

# *HTTPS, TLS, and Certificates*

**Niklas Carlsson**, *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



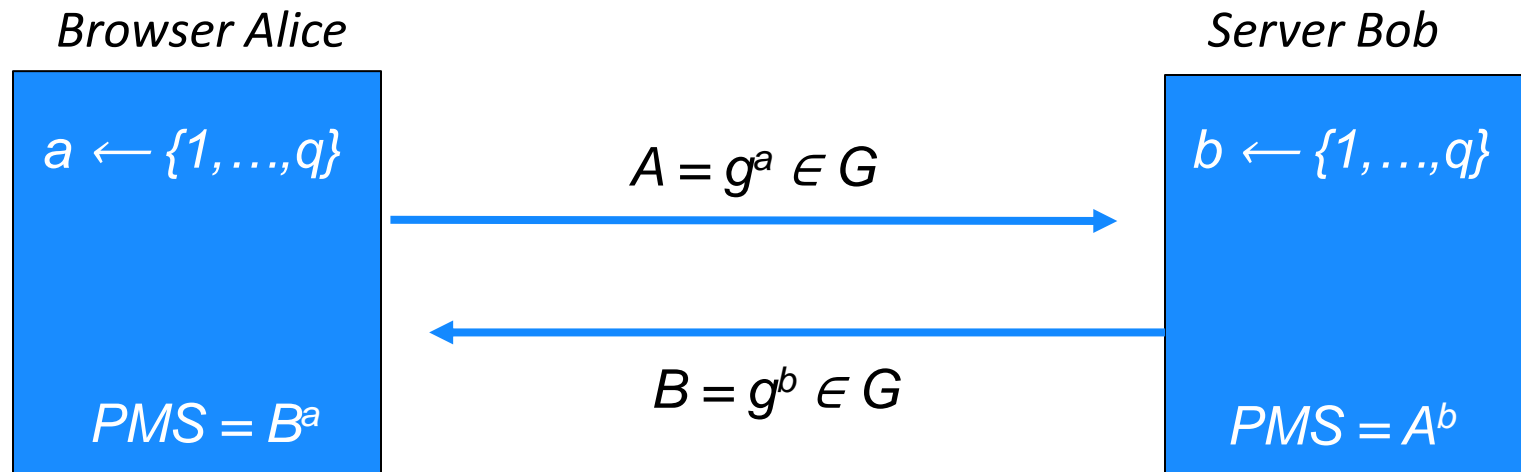
## Examples:

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

# TLS overview: (1) DH key exchange

**Anonymous key exchange secure against eavesdropping:**

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

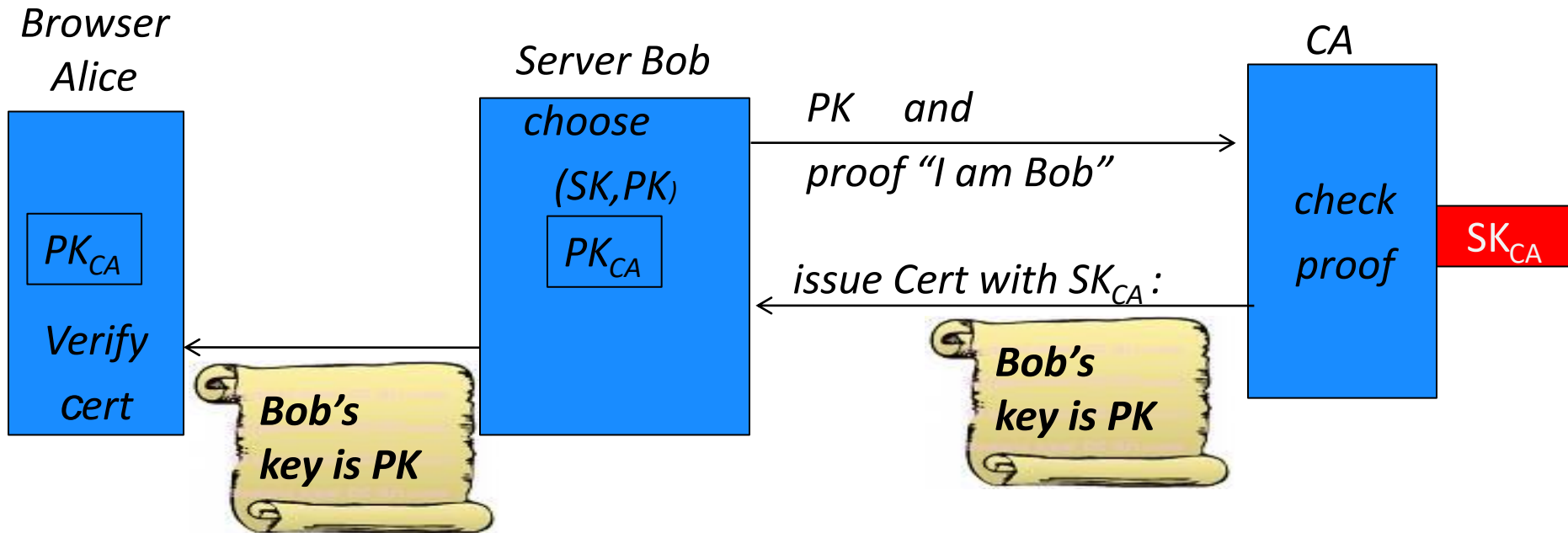


$$PreMasterSecret = g^{ab} = (g^b)^a = B^a = (g^a)^b = A^b$$

Used to establish a shared secret  
Group  $G$  is publicly known

# TLS overview: (2) Certificates

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



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

# Motivation and high-level problem

- Private and confidential communication important
  - 
  -



E.g., HTTPS does HTTP over TLS

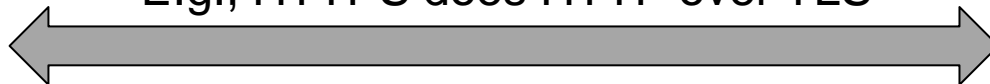


# Motivation and high-level problem

- Private and confidential communication important
  - 
  -



E.g., HTTPS does HTTP over TLS



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



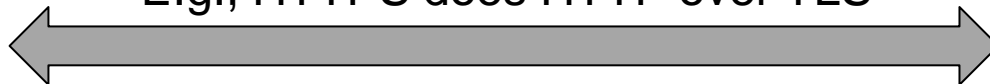


# Motivation and high-level problem

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



E.g., HTTPS does HTTP over TLS



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

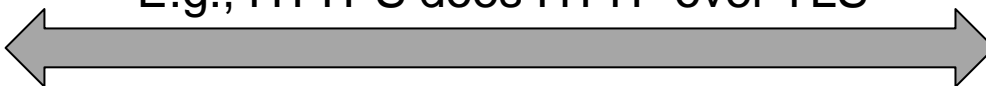


# Motivation and high-level problem

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



User need to trust FB's public key is FBs



E.g., HTTPS does HTTP over TLS

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

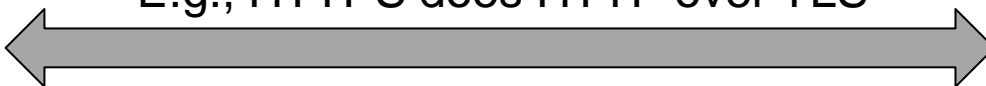


# Motivation and high-level problem

- 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

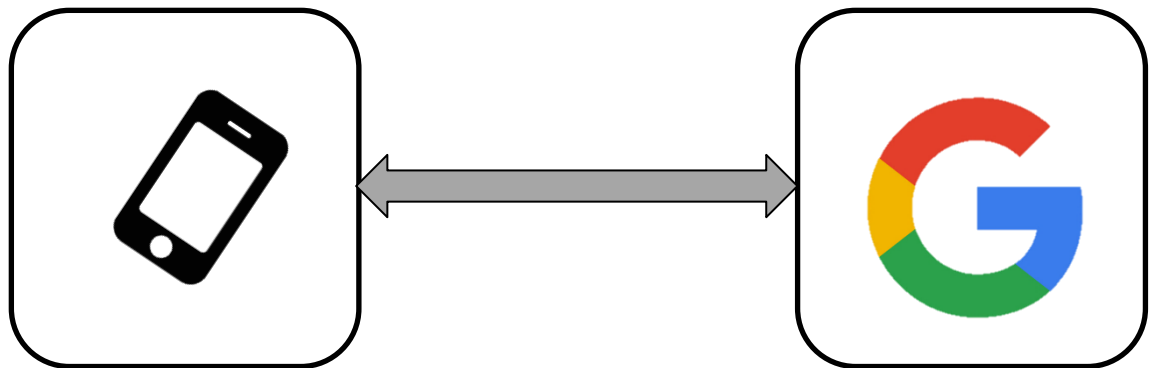
User need to **trust** Google's public key is Google's



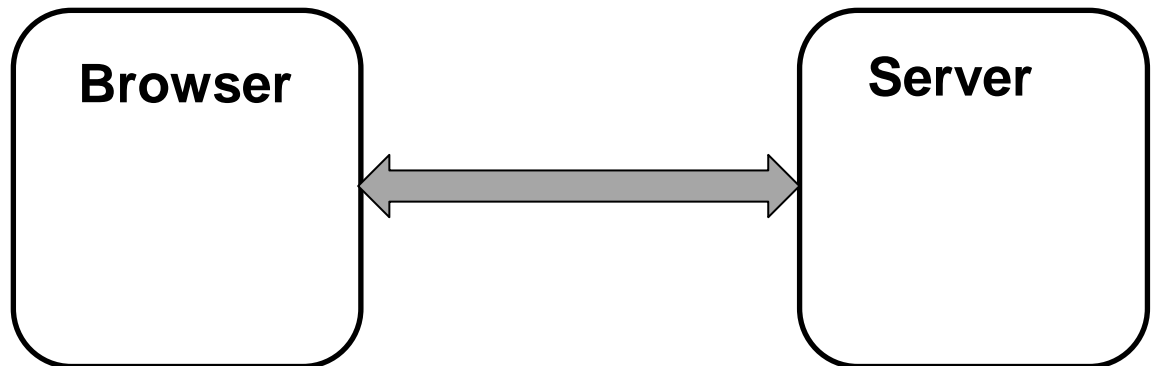
# Certification of public keys



# Certification of public keys

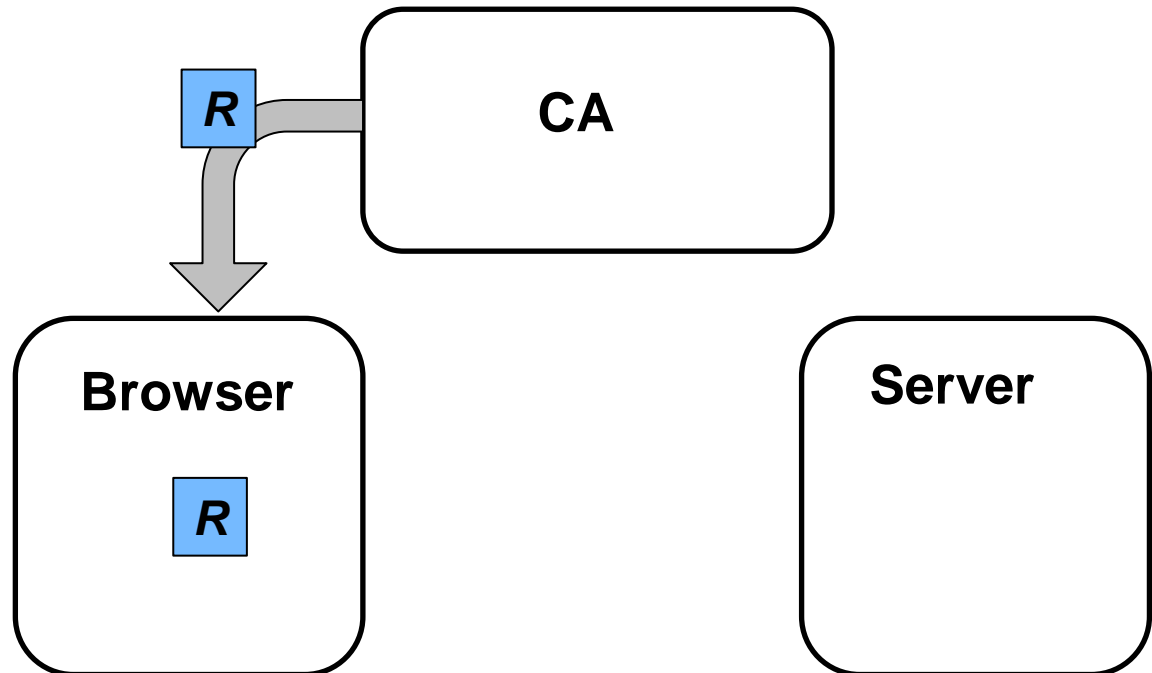


# Certification of public keys



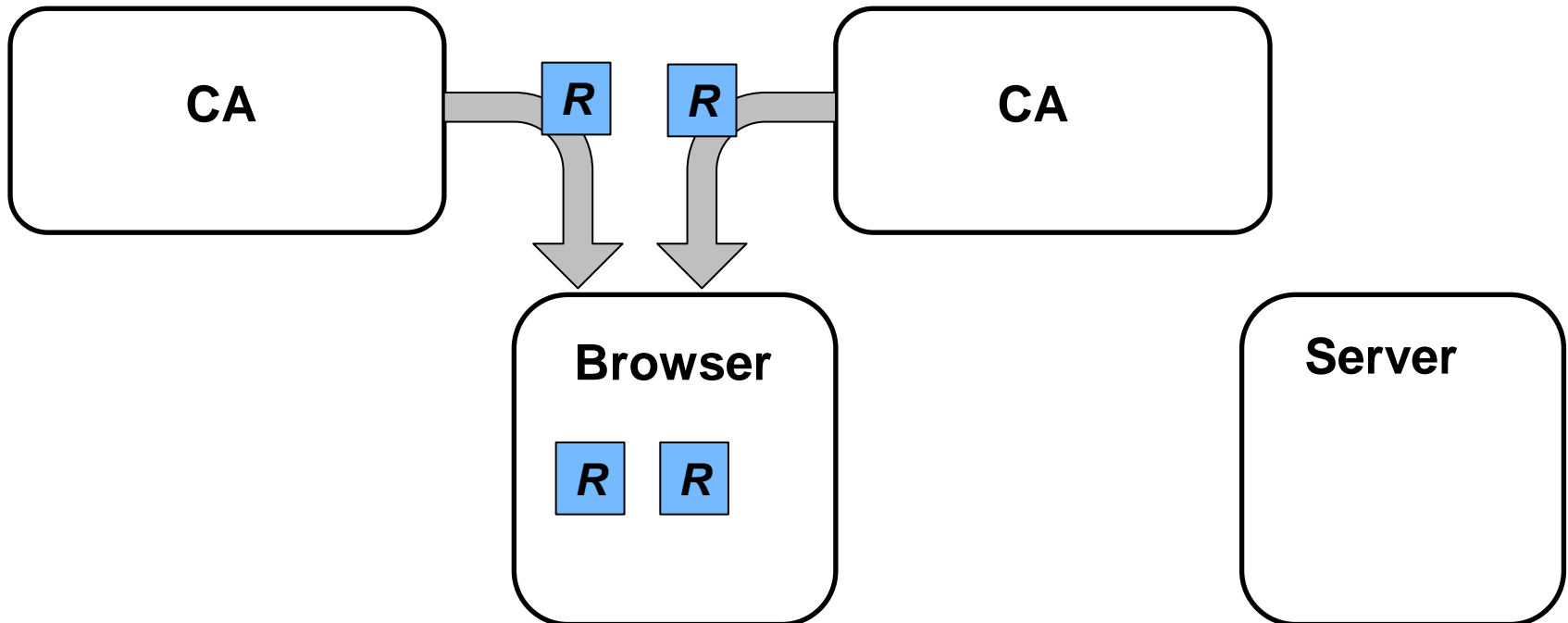
# Certification of public keys

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



# Certification of public keys

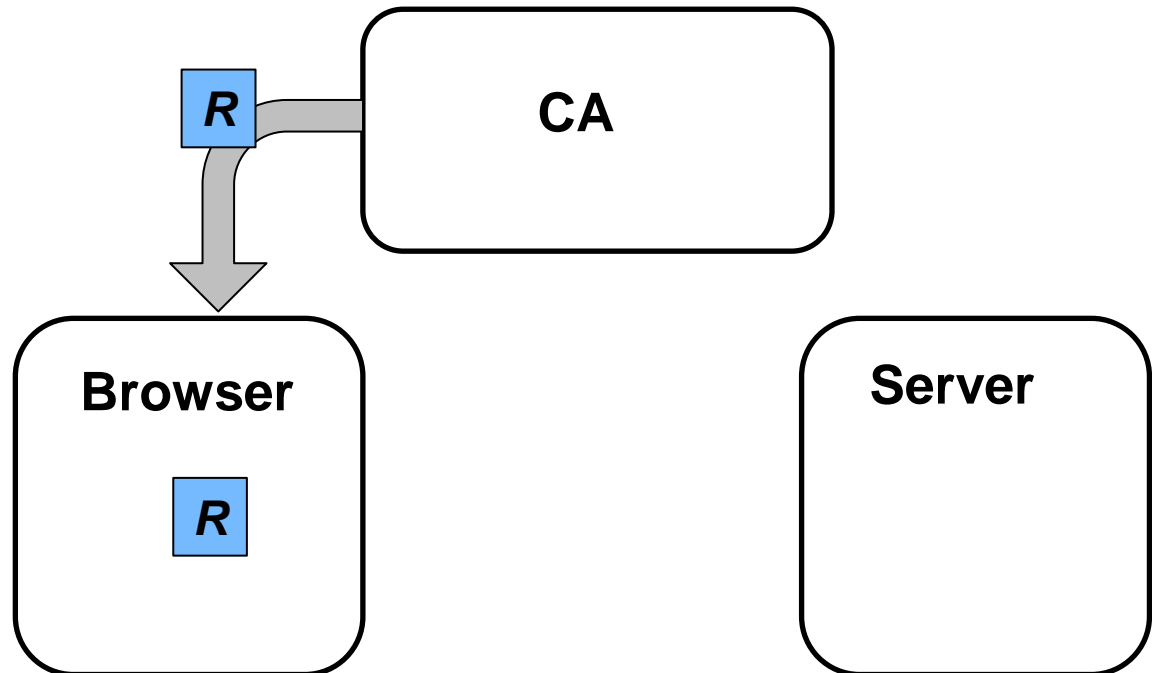
- Browsers have trust stores with root certs (of CAs)





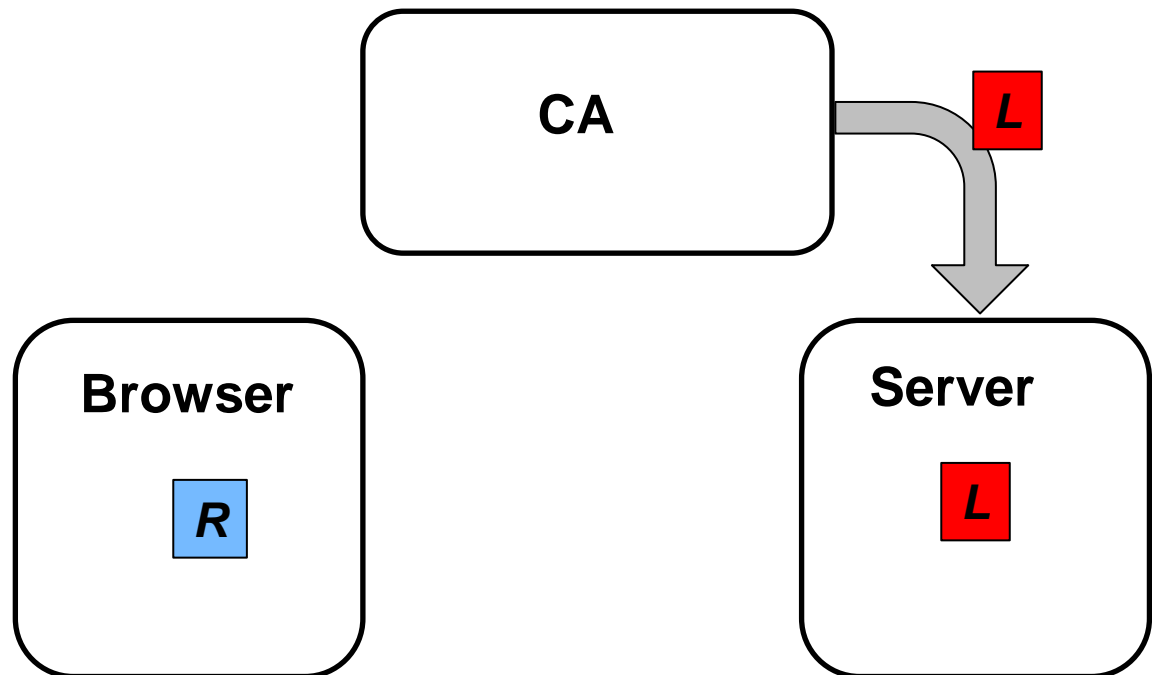
# Certification of public keys

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



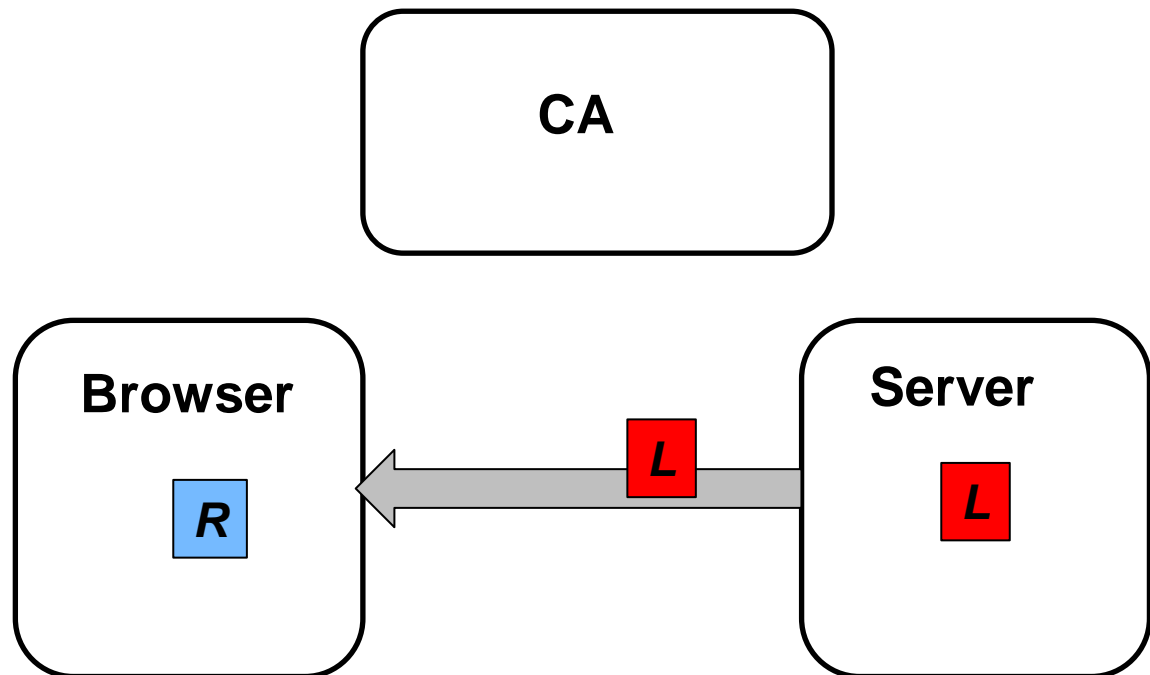
# Certification of public keys

- 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



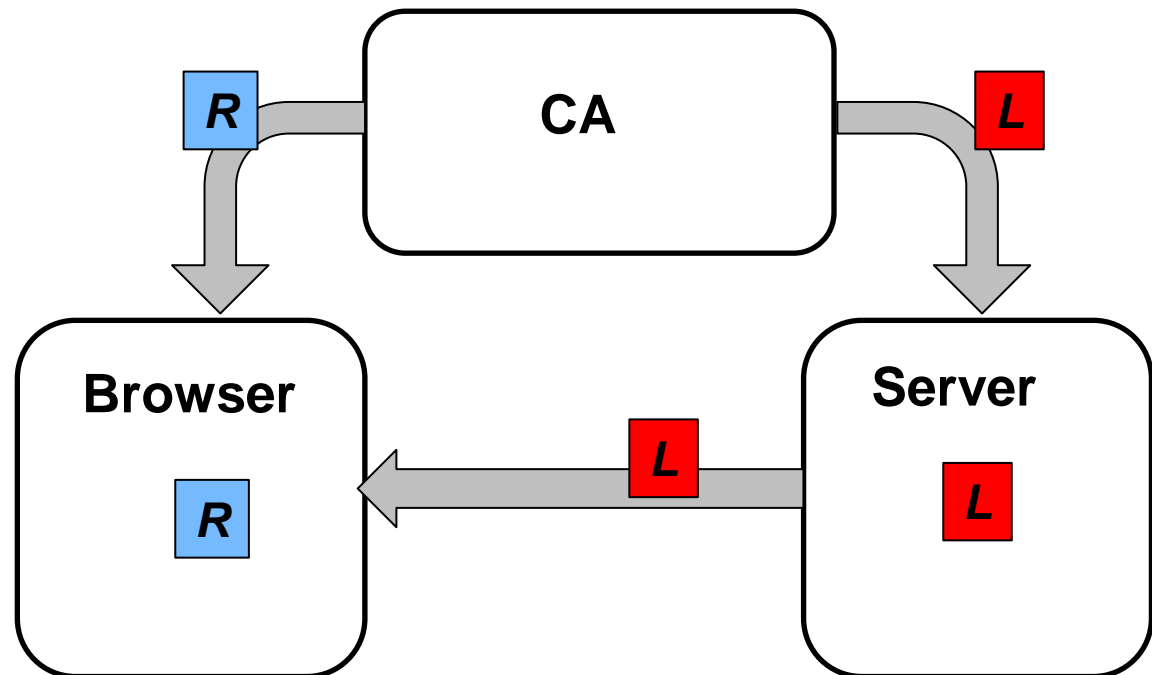
# Certification of public keys

- 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

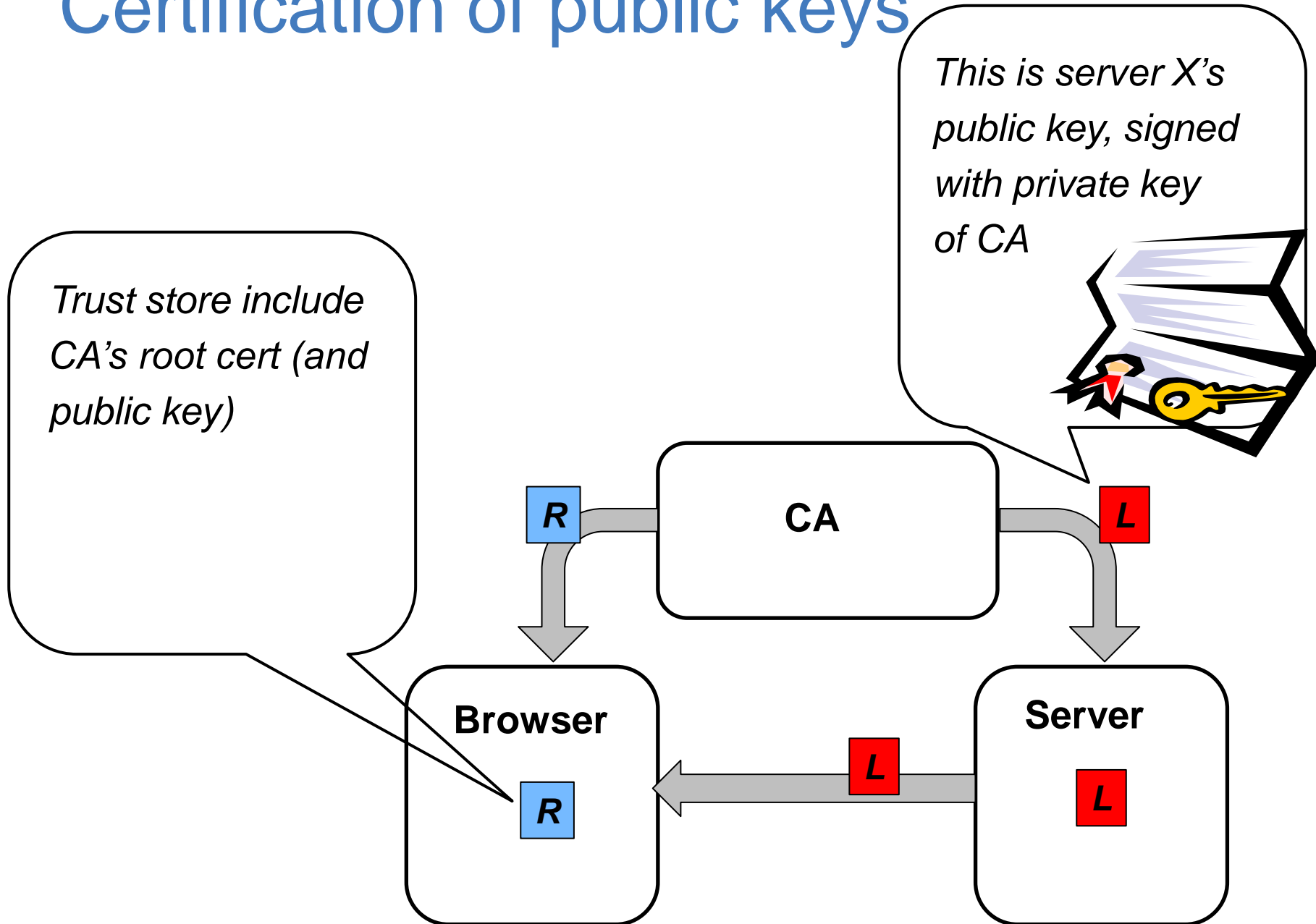


# Certification of public keys

- 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

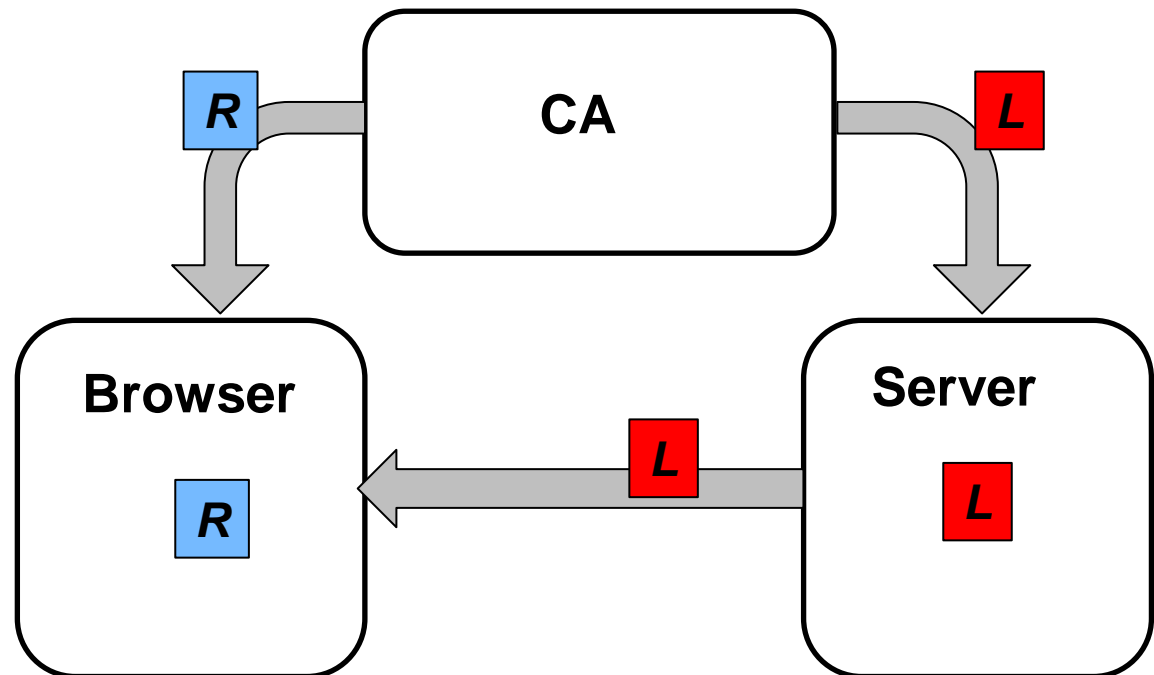


# Certification of public keys



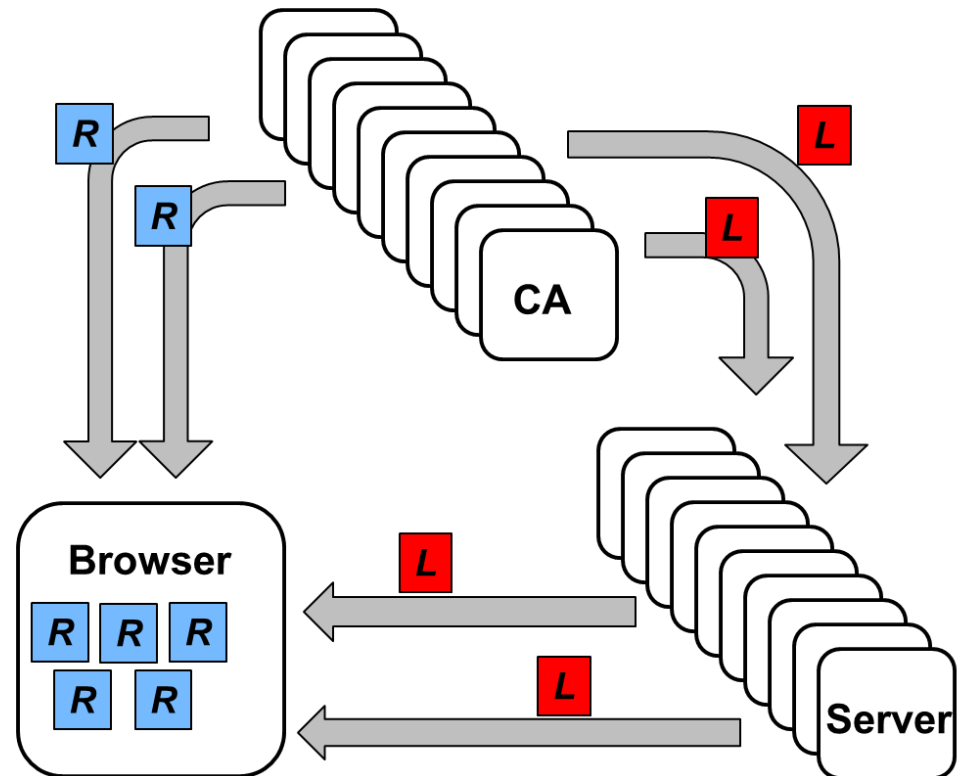
# Certification of public keys

- 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





**www.bankofamerica.com**

Issued by: Entrust Certification Authority - L1M  
Expires: Thursday, June 6, 2019 at 9:57:43 AM Pacific Daylight Time



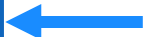
✔ This certificate is valid

*Sample certificate:*

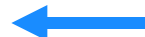
<b>Organization</b>	Bank of America Corporation
<b>Business Category</b>	Private Organization
<b>Organizational Unit</b>	eComm Network Infrastructure
<b>Serial Number</b>	2927442
<b>Common Name</b>	www.bankofamerica.com



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



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





# 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  $\approx$  60*

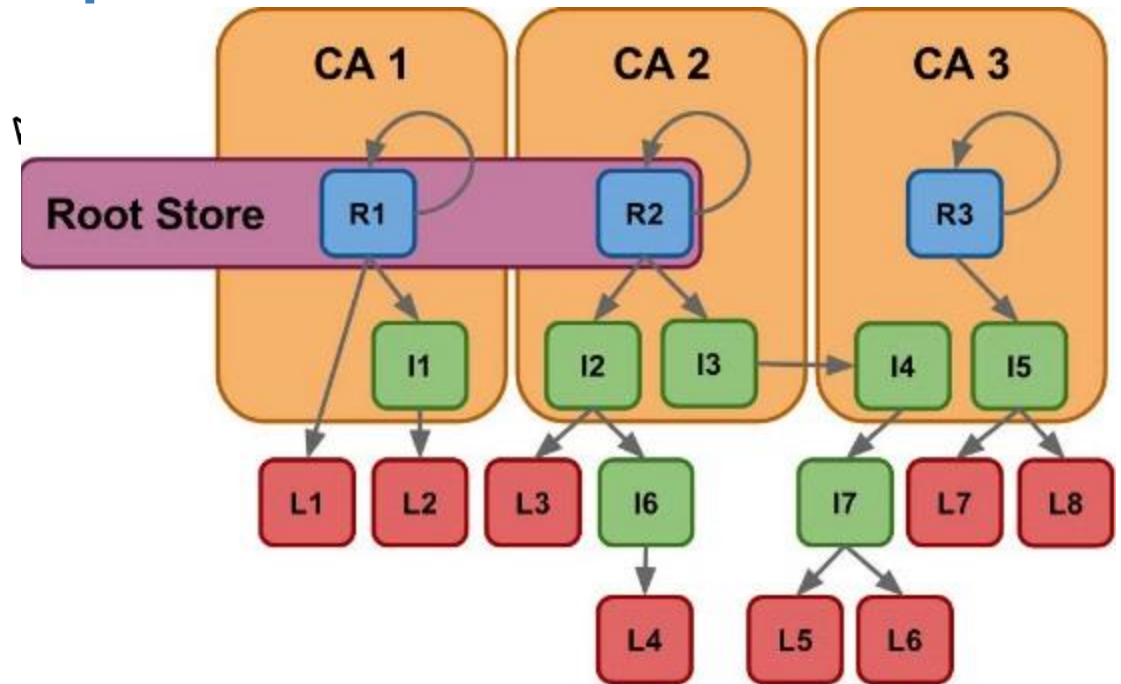
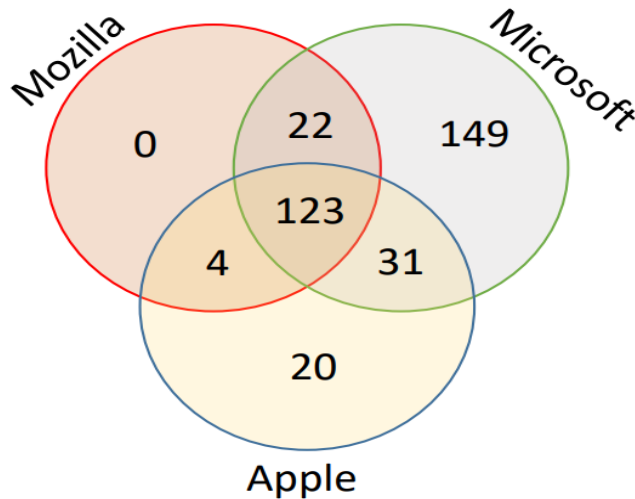
*Intermediate CAs  $\approx$  1200*



 Entrust.net C...Authority (2048)	Jul 24, 2029 7:15:12 AM
 Entrust.net S...ification Authority	May 25, 2019 9:39:40 AM
 ePKI Root Certification Authority	Dec 19, 2034 6:31:27 PM
 Equifax Secu...rtificate Authority	Aug 22, 2018 9:41:51 AM
 Equifax Secure eBusiness CA-1	Jun 20, 2020 9:00:00 PM
 Equifax Secure eBusiness CA-2	Jun 23, 2019 5:14:45 AM
 Equifax Secu...l eBusiness CA-1	Jun 20, 2020 9:00:00 PM
 Federal Common Policy CA	Dec 1, 2030 8:45:27 AM
 FNMT Clase 2 CA	Mar 18, 2019 8:26:19 AM
 GeoTrust Global CA	May 20, 2022 9:00:00 PM
 GeoTrust Pri...ification Authority	Jul 16, 2036 4:59:59 PM
 Global Chambersign Root	Sep 30, 2037 9:14:18 AM



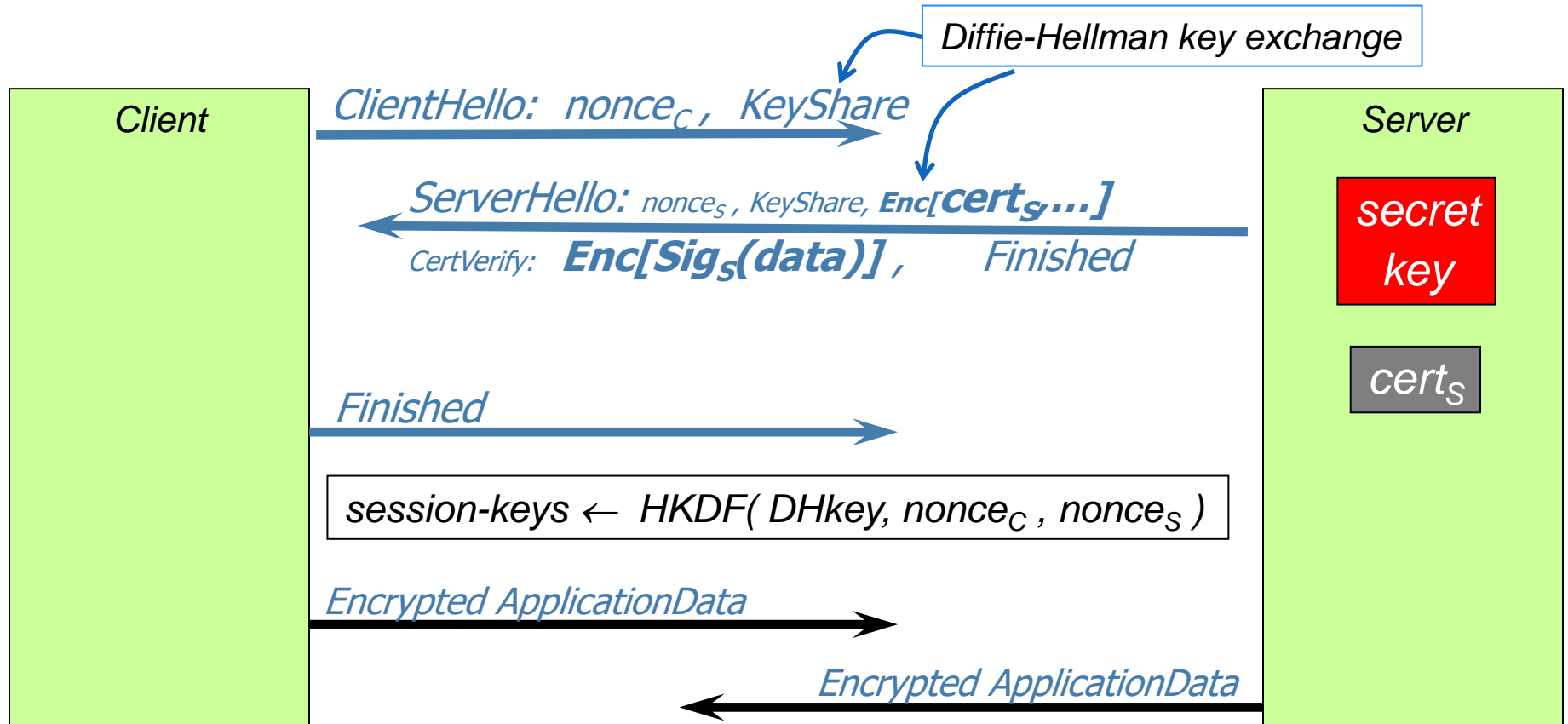
# Trust landscape



[Korzhitskii & Carlsson, 2020]

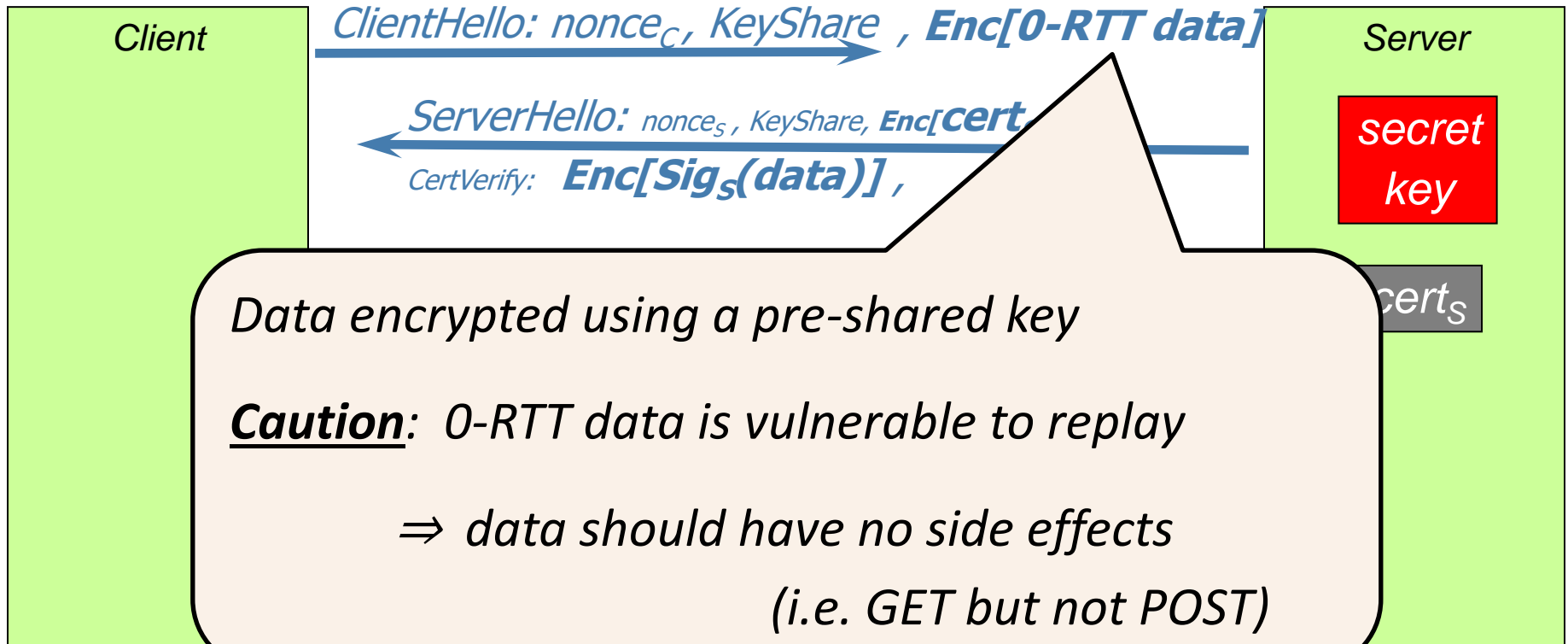
- Delegation of trust to intermediates (I<sub>i</sub>)
- Browsers trust that the servers that can present certs (L<sub>i</sub>) that map to (trusted) root certs are who they claim to be
- Impersonation
  - Any trusted CA (R<sub>i</sub>) or intermediate (I<sub>i</sub>) can issue rogue certs
  - Very difficult to know all certs issued in once name

# 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

■ Connection - secure (strong TLS 1.3)

The connection to this site is encrypted and authenticated using TLS 1.3 (a strong protocol), X25519 (a strong key exchange), and AES\_128\_GCM (a strong cipher).

*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

July 2018 (Chrome 68)

 Not secure | example.com

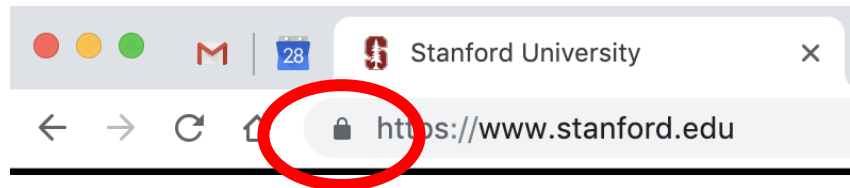
## Chrome's gradual blocking of mixed content

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

# HTTPS in the Browser



# The lock icon: TLS indicator

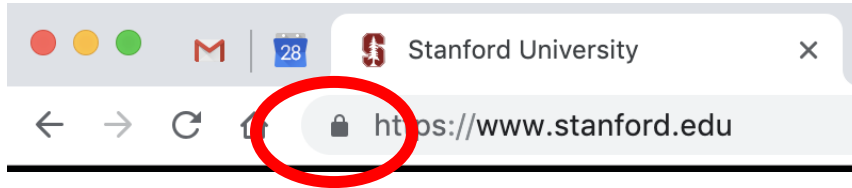


## Intended goal:

- 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



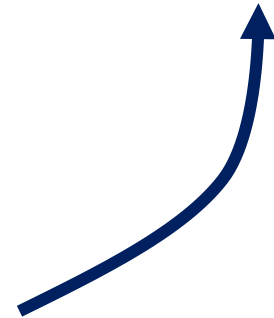
All elements on the page fetched using HTTPS

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

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

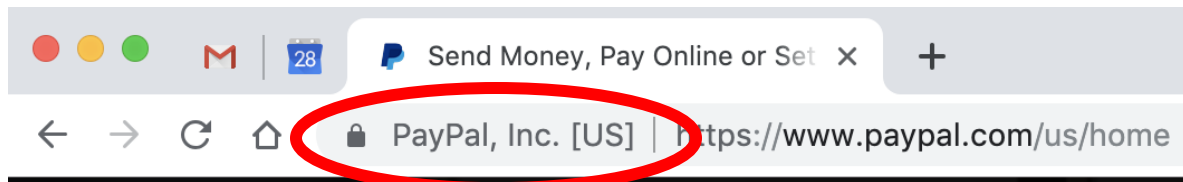


# The lock UI: Extended Validation (EV) Certs

Harder to obtain than regular certs

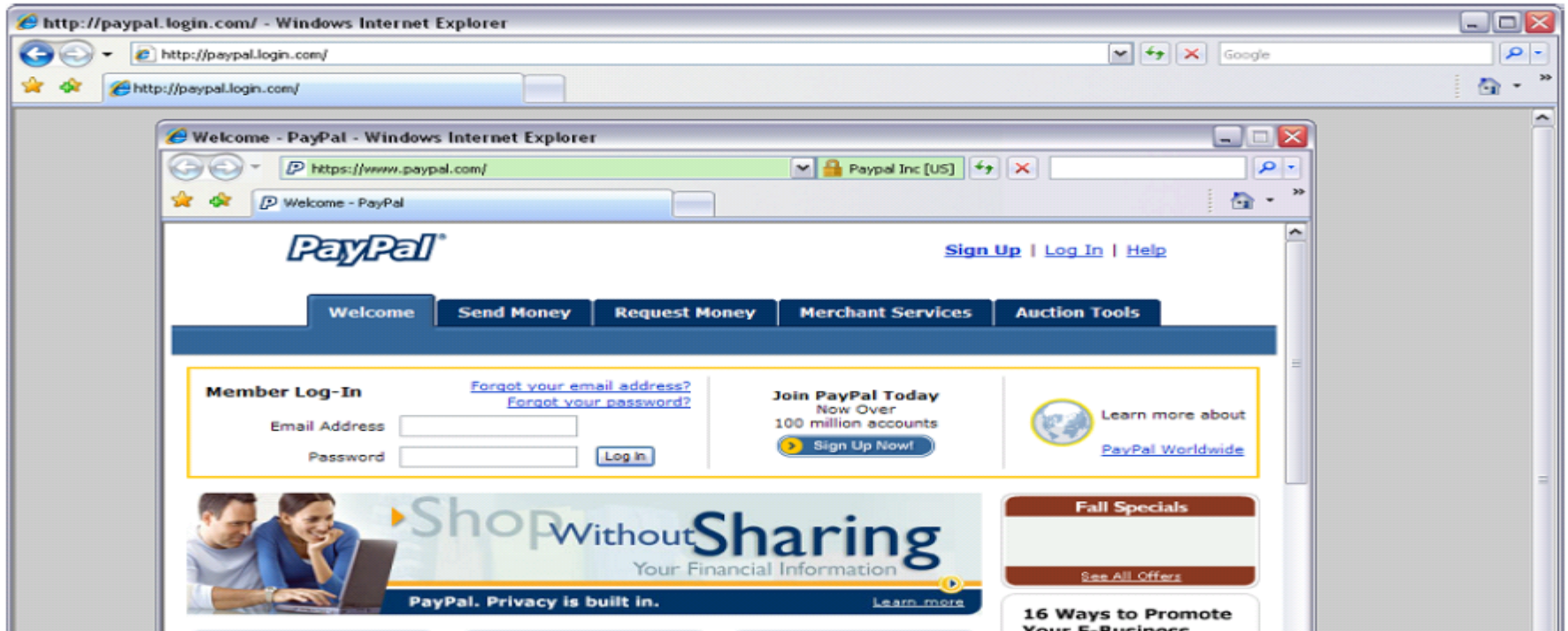
- requires human at CA to approve cert request
- no wildcard certs (e.g. \*.stanford.edu )

Helps block “semantic attacks”: [www.bankofthevest.com](http://www.bankofthevest.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

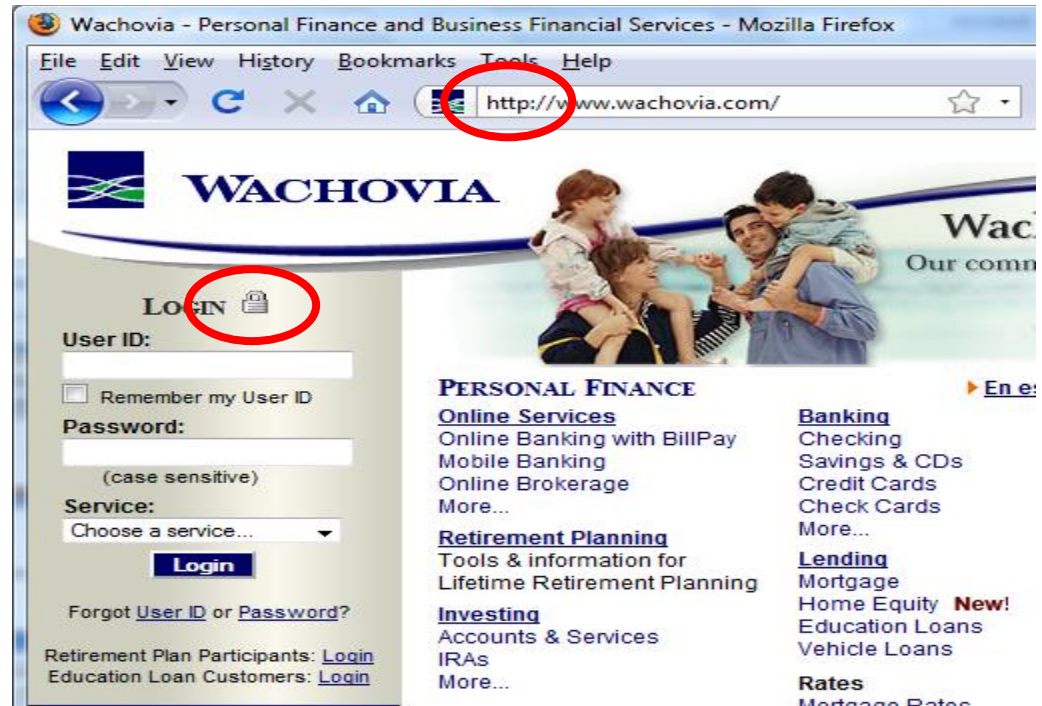
Suppose user lands on HTTP login page.

- say, by typing HTTP URL into address bar

View source:

```
<form method="post"
```

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



(old site)

# HTTPS and login pages: guidelines

General guideline:

Response to

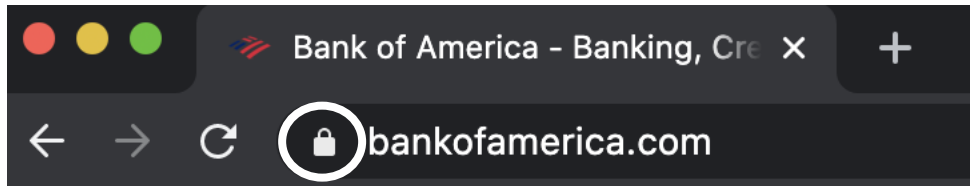
<http://login.site.com>

should be

Location: <https://login.site.com>

*(redirect)*

*Should be the response  
to every HTTP request ...*



# Problems with HTTPS and the Lock Icon

# 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

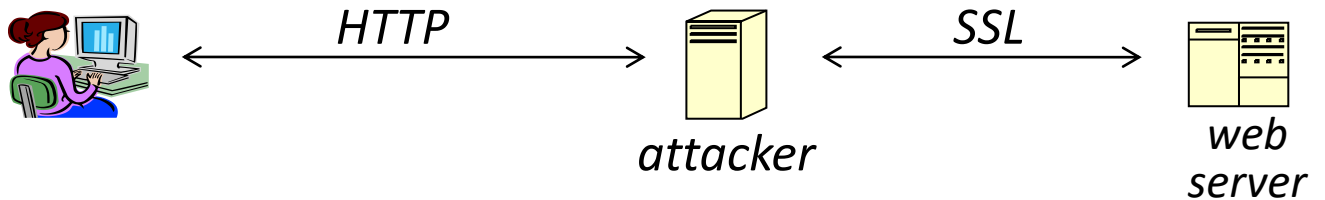


# 1. HTTP $\Rightarrow$ HTTPS upgrade

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]



<code>&lt;a href=https://...&gt;</code>	$\rightarrow$	<code>&lt;a href=http://...&gt;</code>	
Location: https://...	$\rightarrow$	Location: http://...	(redirect)
<code>&lt;form action=https://... &gt;</code>	$\rightarrow$	<code>&lt;form action=http://...&gt;</code>	

# Tricks and Details

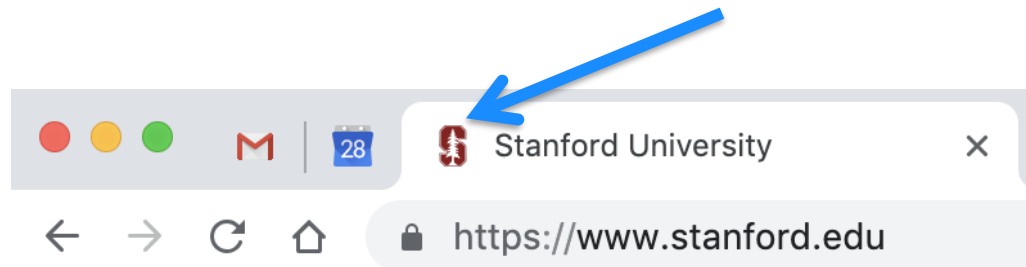
Tricks: drop-in a clever fav icon (older browsers)



→



⇒ fav icon no longer presented in address bar

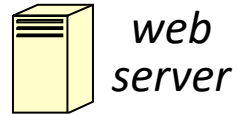


Number of users who detected HTTP downgrade: 0

# Defense: Strict Transport Security (HSTS)

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

*(ignored if not over HTTPS)*



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/>

Enter a domain for the HSTS preload list:

paypal.com

Check status and eligibility

*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 ``

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

Solution: gradual transition using CSP

## Content-Security-Policy: upgrade-insecure-requests

```

```

```

```

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

```
<a href="http://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

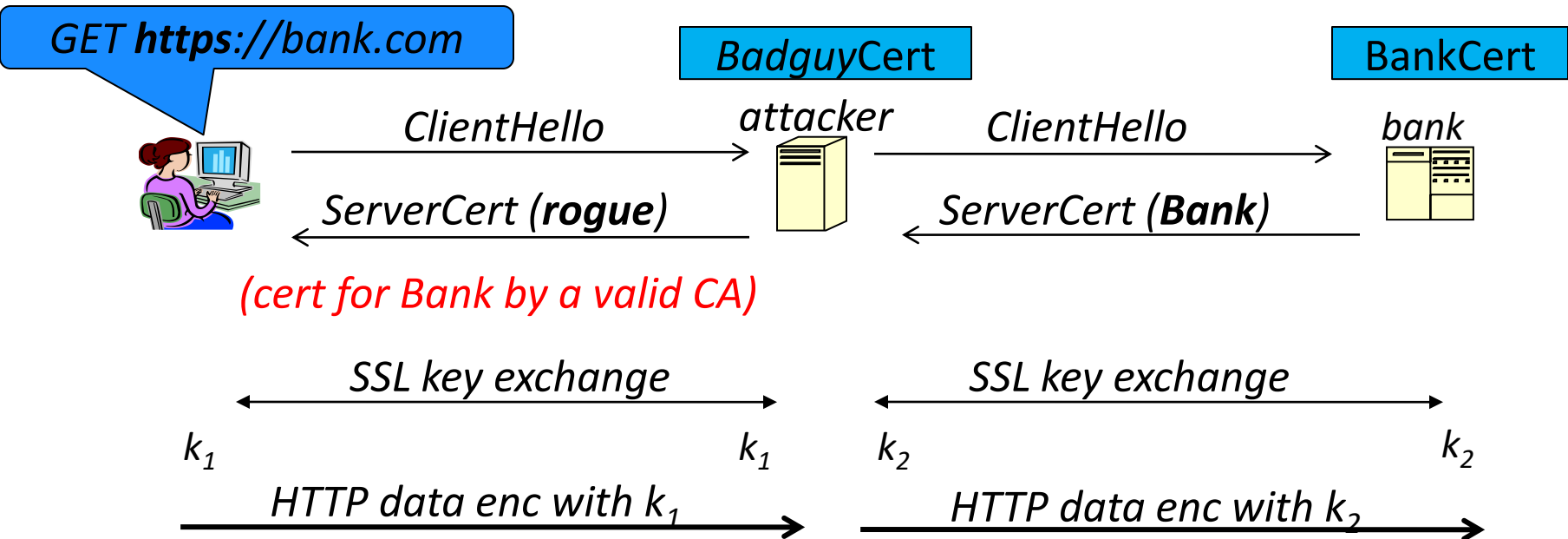
(2) Chrome restricts India CCA root to only seven Indian domains

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

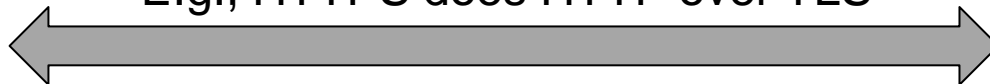


# Motivation and high-level problem

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



E.g., HTTPS does HTTP over TLS

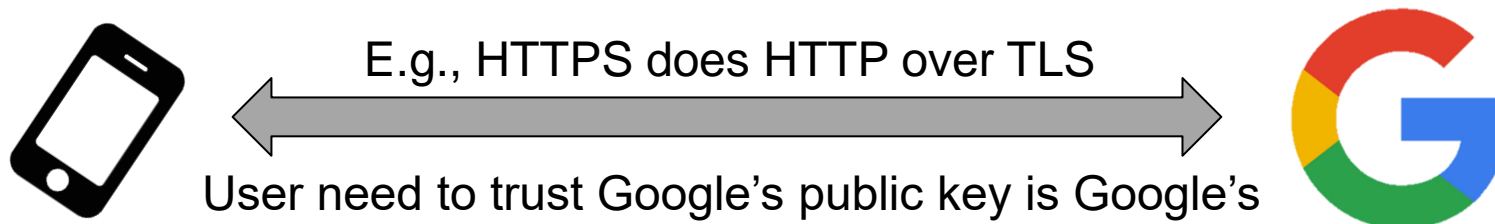


User need to trust Google's public key is Google's



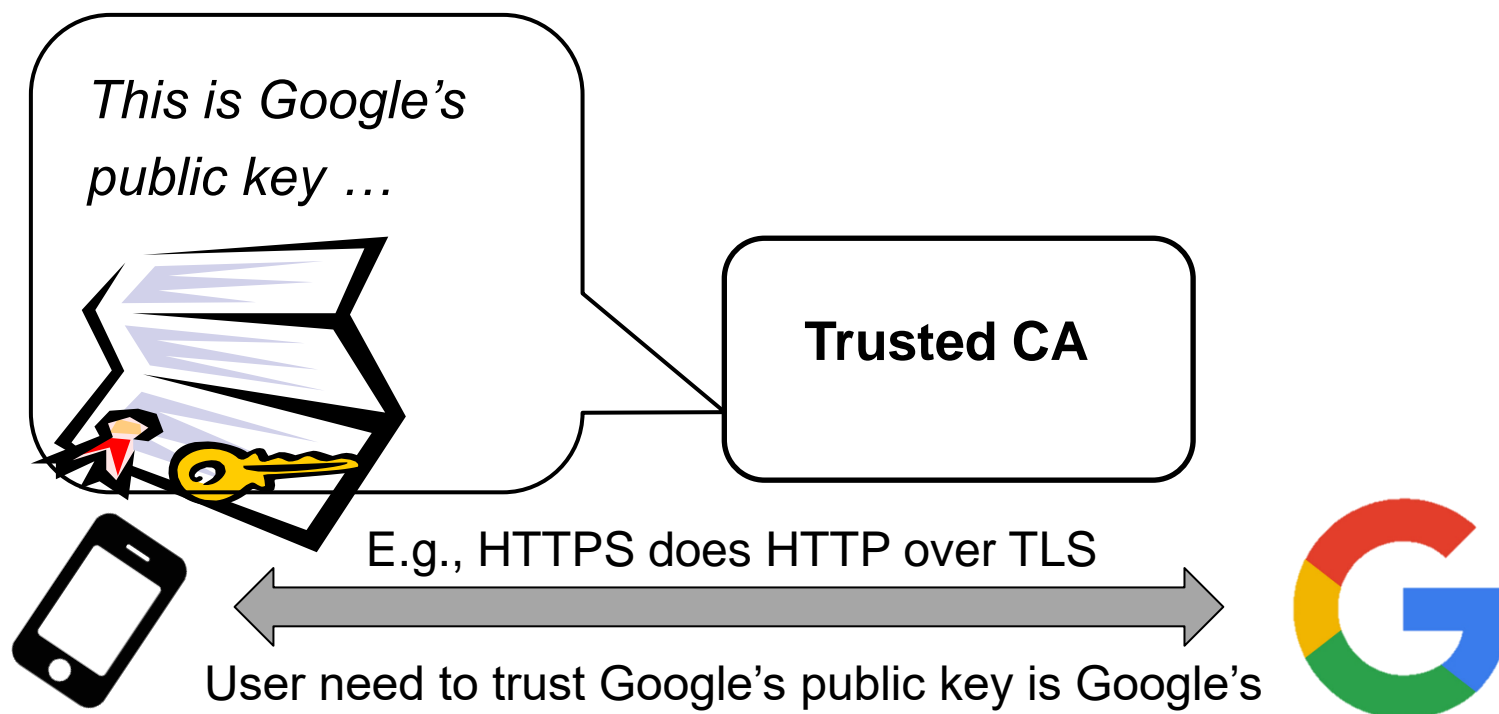
# Motivation and high-level problem

- 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



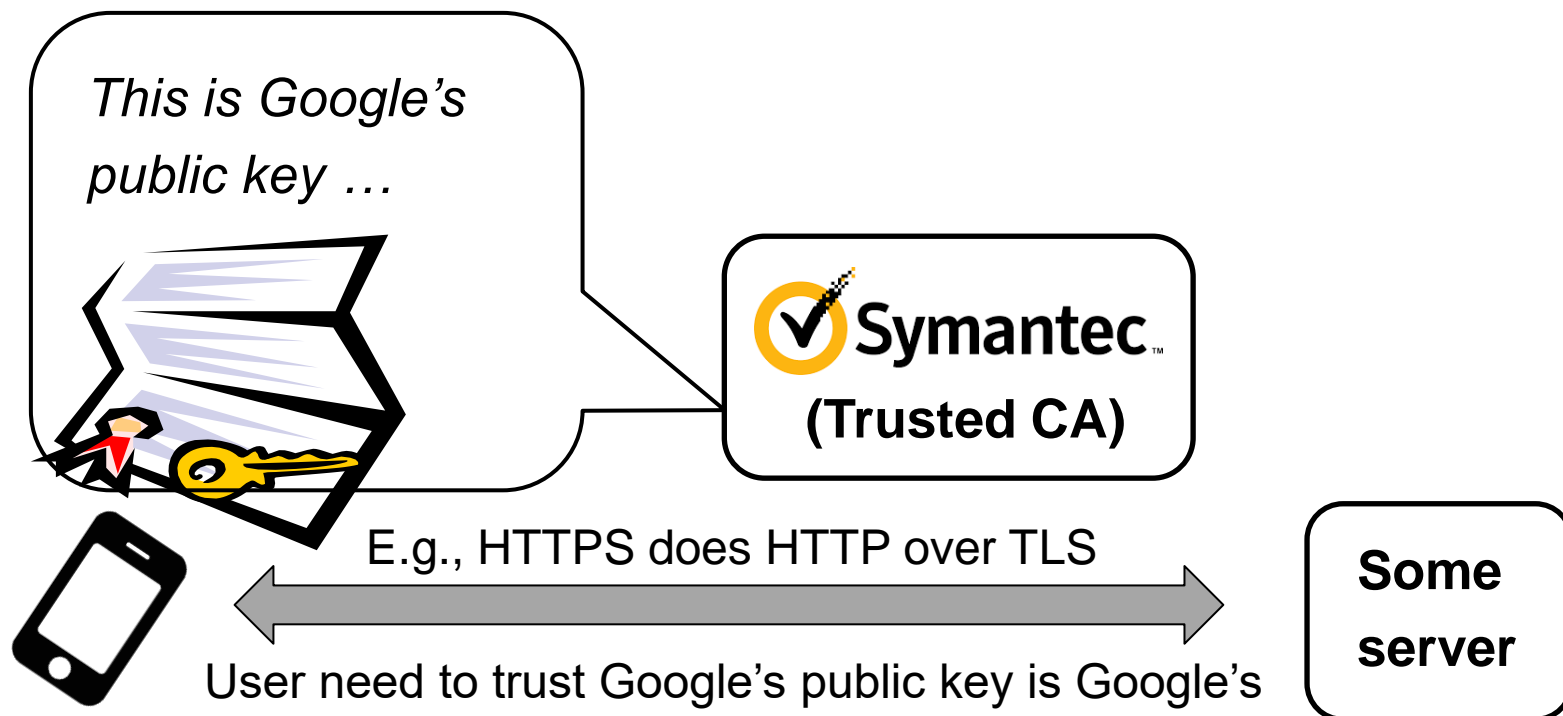
# Motivation and high-level problem

- 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

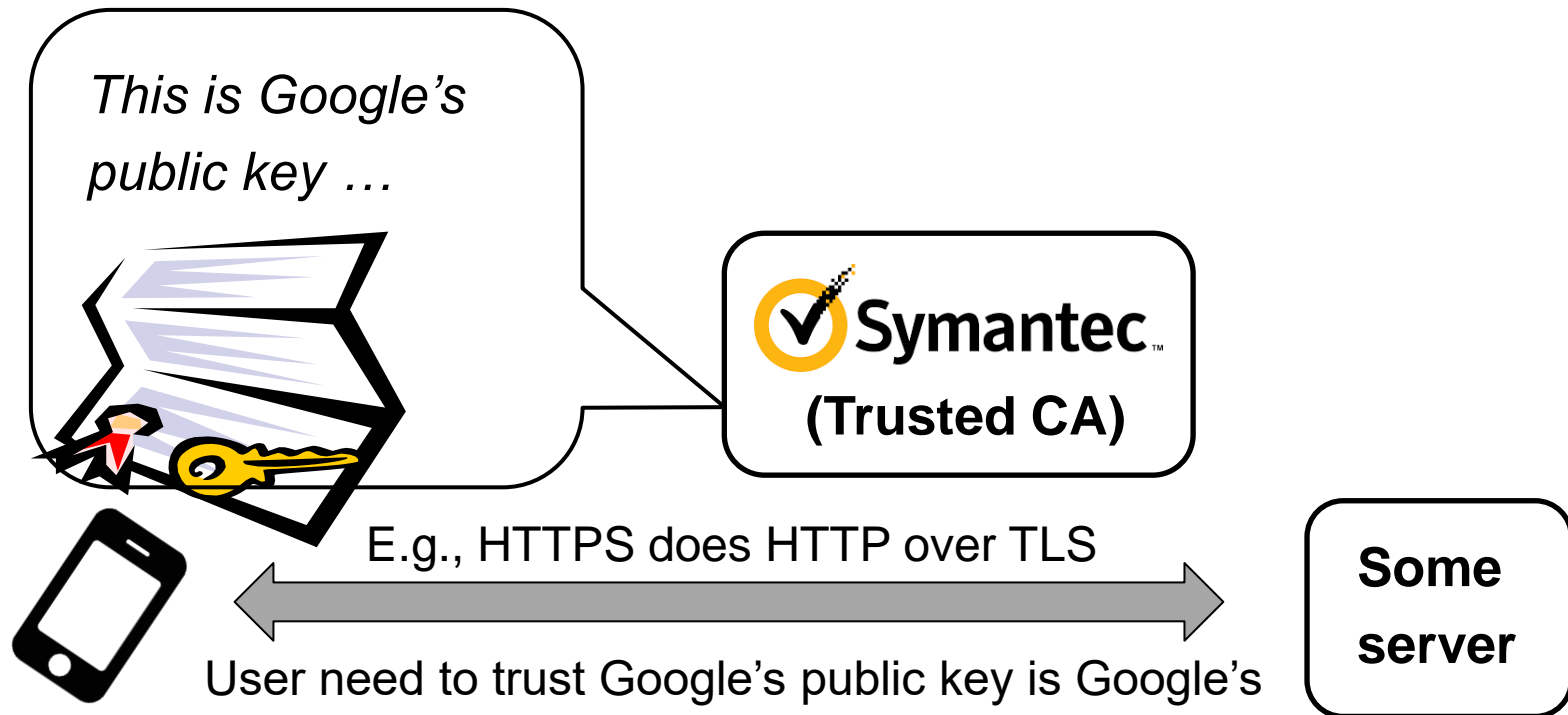
- However, mistakes happen ...
  - E.g., in Oct. 2015, Google discovered (using CT) that Symantec had issued test certificates for 76 domains that they did not own (including Google domains) and another 2,458 unregistered domains ...



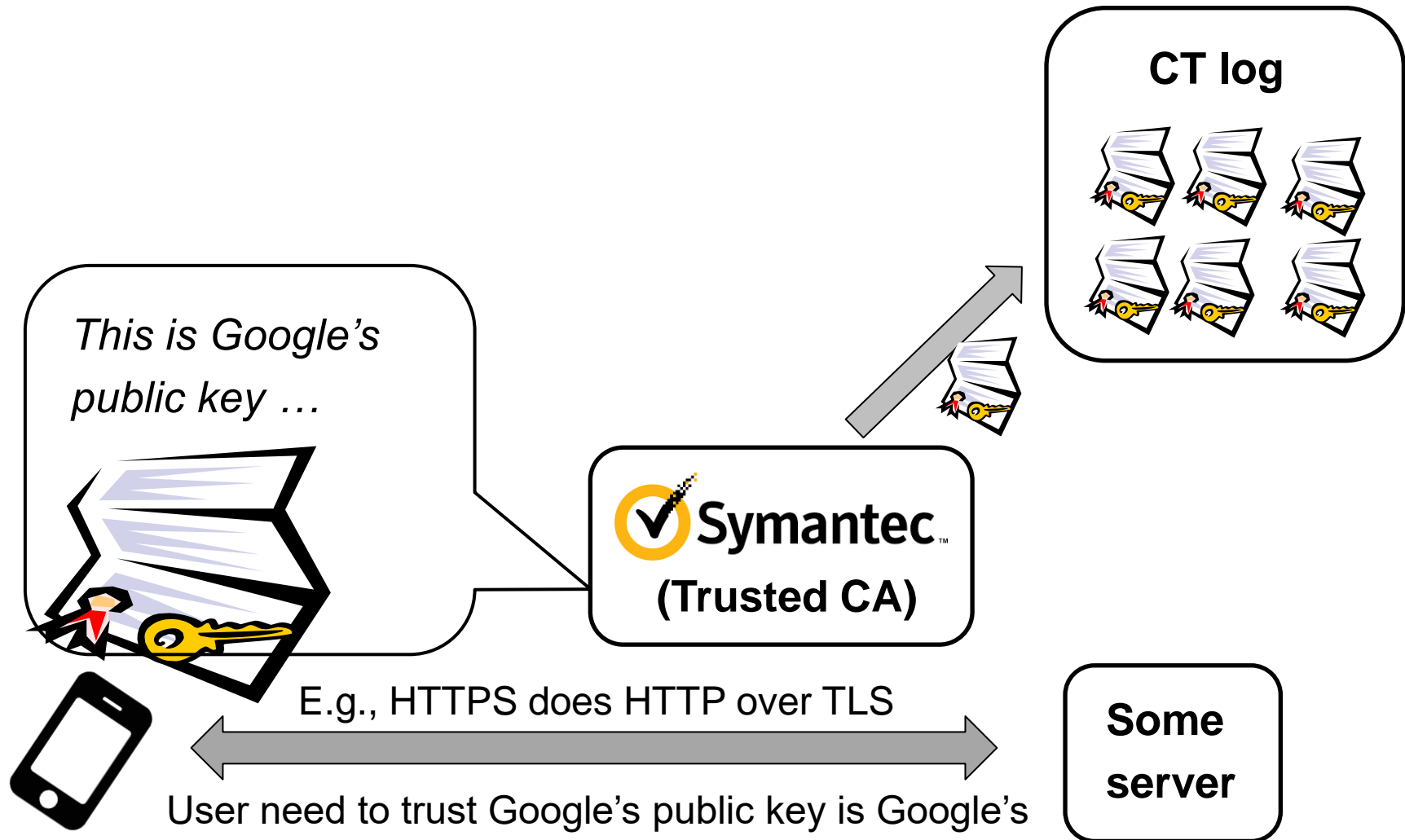
# CT: Emerging trust-monitoring solution

- 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 ...

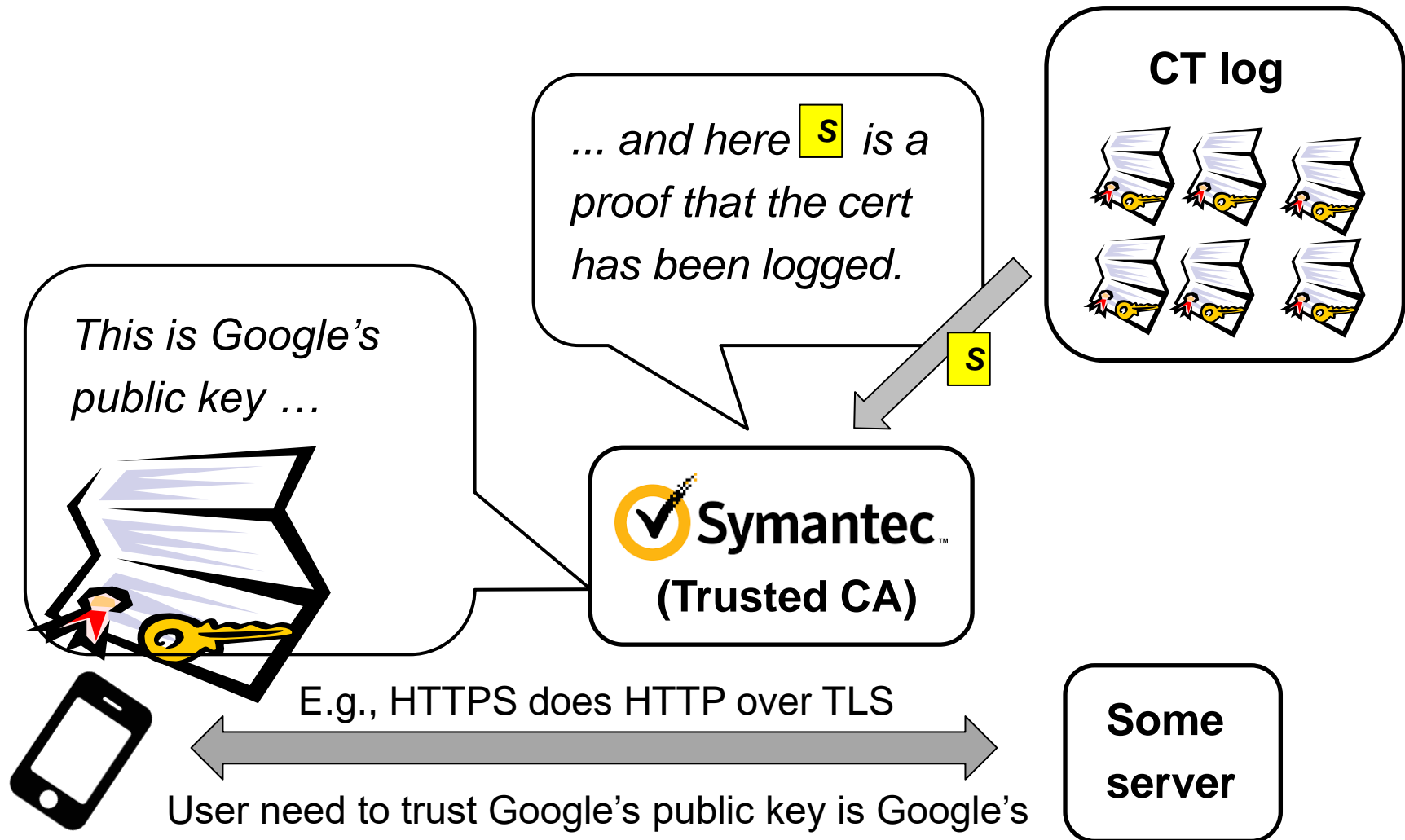
# CT: Emerging trust-monitoring solution



# CT: Emerging trust-monitoring solution

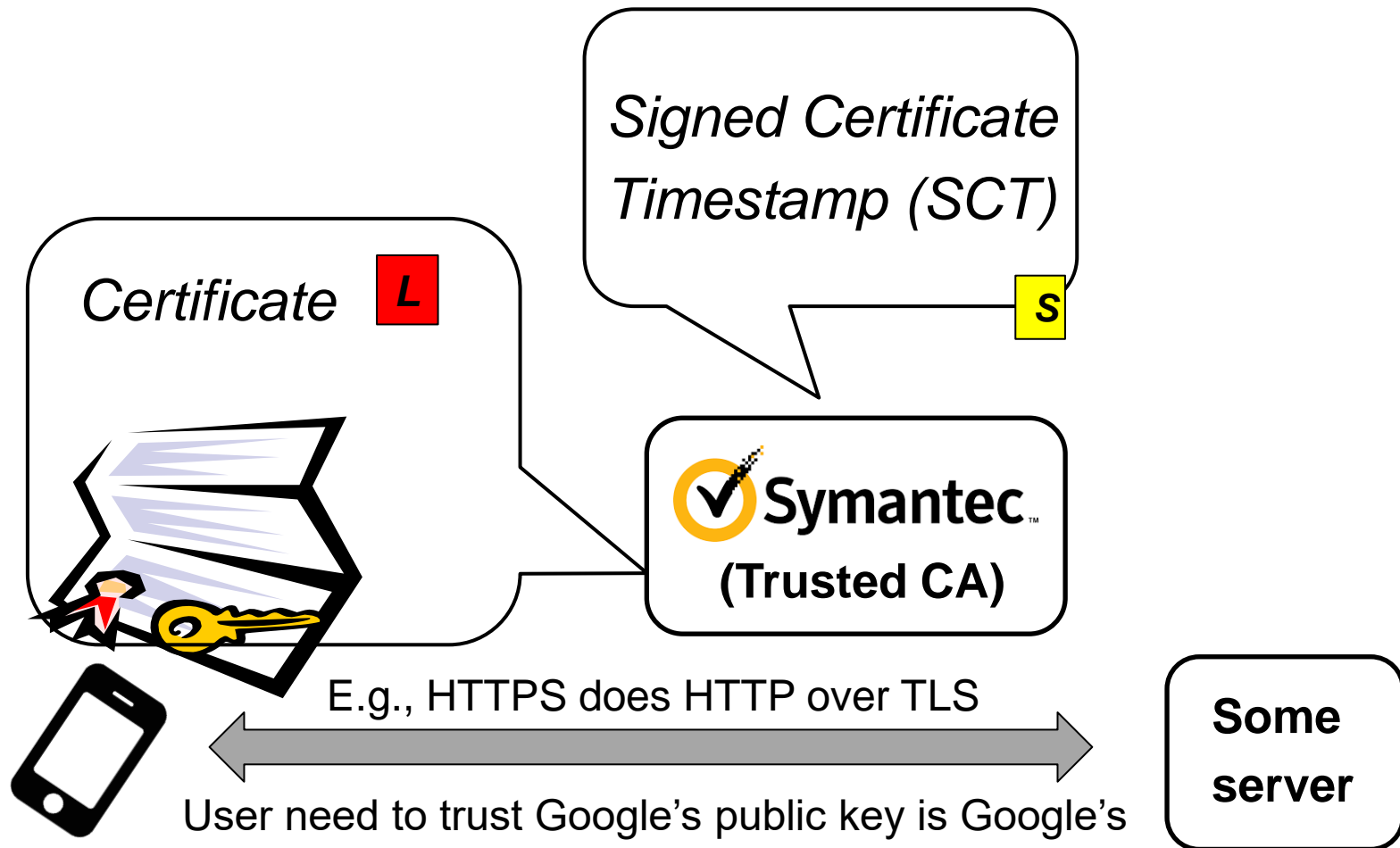


# CT: Emerging trust-monitoring solution



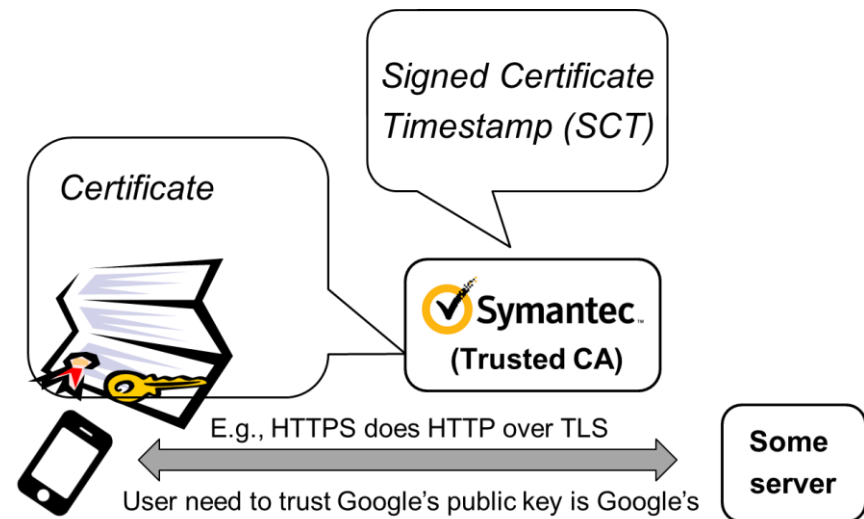


# CT: Emerging trust-monitoring solution

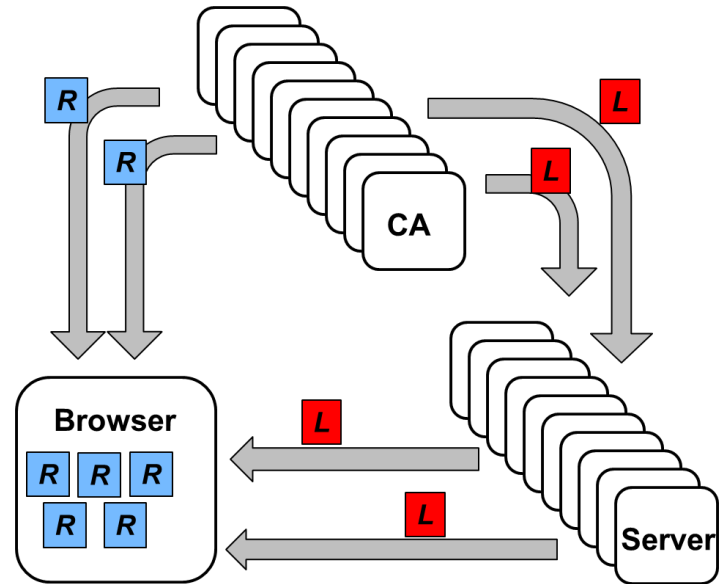


# Signed Certificate Timestamps (SCTs)

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

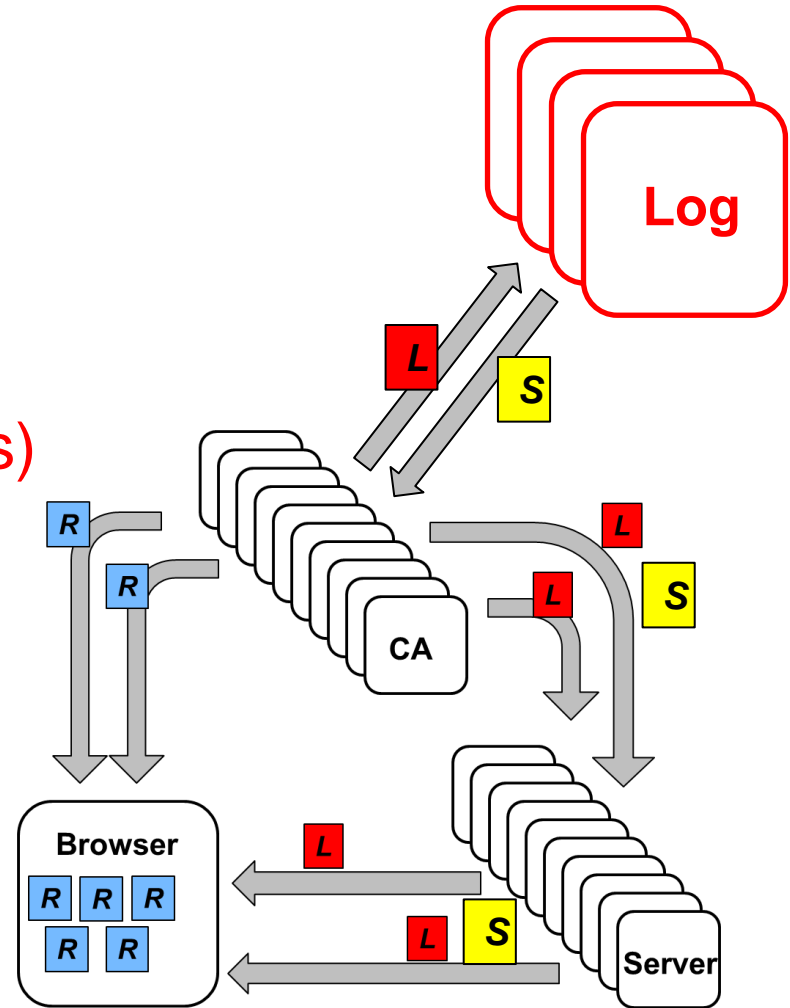


# Certification Transparency (CT)



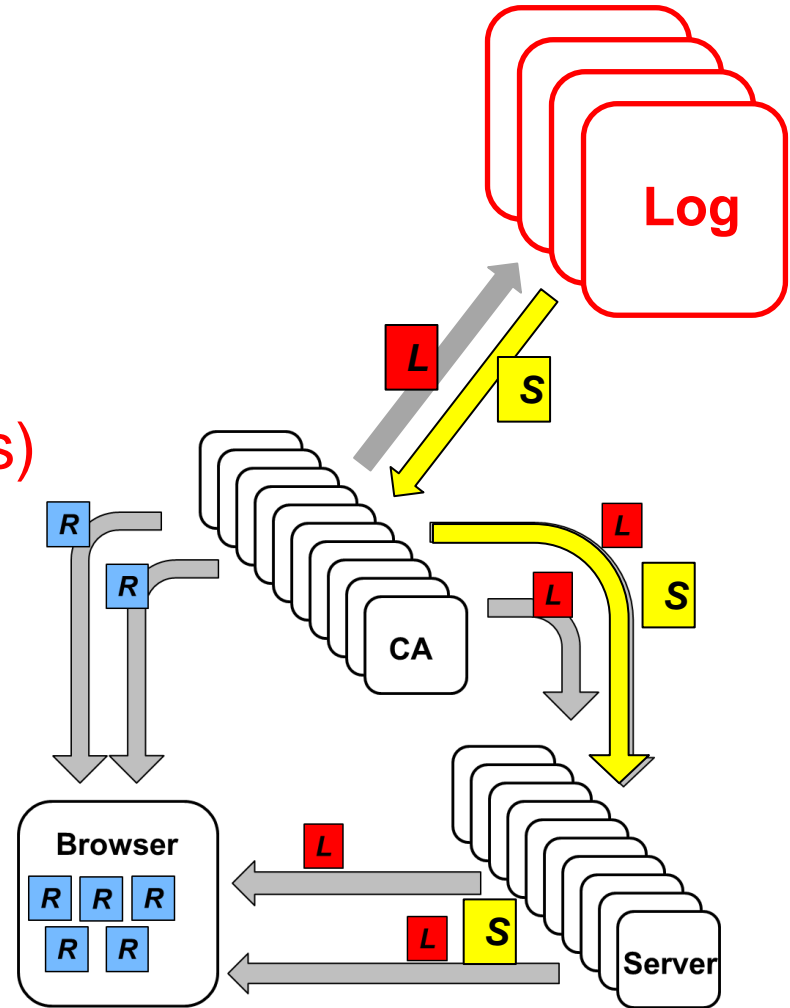
# Certification Transparency (CT)

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



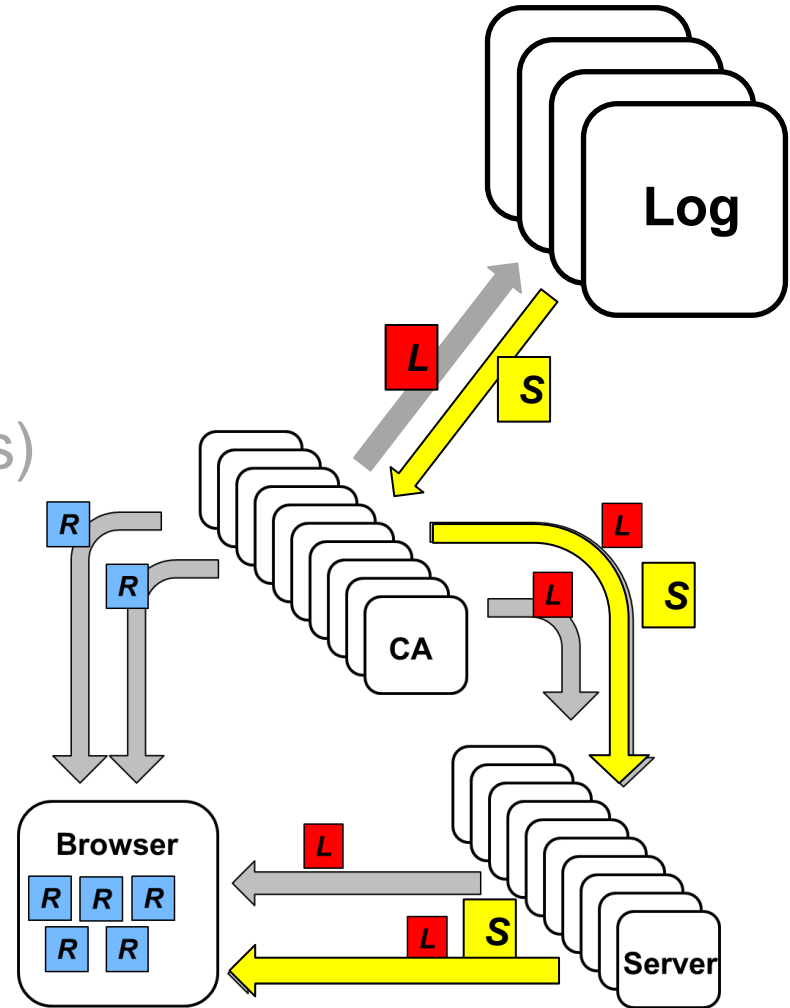
# Certification Transparency (CT)

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



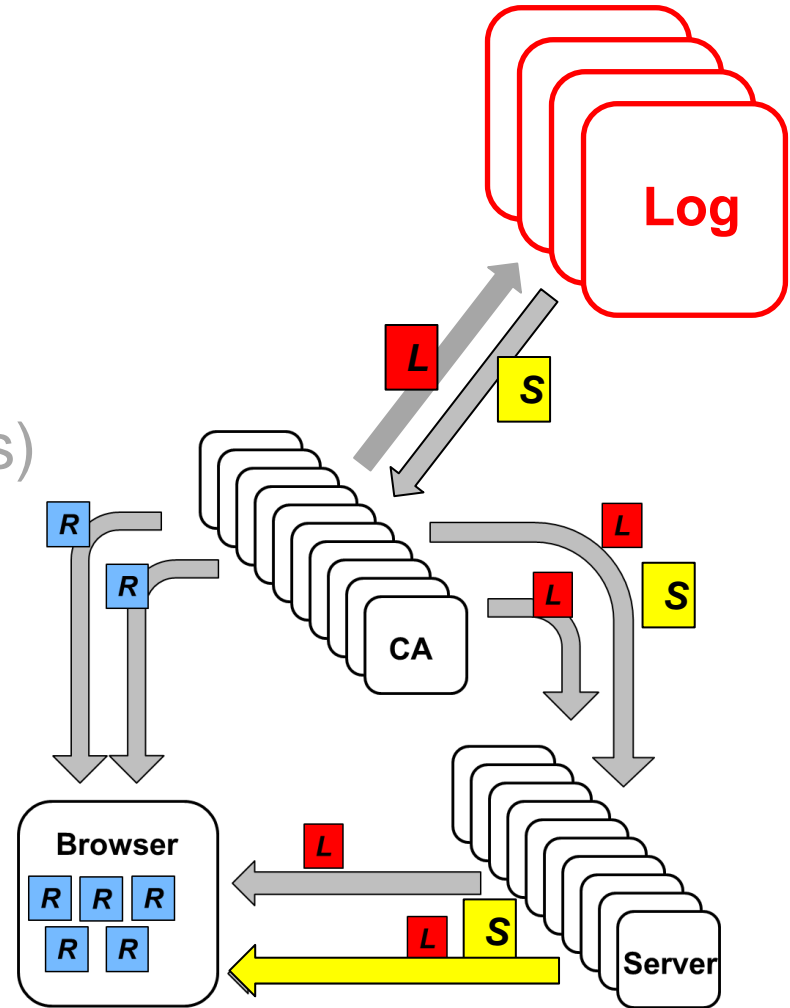
# Certification Transparency (CT)

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

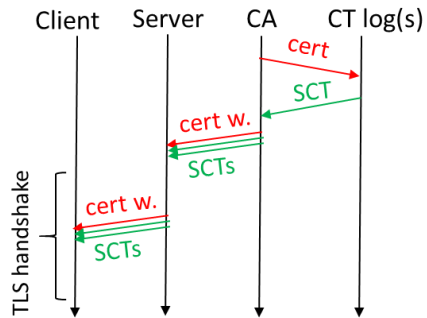
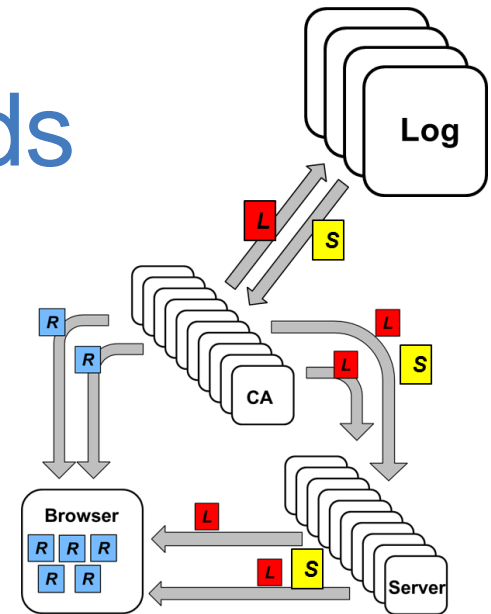


# Certification Transparency (CT)

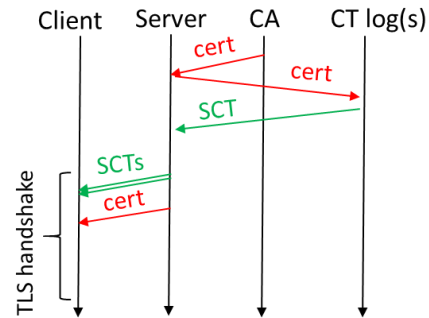
- Logs
  - Public record of certs
  - Append only (Merkle trees)
  - Create SCTs
- **SCTs**
  - **Proof cert is logged**



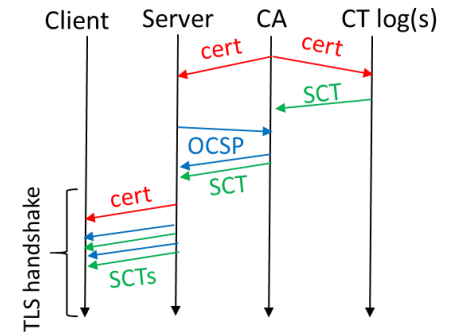
# Three SCT delivery methods



(a) X.509v3 extension



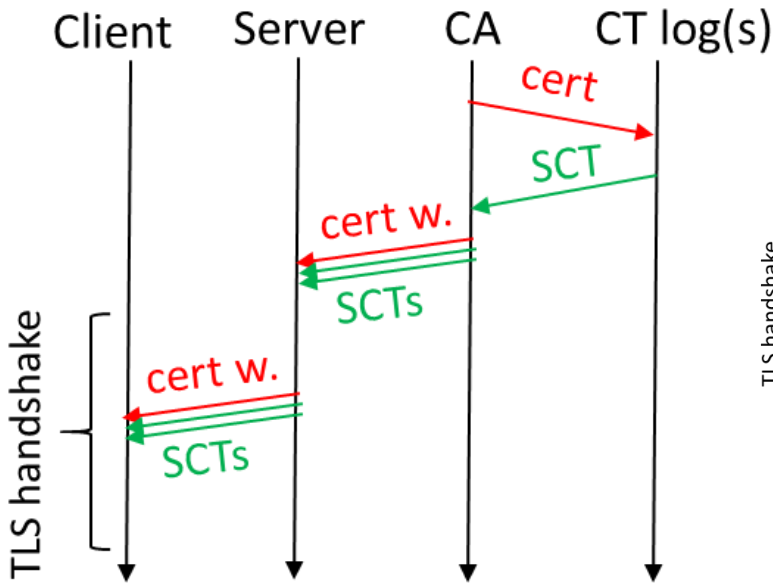
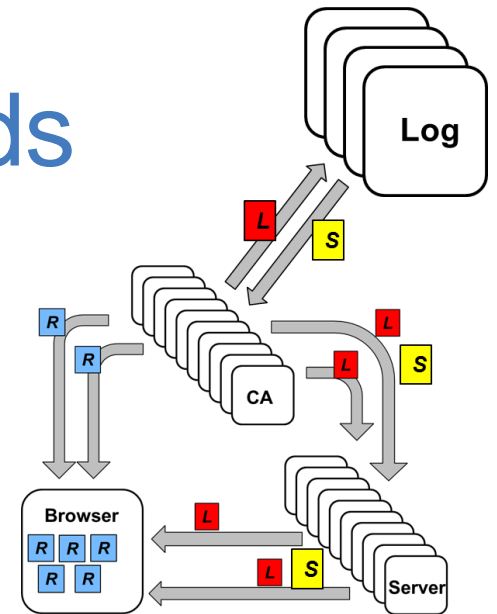
(b) TLS extension



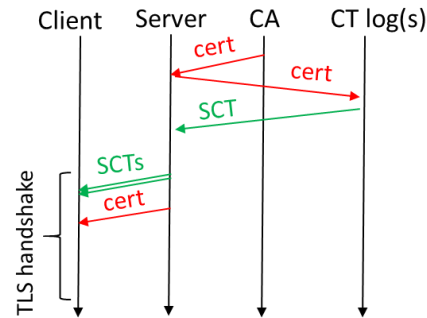
(c) OCSP stapling



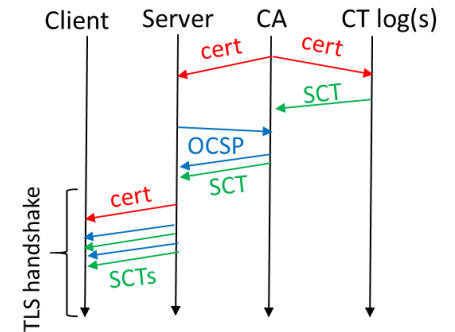
# Three SCT delivery methods



(a) X.509v3 extension

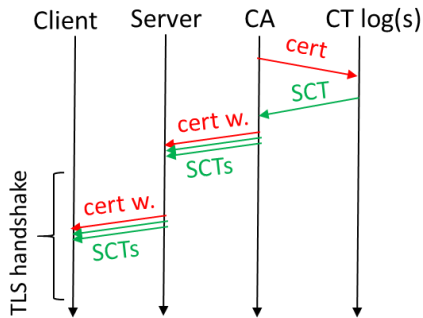
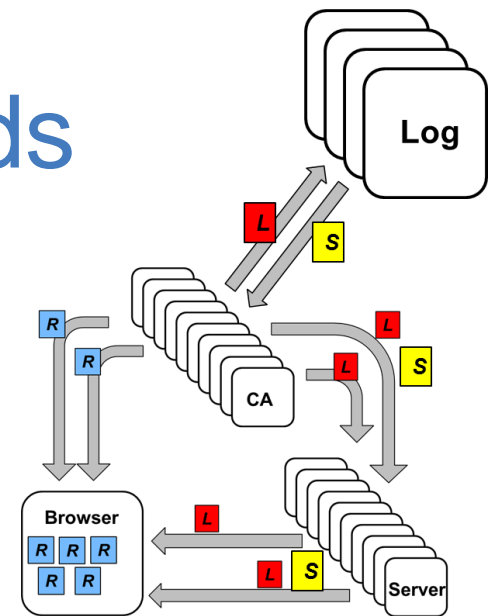


(b) TLS extension

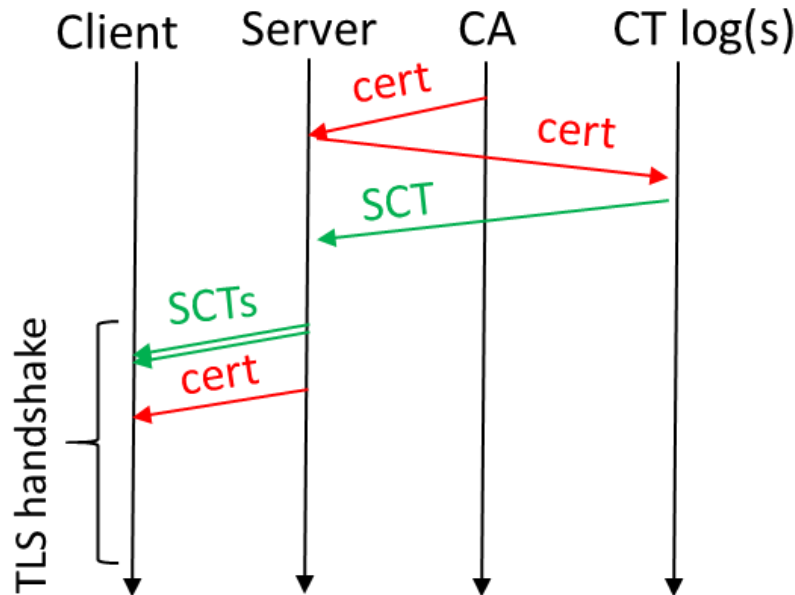


(c) OCSP stapling

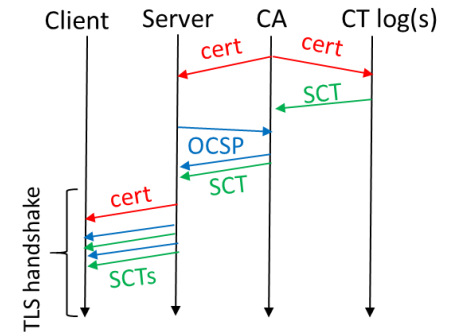
# Three SCT delivery methods



(a) X.509v3 extension

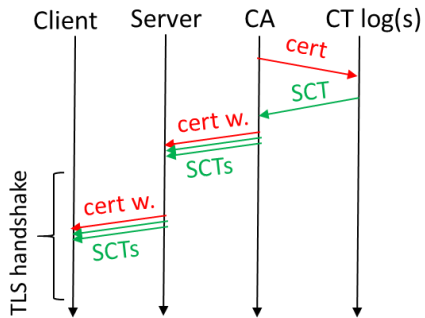
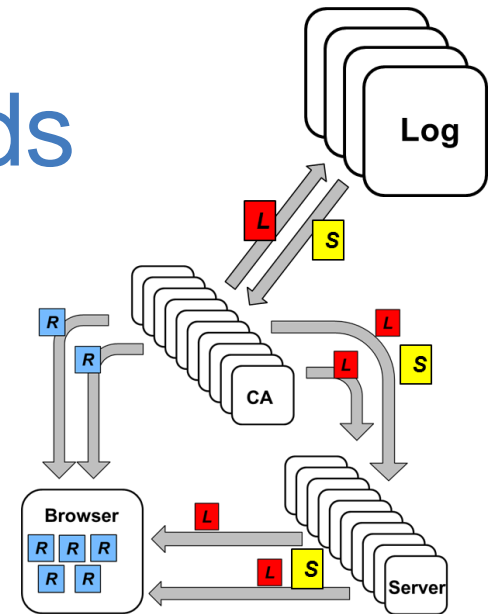


(b) TLS extension

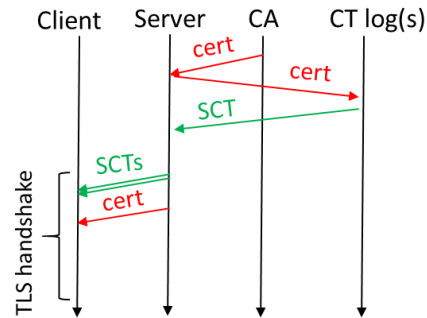


(c) OCSP stapling

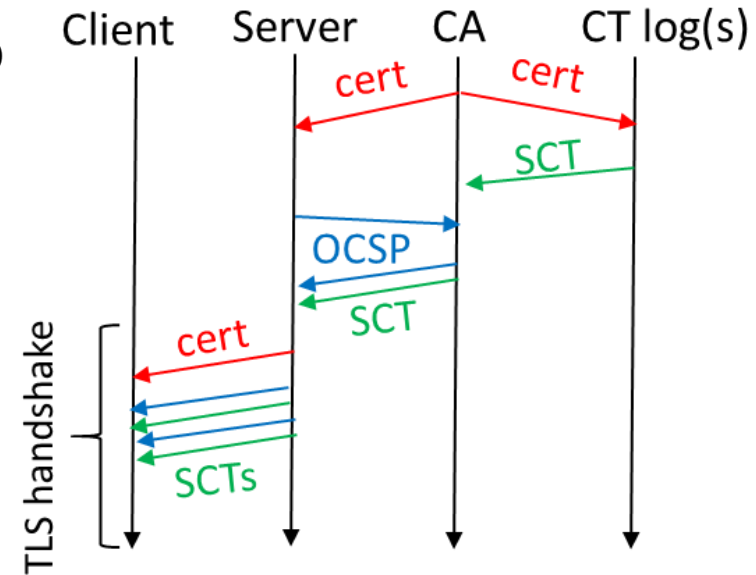
# Three SCT delivery methods



(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*



Your connection is not private

Attackers might be trying to steal your information from [choosemyreward.chase.com](#) (for example, passwords, messages, or credit cards). NET::ERR\_CERTIFICATE\_TRANSPARENCY\_REQUIRED

# 3. Mixed Content: HTTP and HTTPS

Page loads over HTTPS, but contains content over HTTP

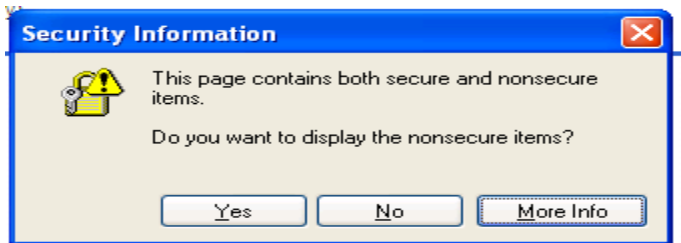
(e.g. `<script src="http://.../script.js">` )

 *never write this*

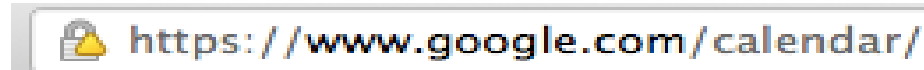
⇒ Active network attacker can hijack session

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>`

 Secure | https://mixed-script.badssl.com



*(script is blocked, click to load)*

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

 https://mixed.badssl.com

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)

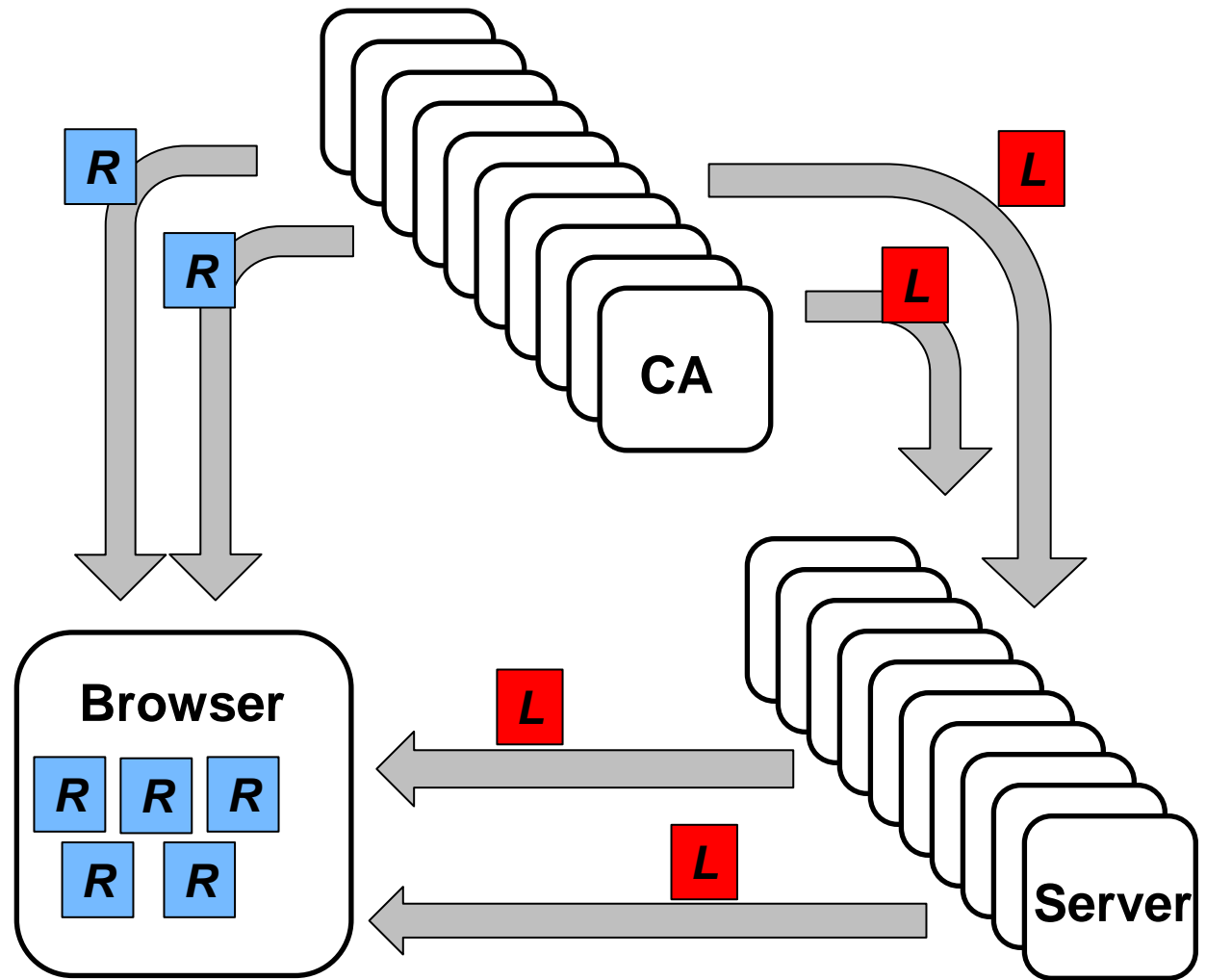
# Credits and some more slides

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

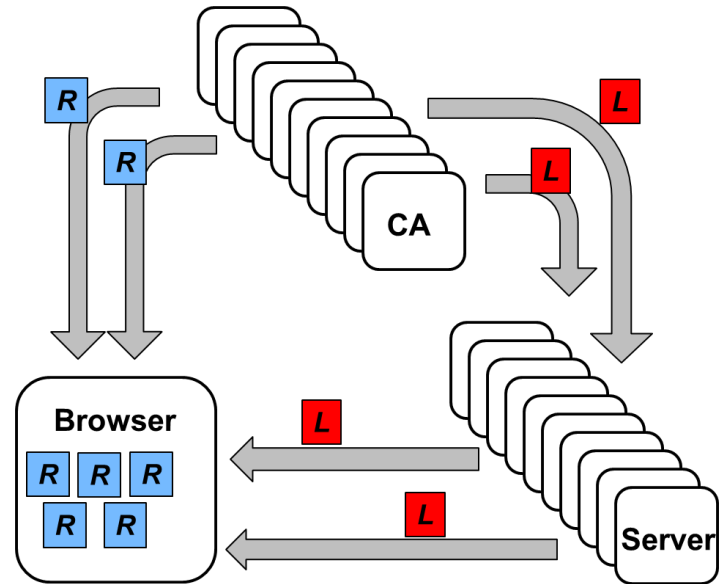
- *Nikita Korzhitskii and Niklas Carlsson, Characterizing the Root Landscape of Certificate Transparency Logs, Proc. IFIP Networking, Paris, France, June 2020, pp. 190--198.*
- *Carl Nykvist, Linus Sjöstrom, Josef Gustafsson, and Niklas Carlsson, Server-side Adoption of Certificate Transparency, Proc. 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.*



# Certification Transparency (CT)

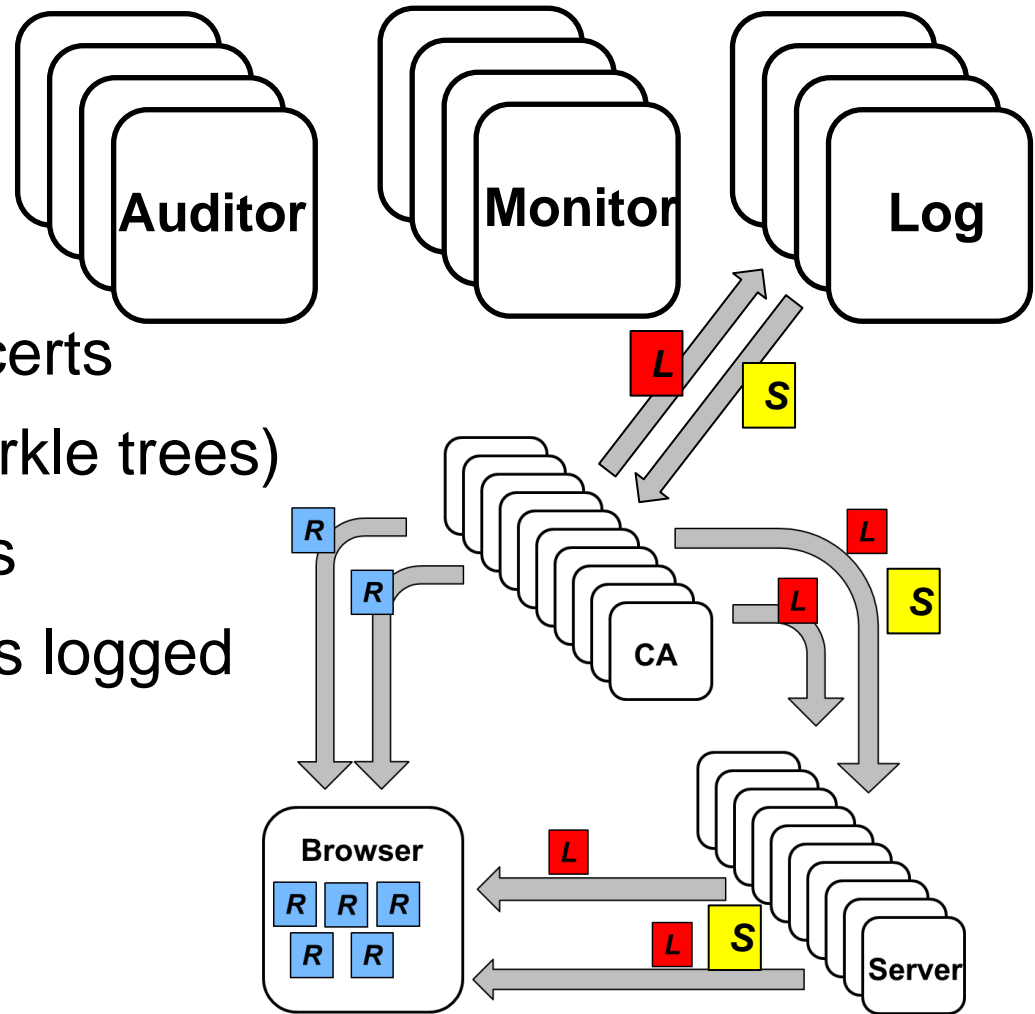


# Certification Transparency (CT)



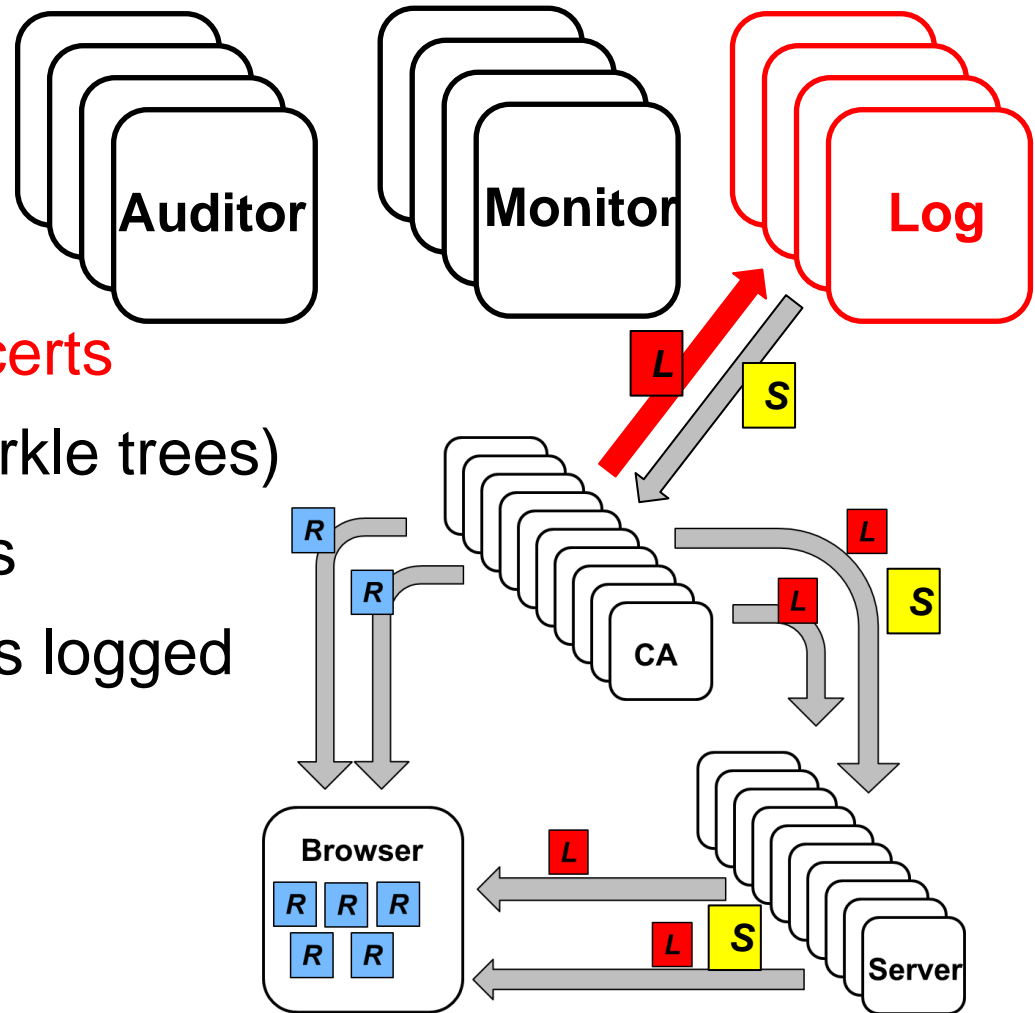
# Certification Transparency (CT)

- 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



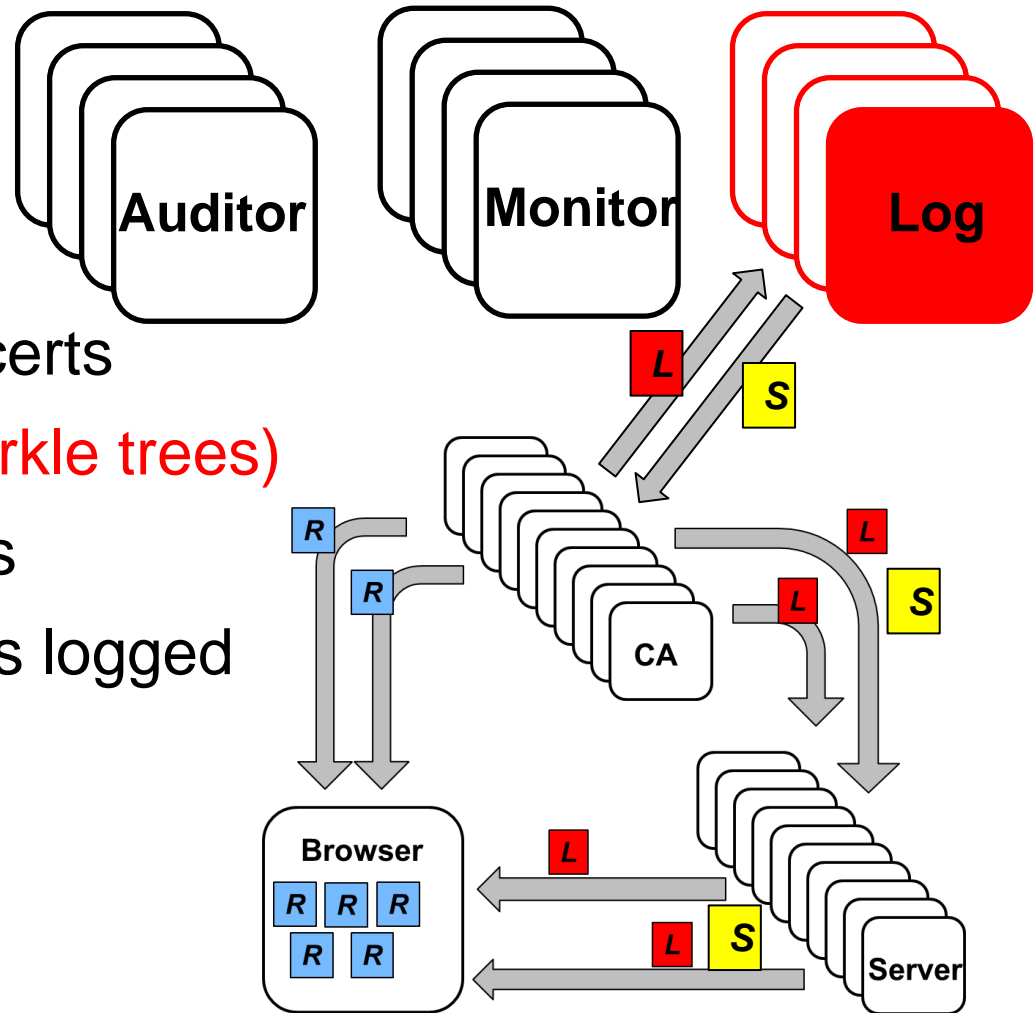
# Certification Transparency (CT)

- **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



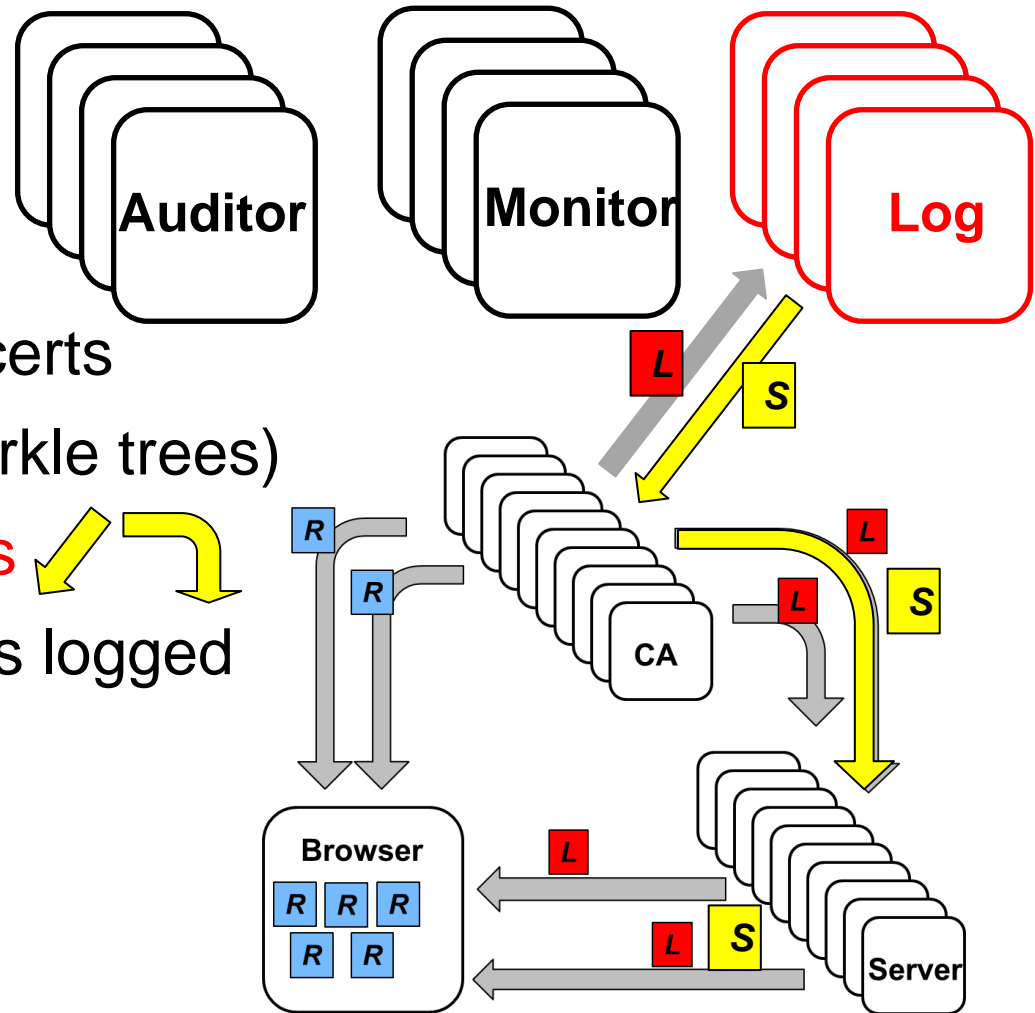
# Certification Transparency (CT)

- 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



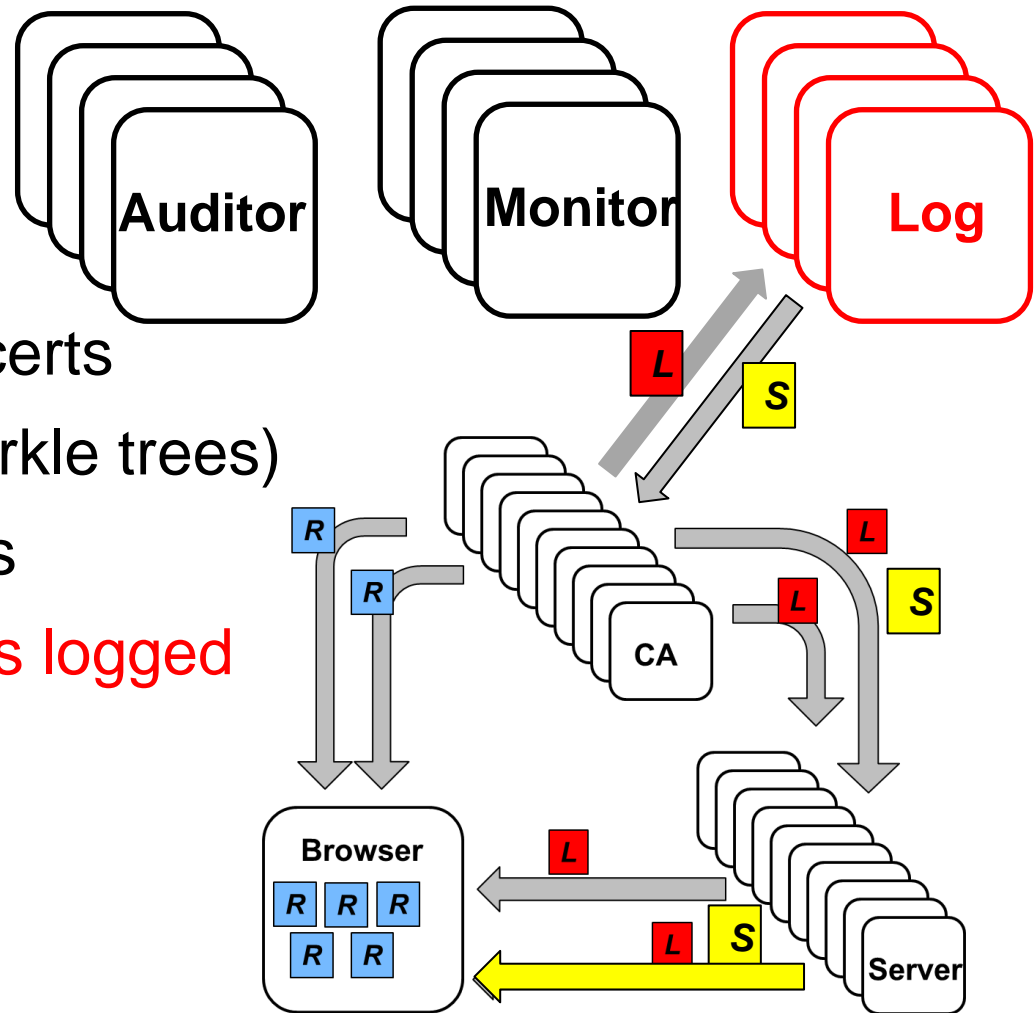
# Certification Transparency (CT)

- 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



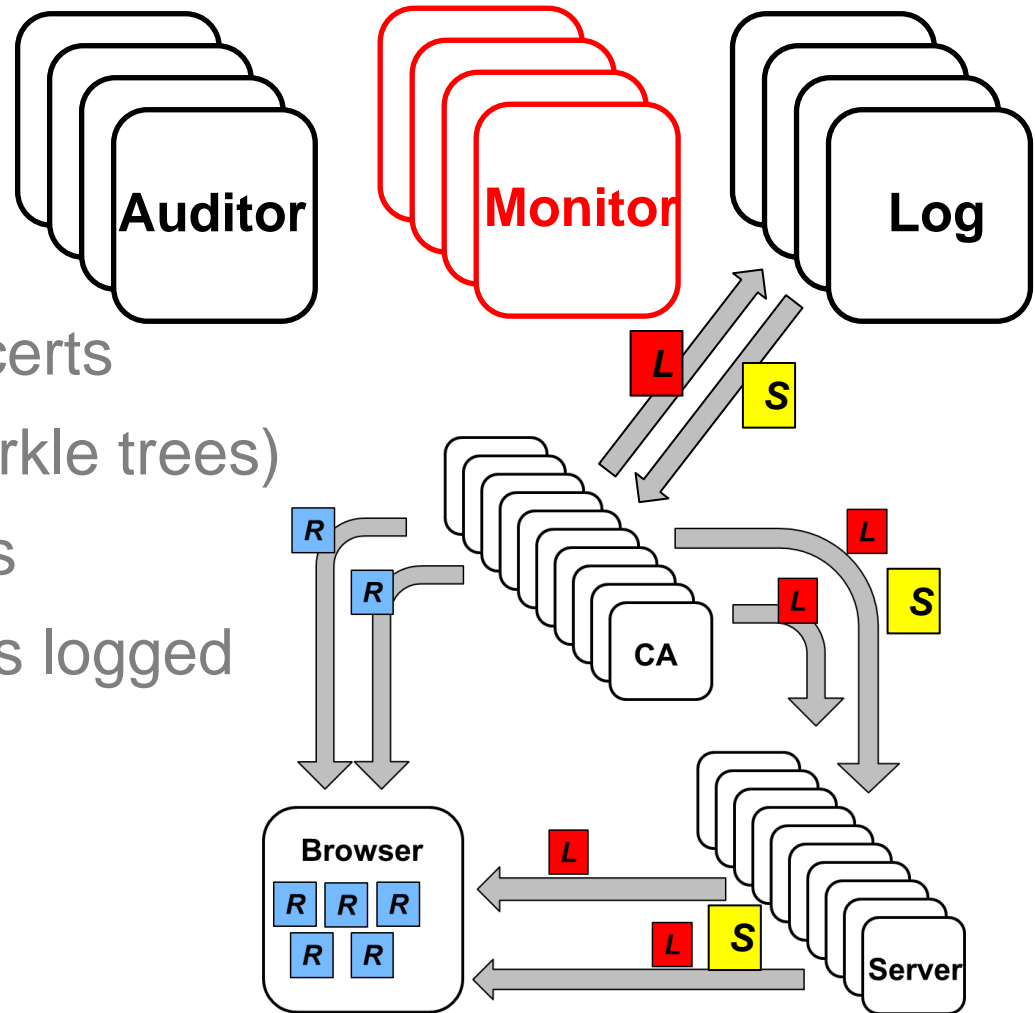
# Certification Transparency (CT)

- 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**

