HTTPS, TLS, and Certificates

Niklas Carlsson, *Linköping University*

LINKÖPING UNIVERSITY

Web security

HTTPS and the Lock Icon

Goals for this lecture

Brief overview of HTTPS:

- How the SSL/TLS protocol works (very briefly)
- How to use HTTPS

Integrating HTTPS into the browser

- Lots of user interface problems to watch for

Threat Model: Network Attacker

Network Attacker:

- Controls network infrastructure: Routers, DNS
- Eavesdrops, injects, blocks, and modifies packets

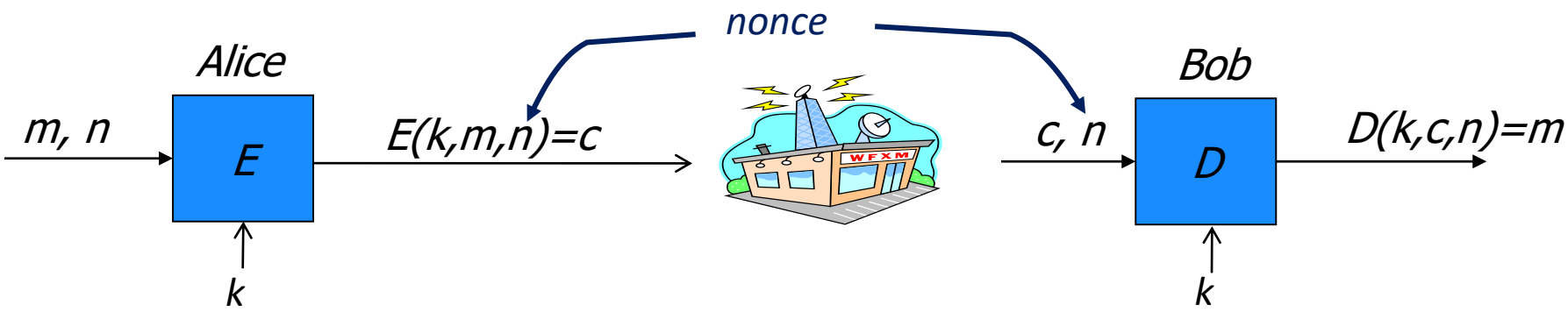
Examples:

- Wireless network at Internet Café
- Internet access at hotels (untrusted ISP)

Crypto Concepts

Symmetric key cryptography

Building block: symmetric cipher



E, D: cipher k: shared secret key (e.g., 128 bits)

m, c: plaintext, ciphertext n: nonce (non-repeating)

Encryption algorithm is publicly known

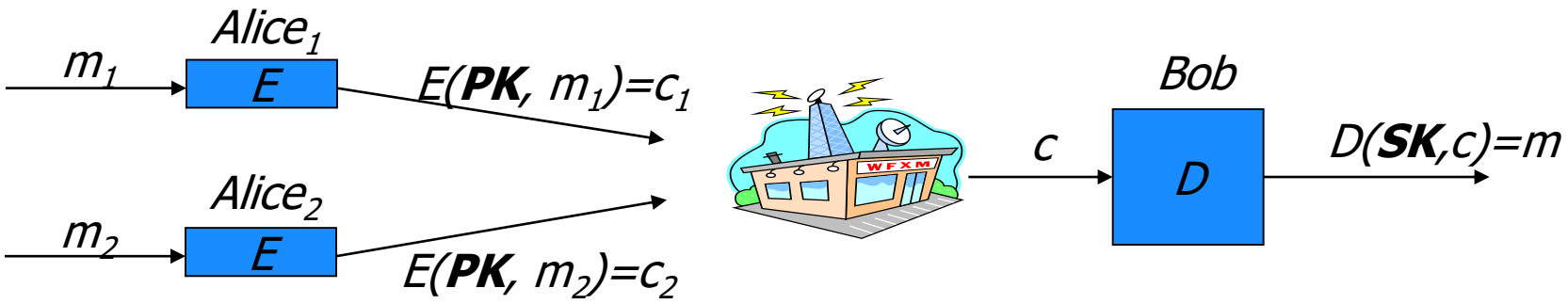
⇒ never use a proprietary cipher

Crypto Concepts

Public key cryptography

Public-key encryption

Tool for managing or generating symmetric keys



- E – Encryption alg. PK – <u>Public</u> encryption key
- D – Decryption alg. SK – <u>Private</u> decryption key

Algorithms E, D are publicly known.

Building block: trapdoor permutations

1. Algorithm KeyGen: outputs pk and sk

2. Algorithm F(pk, ·) : a one-way function
   - Computing y = F(pk, x) is easy
   - <u>One-way</u>: given random y finding x s.t. y = F(pk,x) is difficult

3. Algorithm F^{-1}(sk, ·) : Invert F(pk, ·) using trapdoor sk

$F^{-1}(sk, y) = x$

Example: RSA

1. KeyGen: generate two equal length primes p, q

   set $N \leftarrow p \cdot q$ (3072 bits $\approx$ 925 digits)

   set $e \leftarrow 2^{16}+1 = 65537$; $d \leftarrow e^{-1} \pmod{\varphi(N)}$

   pk = (N, e) ; sk = (N, d)

Re. choice of e, d: A mote general description is to pick e,d such that:

1. $1 < e < (p-1)(q-1)$ and $(p-1)(q-1)$ is not divisible by e
2. $d \cdot e \equiv 1 \bmod (p-1)(q-1)$

Example: RSA

1. KeyGen: generate two equal length primes p, q

   set $N \leftarrow p \cdot q$ (3072 bits $\approx$ 925 digits)

   set $e \leftarrow 2^{16}+1 = 65537$; $d \leftarrow e^{-1} \pmod{\varphi(N)}$

   pk = (N, e) ; sk = (N, d)

2. RSA(pk, x) : $x \rightarrow (x^e \bmod N)$

   Inverting this function is believed to be as hard as factoring N

3. RSA^{-1}(sk, y) : $y \rightarrow (y^d \bmod N)$

Intuitive example (but requires lots of hidden math+care …):

$y^d \bmod N = (x^e \bmod N)^d \bmod N = x^{ed} \bmod N = x \bmod N = x$

Public Key Encryption with a TDF

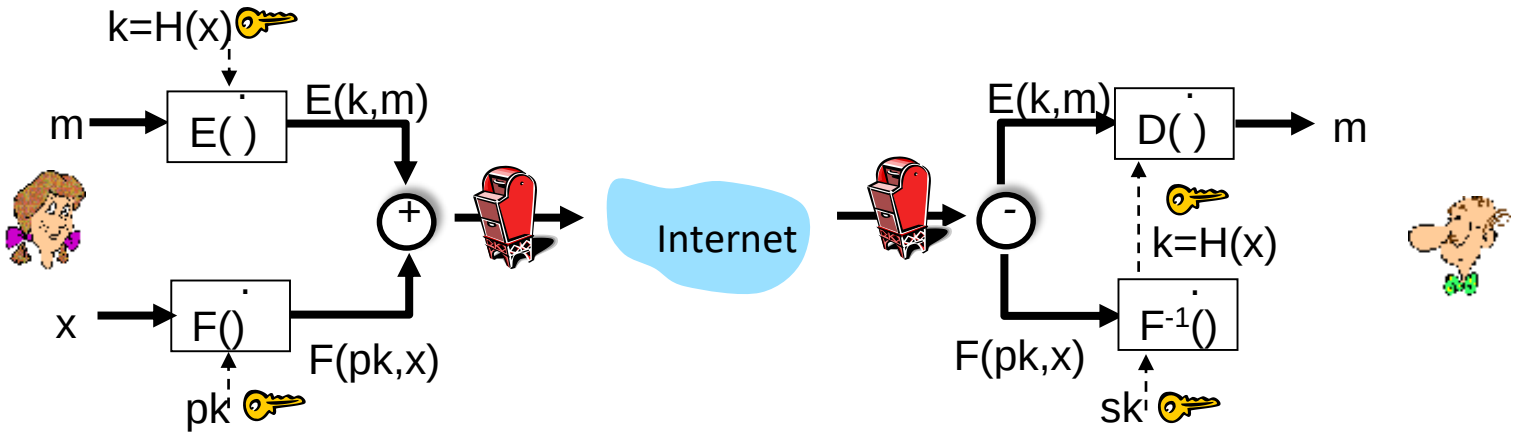
KeyGen: generate pk and sk

c_0	c_1

Encrypt(pk, m):

- choose random $x \in$ domain(F) and set $k \leftarrow H(x)$
- $c_0 \leftarrow F(pk, x)$, $c_1 \leftarrow E(k, m)$ (E: symmetric cipher)
- send $c = (c_0, c_1)$

Decrypt(sk, $c=(c_0,c_1)$): $x \leftarrow F^{-1}(sk, c_0)$, $k \leftarrow H(x)$, $m \leftarrow D(k, c_1)$



Digital signatures

Goal: bind document to author

- Problem: attacker can copy Alice's sig from one doc to another

Main idea: make signature depend on document

Example: signatures from trapdoor functions (e.g., RSA)

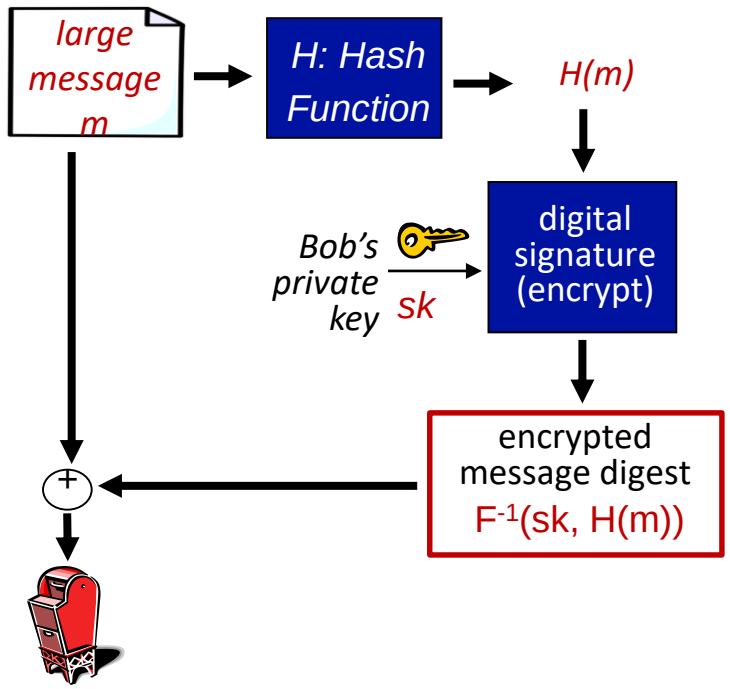
sign(sk, m) := F^{-1} (sk, H(m))

verify(pk, m, sig) := accept if F(pk, sig) = H(m)

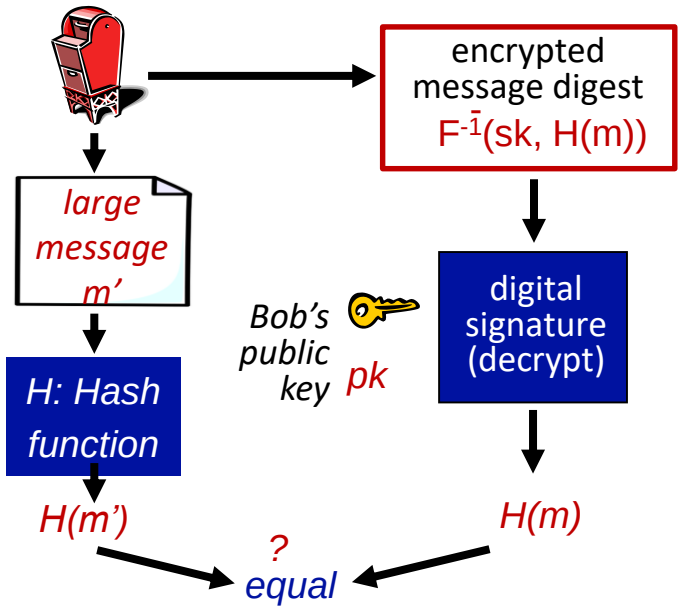
Note: With RSA we have $F(pk, F^{-1}(sk, m)) = F^{-1}(sk, F(pk, m)) = m$

Digital signature

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Note: With RSA we have $F(pk, F^{-1}(sk, m)) = F^{-1}(sk, F(pk, m)) = m$

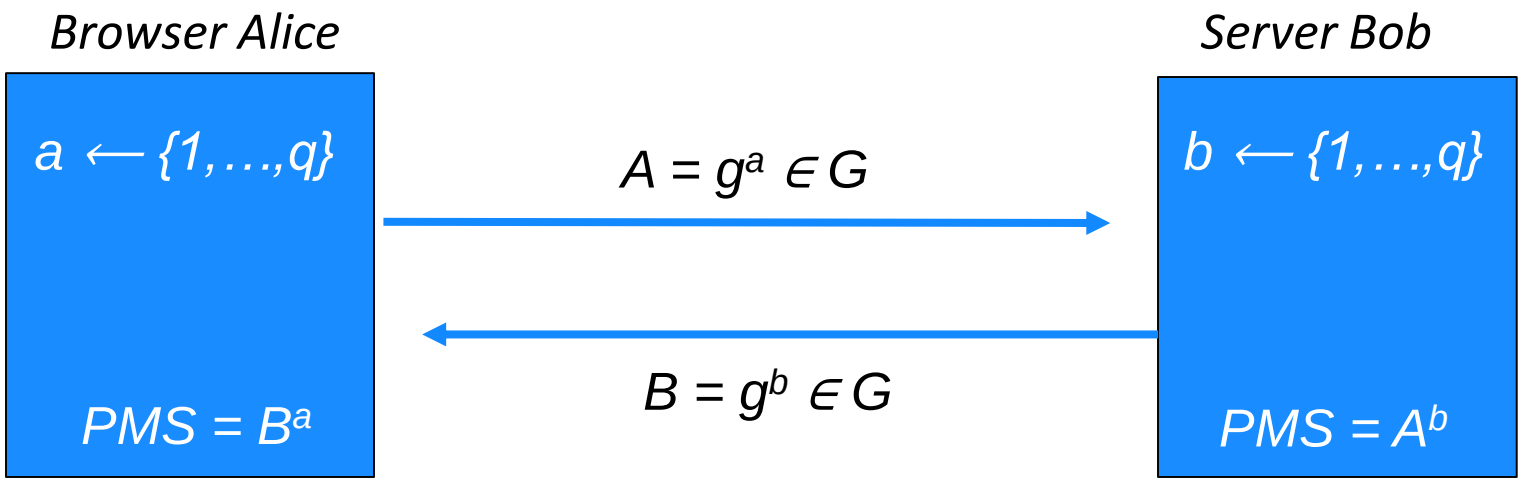
TLS

Building blocks

TLS overview: (1) DH key exchange

Anonymous key exchange secure against eavesdropping:

The Diffie-Hellman protocol in a group $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$



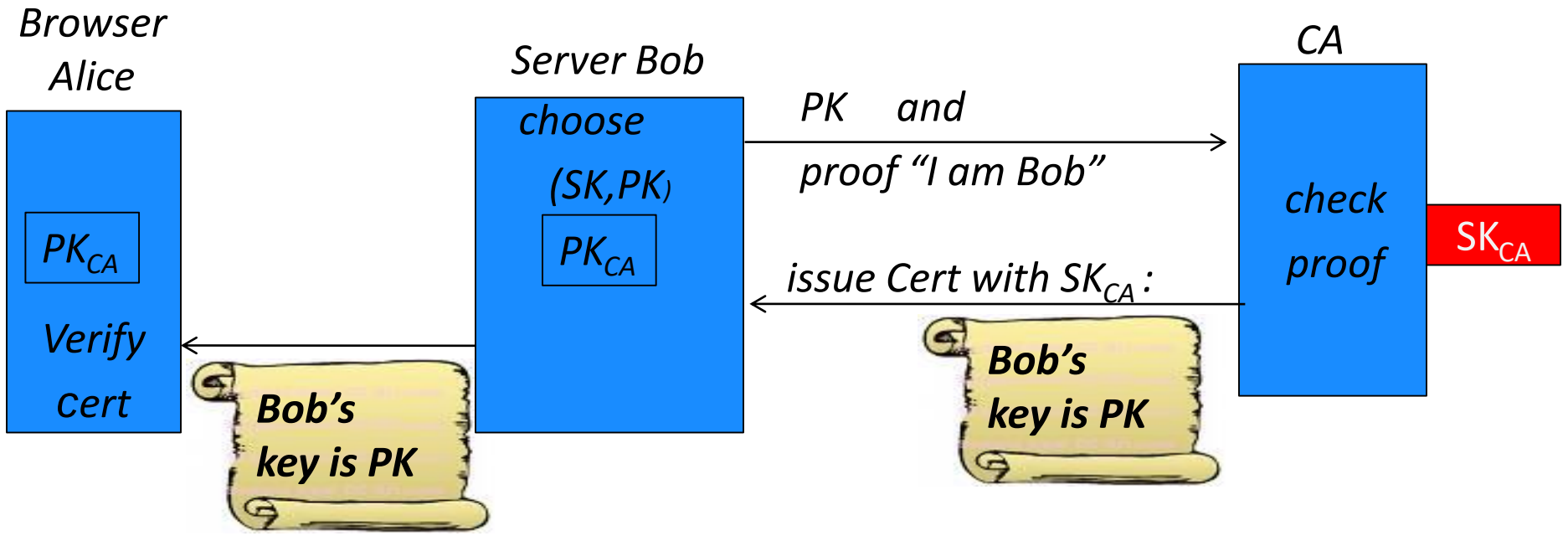
$$PreMasterSecret = g^{ab} = (g^b)^a = B^a = (g^a)^b = A^b$$

Used to establish a shared secret

Group G is publicly known

TLS overview: (2) Certificates

How does Alice (browser) obtain PK_{Bob} ?

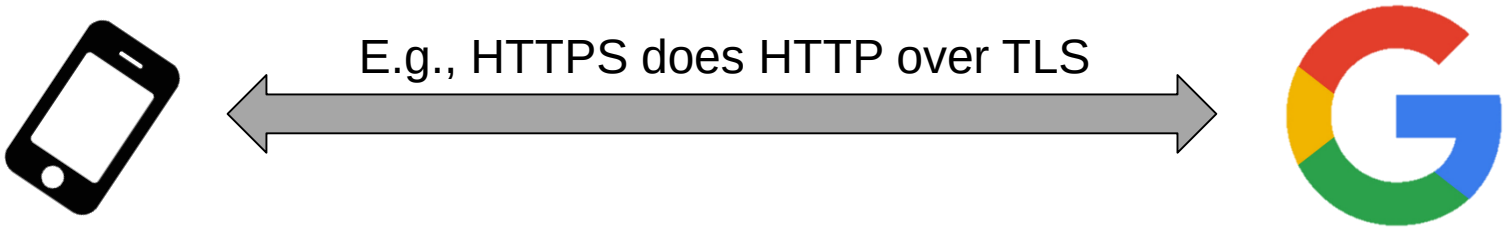


Bob uses Cert for an extended period (e.g. *one year)*

Certificates

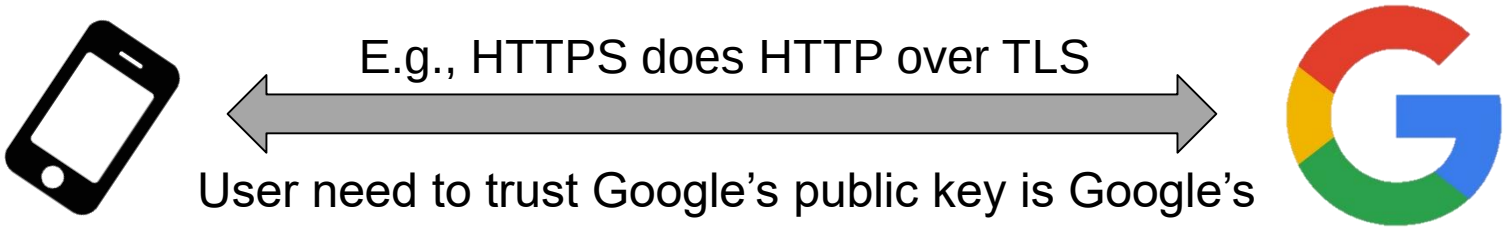
Motivation and high-level problem

- Private and confidential communication important
  -
  -



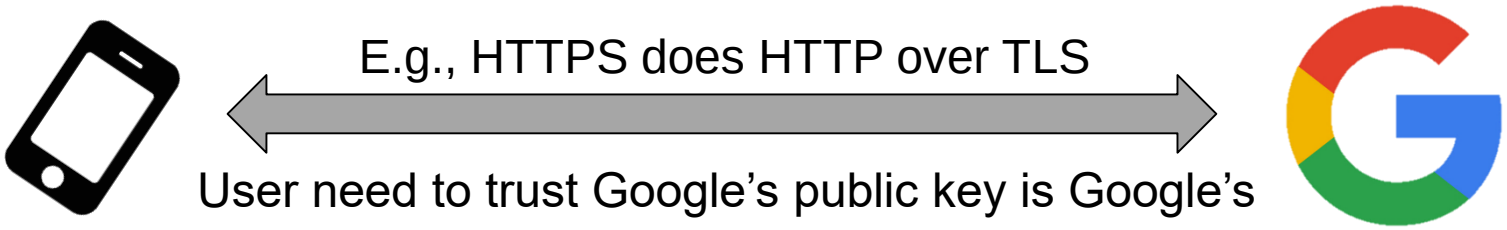
Motivation and high-level problem

- Private and confidential communication important
  -
  -



Motivation and high-level problem

- Private and confidential communication important
  - Billions of devices
  - Millions of services
-
  -

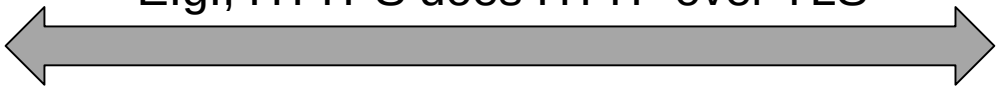


Motivation and high-level problem

- Private and confidential communication important
  - Billions of devices
  - Millions of services
-
  -

User need to trust FB's public key is FBs

E.g., HTTPS does HTTP over TLS



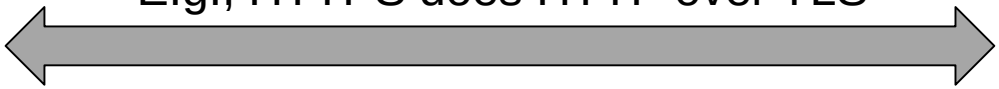
User need to trust Google's public key is Google's

Motivation and high-level problem

- Private and confidential communication important
  - Billions of devices
  - Millions of services
-
  -

User need to **trust** FB's public key is FB's

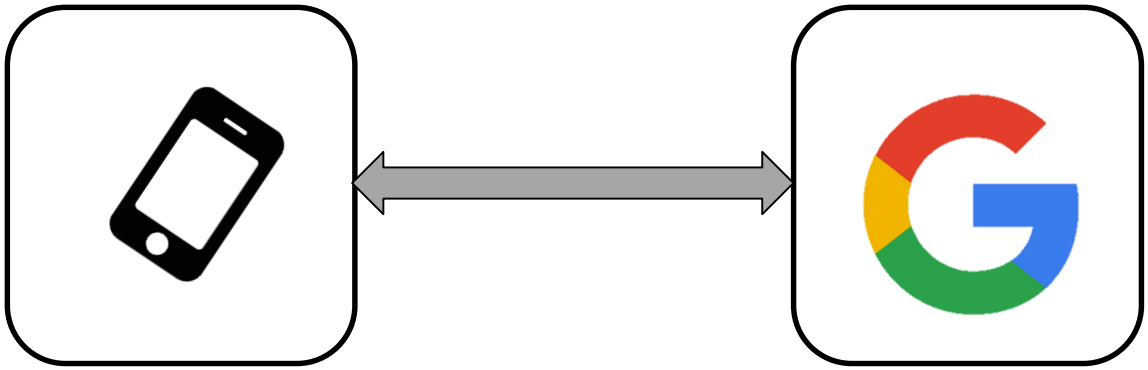
E.g., HTTPS does HTTP over TLS



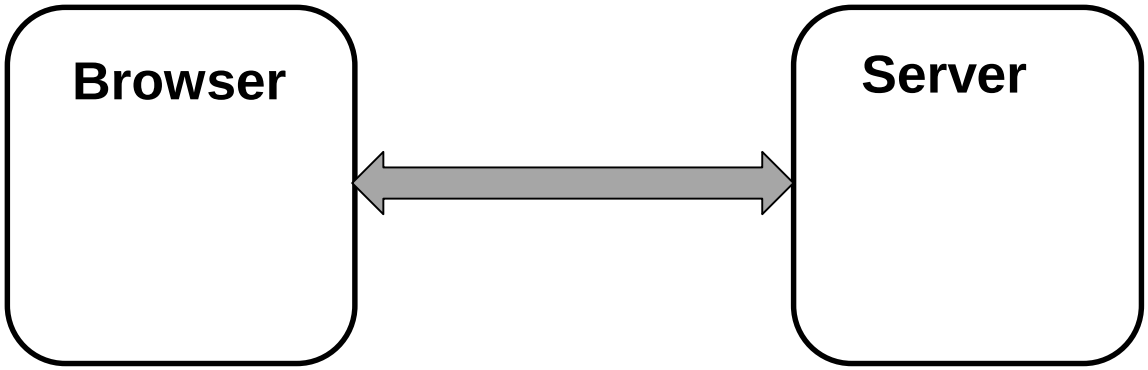
User need to **trust** Google's public key is Google's

Certification of public keys

Certification of public keys

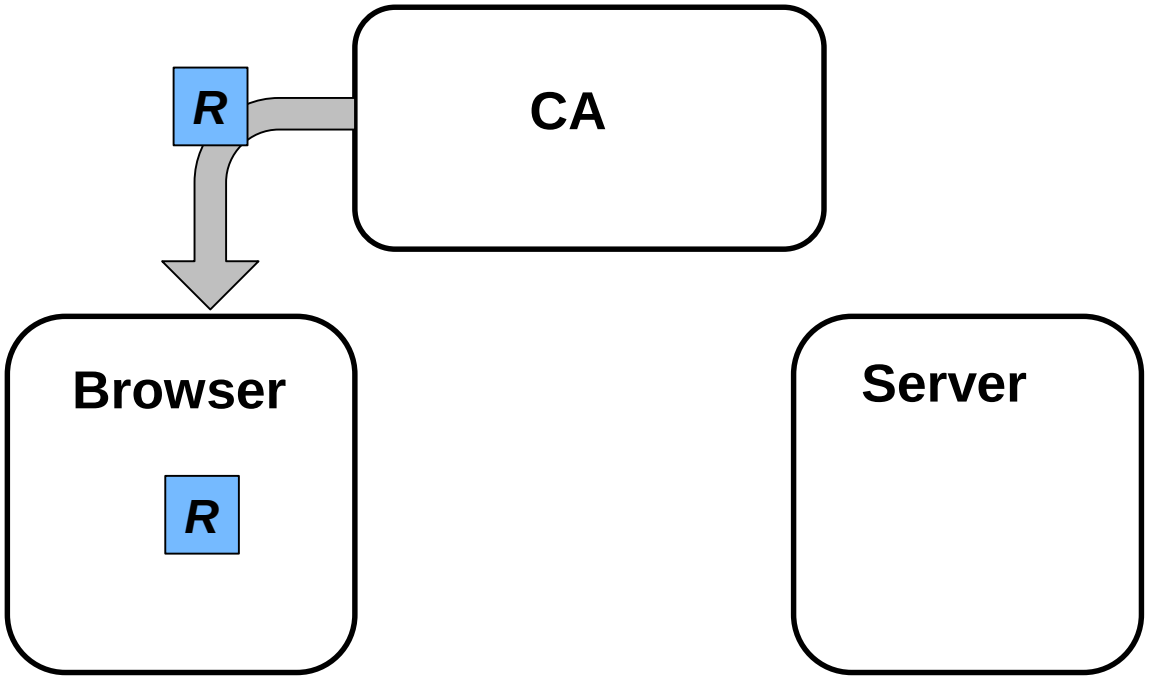


Certification of public keys



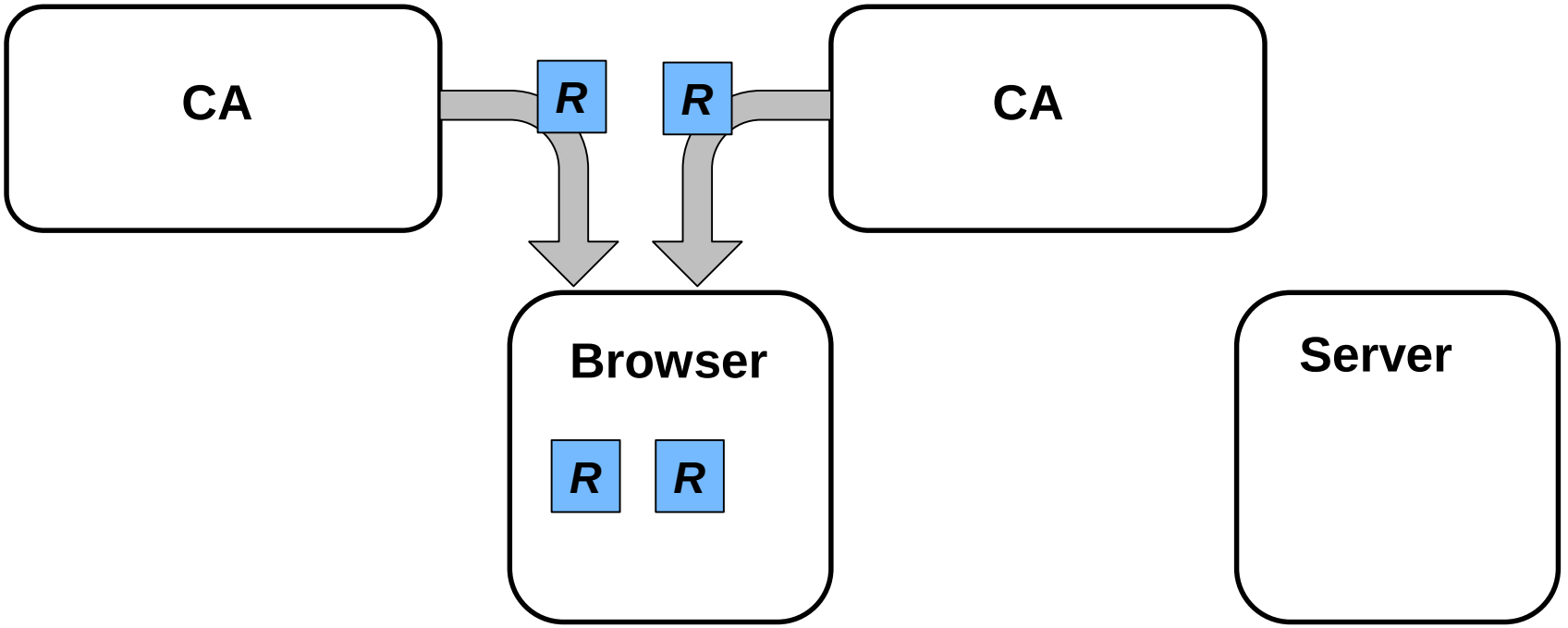
Certification of public keys

- Browsers have trust stores with root certs (of CAs)



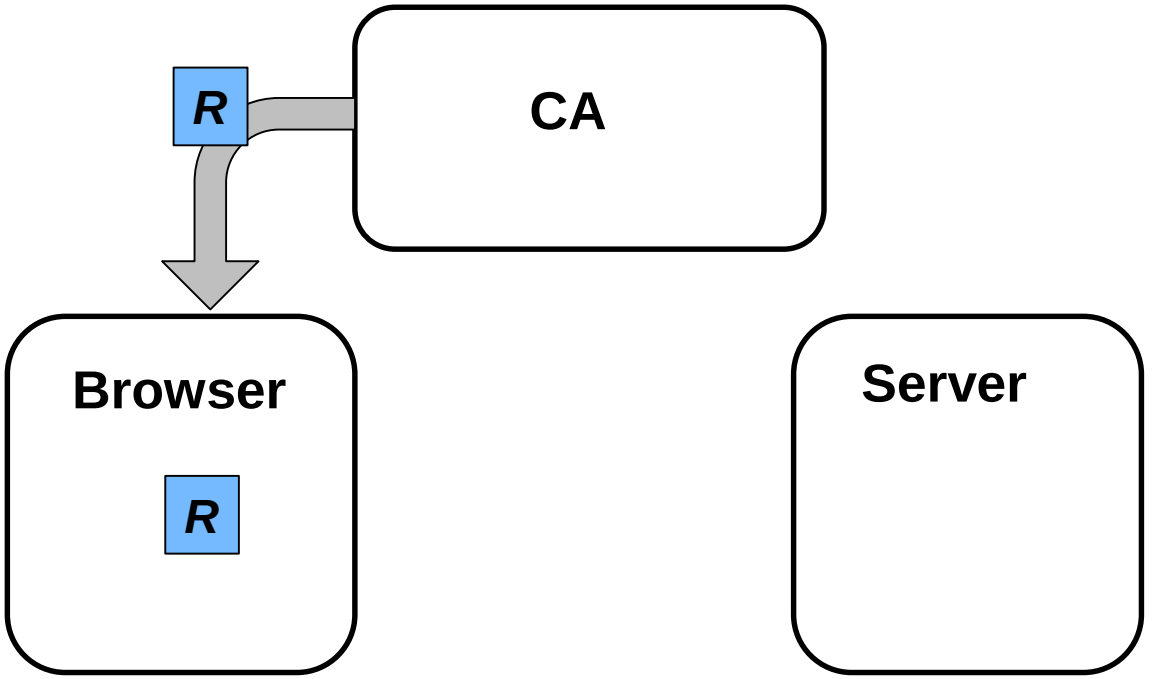
Certification of public keys

- Browsers have trust stores with root certs (of CAs)



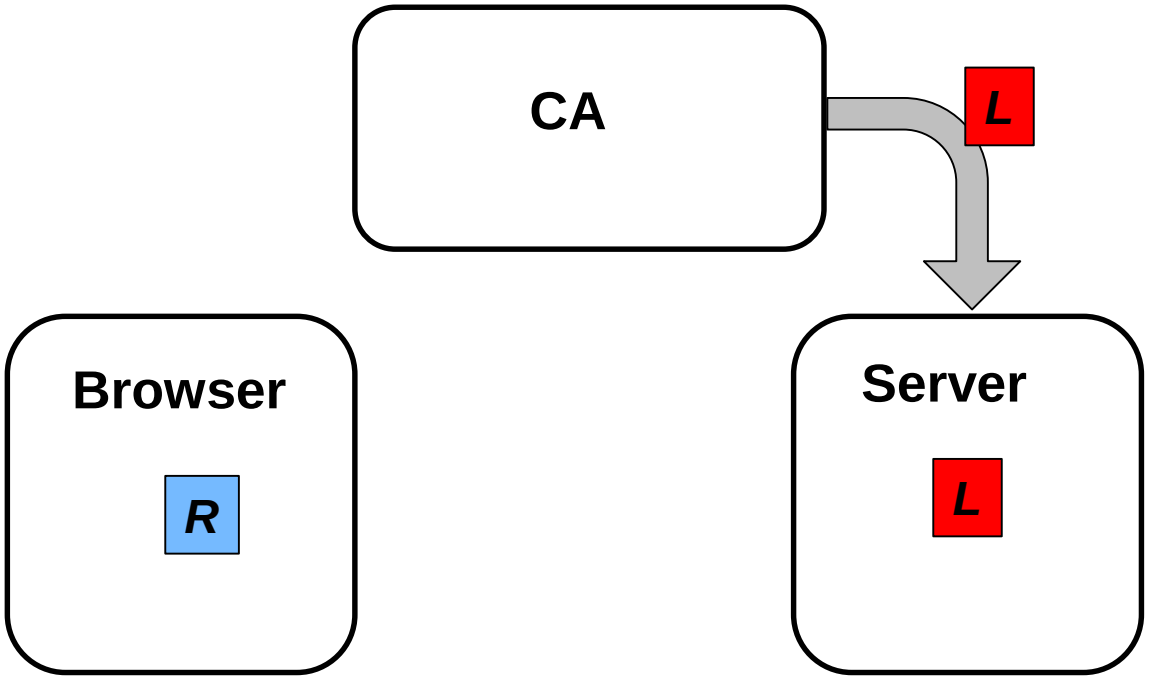
Certification of public keys

- Browsers have trust stores with root certs (of CAs)



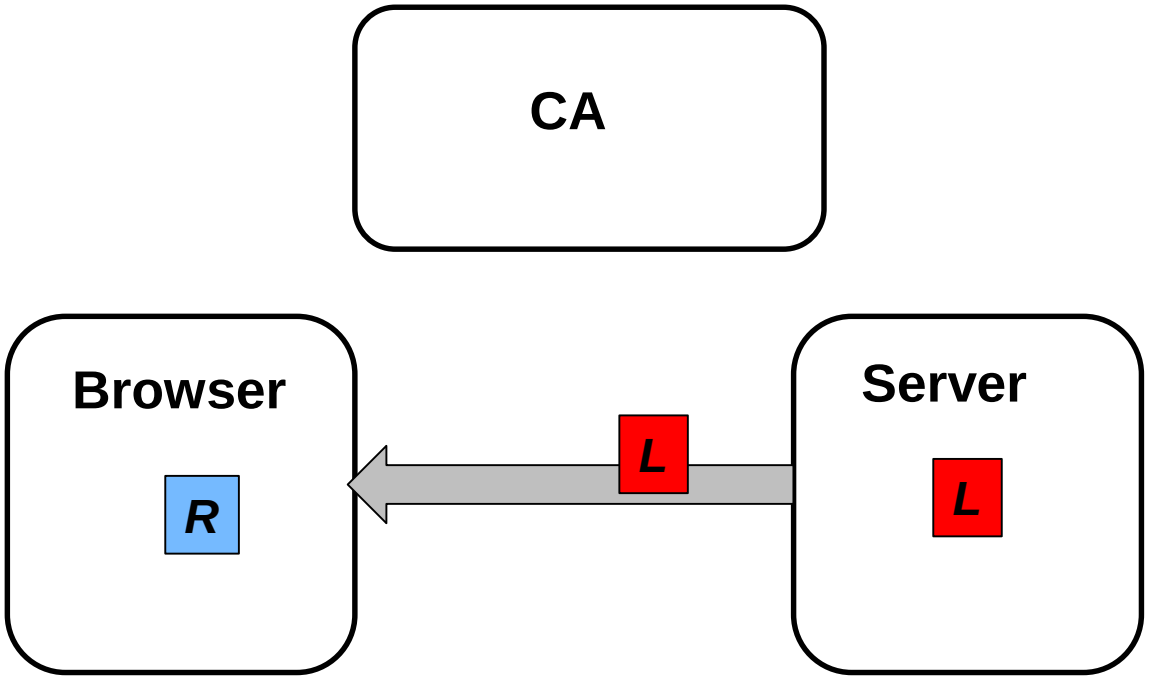
Certification of public keys

- Browsers have trust stores with root certs (of CAs)
- CAs use private key to sign certs for servers/domains
  - Certs are proof that public key belongs to server/domain



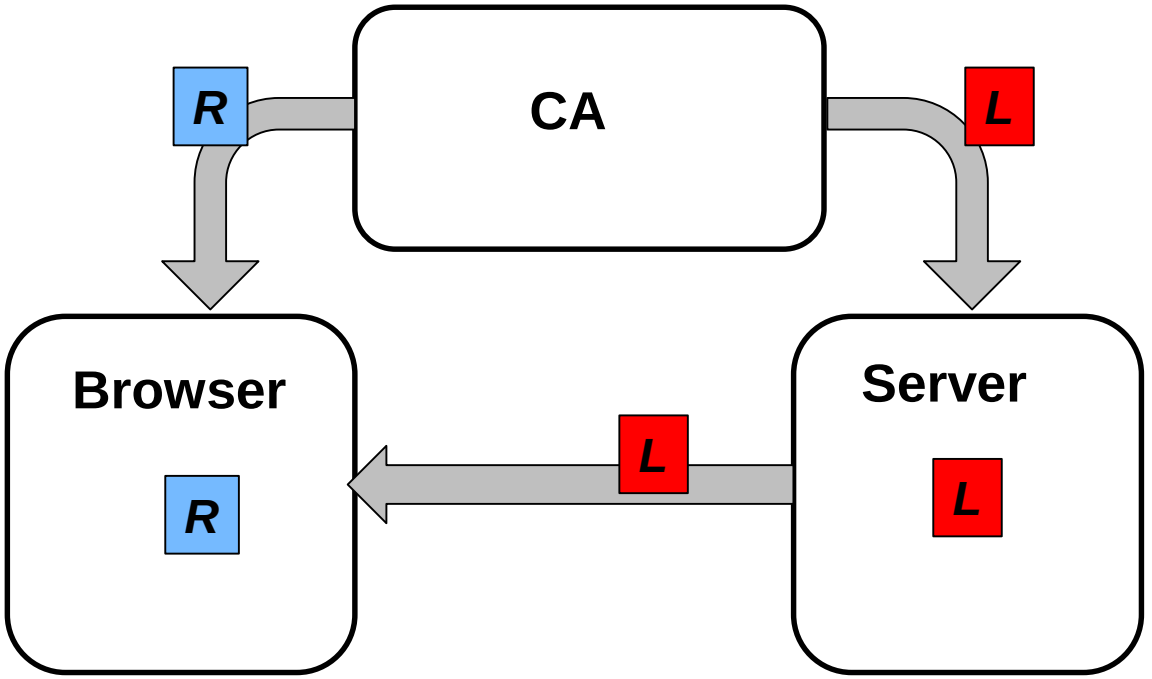
Certification of public keys

- Browsers have trust stores with root certs (of CAs)
- CAs use private key to sign certs for servers/domains
  - Certs are proof that public key belongs to server/domain
  - Signature of certs can be validated using keys in root store

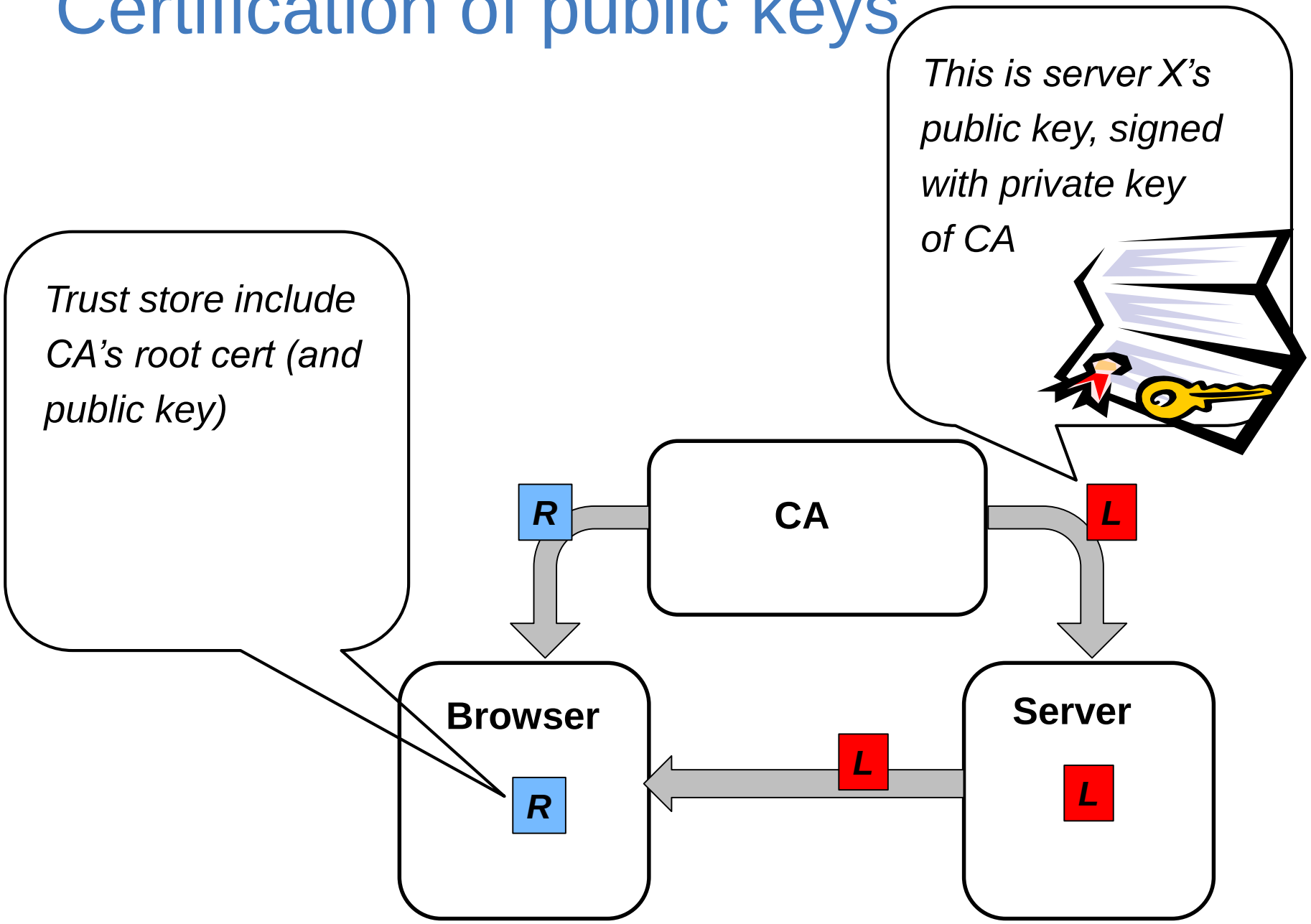


Certification of public keys

- Browsers have trust stores with root certs (of CAs)
- CAs use private key to sign certs for servers/domains
  - Certs are proof that public key belongs to server/domain
  - Signature of certs can be validated using keys in root store

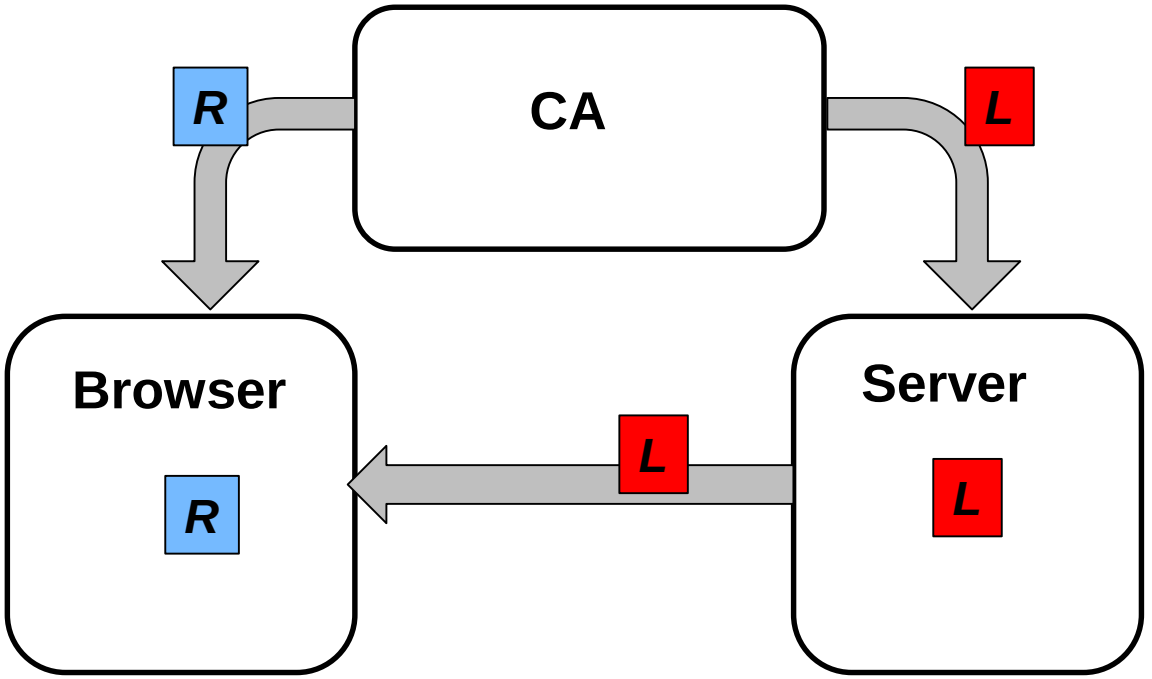


Certification of public keys



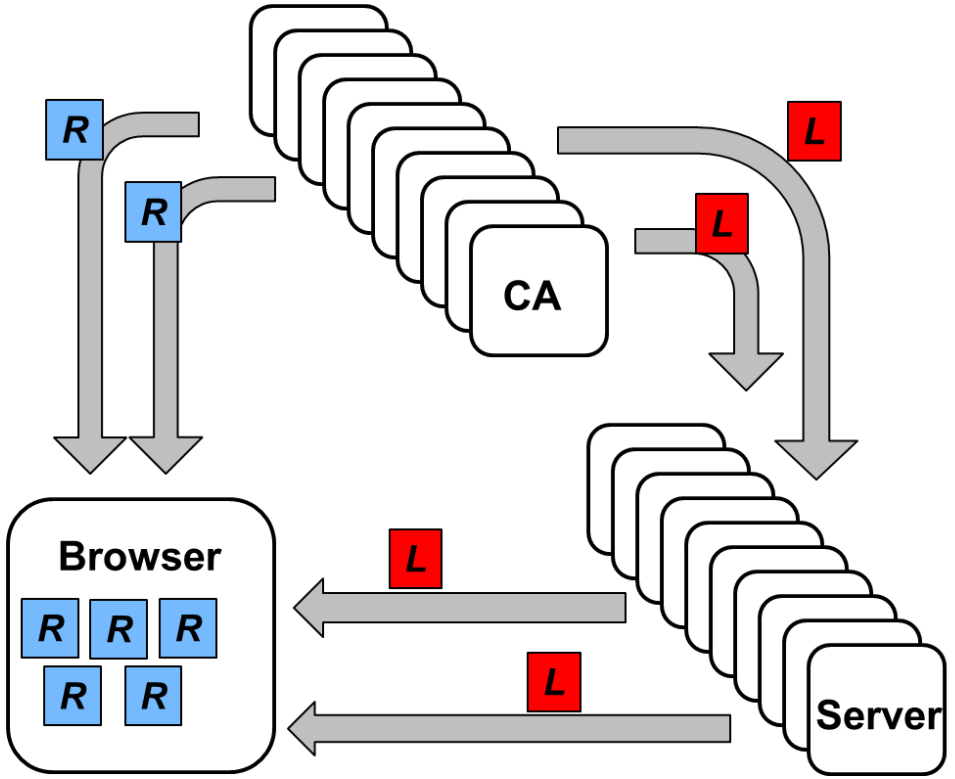
Certification of public keys

- Browsers have trust stores with root certs (of CAs)
- CAs use private key to sign certs for servers/domains
  - Certs are proof that public key belongs to server/domain
  - Signature of certs can be validated using keys in root store
-
  -
  -



Certification of public keys

- Browsers have trust stores with root certs (of CAs)
- CAs use private key to sign certs for servers/domains
  - Certs are proof that public key belongs to server/domain
  - Signature of certs can be validated using keys in root store
- In practice, many
  - Many CAs, servers
  - Varying trust+security



Sample certificate:

www.bankofamerica.com
Issued by: Entrust Certification Authority - L1M
Expires: Thursday, June 6, 2019 at 9:57:43 AM Pacific Daylight Time
This certificate is valid

Organization	Bank of America Corporation
Business Category	Private Organization
Organizational Unit	eComm Network Infrastructure
Serial Number	2927442
Common Name	www.bankofamerica.com

Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	256 bytes : BE E5 23 1D 17 9A 68 05 ...
Exponent	65537
Key Size	2,048 bits
Key Usage	Encrypt, Verify, Wrap, Derive

Signature *(by CA)* 256 bytes : 39 D0 09 7E 99 C6 B3 01 ...

Certificates on the web

Subject's CommonName can be:

- An explicit name, e.g. cs.stanford.edu , or
- A wildcard cert, e.g. *.stanford.edu or cs*.stanford.edu

matching rules:

"*" must occur in leftmost component, does not match "."

example: *.a.com matches x.a.com but not y.x.a.com

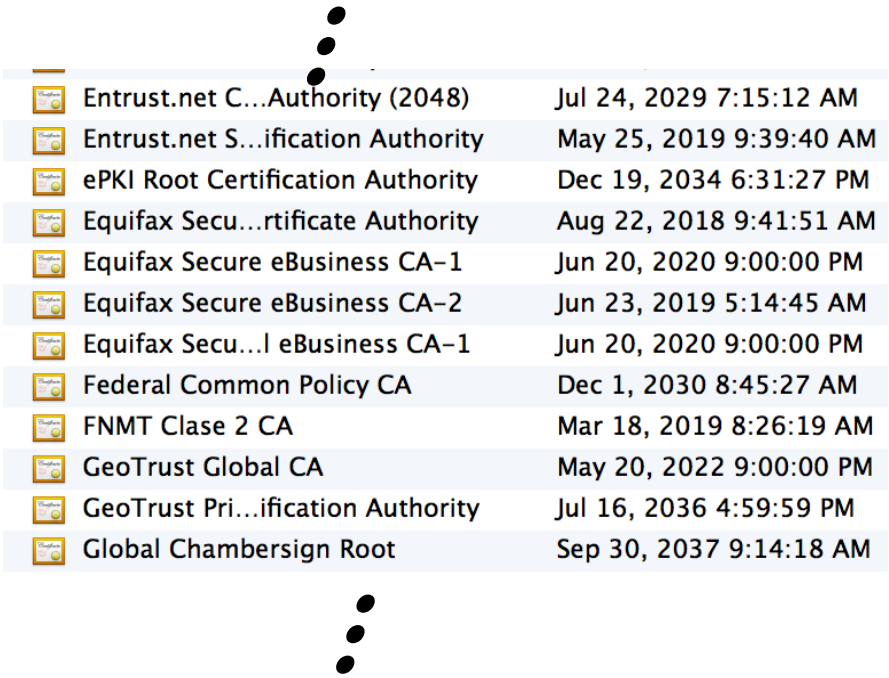
(as in RFC 2818: "HTTPS over TLS")

Certificate Authorities (CAs) and root/trust stores

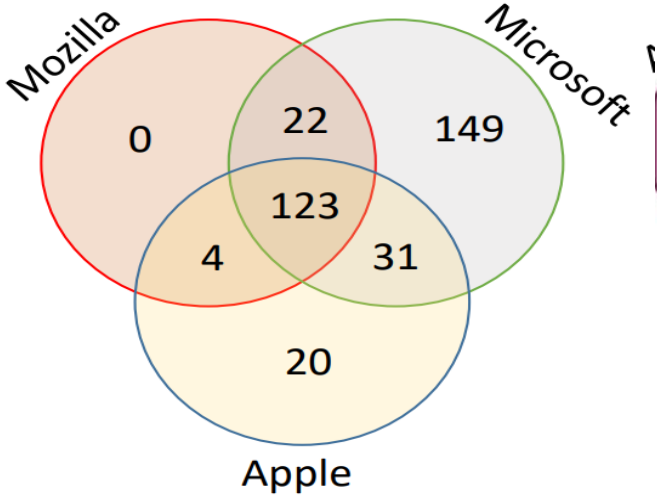
Browsers accept certificates from a large number of CAs

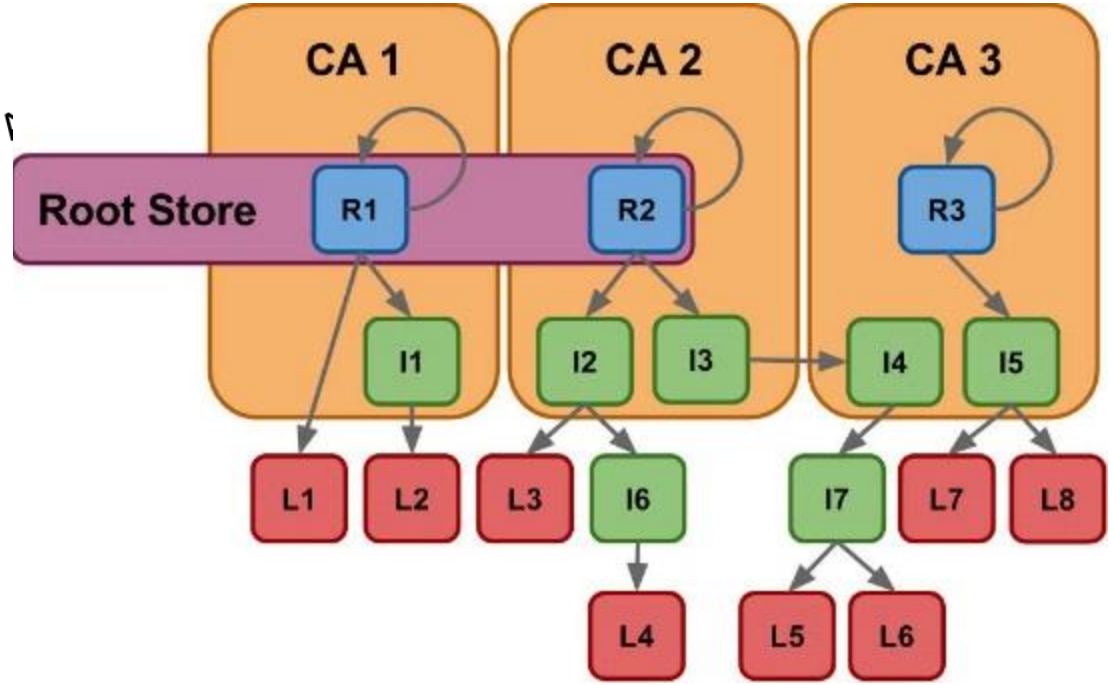
Top level CAs ≈ 60

Intermediate CAs ≈ 1200



Trust landscape





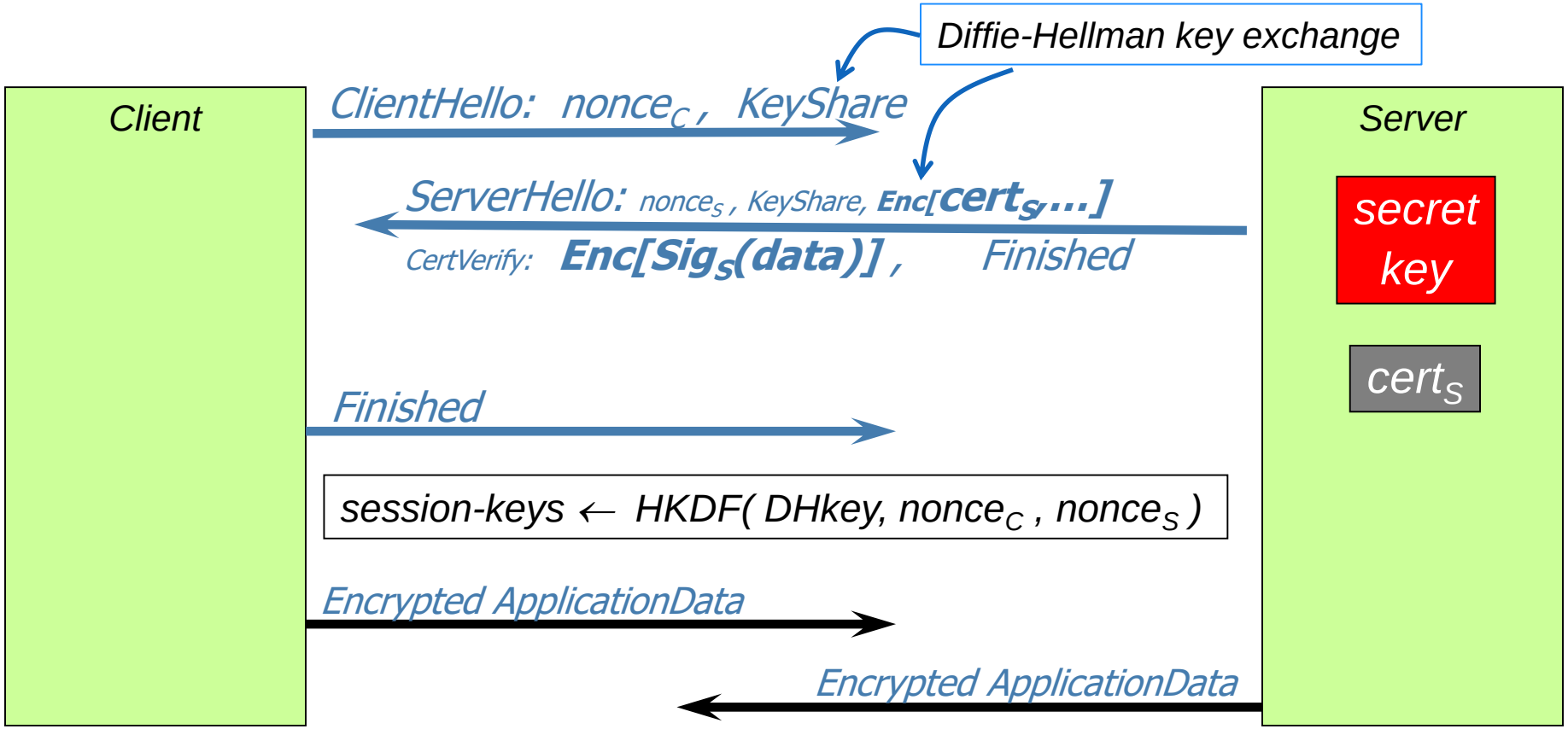
[Korzhitskii & Carlsson, 2020]

- Delegation of trust to intermediates (Ii)
- Browsers trust that the servers that can present certs (Li) that map to (trusted) root certs are who they claim to be
- Impersonation
  - Any trusted CA (Ri) or intermediate (Ii) can issue rogue certs
  - Very difficult to know all certs issued in once name

TLS 1.3

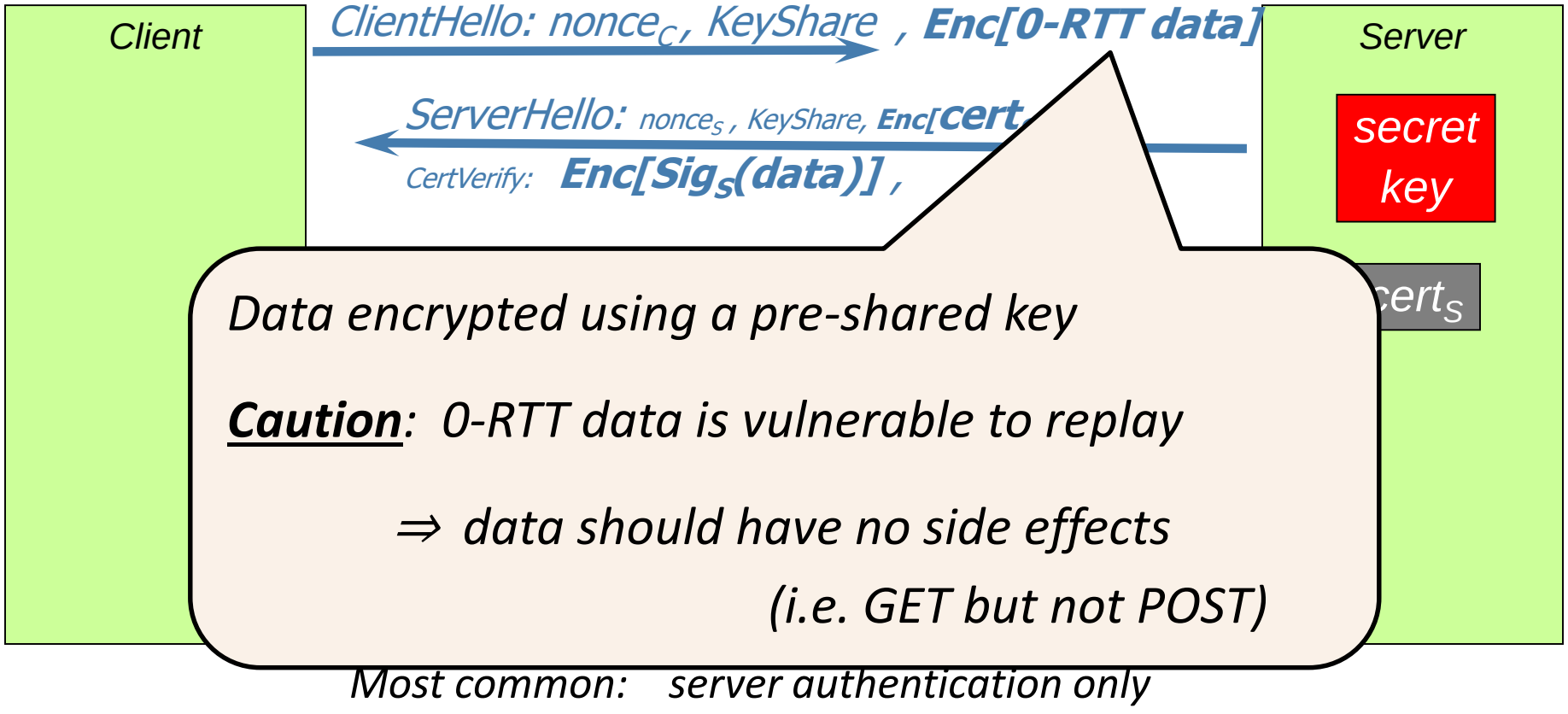
Back to TLS

TLS 1.3 session setup (simplified)



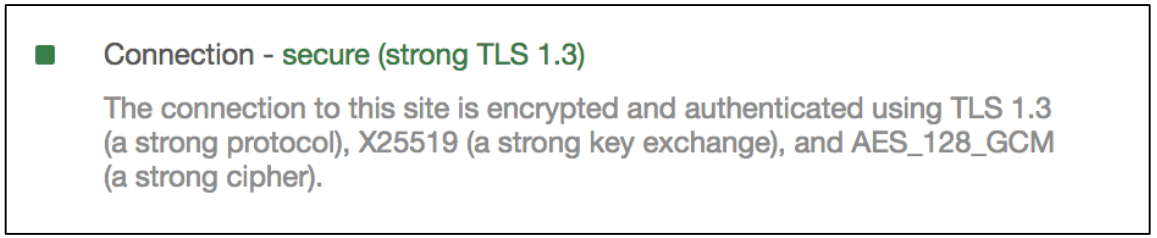
Most common: server authentication only

TLS 1.3 session setup: optimization (and caution)



Most common: server authentication only

Properties



Gmail

Nonces: prevent replay of an old session

Forward secrecy: server compromise does not expose old sessions

Some identity protection: certificates are sent encrypted

One sided authentication:

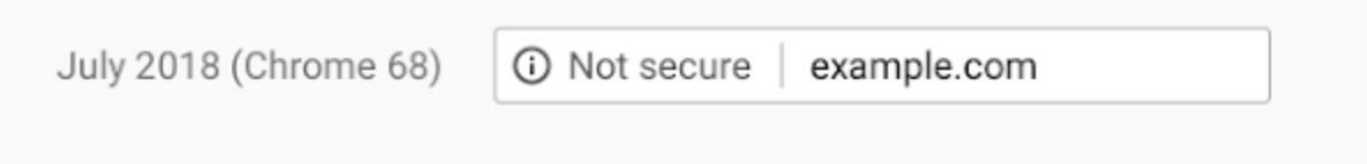
- Browser identifies server using server-cert
- TLS has support for mutual authentication
  - Rarely used: requires a client pk/sk and client-cert

HTTPS for all web traffic?

Old excuses:

- Crypto slows down web servers (not true anymore)
- Some ad-networks still do not support HTTPS
  - reduced revenue for publishers

Since July 2018: Chrome marks HTTP sites as insecure



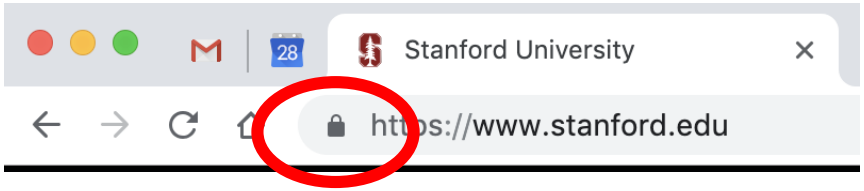
Chrome's gradual blocking of mixed content

https://blog.chromium.org/2020/02/protecting-users-from-insecure.html

HTTPS in the Browser

LiU EXPANDING REALITY

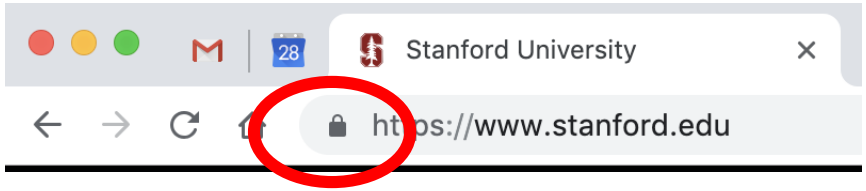
The lock icon: TLS indicator



Intended goal:

- Provide user with identity of page origin
- Indicate to user that page contents were not viewed or modified by a **network attacker**

When is the (basic) lock icon displayed



Extension	Subject Alternative Name (2.5.29.17)
Critical	NO
DNS Name	*.google.com
DNS Name	*.android.com
DNS Name	*.appengine.google.com
DNS Name	*.cloud.google.com
DNS Name	*.google-analytics.com
DNS Name	*.google.ca
DNS Name	*.google.cl
DNS Name	*.google.co.in
DNS Name	*.google.co.jp
DNS Name	*.google.co.uk
DNS Name	*.google.com.ar
DNS Name	*.google.com.au

All elements on the page fetched using HTTPS

For all elements:

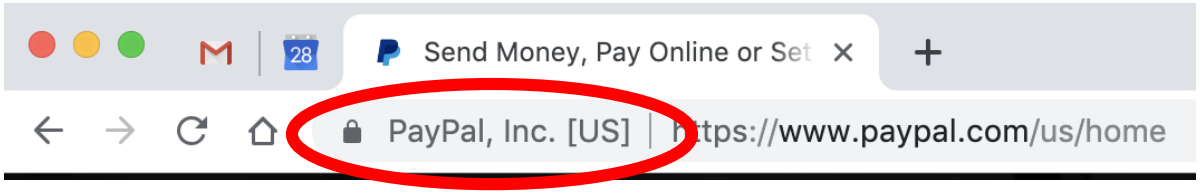
- HTTPS cert issued by a CA trusted by browser
- HTTPS cert is valid (e.g. not expired)
- Domain in URL matches:
  CommonName or **SubjectAlternativeName** in cert

The lock UI: Extended Validation (EV) Certs

Harder to obtain than regular certs

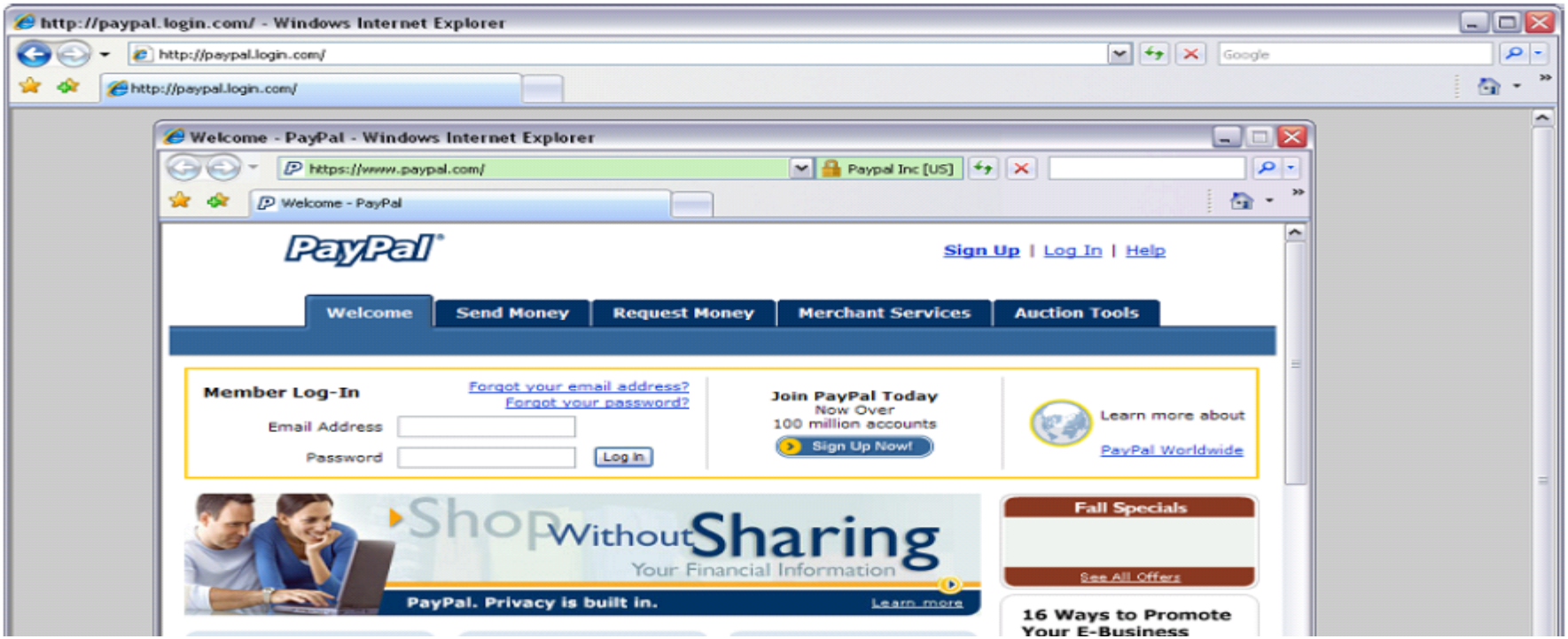
- requires human at CA to approve cert request
- no wildcard certs (e.g. *.stanford.edu)

Helps block “semantic attacks”: www.bankofthevvest.com



This UI is ineffective: removed from Chrome in 2019.

A general UI attack: picture-in-picture



Trained users are more likely to fall victim to this [JSTB'07]

HTTPS and login pages: incorrect usage

Suppose user lands on HTTP login page.

- say, by typing HTTP URL into address bar

(old site)

View source:

<form method="post"

*action="**https**://onlineservices.wachovia.com/..."*

HTTPS and login pages: guidelines

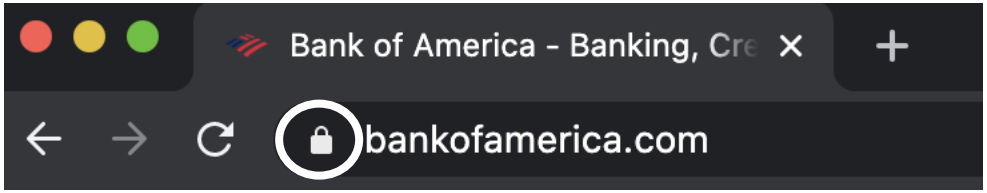
General guideline:

Response to http://login.site.com

should be Location: https://login.site.com

(redirect)

Should be the response to every HTTP request …



Problems with HTTPS and the Lock Icon

LiU EXPANDING REALITY

Problems with HTTPS and the Lock Icon

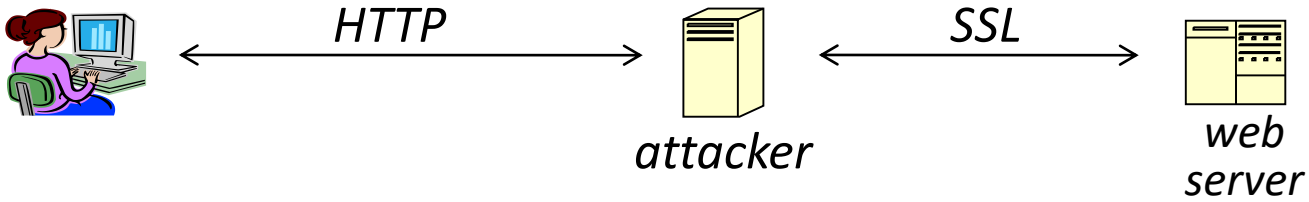
1. Upgrade from HTTP to HTTPS
2. Forged certs
3. Mixed content: HTTP and HTTPS on the same page
4. Does HTTPS hide web traffic?
    - Problems: traffic analysis, compression attacks

1. HTTP ⇒ HTTPS upgrade

Common use pattern:

- browse site over HTTP; move to HTTPS for checkout
- connect to bank over HTTP; move to HTTPS for login

SSL_strip attack: prevent the upgrade [Moxie'08]



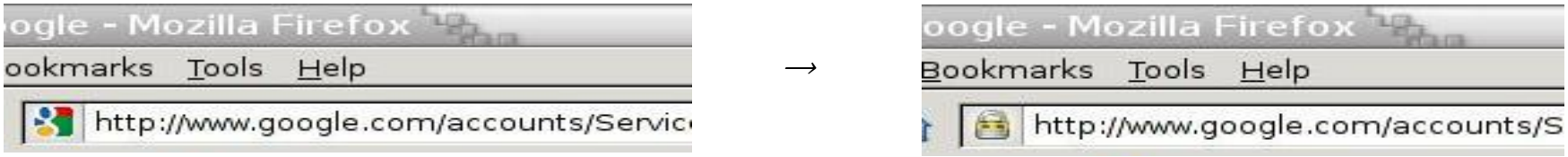
<a href=https://…> ⟶ <a href=http://…>

Location: https://... ⟶ Location: http://... (redirect)

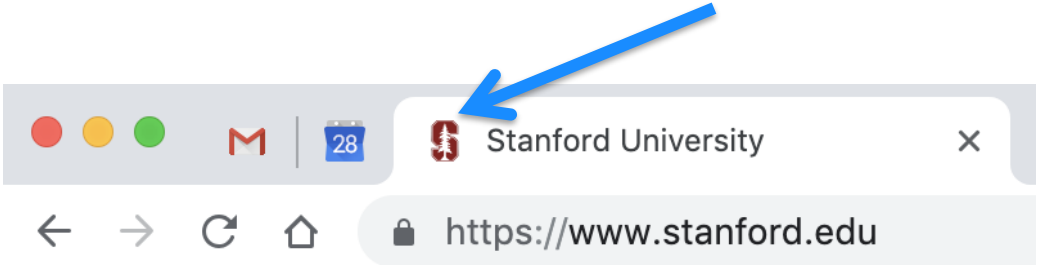
<form action=https://… > ⟶ <form action=http://…>

Tricks and Details

Tricks: drop-in a clever fav icon (older browsers)

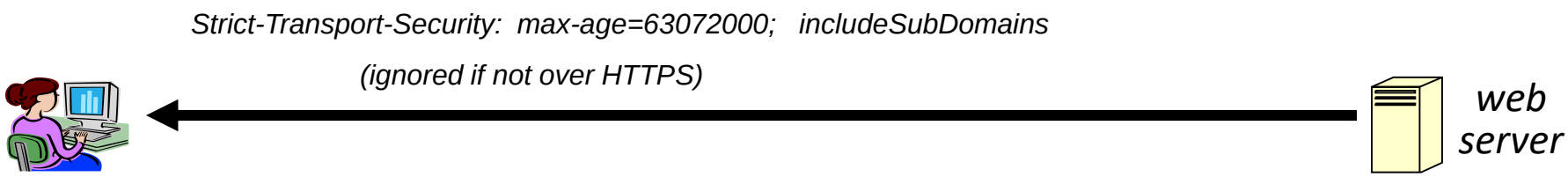


⇒ fav icon no longer presented in address bar



Number of users who detected HTTP downgrade: 0

Defense: Strict Transport Security (HSTS)



Header tells browser to always connect over HTTPS

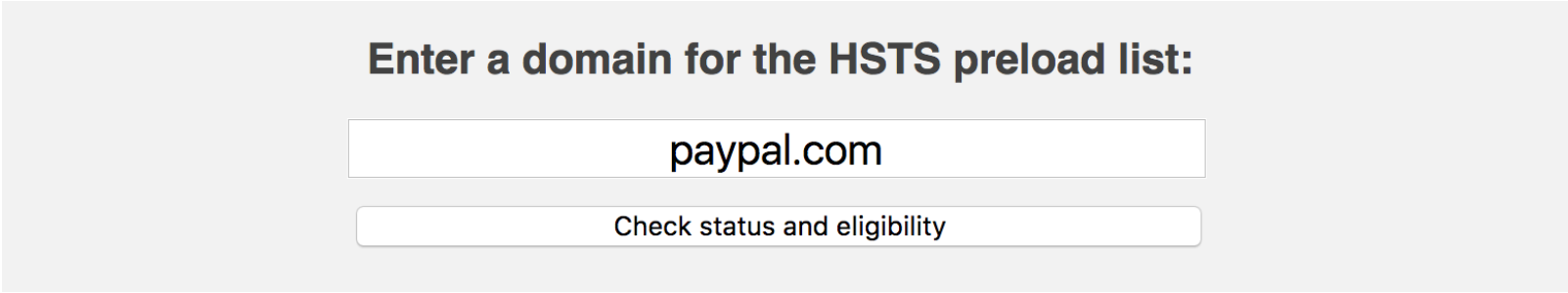
Subsequent visits must be over HTTPS (self signed certs result in an error)

- Browser refuses to connect over HTTP or if site presents an invalid cert
- Requires that entire site be served over valid HTTPS

HSTS flag deleted when user “clears private data” : security vs. privacy

Preloaded HSTS list

https://hstspreload.org/



Strict-Transport-Security: max-age=63072000; includeSubDomains; ***preload***

Preload list hard-coded in Chrome source code. Examples:
Google, Paypal, Twitter, Simple, Linode, Stripe, Lastpass, …

CSP: upgrade-insecure-requests

The problem: many pages use **<img src="http://site.com/img">**

- Makes it difficult to migrate a section of a site to HTTPS

Solution: gradual transition using CSP

Content-Security-Policy: upgrade-insecure-requests

```
<img src="http://site.com/img">
<img src="http://othersite.com/img">
<a href="http://site.com/img">
<a href="http://othersite.com/img">
```

```
<img src="https://site.com/img">
<img src="https://othersite.com/img">
<a href="https://site.com/img">
<a href="http://othersite.com/img">
```

2. Certificates: wrong issuance

2011: **Comodo** and **DigiNotar** CAs hacked, issue certs for Gmail, Yahoo! Mail, …

2013: **TurkTrust** issued cert. for gmail.com (discovered by pinning)

2014: **Indian NIC** (intermediate CA trusted by the root CA **IndiaCCA**) issue certs for Google and Yahoo! domains

Result: (1) India CCA revoked NIC's intermediate certificate

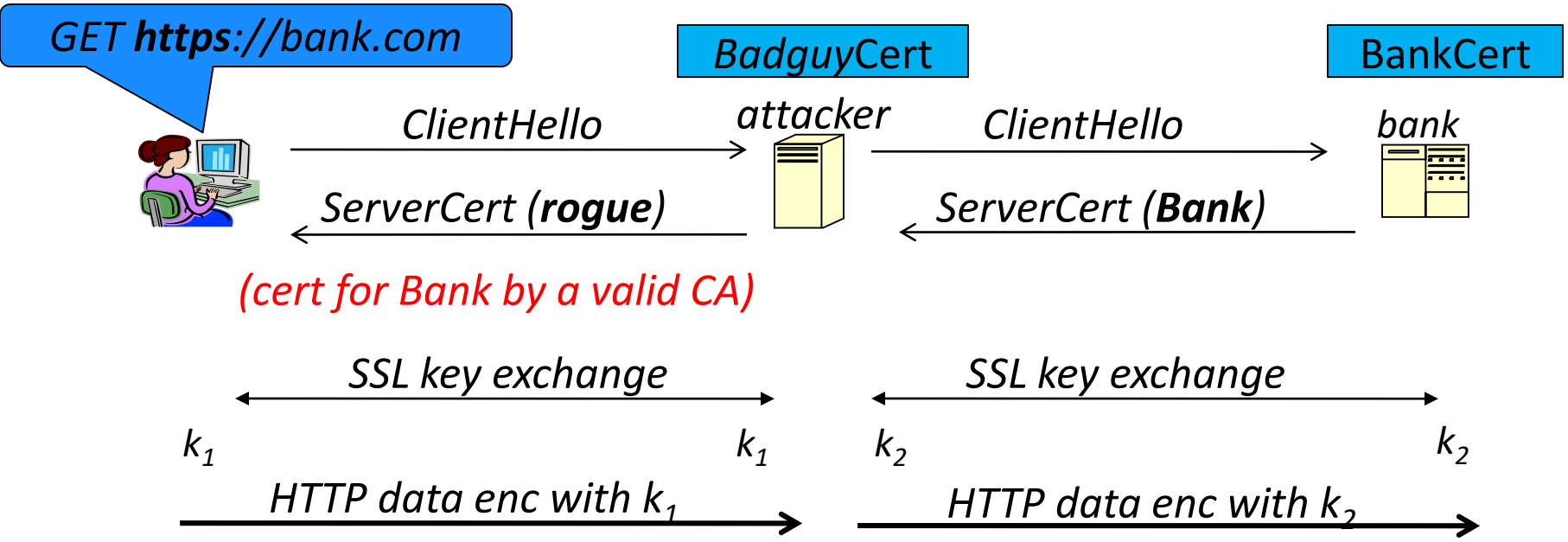
(2) Chrome restricts India CCA root to only seven Indian domains

2016: **WoSign** (Chinese CA) issues cert for GitHub domain (among other issues)

Result: WoSign certs no longer trusted by Chrome and Firefox

⇒ enables eavesdropping w/o a warning on user's session

Man in the middle attack using rogue cert



Attacker proxies data between user and bank.
Sees all traffic and can modify data at will.

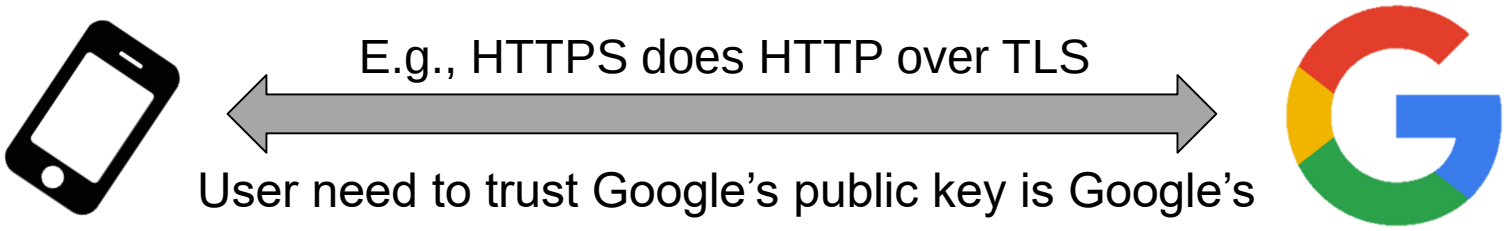
What to do? (many good ideas)

1. **Public-key pinning (static pins)**
   - Hardcode list of allowed CAs for certain sites (Gmail, facebook, …)
   - Browser rejects certs issued by a CA not on list
   - Now deprecated (because often incorrectly used in practice)

2. **Certificate Transparency (CT)**: [LL'12]
   - idea: CA's must advertise a log of all certs. they issued
   - Browser will only use a cert if it is published on (two) log servers
     - Server attaches a signed statement from log (SCT) to certificate
   - Companies can scan logs to look for invalid issuance

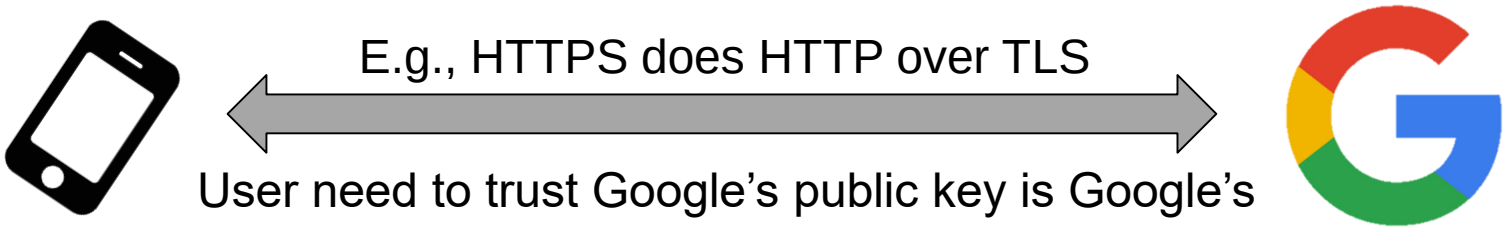
Motivation and high-level problem

- Private and confidential communication important
  - Billions of devices
  - Millions of services
- Certification Authorities (CAs) issue certificates
  - Proof of identity (signed with their private key)



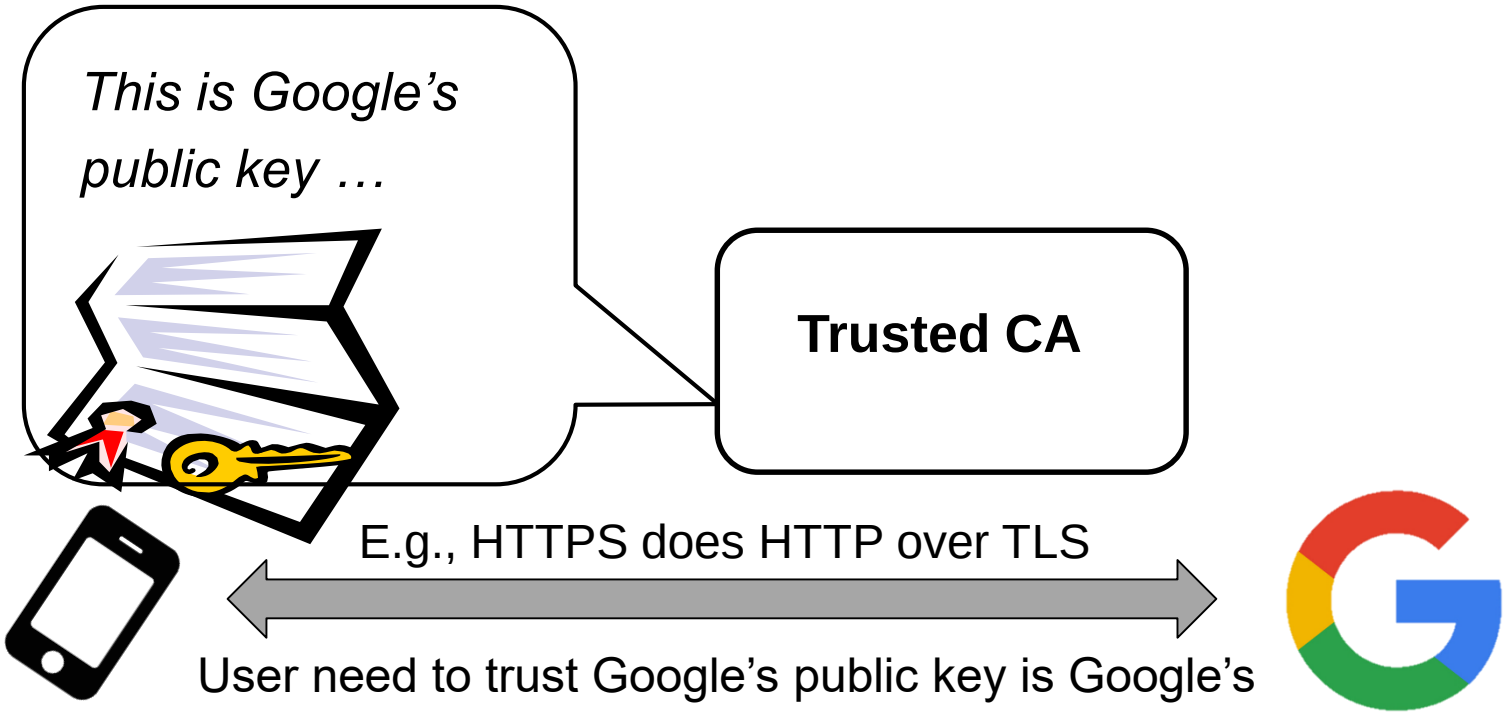
Motivation and high-level problem

- If CAs in our trust (root) store (e.g., Symantec/ Verisign) tells us that a public key belongs to Google, our browsers (and us) trust that this is the case



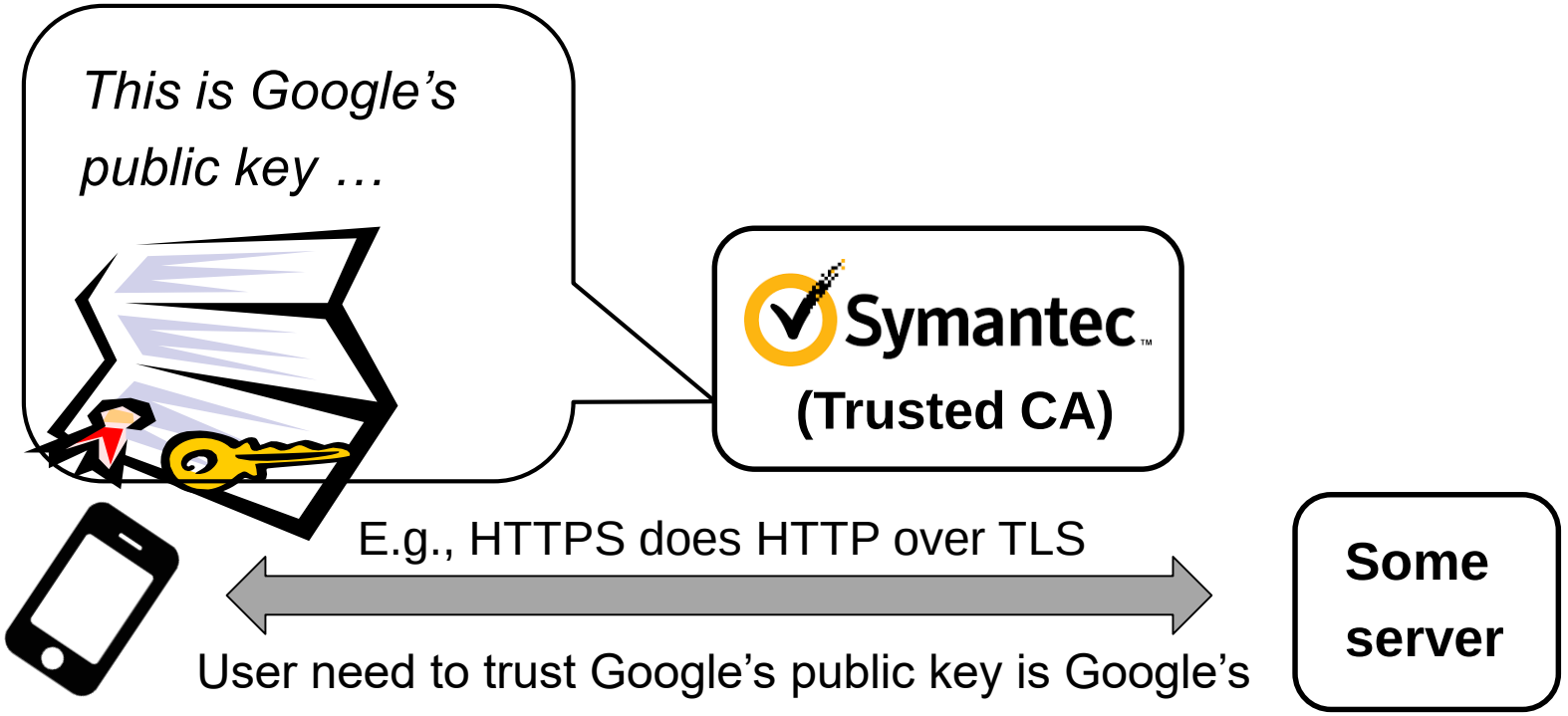
Motivation and high-level problem

- If CAs in our trust (root) store (e.g., Symantec/ Verisign) tells us that a public key belongs to Google, our browsers (and us) trust that this is the case



Motivation and high-level problem

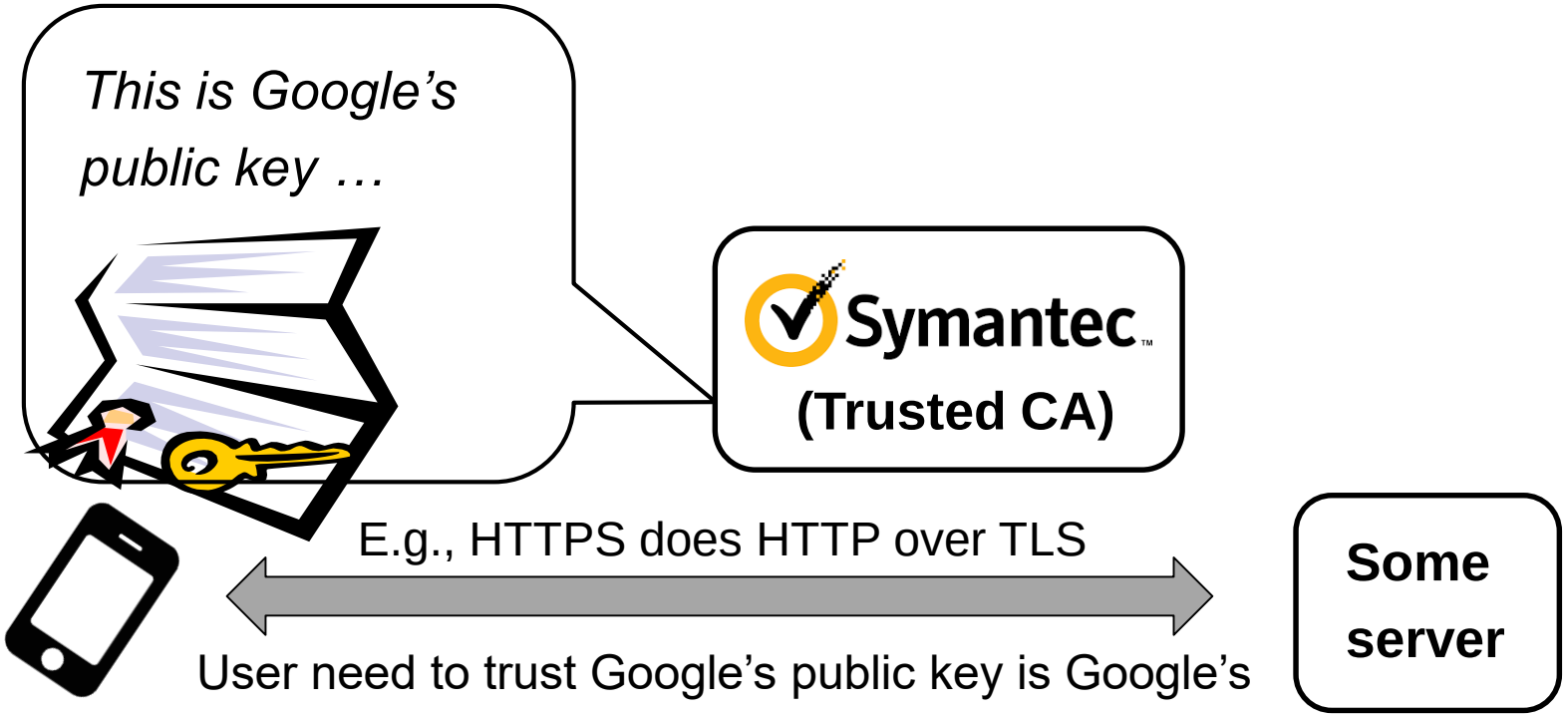
- However, mistakes happen ...
  - E.g., in Oct. 2015, Google discovered (using CT) that Symantec had issued test certificates for 76 domains that they did not own (including Google domains) and another 2,458 unregistered domains …



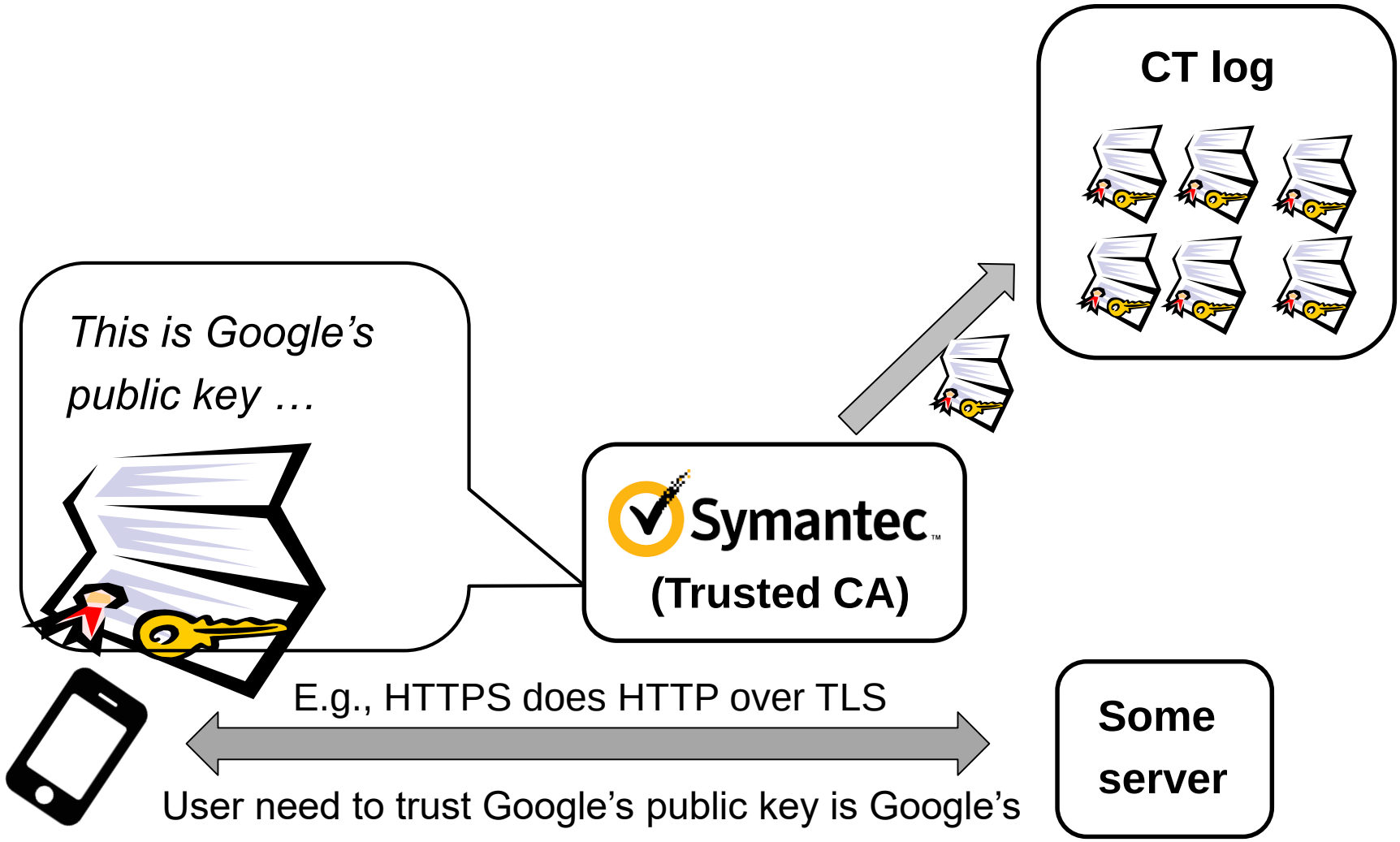
CT: Emerging trust-monitoring solution

- Since then, Google has demanded that Symantec logs all their certificates in public (append-only) CT logs
- Since Jan. 2015, the Chrome browser requires all EV certificates be logged in 1 Google log and 1 other log
  - Mozilla planning to make similar demands
  - Both Chrome and Mozilla expected to implement policies for DV certificates too …

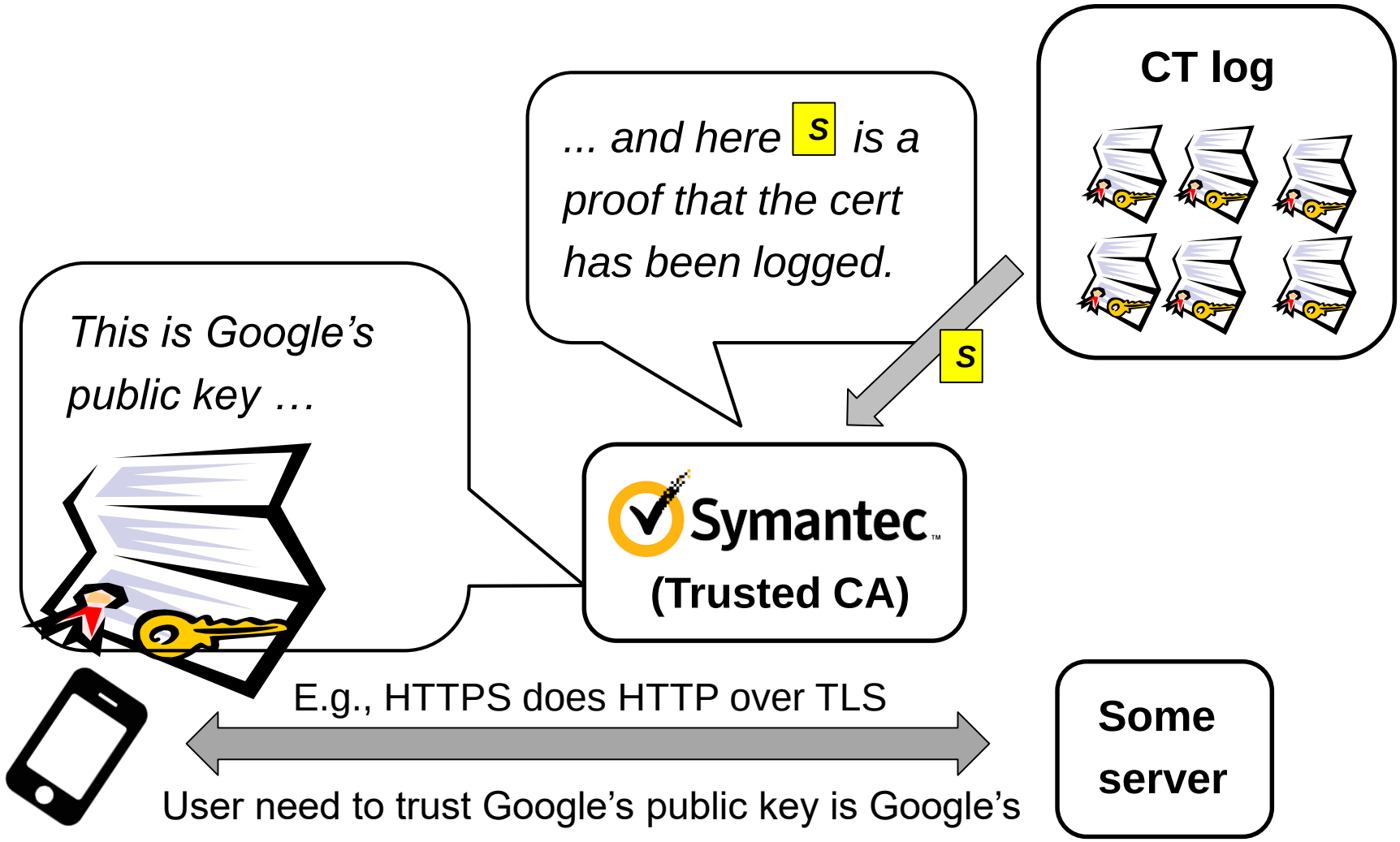
CT: Emerging trust-monitoring solution



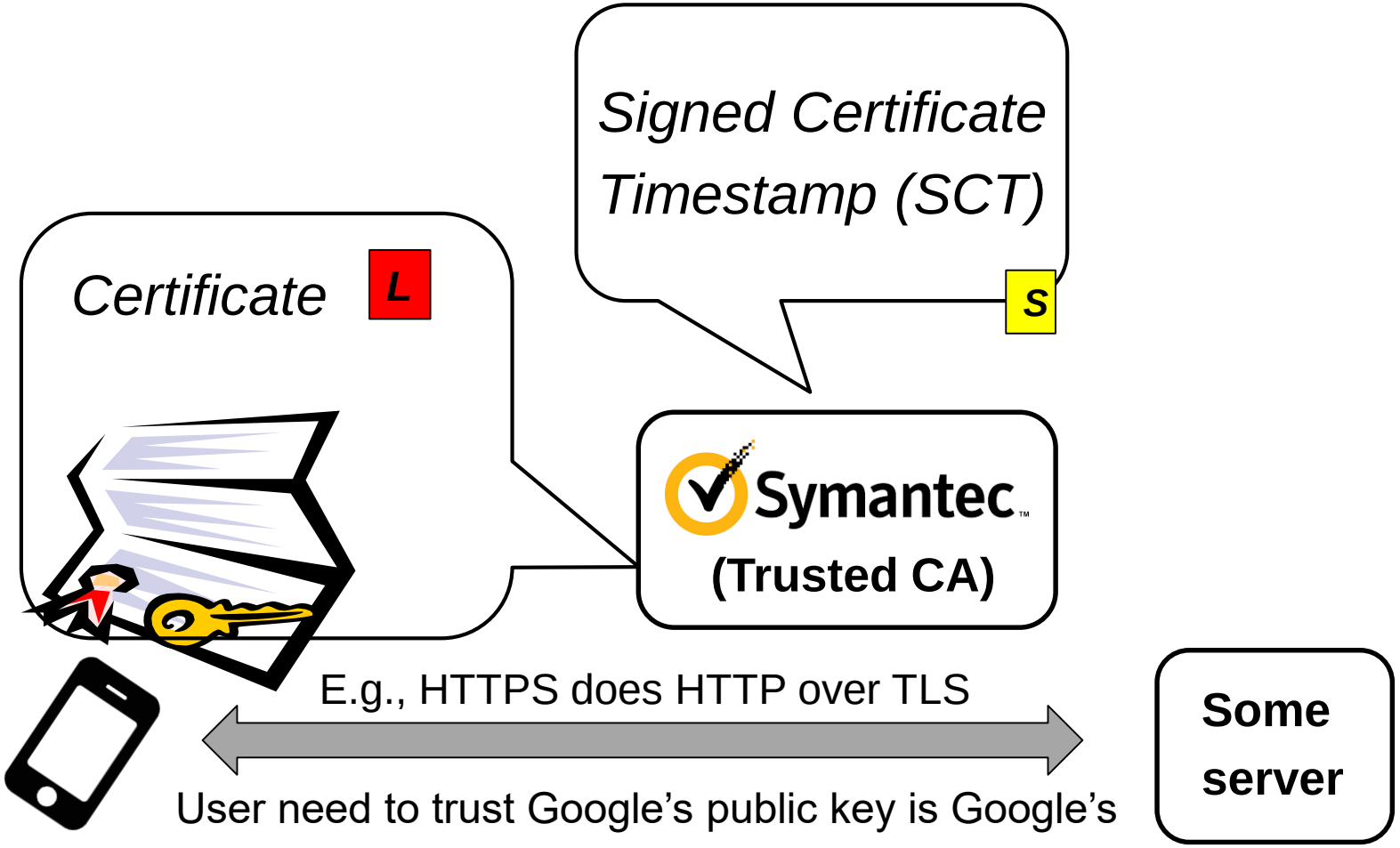
CT: Emerging trust-monitoring solution



CT: Emerging trust-monitoring solution

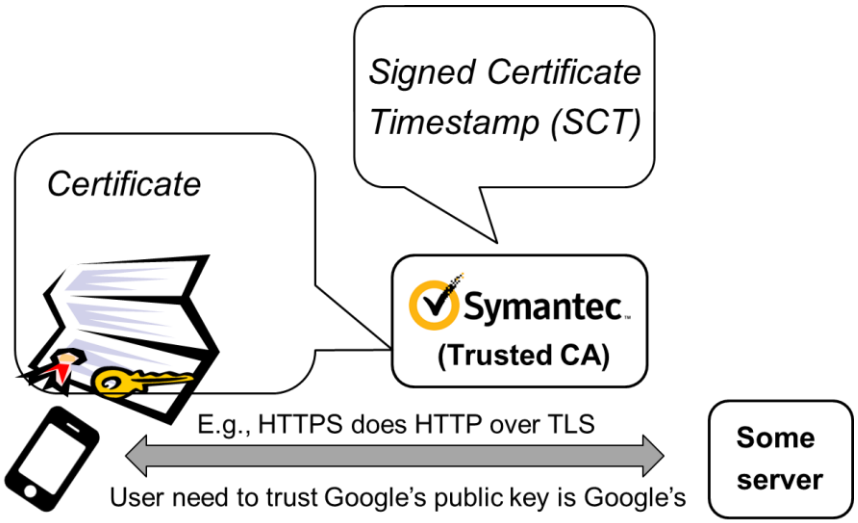


CT: Emerging trust-monitoring solution

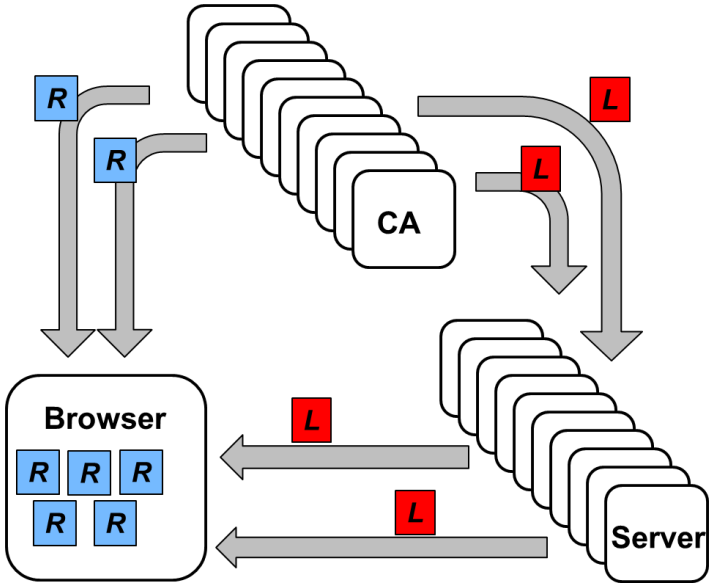


Signed Certificate Timestamps (SCTs)

- SCTs delivered three different ways
  - X.509v3 extension
  - TLS extension
  - OSCP stapling

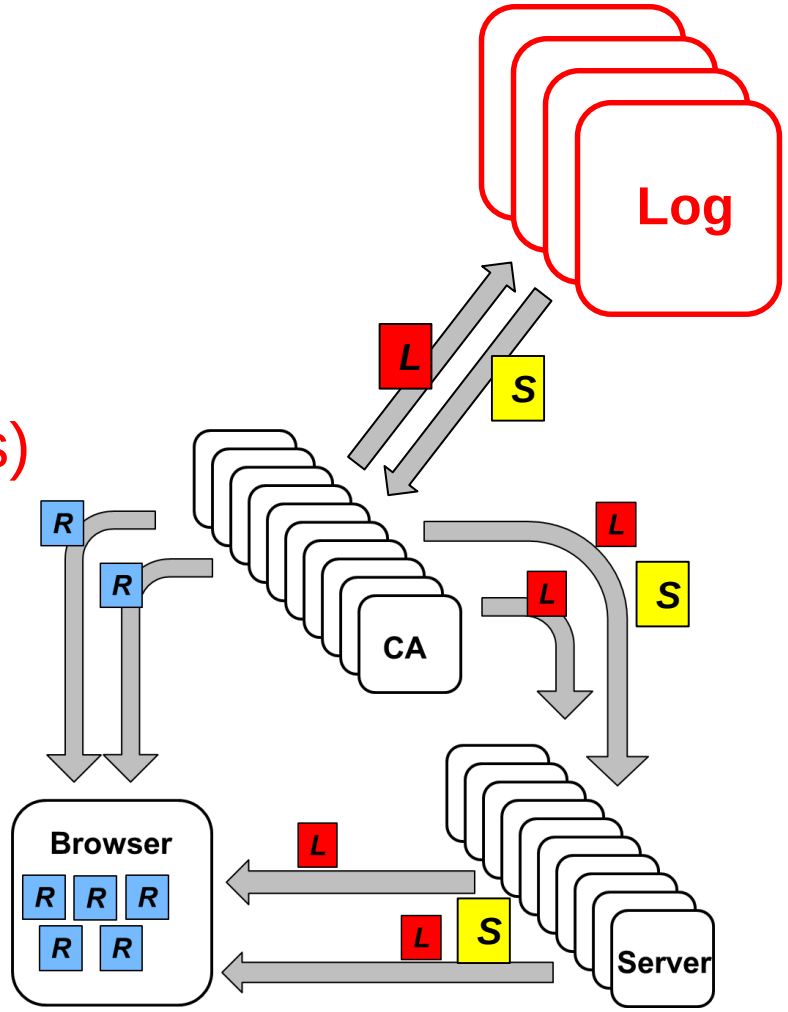


Certification Transparency (CT)



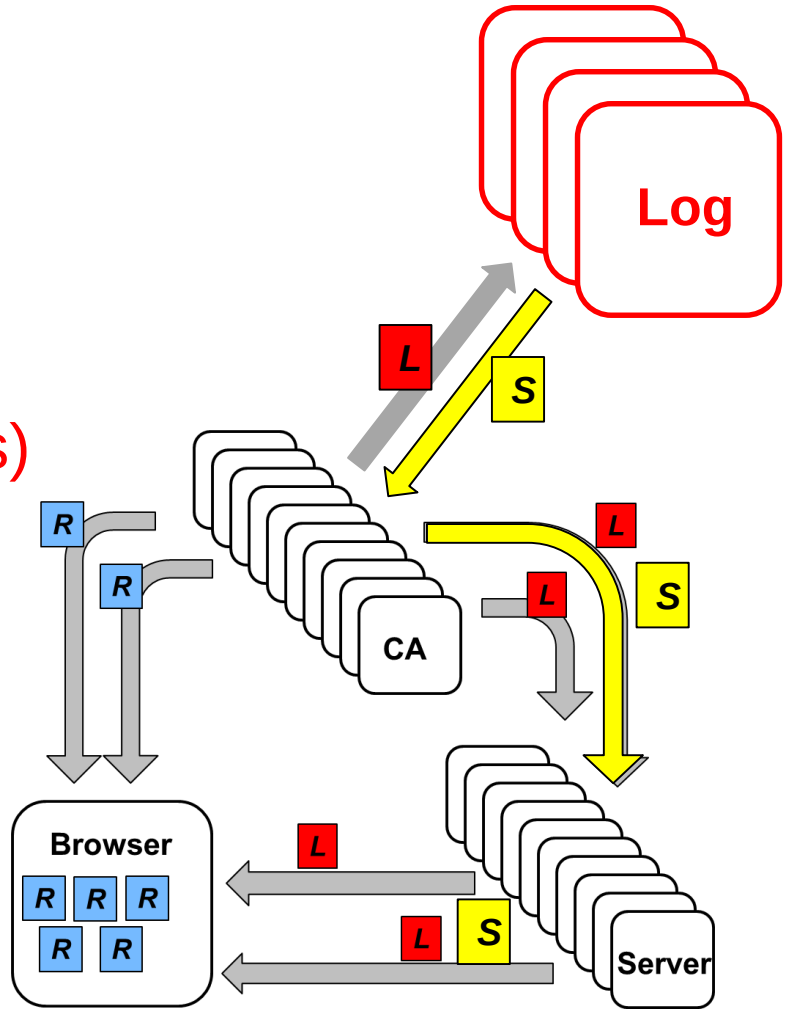
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Create SCTs
-
  -



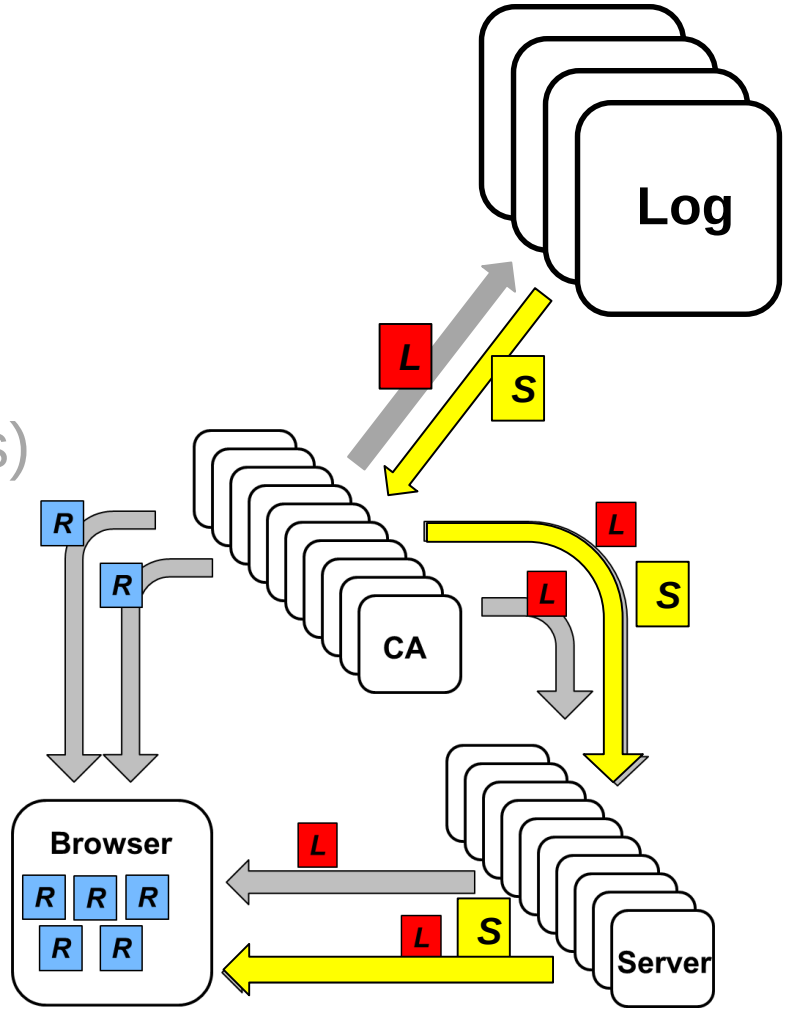
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Create SCTs
-
  -



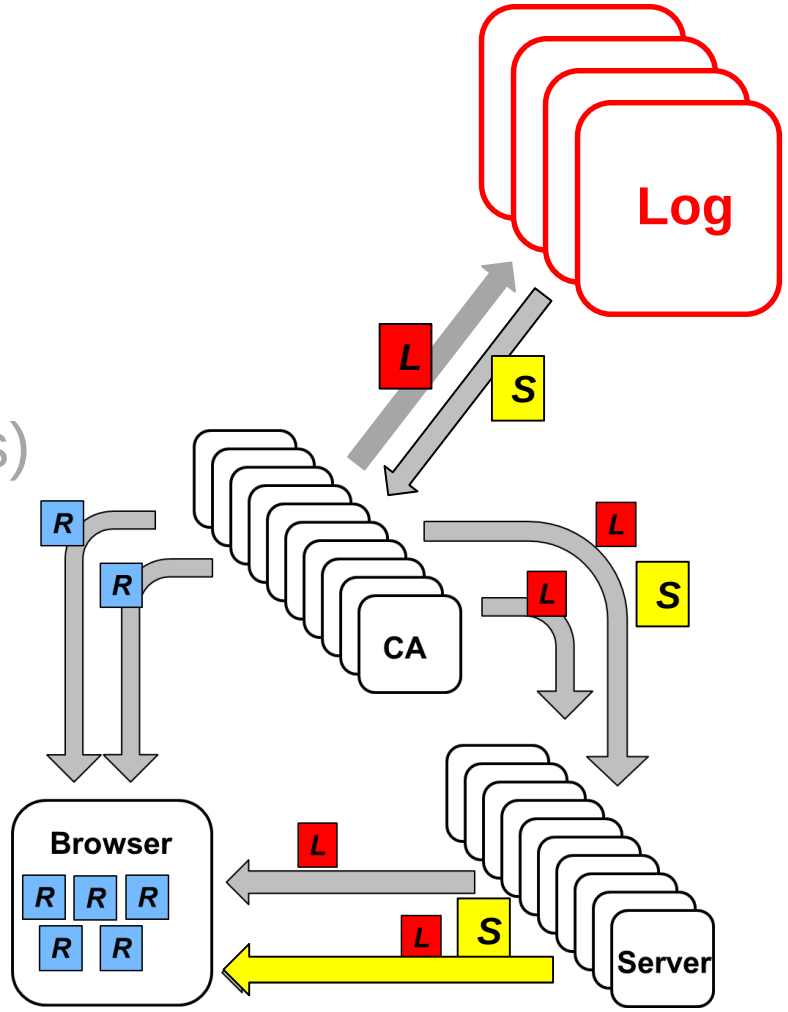
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Create SCTs
- SCTs
  - Proof cert is logged

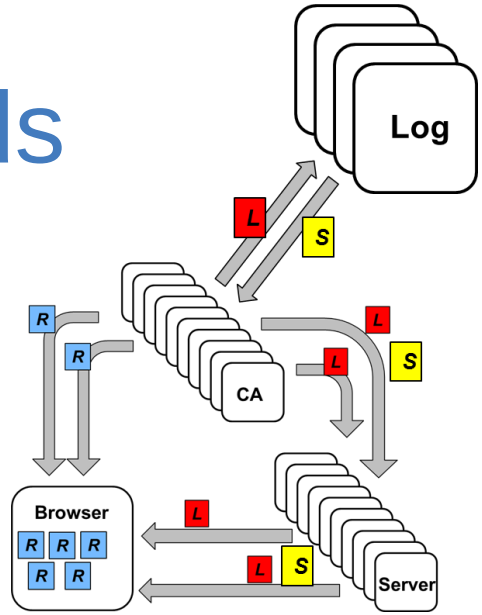


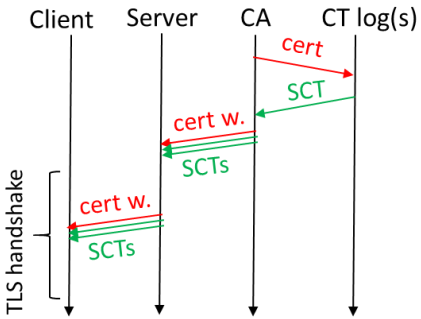
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Create SCTs
- SCTs
  - Proof cert is logged

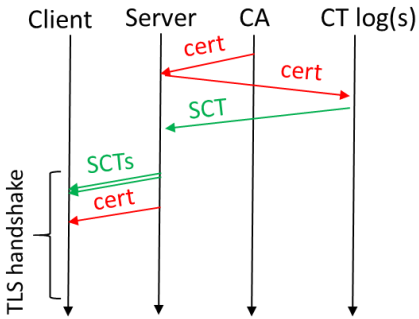


Three SCT delivery methods

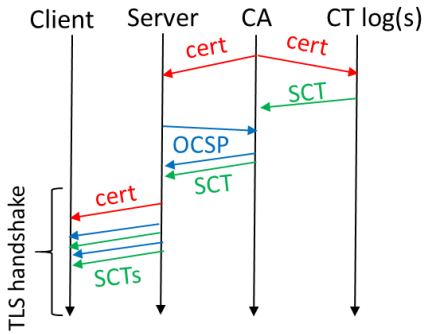




(a) X.509v3 extension

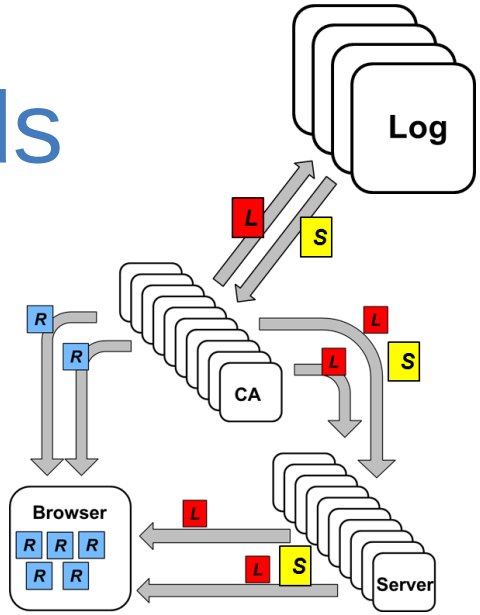


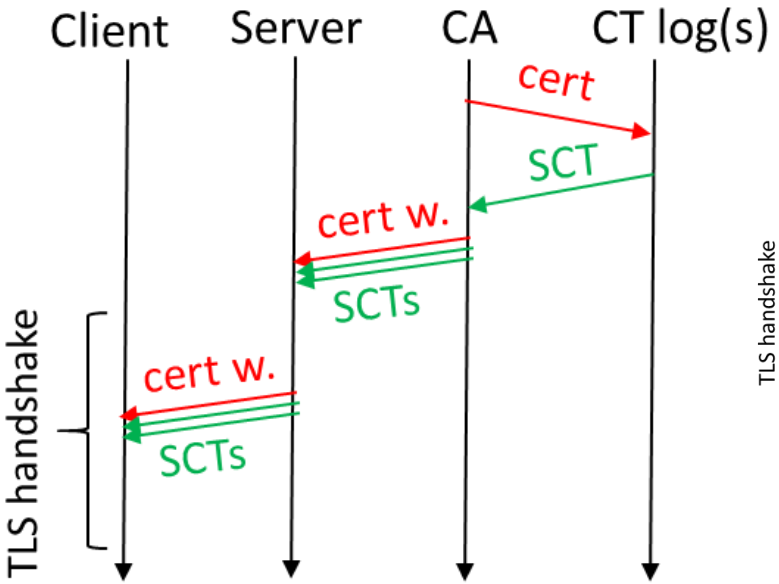
(b) TLS extension



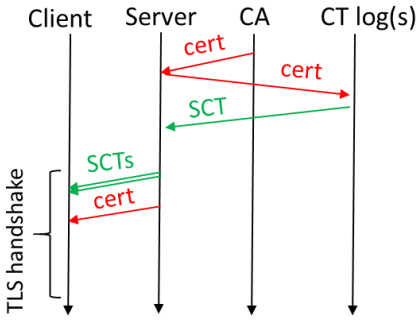
(c) OCSP stapling

Three SCT delivery methods

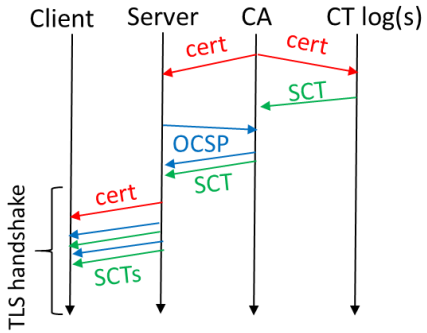




(a) X.509v3 extension

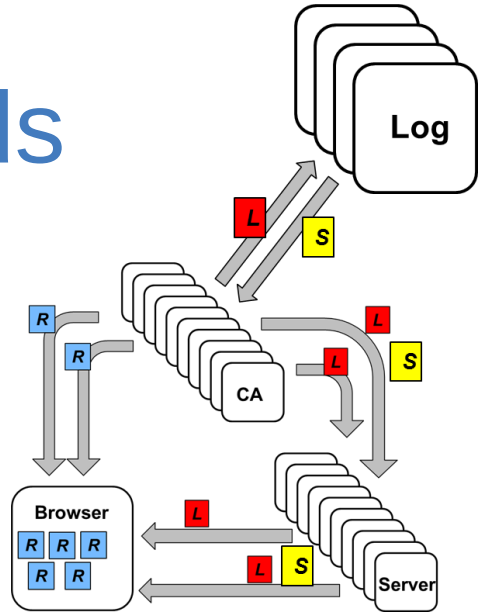


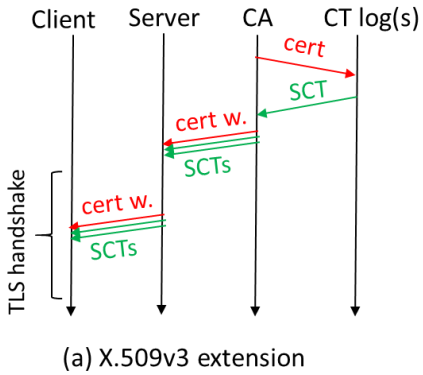
(b) TLS extension



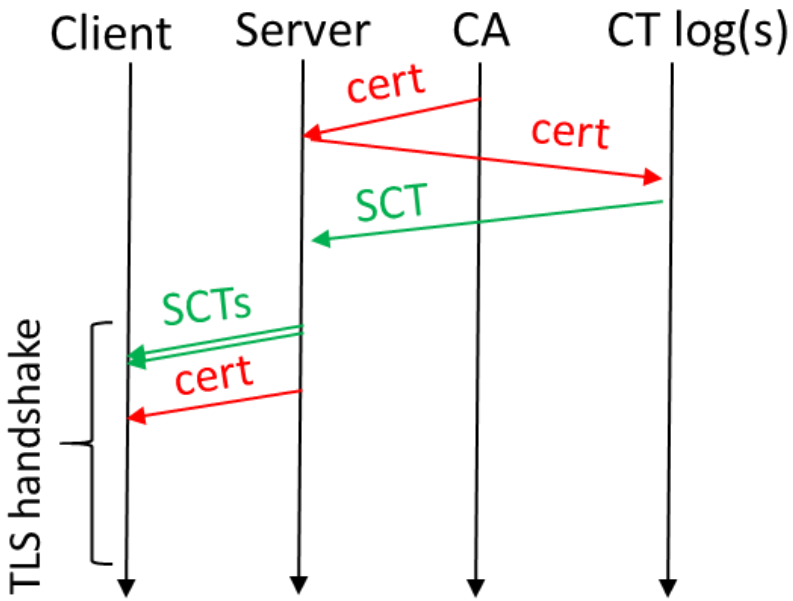
(c) OCSP stapling

Three SCT delivery methods

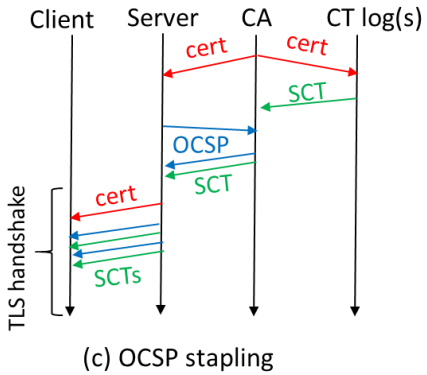




(a) X.509v3 extension

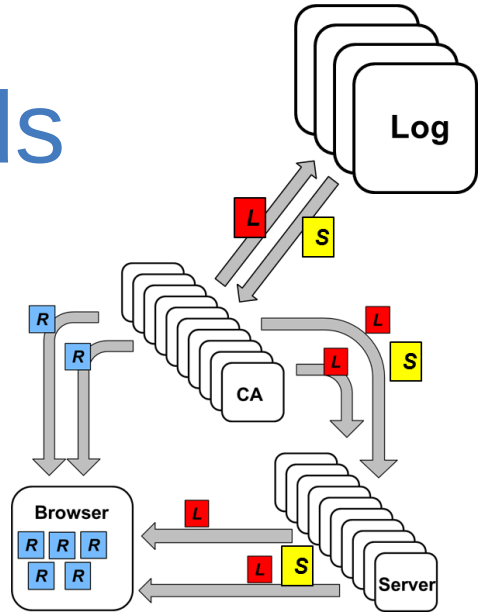


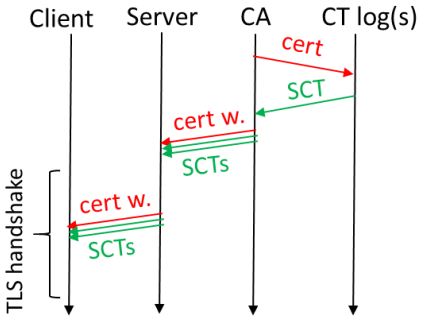
(b) TLS extension



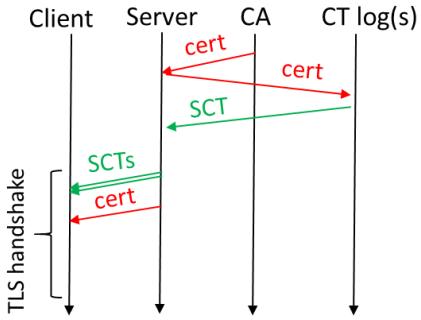
(c) OCSP stapling

Three SCT delivery methods

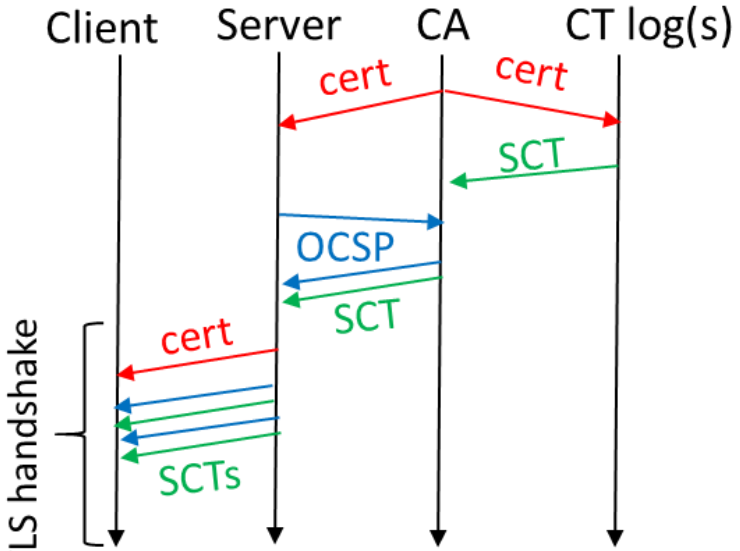




(a) X.509v3 extension



(b) TLS extension



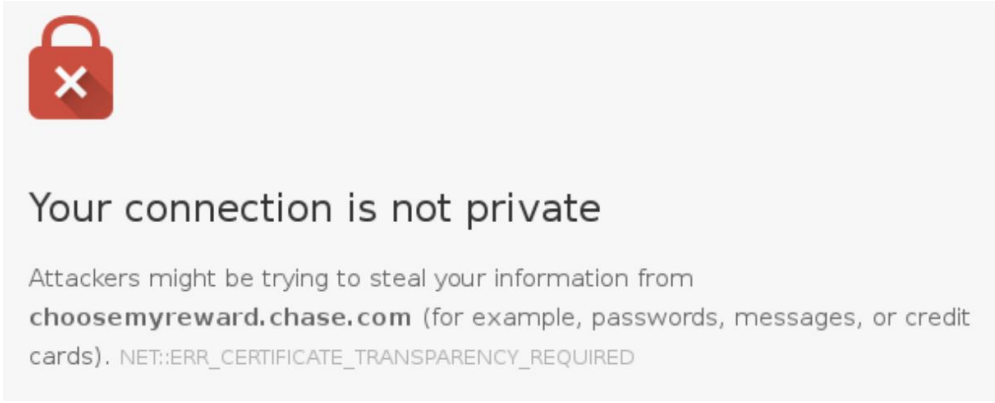
(c) OCSP stapling

CT requirements

April 30, 2018: CT required by chrome

- Required for all certificates with a path to a trusted root CA

  (not required for an installed root CA)
- Otherwise: HTTPS errors



Cert for crypto.stanford.edu published on five logs:

cloudflare_nimbus2018

google_argon2018, google_aviator

google_pilot, google_rocketeer

3. Mixed Content: HTTP and HTTPS

Page loads over HTTPS, but contains content over HTTP

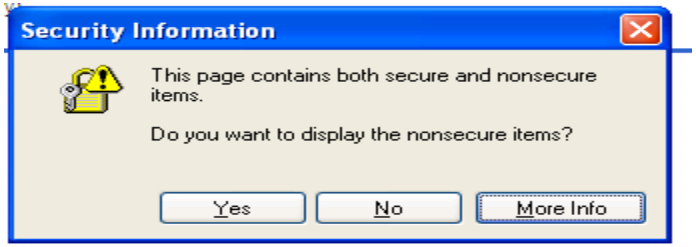
(e.g. <script src="http://.../script.js>)

never write this

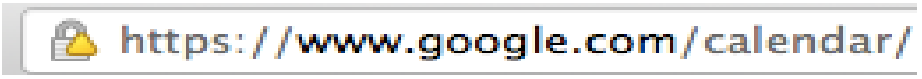
⇒ Active network attacker can hijack session

by modifying script en-route to browser

IE7:



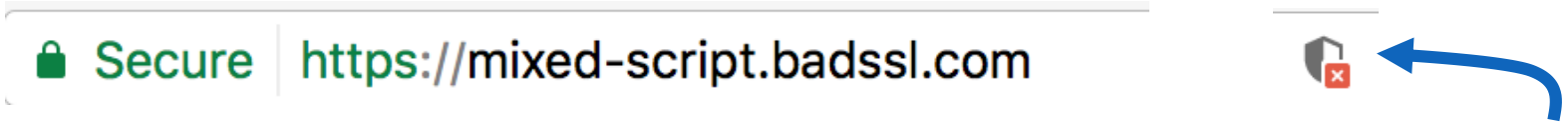
Old Chrome:



Mostly ignored by users …

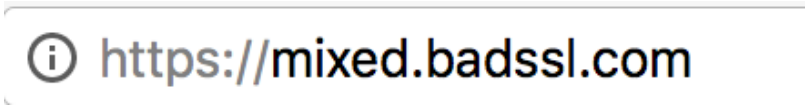
https://badssl.com (Chrome 73, 2019)

Mixed script: **<script src="http://mixed-script.badssl.com/nonsecure.js"></script>**



(script is blocked, click to load)

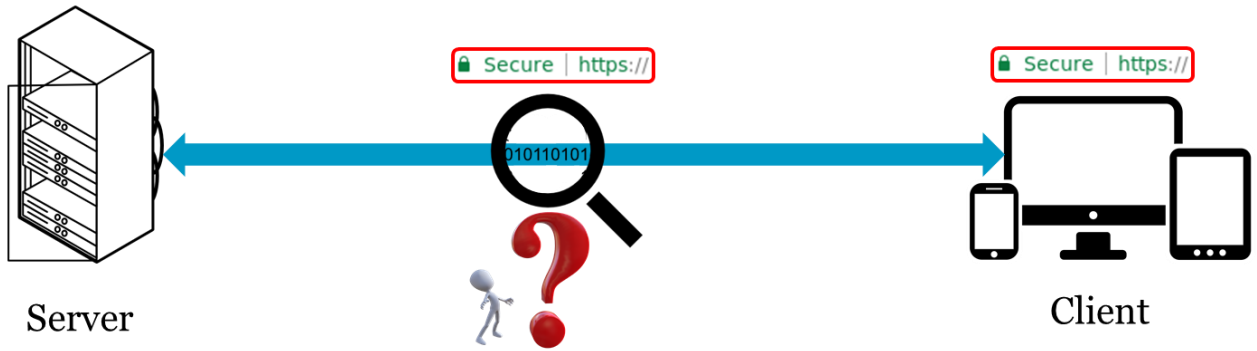
Mixed form: **<form action="http://http.badssl.com/resources/submit.html">**



Form loaded, but no HTTPS indicator

4. Peeking through SSL: traffic analysis

- Network traffic reveals length of HTTPS packets
  - TLS supports up to 256 bytes of padding
- Interactions expose specific internal state of the page and internal state of the client ...



E.g., BUFFEST (Krishnamoorthi et al. 2017)

Credits and some more slides

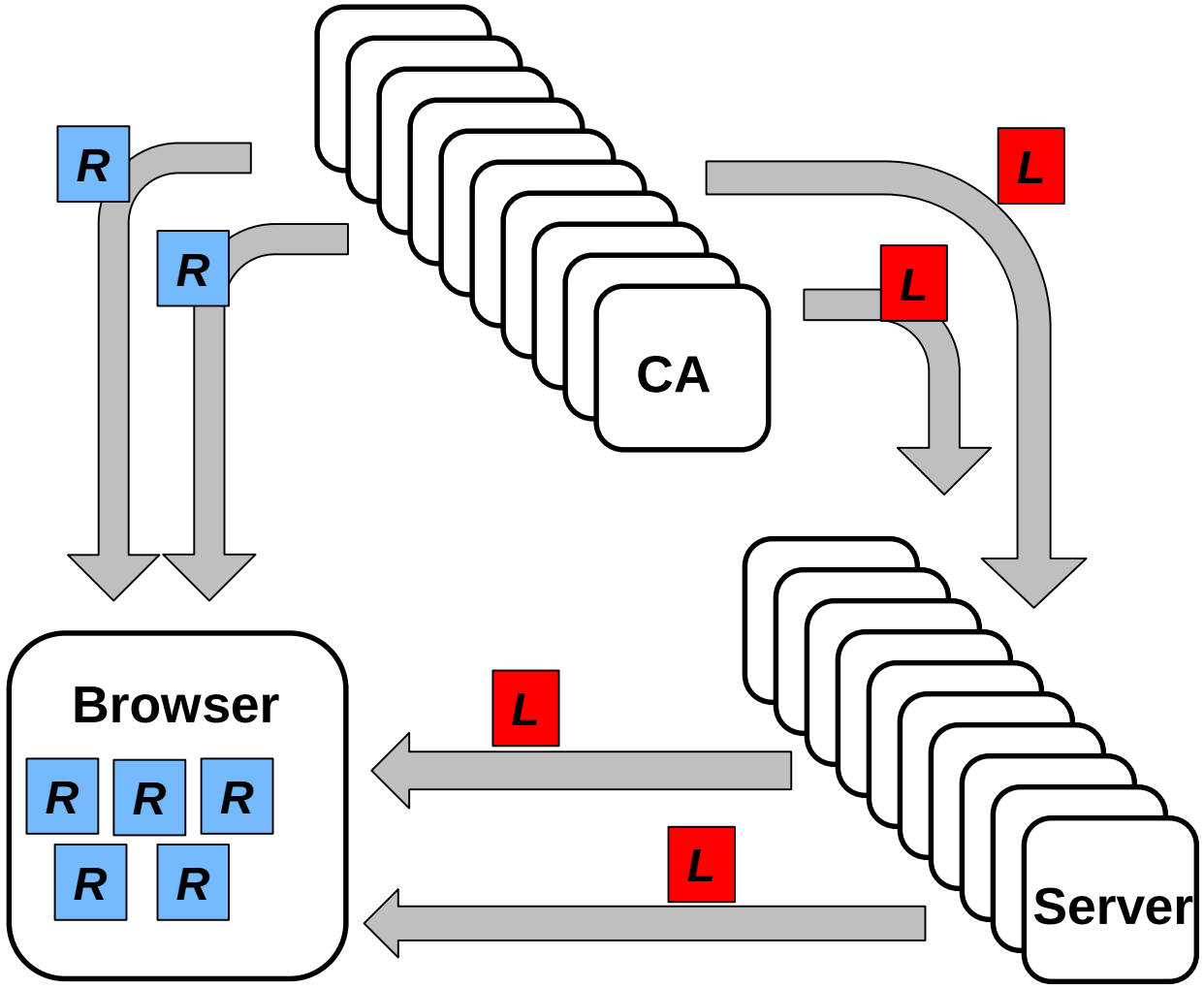
These slides heavily borrow from slides by Dan Boneh and research-presentation slides of some of our prior works, including

- *Nikita Korzhitskii and Niklas Carlsson, Characterizing the Root Landscape of Certificate Transparency Logs, Proc. IFIP Networking, Paris, France, June 2020, pp. 190--198.*
- *Carl Nykvist, Linus Sjostrom, Josef Gustafsson, and Niklas Carlsson, Server-side Adoption of Certificate Transparency, Proc. Passive and Active Measurement Conference (PAM), Berlin, Germany, Mar. 2018.*
- *Josef Gustafsson, Gustaf Overier, Martin Arlitt, and Niklas Carlsson, A First Look at the CT Landscape: Certificate Transparency Logs in Practice, Proc. Passive and Active Measurement Conference (PAM), Sydney, Australia, Mar. 2017, pp. 87-99.*
- *Vengatanathan Krishnamoorthi, Niklas Carlsson, Emir Halepovic and Eric Petajan, BUFFEST: Predicting Buffer Conditions and Real-time Requirements of HTTP(S) Adaptive Streaming Clients, Proc. ACM Multimedia Systems (ACM MMSys), Taipei, Taiwan, June 2017, pp. 76--87.*

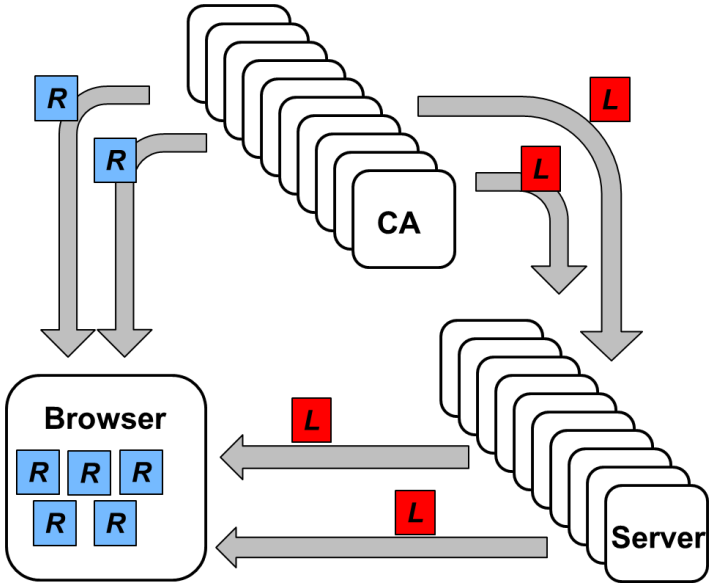
LINKÖPING UNIVERSITY

Niklas Carlsson (niklas.carlsson@liu.se)
www.ida.liu.se/~nikca89/

Certification Transparency (CT)

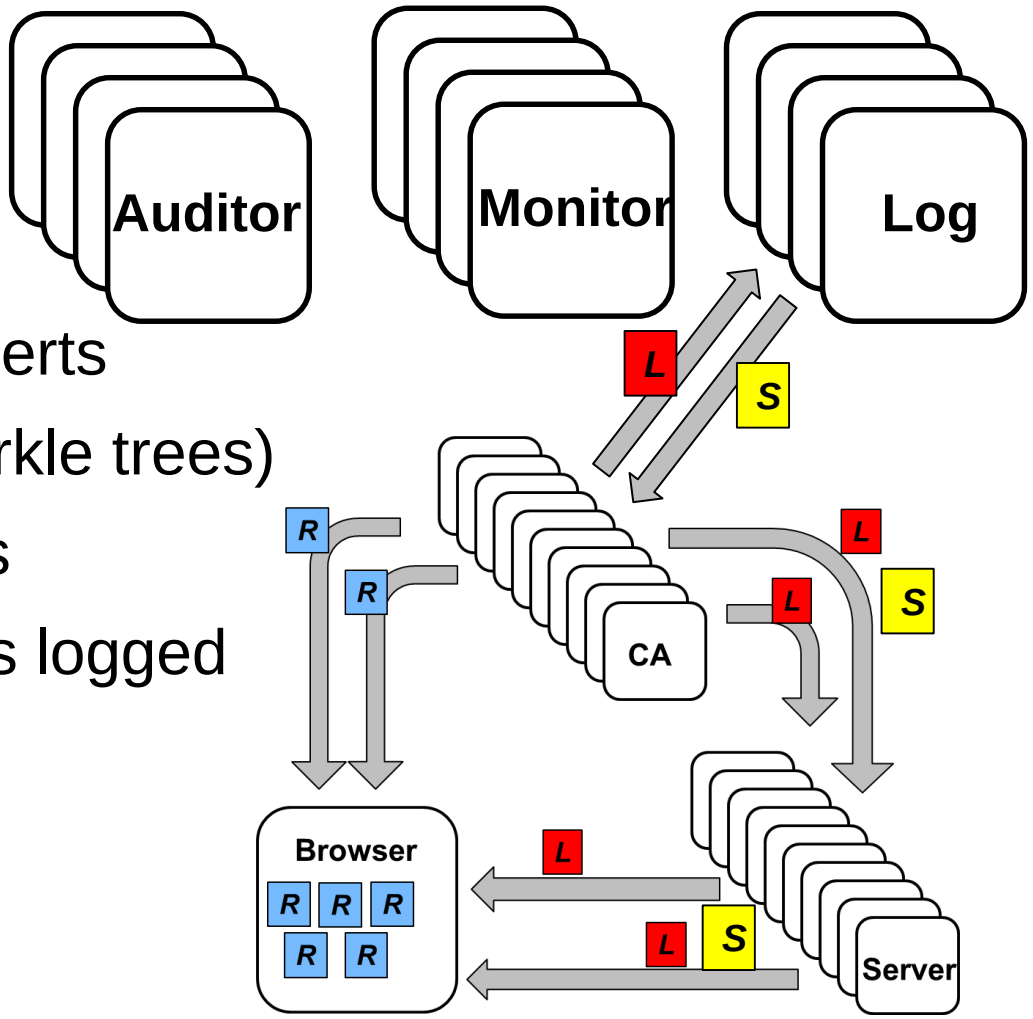


Certification Transparency (CT)



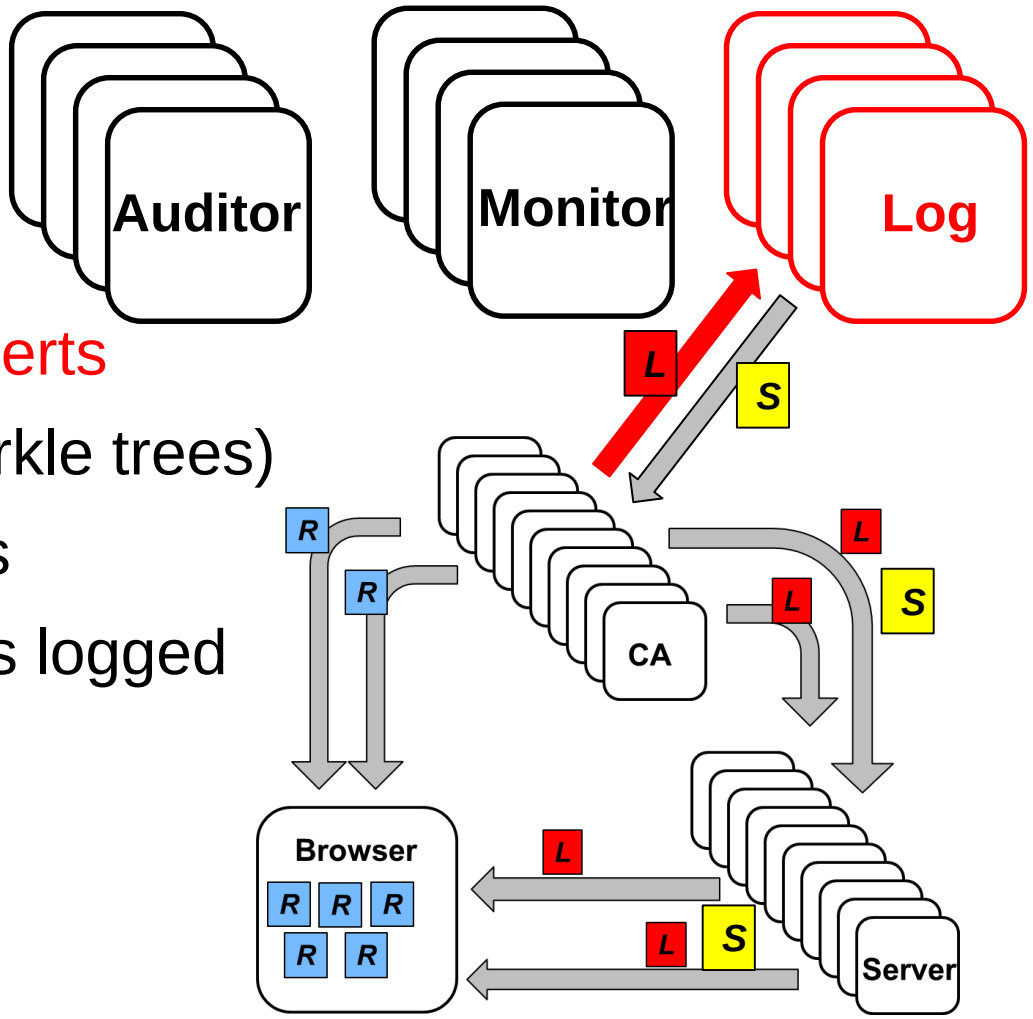
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior



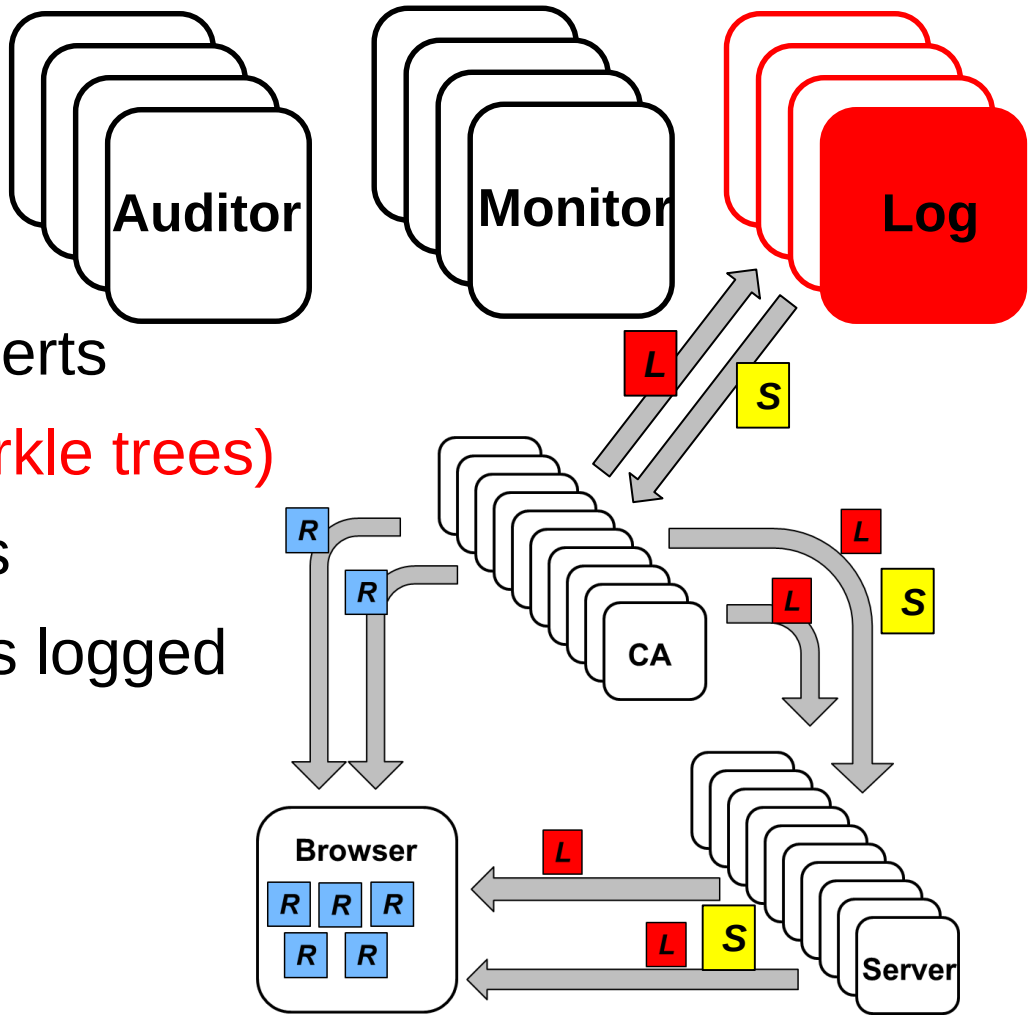
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior



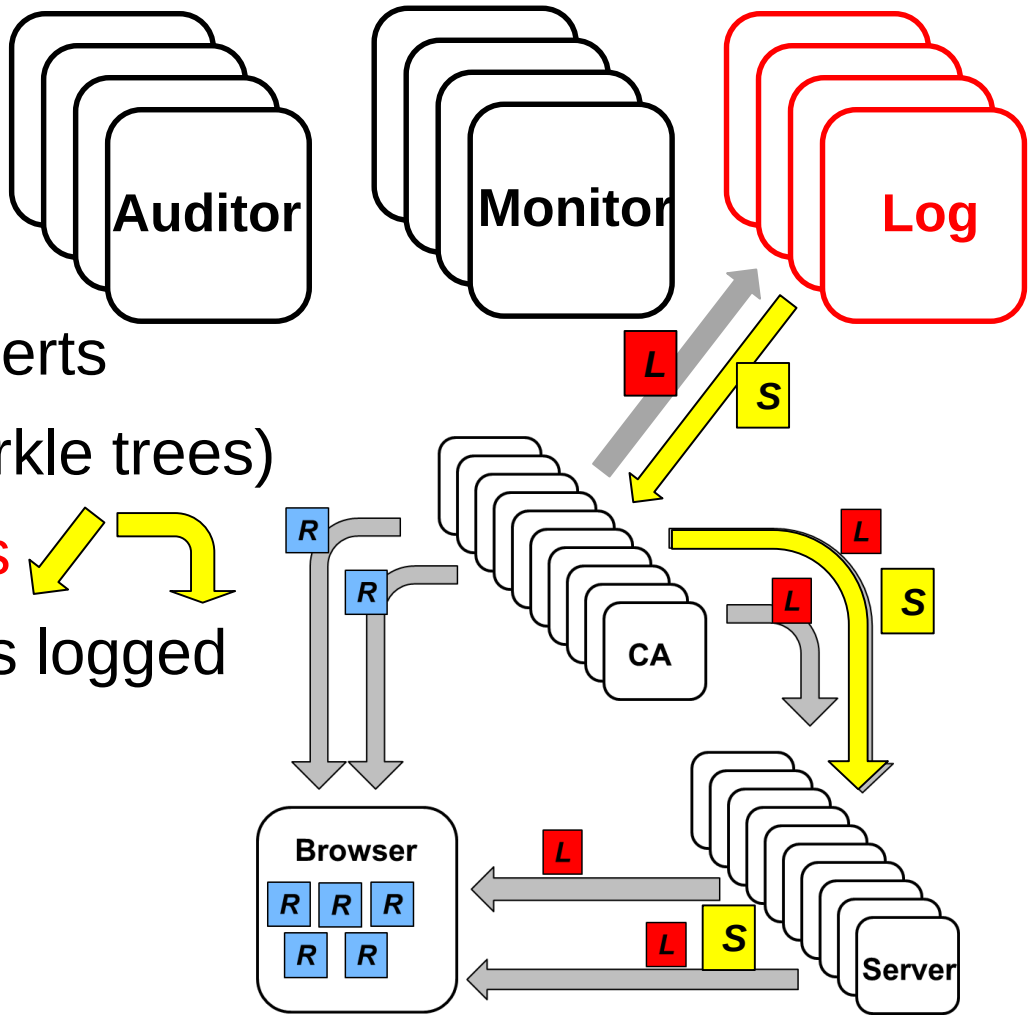
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior



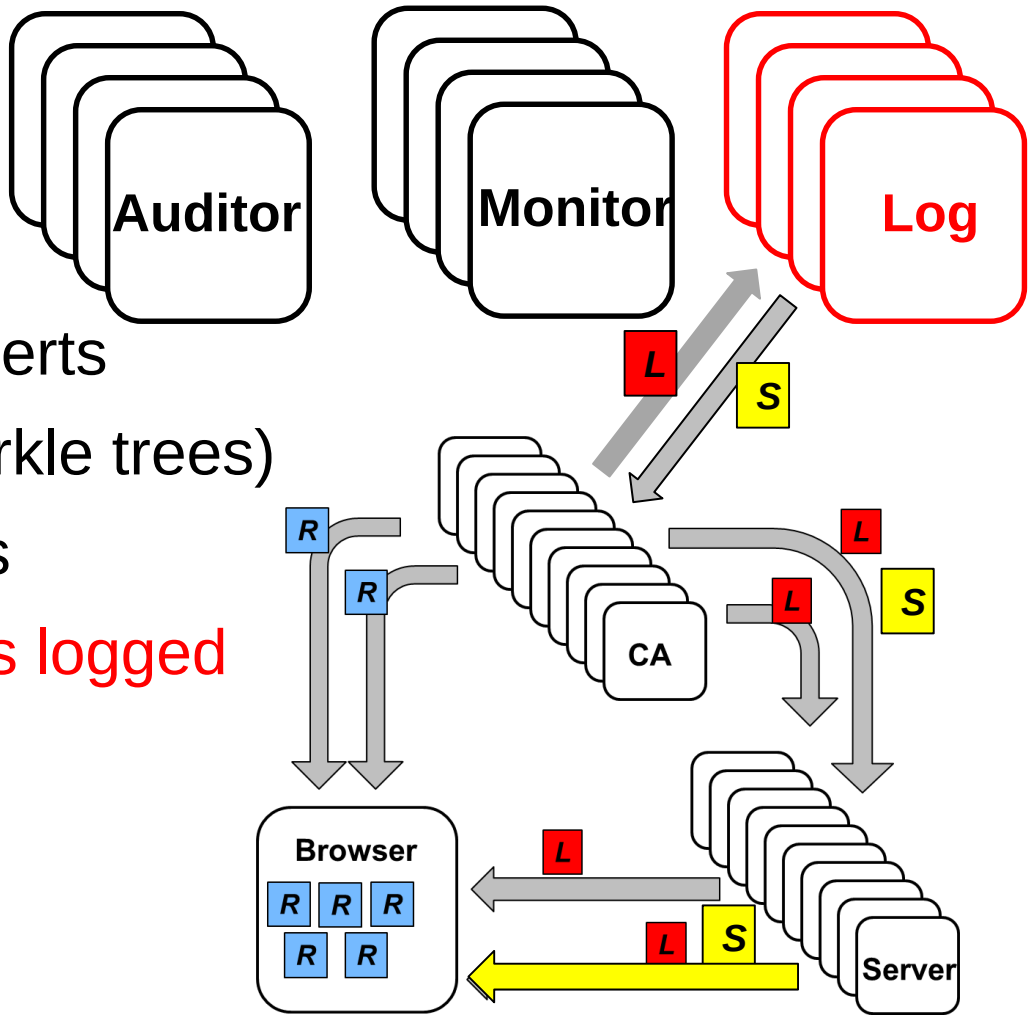
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior



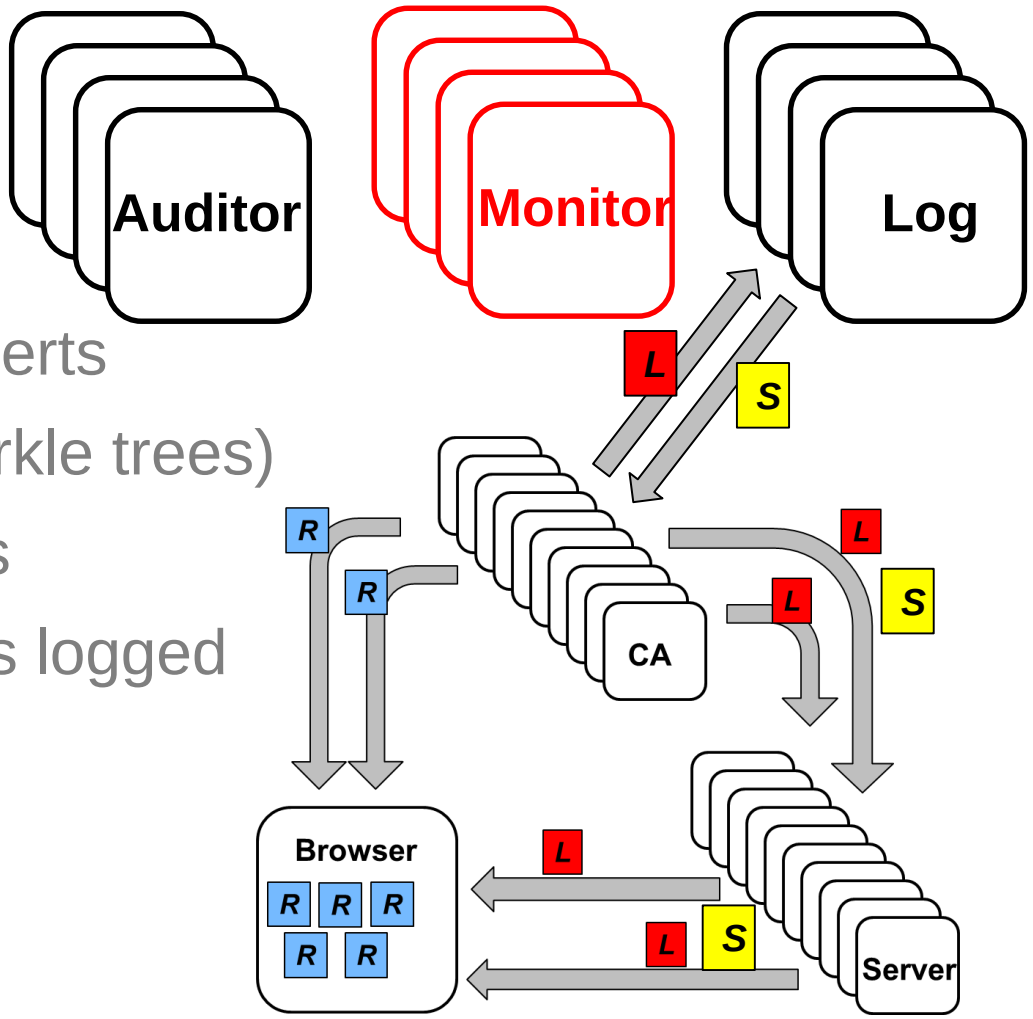
Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior



Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior



Certification Transparency (CT)

- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Servers get SCTs
  - SCTs proof cert is logged
- Monitors
  - Assert log content
- Auditors
  - Assert log behavior