TEL AVIV UNIVERSITY

Information Security – Theory vs. Reality

0368-4474-01, Winter 2011

Lecture 1: Introduction

Eran Tromer

Case I

Cryptographic algorithms
vs. the real world

Cryptographic algorithms

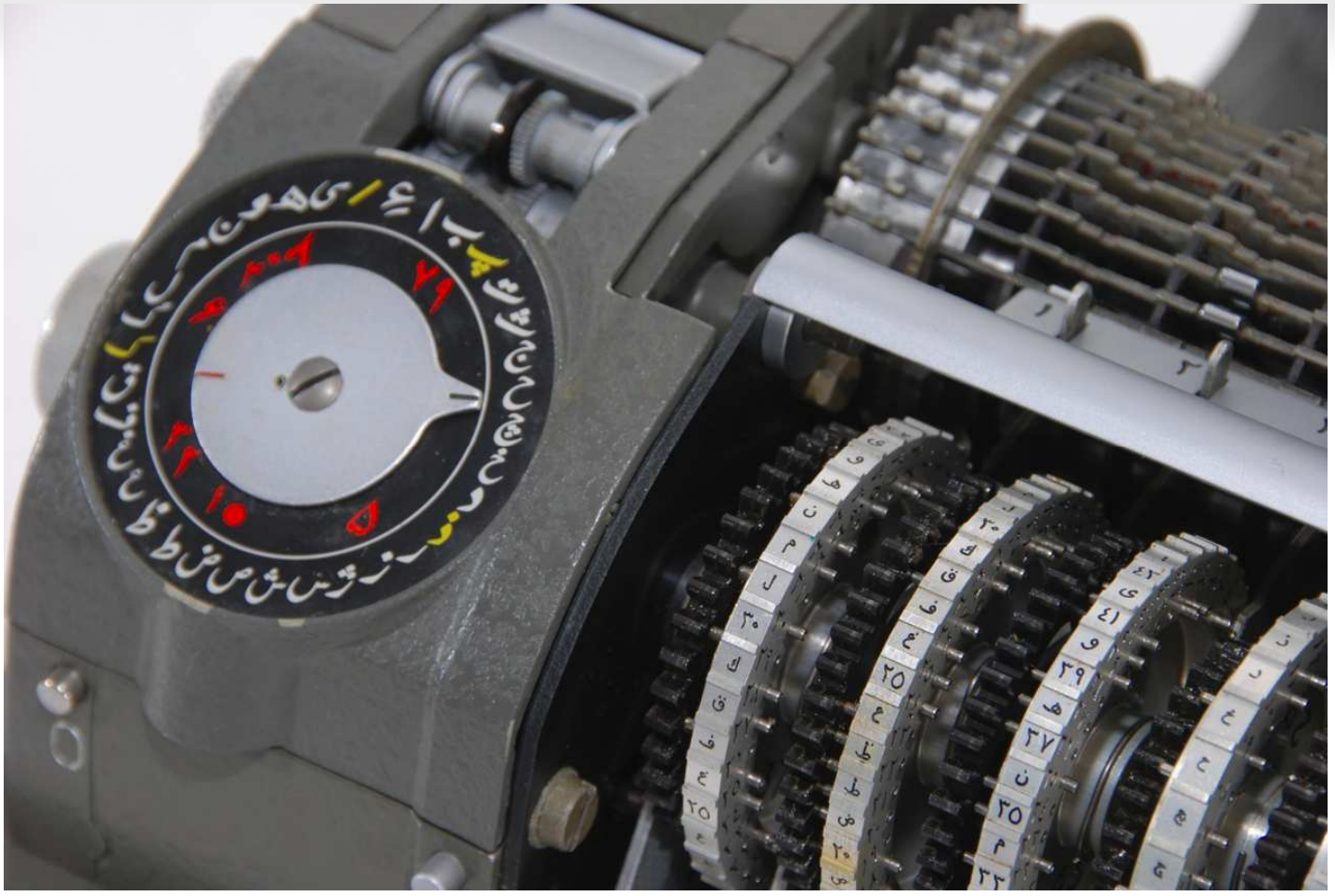
- Model: Input (plaintext and key) $\longrightarrow$  $\longrightarrow$ Output (ciphertext)
- Formal security definitions (CPA, CCA1, CCA2, …)
- Well-studied algorithms (RSA, AES, DES, …)
- Algorithmic attacks are believed infeasible.

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.)



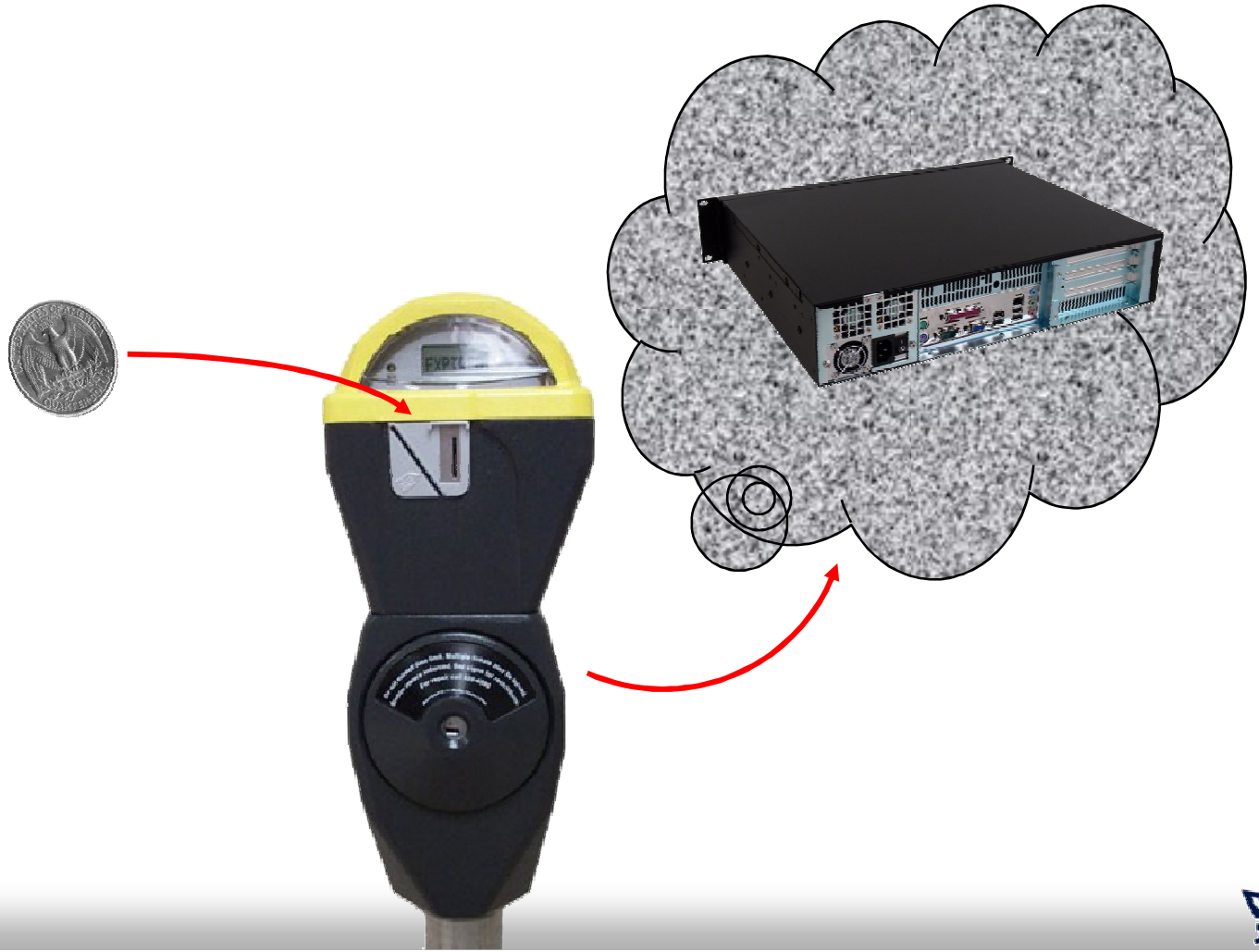
- *"The combined MI5/GCHQ operation enabled us to read the Egyptian ciphers in the London Embassy throughout the Suez Crisis."*

Case II

Architectural attacks

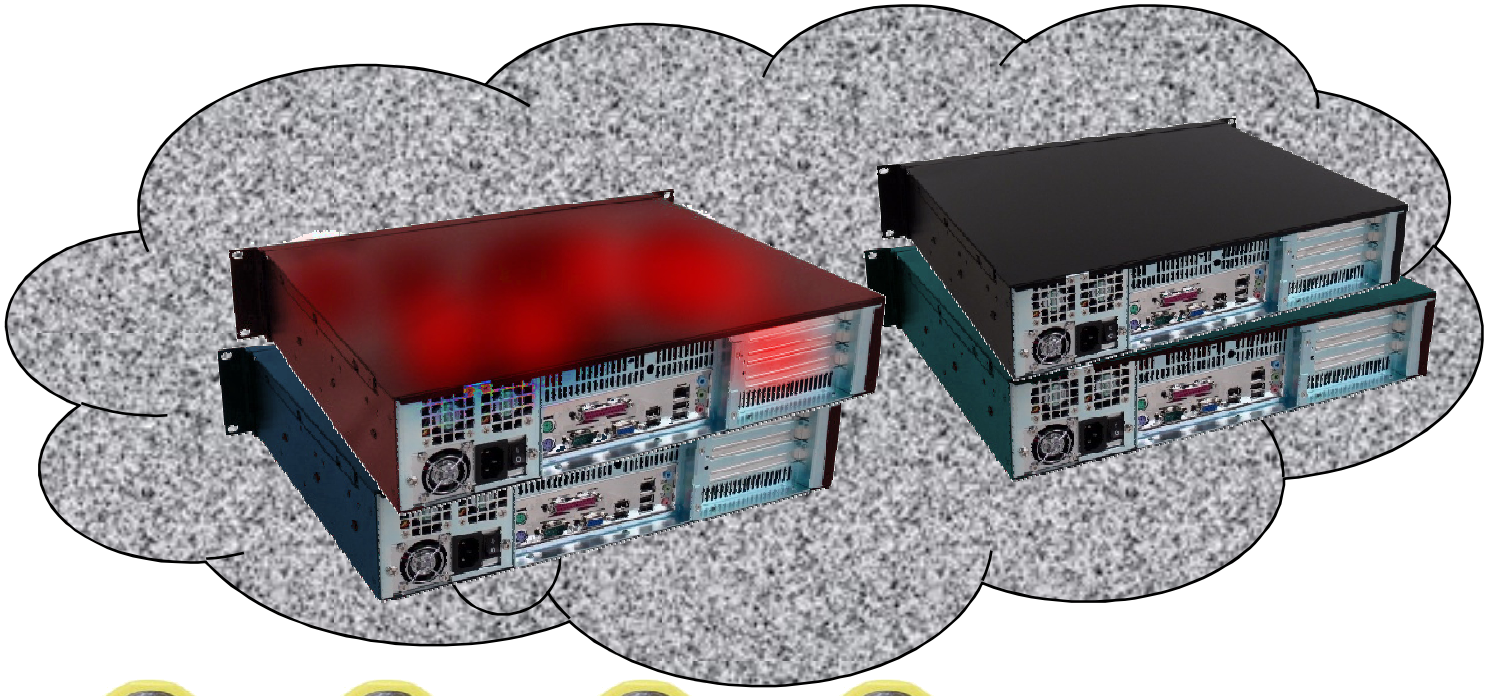
Cloud Computing (Infrastructure as a Service)

Instant virtual machines



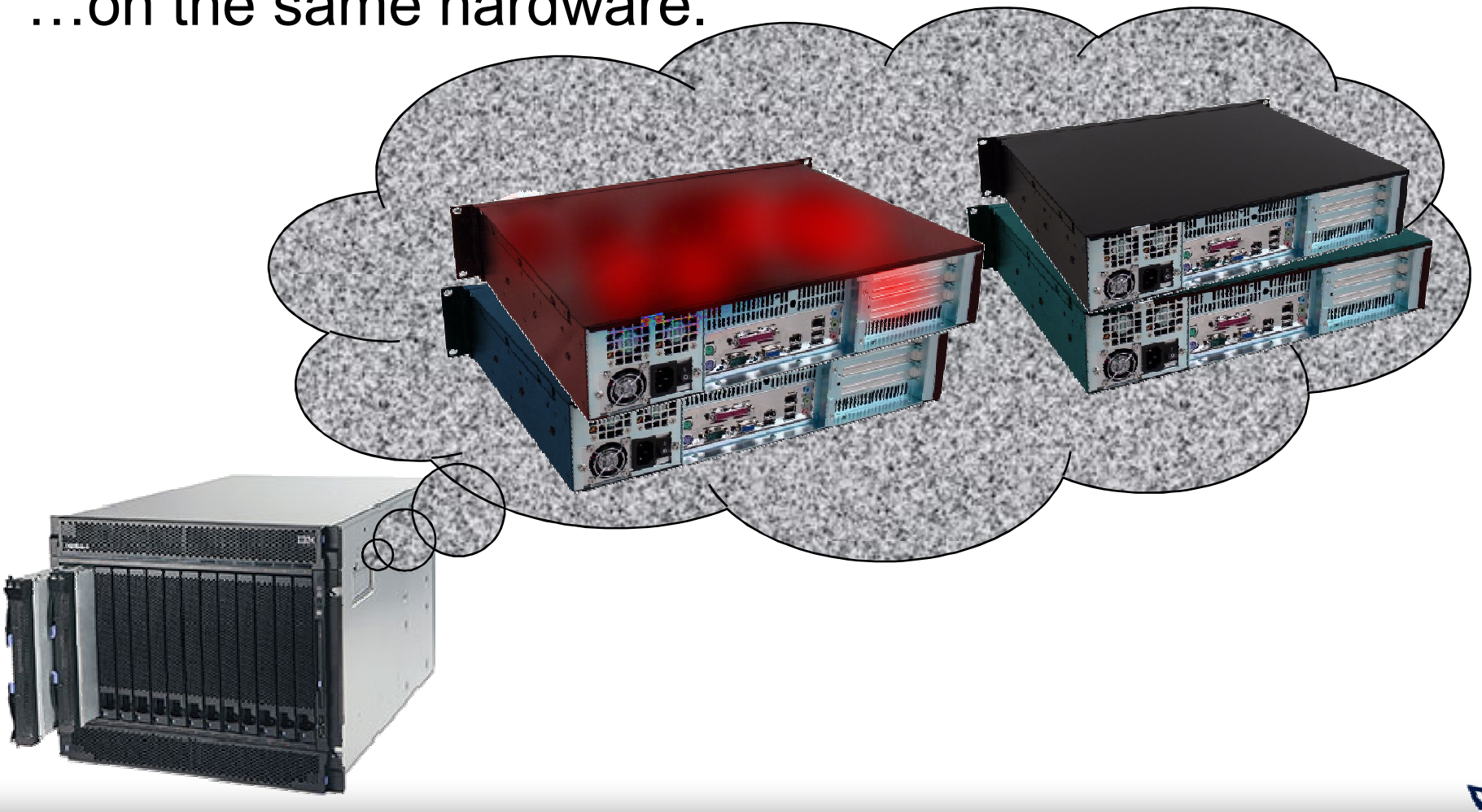
Public Clouds
(Amazon EC2, Microsoft Azure, Rackspace Mosso)

Instant virtual machines
... for anyone



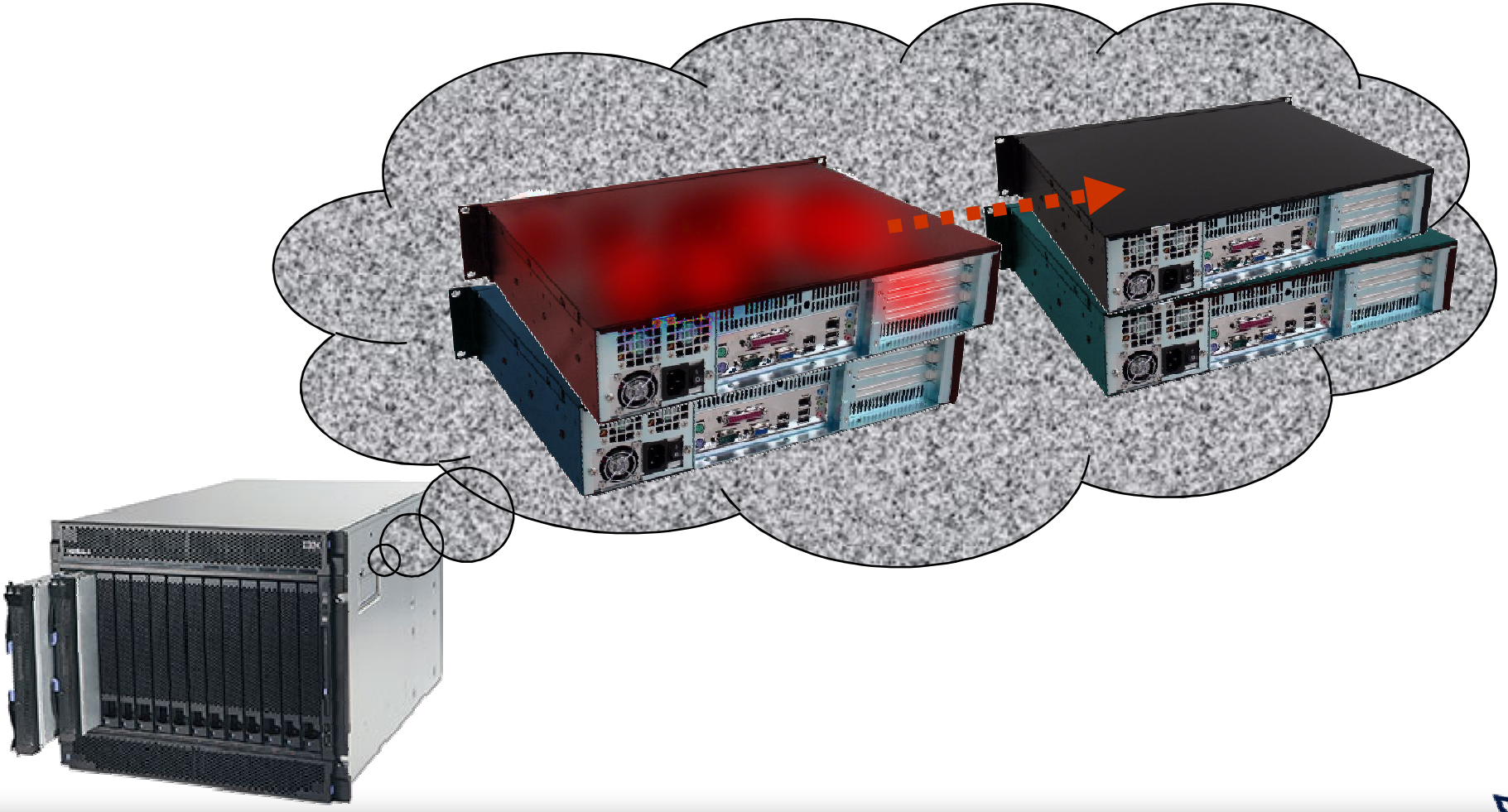
Virtualization

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



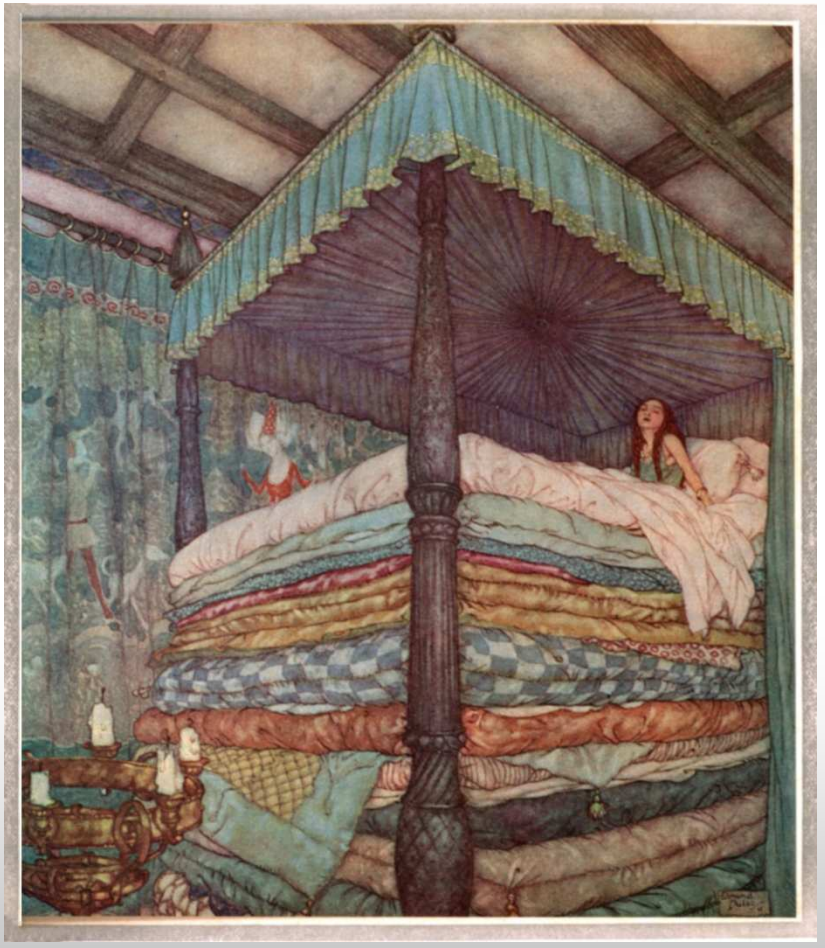
Virtualization

What if someone running on that hardware is malicious?



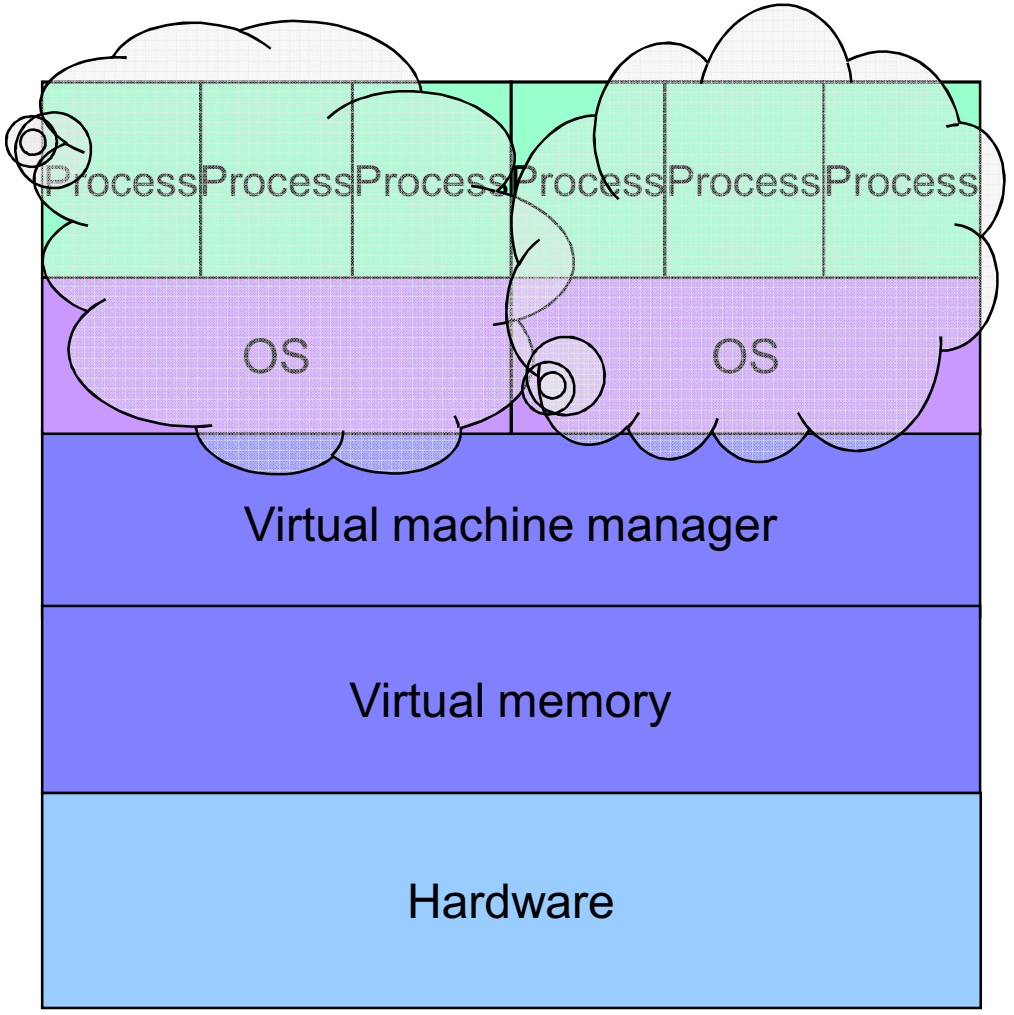
The Princess and the Pea

A Tale of Virtualization and Side Channels

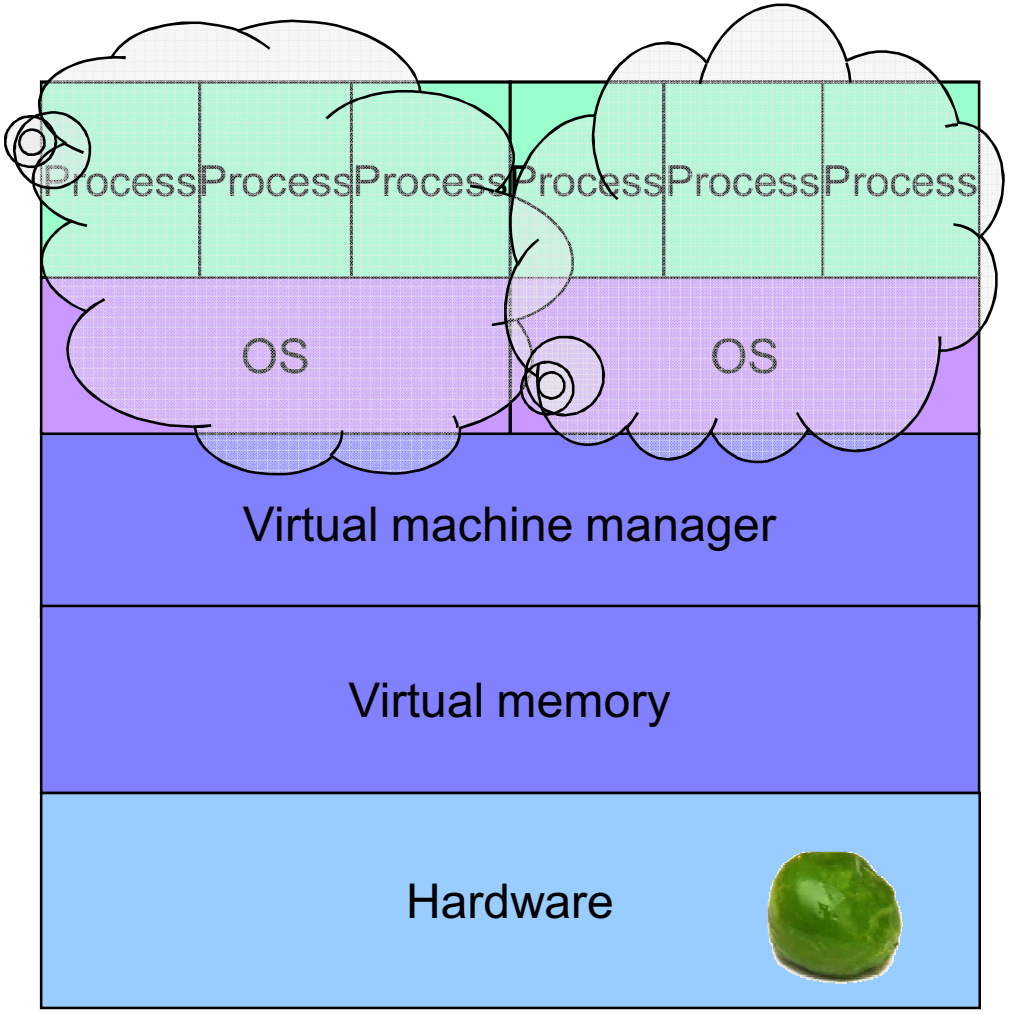


Virtualization: textbook description

20 mattresses

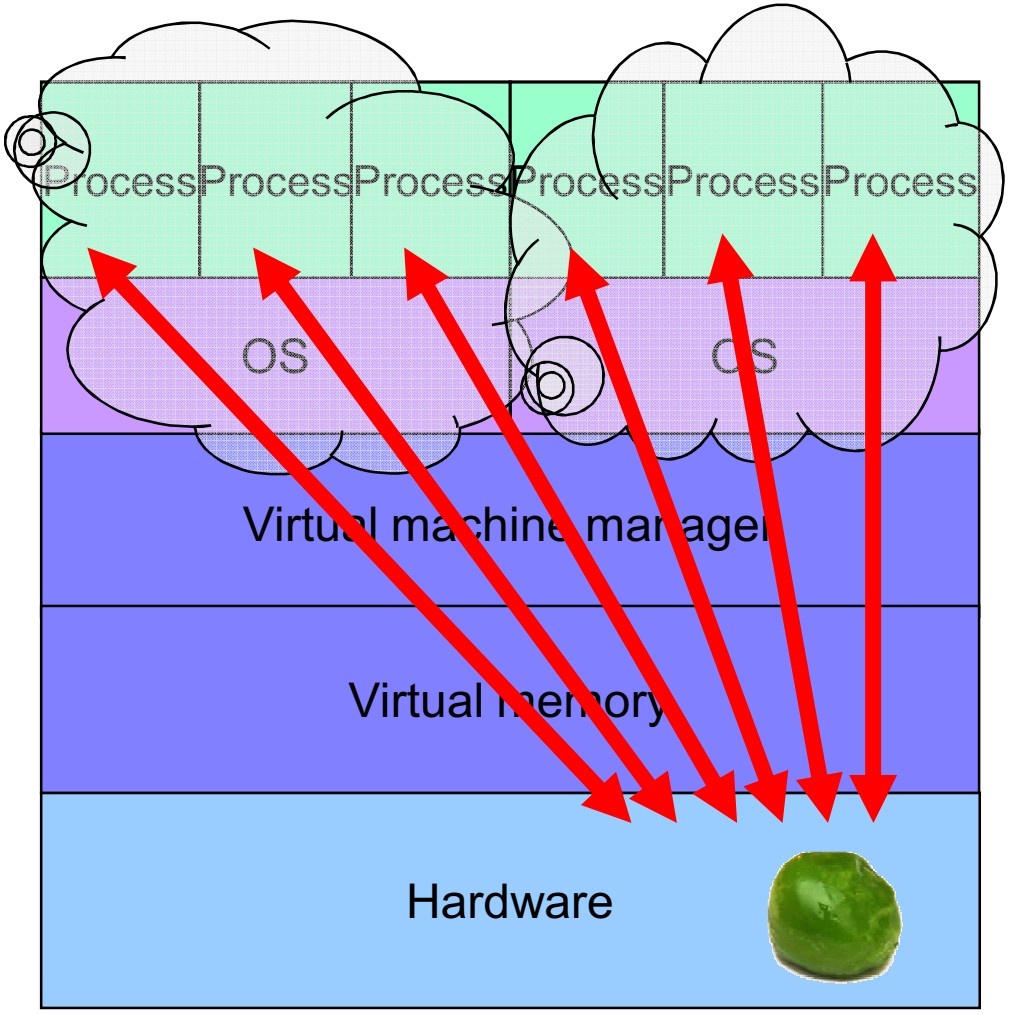


Cross-talk through architectural channels



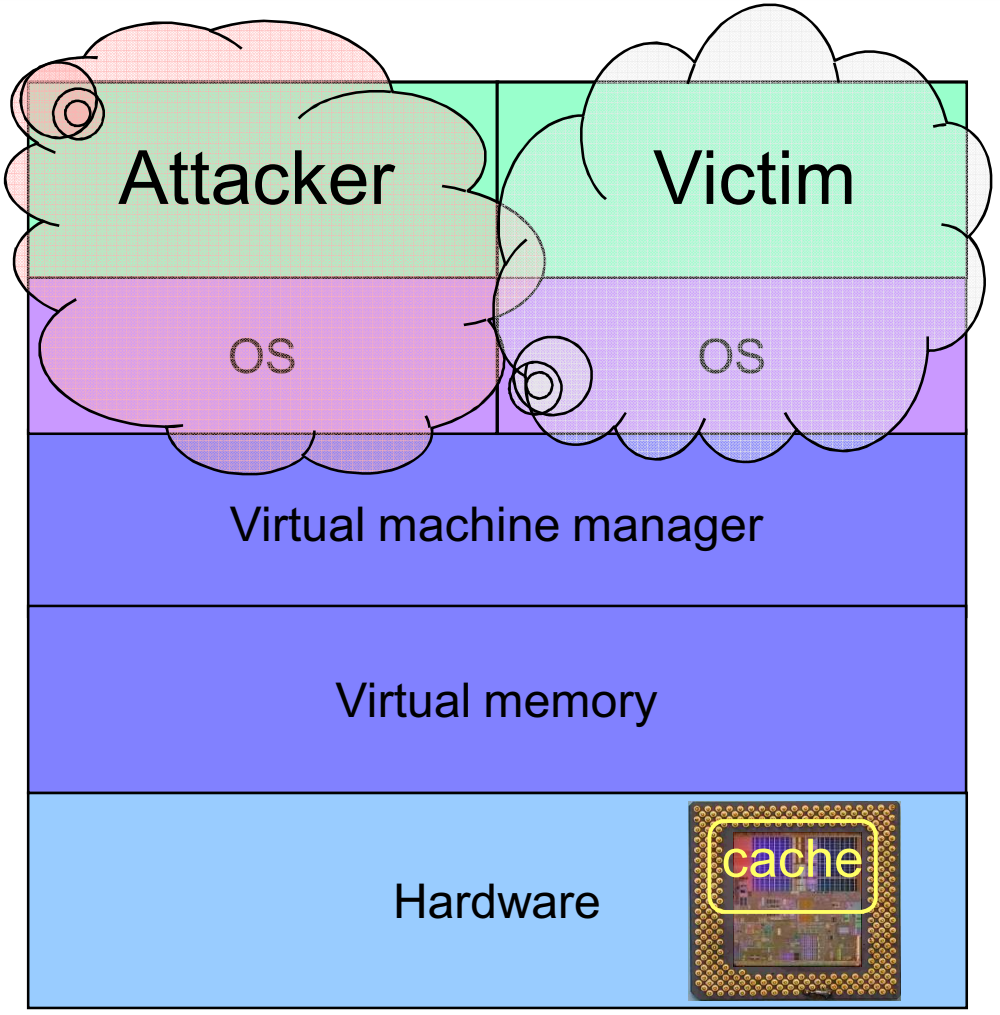
Cross-talk through architectural channels

- Contention for shared hardware resources



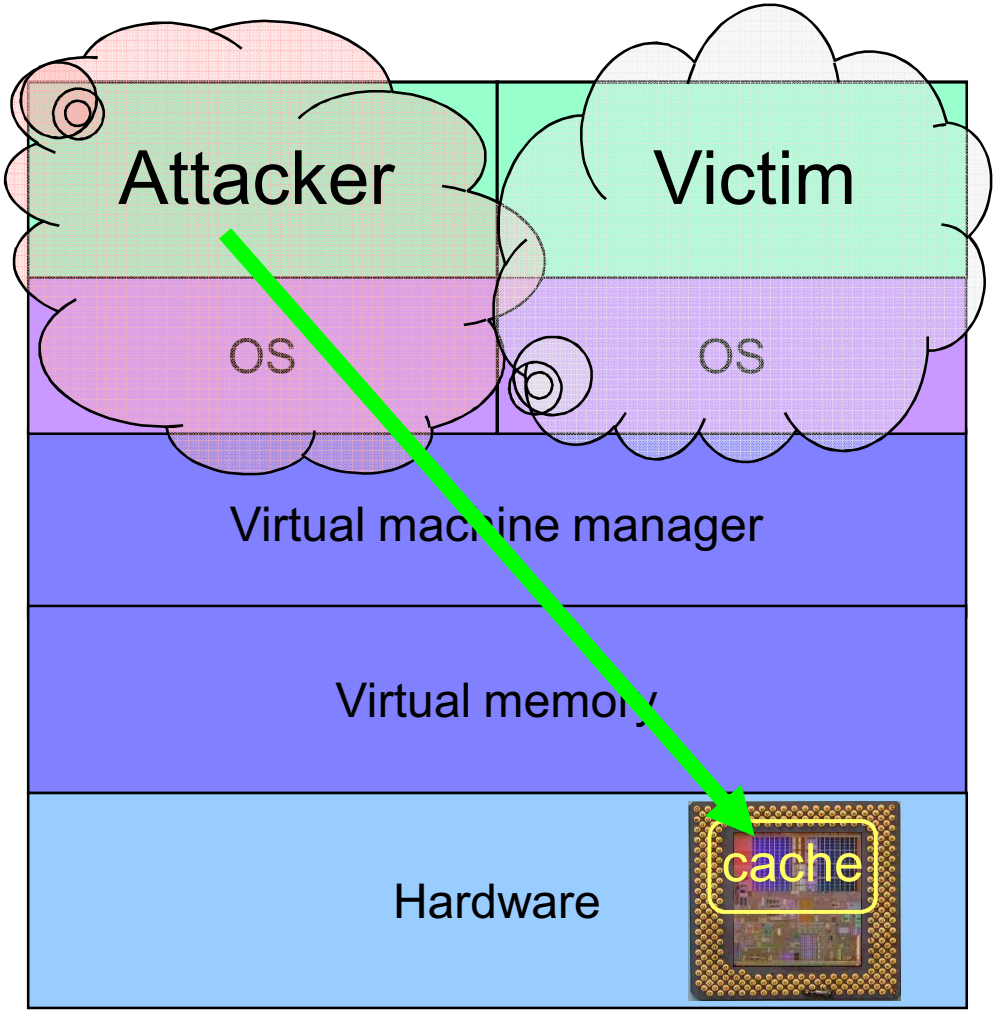
Cross-talk through architectural channels

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



Cross-talk through architectural channels

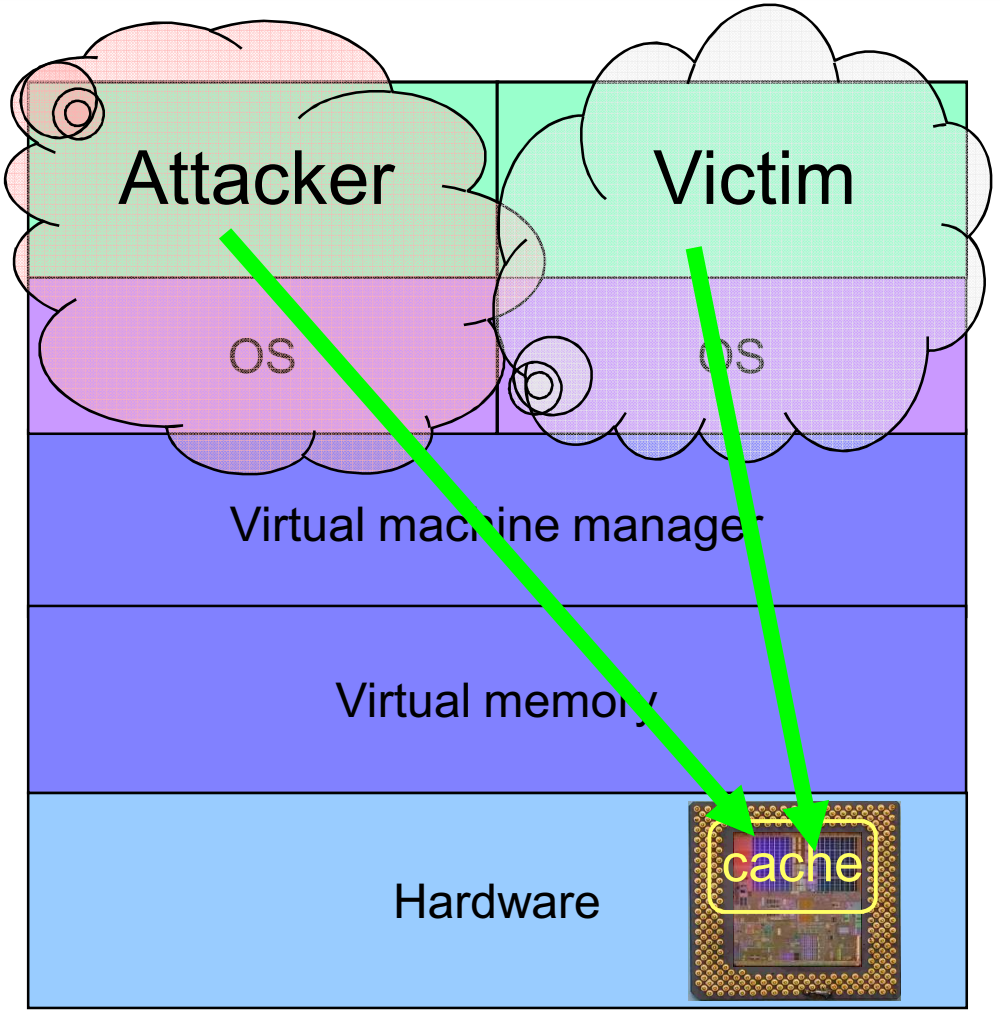
- Contention for shared hardware resources
- Example: contention for **CPU data cache**



<1 ns latency

Cross-talk through architectural channels

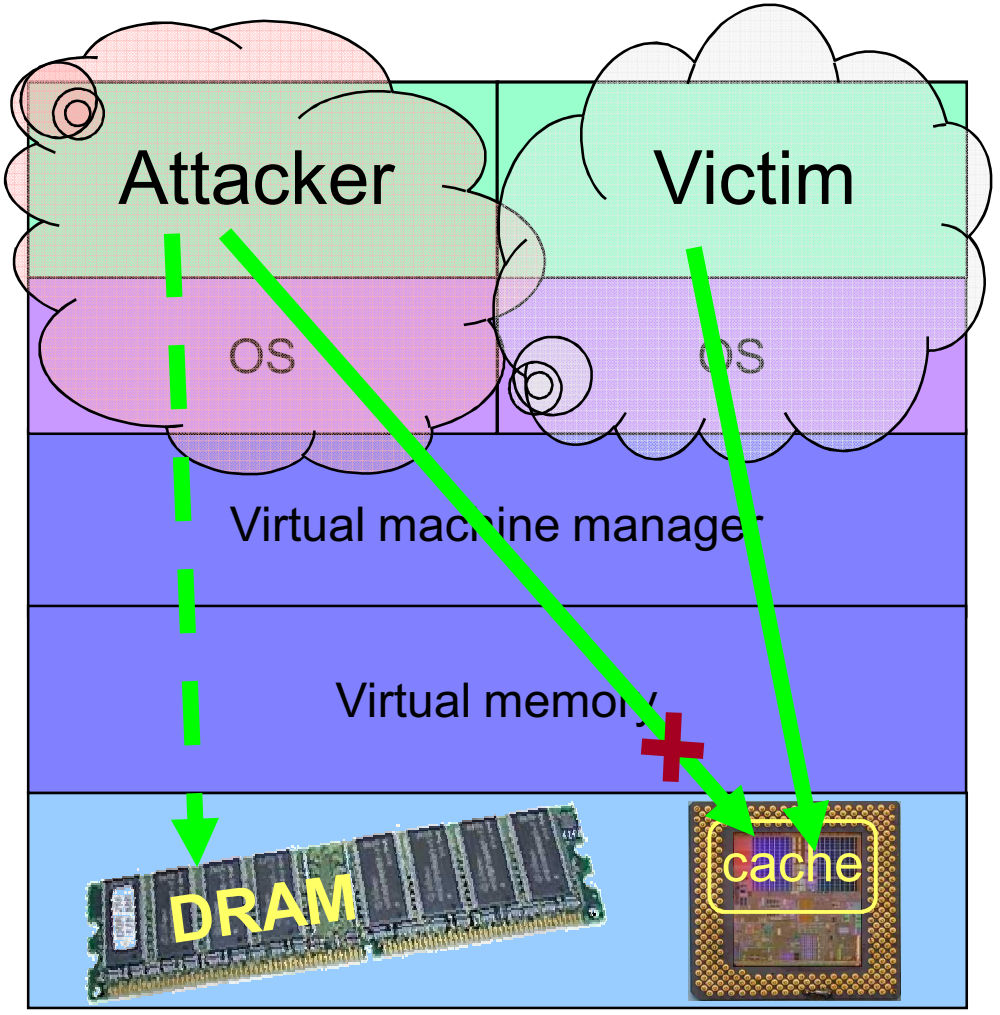
- Contention for shared hardware resources
- Example: contention for **CPU data cache**



<1 ns latency

Cross-talk through architectural channels

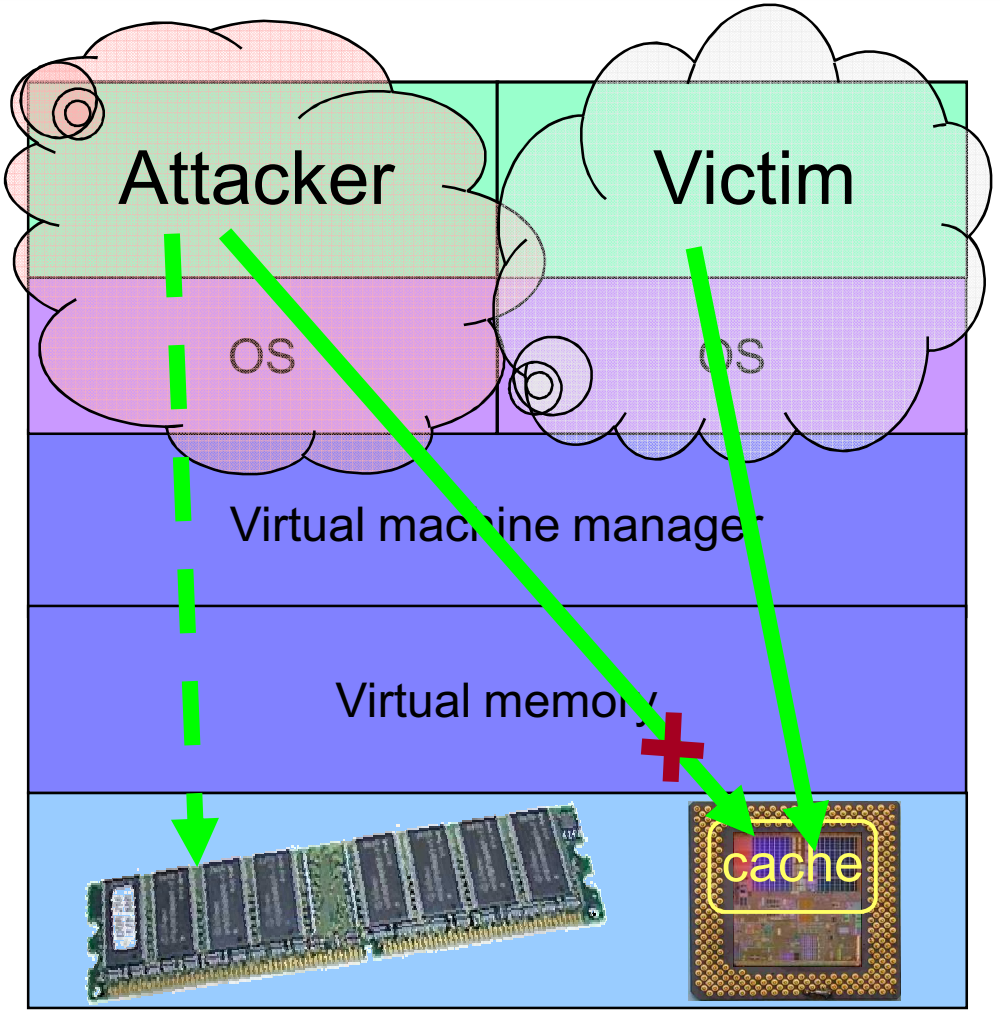
- Contention for shared hardware resources
- Example: contention for **CPU data cache**



~100 ns latency <1 ns latency

Cross-talk through architectural channels

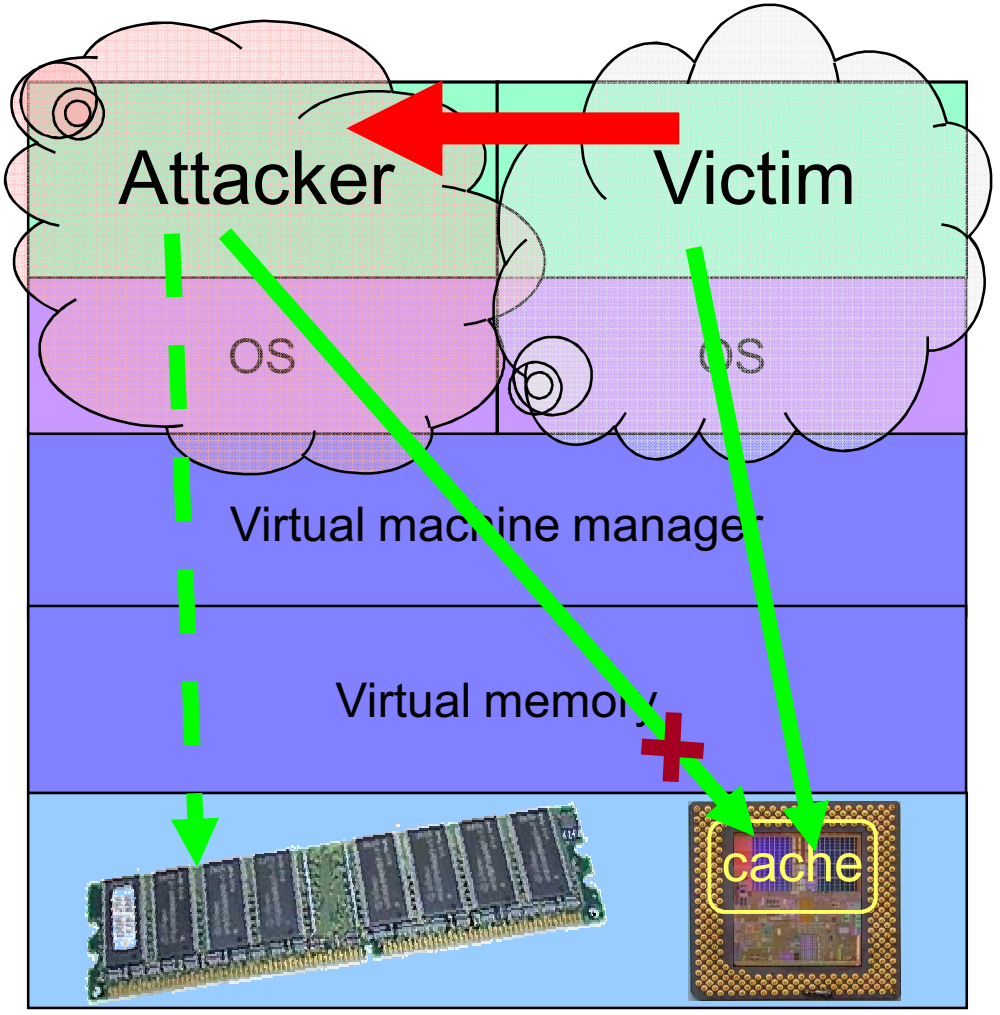
- Contention for shared hardware resources
- Example: contention for CPU data cache leaks memory access patterns.



~100 ns latency <1 ns latency

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.



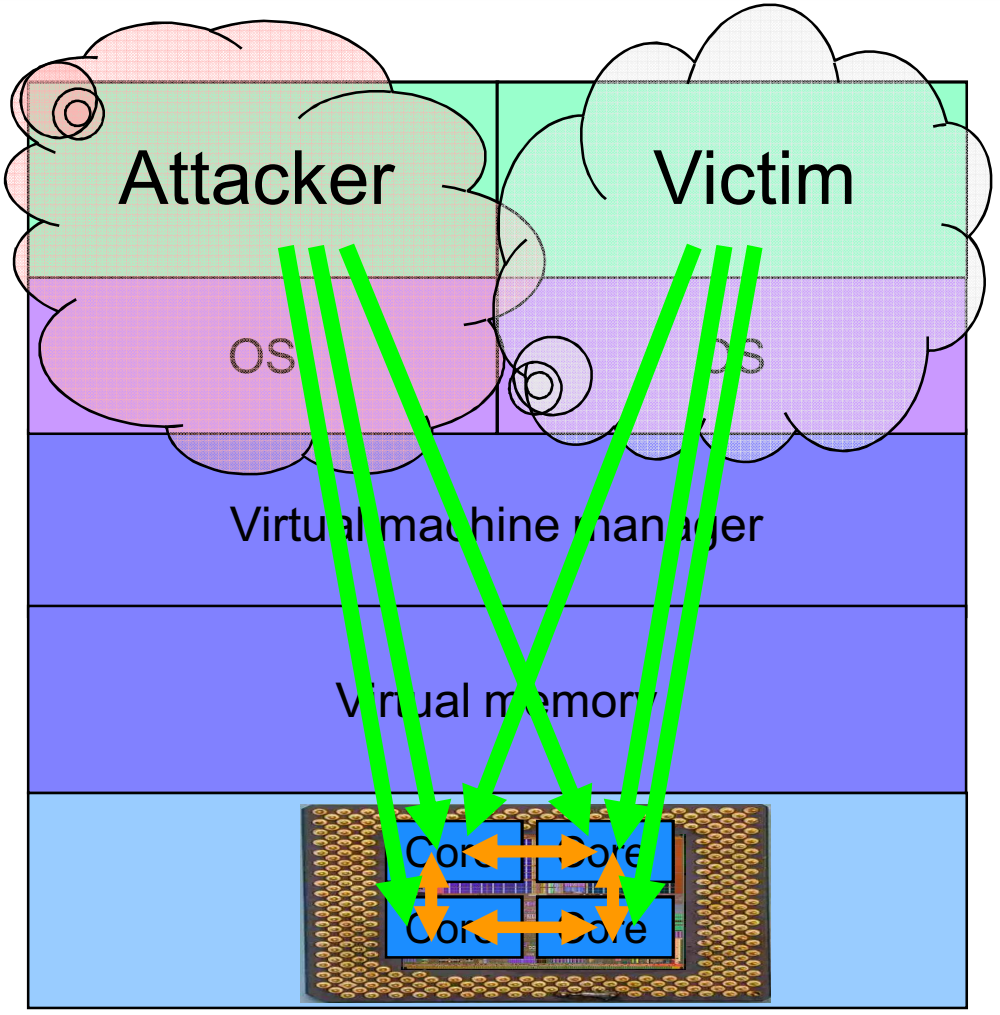
The multicore/multithread world

CPU parallelism in is becoming ubiquitous:

- Multicore
- Simultaneous multithreading ("HyperThreading")
- SMP

This trend is fundamental: CPU transistor count keeps growing exponentially, but clock rate and on-chip bandwidth aren't.

Parallelism aggravates architectural side channels.



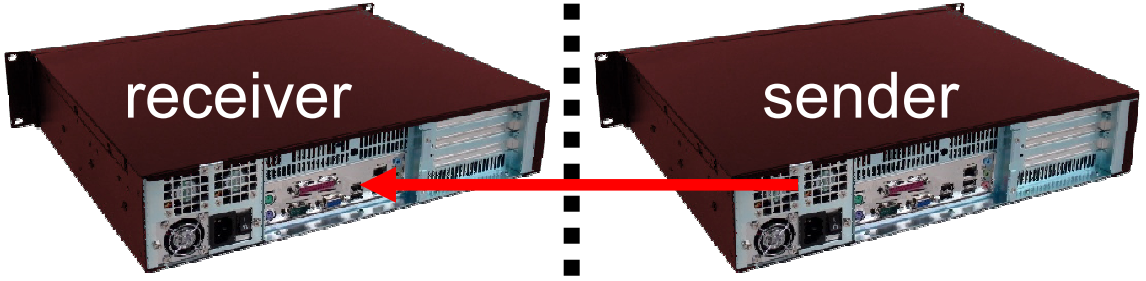
Inter-core and inter-thread crosstalk

Attack types

Inadvertent information channels between processes running on the same system:

- Side channels

- Covert channels
  collaborate to circumvent mandatory access controls



Most generally:

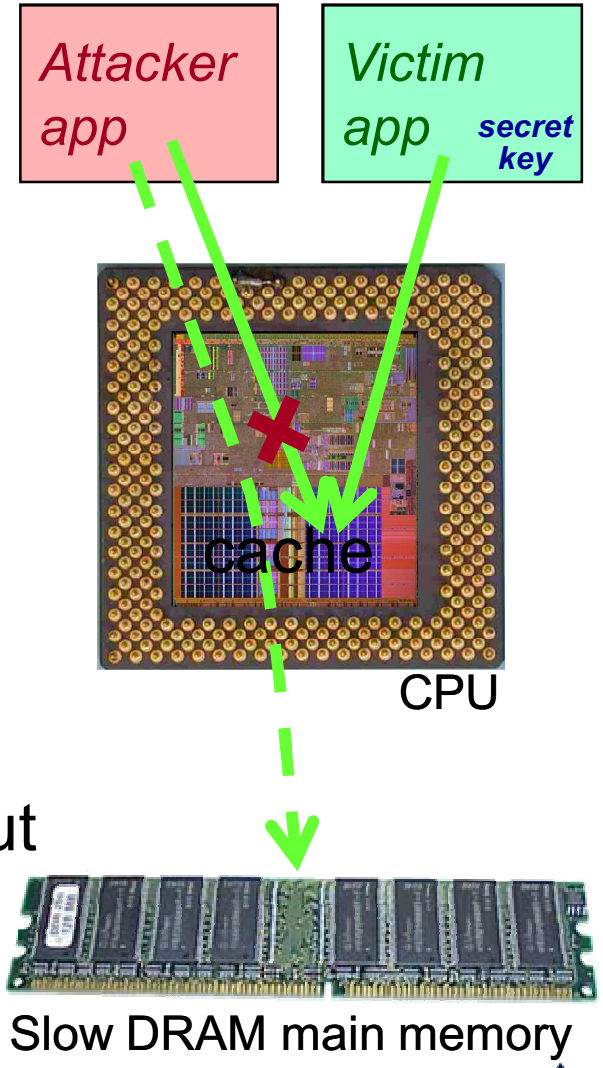
- Violate information flow control
  (e.g., Flume)

[Krohn Tromer 09]

Cache attacks

[Osvik Shamir Tromer 05]

- 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 <u>data</u> is subject to memory protection…
- But the <u>metadata</u> 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:

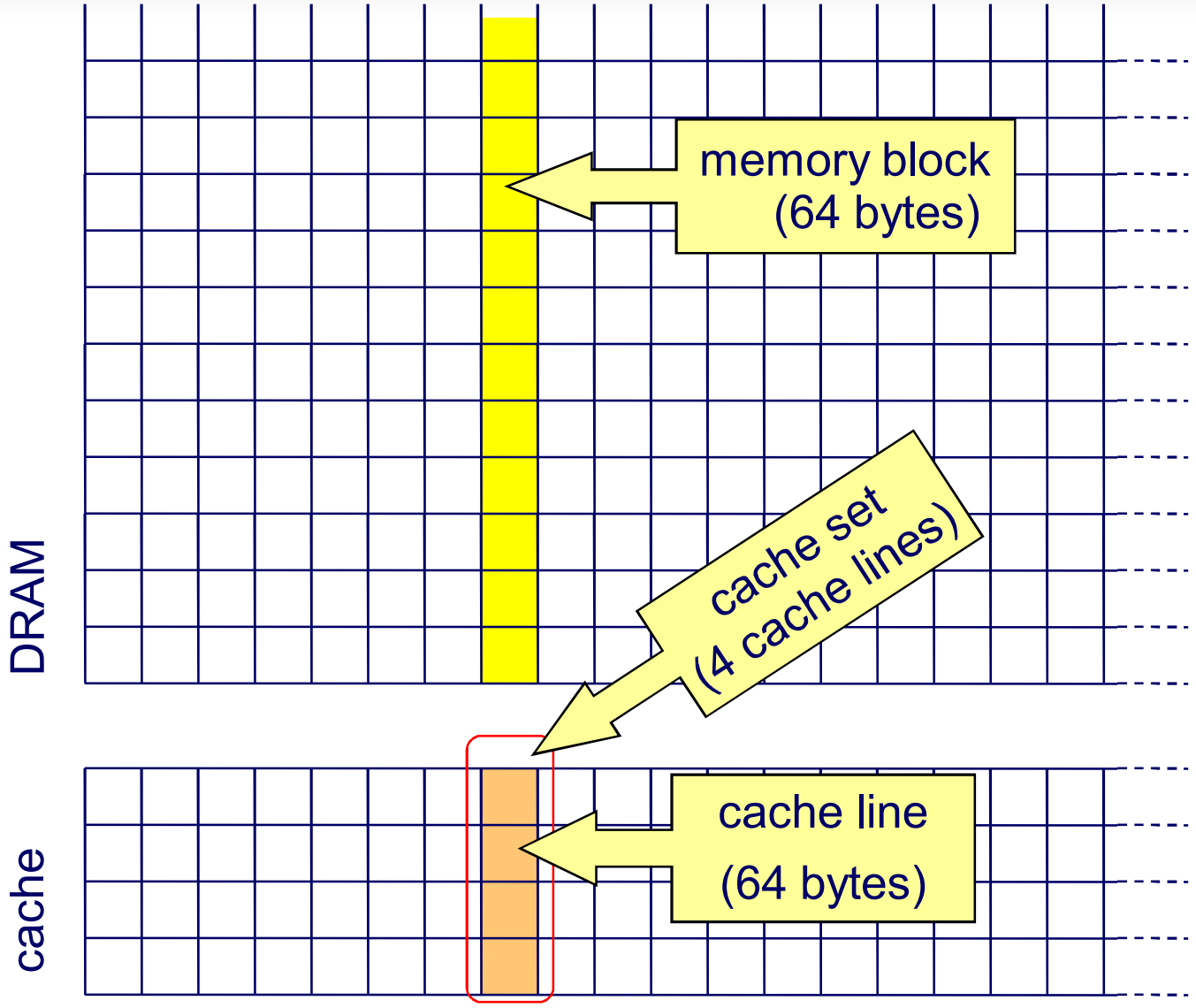
- Multiple indices per cache line
- Uncertain messages
- Noise

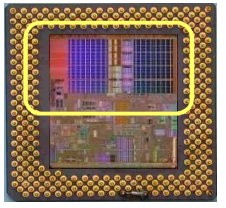
Requires further cryptographic and statistical analysis.

How to learn <u>addresses</u>?

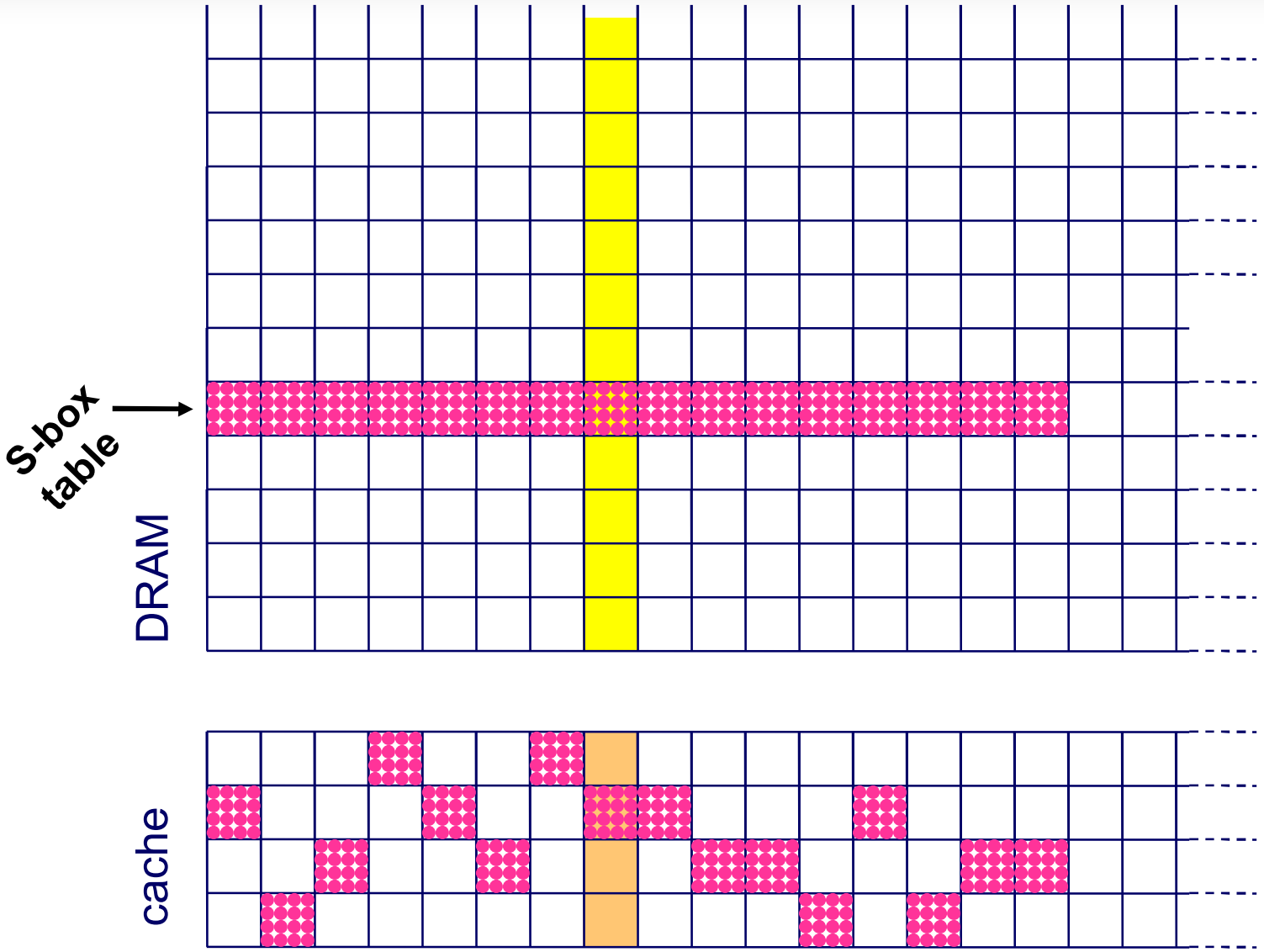
lookup index = plaintext ⊕ key

Associative memory cache

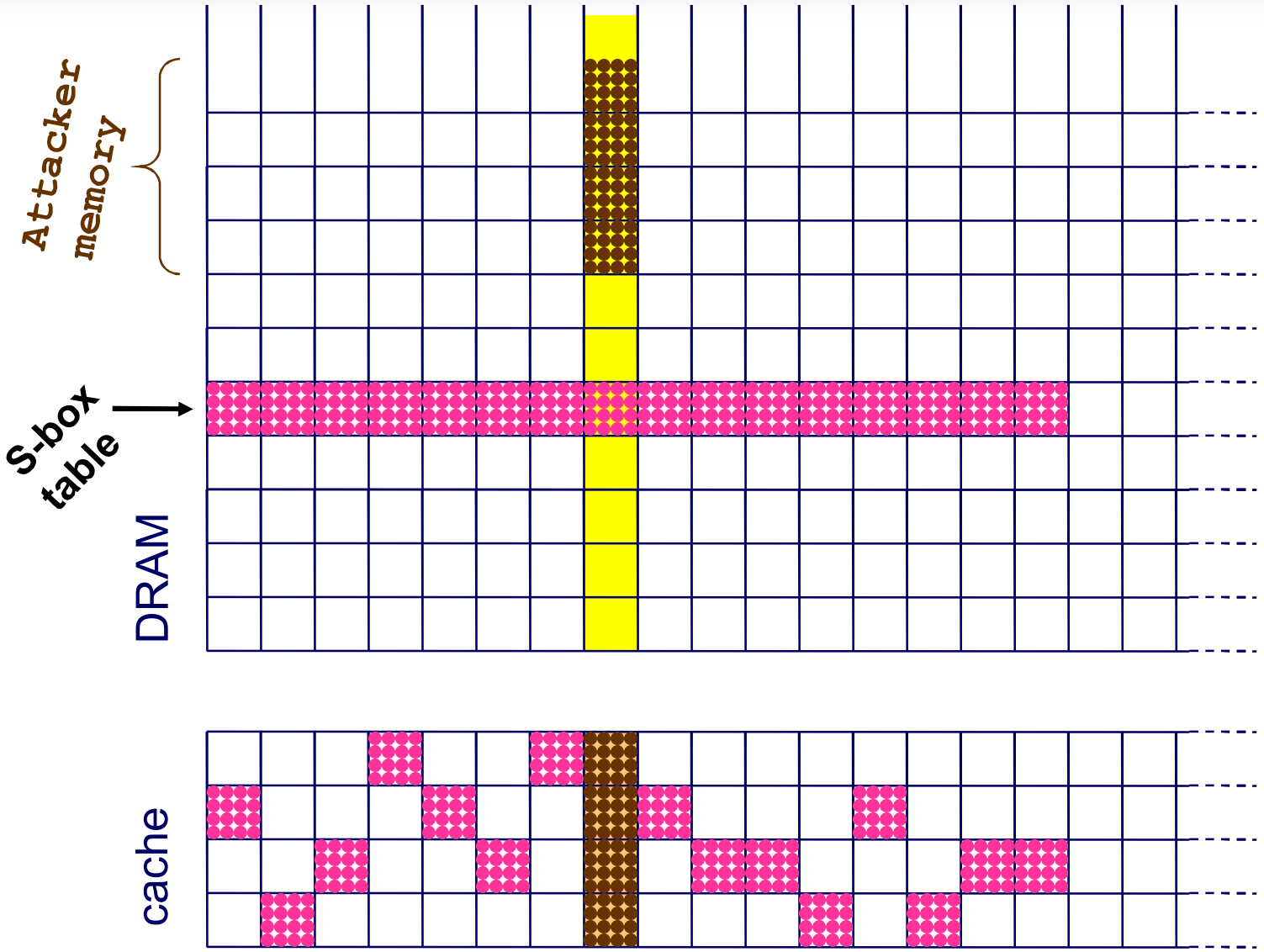




Victim's memory



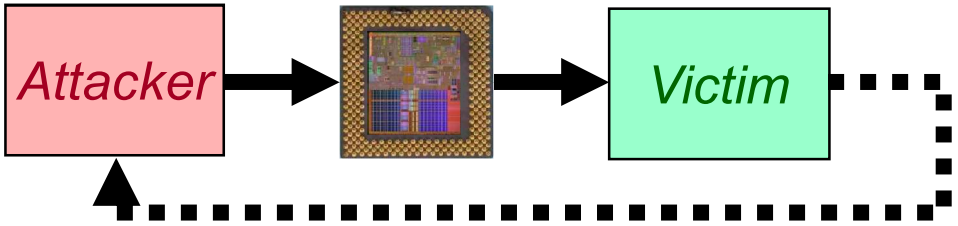
Detecting victim's memory accesses



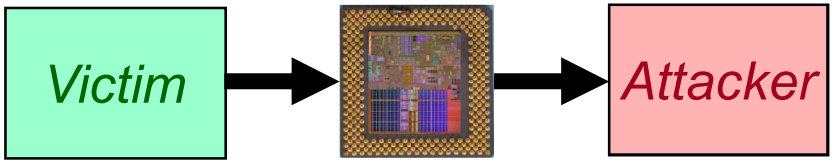
Measurement technique

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

- Effect of the cache on the victim

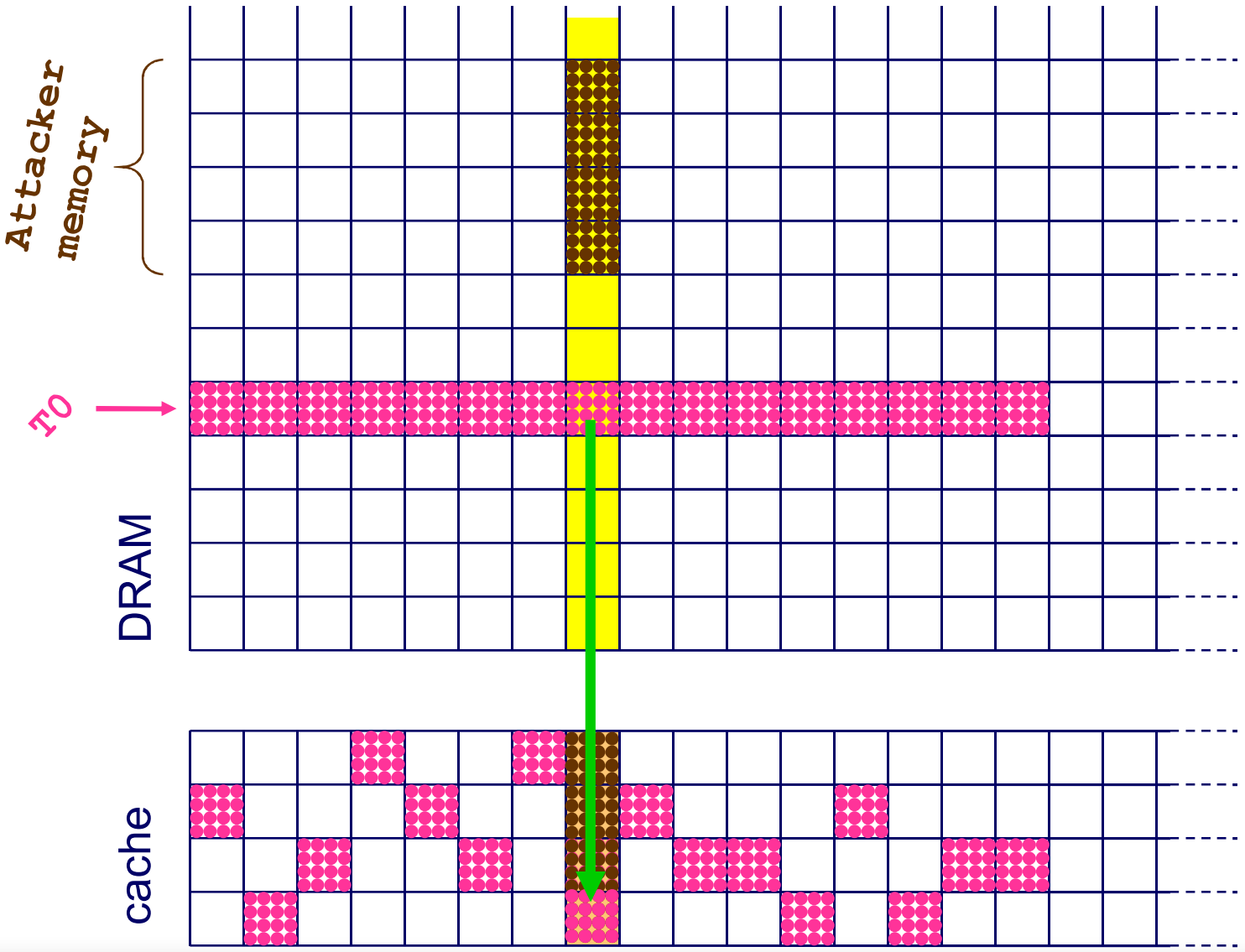


- Effect of victim on the cache



Measuring effect of cache on encryption:

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

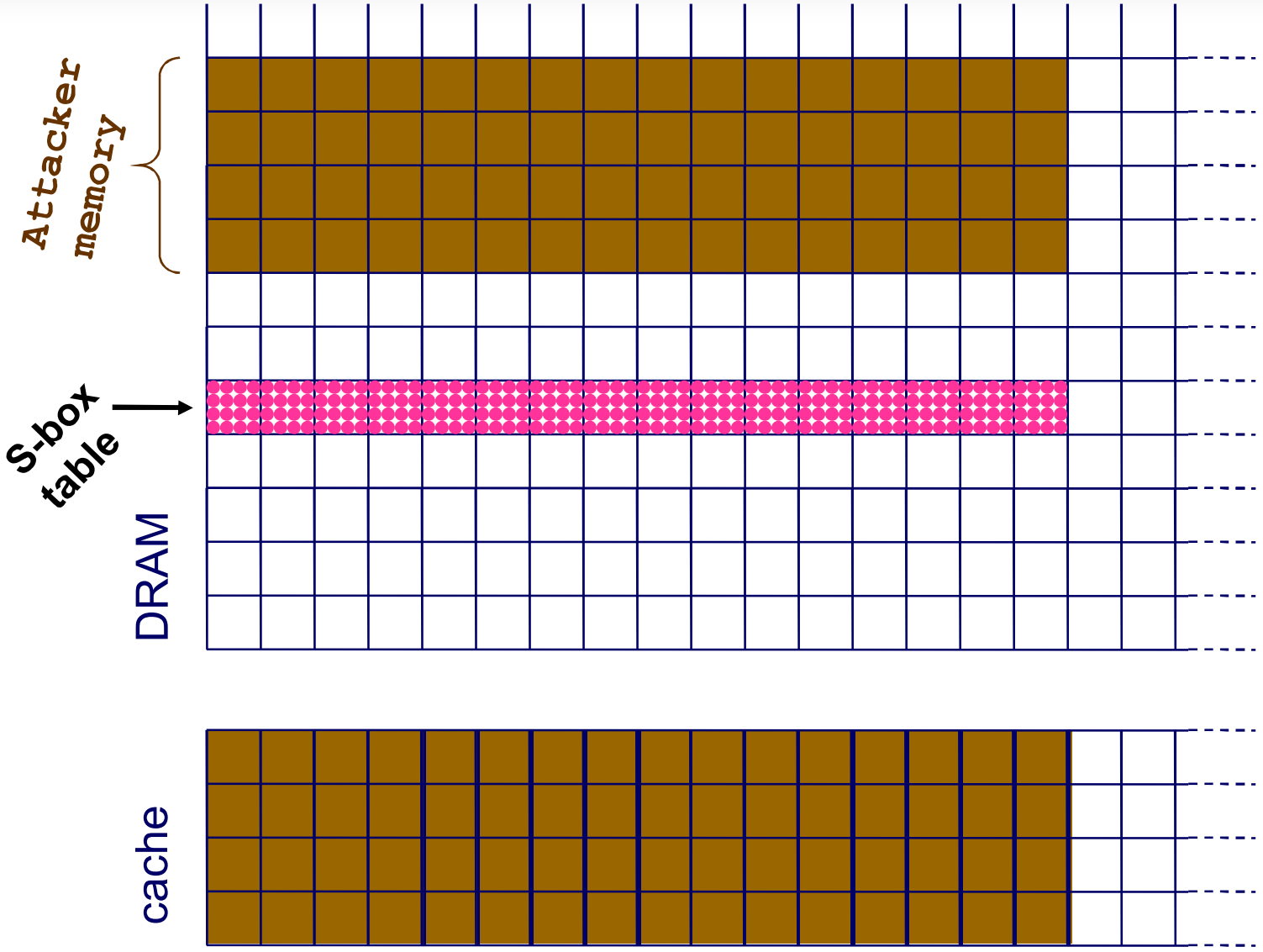


1. Victim's data fully cached

2. Attacker evicts victim's block

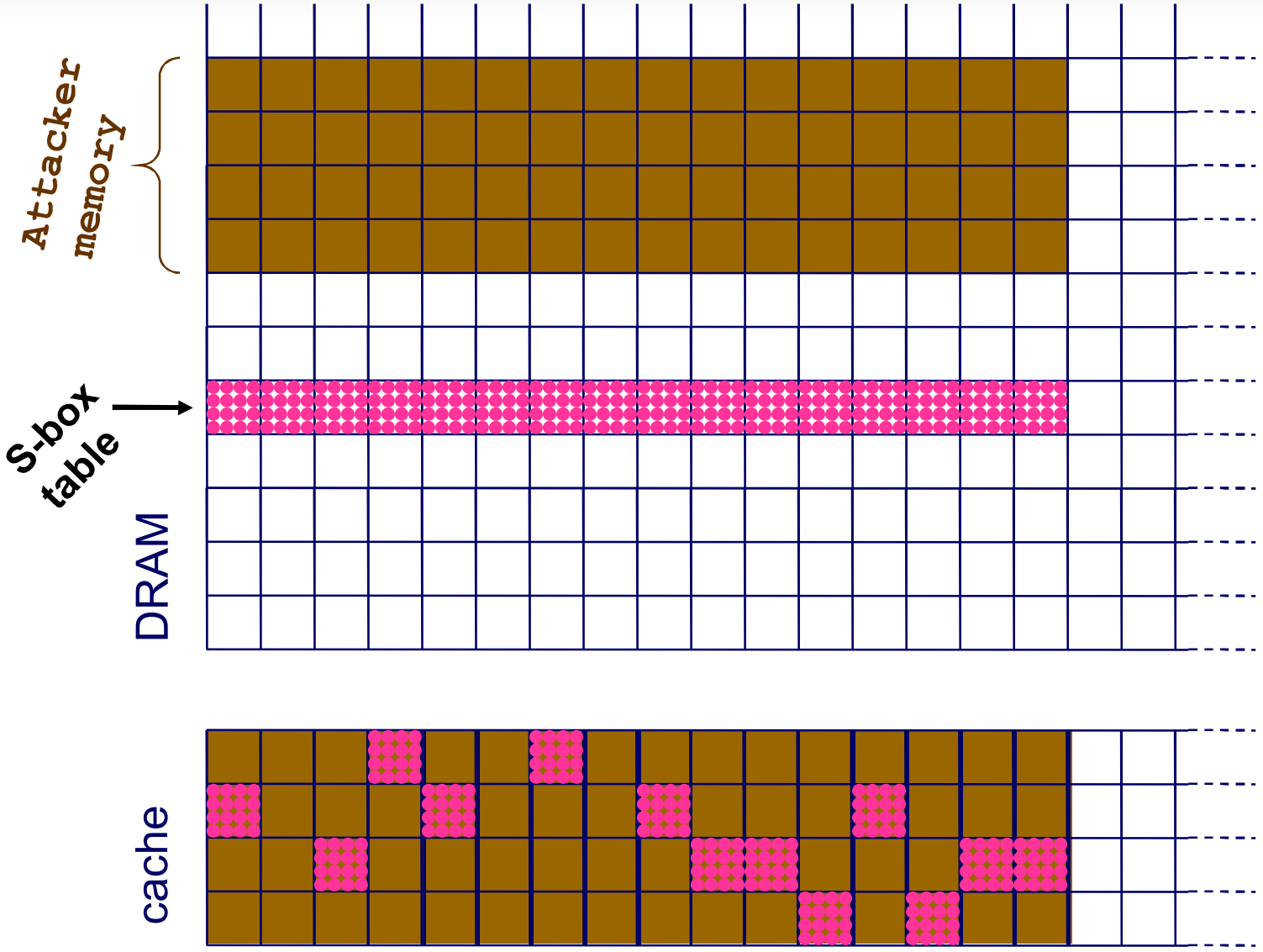
3. Attacker times the victim's next run. Slowdown?

Measuring effect of encryption on cache:
Attacker checks which of its own data was evicted by the victim.



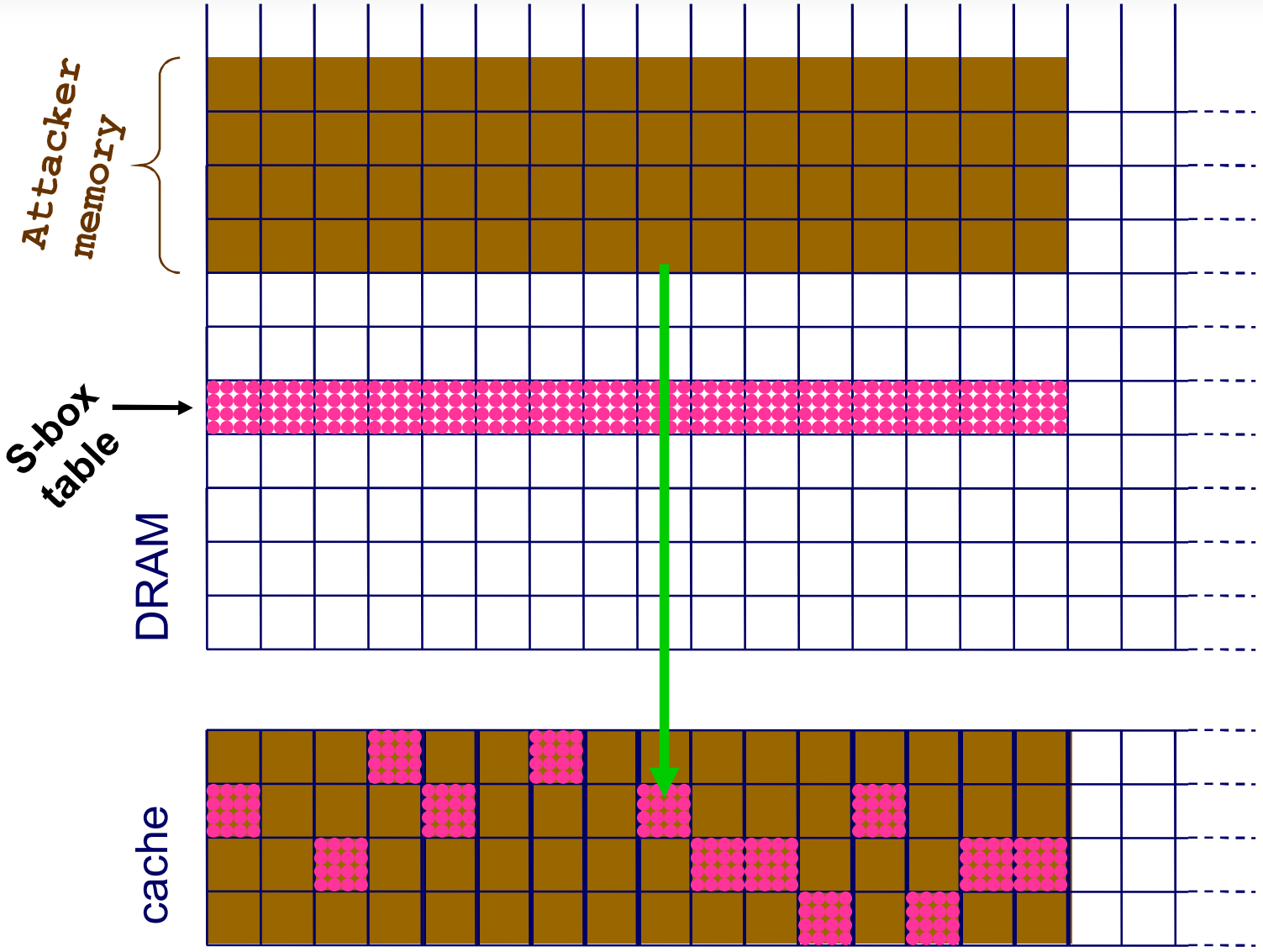
1. Fill cache with attacker's data

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

Measuring effect of encryption on cache:
Attacker checks which of its own data was evicted by the victim.

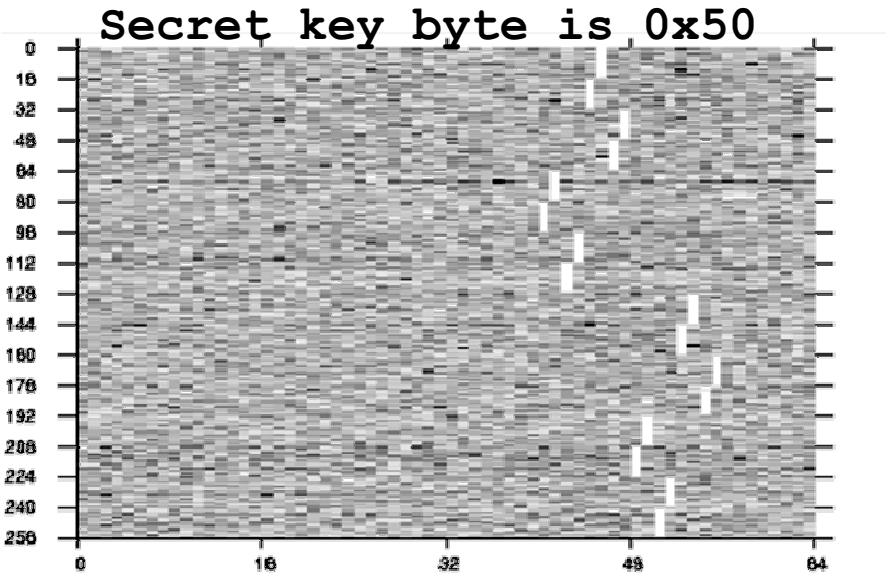


1. Fill cache with attacker's data
2. Trigger a single encryption
3. Access attacker memory again and see which cache sets are slow

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”

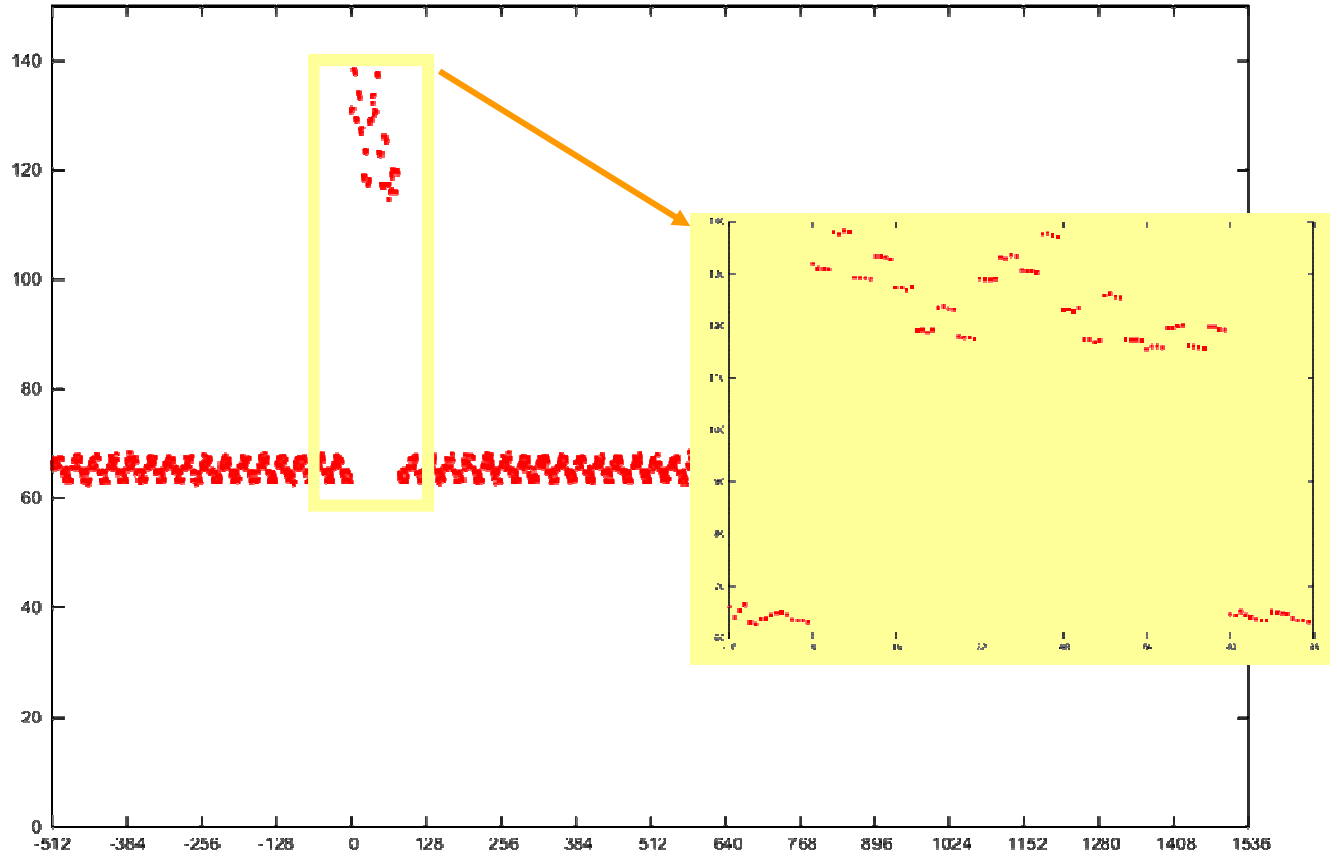
- Obtaining parallelism:
  - **Hyper**Threading (simultaneous multithreading)
  - Multi-core, shared caches, cache coherence
  - (Also: interrupts, scheduler)
- Attack vector:
  - Monitor cache statistics in real time
  - Encryption process is not communicating with anyone
    (no I/O, no IPC).
  - No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

Experimental results

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

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

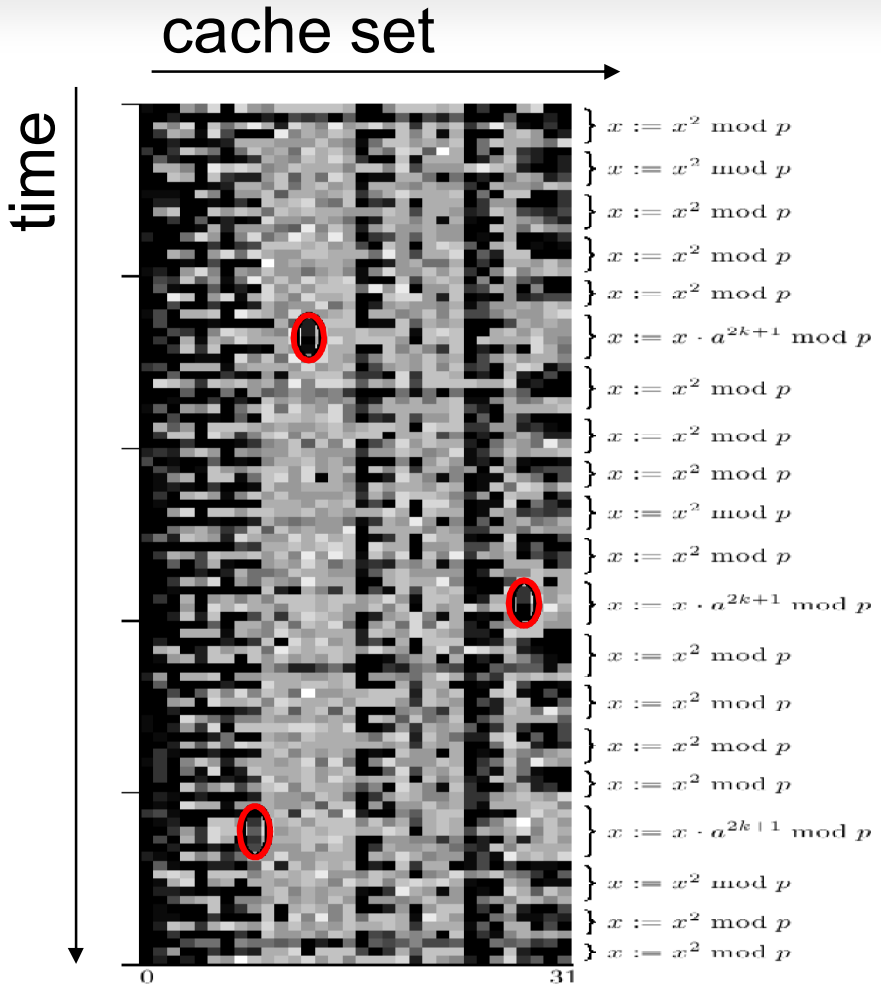
  Recovery of 45.7 key bits in one minute.



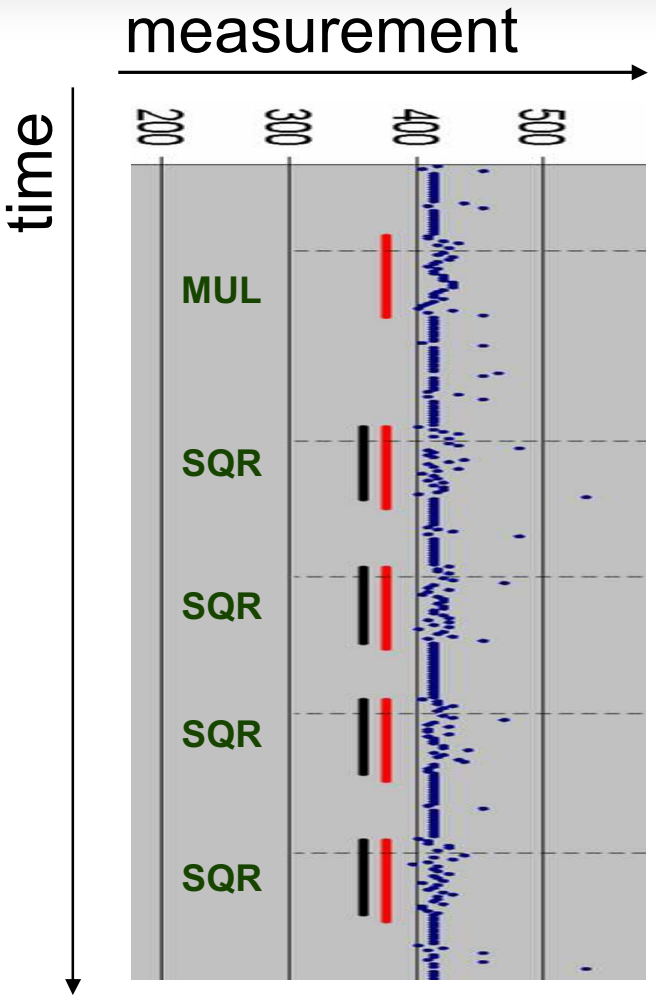
Other architectural attacks

- Covert channels [Hu ’91, ‘92]
- Hardware-assisted
  - Power trace [Page ’02]
- Timing attacks via internal collisions
  [Tsunoo Tsujihara Minematsu Miyuachi ’02]
  [Tsunoo Saito Suzaki Shigeri Miyauchi ’03]
- Model-less timing attacks [Bernstein ’04]
- RSA [Percival ’05]
- Exploiting the scheduler [Neve Seifrert ’07]
- Instruction cache Aciicmez ’07]
  - Exploits difference between code paths
  - Attacks are analogous to data cache attack
- Branch prediction [Aciicmez Schindler Koc ’06–’07]
  - Exploits difference in choice of code path
  - BP state is a shared resource
- ALU resources [Aciicmez Seifert ’07]
  - Exploits contention for the multiplication units
- *Many followups*

Example: attacks on RSA



Cache attack using HyperThreading
[Percival 05]



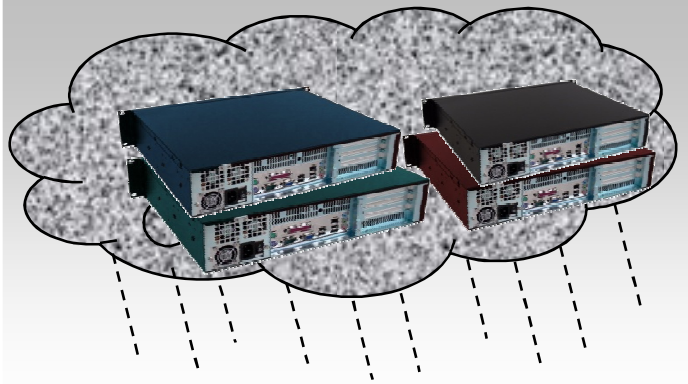
ALU multiplier attack
[Aciicmez Seifert 2007]

Implications?

Implications

- Multiuser systems
- Untrusted code, even if sandboxed
  (e.g., ActiveX, Java applets, managed .NET, JavaScript, Google Native Client, Silverlight)

  [work in progress]
- Digital right management
  The trusted path is leaky
  (even if verified by TPM attestation, etc.)
- Remote network attacks [work in progress]
- <u>Virtual machines</u>



Hey, You, Get Off of My Cloud! Exploring Information Leakage in Third-Party Compute Clouds

[Ristenpart Tromer Shacham Savage 09]

Virtualization

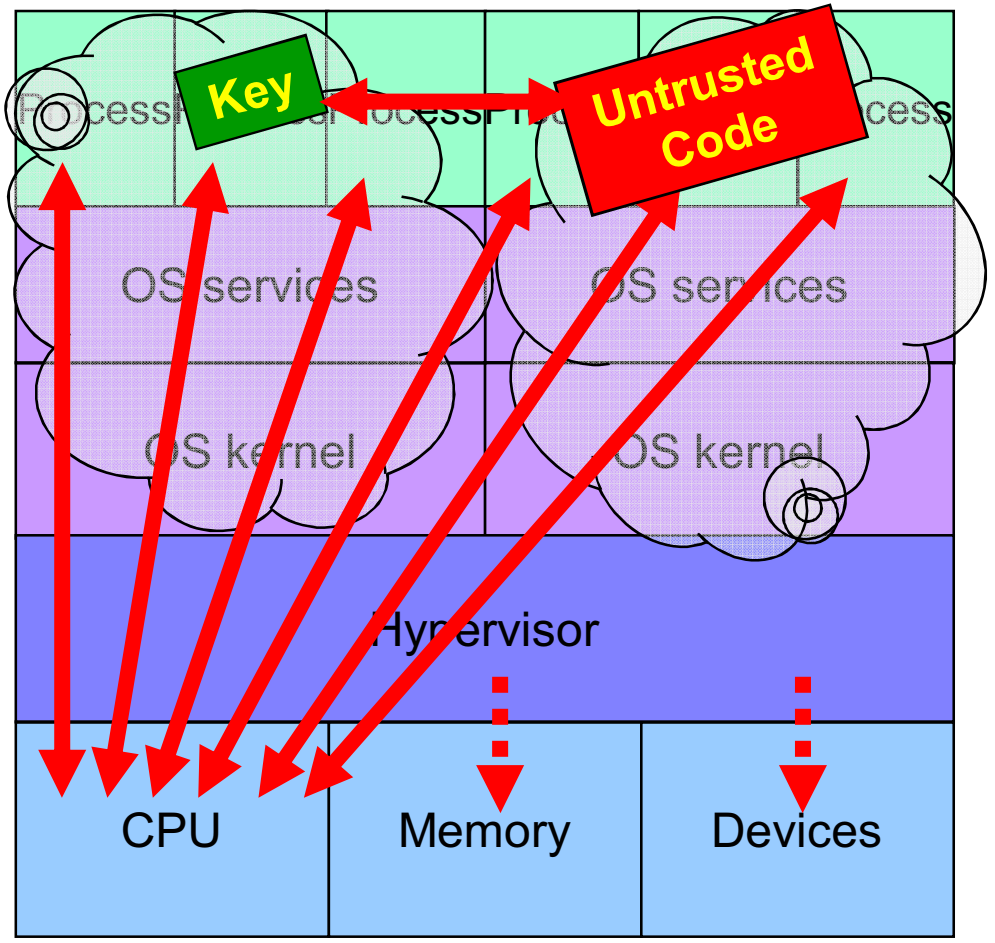
US patent 6,922,774 (NSA)

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.

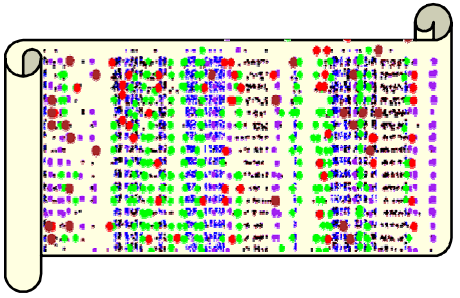
Architectural attacks in cloud computing: difficulties

- 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?

Hey, You, Get Off of My Cloud!
Exploring Information Leakage in Third-Party Compute Clouds

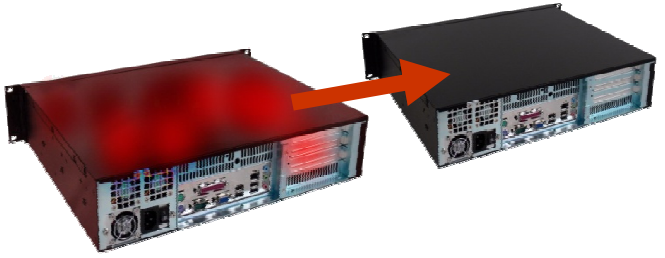
[Ristenpart Tromer Shacham Savage 09]

Demonstrated, using Amazon EC2 as a study case:



- 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 can be exfiltrated across VM boundary.

All via standard customer capabilities, using our own VMs to simulate targets.
We believe these vulnerabilities are general and apply to most vendors.

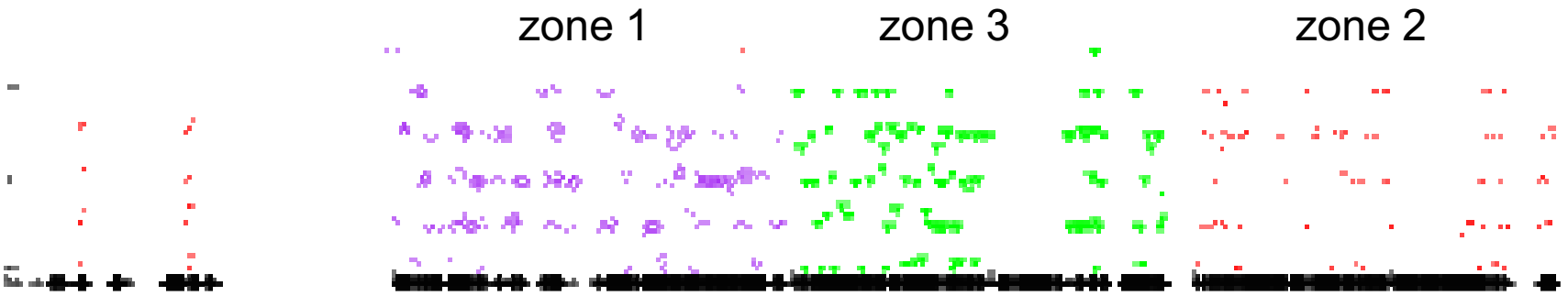
Cloud cartography

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

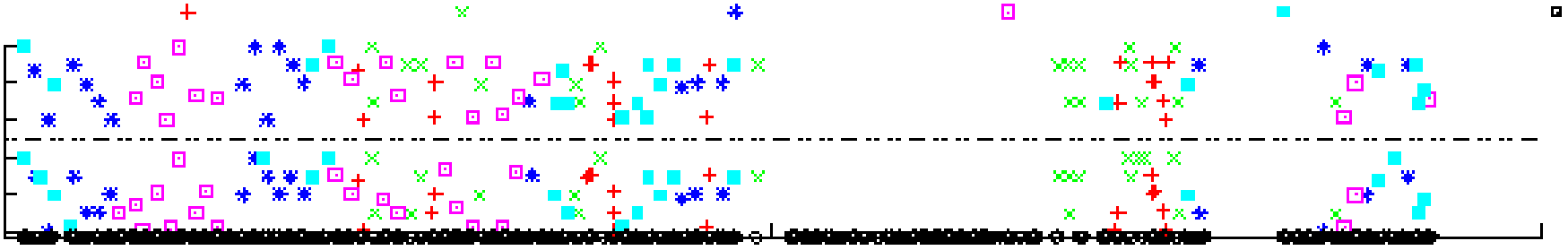
- On EC2, VMs can be co-resident only if they have identical creation parameters:
  - Region (US/Europe)
  - Availability zone (data center)
  - Instance type (machine pool)
- The cloud-internal IP addresses assigned to VMs are strongly correlated with their creation parameters.

We mapped 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

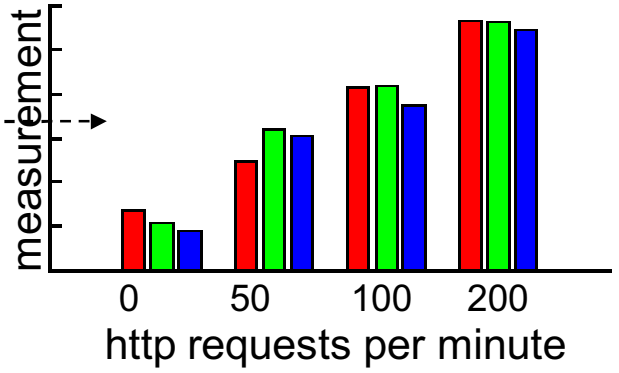
- EC2-specific:
  - Internal IP address are close
- Xen-specific:
  - Obtain and compare Xen Dom0 address
- Generic:
  - Network latency
  - Cross-VM architectural channels:
    send HTTP requests to target and observe correlation with cache utilization

Exploiting co-residence: cross-VM attacks

- Demonstrated:
  - 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]



- Work in progress:
  - Key leakage
  - More data exfiltration

Case III

Hardware

Wonderful software techniques

- Software engineering (review, tests)
- Formal verification
- Static analysis
- Dynamic analysis
- Programming languages type safety
- Reference monitors
- Sandboxing
- Information flow control
- …

F-35 Joint Strike Fighter

Information technology supply chain: headlines

The New York Times (May 9, 2008)

"F.B.I. Says the Military Had Bogus Computer Gear"

ars technica (October 6, 2008)

"Chinese counterfeit chips causing military hardware crashes"

The New York Times (May 6, 2010)

"A Saudi man was sentenced [...] to four years in prison for selling counterfeit computer parts to the Marine Corps for use in Iraq and Afghanistan."

Theory Strikes Back (later in the semester)

- Fully homomorphic encryption
- Multiparty computation
- Probabilistically checkable proofs
- Leakage-resilient cryptography

Course duties

- Questionnaire
- Exercises
- Reading
- Exams
  - Moed A: 20/02/12
  - Moed B: 27/04/12

Exercises

- 6 exercises, 4% each
- Due in 13 days
  (posted in an evening after class, due by a midnight before class)
- No late submissions
- Exercise + anti-exercise
  - Exercise: “in class we saw system X; suggest an attack Y”
  - Anti-exercise: “here is a description of a claimed attack Y on X, submitted by randomly-chosen student from class; assess Y and suggest a defense Z against this attack.”

Exercise assessment and grading

- Each includes a self-assessment
  - Point out flaws in own answers
  - Self-grade accordingly
- Each anti-exercise also includes assessing the preceding exercise
- Exercise grading:
  - Flawless answer merits full credit
  - Flawed answers with correct analysis of flaws in self-assessment merits 50% credit
  - Flawed answer with missing or incorrect analysis of flaw gets 0% credit
- Checking and final grading by course staff

Resources

- Course website: http://www.cs.tau.ac.il/~tromer/courses/infosec11/
- Mailing list (see website)
- The course material is not covered by any single book, but the most pertinent one is: Ross Anderson, *Security Engineering*, 2nd ed.
- Additional reading material during the semester.