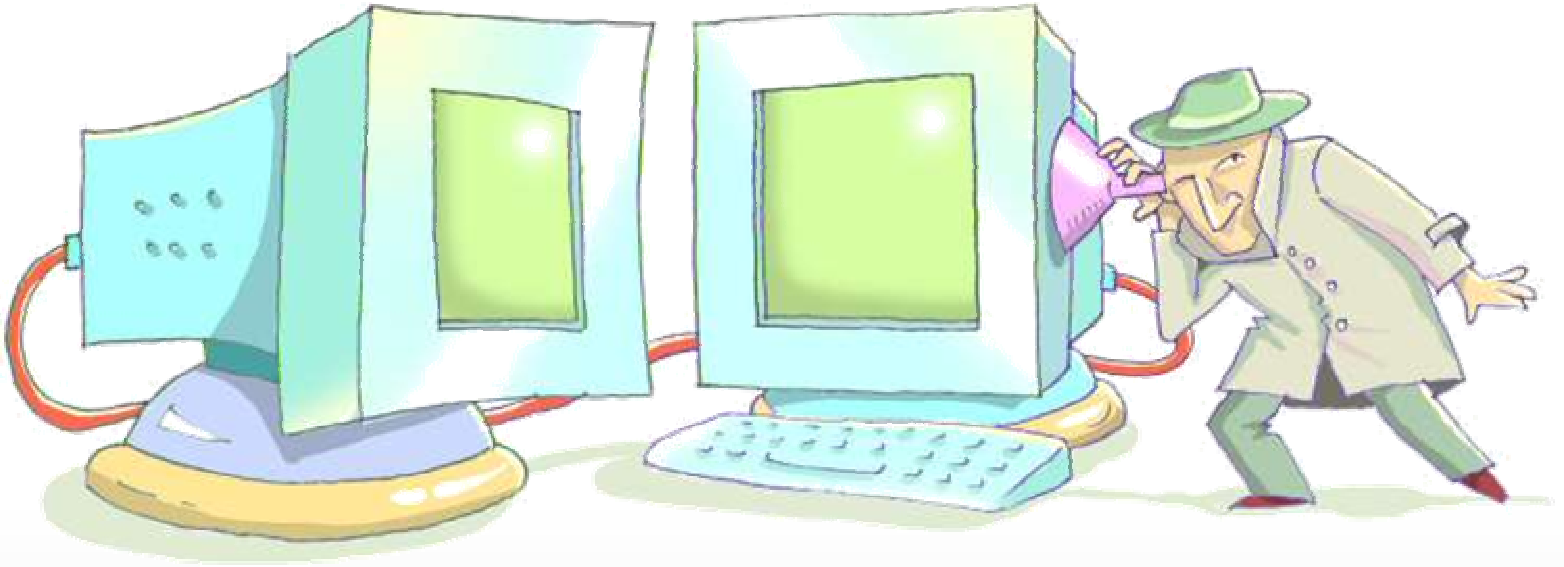
Architectural Side Channels in Cloud Computing



Eran Tromer
MIT
Includes joint works with Adi Shamir, Dag Arne Osvik
Stefan Savage, Hovav Shacham, Thomas Ristenpart

This talk

This talk will not be about

- AC^0 circuits
- Proofs
- Constructions

But:

- "Computation leaks information"
- *a whole lot of it!*

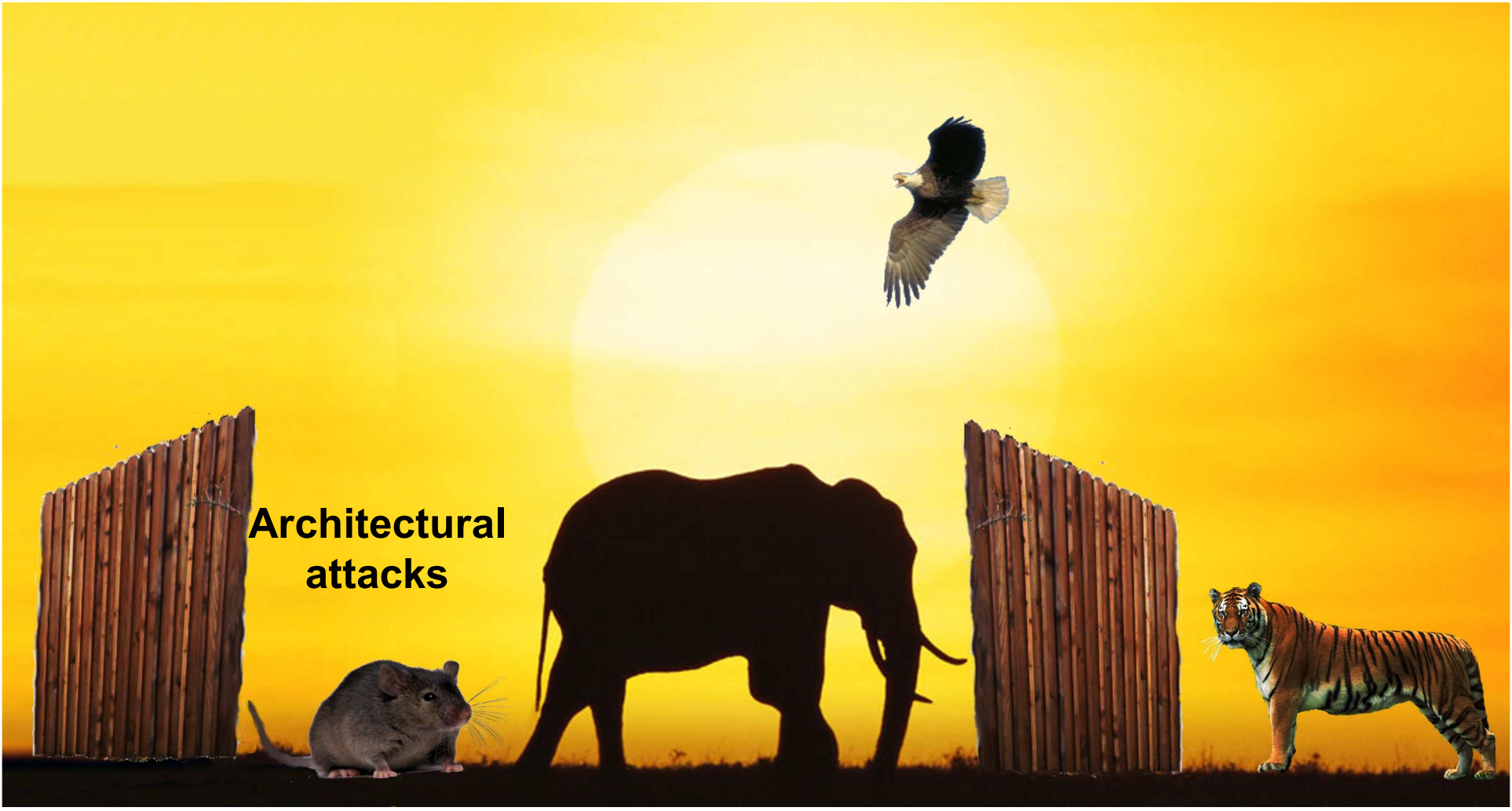
Hazards to cryptographic systems

Let's think **inside** the box

Physical attacks
- Electromagnetic
- Power
- Faults
- Acoustic
- Visible light
- (Timing)

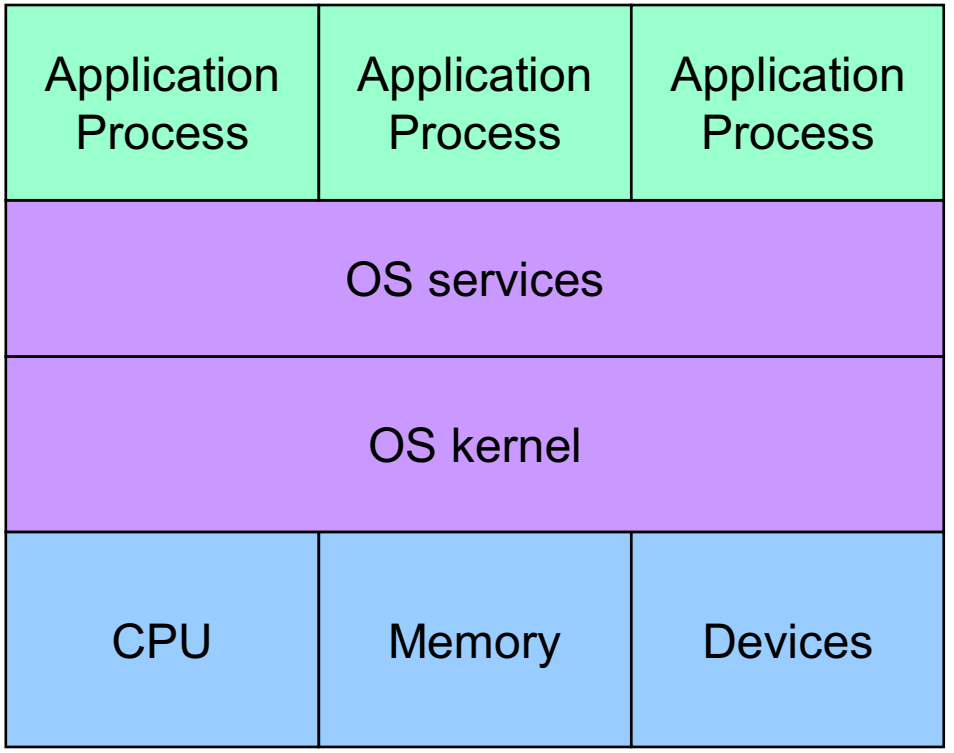
"Traditional" attacks
- Bad specification
- Insecure algorithm
- Code bugs
- Hardware intrusion
- Software intrusion

Hazards to cryptographic systems



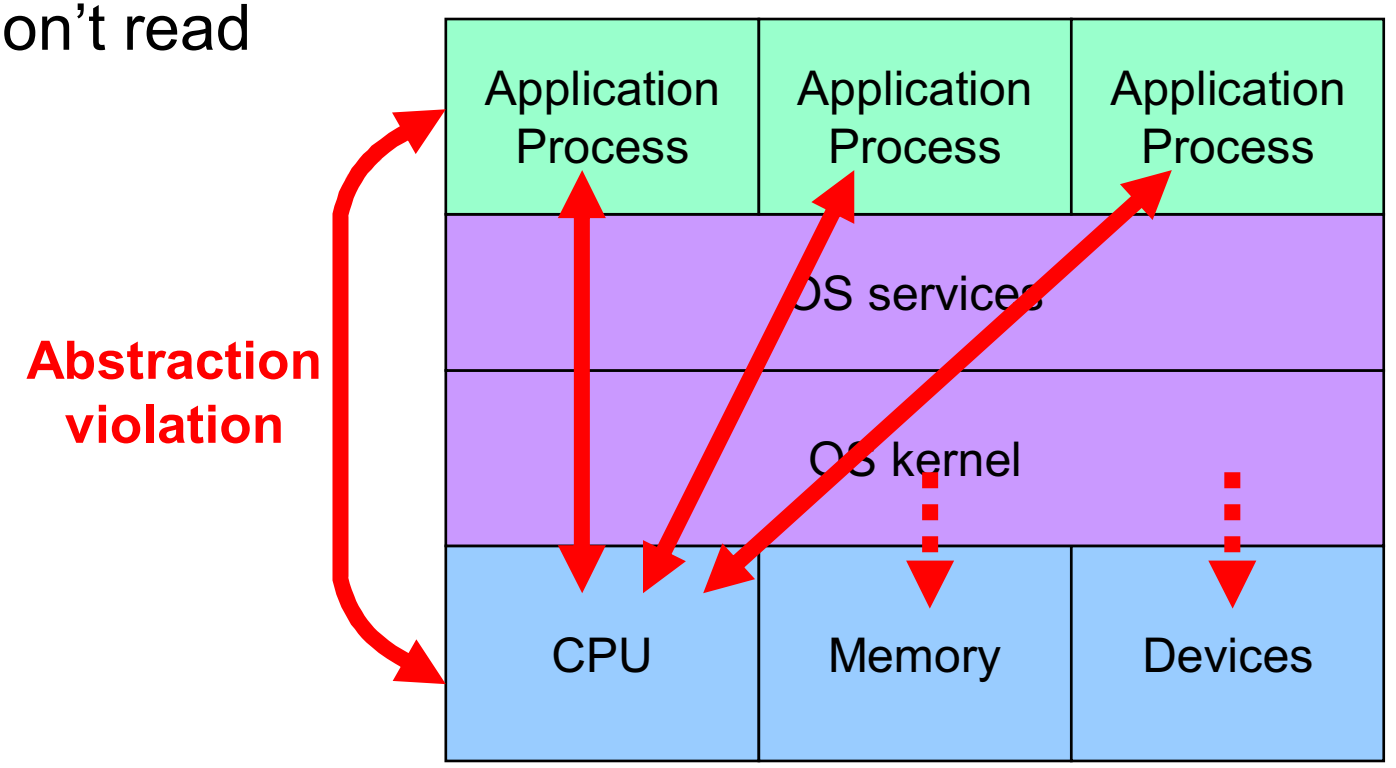
The ideal world

Textbook computer architecture:



The real world

Transistors don't read textbooks.



Inter-processcrosstalk creates side channels and covert channels

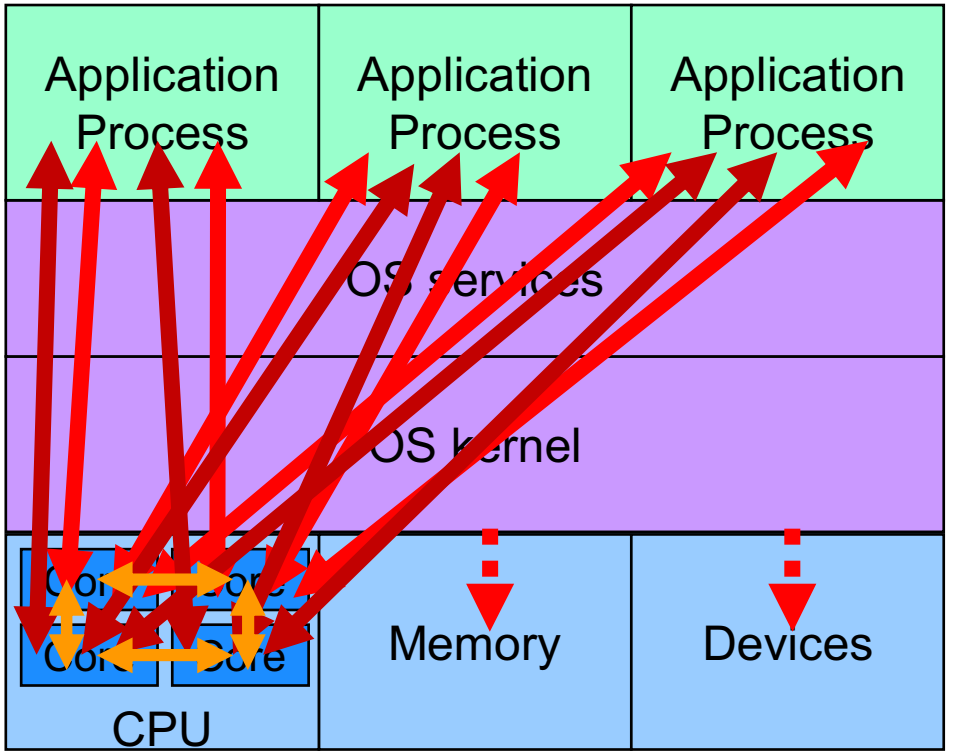
The multicore/multithread world

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 microarchitectural side channels.



Inter-core and inter-thread crosstalk

Attack types

Inadvertent information channels between processes running on the same system:

- Side channels (attacker and victim)
- Covert channels (cooperate to circumvent mandatory access control)

Most generally:

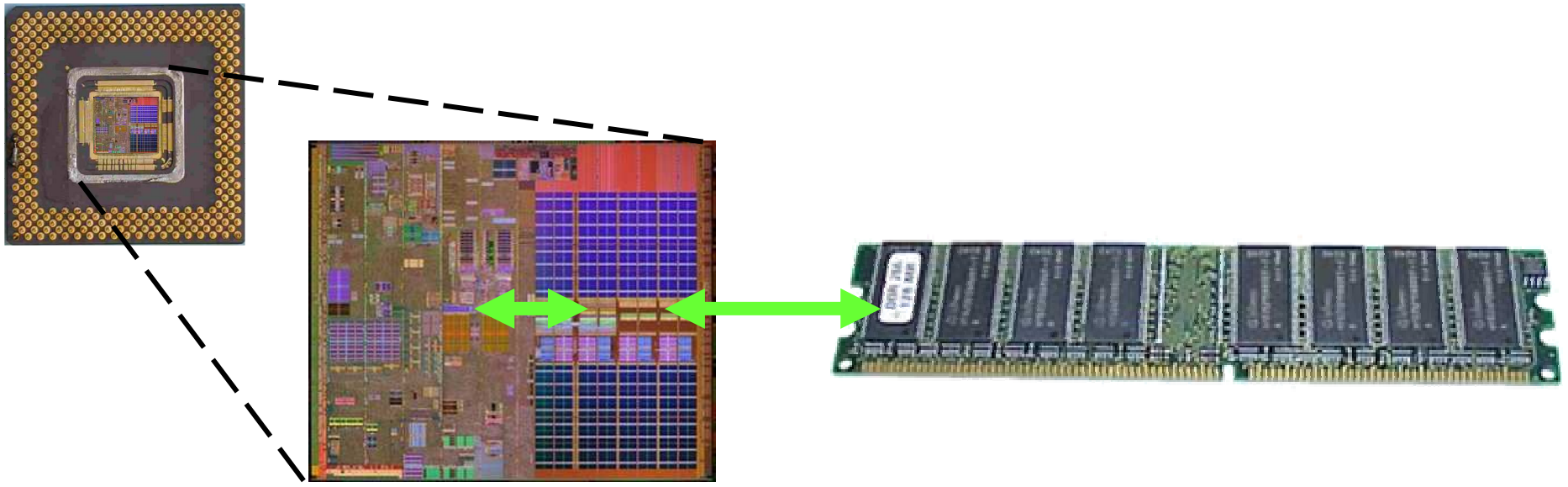
- Violate mandatory information control (e.g., Flume) [Krohn Tromer 09]

Cache attacks

Cache attacks

- Pure software
- No special privileges
- No interaction with the cryptographic code (some variants)
- Very efficient
  (e.g., full AES key extraction from Linux encrypted partition in 65 milliseconds)
- Compromise otherwise well-secured systems
  (e.g., see NIST AES process)
- “Commoditize” side-channel attacks
  easily deployed software breaks many common systems

Why cache analysis?

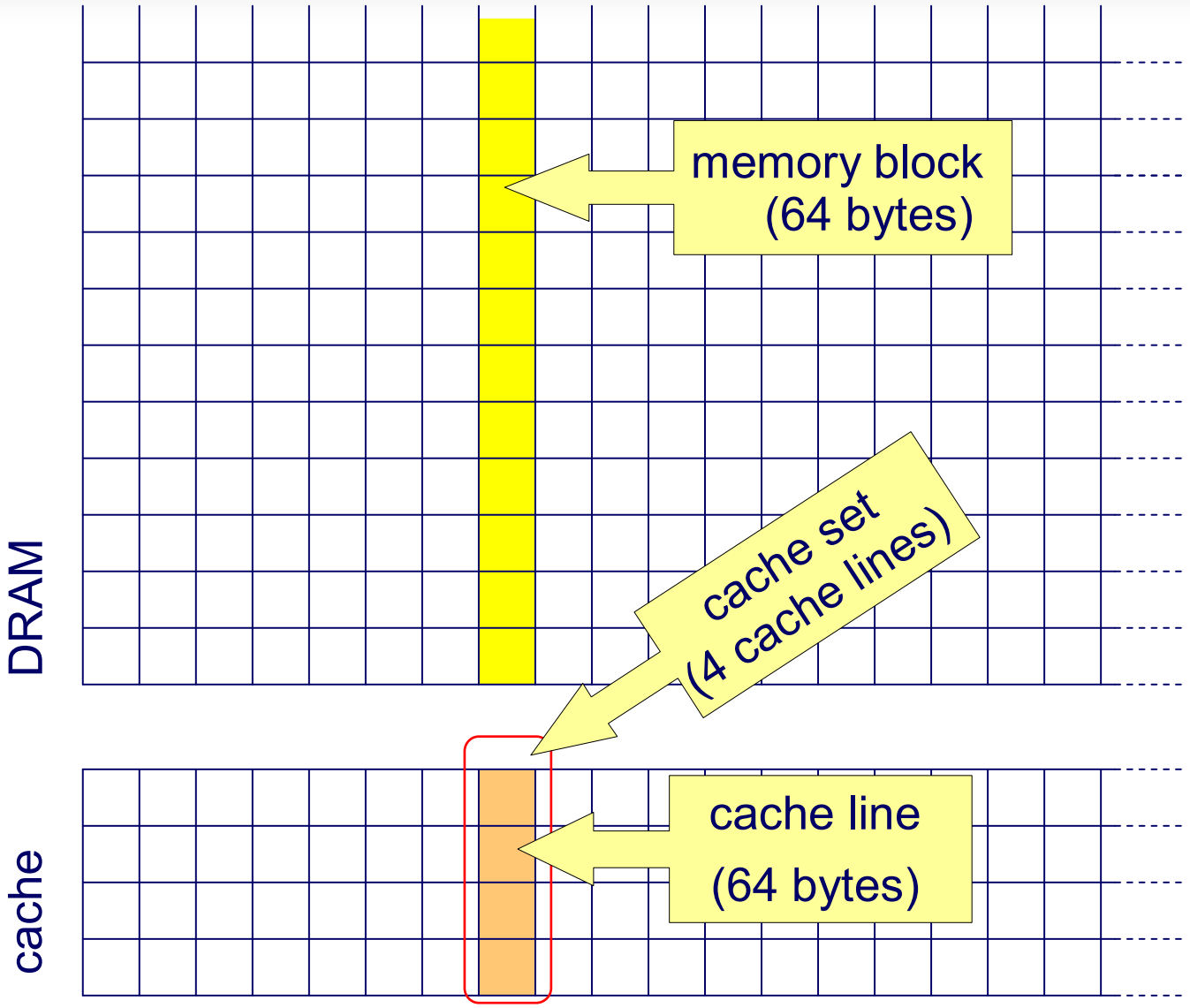


	CPU core	cache	Main memory
Annual speed increase:	60% (until recently)		7-9%
Typical latency:		0.3ns	50-150ns ⟶ **timing gap**

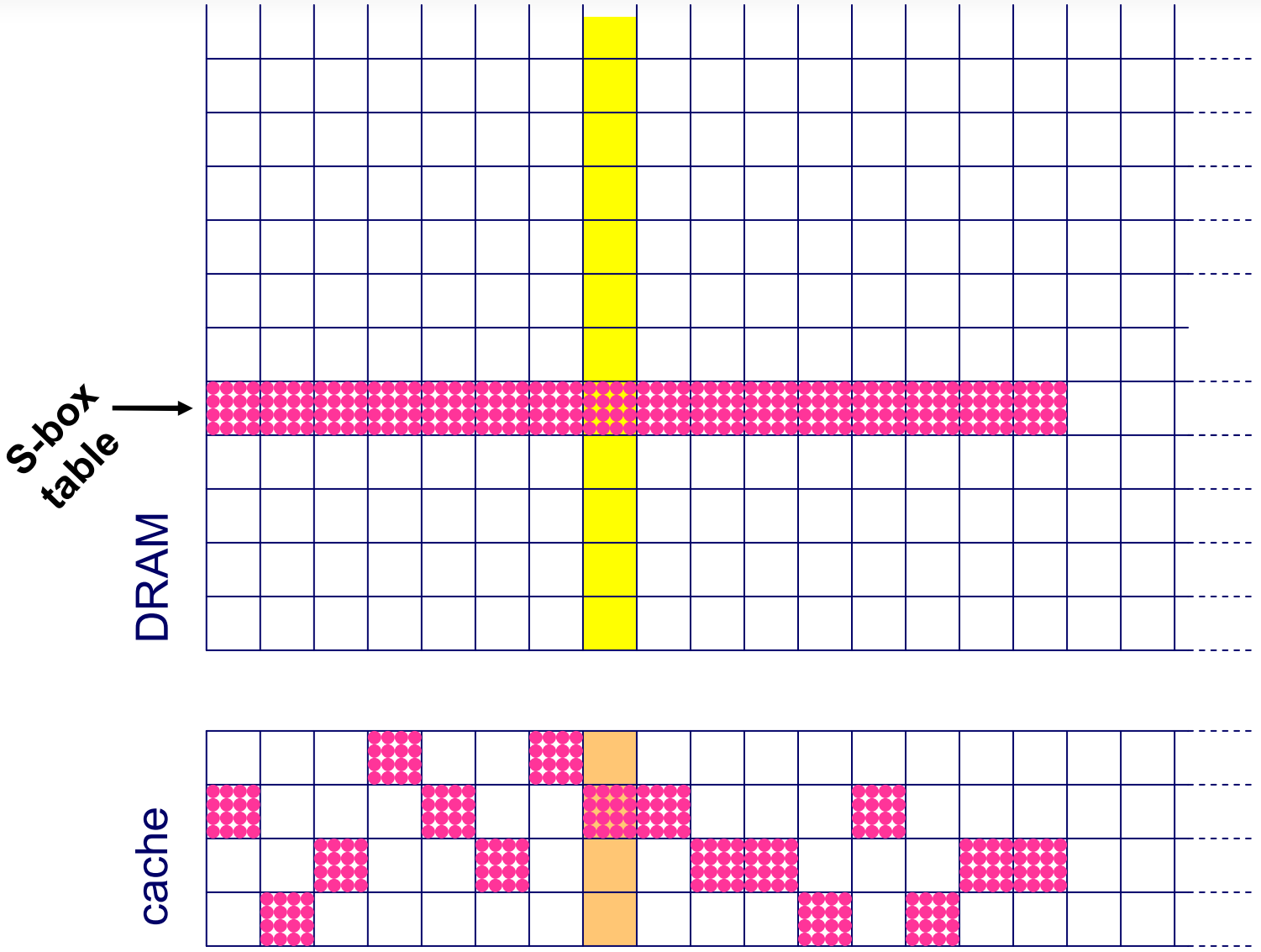
Address leakage

- The cache is a shared resource:
  cache state affects, and is affected by, all processes,
  leading to crosstalk between processes.
- The cached data is subject to memory protection…
- But the metadata leaks information about memory access patterns: which addresses are being accessed.

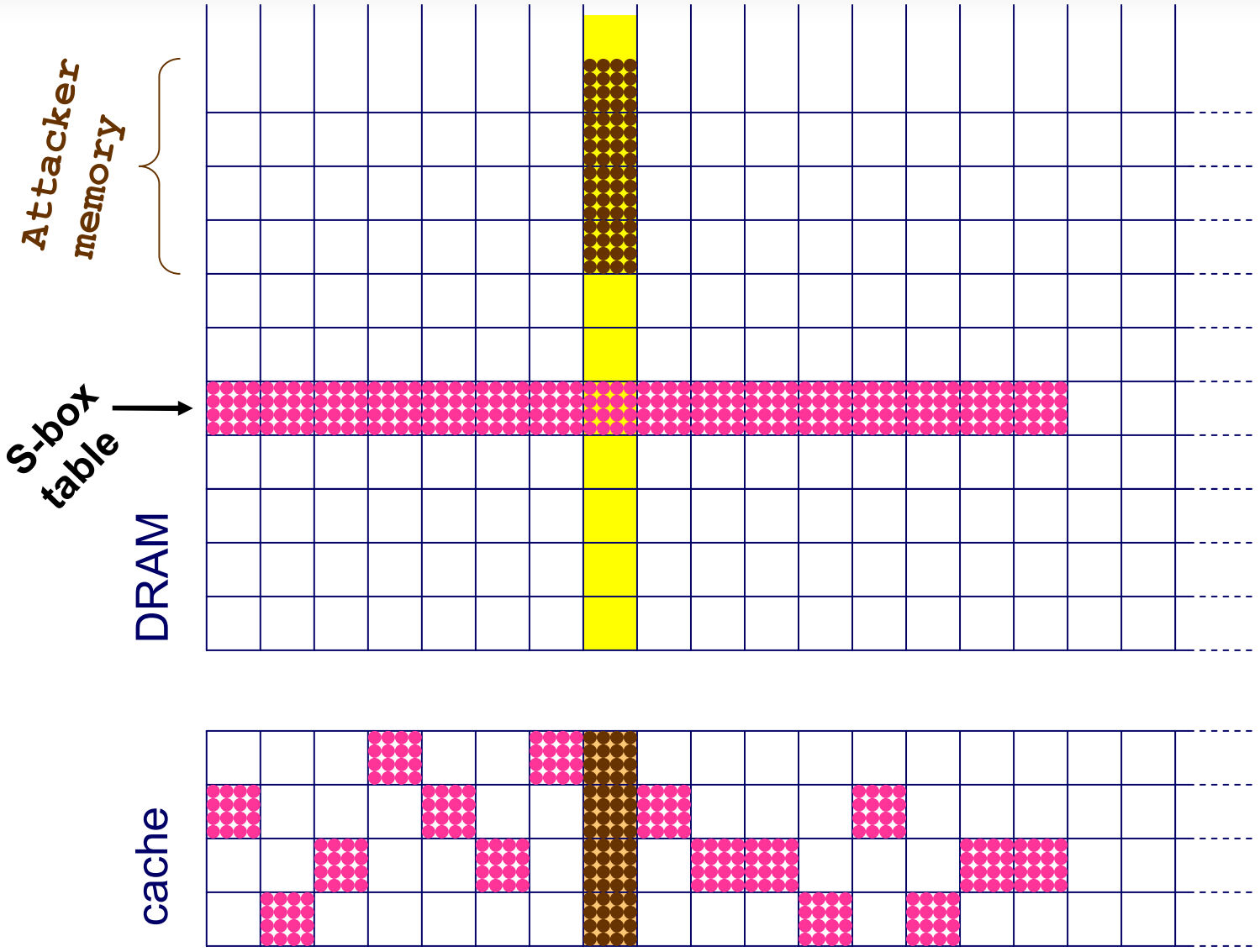
Associative memory cache



S-box tables in memory



Detecting access to AES tables (basic idea)



Measurement technique

Inter-process crosstalk can be exploited in two ways:

- Effect of the cache on the encryption (timing)

- Effect of the encryption on the cache

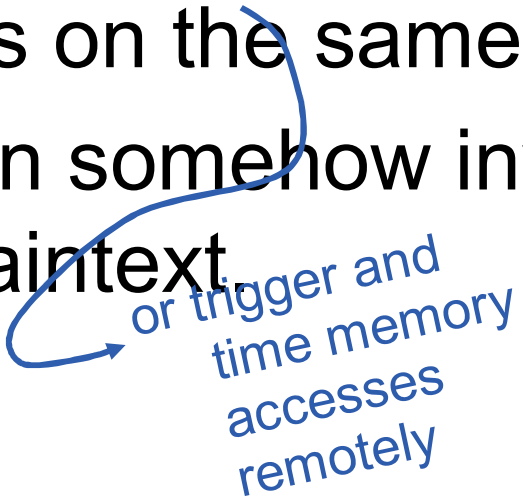
A typical software implementation of AES

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

...

/* 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]];
```

lookup index = plaintext ⊕ key

Scenario 1: Synchronous attack

- A software service performs AES encryption using a secret key.
- An attacker process runs on the same CPU.
- The attacker process can somehow invoke the service on known plaintext.



- Examples:
  - Encrypted disk partition + filesystem
  - IP/Sec, VPN

  → the attacker can discover the secret key.

Synchronous attack on AES

- Measure (possibly noisy) cache usage of many encryptions of known plaintexts.
- Guess the first key byte. For each hypothesis:
  - For each sampled plaintext, predict which cache line is accessed by "`T0[p[ 0]⊕k[ 0]]`"
- Identify the hypothesis which yields maximal correlation between predictions and measurements.
- Proceed for the rest of the key bytes.
- Practically, a few hundred samples suffice.

Got 64 bits of the key (high nibble of each byte)!

- Use these partial results to mount attack further AES rounds, exploiting S-box nonlinearity.

A few thousand samples for complete key recovery.

Experimental results [Osvik Shamir Tromer 05]

- Synchronous attack on OpenSLL AES encryption library call:

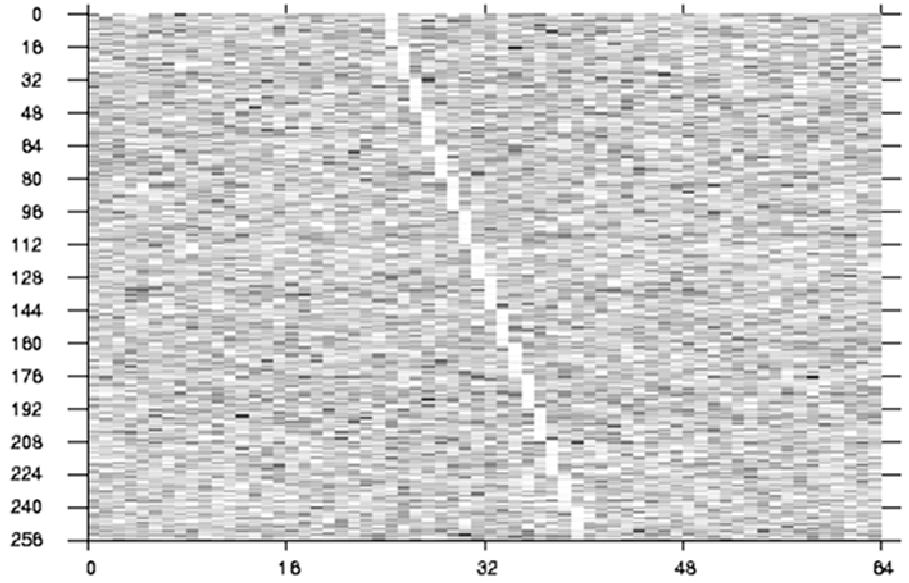
  Full key recovery by analyzing 300 encryptions (13ms)

- Synchronous attack on an AES encrypted filesystem (Linux **`dm-crypt`**):

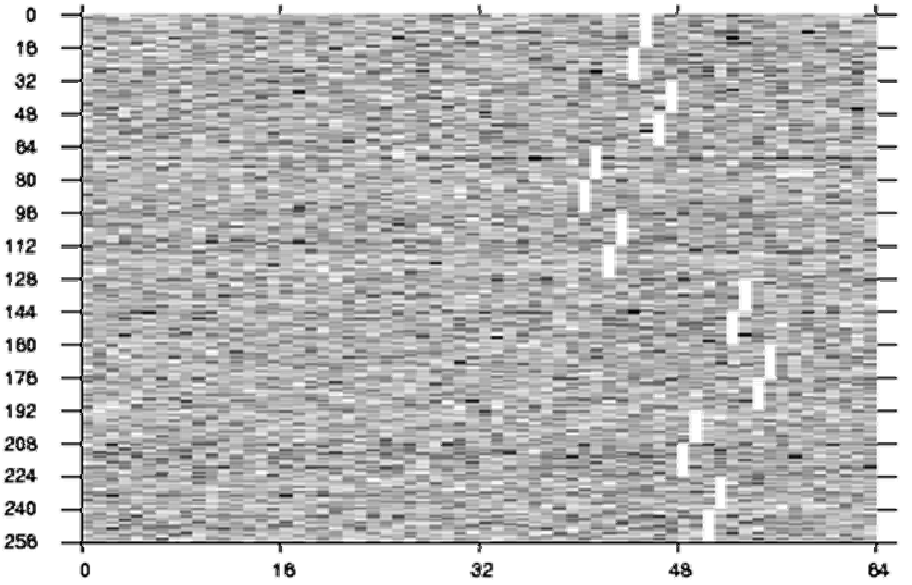
  Full key recovery by analyzing 800 write operations (65ms)

Experimental example: synchronous attack I

`k[0]=0x00`

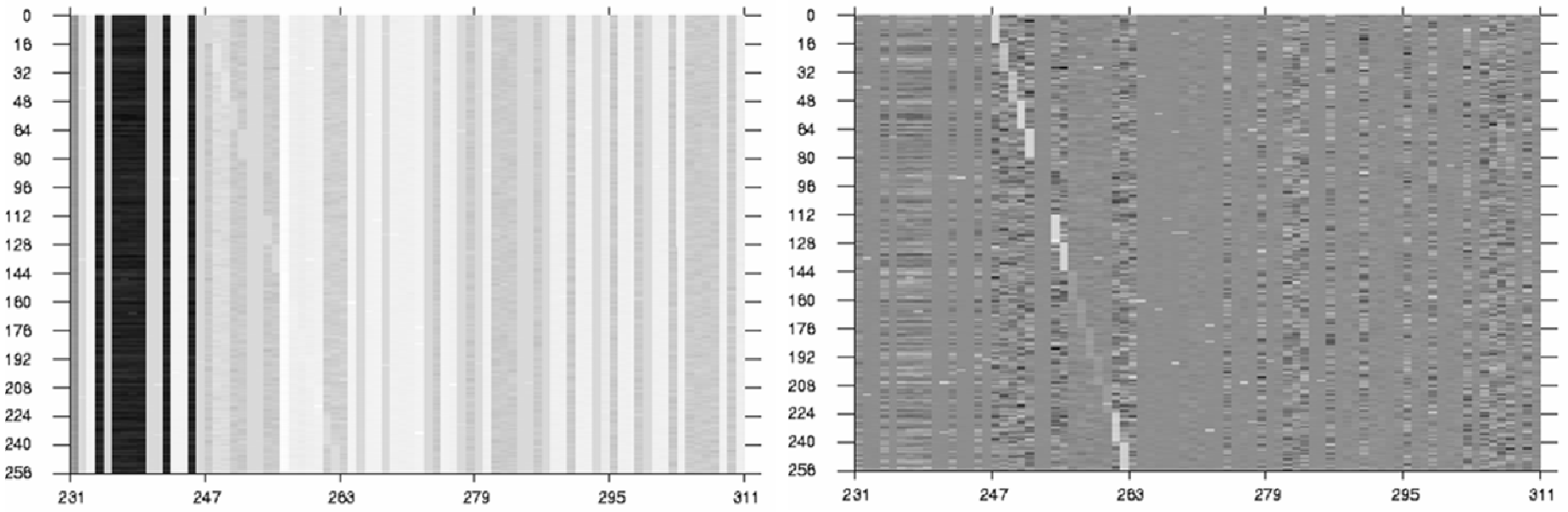


`k[5]=0x50`



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)

Experimental example: synchronous attack II



Measuring a Linux 2.6.11 dm-crypt encrypted filesystem with ECB AES on Athlon 64, using 30,000 samples.
Horizontal axis: evicted cache set
Vertical axis: `p[0]`
Brightness: cache probing time (normalized)

Left: raw. Right: after subtracting cache set average.

Scenario 2: asynchronous attack

- Someone runs encryptions computations using a secret key.
- An attacker process runs on the same CPU at (roughly) the same time.
- The plaintext/ciphertext has a non-uniform (conditional) distribution:
  - English
  - Formatted data
  - Headers
  - Ciphertext gleaned from wire
- Examples: just about any use of crypto on a multi-user system

→ attacker can (partially?) discover the secret key

Asynchronous attack (basic idea)

Compare two distributions:

- Measured memory accesses statistics.
- Predicted memory accesses statistics, under the given plaintext distribution and the key hypothesis.

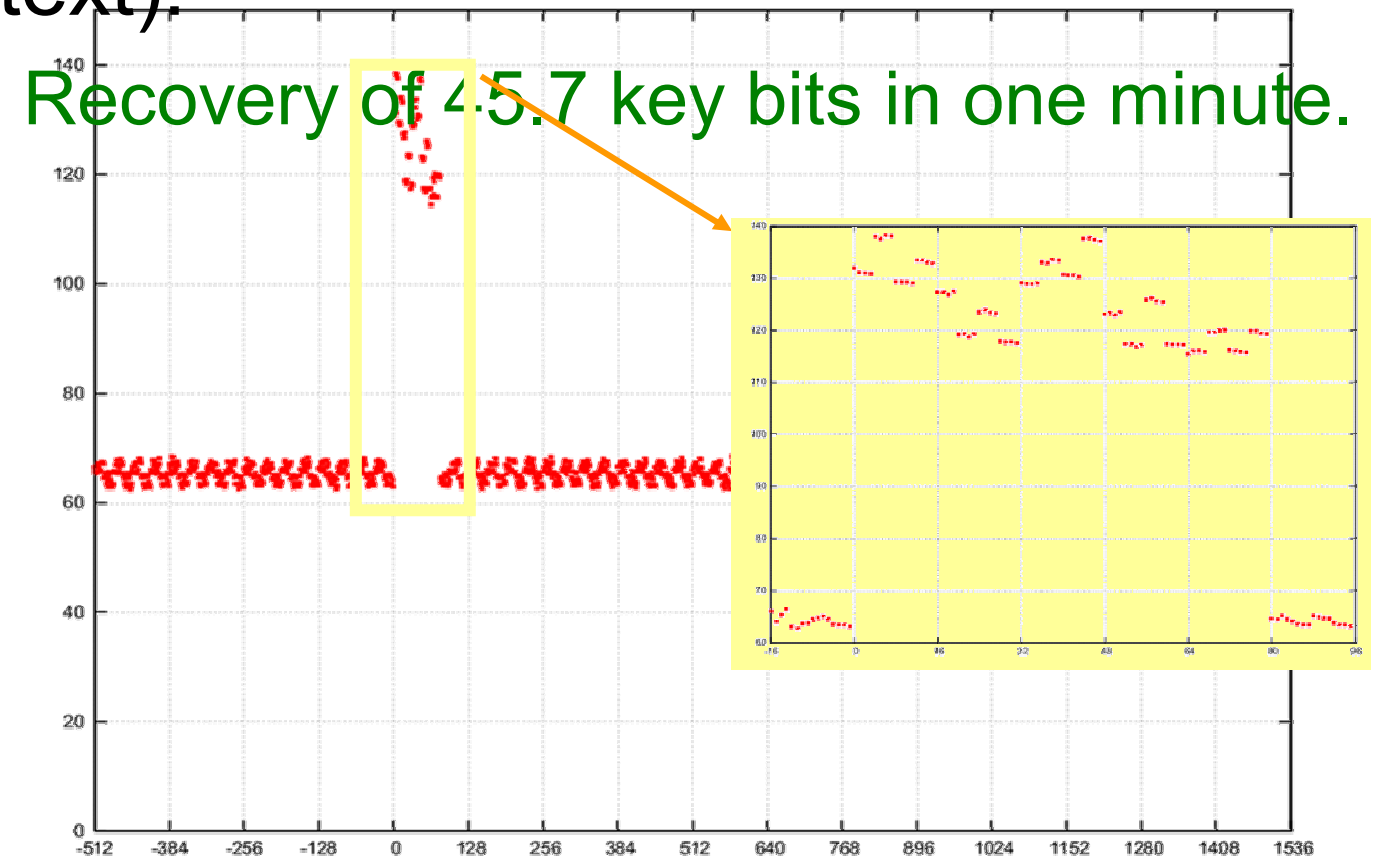
Find key that yields best correlation.

“Hyper Attack”

- Obtaining parallelism:
  - **Hyper**Threading (simultaneous multithreading)
  - Multi-core, shared caches, cache coherence
  - Interrupts
- Attack model:
  - 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]

- Asynchronous attack on AES (independent process doing batch encryption of text):

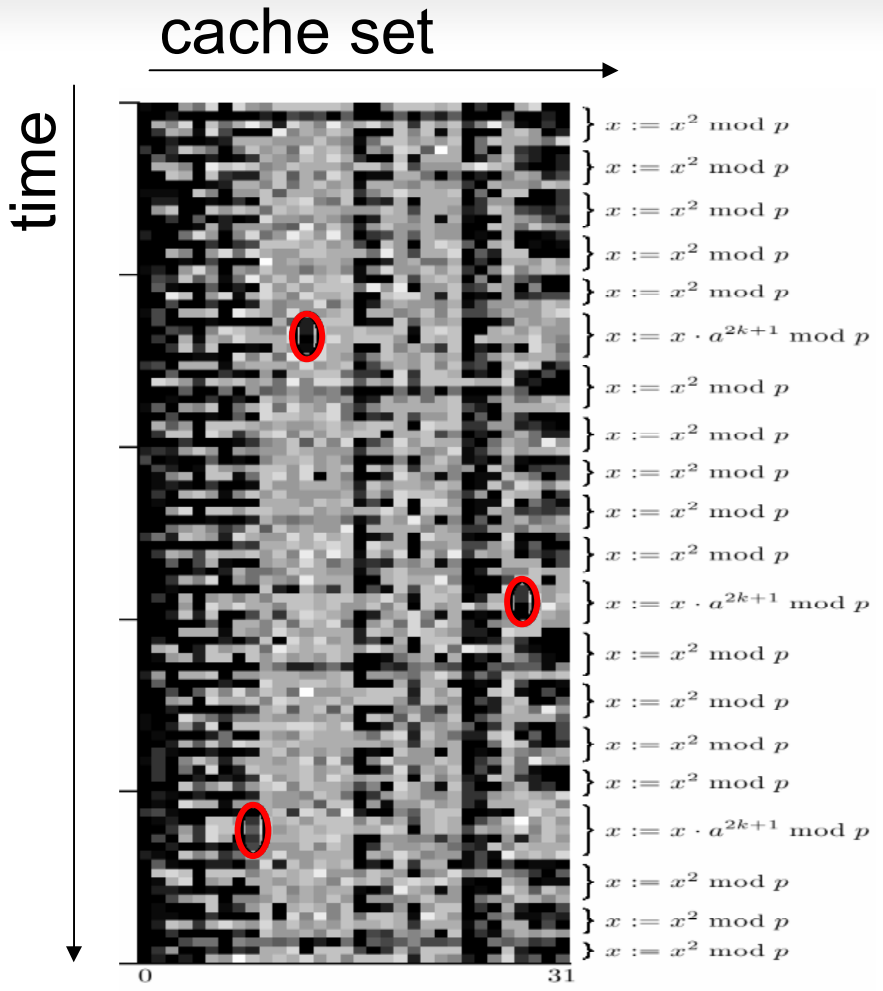


More attacks!

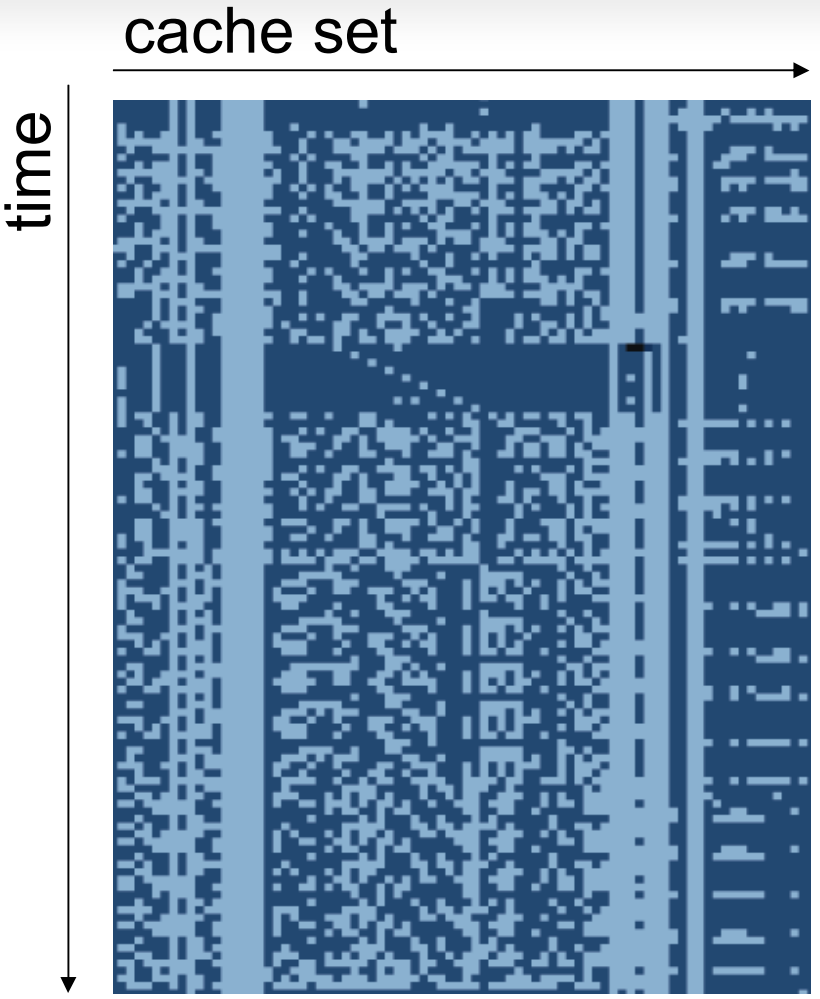
Other cache attacks

- Convert channels [Hu '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]

High-temporal-resolution cache monitoring



Secret key read from RSA decryption (OpenSSL library) using HyperThreading [Per05]



AES in another process, (4 keys, 5 runs each) using scheduler [NS07]

Other architectural attacks

Induced by contention for shared resources:

- Data cache [Hu91][Bernstein05][Osvik ShamirTromer 05] [Percival05]
- 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

Implications

Implications and Extensions

- Multiuser systems
- Virtual machines
  - Examples: `jail()`, Xen, UML, VMware, Virtual PC
  - Breaks NSA US Patent 6,922,774
  - No guarantees in AMD ”Secure Virtual Machine” or Intel “Virtualization Technology” (VMX) specs either
- DRM
  The trusted path is leaky (even if verified by TPM attestation, NGSCB, etc.).
- Untrusted code
  (e.g., ActiveX, Java applets, managed .NET, JavaScript)
- Remote network attacks

Cloud Computing

raining on the parade

The virtualized world

Virtualization is increasingly popular:

- Cost reduction
- Management
- Compatibility

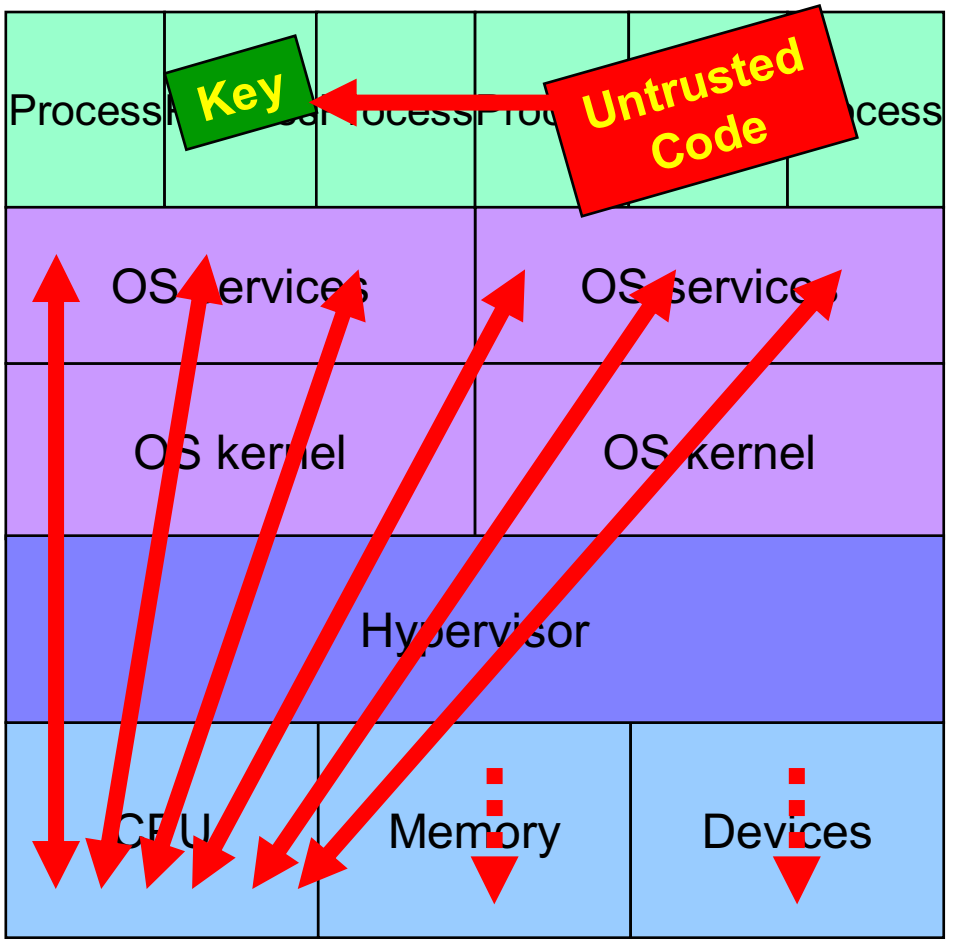
Also touted for its security benefits:

- Isolation
- Sandboxing

But many side-channel attacks are oblivious to virtualization – it's the same underlying hardware!

Gartner: "*through 2009, 60% of production VMs will be less secure than their physical counterparts.*"

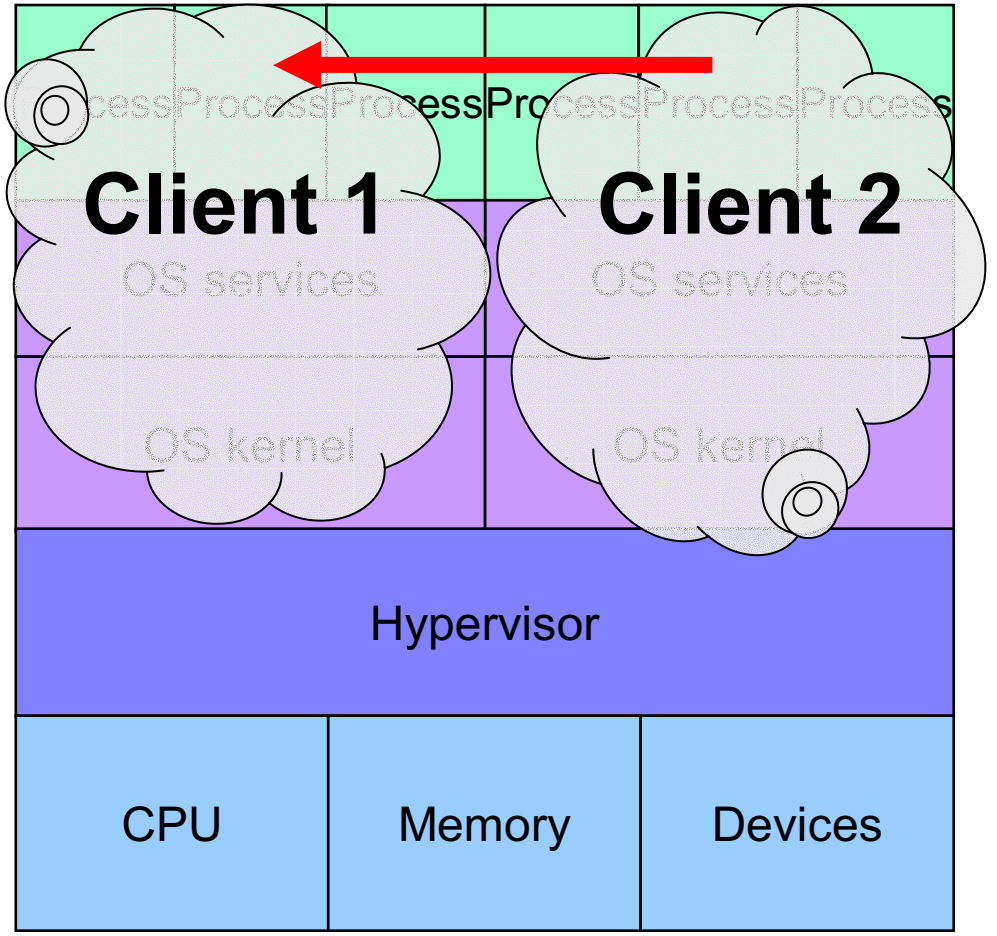
US patent 6,922,774 (NSA)



Vulnerability of public “Cloud Computing”

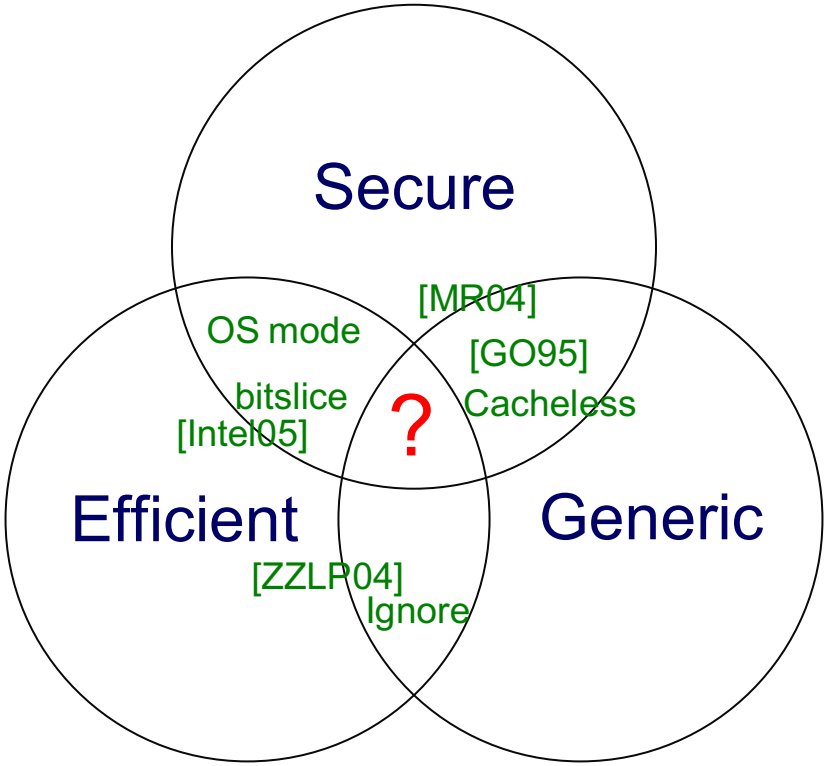
[Ristenpart Tromer Shacham Savage 09]

[details omitted from online version]



Countermeasures

Solutions


Secure
[MR04]
OS mode
[GO95]
bitslice
[Intel05]
?
Cacheless
Efficient
Generic
[ZZLP04]
Ignore

Hardware / platform

- Lock down the cache
  - Performance
  - Manual
- Randomize the address-to-cacheline mapping [Wang Lee 08]
- Secure mode with guarantees semantics
- Add AES instructions to new Pentium chips

General cryptographic transformations

- ~~Fully homomorphic encryption~~
- Obfuscation
- Oblivious RAM

[Goldreich 87][Goldreich Ostrovsky 95]

Protecting specific functionality

- Pick the right primitives (Rijndael vs. Serpent)
- AES using bitslicing
- Memory-obllivious modular exponentiation (OpenSSL)
- Moni's upcoming talk

Models

Model for cache attacks

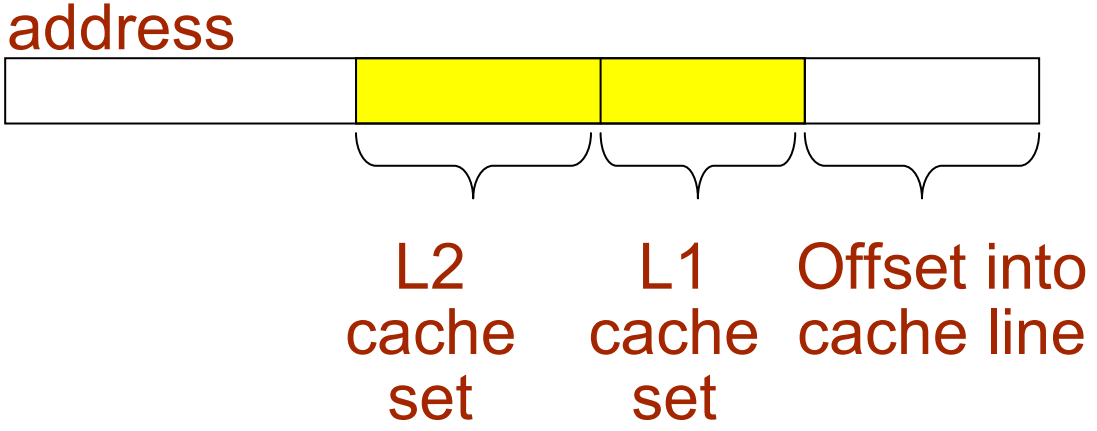
- Full address transcript: (time,address)
  - Analogous to “program counter model”

    [Molnar Piotrowski Schultz Wagner 2005]
- Too strong – requires full-fledged oblivious RAM
  - Logarithmic lower bound on overhead

    [Goldreich Ostrovsky 95]

Model for cache attacks

- Restrict which bits leak:



- Inaccurate (e.g., cache bank collisions in Athlon 64)
- Generalized:
  Suppose $(\text{time}, f(\text{address}))$ leaks, for adaptively chosen f;
  What classes of f can we handle?

Models for cache attacks

- Restrict temporal resolution
- In many settings, attacker can't individual observe every element in the victim's address transcript.
- Model: $(f(\text{time}),\text{address})$
- But:
  - HyperThreading attacks [Osvik Shamir Tromer 05][Percival 06]
  - Scheduler attacks [Neve Seifert 06]
  - Cache state captures temporal dependencies

More models

- "High-level" models
  - Bounded leakage amount
  - Computationally-bounded leakage
  - Other recent machinery from physical circuits
- Other architectural attacks

Open problems

- Find good models (clean, realistic and useful)
- Fix the hardware
- More leaky-cache-resilient primitives
- Transform existing implementations at the system level [work in progress]
- Cryptographic program transformation, generalizing Oblivious RAM

Thanks!