CSE 127: Introduction to Security

Lecture 7: Side Channel Attacks

George Obaido

Winter 2022

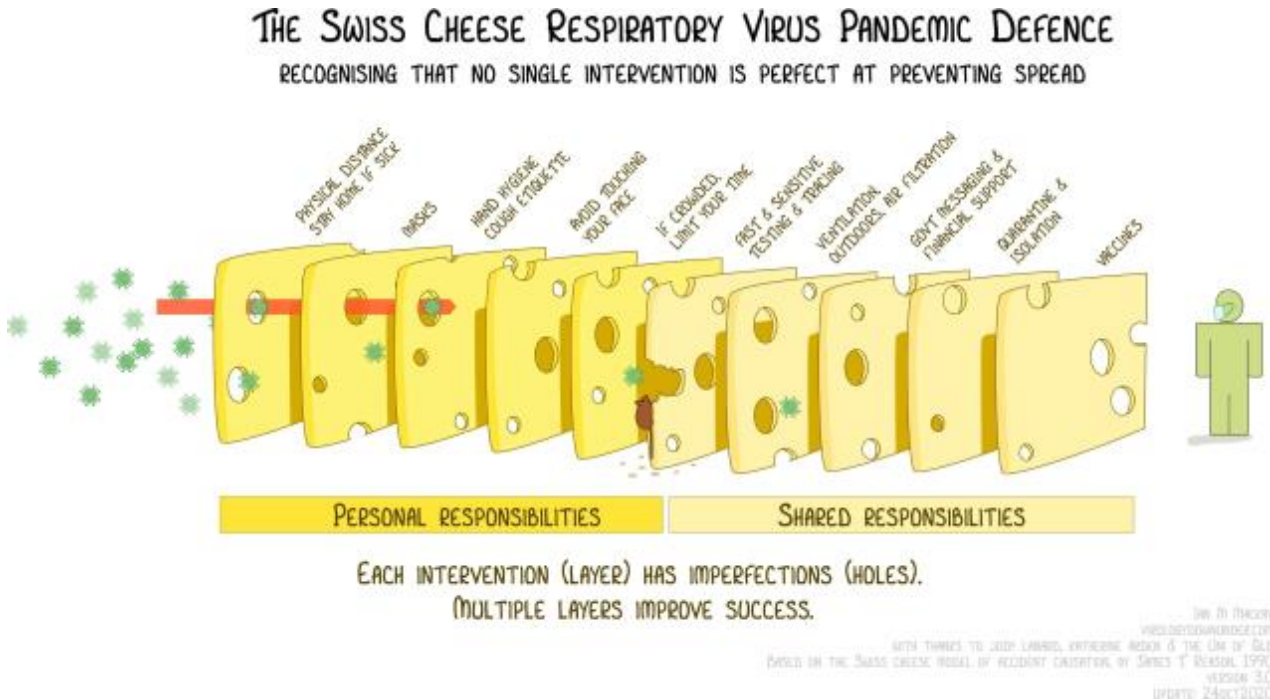
Some material from Dan Boneh, Stefan Savage, Deian Stefan, Keegan Ryan, Nadia Heninger

From Last Time:

Principles: Defense in depth

5. Principles: Defense in depth

We do not expect any of our defenses to be perfect.



Last. Principles: Keep it simple

6. Principles: Keep it simple

We *have* to trust some components of our system.

In general, keeping the Trusted Computing Base **small and simple** makes it easier to verify.

- In theory a hypervisor can be less complex than a full host operating system.
- A small OS kernel has less attack surface than one with many features.

Principles of secure system design

1. Least privilege
2. Privilege separation
3. Complete mediation
4. Fail safe/closed
5. Defense in depth
6. Keep it simple

How can attackers access protected data?

- Find a bug in an unprotected program
- Find a bug in the kernel, VMM, or runtime system providing protection
- Find a hardware bug that lets you bypass isolation

The power of abstraction in computer science

"All problems in computer science can be solved by another level of indirection." – David Wheeler

- Computer systems are often built on layers of abstraction
- Physics → hardware → operating system → applications
- An ideal abstraction allows each layer to treat the layer below as a black box with well-defined behavior

Side channels

Implementations have artifacts and side effects

- How long, how fast, how loud, how hot
- A side channel is a source of information beyond the output specified by an abstraction.
    - Mostly “unintended” emissions of information.

Today

- Overview and history of side channels
- Cache side channels and countermeasures

Soviet Great Seal Bug

- 1945 Soviet gift to US ambassador
- Contained passive listening device
- Would transmit when illuminated at a particular radio frequency
- Discovered 7 years later (in 1952).

- https://en.wikipedia.org/wiki/The_Thing_(listening_device)
- https://www.youtube.com/watch?v=qo4PnkXT2jE
- https://historyofspies.com/great-seal-bug/

TEMPEST: US/NATO side channel codename

- WWII: Bell Telephone discovers electromagnetic leakage in one-time pad teleprinters: 100-ft radius
- 1951: CIA rediscovers teleprinter leakage; 200-ft radius
- 1964: TEMPEST shielding rules established

https://military-history.fandom.com/wiki/Tempest_(codename)

van Eck Phreaking

"Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?" Wim van Eck 1985

- 1985: Wim van Eck demonstrates side channel image recovery from CRT monitors with off-the-shelf equipment

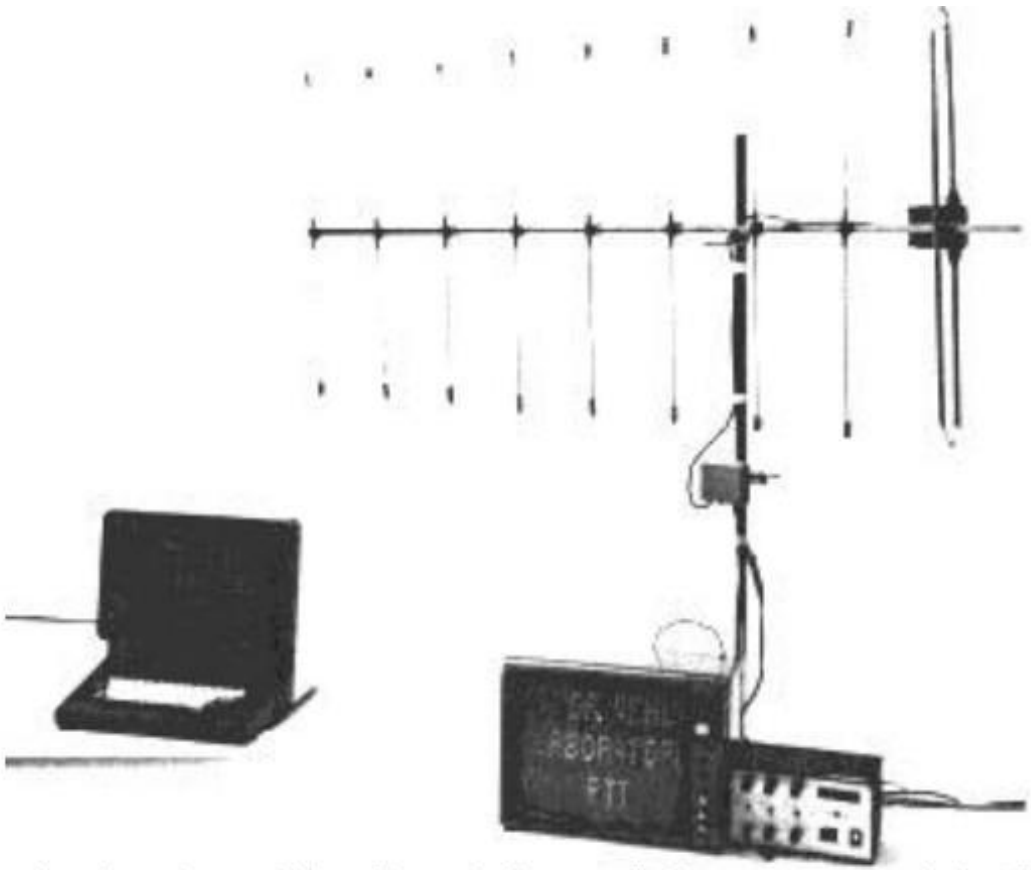
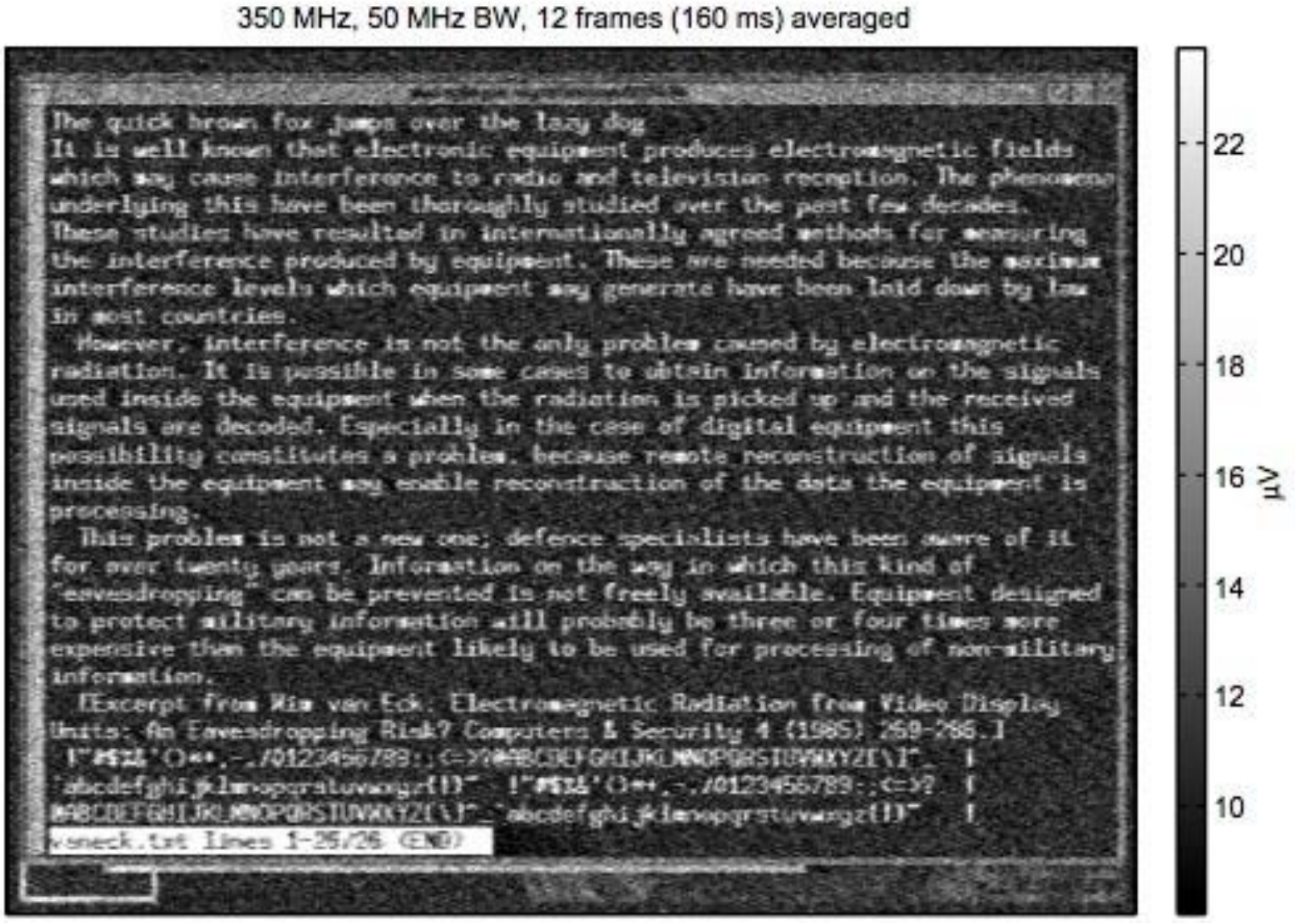


Fig. 1. Eavesdropping eel-up using a variable oscillator and a frequency divider to restore synchronization. The picture on the TV is picked up from the radiation of the VDU in the background.

https://www.youtube.com/watch?v=mcV6izFG3vQ

“Electromagnetic Eavesdropping Risks of Flat-Panel Displays” Kuhn 2004

- Image displays simultaneously along line
- Pick up radiation from screen connection cable



Examples of side channels

Consumption: How much of a resource is being used to perform an operation?

- Timing
  - Different execution time due to program branches
  - Cache timing attacks
- Power consumption: Consumption from Microprocessors.
- Network traffic: Leaks through packet sizes.

Emission: What out-of-band signal is generated in the course of performing the operation?

- Electromagnetic radiation
  - Voltage running through a wire produces a magnetic field
- Sound (acoustic attacks)
  - Capacitors discharging can make noises
- Many other attacks exist!

Consumption side channels

How long does this password check take?

```
char pwd[] = "z2n34uzbnqhw4i";

//...

int check_password(char *buf) {
    return strcmp(buf, pwd);
}
```

“Timing Analysis of Keystrokes and Timing Attacks on SSH”

Song Wagner Tian 2001

- In interactive SSH, keystrokes sent in individual packets
- Build model of inter-keystroke delays by finger, key pair
- Measure packet timing off network.

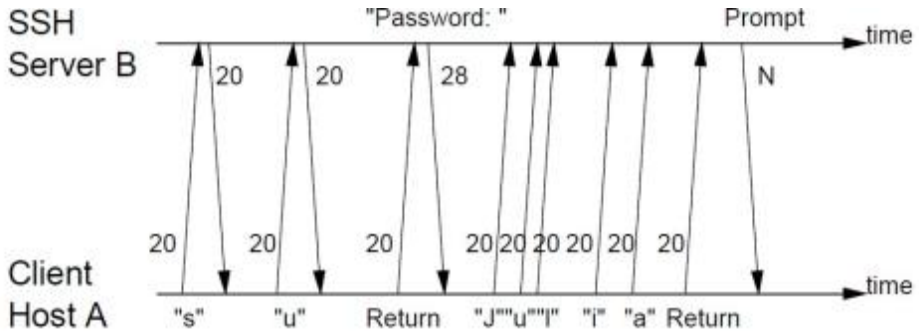


Figure 1: The traffic signature associated with running SU in a SSH session. The numbers in the figure are the size (in bytes) of the corresponding packet payloads.

https://www.usenix.org/conference/10th-usenix-security-symposium/timing-analysis-keystrokes-and-timing-attacks-ssh

Power Analysis Attacks

Kocher Jaffe Jun 98

Side-channel attacks can also leak cryptographic secrets.

Simple power analysis (SPA) and differential power analysis (DPA) exploit secret-dependent power consumption.

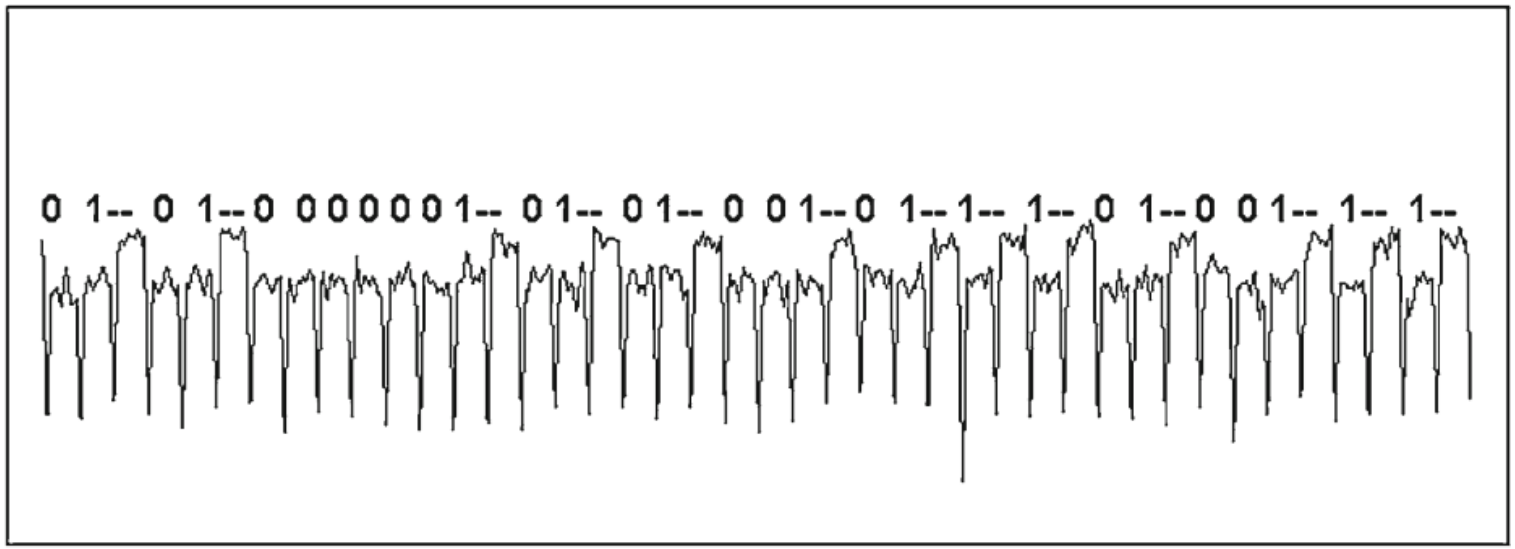
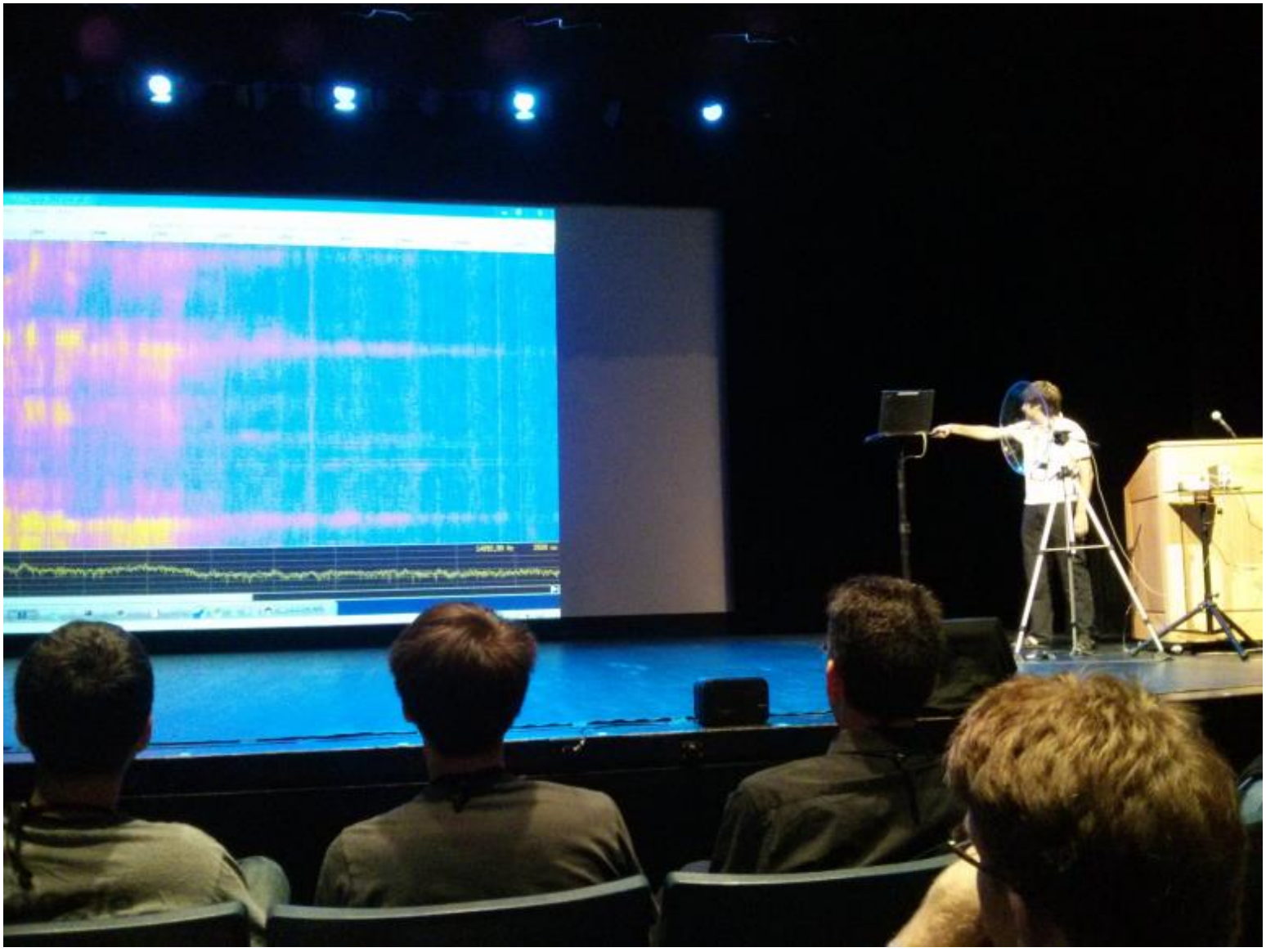


Fig. 11 SPA leaks from an RSA implementation

https://anysilicon.com/side-channel-attacks-differential-power-analysis-dpa-simple-power-analysis-spa-works/

Acoustic Attacks

Genkin Shamir Tromer 2014



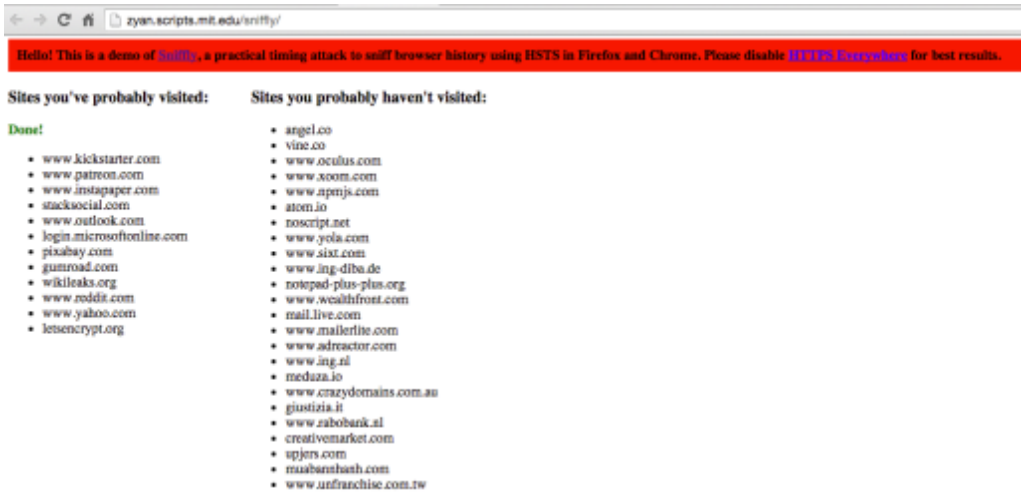
https://www.cnn.com/2018/12/12/health/cuba-acoustic-attack-symptoms-study/index.html

Browser History (BH) Sniffing

Jang, Jhala, Lerner, Shacham 2010

- Default web browser behavior: unvisited links are blue and visited links are purple.
- Text display attributes available to scripts via DOM.
- Victim browser visits malicious website. Malicious website enumerates URLs in invisible portion of site to sniff browser history.
- Fixed in browsers, but surprisingly hard to eliminate all the information leaks.

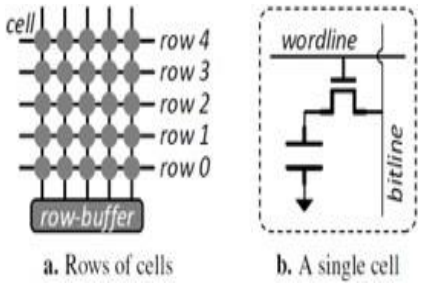
Sniffly: Proof-of-concept BH Sniffing



https://arstechnica.com/information-technology/2015/10/unpatched-browser-weaknesses-can-be-exploited-to-track-millions-of-web-users/

Rowhammer attacks

Seaborn and Dullien 2015

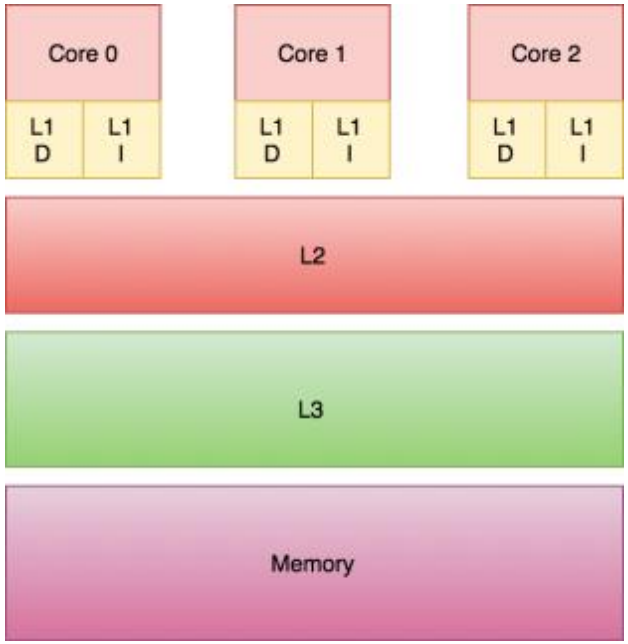


- DRAM cells are grouped into rows
- All cells in a row are refreshed together
  - Repeatedly opening and closing a row within a refresh interval causes disturbance errors in adjacent rows.
  - Attacker running attack process on same machine as victim can cause bits to fip in victim's memory

https://therecord.media/google-says-rowhammer-attacks-are-gaining-range-as-ram-is-getting-smaller/

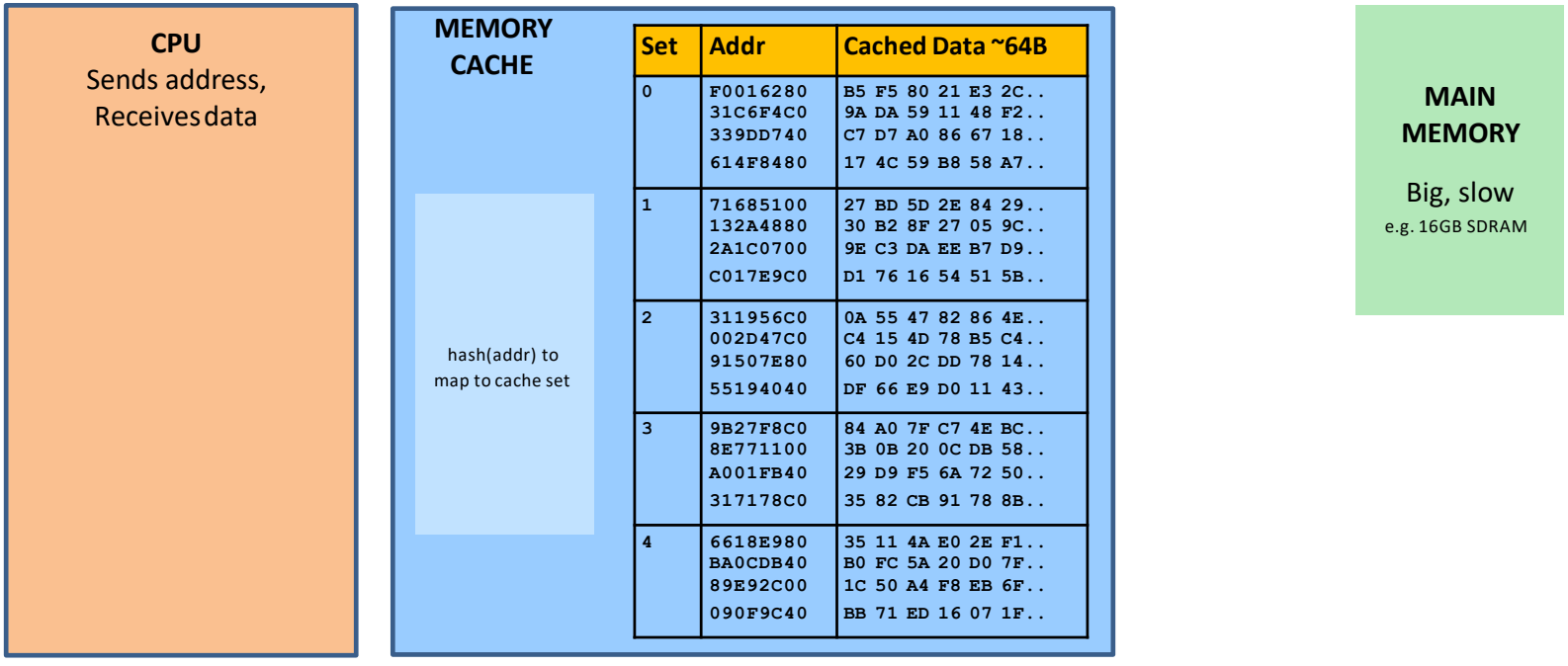
Memory and cache

- Main memory is large and slow
- Processors have faster, smaller caches to store more recently used memory closer to cores
- Caches organized in hierarchy: closer to the core are faster and smaller



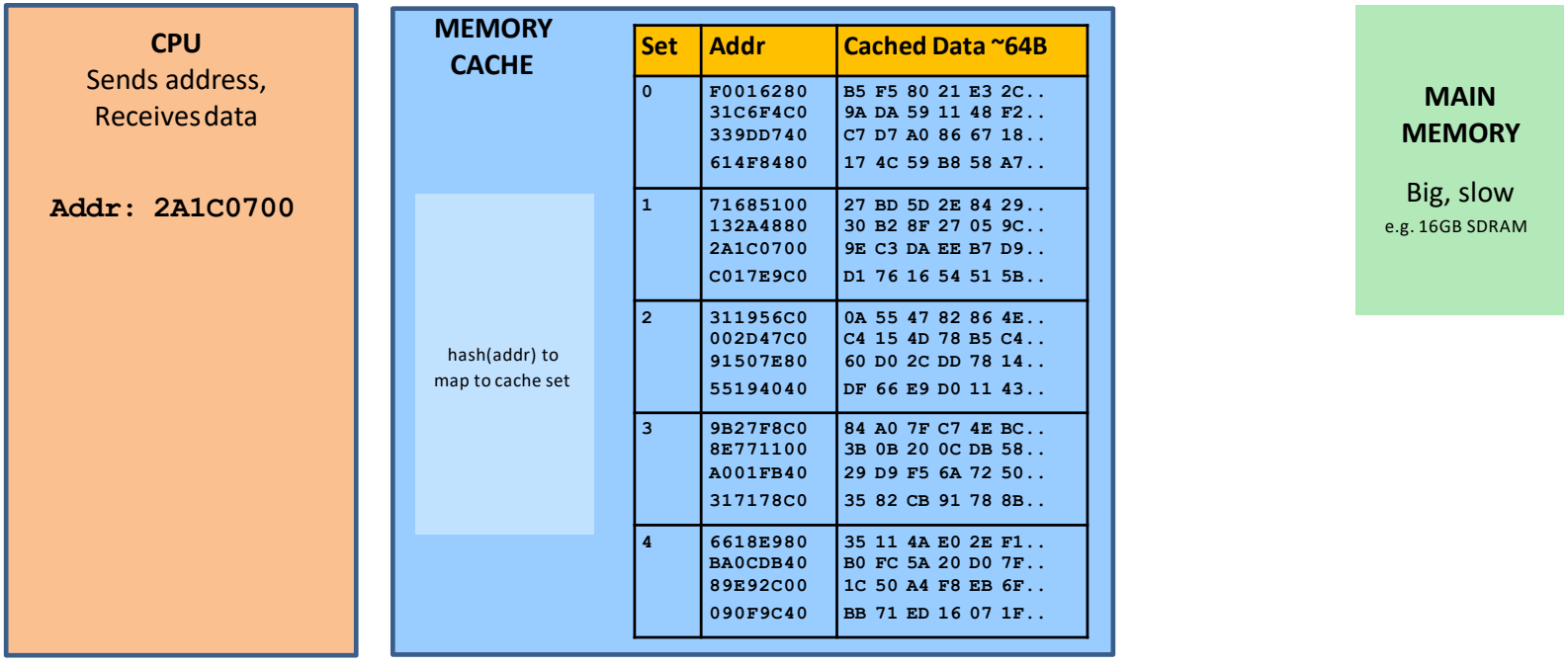
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



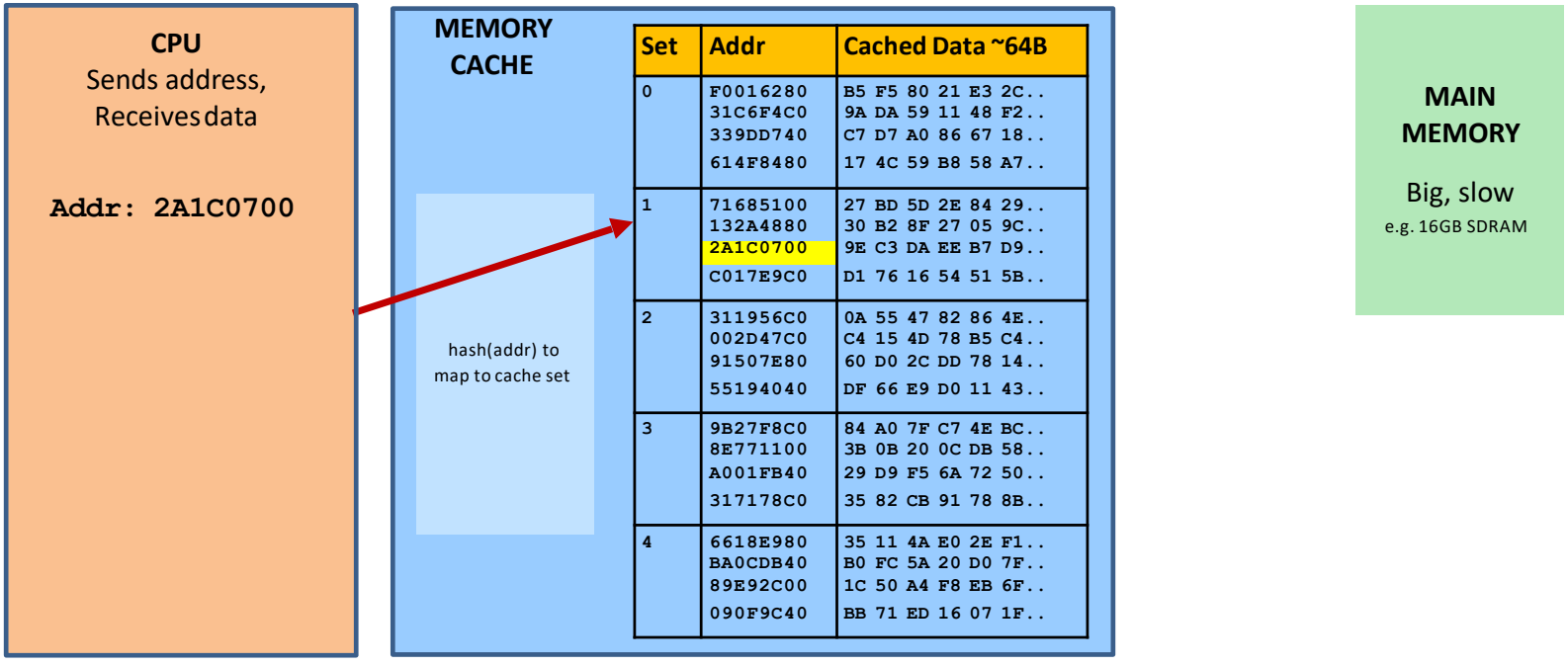
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



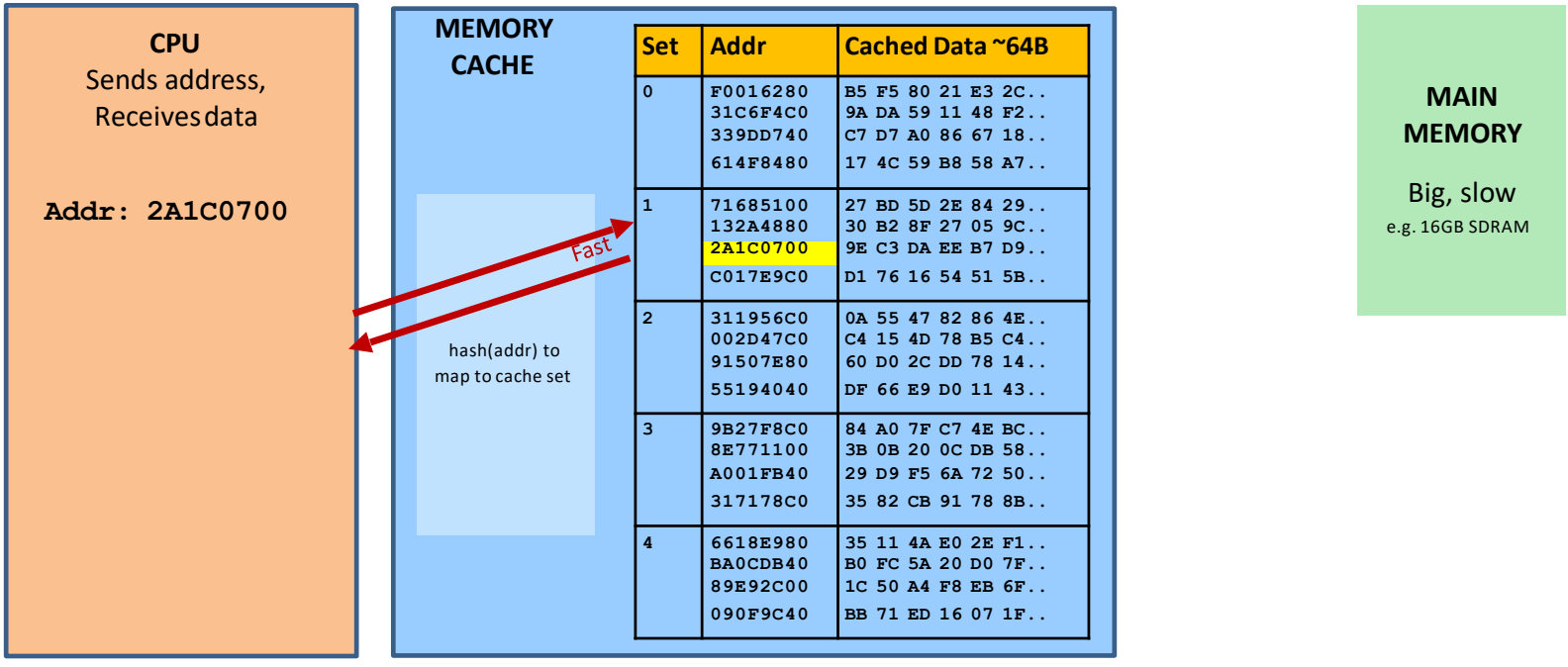
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



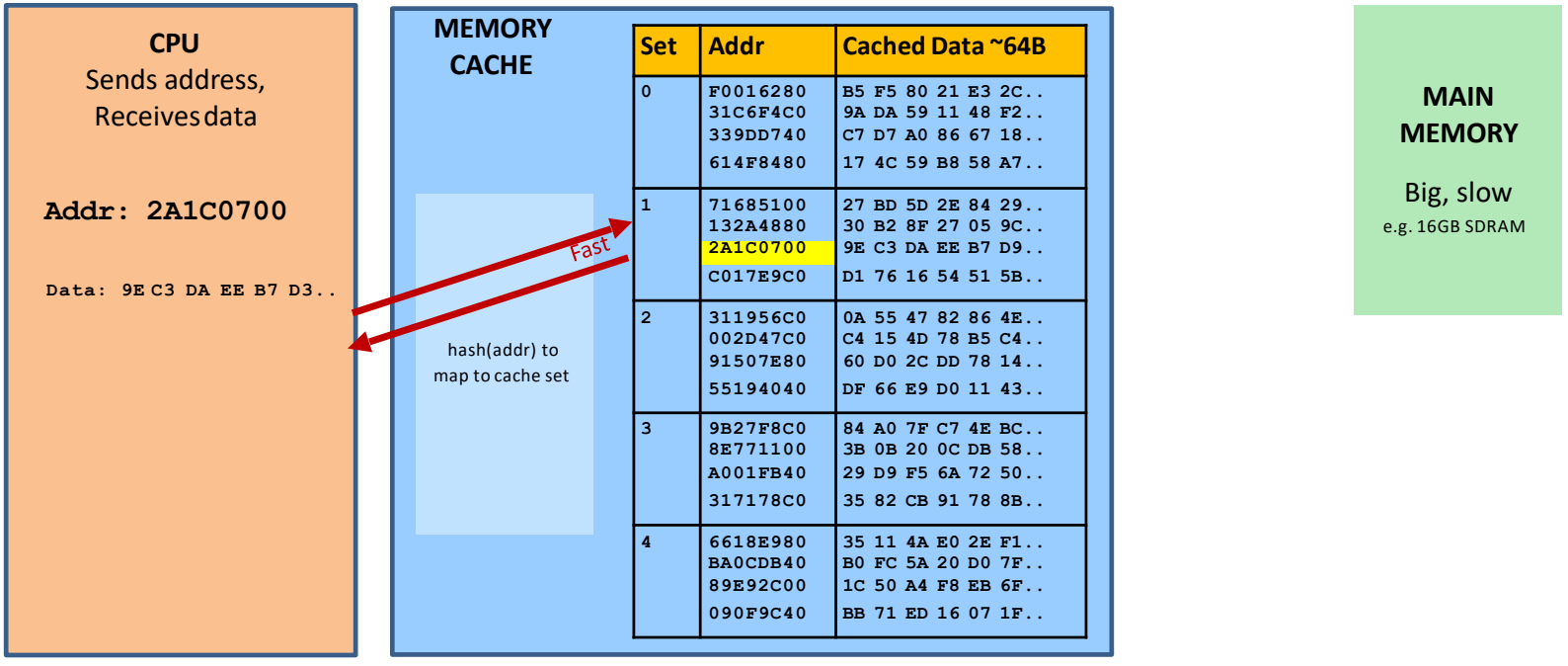
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



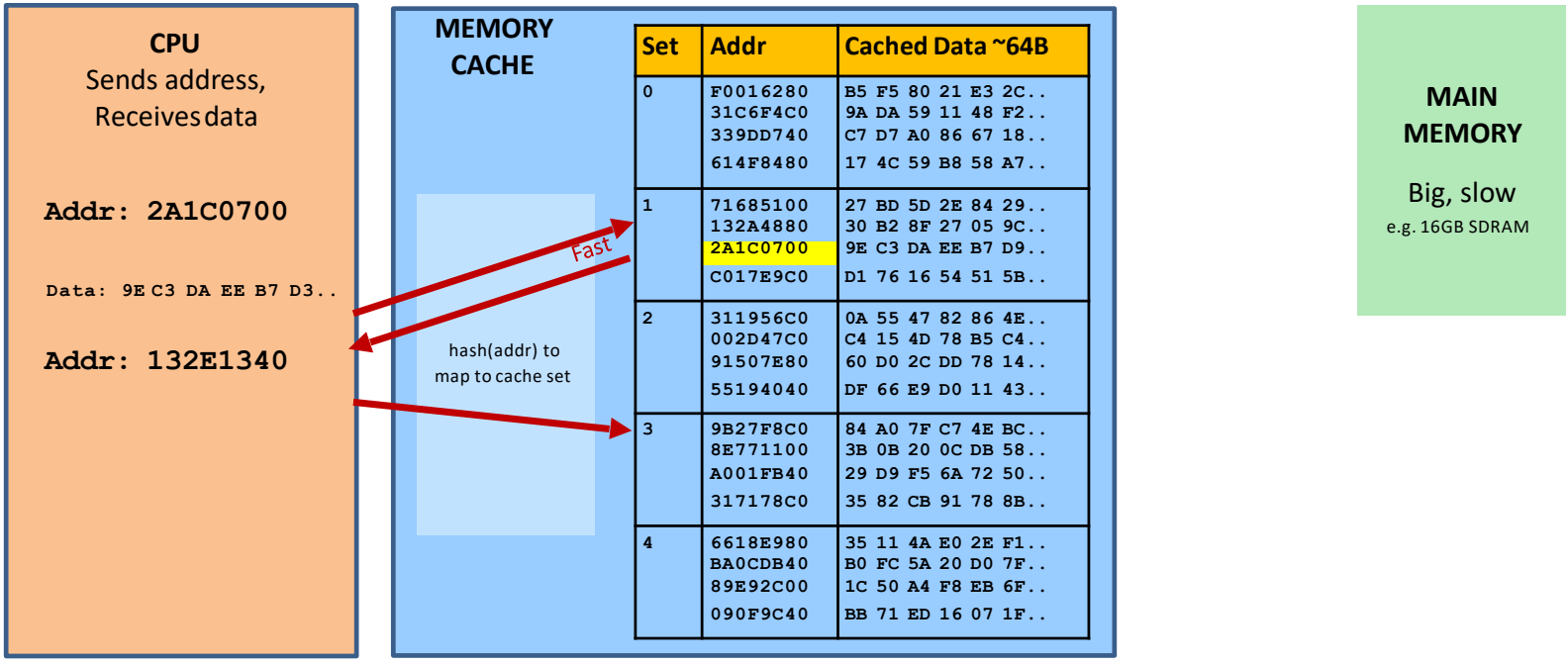
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



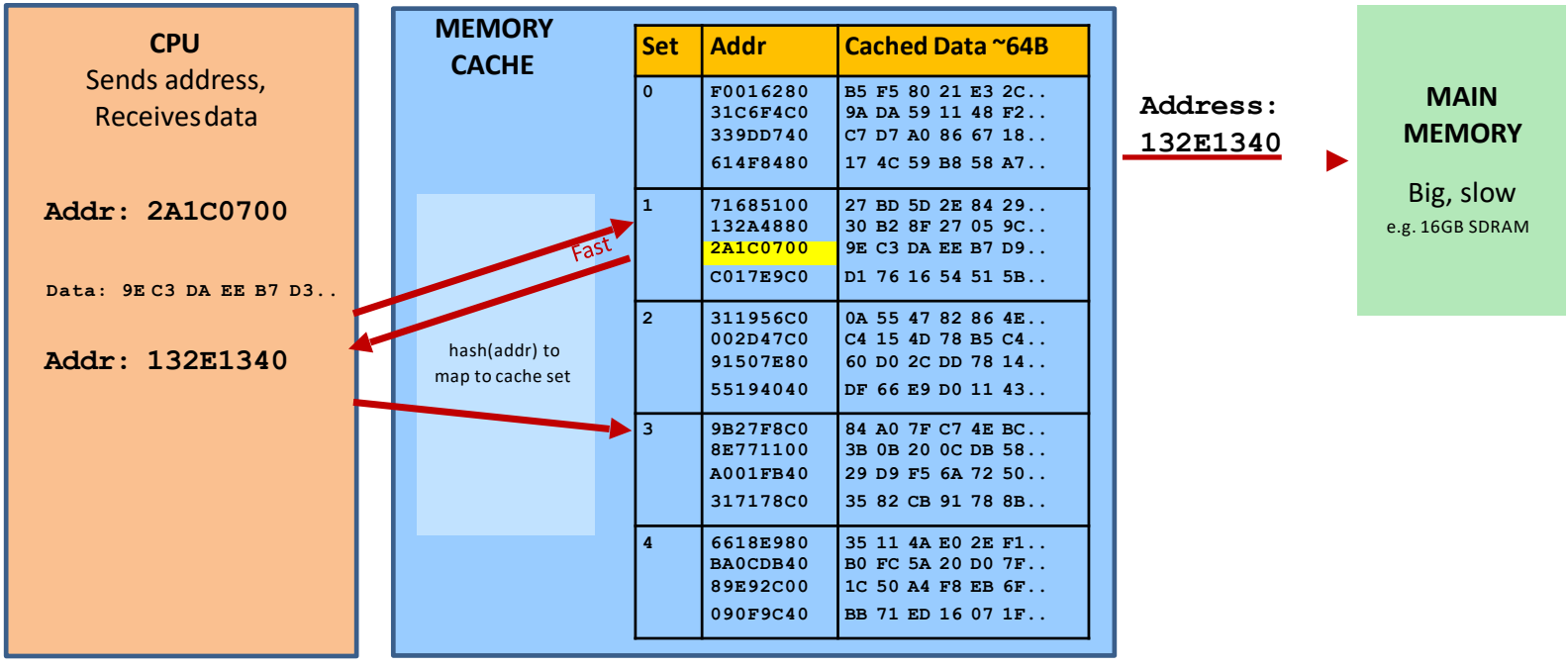
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



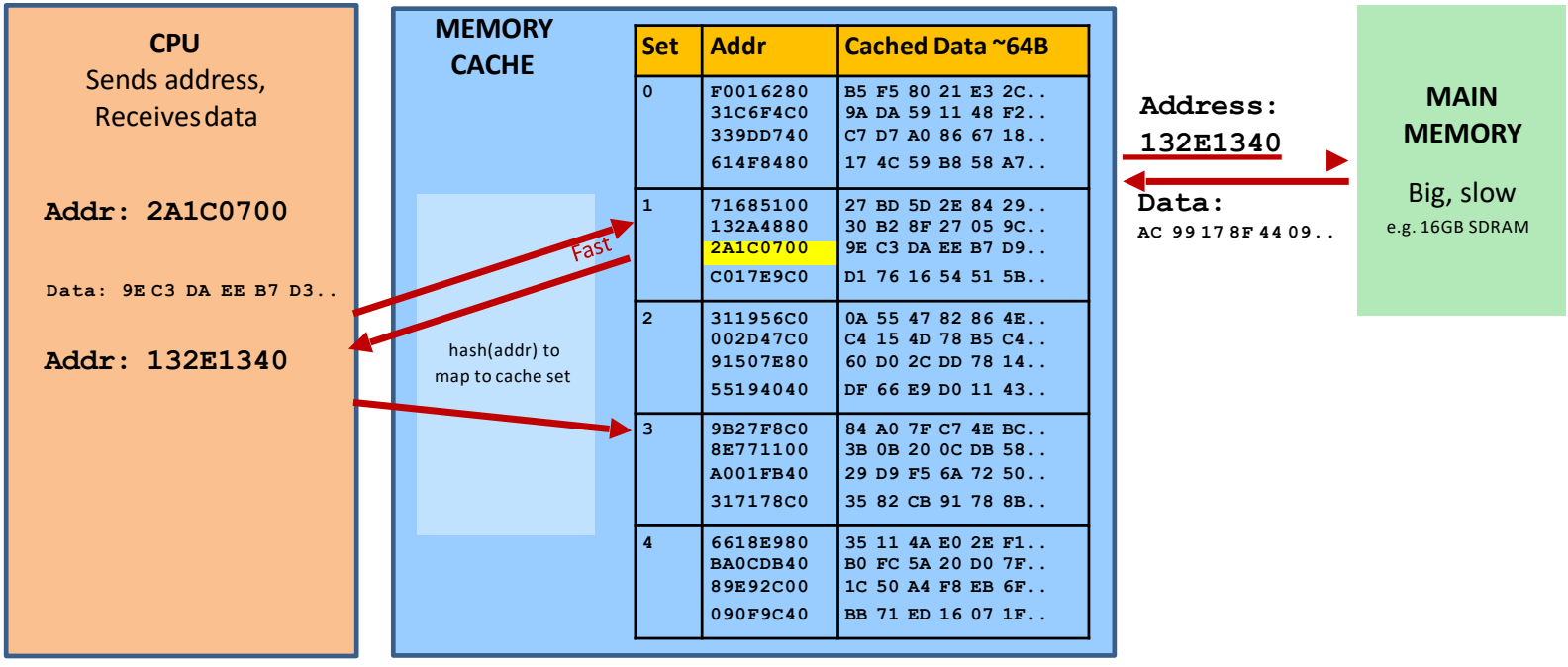
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



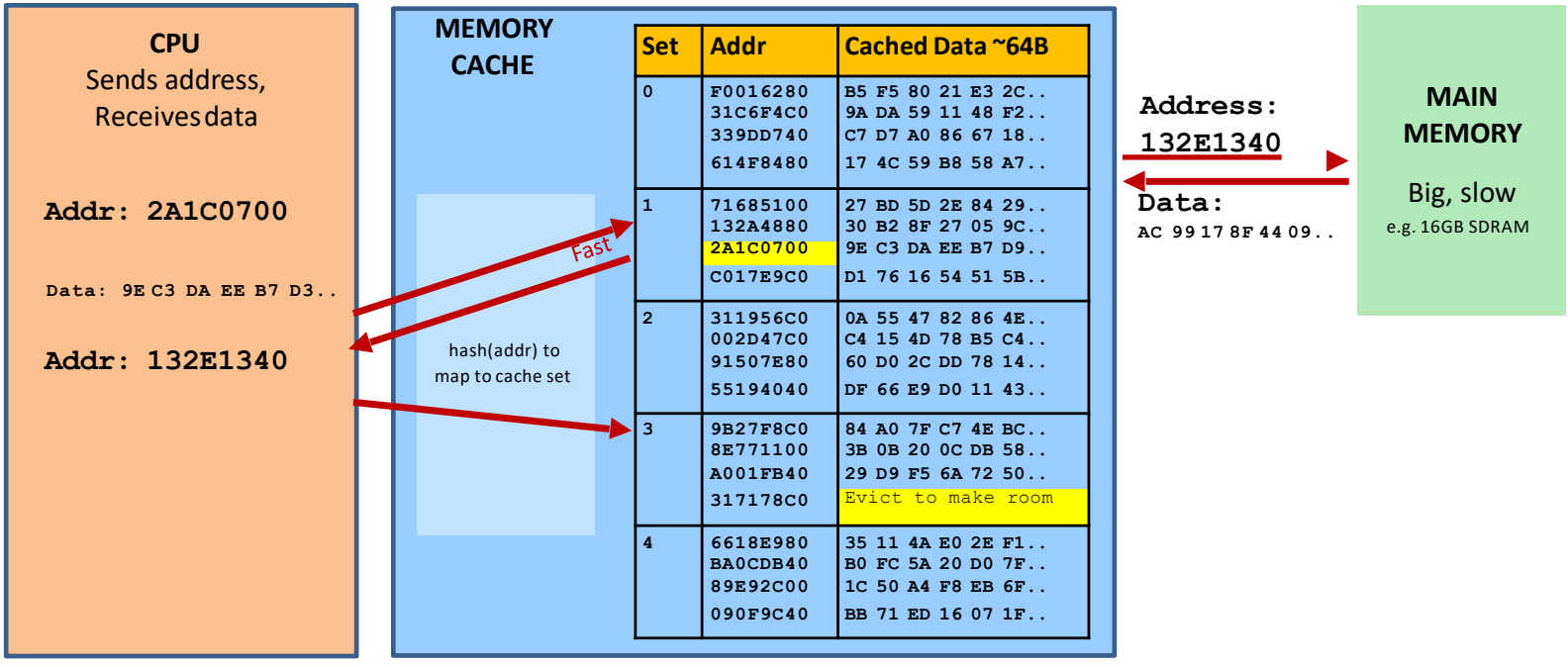
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



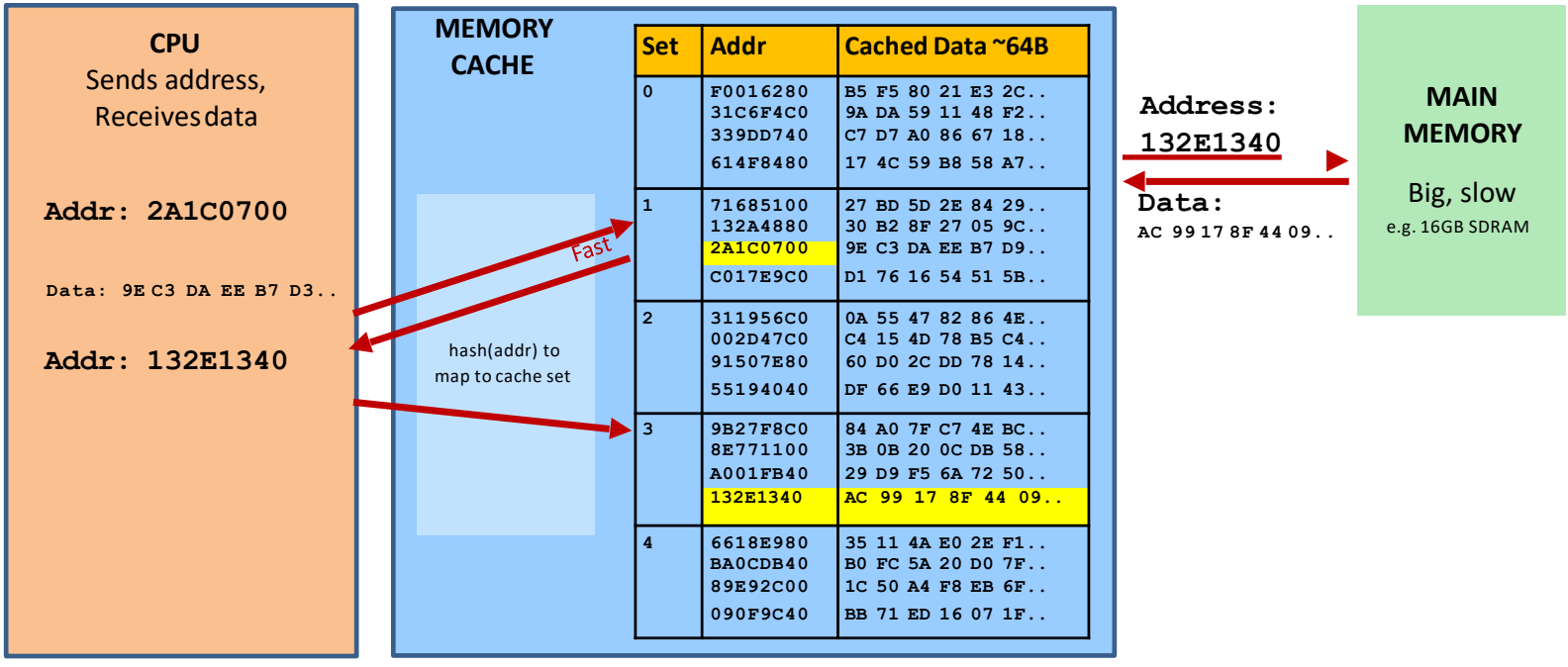
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



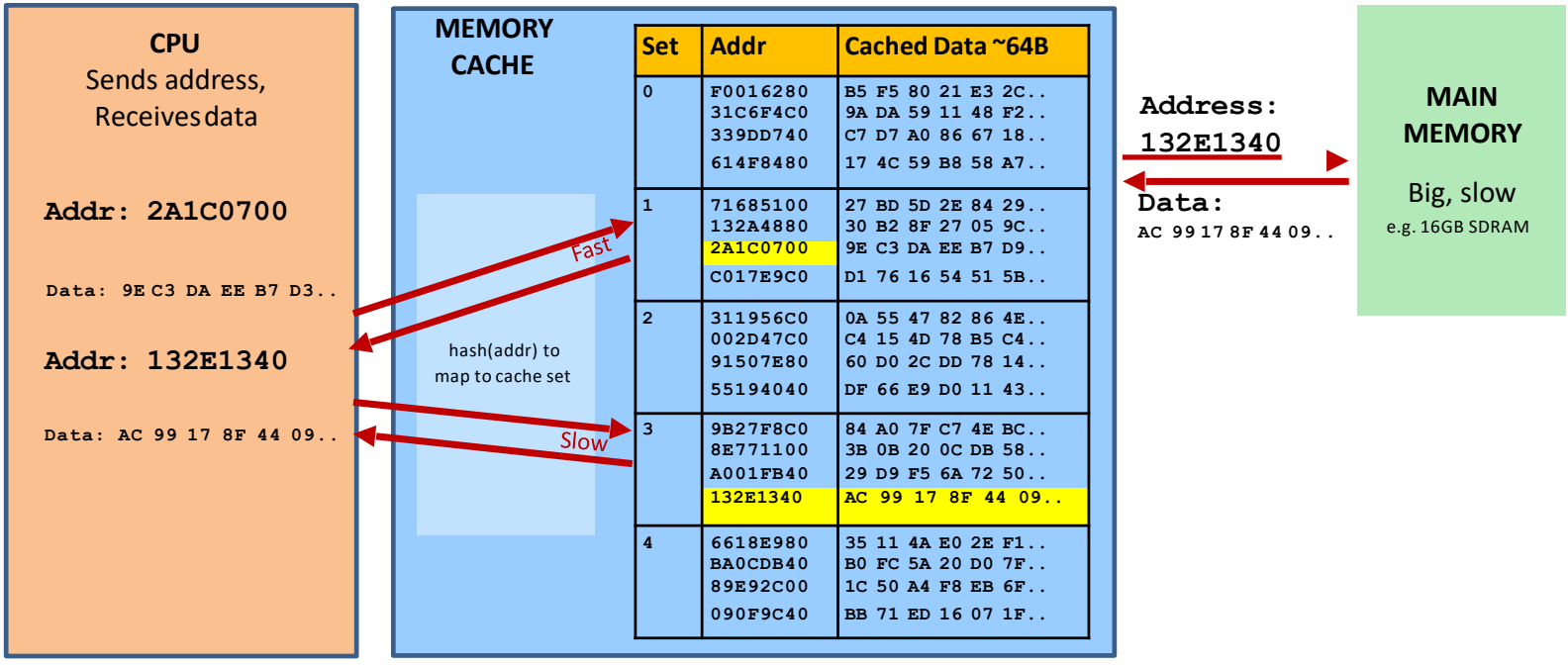
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



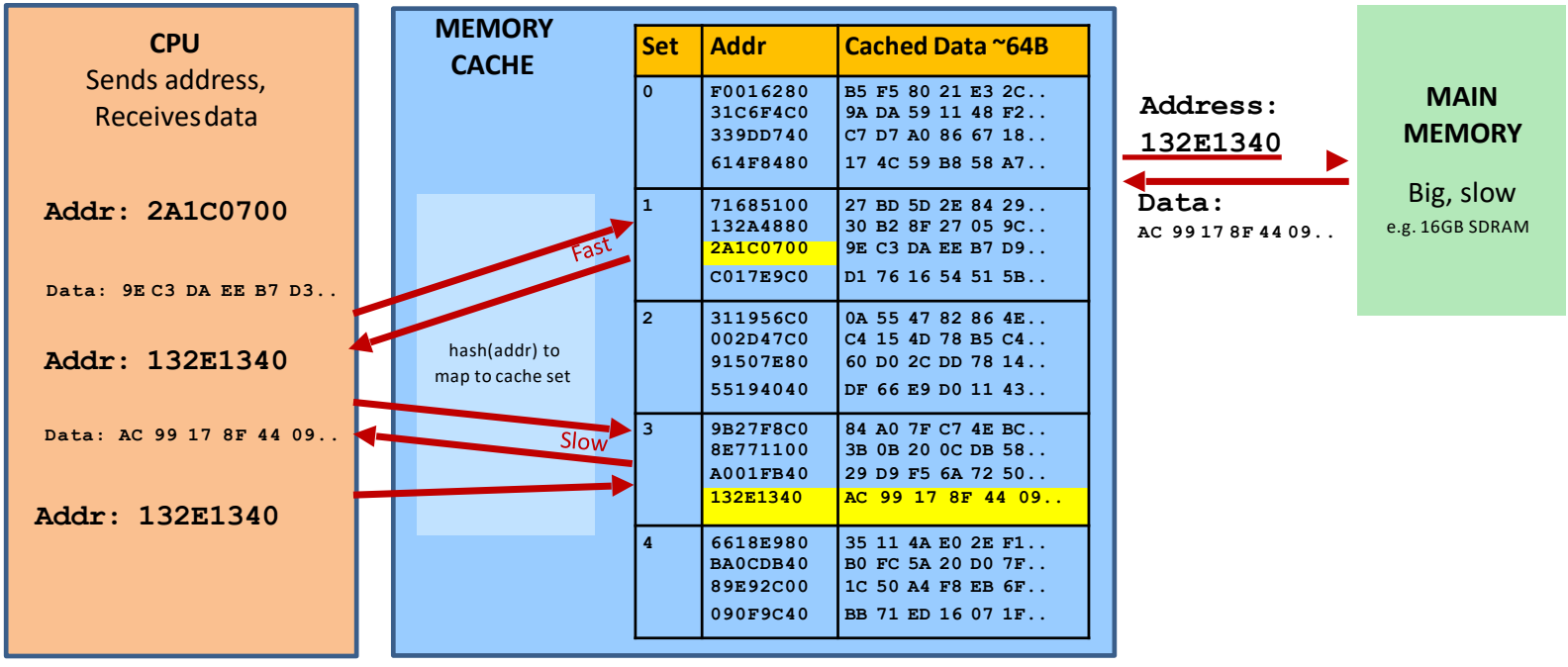
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



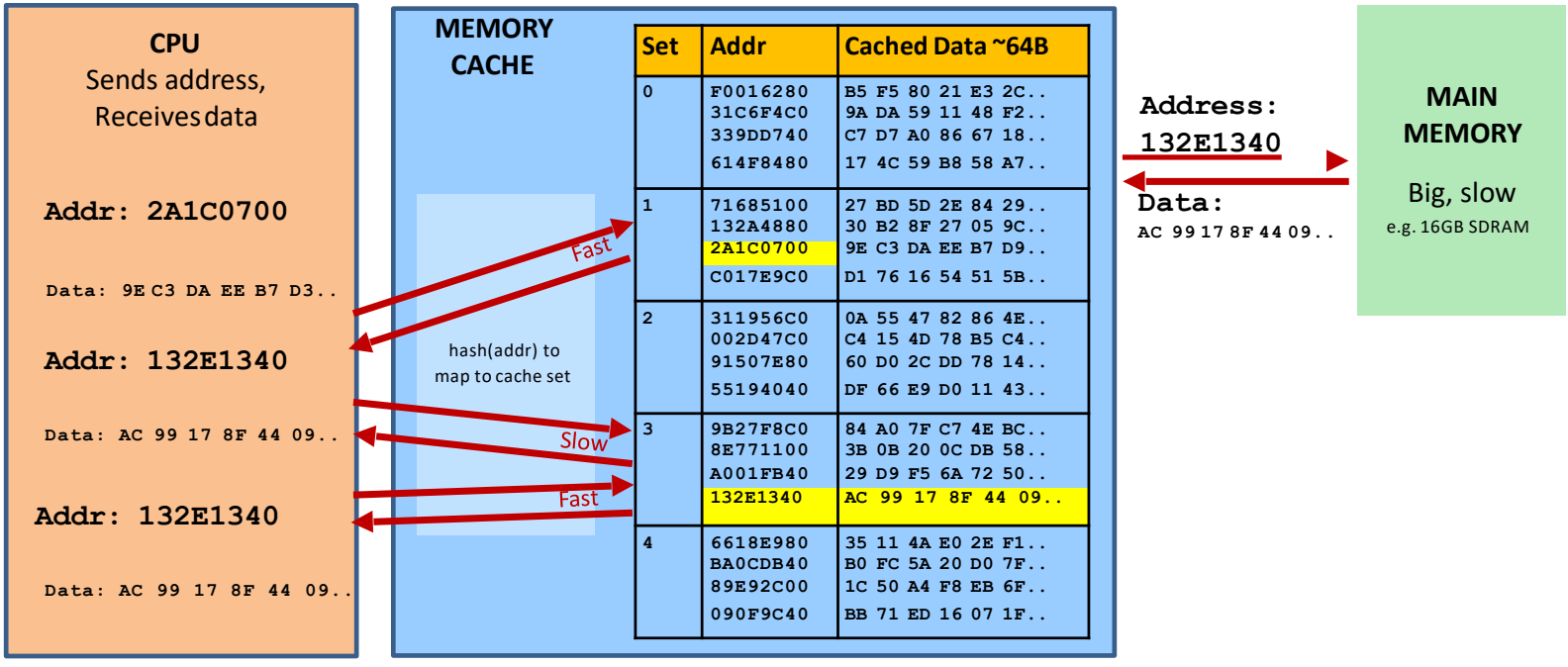
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



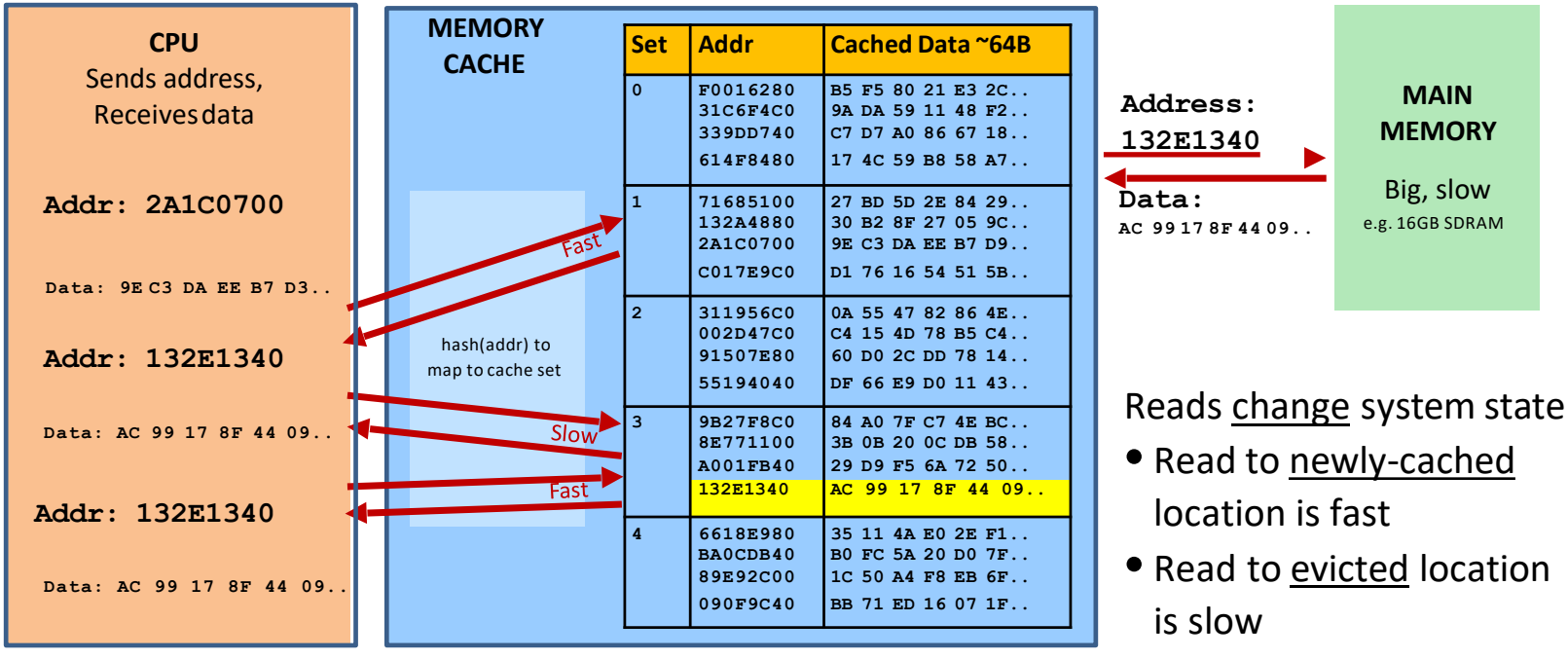
Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



Memory and cache

Caches hold local (fast) copy of recently-accessed 64-byte chunks of memory



Reads change system state:

- Read to newly-cached location is fast
- Read to evicted location is slow

Cache timing side channel attacks

- Caches are a shared system resource
- Not isolated by process, VM, or privilege level
- An attacker who can run code on same physical hardware can abuse this shared resource to learn information from another process.

Cache timing attack options

- **Prime:** Place a known address in the cache by reading it
- **Evict:** Access memory until address is no longer cached (force capacity misses)
- **Flush:** Remove an address from the cache (`clflush` on x86)
- **Measure:** Precisely (down to the cycle) how long it takes to do something (`rdtsc` on x86)
- **Attack form:** Manipulate cache into known state, make victim run, infer what changed after run

Three basic techniques

- Evict and time
  - Evict things from the cache and measure if victim slows down as a result
- Prime and probe
  - Place things in the cache, run the victim, and see if you slow down as result
- Flush and reload
  - Flush a particular line from the cache, run the victim, and see if your accesses are still fast

Next: Mitigating side channels and Web Intro

https://www.microsoft.com/en-us/research/video/verifying-constant-time-implementations/