Communication

Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

Fragment Identifier Messaging

Send information by navigating a frame

- http://gadget.com/**#hello**

Navigating to fragment doesn't reload frame

- No network traffic, but frame can read its fragment

Not a secure channel

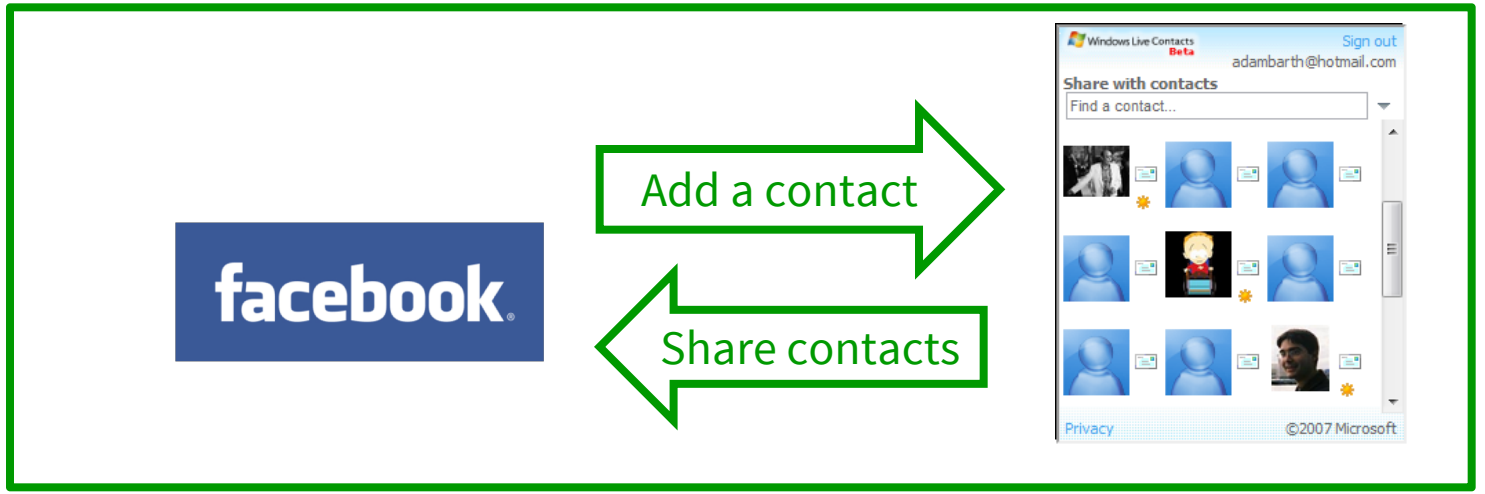
- Confidentiality ✓
- Integrity ✓
- Authentication ✗

window.postMessage

API for inter-frame communication

- Supported in current browsers

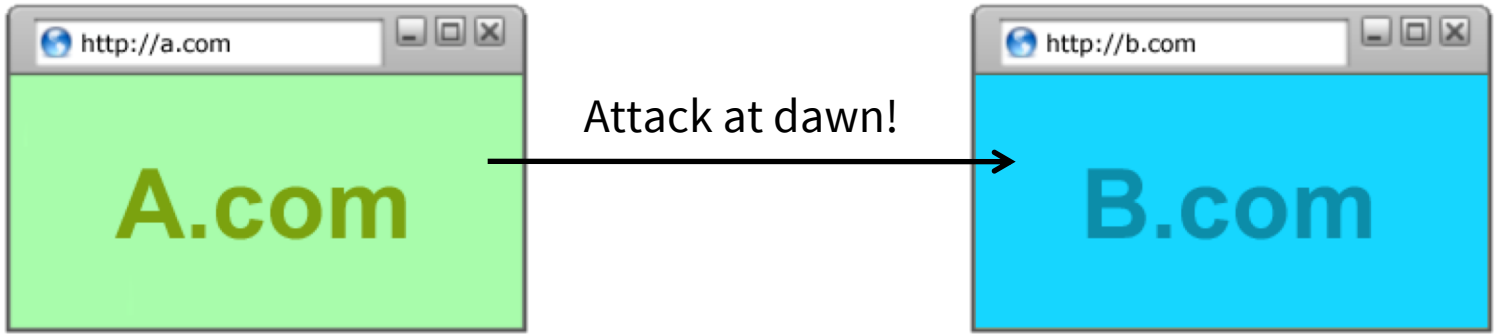
- A network-like channel between frames



postMessage syntax

```
frames[0].postMessage("Attack at dawn!",
                      "http://b.com/");
```

```
window.addEventListener("message", function (e) {
  if (e.origin == "http://a.com") {
    ... e.data ... }
}, false);
```



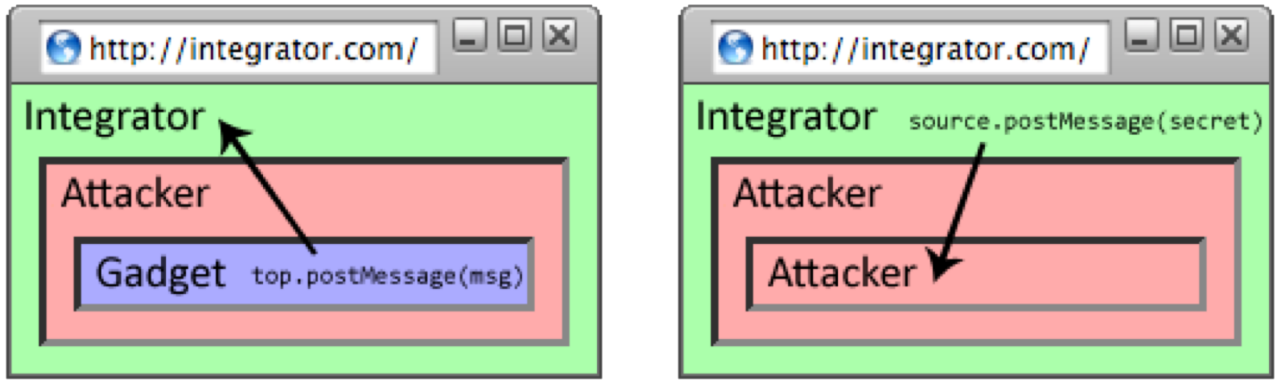
Why include “targetOrigin”?

What goes wrong?

```
frames[0].postMessage("Attack at dawn!");
```

Messages sent to *frames*, not principals

- When would this happen?



Two-way communication

A method call is associated with a response

Can build this on top of postMessage

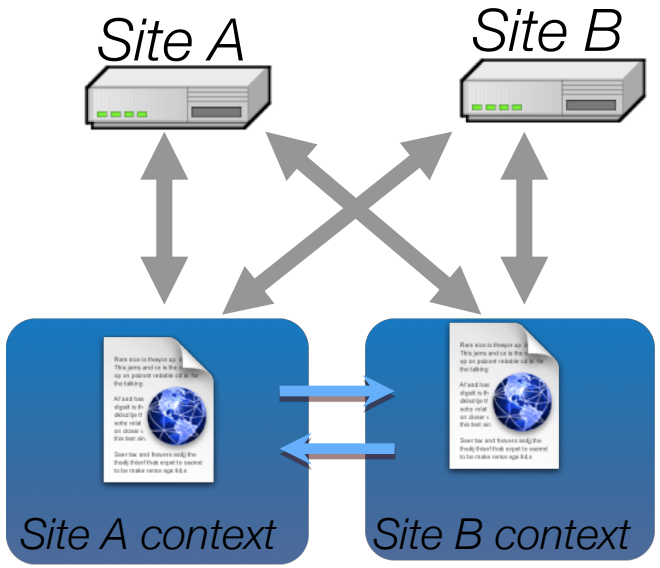
- Messenger: Each time you call a method in the iframe, you pass a reply function that is called with the results of that method call.

jQuery postMessage plugin

Wraps the postMessage API and simplifies its usage.

Works in browsers that do not support postMessage method by using fragment navigation (hash portion of the url)

Network communication



- Cross-origin network requests
  - Access-Control-Allow-Origin: <list of domains>
  - Access-Control-Allow-Origin: *

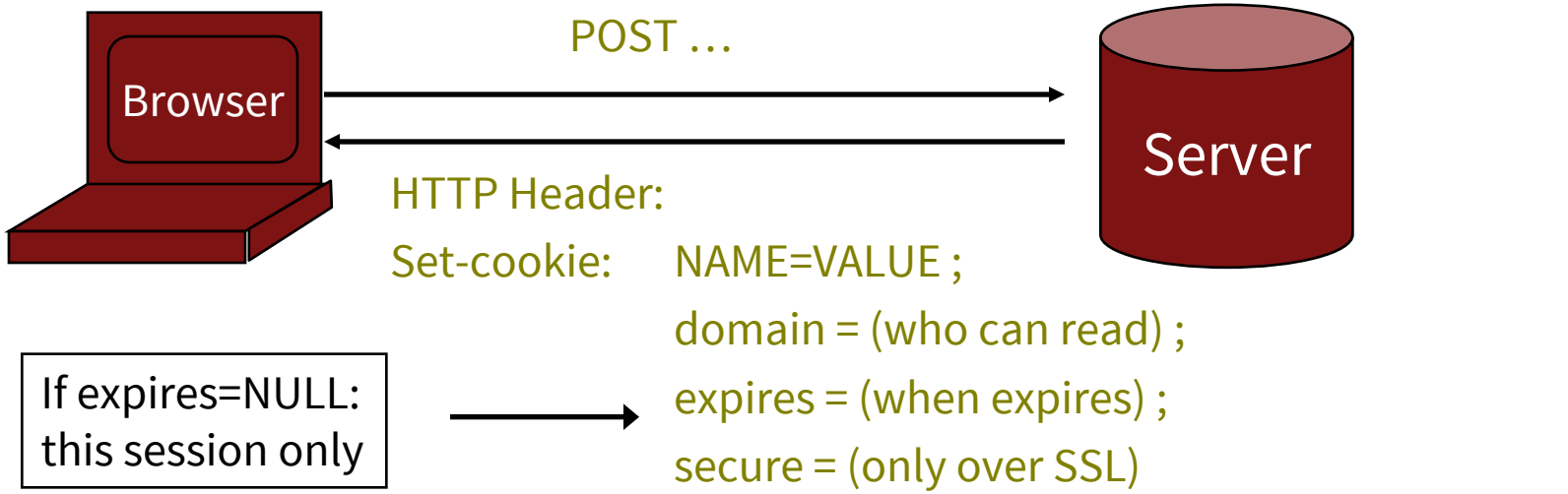
Client State

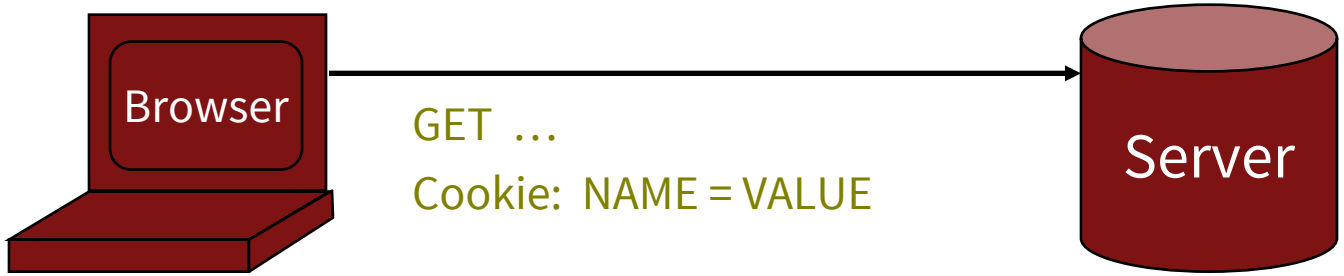
Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

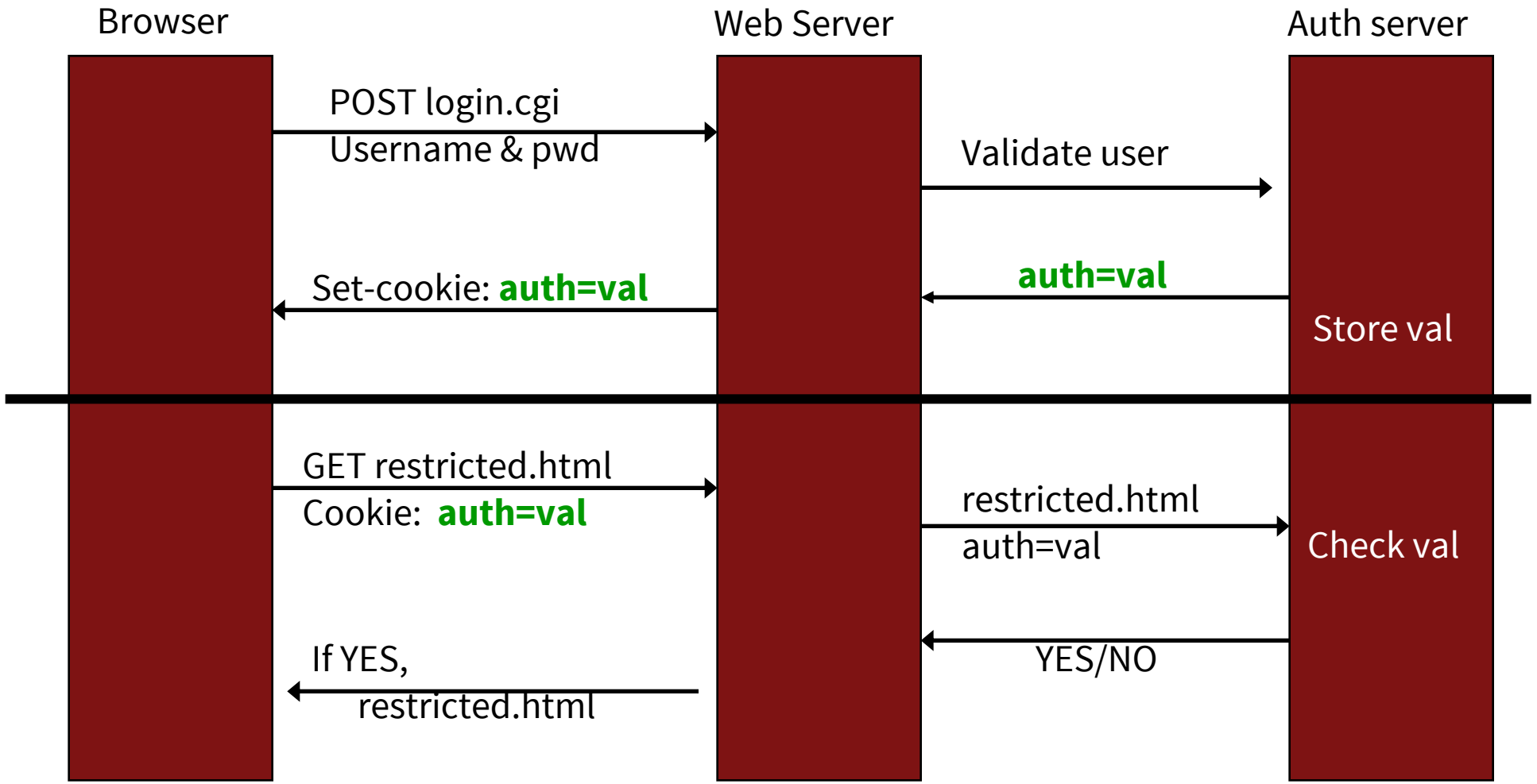
Cookies

Used to store state on user's machine





Cookie authentication



Cookie Security Policy

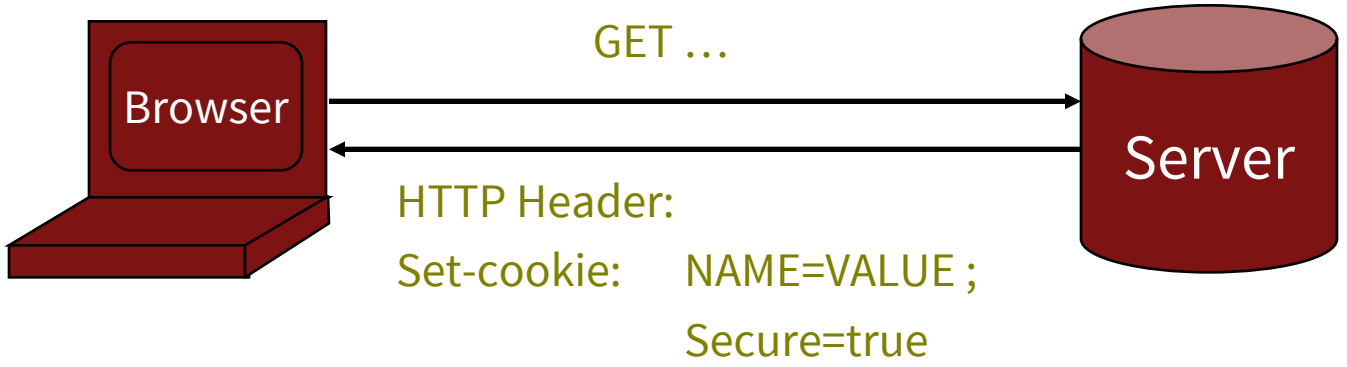
Uses:

- User authentication
- Personalization
- User tracking: e.g. Doubleclick (3rd party cookies)

Origin is the tuple **<domain, path>**

- Can set cookies valid across a domain suffix
- Complicated and implementation-specific rules for selecting cookie values, when many cookies apply

Secure Cookies



- Provides confidentiality against network attacker
  - Browser will only send cookie back over HTTPS

- … but no integrity
  - Can rewrite secure cookies over HTTP
    - ⇒ network attacker can rewrite secure cookies
    - ⇒ can log user into attacker's account

httpOnly Cookies

- Cookie sent over HTTP(s), but not accessible to scripts
  - cannot be read via document.cookie
  - Helps prevent cookie theft via XSS

… but does not stop most other risks of XSS bugs

HTML5 Local Storage

Based on named key/value pairs

- Store data based on a named key (a string)
- Retrieve that data with the same key
- Data can be any type supported by JavaScript
  - Including strings, Booleans, integers, floats
  - But data is actually stored as a string

    Need to use functions like parseInt() or parseFloat() to coerce your retrieved data into the expected JavaScript datatype

Some browsers also implement Web SQL Database

- Other forms of local storage would also be useful

Example

Save a sentence in Local Storage :

```
localStorage.setItem(1,'This is a sample sentence');
```

Retrieve it:

```
var data = localStorage.getItem(1);
```

Local Storage supports length, removeItem() and clear().

Security issues

Storage per origin

- Origin is: scheme, host, port

Could be accessed by user with local access (varies by browser)

Can be accessed by JavaScript in page

- no httpOnly so vulnerable to XSS attacks

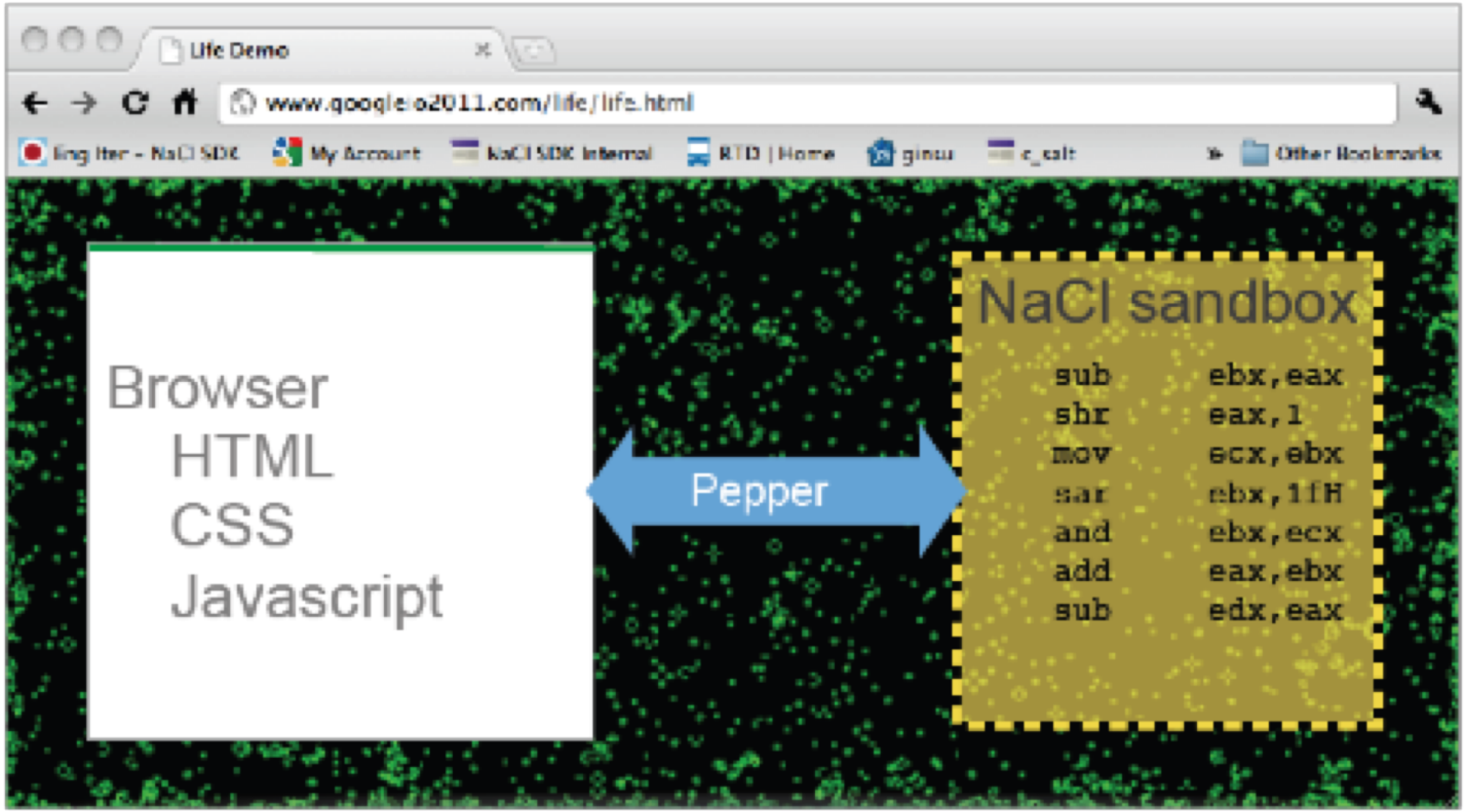
XSS attacks can read local storage

- Do not store sensitive information

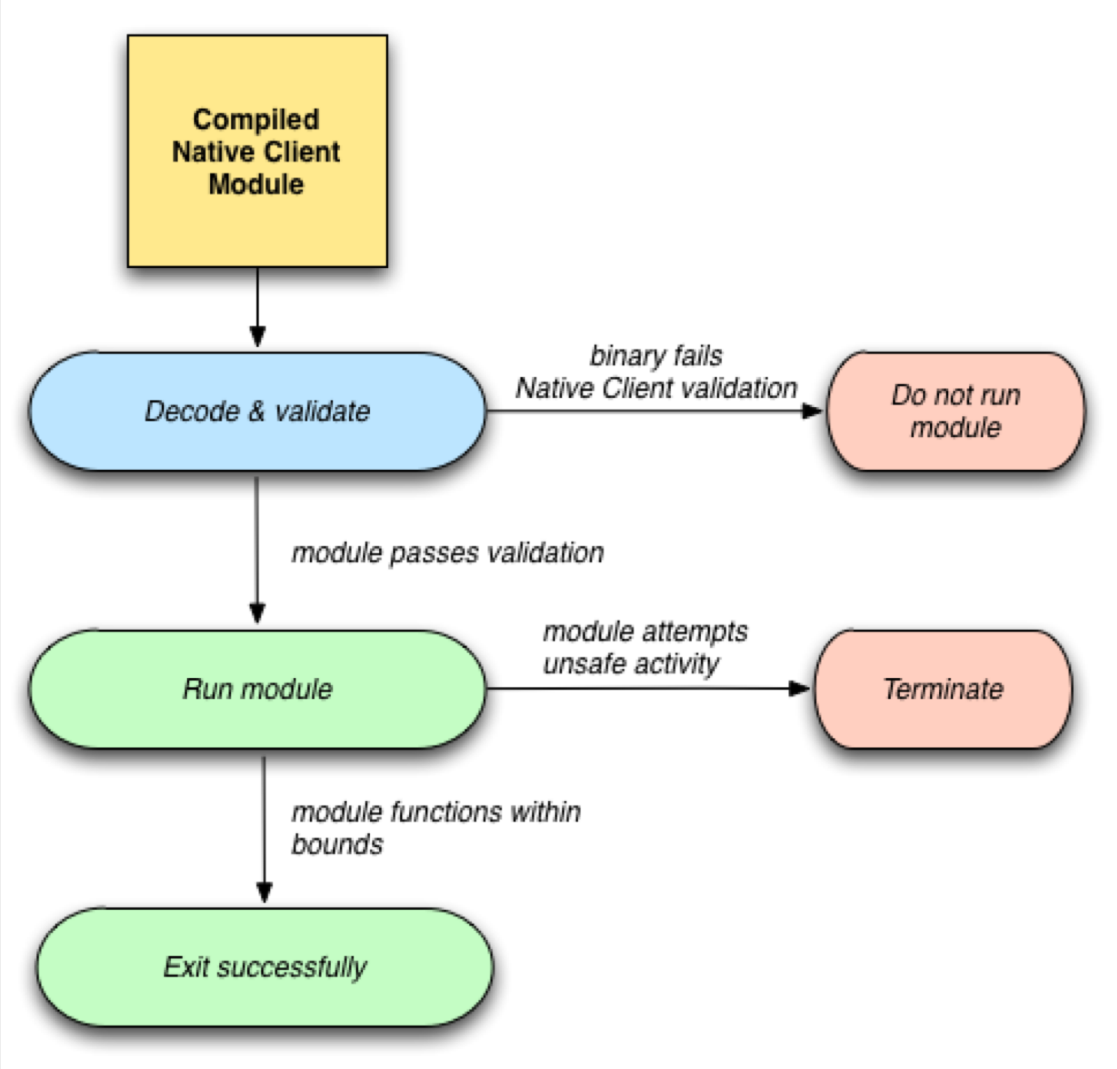
XSS attacks can write local storage

- Do not trust data read from local storage

Native Client



Sandboxed native code



Sandbox techniques

Static analysis

- No loads or stores permitted outside the data sandbox
  - Enforced by operating system protection mechanisms
- No unsafe instructions
  - Examples: syscall, int, and lds.
- Control flow integrity
  - All direct, indirect branches target a safe instruction

Dynamic monitoring

- Native Client runtime mediates system calls

Click-Jacking

Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

Clickjacking

Attacker overlays multiple transparent or opaque frames to trick a user into clicking on a button or link on another page

Clicks meant for the visible page are hijacked and routed to another, invisible page

Clickjacking in the Wild

Google search for “clickjacking” returns 342,000 results… this is not a hypothetical threat!

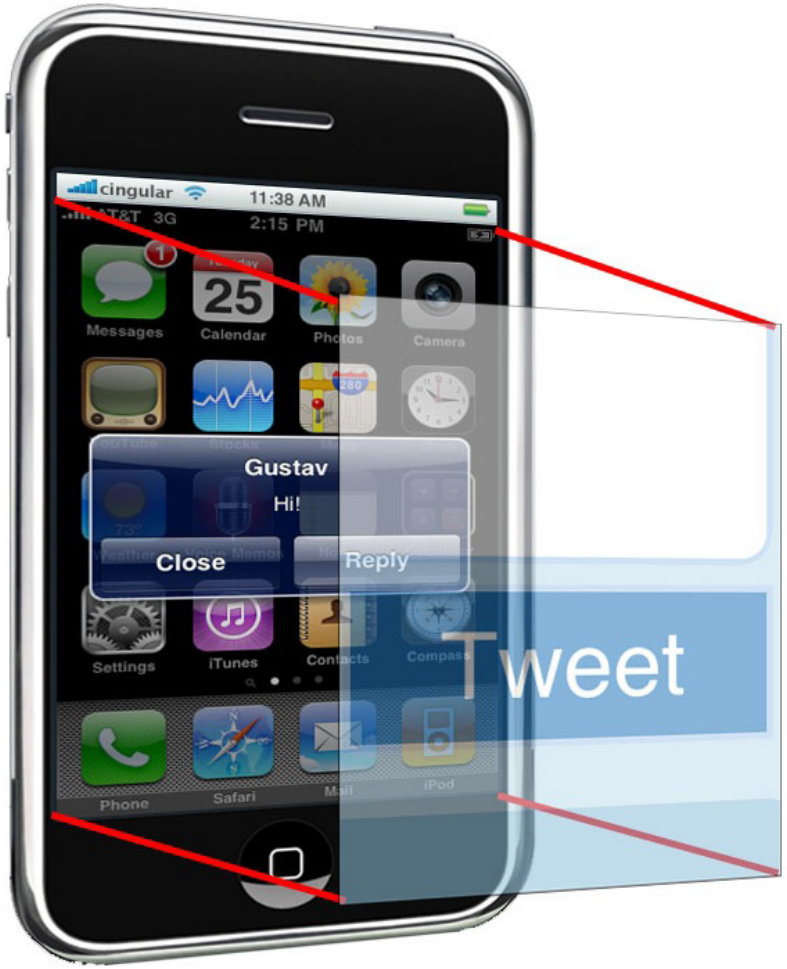
Summer 2010: Facebook worm superimposes an invisible iframe over the entire page that links back to the victim's Facebook page

- If victim is logged in, automatically recommends link to new friends as soon as the page is clicked on

Many clickjacking attacks against Twitter

- Users send out tweets against their will

Tap-jacking


cingular
11:38 AM
AT&T 3G
2:15 PM
Messages
Calendar
Photos
Camera
Gustav
Hi!
Close
Reply
Settings
iTunes
Contacts
Compass
Tweet
Phone
Safari
Mail
iPod

Frame Busting

Module 2: Web Background and the Browser Security Model

Stanford | Stanford Center for Professional Development

Frames

Embed HTML documents in other documents

```
<iframe name="myframe"
      src="http://www.google.com/">
          This text is ignored by most
browsers.
</iframe>
```

Frame Busting

Goal: prevent web page from loading in a frame

- example: opening login page in a frame will display correct passmark image

Frame busting:

```
if   (top != self)
   top.location.href = location.href
```

Better Frame Busting

Problem: **Javascript OnUnload event**

```
<body onUnload="javascript: cause_an_abort;)">
```

Try this instead:

```
if   (top != self)
   top.location.href = location.href
else {  …  code of page here …}
```

Summary

- Web security goals and threat models
- HTTP
- Rendering: Html, DOM, embedded content, JavaScript
- Isolation: frames, same-origin policy, HTML5 sandboxing
- Communication: fragment, post-message, cross-origin request
- Frame navigation: Same-origin policy, descendant policy
- Client storage: Cookies, Local storage, Native Client
- Click-jacking, tap-jacking, frame busting

MODULE 3

TLS 1.3

Compression attacks

Password Breaches

Certificates on the Web

Abusing Mobile Sensors

Topics for this section

TLS attacks and defenses:

Compression attacks: CRIME and BREACH, TLS 1.3

Password breaches and 2nd factor authentication

Certificate Authorities: compromises, Lets encrypt, universal TLS

New hardware security support: Intel SGX

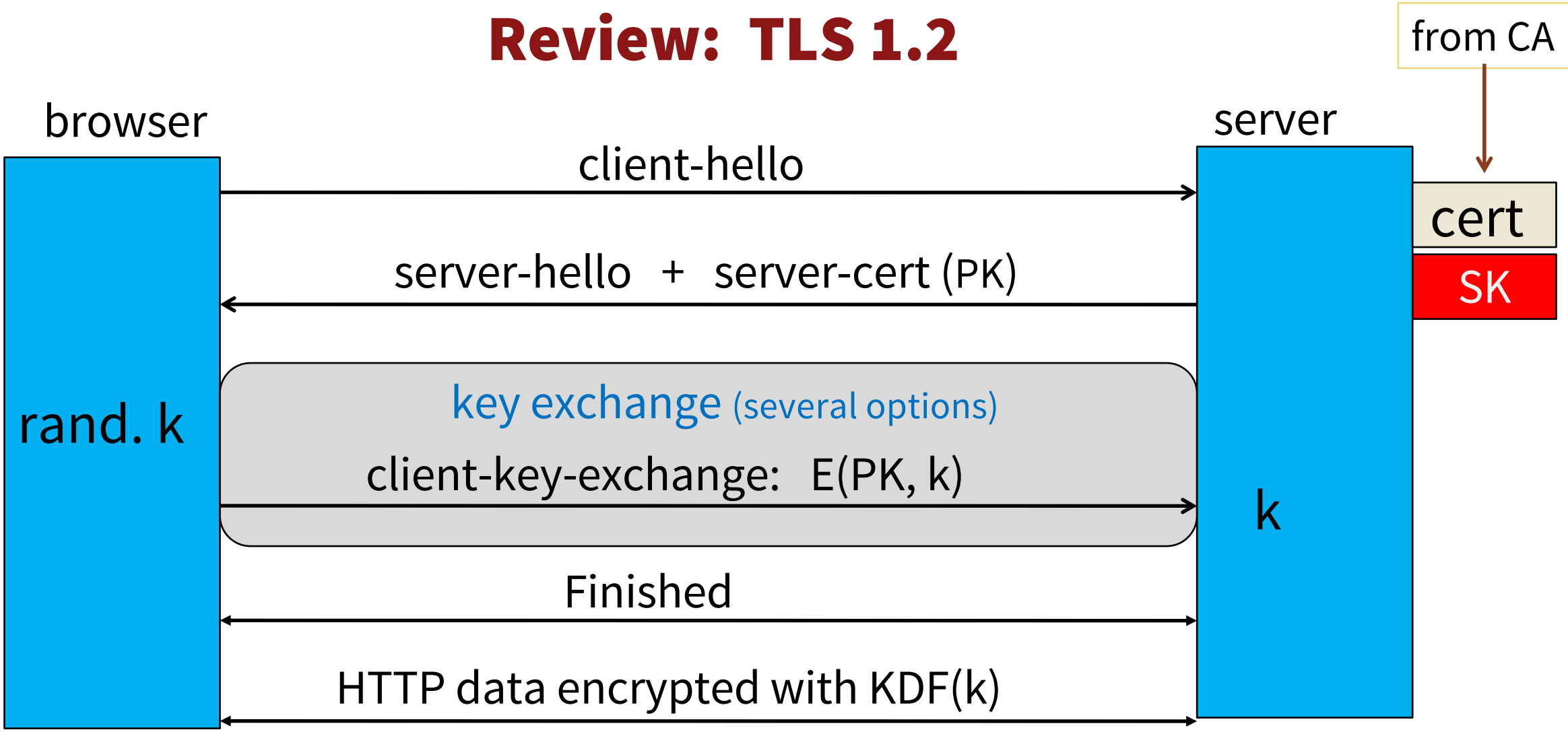
Sensor abuse on mobile phones

TLS 1.3

Module 3: Attacks and Defenses

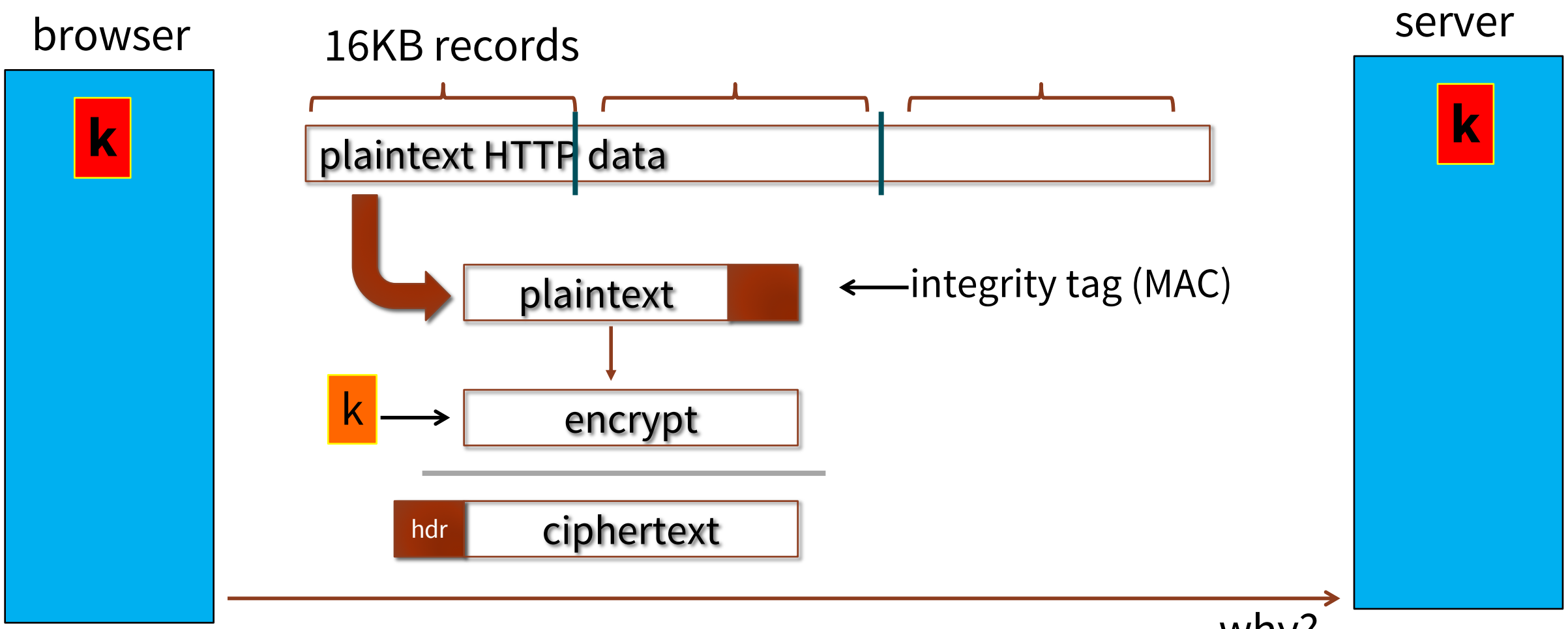
Stanford | Stanford Center for Professional Development

Review: TLS 1.2



most common: server authentication only

Review: TLS record encryption (original design)



encryption method is called **MAC-then-encrypt** : why?

the reason for many attacks on TLS (BEAST, Lucky13, POODLE, …)

TLS 1.3: a new version of TLS (2017)

Record encryption:

- mandatory method: **AES128-GCM**

  fast on x86 (AES-NI) : Intel Skylake, 0.68 cycles/byte

- On weaker processors: **CHACHA20_POLY1305**

  fast in software

Both methods provide authenticated encryption

TLS 1.3: a new version of TLS (2017)

Session setup:

- **Forward secrecy required** (non-forward secure method is deprecated)
- **Zero round-trip setup option**:
  client can send encrypted data on first flow (after client-hello)
- **Server certificate is encrypted** (previously, sent in the clear)
  stronger privacy when server has multiple certificates
- **Initiate TLS session from a pre-shared secret**, if one exists
  more general than session-resume in TLS 1.2

Compression Attacks

Module 3: Attacks and Defenses

Stanford | Stanford Center for Professional Development

Compression and Encryption

Strong desire to combine compression and encryption

How?

Option 1: first encrypt and then compress

Does not work … ciphertext looks like a random string

Compression and Encryption

Option 2: first compress and then encrypt

Used in many Internet protocols (TLS, HTTPS, QUIC, …)

POST /bank.com/buy?id=aapl
Cookie: uid=JhPL8g69684rksfsdg

Recall in TLS: 16KB records

Support for compression before encryption

Trouble … [Kelsey'02]

Compress-then-encrypt reveals information:

POST /bank.com/buy?id=aapl
Cookie: uid=JhPL8g69684rksfsdg

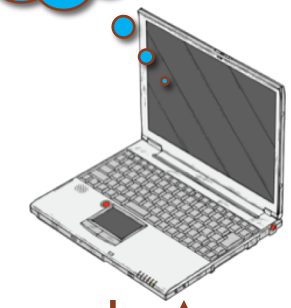
POST /bank.com/buy?id=g**oo**g
C**oo**kie: uid=JhPL8g69684rksfsdg

Second message compresses better than first:

network observer can distinguish the two messages!

Even worse: the CRIME attack [RD'2012]

(simplified)



POST /bank.com/buy?id=aapl

Cookie: **uid=JhPL8g69684rksfsdg**

Host: bank.com

Javascript

Goal: steal user's bank cookie

Javascript can issue requests to Bank, but cannot read Cookie value

Even worse: the CRIME attack [RD'2012]

(simplified)

16KB

POST /bank.com/buy?**uid=A11111**…
Cookie: **uid=**J hPL8g69684rksfsdg
Host: bank.com

Observe ciphertext size

Even worse: the CRIME attack [RD'2012]

(simplified)

16KB

POST /bank.com/buy?**uid=B11111**…
Cookie: **uid=**J hPL8g69684rksfsdg
Host: bank.com

Observe ciphertext size

Even worse: the CRIME attack [RD'2012]

(simplified)

16KB

POST /bank.com/buy?**uid=J11111**…
Cookie: **uid=J** hPL8g69684rksfsdg
Host: bank.com

Ciphertext size is slightly shorter

⇒ first character of Cookie is "J"

Even worse: the CRIME attack [RD'2012]

(simplified)

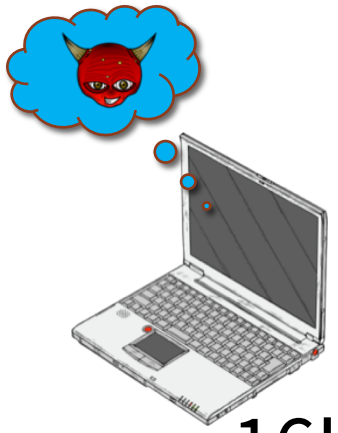
16KB

POST /bank.com/buy?**uid=Ja1111**…
Cookie: **uid=J**h PL8g69684rksfsdg
Host: bank.com

Observe ciphertext size

Even worse: the CRIME attack [RD'2012]

(simplified)



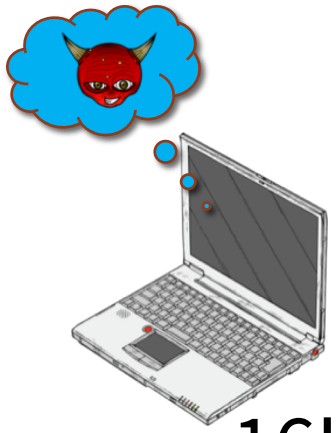
16KB

POST /bank.com/buy?**uid=Jh1111**…
Cookie: **uid=Jh** PL8g69684rksfsdg
Host: bank.com

Ciphertext size is slightly shorter
⇒ 2nd character of Cookie is "h"

Even worse: the CRIME attack [RD'2012]

(simplified)



16KB

POST /bank.com/buy?**uid=Jh1111**…
Cookie: **uid=J**h PL8g69684rksfsdg
Host: bank.com

Recover entire cookie after
256 × (len of Cookie) attempts

Takes several seconds (simplified)

What to do?

The problem:

Observed ciphertext length reveals compression amount ⇒
reveals plaintext info … no good solution

Non-defense: add a random length pad to ciphertext

First defense: compression disabled in TLS (and others, e.g., SPDY)

Problem: compression also done in HTTP layer

⇒ BREACH attack [PHG'13]

… much harder to disable HTTP compression in practice

What to do? [PHG'13]

Many web sites are impacted …

A proposed defense:

- Application layer "tags" sensitive data fields in HTTP requests and responses (cookies, PII, etc.)
- HTTP-level compression only applied to non-sensitive fields

… but not easy to implement

Password Breaches

Module 3: Attacks and Defenses

Stanford | Stanford Center for Professional Development

A (small) sample of password breaches

(server-side compromise)

2012: **Linked-in**: 6 million passwords (hashed, **unsalted**)

2013:

Twitter: 250,000 passwords (hashed, salted)

Evernote: 50 million records: usernames, emails, **hashed passwords**

Adobe: 38 million records

email addrs., **password hints**, and **encrypted** passwords

2015:

LastPass: stolen email addr., **hashed master passwords** (and salts)

Weak password choice

Users frequently choose weak passwords: (adobe list, 2013)

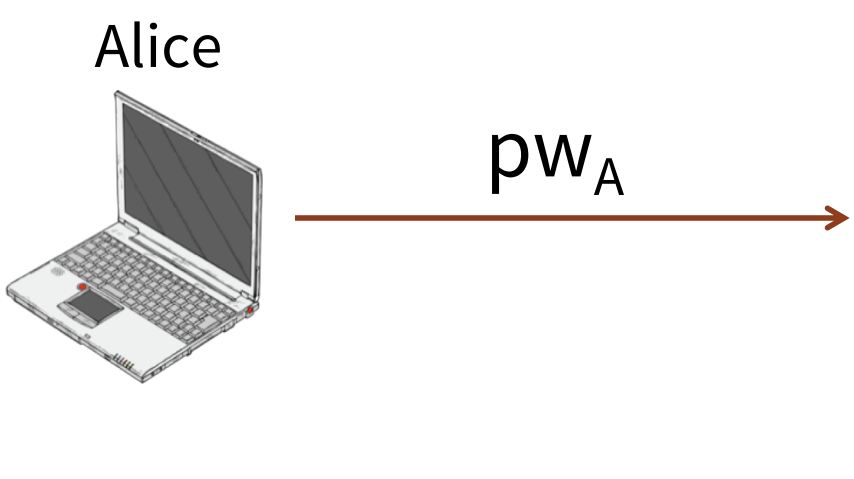
Password:	123456	123456789	password	adobe123	12345678	qwerty	1234567
Fraction of users:	5%	1.1%	0.9%	0.5%	0.5%	0.5%	0.3%

List of 360,000,000 words covers about 25% of user passwords

Total: 8.8%

How to store passwords

First rule of password storage: **never store passwords in the clear !**



password database

Alice	$\mathbf{S_A}$	$H(pw_A , \mathbf{S_A})$
Bob	$\mathbf{S_B}$	$H(pw_B , \mathbf{S_B})$
…	…	…
id	salt	hash

To validate a given password server checks:

$$H(pw_A , \mathbf{S_A}) \stackrel{?}{=} \text{StoredHash(Alice)}$$

How to hash?

Linked-in: **SHA-1** hashed (**unsalted**) passwords

⇒ 6 days, 90% of pwds. recovered by exhaustive search

The problem: SHA-1 is too fast …
attacker can try all words in dictionary

To hash passwords:

- Use a **keyed** hash function (e.g., HMAC) where key stored in HSM
- In addition: use a **slow**, **space-hard** function

How to hash?

PBKDF2, bcrypt: slow hash functions

- Slowness by “iterating” a crypto hash function like SHA256
- Parameterized number of iterations (e.g., set for 1000 evals/sec)

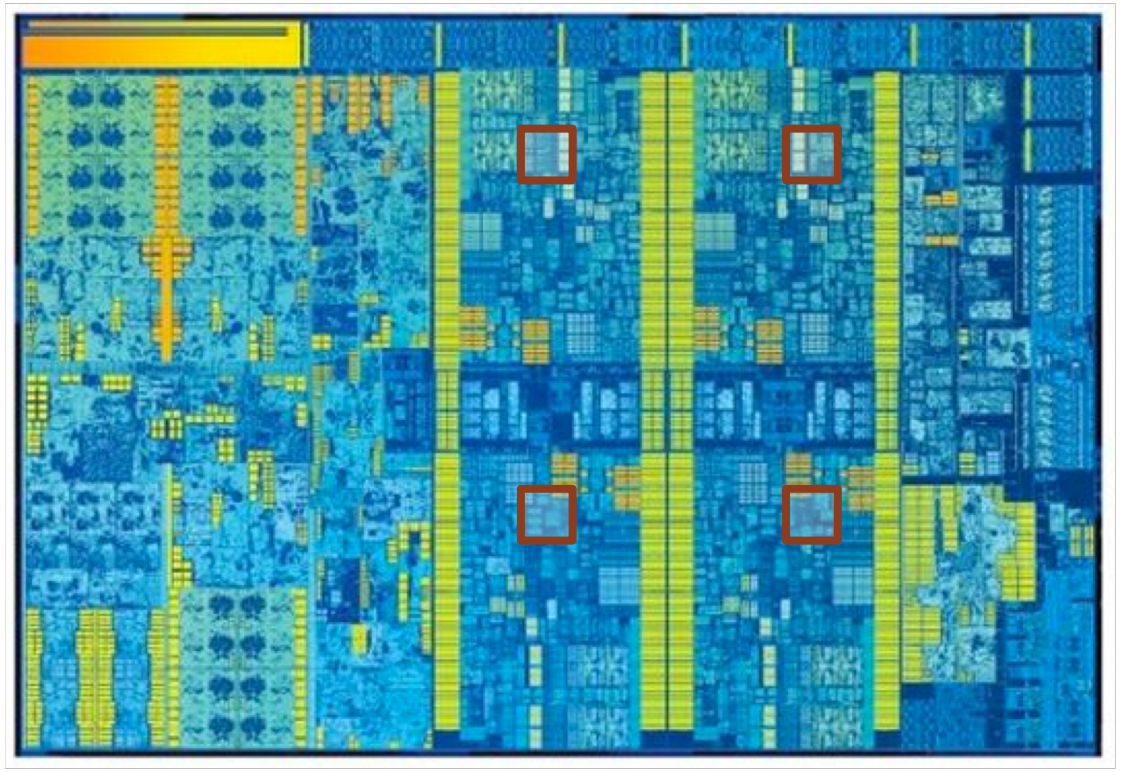
Problem: custom hardware (e.g., GPU) can evaluate hash function much faster than a commodity CPU

⇒ attacker can do dictionary attack much faster than 1000 evals/sec.

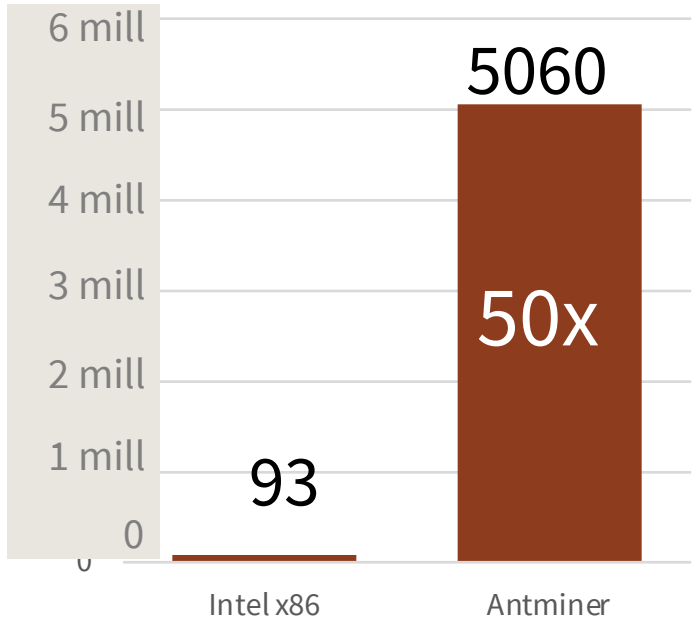
Why is custom hardware faster?

only small part of CPU
used to hash

custom hardware
for Bitcoin mining ($1,695)



Intel Skylake



Antminer S7 5.06TH/s

How to hash?

Scrypt: a slow hash function AND need lots of memory to evaluate

⇒ custom hardware not much faster than commodity CPU

Problem: memory access pattern depends on input password

⇒ local attacker can learn memory access pattern for user's pwd

⇒ eliminates need for memory in an offline dictionary attack

Is there a space-hard function where time is independent of pwd?

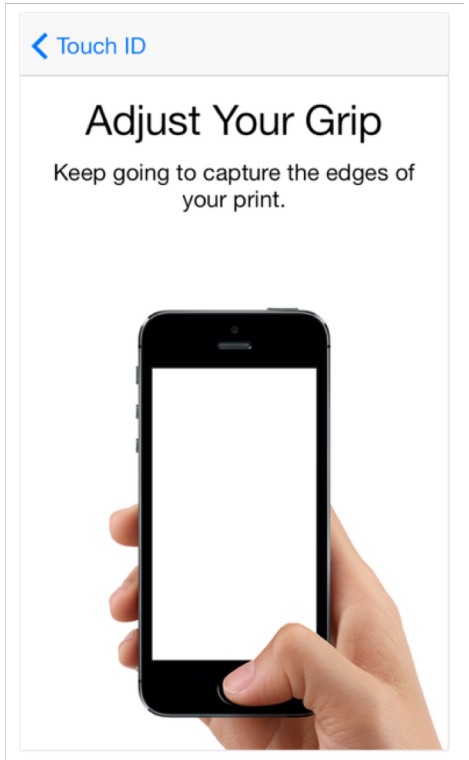
- Pwd hashing competition (2015): **Argon2i** (also see **Balloon** hashing)

Strengthening User Authentication

One option: biometrics:

Fingerprints, retina, facial recognition, …

Benefit: hard to forget



<u>Problems</u>:

Biometrics are not generally secret

Cannot be changed, unlike passwords

⇒ Should primarily be used as a second factor authentication

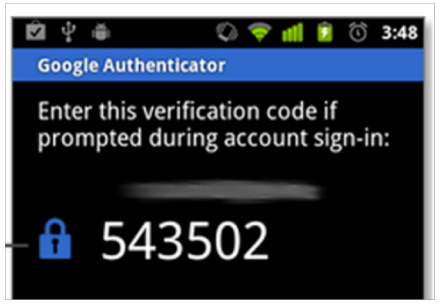
note: CCC'13

2nd factor OTP authentication

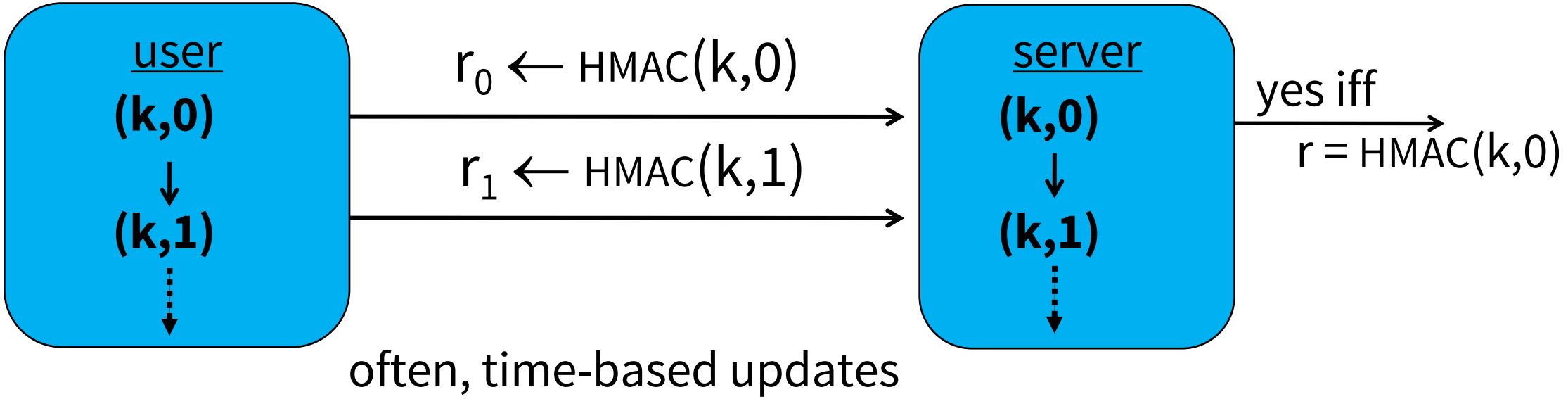
Setup:

Choose random key **k**

On device and server: **sk = (k,0)**



Identification:



Google authenticator

6-digit timed one-time passwords (TOTP) [RFC 6238]

Wide web-site adoption: Gmail, Dropbox, WordPress, …

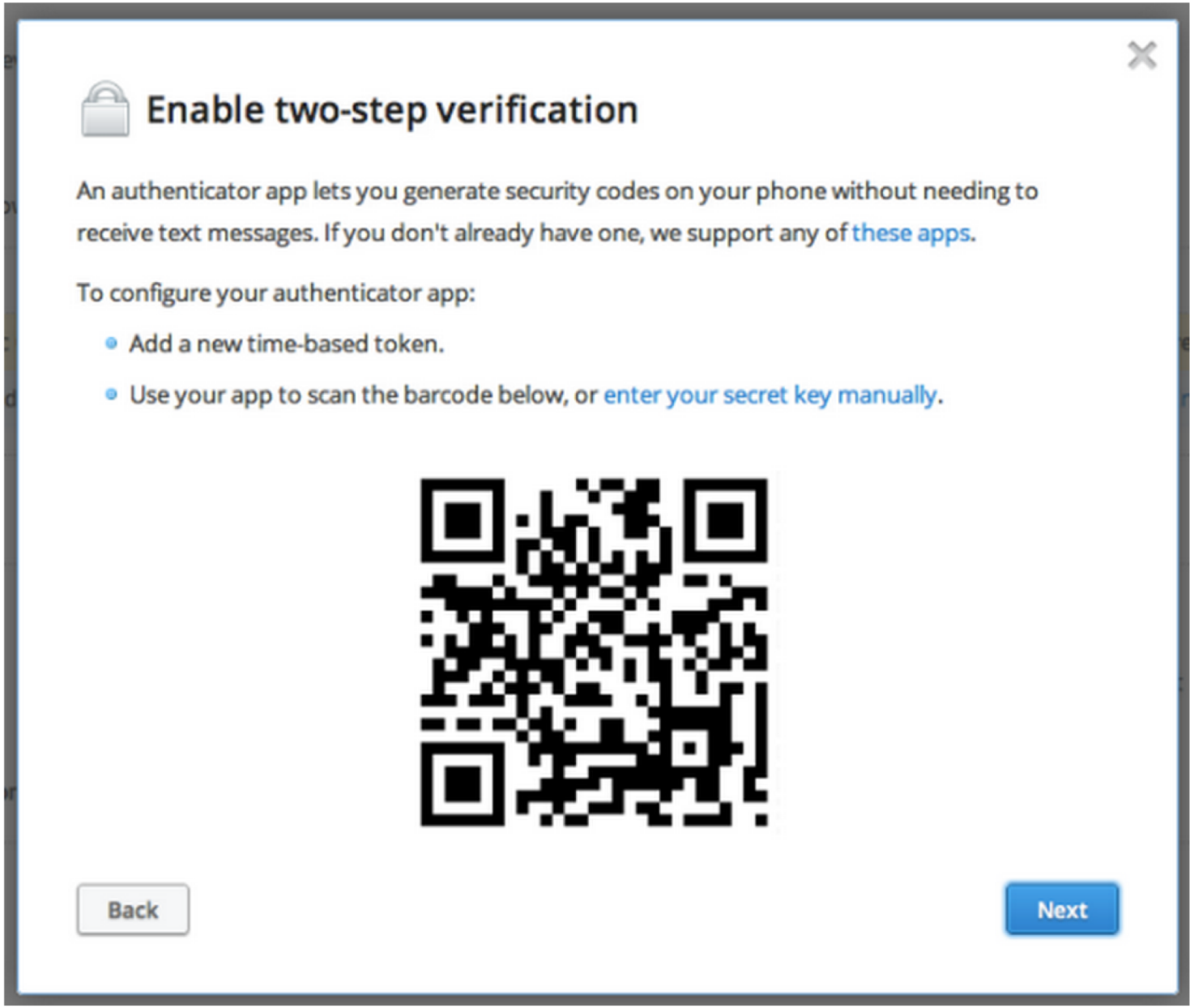
- Open study: 6.4% Gmail user adoption [EuroSec 2015]

To enable TOTP for a user: web site presents QR code with embedded data:

otpauth://totp/Example**:alice@dropbox.com**?
secret=**JBSWY3DPEHPK3PXP** & issuer=Example

(Subsequent user logins require user to present TOTP)

Danger: password reset upon user lockout


Enable two-step verification
An authenticator app lets you generate security codes on your phone without needing to receive text messages. If you don't already have one, we support any of these apps.
To configure your authenticator app:
Add a new time-based token.
Use your app to scan the barcode below, or enter your secret key manually.
Back
Next

Server compromise exposes secrets

March 2011:

RSA announced servers attacked, secret keys stolen

⇒ enabled SecurID user impersonation

Can we do better? Answer: Yes!

Duo (also FIDO U2F)

Signature-based challenge response:

No secrets on server, simple user experience

Certificates on the Web

Module 3: Attacks and Defenses

Stanford | Stanford Center for Professional Development

Certificate Issuance Woes

Wrong issuance:

2011: **Comodo** and **DigiNotar** RAs 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

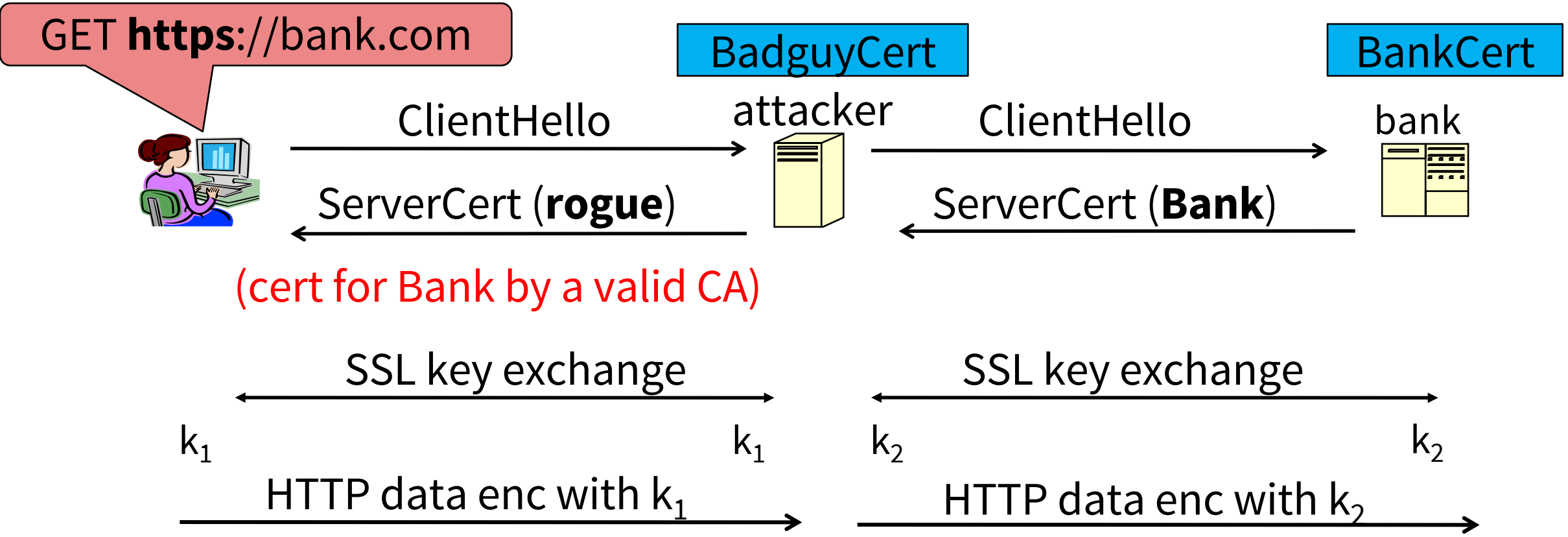
(2) Chrome restricts India CCA root to only seven Indian domains

2015: **MCS** (intermediate CA cert issued by **CNNIC**) issues certs for Google domains

Result: current CNNIC root no longer recognized by Chrome

⇒ 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)

HPKP: HTTP public-key pinning

- HTTP header that lets a site declare CAs that can sign its cert

  Public-Key-Pins: pin-sha256="cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs=";

- on subsequent HTTPS, browser rejects certs issued by other CAs
- TOFU: Trust on First Use

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 on the CT log
- Efficient implementation using Merkle hash trees

- Companies can scan logs to look for invalid issuance

A new CA: Let's encrypt (letsencrypt.org)

A new open Certificate Authority: free certs

- Provisioning via an automated agent running on web server

2016: 800K certs issued

Step 1: install agent on web server

Step 2: agent proves domain ownership (e.g. bank.com) by

DNS record under bank.com or page at fixed URI at bank.com

and send Certificate Signing Request (CSR) to CA

Step 3: Let's encrypt CA checks domain ownership

if valid, issue cert and sends cert to agent

Step 4: agent installs cert on Web server … done

Abusing Mobile Sensors

Module 3: Attacks and Defenses

Stanford | Stanford Center for Professional Development

Sensors on smart phones

Microphone
Camera
GPS
Light sensor
Compass
MEMS Gyroscope / accelerometer
Power meter
Barometer
Heart rate / oximeter (on smart watches)

All have a specific function

Can they be abused ??

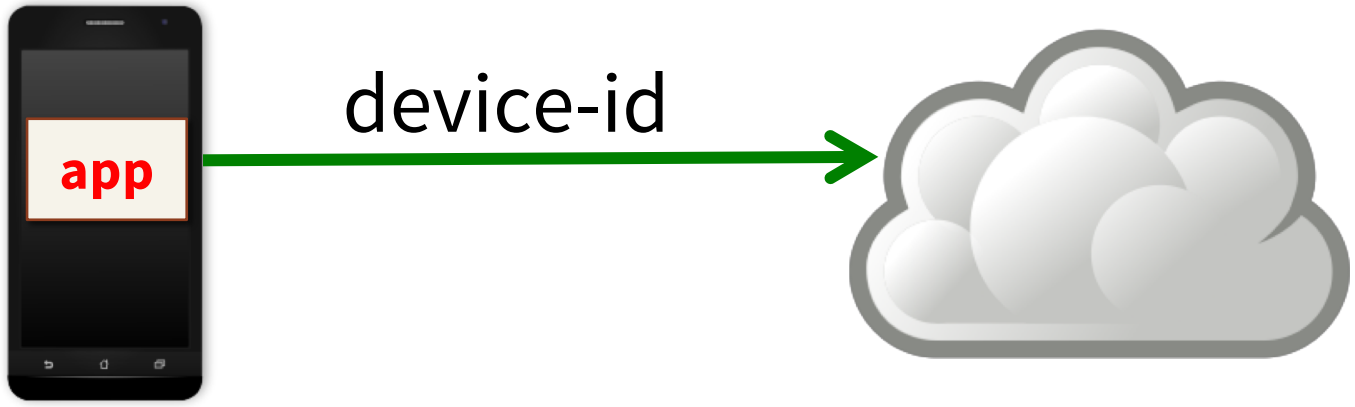
no user permission required

Example 1: fingerprinting

Imperfections in camera sensor can be used to link pictures taken by same phone [LG'06]

Accelerometer gives a stable device fingerprint [BBMN'14, DRXCN'14]

- App. can tell if it has been previously installed on device

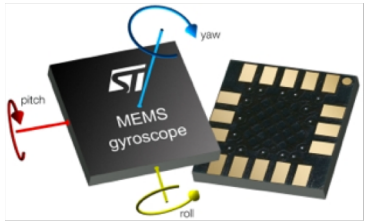


Example 2: Gyrophone [MBN'14]

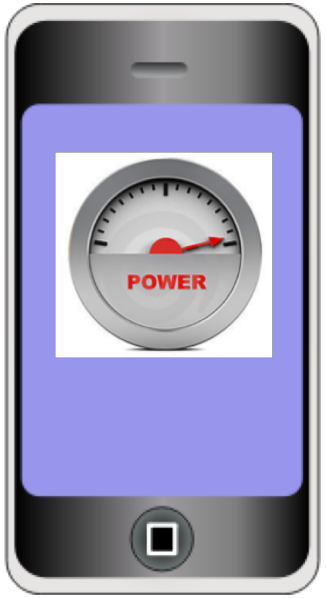
Phone gyroscope: measures vibrations (used for games)

Trouble:

- Gyroscope picks up air vibrations (a.k.a speech)
  - Sample rate (apps.): 200Hz
  - Machine learning ⇒ can recognize some speech



Example 3: Power usage sensor



Modern phones measure power drained from battery

Enables apps to optimize power use

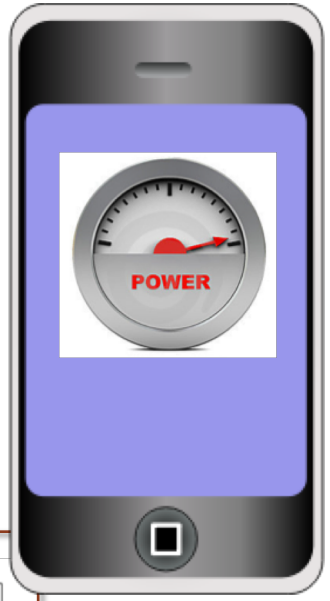
Repeatedly read:

```
/sys/class/power_supply/battery/voltage_now
/sys/class/power_supply/battery/current_now
```

Unrestricted access.

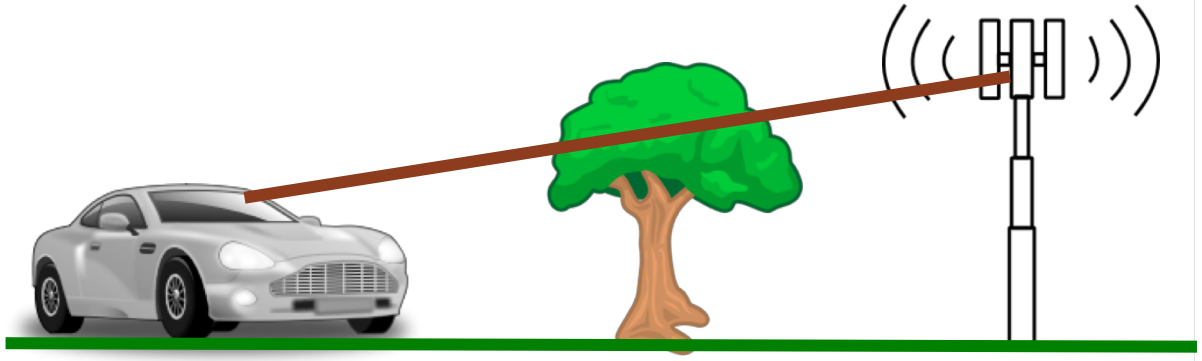
Can this be abused?

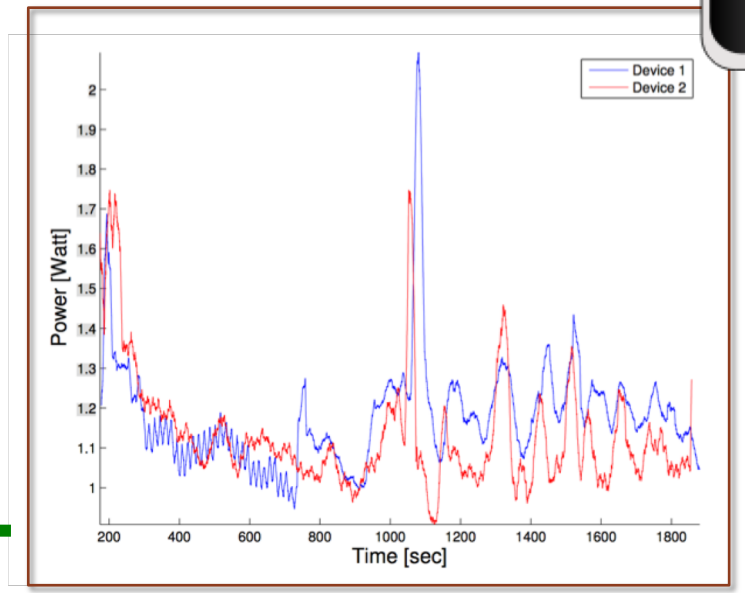
Example 3: Power usage sensor



Can this be abused? [MBSN'15]

Observation: power used by radio depends on distance and obstacles to cell tower





So what?

Our work: [MBSN'15]

power readings + machine learning ⇒ GPS

Why? Routes in a city have unique power fingerprints

Three goals:

✓ 1. identify route car is taking among a known set of routes

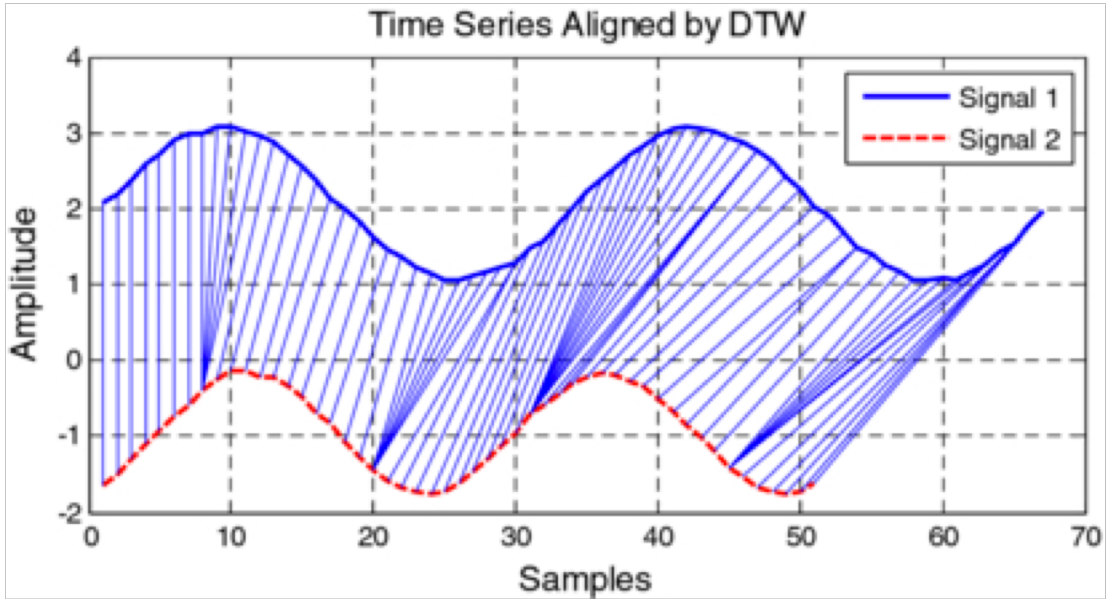
✓ 2. identify car's location along a known route

✓ 3. identify car's route based on a database of pre-measured short segments

Identify location along known route

Main tool: dynamic time warping (DTW)

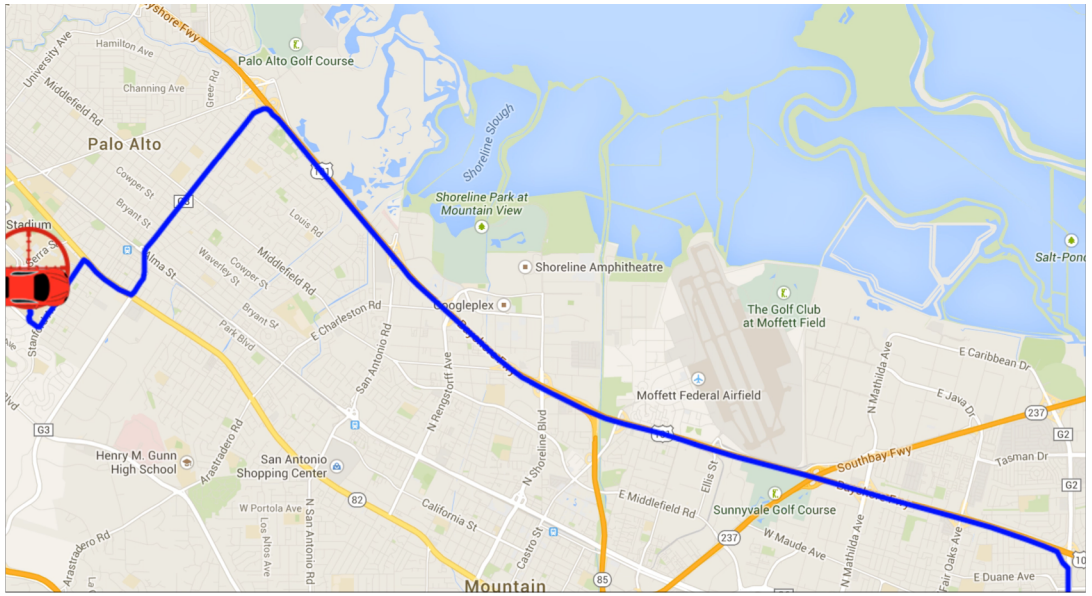
⇒ Aligns pre-recorded data with current samples



Identify location along known route

Main tool: dynamic time warping (DTW)

⇒ Aligns pre-recorded data with current samples



Lessons

Sensors can have unintended consequences

There is risk in giving apps direct access to sensors

Prevention:

- Always require permissions to access sensors
- Reduce data from sensors to min needed for utility or only provide abstract view of sensor data

Final note: limitations of air gaps

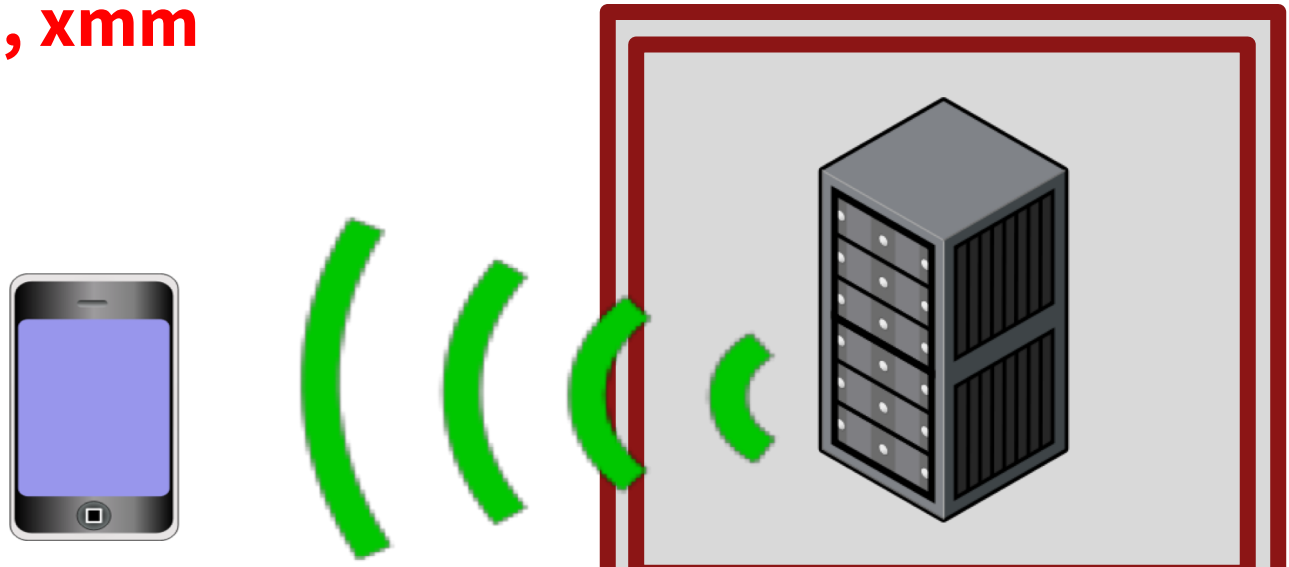
A machine holds sensitive date and is isolated from network

- If it gets infected, can the malware exfiltrate data?

Answer: yes! [Usenix Sec 2015]

- Mimic GSM signals using data bus
- Use x86 instruction: **MOVNTDQ m128, xmm**

- Effective for 60 feet



Stanford | Stanford Center for Professional Development