TEL AVIV UNIVERSITY

Information Security – Theory vs. Reality

0368-4474, Winter 2015-2016

Lecture 1:
Introduction,
Architectural side channels 1/2

Lecturer:

Eran Tromer

Course agenda

Course duties

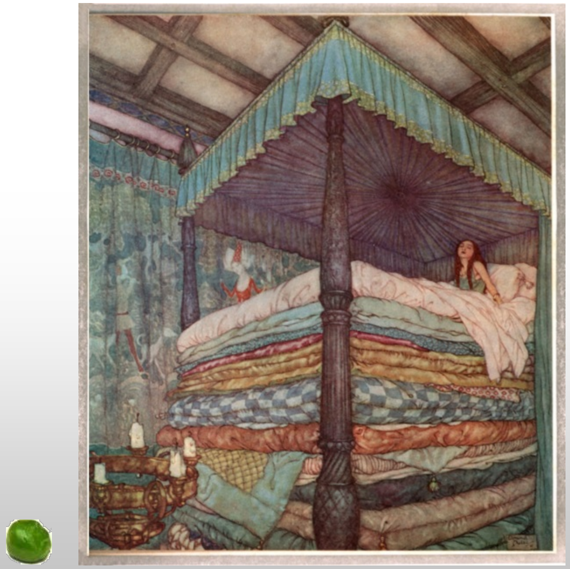
- Questionnaire (on course website)
- Material: everything covered in class (including whiteboard and discussions), and assigned reading as specified.
- Exercises
  - 5 exercises, submitted in individually.
  - 30% of grade
  - All mandatory
  - No late submissions
- Final project
- Lecture summaries: up to 5% bonus

Resources

- Course website:
  `http://cs.tau.ac.il/~tromer/istvr1516`
- Recommended Facebook group: istvr1516
- Mailing list (see website)
- The course material is not covered by any single book. For background and discussion of physical attacks, see:
  Ross Anderson, *Security Engineering*, 2nd ed.
- Additional reading material during the semester.

Course agenda

Advanced topics in applied cryptography and information security, focusing on all the ways our convenient abstractions and careful designs fail in reality – and what to do about it.



Tentative topics

Attacks

- Software side-channel attacks
- Physical side-channel attacks
- Fault attacks
- Hardware security

Defense

- Leakage-resilient cryptography
- Fully-homomorphic encryption
- Computationally-sound proofs

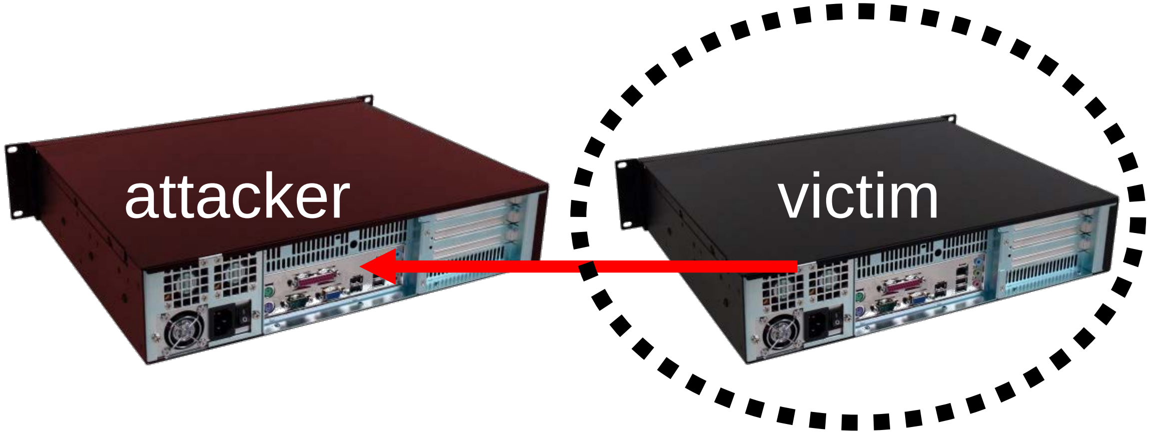
  with applications to Bitcoin
- Multiparty computation
- Obfuscation

Today:
Side-channel attacks

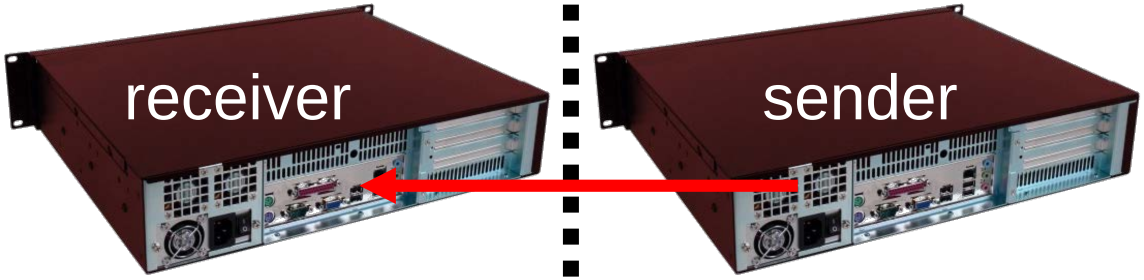
Types of undesired information flow

Inadvertent information channels between processes running on the same system:

- Side channels



- Covert channels
  collaborate to circumvent <u>mandatory access controls</u>



Most generally:

- Violate <u>information flow control</u>

Cryptographic algorithms
vs.
the real world

An example

Cryptographic algorithms

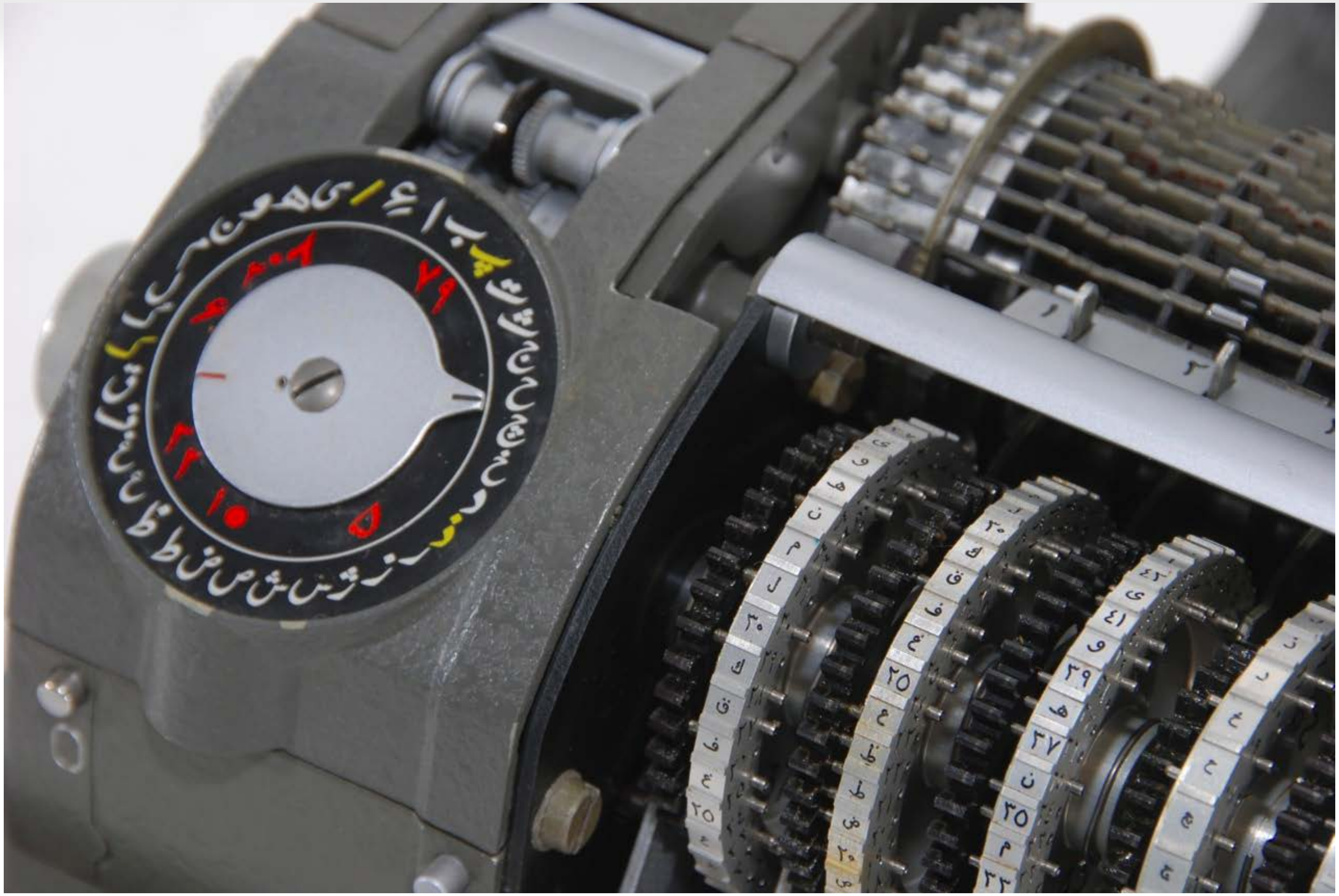
- Model: Input: (plaintext, key) →  → Output (ciphertext)
- Formal security definitions (CPA, CCA1, CCA2, …)
- Well-studied algorithms (RSA, AES, DES, …)
- Algorithmic attacks are believed infeasible.

ENGULF

[Peter Wright, *Spycatcher*, p. 84]

- In 1956, a couple of Post Office engineers fixed a phone at the Egyptian embassy in London.

ENGULF (cont.)

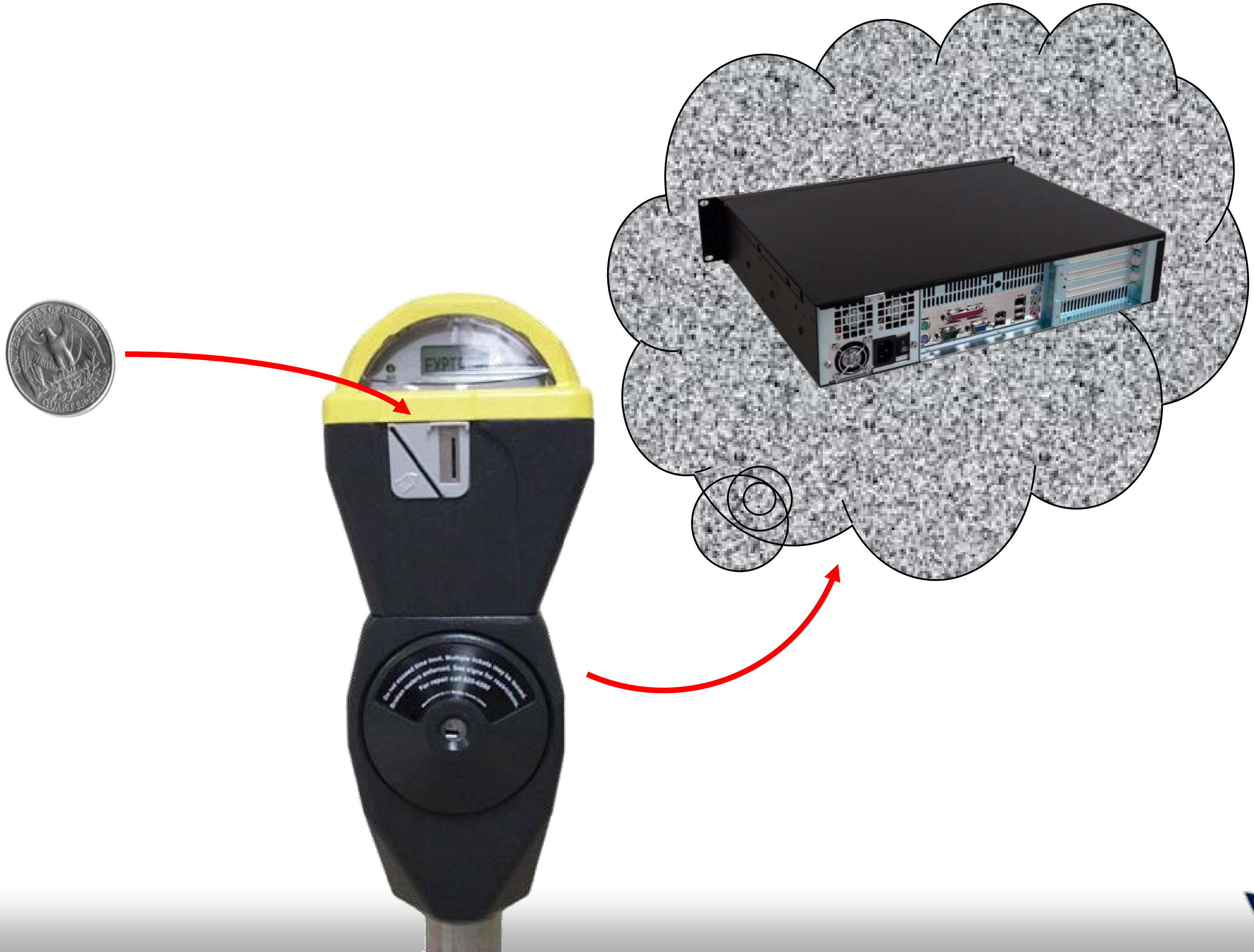


- *“The combined MI5/GCHQ operation enabled us to read the Egyptian ciphers in the London Embassy throughout the Suez Crisis.”*

Architectural side-channel attacks

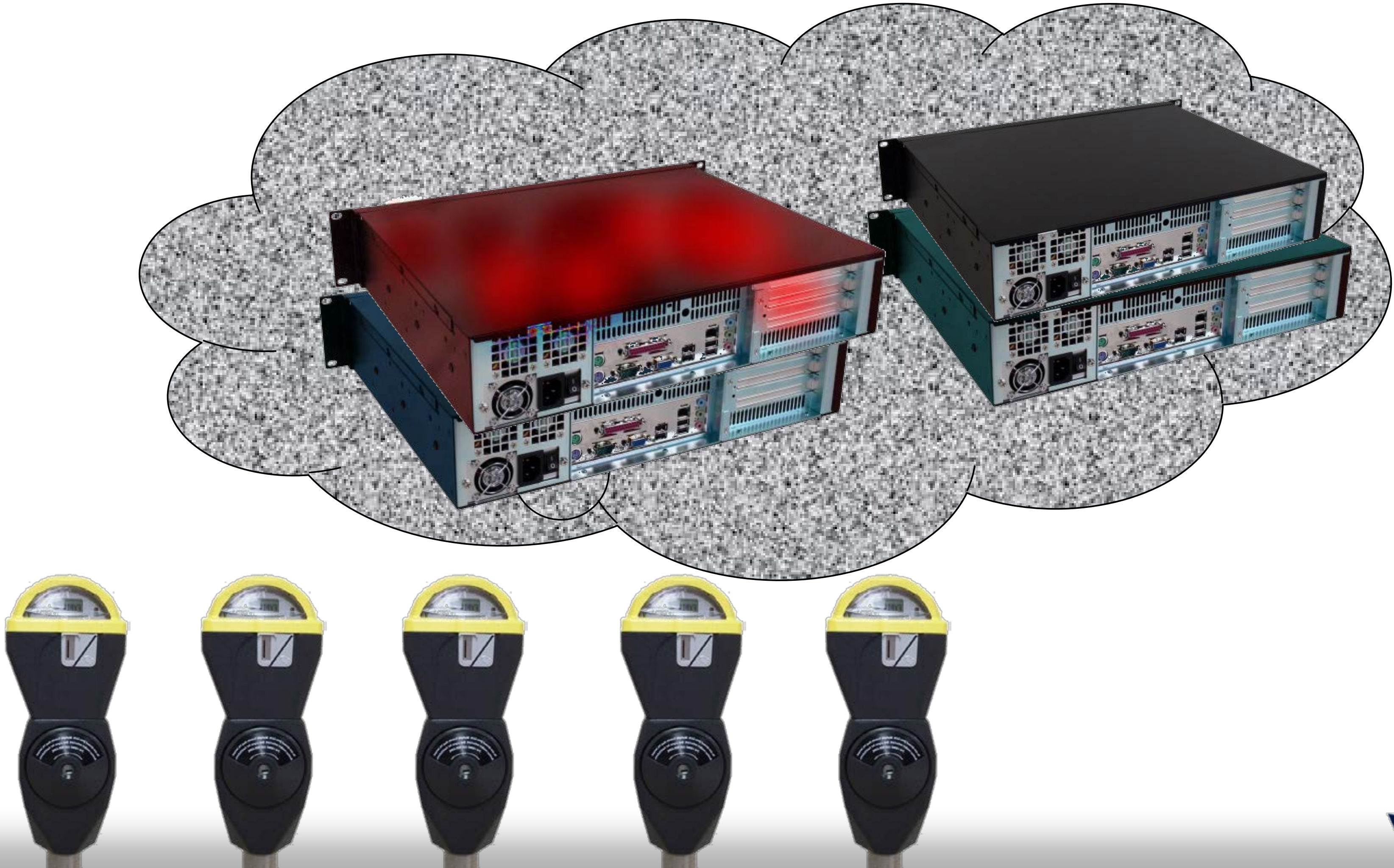
Cloud Computing (Infrastructure as a Service)

Instant virtual machines



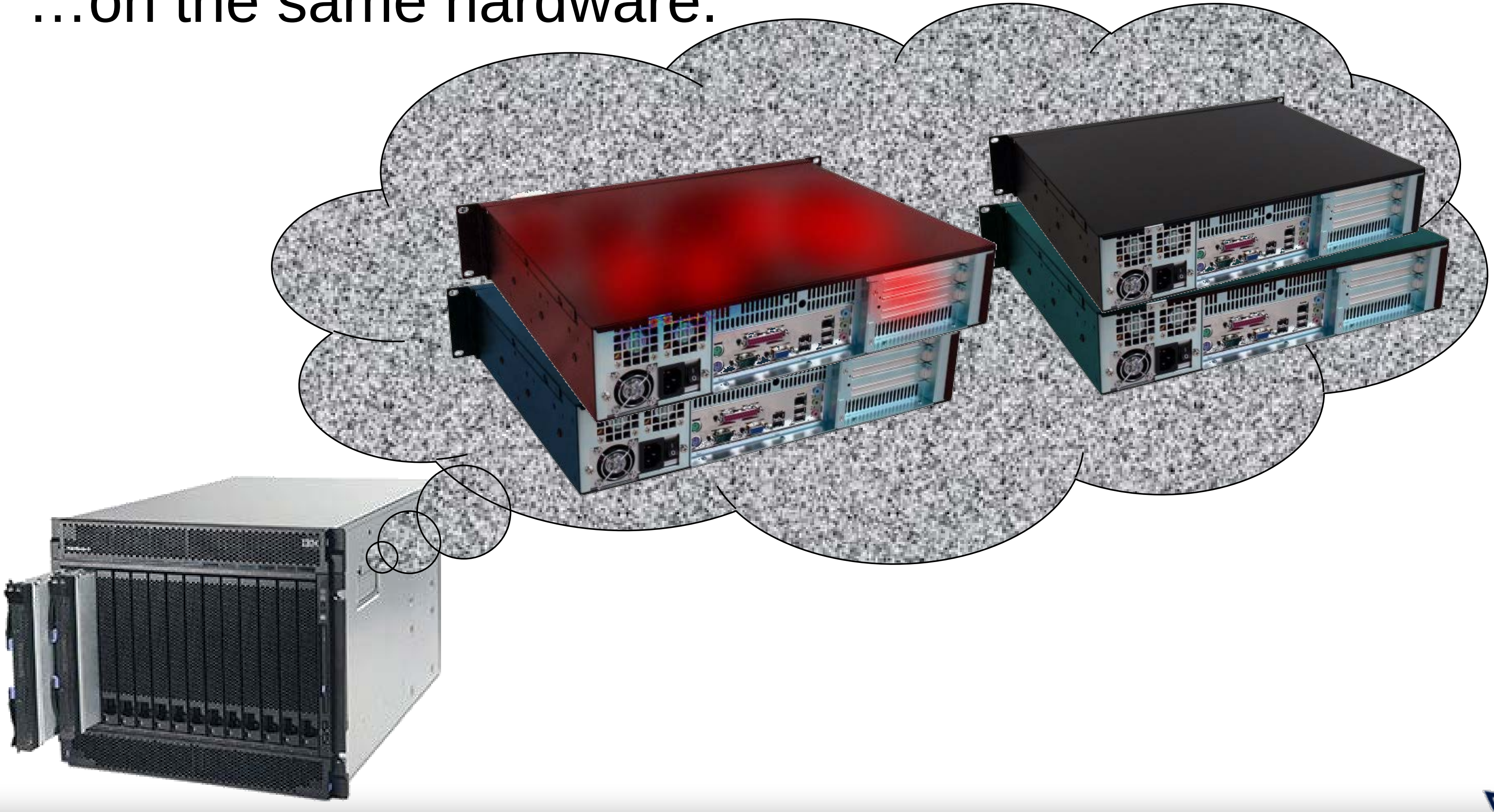
Public Clouds
(Amazon EC2, Microsoft Azure, Rackspace Mosso)

Instant virtual machines
... for anyone



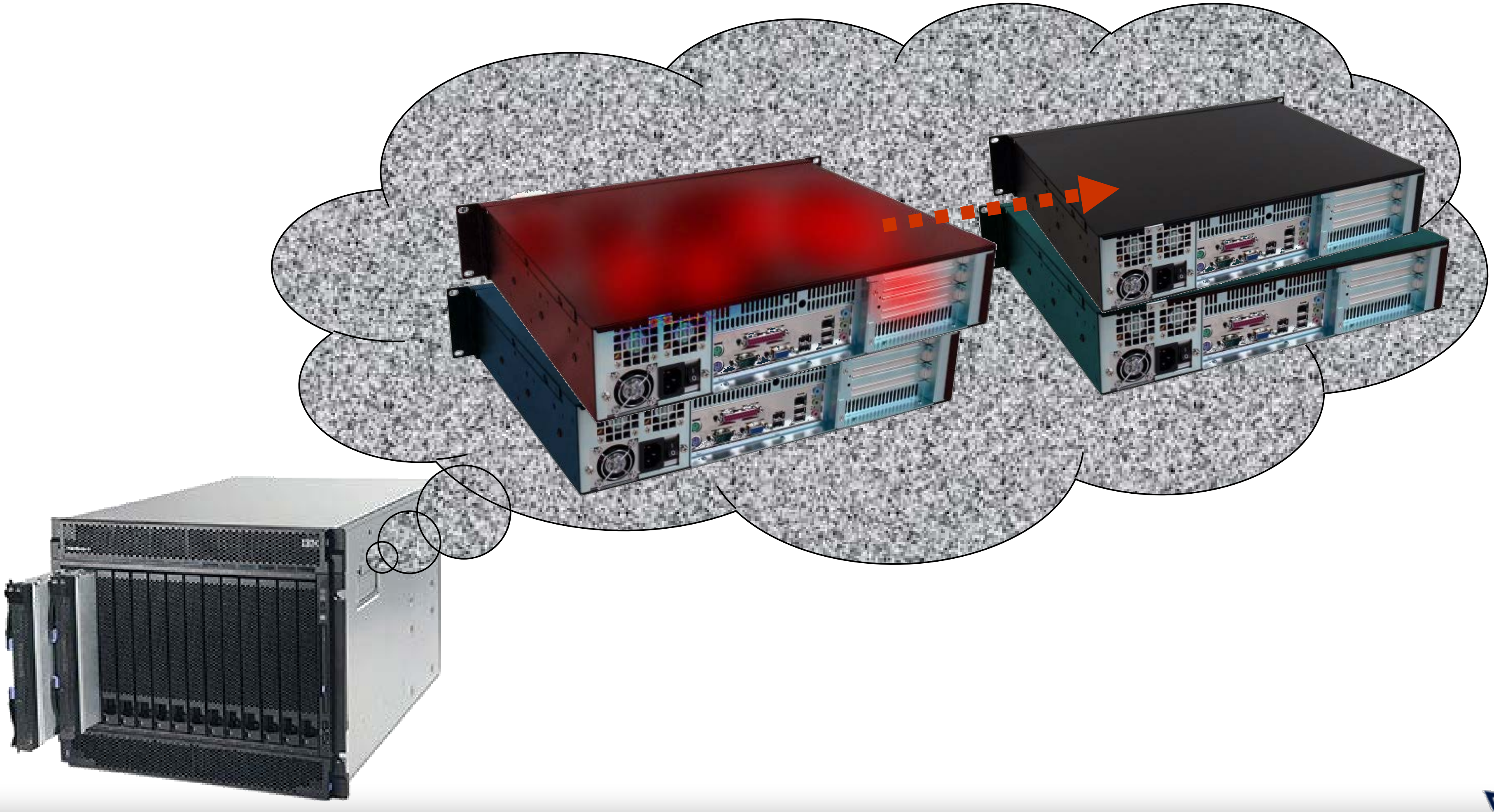
Virtualization

Instant virtual machines
... for anyone
…on the same hardware.

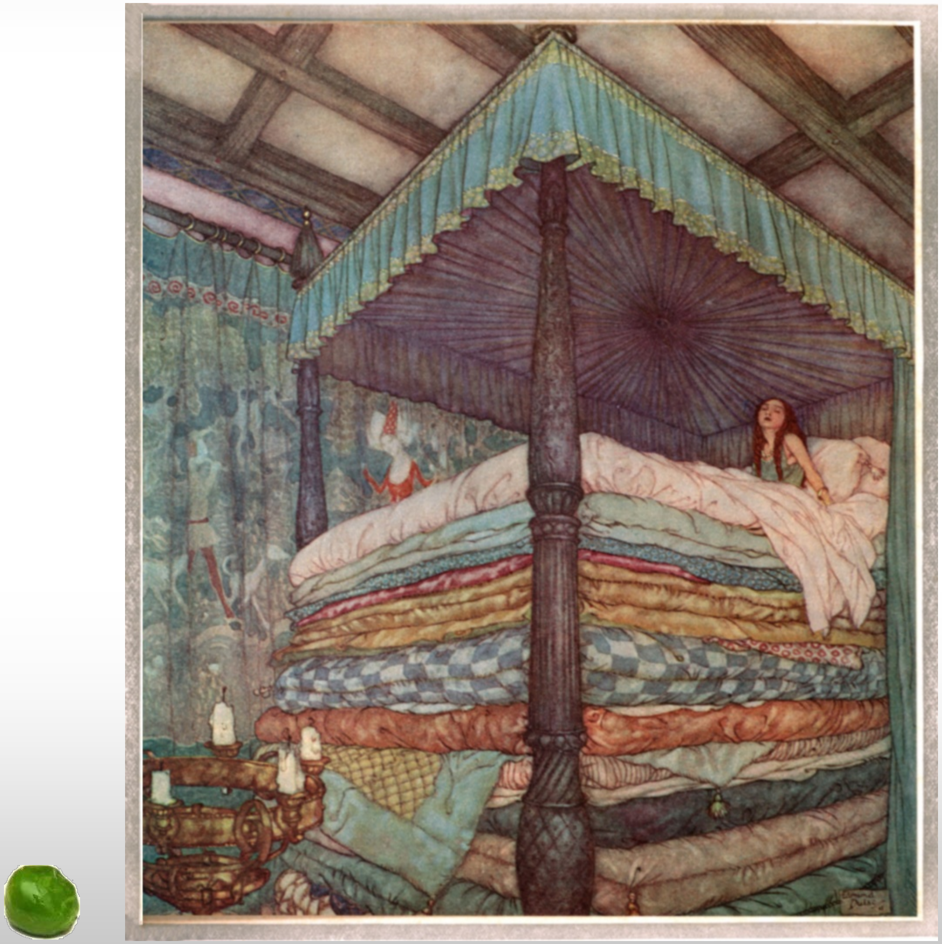


Virtualization

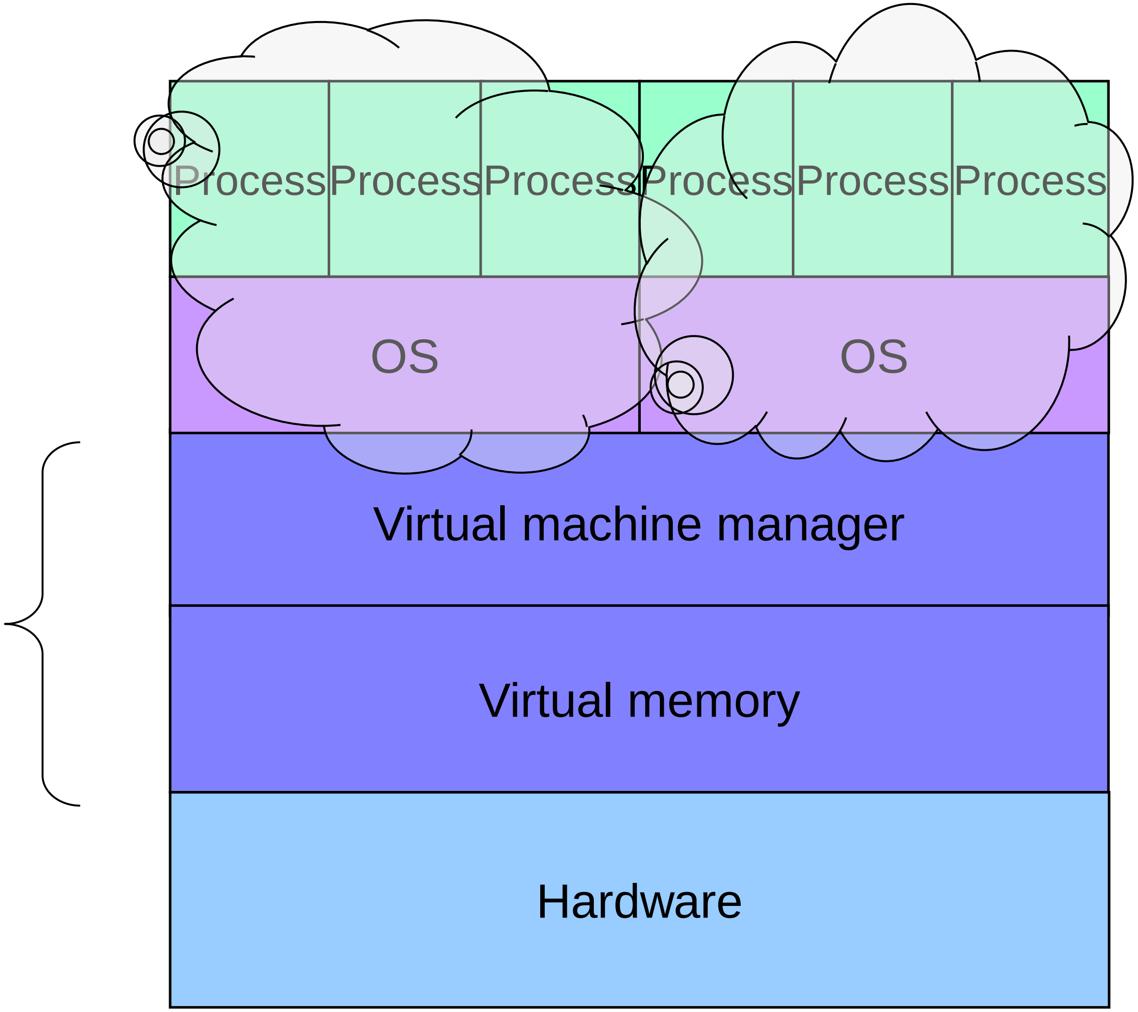
What if someone running on that hardware is malicious?



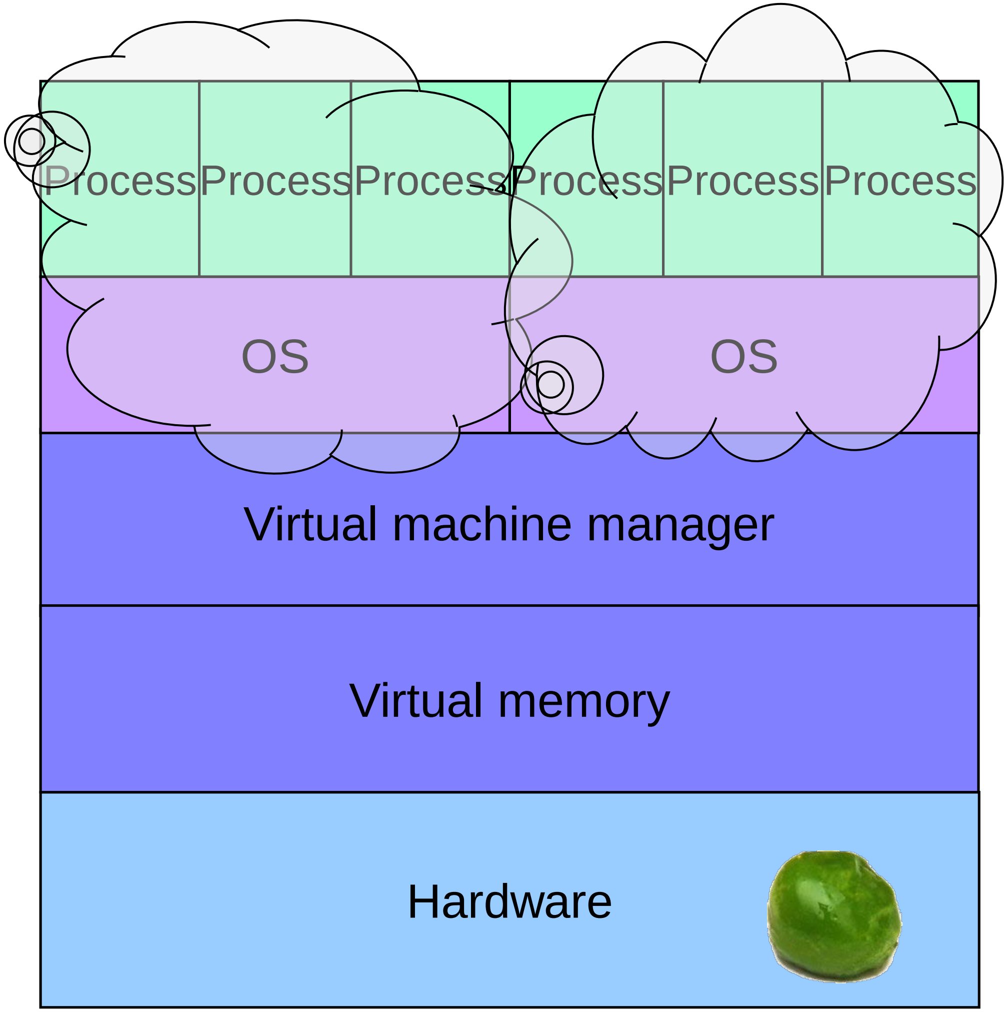
A Tale of Virtualization and Side Channels



Virtualization: textbook description

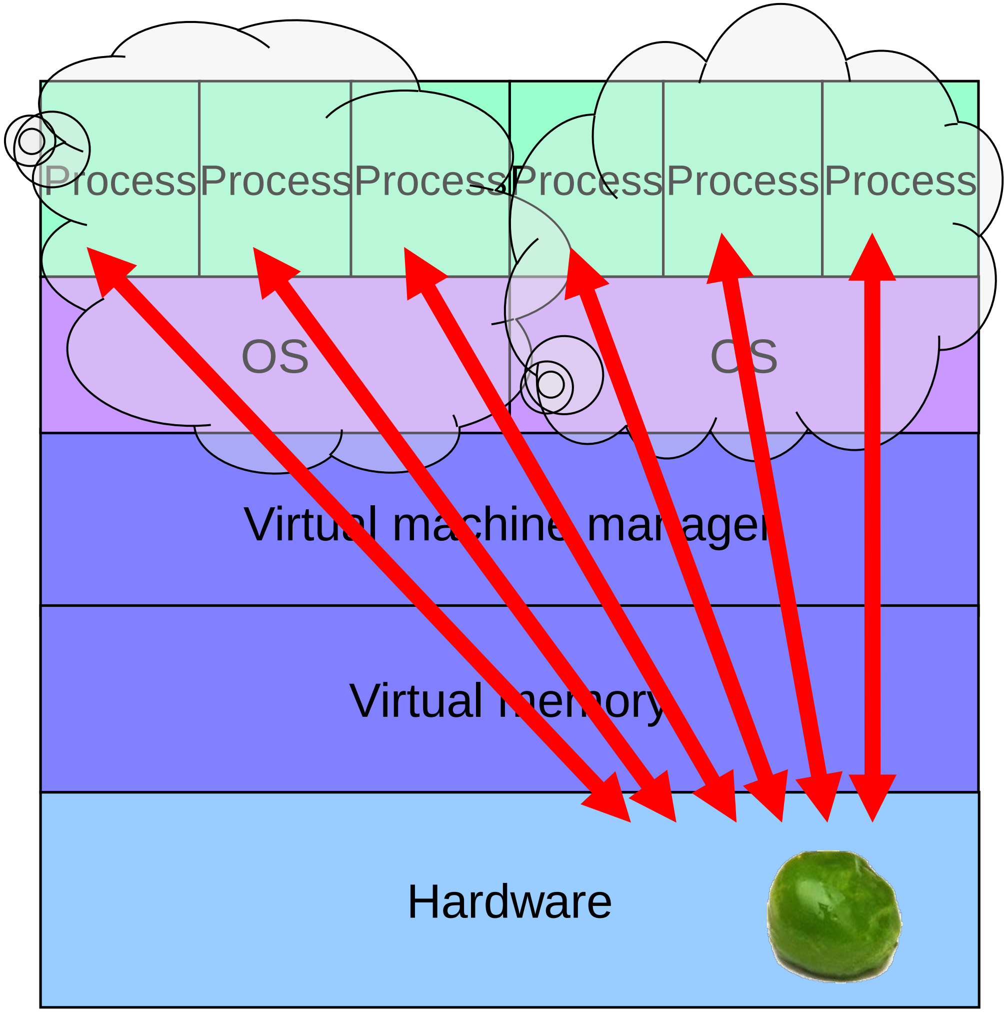


Cross-talk through architectural channels



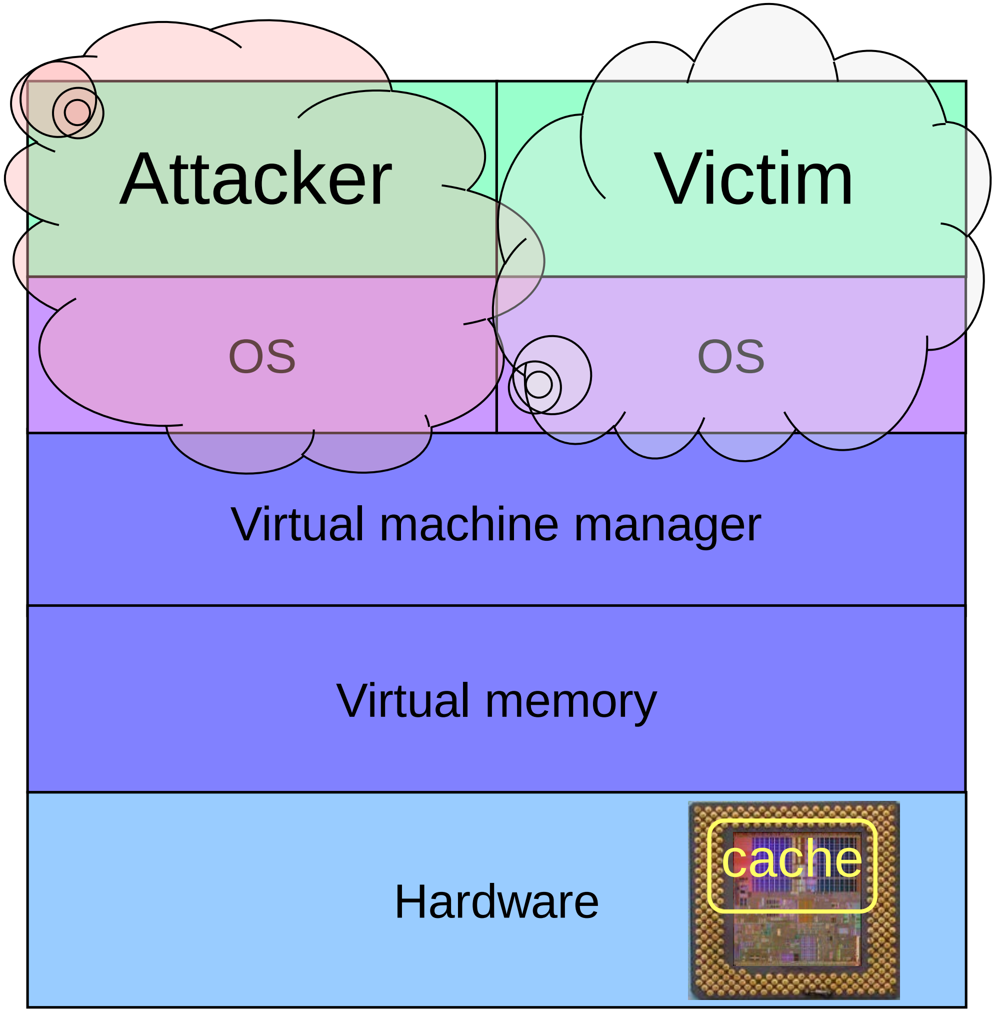
Cross-talk through architectural channels

- Contention for shared hardware resources



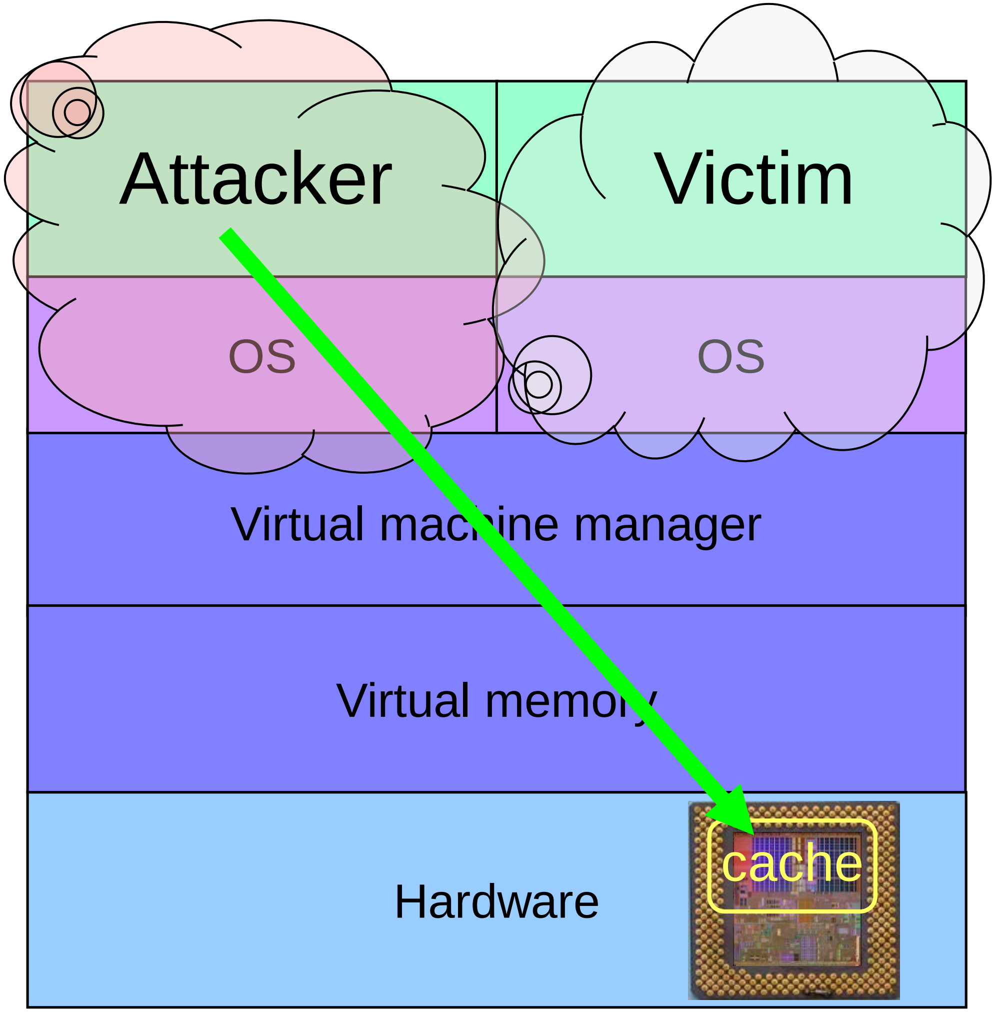
Cross-talk through architectural channels

- Contention for shared hardware resources
- Example: contention for **CPU data cache**



Cross-talk through architectural channels

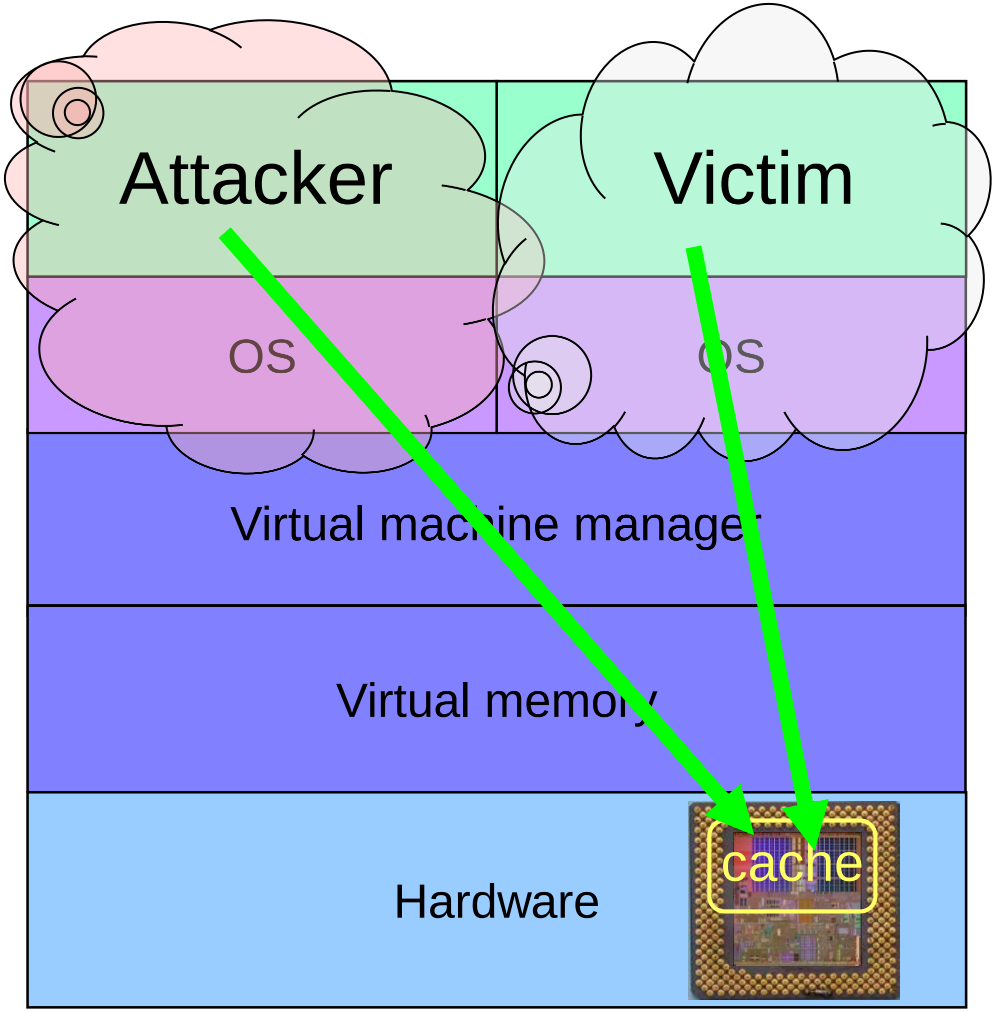
- Contention for shared hardware resources
- Example: contention for **CPU data cache**



<1 ns latency

Cross-talk through architectural channels

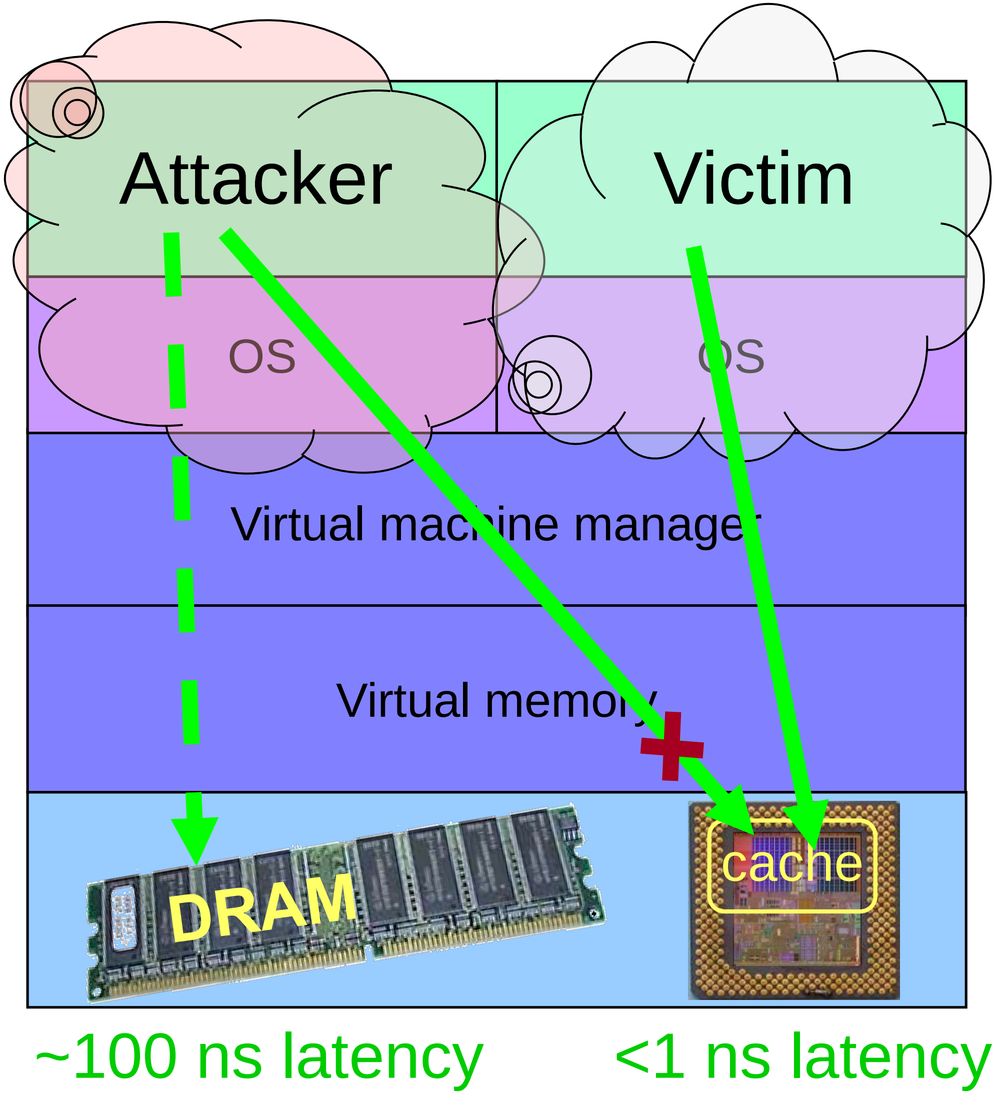
- Contention for shared hardware resources
- Example: contention for **CPU data cache**



<1 ns latency

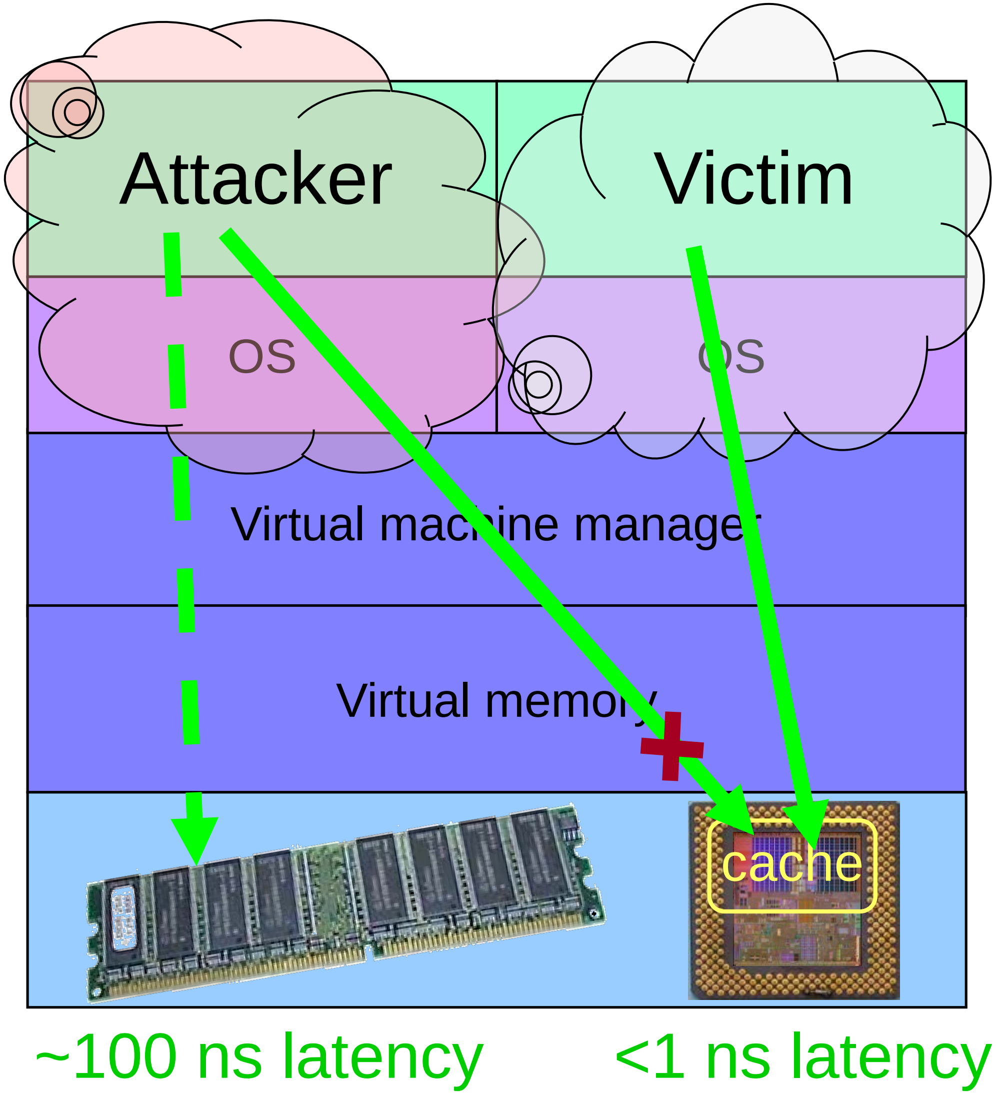
Cross-talk through architectural channels

- Contention for shared hardware resources
- Example: contention for **CPU data cache**



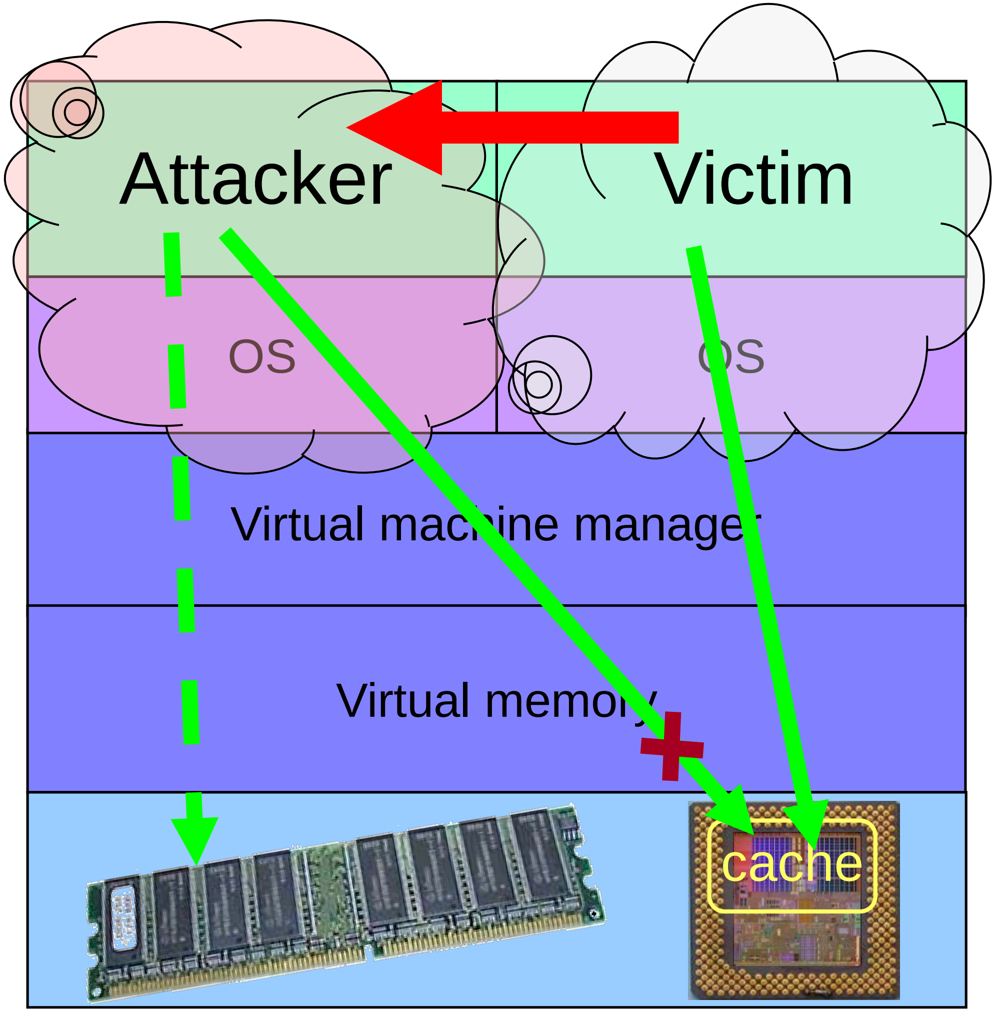
Cross-talk through architectural channels

- Contention for shared hardware resources
- Example: contention for CPU data cache leaks memory access patterns.



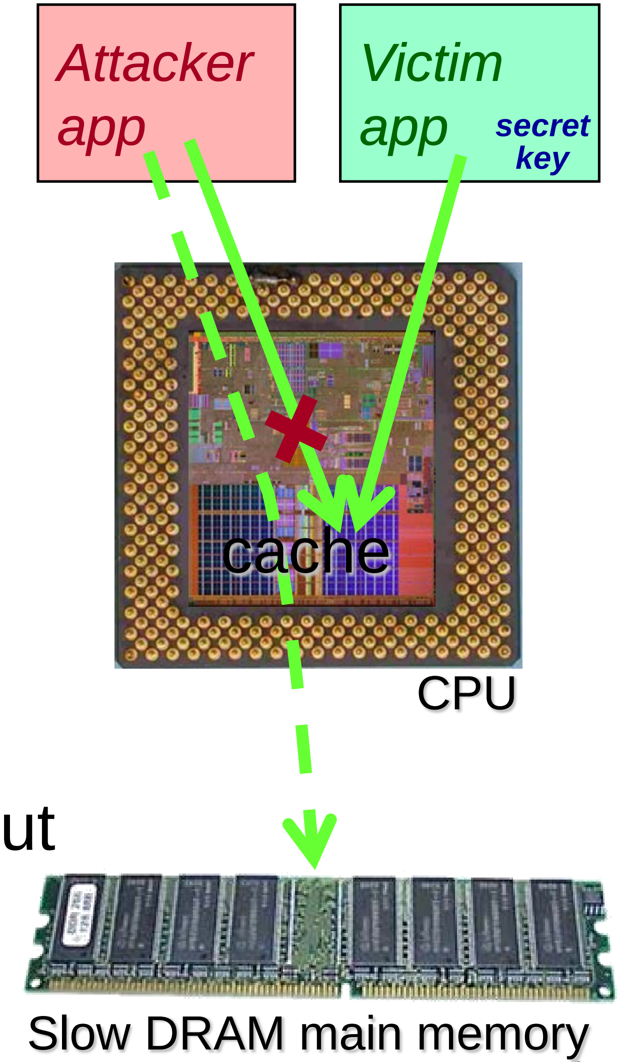
Cross-talk through architectural channels

- Contention for shared hardware resources
- Example: contention for CPU data cache leaks memory access patterns.
- This is sensitive information! Can be used to steal encryption keys in few milliseconds of measurements.



Cache attacks

- CPU core contains small, fast memory **cache** shared by all applications.
- Contention for this shared resources mean *Attacker* can observe slow-down when *Victim* accesses its own memory.
- From this, *Attacker* can deduce the memory access patterns of *Victim*.
- The cached data is subject to memory protection…
- But the metadata leaks information about memory access patterns: addresses and timing.



Example: breaking AES encryption via address leakage

(NIST FIPS 197; used by WPA2, IPsec, SSH, SSL, disk encryption, …)

```
char p[16], k[16];                          // plaintext and key
int32 Col[4];                               // intermediate state
const int32 T0[256],T1[256],T2[256],T3[256]; // lookup tables
...

/* Round 1 */
Col[0]← T0[p[ 0]⊕k[ 0]] ⊕ T1[p[ 5]⊕k[ 5]] ⊕
        T2[p[10]⊕k[10]] ⊕ T3[p[15]⊕k[15]];
Col[1]← T0[p[ 4]⊕k[ 4]] ⊕ T1[p[ 9]⊕k[ 9]] ⊕
        T2[p[14]⊕k[14]] ⊕ T3[p[ 3]⊕k[ 3]];
Col[2]← T0[p[ 8]⊕k[ 8]] ⊕ T1[p[13]⊕k[13]] ⊕
        T2[p[ 2]⊕k[ 2]] ⊕ T3[p[ 7]⊕k[ 7]];
Col[3]← T0[p[12]⊕k[12]] ⊕ T1[p[ 1]⊕k[ 1]] ⊕
        T2[p[ 6]⊕k[ 6]] ⊕ T3[p[11]⊕k[11]];
```

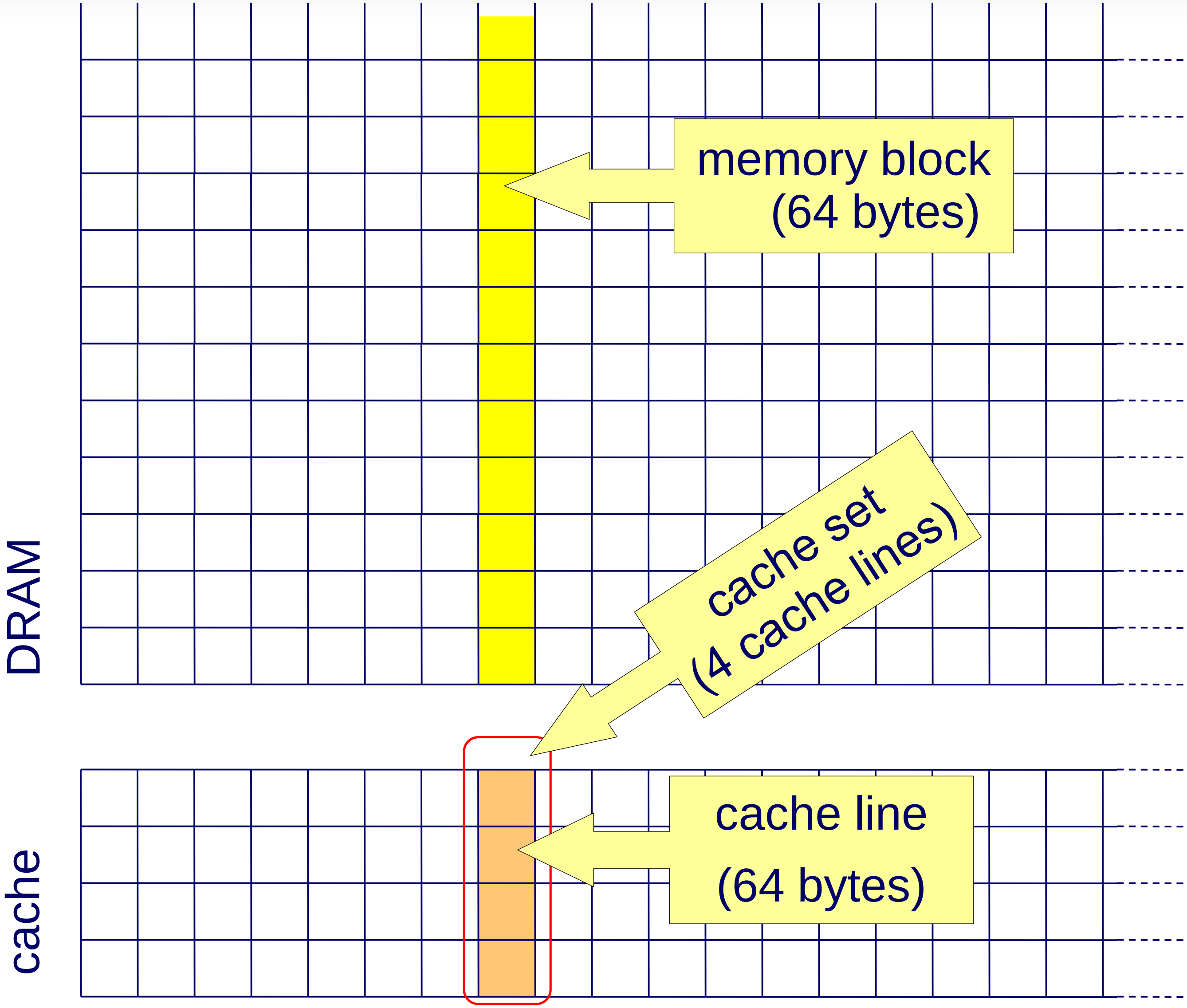
Complications:
- Multiple indices per cache line
- Uncertain messages
- Noise

Requires further cryptographic and statistical analysis.

How to learn addresses?

lookup index = plaintext ⊕ key

Associative memory cache



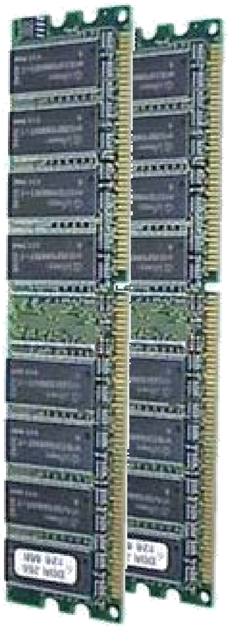
DRAM

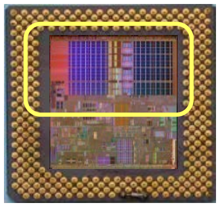
memory block
(64 bytes)

cache set
(4 cache lines)

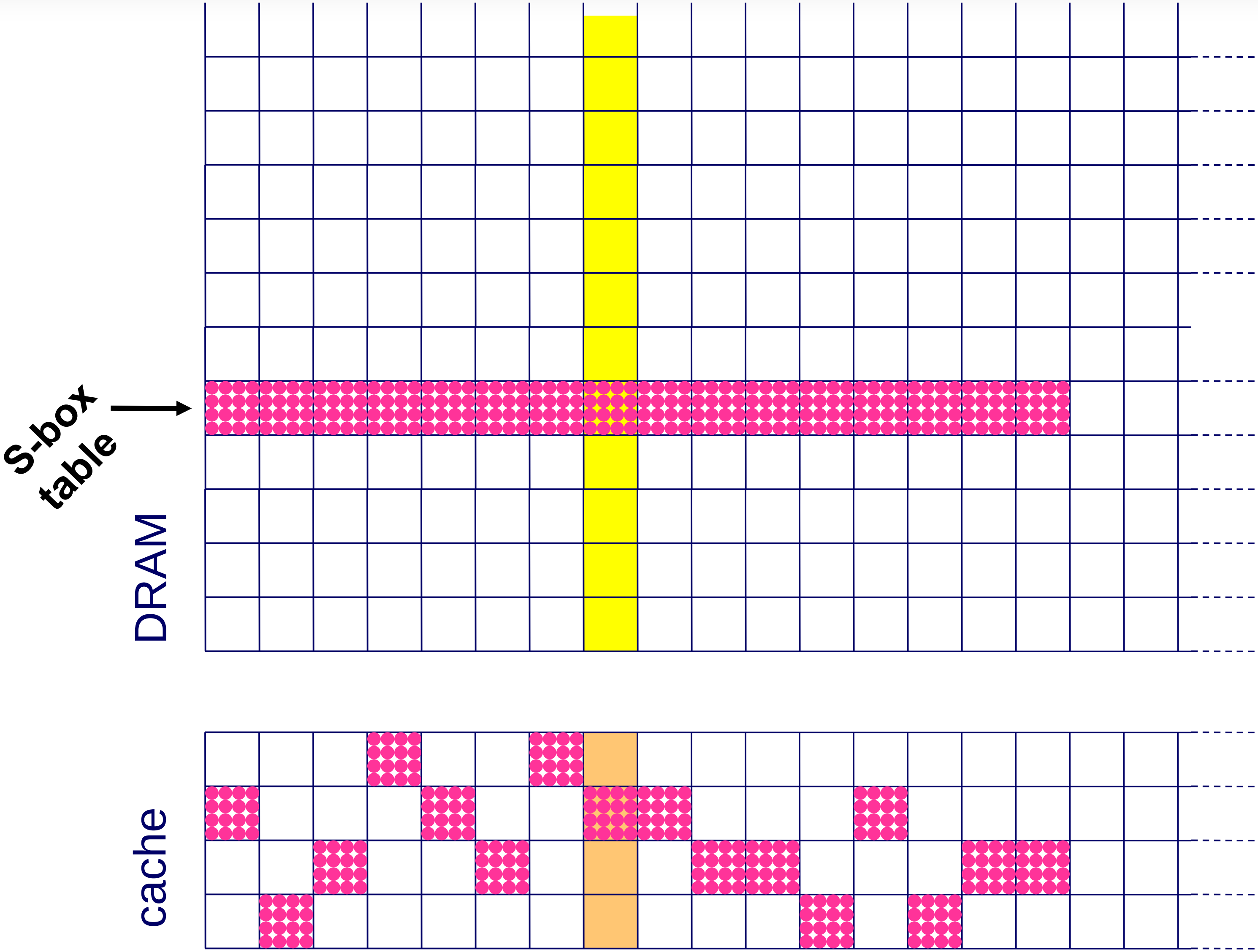
cache

cache line
(64 bytes)

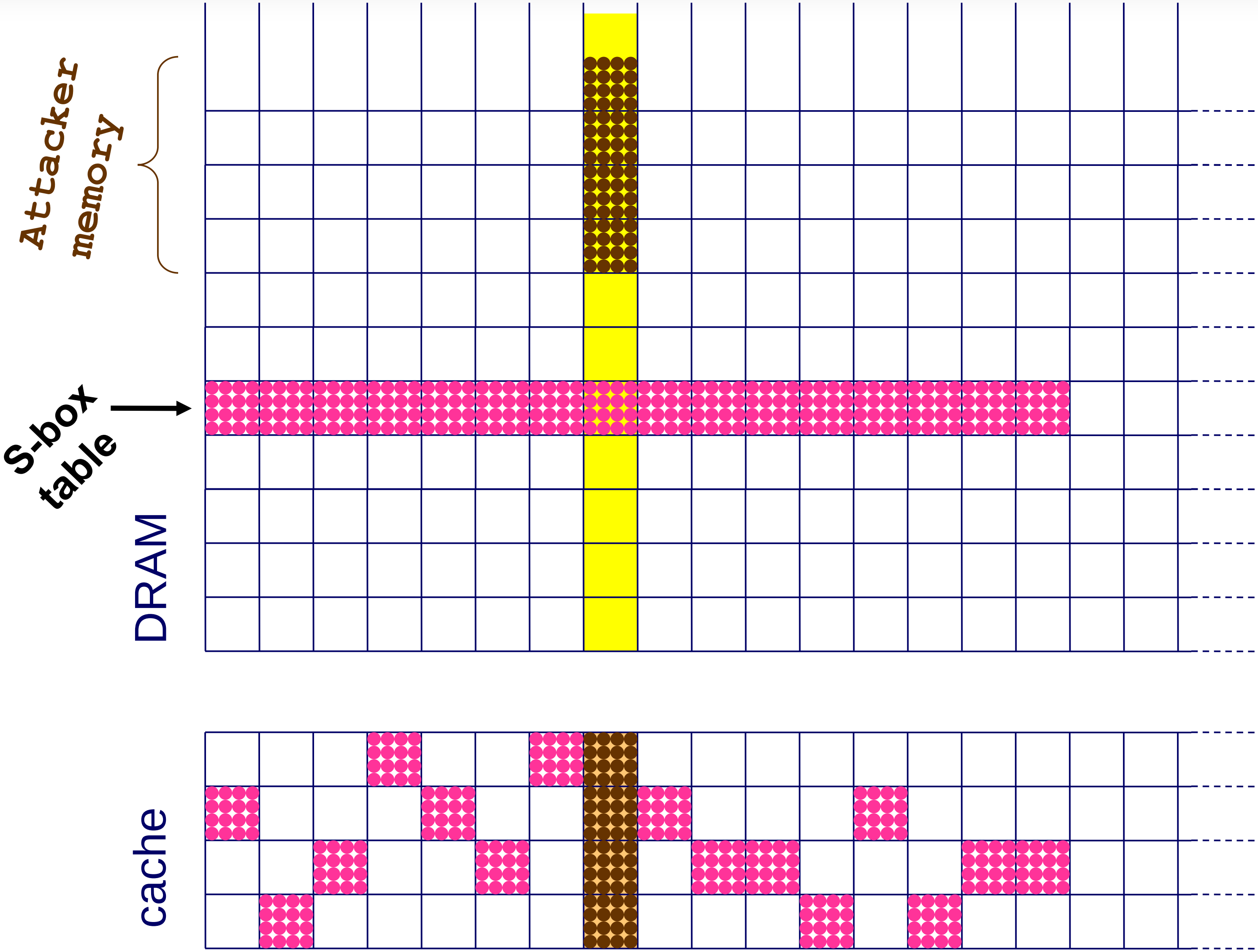




Victim's memory



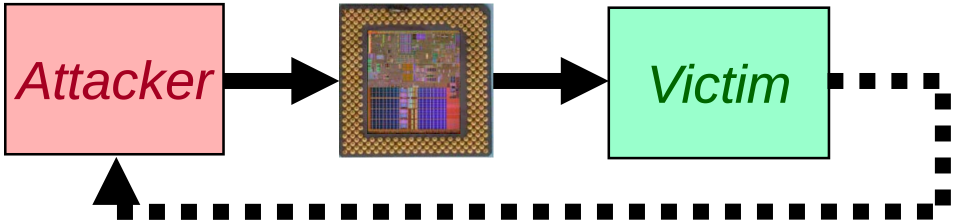
Detecting victim's memory accesses



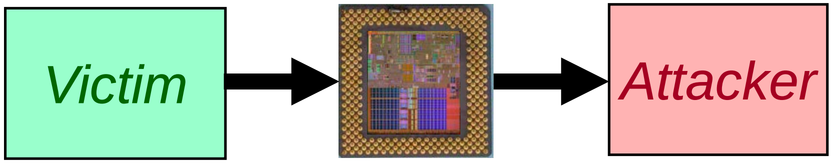
Measurement technique

Attacker can exploit cache-induced crosstalk as an input or as an output:

- Effect of the cache on the victim

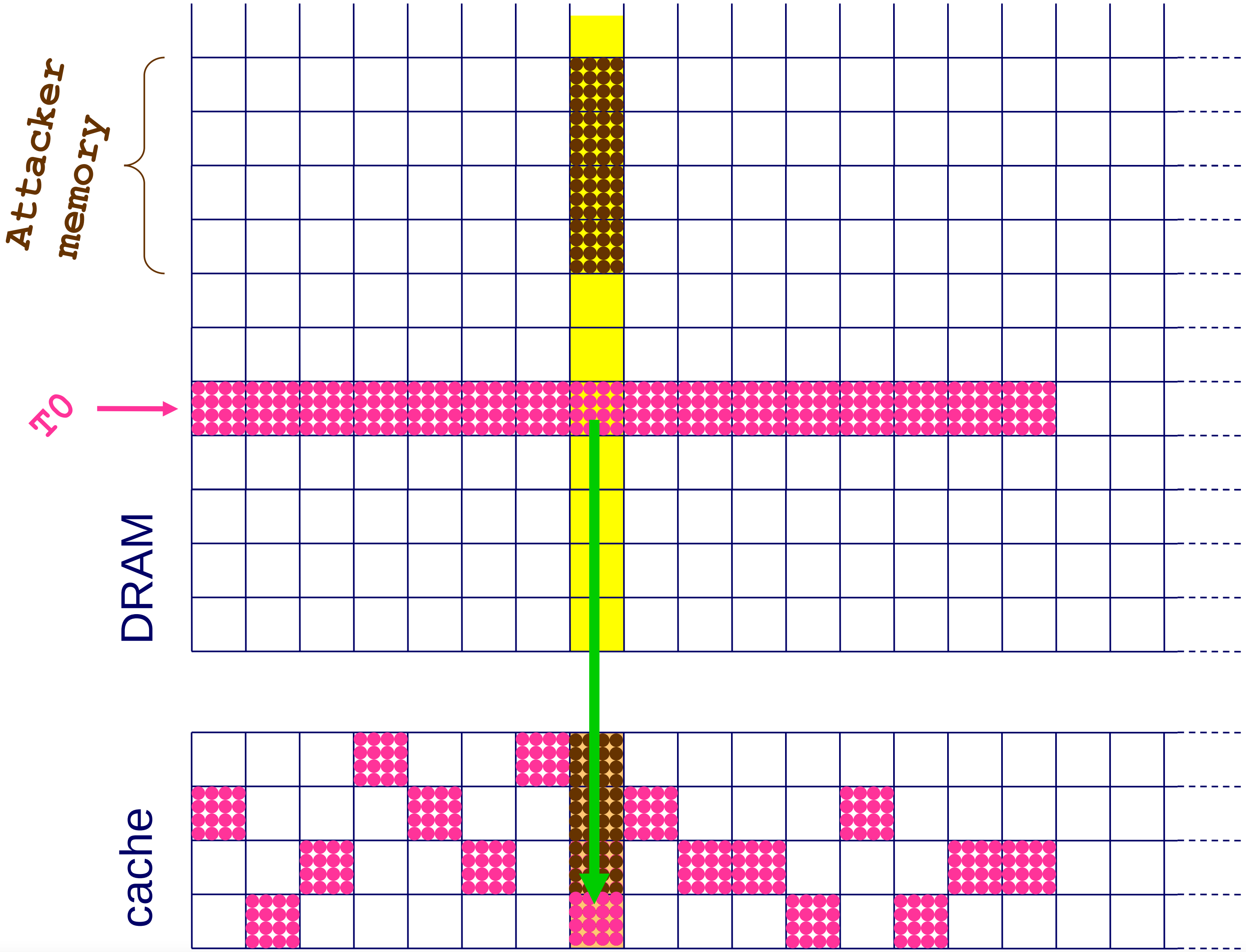


- Effect of victim on the cache



Evict+Time: Measuring effect of cache on encryption

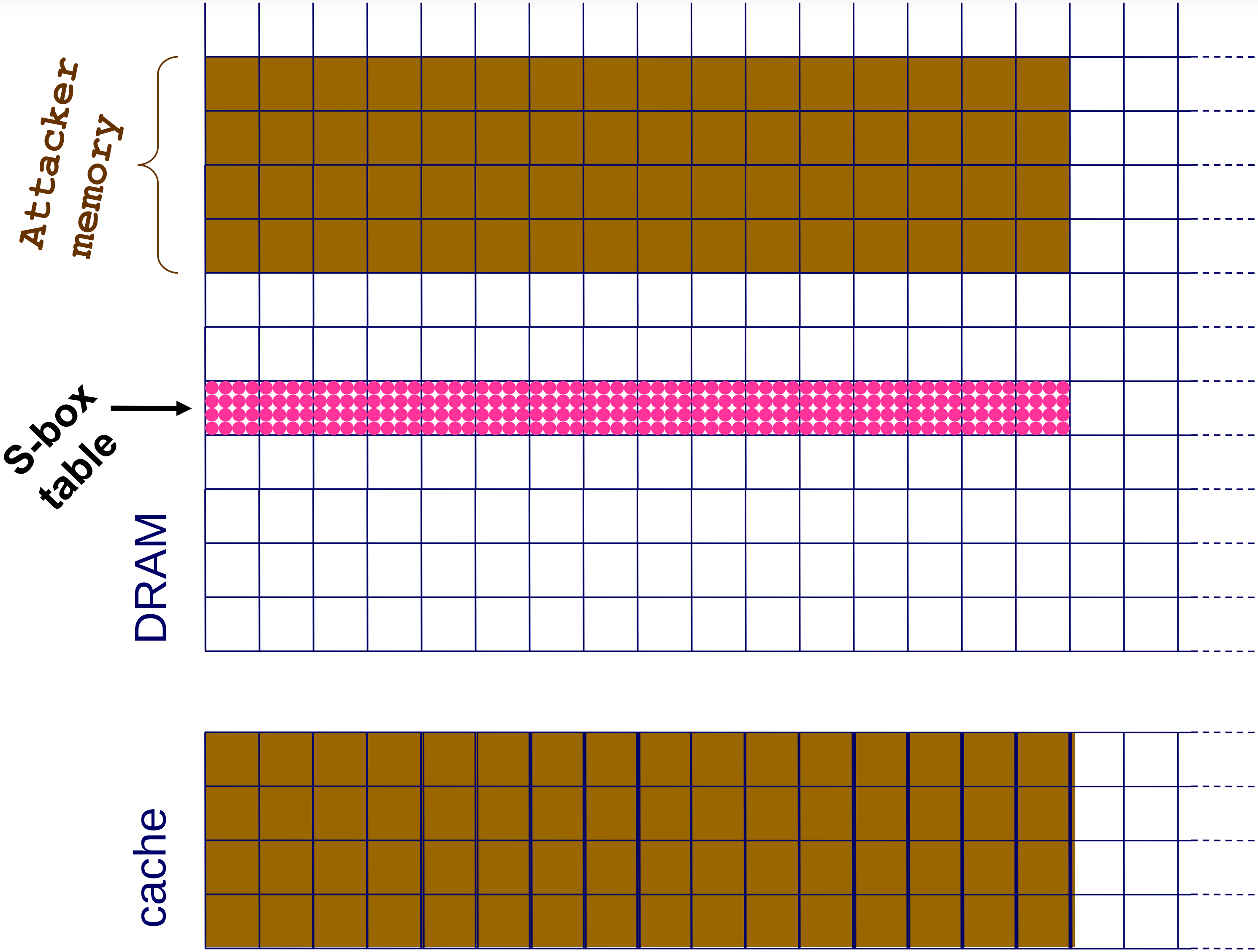
Attacker manipulates cache states and measures effect on victim's running time.



1. Victim's data fully cached
2. Attacker evicts victim's block
3. Attacker times the victim's next run. Slowdown?

Prime+Probe: Measuring effect of encryption on cache

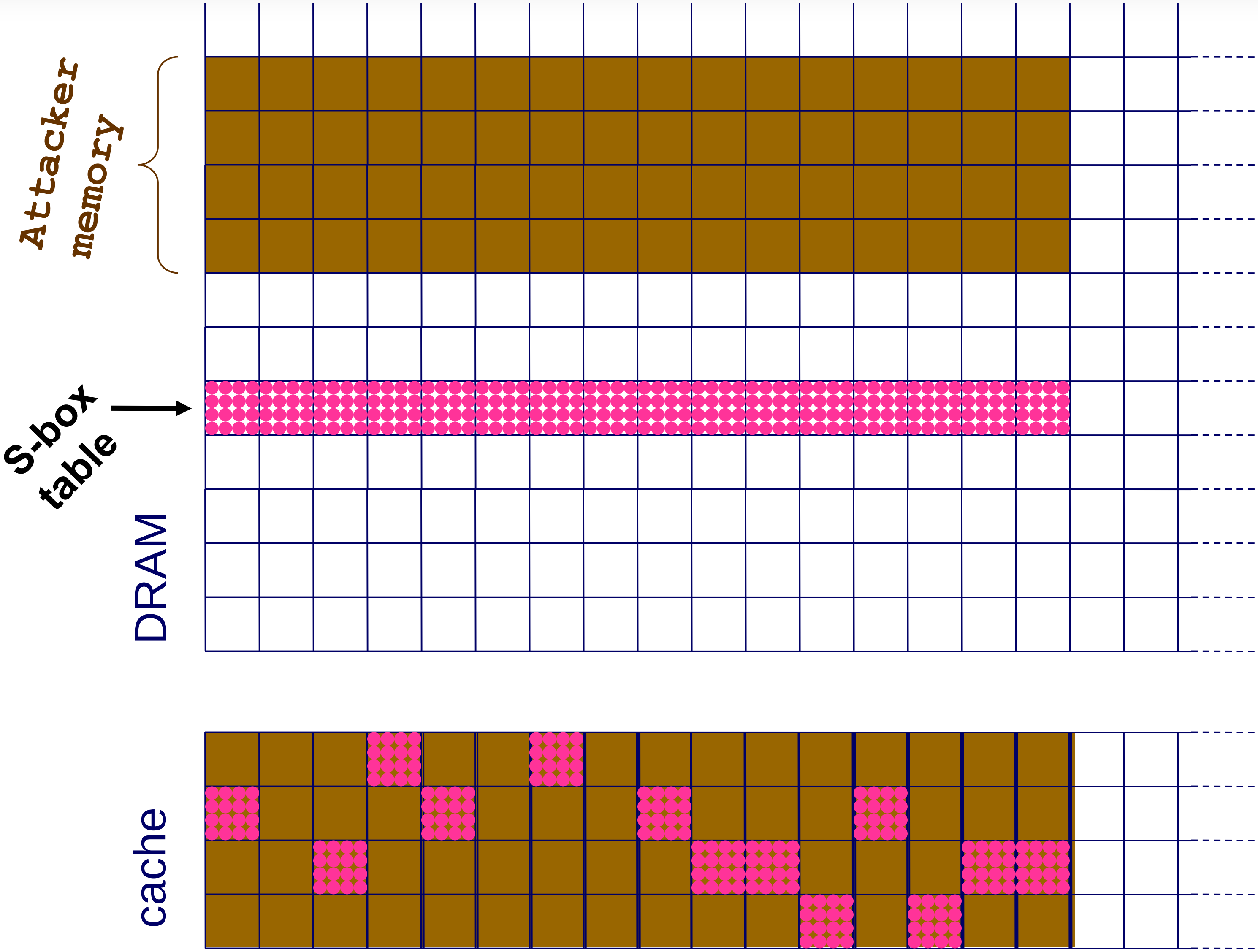
Attacker checks which of its own data was evicted by the victim.



1. Fill cache with attacker's data

Prime+Probe: Measuring effect of encryption on cache

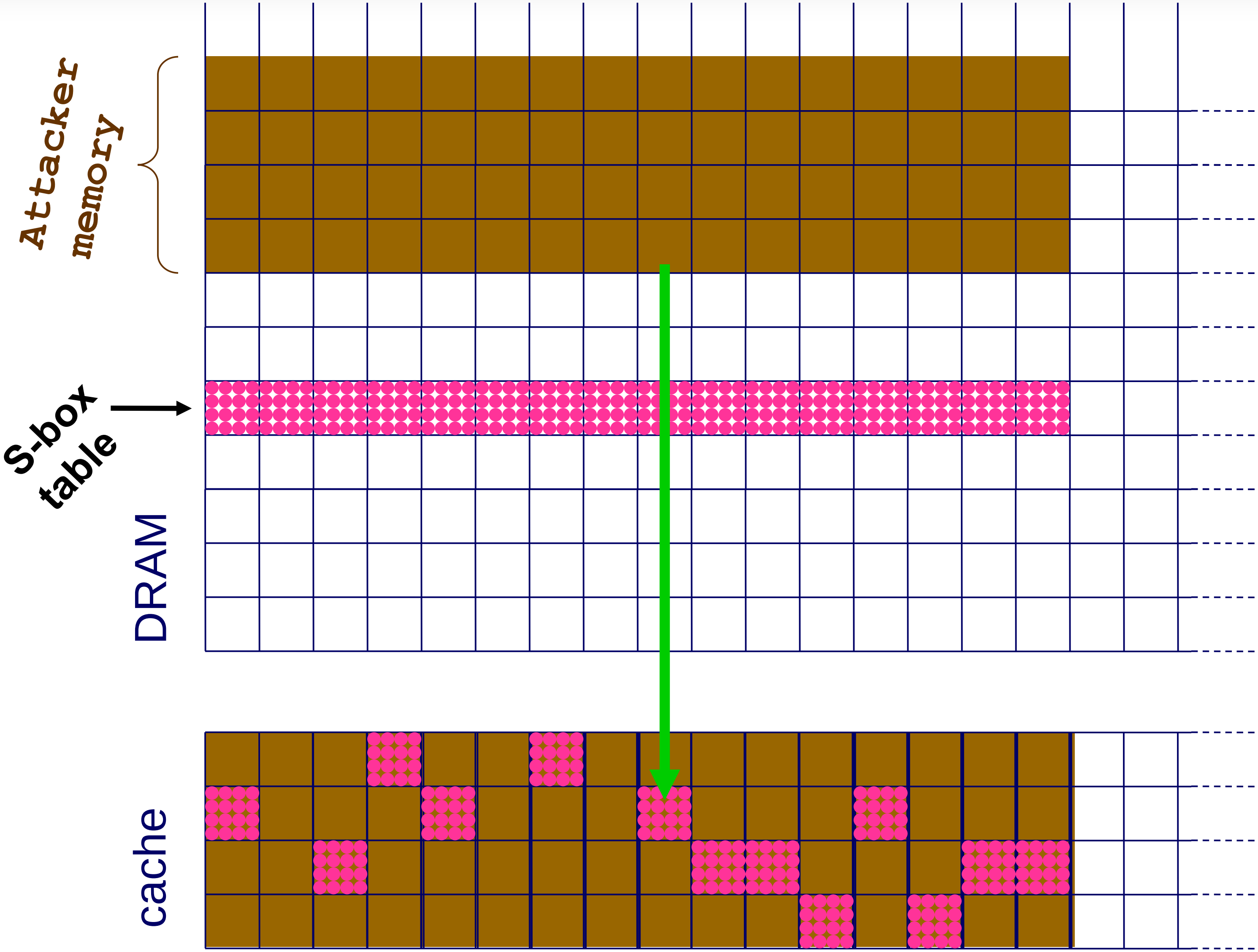
Attacker checks which of its own data was evicted by the victim.



1. Fill cache with attacker's data
2. Trigger a single encryption

Prime+Probe: Measuring effect of encryption on cache

Attacker checks which of its own data was evicted by the victim.

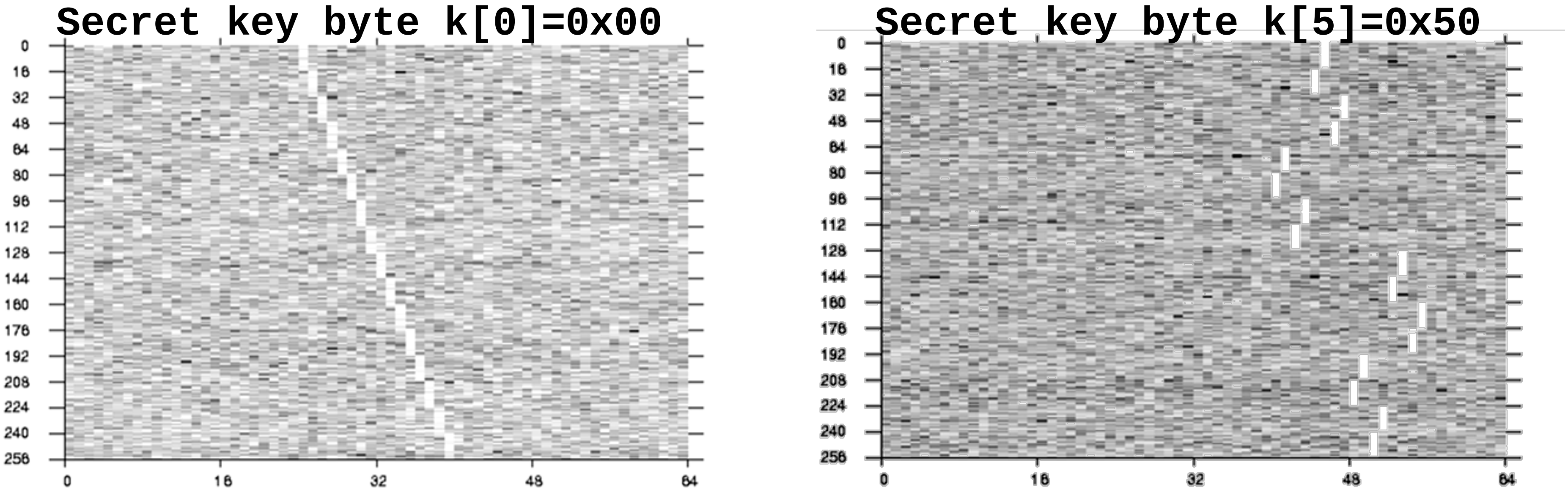


1. Fill cache with attacker's data
2. Trigger a single encryption
3. Access attacker memory again and see which cache sets are slow

Experimental results

[Osvik Shamir Tromer 05]
[Tromer Osvik Shamir 09]

- Attack on OpenSLL AES encryption library call:
  Full key extracted from 13ms of measurements (300 encryptions)
- Attack on an AES encrypted filesystem (Linux **`dm-crypt`**):
  Full key extracted from 65ms of measurements (800 I/O ops)



Measuring a “black box” OpenSSL encryption on Athlon 64, using 10,000 samples. Horizontal axis: evicted cache set. Vertical axis: `p[0]` (left), `p[5]` (right). Brightness: encryption time (normalized)

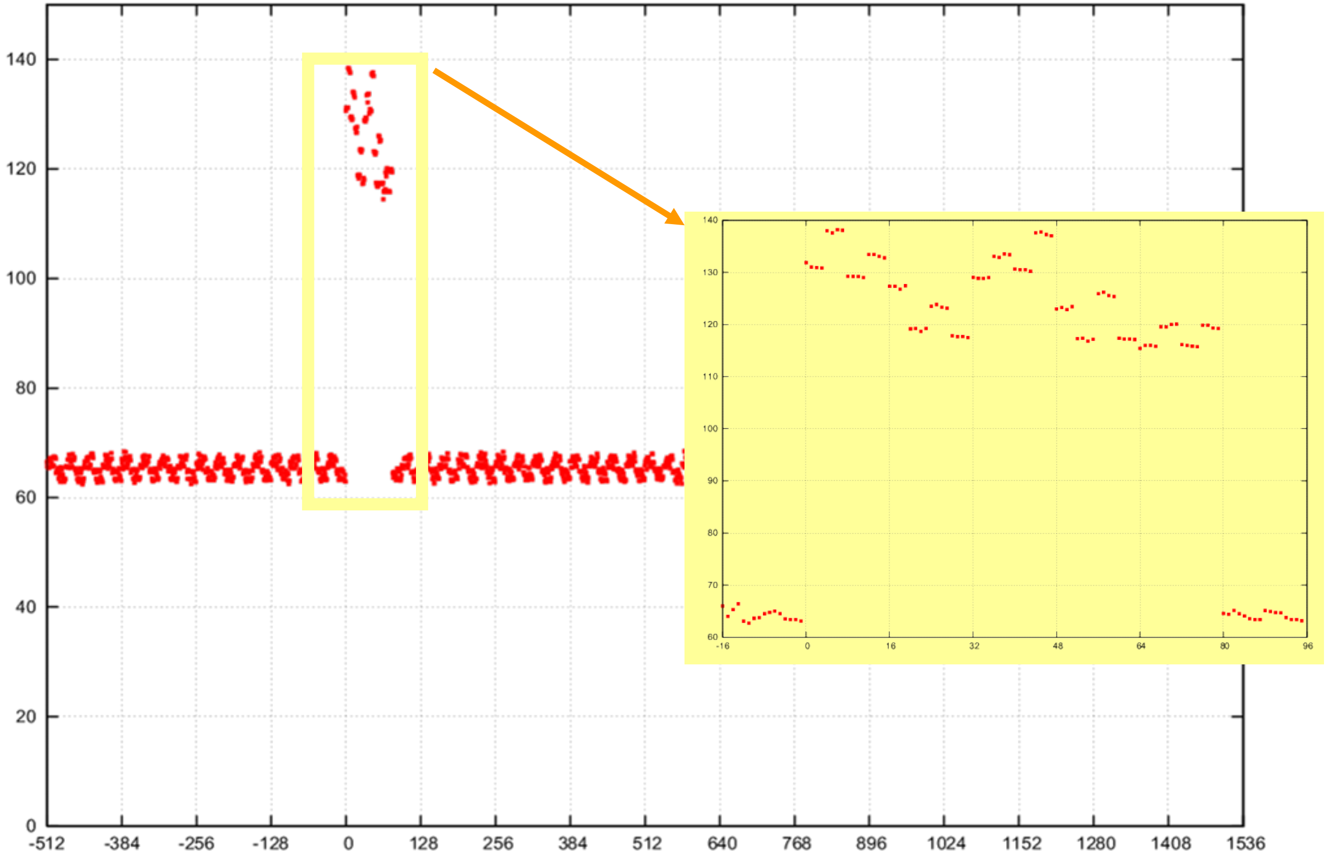
Extension: “Hyper Attacks”

- Obtaining parallelism:
  - **Hyper**Threading (simultaneous multithreading)
  - Multi-core, shared caches, cache coherence
  - (Also: interrupts, scheduler)
- Attack vector:
  - Monitor cache statistics in real time
  - Encryption process is not communicating with anyone
    (no I/O, no IPC).
  - No special measurement equipment
  - No knowledge of either plaintext of ciphertext

Experimental results

- “Hyper Attack” attack on AES
  (independent process doing batch encryption of text):

  Recovery of 45.7 key bits in one minute.



Implications?

Implications

- Multiuser systems (e.g, Android)
- Untrusted code, even if sandboxed
  (e.g., ActiveX, Java applets, managed .NET, JavaScript, Google Native Client, Silverlight)
- Digital right management
  The trusted path is leaky
  (even if verified by TPM attestation, etc.)
- Remote network attacks
  Virtual machines

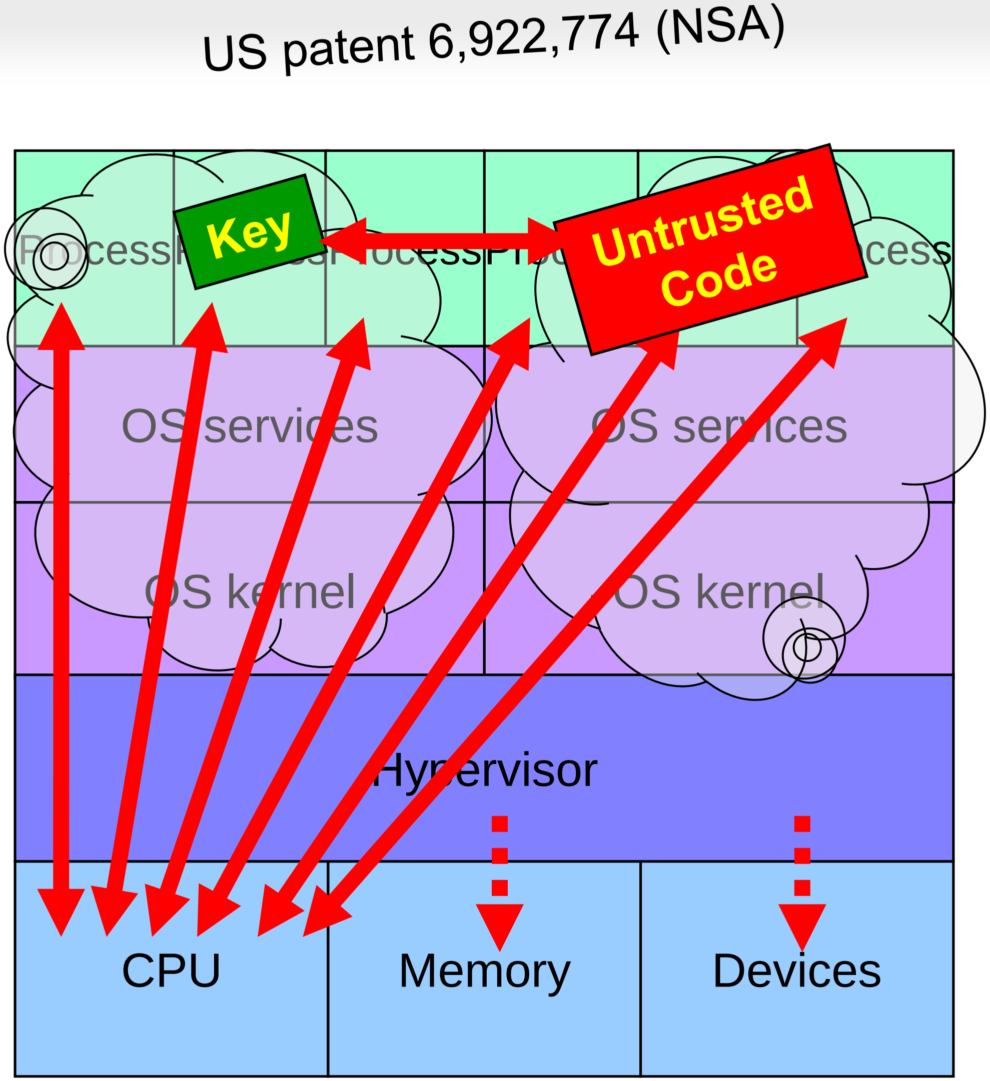
Virtualization

Touted for its security benefits:

- Isolation
- Sandboxing
- Management
- Monitoring
- Recovery
- Forensics (replay)

All true.

But many side-channel attacks are oblivious to virtualization. (It’s the same underlying hardware!)



This creates inherent new risks.

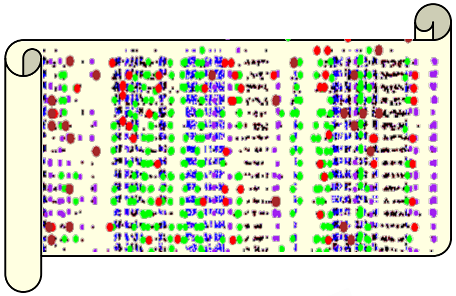
Architectural attacks in cloud computing: difficulties

- How can the attacker reach a target VM?
- How to exploit it? Practical difficulties:
  - Core migration
  - Extra layer of page-table indirection
  - Coarse hypervisor scheduler
  - Load fluctuations
  - Choice of CPU

- Is the “cloud” really vulnerable?

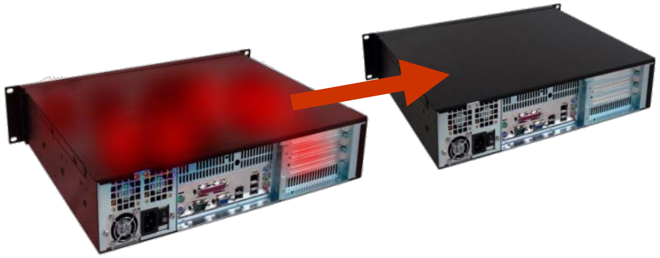
Demonstrated using Amazon EC2 as cast study:

- Cloud cartography
  Mapping the structure of the “cloud” and locating a target on the map.



- Placement vulnerabilities
  An attacker can place his VM on the same physical machine as a target VM (40% success for a few dollars).

- Cross-VM exfiltration
  Once VMs are co-resident, information and secret keys can be exfiltrated across VM boundary.



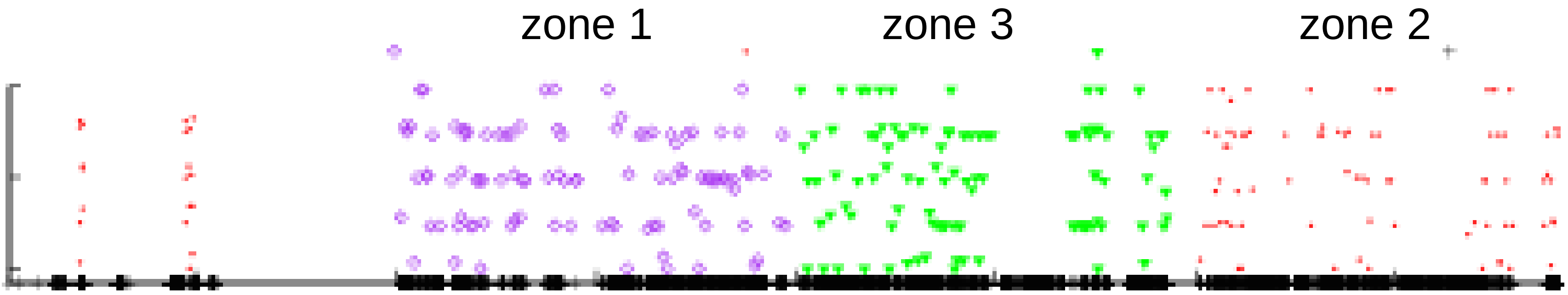
Cloud cartography

Where in the world is the target VM, and how can I get there?

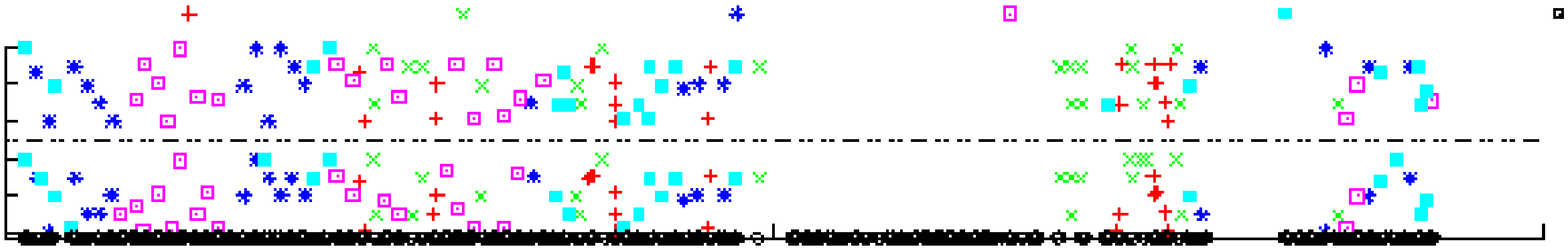
- On EC2, VMs can be co-resident only if they have identical creation parameters:
  - Region (US/Europe)
  - Availability zone (data center)
  - Instance type (machine pool)
- The cloud-internal IP addresses assigned to VMs are strongly correlated with their creation parameters.

Mapping out this correlation:

Cloud cartography (example)



IP address (position) vs. zone (color)



IP address (position) vs. instance type (color)

Deduced:
Heuristic rules for mapping IP address to creation parameters.

Achieving co-residence

- Overall strategy:
  - Derive target's creation parameters
  - Create similar VMs until co-residence is detected.
- Improvement:
  - Target fresh (recently-created) instances, exploiting EC2's sequential assignment strategy
  - Conveniently, one can often *trigger* new creation of new VMs by the victim, by inducing load (e.g., RightScale).

- Success in hitting a given (fresh) target: ~40% for a few dollars

  Reliable across EC2 zones, accounts and times of day.

Detecting co-residence

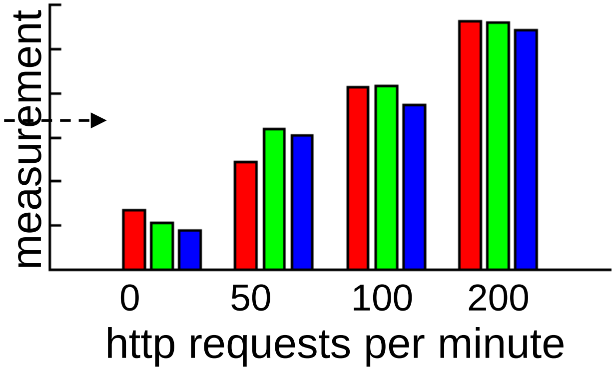
- EC2-specific:
  - Internal IP address are close
- Xen-specific:
  - Obtain and compare Xen Dom0 address
- Generic:
  - Network latency
  - Cross-VM architectural channels:
    send HTTP requests to target and observe correlation with cache utilization

Exploiting co-residence: cross-VM attacks

- Demonstrated:

[Ristenpart Tromer Shaham Savage ‘09]

[Zhang Juels Reiter Ristenpart ‘12]



  - Measuring VMs load (average/transient)
  - Estimating web server traffic
  - Robust cross-VM covert channel
  - Detecting keystroke timing in an SSH session across VMs

    (on a similarly-configured Xen box)

    → keystroke recovery [Song Wagner Tian 01]
  - Stealing ElGamal secret keys[Zhang Juels Reiter Ristenpart 2012]
  - Stealing RSA secret keys[Inci Gulmezoglu Irazoqui Eisenbarth Sunar 2015]
  - Stealing AES secret keys [Irazoqui Inci Eisenbarth Sunar 2014]

Architectural attacks (continued)

L3 Flush+Reload attack

[Yarom Falkner 2014]

- Target outermost cache, shared between all CPU cores (typically L3)
- RSA key extraction from GnuPG 1.4.13
- Target specific memory block (instead of cache set)
- Exploits memory deduplication (content-based page sharing)
  - Common code, libraries, data across VMs
  - Supposedly safe (nominally, no new information flow)

L3 flush+reload attack (cont.)

To measure a memory block b, the attacker:

- Achieve page sharing of b with victim
- Flush block b using x86 `clflush` instruction
  - Flushes block from all cache levels
  - Normally used for synchronization / performance
- Wait until victim runs
- Measure time to read the block b
  - Fast → victim accessed b
  - Slow → victim did not access b