



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

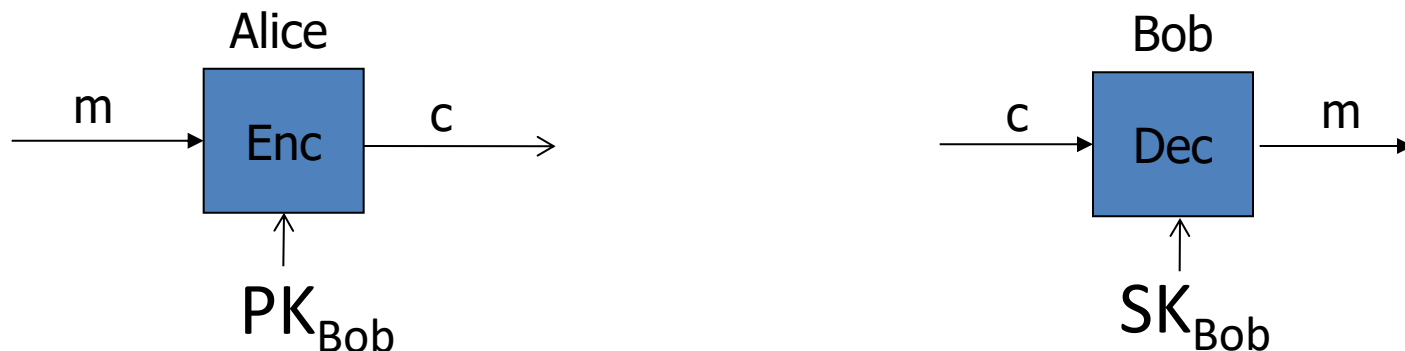


## Examples:

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

# TLS overview

## Public-key encryption:

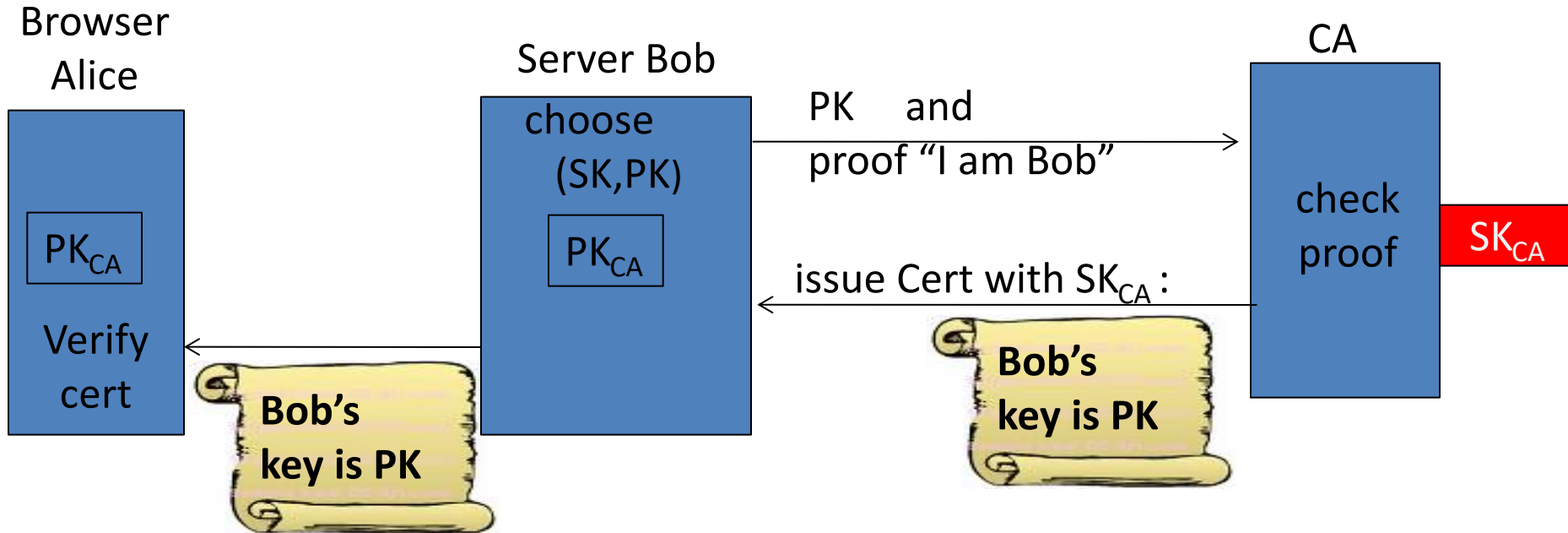


- Bob generates  $(SK_{Bob}, PK_{Bob})$
- Alice: using  $PK_{Bob}$  encrypts messages and only Bob can decrypt



# Certificates

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



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



## mail.google.com

Issued by: Google Internet Authority G3

Expires: Wednesday, June 20, 2018 at 6:25:00 AM Pacific Daylight Time

✓ This certificate is valid

### ▼ Details

<b>Subject Name</b>	
<b>Country</b>	US
<b>State/Province</b>	California
<b>Locality</b>	Mountain View
<b>Organization</b>	Google Inc
<b>Common Name</b>	mail.google.com

<b>Issuer Name</b>	
<b>Country</b>	US
<b>Organization</b>	Google Trust Services
<b>Common Name</b>	Google Internet Authority G3
<b>Serial Number</b>	3495829599616174946
<b>Version</b>	3
<b>Signature Algorithm</b>	SHA-256 with RSA Encryption

<b>Public Key Info</b>	
<b>Algorithm</b>	Elliptic Curve Public Key ( 1.2.840.10045.2.1 )
<b>Parameters</b>	Elliptic Curve secp256r1 ( 1.2.840.10045.3.1.7 )
<b>Public Key</b>	65 bytes : 04 D5 63 FC 4D F9 4E 91 ...
<b>Key Size</b>	256 bits
<b>Key Usage</b>	Encrypt, Verify, Derive
<b>Signature</b>	256 bytes : 3F FE 04 7B BE B0 32 1D ...

Sample certificate:

# 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  $\approx$  60

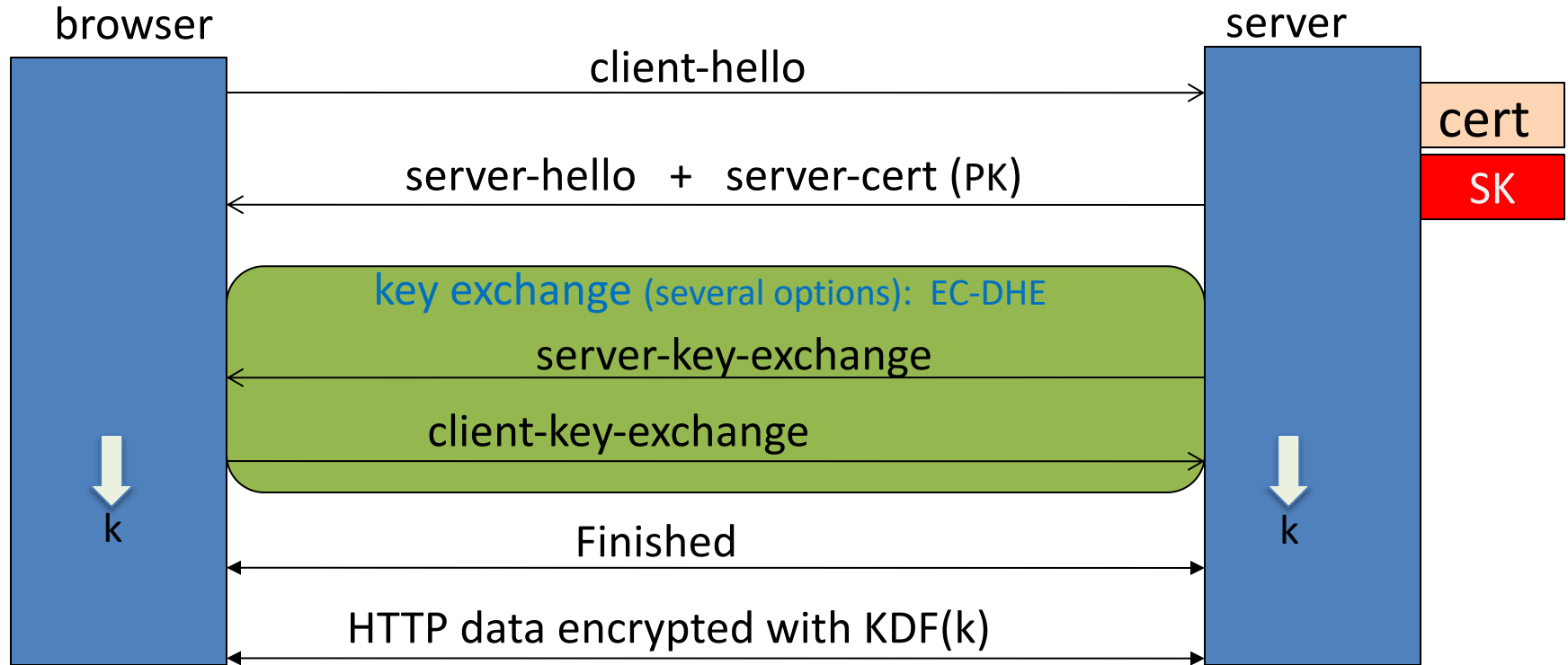
Intermediate CAs  $\approx$  1200

•  
•  
•

	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

•  
•  
•

# Brief overview of TLS 1.2



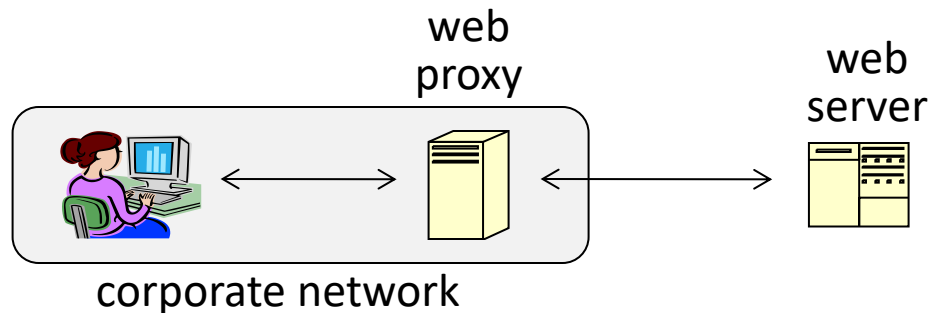
Most common: server authentication only

# Integrating SSL/TLS with HTTP: HTTPS

Two complications

## Web proxies

solution: browser sends  
**CONNECT domain-name**  
before client-hello



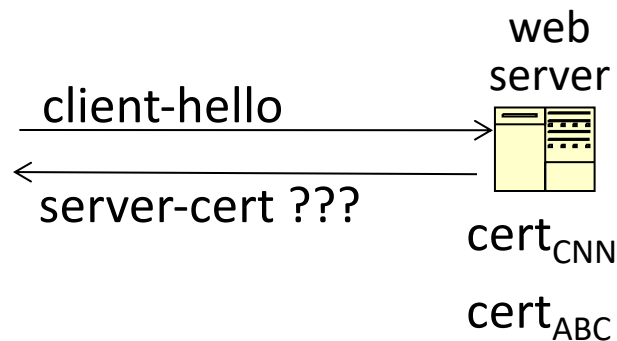
## Virtual hosting:

two sites hosted at same IP address.

solution in TLS 1.1: SNI (June 2003)

client\_hello\_extension: server\_name=cnn.com

implemented since FF2 and IE7 (vista)



# 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
- Incompatible with virtual hosting (older browsers)

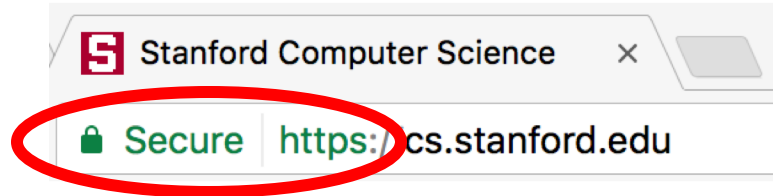
July 2018: Chrome will mark HTTP sites as insecure



# HTTPS in the Browser



# The lock icon: SSL indicator



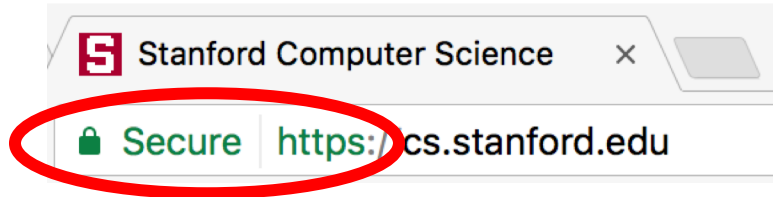
Intended goal:

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



In reality: many problems (next few slides)

# When is the (basic) lock icon displayed



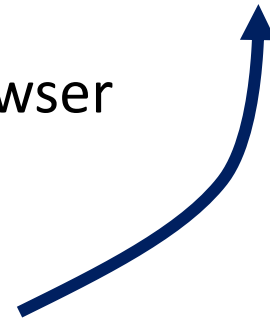
All elements on the page fetched using HTTPS

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

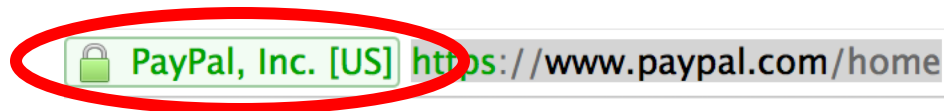


# The lock UI: Extended Validation Certs

Harder to obtain than regular certs

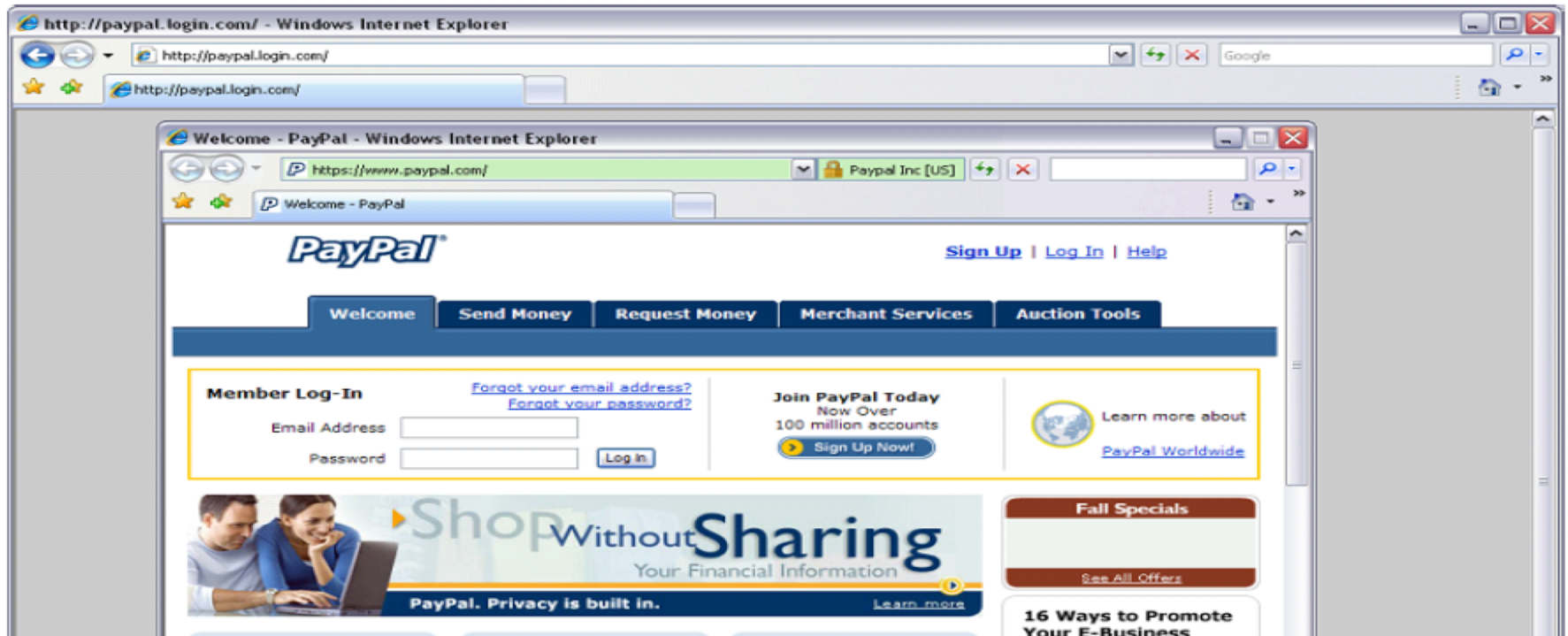
- requires human at CA to approve cert request
- no wildcard certs (e.g. \*.stanford.edu )

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



note: HTTPS-EV and HTTPS are in the same origin

# 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

Users often land on login page over HTTP:

- Type HTTP URL into address bar
- Google links to HTTP page

View source:

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

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



(old site)

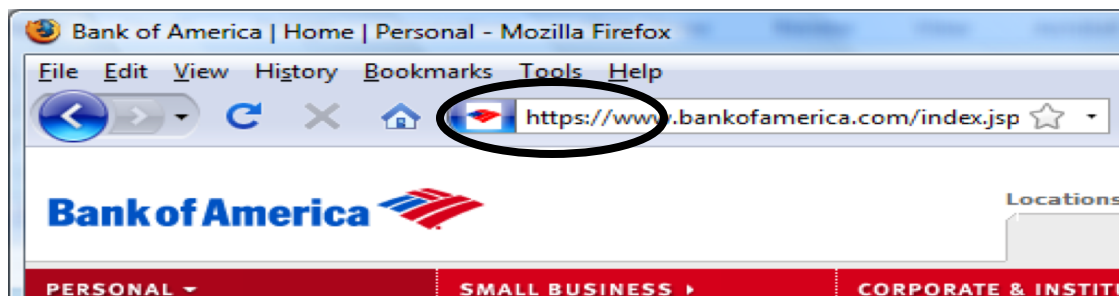
# HTTPS and login pages: guidelines

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

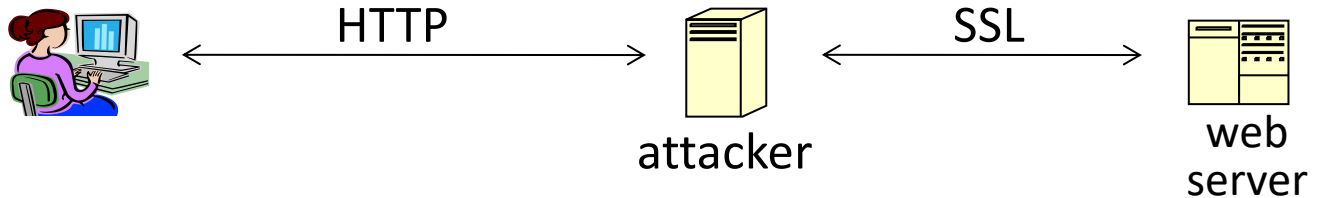


# 1. HTTP $\Rightarrow$ HTTPS upgrade

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: <code>https://...</code>	$\rightarrow$	Location: <code>http://...</code>	(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



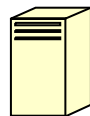
Number of users who detected HTTP downgrade: 0

# Defense: Strict Transport Security (HSTS)



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

(ignored if not over HTTPS)



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

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

Always use protocol relative URLs

```

```

## 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

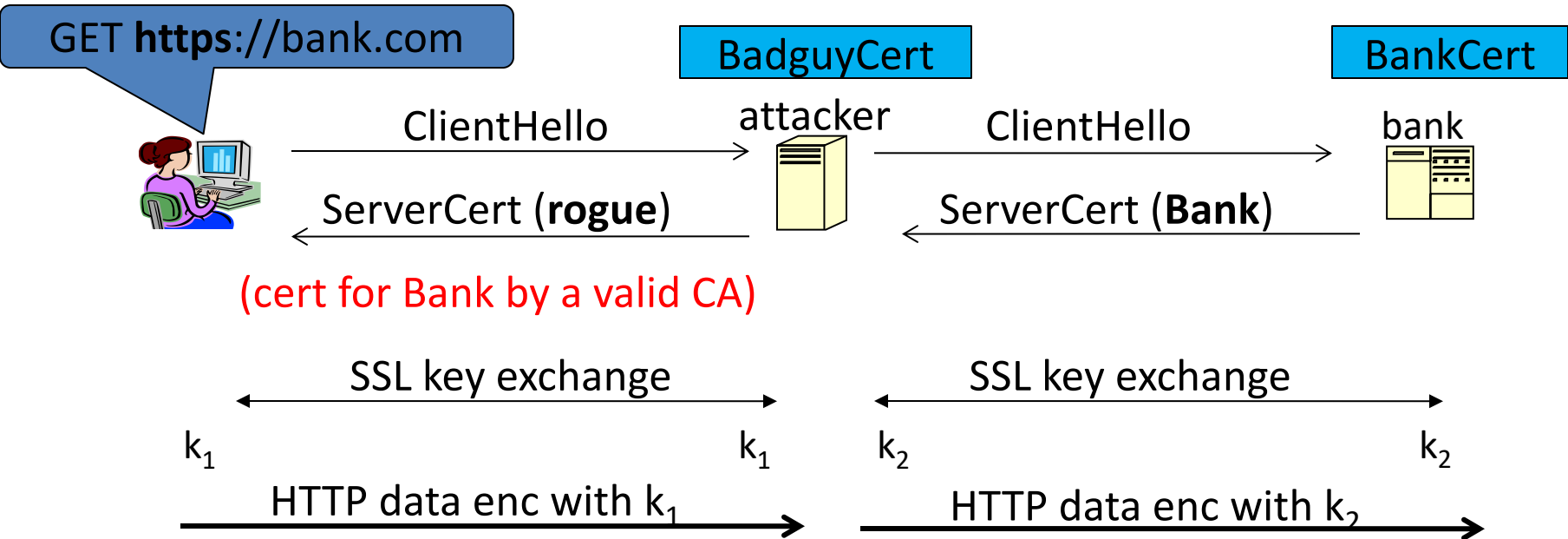
(2) Chrome restricts India CCA root to only seven Indian domains

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

## 1. 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



# 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

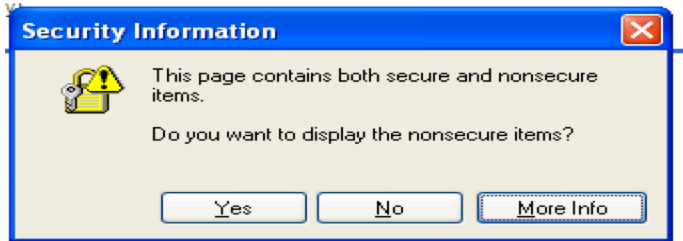
Page loads over HTTPS, but contains content over HTTP

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

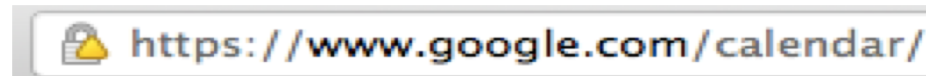
 never write this

⇒ Active network attacker can hijack session  
by modifying script en-route to browser

IE7:



Old Chrome:



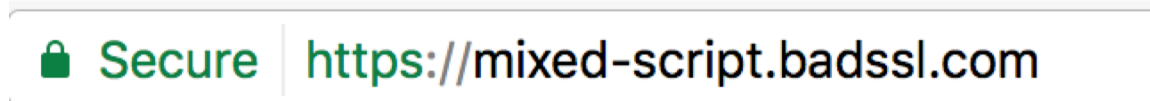
Mostly ignored by users ...

# https://badssl.com

(Chrome 65, 2018)

Mixed script:

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



(script is blocked, click to load)

---

Mixed image:

```

```

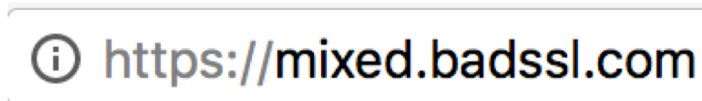


Image loaded, but no HTTPS indicator

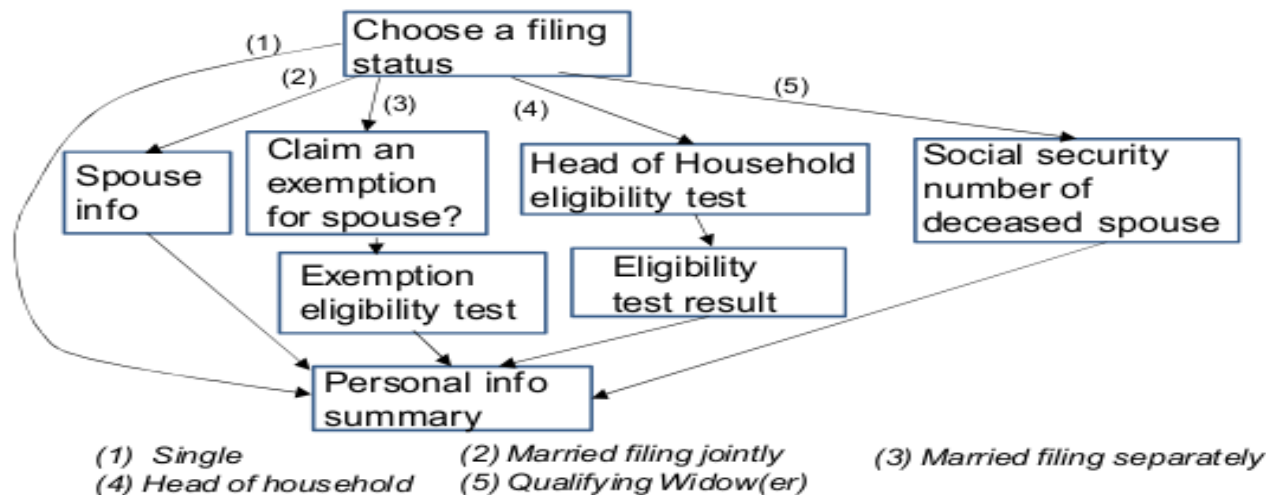
## 4. Peeking through SSL: traffic analysis

- Network traffic reveals length of HTTPS packets
  - TLS supports up to 256 bytes of padding
- AJAX-rich pages have lots and lots of interactions with the server
- These interactions expose specific internal state of the page



Chen, Wang, Wang, Zhang, 2010

# Peeking through SSL: an example [CWWZ'10]



Vulnerabilities in an online tax application

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

THE END