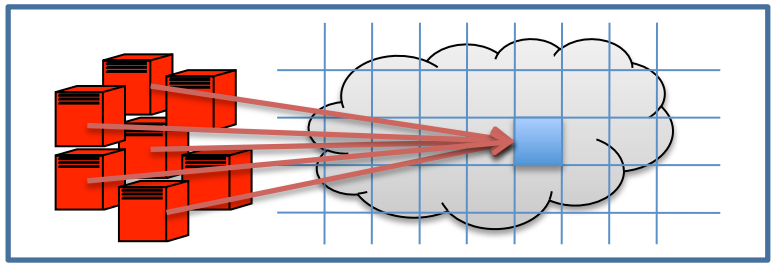
Virtual Security:
Information Leakage in Clouds and VM Reset Vulnerabilities



Thomas Ristenpart
University of Wisconsin

Today's talk in one slide

Third-party clouds:

"cloud cartography" to map internal infrastructure

get malicious VM on same physical server as victim

side-channels **might** leak confidential data of victim

Exploiting a placement vulnerability:
knowingly getting attack VM on server of victim

Joint with Eran Tromer, Hovav Shacham, Stefan Savage

Virtual machine snapshot technology:

run a VM twice from same snapshot

software re-uses cryptographic randomness

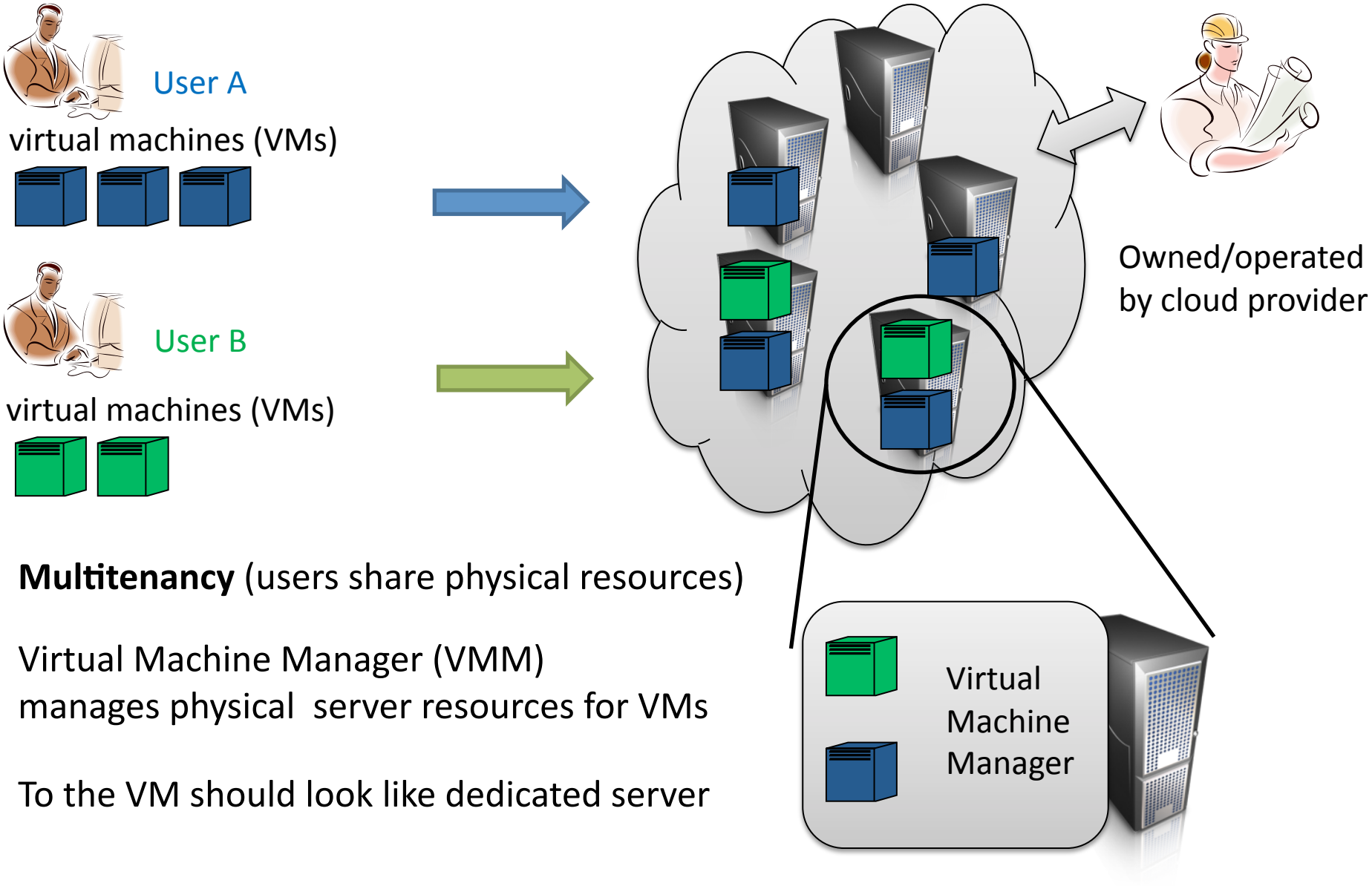
expose TLS sessions or steal TLS server secret key

Joint with Scott Yilek

Exploiting a reset vulnerability:
software unaware of resets, crypto fragile

A simplified model of public cloud computing

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

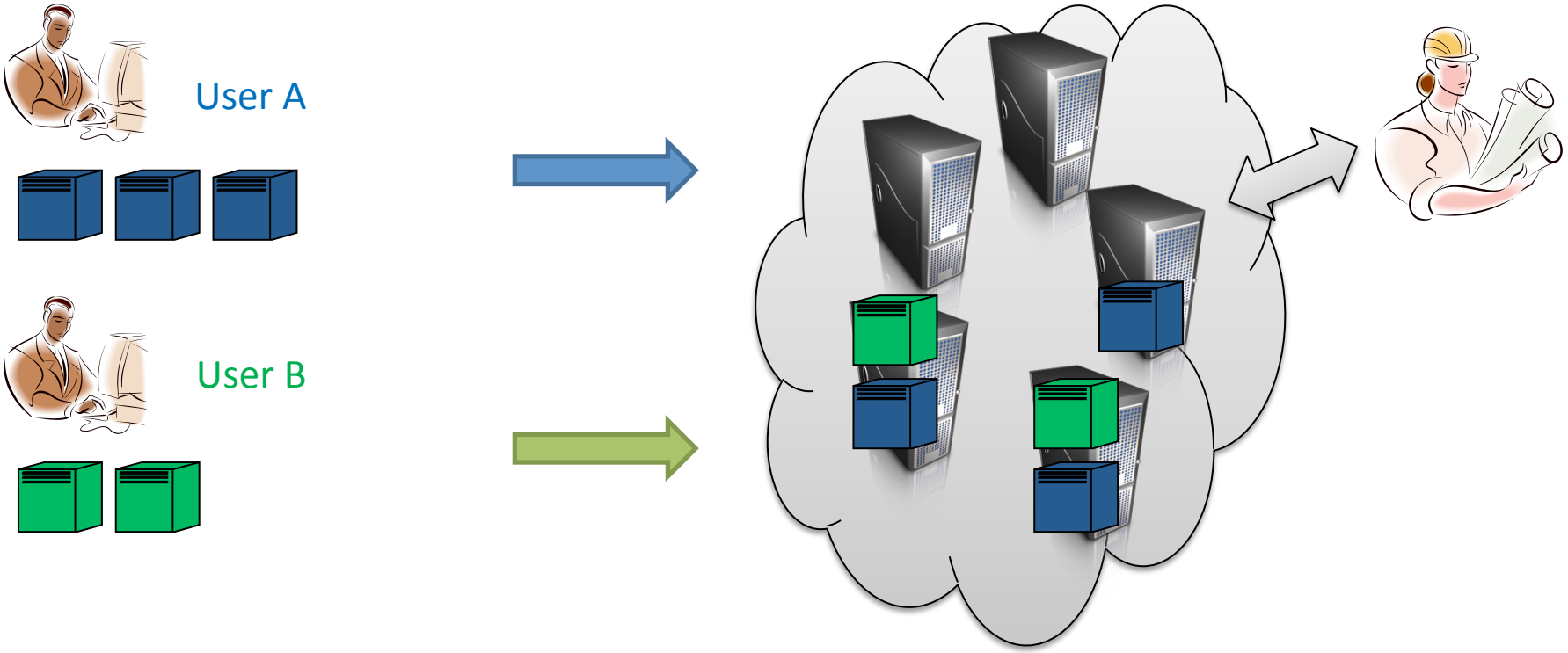


Multitenancy (users share physical resources)

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

To the VM should look like dedicated server

Trust models in cloud computing



Users must trust third-party provider to

not spy on running VMs / data

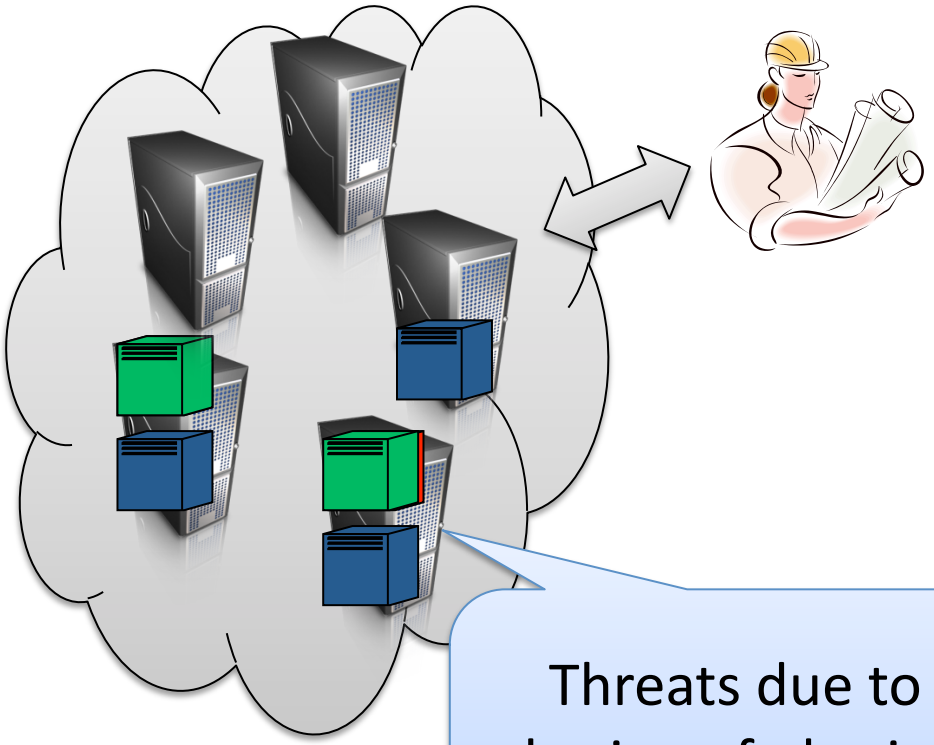
secure infrastructure from external attackers

secure infrastructure from internal attackers

Trust models in cloud computing

User A

Bad guy

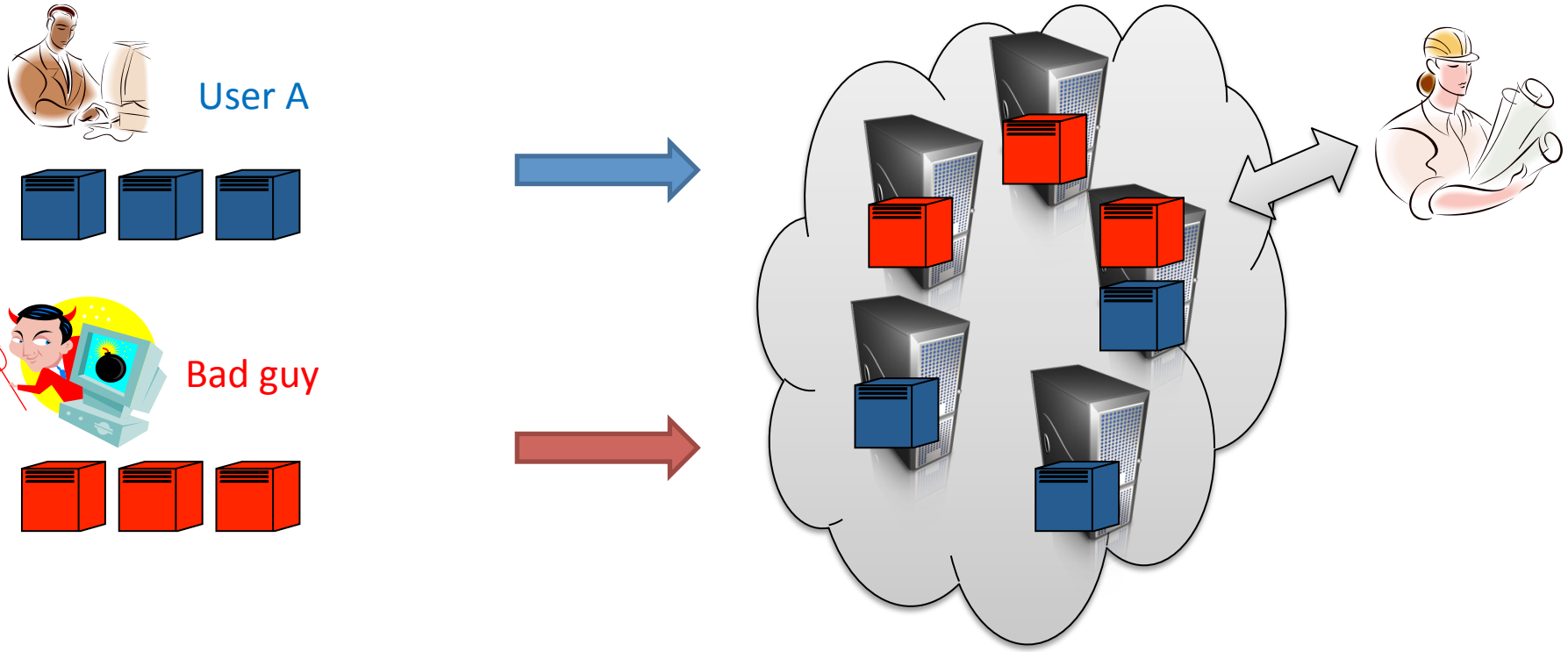


Threats due to sharing of physical infrastructure ?

Users must trust third-party provider to

- not spy on running VMs / data
- secure infrastructure from external attackers
- secure infrastructure from internal attackers
  - Your business competitor
  - Script kiddies
  - Criminals
  - …

One potential threat:



Attacker identifies one or more victims VMs in cloud

1) Achieve advantageous placement

- Attacker launches VMs
- VMs each check for co-residence on same server as victim

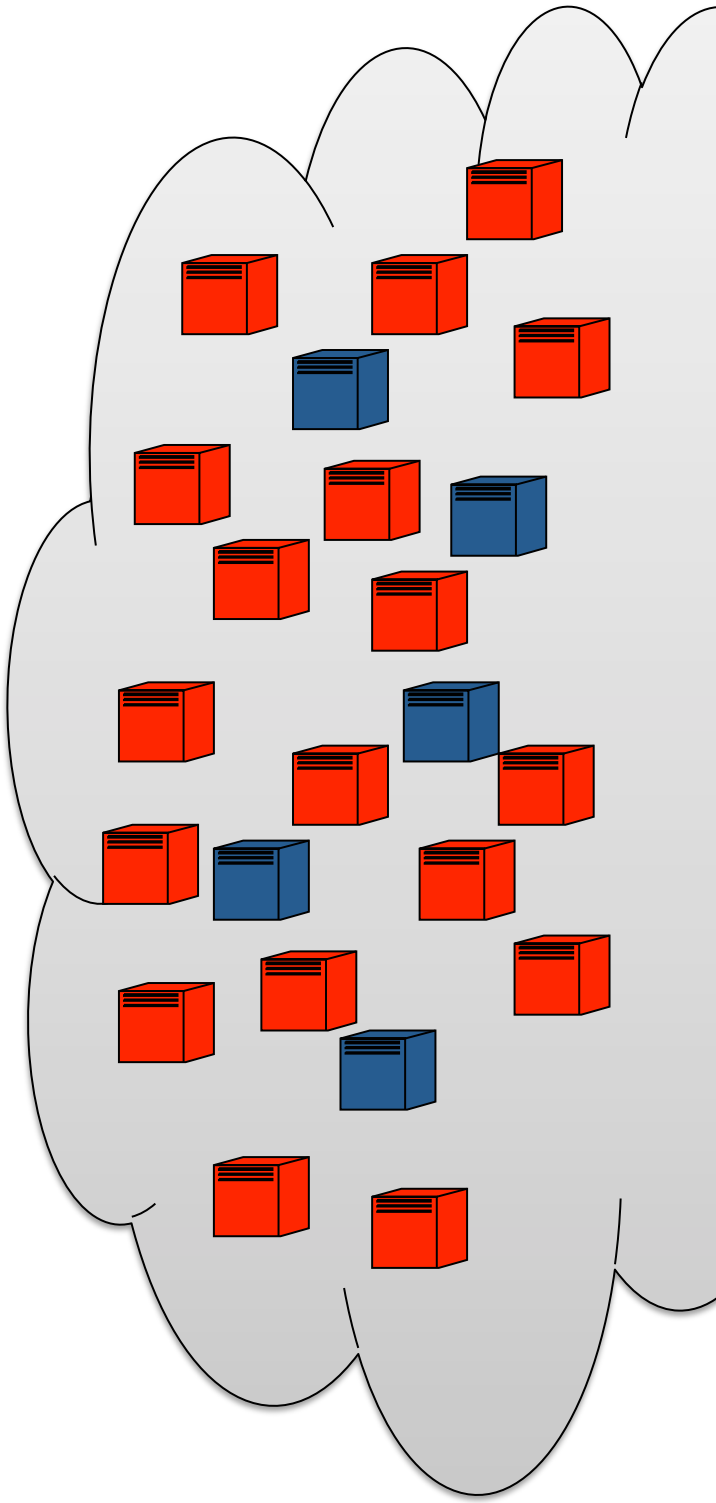
2) Launch attacks using physical proximity

Exploit VMM vulnerability DoS Side-channel attack

1 or more targets in the cloud and we want to attack them from same physical host

Launch lots of instances (over time), with each attempting an attack

Can attackers do better?



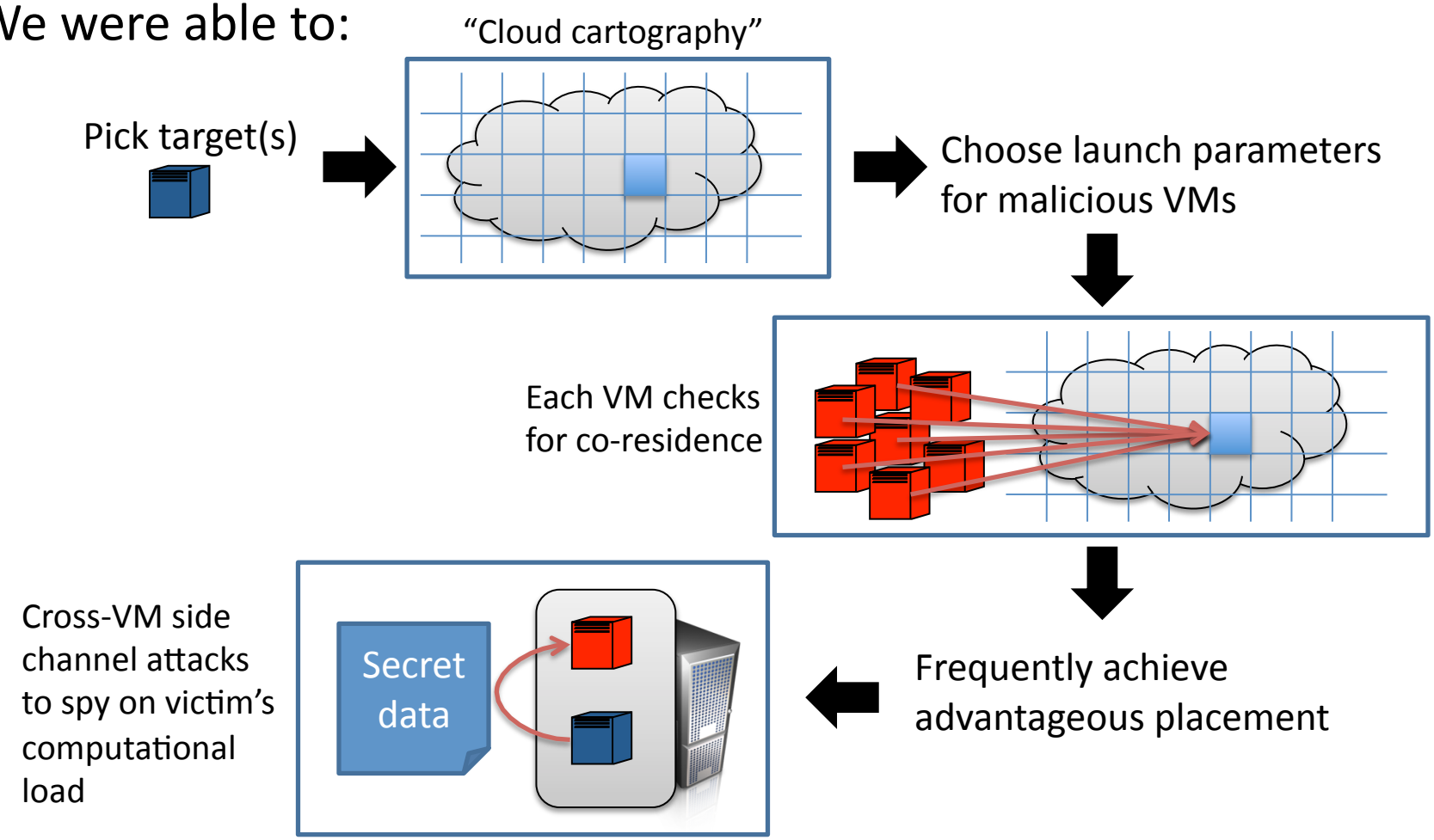
We performed a case study with Amazon's EC2

1) given no insider information

2) restricted by (the spirit of) Amazon's acceptable use policy (AUP)

(using only Amazon's customer APIs and very restricted network probing)

We were able to:



Some info about EC2 service (Fall 2008)

Linux-based VMs available
Uses Xen-based VM manager

launch parameters

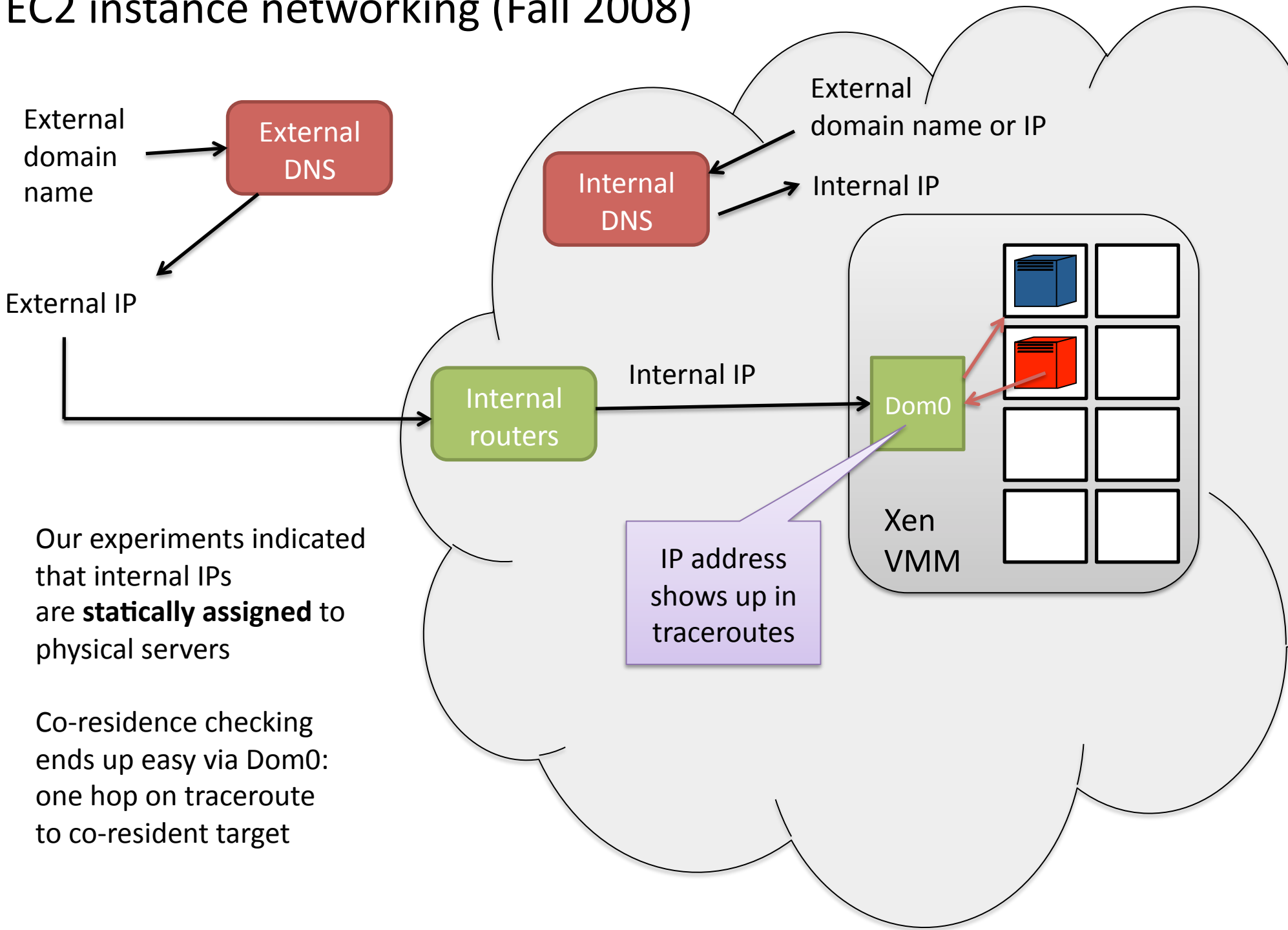
- User account
- 3 "availability zones" (Zone 1, Zone 2, Zone 3)
- 5 instance types (various combinations of virtualized resources)

Type	gigs of RAM	EC2 Compute Units (ECU)
m1.small (default)	1.7	1
m1.large	7.5	4
m1.xlarge	15	8
c1.medium	1.7	5
c1.xlarge	7	20

1 ECU = 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor

Limit of 20 instances at a time per account.
Essentially unlimited accounts with credit card.

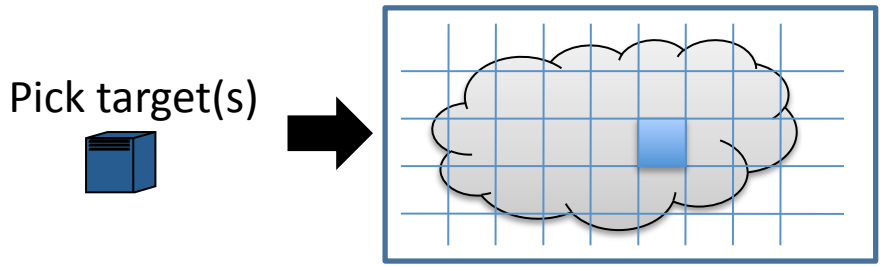
EC2 instance networking (Fall 2008)



Our experiments indicated that internal IPs are **statically assigned** to physical servers

Co-residence checking ends up easy via Dom0: one hop on traceroute to co-resident target

Cloud cartography

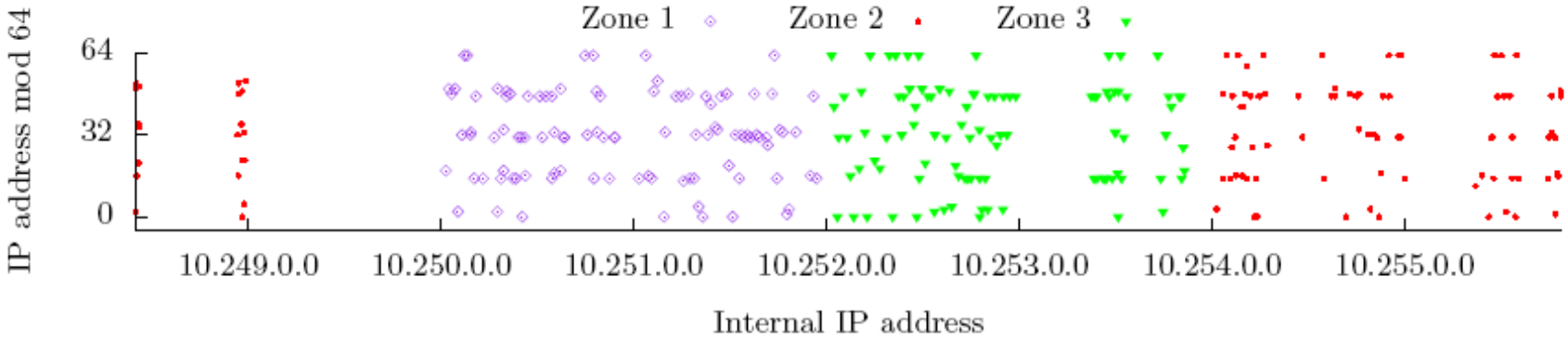


launch parameters

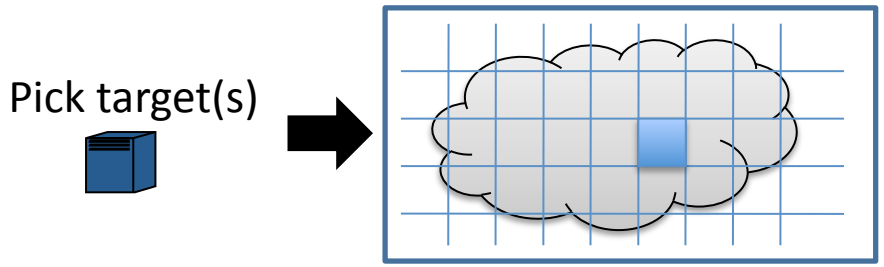
3 "availability zones"
(Zone 1, Zone 2, Zone 3)

5 instance types
(m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge)

User account



Cloud cartography

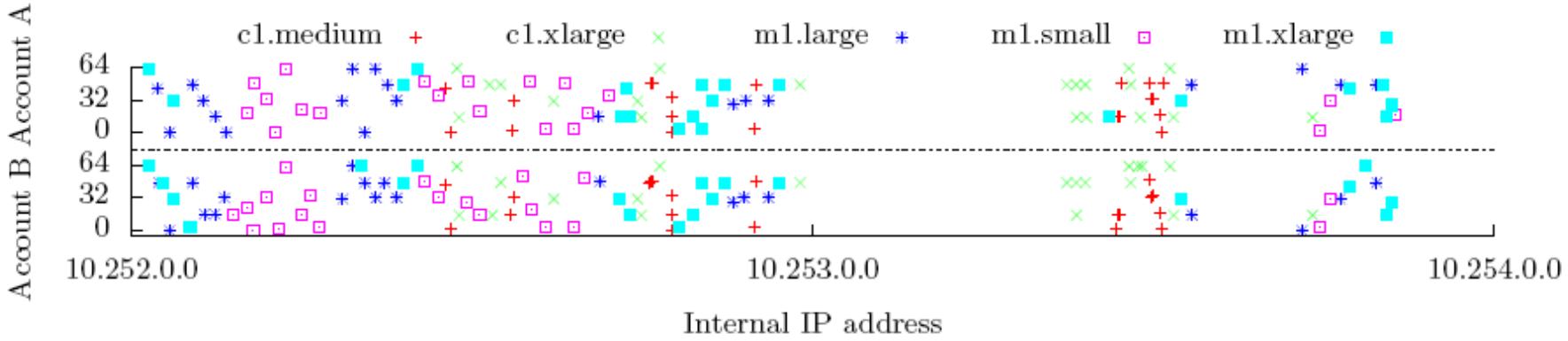


launch parameters

3 “availability zones”
(Zone 1, Zone 2, Zone 3)

5 instance types
(m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge)

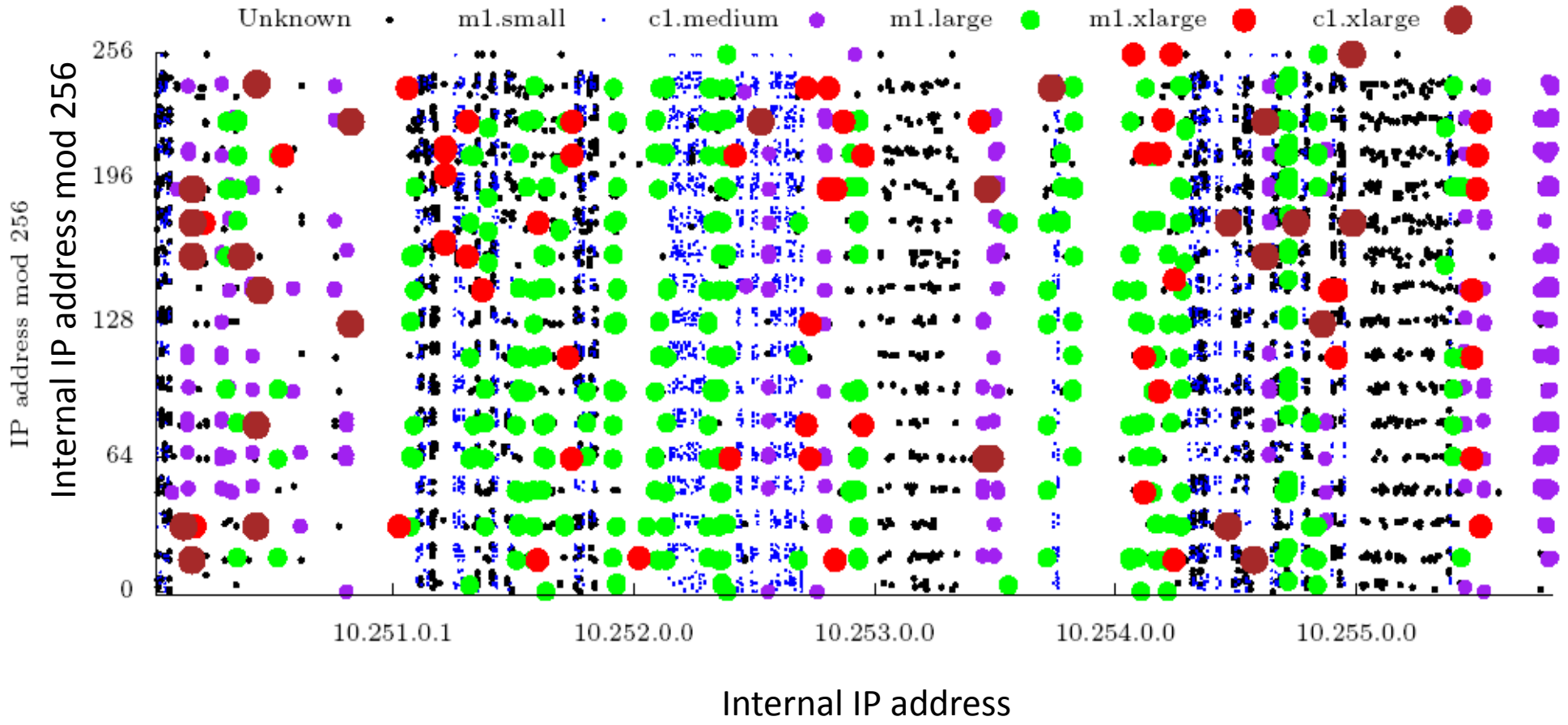
User account



Associate to each /24 an estimate of Availability zone and Instance Type

External IP → (DNS) Internal IP → (/24) Availability zone, Instance Type

Mapping 6,577 public HTTP servers running on EC2 (Fall 2008)



Achieving co-residence

"Brute-forcing" co-residence

Attacker launches many VMs over a relatively long period of time in target's zone and of target type

Experiment:

1,686 public HTTP servers as stand-in "targets" running m1.small and in Zone 3 (via our map)

1,785 "attacker" instances launched over 18 days

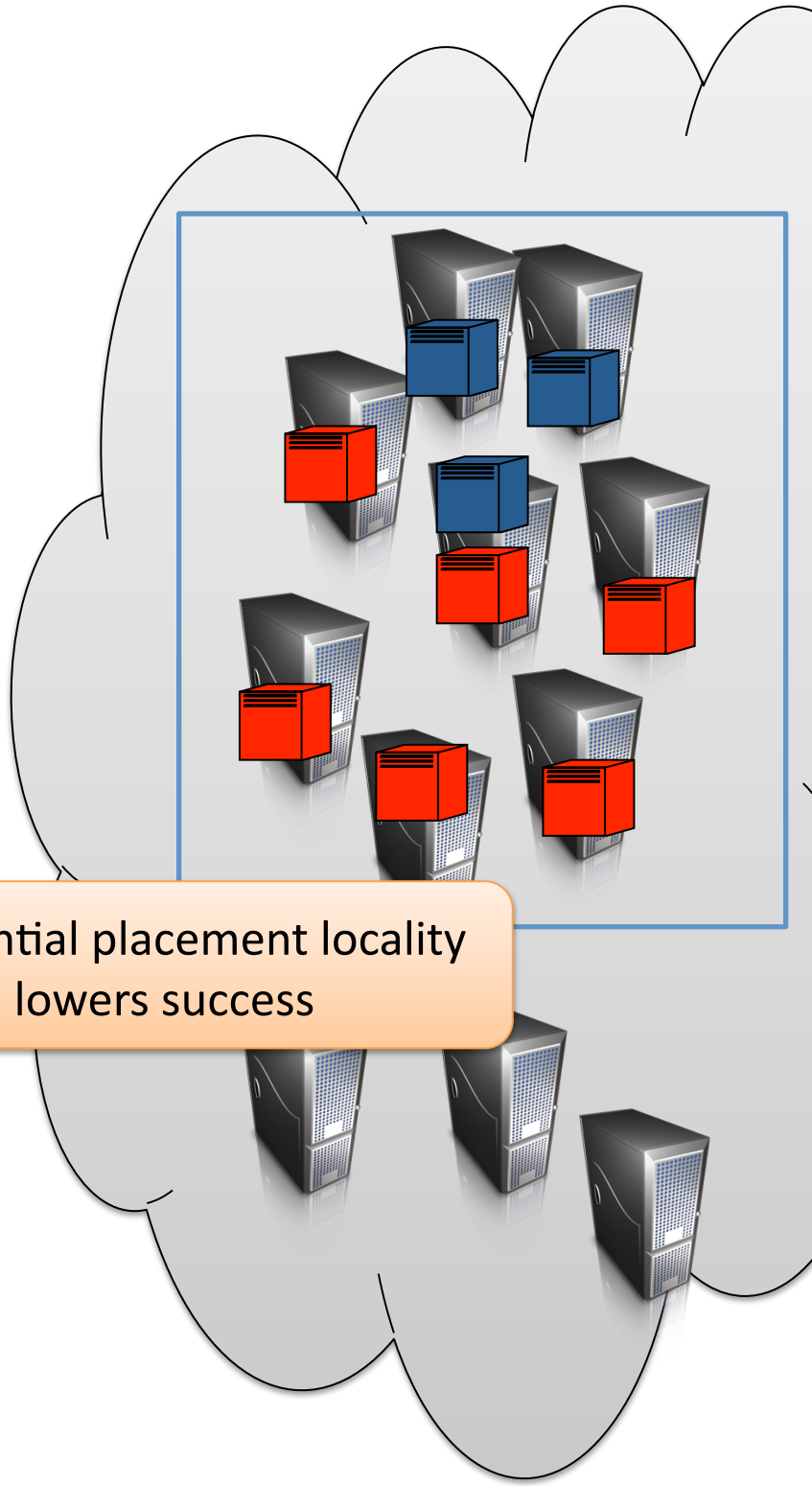
Each checked co-residence against all targets

Results:

78 unique Dom0 IPs

141 / 1,686 (8.4%) had attacker co-resident

Lower bound on true success rate



Sequential placement locality lowers success

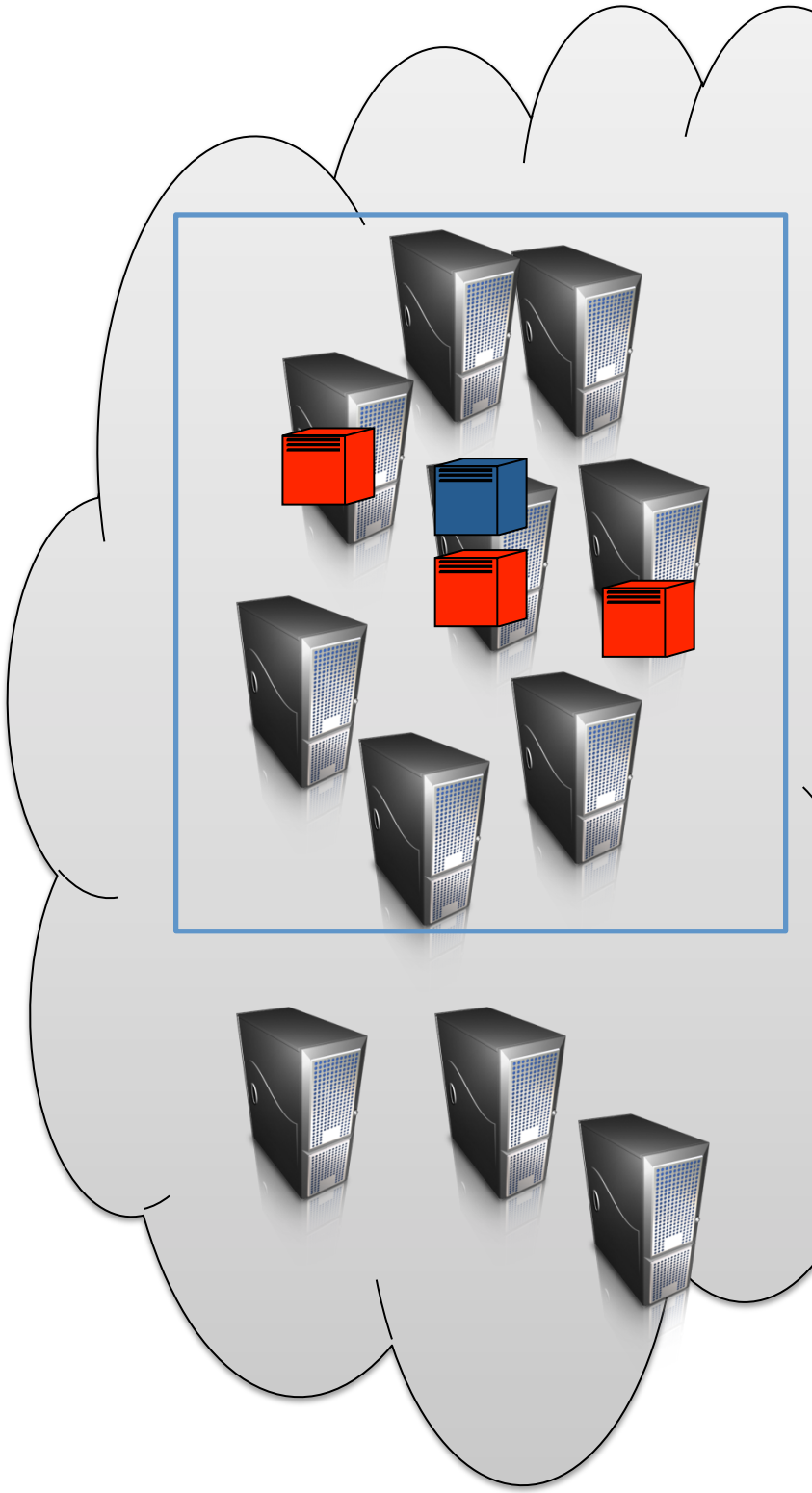
Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

Launch many instances in parallel near time of target launch

Attackers might arrange this due to dynamic nature of cloud use:

- Auto-scaling services (Amazon, RightScale, …)
- Cause target VM to crash, relaunch
- Wait for maintenance cycles
- …



Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

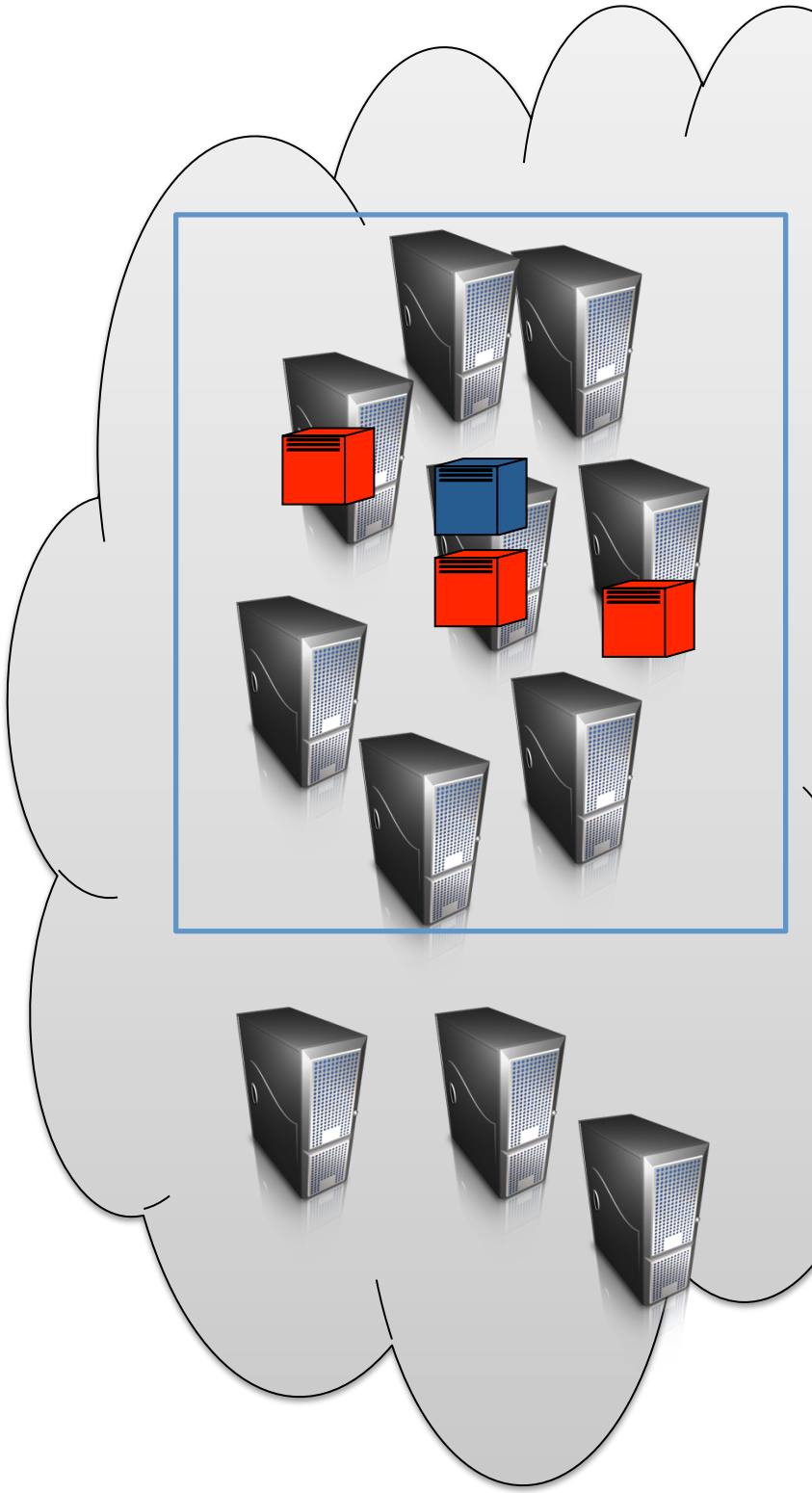
Launch many instances in parallel near time of target launch

<u>Experiment:</u>

Repeat for 10 trials:

1) Launch 1 target VM (Account A)
2) 5 minutes later, launch 20 “attack” VMs (alternate using Account B or C)
3) Determine if any co-resident with target

4 / 10 trials succeeded

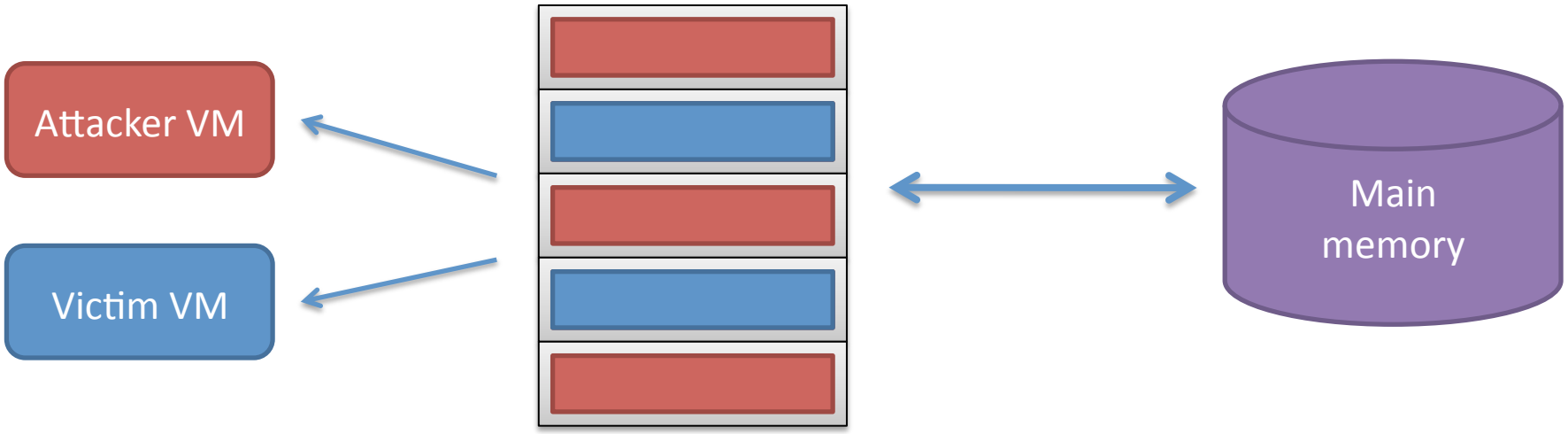


Attacker has uncomfortably good chance
at achieving co-residence with your VM

What can the attacker then do?

Cross-VM load measurement using CPU cache contention

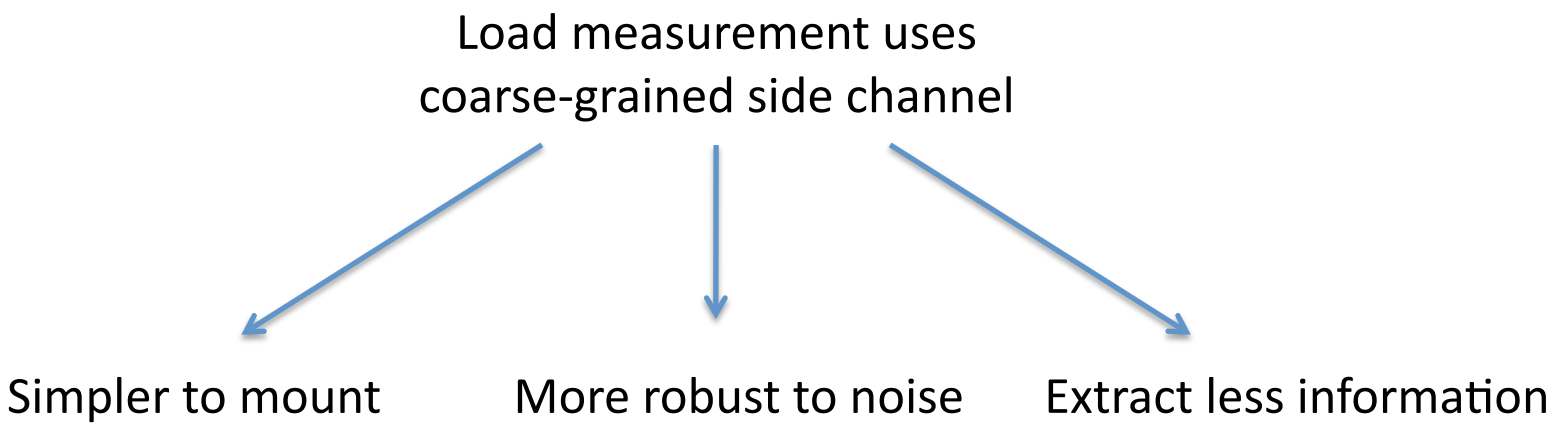
Extends techniques of [Osvik, Shamir, Tromer – '05]



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

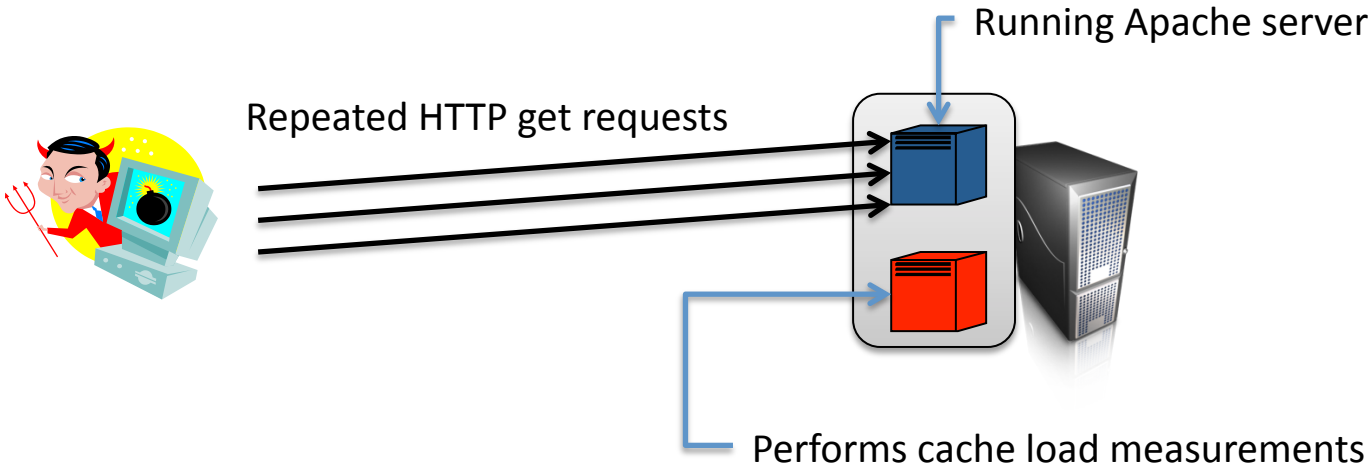
2) Busy loop (allow victim to run)

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



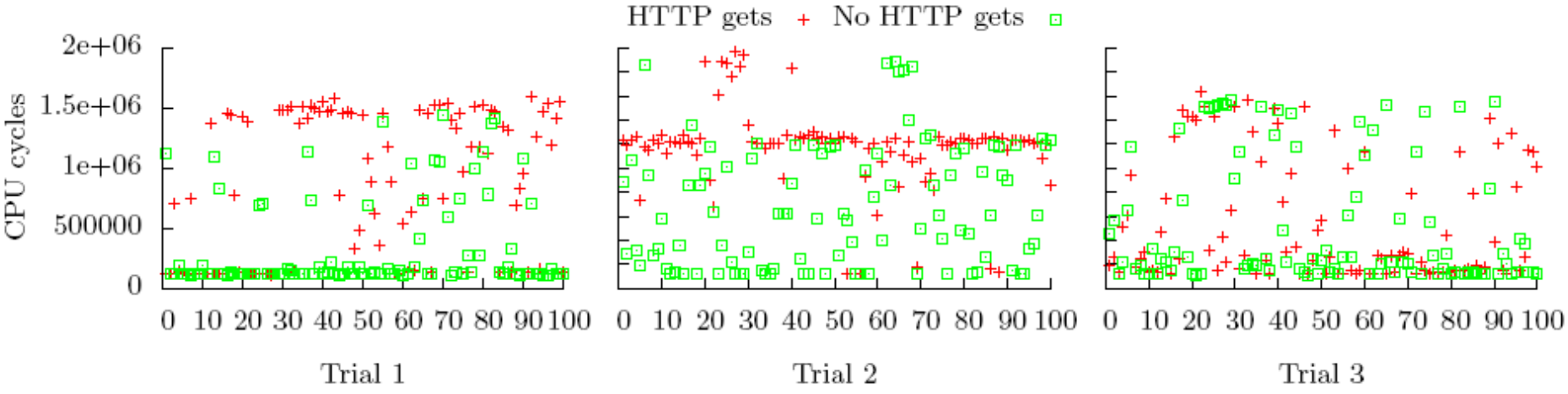
coarse side channels could be damaging
in hands of clever attackers

Cache-based load measurement to determine co-residence



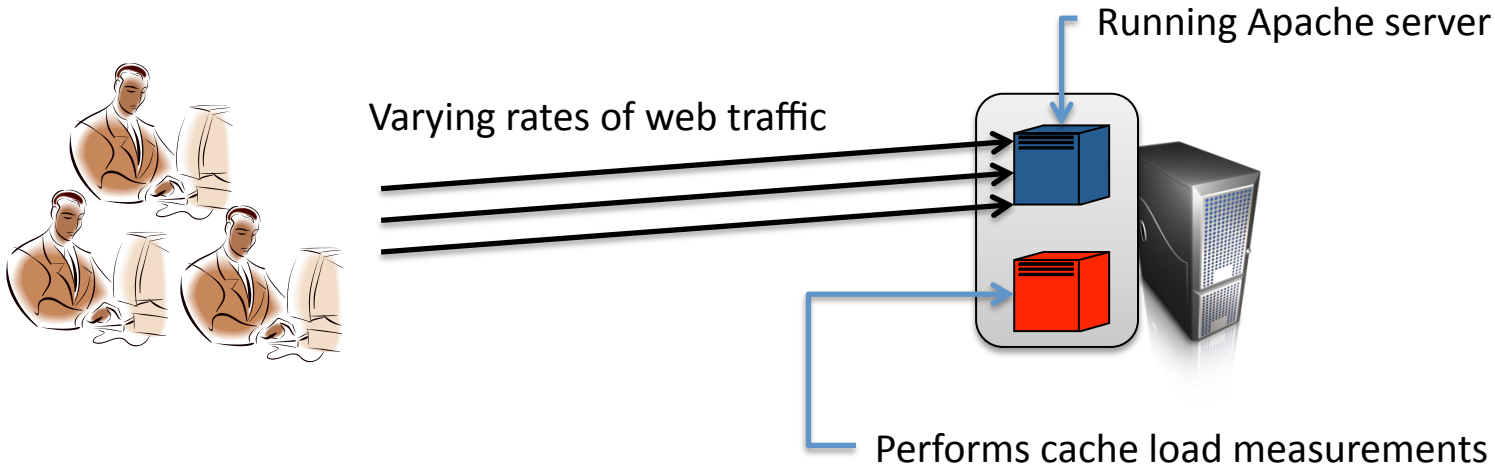
3 pairs of instances, 2 pairs co-resident and 1 not

100 cache load measurements during **HTTP gets** (1024 byte page) and with **no HTTP gets**

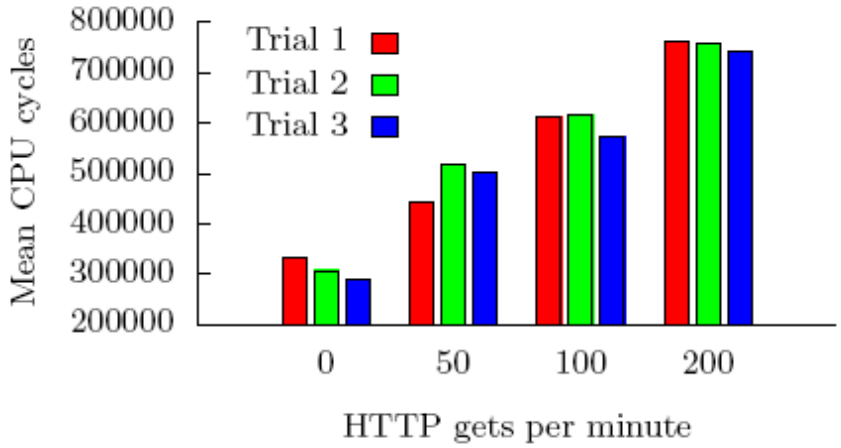


Instances co-resident | Instances co-resident | Instances NOT co-resident

Cache-based load measurement of traffic rates



3 trials with 1 pair of co-resident instances:
1000 cache load measurements during
0, 50, 100, or 200 **HTTP gets** (3 Mbyte page) per minute for ~1.5 mins

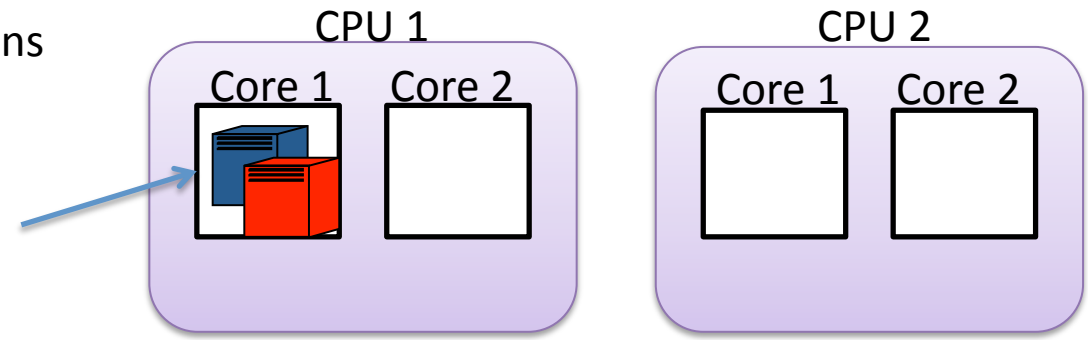


More on cache-based physical channels

Keystroke timing in experimental testbed similar to EC2 m1.small instances

AMD Opterons

VMs pinned to core

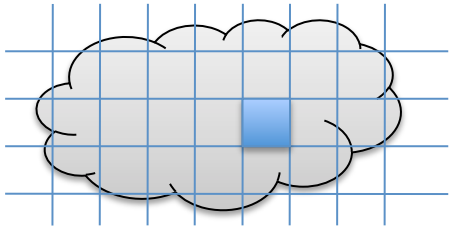


If VMs pinned to same core, then cache-load measurements allow cross-VM keystroke detection

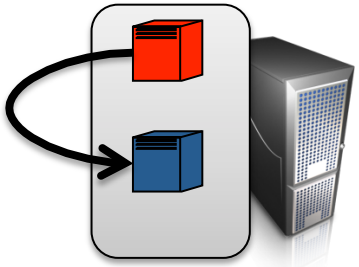
Keystroke timing of this form might be sufficient for the password recovery attacks of [Song, Wagner, Tian 01]

What can cloud providers do?

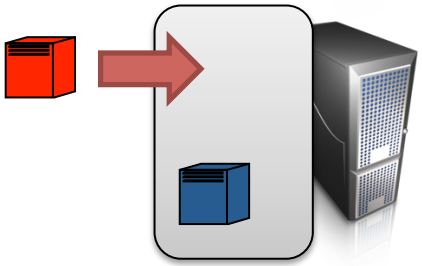
1) Cloud cartography



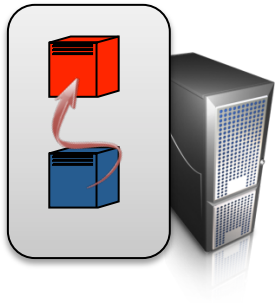
2) Checking for co-residence



3) Achieving co-residence



4) Side-channel information leakage



Possible counter-measures:

- Random Internal IP assignment
- Isolate each user's view of internal address space

- Hide Dom0 from traceroutes

- Allow users to opt out of multitenancy

- Hardware or software countermeasures to stop leakage [Ber05,OST05,Page02,Page03, Page05,Per05]

Today's talk in one slide

Third-party clouds:

"cloud cartography" to map internal infrastructure

get malicious VM on same physical server as victim

side-channels **might** leak confidential data of victim

Exploiting a placement vulnerability:
knowingly getting attack VM on server of victim

Joint with Eran Tromer, Hovav Shacham, Stefan Savage

Virtual machine snapshot technology:

run a VM twice from same snapshot

software re-uses cryptographic randomness

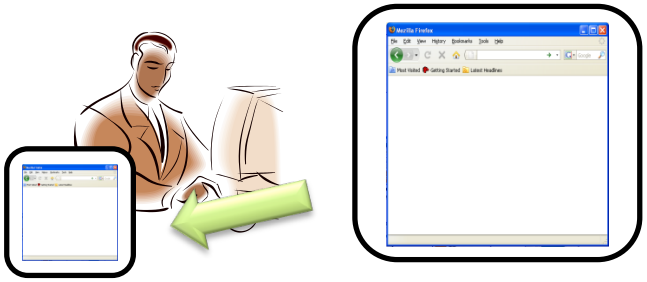
expose TLS sessions or steal TLS server secret key

Joint with Scott Yilek

Exploiting a reset vulnerability:
software unaware of resets, crypto fragile

Virtual machines and snapshots can improve security

Snapshot records exact state of VM, including persistent storage and active memory.



"**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]

vmware

"Your dad can do his [private] surfing on the virtual machine and can even set it to reset itself whenever the virtual computer is restarted, so there's no need to worry about leaving tracks. … I recommend VMware because you can download a free version of VMware Server for home use. "
[Rescorla, http://www.thestranger.com/seattle/SavageLove?oid=490850]

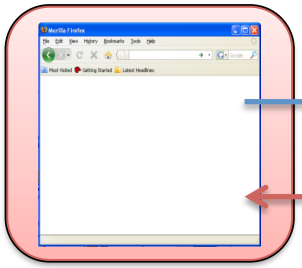
Example: using a VM snapshot for browser security

Fresh VM

Load browser

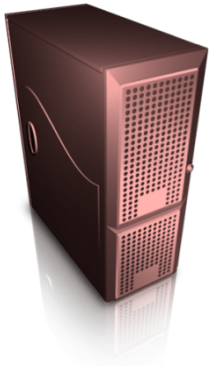
Take snapshot

Each new browsing session, reset VM by resuming from snapshot



http://www.freesoftware.com/

browser exploit



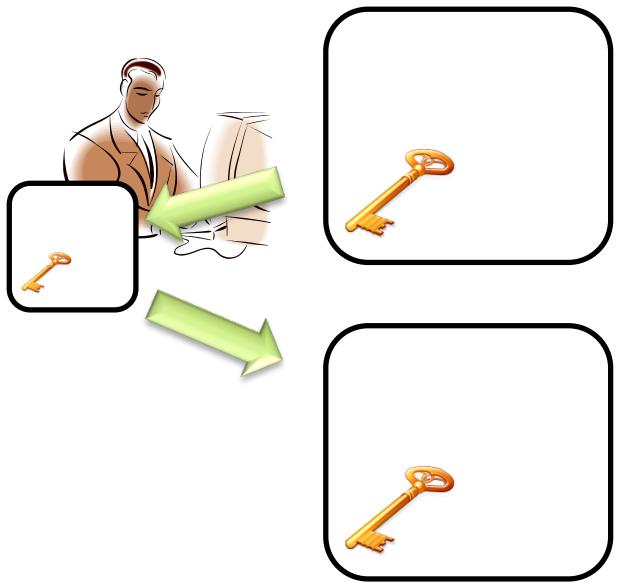
Virtual machine compromised

Resetting to snapshot removes malware!

Can virtualization introduce security problems?

[Garfinkel, Rosenblum 05] discuss possibility that snapshot use could lead to (what we call) reset vulnerabilities

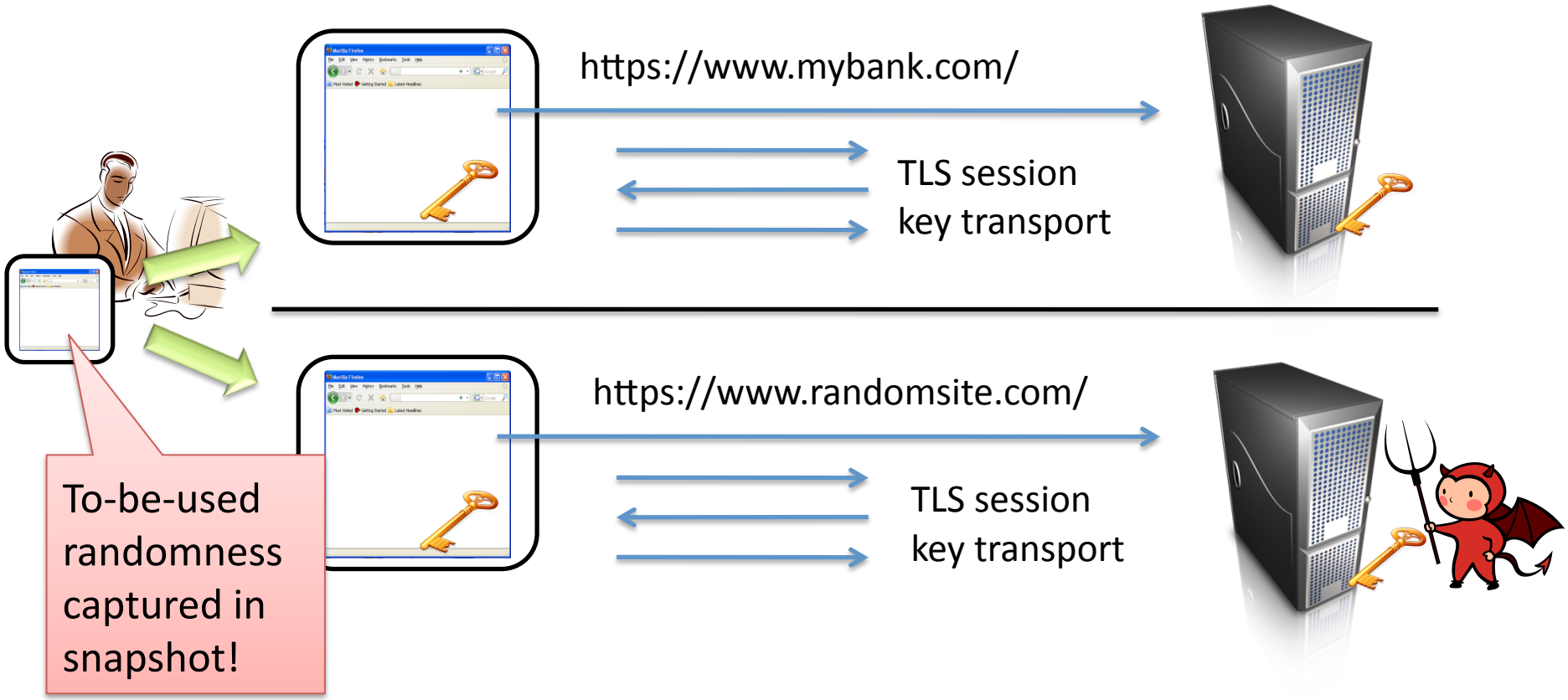
Problems might stem from reuse of security-critical state



Hypothetical example:
reuse of a one-time-only cryptographic key

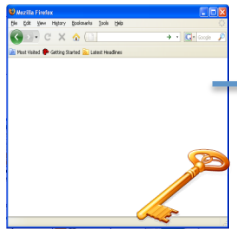
We show vulnerabilities exist in practice:

[R., Yilek – 2010]



Browser sends first session's secret key material to next site visited after reset

Recent versions of Firefox, Chrome allow session compromise attacks (we notified developers) in VMWare Server 1.0, VirtualBox 3.0



https://www.mybank.com/

TLS session
key transport

A logical timeline of events

User launches browser in VM

Randomness gathered by browser random number generator (RNG)

User snapshots VM

Snapshot later run.

User requests https page

Randomness used by TLS key transport

TLS key transport client

pk_{server} → Encrypt

ctxt

ctxt sent to server

A second run from snapshot leads to same secret key being sent to (different) server

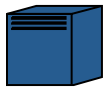
TLS Client	Guest OS	Potential session compromise?	Comments
Firefox 3.5	Windows XP	Yes	<100 mouse events
Chrome 3.0	Windows XP	No	Same secret key material to same server
IE 6.0	Windows XP	No	Same secret key material to same server
Safari 4.0	Windows XP	No	--
Firefox 3.0	Ubuntu Linux	Yes	<100 mouse events
Chrome 4.0	Ubuntu Linux	Yes	--

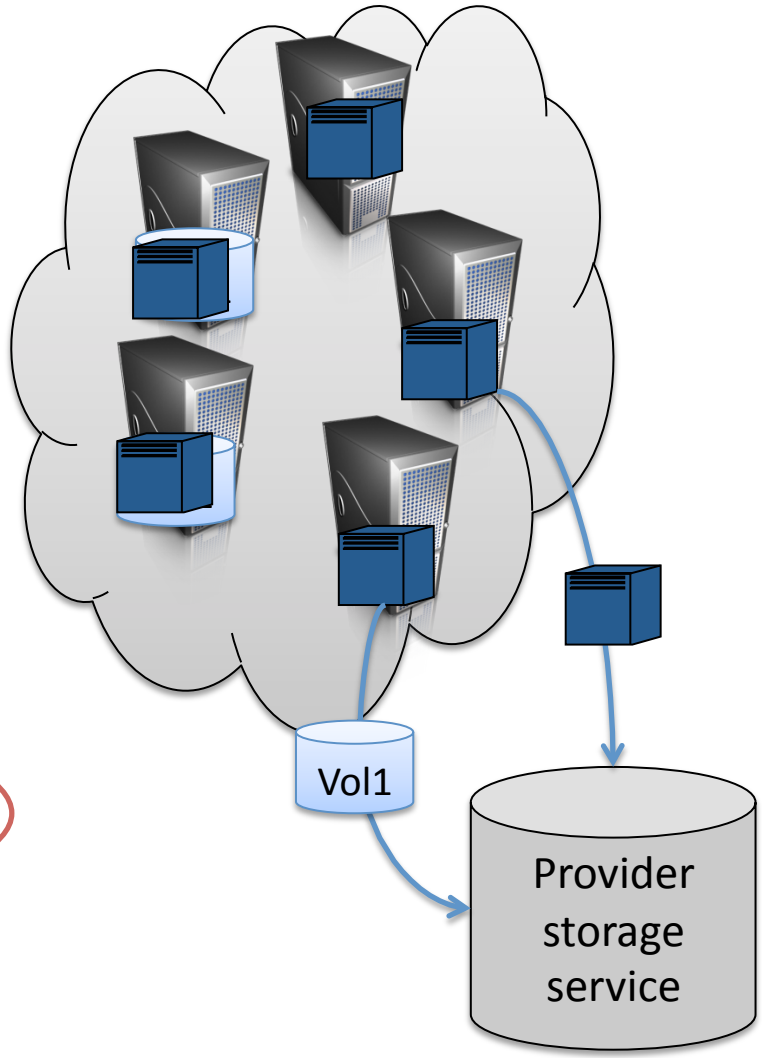
Results hold for both the VMWare Server 1.0 and VirtualBox 3.0 virtual machine managers

Potential for problems anywhere snapshots used

User A

virtual machines (VMs)



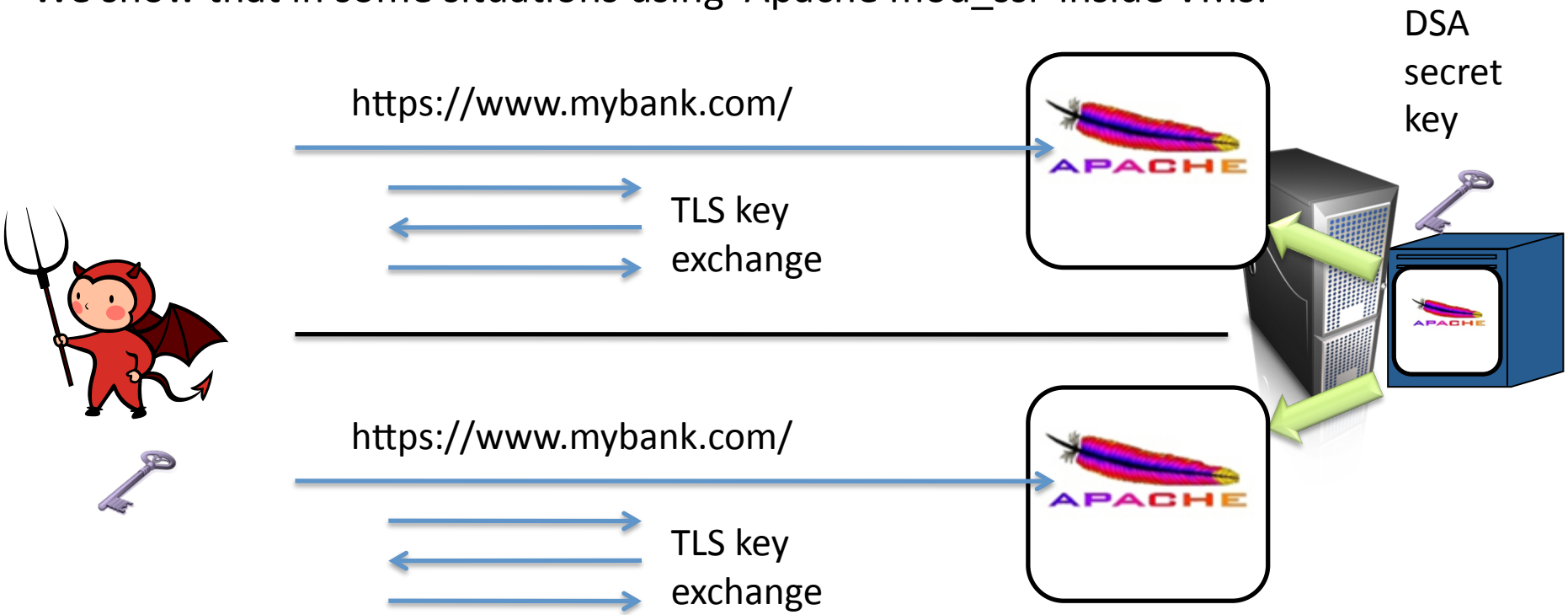


Volume snapshots save persistent storage

Full-state snapshots save entire state of VM

Potential for problems anywhere snapshots used

We show that in some situations using Apache mod_ssl inside VMs:

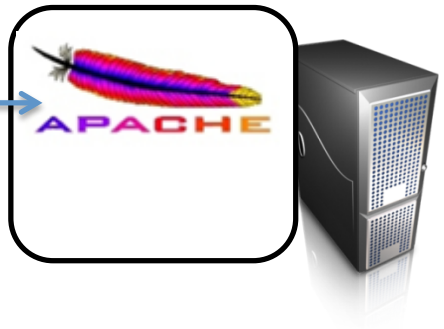


Key extraction might be possible

DSA secret key allows impersonating server

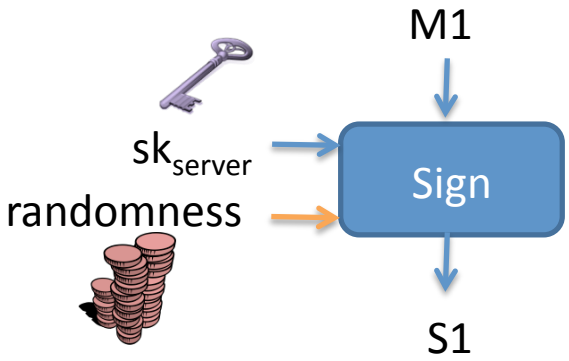
https://www.mybank.com/

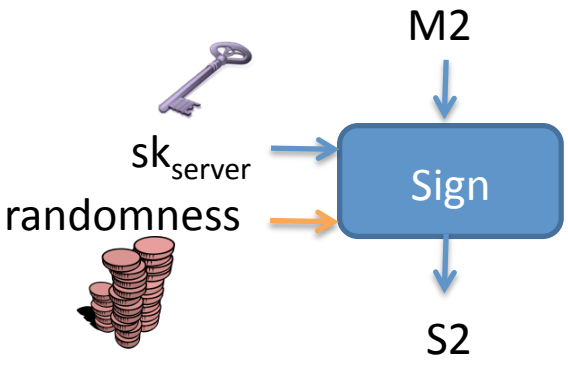
TLS key exchange



DSA secret key

A few minutes with pen & paper --or-- just check wikipedia article on DSA:





If adversary gets (M1,S1) and (M2,S2) then adversary easily computes sk_{server}

https://www.mybank.com/

TLS key exchange

DSA secret key

A logical timeline of events

Adminstrator launches Apache daemon

Init childs' RNGs

Apache children processes forked

User snapshots VM

Snapshot later run.

HTTPS request handled by a child

Randomness generated & used to sign

RNG updated with time, child PID, stack

DSA signing

key exch msg

sk_{server}

Sign

sig

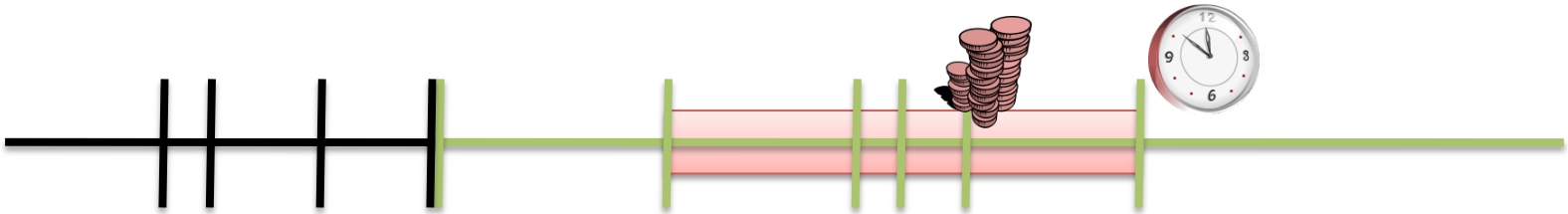
sig sent to client

https://www.mybank.com/

TLS key exchange

DSA secret key

A logical timeline of events



VM clock synch

HTTPS request handled by a child

Guests' network up

Randomness generated & used to sign

User snapshots VM

Snapshot later run.

RNG updated with time, child PID, stack

DSA signing

key exch msg

sk_{server}

Sign

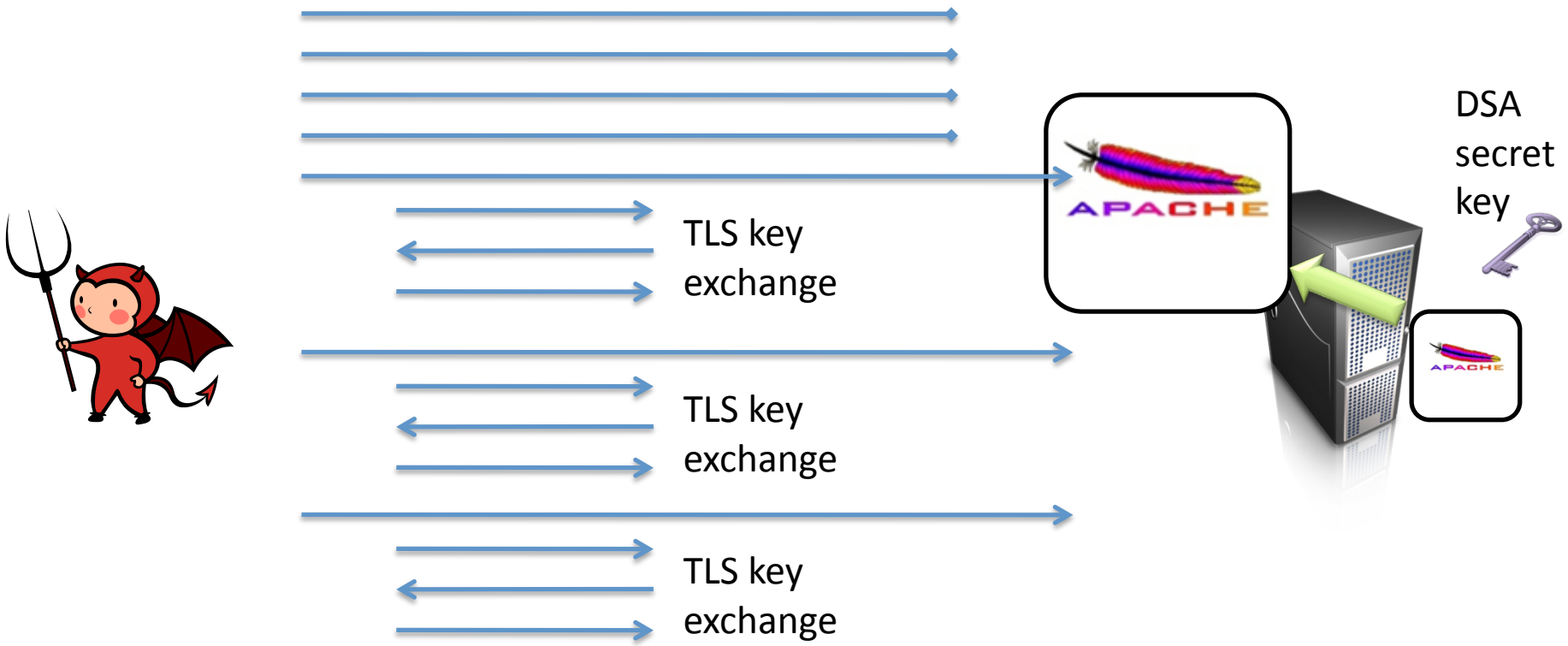
sig

sig sent to client

VM managers we looked at synchronized guest's time with Internet

This would seem to imply that DSA randomness would be different each time

Experimenting with DSA key extraction



This is one trial.

- 5 trials w/ rebooting physical server
- 5 trials w/o rebooting physical server

Looked for reuse of randomness across pairs of successful connections

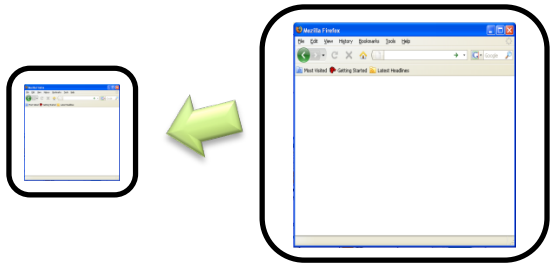
Repeat for both VMMs

Experimenting with DSA key extraction

VMM	Time sync?	Always reboot physical machine?	# pairs w/ repeat sesion IDs	# pairs w/ DSA key extractable
VirtualBox	Yes	No	10/10	10/10
VirtualBox	Yes	Yes	10/10	10/10
VMWare	Yes	No	0/10	0/10
VMWare	Yes	Yes	4/10	3/10
VMWare	No	No	6/10	6/10
VMWare	No	Yes	3/10	1/10

Problems at the intersection of technologies

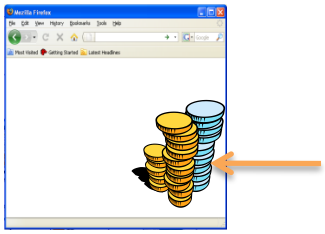
virtualization



- Snapshot technology allows freezing VM at arbitrary point

- Transparent to guest

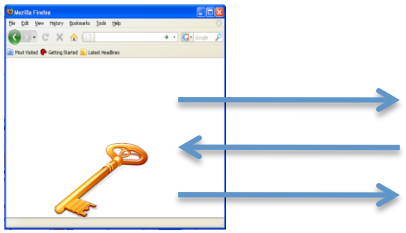
random number generation



- Applications often cache randomness for later use

- Applications unaware of snapshots

cryptography



- Crypto schemes fail spectacularly when RNGs fail

Applications not designed for resets.
Other security problems lurking?

Crypto operations fail spectacularly given bad randomness

Example situation:	randomness quality:	traditional crypto:	hedged crypto:
Proper RNG	Good	Strongest	Strongest
VM resets	Repeated	No security	Stronger
Debian OpenSSL bug	Predictable	No security	Strong

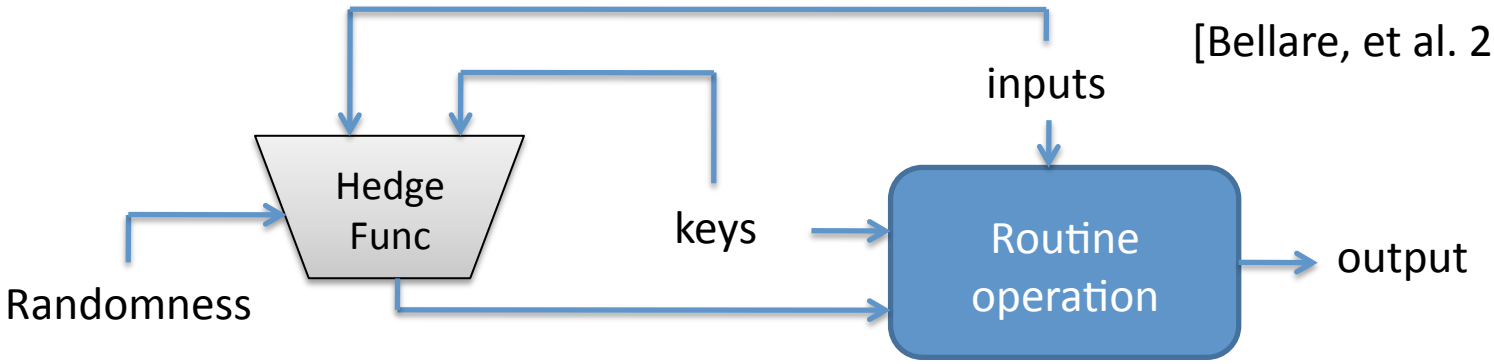
Hedged cryptography

[Bellare, Brakerski, Naor, R., Segev, Shacham, Yilek 2009]

Cryptographic operations should be as-secure-as-possible
in face of bad randomness

General hedging framework [R., Yilek 2010]

Integrates approaches from
[Bellare, et al. 2009] [Yilek 2010]



Hedging is backwards-compatible, allowing immediate deployability

Hedging does not solve RNG failures, but provides improved defense-in-depth

Today's talk in one slide

Third-party clouds:

"cloud cartography" to map internal infrastructure

get malicious VM on same physical server as victim

side-channels **might** leak confidential data of victim

Exploiting a placement vulnerability:
knowingly getting attack VM on server of victim

Joint with Eran Tromer, Hovav Shacham, Stefan Savage

Virtual machine snapshot technology:

run a VM twice from same snapshot

software re-uses cryptographic randomness

expose TLS sessions or steal TLS server secret key

Joint with Scott Yilek

Exploiting a reset vulnerability:
software unaware of resets, crypto fragile

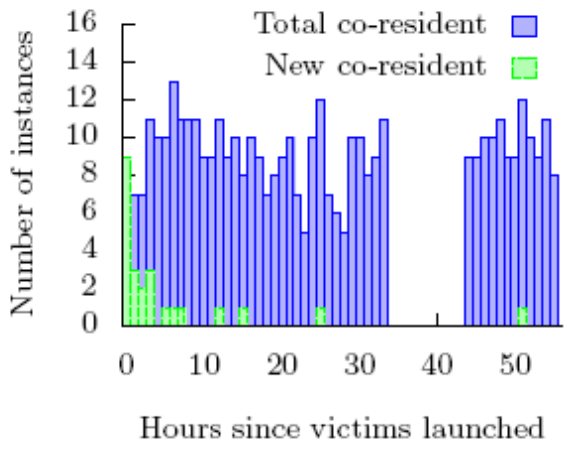
Achieving co-residence

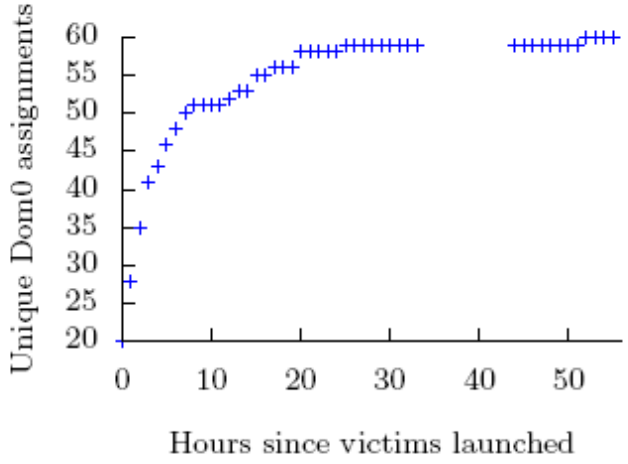
Instance flooding near target launch abuses parallel placement locality

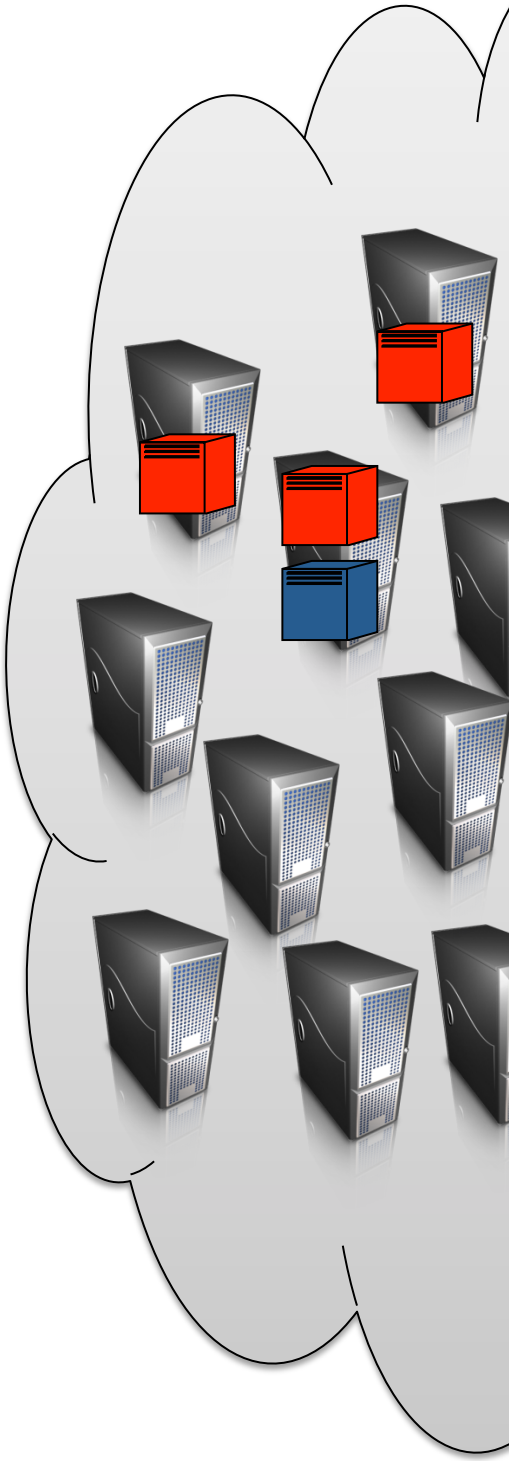
How long is parallel placement locality good for?

Experiment:

40 “target” VMs (across two accounts)
20 “attack” VMs launched hourly





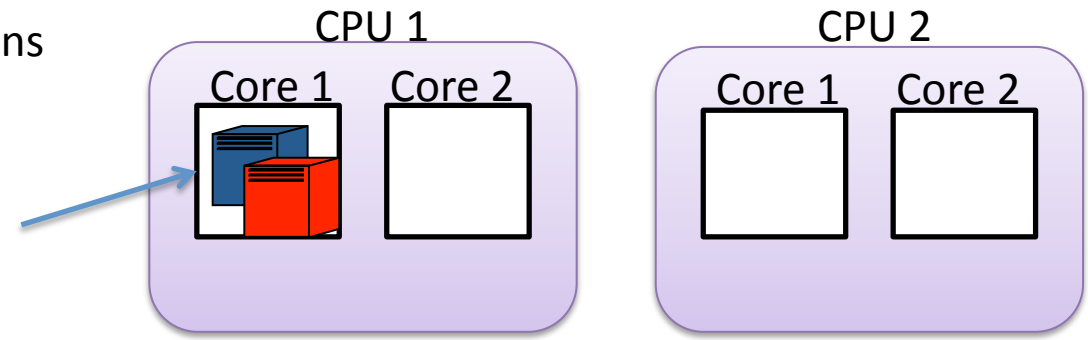


More on cache-based physical channels

Keystroke timing in experimental testbed similar to EC2 m1.small instances

AMD Opterons

VMs pinned to core



If VMs pinned to same core, then cache-load measurements allow cross-VM keystroke detection

Keystroke timing of this form might be sufficient for the password recovery attacks of [Song, Wagner, Tian 01]

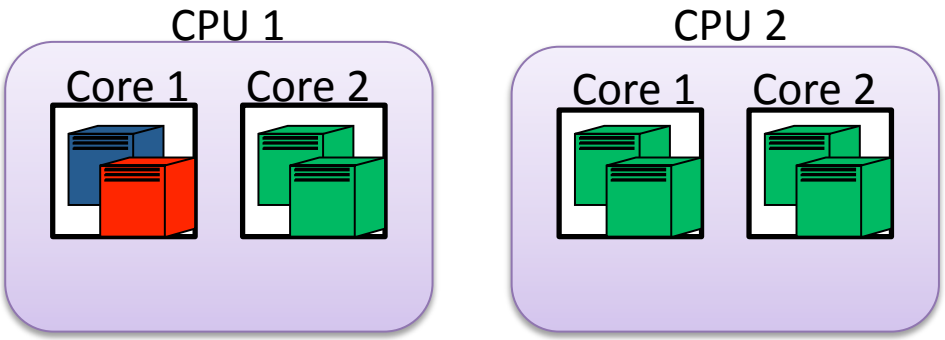
Cryptographic side-channels?

Cache-based side channels shown to leak RSA, AES keys [B05,P05,OST06] in non-VM settings

Translating such attacks to cross-VM setting faces hurdles:

Core migration
Noise due to other VMs
No hyperthreading
Double indirection of memory addresses
….

} Fine-grained side channels challenging



Open question: realizing such attacks in cloud setting

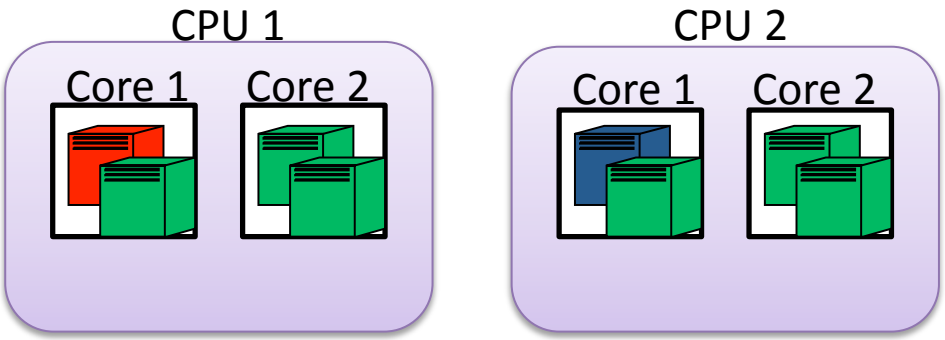
Cryptographic side-channels?

Cache-based side channels shown to leak RSA, AES keys [B05,P05,OST06] in non-VM settings

Translating such attacks to cross-VM setting faces hurdles:

Core migration
Noise due to other VMs
No hyperthreading
Double indirection of memory addresses
….

} Fine-grained side channels challenging



Open question: realizing such attacks in cloud setting

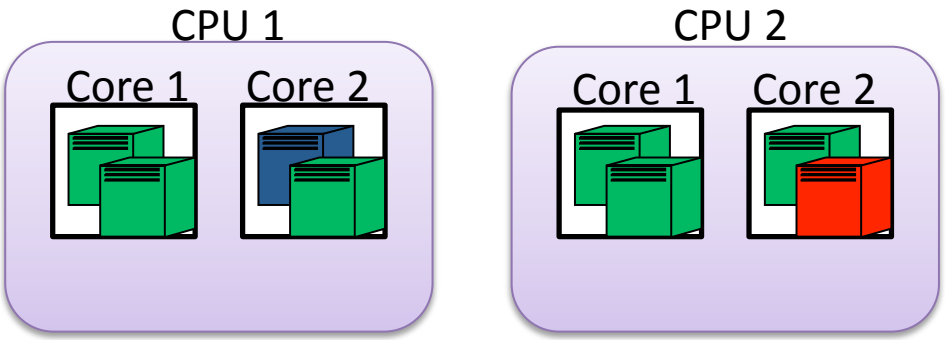
Cryptographic side-channels?

Cache-based side channels shown to leak RSA, AES keys [B05,P05,OST06] in non-VM settings

Translating such attacks to cross-VM setting faces hurdles:

Core migration
Noise due to other VMs
No hyperthreading
Double indirection of memory addresses
….

} Fine-grained side channels challenging



Open question: realizing such attacks in cloud setting

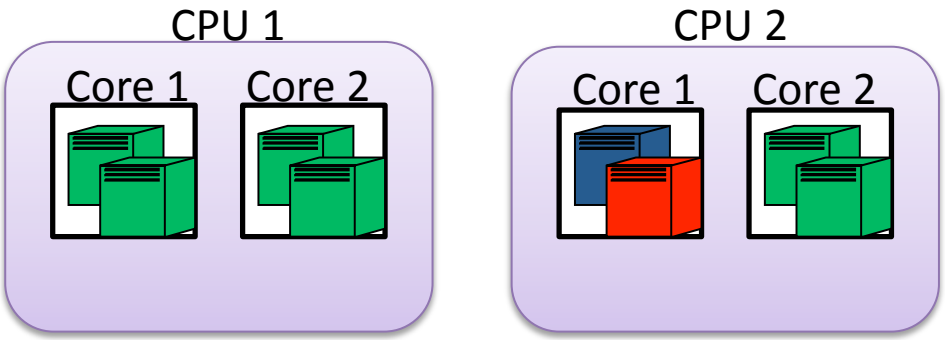
Cryptographic side-channels?

Cache-based side channels shown to leak RSA, AES keys [B05,P05,OST06] in non-VM settings

Translating such attacks to cross-VM setting faces hurdles:

Core migration
Noise due to other VMs
No hyperthreading
Double indirection of memory addresses
….

} Fine-grained side channels challenging



Open question: realizing such attacks in cloud setting