Secure Channels:
Practical Pitfalls and Insufficiency

Security Analysis of HTTPS So Far…

- Needed to argue security

- We assumed that:
  - Cryptographic primitives are secure
  - Interacting end points are uncompromised
  - A “Perfect” Protocol achieves its stated / defined properties both in design and implementation
  - Attacker can do anything within its defined power

Assumptions in the threat model

- User is using a secure channel
- Crypto primitives are secure
- TLS protocol design is secure
- TLS protocol implementation is secure
- Certificate issuers are uncompromised
- Users check browser UI correctly
- Alice & Bob’s secrets are secure
- Entities are authenticated correctly

Question (I) is Important!

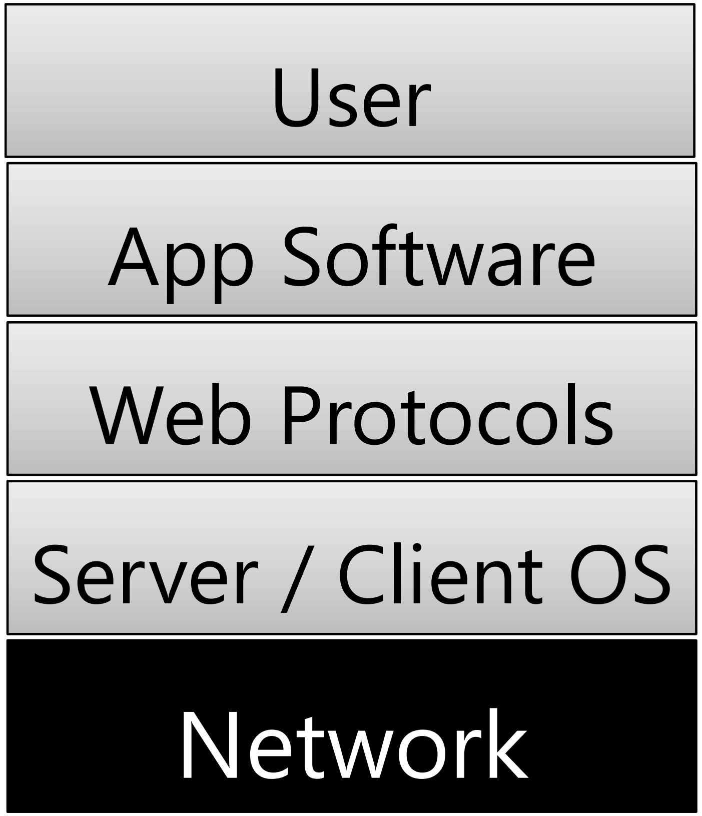
- You visit https://gmail.com on a WiFi network with no cert errors. Can you safely assume that your email will reach Gmail safe from ALL network attackers?

- Stick to the threat model – assume the assumptions hold and then check:
  (A) Defeats DNS Cache Poisoning?
  (B) Defeats BGP Route Hijacking?
  (C) Defeats TCP / IP attacks?

Yes, it does!

Secure Channels:
Theory vs. Practice

How Do Systems Fail In Practice?



User

App Software

Web Protocols

Server / Client OS

Network

- Threat Model:
  - Attackers (Eve & Mallory)
  - Assumptions
  - Desired Security Property:
    - The “CIA” of secure channels
- Attackers can win by going “outside the threat model” in practice
- 2 Ways to go “outside the model”:
  - Attack the assumptions
  - Violate other security properties that are not captured by the threat model

Revisit:
Assumptions in the threat model

- **User is using a secure channel**
- Users check browser UI correctly
- Certificate issuers are uncompromised
- Alice & Bob’s secrets are secure
- Crypto primitives are secure
- TLS protocol design is secure
- TLS protocol implementation is secure
- Entities are authenticated correctly

Secure Channels:
Practical Pitfalls in HTTPS

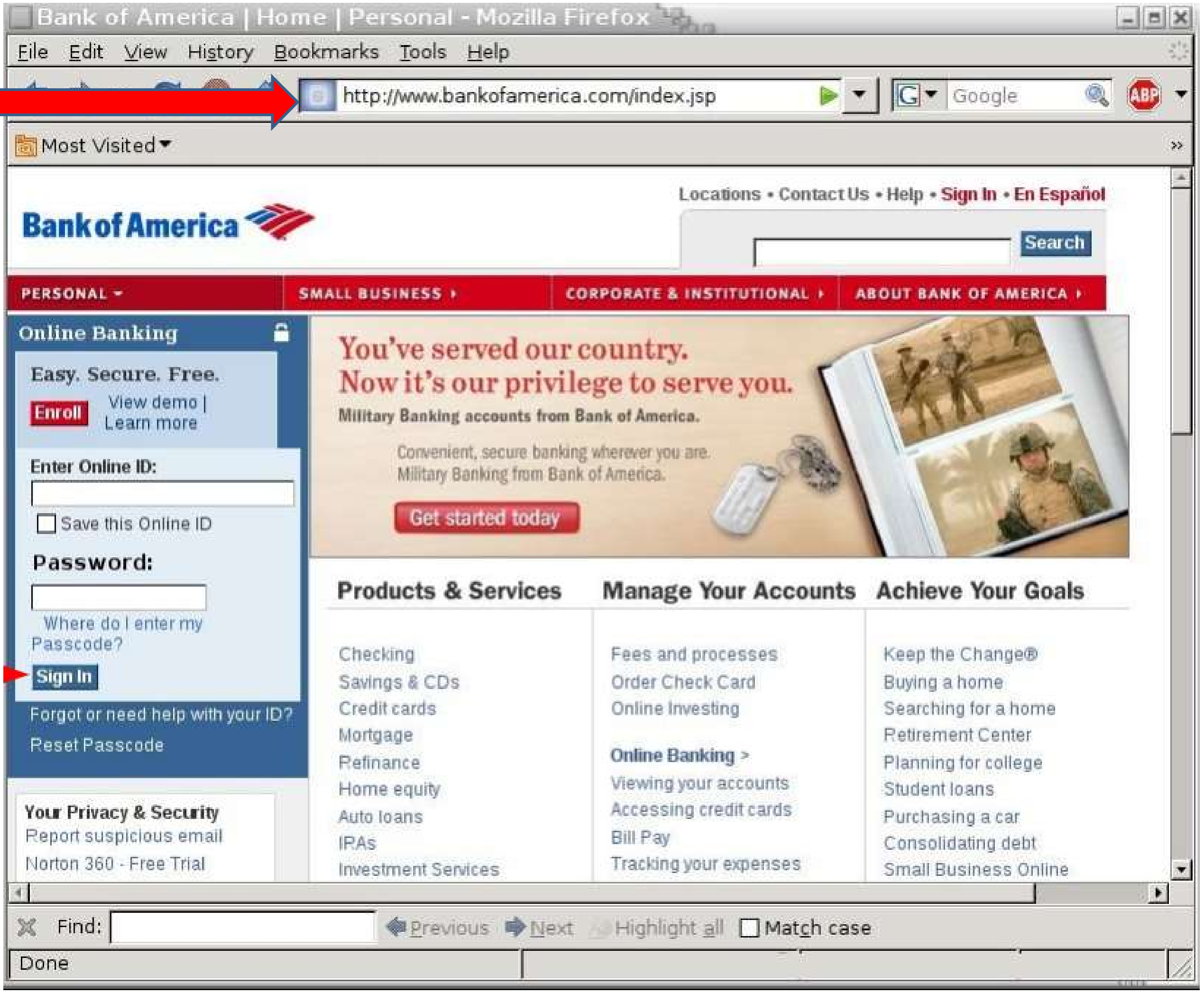
Practical Pitfalls In HTTPS:

The Secure Channel Isn‘t Used

Quiz

Suppose Alice clicks on http://bankof...com , and server redirects to https://bankof....com
But, Alice sees this page below.

HTTP



Bank of America | Home | Personal - Mozilla Firefox

http://www.bankofamerica.com/index.jsp

Easy. Secure. Free.

Enter Online ID:

□ Save this Online ID

Password:

Where do I enter my Passcode?

Forgot or need help with your ID?
Reset Passcode

Your Privacy & Security
Report suspicious email
Norton 360 - Free Trial

You've served our country.
Now it's our privilege to serve you.
Military Banking accounts from Bank of America.

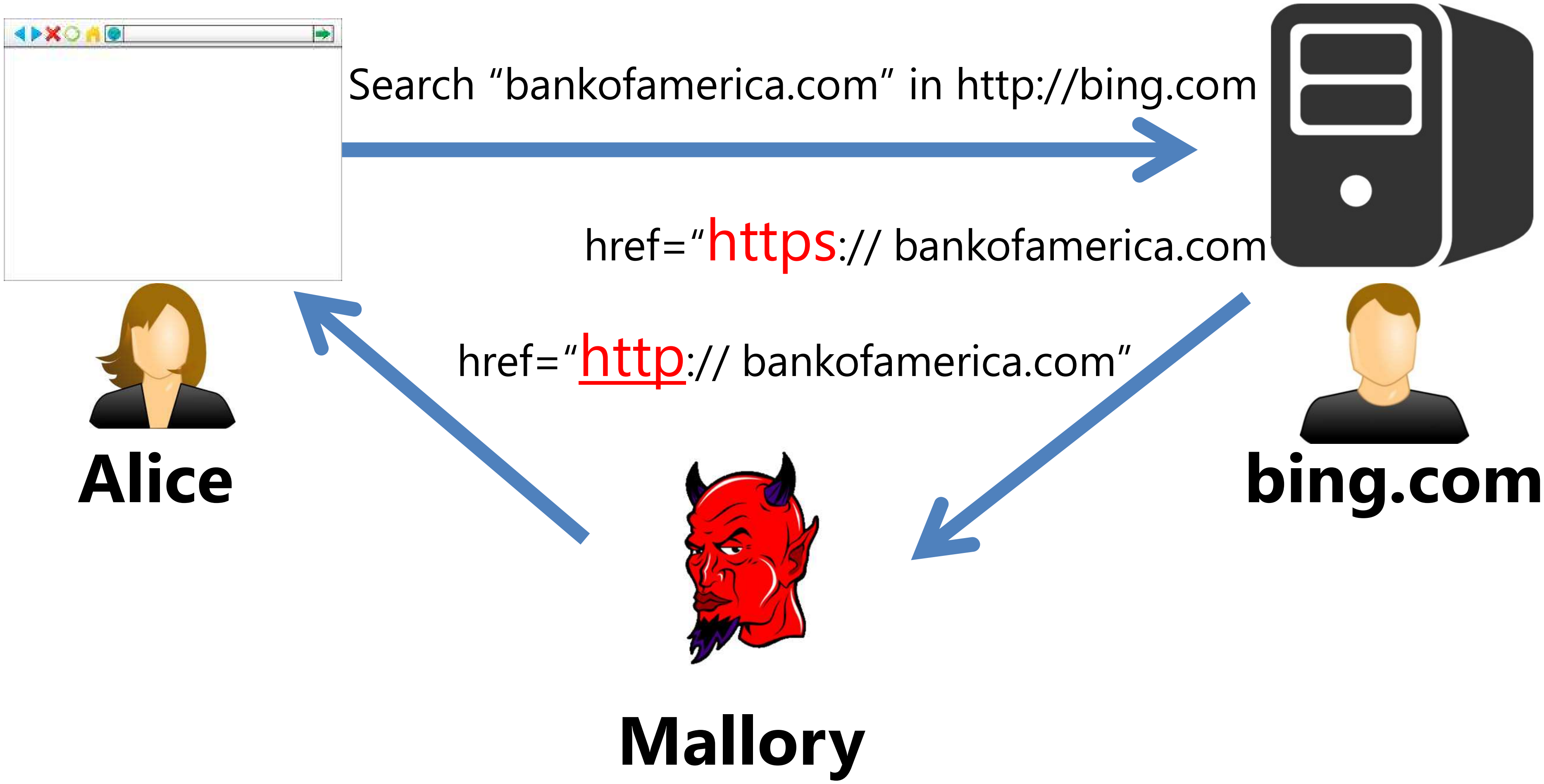
Convenient, secure banking wherever you are.
Military Banking from Bank of America.

Products & Services	Manage Your Accounts	Achieve Your Goals
Checking	Fees and processes	Keep the Change®
Savings & CDs	Order Check Card	Buying a home
Credit cards	Online Investing	Searching for a home
Mortgage		Retirement Center
Refinance	Online Banking >	Planning for college
Home equity	Viewing your accounts	Student loans
Auto loans	Accessing credit cards	Purchasing a car
IRAs	Bill Pay	Consolidating debt
Investment Services	Tracking your expenses	Small Business Online

Has something unsafe happened? How?

DEFCON 17: More Tricks For Defeating SSL --- Moxie

HTTP Downgrade



Search “bankofamerica.com” in http://bing.com

href=“https:// bankofamerica.com

href=“http:// bankofamerica.com”

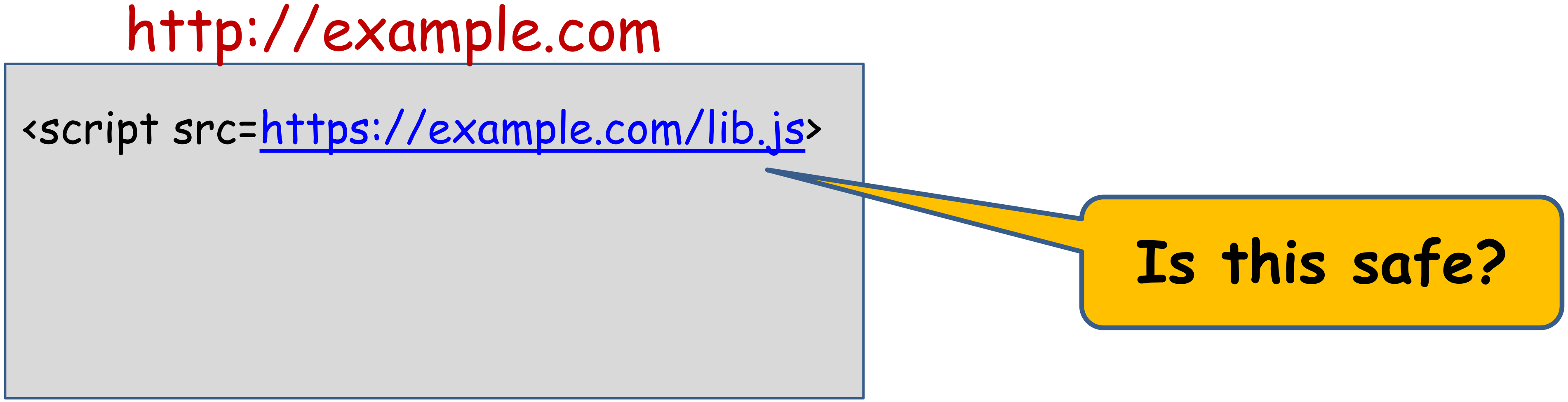
Alice

bing.com

Mallory

Think about this…

- Going from an HTTP to and HTTPS sub-resource.



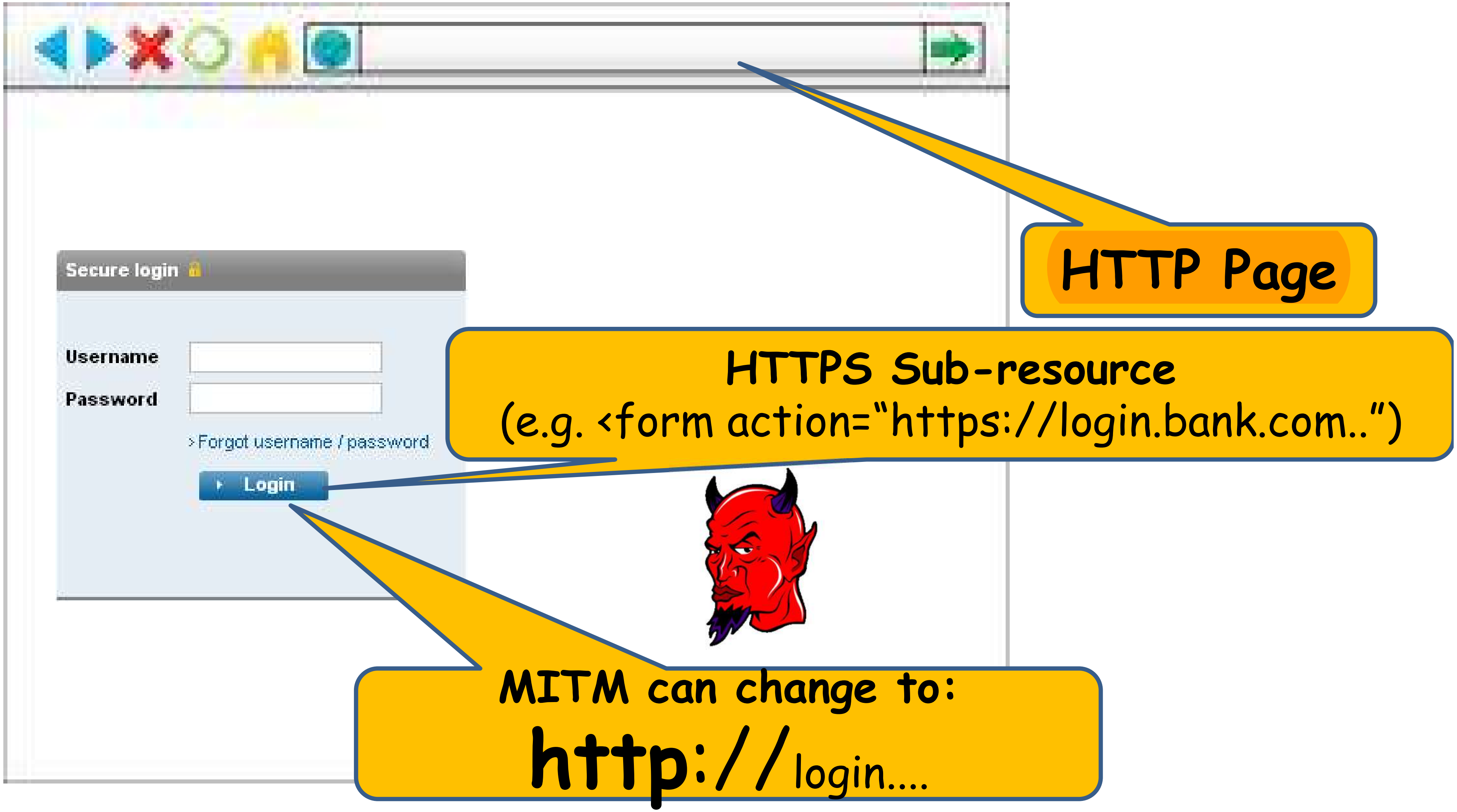
http://example.com

```
<script src=https://example.com/lib.js>
```

Is this safe?

More opportunities for downgrades…

- No security warning for sub-resource loads



HTTP Page

HTTPS Sub-resource
(e.g. <form action="https://login.bank.com..")

MITM can change to:
http:// login....

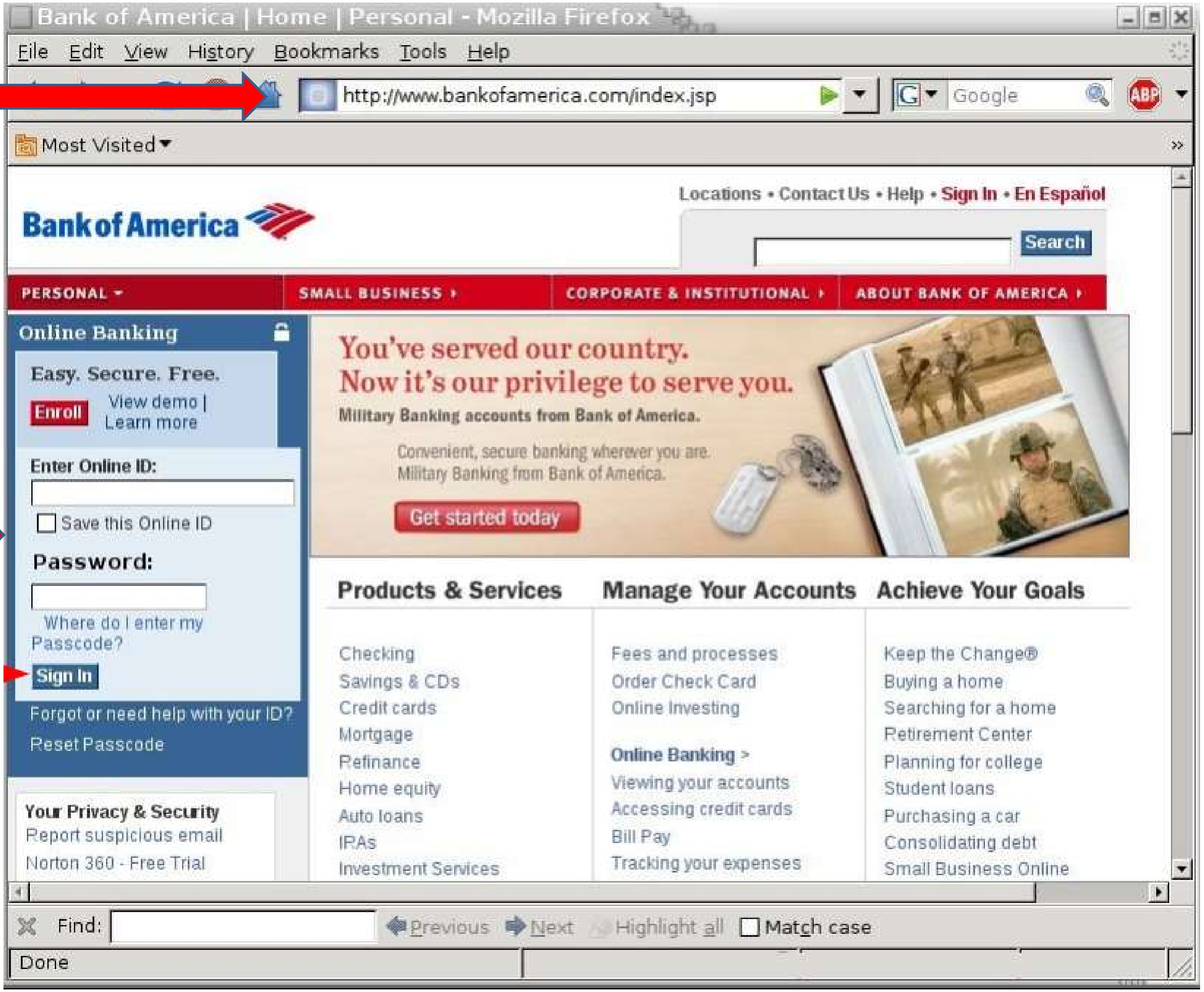
See Moxie BH'09

Quiz

HTTP

<iframe> over HTTPS

Is this safe?


Bank of America | Home | Personal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.bankofamerica.com/index.jsp
Google
Most Visited
Locations • Contact Us • Help • Sign In • En Español
Search
PERSONAL
SMALL BUSINESS
CORPORATE & INSTITUTIONAL
ABOUT BANK OF AMERICA
Enroll
Sign In
Get started today
Find:
Previous
Next
Highlight all
Match case
Done

Bank of America

Online Banking

Easy. Secure. Free.

View demo | Learn more

Enter Online ID:

□ Save this Online ID

Password:

Where do I enter my Passcode?

Forgot or need help with your ID?

Reset Passcode

Your Privacy & Security

Report suspicious email

Norton 360 - Free Trial

You've served our country.
Now it's our privilege to serve you.

Military Banking accounts from Bank of America.

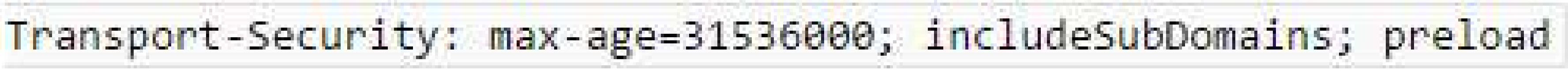
Convenient, secure banking wherever you are.
Military Banking from Bank of America.

Products & Services	Manage Your Accounts	Achieve Your Goals
Checking	Fees and processes	Keep the Change®
Savings & CDs	Order Check Card	Buying a home
Credit cards	Online Investing	Searching for a home
Mortgage		Retirement Center
Refinance	Online Banking >	Planning for college
Home equity	Viewing your accounts	Student loans
Auto loans	Accessing credit cards	Purchasing a car
IRAs	Bill Pay	Consolidating debt
Investment Services	Tracking your expenses	Small Business Online

Can Mallory intercept Alice's password without Alice noticing?

Defense Against HTTP Downgrade

- HSTS: **HTTP Strict Transport Security**
- Idea: Server supplies a header over HTTPS



```
Transport-Security: max-age=31536000; includeSubDomains; preload
```

- Browser never issues any HTTP request to this site if it receives this header

The old way of using the server to send a https redirection request - in an attempt to "upgrade" the user to https is not safe

Insecure Cookies

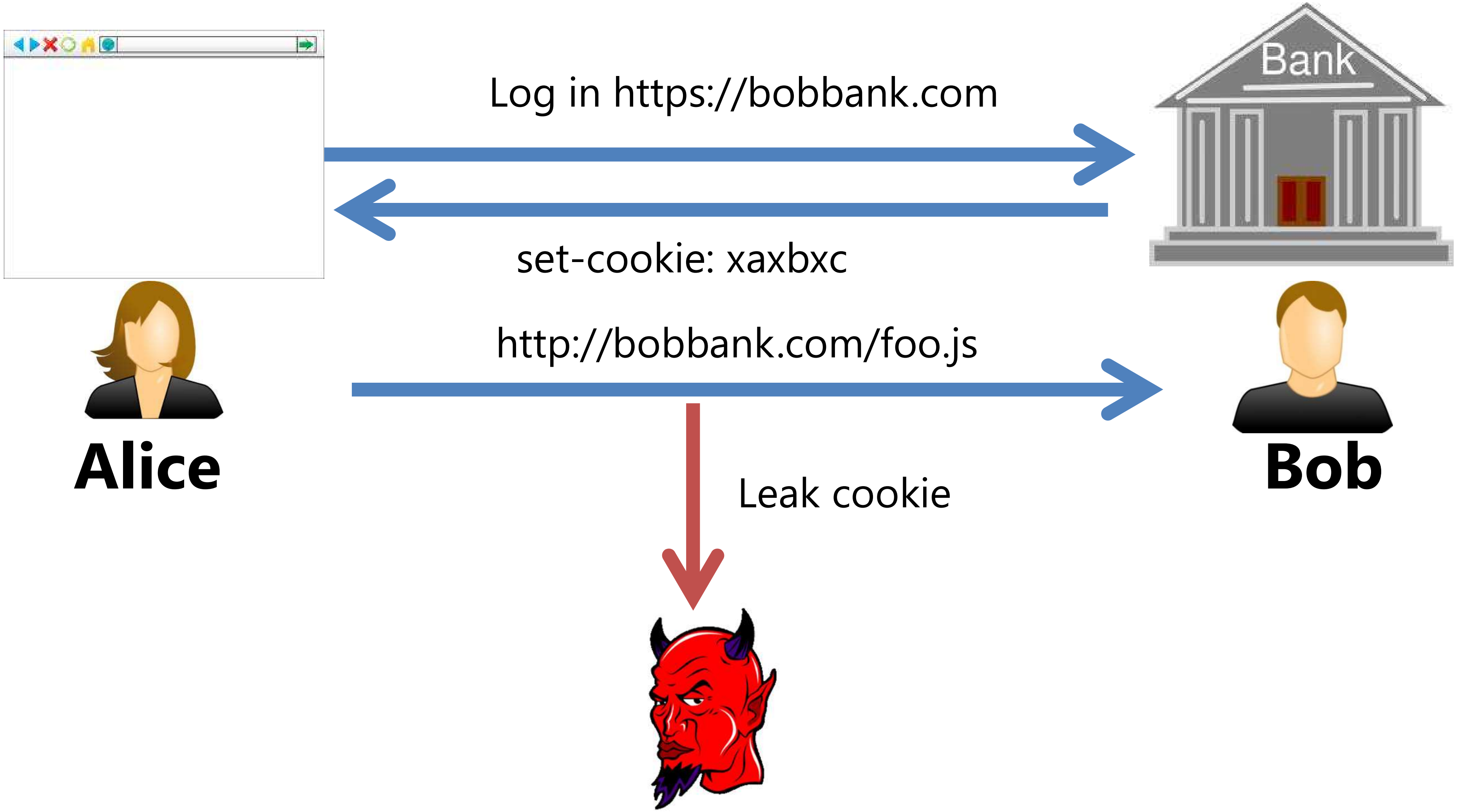
https://example.com

<img src=http://example.com/logo.gif>

Is this safe?
[Images can't run code]

Leaks the cookie on to HTTP traffic
(if the 'secure' flag is unset for cookies)

Insecure Cookies



Log in https://bobbank.com

set-cookie: xaxbxc

http://bobbank.com/foo.js

Leak cookie

Why? Because, HTTP Strict Transport Security (HSTS) and 'Secure' flag for cookies are turned off here

Secure Channel For Web Cookies?

- Does the web have a secure channel for cookies?
- Confidentiality – Yes!
  - Over HTTPS only using ‘Secure’ keyword
    - Won’t be sent over HTTP
  - Can be read by JS via DOM API
- Integrity – No!
  - Can be written by HTTP requests
    - E.g. Set-cookie: SID=bad; secure
    - It will override the previously set Secure cookie
  - Can be written / deleted via JavaScript
    - **evil.example.com** can set cookies for **example.com**

Cookies Lack Integrity: Real-World Implications

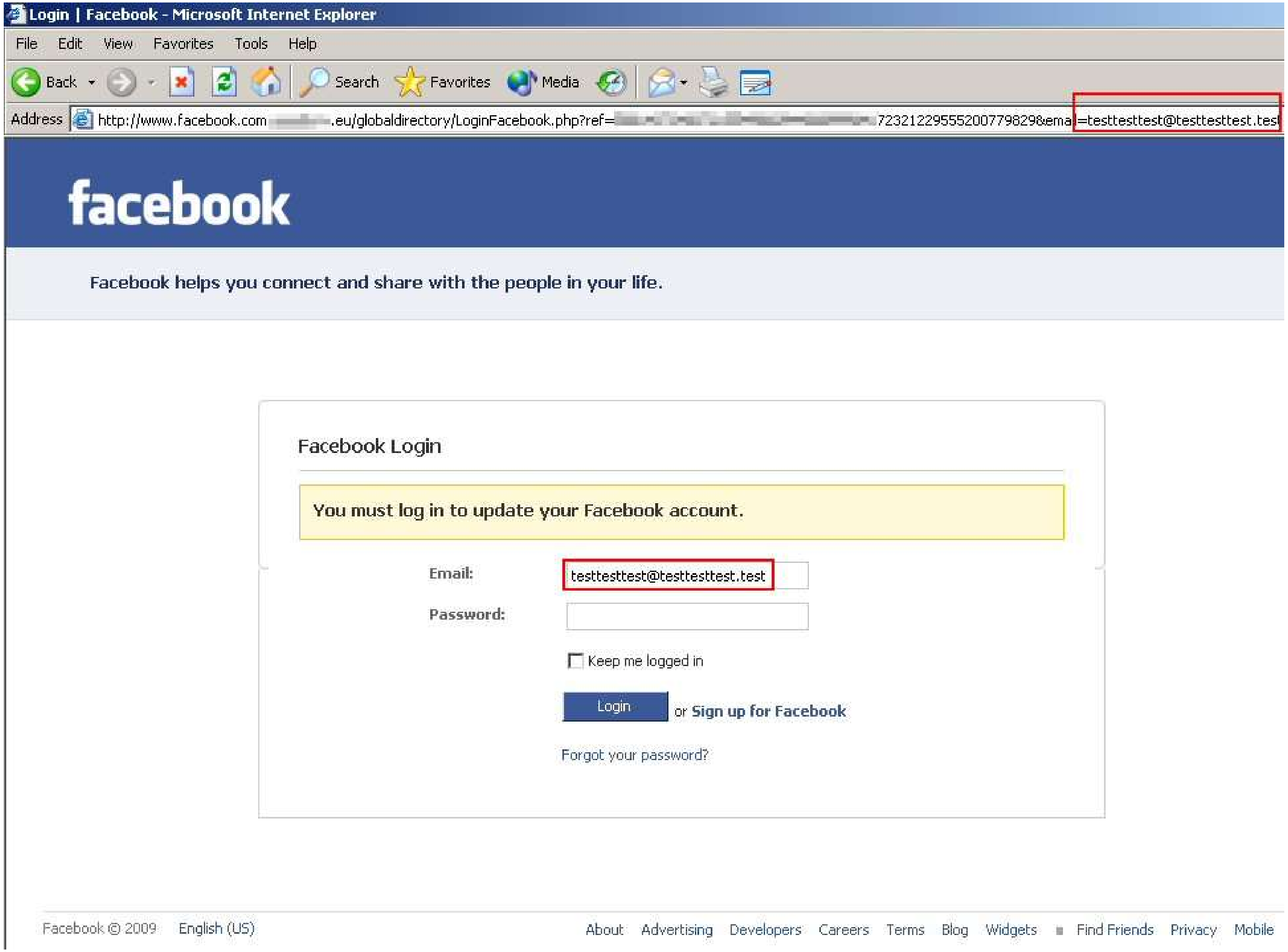
Revisit:
Assumptions in the threat model

- User is using a secure channel
- **Users check browser UI correctly**
- Certificate issuers are uncompromised
- Alice & Bob’s secrets are secure
- Crypto primitives are secure
- TLS protocol design is secure
- TLS protocol implementation is secure
- Entities are authenticated correctly

Practical Pitfalls In HTTPS:
UI Confusion

Phishing Attack

www.bankofthewest.com vs www.bankofthevvest.com



http://www.facebook.com[redacted].eu/globaldirectory/LoginFacebook.php?ref=[redacted]72321229555200779829&emai=testtesttest@testtesttest.test

Facebook helps you connect and share with the people in your life.

Facebook Login

You must log in to update your Facebook account.

Email: testtesttest@testtesttest.test

Password:

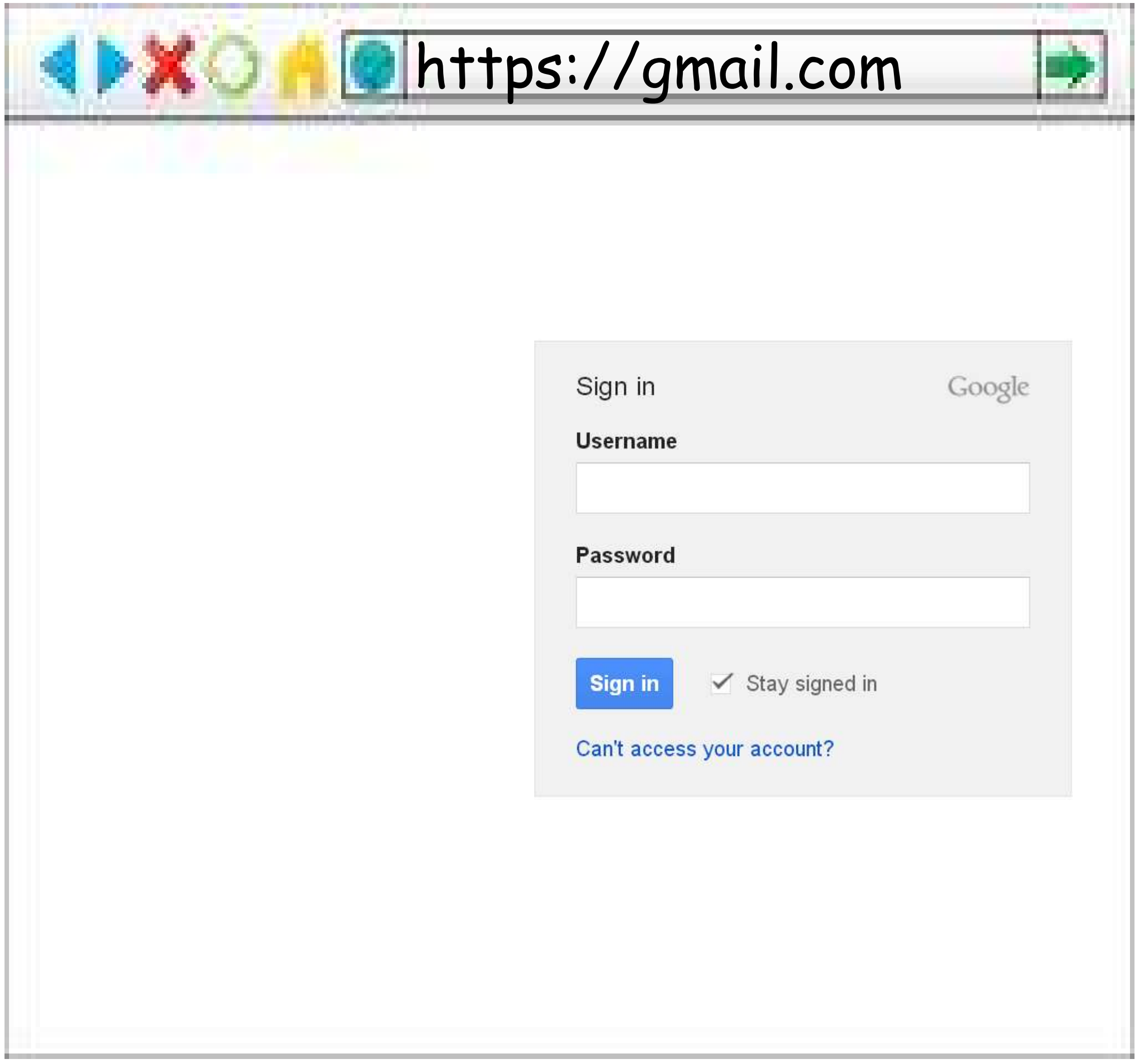
□ Keep me logged in

Login or Sign up for Facebook

Forgot your password?

Facebook © 2009 English (US) About Advertising Developers Careers Terms Blog Widgets Find Friends Privacy Mobile

The User Misinformed…



https://gmail.com

Sign in Google

Username

Password

Sign in ☑ Stay signed in

Can't access your account?

Null-byte Certificate

gmail.com\0.evil.com

Eric Johanson’05

The User Misinformed…

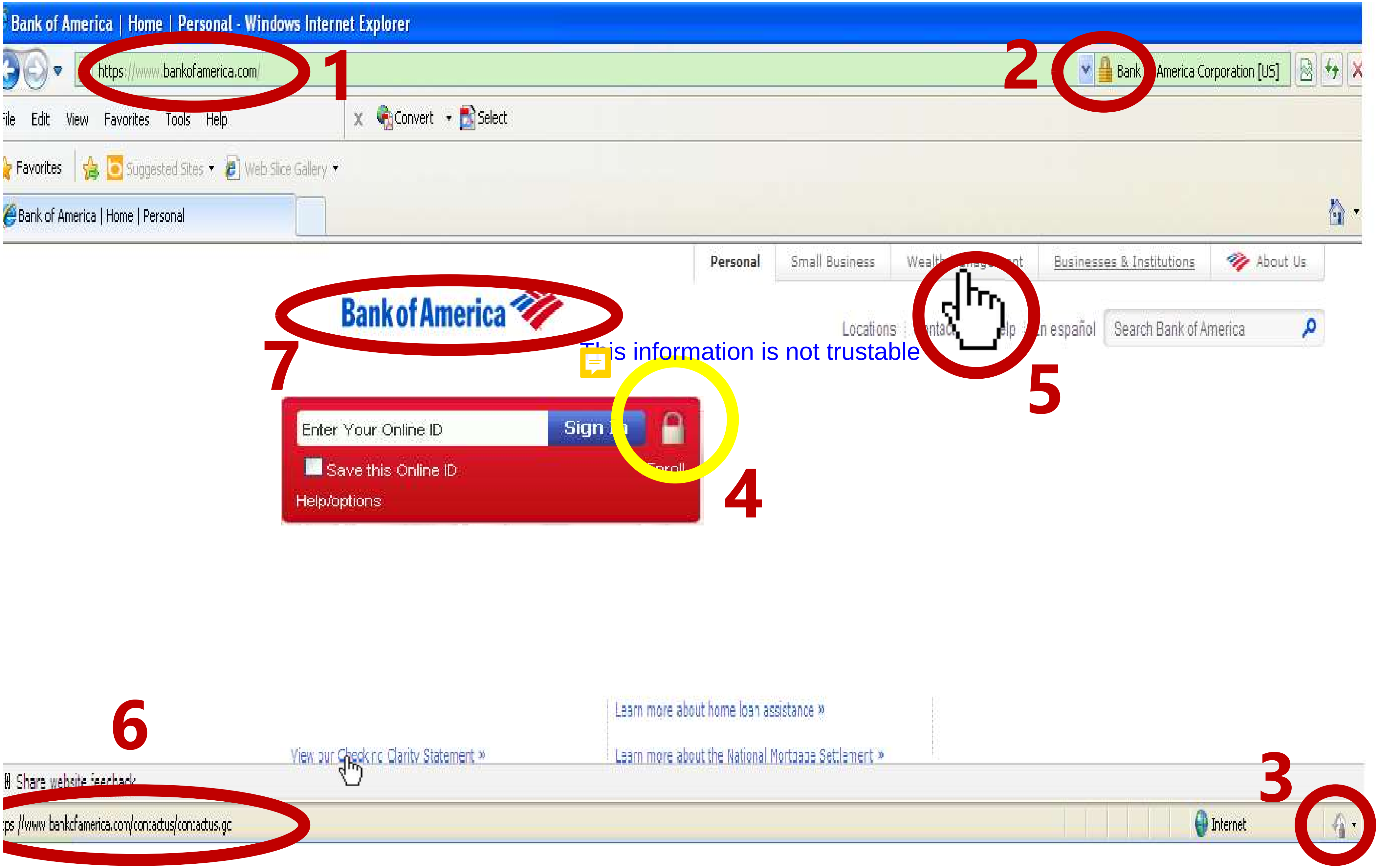
https://paypal.com

Registered Cert for:

p&#1072;ypal.com

Eric Johanson'05

The User Misinformed…



Bank of America | Home | Personal - Windows Internet Explorer

https://www.bankofamerica.com/

Bank of America Corporation [US]

This information is not trustable

Enter Your Online ID

Sign In

□ Save this Online ID

Help/options

Learn more about home loan assistance »

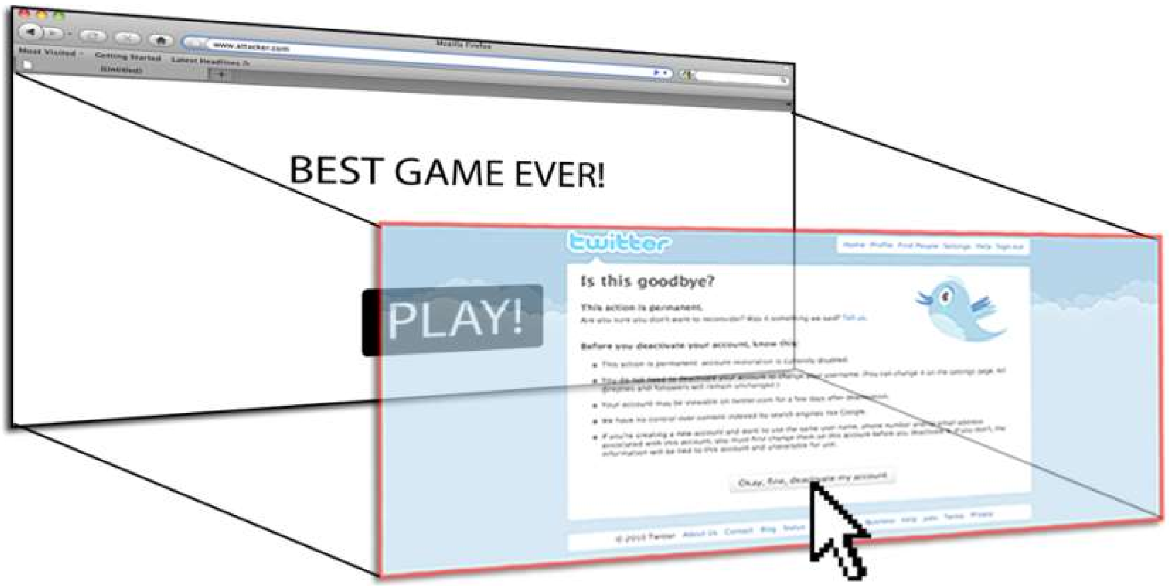
Learn more about the National Mortgage Settlement »

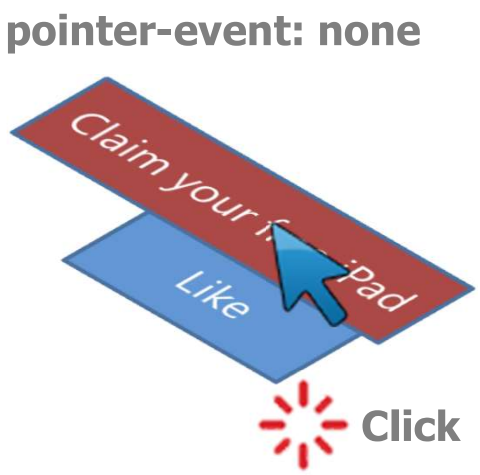
View our Checking Clarity Statement »

ps://www.bankcfamerica.com/contactus/contactus.gc

Clickjacking

- Pages can embed iframes from 3rd-party
  - Any site can host another in <iframe>
  - Frames can overlap
  - CSS controls the transparency, location of frames
- How to trick users?
  - E.g. `opacity : 0.1`, or `pointer-events: none`





[Clickjacking – Hansen, Grossman 2008](#)

Mixing HTTP and HTTPS

- Mixed Content
  - HTTP resources in HTTPS pages

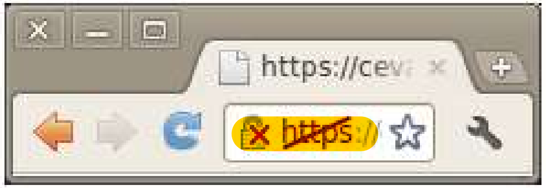
https://example.com

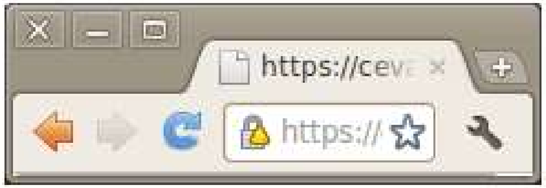
```
<script src=http://example.com/lib.js>
```

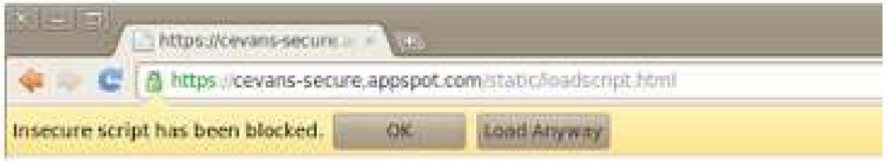
Is this safe?

Attacker can corrupt JS
and include payload

- What do browsers do for mixed content?
  - Legacy: Ignore, No security warning.
  - Recent: Block and present new UI indicators



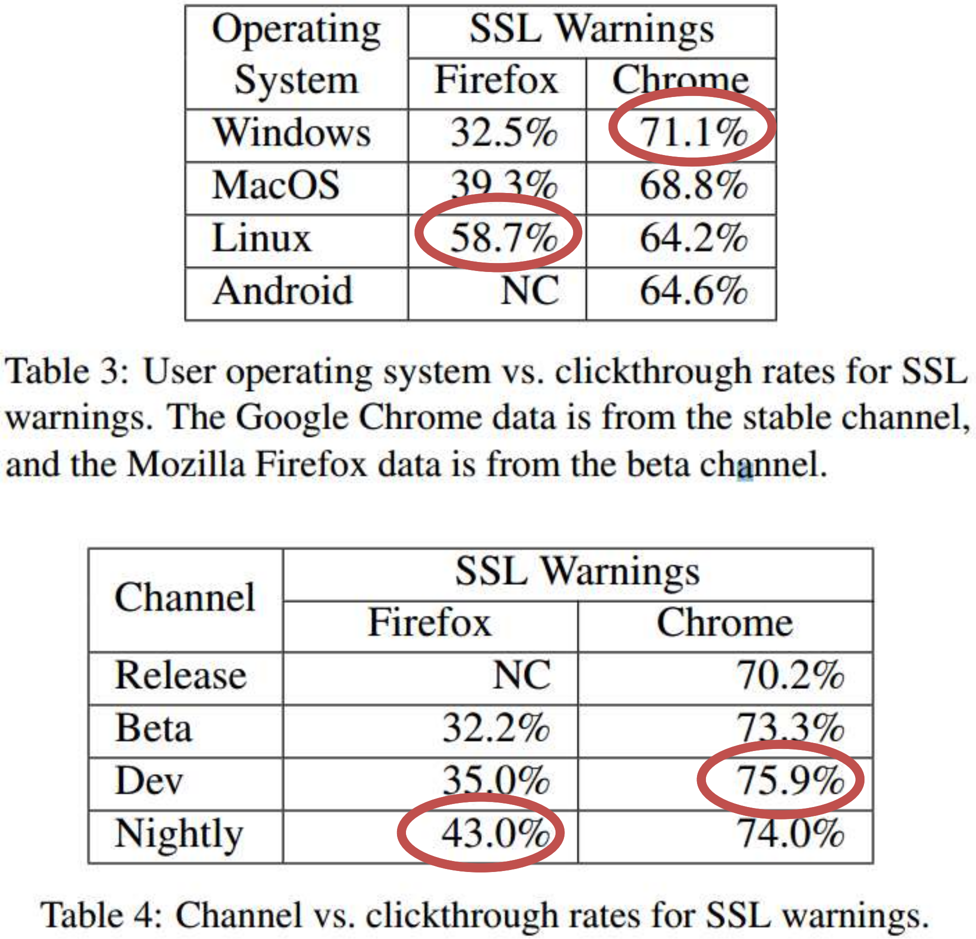




Insecure script has been blocked.

Coopting the User to Click-through

- Do users pay attention to cert warnings?



Operating System	SSL Warnings	
	Firefox	Chrome
Windows	32.5%	71.1%
MacOS	39.3%	68.8%
Linux	58.7%	64.2%
Android	NC	64.6%

Table 3: User operating system vs. clickthrough rates for SSL warnings. The Google Chrome data is from the stable channel, and the Mozilla Firefox data is from the beta channel.

Channel	SSL Warnings	
	Firefox	Chrome
Release	NC	70.2%
Beta	32.2%	73.3%
Dev	35.0%	75.9%
Nightly	43.0%	74.0%

Table 4: Channel vs. clickthrough rates for SSL warnings.

Akhawe, Devdatta, and Adrienne Porter Felt. "Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness." Usenix Security. 2013.

Revisit:
Assumptions in the threat model

- User is using a secure channel
- Users check browser UI correctly
- **Certificate issuers are uncompromised**
- Alice & Bob’s secrets are secure
- Crypto primitives are secure
- TLS protocol design is secure
- TLS protocol implementation is secure
- Entities are authenticated correctly

Practical Pitfalls in HTTPS:
Comprised Certificate Authorities

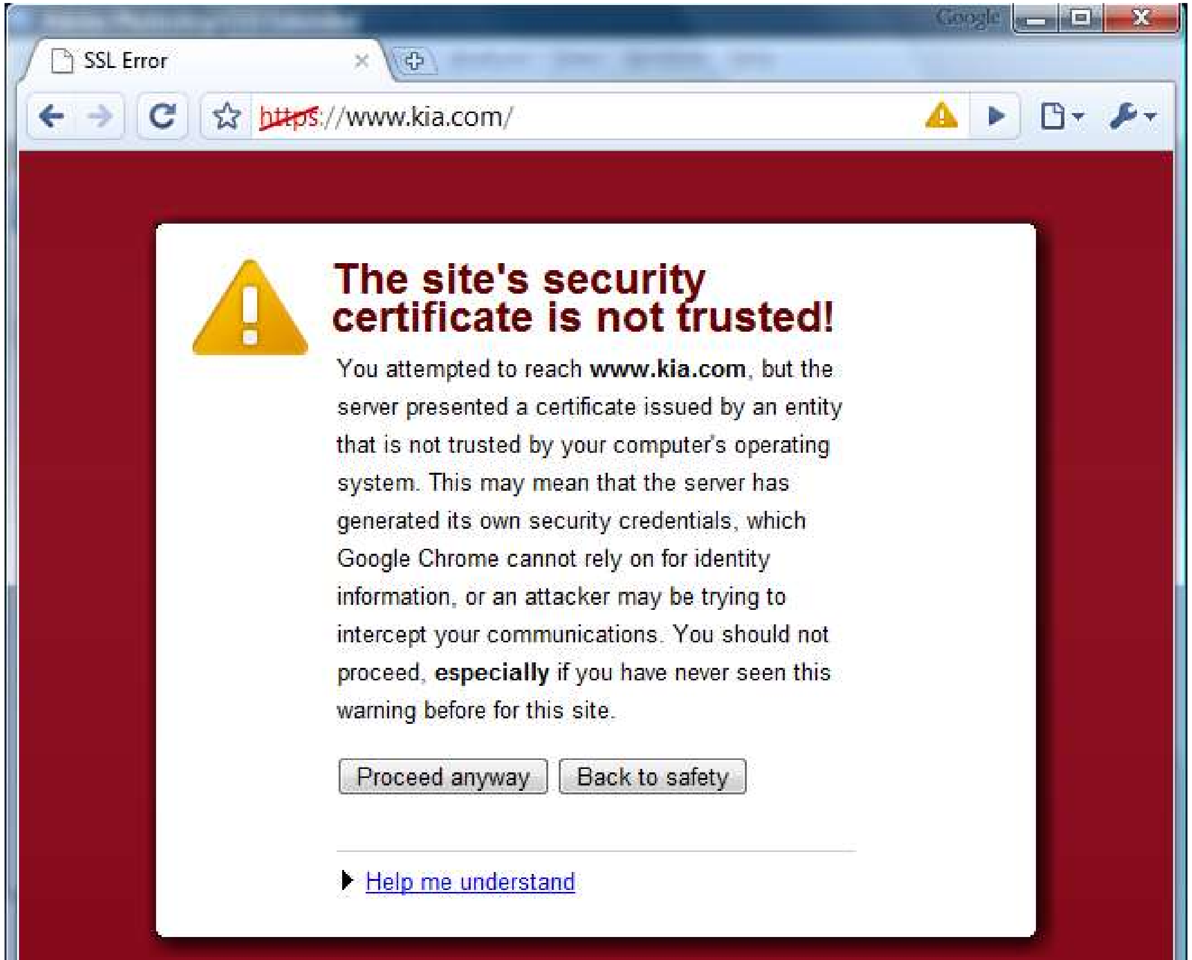
How Do I Get an SSL cert?

- Get a Root CA to issue you one
  - You can get some for free ☺
  - Paid ones: $10 - $50 / yr (not costly)

- What do they check for before issuing cert?
  - Valid email
  - You own the domain you want cert for?
    - E.g. you are the admin at http://evil.com
  - Sometimes a bit deeper, but basically that’s it!

Can I Be A CA?

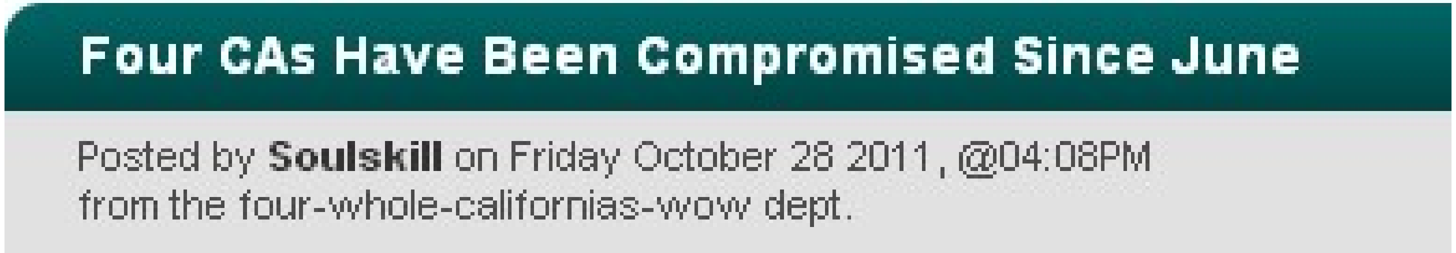
- Yes,
  - Self-sign certificates
  - Customers need to add you as root CA



The site's security certificate is not trusted!

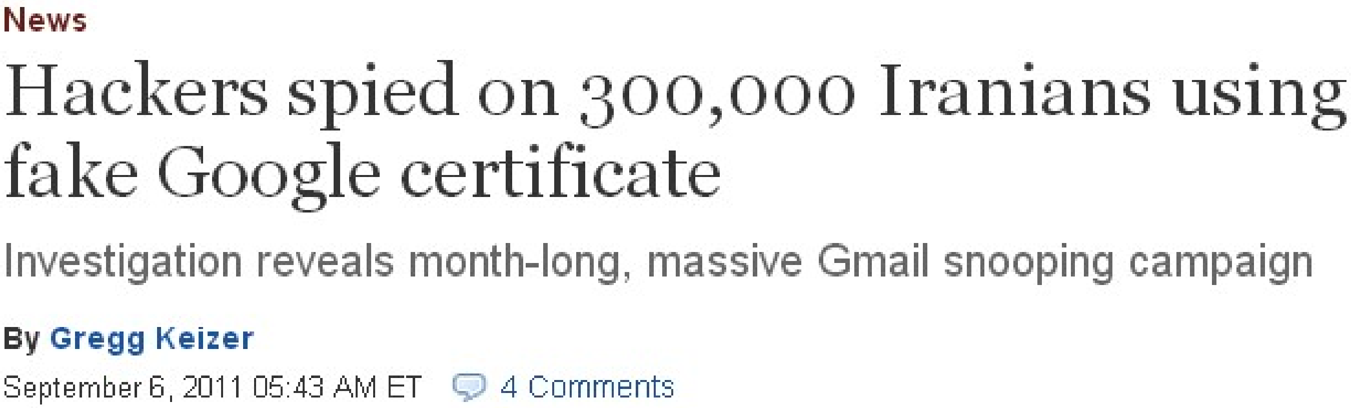
You attempted to reach **www.kia.com**, but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Google Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications. You should not proceed, **especially** if you have never seen this warning before for this site.

Compromised CAs?



Four CAs Have Been Compromised Since June

Posted by **Soulskill** on Friday October 28 2011, @04:08PM
from the four-whole-californias-wow dept.



News

Hackers spied on 300,000 Iranians using fake Google certificate

Investigation reveals month-long, massive Gmail snooping campaign

By Gregg Keizer
September 6, 2011 05:43 AM ET   4 Comments

See Moxie'11

Defenses Against Compromised Certs

- How to Detect If Being Served Bad Cert
  - Certificate Pinning
  - Certificate Revocation
  - Certificate Transparency

- Certificate Pinning
  - Browser “pins” or caches certain certificates after the first visit (e.g. Gmail.com)
  - Issues: How many and which certs to pin?

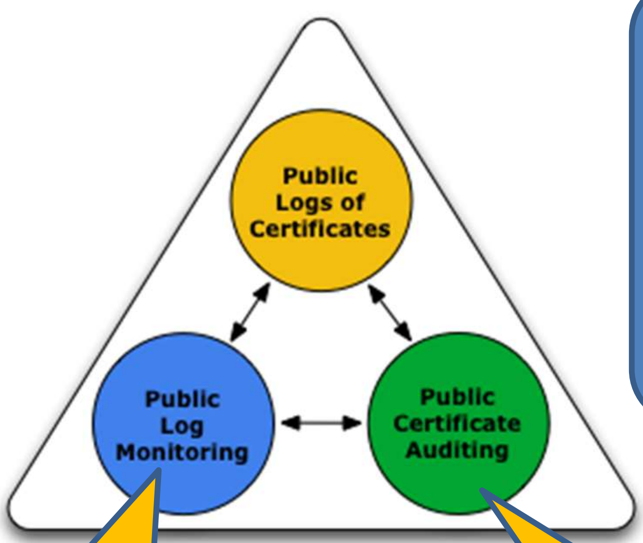
Certificate Revocation

- Idea: CA can revoke compromised certs.
- Supported by OCSP
  - CA signs a revocation list
  - Problems?
    - Time windows after compromise
    - Privacy
    - Implementation bugs (replay attacks)
  - Improvements: OCSP stapling (see Wikipedia)
    - Network costs increase

– PR

A Mitigation for Compromised Certs: Certificate Transparency

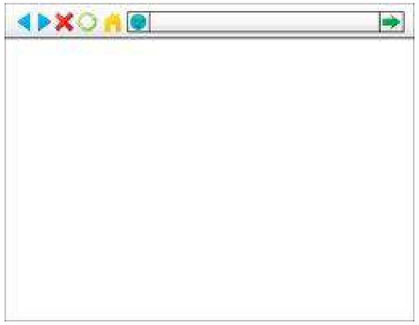
- Idea: Publicly audit all SSL certs.



Public Cert. Log

(Append-only)

Cert. Verify?



Are certificates Safe?

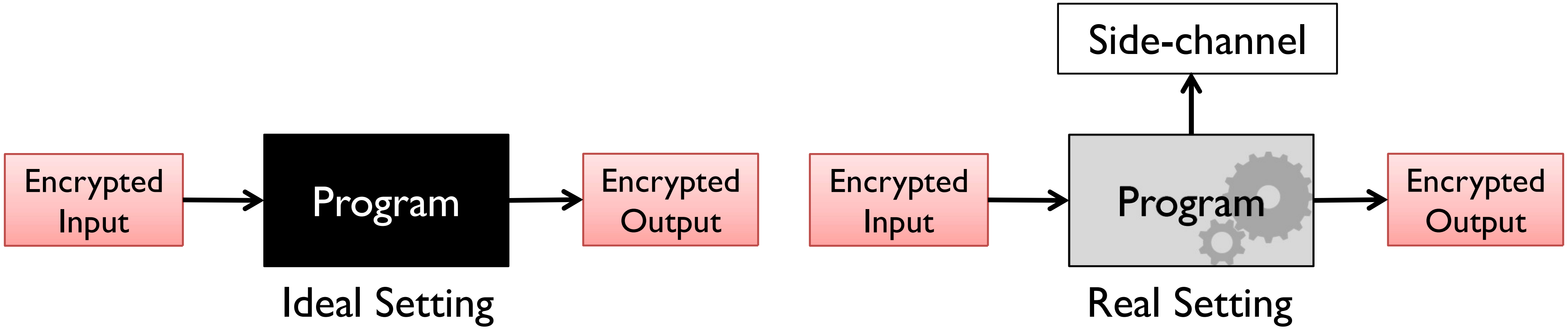
Is the Log Functioning correctly?

Revisit: Assumptions in the threat model

- User is using a secure channel
- Users check browser UI correctly
- Certificate issuers are uncompromised
- **Alice & Bob’s secrets are secure**
- Crypto primitives are secure
- TLS protocol design is secure
- TLS protocol implementation is secure
- Entities are authenticated correctly

Practical Pitfalls in HTTPS:
Side-channel Leakage

What is a side-channel?



Side-channel: *A side-channel is an unintended source of information leakage that is not designed as the primary means of communication*

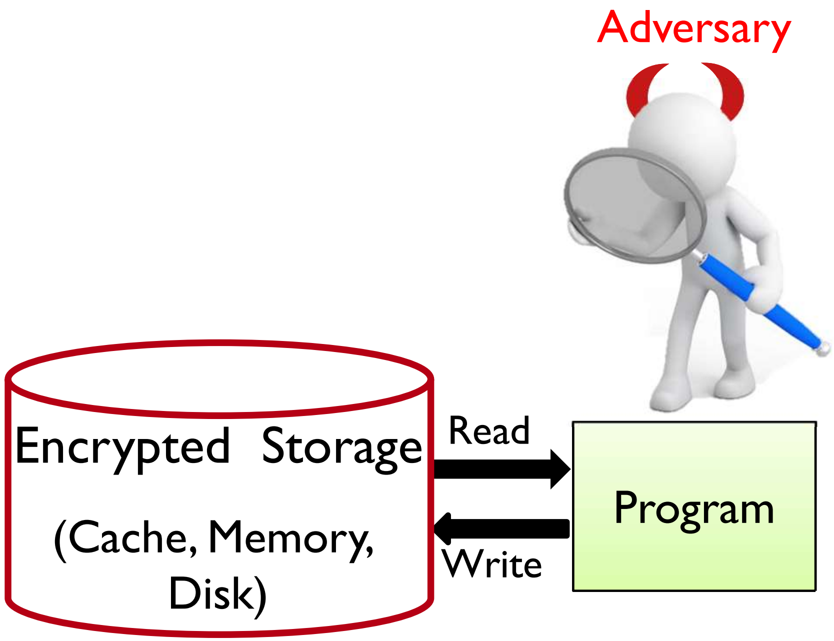
Types of side-channels

Attacker's knowledge in encrypted computation

- Program logic is public

Side-Channels

- Size of data
- Timing Channel
- Data Access Patterns
- Power Channel
- Sound
- Electromagnetic radition



Adversary

Encrypted Storage

(Cache, Memory, Disk)

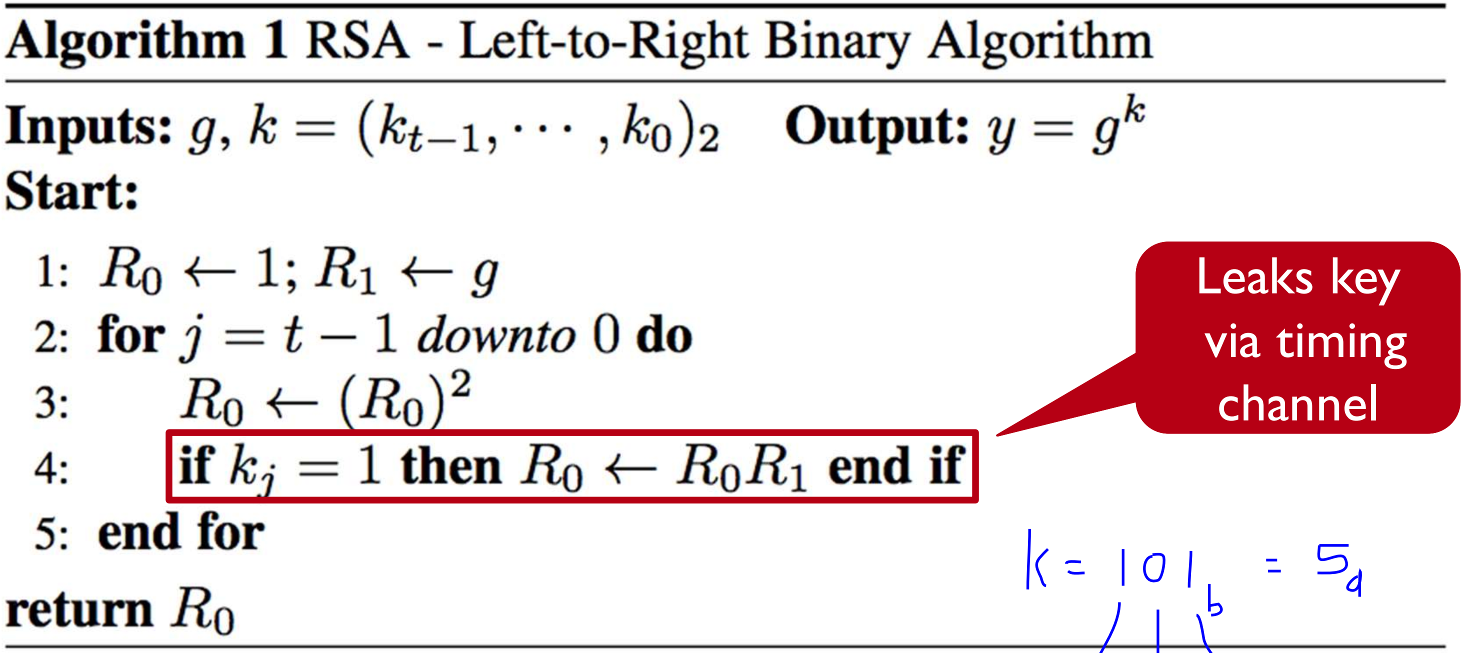
Read

Write

Program

Timing Channel

Cryptographic protocols: Exponentiation is implemented via square-and-multiply
RSA has *y* = *g*^*k* *mod N*



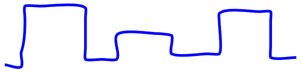
Algorithm 1 RSA - Left-to-Right Binary Algorithm

```
Inputs: $g,\ k = (k_{t-1}, \cdots, k_0)_2$    Output: $y = g^k$
Start:
  $R_0 \leftarrow 1;\ R_1 \leftarrow g$
  for $j = t-1$ downto $0$ do
      $R_0 \leftarrow (R_0)^2$
      if $k_j = 1$ then $R_0 \leftarrow R_0 R_1$ end if
  end for
return $R_0$
```

Leaks key via timing channel

$k = 101_b = 5_d$

$g^4 \times g^0 \times g^1 = g^5$



Fixing the Timing Channel

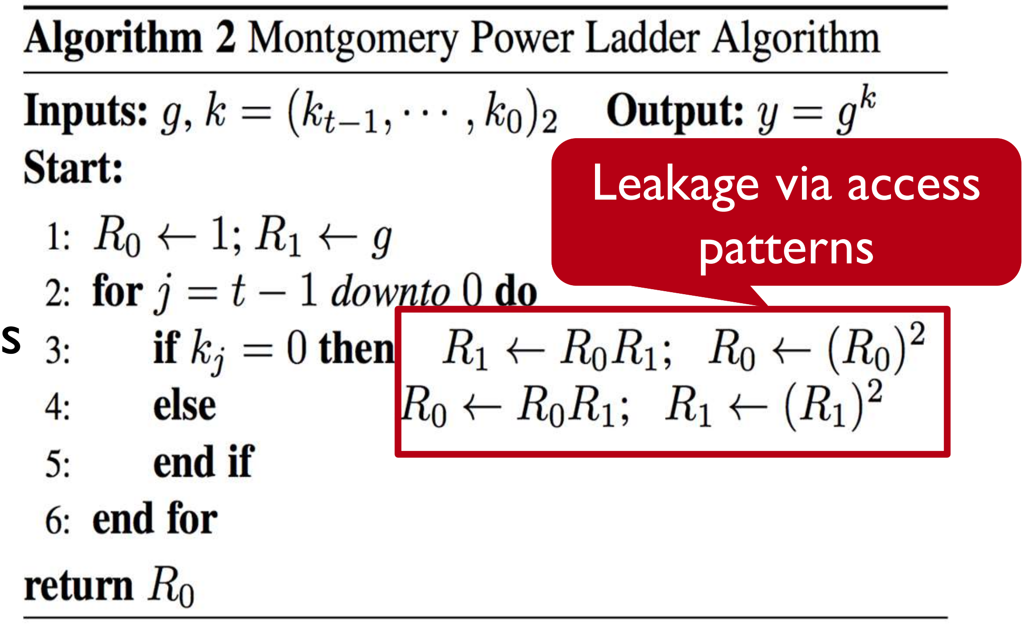
Same computation on both the branches

Is there any other leakage channel?

- YES

Memory access patterns reveal key bits

- Order of accessing R_0 and R_1
- Can be fixed using deterministic address patterns, or randomization

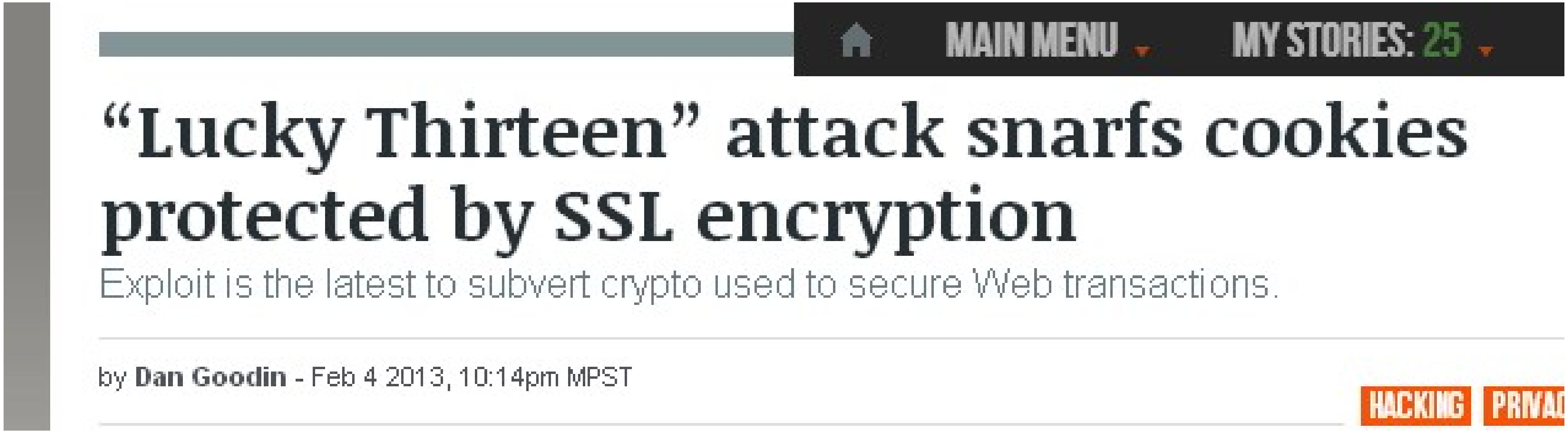


Algorithm 2 Montgomery Power Ladder Algorithm

```
Inputs: $g, k = (k_{t-1}, \cdots, k_0)_2$   Output: $y = g^k$
Start:
  $R_0 \leftarrow 1; R_1 \leftarrow g$
  for $j = t - 1$ downto 0 do
      if $k_j = 0$ then  $R_1 \leftarrow R_0 R_1; \quad R_0 \leftarrow (R_0)^2$
      else               $R_0 \leftarrow R_0 R_1; \quad R_1 \leftarrow (R_1)^2$
      end if
  end for
return $R_0$
```

Leakage via access patterns

Side-Channels Flaws: Timing



“Lucky Thirteen” attack snarfs cookies protected by SSL encryption

Exploit is the latest to subvert crypto used to secure Web transactions.

by **Dan Goodin** - Feb 4 2013, 10:14pm MPST

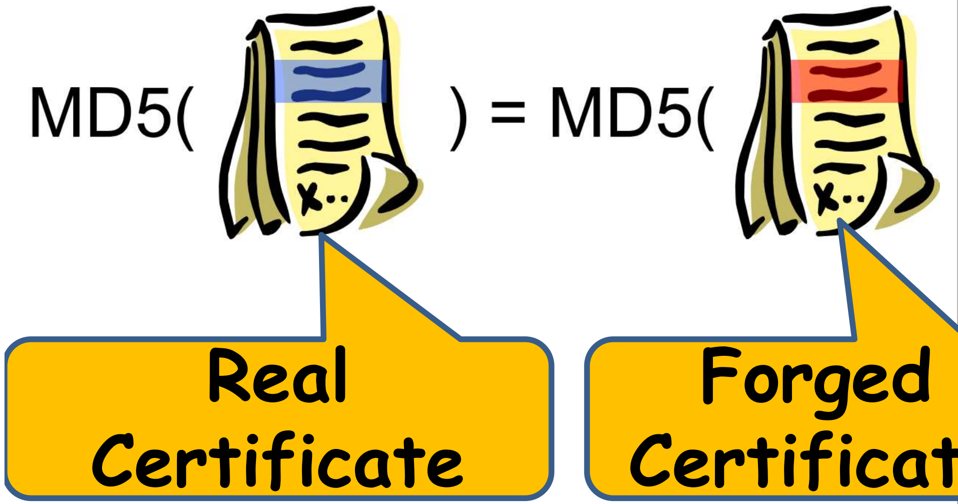
Revisit:
Assumptions in the threat model

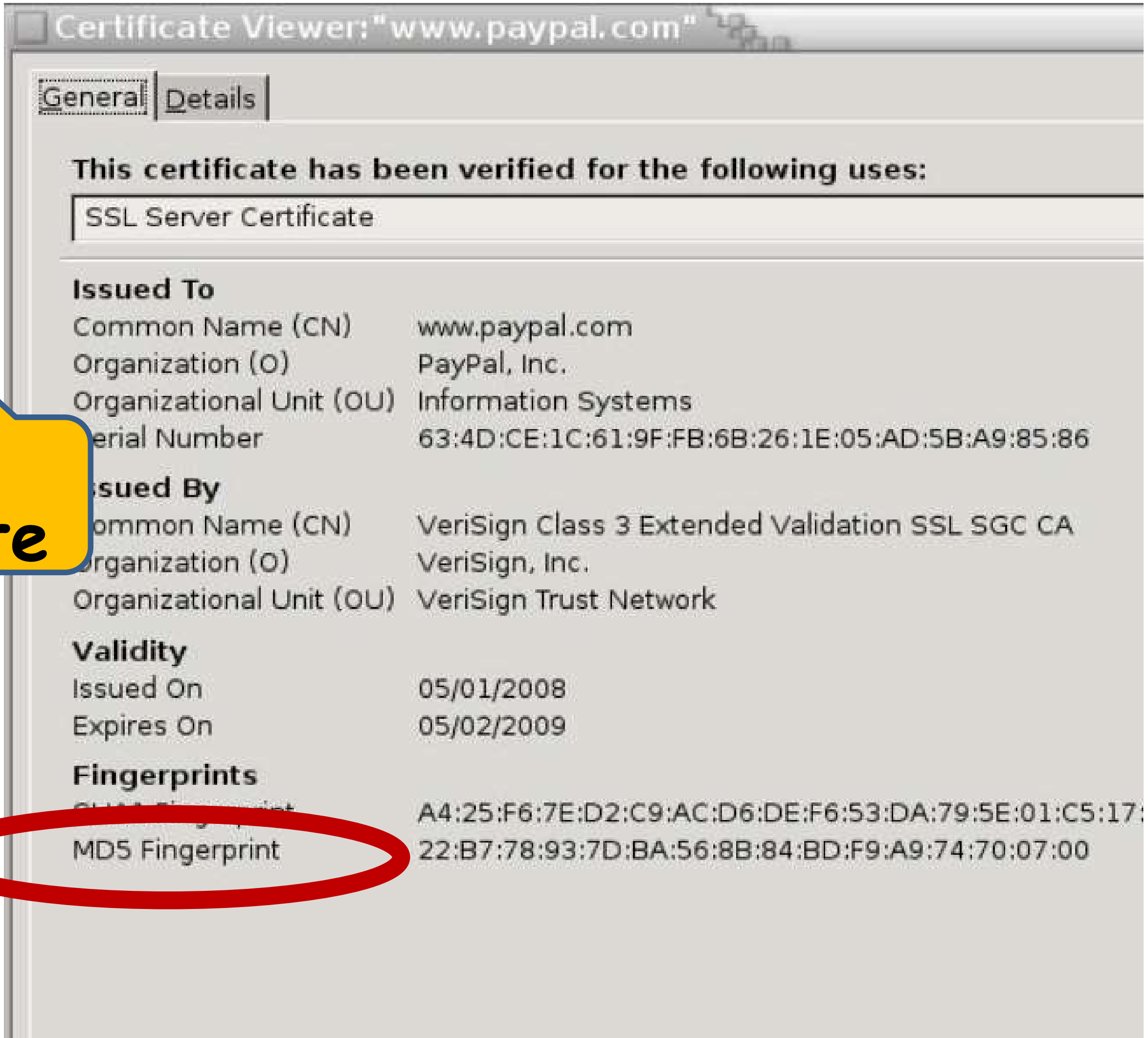
- User is using a secure channel
- Users check browser UI correctly
- Certificate issuers are uncompromised
- Alice & Bob’s secrets are secure
- **Crypto primitives are secure**
- **TLS protocol design is secure**
- **TLS protocol implementation is secure**
- Entities are authenticated correctly

Practical Pitfalls in HTTPS:
Cryptographic Implementation Errors

Broken Crypto Primitives: Broken MD5 leads to Forged Certs.

- Can attack the cryptographic signing [Sotirov et al.]
- MD5 is a broken hash: can have collisions [2004, 2007]





This certificate has been verified for the following uses:

SSL Server Certificate

Issued To

Common Name (CN)	www.paypal.com
Organization (O)	PayPal, Inc.
Organizational Unit (OU)	Information Systems
erial Number	63:4D:CE:1C:61:9F:FB:6B:26:1E:05:AD:5B:A9:85:86

sued By

ommon Name (CN)	VeriSign Class 3 Extended Validation SSL SGC CA
rganization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network

Validity

Issued On	05/01/2008
Expires On	05/02/2009

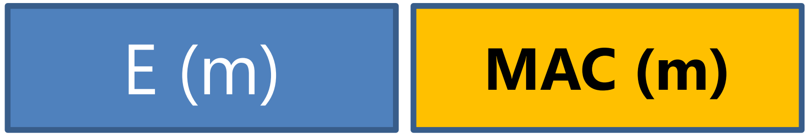
Fingerprints

SHA1 Fingerprint	A4:25:F6:7E:D2:C9:AC:D6:DE:F6:53:DA:79:5E:01:C5:17:
MD5 Fingerprint	22:B7:78:93:7D:BA:56:8B:84:BD:F9:A9:74:70:07:00

See Sotirov'09

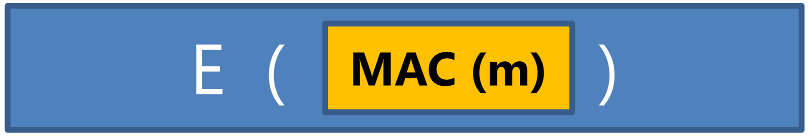
Improper Use of Crypto Primitives

- MAC => integrity, Enc => confidentiality
- Which of these is a secure MAC+Enc scheme?

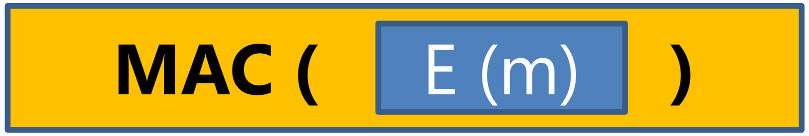

E (m)

MAC (m)

- Example 1: SSH
  - Encrypt – **and**- MAC
  - Clearly Insecure! (Why?)


E (**MAC (m)**)

- Example 2: SSL (Used in HTTPS)
  - **MAC – then – encrypt**
  - Can be insecure
  - Encryption is malleable! (See later)


MAC (E (m) **)**

- Example 3: IPSec
  - **Encrypt - then - MAC**
  - Provably Secure

Other Crypto Implementation Flaws

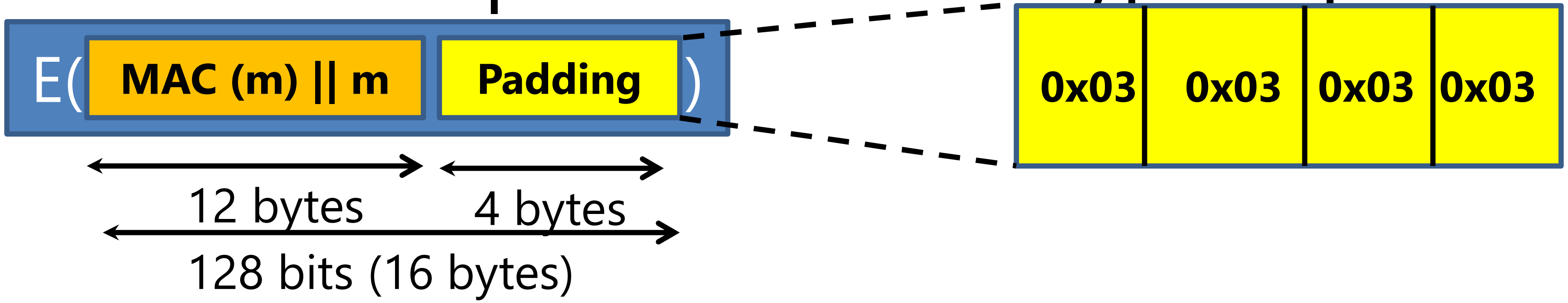
- Timing side-channels:
  - Vulnerable RSA PKCS#1 v1.5 **[1998]**
  - Compression
    - CRIME **[2012]**, new one – BREACH **[Aug 2013]**

- Renegotiation attacks **[Rescola 2009]**
- IV in CBC mode incorrect => BEAST **[2011]**
- Uses RC4 (no padding needed)
  - RC4 is totally broken! (gives biased stream)
- Browsers treat SSL errors lightly! [CS5331]
- Dual EC in ANSI, ISO standard has backdoors
- Replay Attacks in WPA2 [CCS 2017]

- **[Optional]** Reading: Slides from Vitaly Shmatikov
- **[Optional]** Reading: Analysis of the SSL 3.0 protocol

Improper Use of Crypto in SSL: Vaudenay's – Padding Oracle Attack

- Older SSL implemented encrypted packets as:



```
If (Dec(C) == OK-PAD)
{
    P = Plaintext(C);
    if (CheckPad(P)) {
        send ("BAD-PAD"); exit();
    }
    if (MAC (RemovePad(P)) !=MacTag))
    {send("BAD-MAC"); exit(); }
}
```

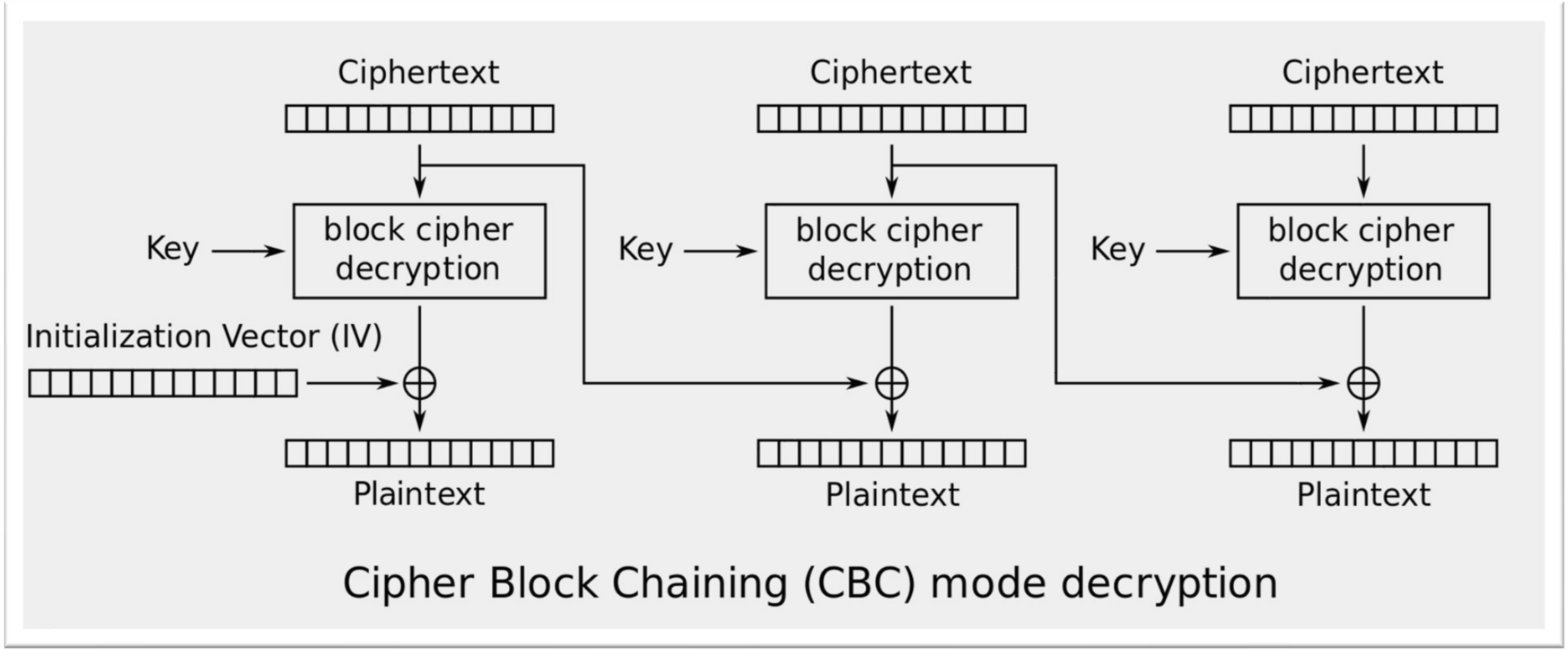
Spot the Security Bug with this code

Hint: This line…

Can distinguish BAD-MAC and BAD-PAD

Improper Use of Crypto Operations: Vaudenay's – Padding Oracle Attack

- Observe: CBC-mode encryption is malleable!



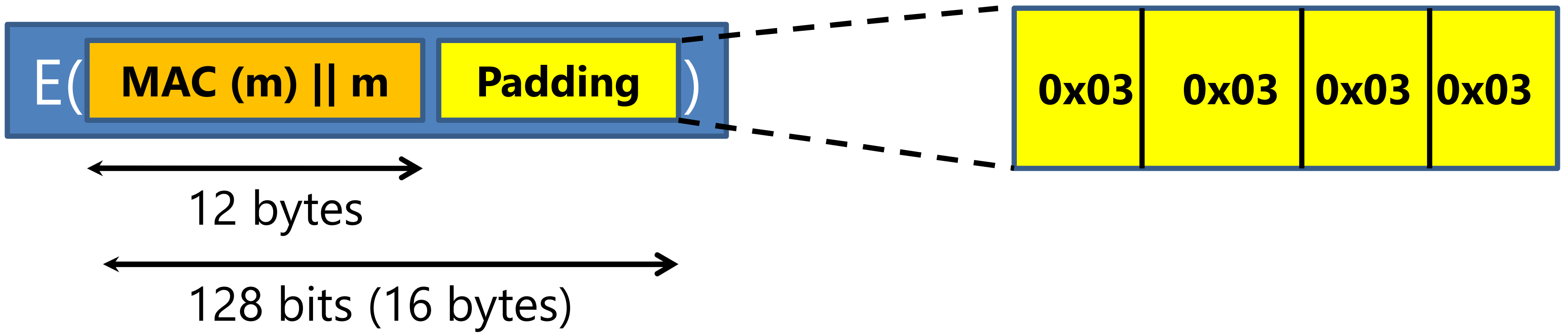
$$P_i = D_K(C_i) \oplus C_{i-1},$$
$$C_0 = IV.$$

A single bit flip of IV will cause the decrypted plaintext P1 to be single bit flipped.

Reference: Padding Oracle Attacks (Wikipedia)

Improper Use of Crypto in SSL: Vaudenay's – Padding Oracle Attack

- Older SSL implemented encrypted packets as:



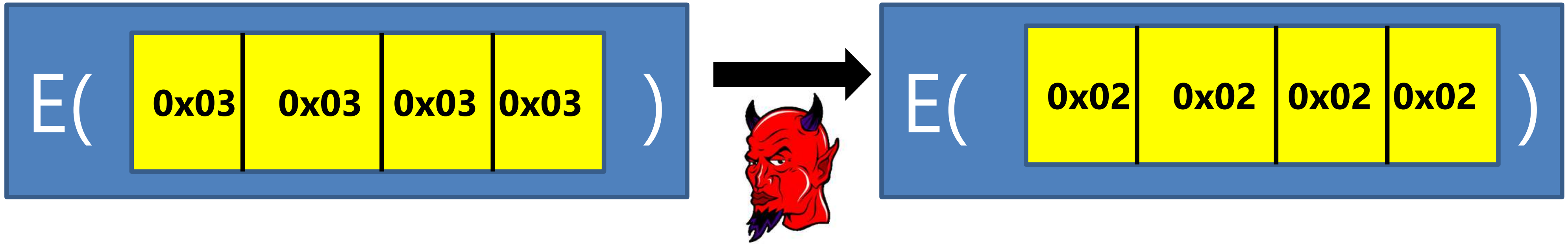
E(MAC (m) || m Padding)

0x03 0x03 0x03 0x03

12 bytes

128 bits (16 bytes)

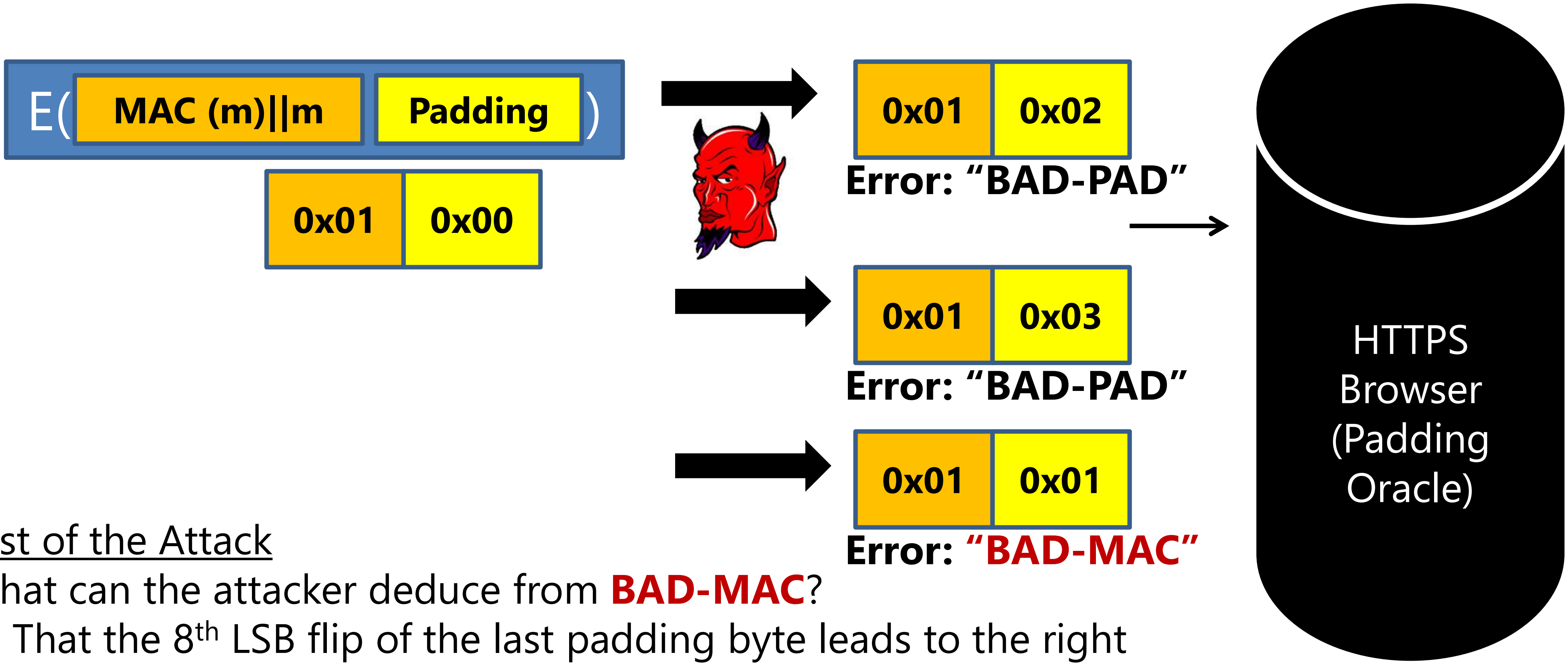
- CBC-mode encryption is malleable, so:
  - The attacker can bit-flip the padding ciphertext



E(0x03 0x03 0x03 0x03)

E(0x02 0x02 0x02 0x02)

SSL Protocol Design Flaws: Vaudenay's attack [2002]



E(MAC (m)||m Padding)

0x01 0x00

0x01 0x02

Error: “BAD-PAD”

0x01 0x03

Error: “BAD-PAD”

0x01 0x01

Error: **“BAD-MAC”**

HTTPS Browser (Padding Oracle)

Gist of the Attack

What can the attacker deduce from **BAD-MAC**?

- That the 8th LSB flip of the last padding byte leads to the right value of last byte of the MAC tag!

How many correct values are there for the last padding byte?

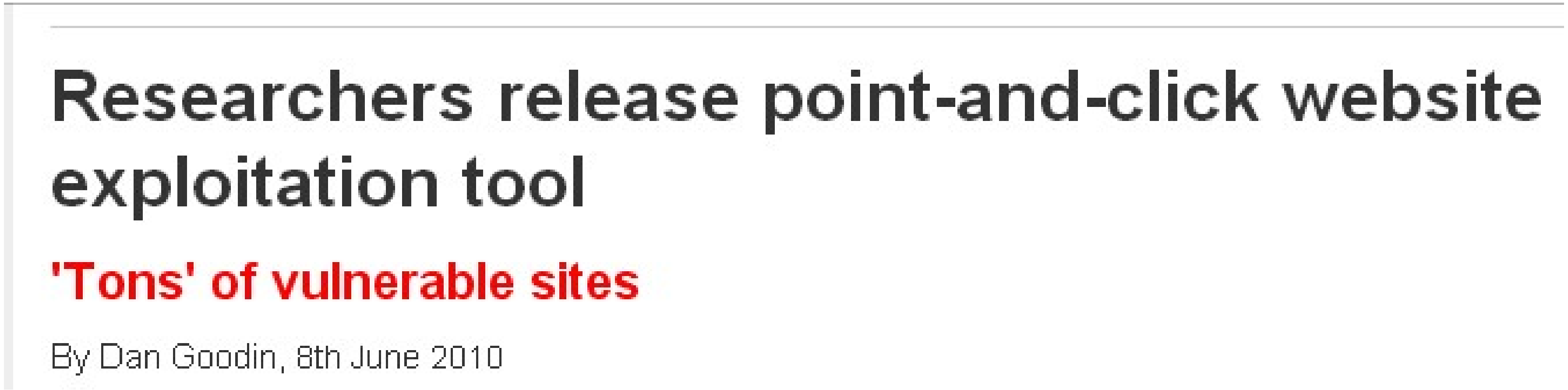
- 0 to 15!

(The full attack @ the provided link!)

Reference: [Padding Oracle Attacks (Katz - Youtube)]

SSL Protocol Design Flaws: Vaudenay's attack [2002]

- POET toolkit [2010]



Researchers release point-and-click website exploitation tool

'Tons' of vulnerable sites

By Dan Goodin, 8th June 2010

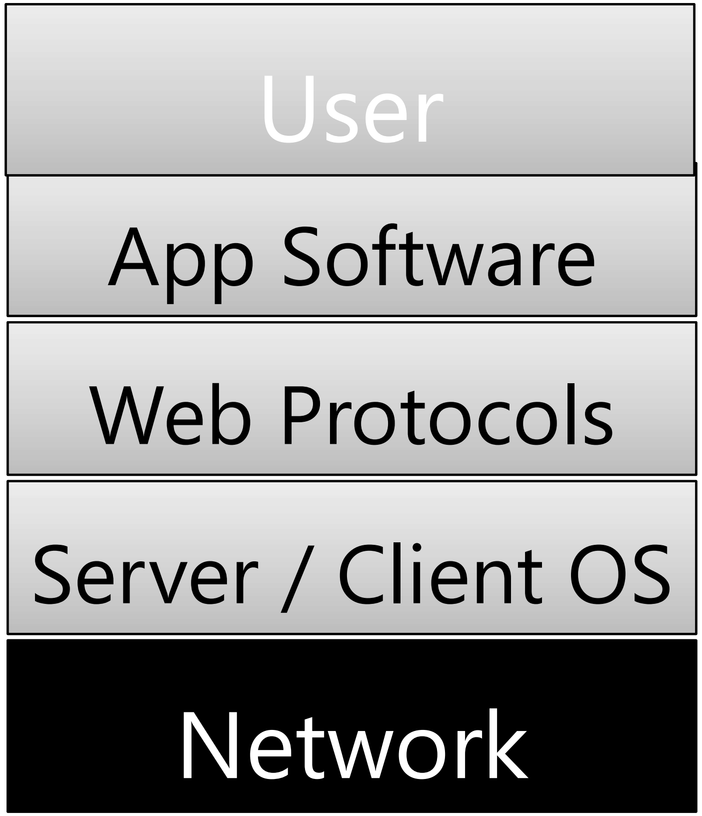
- More Padding Oracle Attacks
  - Bleichenbacher's Attacks [1998]
    - In RSA PKCS# 1 v1.5 (used in TLS)

Summary of HTTPS Failures

- HTTPS errors
- Crypto Usage & Implementation Flaws
- Side-Channel Attacks
- UI Hijacking and Confusion
- Compromised CA

- **[Optional]** Reading: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements

How do Secure Channels Fail?



User

App Software

Web Protocols

Server / Client OS

Network

- In practice, they fail in 2 ways:
  - Attack the assumptions
  - Violate other security properties that are not captured by the threat model
    - HTTPS only provides "CIA" for network (not availability)
    - Attack other layers!

Important Principles:

(1) State threat model, else there's no security argument!
(2) Assumptions can fail, but that's not a flawed argument
(3) Choose reasonable assumptions in your threat model