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

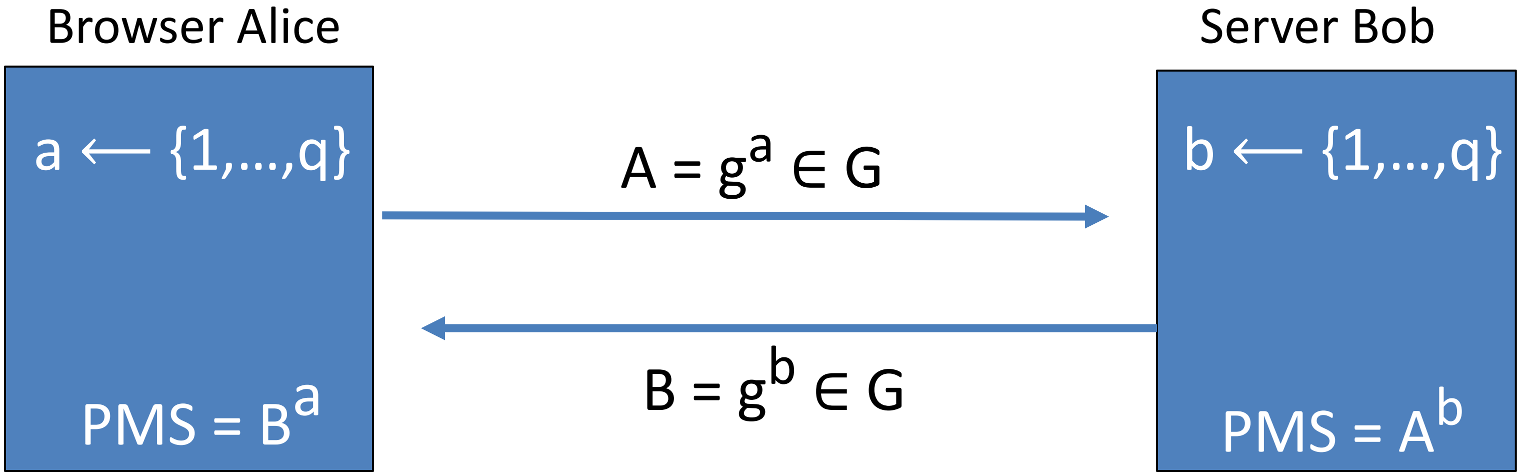
Examples:

- 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 $G = \{1, g, g^2, g^3, \ldots, g^{q-1}\}$



Browser Alice

$a \longleftarrow \{1,\ldots,q\}$

$PMS = B^a$

$A = g^a \in G$

$B = g^b \in G$

Server Bob

$b \longleftarrow \{1,\ldots,q\}$

$PMS = A^b$

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

(2) 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 contents of document

Def: a signature scheme is a tuple of three algorithms:

- **Gen()** → (pk, sk)
- **Sign**(sk, msg) → *sig*
- **Verify**(pk, msg, sig) → ‘accept’ or ‘reject’

sign msg using sk

verify (msg,sig) using pk

(2) Digital signatures

Security (informal): adversary who sees signatures on many messages of its choice, cannot forge a signature on a new message.

Def: a signature scheme is a tuple of three algorithms:

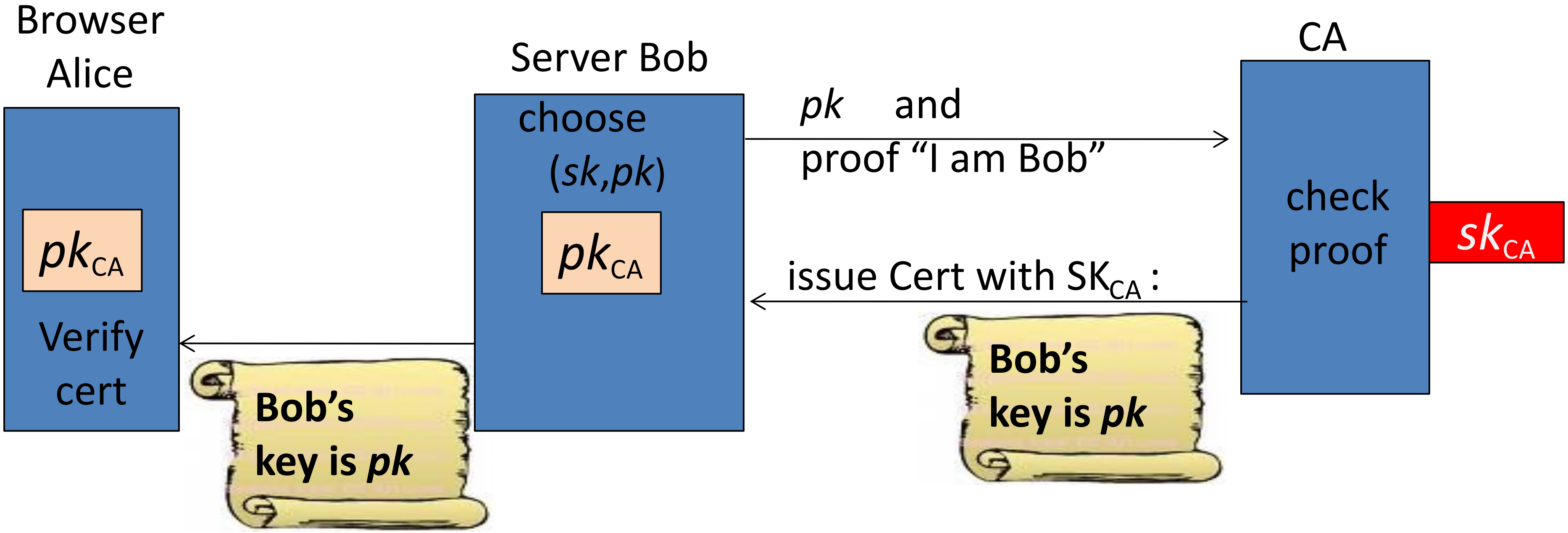
- **Gen()** → (pk, sk)
- **Sign**(sk, msg) → *sig*
- **Verify**(pk, msg, sig) → 'accept' or 'reject'

sign msg using sk

verify (msg,sig) using pk

(3) Certificates

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



Browser Alice

pk_{CA}

Verify cert

Server Bob

choose (sk,pk)

pk_{CA}

pk and proof “I am Bob”

CA

check proof

sk_{CA}

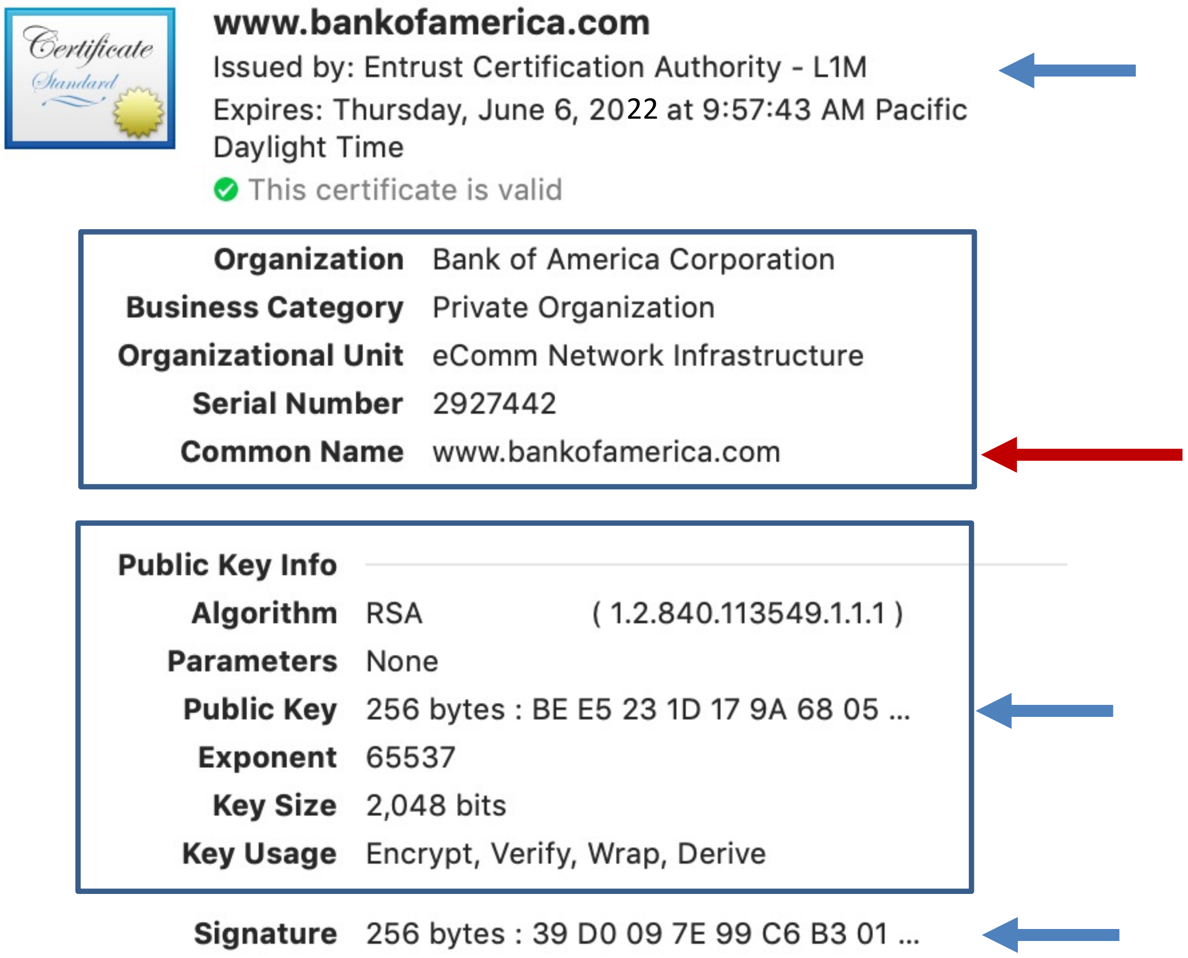
issue Cert with SK_{CA} :

Bob’s key is *pk*

Bob’s key is *pk*

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

Sample certificate:



www.bankofamerica.com

Issued by: Entrust Certification Authority - L1M

Expires: Thursday, June 6, 2022 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	(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	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")

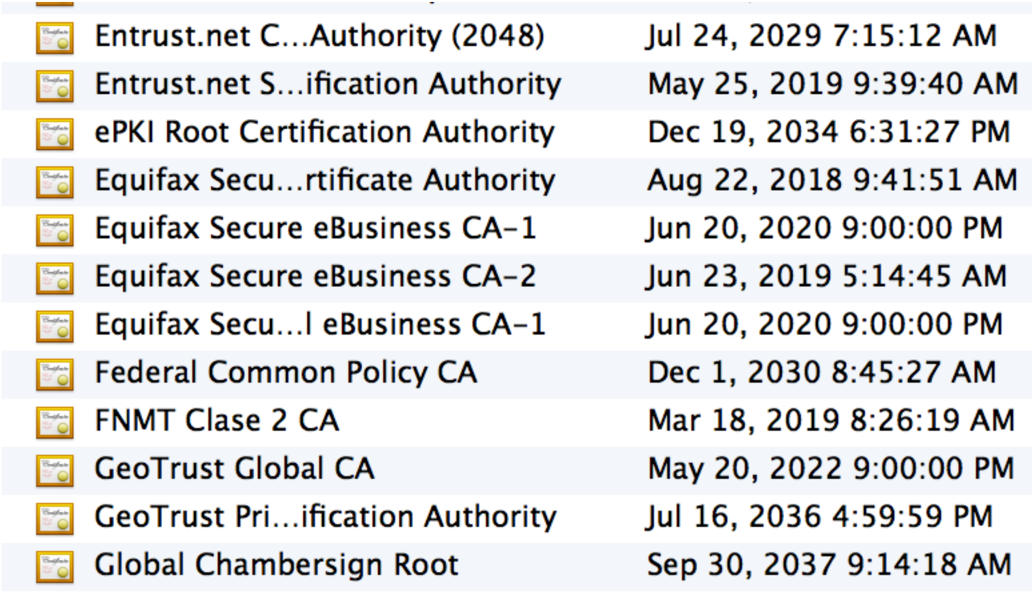
Certificate Authorities

Browsers accept certificates from a large number of CAs

Top level CAs ≈ 60

Intermediate CAs ≈ 1200

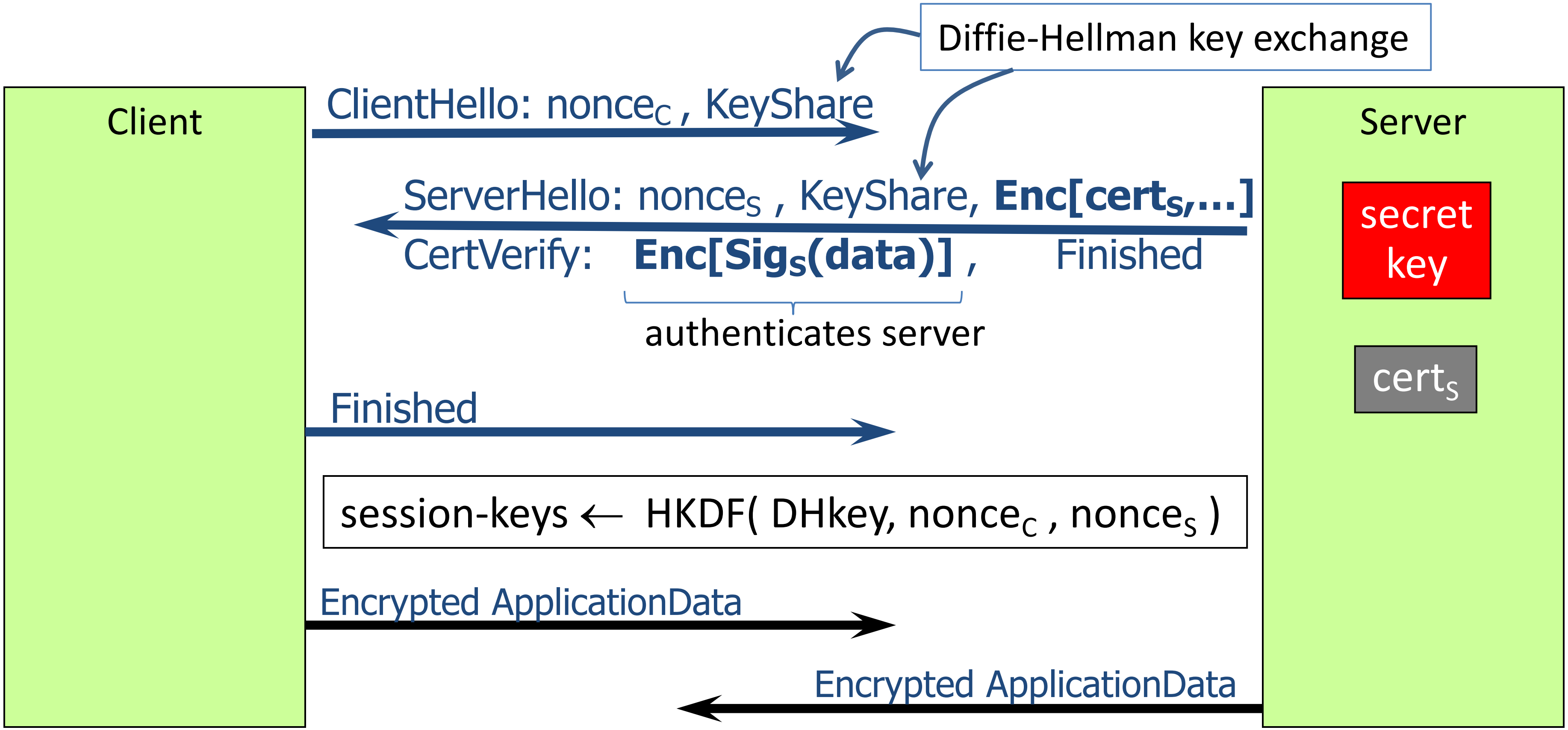
⋮



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

⋮

(4) TLS 1.3 session setup (simplified)



Diffie-Hellman key exchange

Client

Server

secret key

cert_S

ClientHello: nonce_C , KeyShare

ServerHello: nonce_S , KeyShare, $\textbf{Enc[cert}_S\textbf{,...]}$

CertVerify: $\underbrace{\textbf{Enc[Sig}_S\textbf{(data)]}}_{\text{authenticates server}}$, Finished

Finished

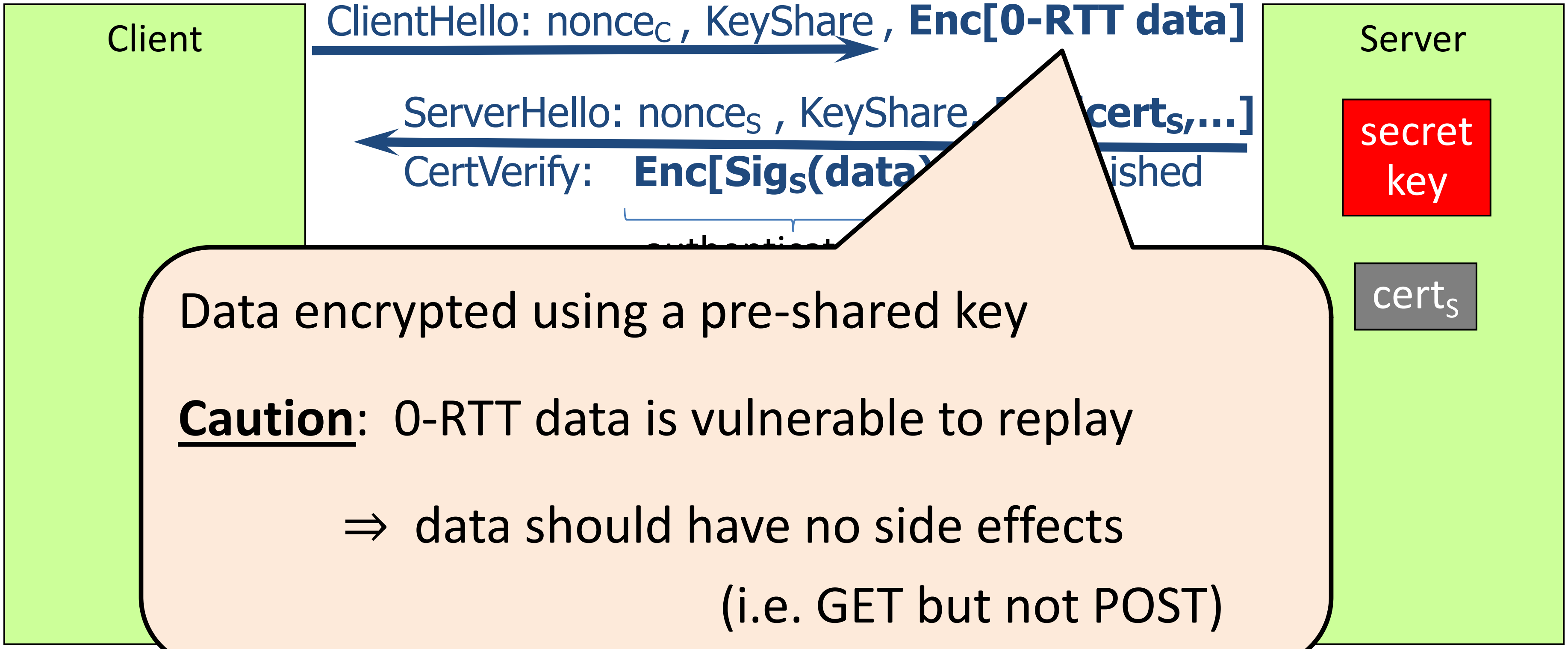
session-keys ← HKDF(DHkey, nonce_C , nonce_S)

Encrypted ApplicationData

Encrypted ApplicationData

Most common: server authentication only

(3) TLS 1.3 session setup: optimization (and caution)



Client
ClientHello: nonce_C , KeyShare , **Enc[0-RTT data]**
Server
secret key
certs_S
ServerHello: nonce_S , KeyShare, E **certs_S,…]**
CertVerify: **Enc[Sig_S(data)** ished
authenticat

Data encrypted using a pre-shared key

Caution: 0-RTT data is vulnerable to replay

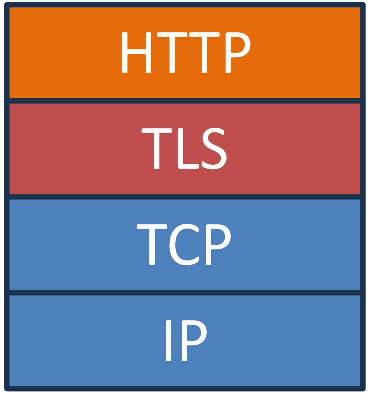
⇒ data should have no side effects

(i.e. GET but not POST)

Most common: server authentication only

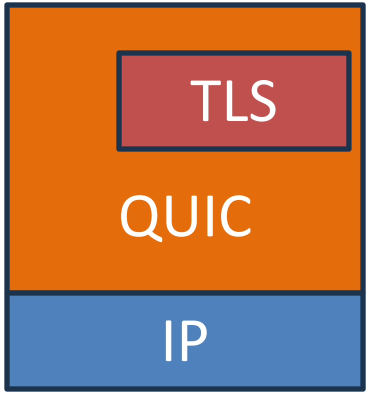
Integrating TLS with HTTP: HTTPS

HTTP/2
(2015)



TCP handshake ⇒ TLS handshake

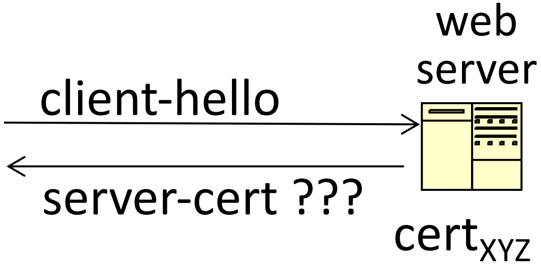
HTTP/3
(2022)



integrated handshake
(encrypting by default)

Integrating TLS with HTTP: HTTPS

A complication:



Virtual hosting: many sites hosted at same IP address

solution since TLS 1.1: SNI (2003)

client_hello_extension: **server_name=cnn.com**

… but SNI defeats privacy benefit of encrypted cert in TLS 1.3.

Solution: **enc. client hello (ECH)** [encrypted with pk in server DNS]

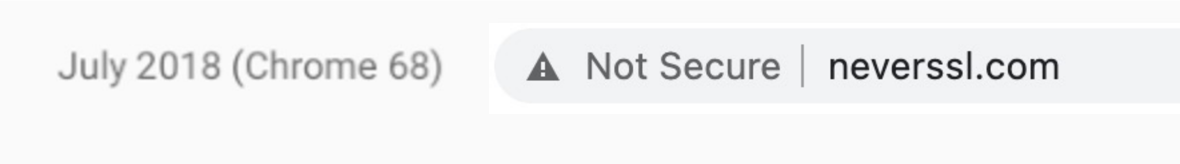
HTTPS for all web traffic?

Old excuse:

- Crypto slows down web servers

⟹ no longer true (thanks to AES-NI)

Since July 2018: Chrome marks HTTP sites as insecure



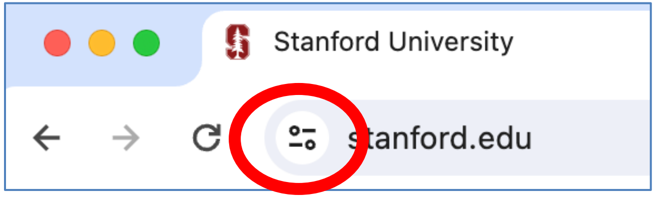
July 2018 (Chrome 68)

⚠ Not Secure | neverssl.com

HTTPS in the Browser

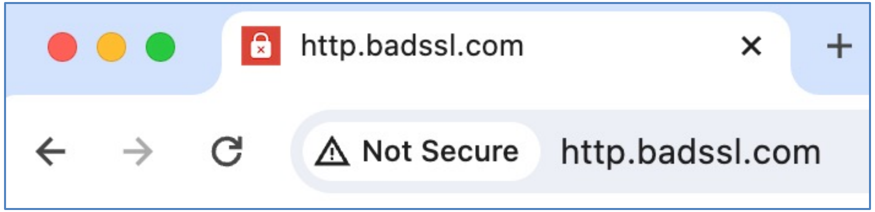
The TLS indicator

(HTTPS site)



vs.

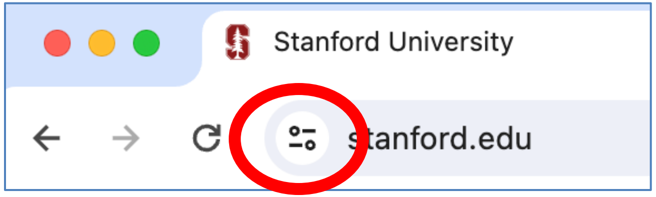
(HTTP site)



Intended goal:

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

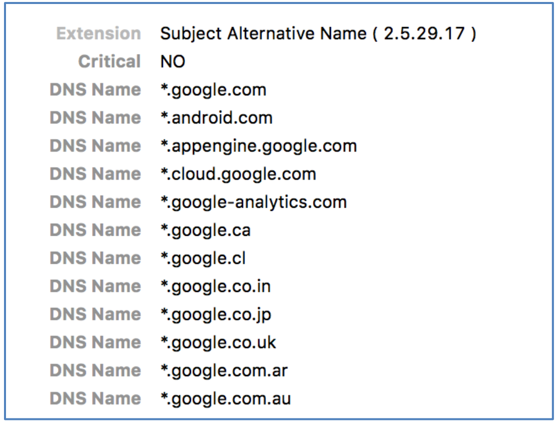
When is the TLS icon displayed



All elements on the page fetched using HTTPS

For all elements:

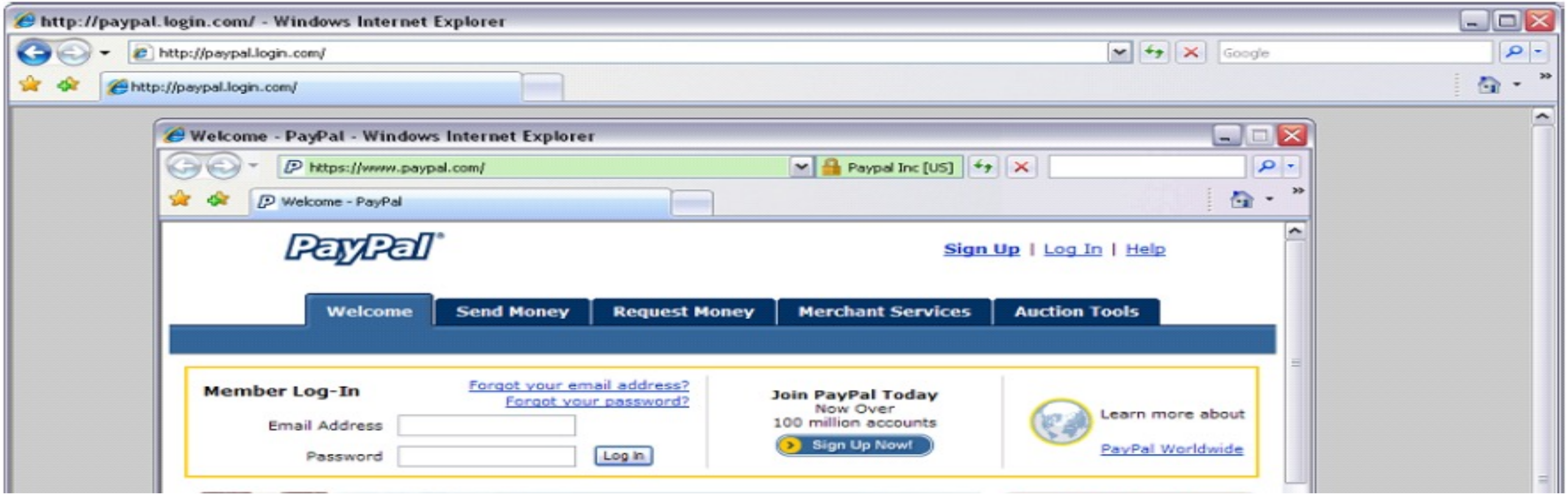
- HTTPS cert issued by a CA trusted by browser
- HTTPS cert is valid (e.g. not expired)
- Domain in URL matches:
  CommonName or **SubjectAlternativeName** in cert



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

Positive security indicators are dangerous

The lock icon is a **positive security indicator**. Problem: picture-in-picture attacks.



http://paypal.login.com/ - Windows Internet Explorer

http://paypal.login.com/

Welcome - PayPal - Windows Internet Explorer

https://www.paypal.com/

Paypal Inc [US]

Welcome - PayPal

Member Log-In

Email Address

Password

Join PayPal Today
Now Over
100 million accounts

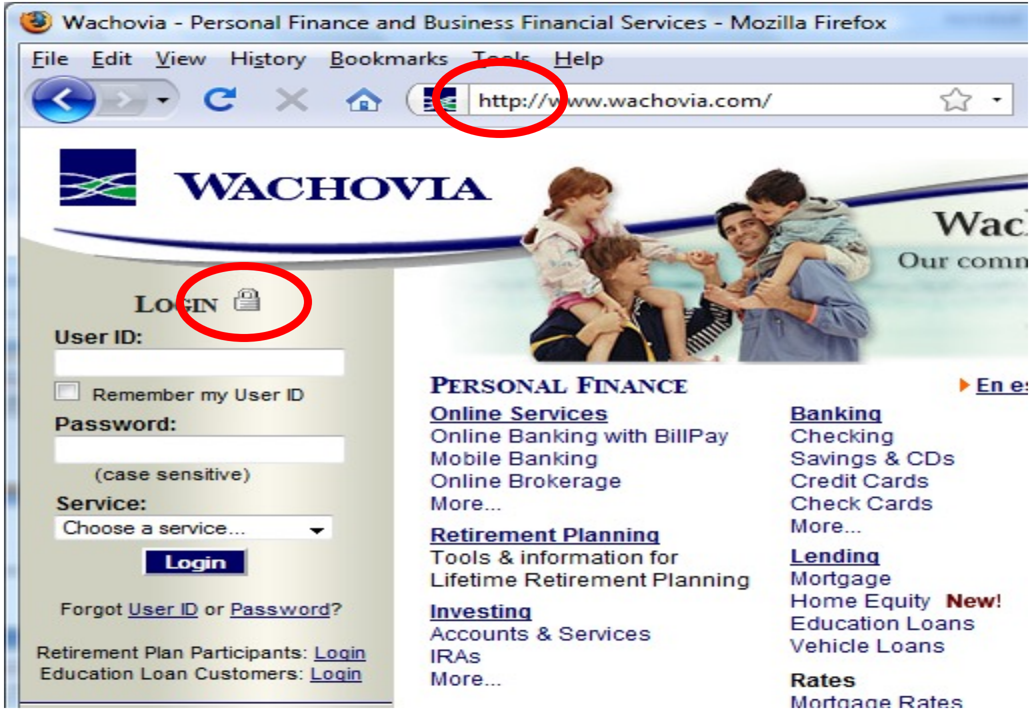
Learn more about PayPal Worldwide

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

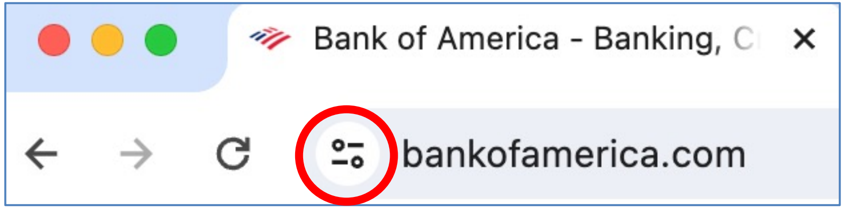
General guideline:

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

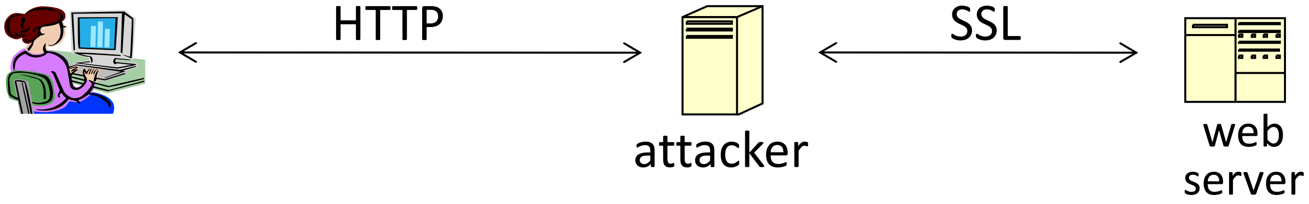
4. Does HTTPS hide web traffic?
   - Problems: traffic analysis, compression attacks

1. HTTP ⇒ HTTPS upgrade

Suppose user does:

- connect to bank site over HTTP;   bank redirects to HTTPS

SSL_strip attack: prevent the upgrade [Moxie'08]



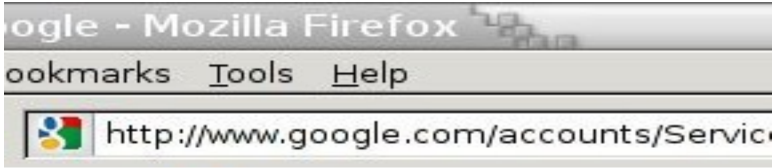
<a href=https://…>   ⟶   <a href=http://…>

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

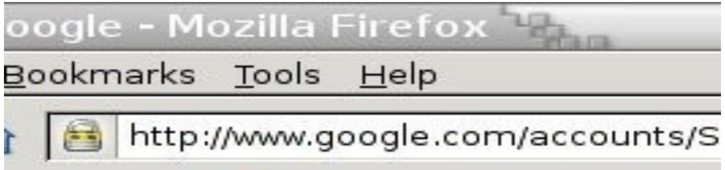
<form action=https://… >   ⟶   <form action=http://…>

Tricks and Details

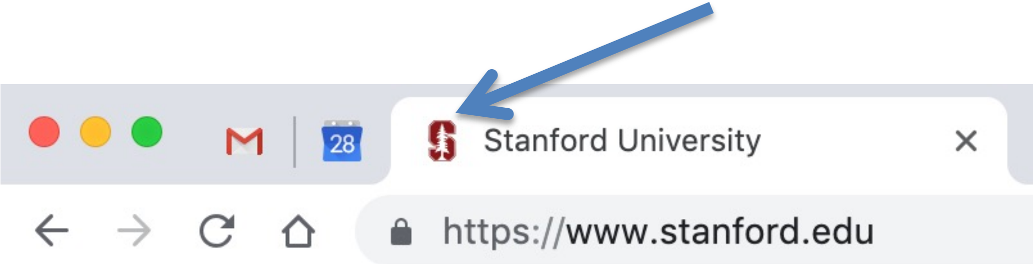
UI design flaw in old browsers: location of fav icon



→

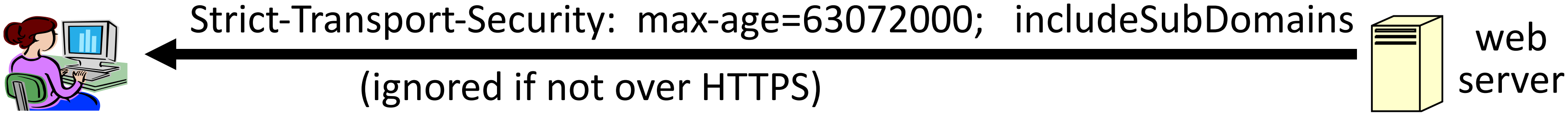


⇒ fav icon no longer presented in address bar



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)
web server

Header tells browser to always connect over HTTPS

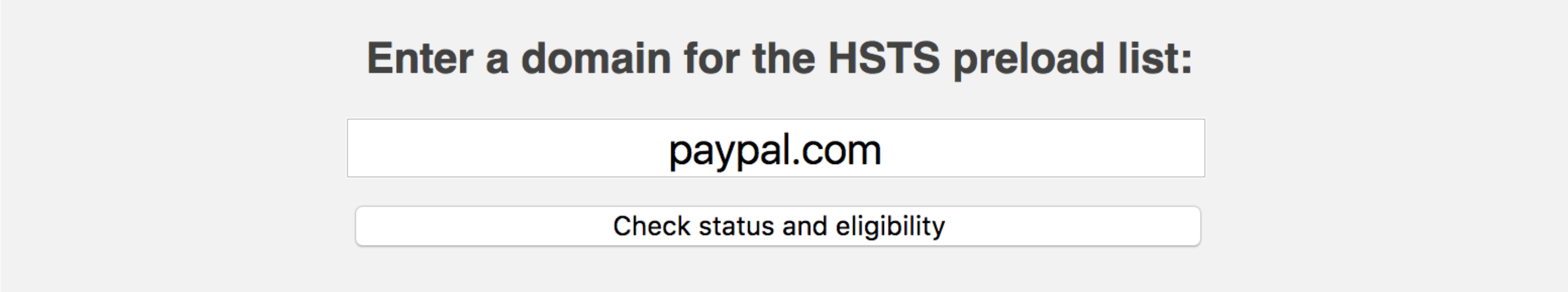
Subsequent visits must be over HTTPS (self signed certs result in an error)

- 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

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

Preload list hard-coded in Chrome source code. Examples:

Google, Paypal, Twitter, Simple, Linode, Stripe, Lastpass, …

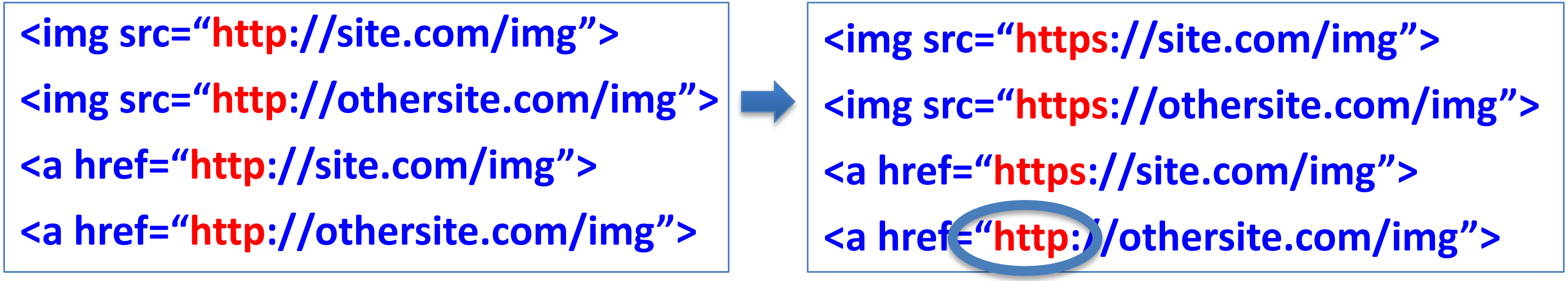
CSP: upgrade-insecure-requests

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

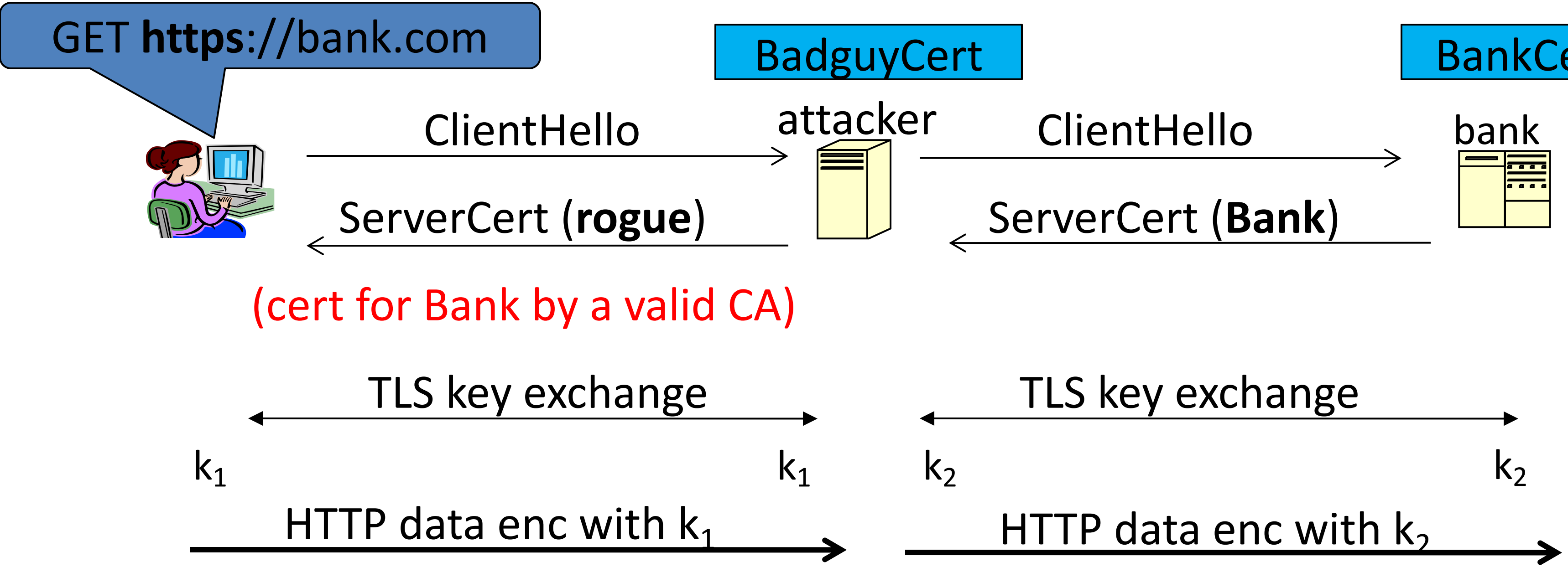
(2) Chrome restricts India CCA root to only seven Indian domains

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



GET **https**://bank.com

BadguyCert

BankCert

ClientHello

attacker

ClientHello

bank

ServerCert (**rogue**)

ServerCert (**Bank**)

(cert for Bank by a valid CA)

TLS key exchange

TLS key exchange

k_1 k_1 k_2 k_2

HTTP data enc with k_1

HTTP data enc with k_2

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

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

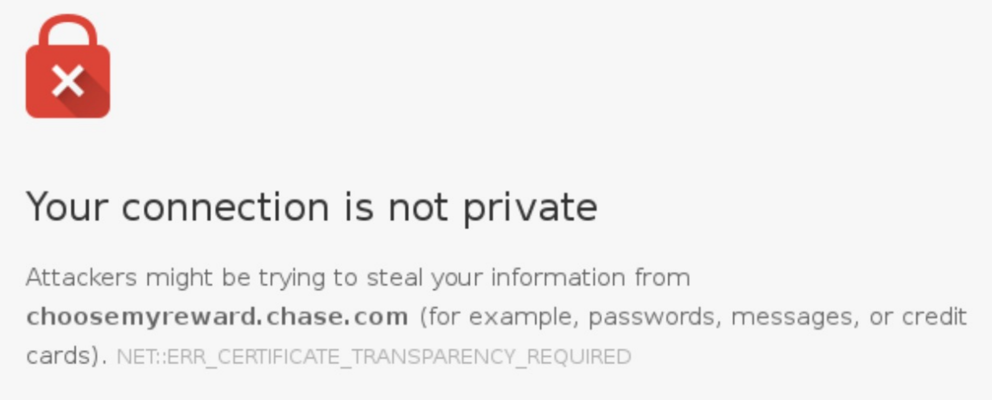
CT requirements

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



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

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

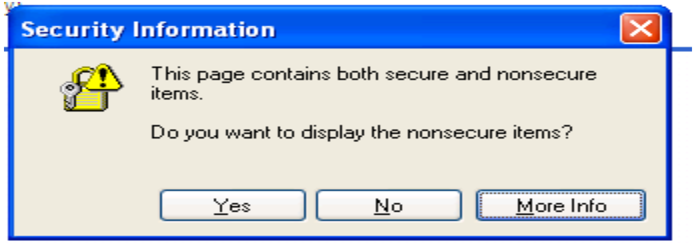
(e.g. <script src=“http://.../script.js>)

never write this

⇒ Active network attacker can hijack session

by modifying script en-route to browser

IE7:

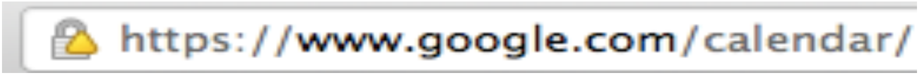


Security Information

This page contains both secure and nonsecure items.

Do you want to display the nonsecure items?

Old Chrome:

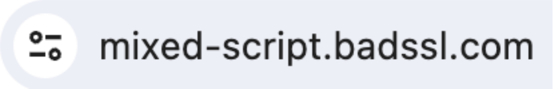


https://www.google.com/calendar/

Mostly ignored by users …

https://badssl.com (Chrome 124, 2024)

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

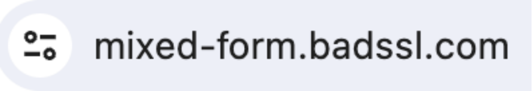


mixed-script.badssl.com

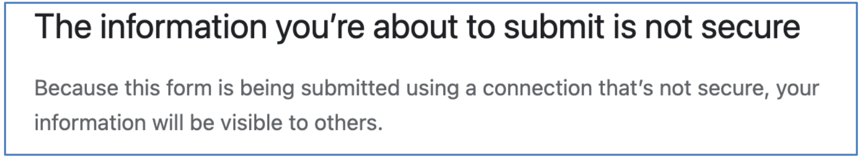
(no visible warning)

script is not loaded! developer tools show an error.

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



mixed-form.badssl.com



The information you’re about to submit is not secure

Because this form is being submitted using a connection that’s not secure, your information will be visible to others.

Warning if user tries to submit data

4. Peeking through TLS: traffic analysis

- Network traffic reveals length of HTTPS packets
  - TLS supports up to 256 bytes of padding

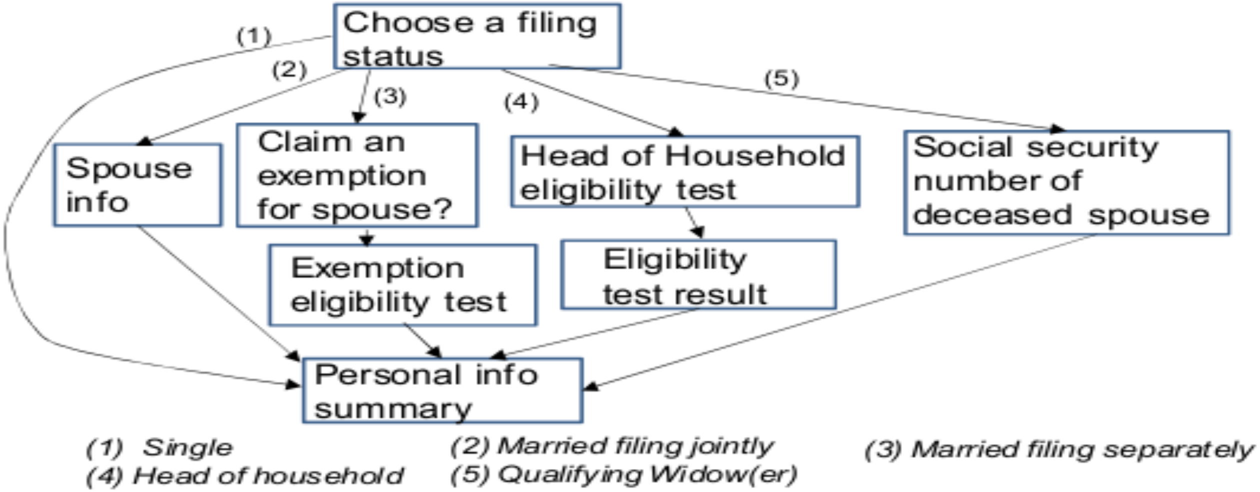
- Some sites interact frequently with the web server
  - These interactions expose specific internal state of the page

BAM!

Chen, Wang, Wang, Zhang, 2010 (tax web site)

IOactive, 2012 (Google maps)

Peeking through TLS: an example [CWWZ'10]


Choose a filing status
(1)
(2)
(3)
(4)
(5)
Spouse info
Claim an exemption for spouse?
Head of Household eligibility test
Social security number of deceased spouse
Exemption eligibility test
Eligibility test result
Personal info summary
(1) Single
(2) Married filing jointly
(3) Married filing separately
(4) Head of household
(5) Qualifying Widow(er)

Vulnerabilities in an online tax application

No easy fix. Can also be used to ID Tor traffic

THE END