TLS and Certificates

CMSC 23200/33250, Winter 2021, Lecture 10

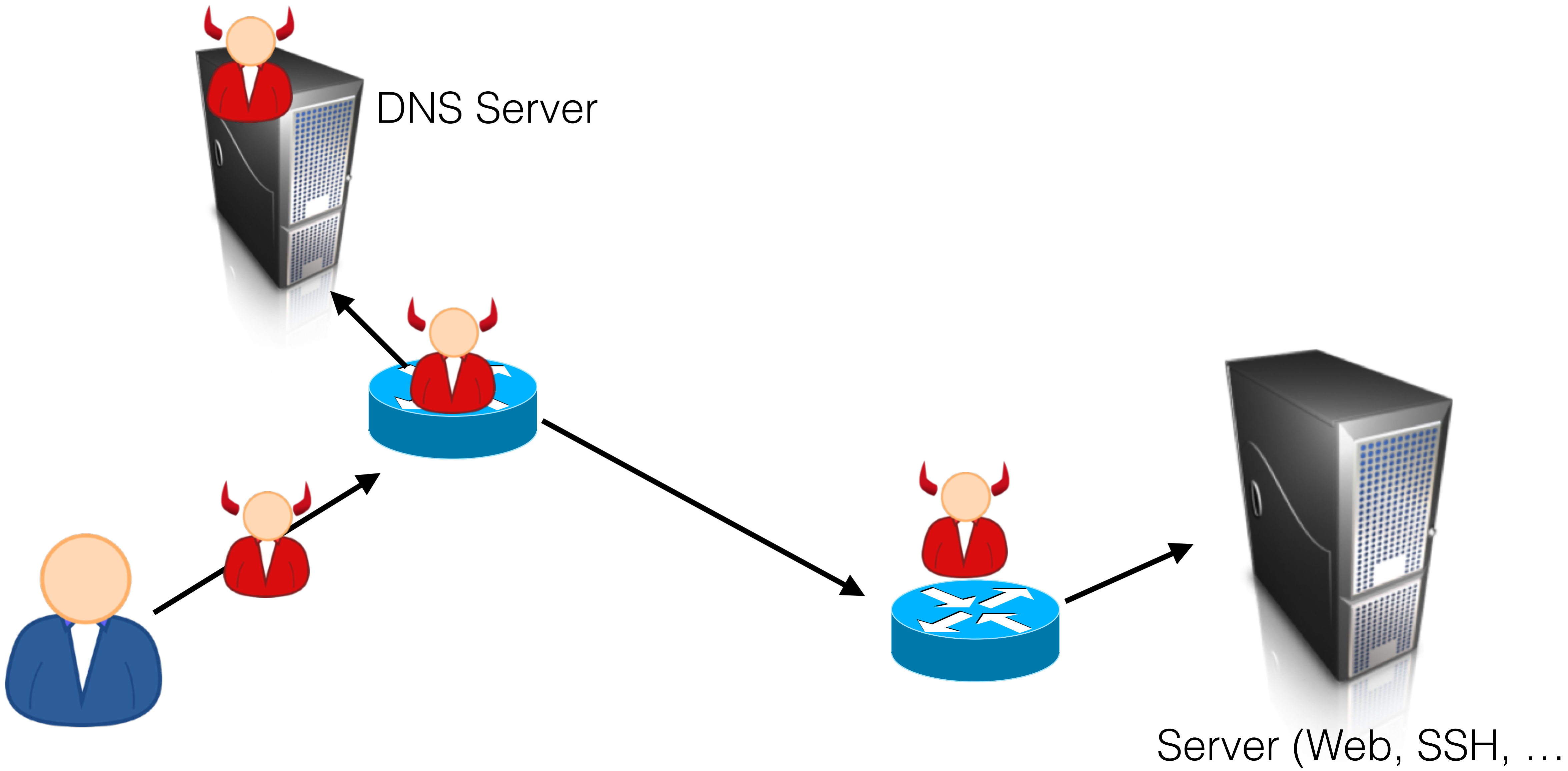
David Cash & Blase Ur

University of Chicago

Outline

- Securing a Connection to a Server: Threat Model
- Overview of how TLS works
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs, Attacks, Countermeasures
- Revoking Certs (Attacks, Countermeasures, …)
- A Closer Look at TLS 1.3 (2018 — Present)

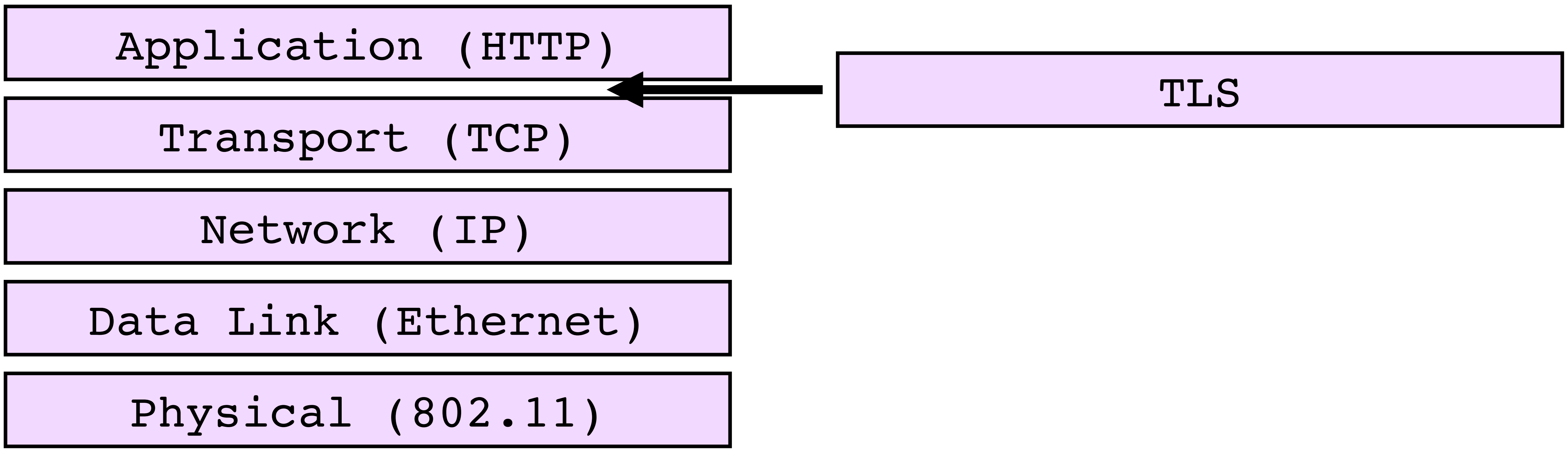
Threat Model for Secure Channels on the Internet



- Malicious/eavesdropping infrastructure
  - Examples: router, person at coffeeshop, ISP…
- Malicious DNS Server (who may lie)

- **Not** in threat model: Compromised endpoint

TLS in the Protocol Stack



- TLS takes requests from application (e.g. browser speaking HTTP)
- TLS uses TLS connection to communicate with other host

History: SSL/TLS

- SSL = “Secure Sockets Layer”
- TLS = “Transport Layer Security” (renaming of SSL)

Netscape

IETF

1993	1995	1996	1999	2006	2008	August 2018
SSL v1.0	SSL v2.0	SSL v3.0	TLS v1.0	TLS v1.1	TLS v1.2	TLS v1.3
Insecure	Insecure	Major issues	Major issues	Major issues	Worked but time for an upgrade	Going okay?

Security

It's official: TLS 1.3 approved as standard while spies weep

Now all you lot have to actually implement it

By Kieren McCarthy in San Francisco 13 Aug 2018 at 22:19 26 SHARE ▼

An overhaul of a critical internet security protocol has been completed, with TLS 1.3 becoming an official standard late last week.

TLS Adoption

- Originally for financial transactions

TLS Adoption

Official Blog

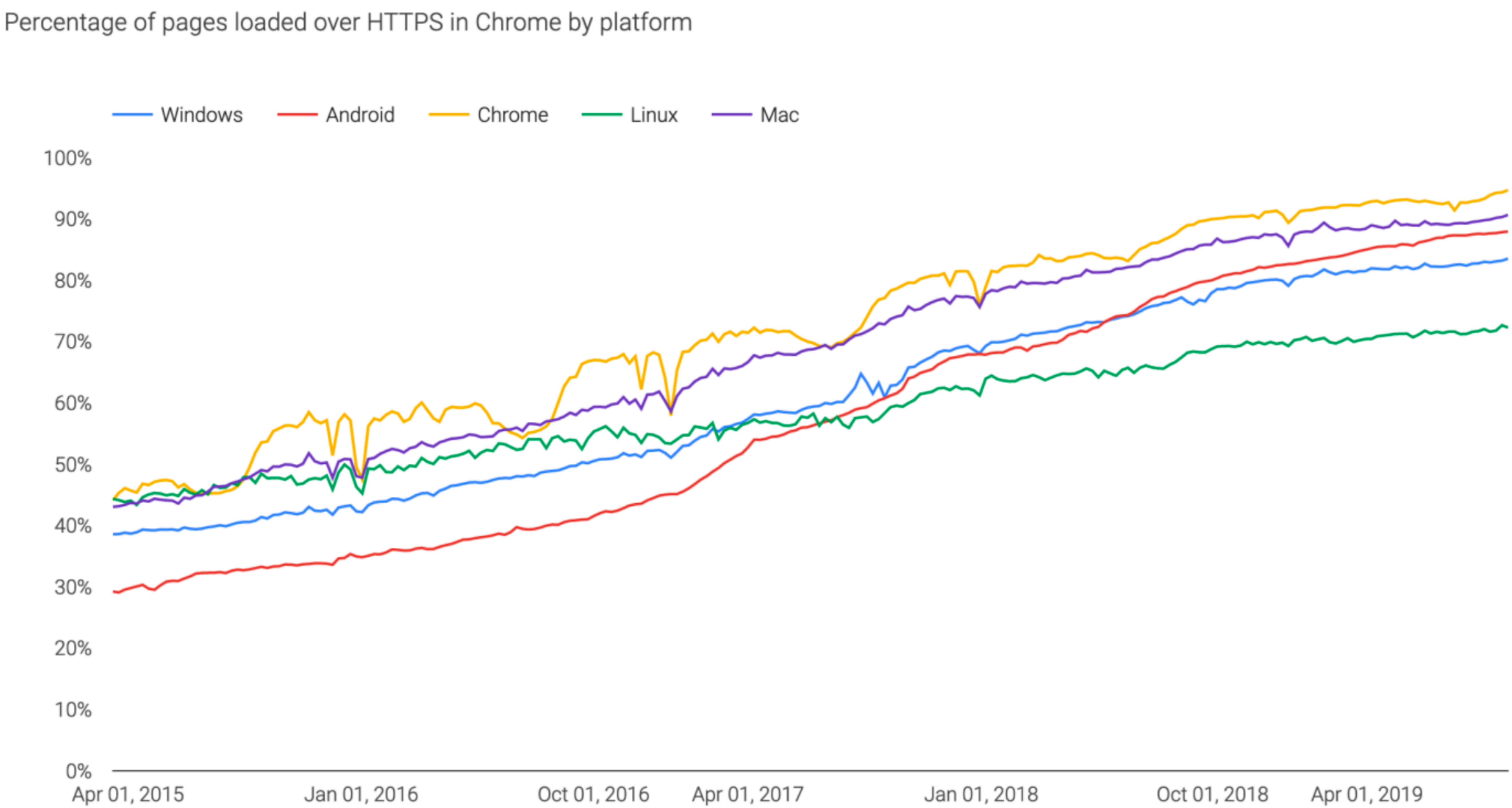
Insights from Googlers into our products, technology, and the Google culture

Search more securely with encrypted Google web search

May 21, 2010

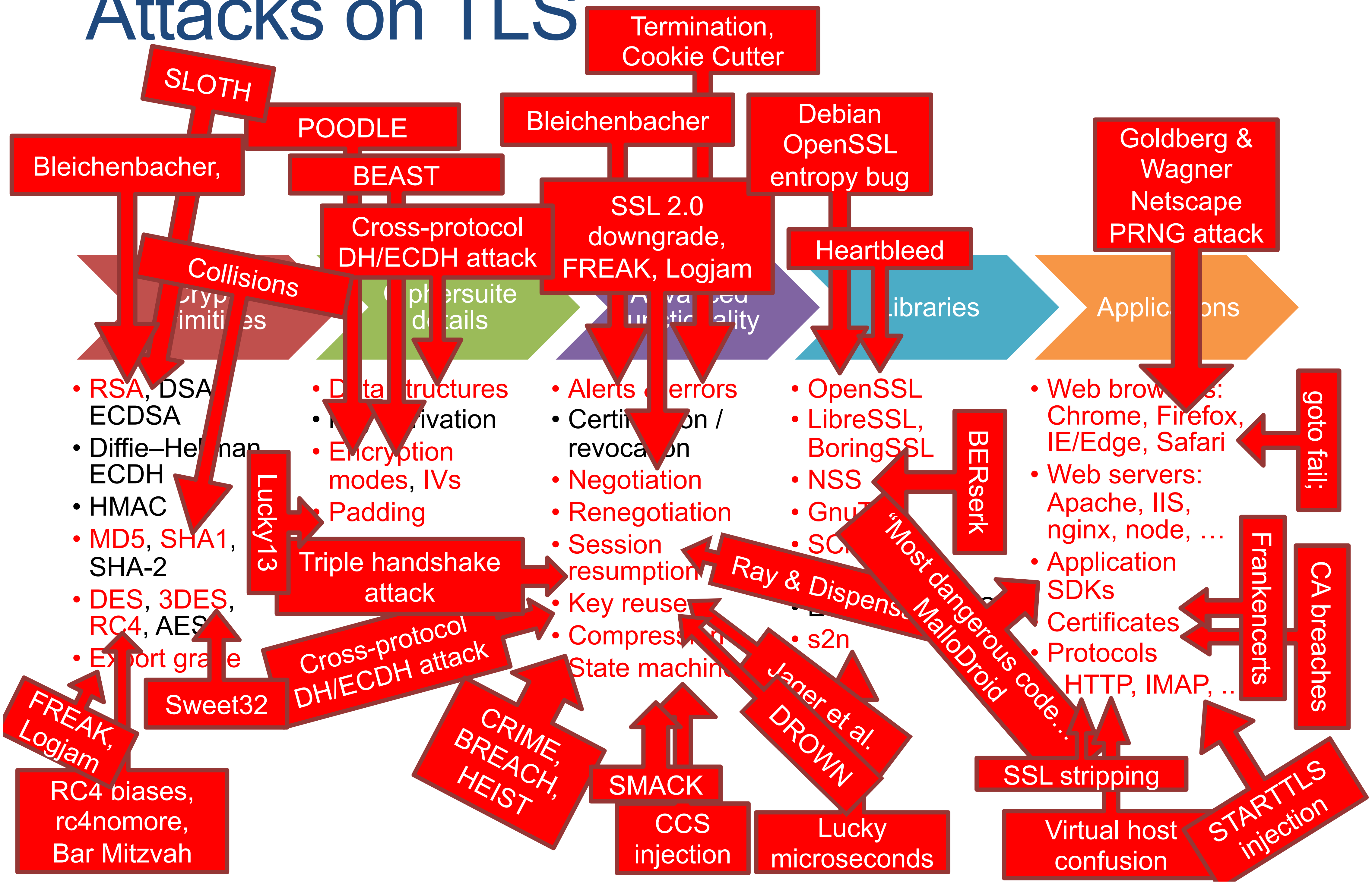
Update June 25, 2010: Since we introduced our encrypted search option last month, we've been listening closely to user feedback. Many users appreciate the capability to perform searches with better protection against snooping from third parties. We've also heard about some challenges faced by various school districts, and today, we want to inform you that we've moved encrypted search from https://www.google.com to https://encrypted.google.com. The site functions in the same way. For more information on this change, please read on here.

TLS Adoption



(Source: transparencyreport.google.com, via Matt Green)

Attacks on TLS



Template For Secure Channels (TLS, SSH, IPSec, …)

Key Exchange (“Handshake”)

`uchicago.edu`

A →

← B

Symmetric Encryption (“Record Protocol”)

$A \leftarrow g^a$

$K \leftarrow \mathrm{Hash}(B^a)$

← <encrypted data>

<encrypted data> →

← <encrypted data>

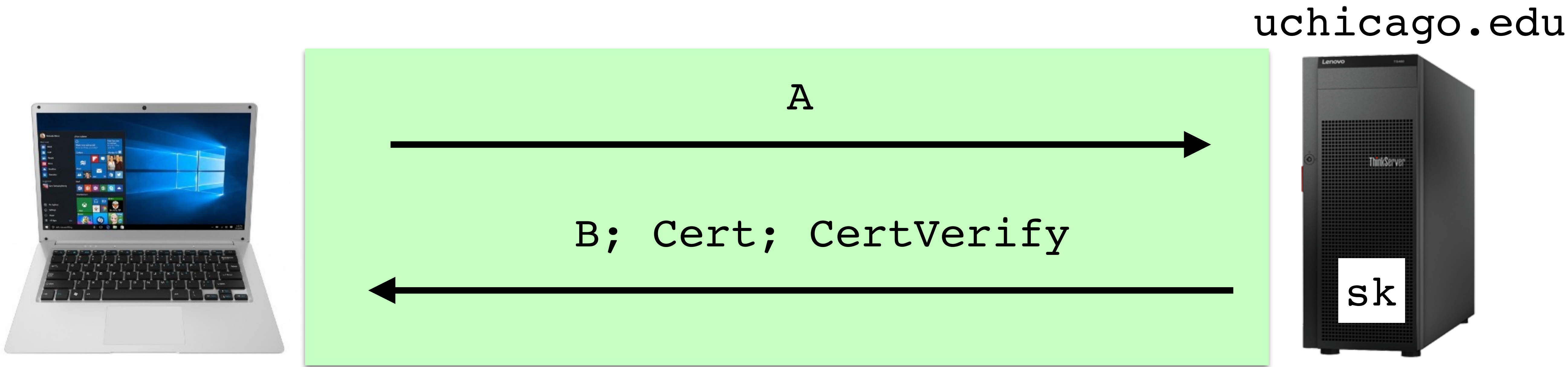
…

$B \leftarrow g^b$

$K \leftarrow \mathrm{Hash}(A^b)$

- Template can be secure against passive adversaries.
- But template pictured provides no authentication.

Authentication with Certificates (="Certs")



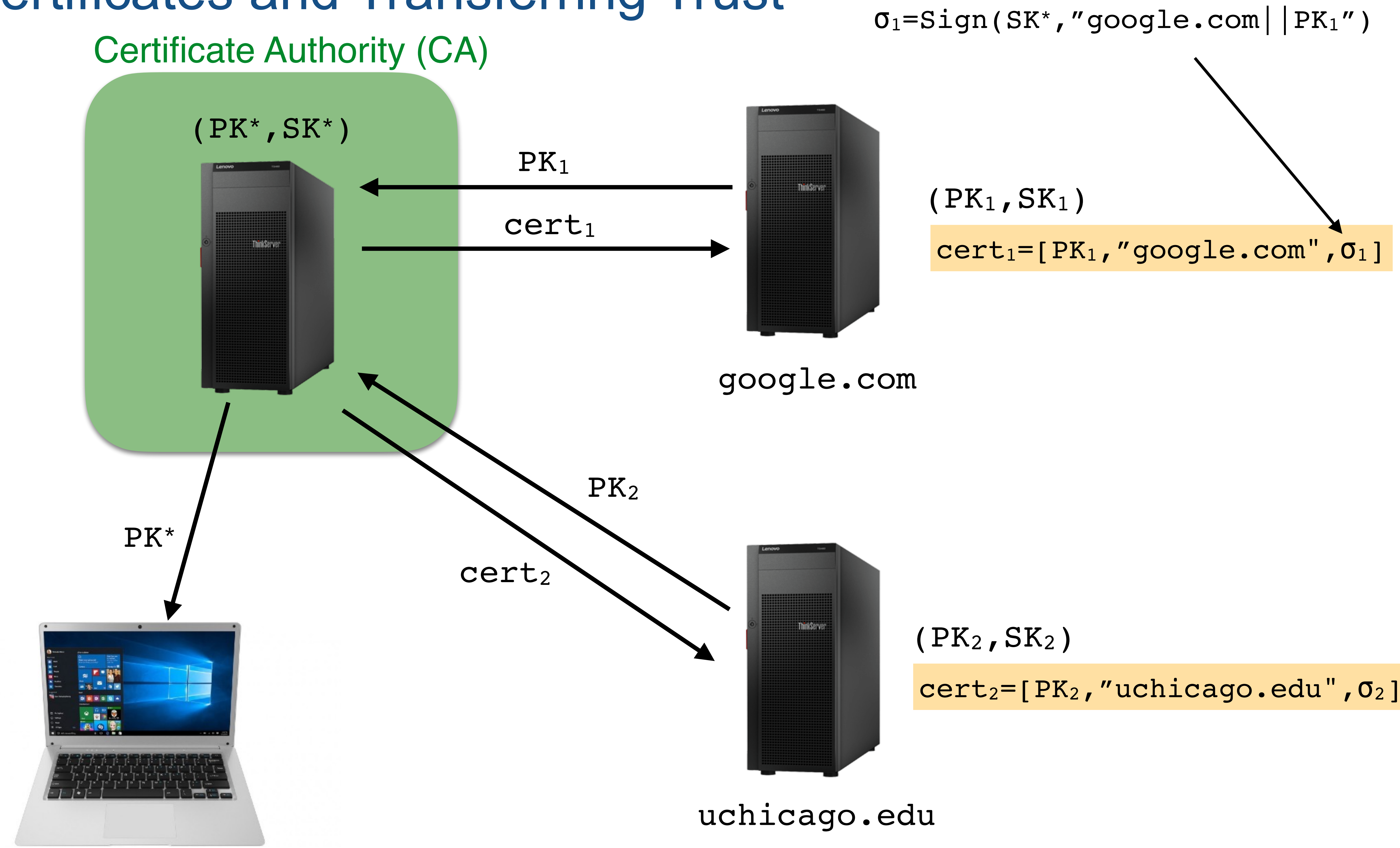
- `Cert` is a document saying
  "The public key of **`uchicago.edu`** is **`pk=0x7b5532…`**"

- `CertVerify` is signature of handshake that verifies under **`pk`**:
  `CertVerify=Sign(sk,handshake)`

- Randomness ensures transcript changes each run.

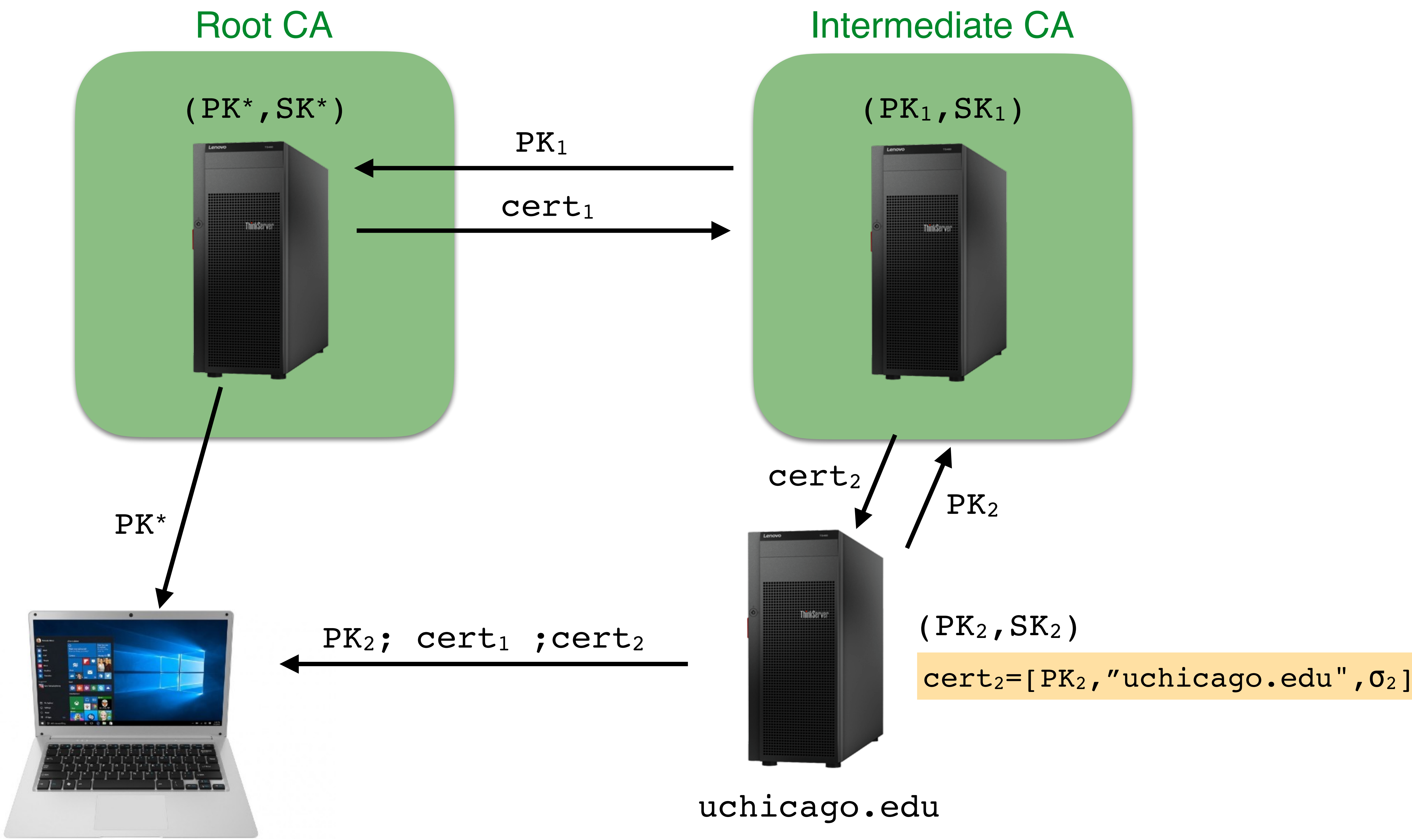
- Many, many details omitted.

Certificates and Transferring Trust



- Trusted CA “issues certs”, i.e. signs public keys of other orgs.

Intermediate CAs and Cert Chains



$\mathtt{PK^*}$ bound to root ⇒ $\mathtt{PK_1}$ bound to CA ⇒ $\mathtt{PK_2}$ bound to uchicago.edu

USERTrust RSA Certification Authority

↳ InCommon RSA Server CA

↳ *.uchicago.edu

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Public Key 256 bytes : CA E9 01 25 77 E9 74 B8 CB F7 99 DA D6 87 79 35 D7 31 CA D7 83 11 83 32 FA FA 43 CC C8 85 7B 76 EF 79 BB 4B 8B E0 35 87 EE A4 34 17 DC 5A 0D 5A 04 D3 F1 BA E7 98 9F 49 FC D5 B9 2C FB C8 DD 36 47 4D 07 FE 41 11 75 B0 42 F7 6D 40 4C BF F5 B6 C7 FE 05 0D DE 3B 7C E9 9F 6A 1C 1C 89 2E AA E8 F5 E3 5B 04 55 16 B0 48 92 C7 F9 37 11 89 F8 C5 85 C1 24 96 71 6F 78 B6 6B 35 39 92 8C EF 17 91 D1 97 D7 EF 93 6E 95 F1 EE C6 0D 5A EA 39 C6 4E 33 E2 CA F2 9A 41 F4 A2 41 9C E8 EA 46 FB EF 71 C0 A6 D3 C6 A5 94 81 4B 12 5E 80 63 87 7C 2F A6 8A A5 9A 31 9E 81 63 7F 0F 26 25 B6 6D 62 C2 AD B4 E7 68 FD C9 F8 86 2C 3F F8 E1 59 F3 3E 73 08 DF 6C 92 98 21 D2 AD EF 23 E7 33 A2 D4 5E 67 74 E3 AB 08 DF 15 31 9A 9D 3B 36 7D 6B 77 48 60 17 A4 10 F3 17 77 53 E0 21 D9 F9 A4 12 0F 39 DA D1

Exponent 65537

Key Size 2,048 bits

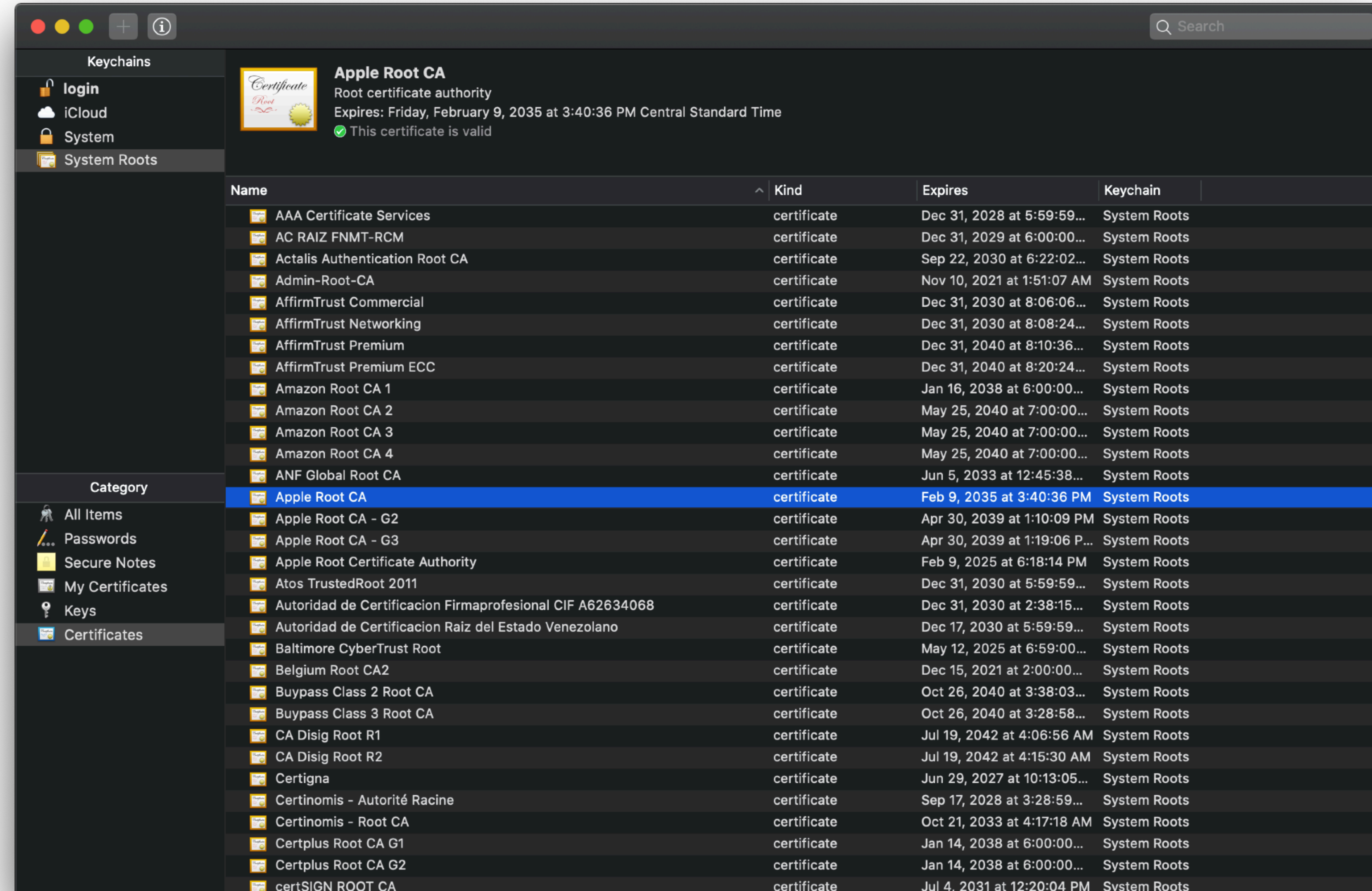
Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes : 11 F9 F9 6D C6 92 D1 B9 E7 13 E6 0D BA E6 19 65 BB 16 4B DE E1 C2 3A 62 55 D1 61 80 93 F0 2A B2 7D 9E 76 CE 10 4A D6 96 4E 5C 00 5D BD 8C 83 74 CF C1 14 91 2B 15 4B 2D 67 4A 84 A2 A4 54 7A B1 C9 8E F5 A7 93 8D 30 BF 0C 9B EF 98 36 D6 4B BD B6 11 63 C2 51 23 71 7B 8D 4C 9B B7 AD A9 FE A8 4E 48 B2 83 A1 36 75 97 2B 36 4A 72 C4 AA C6 B6 A8 4A C0 F4 37 BD 0E 85 B1 A8 FB EC B6 B5 BB A8 C2 C0 BB B7 47 D7 D4 DB 05 80 72 BA CB C7 79 81 63 CC 55 D7 68 9C 41 2B E7 D9 F0 C2 8F 11 15 7D C5 D5 34 27 5C 7C B5 D9 A8 3F 3C DF C5 1D AA 52 03 19 AE 5B FC FF 42 68 15 A3 01 CB F8 0E FE 9B A1 76 B8 43 1C 6B 9C 57 38 87 81 3B 4A 33 98 09 CF 25 F4 75 34 AE 1E 7B CD 0F EF A0 4C 5B 92 B7 F1 FD 66 1B 49 67 B0 65 5A 90 1D 1D 54 D2 CF FF FD 07 DC 7A 88 56 51 55 16 7F 83 D4 FC 19 F4 28

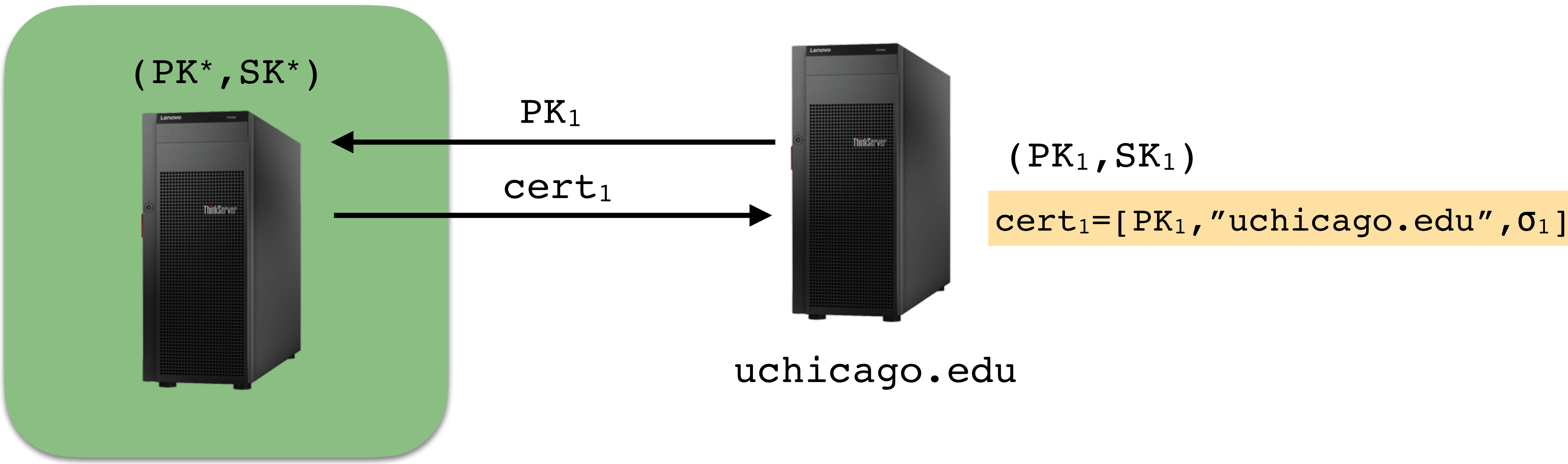
X.509 Certificates Include

- Serial number
- CA info (public key, name, etc)
- Common name of subject
- Public key of subject
- Expiration date
- Supported protocols
- Extensions (possibly many)

Root Certificates



Issuing Certificates: Validation

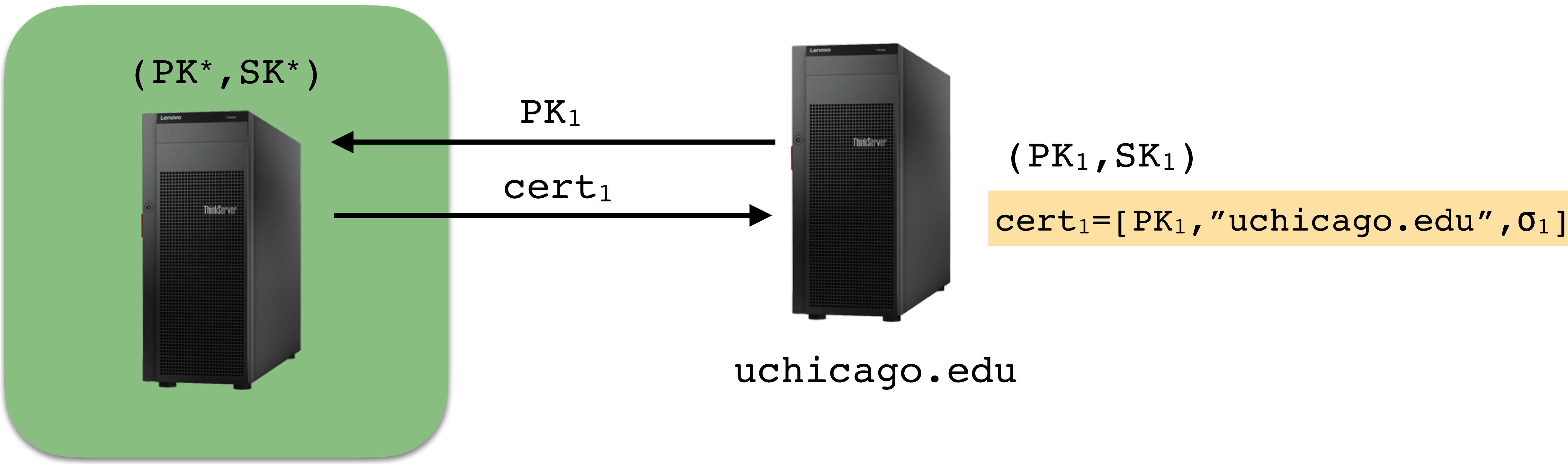


- CA must check that key really does to "`google.com`"

Domain Validation (DV): Check that party with that key can control domain.

Org. Validation (OV) and **Extended Validation (EV)**: Also check company name, location etc via public records.

ACME Protocol by Let’s Encrypt (Assignment 5 or 6)



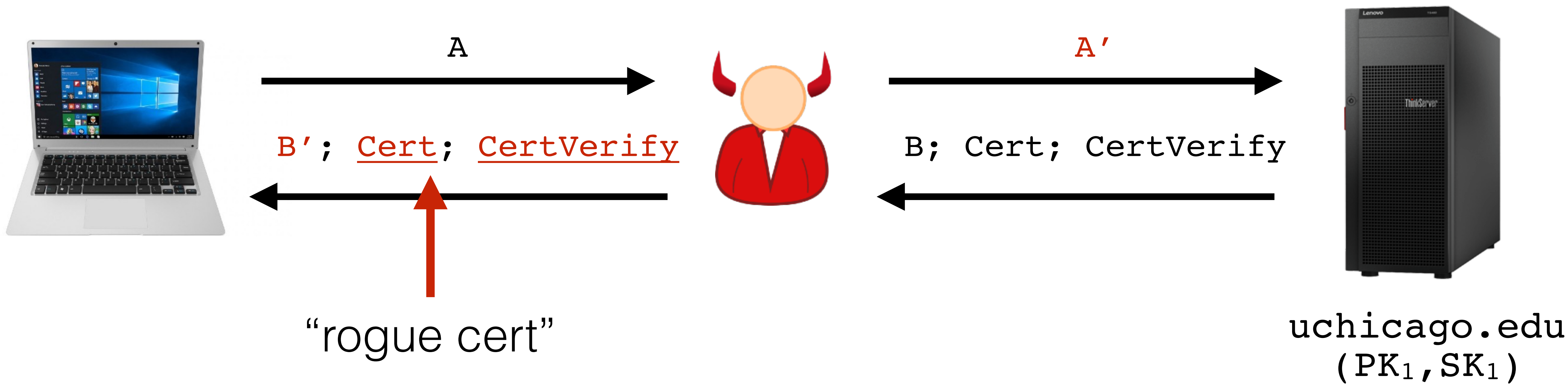
1. Requestor submits public key and request to CA
2. CA gives a challenge to requestor
3. Requestor places challenge on web server, proving ownership
4. CA then issues cert

- For wildcard certs (`*.uchicago.edu`) similar protocol used, but with DNS server… why?

Let’s Encrypt

certbot

What if you had "valid" cert for uchicago.edu?



- "Machine-in-the-middle" can read/change all traffic undetected
- Needs access to network path (or DNS)

CA Security

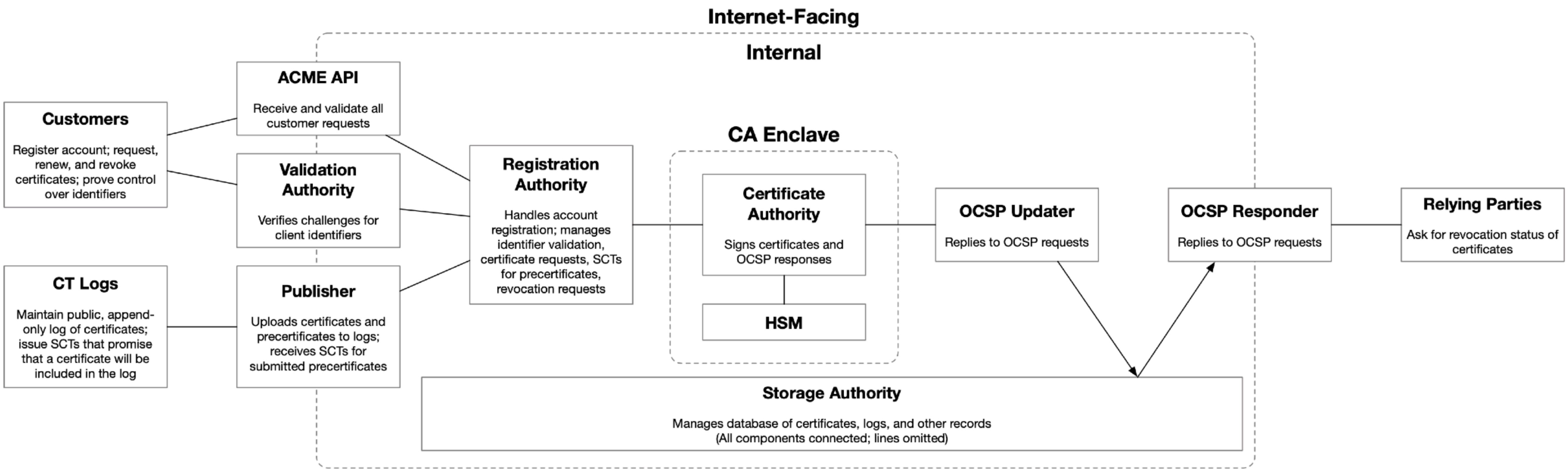


Figure 3: Boulder architecture. Let's Encrypt developed and operates a Go-based open-source CA software platform named Boulder, which is composed of single-purpose components that communicate over gRPC, as illustrated here. The certificate lifecycle unfolds roughly from left to right in the diagram.

[Aas et al, 2019]

CA Security Incidents

- 2011, Root CA Comodo: Login credentials stolen. Hacker issues certs for mail.google.com, login.live.com, www.google.com, login.yahoo.com…
- 2011, Root CA DigiNotar: Hacker issues rogue cert for *.google.com, others. Used to PitM by Iranian government.
- 2013, Root CA TurkTrust: Accidentally issues intermediate CA cert, used to issue gmail.com cert.
- …
- 2019, Root CA Comodo: Pushes email login credentials to public GitHub repo…

(Slide inspiration: Dan Boneh)

Countermeasure: Public-Key Pinning

- Site can tell client to only accept certs from certain CAs
- Helped discover some rogue certs from previous slide

- But… if site gets hacked… attacker can pin a malicious cert!
- Deprecated now.

Countermeasure: Revocation

- Blacklist certificates so they are no longer accepted

Mass Revocation: Millions of certificates revoked by Apple, Google & GoDaddy

The DarkMatter debate is already having industry-wide ramifications

Millions of SSL/TLS certificates – among other digital certificates – are being revoked right now as a result of an operational error that caused the generation of non-compliant serial numbers.

March 3, 2020 0

Let’s Encrypt to Revoke 3 Million SSL Certificates on March 4

The world’s leading free SSL provider announces that millions of certificates are being revoked due to a bug they discovered days ago – giving subscribers potentially only hours to respond

Certification Revocation: Cert. Revocation Lists (CRLs)

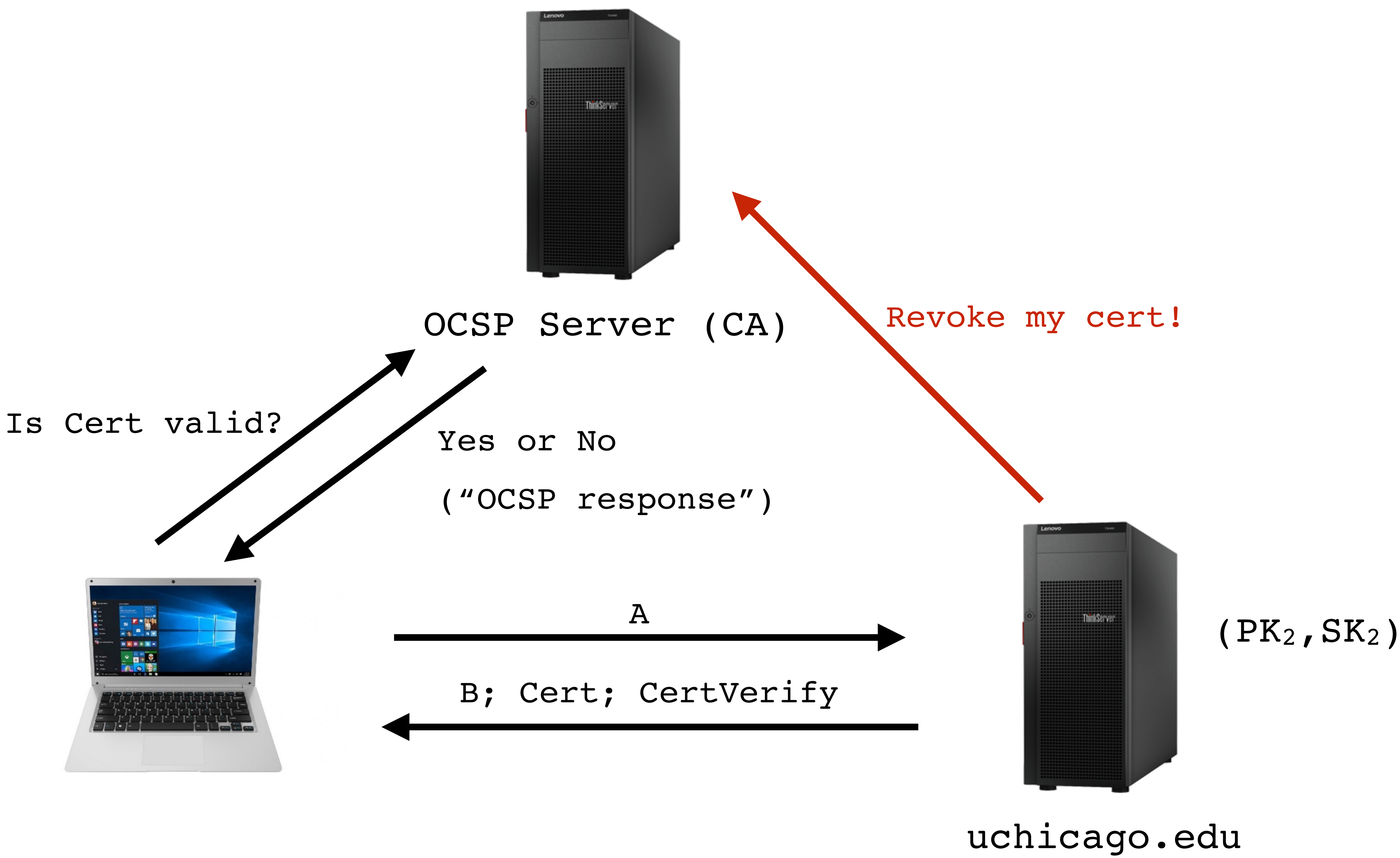
CA's CRL Server

`(PK*,SK*)`

- CA provides list of revoked certs
- List will get big, hard to keep current

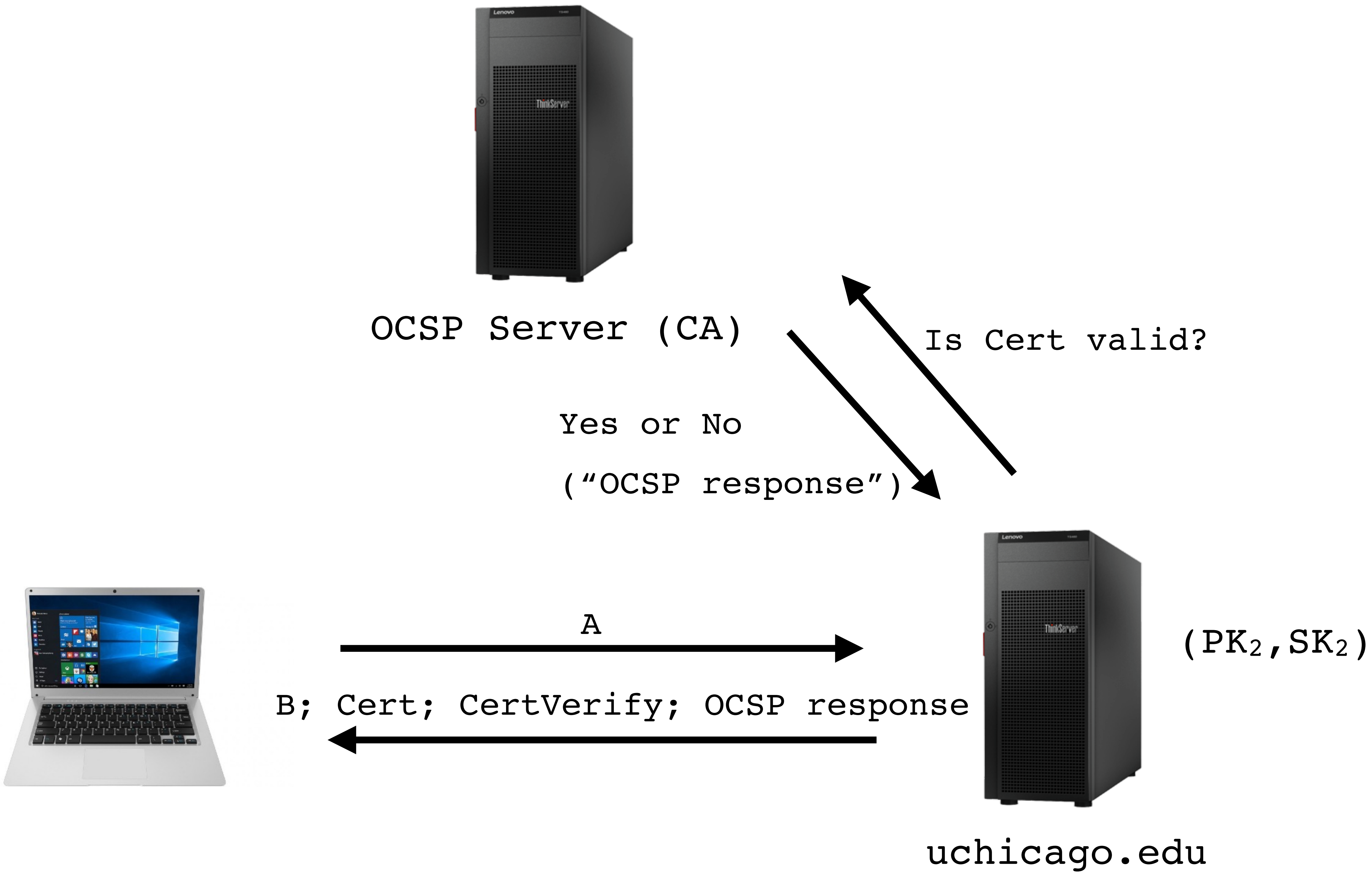
```
Revoked serial numbers:
09823342365
23423482349
98072344456
...
```

Online Certificate Status Protocol (OCSP)



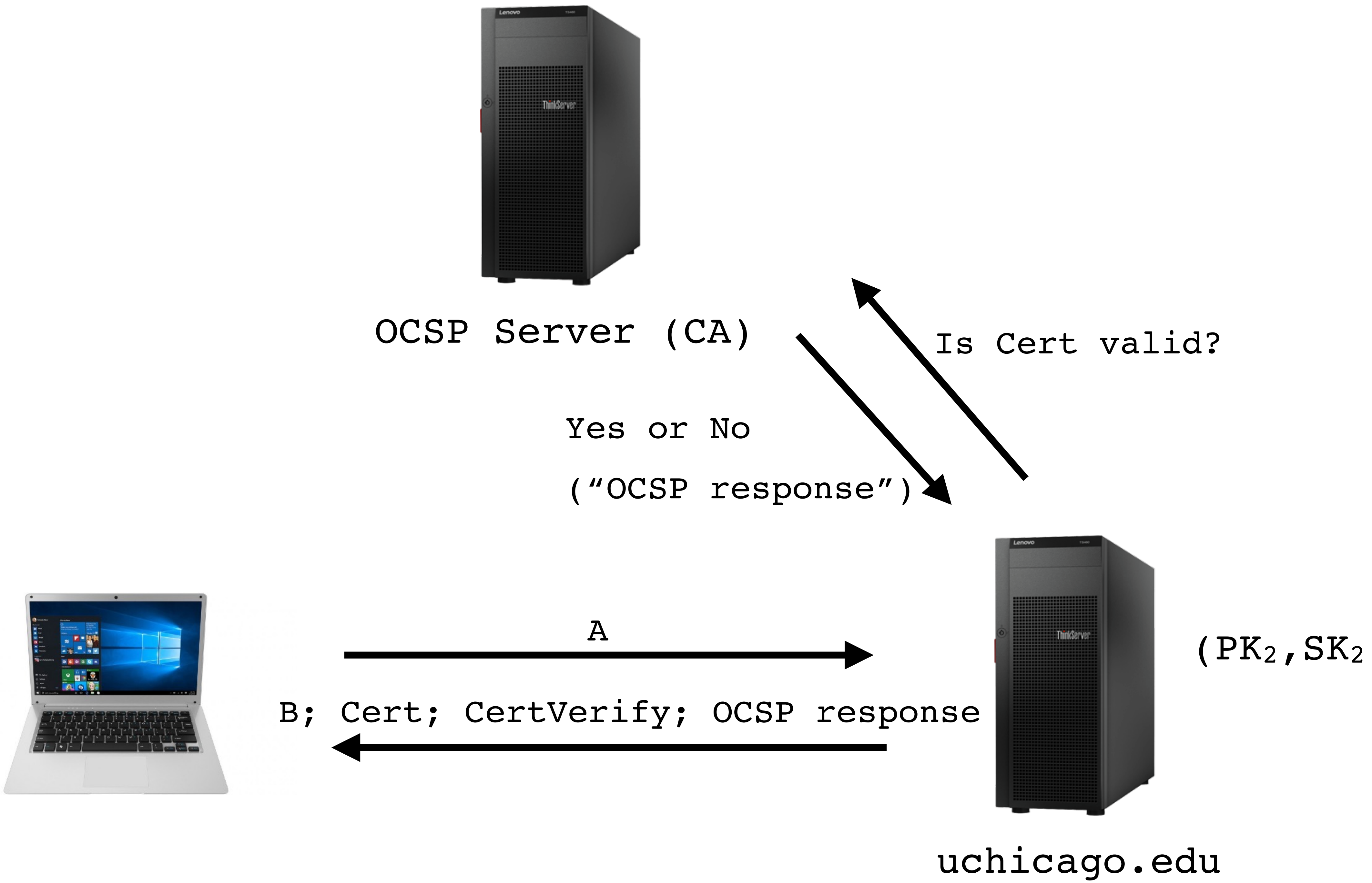
- Add another server to connect to, slowing connection
- What if OCSP server times out?
- Privacy problem?

OCSP Stapling



- TLS Extension that allows for OCSP response to be included with cert
- Client checks CA signature and time-stamp on response (~hours old).
- Certs can have "must staple" extension.

OCSP Stapling

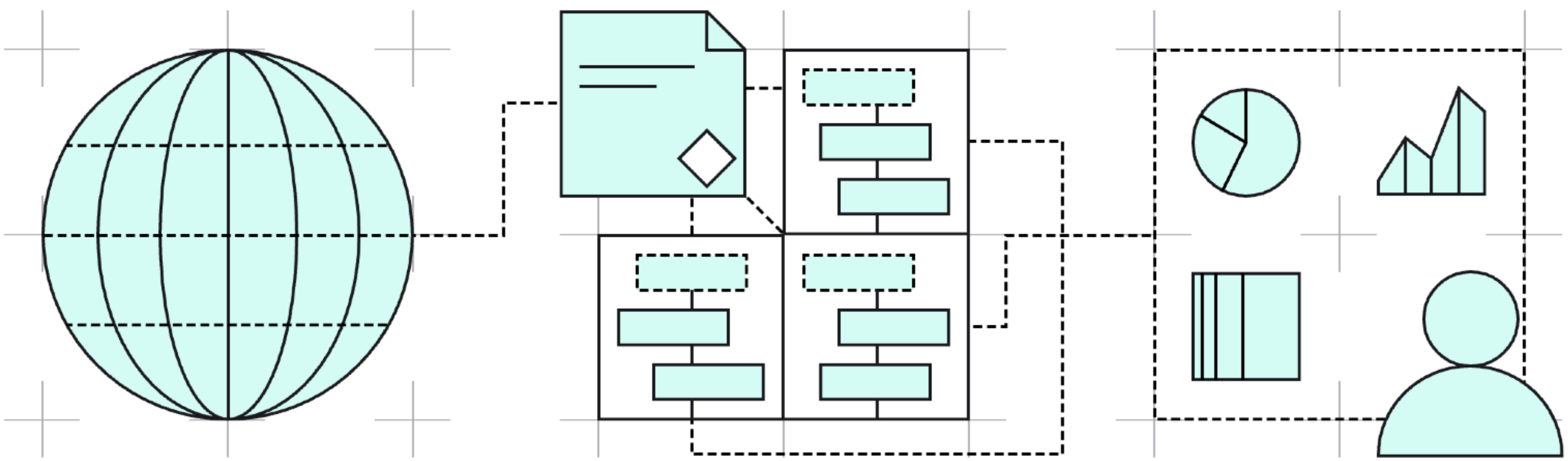


- Problems?
  - OCSP server goes down => uchicago.edu goes down (or ignore)

Certificate Transparency (CT)

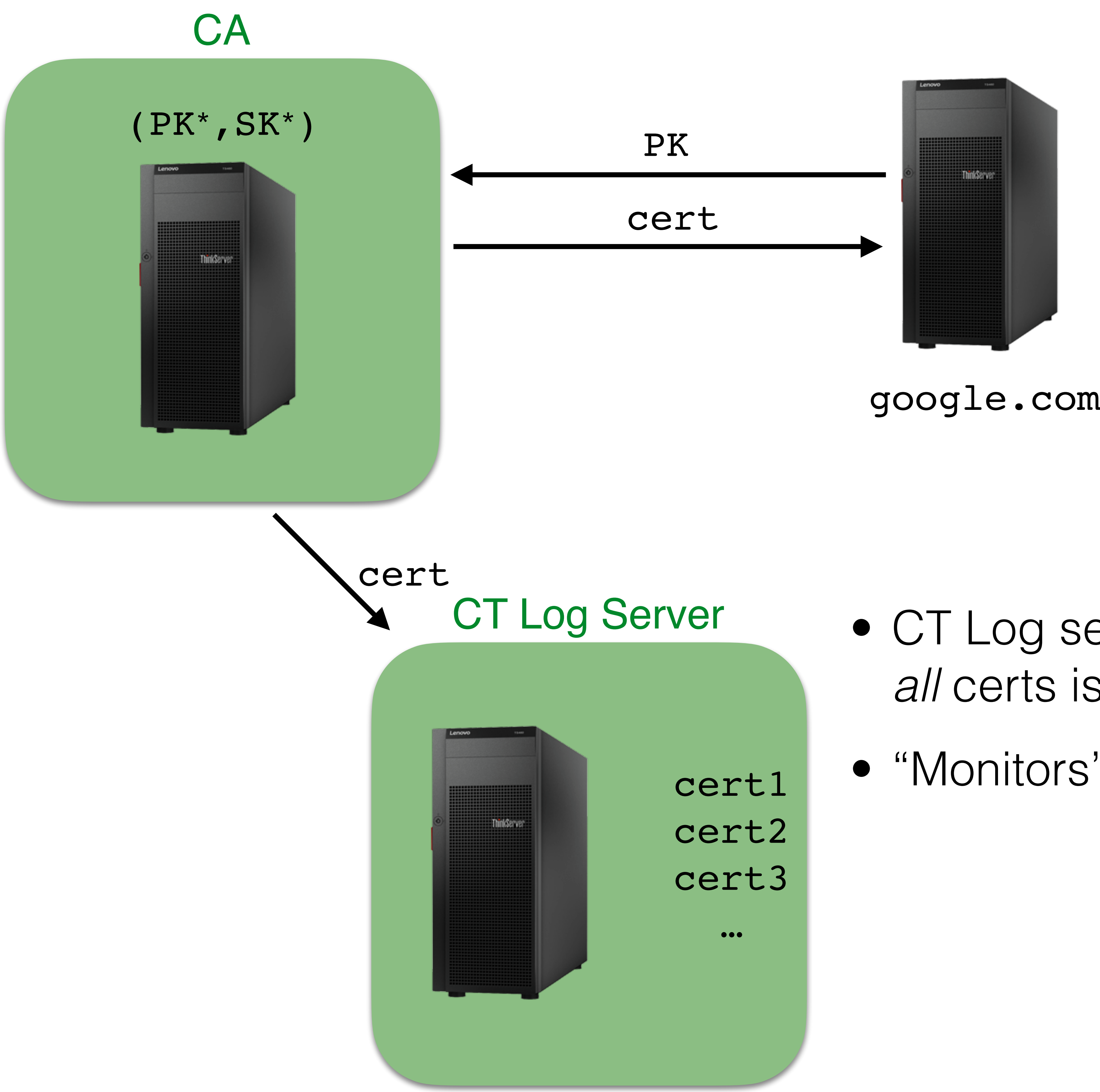
Certificate Transparency   How CT works   Actors ▼   Our story   GitHub

Working together to detect maliciously or mistakenly issued certificates.



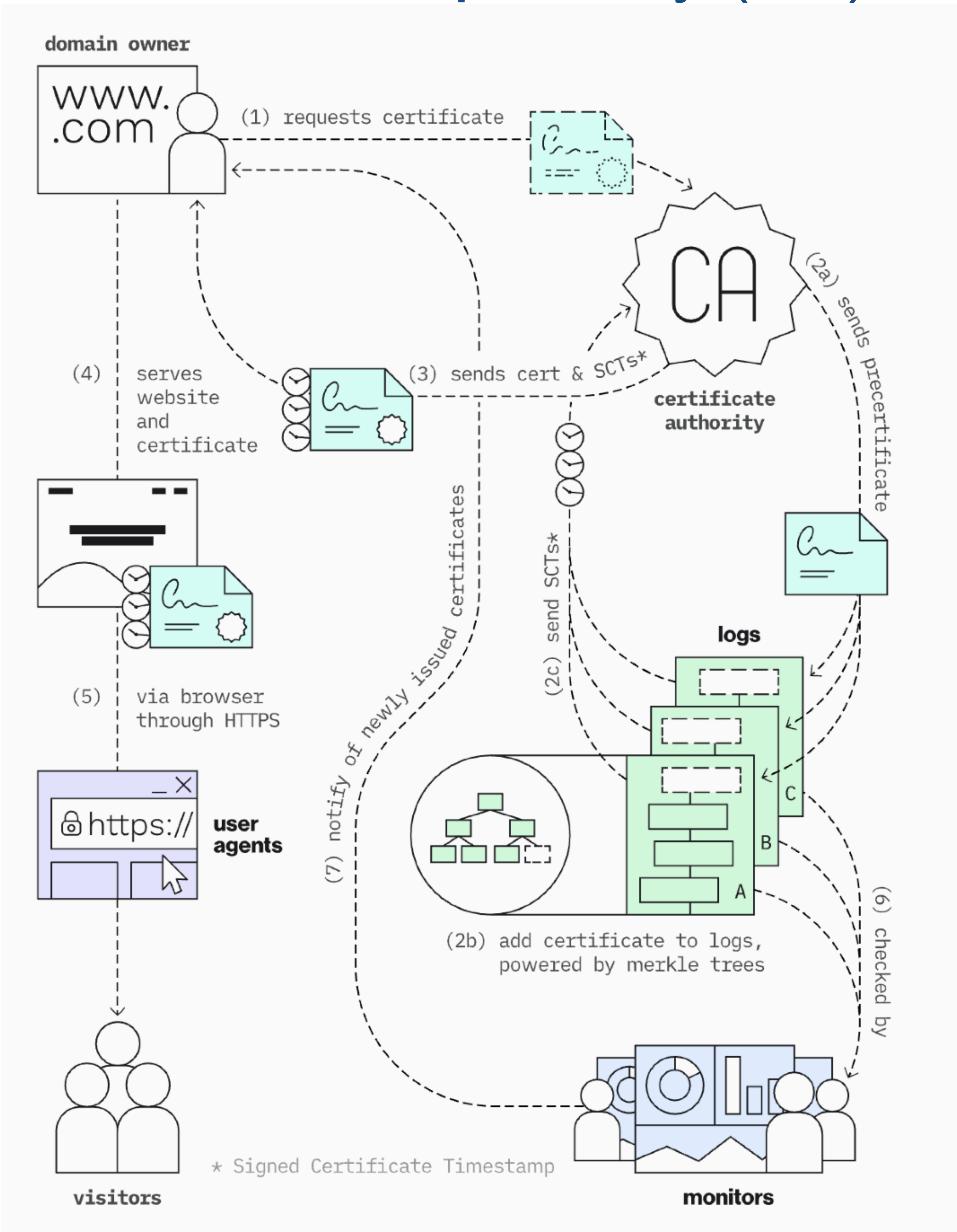
An ecosystem that makes the issuance of website certificates transparent and verifiable.

Certificate Transparency (CT)



- CT Log server maintains a list of *all* certs issued by CA(s).
- “Monitors” check for improper certs.

Certificate Transparency (CT)

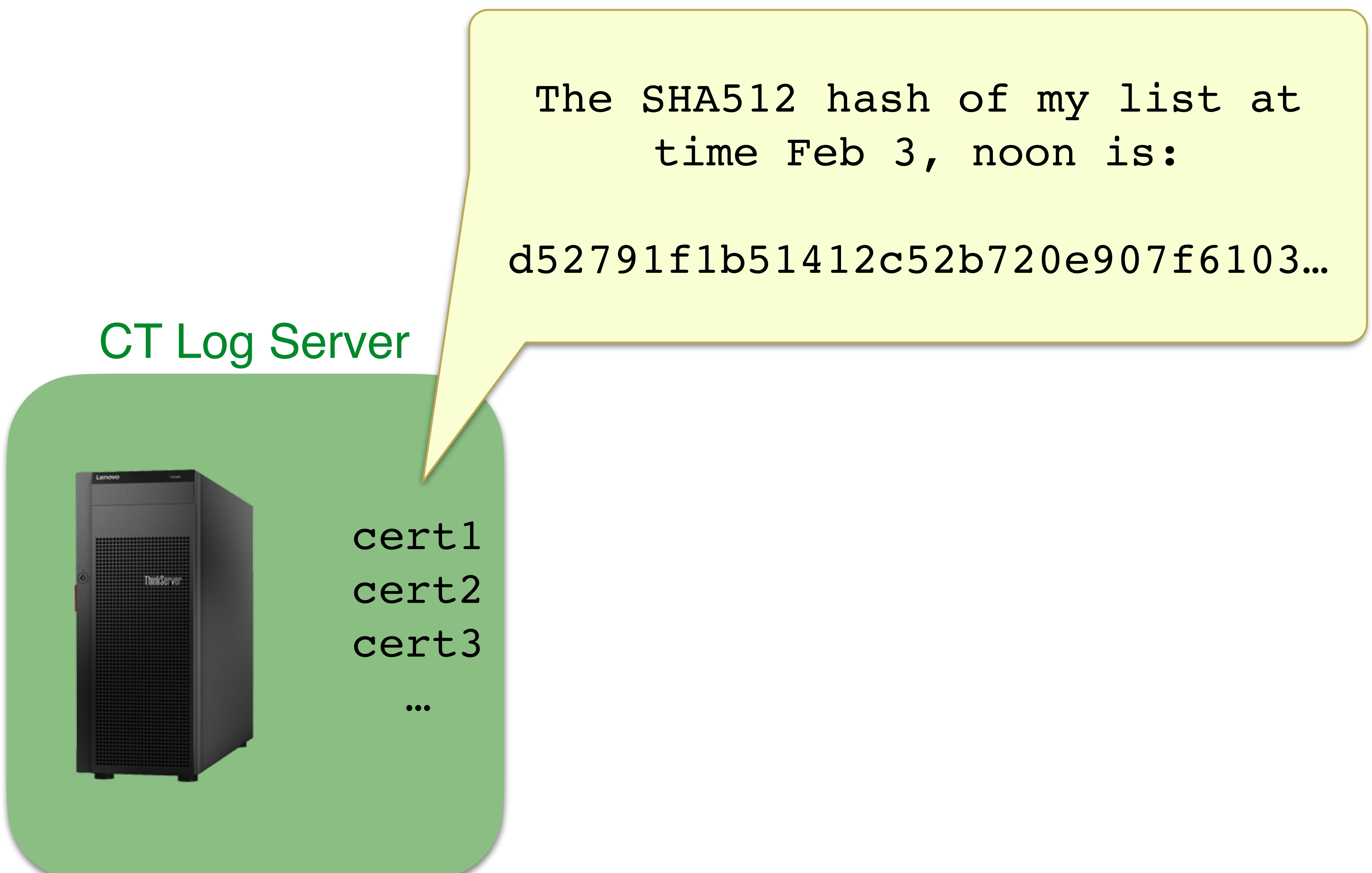


(https://certificate.transparency.dev/howctworks/)

- How do CT and OCSP compare?
  - What problems do they solve?

Challenges with CT

- List is huuuuge
- Trust the CT Log?
- Who checks?
- Privacy?



Yet More Problems with Certs…

[Moxie'2009]

- Step 1: Blase requests cert for domain

`google.com\0.blaseur.com`

- Step 2: Blase can validate this domain; He owns **`blaseur.com`**
- Step 3: Blase MitM's a victim, and presents his cert as a **`google.com`** cert
- Result: Browser runs

```
strcmp("google.com", "google.com\0.blaseur.com")
```

which returns 1, accepting cert.

Back to TLS

uchicago.edu

A

B; Cert; CertVerify

<encrypted data>

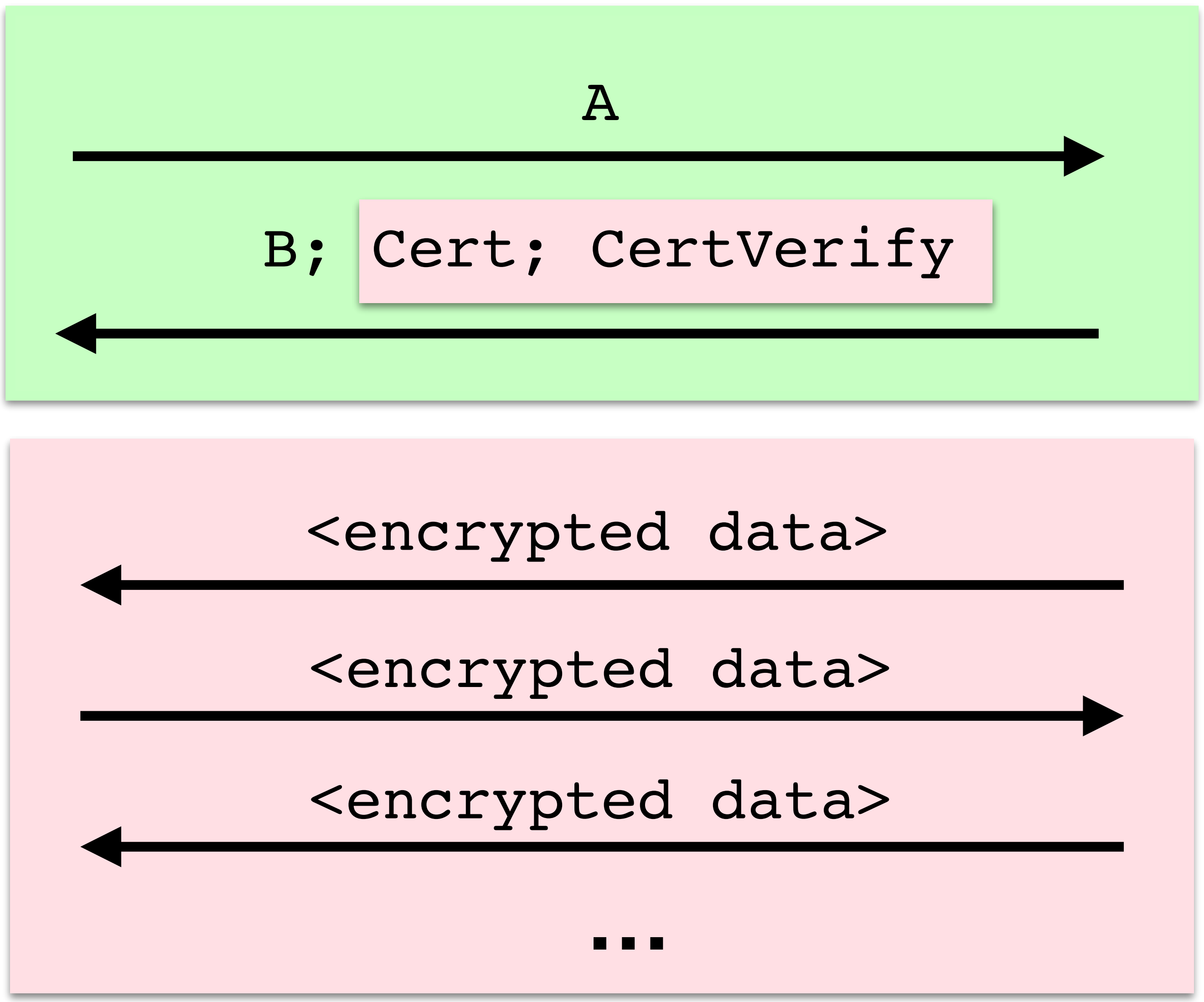
<encrypted data>

<encrypted data>

...

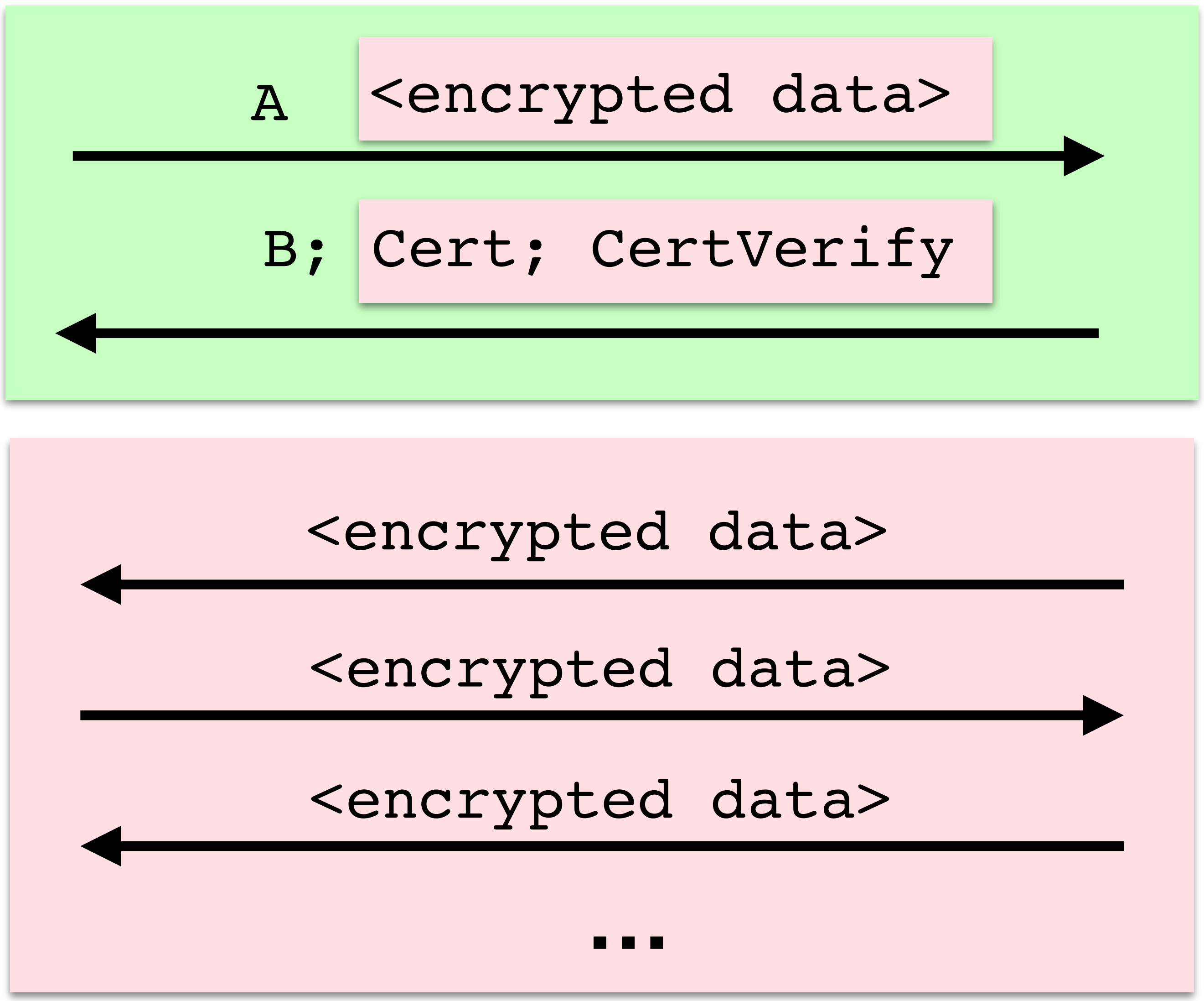
Some Changes from TLS 1.2 to TLS 1.3

1. Disallowed old crypto algorithms that caused problems ("ciphersuites")
2. Reduced number of round trips for handshake
3. Numerous hacks to make connections also valid TLS 1.2; Why?
4. Aggressively encrypts handshake: Why?

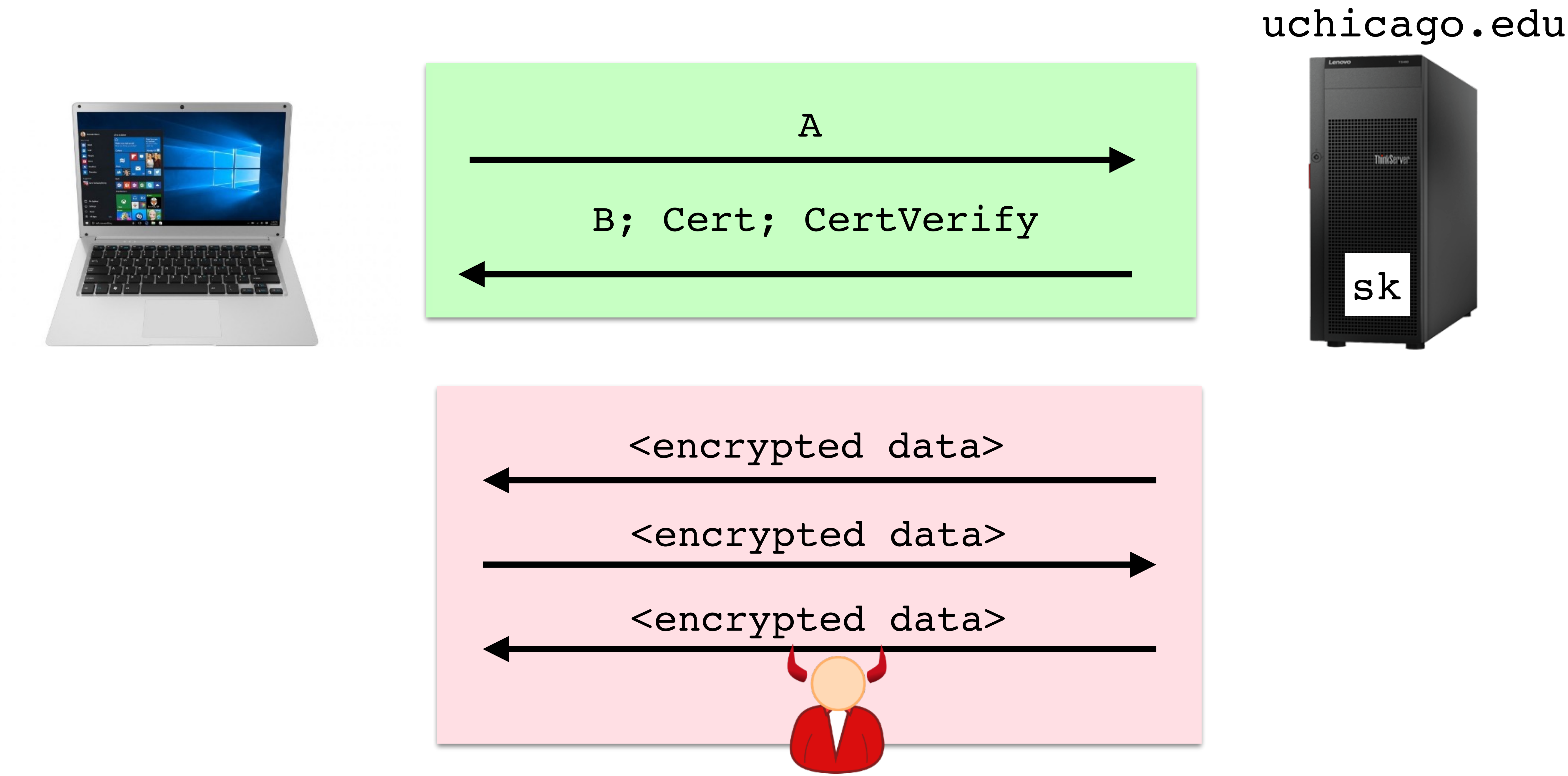


Interesting Addition to TLS 1.3: “0-RTT”

- When resuming old session, 0-RTT enables symmetric encryption on first message
- Security concerns?



Traffic Analysis (Assignment 4)



- Adversary can still see packet lengths through encryption.
- Can often reveal which pages are being loaded and other behavior.

Case Study: Netflix enables TLS

- [Reed, Kranch'2017]: Exploit video encoding timing and lengths to identify which video is being streamed

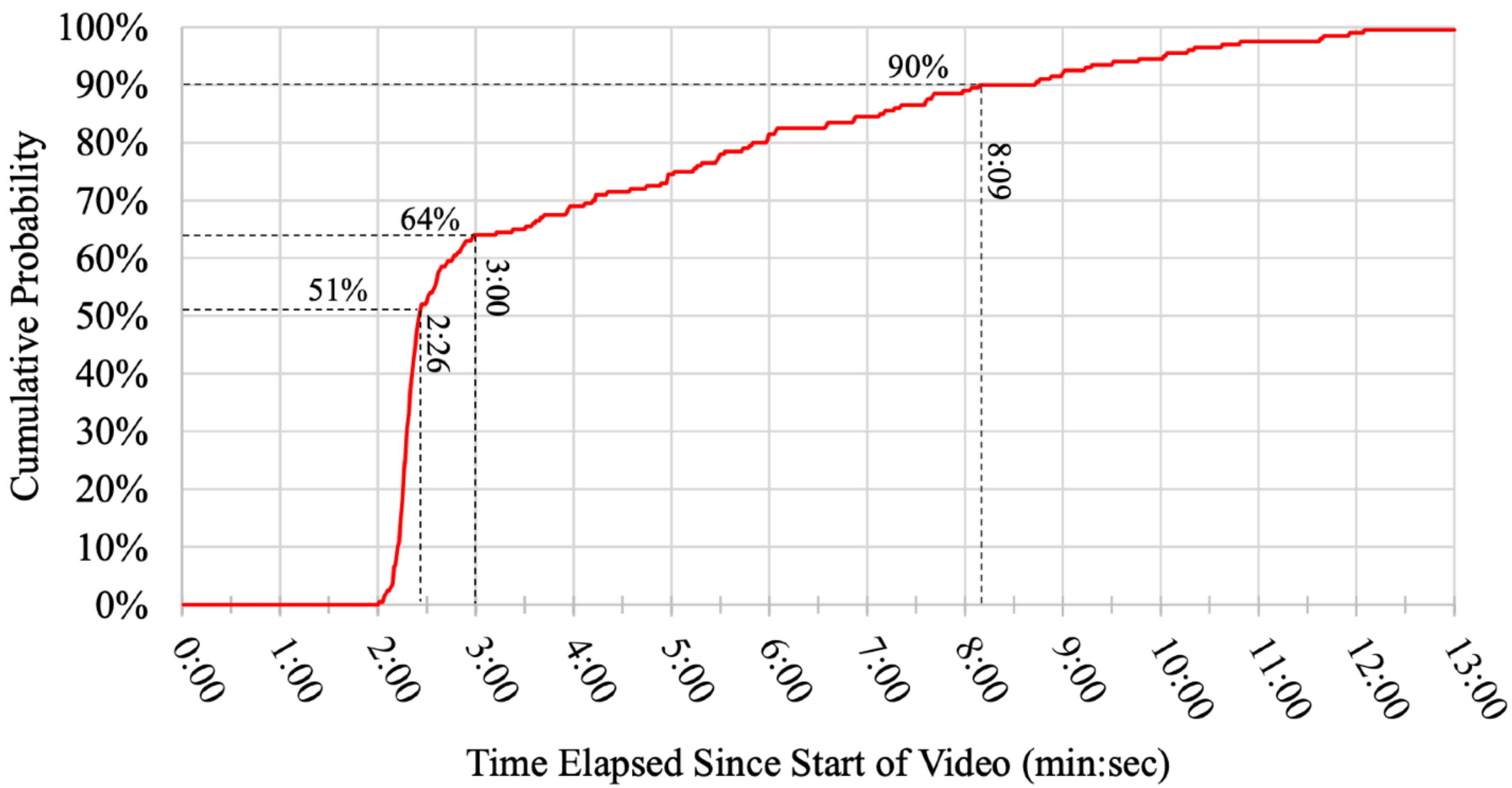


Figure 4: Cumulative probability of identifying a video before a specified amount of time has elapsed.

The End