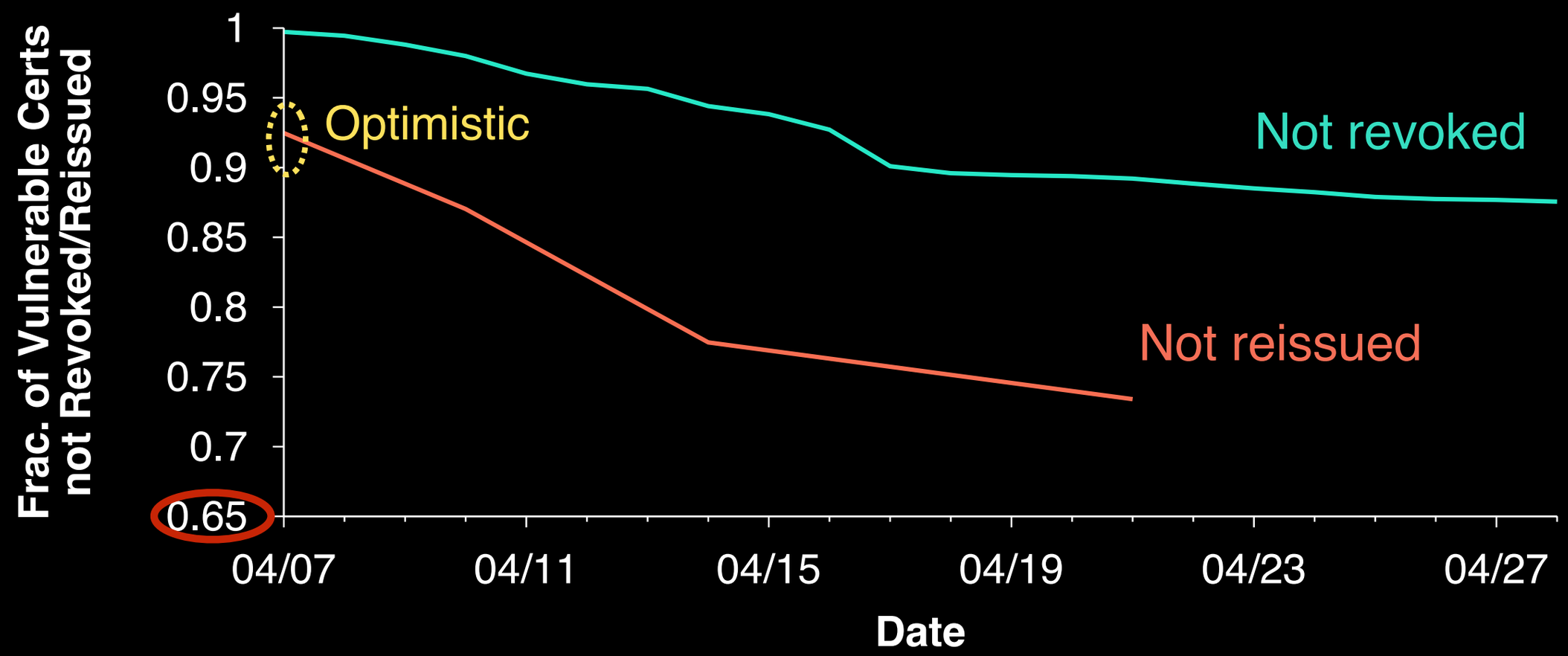


Similar pattern to patches:

Exponential drop-off, then levels out

After 3 weeks: 13% Revoked 27% Reissued

Certificate update rates

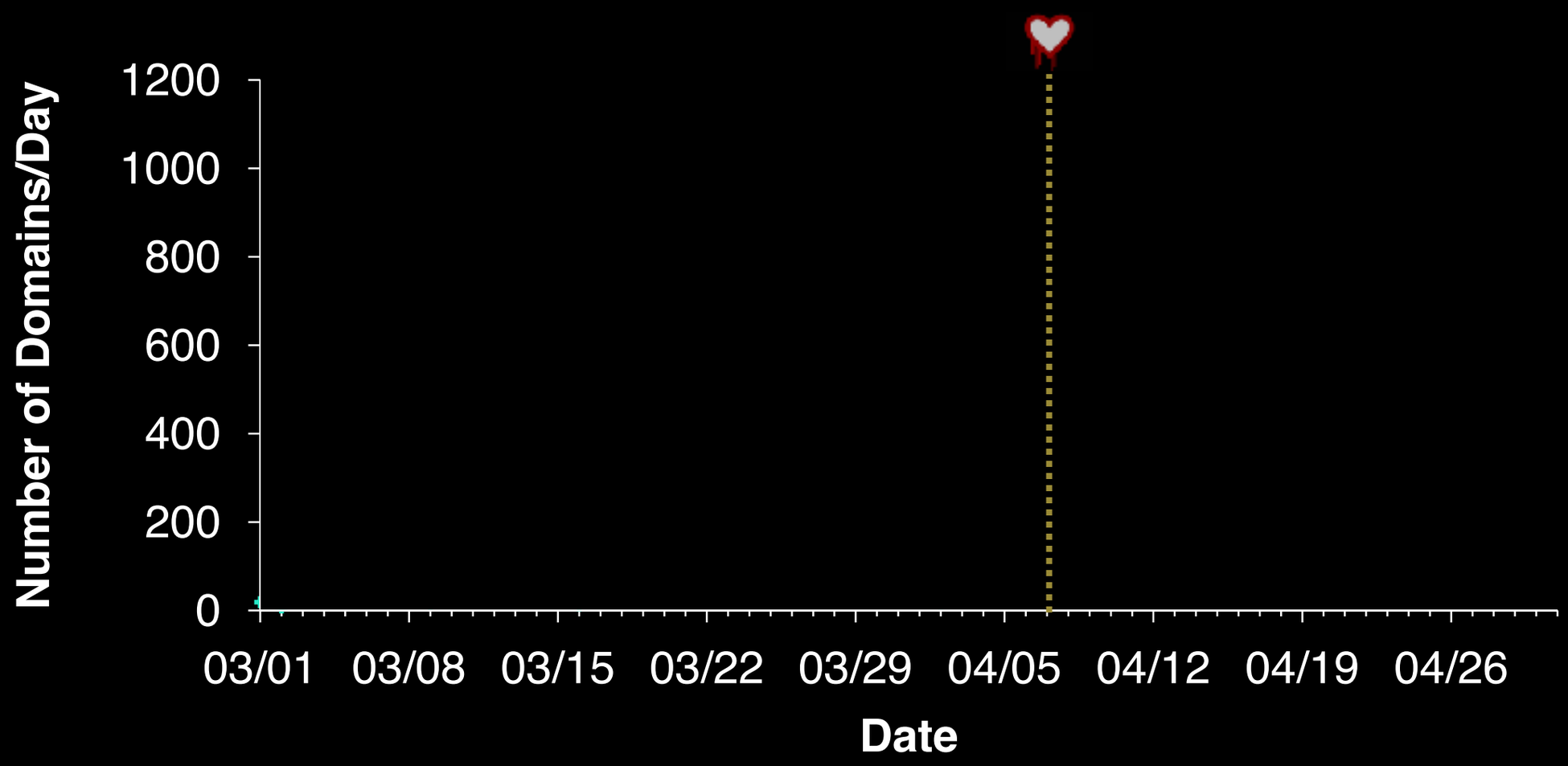


Similar pattern to patches:

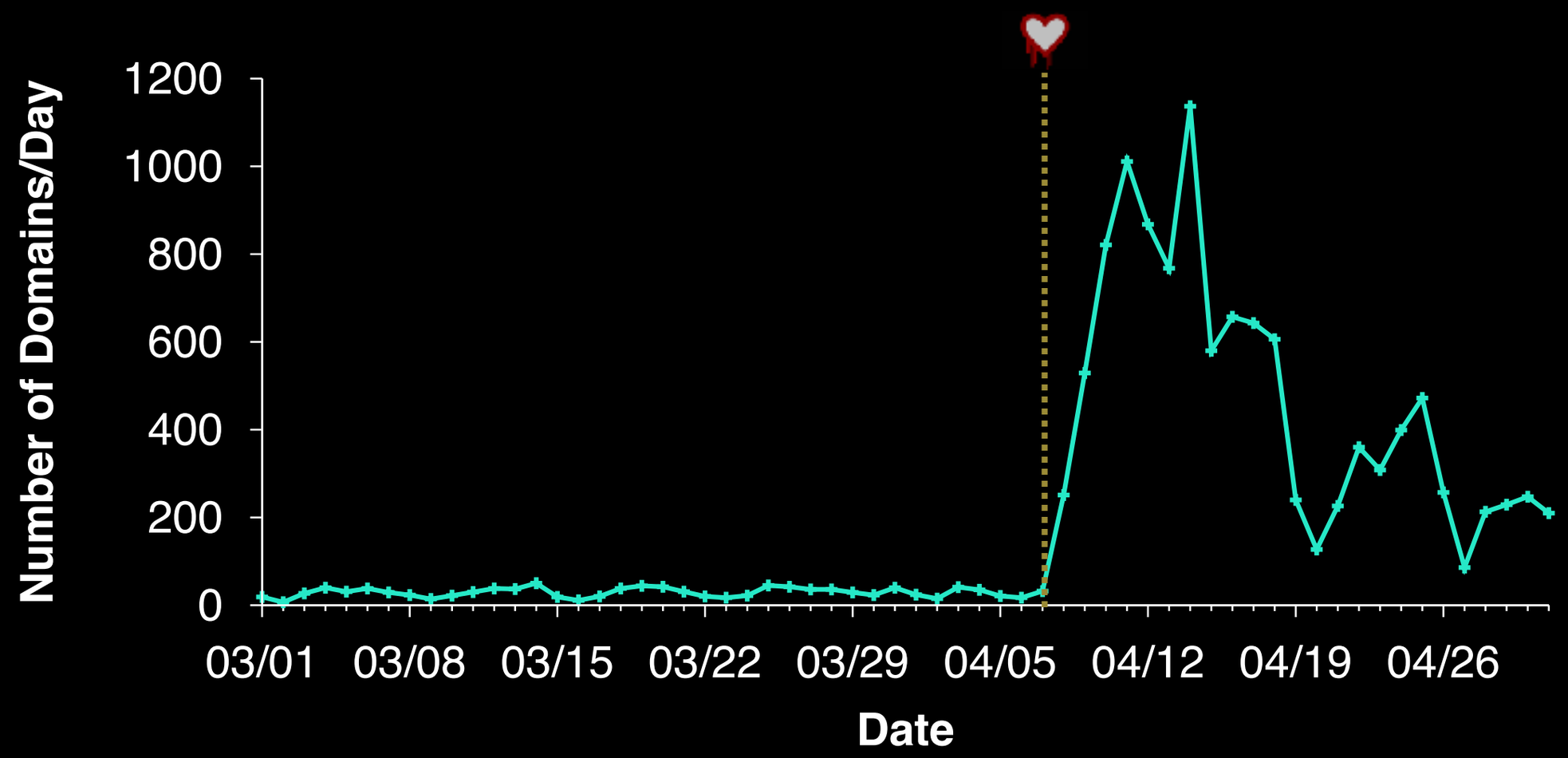
Exponential drop-off, then levels out

After 3 weeks: 13% Revoked 27% Reissued

How quickly were certs revoked?

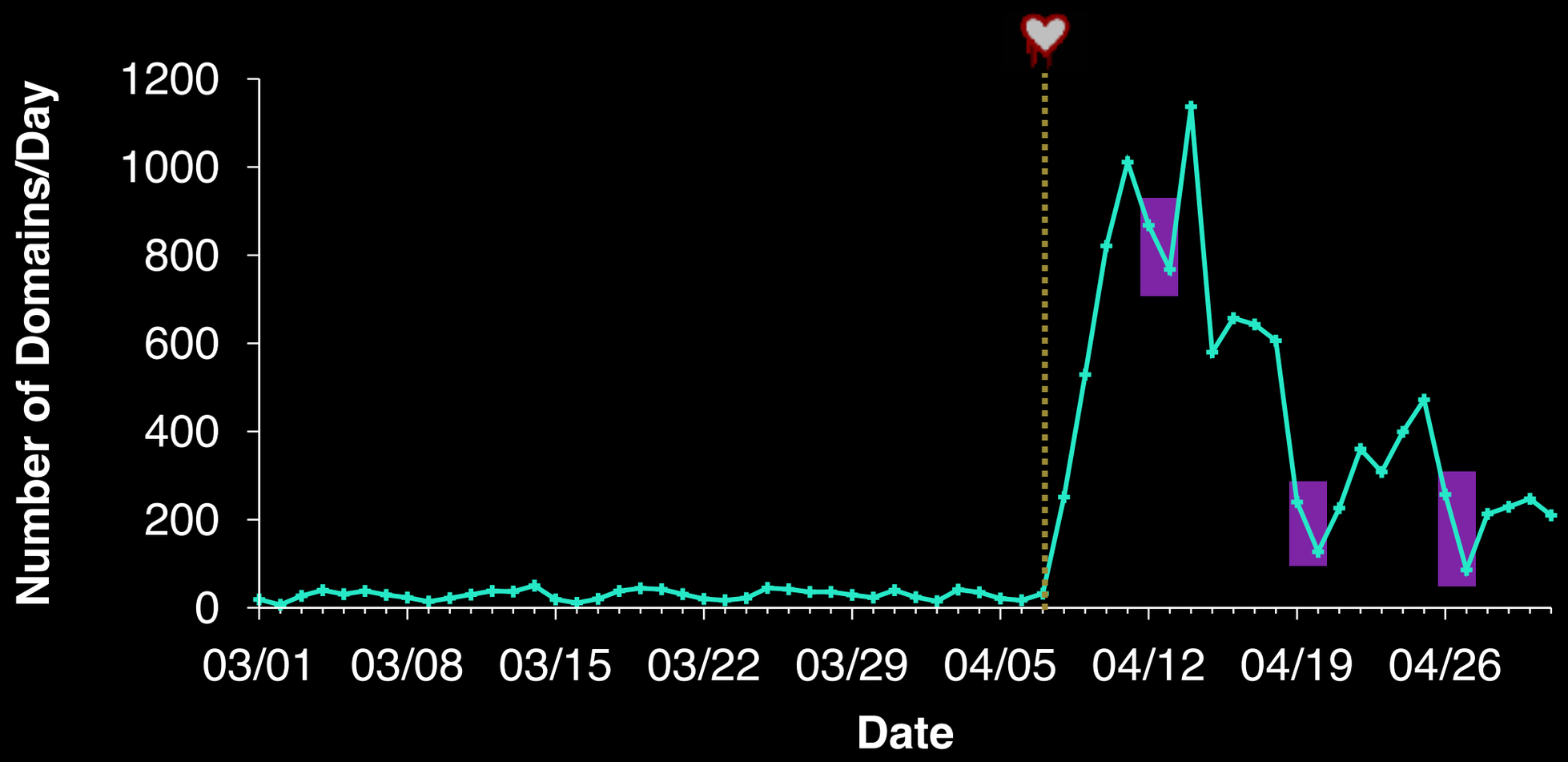


How quickly were certs revoked?



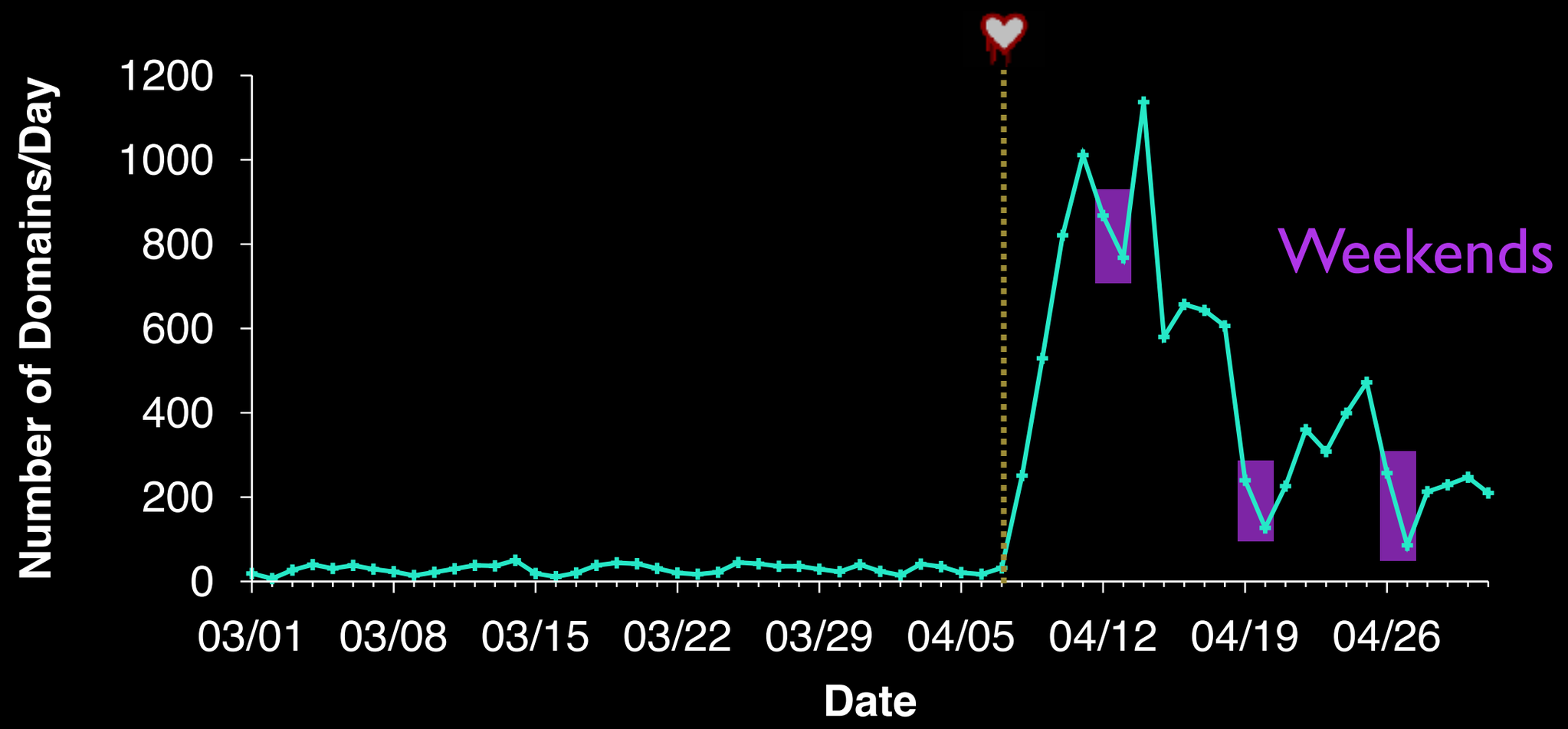
Reaction ramps up quickly

How quickly were certs revoked?



Reaction ramps up quickly

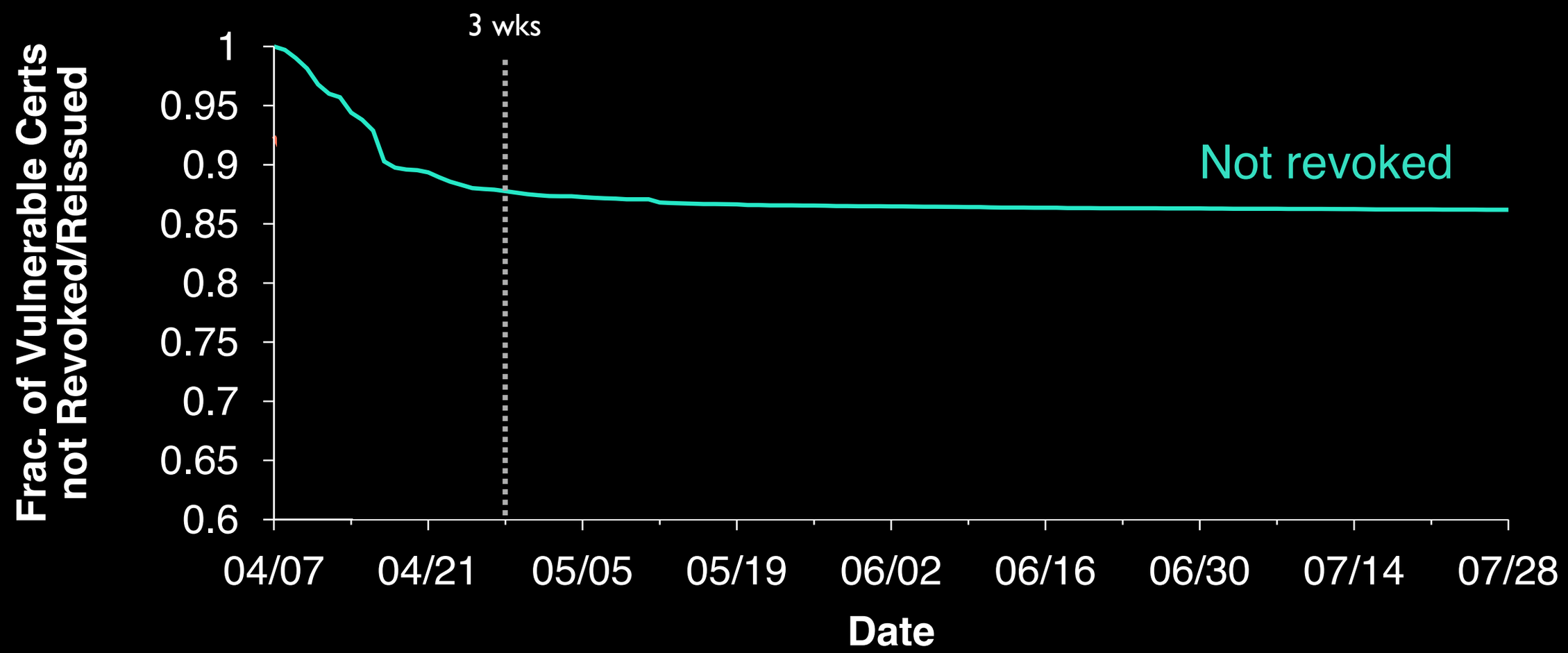
How quickly were certs revoked?



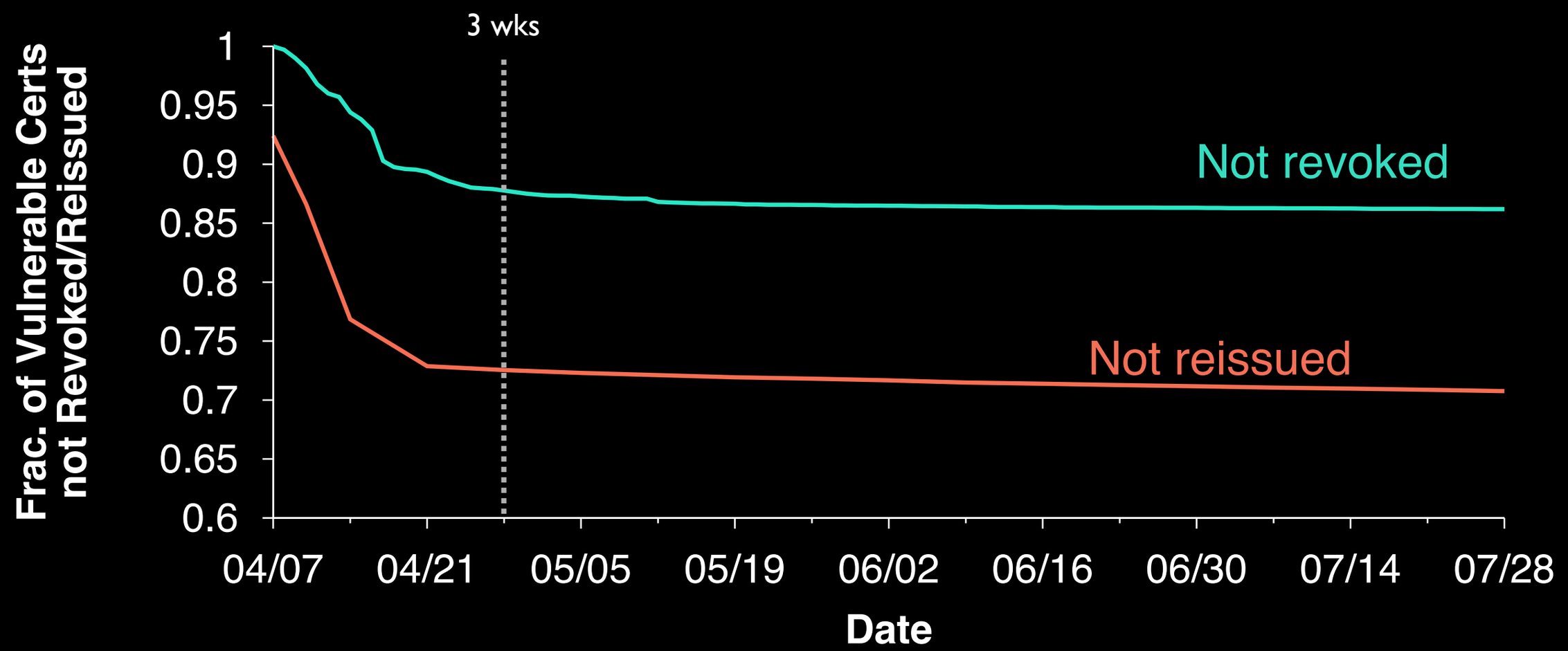
Reaction ramps up quickly

Security takes the weekends off

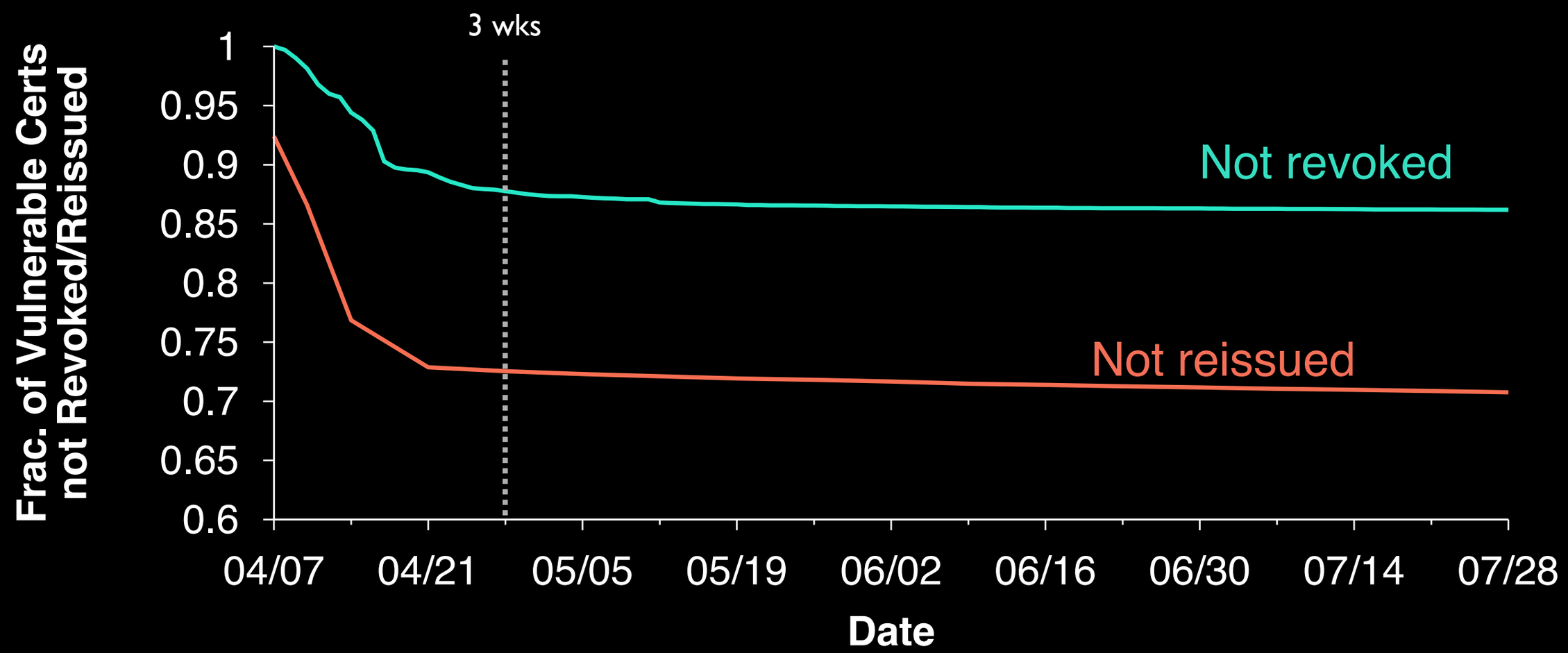
Certificate update rates



Certificate update rates



Certificate update rates

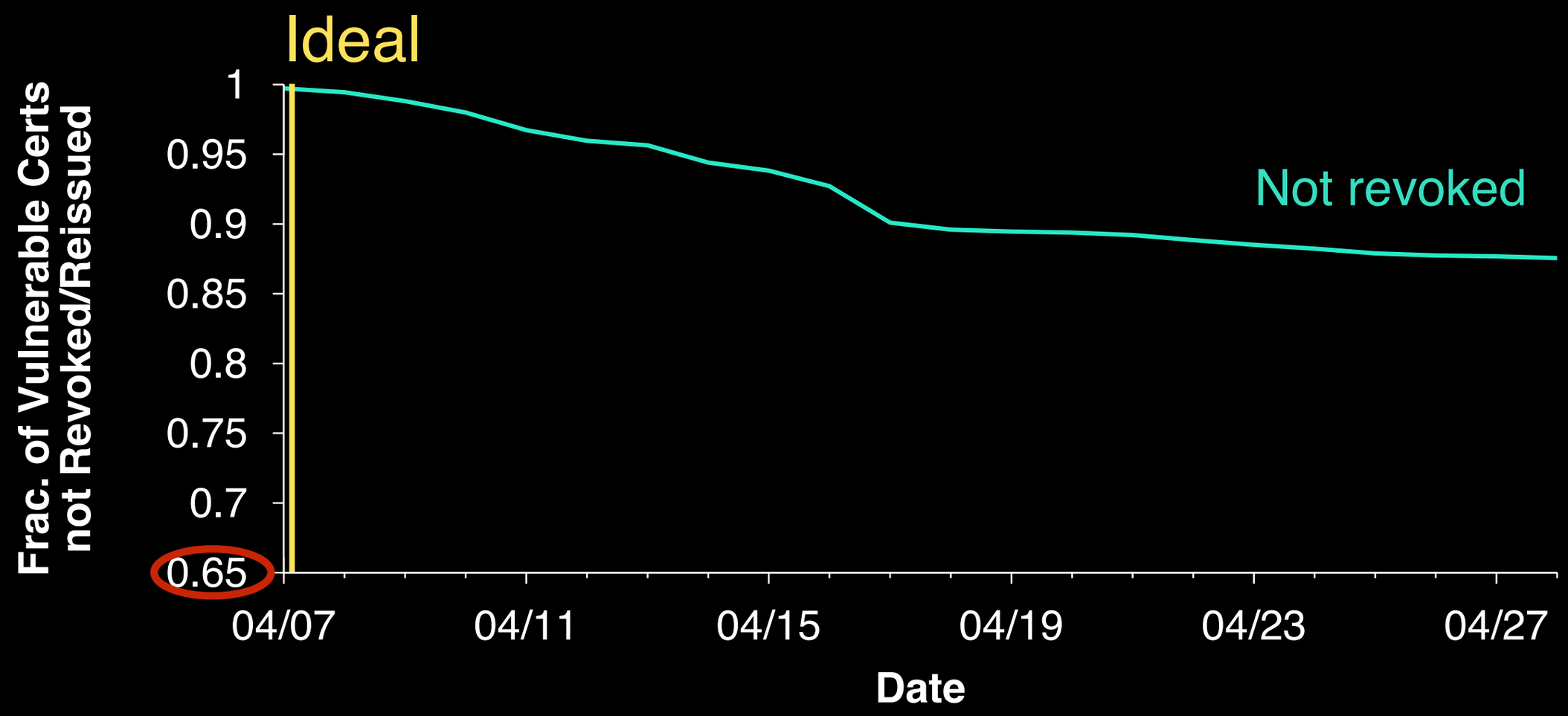


Similar pattern to patches:

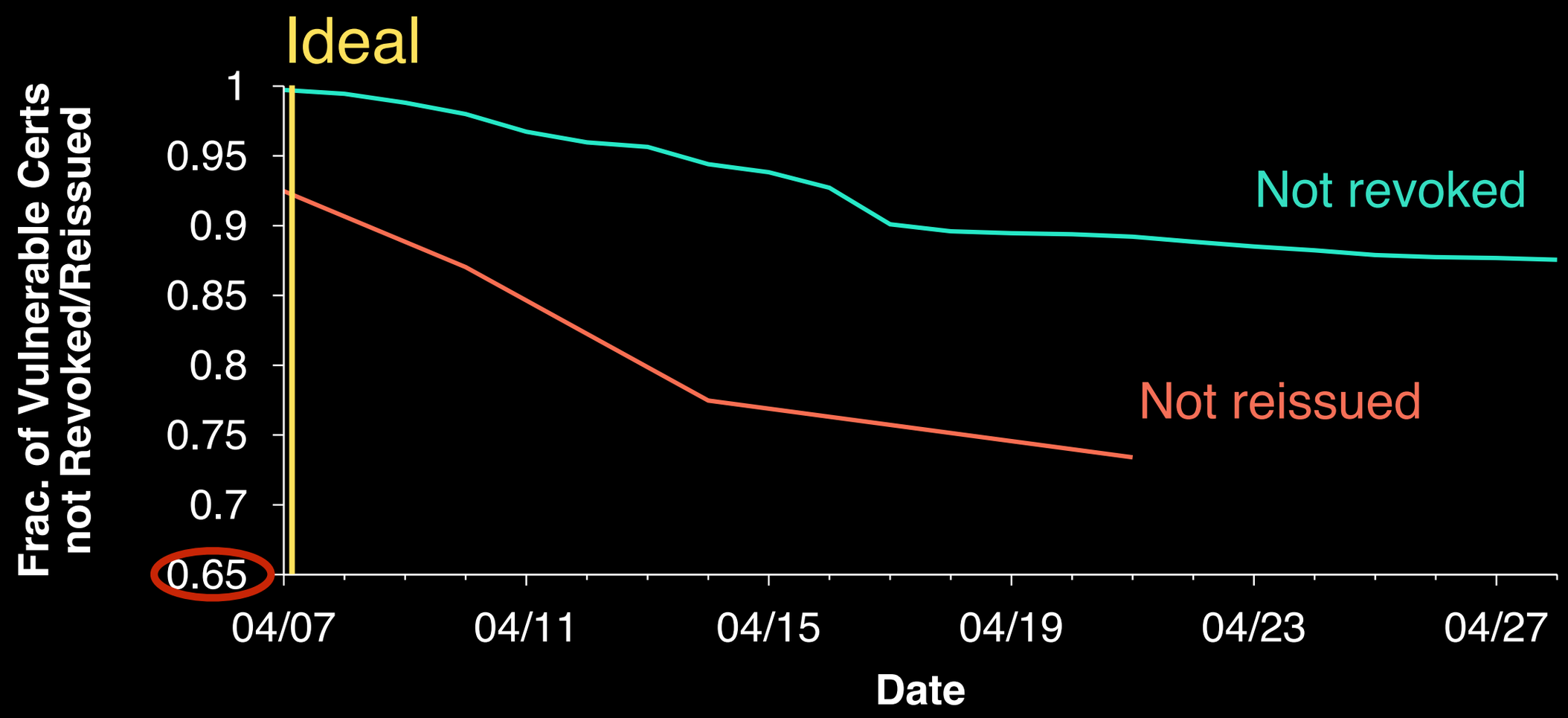
Exponential drop-off, then levels out

After 3 weeks: 13% Revoked 27% Reissued

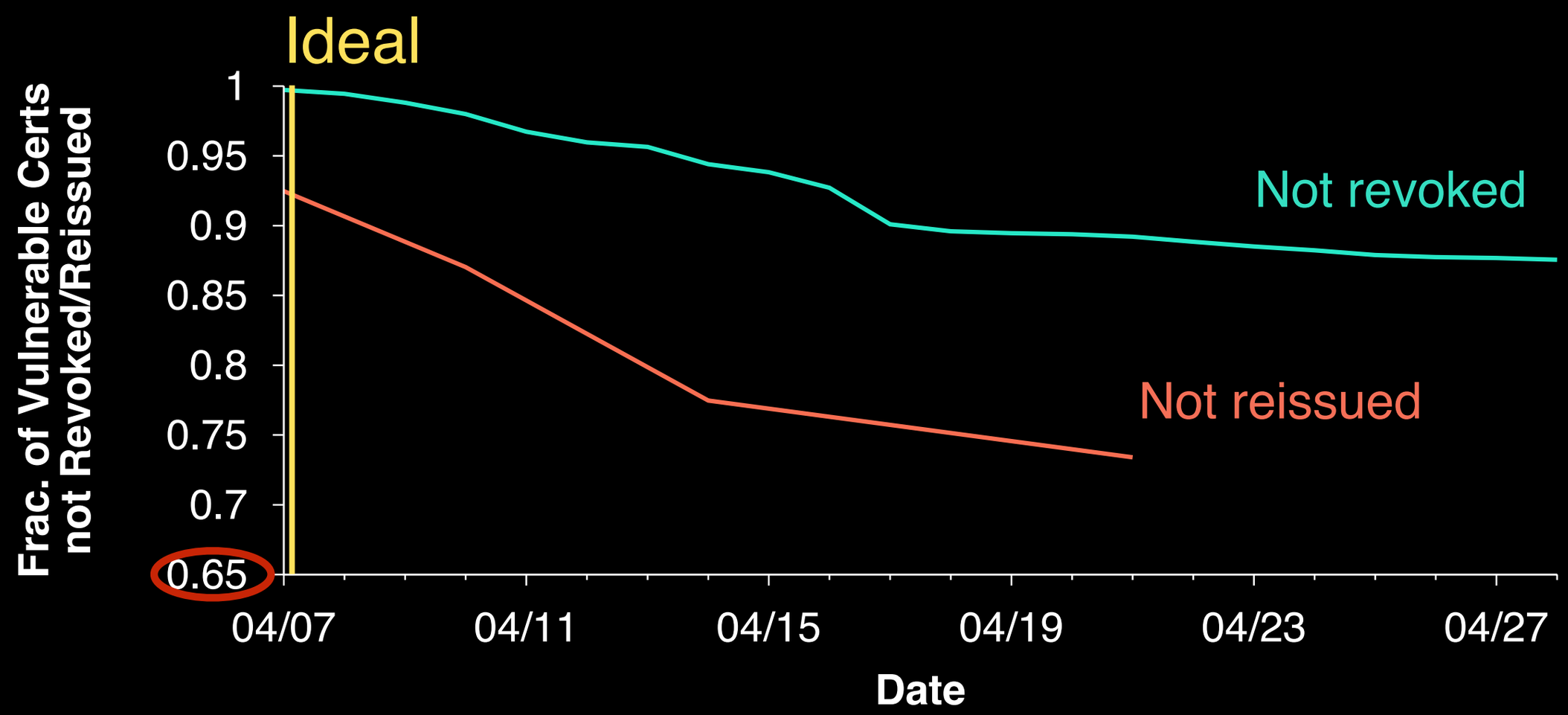
Certificate update rates



Certificate update rates



Certificate update rates

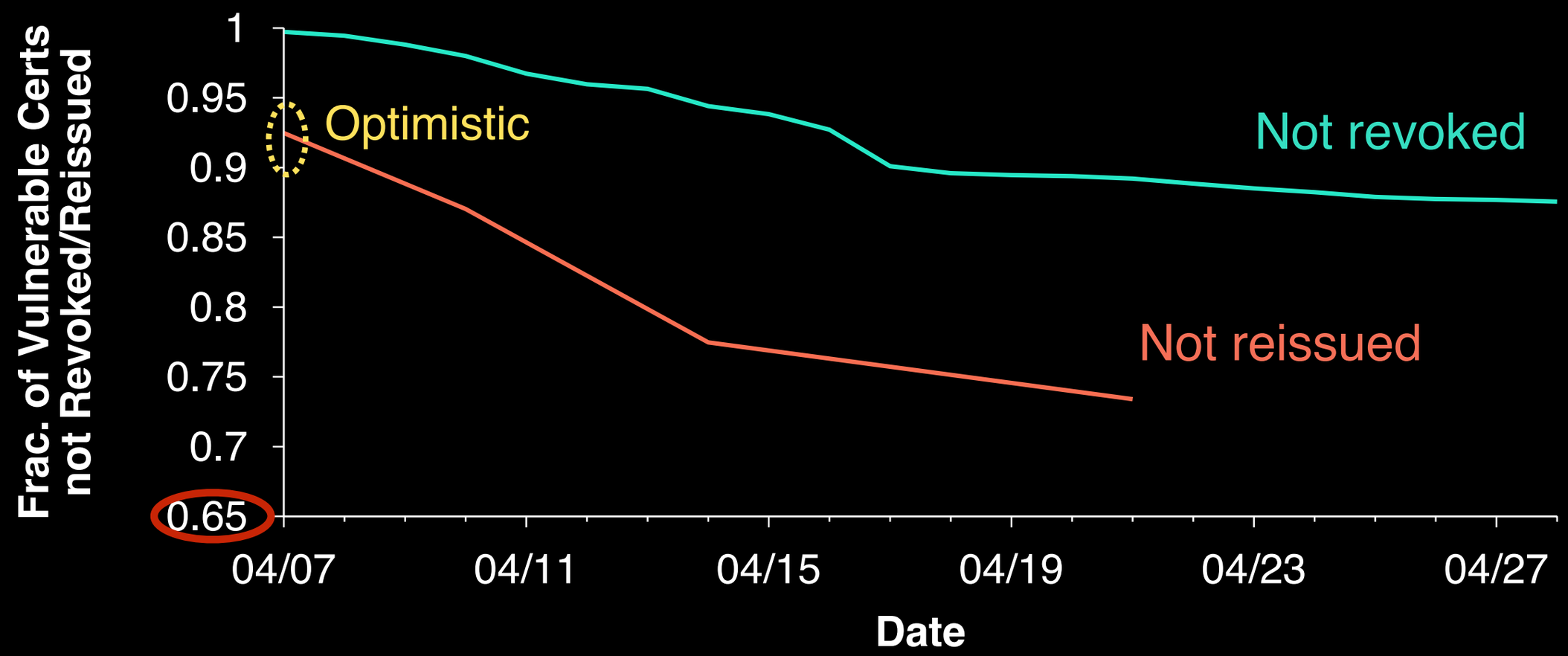


Similar pattern to patches:

Exponential drop-off, then levels out

After 3 weeks: 13% Revoked 27% Reissued

Certificate update rates

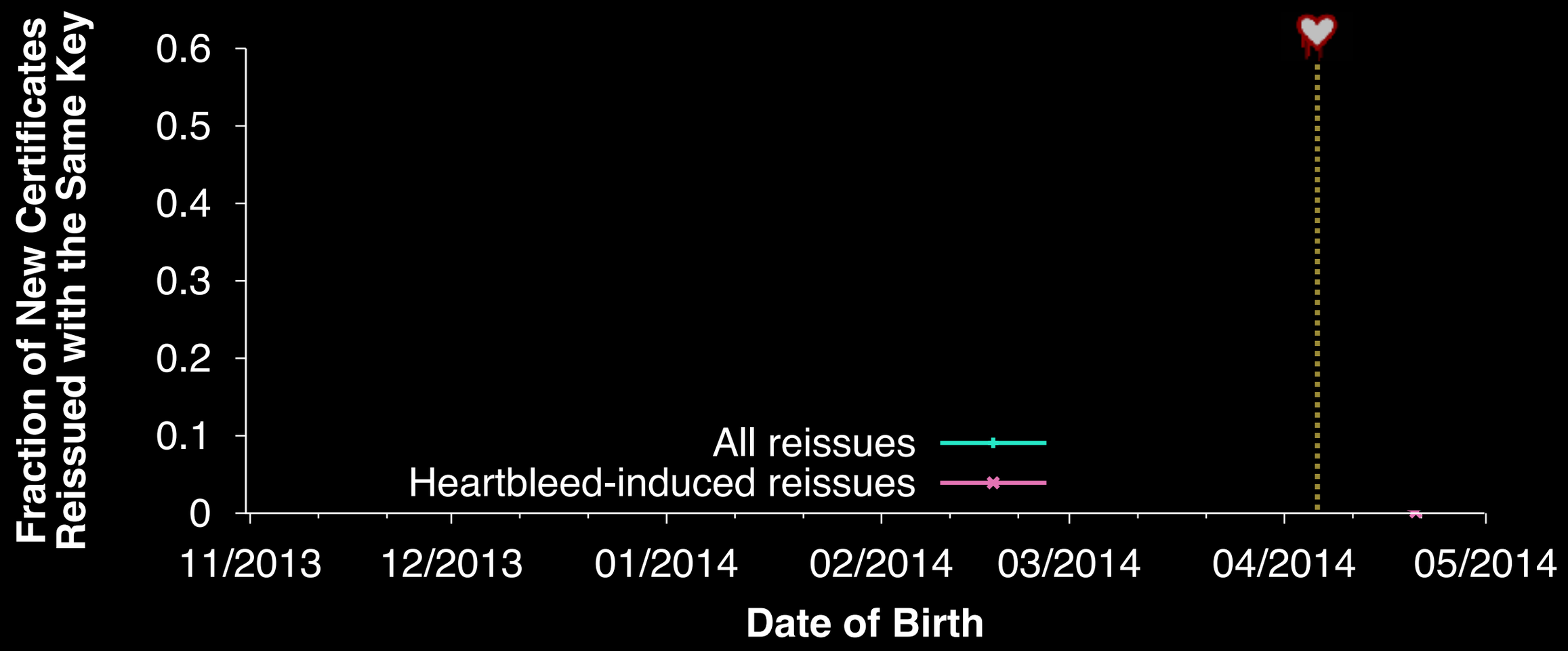


Similar pattern to patches:

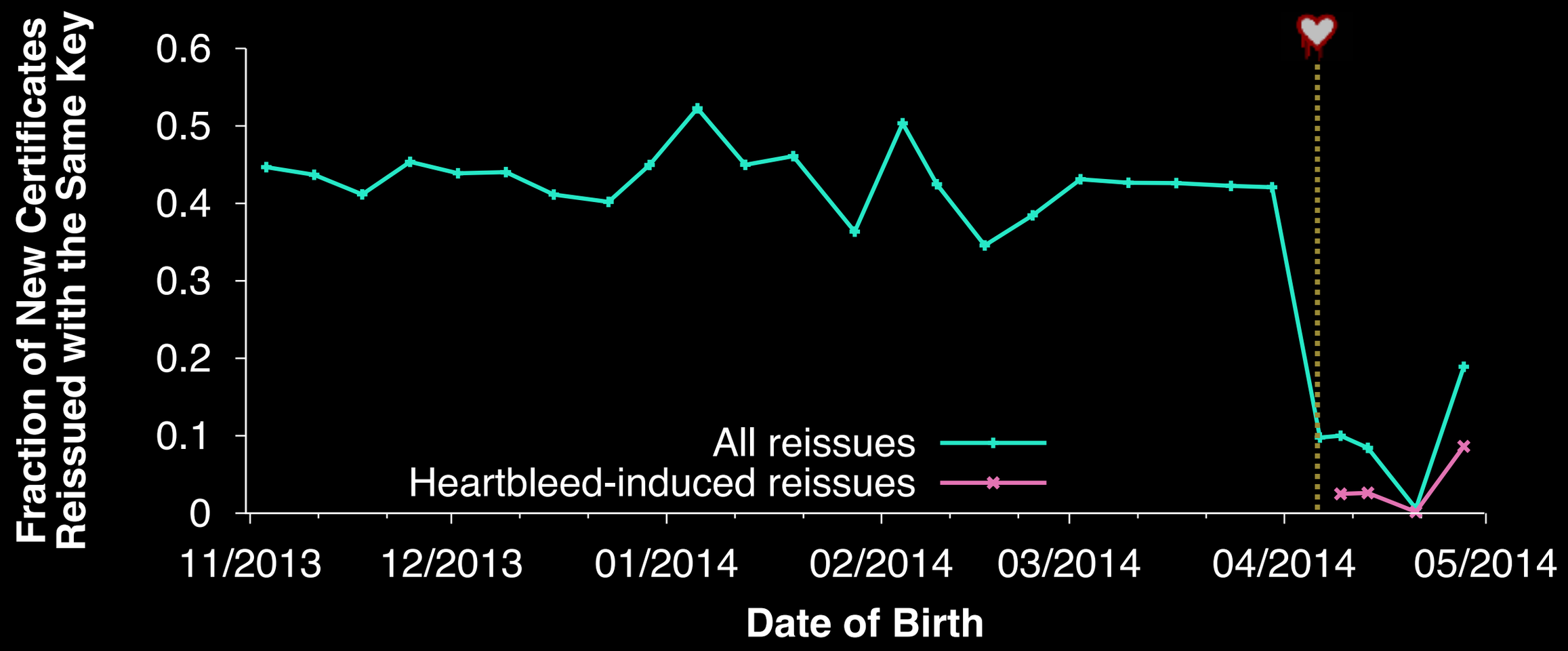
Exponential drop-off, then levels out

After 3 weeks: 13% Revoked 27% Reissued

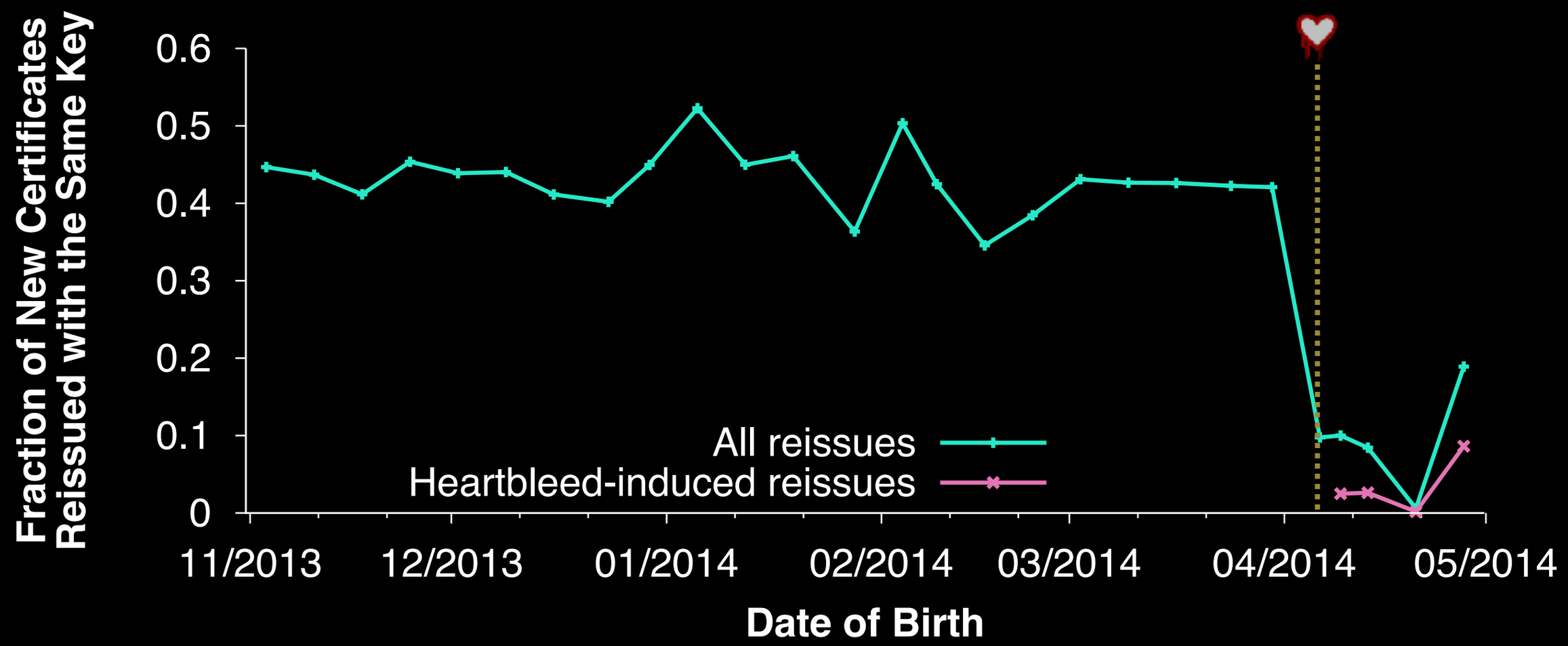
Reissue ⇒ New key?



Reissue ⇒ New key?



Reissue ⇒ New key?



Reissuing the same key is common practice

4.1% Heartbleed-induced

The ugly truth of revocations

93% Patched

13% Revoked

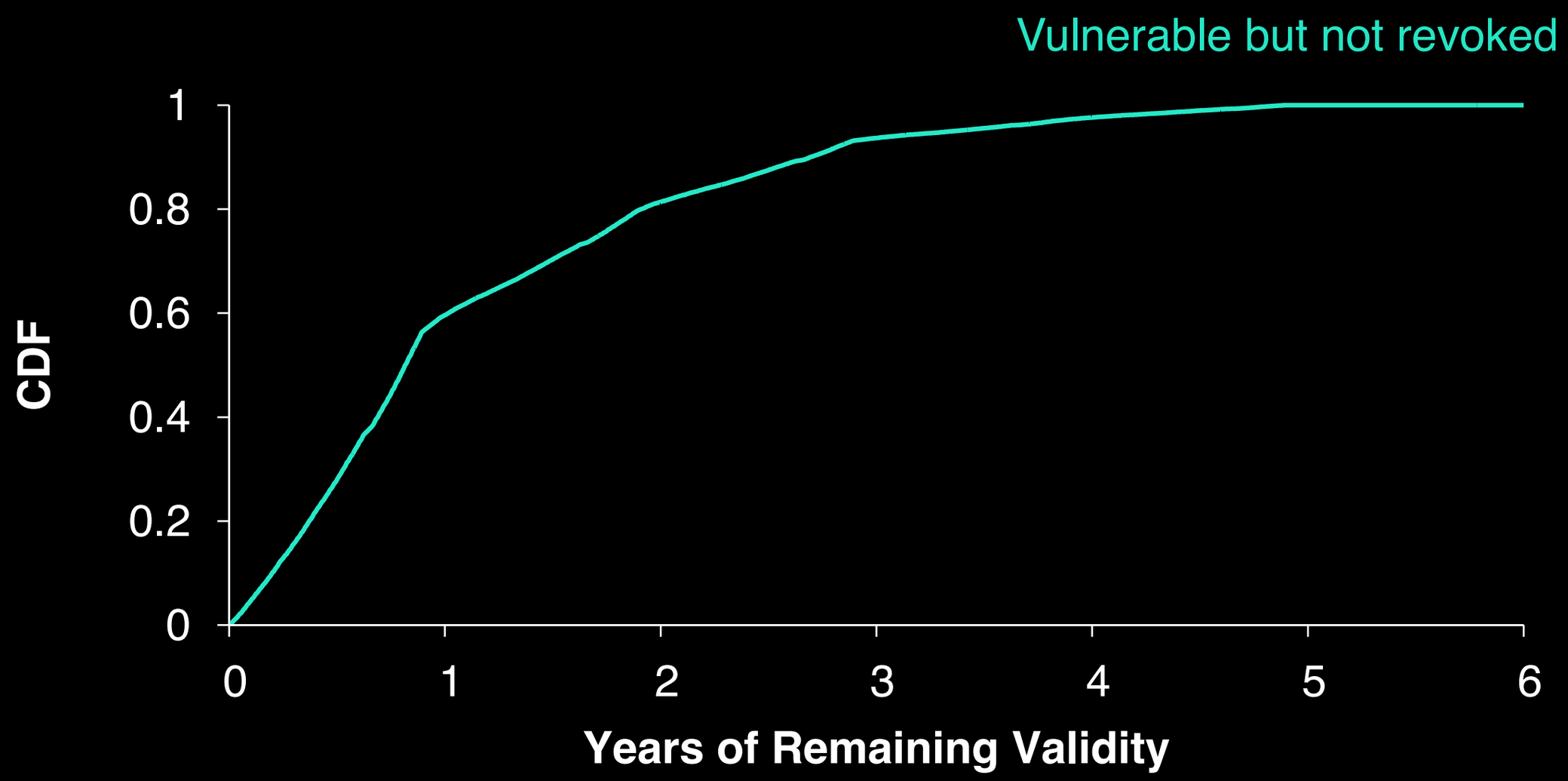
27% Reissued

Security is supposed to be a fundamental design goal, but

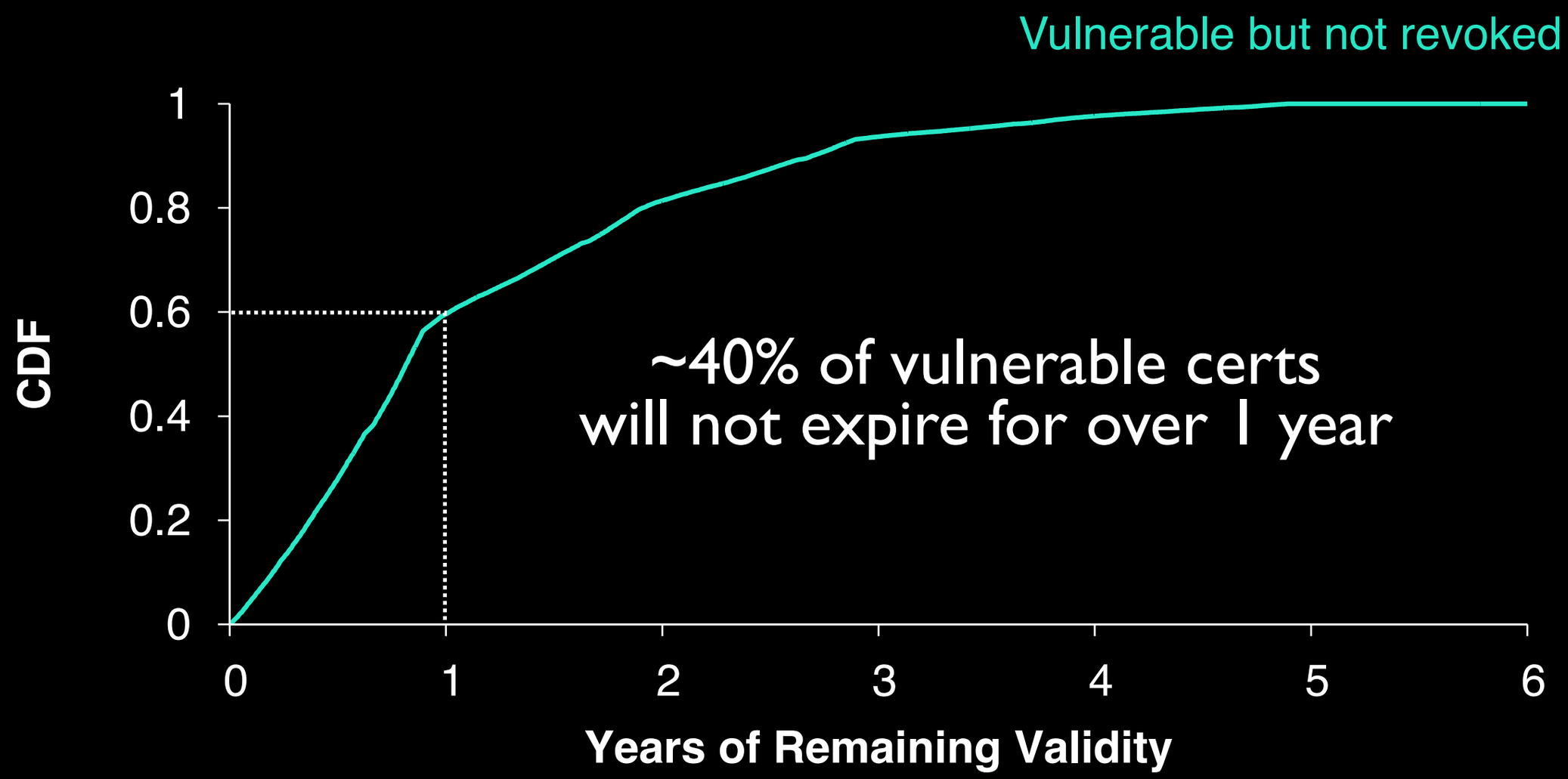
- Administrators trade off security for *ease of maintenance/cost*
- Certificate authorities trade off security for *profit*

Can we wait for expiration?

Can we wait for expiration?



Can we wait for expiration?



Can we wait for expiration?

We may be dealing with Heartbleed for *years*

Testing browser behavior

Revocation protocols

- Browsers *should* support all major protocols
  - CRLs, OCSP, OCSP stapling

Availability of revocation info

- Browsers *should* reject certs they cannot check
  - E.g., because the OCSP server is down

Chain lengths

- Browsers *should* reject a cert if *any* on the chain fail
  - Leaf, intermediate(s), root

Testing browser behavior

Revocation protocols

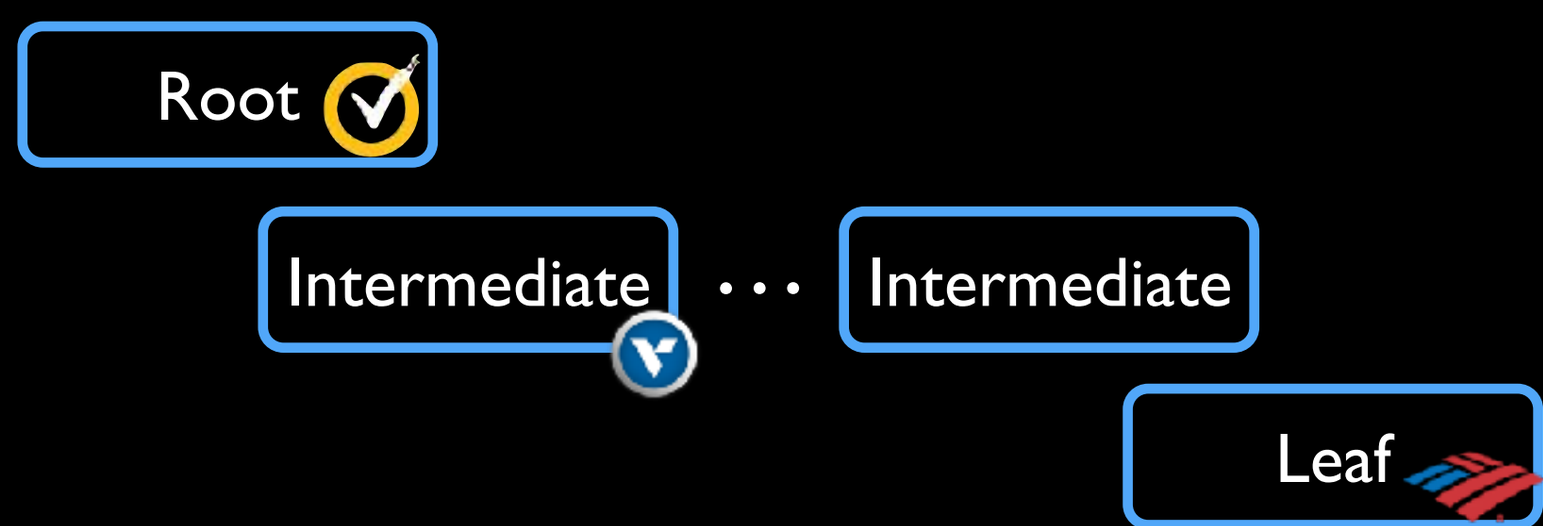
- Browsers *should* support all major protocols
  - CRLs, OCSP, OCSP stapling

Availability of revocation info

- Browsers *should* reject certs they cannot check
  - E.g., because the OCSP server is down

Chain lengths

- Browsers *should* reject a cert if *any* on the chain fail
  - Leaf, intermediate(s), root



Testing browser behavior

Revocation protocols

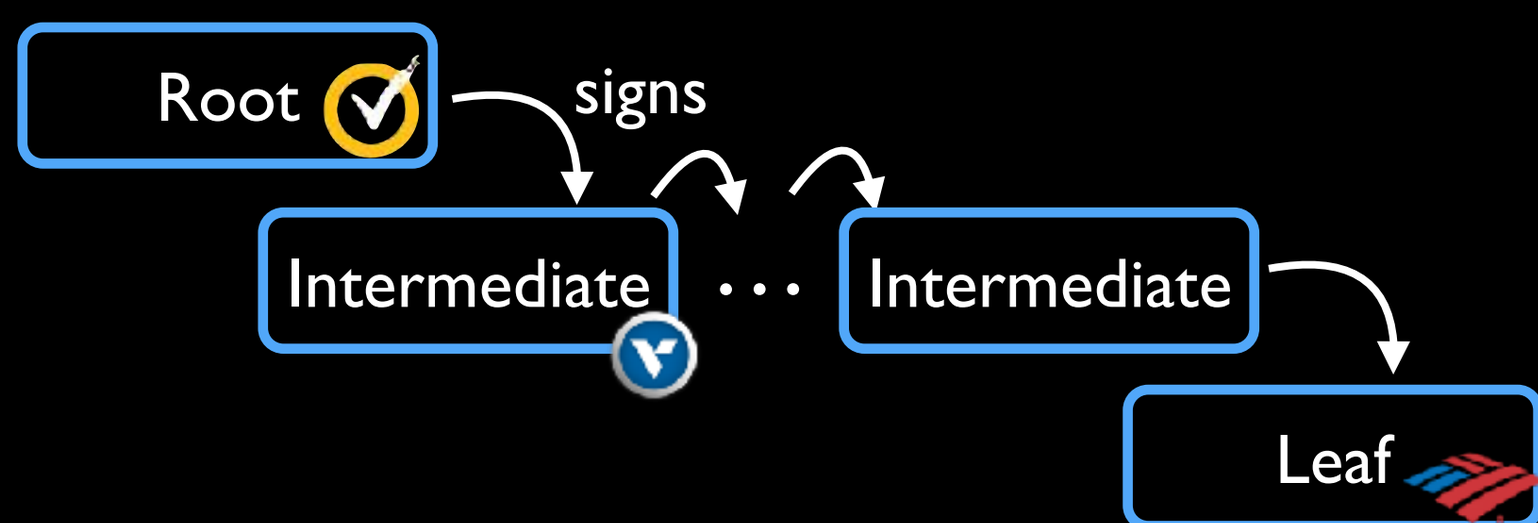
- Browsers *should* support all major protocols
  - CRLs, OCSP, OCSP stapling

Availability of revocation info

- Browsers *should* reject certs they cannot check
  - E.g., because the OCSP server is down

Chain lengths

- Browsers *should* reject a cert if *any* on the chain fail
  - Leaf, intermediate(s), root



Testing browser behavior

Revocation protocols

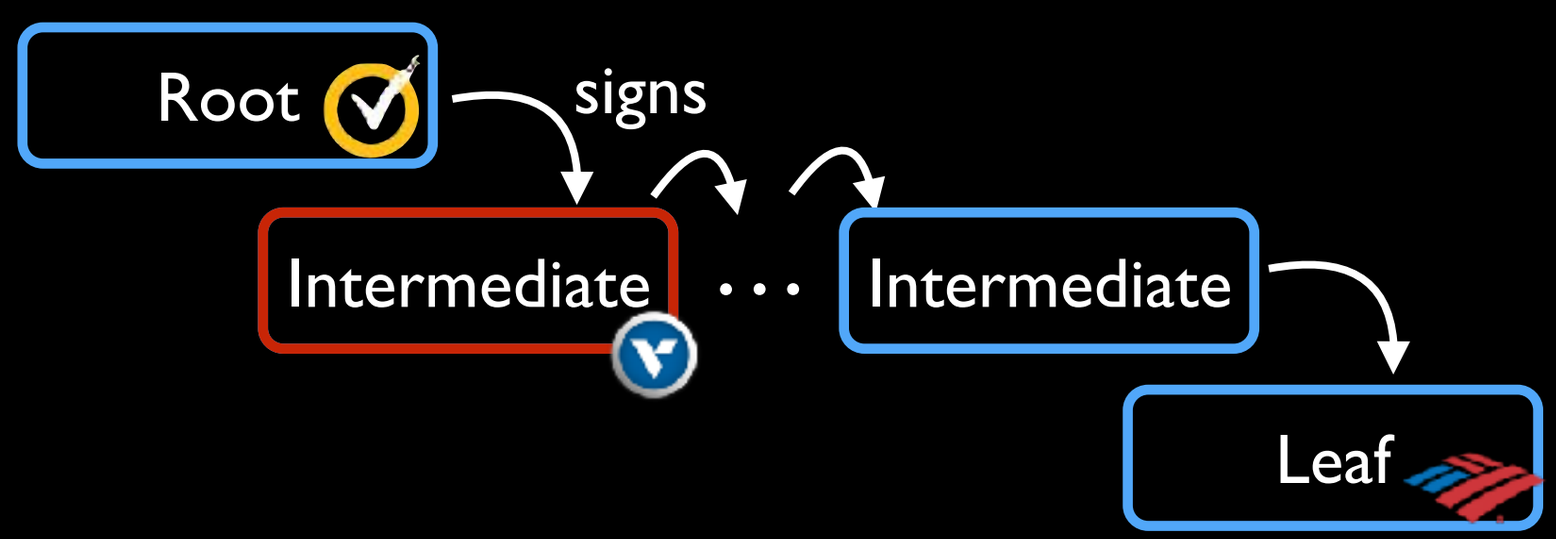
- Browsers *should* support all major protocols
  - CRLs, OCSP, OCSP stapling

Availability of revocation info

- Browsers *should* reject certs they cannot check
  - E.g., because the OCSP server is down

Chain lengths

- Browsers *should* reject a cert if *any* on the chain fail
  - Leaf, intermediate(s), root



Testing browser behavior

Revocation protocols

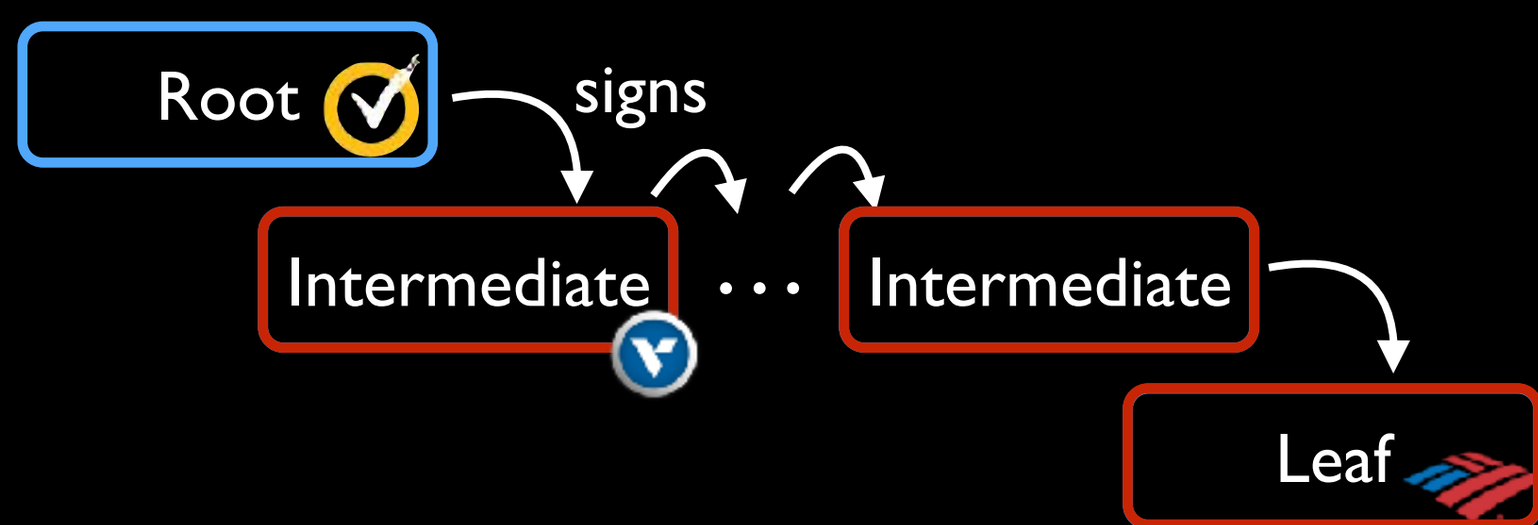
- Browsers *should* support all major protocols
  - CRLs, OCSP, OCSP stapling

Availability of revocation info

- Browsers *should* reject certs they cannot check
  - E.g., because the OCSP server is down

Chain lengths

- Browsers *should* reject a cert if *any* on the chain fail
  - Leaf, intermediate(s), root



Results across all browsers

		Desktop Browsers									Mobile Browsers			
		Chrome 42			Firefox	Opera		Safari	IE		iOS	Andr. 4.1–5.1		IE
		OS X	Win.	Linux	35–37	12.17	28.0	6–8	7–9	10–11	6–8	Stock	Chrome	8.0
	CRL													
Int. 1	Revoked	EV	✓	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	EV	✓	–	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
Int. 2+	Revoked	EV	EV	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Leaf	Revoked	EV	EV	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	A	✗	✗	✗	✗
	OCSP													
Int. 1	Revoked	EV	EV	EV	EV	✗	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	L/W	✗	✓	✓	✗	✗	✗	✗
Int. 2+	Revoked	EV	EV	EV	EV	✗	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Leaf	Revoked	EV	EV	EV	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	A	✗	✗	✗	✗
	OCSP Stapling													
	Request OCSP Staple	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	I	I	✗
	Respect Revoked Staple	✗	✓	–	✓	✓	L/W	–	✓	✓	–	–	–	–

✓ Passes test

✗ Fails test

EV Passes for EV certs

I Ignores OCSP Staple

A Pops up alert to user

L/W Passes on Linux/Win.

Results across all browsers

		Chrome 42		
		OS X	Win.	Linux
	CRL			
Int. 1	Revoked	EV	✓	EV
	Unavailable	EV	✓	–
Int. 2+	Revoked	EV	EV	EV
	Unavailable	✗	✗	–
Leaf	Revoked	EV	EV	EV
	Unavailable	✗	✗	–
	OCSP			
Int. 1	Revoked	EV	EV	EV
	Unavailable	✗	✗	–
Int. 2+	Revoked	EV	EV	EV
	Unavailable	✗	✗	–
Leaf	Revoked	EV	EV	EV
	Unavailable	✗	✗	–
	OCSP Stapling			
	Request OCSP Staple	✓	✓	✓
	Respect Revoked Staple	✗	✓	–

Chrome

Generally, only checks for EV certs
~3% of all certs

Allows if revocation info unavailable

Supports OCSP stapling

✔ Passes test
✗ Fails test

EV Passes for EV certs
I Ignores OCSP Staple

A Pops up alert to user
L/W Passes on Linux/Win.

Results across all browsers

		Deskt Firefox 35–37
	CRL	
Int. 1	Revoked	✗
	Unavailable	✗
Int. 2+	Revoked	✗
	Unavailable	✗
Leaf	Revoked	✗
	Unavailable	✗
	OCSP	
Int. 1	Revoked	EV
	Unavailable	✗
Int. 2+	Revoked	EV
	Unavailable	✗
Leaf	Revoked	✓
	Unavailable	✗
	OCSP Stapling	
	Request OCSP Staple	✓
	Respect Revoked Staple	✓

Firefox

Never checks CRLs
Only checks intermediates for EV certs

Allows if revocation info unavailable

Supports OCSP stapling

✔ Passes test
✗ Fails test

EV Passes for EV certs
I Ignores OCSP Staple

A Pops up alert to user
L/W Passes on Linux/Win.

Results across all browsers

		Safari 6–8
	CRL	
Int. 1	Revoked	✓
	Unavailable	✓
Int. 2+	Revoked	✓
	Unavailable	✗
Leaf	Revoked	✓
	Unavailable	✗
	OCSP	
Int. 1	Revoked	✓
	Unavailable	✗
Int. 2+	Revoked	✓
	Unavailable	✗
Leaf	Revoked	✓
	Unavailable	✗
OCSP Stapling		
Request OCSP Staple		✗
Respect Revoked Staple		–

Safari

Checks CRLs and OCSP

Allows if revocation info unavailable
Except for first intermediate, for CRLs

Does *not* support OCSP stapling

✔ Passes test
✗ Fails test

EV Passes for EV certs
I Ignores OCSP Staple

A Pops up alert to user
L/W Passes on Linux/Win.

Results across all browsers

		IE 7–9	IE 10–11
	CRL		
Int. 1	Revoked	✓	✓
	Unavailable	✓	✓
Int. 2+	Revoked	✓	✓
	Unavailable	✗	✗
Leaf	Revoked	✓	✓
	Unavailable	✗	A
	OCSP		
Int. 1	Revoked	✓	✓
	Unavailable	✓	✓
Int. 2+	Revoked	✓	✓
	Unavailable	✗	✗
Leaf	Revoked	✓	✓
	Unavailable	✗	A
OCSP Stapling			
Request OCSP Staple		✓	✓
Respect Revoked Staple		✓	✓

Internet Explorer

Checks CRLs *and* OCSP

Often rejects if revocation info unavailable
Pops up alert for leaf in IE 10+

Supports OCSP stapling

✔ Passes test
✗ Fails test

EV Passes for EV certs
I Ignores OCSP Staple

A Pops up alert to user
L/W Passes on Linux/Win.

Results across all browsers

		Mobile Browsers			
		iOS 6–8	Andr. 4.1–5.1 Stock	Andr. 4.1–5.1 Chrome	IE 8.0
	CRL				
Int. 1	Revoked	✗	✗	✗	✗
	Unavailable	✗	✗	✗	✗
Int. 2+	Revoked	✗	✗	✗	✗
	Unavailable	✗	✗	✗	✗
Leaf	Revoked	✗	✗	✗	✗
	Unavailable	✗	✗	✗	✗
	OCSP				
Int. 1	Revoked	✗	✗	✗	✗
	Unavailable	✗	✗	✗	✗
Int. 2+	Revoked	✗	✗	✗	✗
	Unavailable	✗	✗	✗	✗
Leaf	Revoked	✗	✗	✗	✗
	Unavailable	✗	✗	✗	✗
OCSP Stapling					
Request OCSP Staple		✗	I	I	✗
Respect Revoked Staple		–	–	–	–

Mobile Browsers

Uniformly *never* check

Android browsers request Staple
…and promptly ignore it

✔ Passes test
✗ Fails test

EV Passes for EV certs
I Ignores OCSP Staple

A Pops up alert to user
L/W Passes on Linux/Win.

PKI CONCLUSION

The PKI is how we know with whom we are communicating online

The PKI's job is to bind human-understandable identities (domain names) to cryptographic keys (public keys)

The central mechanism for this is **certificates**: digital signatures from trusted entities that tie domain names and public keys together

TLS along with Diffie-Hellman leverages public key crypto to arrive at ephemeral *session keys* (symmetric keys)

There is significant *mismanagement* in today's PKI:

- Websites don't revoke or get new certs ("reissue") when they should
- Browsers don't check for revocations when they should
- Websites share their private keys with their hosting providers

Improving the web's PKI is an active area of research (securepki.org)