CSE 484 / CSE M 584: Computer Security and Privacy

Web Tracking (Continued)
Side Channels

Autumn 2018

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Ada Lerner, John Manferdelli, John Mitchell, Franziska Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Admin

- Lab 2 out Nov 5, due Nov 20, 4:30pm

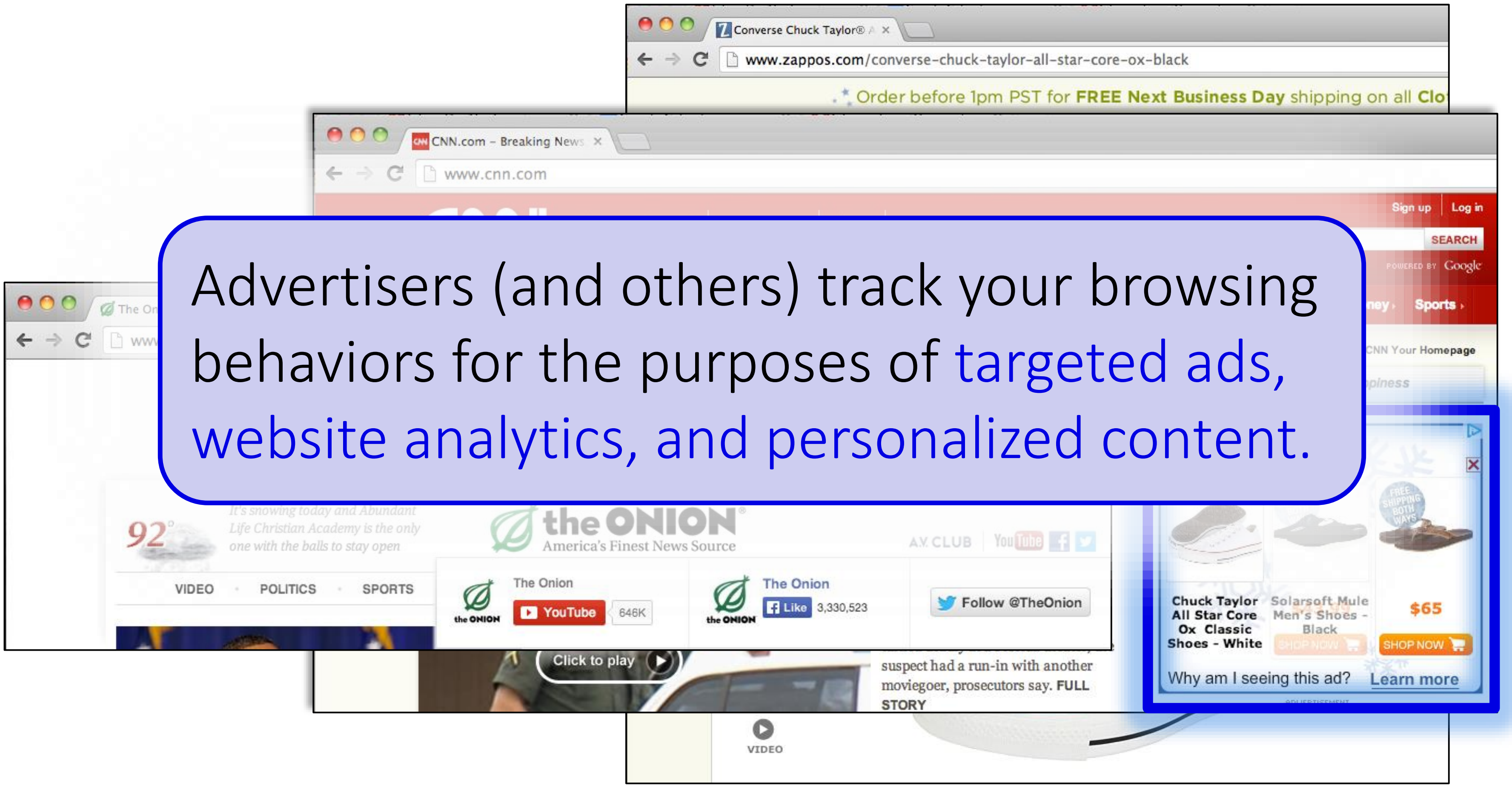
- Looking ahead:
- HW 3 out ~Nov 19, due ~Nov 30
- Lab 3 out ~Nov 26, due Dec 7 (Quiz Section on Nov 29)

- No class Nov 12 (holiday)
- No class Nov 21; video review assignment instead

Admin

- Final Project Proposals: Nov 16 – group member names and brief description
- Final Project Checkpoint: Nov 30 – preliminary outline and references
- Final Project Presentation: Dec 10 – 12-15-minute video – **must** be on time
- Explore something of interest to you, that could hopefully benefit you or your career in some way – technical topics, current events, etc

Review: Ads That Follow You



Advertisers (and others) track your browsing behaviors for the purposes of targeted ads, website analytics, and personalized content.

Review: Tracking Technologies

- HTTP Cookies
- HTTP Auth
- HTTP Etags
- Content cache
- IE userData
- HTML5 protocol and content handlers
- HTML5 storage
- Flash cookies
- Silverlight storage
- TLS session ID & resume
- Browsing history
- window.name
- HTTP STS
- DNS cache
- "Zombie" cookies that respawn (http://samy.pl/evercookie)

Review: Fingerprinting Web Browsers

- User agent
- HTTP ACCEPT headers
- Browser plug-ins
- MIME support
- Clock skew
- Installed fonts
- Cookies enabled?
- Browser add-ons
- Screen resolution
- HTML5 canvas (differences in graphics SW/HW!)

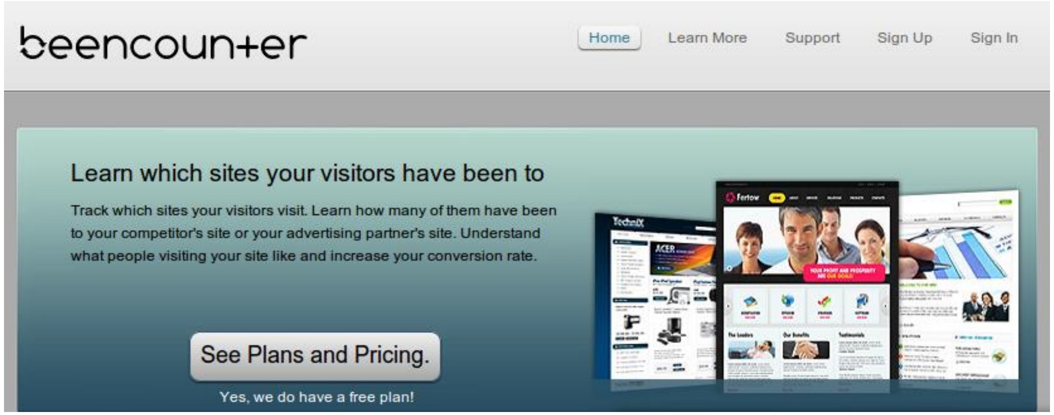
EFF's Panopticlick

- https://panopticlick.eff.org/

History Sniffing

How can a webpage figure out which sites you visited previously?

- Color of links
  - CSS :visited property
  - getComputedStyle()
- Cached Web content timing
- DNS timing



How Websites Get Your Identity

Personal trackers

Leakage of identifiers

GET http://ad.doubleclick.net/adj/...
Referer: http://submit.SPORTS.com/...?email=jdoe@email.com
Cookie: id=35c192bcfe0000b1...

Security bugs

Third party buys your identity

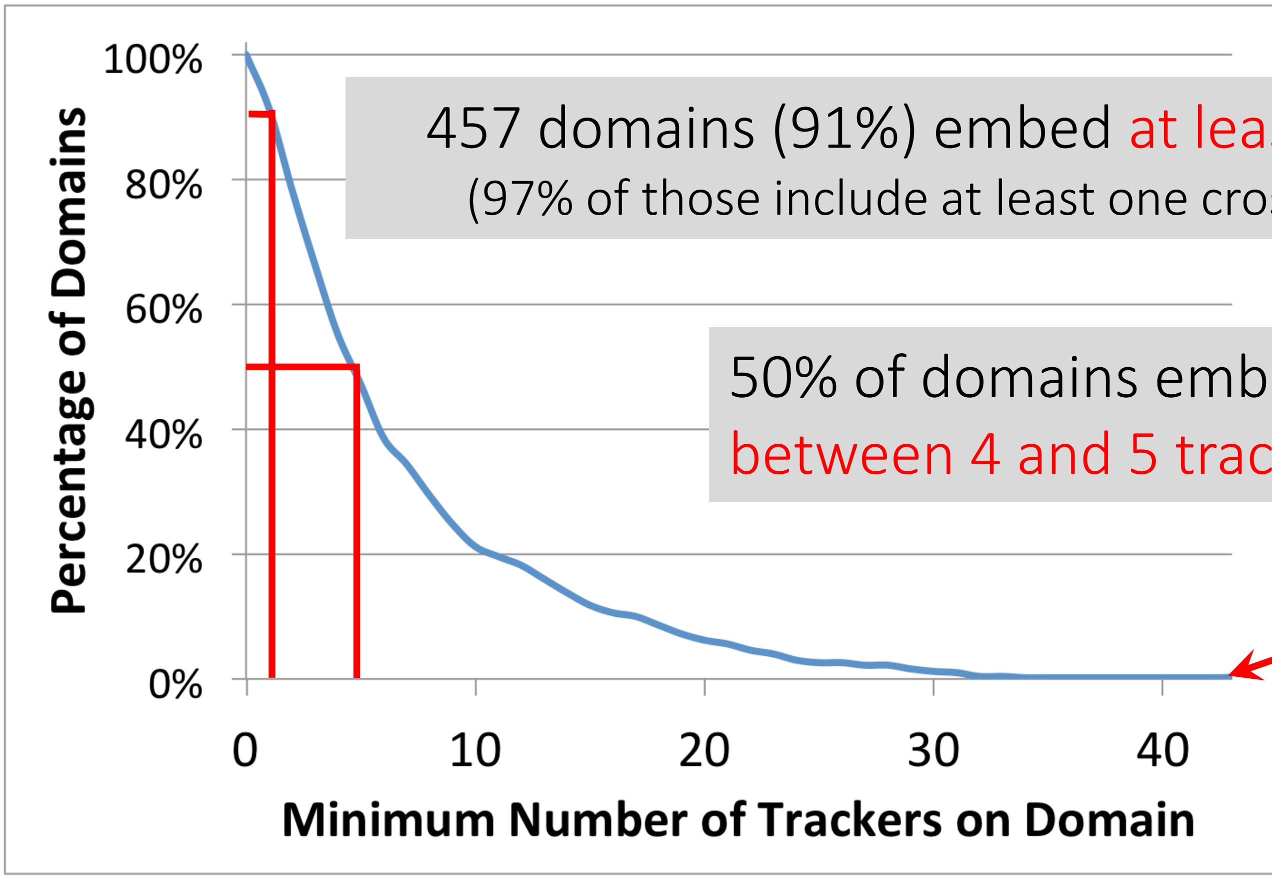
Measurement Study (2011)

- **Questions:**
  - How prevalent is tracking (of different types)?
  - How much of a user's browsing history is captured?
  - How effective are defenses?
- **Approach:** Build tool to automatically crawl web, detect and categorize trackers based on our taxonomy.

Longitudinal studies since then: tracking has increased and become more complex.

How prevalent is tracking?

524 unique trackers on Alexa top 500 websites (homepages + 4 links)

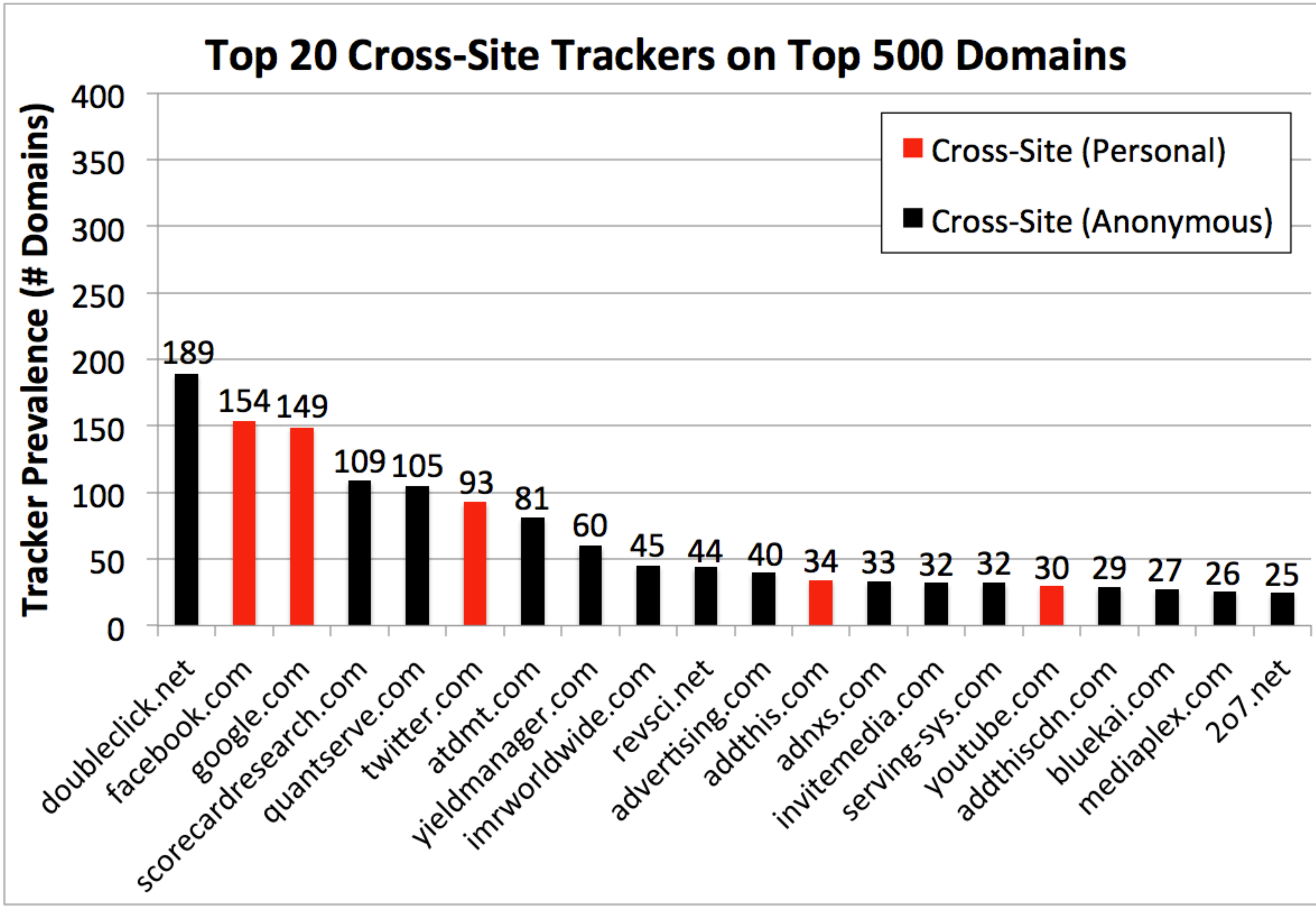


457 domains (91%) embed at least one tracker.
(97% of those include at least one cross-site tracker.)

50% of domains embed between 4 and 5 trackers.

One domain includes 43 trackers.

Who/what are the top trackers? (2011)



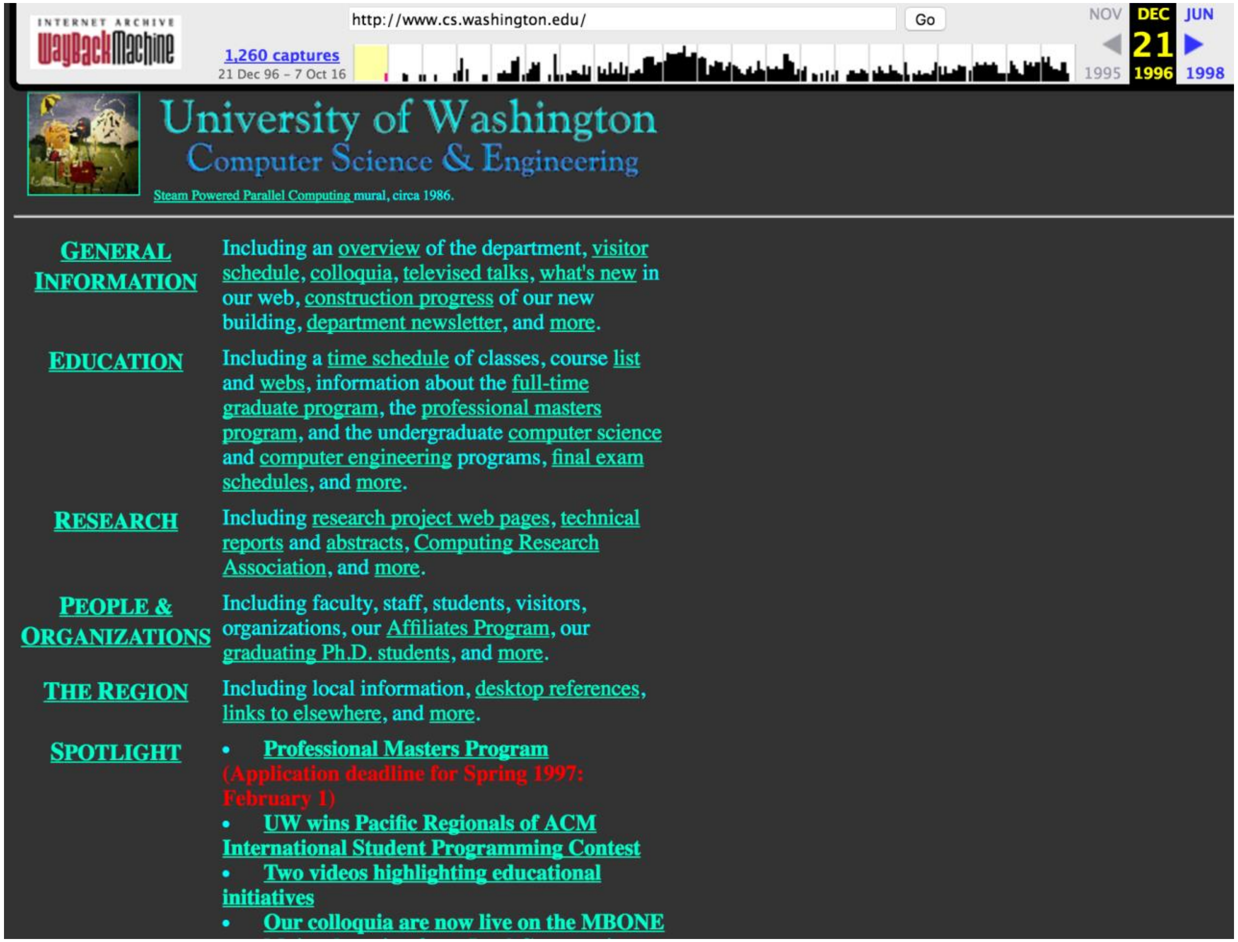
How has this changed over time?

- The web has existed for a while now…
  - What about tracking before 2011? (our first study)
  - What about tracking before 2009? (first academic study)

- Solution: time travel!

  [USENIX Security '16]

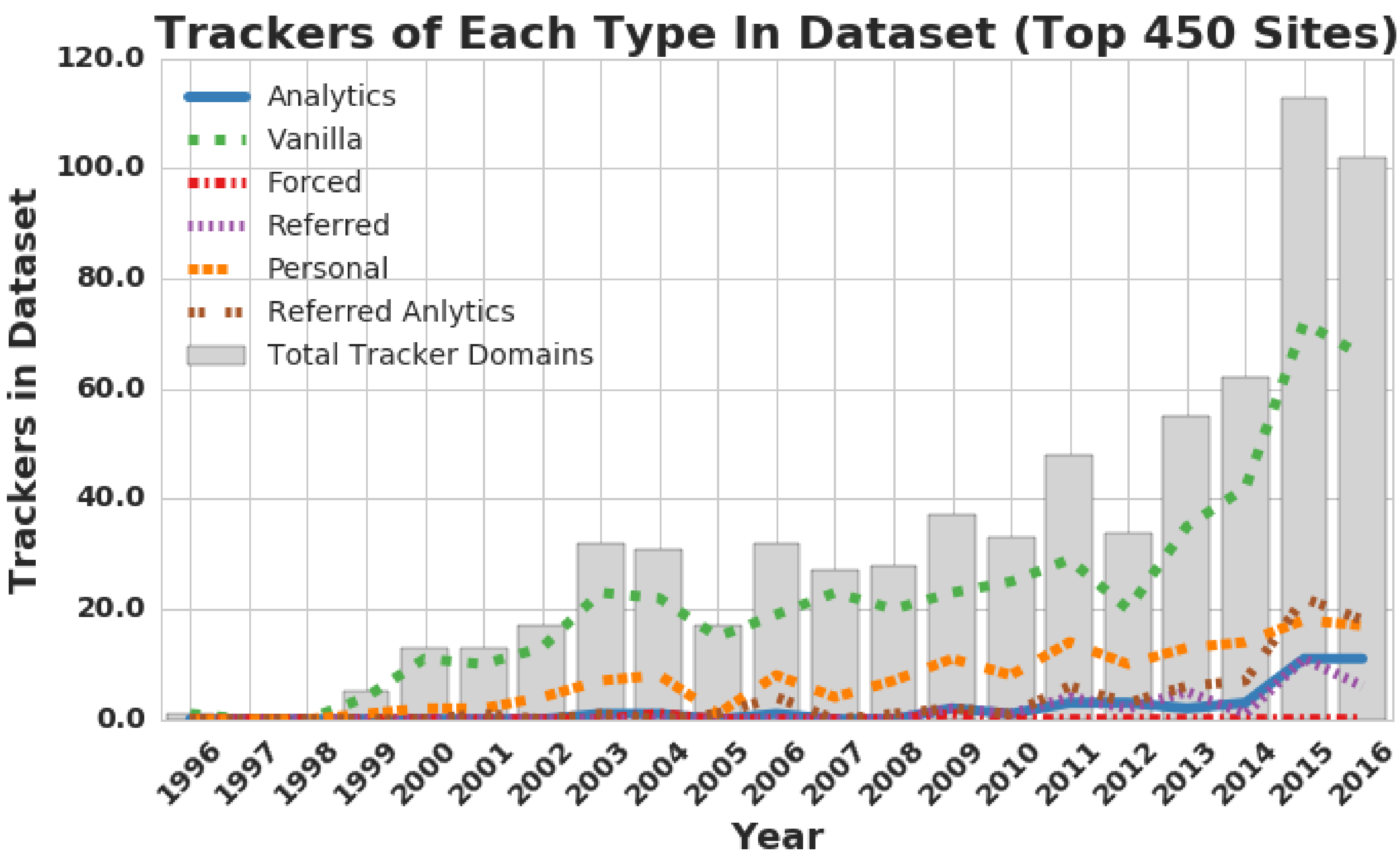
The Wayback Machine to the Rescue



Time travel for web tracking: http://trackingexcavator.cs.washington.edu

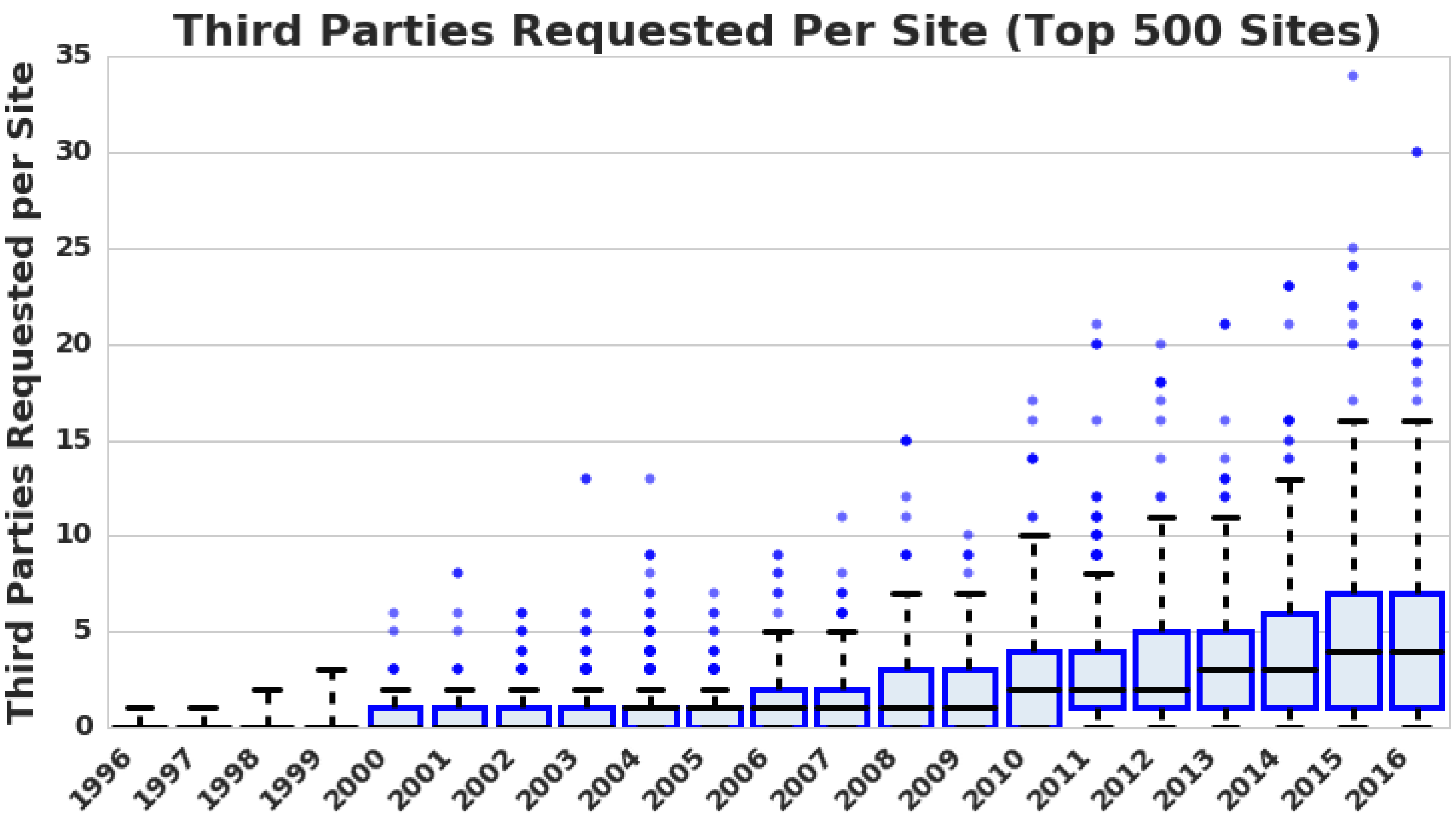
1996-2016: More & More Tracking

- More trackers of more types



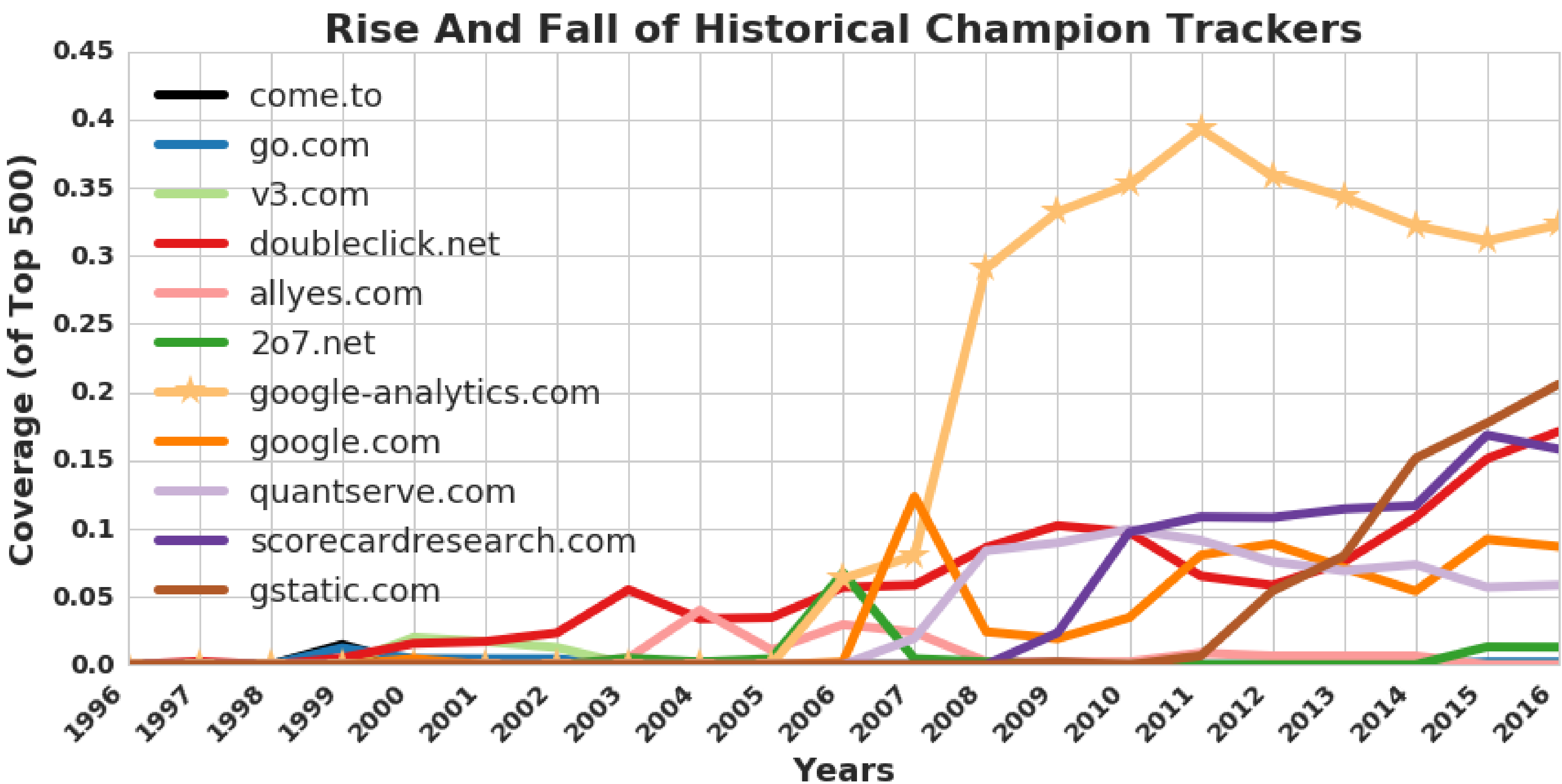
1996-2016: More & More Tracking

- More trackers of more types, more per site



1996-2016: More & More Tracking

- More trackers of more types, more per site, more coverage



ADINT (2017)

- Advertising for Intelligence Gathering
- Adversary can buy ads and use analytics from those ads to learn information about targets
  - Some ad networks provide location-based ad services
- Purchaser of ads can figure out
  - What mobile phone applications are in use in individual homes
  - A target’s movements through the physical world (e.g., stores, doctors offices, etc)

Side Channels

Side Channel Attacks

- Attacks based on information that can be gleaned from the physical implementation of a system, rather than breaking its theoretical properties
  - Most commonly discussed in the context of cryptosystems
  - But also prevalent in many contexts

Examples (on Cryptosystems)

- Timing attacks
- Power analysis

- Good overview: http://www.nicolascourtois.com/papers/sc/sidech_attacks.pdf

If you do something different for secret key bits 1 vs. 0, attacker can learn something…

Example Timing Attacks

- RSA: Leverage key-dependent timings of modular exponentiations
  - https://www.rambus.com/timing-attacks-on-implementations-of-diffie-hellman-rsa-dss-and-other-systems/
  - http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf

- Block Ciphers: Leverage key-dependent cache hits/misses

Power Analysis

- Simple power analysis: Directly read off bits from powerline traces
- Differential power analysis: Look for statistical differences in power traces, based on guesses of a key bit

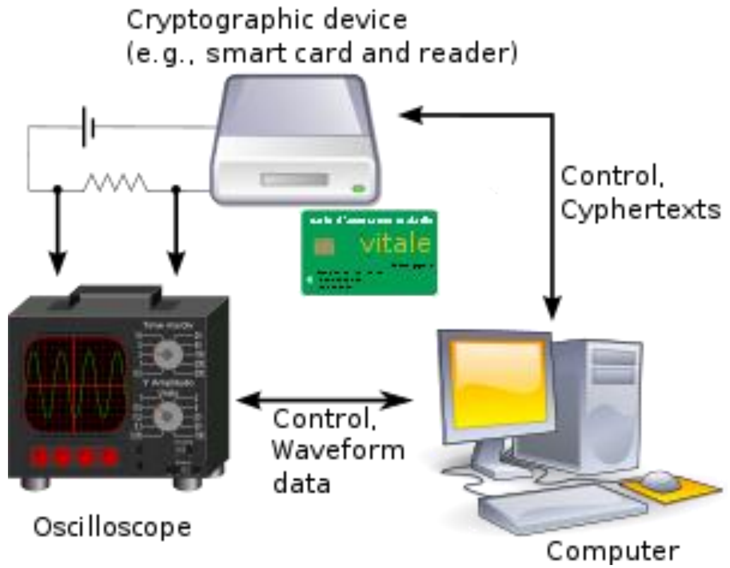
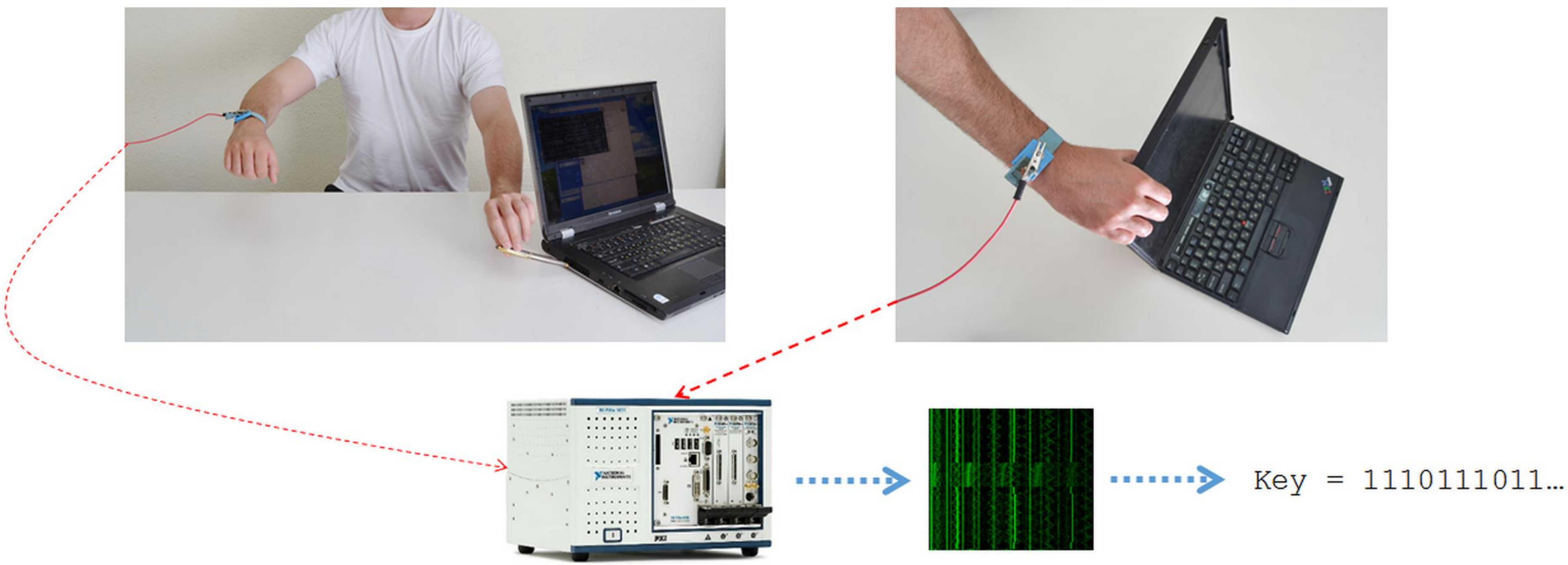


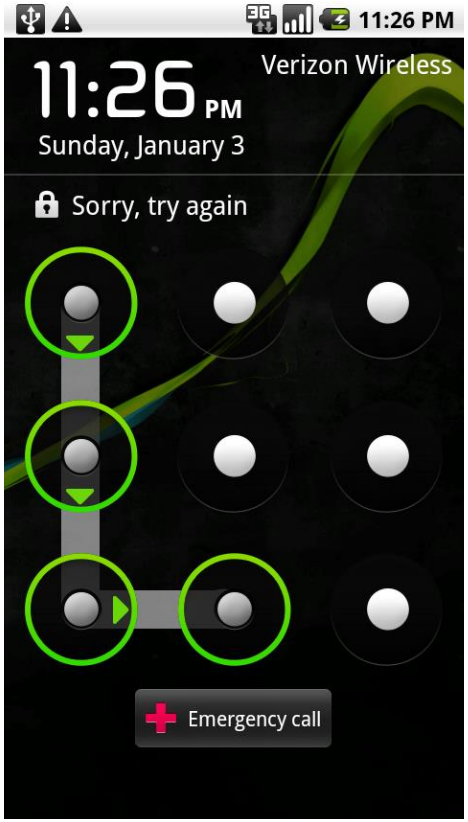
Image from https://en.wikipedia.org/wiki/Power_analysis

Key Extraction via Electric Potential



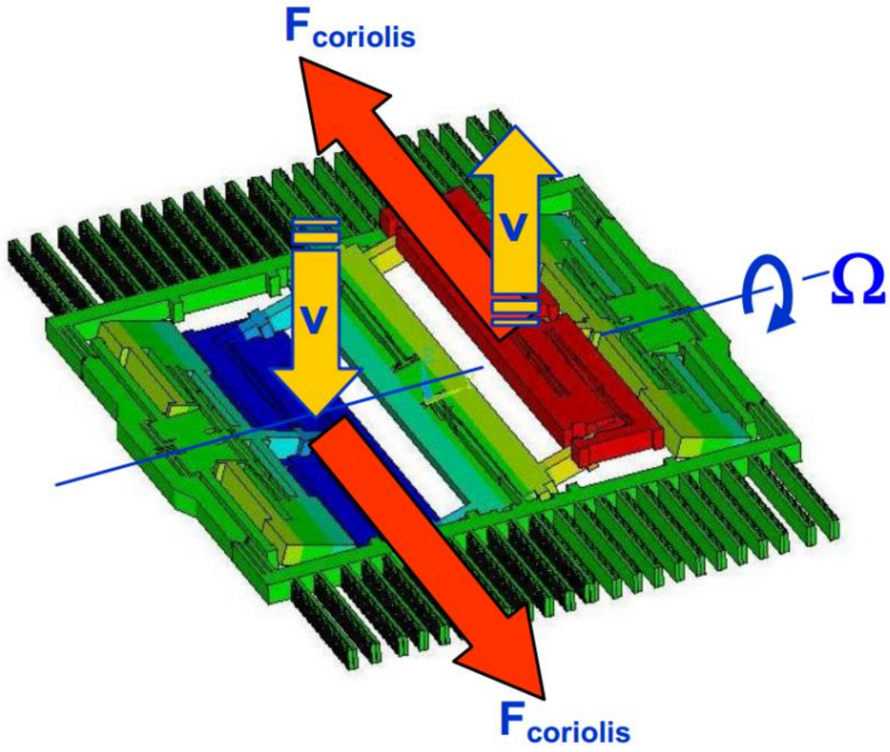
Genkin et al. "Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks On PCs" CHES 2014

Accelerometer Eavesdropping



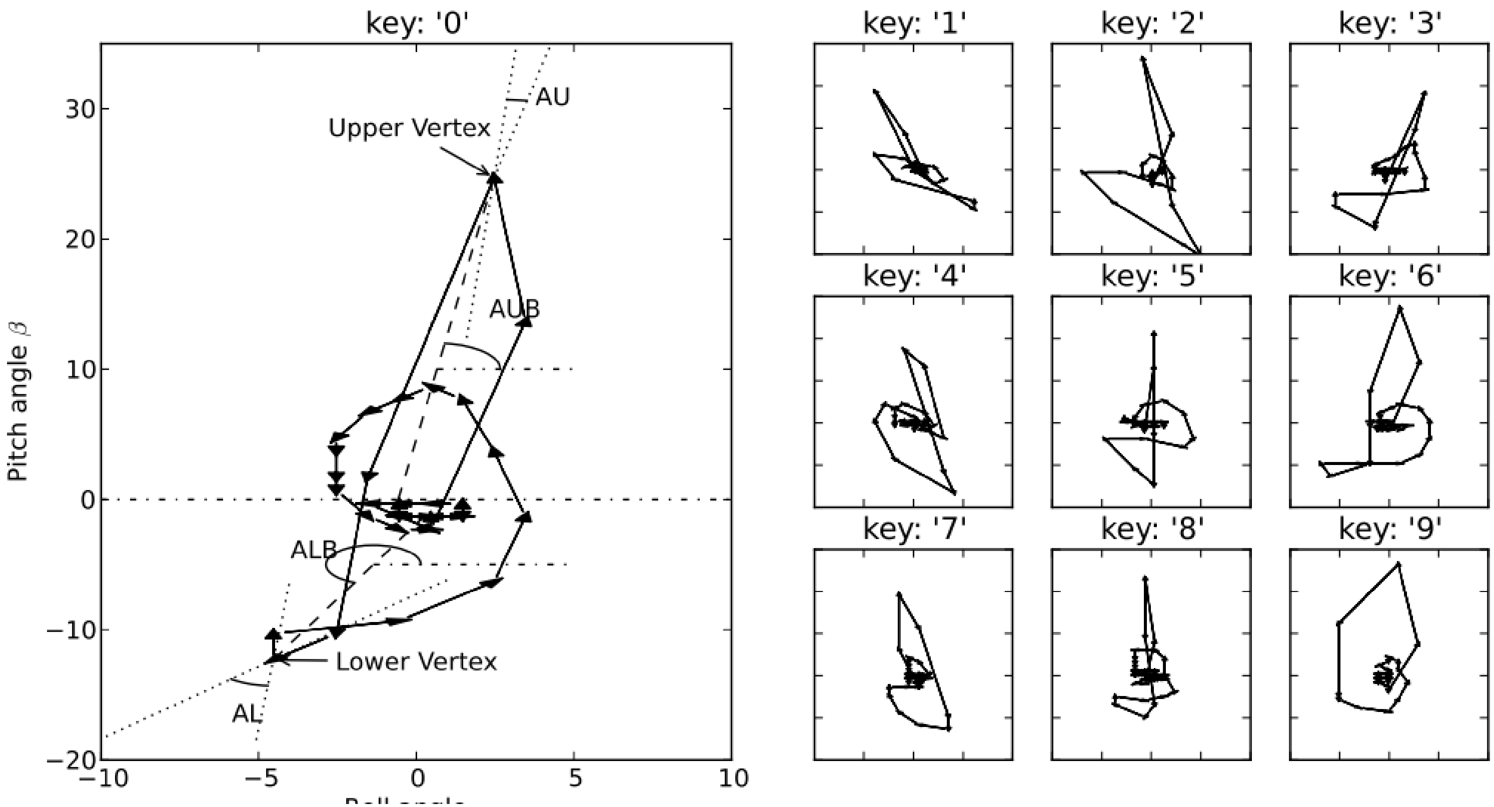
Aviv et al. "Practicality of Accelerometer Side Channels on Smartphones" ACSAC 2012

Gyroscope Eavesdropping



Michalevsky et al. "Gyrophone: Recognizing Speech from Gyroscope Signals" USENIX Security 2014

More Gyroscope



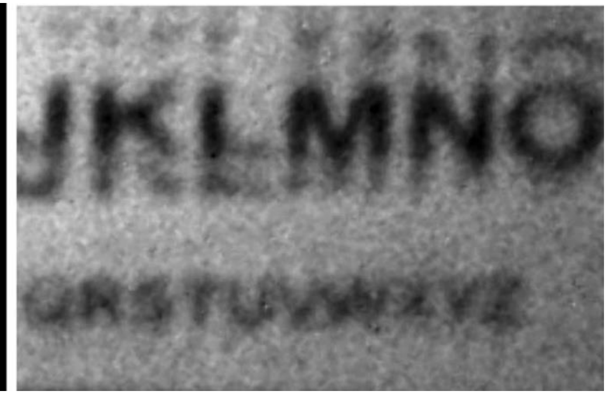
Chen et al. “TouchLogger: Inferring Keystrokes On Touch Screen From Smartphone Motion” HotSec 2011

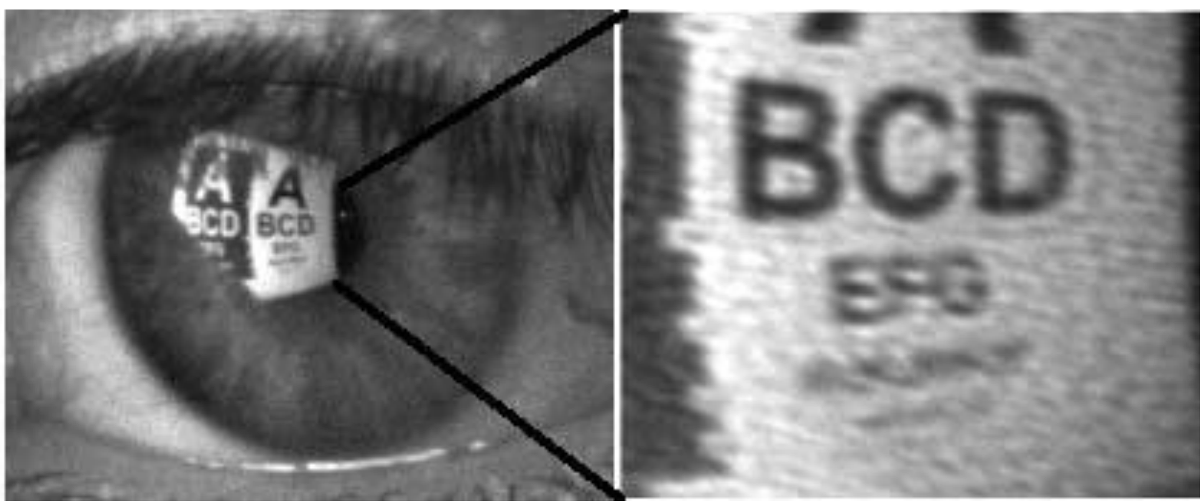
Keyboard Eavesdropping

Zhuang et al. “Keyboard Acoustic Emanations Revisited” CCS 2005
Vuagnoux et al. “Compromising Electromagnetic Emanations of Wired and Wireless Keyboards” USENIX Security 2009

[Backes et al.]

Compromising Reflections





Audio from Video

Davis et al. “The Visual Microphone: Passive Recovery of Sound from Video” SIGGRAPH 2014

Identifying Web Pages: Traffic Analysis

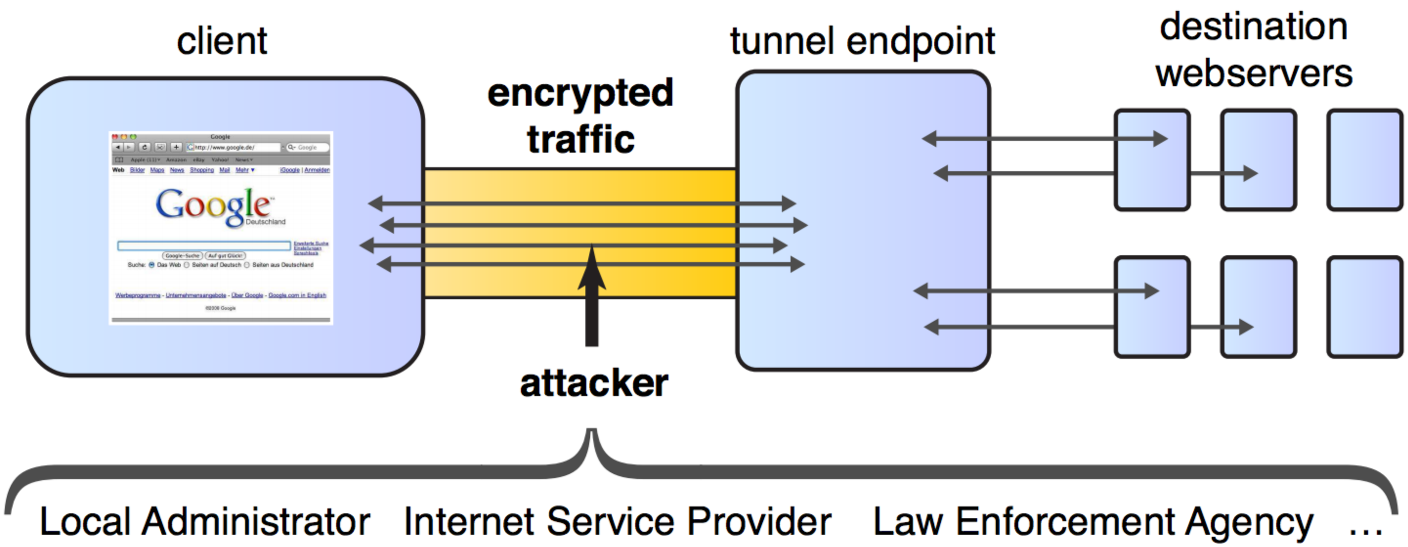
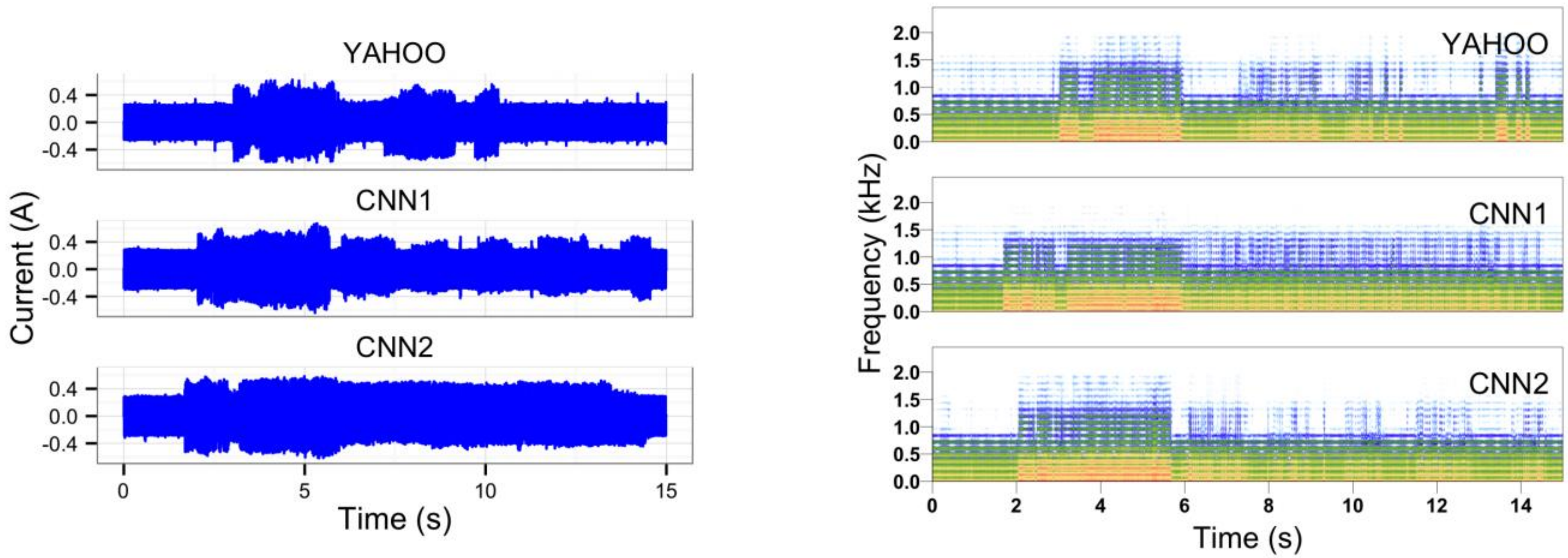


Figure 1: Website fingerprinting scenario and conceivable attackers

Herrmann et al. "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier" CCSW 2009

Identifying Web Pages: Electrical Outlets



(a) Time-domain plots

(b) Spectrogram plots

Fig. 1: Time- and frequency-domain plots of several power traces as a MacBook loads two different pages. In the frequency domain, brighter colors represent more energy at a given frequency. Despite the lack of obviously characteristic information in the time domain, the classifier correctly identifies all of the above traces.

Clark et al. "Current Events: Identifying Webpages by Tapping the Electrical Outlet" ESORICS 2013

Powerline Eavesdropping

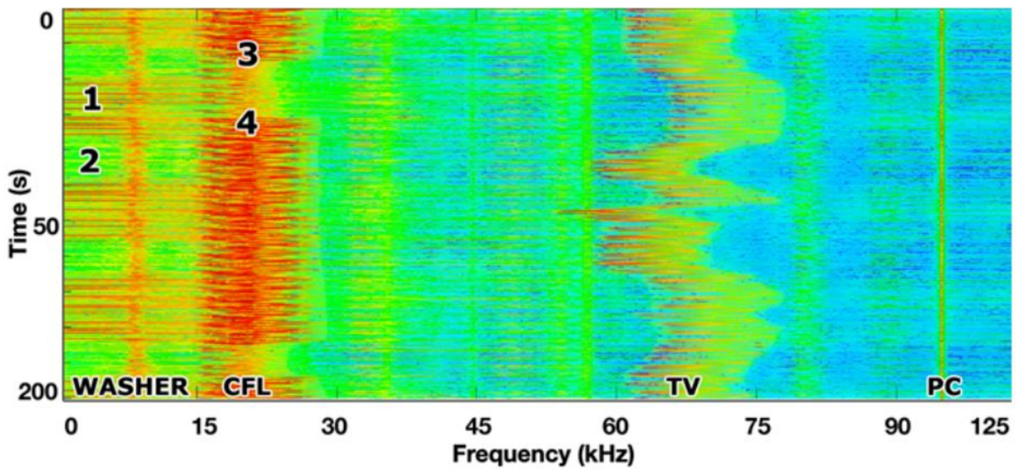


Figure 1: Frequency spectrogram showing various electrical appliances in the home. Washer cycle on (1) and off (2). CFL lamp turning off briefly (3) and then on (4). Note that the TV's (Sharp 42" LCD) EMI shifts in frequency, which happens as screen content changes.

Enev et al.: Televisions, Video Privacy, and Powerline Electromagnetic Interference, CCS 2011

Spectre

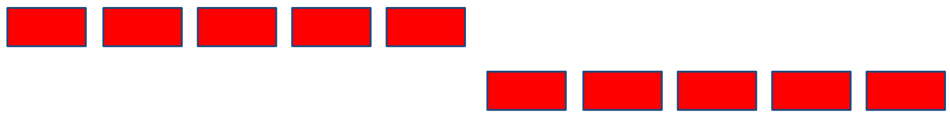
- Exploit speculative execution and cache timing information to extract private information from the same process
  - Example: JavaScript from web page trying to extract information from Browser

- Architecture Background:
  - Hardware architecture provides “promises” to software
  - Those proposes focus on the functional properties of the software, not performance properties
  - Architectures do a lot to try to increase performance

Instruction Speculation Tutorial

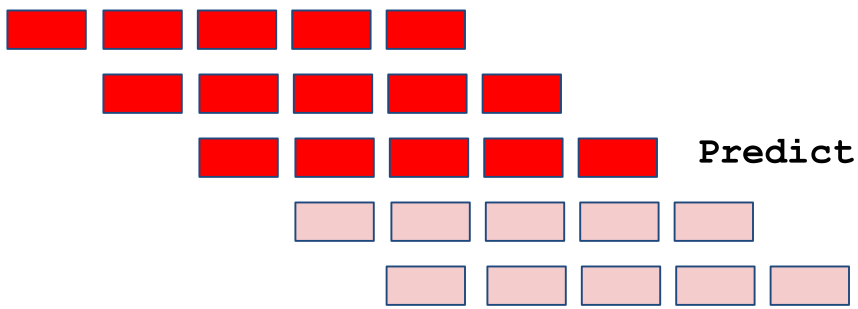
Many steps (cycles) to execute one instruction; time flows left to right →

`add`
`load`



Go Faster: Pipelining, branch prediction, & instruction speculation

`add`
`load`
`branch`
`and`
`store`



`Predict direction: target or fall thru`
`Speculate!`
`Speculate more!`

Speculation correct: Commit architectural changes of **`and`** (register) & **`store`** (memory) go fast!

Mis-speculate: Abort architectural changes (registers, memory); go in other branch direction

Hardware Caching Tutorial

Main Memory (DRAM) 1000x too slow

Add Hardware Cache(s): small, transparent hardware memory

- Like a software cache: speculate near-term reuse (locality) is common
- Like a hash table: an item (block or line) can go in one or few slots

E.g., 4-entry cache w/ slot picked with address (key) modulo 4

0	`--`	**12**?	0	`12`	**07**?	0	`12`	**12**?	0	`12`	**16**?	0	`16`	Note **12**
1	`--`	Miss	1	`--`	Miss	1	`--`	HIT!	1	`--`	Miss	1	`--`	victimized
2	`--`	Insert **12**	2	`--`	Insert **07**	2	`--`	No	2	`--`	Victim **12**	2	`--`	“early” due
3	`--`		3	`--`		3	`07`	changes	3	`07`	Insert **16**	3	`07`	to “alias”

Spectre (Worksheet)

- Consider this code, running as a kernel system call or as part of a cryptographic library.

```
if (x < array1_size)
    y = array2[array1[x] * 256];
```

- Suppose:
  - That an adversary can run code, in the same process.
  - That an adversary can control the value x.
  - That an adversary has access to array2.
  - That the adversary's code cannot just read arbitrary memory in the process.
  - That there is some secret value, elsewhere in the process, that the adversary would like to learn.

- Can you envision a way that an adversary could use their own code, to call a vulnerable function with the above code, to learn the secret information? Leverage branch prediction and cache structure / timing.

Spectre: Key Insights

- Train branch predictor to follow one branch of a conditional
- After branch predictor trained, make the followed branch access information that the code should *not* be allowed to access
- That access information will be loaded into the cache
- After the hardware determines that the branch was incorrectly executed, the logic of the program will be rolled back *but* the cache will still be impacted
- Time reads to cache, to see which cache lines are read more efficiently

Attacker Steps

- Attacker: Execute code with valid inputs, train branch predictor to assume conditional is true
- Attacker: Invoke code with x outside of array1 , array1_size and array2 not cached, but value at array1+x cached // Attacker goal: read secret memory at address array1+x
- CPU: CPU guesses bounds check is true, speculatively reads from array2[array1[x]*256] using malicious x
- CPU: Read from array2 loads data into cache at an address that depends on array1[x] using malicious x
- CPU: Change in cache state not reverted when processor realizes that speculative execution erroneous
- Attacker: Measure cache timings for array2; read of array2[n*256] will be fast for secret byte n (at array1+x)
- Attacker: Repeat for other values of x

Other Types of Side Channels?