More Virtualization

CS642: Computer Security

Topics

- Reset/Randomization problems
- Side channels
- Leaked secrets

What is different about virtual machines

- New operations not formerly possible
  - Snapshot/restore same state *multiple times*
- Changing assumptions
  - Randomness of interrupts
- Multi-tenancy
  - Sharing hardware with your enemies

Virtual Machine Management

- Snapshots
  - Volume snapshot / checkpoint
    - persistent storage of VM
    - must boot from storage when resuming snapshot
  - Full snapshot
    - persistent storage and ephemeral storage (memory, register states, caches, etc.)
    - start/resume in between (essentially) arbitrary instructions
- VM image is a file that stores a snapshot

Uses for Secure Random Numbers

Cryptography

- Keys
- Nonces, initial values (IVs), salts

System Security

- TCP Initial Sequence Numbers (ISNs)
- ASLR
- Stack Canaries

Where can we get secure random numbers?

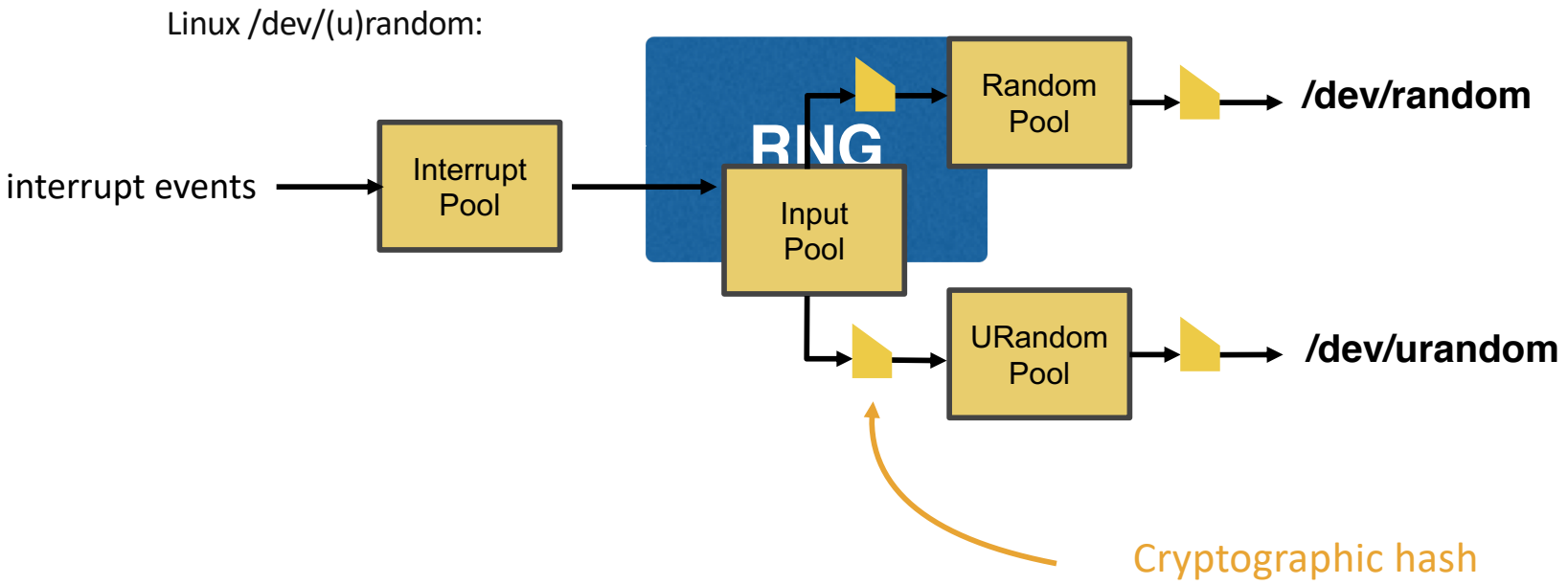
Every OS provides a high-quality RNG

OSX/Linux:

```
cat /dev/urandom
```

Operating System Random Number Generators

Linux RNG

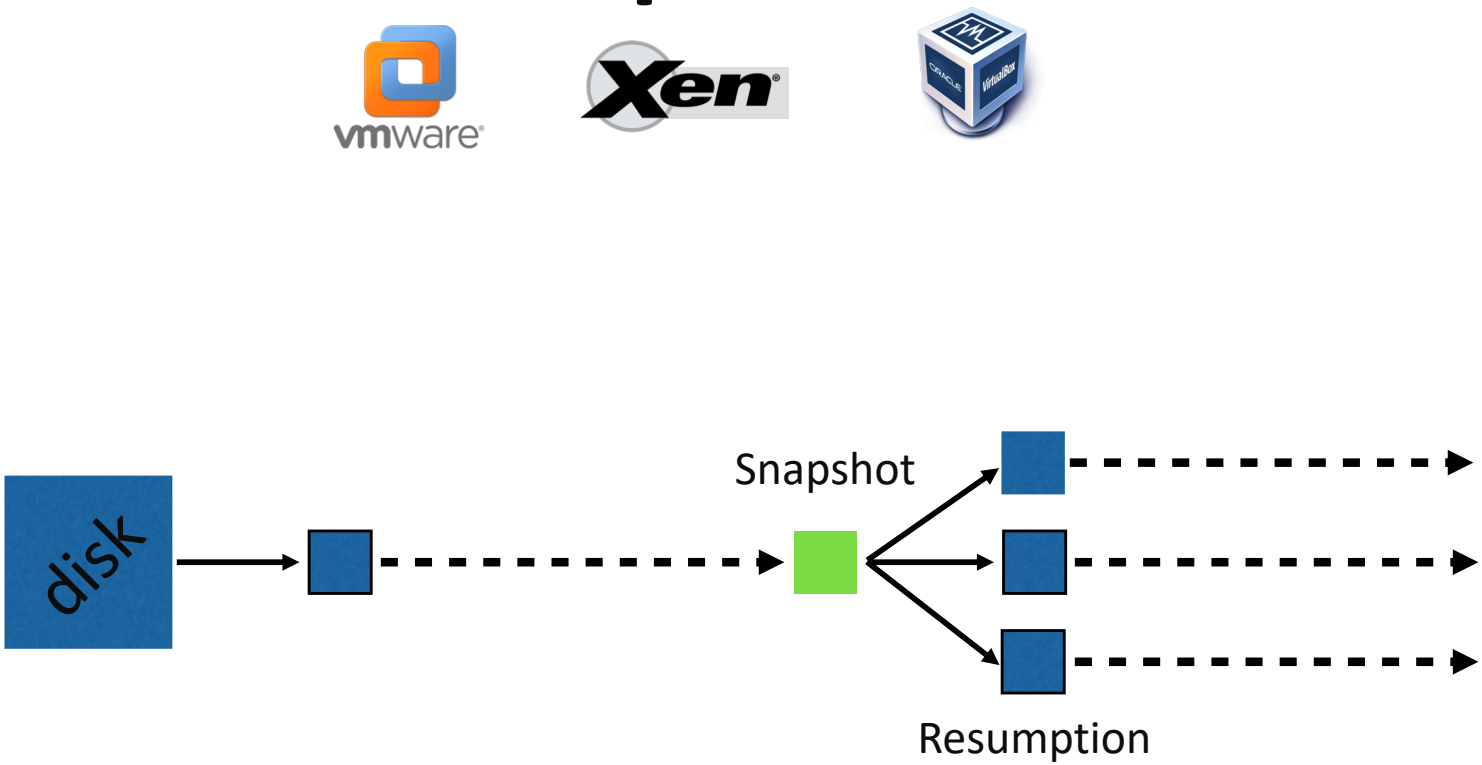

Linux /dev/(u)random:
interrupt events
Interrupt Pool
RNG
Input Pool
Random Pool
/dev/random
URandom Pool
/dev/urandom
Cryptographic hash

RNG Failures

RNG Failures
Predictable Output
Repeated Output
Outputs from a small range (not-statistically uniform)

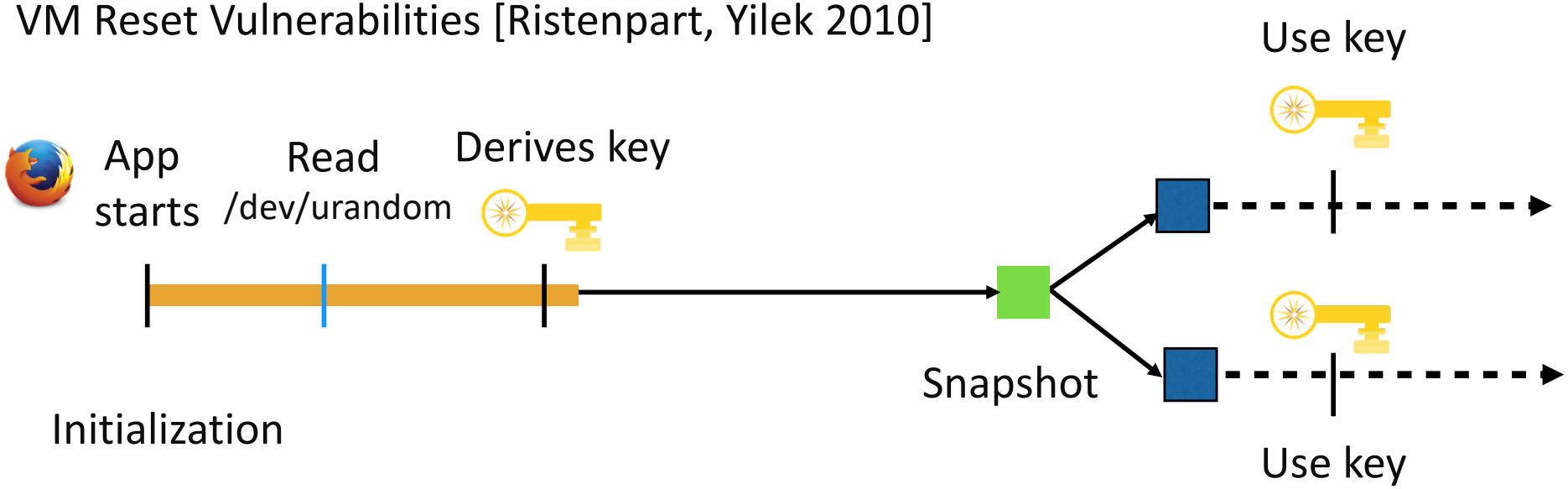
Broken Windows RNG: [DGP 2007]
Broken Linux RNG: [GPR 2008], [LRSV 2012], [DPRVW 2013], [EZJSR 2014]
Factorable RSA Keys: [HDWH 2012]
Taiwan National IDs: [BCCHLS 2013]

Virtual Machine Snapshots



Security Problems with VM Resets

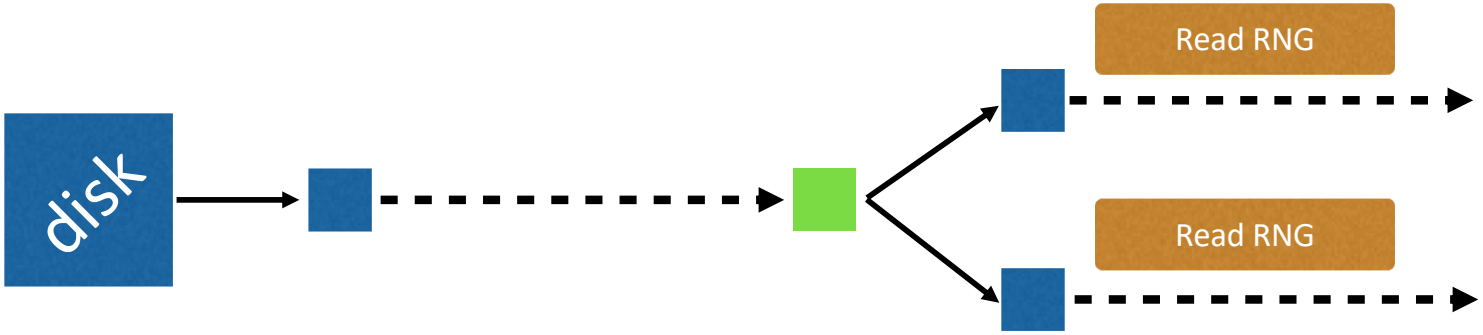
VM Reset Vulnerabilities [Ristenpart, Yilek 2010]



Firefox and Apache reused random values for TLS
Attacker can read previous TLS sessions, recover private keys from Apache

Linux RNG after VM Reset

Not-So-Random Numbers in Virtualized Linux
[Everspaugh, et al, 2014]



Experiment:

- Boot VM in Xen or VMware
- Capture snapshot
- Resume from snapshot, read from /dev/urandom

Repeat: 8 distinct snapshots
20 resumptions/snapshot

/dev/urandom outputs after resumption

Linux RNG is ***not*** reset secure:
7/8 snapshots produce mostly identical outputs

Reset 1		Reset 3
1E6DD331	1E6DD331	1E6DD331
8CC97112	8CC97112	8CC97112
2A2FA7DB	2A2FA7DB	2A2FA7DB
DBBF058C	DBBF058C	DBBF058C
26C334E7	26C334E7	26C334E7
F17D2D20	F17D2D20	45C78AE0
CC10232E	CC10232E	E678DBB2
...	...	...

Reset insecurity and applications

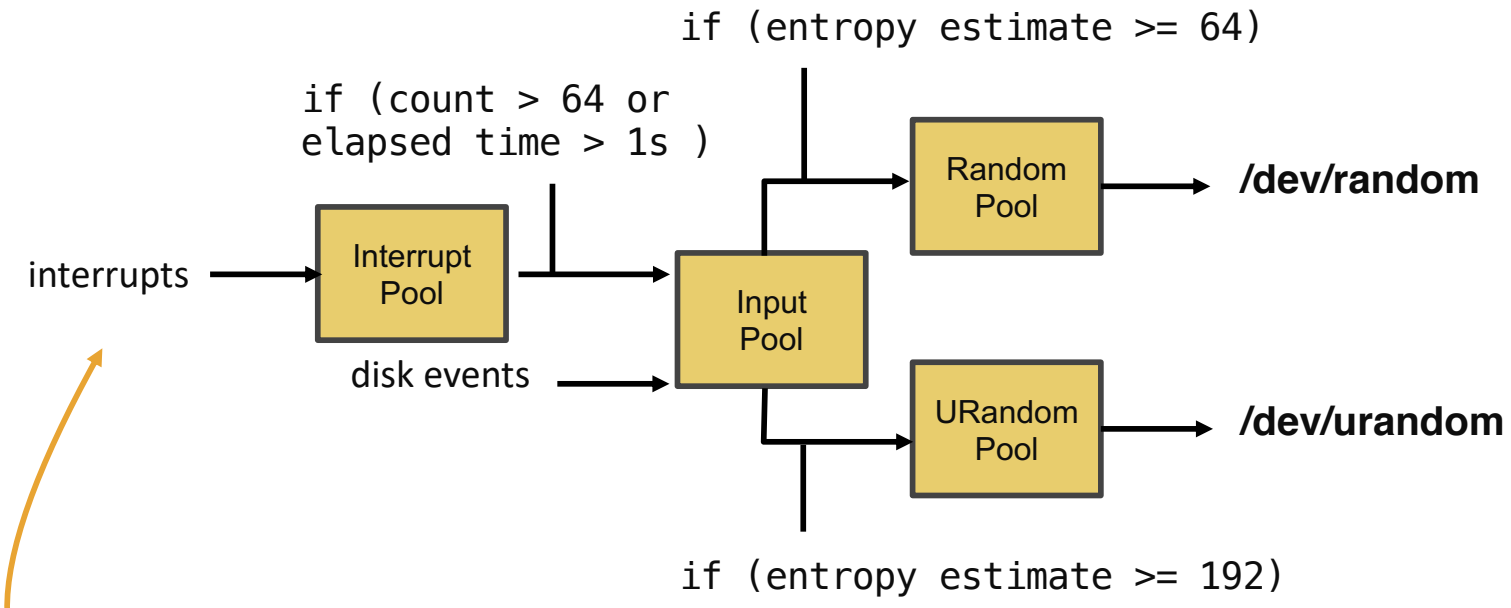
Generate RSA key on resumption:

```
openssl genrsa
```

30 snapshots; 2 resets/snapshot (ASLR Off)

- 27 trials produced identical private keys
- 3 trials produced unique private keys

Why does this happen?



Buffering and thresholds prevent new inputs from impacting outputs

Linux /dev/(u)random

What about other platforms?

FreeBSD

/dev/random produces identical output stream
Up to 100 seconds after resumption

Microsoft Windows 7

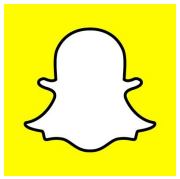
Produces repeated outputs indefinitely

`rand_s`	(stdlib)
`CryptGenRandom`	(Win32)
`RngCryptoServices`	(.NET)

Cloud computing

Cloud providers

Popular customers



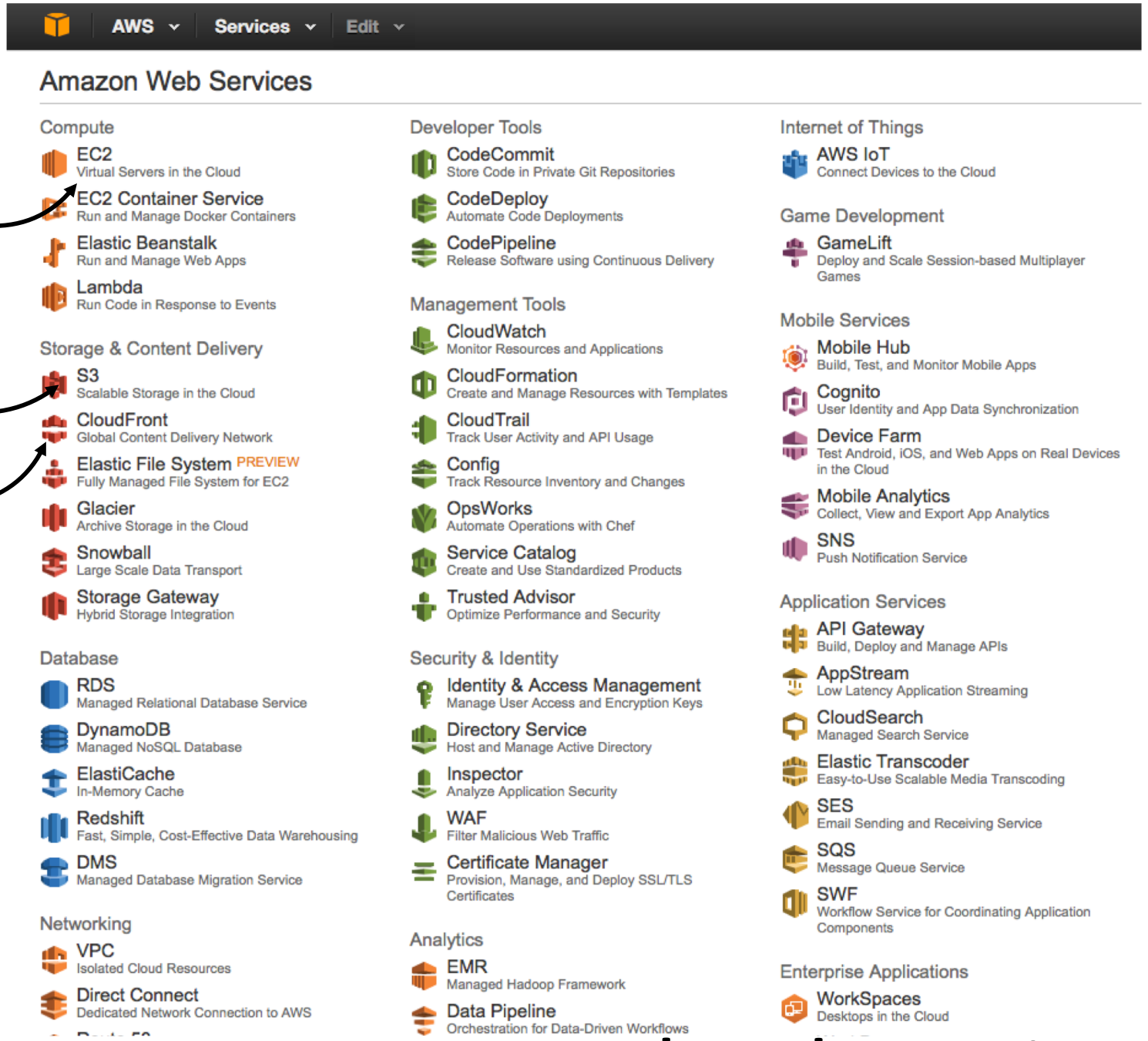
Who can be a customer?

We call these "public clouds"

VMs
Infrastructure-as-a-service

Storage

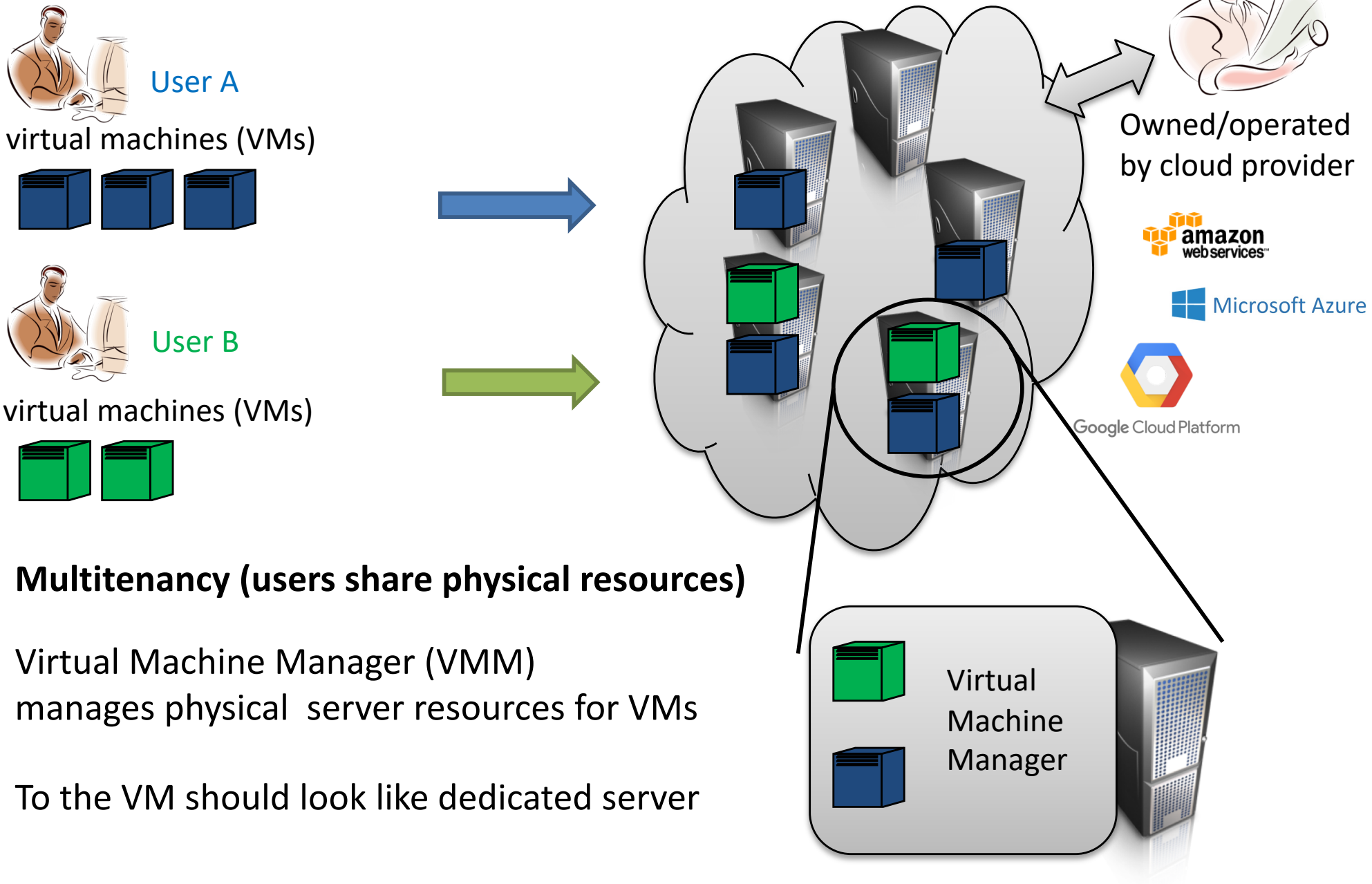
Web Cache/TLS Termination



Cloud Services

A simplified model of public cloud computing

Users run Virtual Machines (VMs) on cloud provider's infrastructure

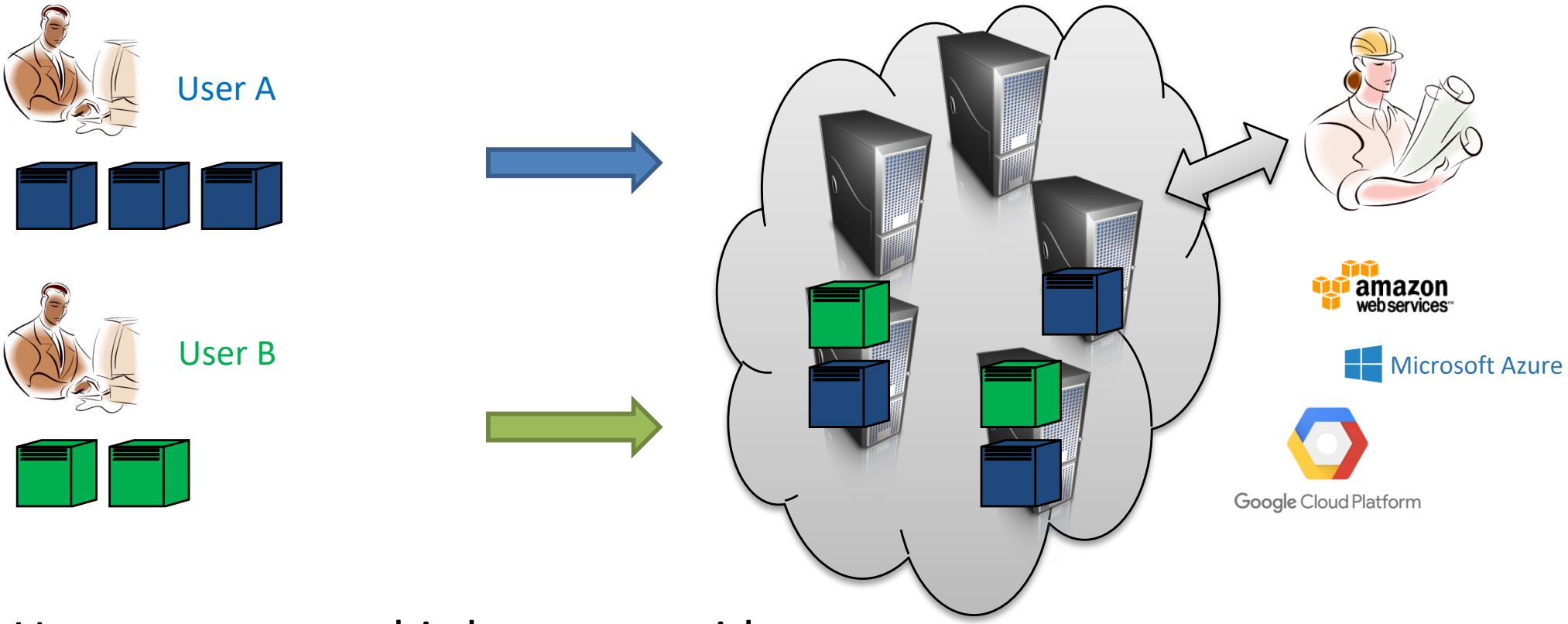


Multitenancy (users share physical resources)

Virtual Machine Manager (VMM)
manages physical server resources for VMs

To the VM should look like dedicated server

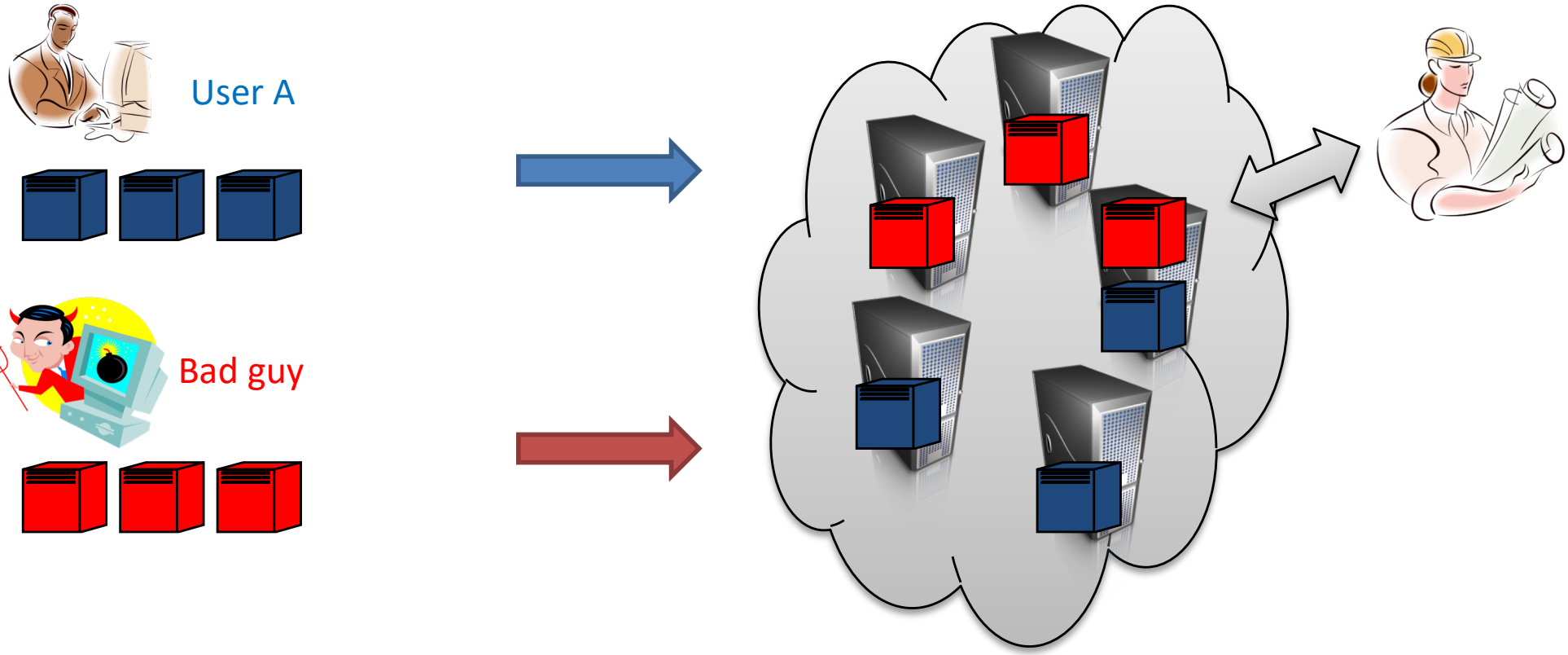
Trust models in public cloud computing



Users must trust third-party provider to

- not spy on running VMs / data
- secure infrastructure from external attackers
- secure infrastructure from internal attackers

A new threat model:



Attacker identifies one or more victims VMs in cloud

1) Achieve advantageous placement via launching of VM instances

2) Launch attacks using physical proximity

Exploit VMM vulnerability DoS Side-channel attack

Anatomy of attack

Checking for co-residence

check that VM is on same server as target
- network-based co-residence checks
- efficacy confirmed by covert channels

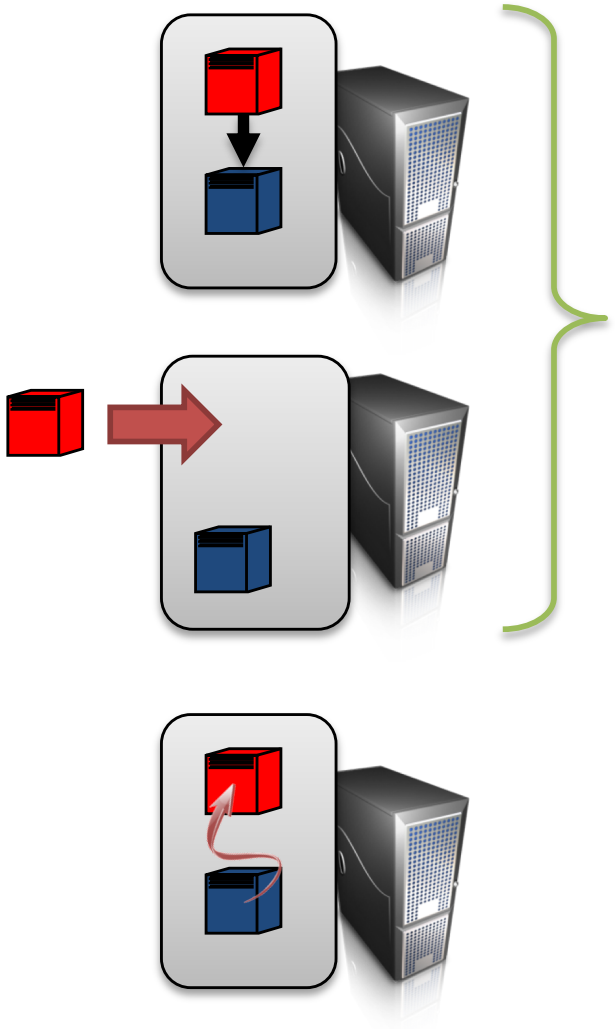
Achieving co-residence

brute forcing placement

instance flooding after target launches

Location-based attacks

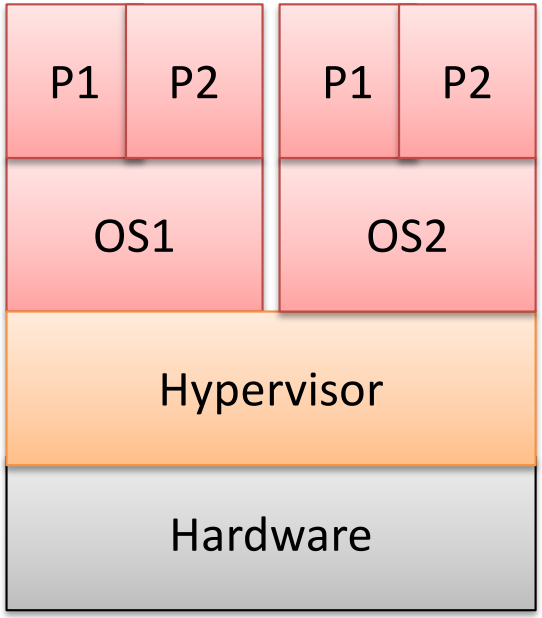
side-channels, DoS, escape-from-VM



Placement vulnerability: attackers can knowingly achieve co-residence with target

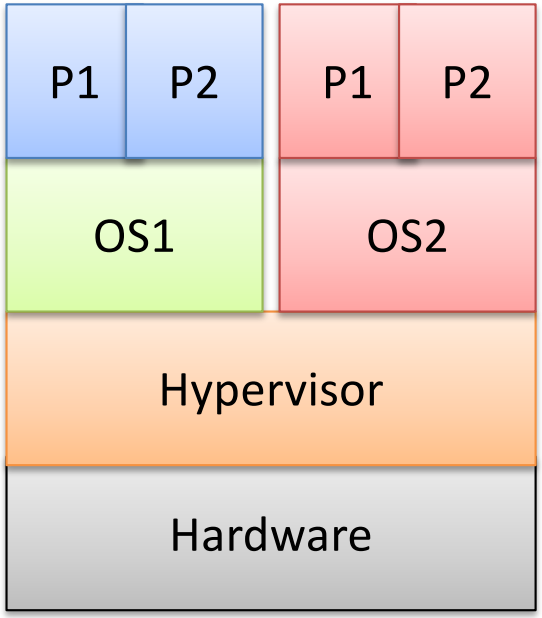
Violating isolation

- Covert channels between VMs circumvent access controls
  - Bugs in VMM
  - Side-effects of resource usage



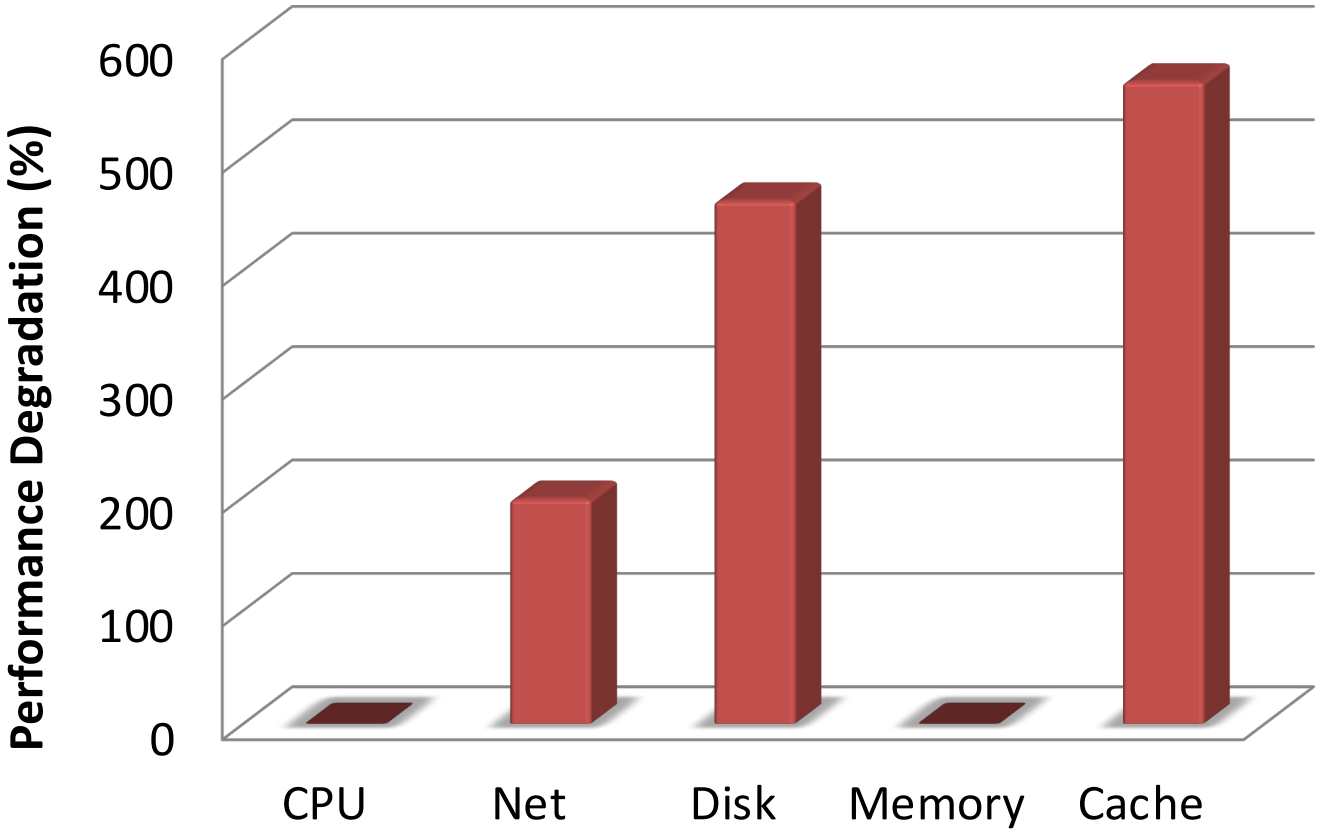
Violating isolation

- Covert channels between VMs circumvent access controls
  - Bugs in VMM
  - Side-effects of resource usage
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Xen scheduler vulnerability



Measuring Resource Contention

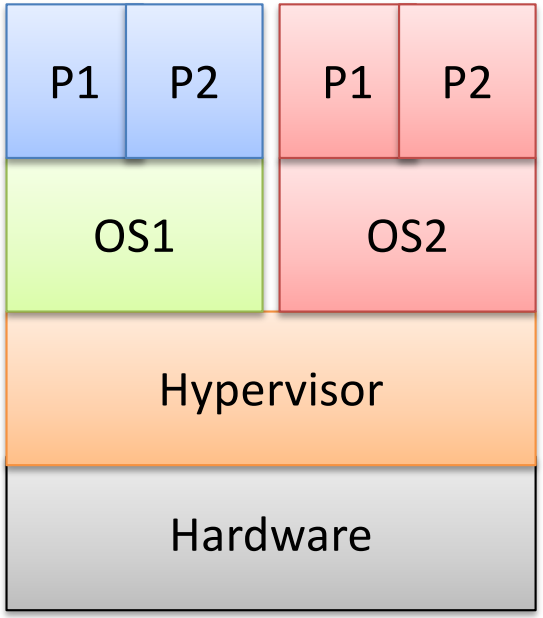
- Contention for the same resource



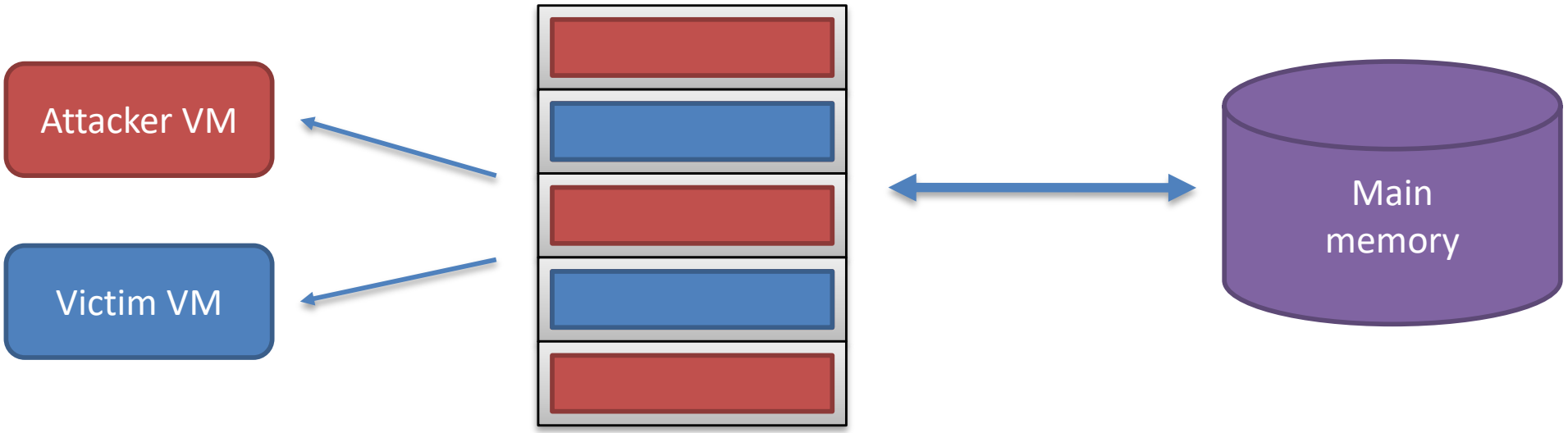
Local Xen Testbed	
Machine	Intel Xeon E5430, 2.66 Ghz
Packages	2, 2 cores per package
LLC Size	6MB per package

Violating isolation

- Covert channels between VMs circumvent access controls
  - Bugs in VMM
  - Side-effects of resource usage
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Xen scheduler vulnerability
- Side channels
  - Spy on other guest via shared resources



Cross-VM side channels using CPU cache contention

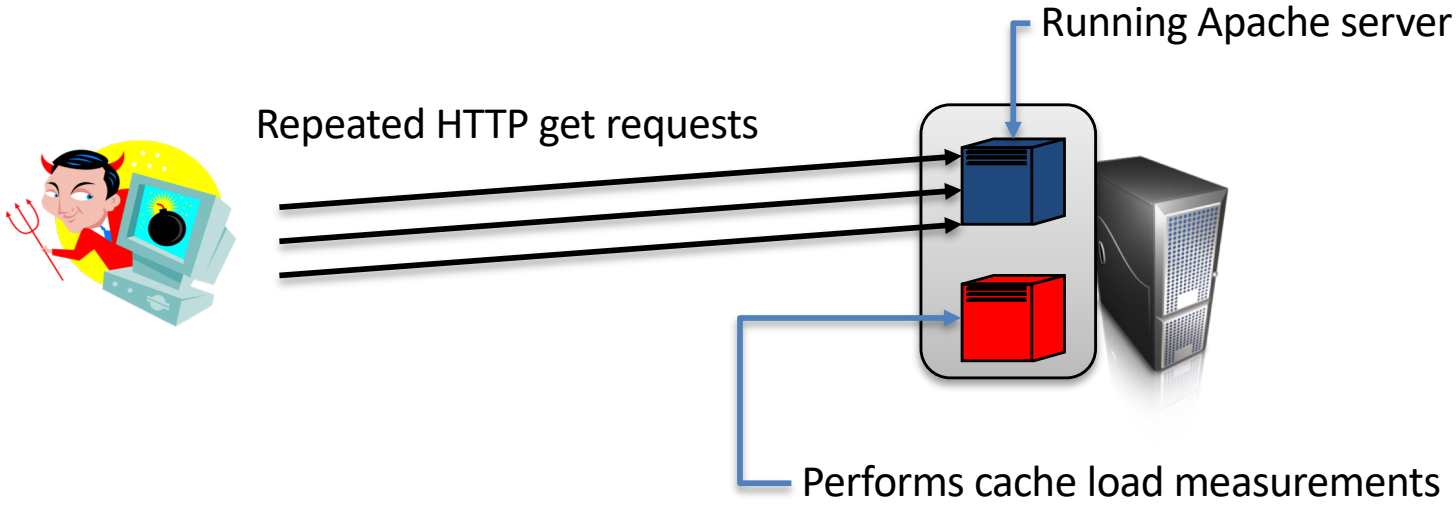


1) Read in a large array (fill CPU cache with attacker data)

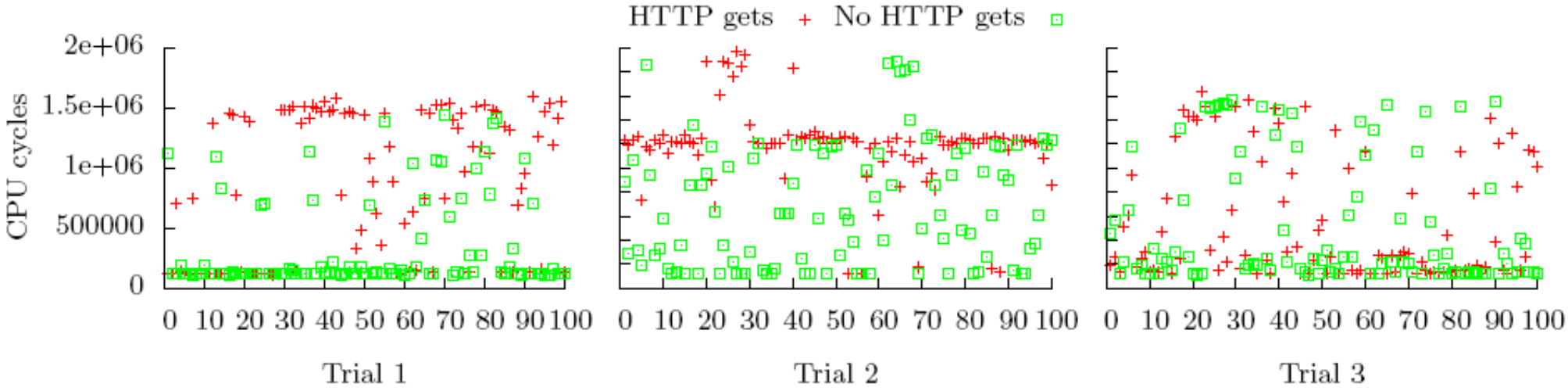
2) Busy loop (allow victim to run)

3) Measure time to read large array (the load measurement)

Cache-based cross-VM load measurement on EC2



3 pairs of instances, 2 pairs co-resident and 1 not
100 cache load measurements during **HTTP gets** (1024 byte page) and with **no HTTP gets**



[*Hey, You, Get Off of my Cloud*, 2009, Ristenpart, et al.]

Square-and-Multiply

/* $y = x^e \bmod N$, from **libgcrypt***/

Modular Exponentiation (x, e, N):

let $e_n \ldots e_1$ be the bits of e

$y \leftarrow 1$

for e_i in $\{e_n \ldots e_1\}$

$y \leftarrow$ **Square**(y) **(S)**

$y \leftarrow$ **Reduce**(y, N) **(R)**

if $e_i = 1$ then

$y \leftarrow$ **Multi**(y, x) **(M)**

$y \leftarrow$ **Reduce**(y, N) **(R)**

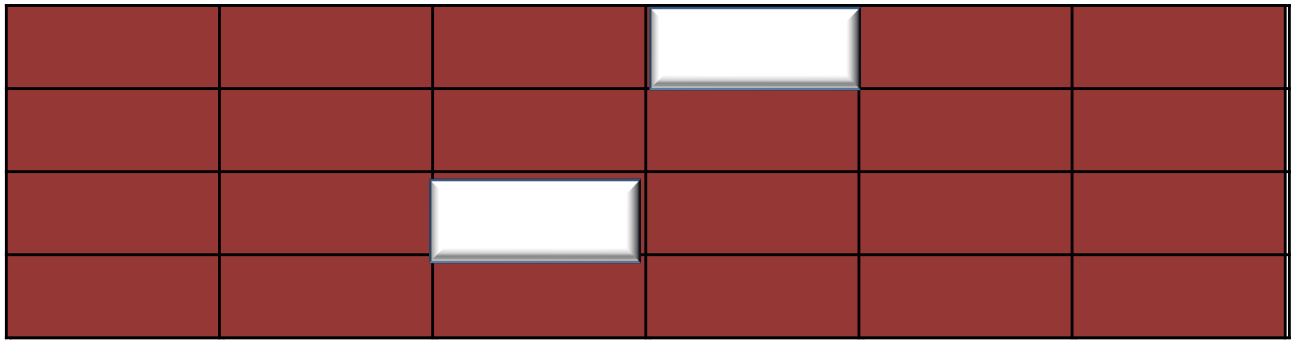
$e_i = 1 \rightarrow$ SRMR

$e_i = 0 \rightarrow$ SR

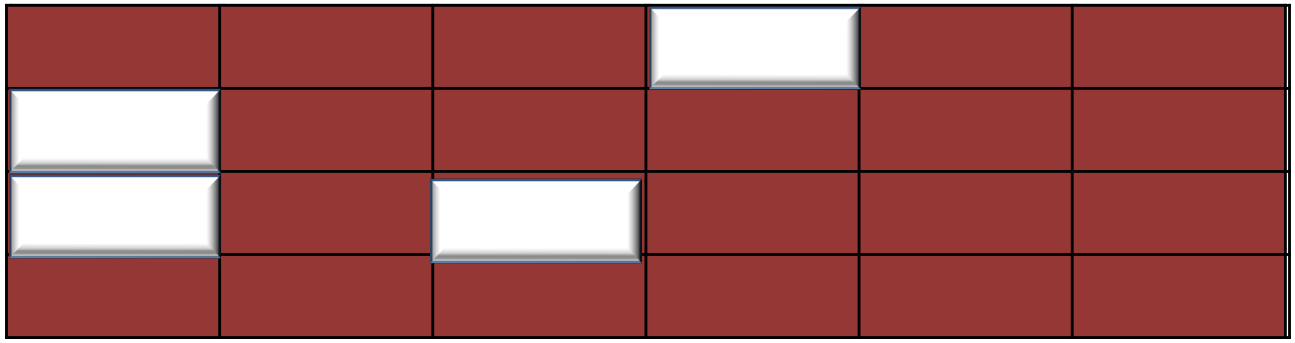
Control flow (sequence of instructions used) leaks secret

Detecting code path

$e_i = 0$



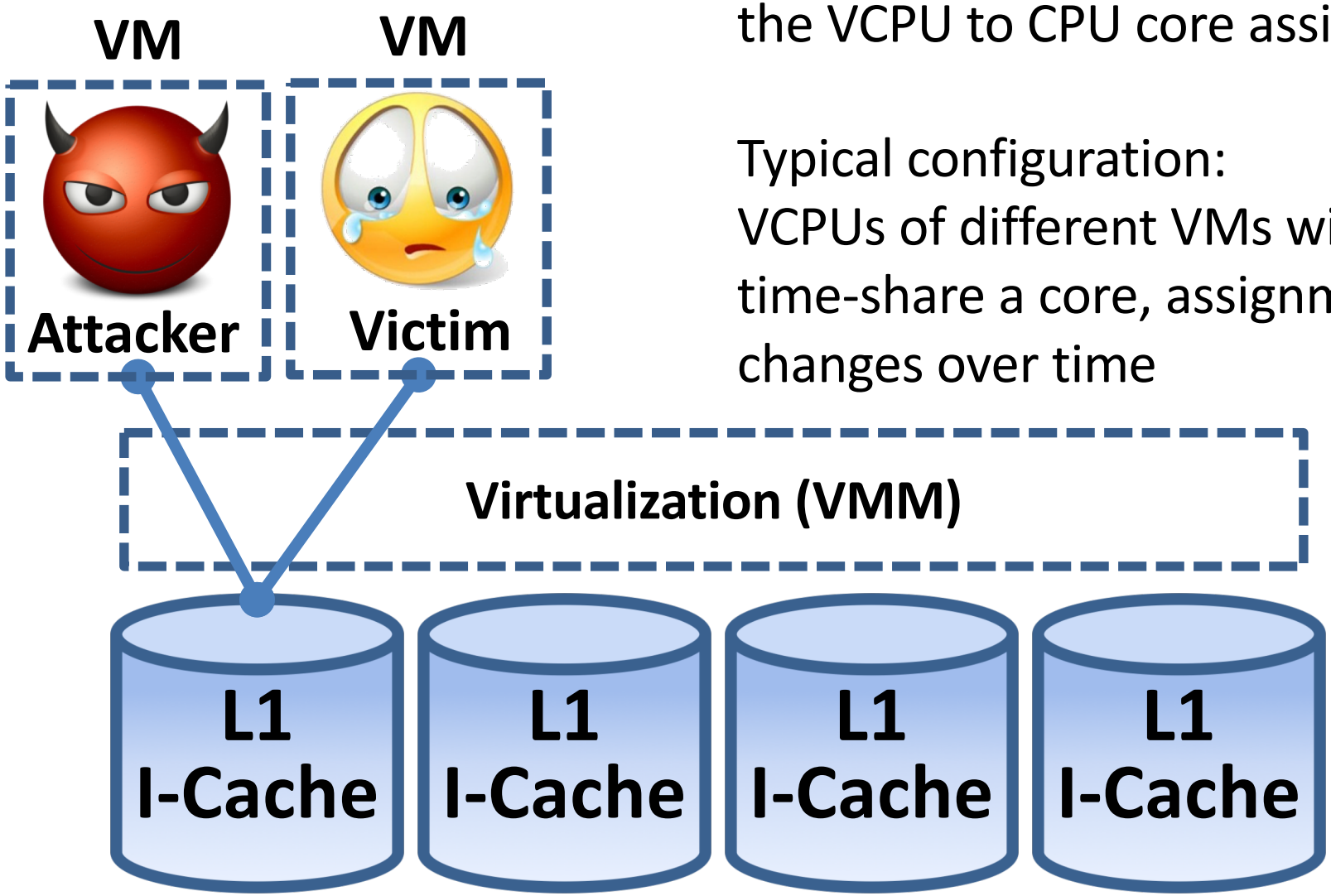
$e_i = 1$: extra instruction cache lines accessed



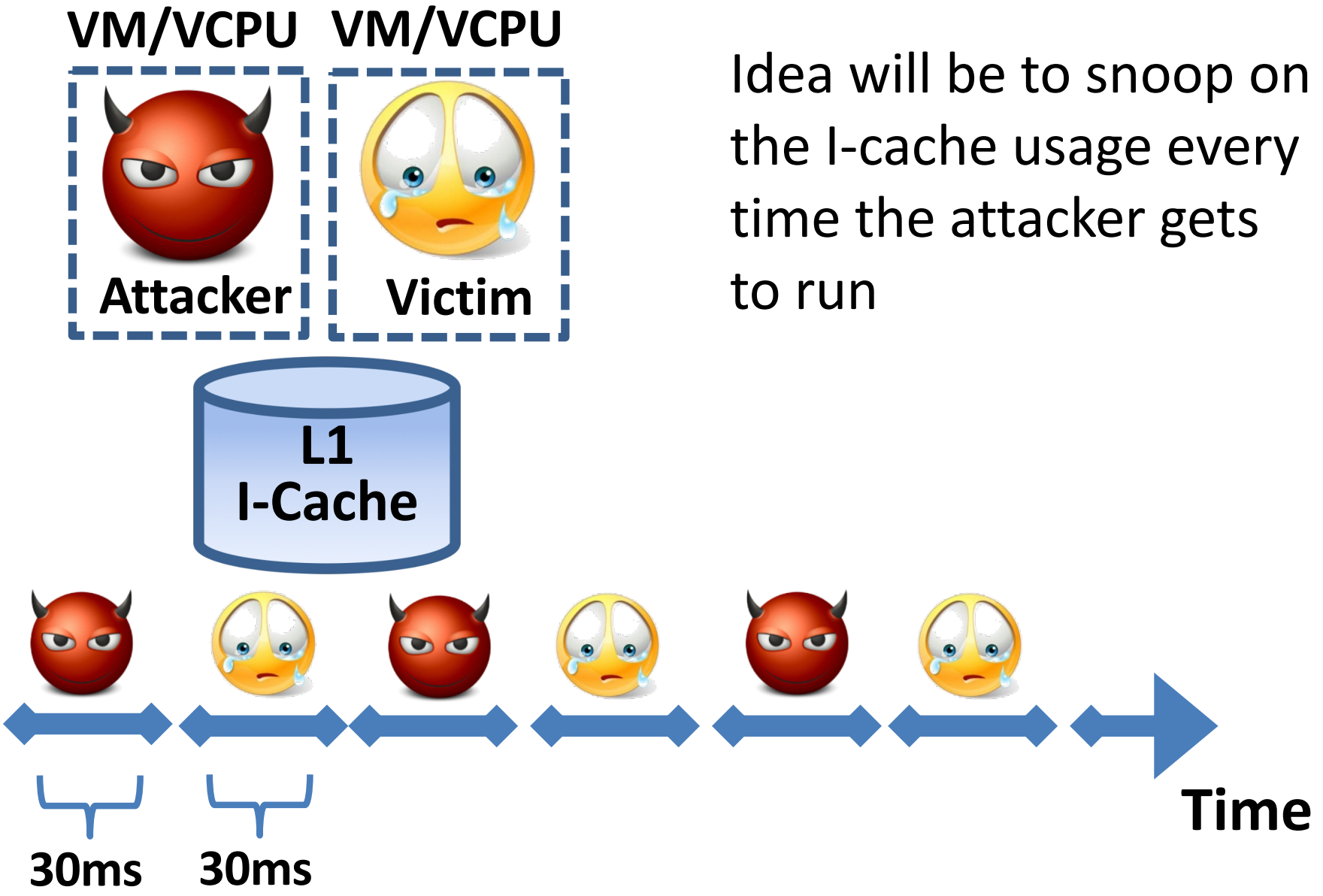
VMM core scheduling

VMM core scheduler determines the VCPU to CPU core assignment

Typical configuration:
VCPUs of different VMs will often time-share a core, assignment changes over time

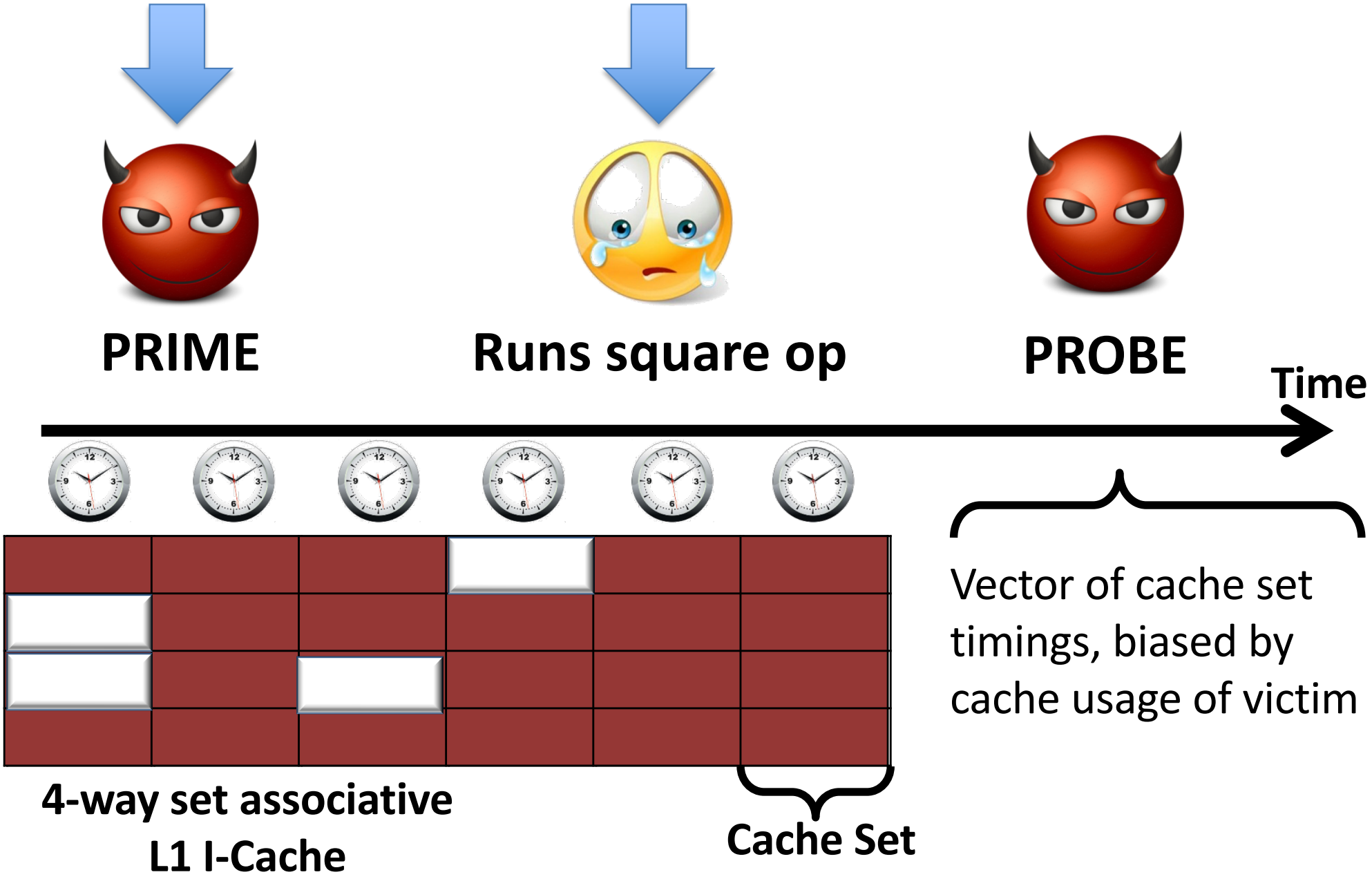


Time-sharing a core

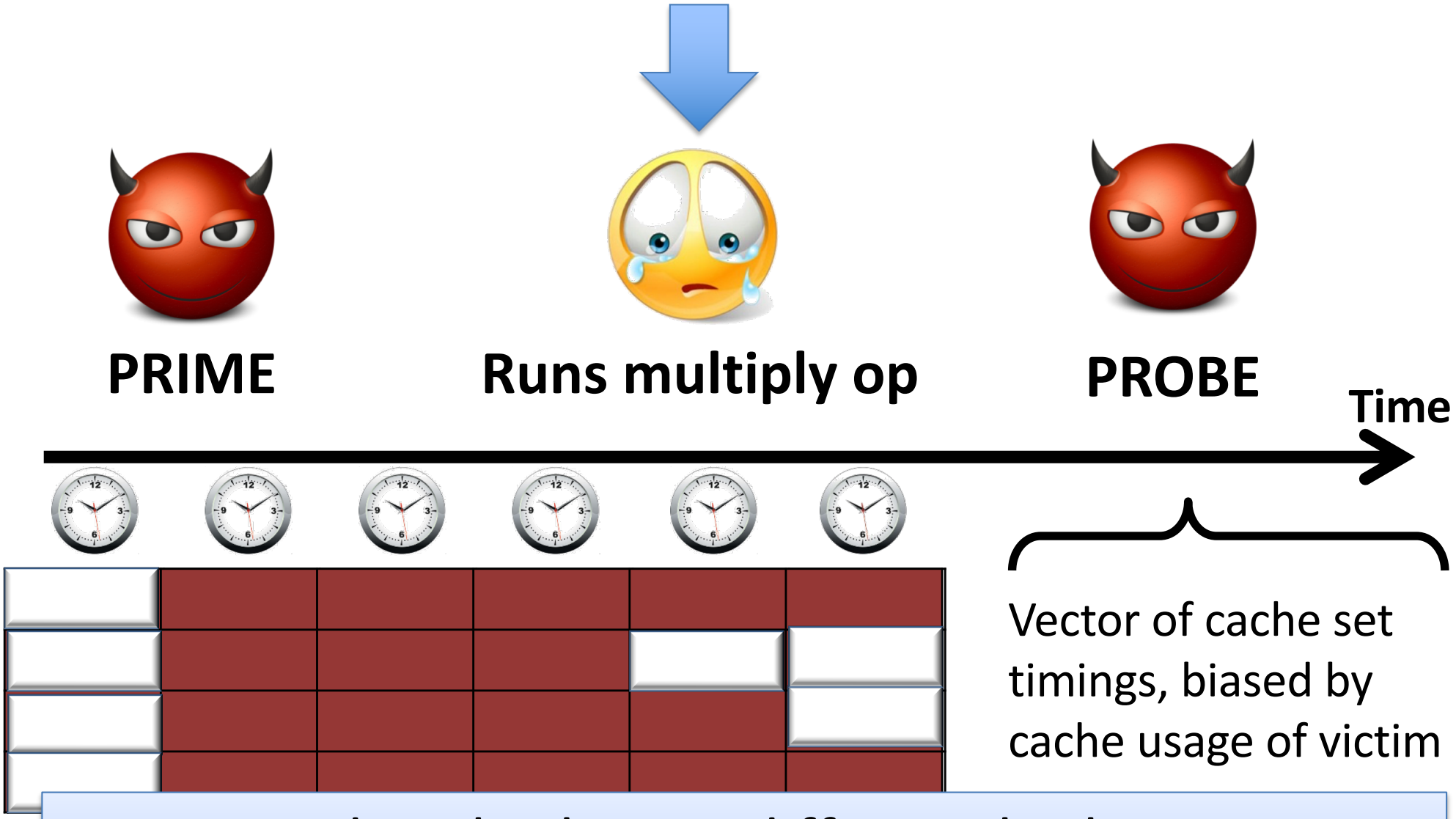


Idea will be to snoop on the I-cache usage every time the attacker gets to run

Prime-Probe Protocol

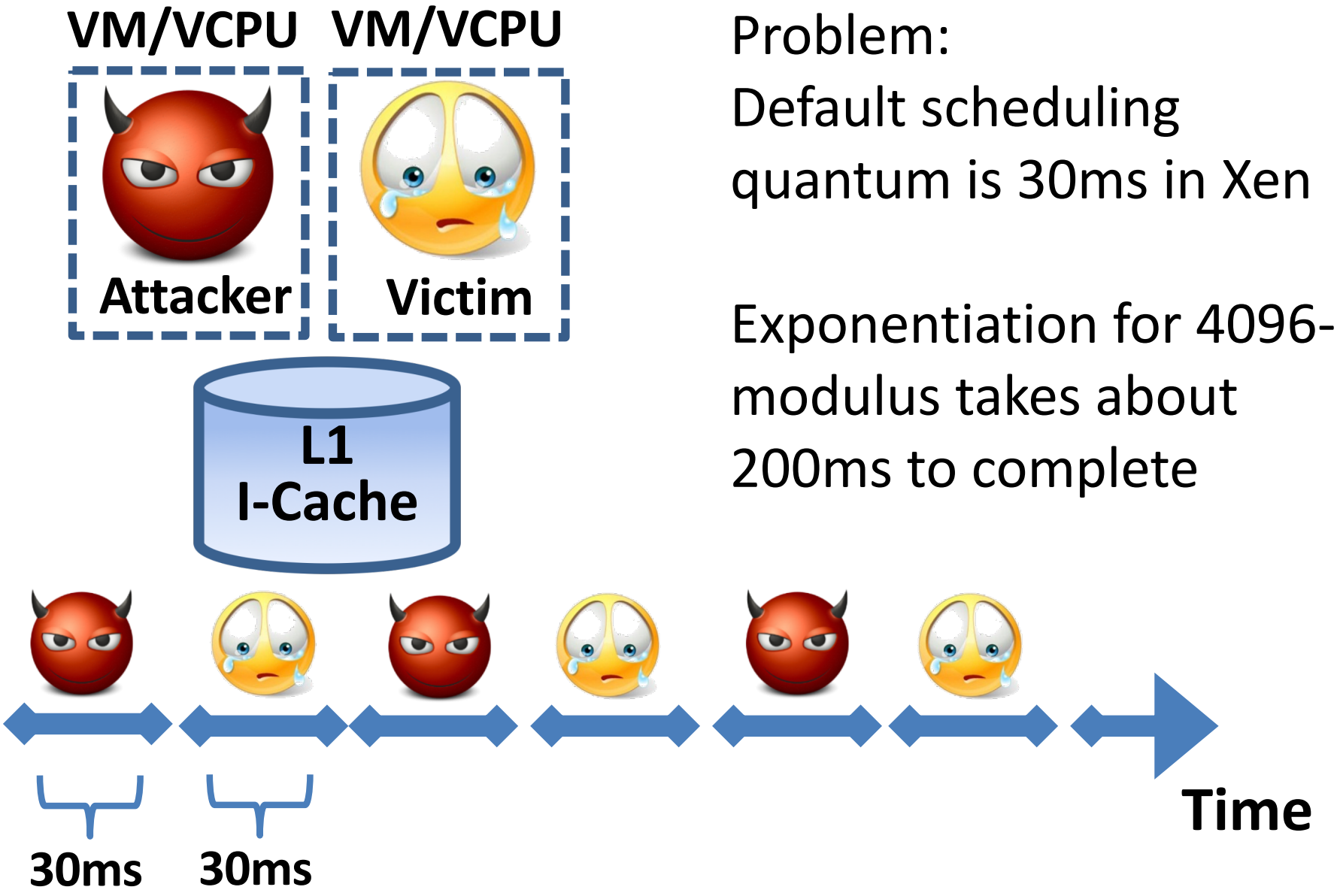


Prime-Probe Protocol



Square and Multiply give different-looking timing vectors (in the absence of noise)

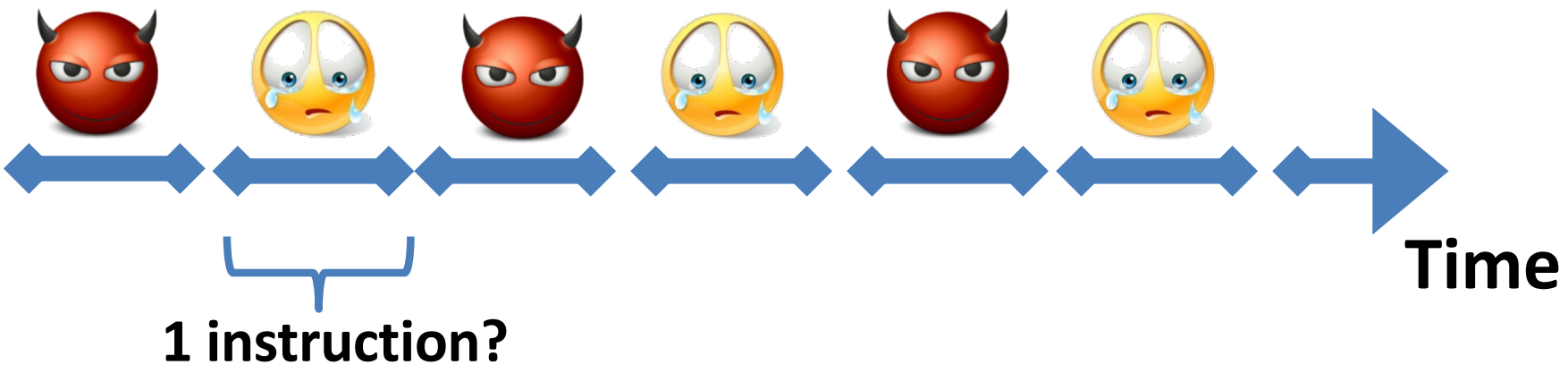
Time-sharing a core



Problem:
Default scheduling quantum is 30ms in Xen

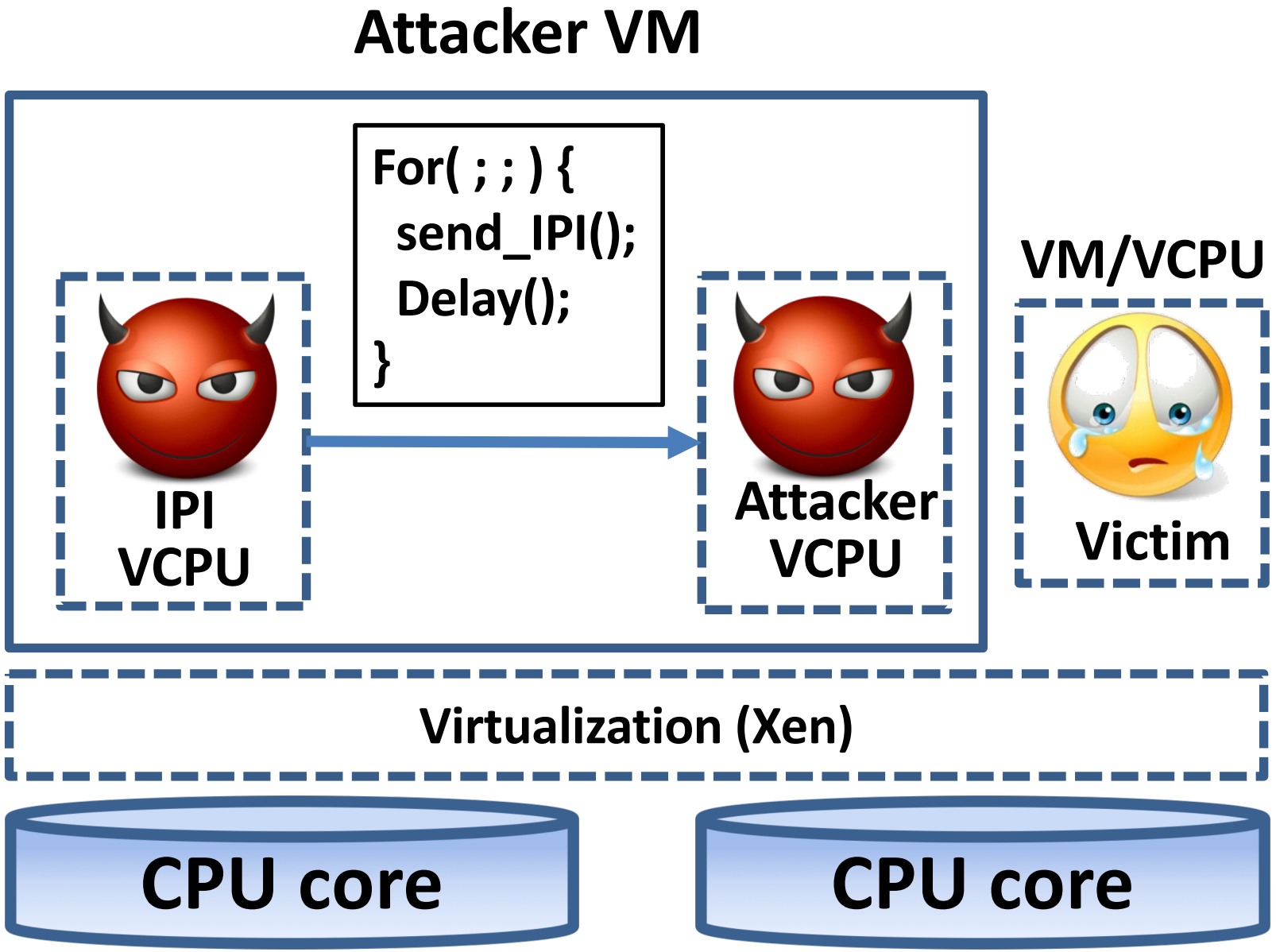
Exponentiation for 4096-bit modulus takes about 200ms to complete

Ideally …

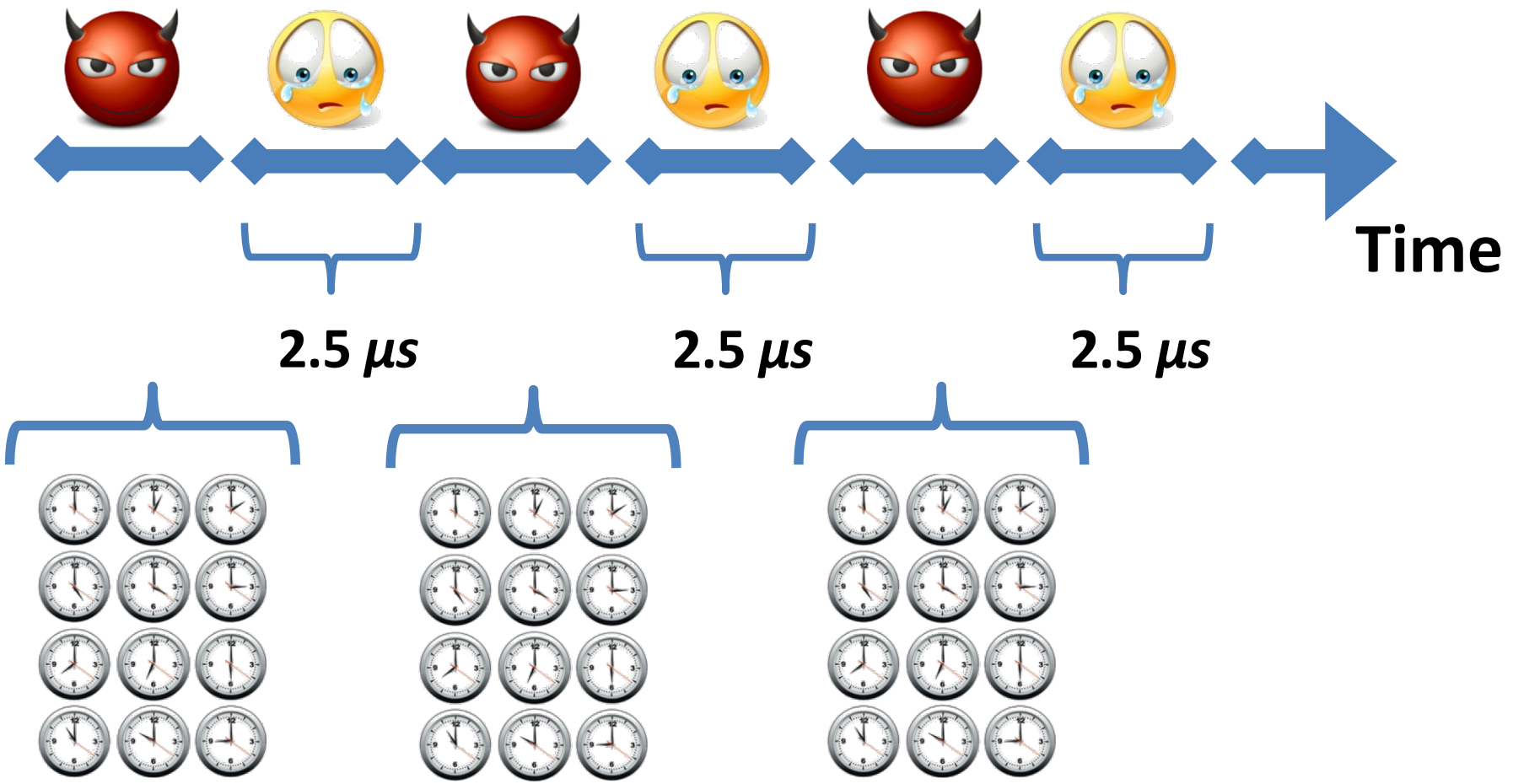


- Use **Interrupts** to preempt the victim:
  - **Inter-Processor interrupts (IPI)!**

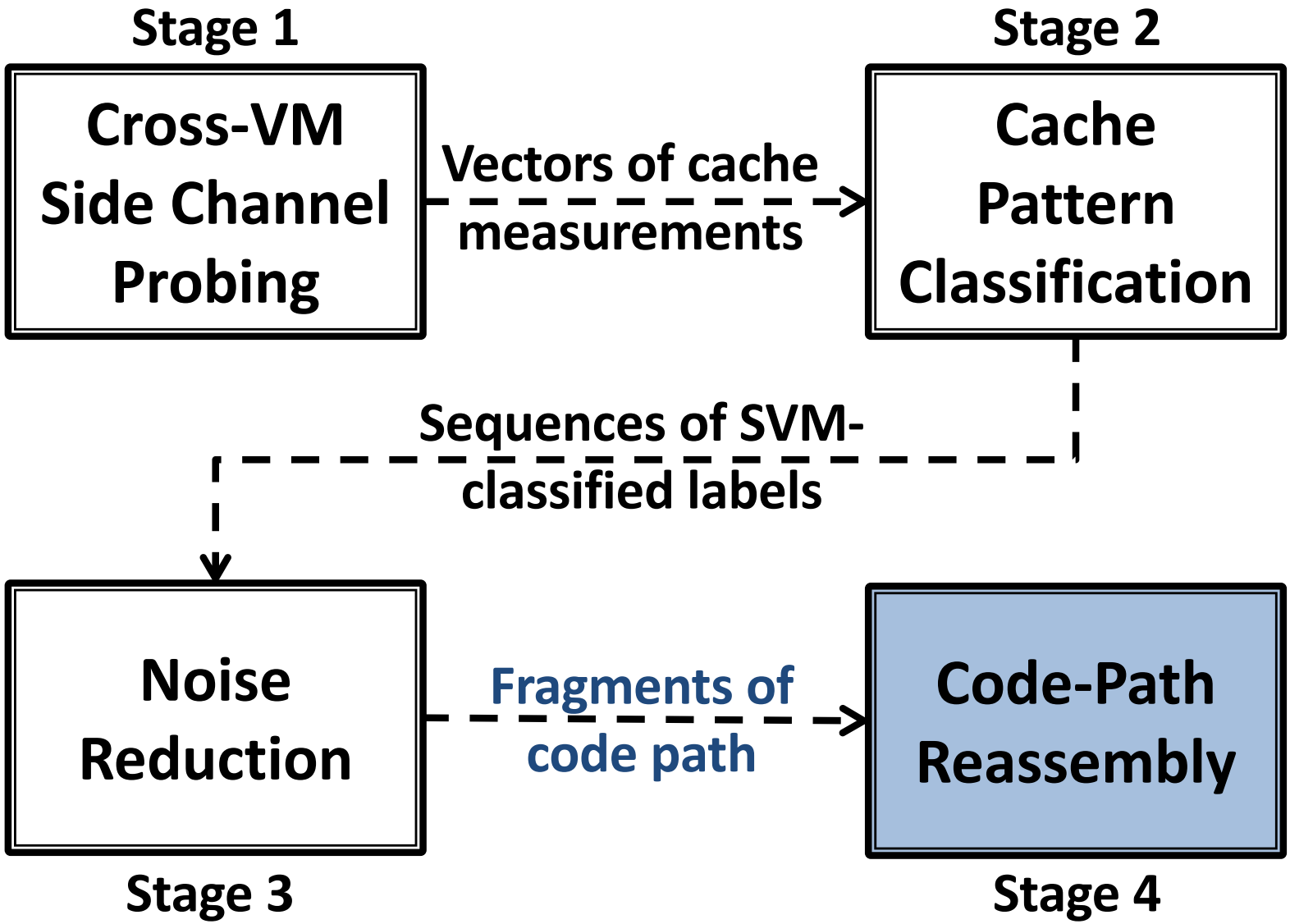
Inter-Processor Interrupts



Cross-VM Side Channel Probing



Outline


Stage 1
Cross-VM Side Channel Probing
Vectors of cache measurements
Stage 2
Cache Pattern Classification
Sequences of SVM-classified labels
Noise Reduction
Fragments of code path
Code-Path Reassembly
Stage 3
Stage 4

Evaluation

- Intel Yorkfield processor
  - 4 cores, 32KB L1 instruction cache
- Xen + linux + GnuPG + libgcrypt
  - Xen 4.0
  - Ubuntu 10.04, kernel version 2.6.32.16
  - Victim runs GnuPG v.2.0.19 (latest)
  - libgcrypt 1.5.0 (latest)
  - ElGamal decryption, 4096 bits

Results

- **Work-Conserving Scheduler**
  - 300,000,000 prime-probe results (6 hours)
  - Over 300 key fragments
  - Brute force the key in ~9800 guesses
- **Non-Work-Conserving Scheduler**
  - 1,900,000,000 prime-probe results (45 hours)
  - Over 300 key fragments
  - Brute force the key in ~6600 guesses

Lessons

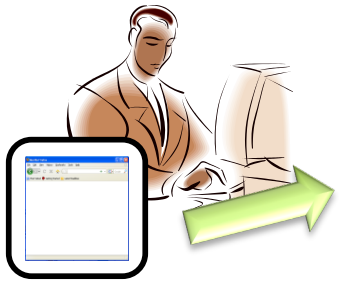
- But don't rely solely on them for:
  - VMM transparency
  - Containment
  - Strong isolation (side channels exist)
- Securing guest OS and host OS still very important for defense-in-depth

Virtual Machine Management

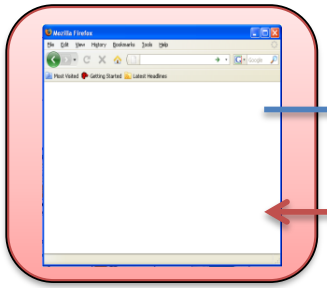
- Snapshots
  - Volume snapshot / checkpoint
    - persistent storage of VM
    - must boot from storage when resuming snapshot
  - Full snapshot
    - persistent storage and ephemeral storage (memory, register states, caches, etc.)
    - start/resume in between (essentially) arbitrary instructions
- VM image is a file that stores a snapshot

Virtual machines and secure browsing

"**Protect Against Adware and Spyware:** Users protect their PCs against adware, spyware and other malware while browsing the Internet with Firefox in a virtual machine."
[http://www.vmware.com/company/news/releases/player.html]

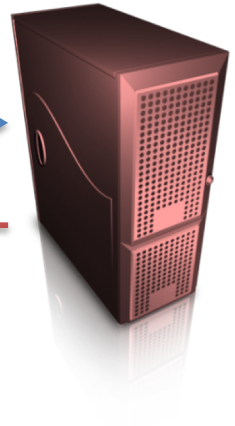


Clean snapshot of VM with browser running



http://www.freesoftware.com/

browser exploit



Virtual machine compromised, but not host OS

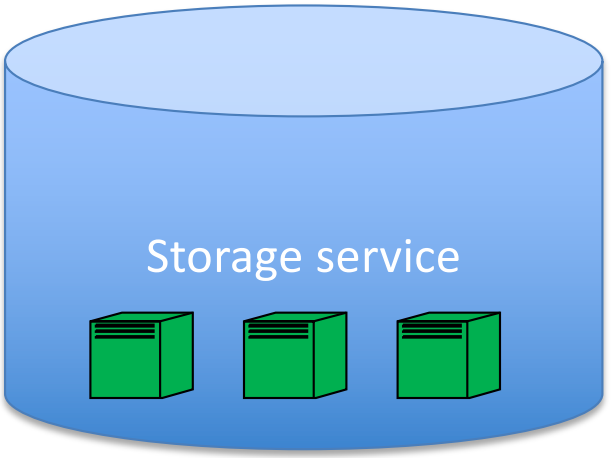
Resetting to snapshot removes malware

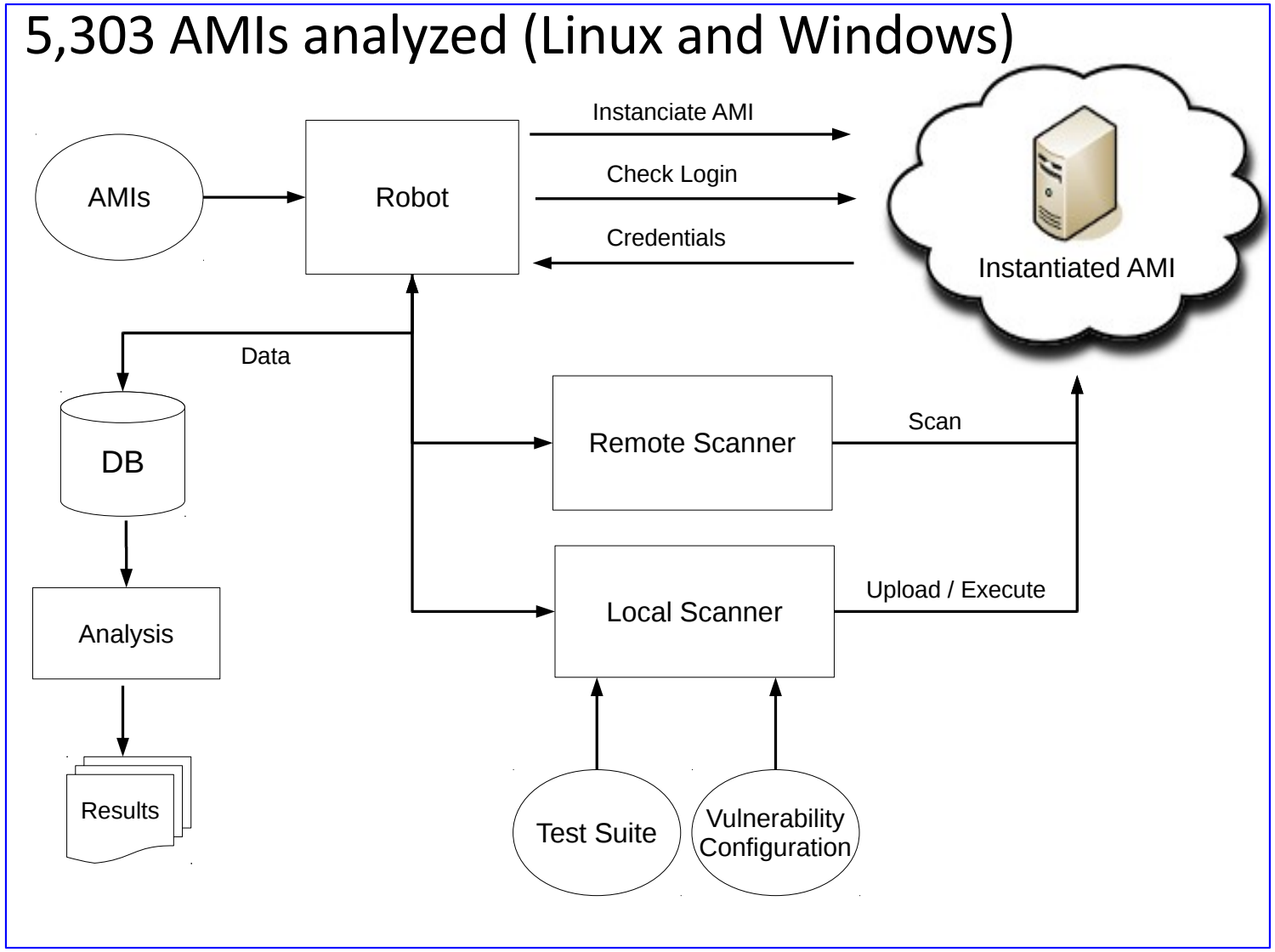
VM Management issues

- Reset vulnerabilities
  - Reuse of randomness
- Lack of diversity
- Identity management / credentials
- Known vulnerabilities

Amazon Machine Images (AMIs)

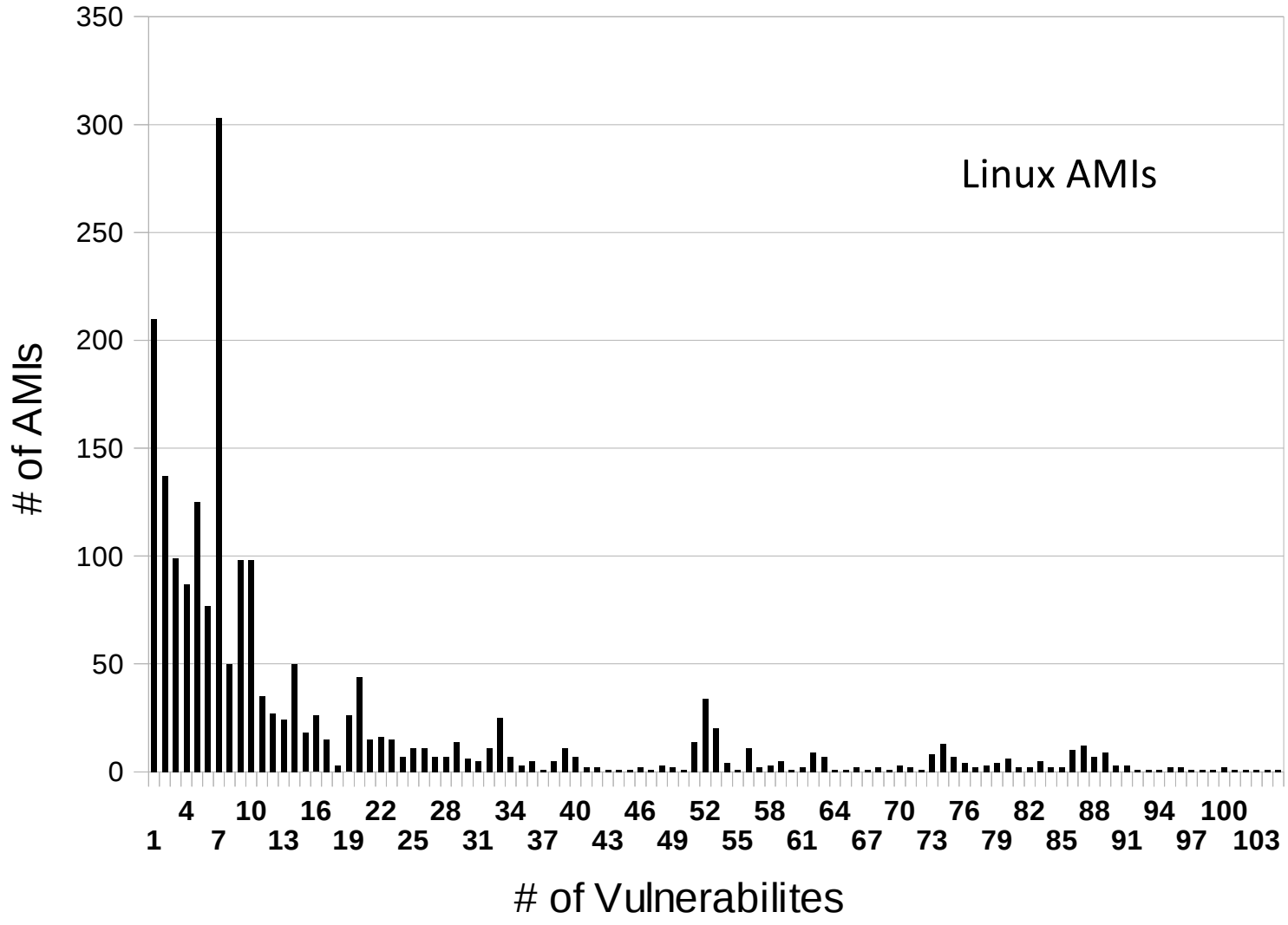
- Users set up volume snapshots / checkpoints that can then be run on the Elastic Compute Cloud (EC2)
- Can be marked as public and anyone can use your AMI





Balduzzi et al. “A Security Analysis of Amazon’s Elastic Compute Cloud Service – Long Version –”, 2011

See also Bugiel et al., “AmazonIA: When Elasticity Snaps Back”, 2011



Also: Malware found on a couple AMIs

Balduzzi et al. analysis

- Backdoors
  - AMIs include SSH public keys within authorized_keys
  - Password-based backdoors

	East	West	EU	Asia	Total
AMIs (%)	34.8	8.4	9.8	6.3	21.8
With Passwd	67	10	22	2	101
With SSH keys	794	53	86	32	965
With Both	71	6	9	4	90
Superuser Priv.	783	57	105	26	971
User Priv.	149	12	12	12	185

Table 2: Left credentials per AMI

Balduzzi et al. analysis

- Credentials for other systems
  - AWS secret keys (to control EC2 services of an account): 67 found
  - Passwords / secret keys for other systems: 56 found

Finding	Total	Image	Remote
Amazon RDS	4	0	4
dDNS	1	0	1
SQL	7	6	1
MySql	58	45	13
WebApp	3	2	1
VNC	1	1	0
Total	74	54	20

Table 3: Credentials in history files

Balduzzi et al. analysis

- Deleted files
  - One AMI creation method does block-level copying

Type	#
Home files (`/home, /root`)	33,011
Images (min. 800x600)	1,085
Microsoft Office documents	336
Amazon AWS certificates and access keys	293
SSH private keys	232
PGP/GPG private keys	151
PDF documents	141
Password file (`/etc/shadow`)	106

Table 5: Recovered data from deleted files

Response

"They told me it's not their concern, they just provide computing power," Balduzzi says. "It's like if you upload naked pictures to Facebook. It's not a good practice, but it's not Facebook's problem."

http://www.forbes.com/sites/andygreenberg/2011/11/08/researchers-find-amazon-cloud-servers-teeming-with-backdoors-and-other-peoples-data/

- Amazon notified customers with vulnerable AMIs
- Made private AMIs of non-responsive customers
- New tutorials for bundling systems
- Working on undelete issues…

Lessons

- New software management practices needed with VM snapshots
- Discussion:
  - New tool support?
  - How much worse is this than non-cloud server deployments?