CSE 484 / CSE M 584: Computer Security and Privacy

Side Channel Attacks

Spring 2019

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Admin

- **HW3 due** today @ 5pm
- **Project checkpoint 2 due** today@ 5pm

- **Monday: Guest lecture** from Peter Ney (UW)
  - Cell phone surveillance (IMSI catchers) and DNA + computer security

- My office hours next week: **Wed 9am** (not Thurs)

Side Channel Attacks

- Attacks based on information that can be gleaned from the physical implementation of a system, rather than breaking its theoretical properties
  - Initially 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…

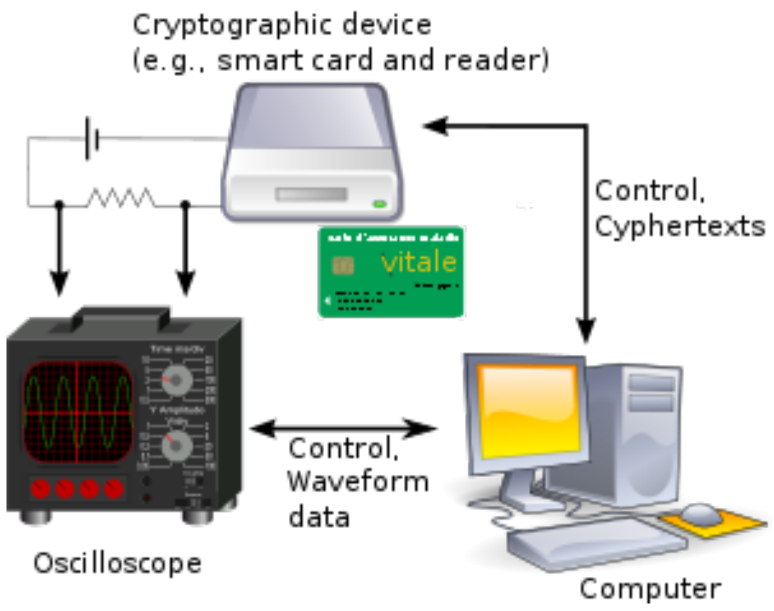
Example Timing Attacks

- **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

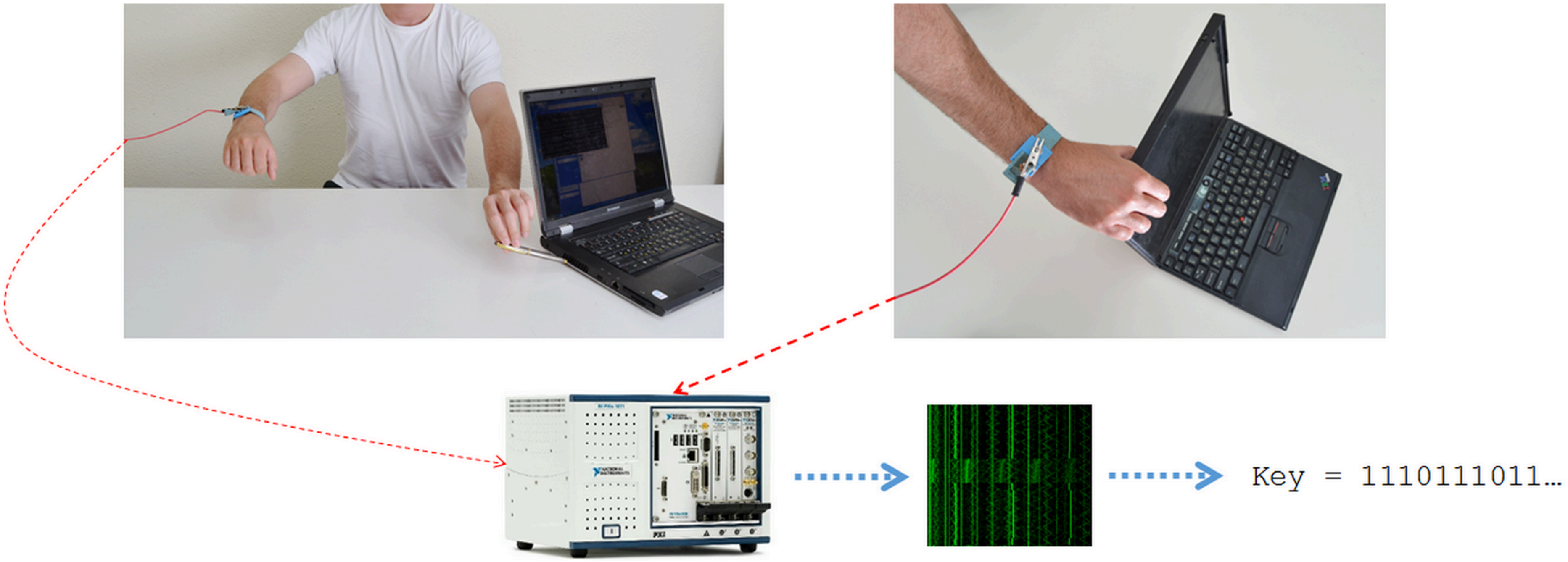
- **Block Ciphers:** Leverage key-dependent cache hits/misses

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



Key Extraction via Electric Potential



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

SPECTRE

Spectre

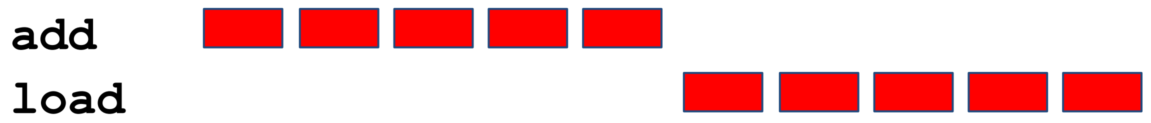
Exploit **(1) speculative execution** and **(2) cache timing information** to extract private information from the same process

- Example: JavaScript running in web page
- Recall Charlie Reis's guest lecture on iframe isolation…
- Note there are other variants (a new one nearly every day!)

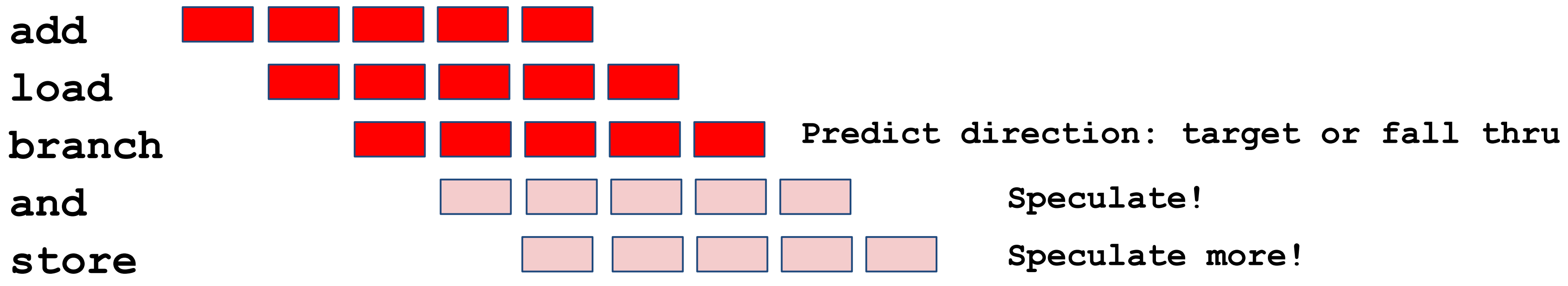
Material from http://research.cs.wisc.edu/multifacet/papers/hill_mark_wisconsin_meltdown_spectre.pptx

Instruction Speculation

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



Go Faster! Pipelining, branch prediction, & instruction speculation



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

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

Speculation affects the cache!

Cache Timing Attacks

- **Basic idea:** Manipulate cache to a known state, wait for victim activity, then examine what has changed.
- Example: "Flush + Reload"
  1. Flush relevant addresses from cache
  2. Wait for victim (here: execute Spectre attack)
  3. Reload relevant addresses, time accesses

     If timing was fast, victim accessed relevant address; if timing was slow, victim did not.

https://conference.hitb.org/hitbsecconf2016ams/materials/D2T1%20-%20Anders%20Fogh%20-%20Cache%20Side%20Channel%20Attacks.pdf

Spectre Threat Model

```
  if (x < array1_size)
     y = array2[array1[x] * 4096];
```

- Suppose that:
  - Adversary can run code, in the same process.
  - Adversary can control the value x.
  - Adversary has access to array2.
  - Adversary cannot just read arbitrary memory in the process.
  - There is some secret value, elsewhere in the process, that the adversary would like to learn.
    - Specifically: **Attacker's goal is to read secret memory at address array1+x**

Spectre Attach Sketch

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

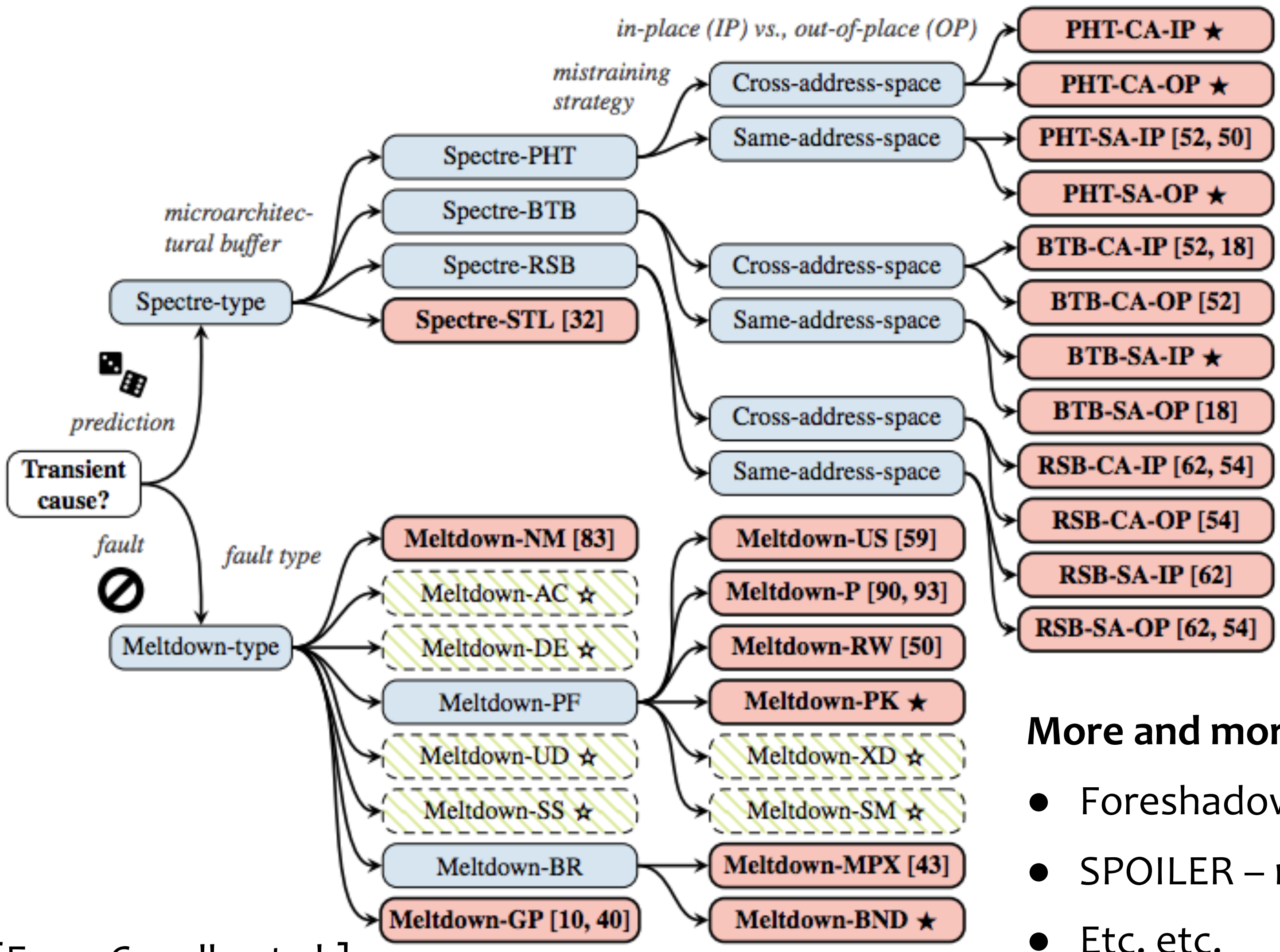
1. Train branch predictor to follow one branch of conditional.
2. After training, make the followed branch access information that the code should *not* be allowed to access (`array1[x]`).
3. The secret information will affect which part of `array2` is loaded into the cache.
4. After the hardware determines that the branch was incorrectly executed, the logic of the program will be rolled back *but* the cache will still be impacted.
5. Time reads to cache (e.g., Flush+Reload) for accesses to `array2`, to see which cache lines are read more efficiently.

Spectre Attack Details

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

- 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]*4096] 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*4096] will be fast for secret byte n (at array1+x)
- Attacker: Repeat for other values of x

It’s A Party



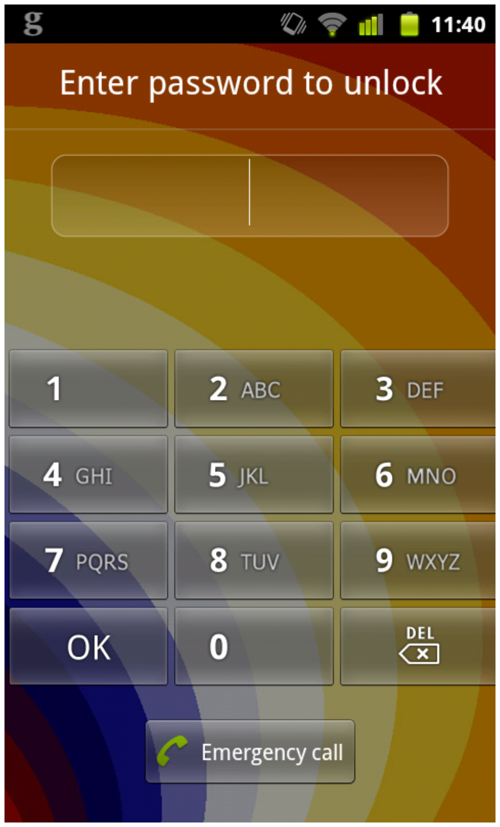
[From Canella et al.]

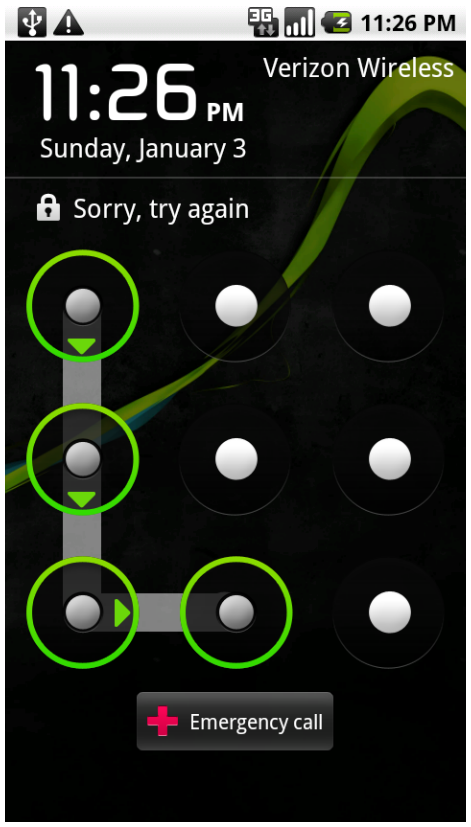
More and more:

- Foreshadow – attacks SGX
- SPOILER – mem dependence
- Etc. etc.

Other “Fun” Examples…

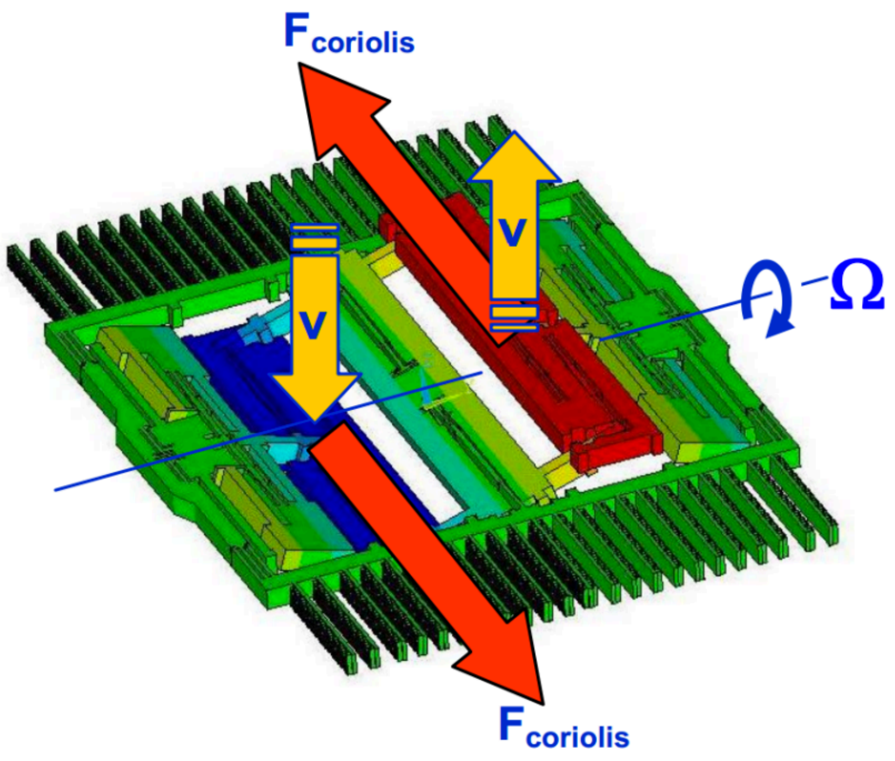
Accelerometer Eavesdropping





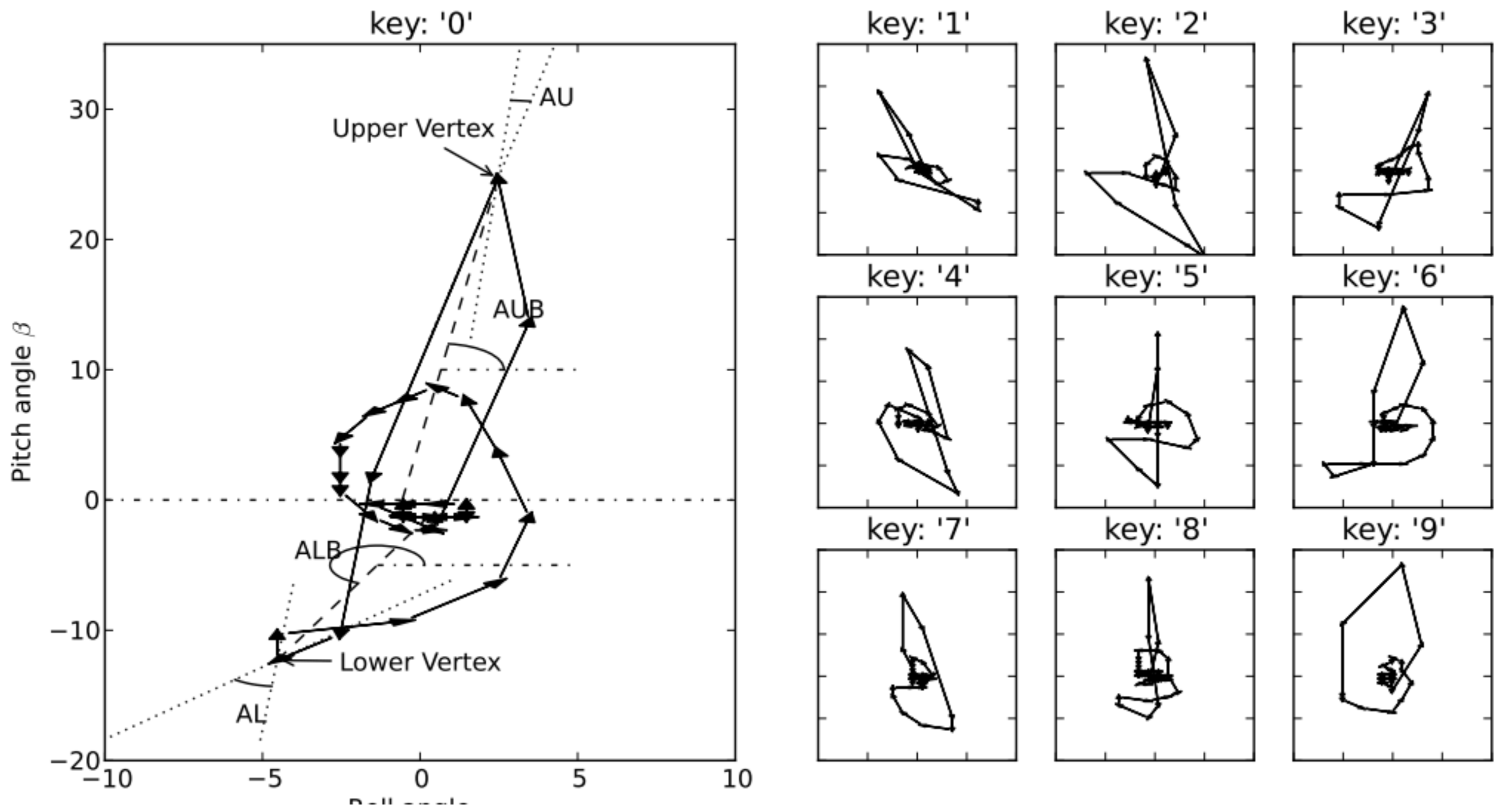
Aviv et al. "Practicality of Accelerometer Side Channels on Smartphones" ACSAC 2012

Gyroscope Eavesdropping



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

More Gyroscope



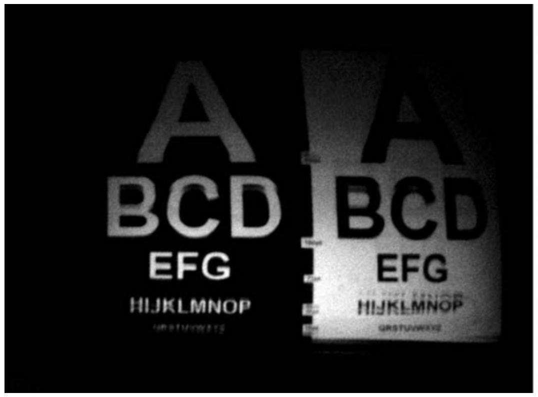
Chen et al. "TouchLogger: Inferring Keystrokes On Touch Screen From Smartphone Motion" HotSec 2011

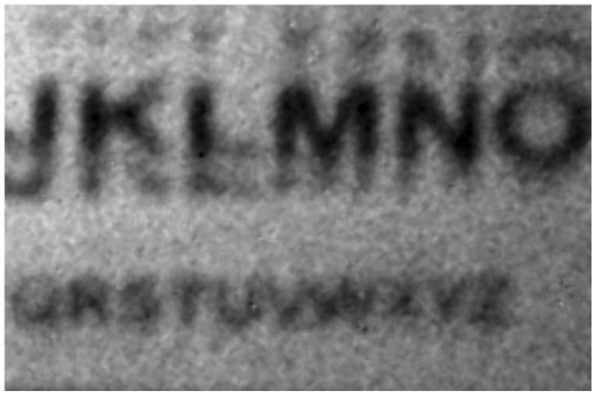
Keyboard Eavesdropping

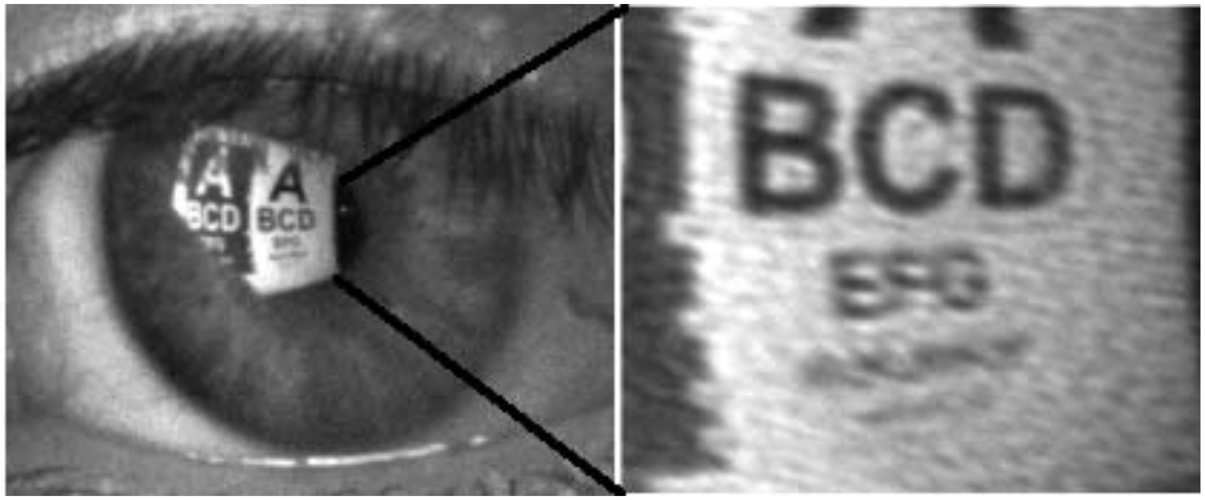
Zhuang et al. “Keyboard Acoustic Emanations Revisited” CCS 2005
Vuagnoux et al. “Compromising Electromagnetic Emanations of Wired and Wireless Keyboards” USENIX Security 2009

[Backes et al.]

Compromising Reflections

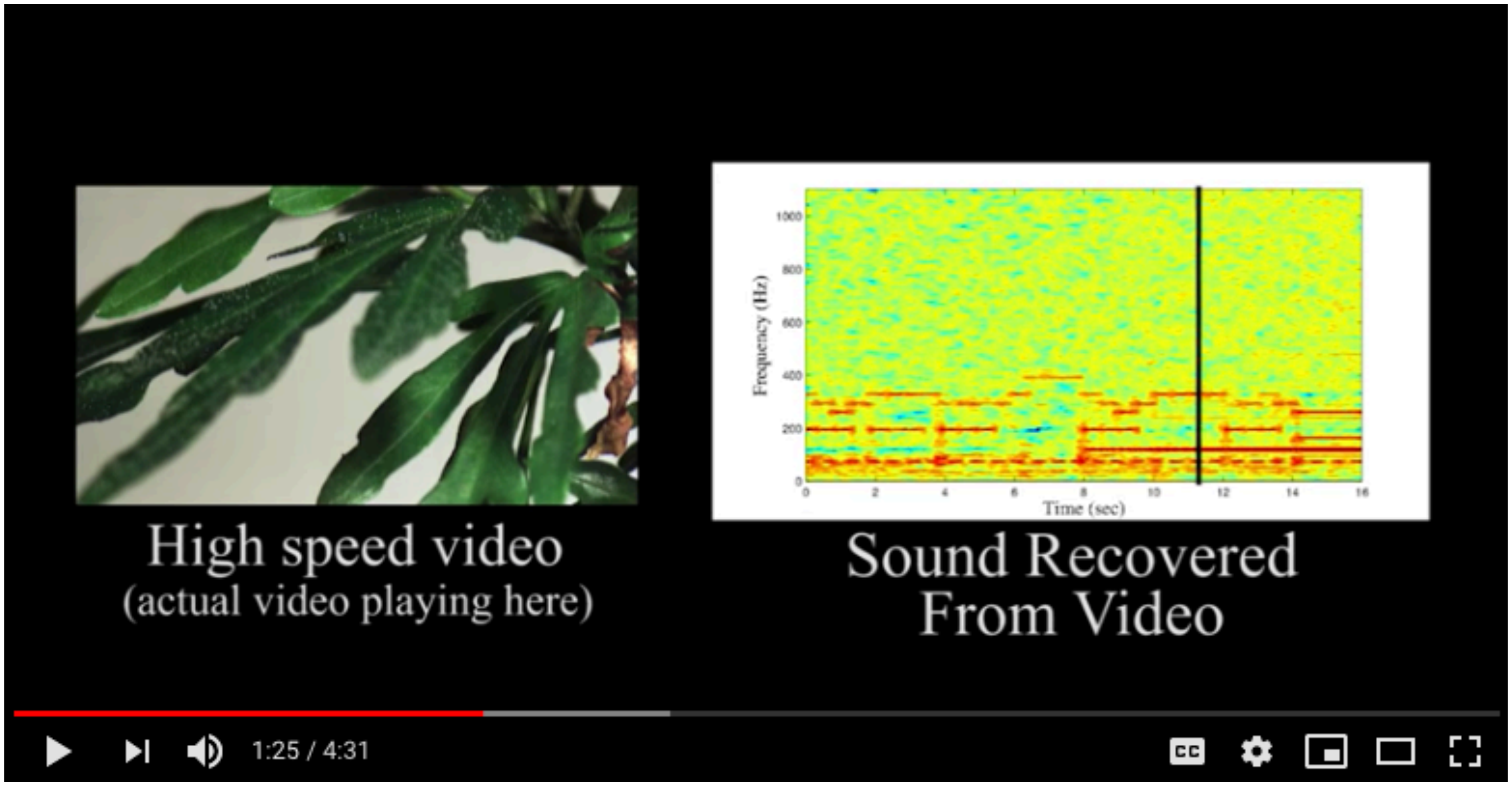






Audio from Video

https://www.youtube.com/watch?v=FKXOucXB4a8



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

Identifying Web Pages: Traffic Analysis

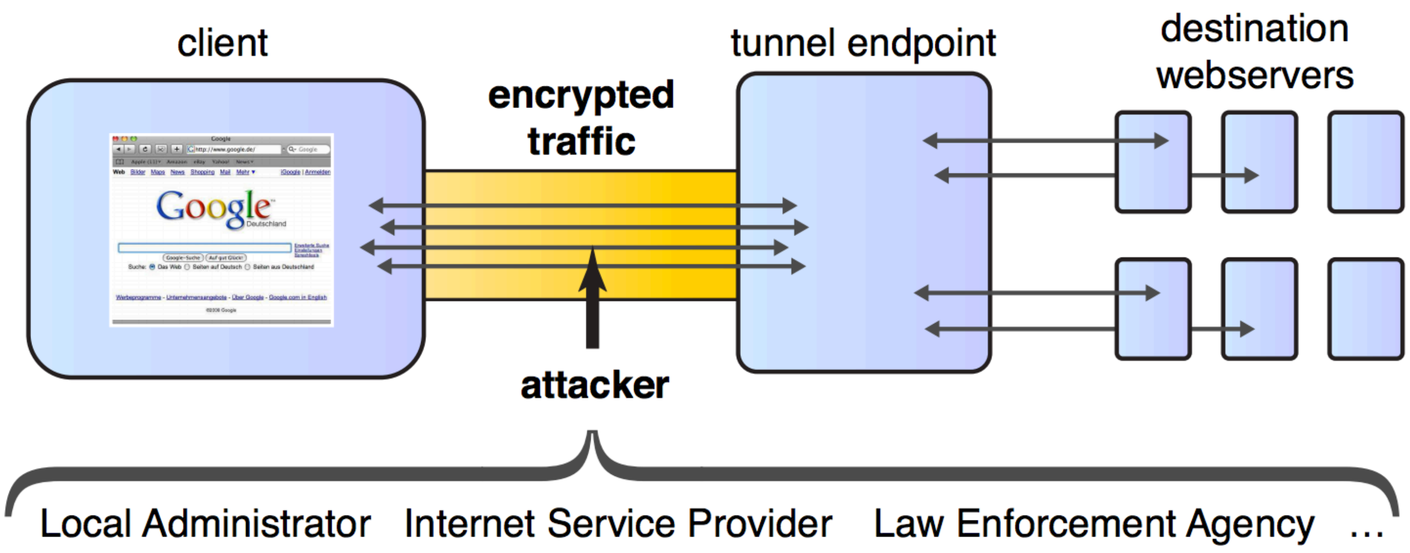
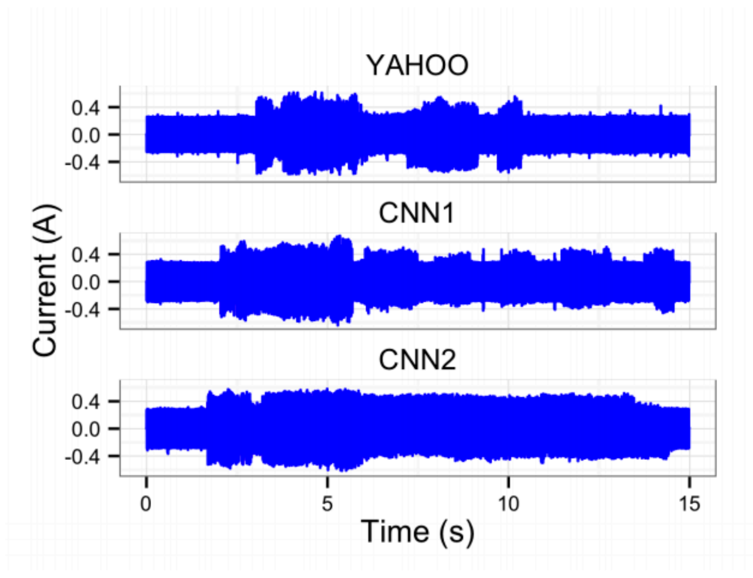


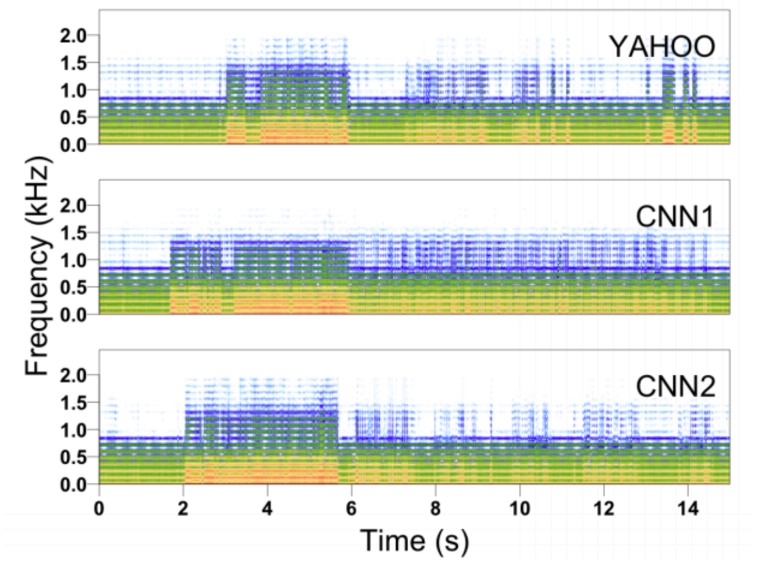
Figure 1: Website fingerprinting scenario and conceivable attackers

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

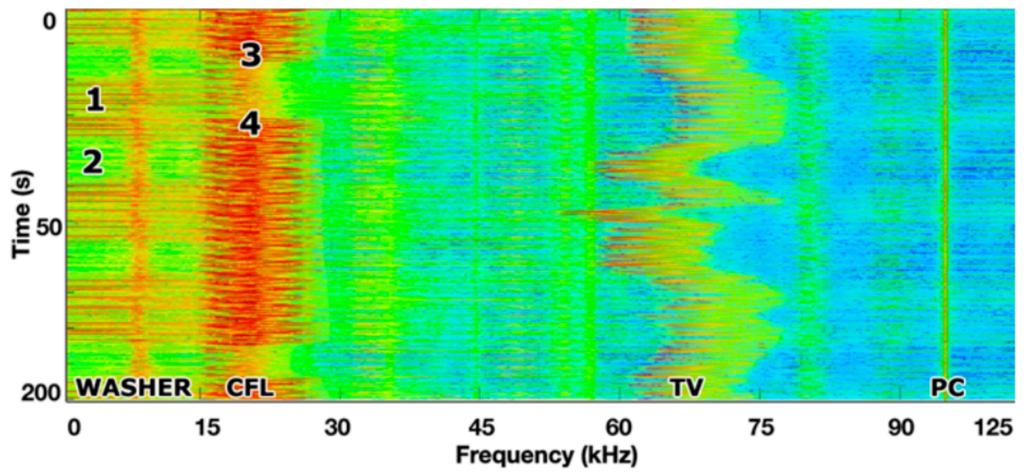
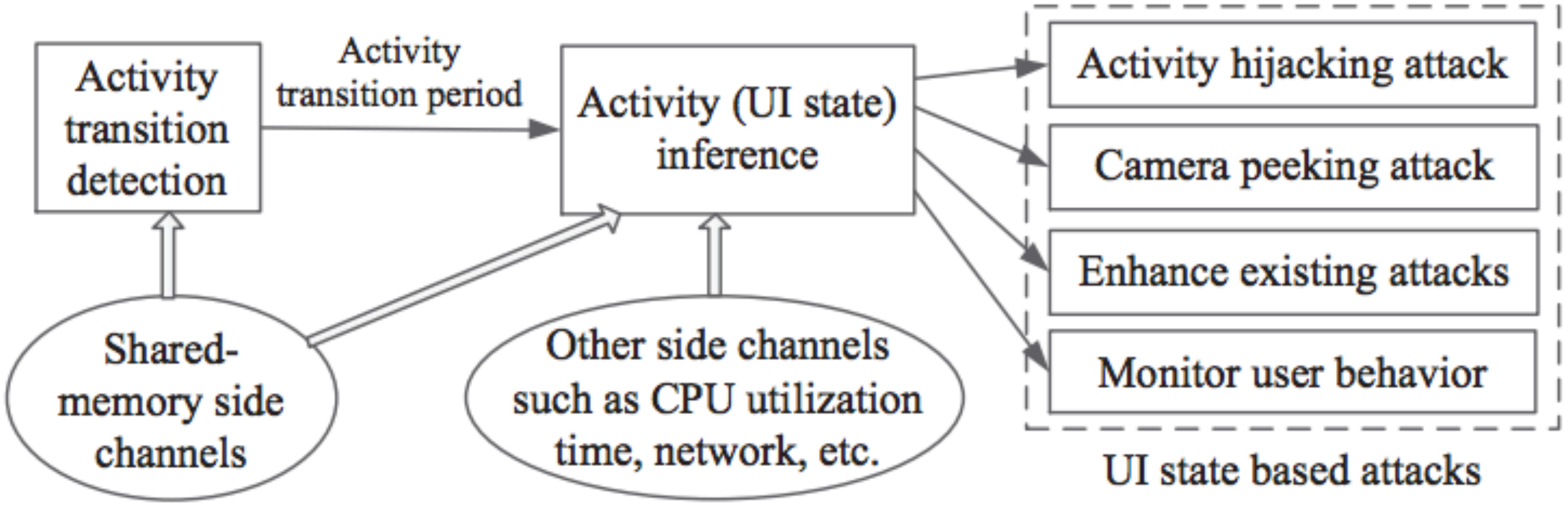


Figure 1: Frequency spectrogram showing various electrical appliances in the home. Washer cycle on (1) and off (2). CFL lamp turning off briefly (3) and then on (4). Note that the TV's (Sharp 42" LCD) EMI shifts in frequency, which happens as screen content changes.

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

Shared Files in Android



Activity transition detection
Activity transition period
Activity (UI state) inference
Activity hijacking attack
Camera peeking attack
Enhance existing attacks
Monitor user behavior
Shared-memory side channels
Other side channels such as CPU utilization time, network, etc.
UI state based attacks

Figure 5: Activity inference attack overview.

Chen et al., "Peeking Into Your App Without Actually Seeing It", USENIX Security 2014

Other Side Channels??