



# HTTPS and the Lock Icon

Dan Boneh

# 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

**Passive attacker:** only eavesdrops on net traffic

**Active attacker:** eavesdrops, injects, blocks, and modifies packets

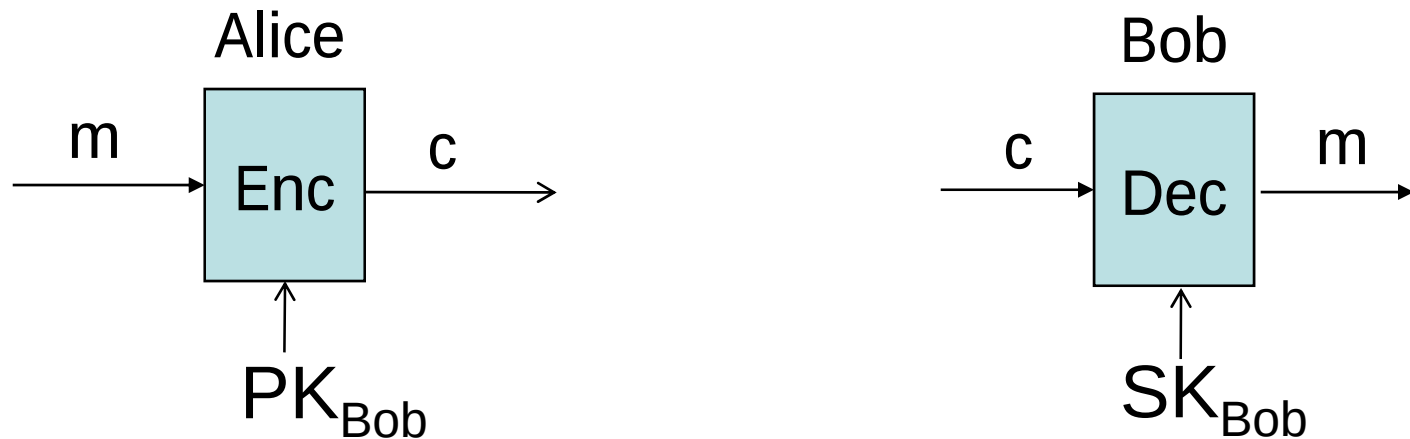
## Examples:

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

# SSL/TLS overview

---

## Public-key encryption:

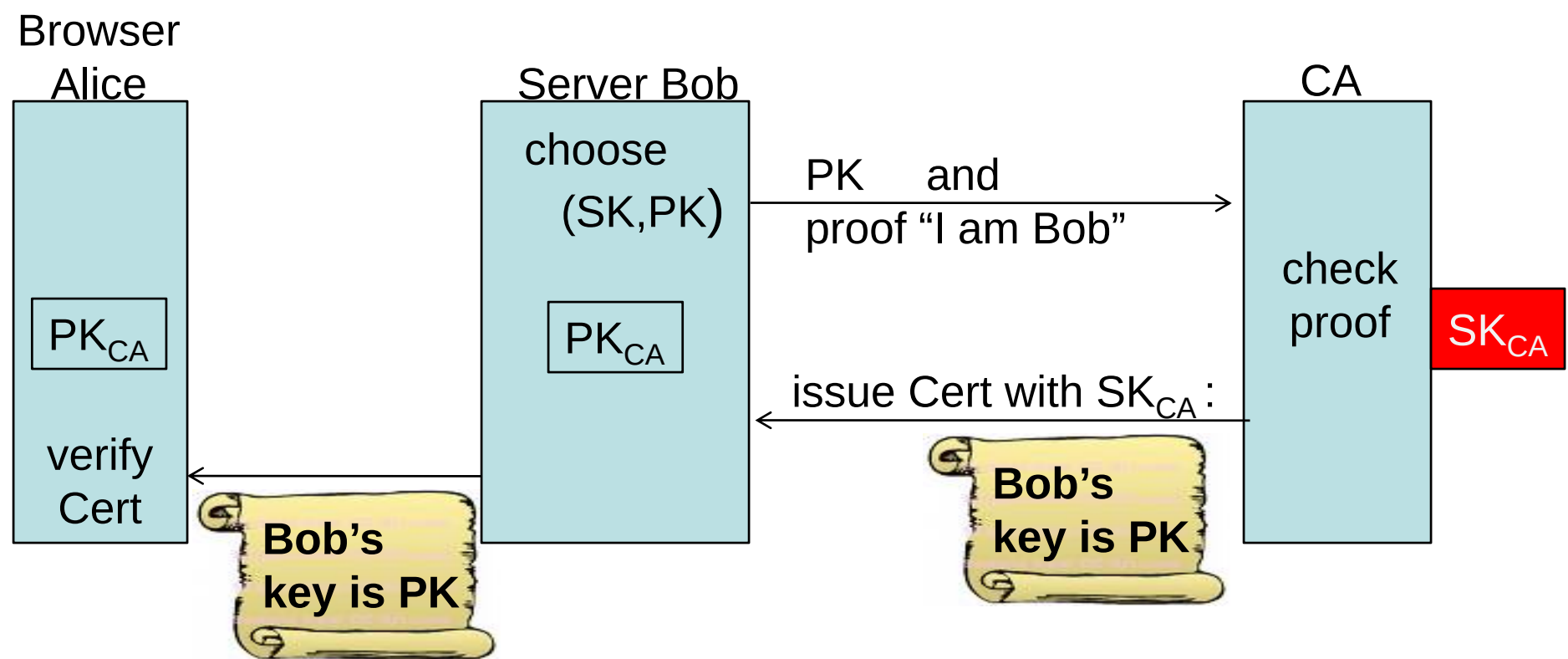


**Bob generates (SK<sub>Bob</sub> , PK<sub>Bob</sub> )**

**Alice: using PK<sub>Bob</sub> encrypts messages  
and only Bob can decrypt**

# Certificates

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



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

# Certificates: example

## Important fields:

Certificate Signature Algorithm

Issuer

▲ Validity

Not Before

Not After

Subject

▲ Subject Public Key Info

Subject Public Key Algorithm

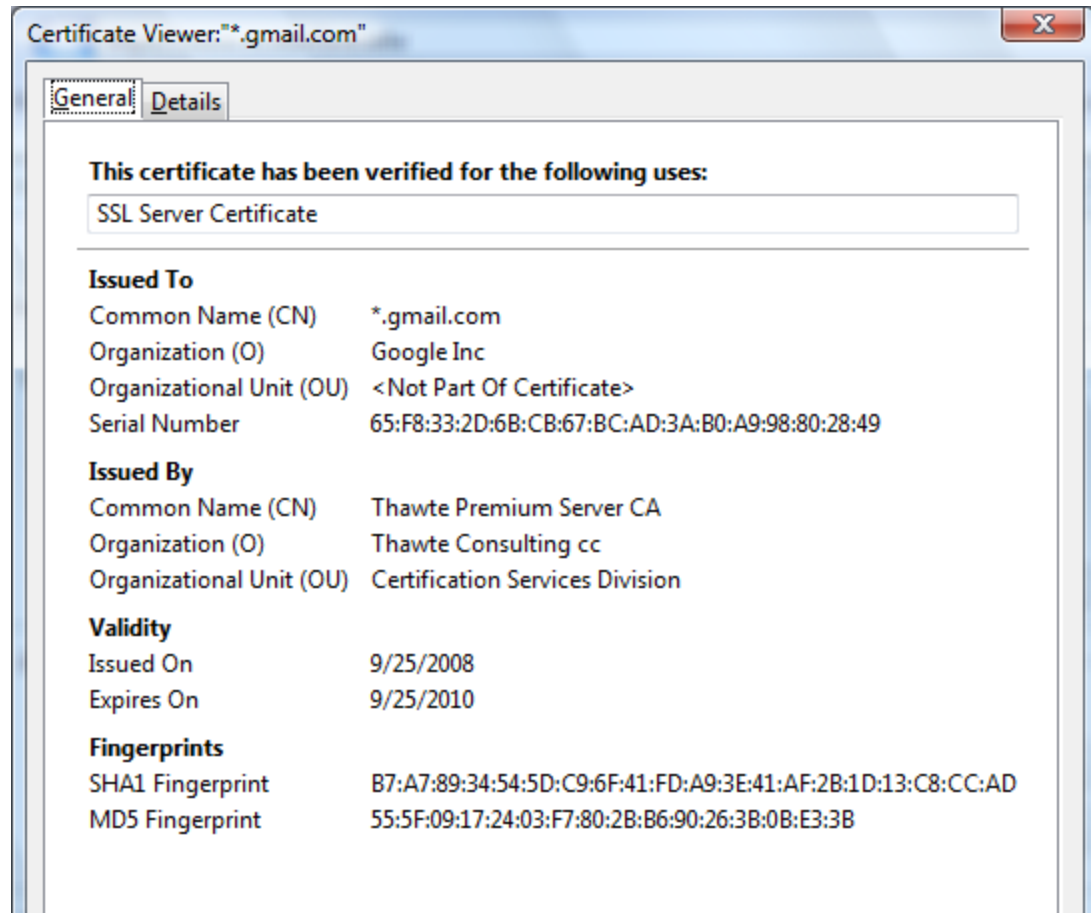
Subject's Public Key

▲ Extensions

### Field Value

Modulus (1024 bits):

```
ac 73 14 97 b4 10 a3 aa f4 c1 15 ed cf 92 f3 9a
97 26 9a cf 1b e4 1b dc d2 c9 37 2f d2 e6 07 1d
ad b2 3e f7 8c 2f fa a1 b7 9e e3 54 40 34 3f b9
e2 1c 12 8a 30 6b 0c fa 30 6a 01 61 e9 7c b1 98
2d 0d c6 38 03 b4 55 33 7f 10 40 45 c5 c3 e4 d6
6b 9c 0d d0 8e 4f 39 0d 2b d2 e9 88 cb 2d 21 a3
f1 84 61 3c 3a aa 80 18 27 e6 7e f7 b8 6a 0a 75
e1 bb 14 72 95 cb 64 78 06 84 81 eb 7b 07 8d 49
```



# Certificates on the web

---

## Subject's CommonName can be:

- An explicit name, e.g. `cs.stanford.edu` , or
- A name with a wildcard character, e.g.  
`*.stanford.edu` or `cs*.stanford.edu`

## matching rules:

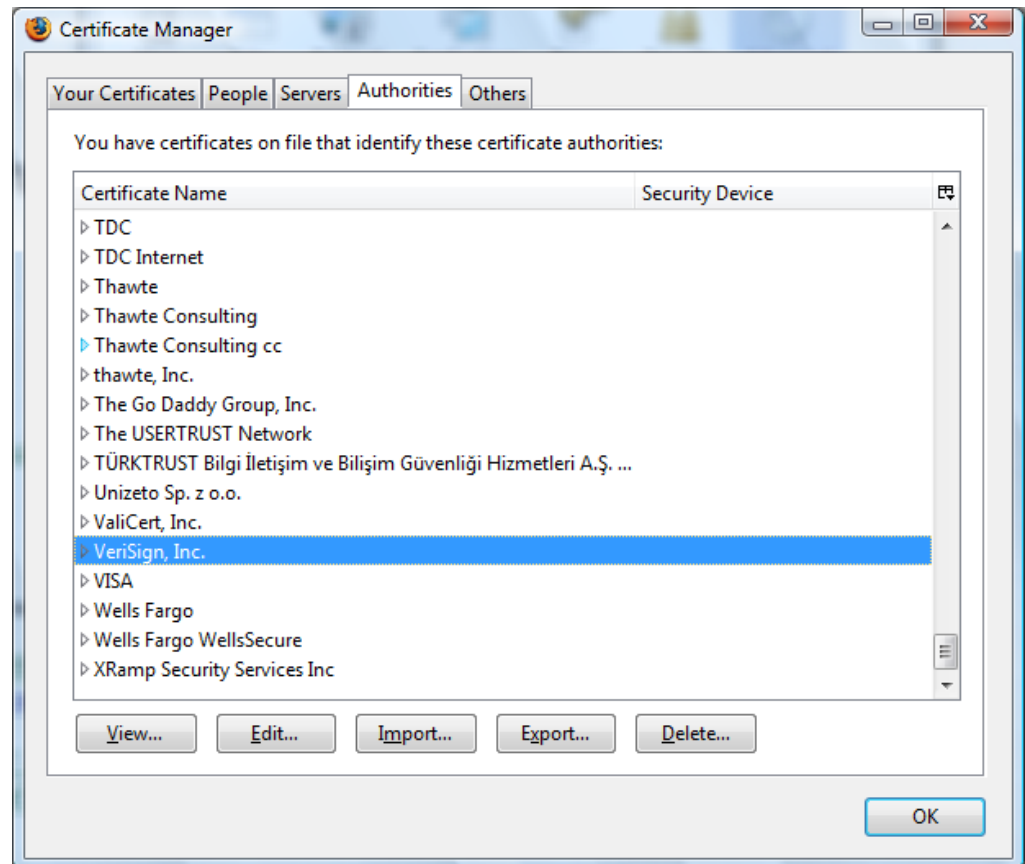
IE7: “\*” must occur in leftmost component, does not match “.”

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

FF3: “\*” matches anything

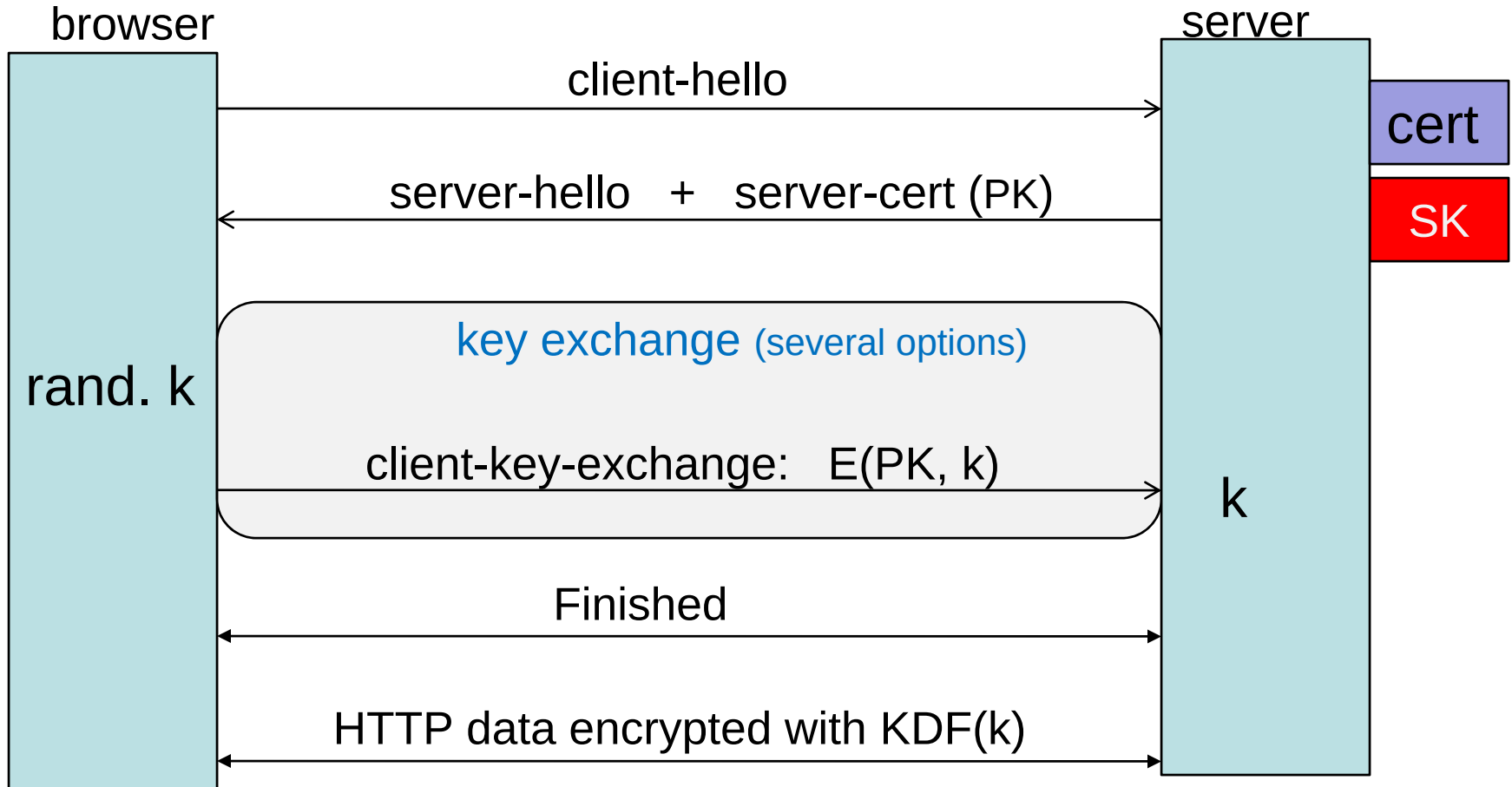
# Certificate Authorities

Browsers accept  
certificates from a  
large number of CAs





# Brief overview of SSL/TLS



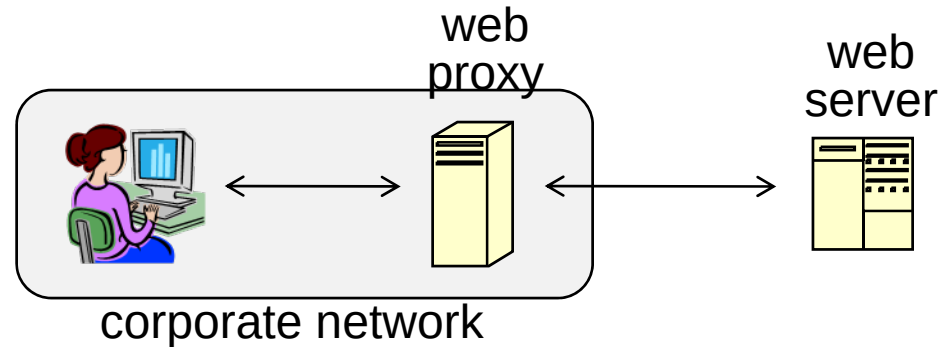
Most common: server authentication only

# Integrating SSL/TLS with HTTP $\Rightarrow$ HTTPS

## Two complications

- **Web proxies**

solution: browser sends  
CONNECT domain-name  
before client-hello (dropped by proxy)



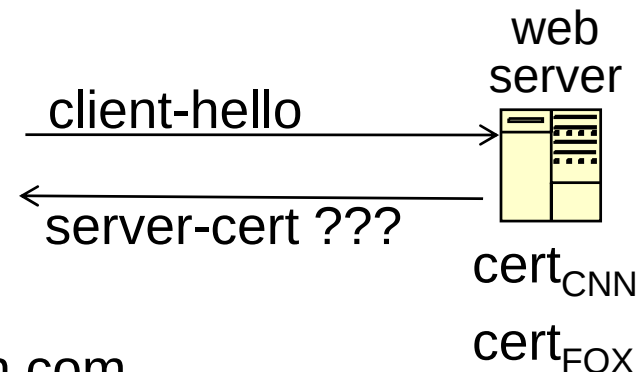
- **Virtual hosting:**

two sites hosted at same IP address.

solution in TLS 1.1 (RFC 4366)

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

implemented in FF2 and IE7 (vista)



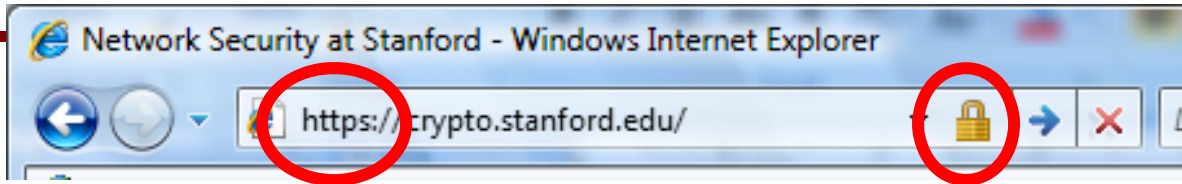
# Why is HTTPS not used for all web traffic?

---

- **Slows down web servers**
- **Breaks Internet caching**
  - ISPs cannot cache HTTPS traffic
  - Results in increased traffic at web site
- **Incompatible with virtual hosting** (older browsers)

# HTTPS in the Browser

# The lock icon: SSL indicator



## Intended goal:

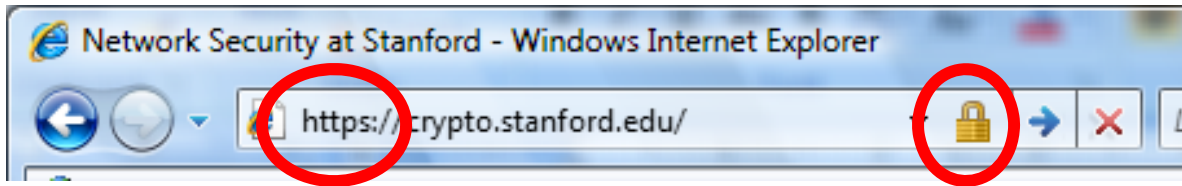
- Provide user with identity of page origin
- Indicate to user that page contents were not viewed or modified by a **network attacker**



## In reality:

- Origin ID is not always helpful  
    example: **Stanford HR is hosted at BenefitsCenter.com**
- Many other problems (next few slides)

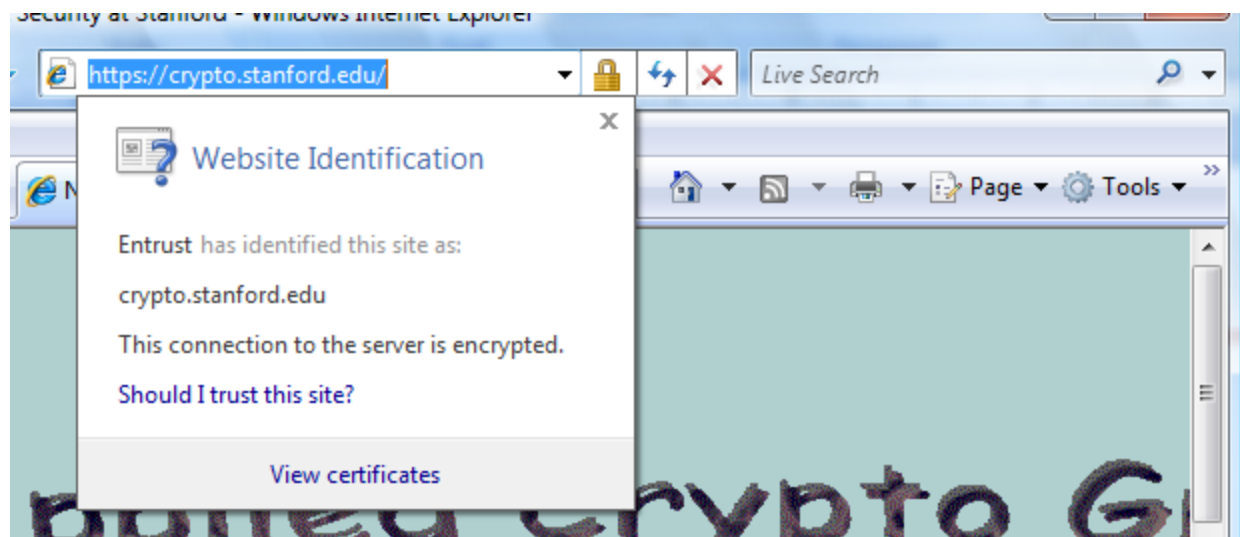
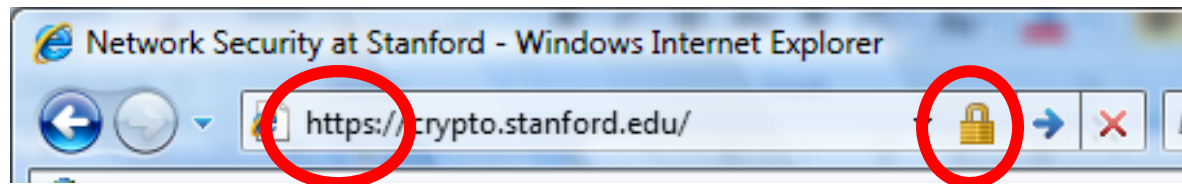
# When is the (basic) lock icon displayed



- **All elements on the page fetched using HTTPS**  
(with some exceptions)
- **For all elements:**
  - HTTPS cert issued by a CA trusted by browser
  - HTTPS cert is valid (e.g. not expired)
  - CommonName in cert matches domain in URL

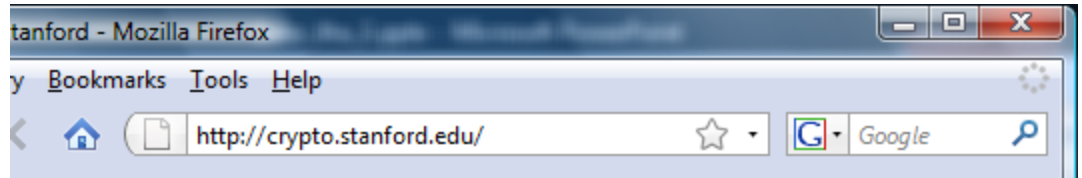
# The lock UI: help users authenticate site

IE7:

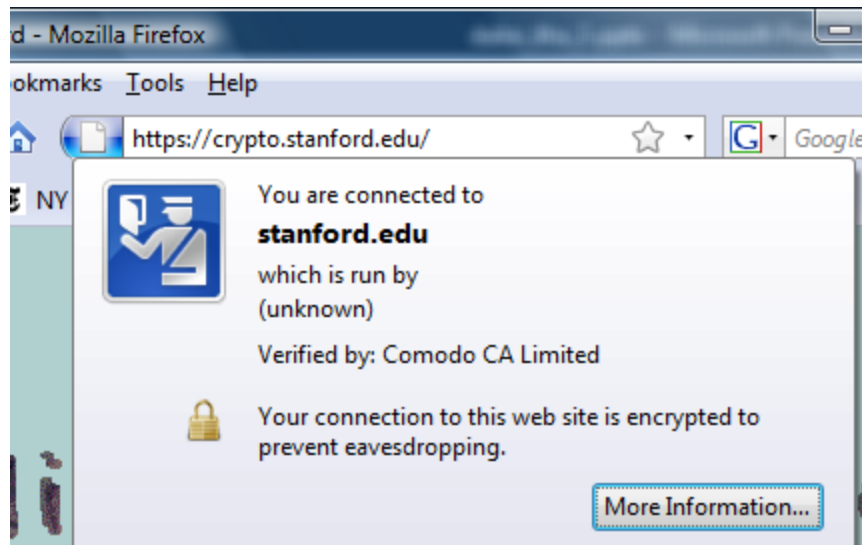
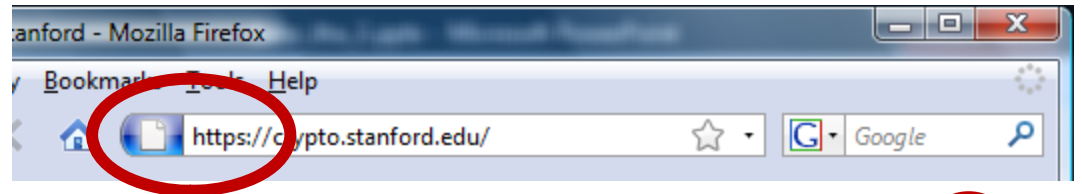


# The lock UI: help users authenticate site

Firefox 3: (no SSL)



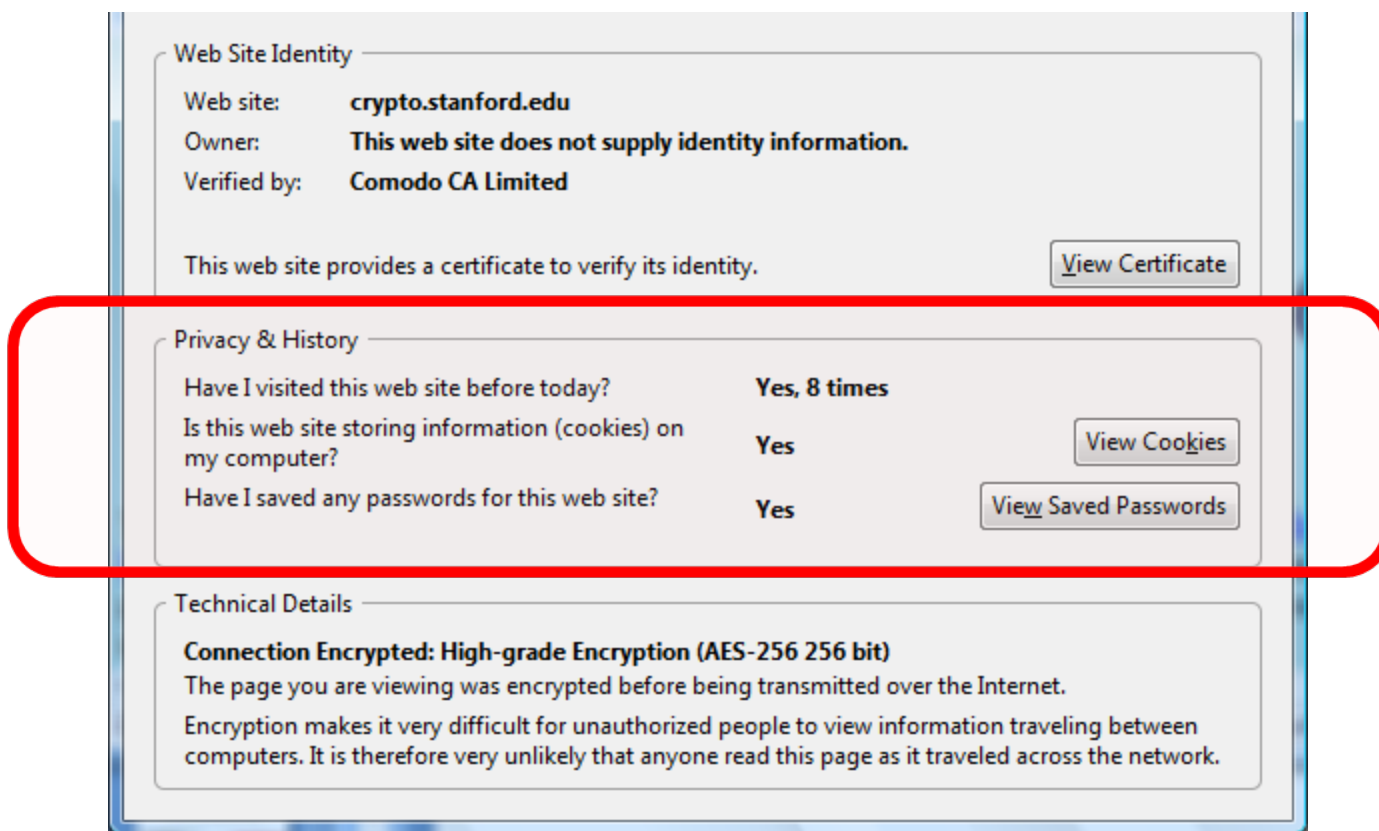
(SSL)





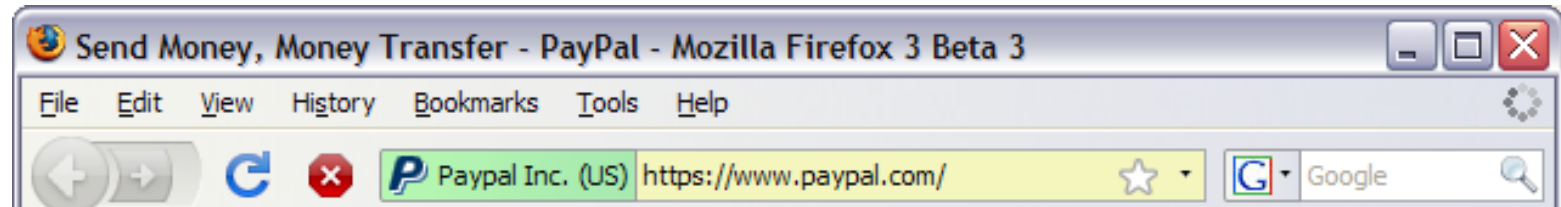
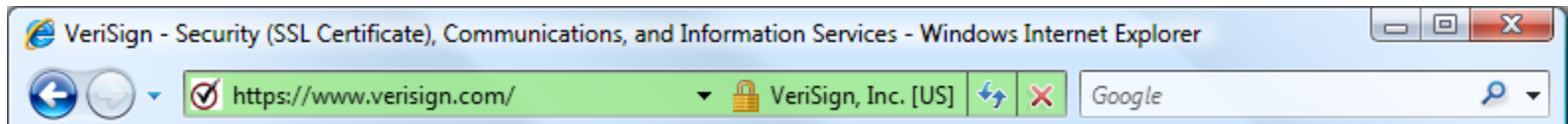
# The lock UI: help users authenticate site

## Firefox 3: clicking on bottom lock icon gives



# The lock UI: Extended Validation (EV) Certs

- **Harder to obtain than regular certs**
  - requires human lawyer at CA to approve cert request
- **Designed for banks and large e-commerce sites**



- **Helps block “semantic attacks”:** [www.bankofthevwest.com](http://www.bankofthevwest.com)

# HTTPS and login pages: incorrect version

Users often land on login page over HTTP:

- Type site's HTTP URL into address bar, or
- Google links to the HTTP page

View source:

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

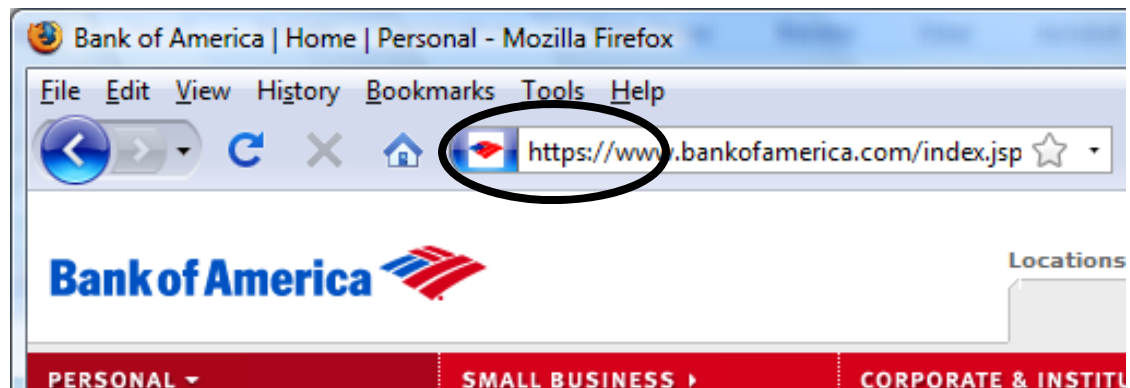
```
action="https://onlineservices.wachovia.com/..."
```



# HTTPS and login pages: guidelines

## General guideline:

- Response to <http://login.site.com>  
should be Redirect: <https://login.site.com>



# Problems with HTTPS and the Lock Icon

# Problems with HTTPS and the Lock Icon

---

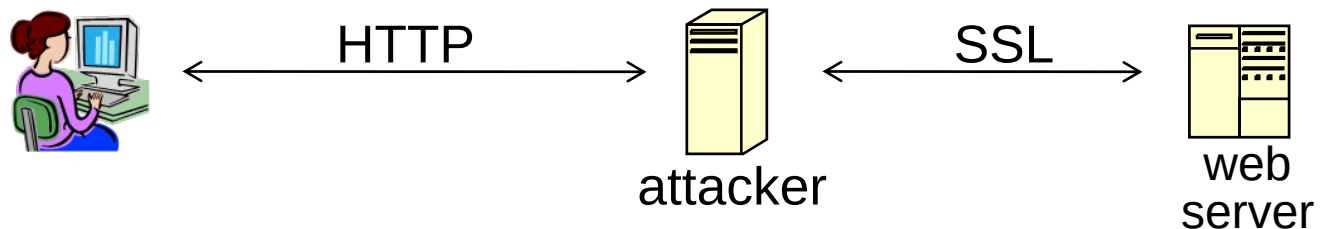
- 1. Upgrade from HTTP to HTTPS**
- 2. Semantic attacks on certs**
- 3. Invalid certs**
- 4. Mixed content**
  - HTTP and HTTPS on the same page
- 5. Origin contamination**
  - Weak HTTPS page contaminates stronger HTTPS page

# 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

Easy attack: prevent the upgrade (ssl\_strip) [Moxie'08]



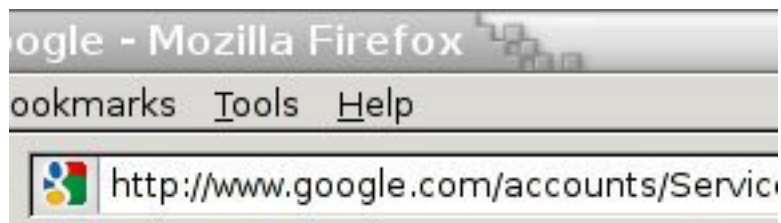
`<a href=https://...>`  $\Rightarrow$  `<a href=http://...>`

Location: **https**://...  $\Rightarrow$  Location: **http**://... (redirect)

`<form action=https://... >`  $\Rightarrow$  `<form action=http://...>`

# Tricks and Details

**Tricks:** drop-in a clever fav icon



## Details:

- Erase existing session and force user to login:  
ssl\_strip injects “Set-cookie” headers to delete existing session cookies in browser.

Number of users who detected HTTP downgrade: 0



## 2. Semantic attacks on certs

---

**International domains:**    **xyz.cn**

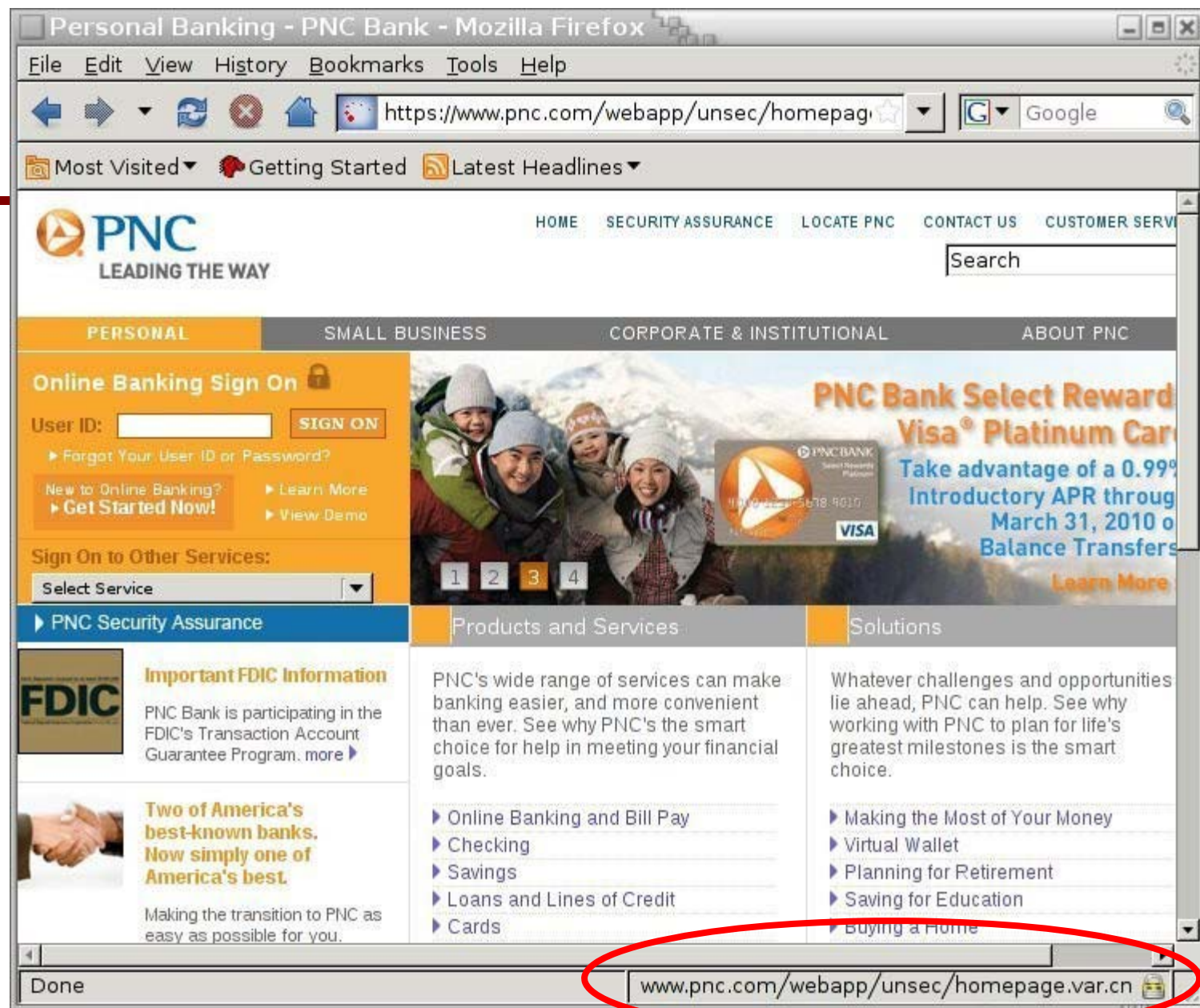
- Rendered using international character set
- Observation: chinese character set contains chars that look like “/” and “?” and “.” and “=”

**Attack:**    buy domain cert for    \*.badguy.cn  
setup domain called:

[www.bank.com/accounts/login.php?q=me.baguy.cn](http://www.bank.com/accounts/login.php?q=me.baguy.cn)

note:    single cert    \*.badguy.cn    works for all sites

Extended validation (EV) certs may help defeat this



[Moxie'08]

### 3. Invalid certs

---

#### Examples of invalid certificates:

- expired:       $\text{current-date} > \text{date-in-cert}$
- CommonName in cert does not match domain in URL
- unknown CA      (e.g. self signed certs)
  - Small sites may not want to pay for cert

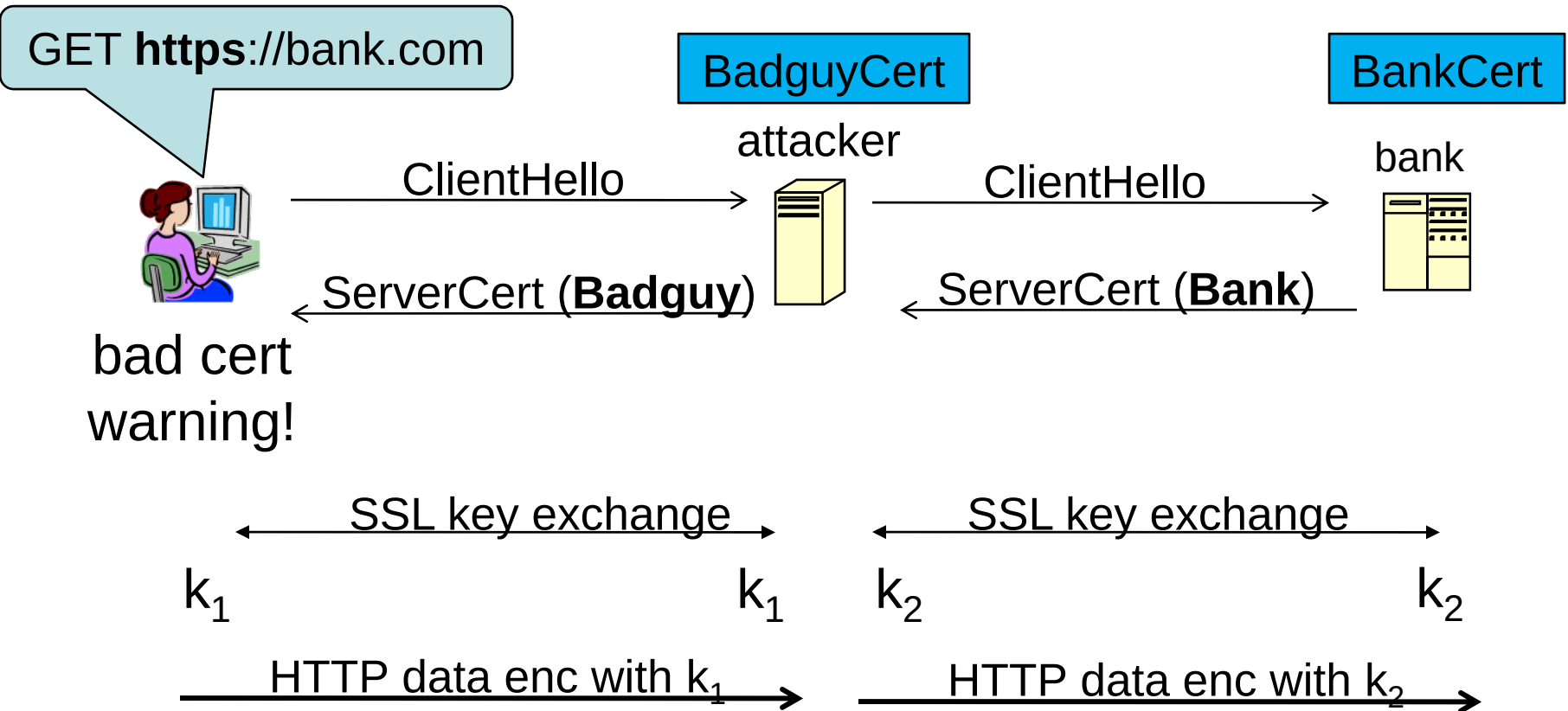
Users often ignore warning:

Is it a misconfiguration or an attack?      User can't tell.

Accepting invalid cert enables man-in-middle attacks

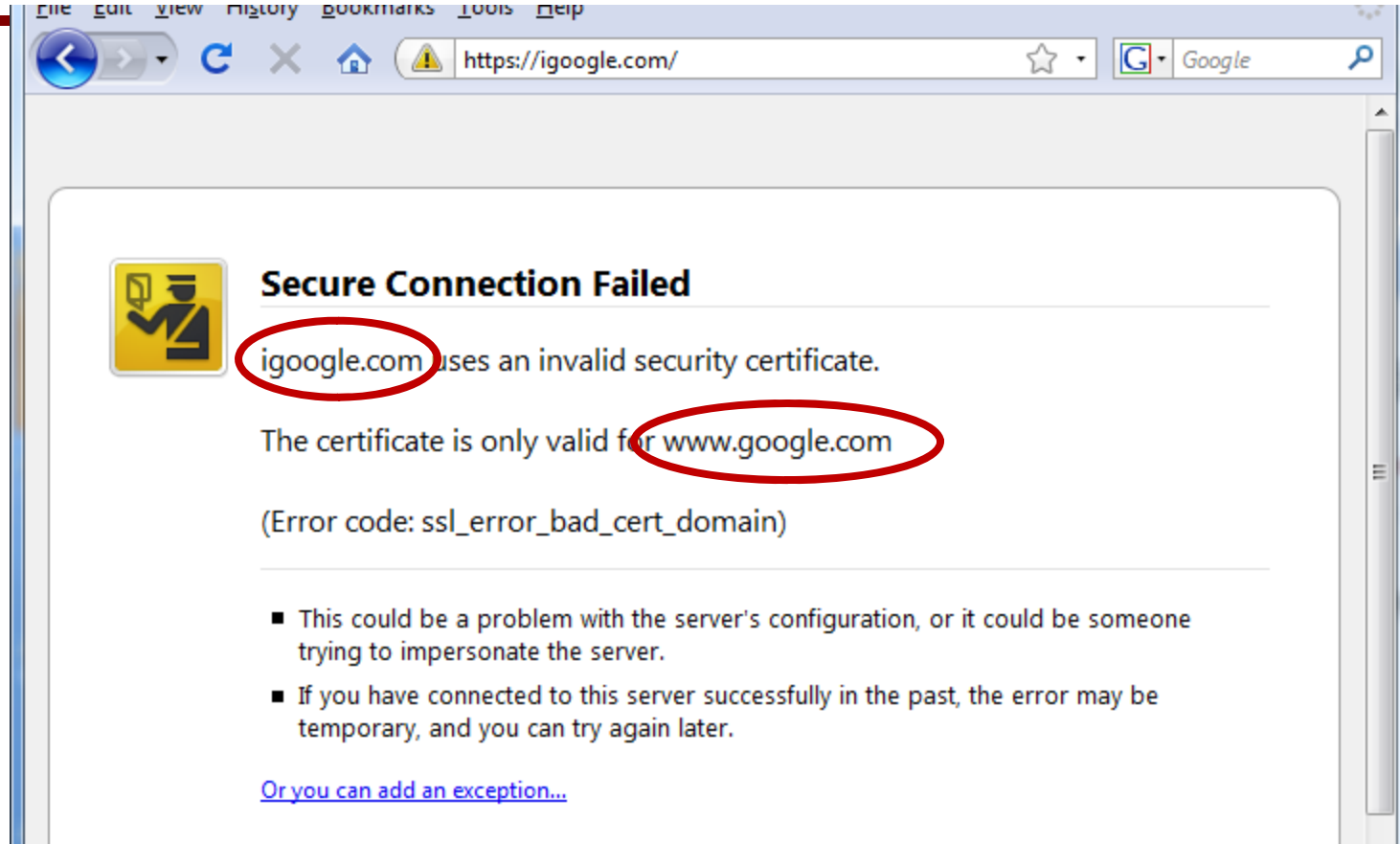
(see <http://crypto.stanford.edu/ssl-mitm> )

# Man in the middle attack using invalid certs



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

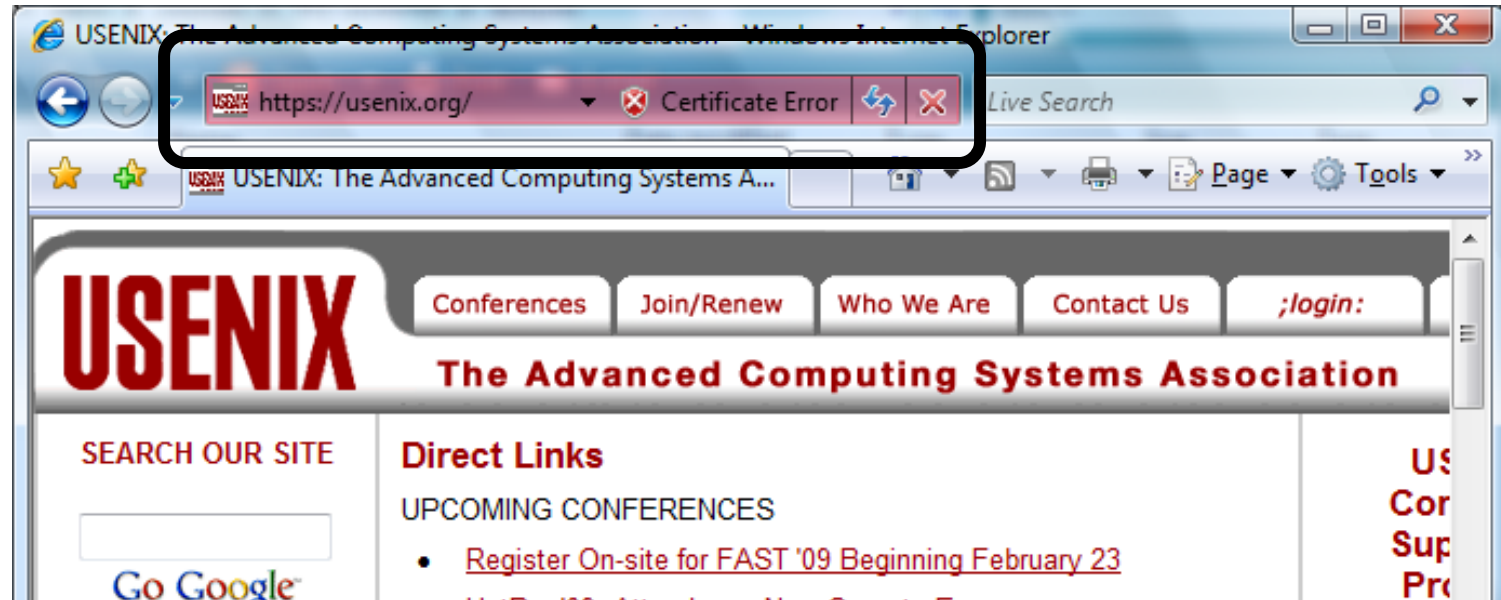
# Firefox: Invalid cert dialog



**Firefox 3.0: Four clicks to get firefox to accept cert**

- page is displayed with full HTTPS indicators

# IE: invalid cert URL bar



## 4. Mixed Content: HTTP and HTTPS

---

Page loads over HTTPS, but contains content over HTTP  
(e.g. `<script src="http://.../script.js">` )

**IE7:** displays mixed-content dialog and no SSL lock

**Firefox 3.0:** displays `!` over lock icon (no dialog by default)

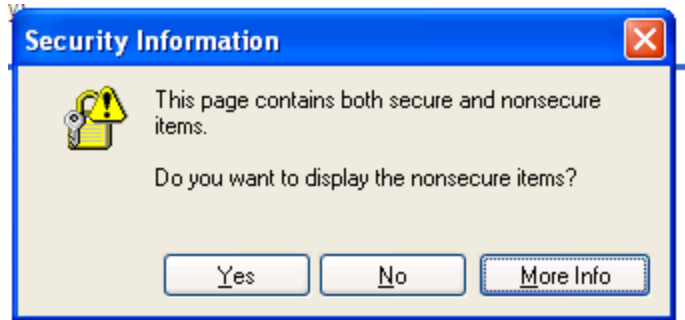
**Both browsers:**

- Flash swf file over HTTP does not trigger warning !!
- note: Flash can script the embedding page

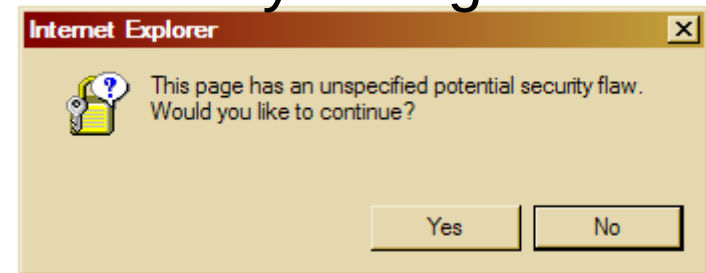
**Safari:** does not attempt to detect mixed content

# Mixed Content: HTTP and HTTPS

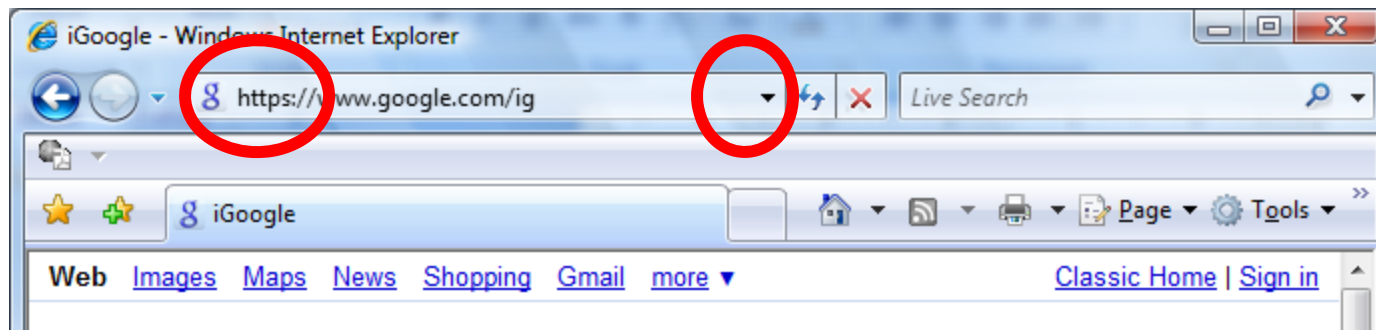
IE7:



silly dialogs



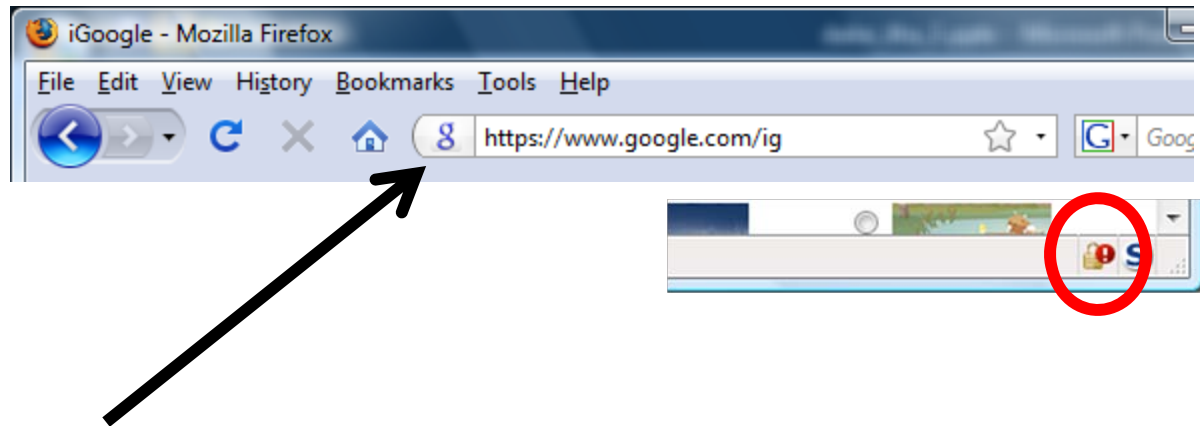
No SSL lock in address bar:



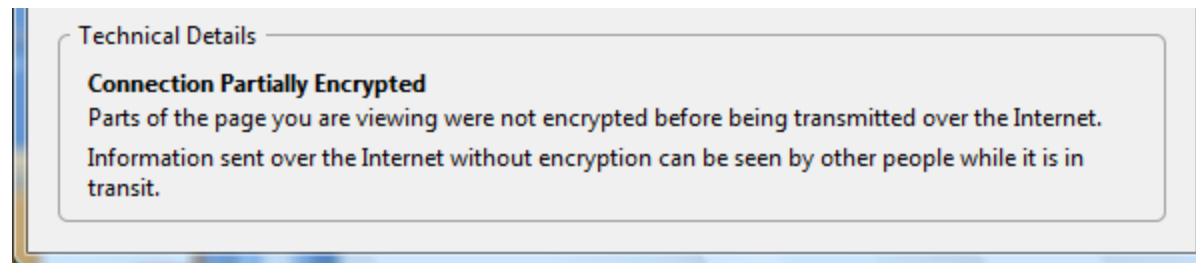


# Mixed Content: HTTP and HTTPS

Firefox 3.0:



- No SSL indicator in address bar
- Clicking on bottom lock gives:



# Mixed content and network attacks

---

banks: after login all content served over HTTPS

**Developer error:** somewhere on bank site write

```
<embed src=http://www.site.com/flash.swf>
```

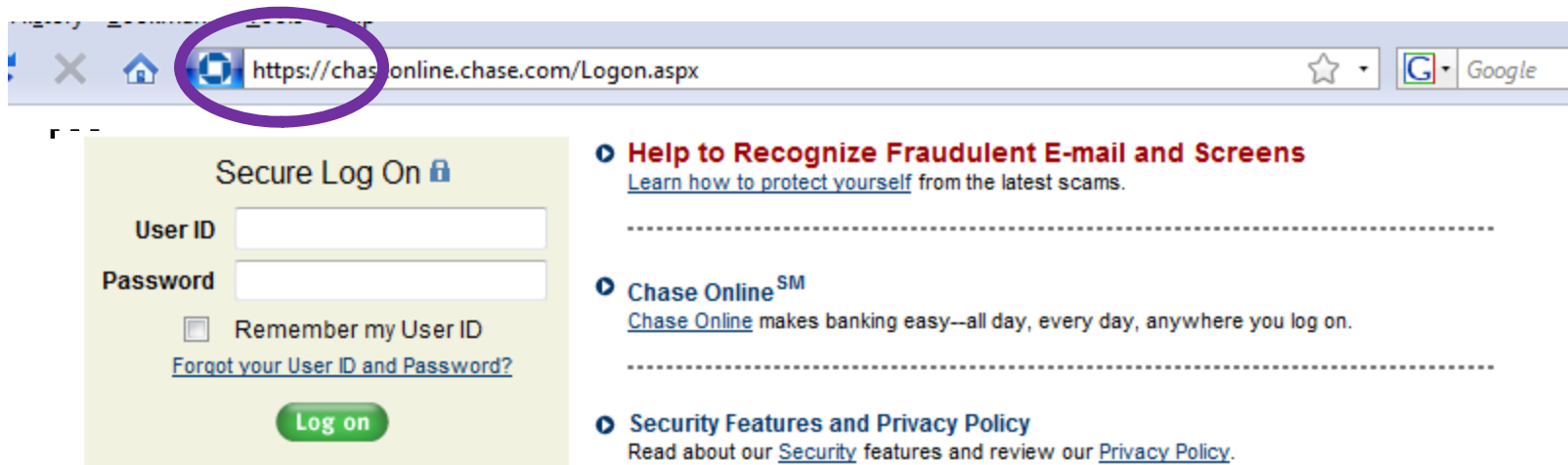
Active network attacker can now hijack session

Better way to include content:

```
<embed src=//www.site.com/flash.swf>
```

served over the same protocol as embedding page

# An Example From an Online Bank

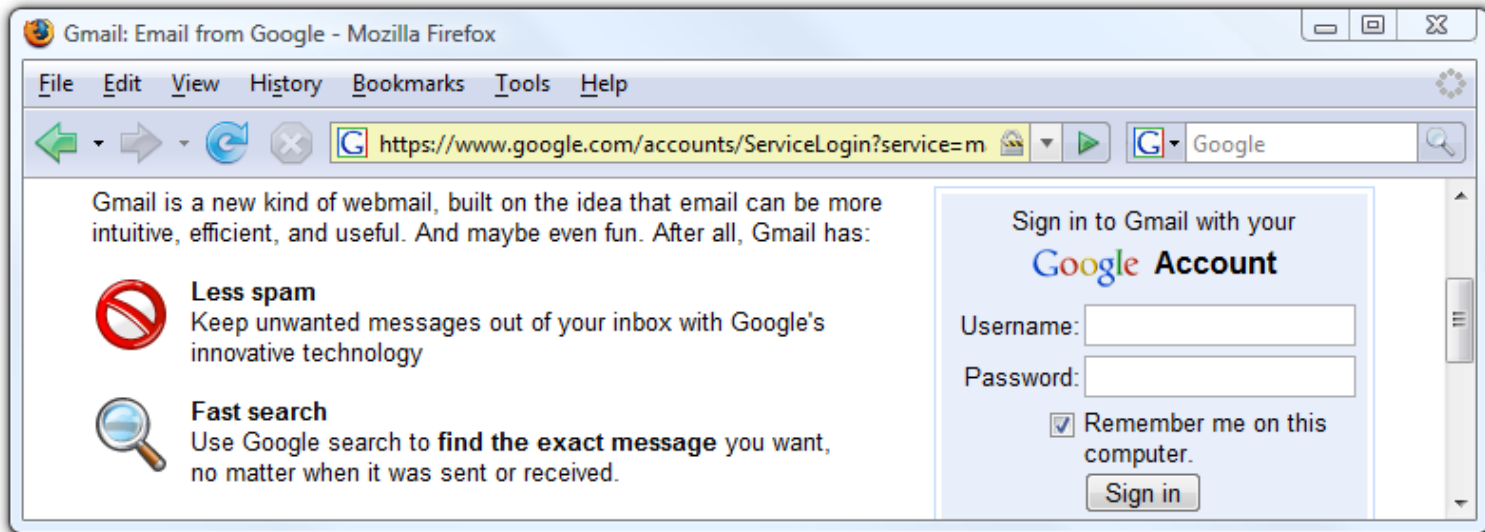


```
var so = new SWFObject("http://mfasa.chase.com/auth/device.swf", ...
```

network attacker can modify SWF file and hijack session

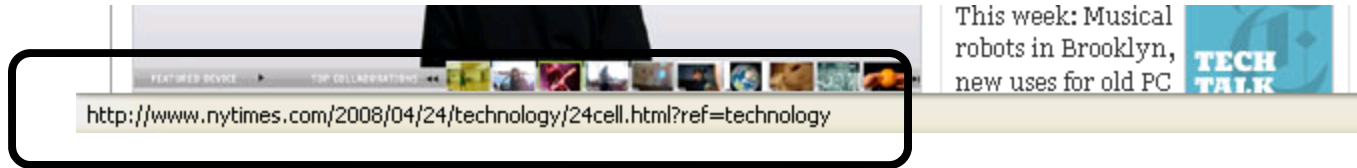
(the site has been fixed)

## 5. Origin Contamination: an example



**safeLock:** removes lock from top page after loading bottom page

# Final note: the status Bar



## Trivially spoofable

```
<a href="http://www.paypal.com/"  
  onclick="this.href = 'http://www.evil.com/';">  
  PayPal</a>
```

**THE END**