DNS, Certificates, and TLS

CMSC 23200, Spring 2025, Lecture 9

Grant Ho

University of Chicago, 04/22/2025

(Slides adapted from Blasé Ur and David Cash)

Logistics

- Discussion Section resumes this Wednesday (04/23)

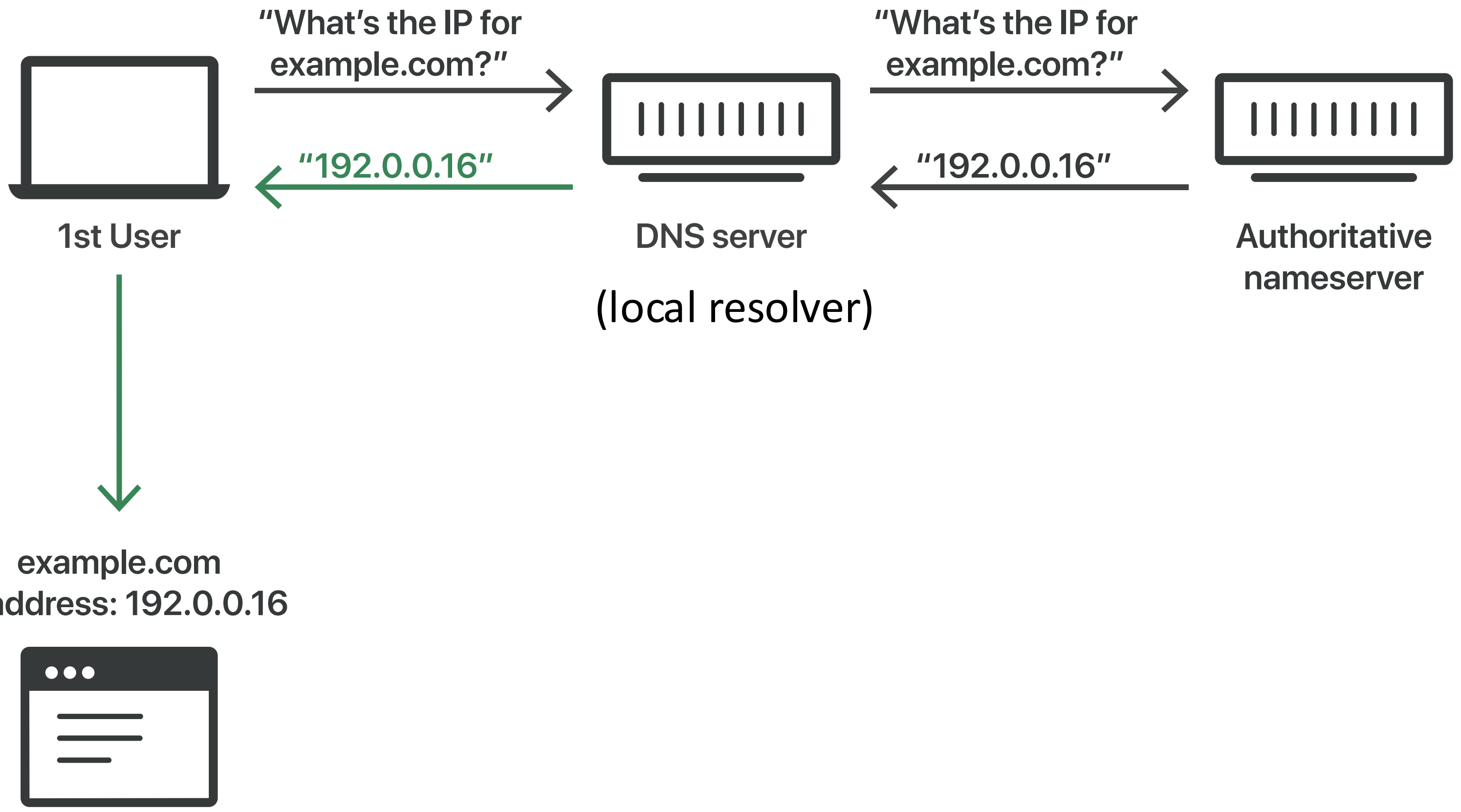
- Assignment 4 released on Friday (4/25)

- Assignment 1 grades released
  Regrade Requests open until Friday (4/25)

Outline

- Wrap-Up: DNS Security
- Secure Network Channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS

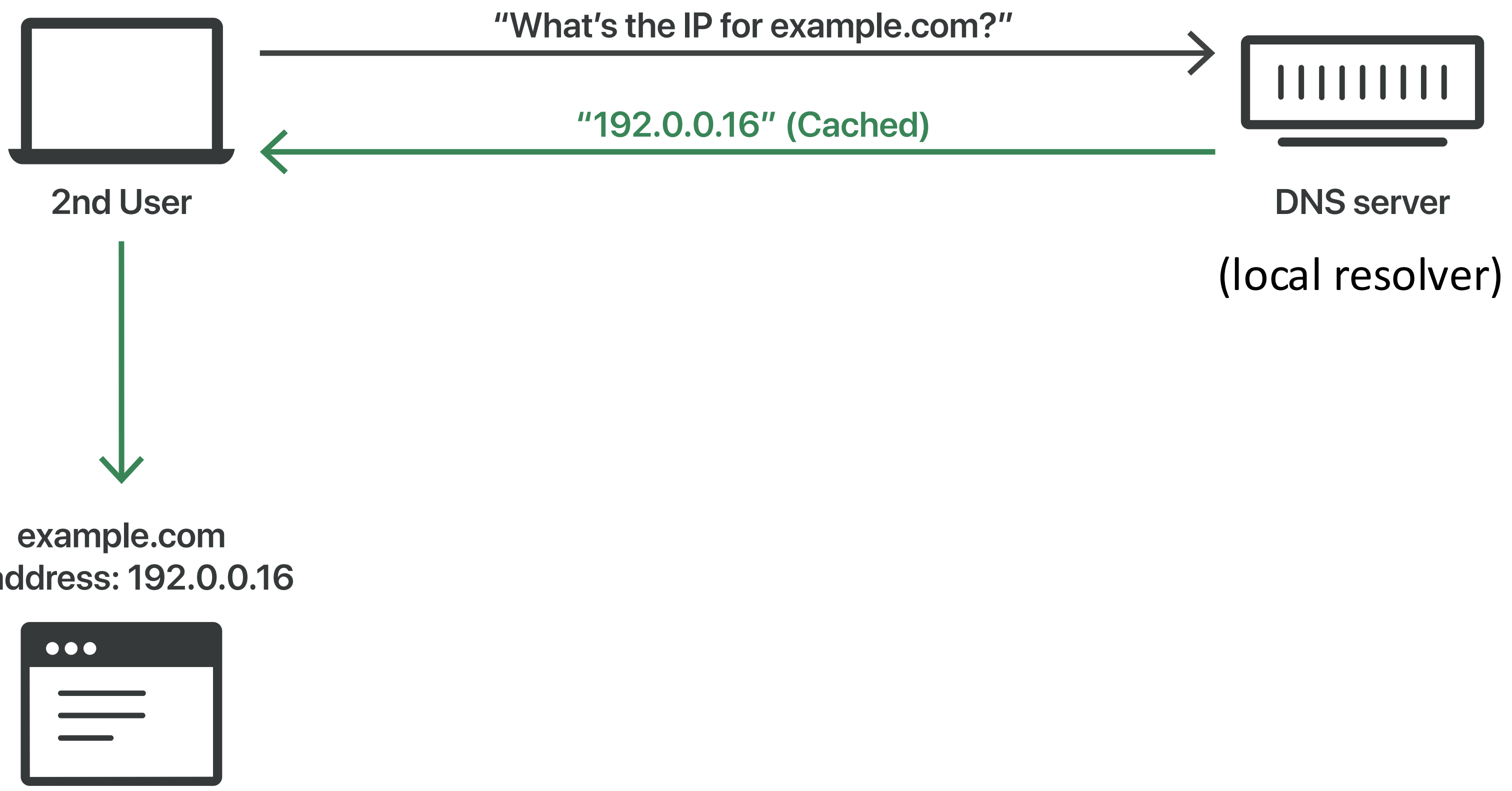
DNS (Uncached)



(local resolver)

Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

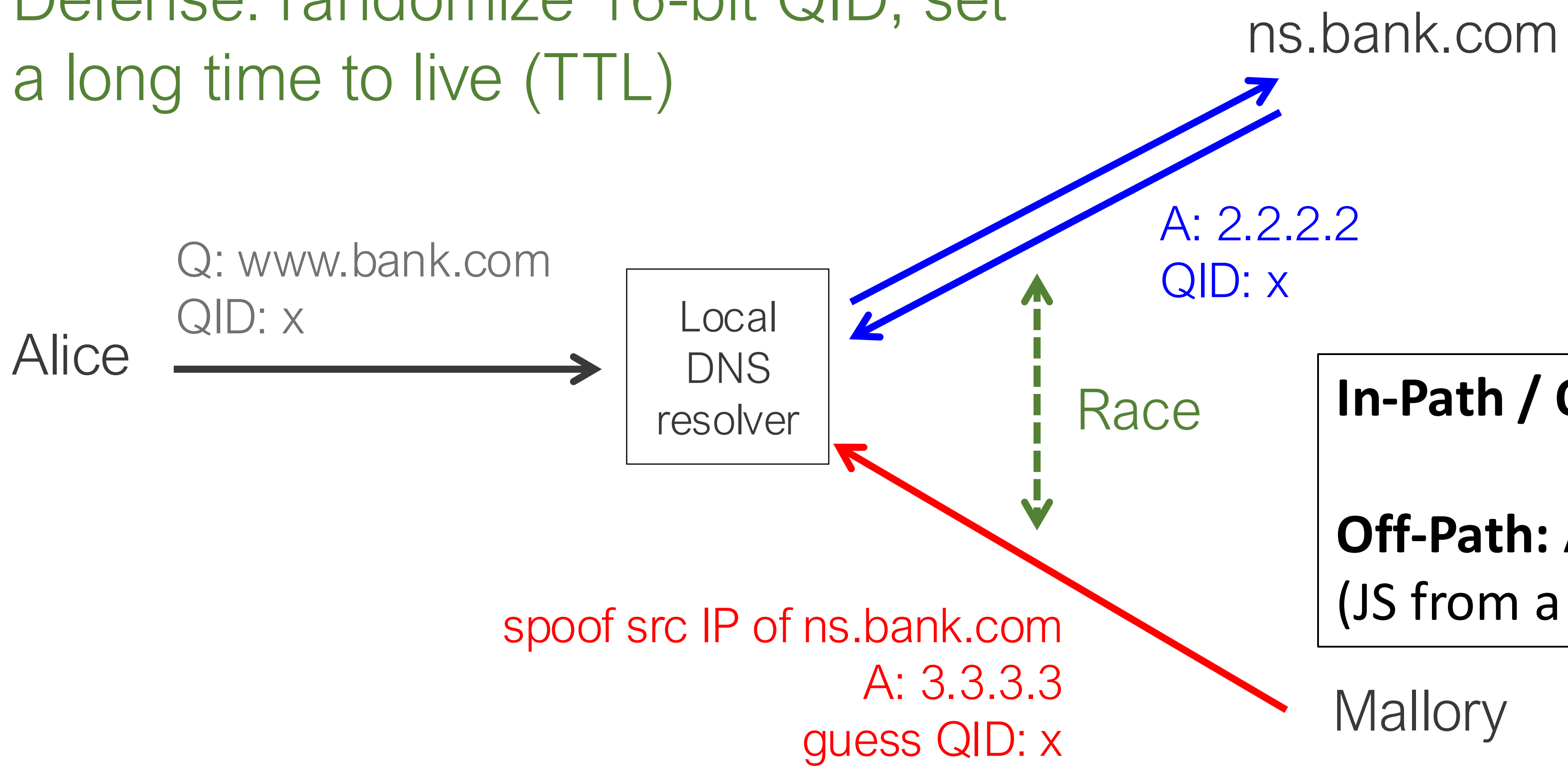
DNS (Cached, Benign)



Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

Cache Poisoning (DNS A Records: Single Name -> IP address)

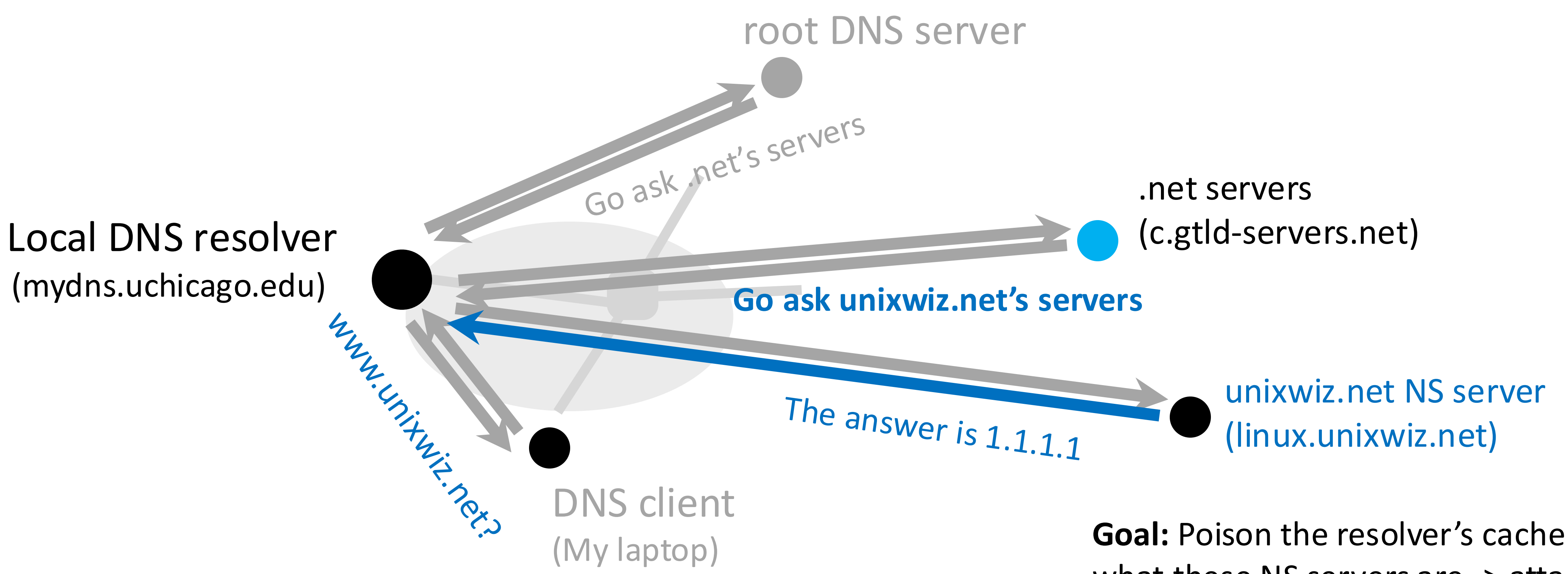
Defense: randomize 16-bit QID, set a long time to live (TTL)



In-Path / On-Path: Easy

Off-Path: Also possible!
(JS from a malicious website)

Cache Poisoning: DNS NS Records



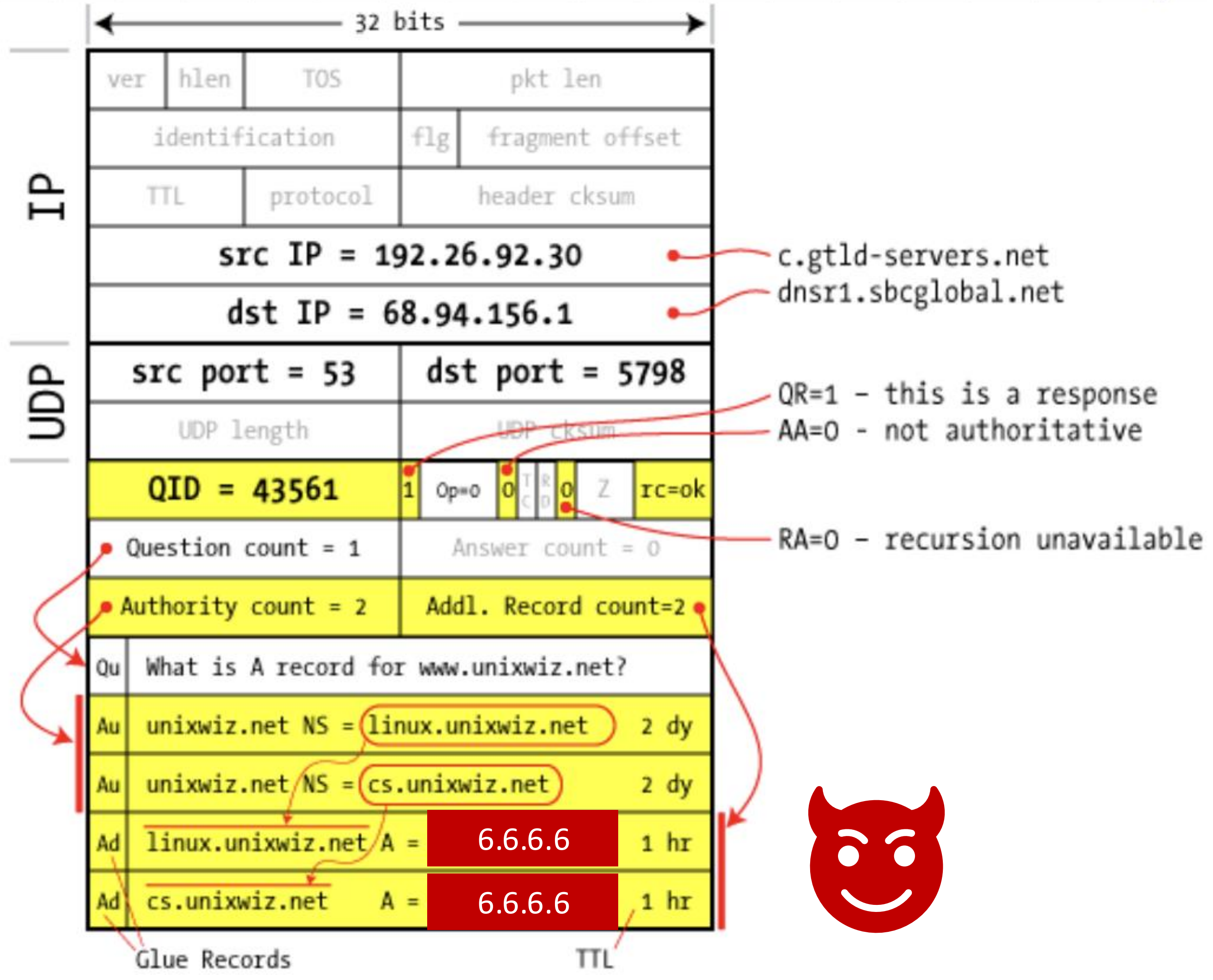
Goal: Poison the resolver's cache of what these NS servers are -> attacker's servers.
(Can control all subdomain IP mappings)

Response Packet w/ NS Info

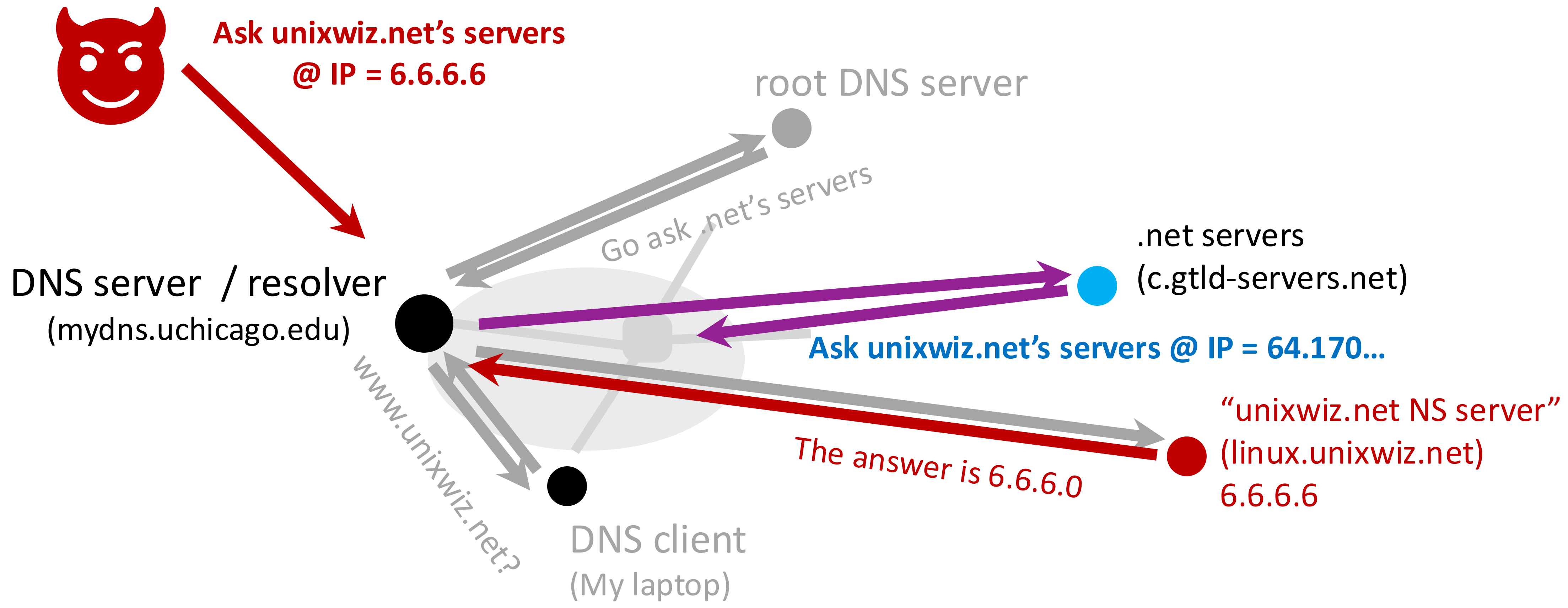
Response by the ".net" TLD nameserver to our local DNS resolver

Authority Section: Who are the name servers you should talk to next?

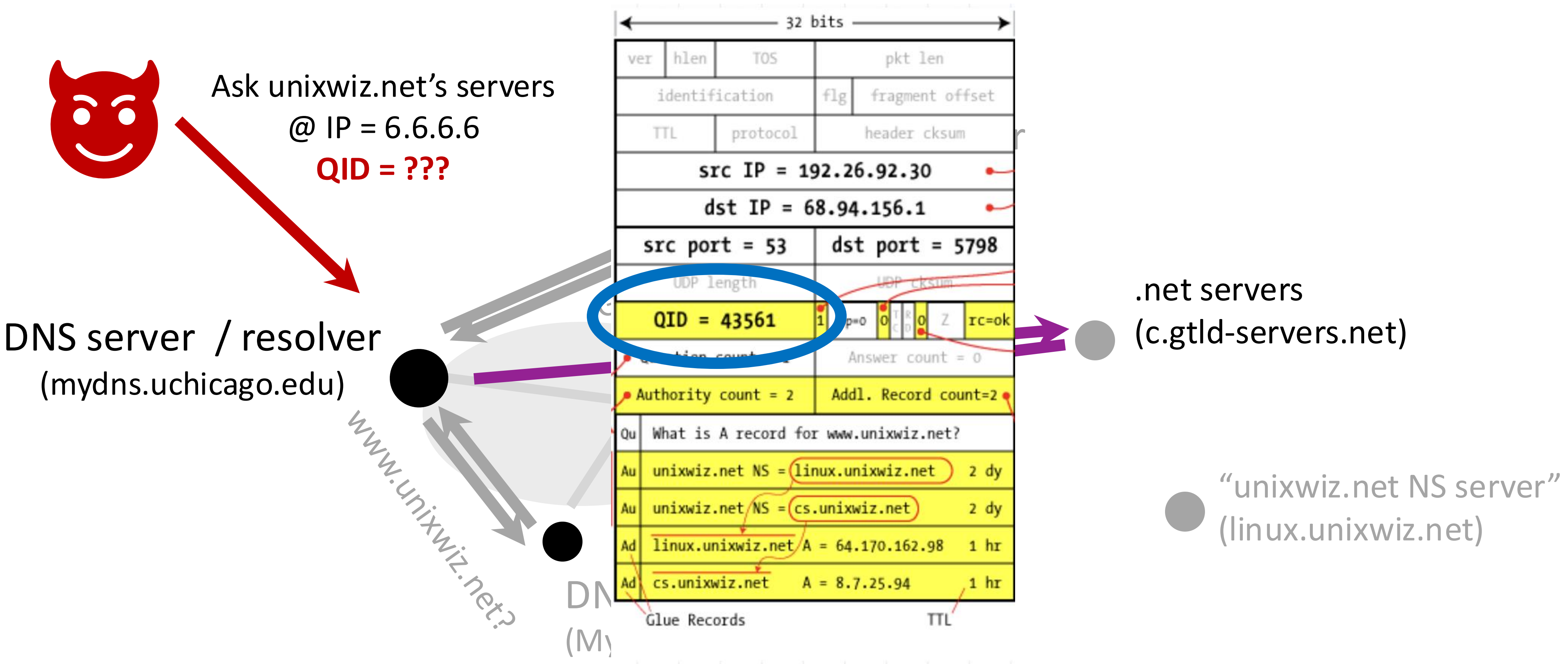
Additional Section [Glue Records]: What are their IP Addresses so you can go ask them?



DNS: Poisoning Authority (NS) Records



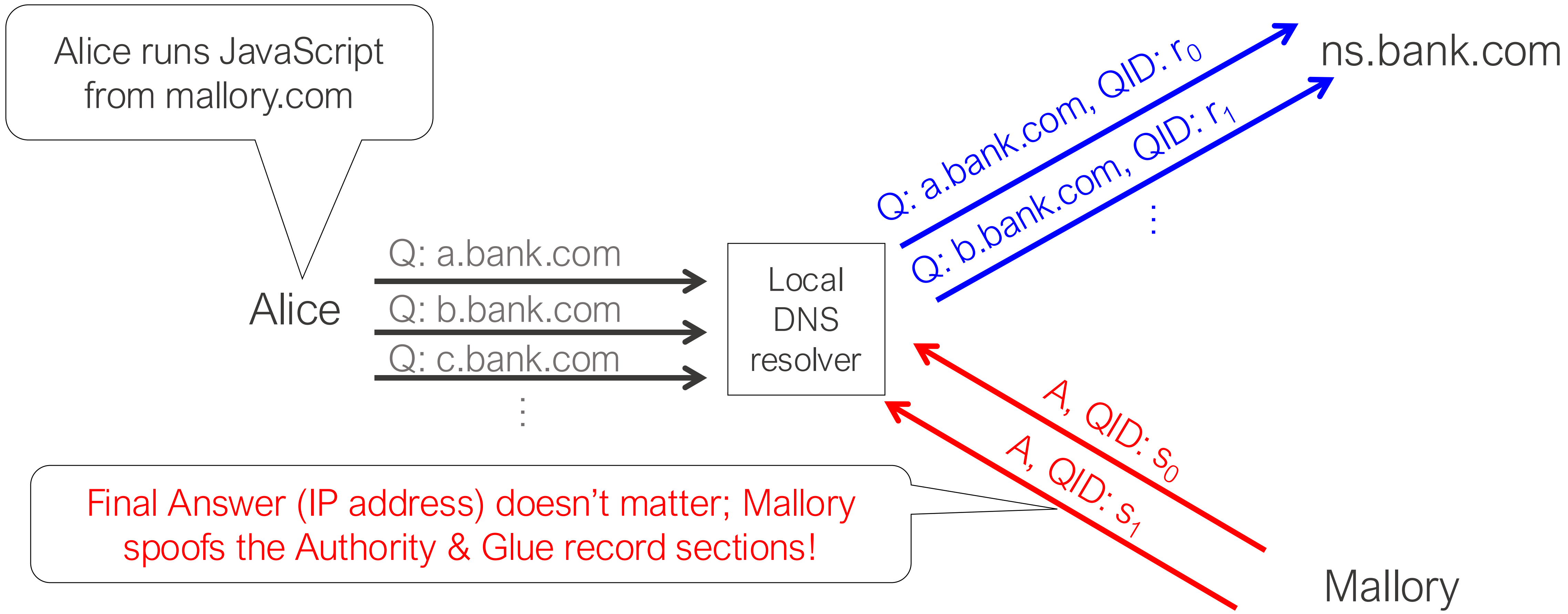
Kaminsky Attack (2008)



Challenge: Attacker needs to guess the correct Query ID.

Can an off-path attacker make this attack work?

Kaminsky Attack (2008)


Alice runs JavaScript from mallory.com
Alice
Q: a.bank.com
Q: b.bank.com
Q: c.bank.com
⋮
Local DNS resolver
Q: a.bank.com, QID: r_0
Q: b.bank.com, QID: r_1
⋮
ns.bank.com
A, QID: s_0
A, QID: s_1
Mallory
Final Answer (IP address) doesn't matter; Mallory spoofs the Authority & Glue record sections!

Mallory wins if any $r_i = s_j$

See http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html for details

Kaminsky Attack (2008)

- **Key Idea:** attacker forces DNS resolver to issue many queries by using many fake subdomains (e.g., z123123.bank.com)
    - Only needs to guess the query ID correctly for one of the queried subdomains (QID: 16 bits = only ~65,000 possible values)
    - Attaches a poisoned authority & glue record [NS info] in their reply
    - Once poisoning succeeds: all un-cached subdomain lookups will ask the attacker's server instead of the domain's real nameserver

- **Defense:** Randomize both the query ID and source port (16 -> 32 bits)
    - Billions of possible values: very low probability of winning the race even with many guesses at a time

General DNS Security: DNSSEC

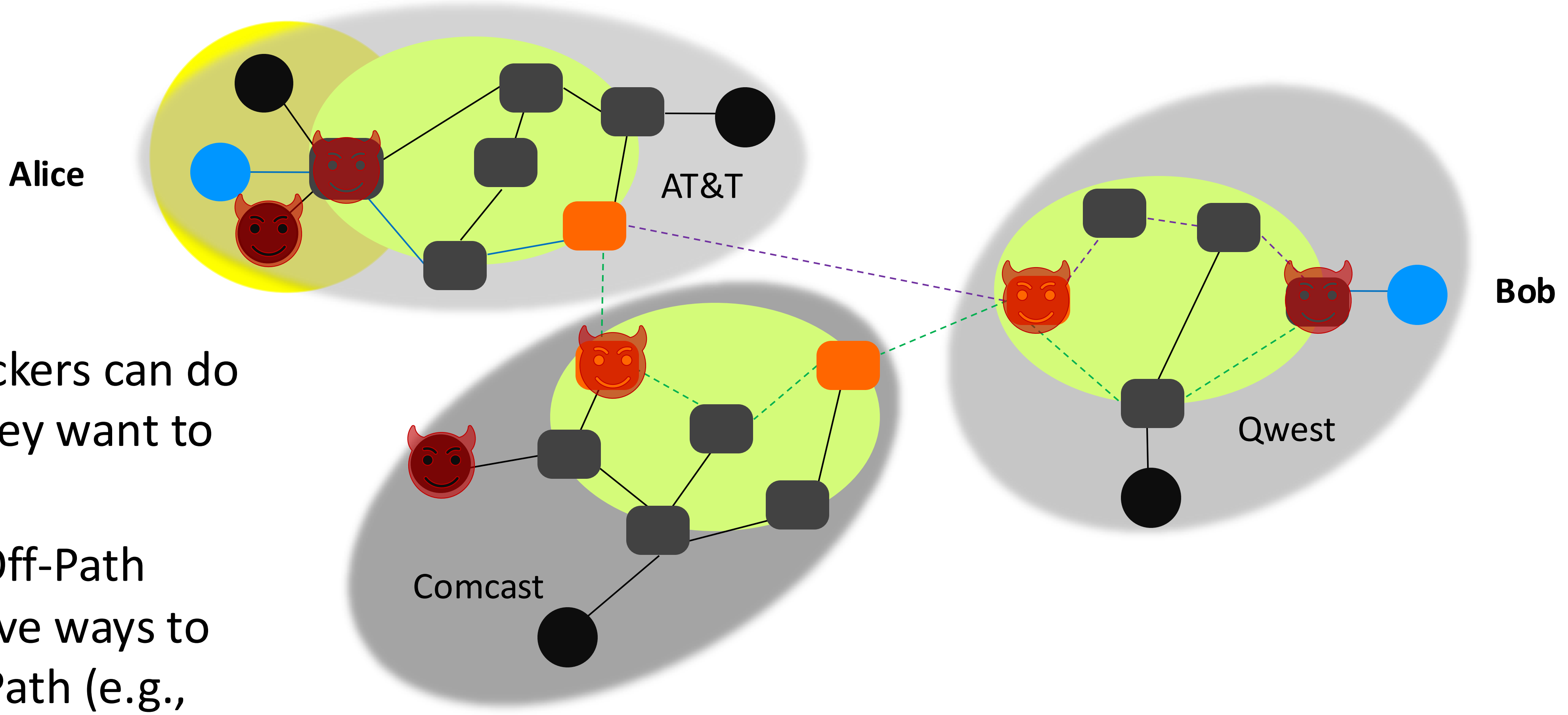
- DNS responses signed
- Higher levels vouch for lower levels
  - e.g., root vouches for .edu, .edu vouches for .uchicago, …
- Root public key published

- Most people don’t use DNSSEC and never will: Use TLS instead

Outline

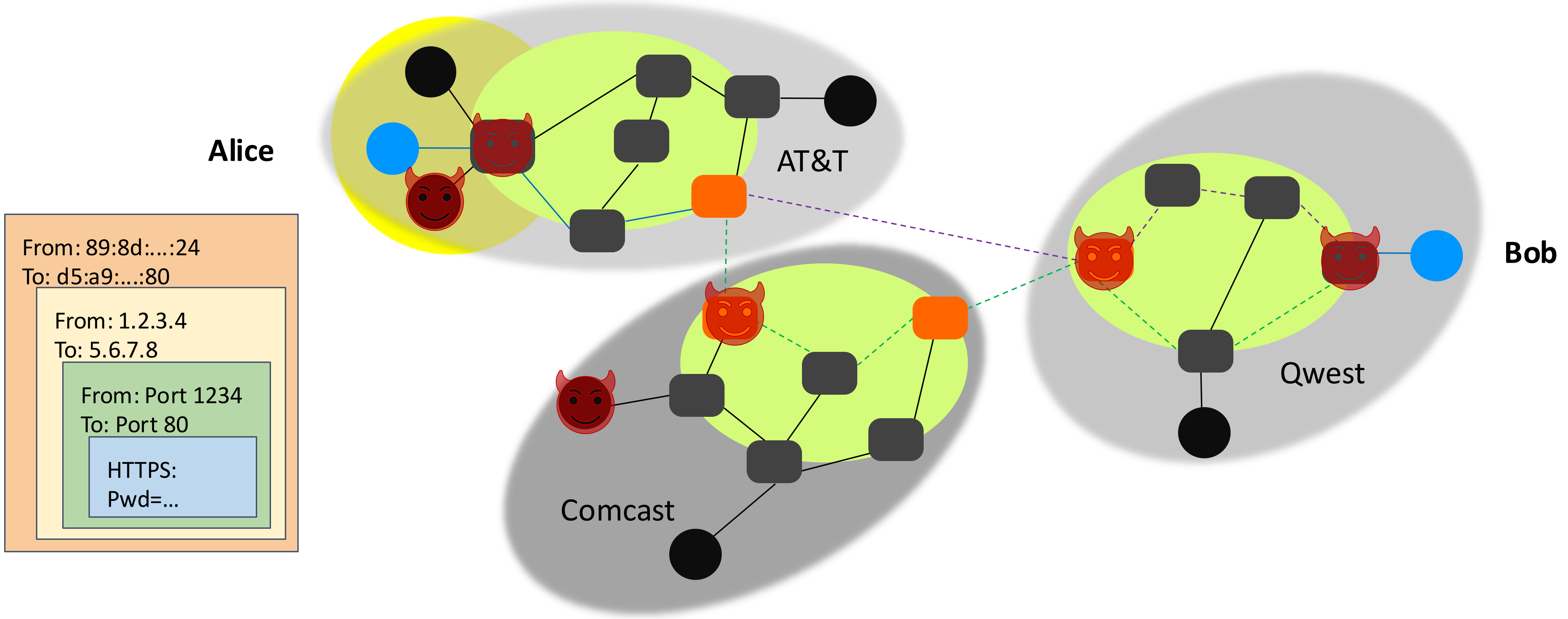
- Wrap-Up: DNS Security
- The Dream: Secure Channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS

The Internet is a Scary Place



- In-Path attackers can do whatever they want to packets
- On-Path & Off-Path attackers have ways to become In-Path (e.g., ARP, DHCP, DNS spoofing)

What if Alice & Bob had a shared cryptographic key?



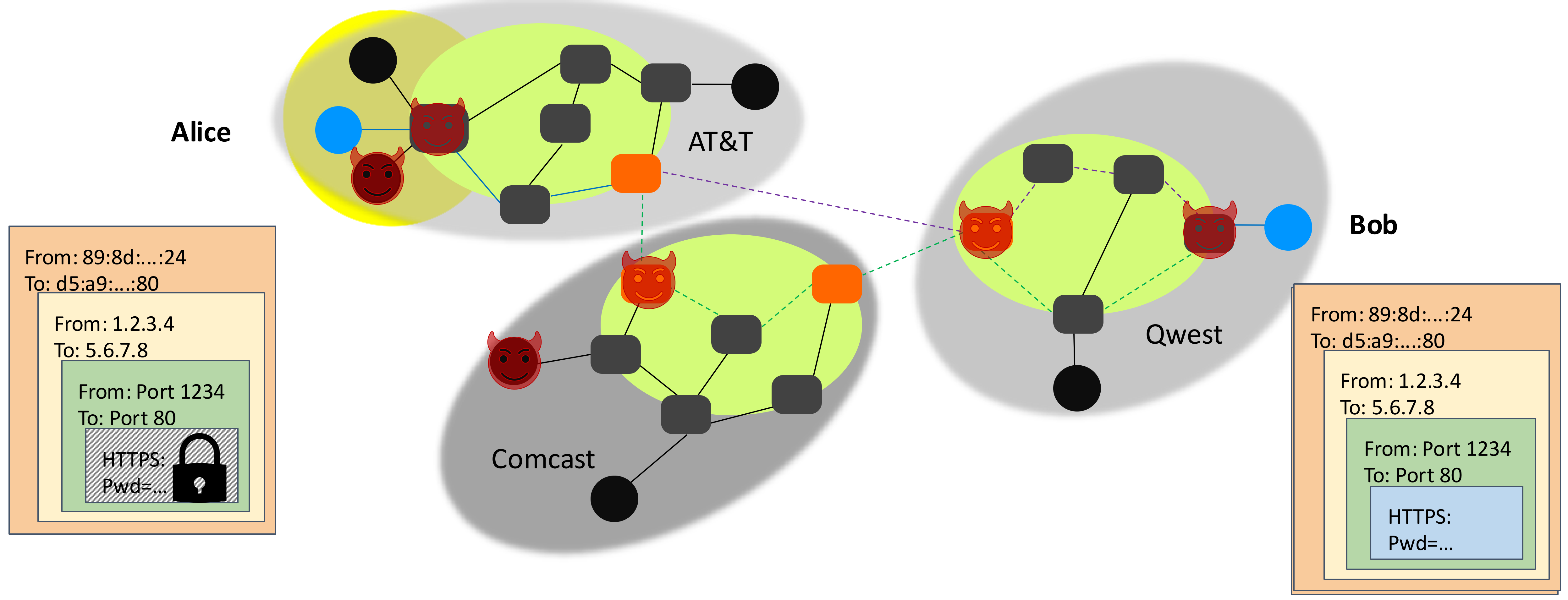
From: 89:8d:...:24
To: d5:a9:...:80

From: 1.2.3.4
To: 5.6.7.8

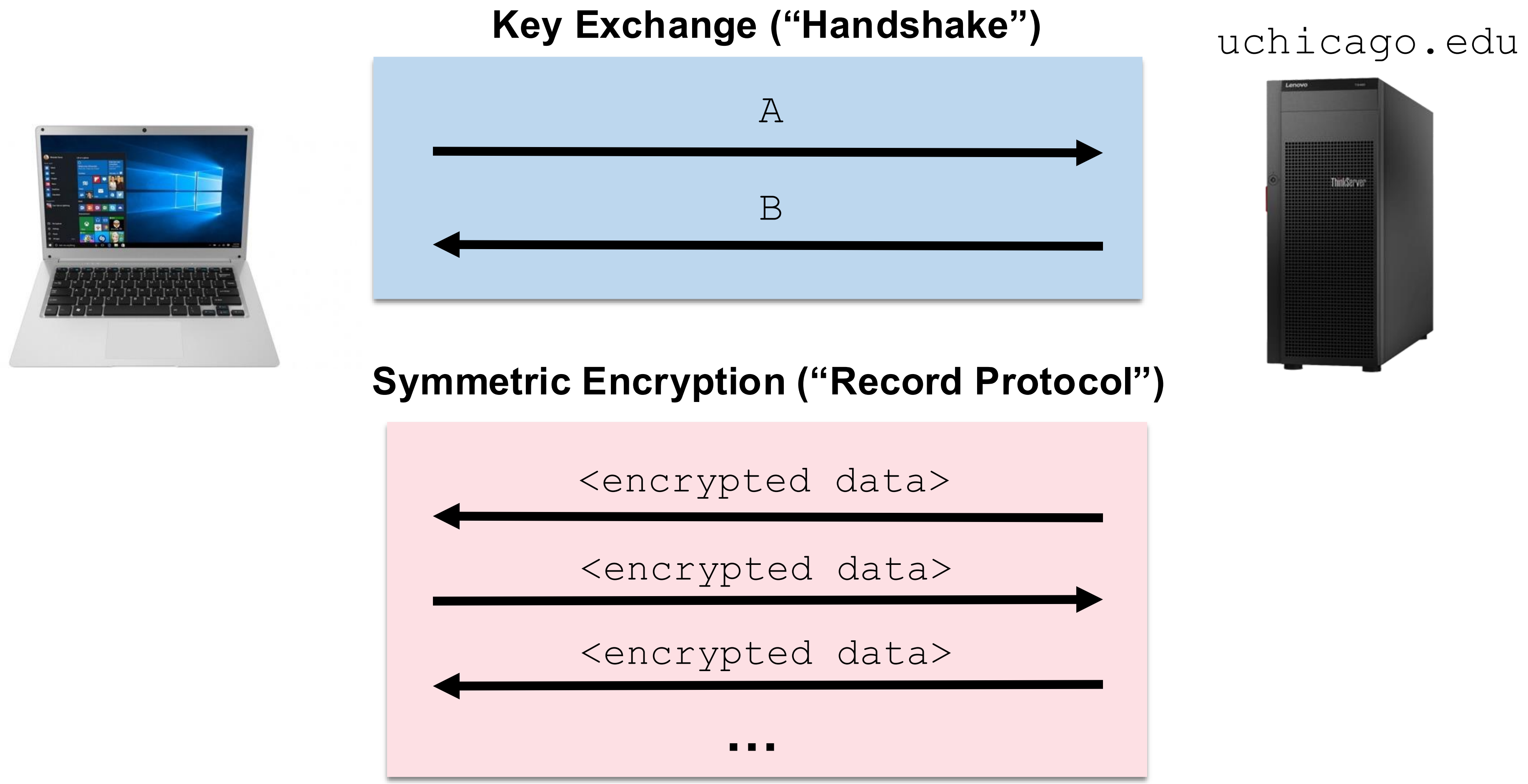
From: Port 1234
To: Port 80

HTTPS:
Pwd=...

What if Alice & Bob had a shared cryptographic key?



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

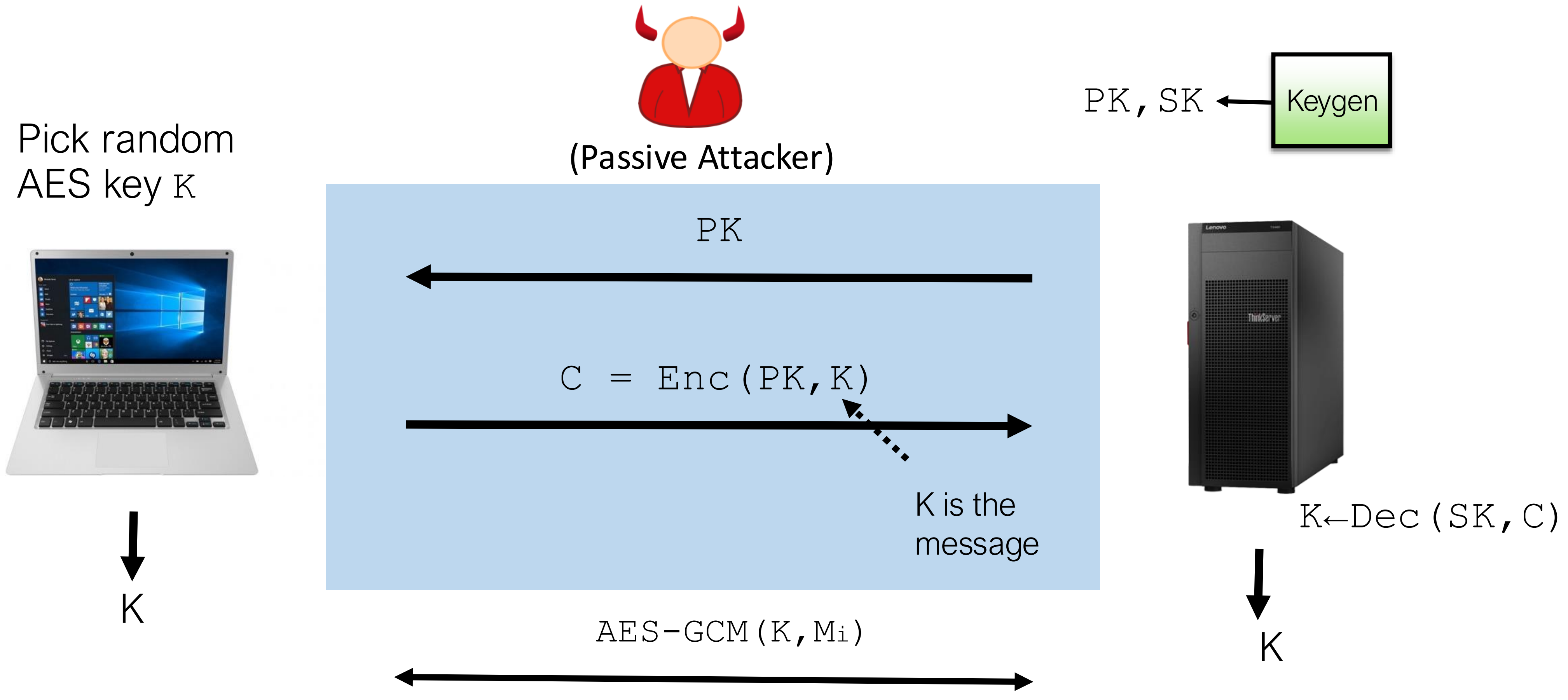

Key Exchange (“Handshake”)
uchicago.edu
A
B
Symmetric Encryption (“Record Protocol”)
<encrypted data>
<encrypted data>
<encrypted data>
…

- Recall: Naïve key exchange secure against ***passive adversaries.***
- But the above template does <u>not</u> provide authentication & integrity.

Recall: Naïve Key Exchange w/ Pub-Key

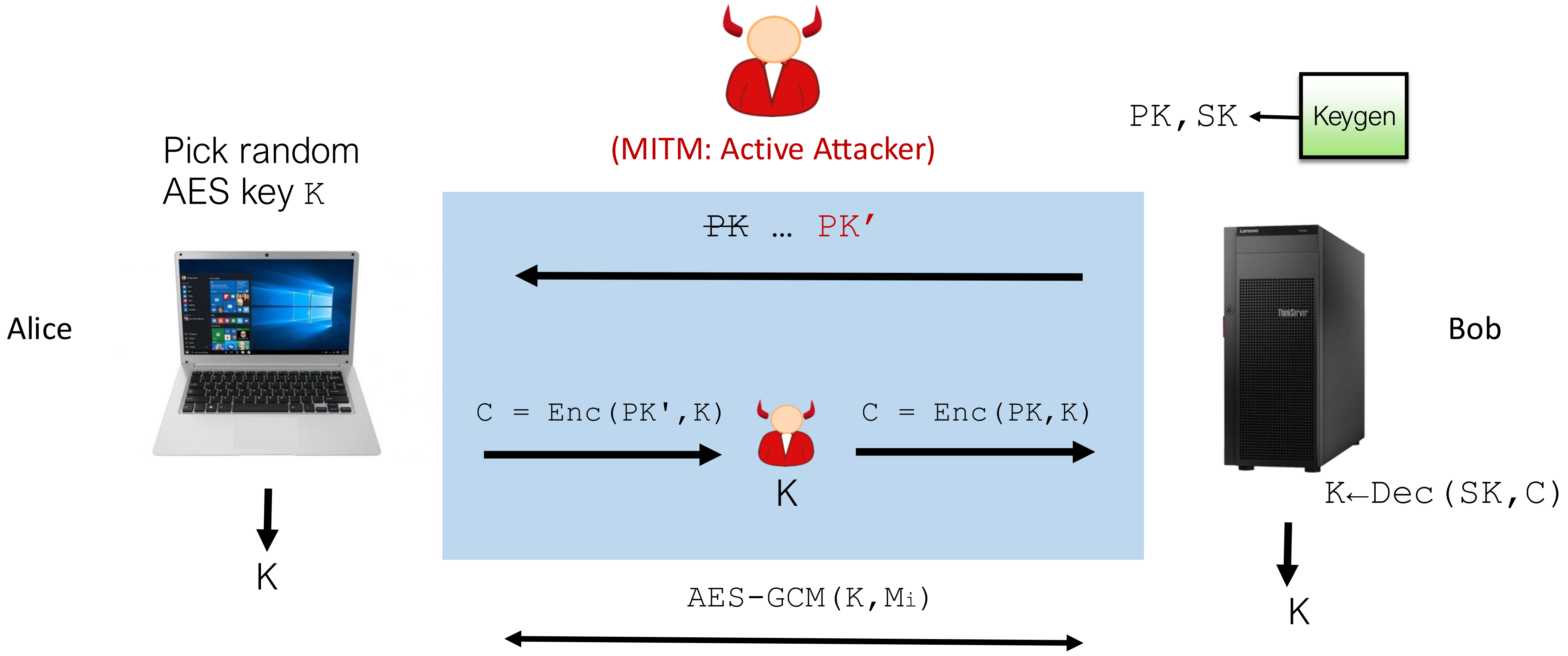
Goal: Establish secret key `K` to use for Symmetric Encryption.

`(KeyGen, Enc, Dec)` is a public-key encryption scheme (e.g., RSA).



Securing Key Exchange against Active (MITM) Attackers

Key Challenge: Authenticity: How do we know that PK is really Bob's?



Recall: Public Crypto Tools

Public Key Encryption

- Encryption key [pk] is public to everyone (anyone can encrypt)
- Only the person with the private key [sk] scan decrypt

Digital Signatures

- Verification key [vk] is public to everyone (anyone can validate signatures)
- Only person with signing key [sk] can generate signatures

Authentication with Certificates ("Certs")

Suppose we had a globally trusted entity, `BlaséInc`.

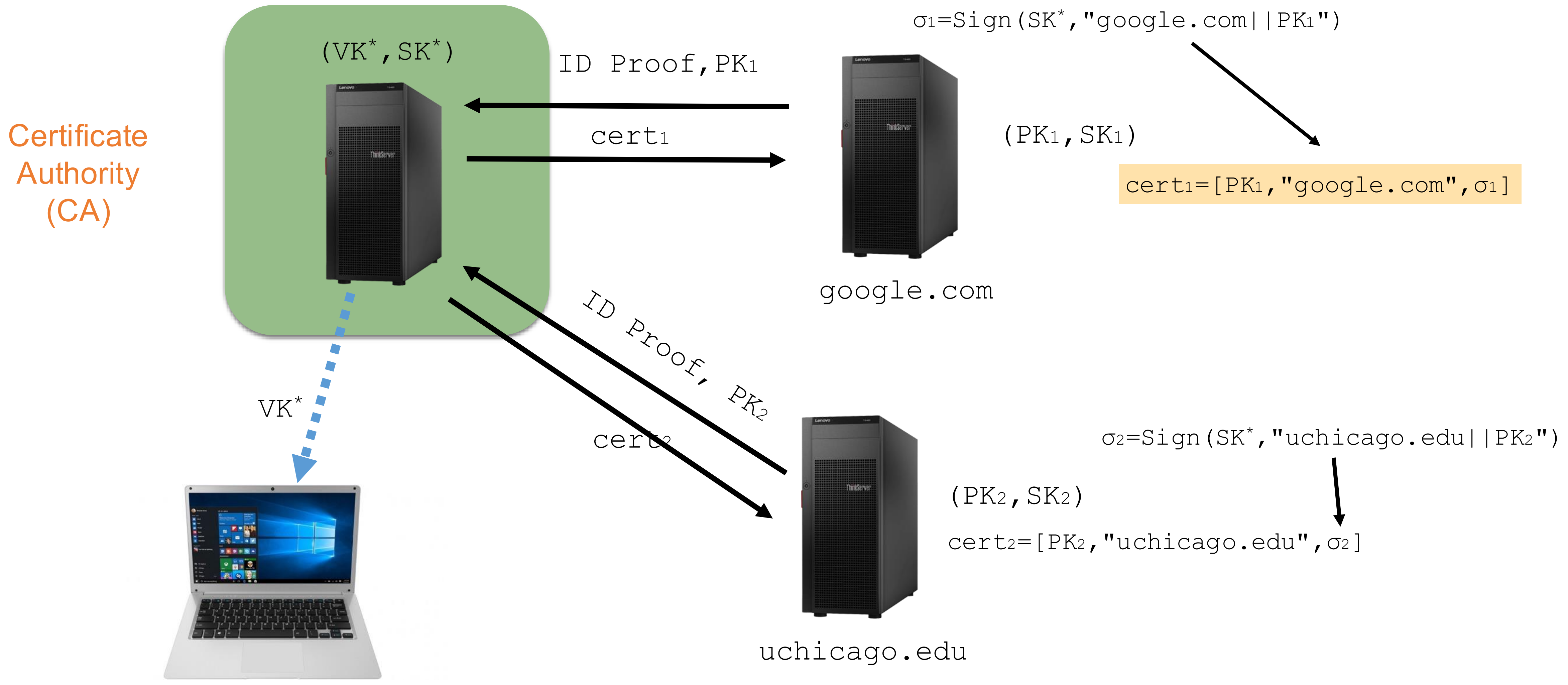
`BlaséInc` could issue **certificates ("certs")** that state what other organizations' public keys are.

Cert = a document that says:

1. An Entity (e.g., UChicago) has a public key that is:
2. `pk=0x7b5532…`, where the document is
3. signed using the `BlaséInc`'s private signing key

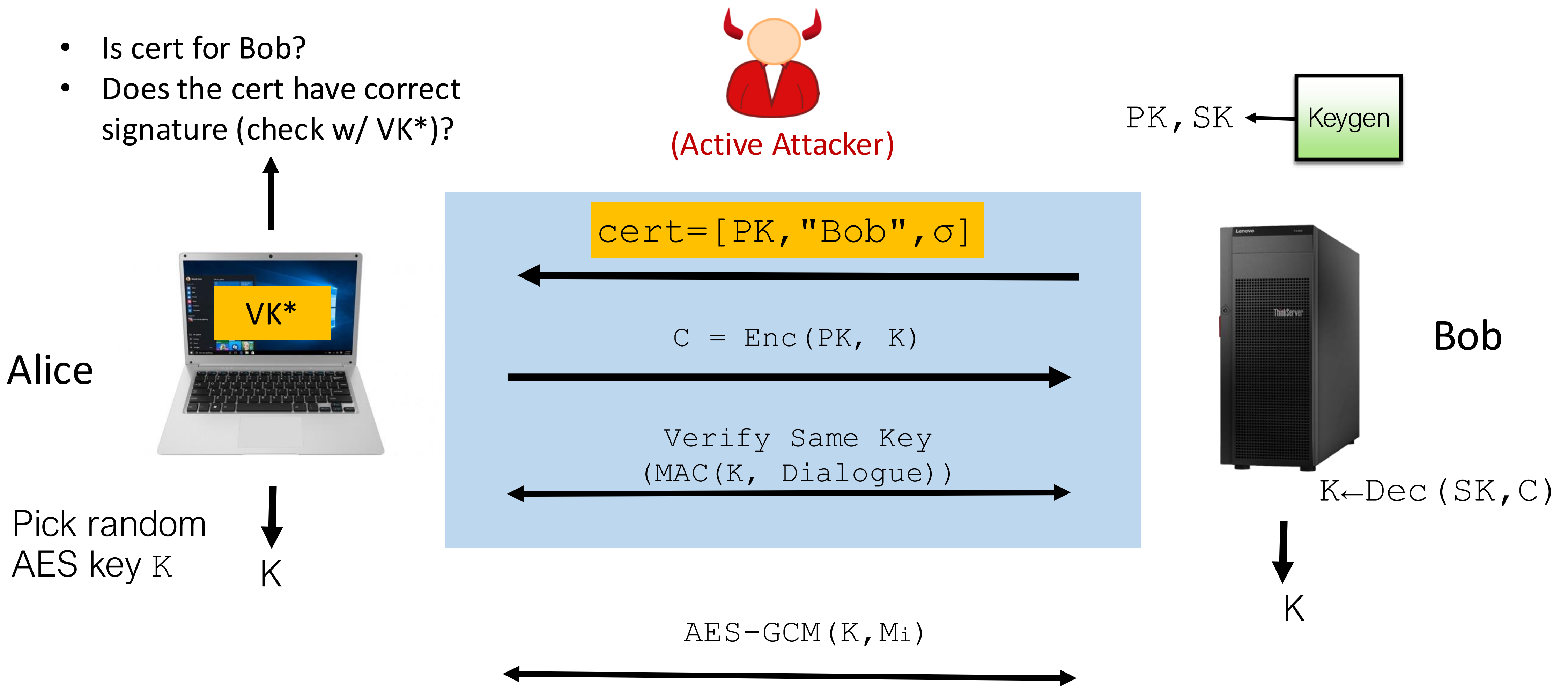
Trusted entity, `BlaséInc`, known as a **Certificate Authority (CA)**

Authentication with Certificates (“Certs”)


(VK^*, SK^*)
Certificate Authority (CA)
ID Proof, PK_1
$cert_1$
σ_1=Sign(SK^*,"google.com||PK_1")
(PK_1, SK_1)
$cert_1$=[PK_1,"google.com",σ_1]
google.com
ID Proof, PK_2
$cert_2$
VK^*
σ_2=Sign(SK^*,"uchicago.edu||PK_2")
(PK_2, SK_2)
$cert_2$=[PK_2,"uchicago.edu",σ_2]
uchicago.edu

VK* pre-installed on every machine by manufacturer or built into OS code.

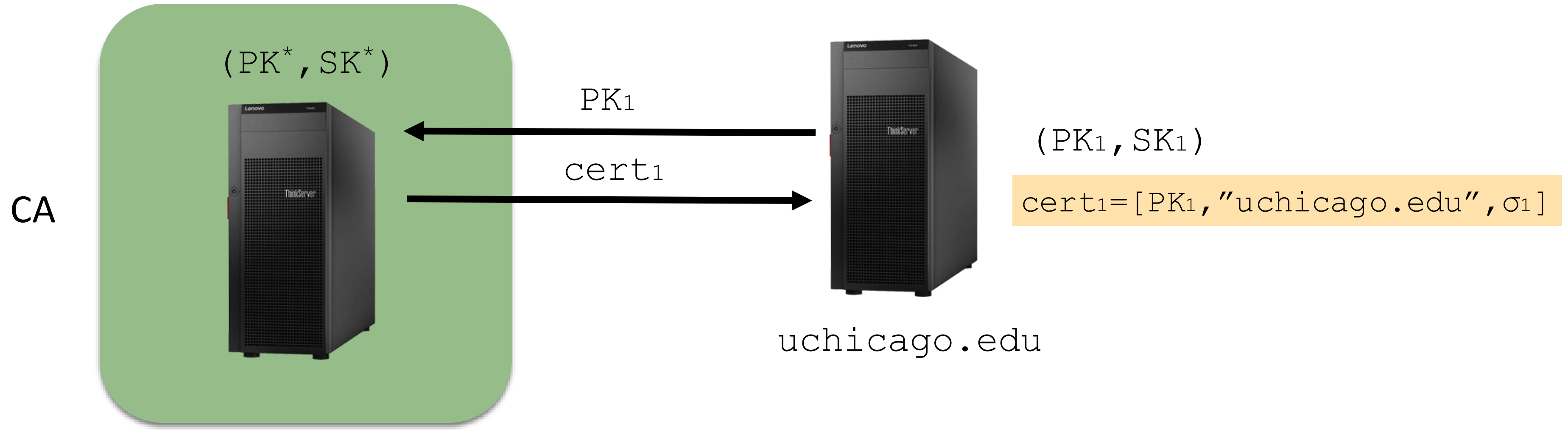
Securing Key Exchange against Active Attackers



Outline

- Wrap-Up: DNS Security
- The Dream: Secure Channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS

Issuing Certificates: Validation

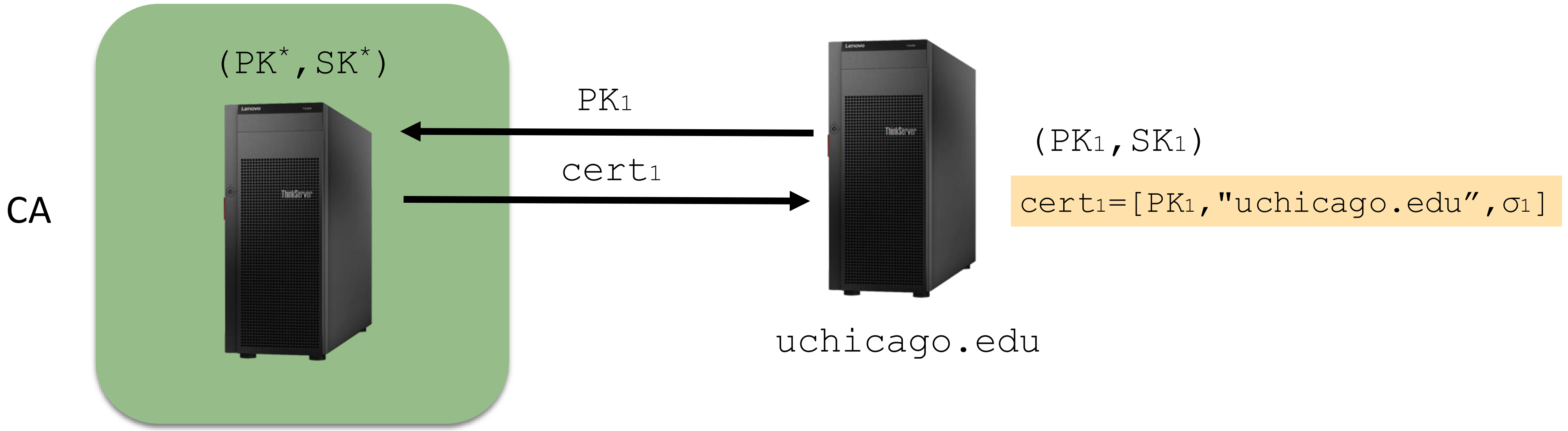


- CA must check that key PK_1 really does belong to "`uchicago.edu`"

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.

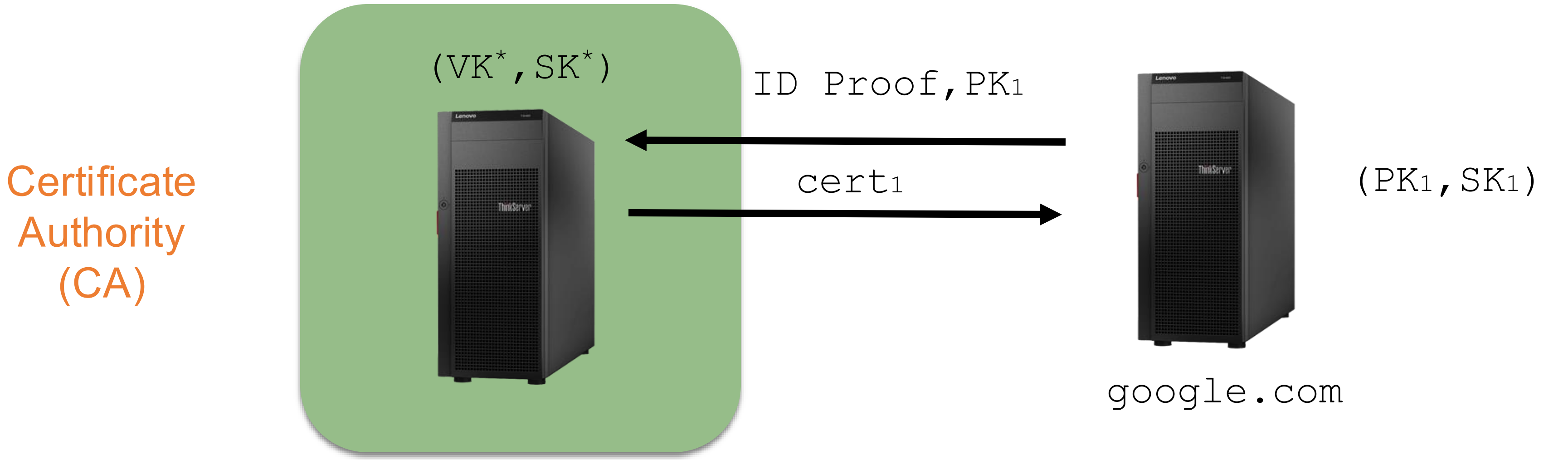
ACME Protocol by Let’s Encrypt



1. Requestor submits public key and request to CA
2. CA gives a *challenge* to requestor
3. Requestor places *challenge* on server or DNS TXT records
4. CA checks *challenge* and then issues cert if challenge matches

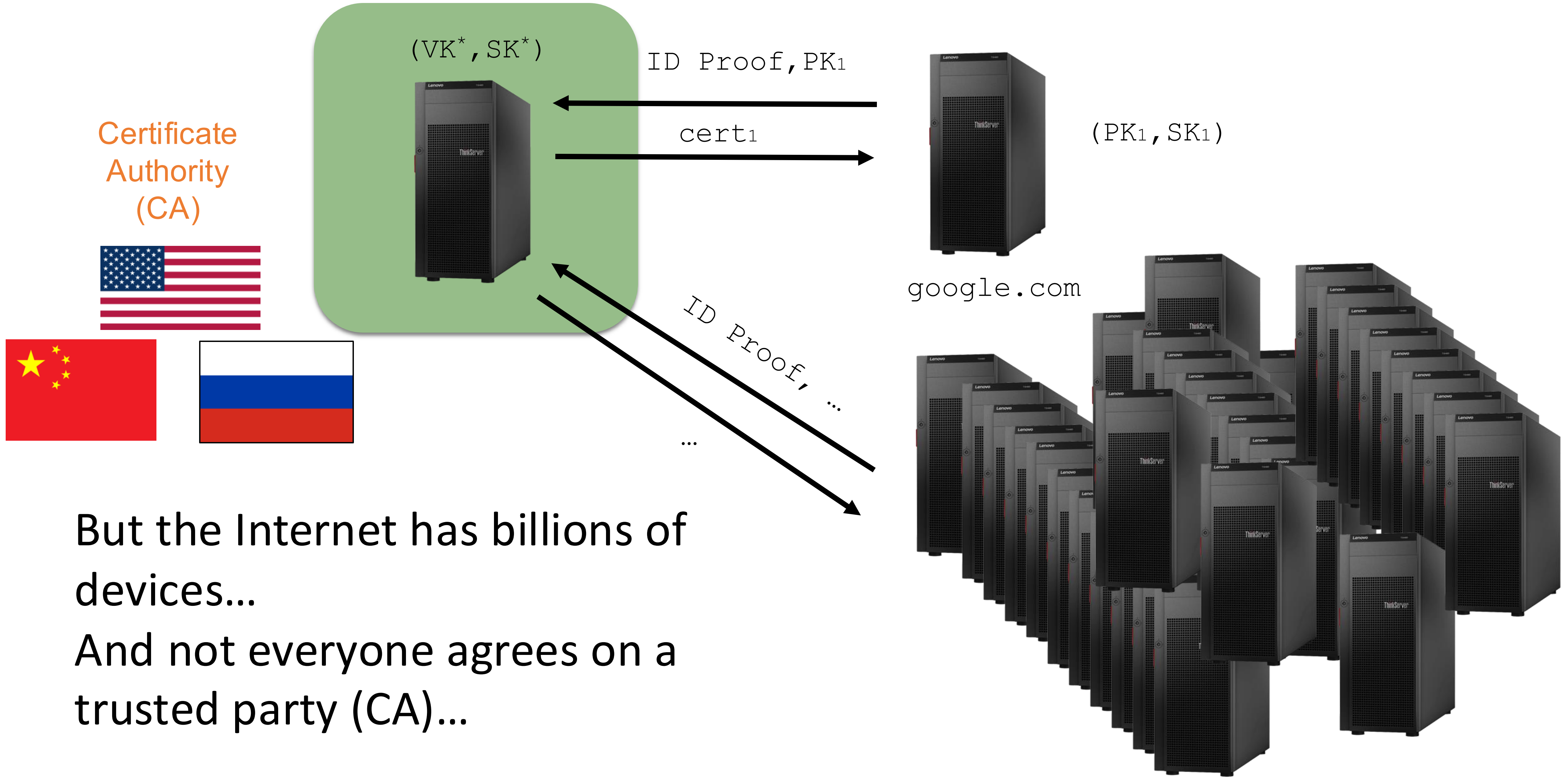
Let’s Encrypt certbot

Scaling Certificates to the Internet



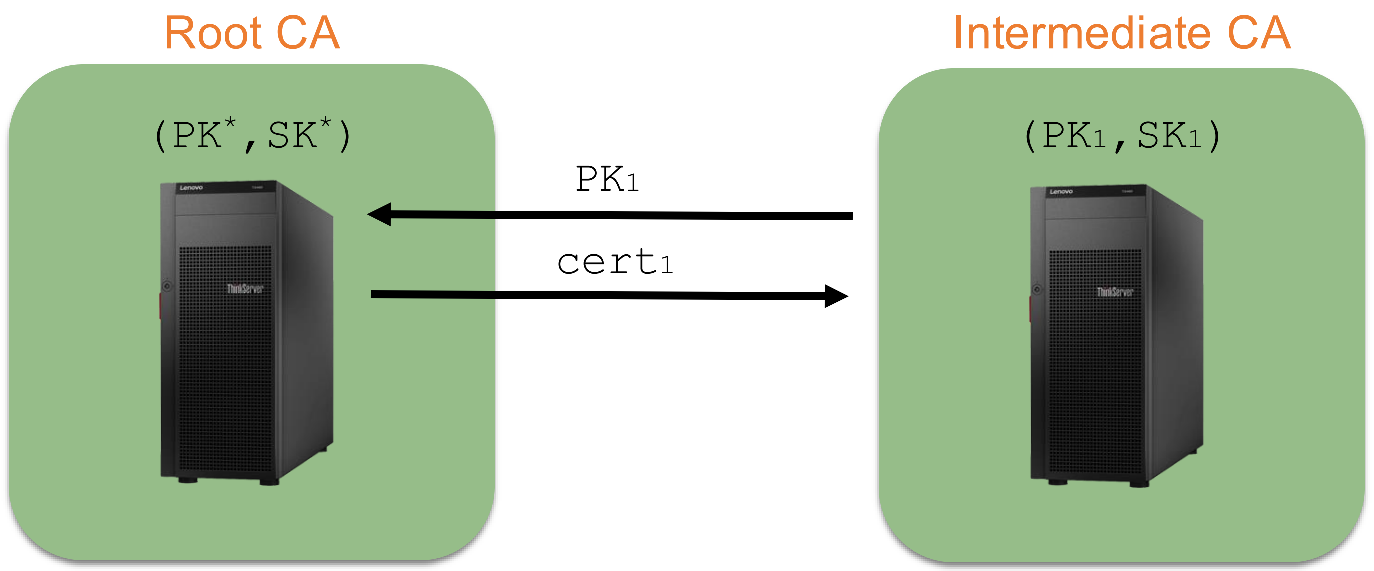
Having one CA works fine if the Internet has just a few entities and everyone agrees that the CA is trustworthy.

Scaling Certificates to the Internet



But the Internet has billions of devices…
And not everyone agrees on a trusted party (CA)…

Scaling: Intermediate CAs and Cert Chains



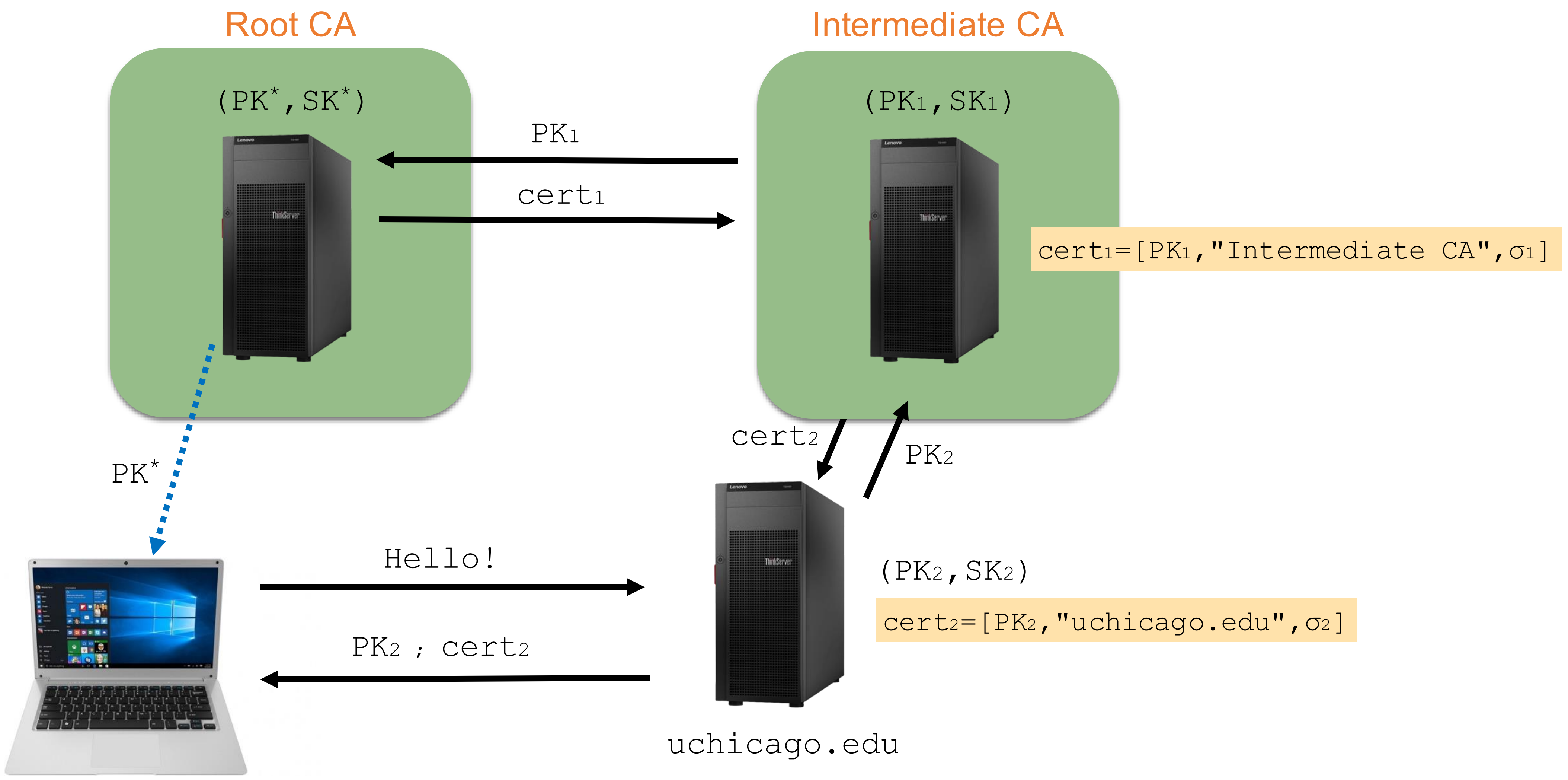
To handle scaling:

- Allow a trusted Root CA to delegate their trust to multiple intermediate CA's
- Any of these intermediate CA's can then create a certificate for someone
    - 100's of intermediate CA's on the Internet

Scaling: Intermediate CAs and Cert Chains

To check PK_2 recursive validation:

1) Check $cert_2$ to make sure PK_2 for uchicago.edu
2) Get PK_1 and $cert_1$ to check sig of $cert_2$
3) If $cert_1$ issued by root CA, use PK* to its check sig.



PK^* bound to Root ⇒ PK_1 bound to CA ⇒ PK_2 bound to uchicago.edu

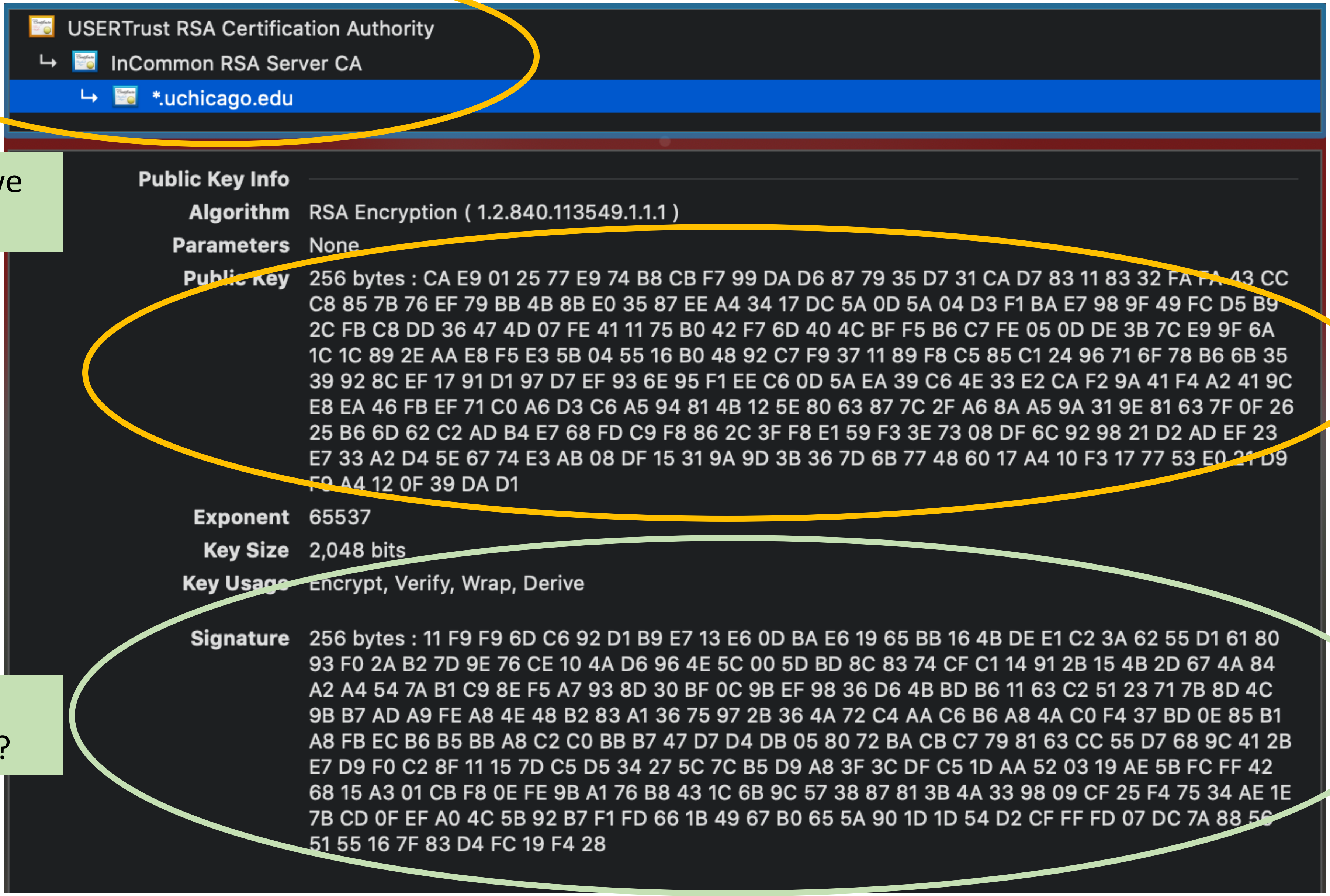
X.509 Certificates

Cert Content Includes:

- Cert’s Serial number
- Cert’s Expiration date
- Common name of subject (e.g., Bob [google.com])
- Public key of subject
- Extensions (possibly many)
- CA info (name of CA that is issuing the cert, etc.)
- CA’s Signature (on hash of cert)

Who are we trusting?

Who's signature?

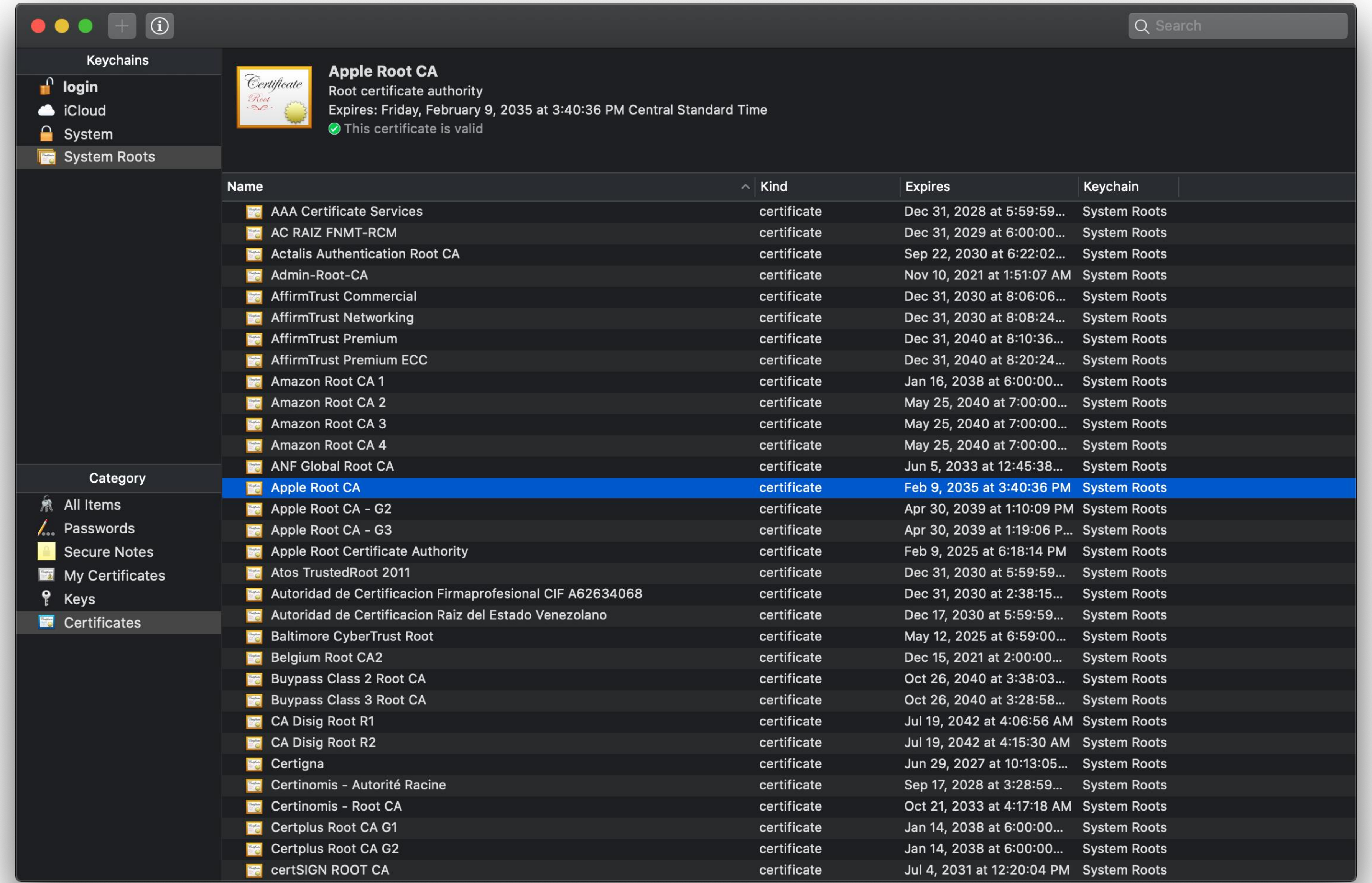


```
USERTrust RSA Certification Authority
↳ InCommon RSA Server CA
    ↳ *.uchicago.edu

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
```

Root CA’s & Root Certificates



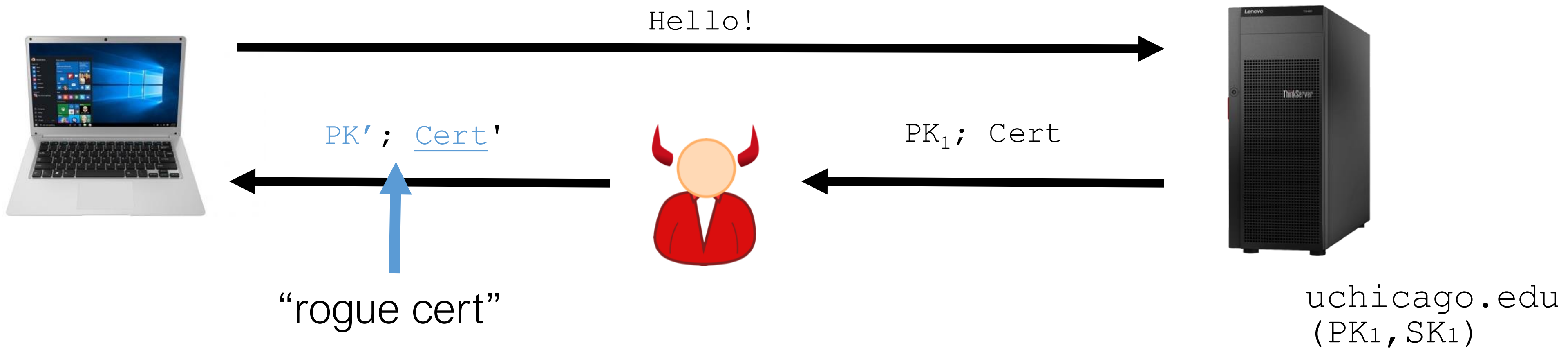
Apple Root CA
Root certificate authority
Expires: Friday, February 9, 2035 at 3:40:36 PM Central Standard Time
✓ This certificate is valid

Name	Kind	Expires	Keychain
AAA Certificate Services	certificate	Dec 31, 2028 at 5:59:59...	System Roots
AC RAIZ FNMT-RCM	certificate	Dec 31, 2029 at 6:00:00...	System Roots
Actalis Authentication Root CA	certificate	Sep 22, 2030 at 6:22:02...	System Roots
Admin-Root-CA	certificate	Nov 10, 2021 at 1:51:07 AM	System Roots
AffirmTrust Commercial	certificate	Dec 31, 2030 at 8:06:06...	System Roots
AffirmTrust Networking	certificate	Dec 31, 2030 at 8:08:24...	System Roots
AffirmTrust Premium	certificate	Dec 31, 2040 at 8:10:36...	System Roots
AffirmTrust Premium ECC	certificate	Dec 31, 2040 at 8:20:24...	System Roots
Amazon Root CA 1	certificate	Jan 16, 2038 at 6:00:00...	System Roots
Amazon Root CA 2	certificate	May 25, 2040 at 7:00:00...	System Roots
Amazon Root CA 3	certificate	May 25, 2040 at 7:00:00...	System Roots
Amazon Root CA 4	certificate	May 25, 2040 at 7:00:00...	System Roots
ANF Global Root CA	certificate	Jun 5, 2033 at 12:45:38...	System Roots
Apple Root CA	certificate	Feb 9, 2035 at 3:40:36 PM	System Roots
Apple Root CA - G2	certificate	Apr 30, 2039 at 1:10:09 PM	System Roots
Apple Root CA - G3	certificate	Apr 30, 2039 at 1:19:06 P...	System Roots
Apple Root Certificate Authority	certificate	Feb 9, 2025 at 6:18:14 PM	System Roots
Atos TrustedRoot 2011	certificate	Dec 31, 2030 at 5:59:59...	System Roots
Autoridad de Certificacion Firmaprofesional CIF A62634068	certificate	Dec 31, 2030 at 2:38:15...	System Roots
Autoridad de Certificacion Raiz del Estado Venezolano	certificate	Dec 17, 2030 at 5:59:59...	System Roots
Baltimore CyberTrust Root	certificate	May 12, 2025 at 6:59:00...	System Roots
Belgium Root CA2	certificate	Dec 15, 2021 at 2:00:00...	System Roots
Buypass Class 2 Root CA	certificate	Oct 26, 2040 at 3:38:03...	System Roots
Buypass Class 3 Root CA	certificate	Oct 26, 2040 at 3:28:58...	System Roots
CA Disig Root R1	certificate	Jul 19, 2042 at 4:06:56 AM	System Roots
CA Disig Root R2	certificate	Jul 19, 2042 at 4:15:30 AM	System Roots
Certigna	certificate	Jun 29, 2027 at 10:13:05...	System Roots
Certinomis - Autorité Racine	certificate	Sep 17, 2028 at 3:28:59...	System Roots
Certinomis - Root CA	certificate	Oct 21, 2033 at 4:17:18 AM	System Roots
Certplus Root CA G1	certificate	Jan 14, 2038 at 6:00:00...	System Roots
Certplus Root CA G2	certificate	Jan 14, 2038 at 6:00:00...	System Roots
certSIGN ROOT CA	certificate	Jul 4, 2031 at 12:20:04 PM	System Roots

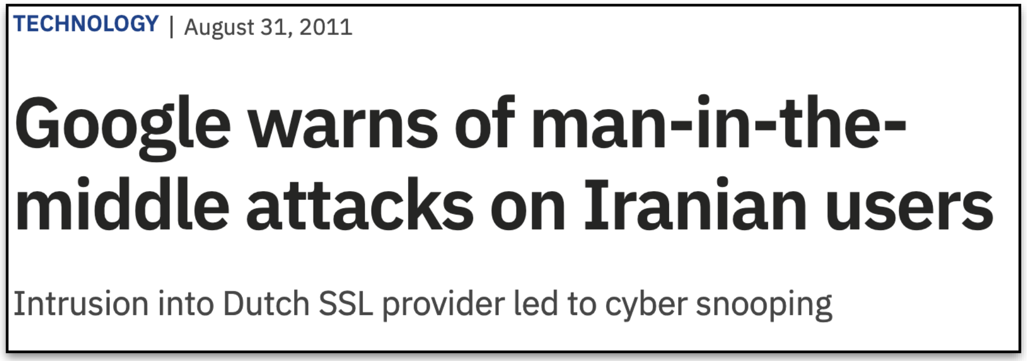
Outline

- Wrap-Up: DNS Security
- The Dream: Secure Channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Certs: Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS

What if attacker got a "valid" cert for uchicago.edu that has their malicious key?



- "Machine-in-the-middle" can read/change all traffic undetected


TECHNOLOGY | August 31, 2011

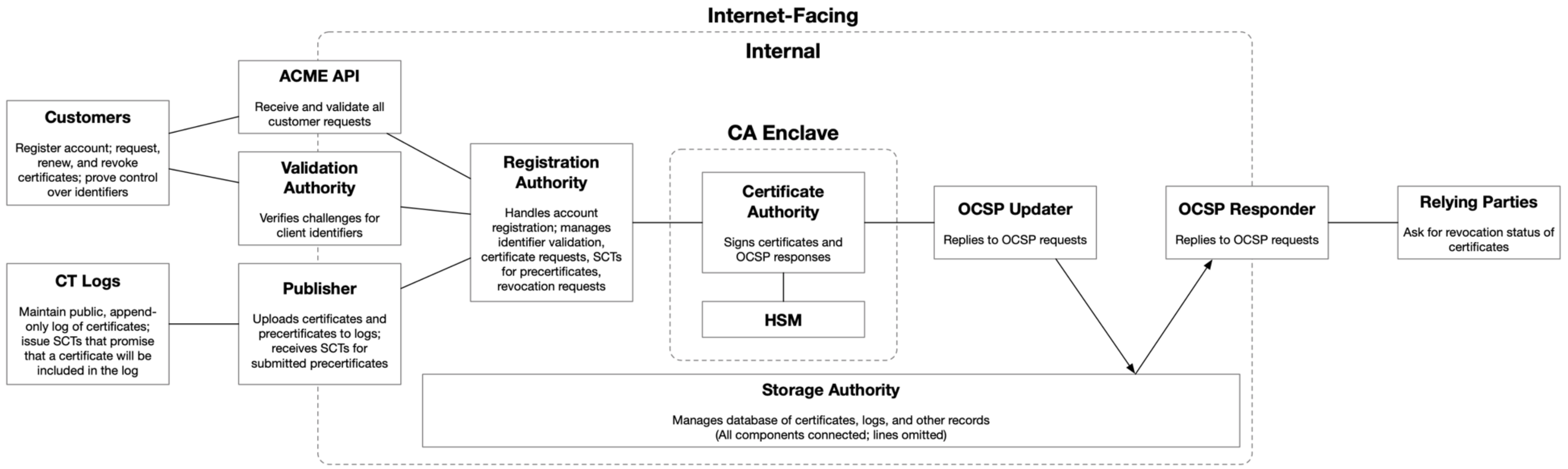
Google warns of man-in-the-middle attacks on Iranian users

Intrusion into Dutch SSL provider led to cyber snooping

CA Security

Some common attacks to get rogue certificate:

- Fool or bypass a CA's validation process
- Compromise a CA organization and generate malicious cert's



Internet-Facing
Internal
CA Enclave
Customers
Register account; request, renew, and revoke certificates; prove control over identifiers
ACME API
Receive and validate all customer requests
Validation Authority
Verifies challenges for client identifiers
CT Logs
Maintain public, append-only log of certificates; issue SCTs that promise that a certificate will be included in the log
Publisher
Uploads certificates and precertificates to logs; receives SCTs for submitted precertificates
Registration Authority
Handles account registration; manages identifier validation, certificate requests, SCTs for precertificates, revocation requests
Certificate Authority
Signs certificates and OCSP responses
HSM
OCSP Updater
Replies to OCSP requests
OCSP Responder
Replies to OCSP requests
Relying Parties
Ask for revocation status of certificates
Storage Authority
Manages database of certificates, logs, and other records
(All components connected; lines omitted)

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.

"Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web", CCS 2019

Sample of 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 MitM 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

- **Goal:** Eliminate Root / Intermediate CA's with bad hygiene or who you don't trust

- Server (e.g., website) can tell client (e.g., browser) to only accept certs signed by certain CA's
  - Code trusted CA keys into client app (e.g., Chrome only trusts certs signed by Google's CA), or
  - Send special application message telling client what to pin (More common)
- Helped discover some rogue certs from previous slide

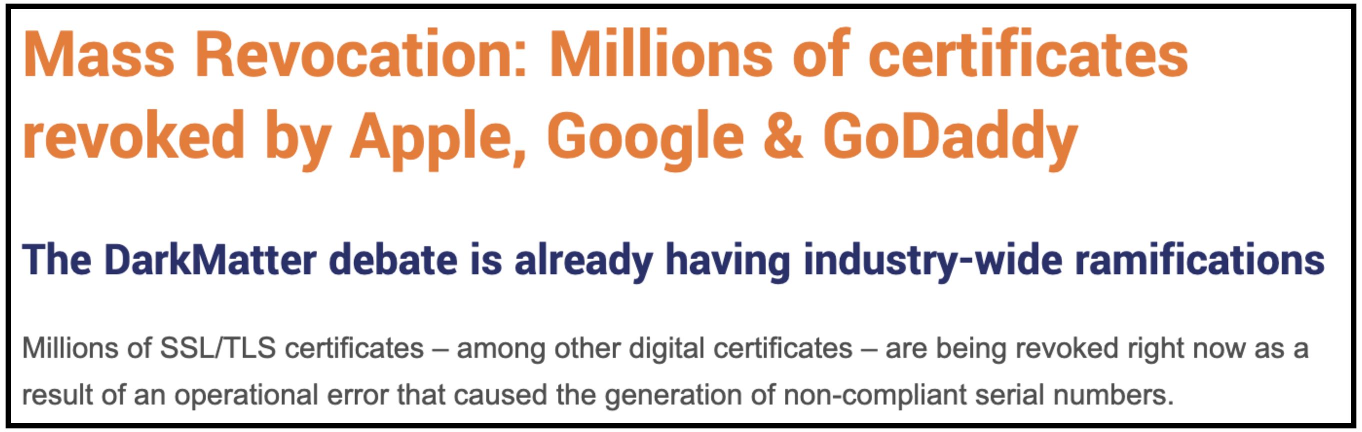
- What are some problems with this defense?
  - If server hacked… attacker can pin a malicious key/cert: will only connect w/ attacker cert!
  - Website error: pin wrong or broken key… website inaccessible!

Now deprecated because of these issues

Countermeasure: Revocation

Publicly list bad (revoked) certificates so they are no longer accepted

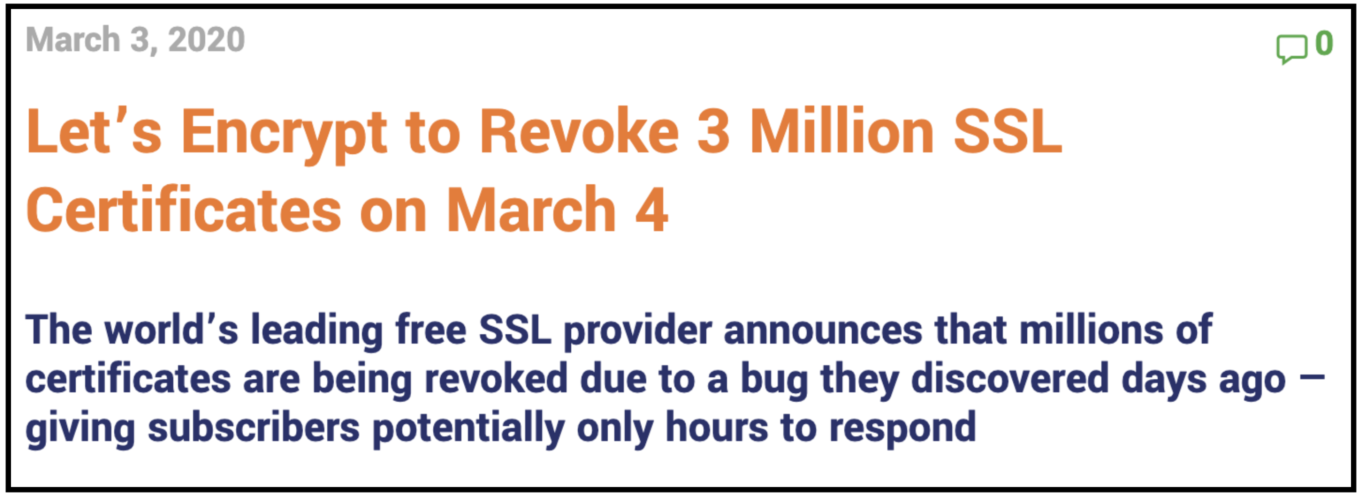
- CA or Server (that was issued cert) can revoke



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.

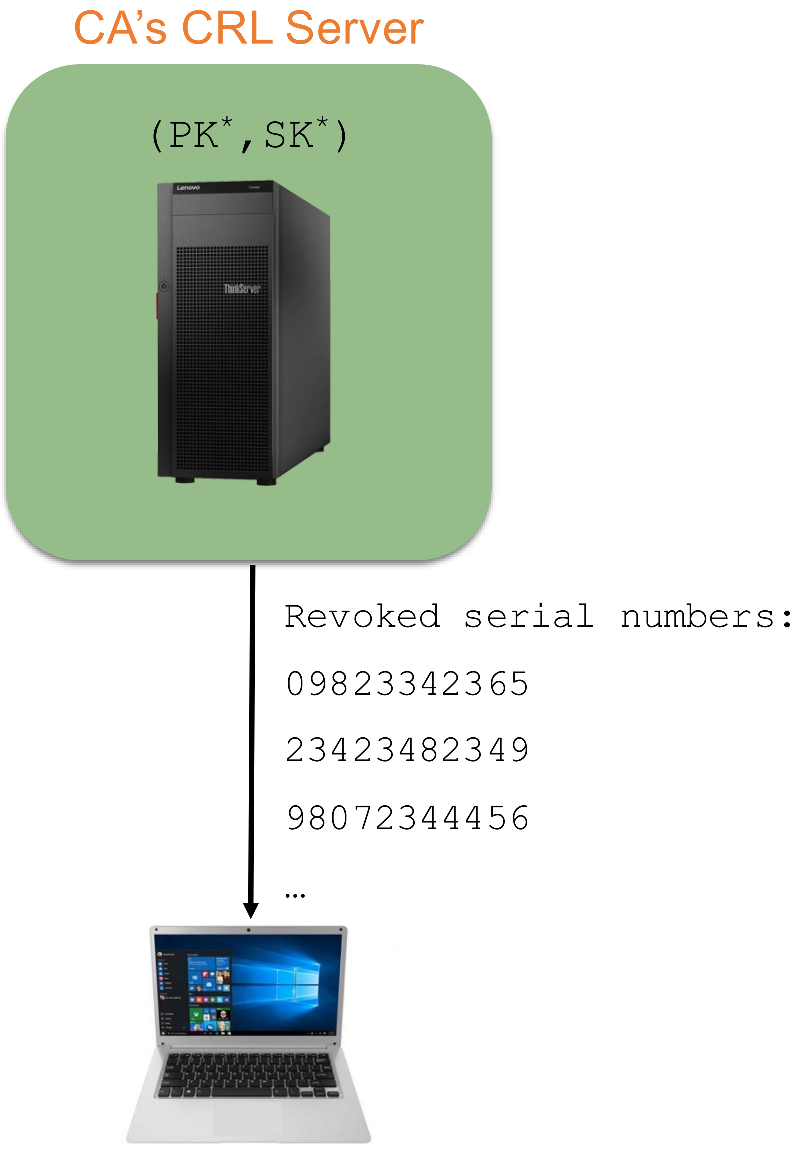


March 3, 2020 0

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

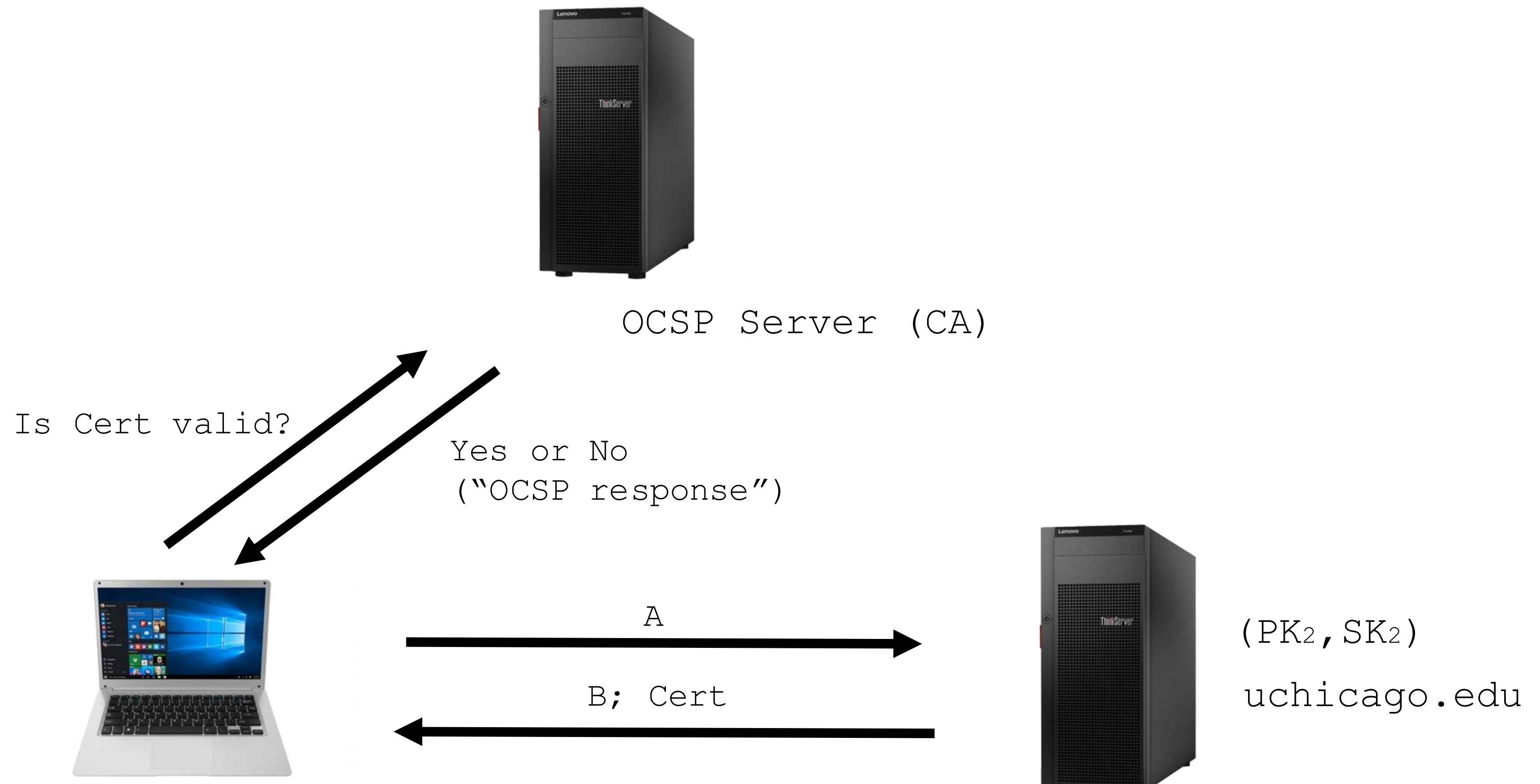
Cert Revocation Lists (CRLs)



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

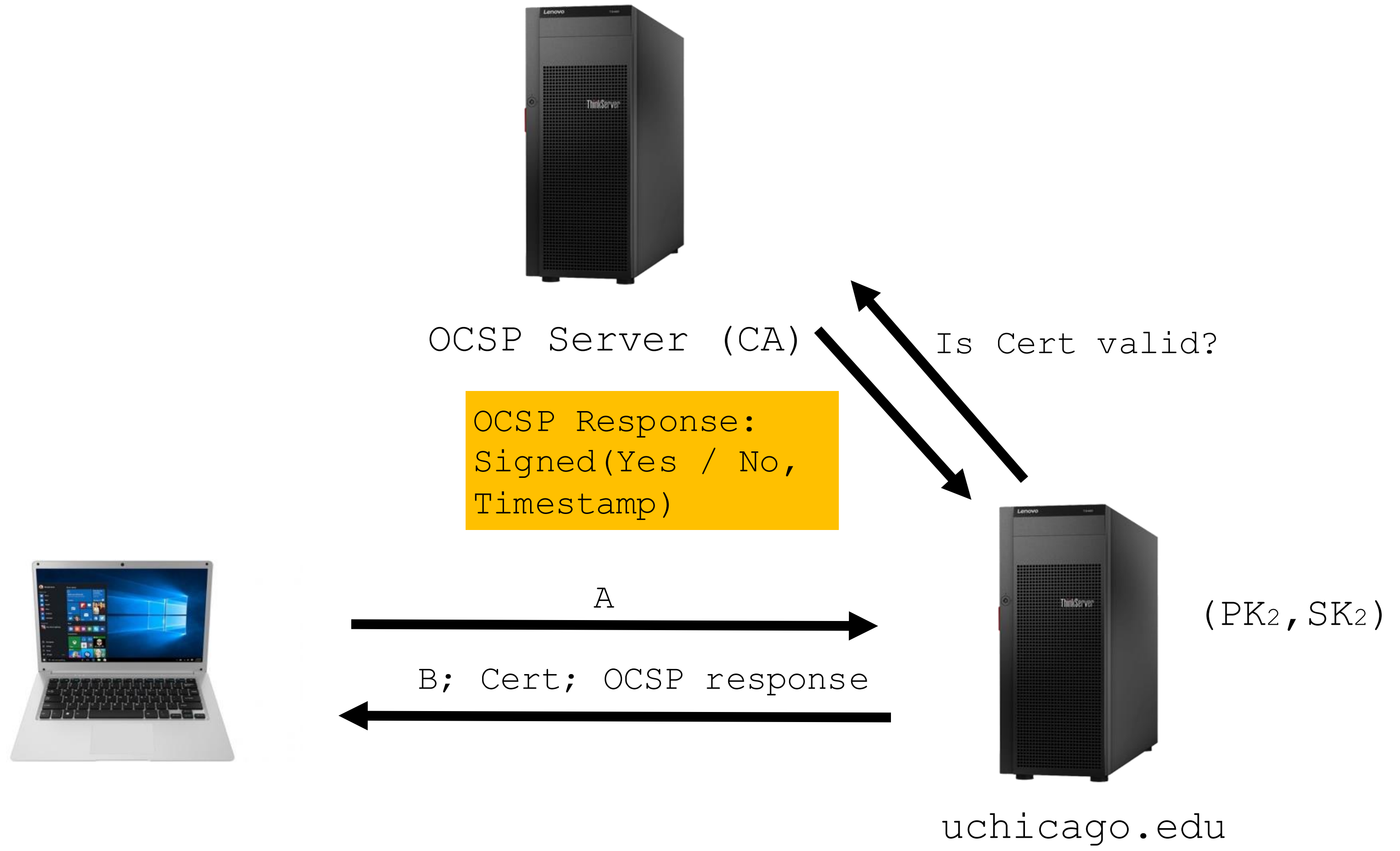
- Each CA provides a list of revoked cert's
- Clients can download CRL and check cert's they receive against the list
- Problems:
  - List will get too large
  - Difficult to keep current

Revocation: Online Certificate Status Protocol (OCSP)



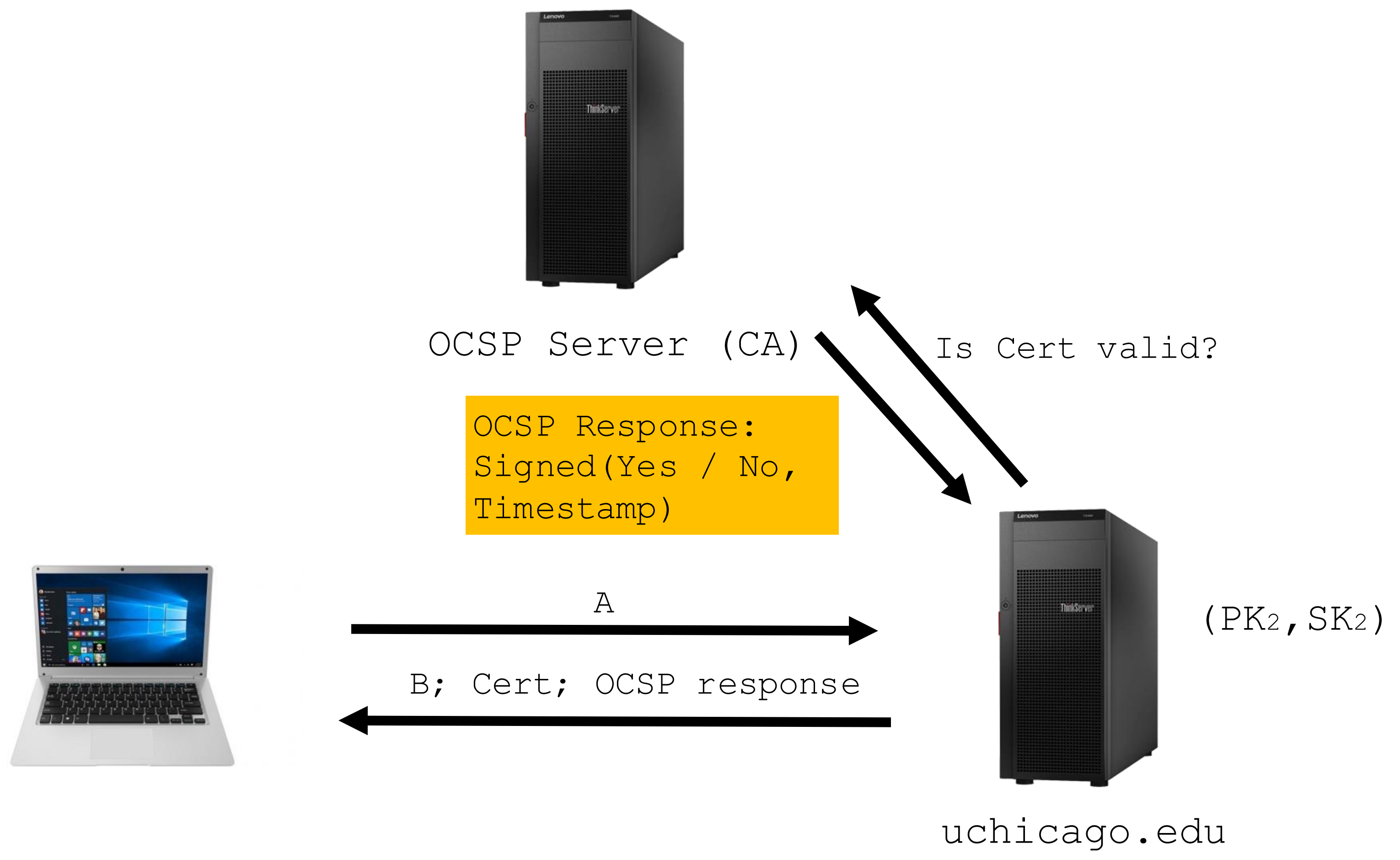
- Add another server to connect to, slowing connection
- What if OCSP server times out?
- Privacy problem?

Revocation: 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.

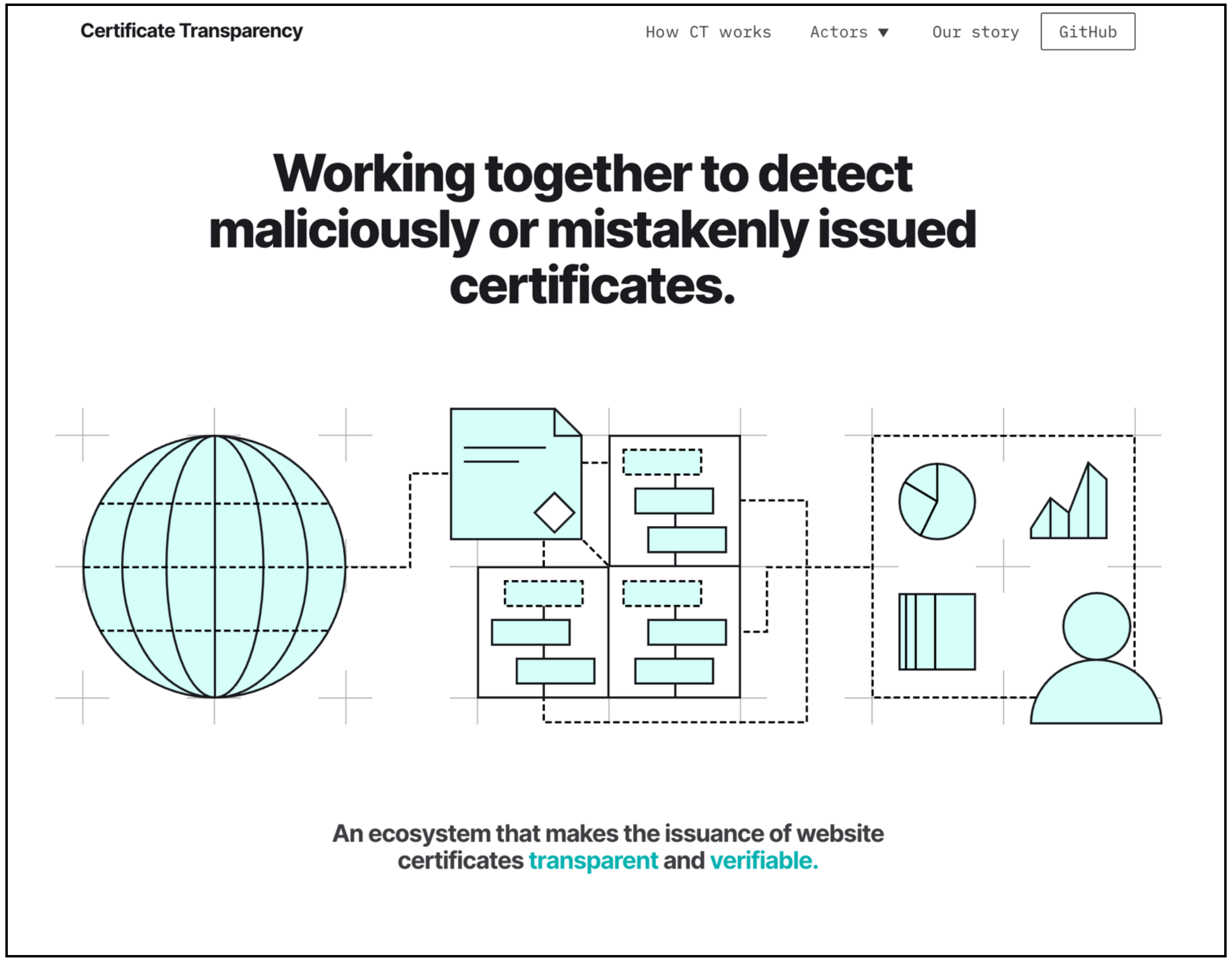
Revocation: OCSP Stapling



Problems?

- OCSP server goes down => uchicago.edu goes down (no OCSP response to attach to cert)

Certificate Transparency (CT) : How do we find rogue certs?



Working together to detect maliciously or mistakenly issued certificates.

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

Scenario: Attackers compromise a CA and create rogue certs for `google.com` that have
(1) attacker's public keys and
(2) valid CA signature

How does Google or the CA discover these rogue certs were issued or in use?

Cert Transparency:

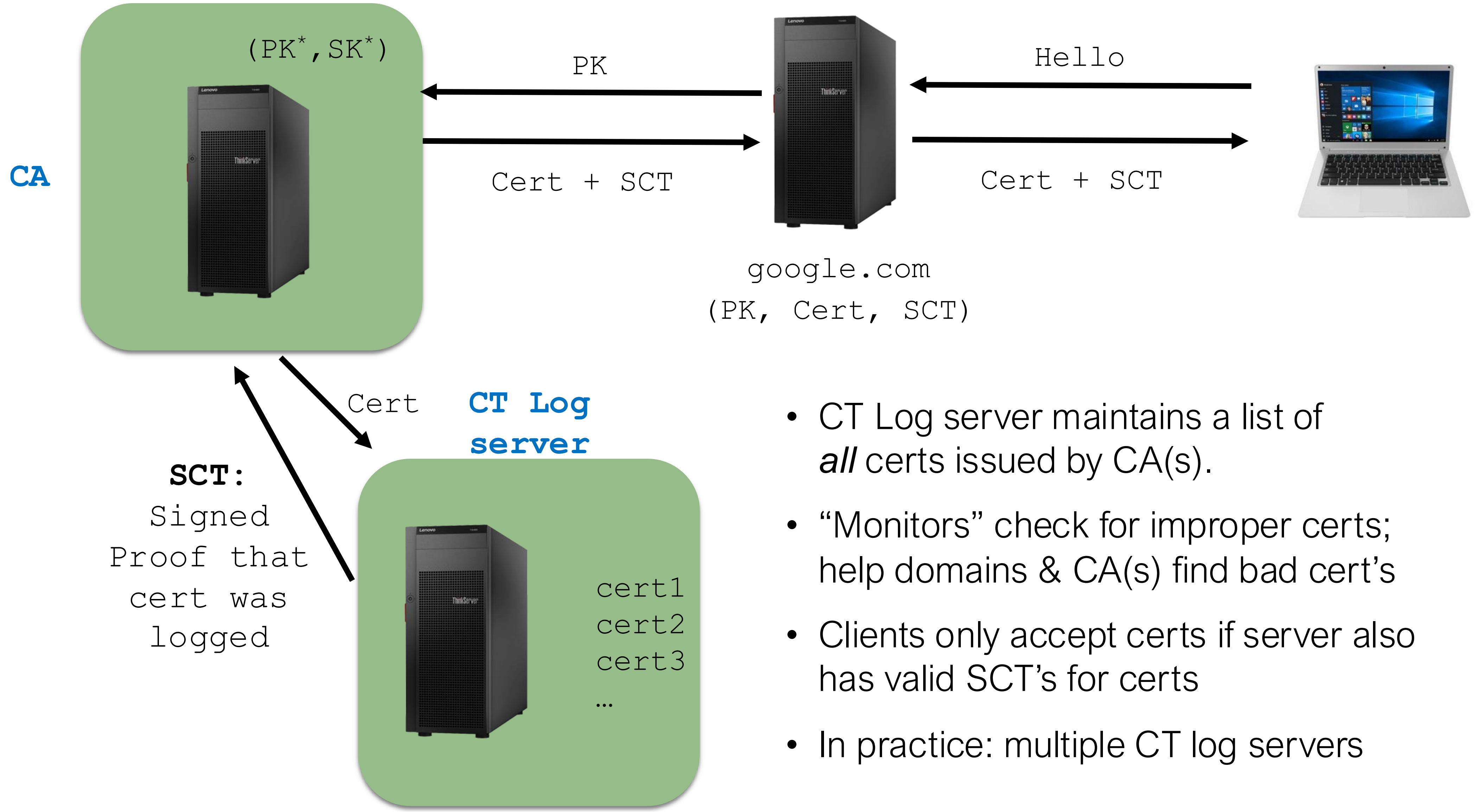
- Require all cert's added to public audit logs
- Domains & CA's can check audit logs for rogue certs & revoke them

Certificate Transparency (CT)

Simplified strategy to find certificates we should revoke:

- An auditor maintains a list (log) of every certificate ever issued
- Whenever a CA issues a cert, they submit (add) cert to this log
- Clients only accept a server’s cert if it appears on the log
- Each server (domain) can now monitor the logs to see if anyone (and who) issued a rogue certificate for them
  - If so, add the rogue cert to revocation lists
  - If CA has pattern of issuing rogue cert’s, ban them

Certificate Transparency (CT)



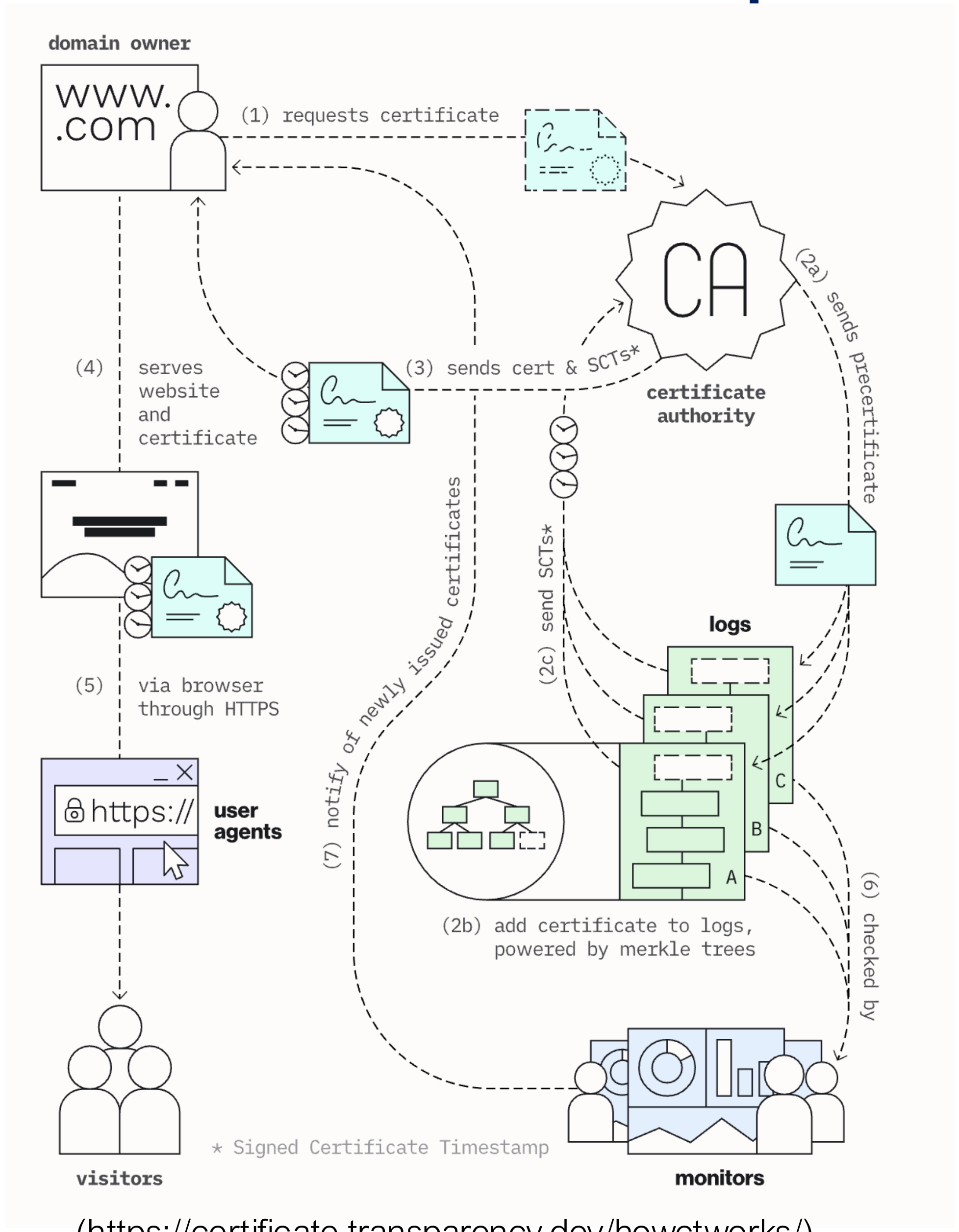
- CT Log server maintains a list of *all* certs issued by CA(s).
- "Monitors" check for improper certs; help domains & CA(s) find bad cert's
- Clients only accept certs if server also has valid SCT's for certs
- In practice: multiple CT log servers

Challenges with CT

- List is huuuuge (every issued cert… solution: temporal sharding)
- Trust the CT Log?
- (Monitors) Who checks the logs?
- Privacy (e.g., enterprise has private servers)?

CT Log Server
cert1
cert2
cert3
…

Cert Transparency & OCSP


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

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

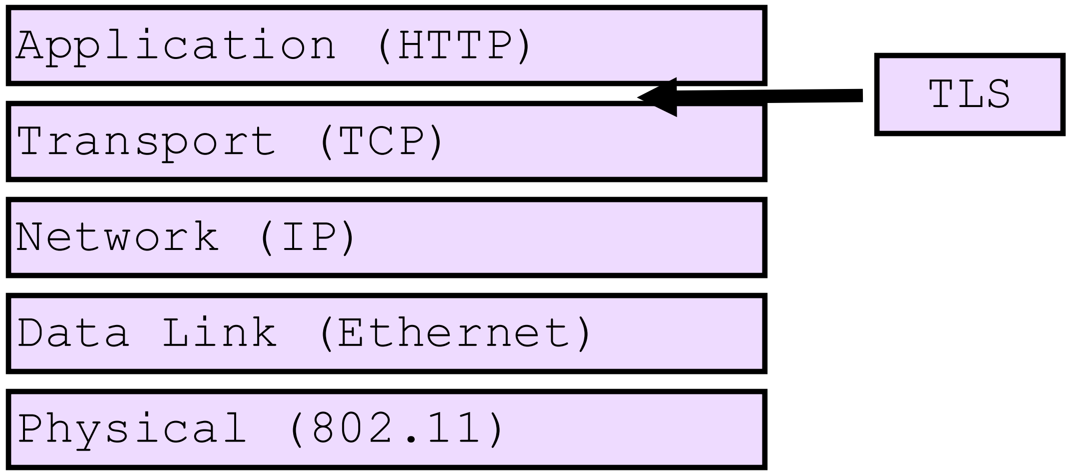
How do CT and OCSP compare?

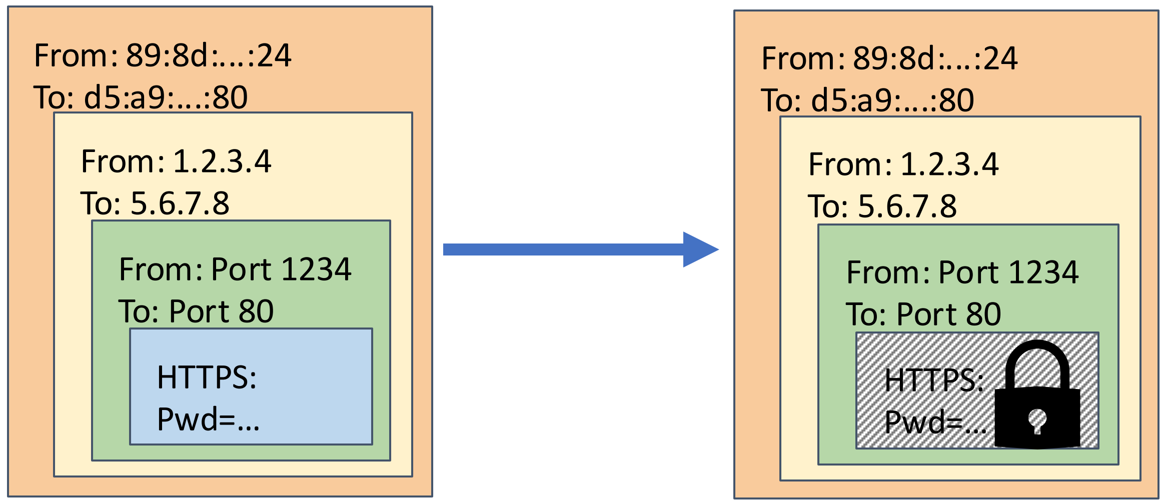
- OCSP: Allows clients to determine if a cert is valid
- CT: Allows domains (cert owners) and CA's to find malicious cert's

Outline

- Wrap-Up: DNS Security
- The Dream: Secure Channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS

TLS in the Protocol Stack

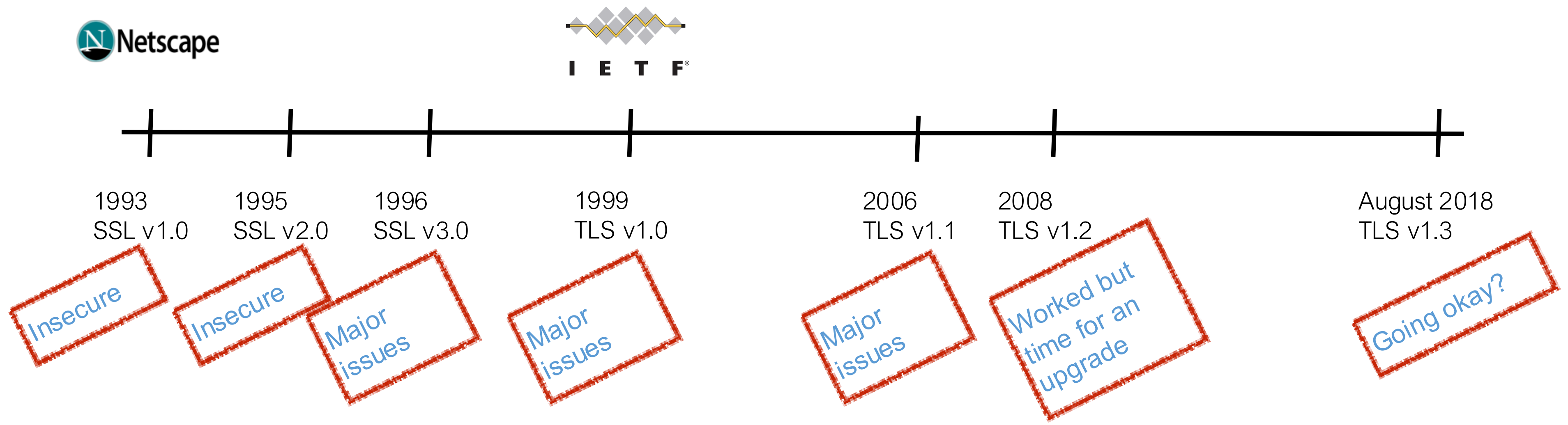




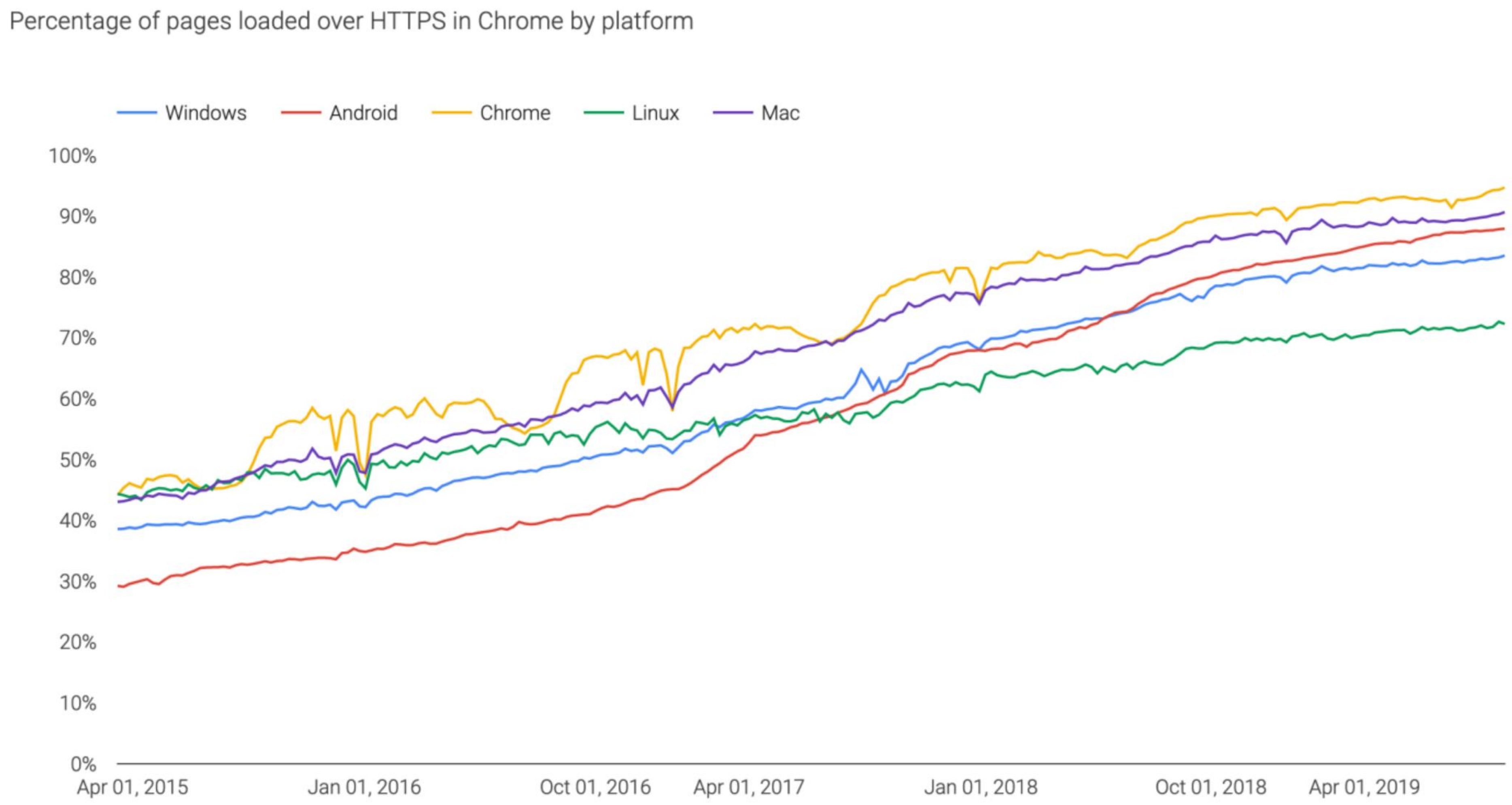
- **Goal:** Allow any application using TCP to transmit data with E2E security
- TLS takes requests from applications (e.g. browser speaking HTTP) and transmits them securely to another host on the Internet

History: SSL/TLS

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



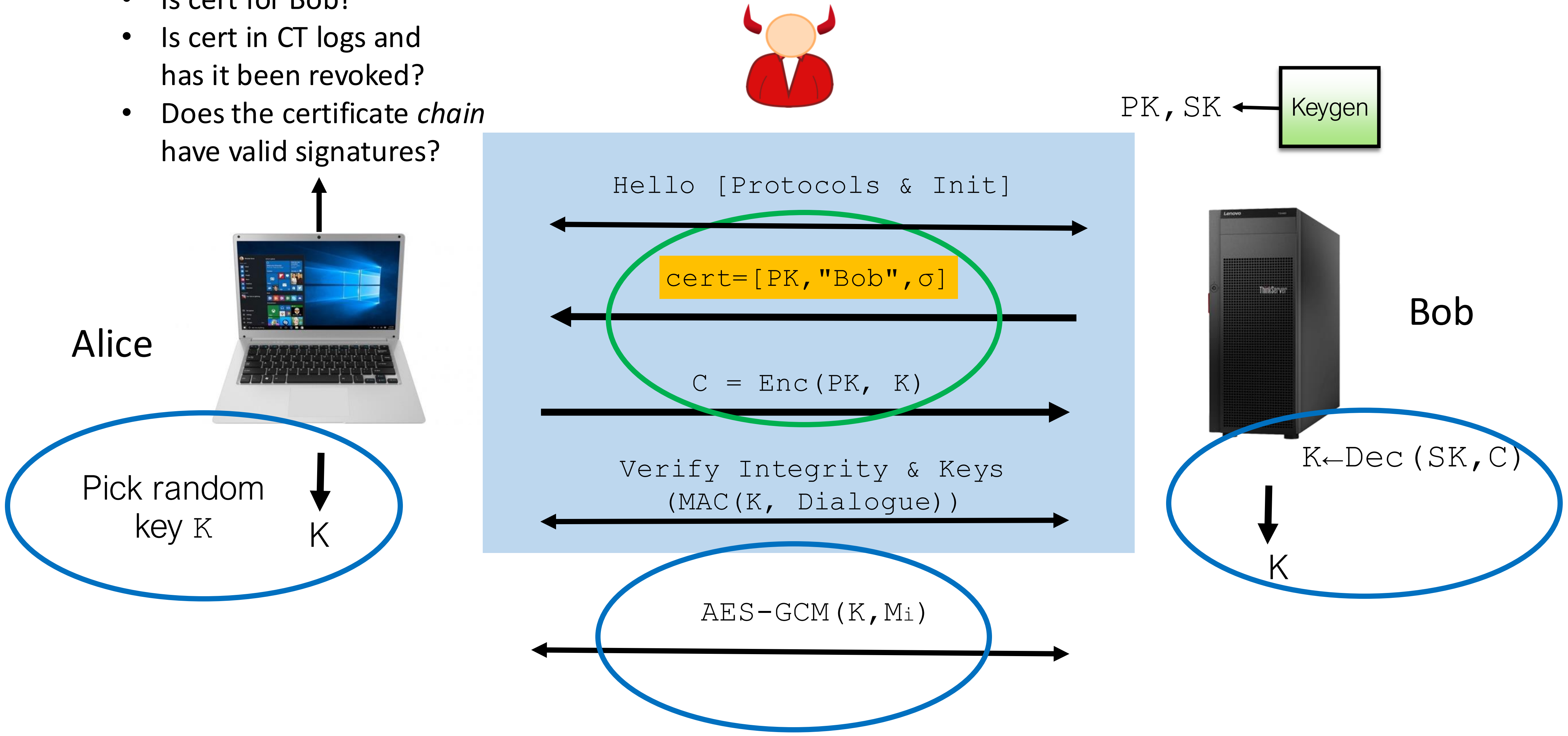
TLS Adoption (HTTPS)



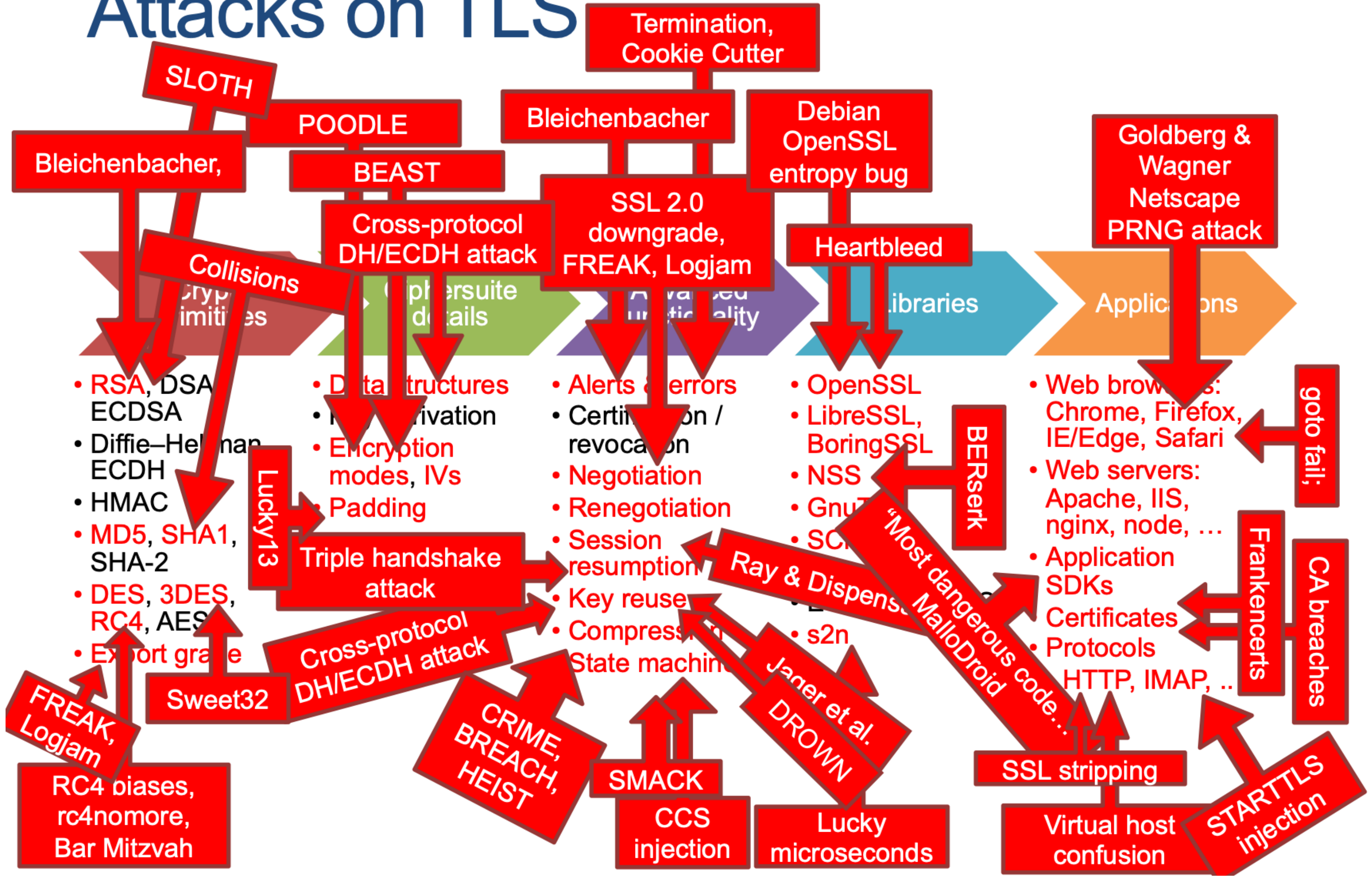
(Source: transparencyreport.google.com, via Matt Green)

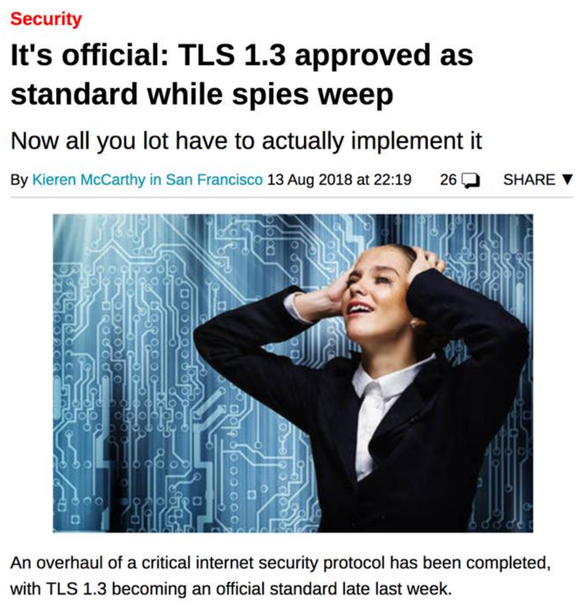
TLS Protocol: Very Similar to Our Template

- Is cert for Bob?
- Is cert in CT logs and has it been revoked?
- Does the certificate *chain* have valid signatures?



Attacks on TLS





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.