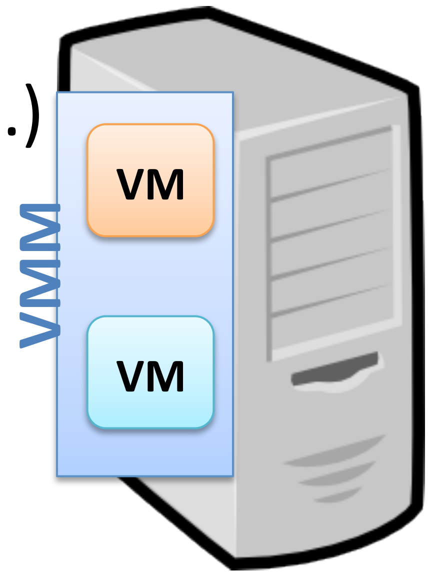
Scheduler-based Defenses against Cross-VM Side-channels

Venkat(anathan) Varadarajan,
Thomas Ristenpart,
and Michael Swift

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON
DEPARTMENT OF COMPUTER SCIENCES

Shared Resources and Isolation

- IaaS Public clouds (Amazon EC2, Azure, etc.)
  - Multi-tenancy
- VMs share many resources
  - CPU, cache, memory, disk, network, etc.
- Virtual Machine Managers (VMM)
  - Goal: Provide Isolation
- Deployed VMMs don't perfectly isolate VMs
  - Side-channels [Ristenpart et al. '09, Zhang et al. '12]
  - Other attacks: Performance Degradation, RFA [Varadarajan et al. '12]



Example Cache Side-channel*

Control-flow Side-channel

- secret key bits directly affect instruction sequence executed

Modular Exponentiation Algorithm:

```
SQUAREMULT(x, e, N):
Let e_n , ..., e_1 be the bits of e
y ← 1
for i = n down to 1 do
    y ← SQUARE(y)
    y ← MODREDUCE(y, N)
    if e_i = 1 then
        y ← MULT(y, x)
        y ← MODREDUCE(y, N)
    end if
end for
return y
```

Secret

Executes when secret bit is ***either 0 or 1***

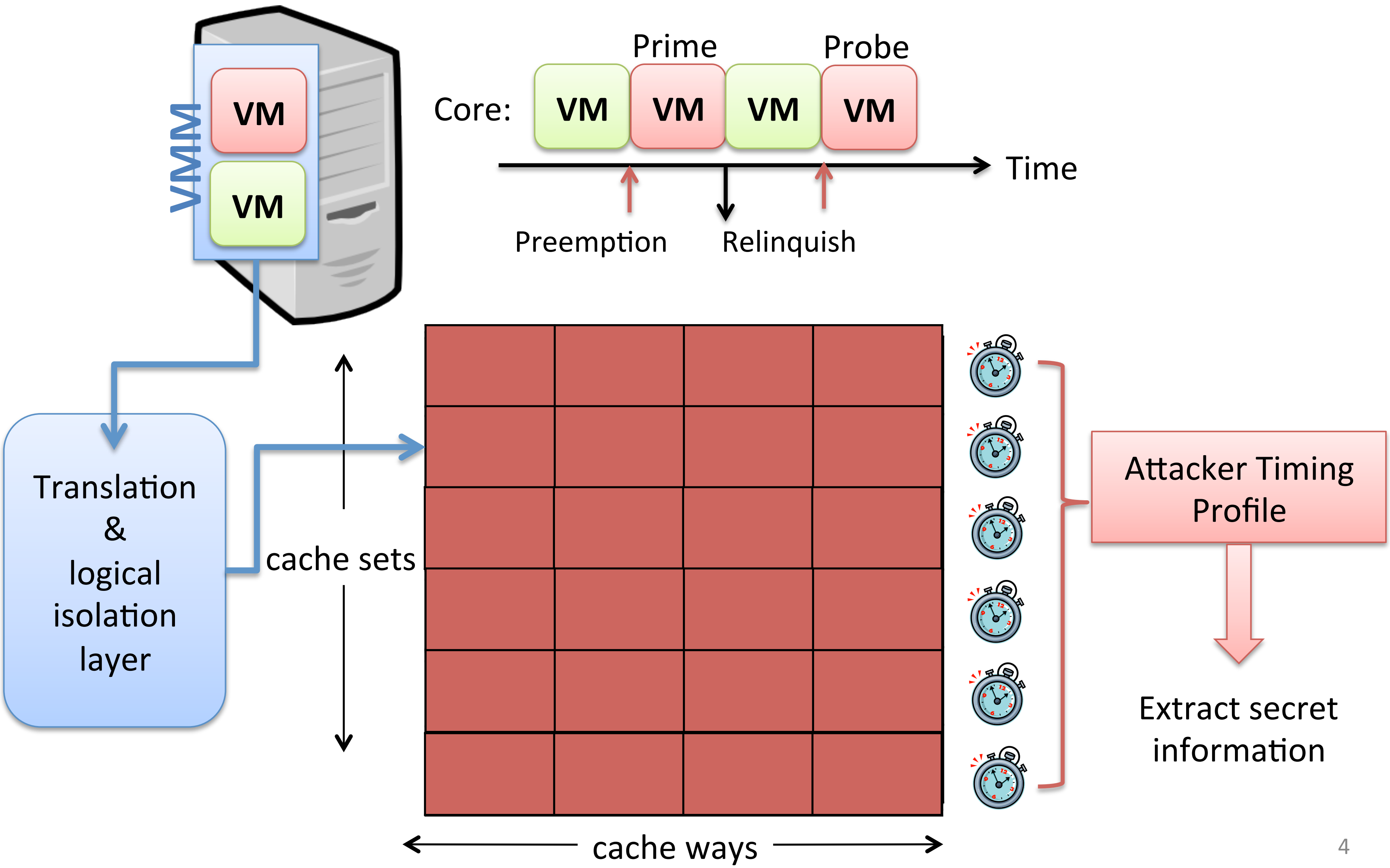
Executes ***only*** when secret key bit is 1

I-cache usage leaks secret

- Operations: Square (S), Reduce (R), and Multiply (M).
- e_i = 1 bit: S→R→M→R
- e_i = 0 bit: S→R (and, NOT followed by M→R).

* Zhang, Juels, Reiter and Ristenpart: Cross-VM Side-channel Attack, CCS 2012

Cache-based Side-channels



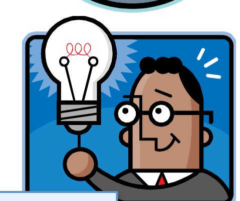
Defenses against Side-channels

Access-driven side-channel attacks rely on:

1. **Sharing**
   1.1. Resource Partitioning or Hard isolation
   Problems: low utilization, high service cost
   1.2. Specialized Hardware
   Problems: high cost, non-commodity

2. **Access to high-resolution timers**
   – Reduce resolution, add noise
   – Problems: Loss of feature or high overhead

3. **Vulnerabilities in CPU scheduler**
   – Managing preemptions – *soft isolation*



Scheduler is *exploited* in side-channels →
NO prior research on secure scheduler designs!

Soft Isolation Mechanisms

Goals:

1. Reduce risk of sharing

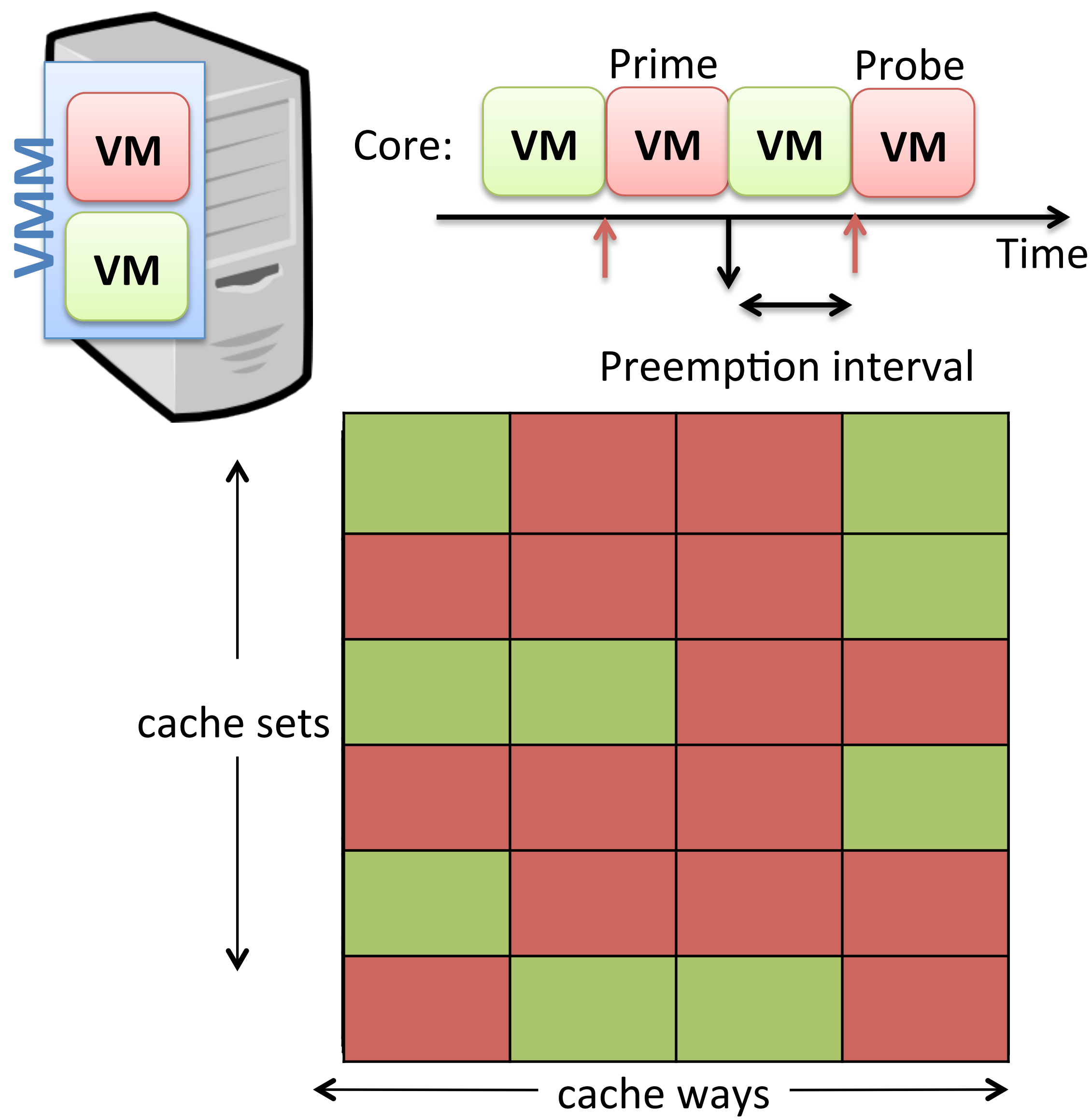
2. Monotonically improve security

3. Low performance overhead

Challenges:

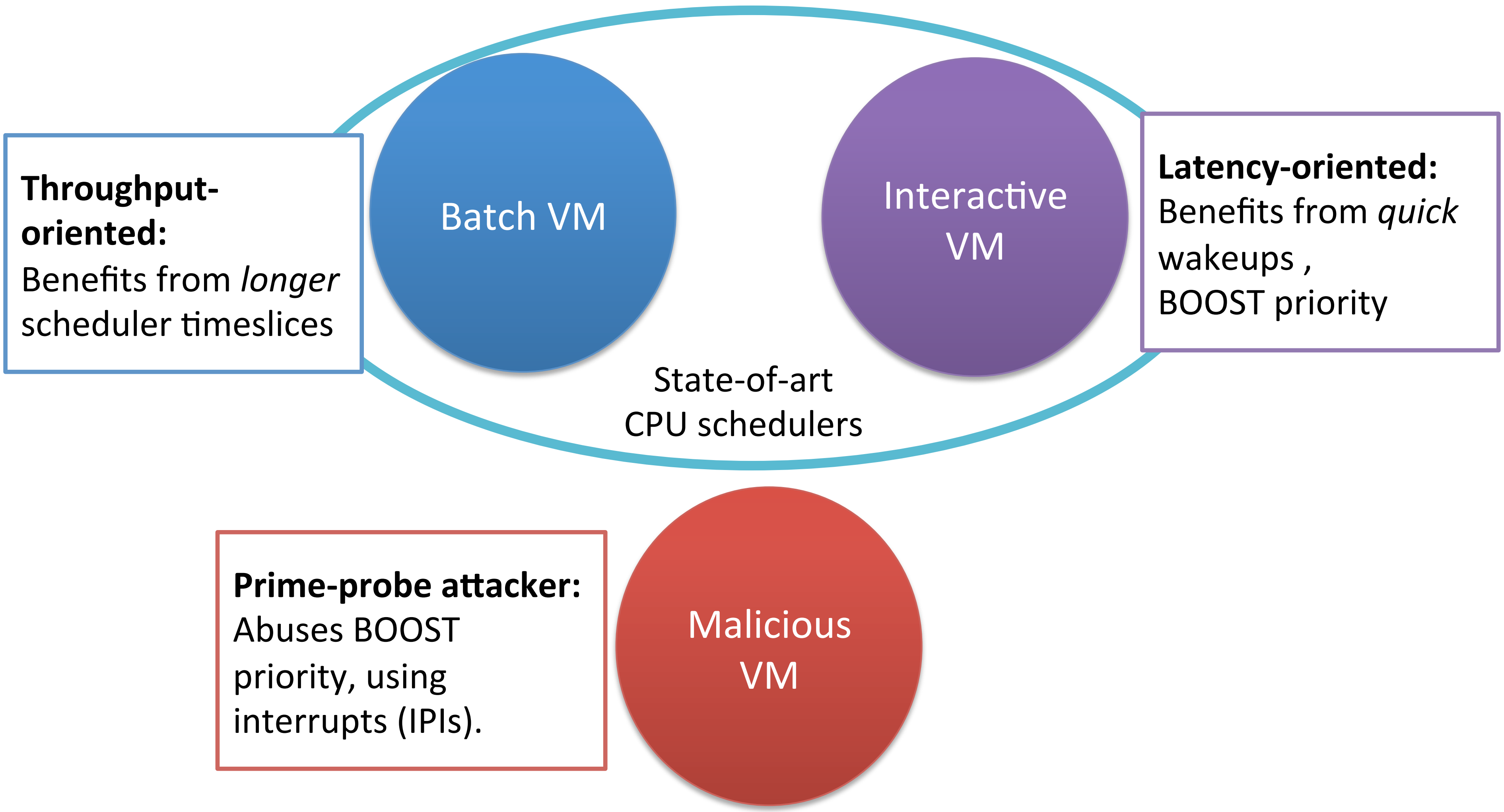
- Unintuitive impacts of scheduler changes
- No standard benchmarks
- No security evaluation methods

Prime-Probe Side-channel Attack



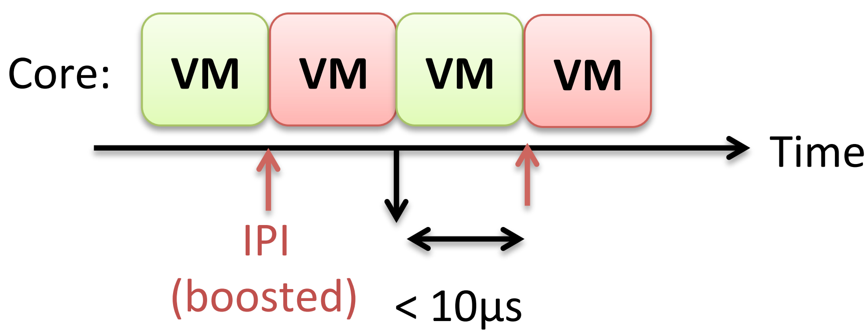
Shorter the preemption interval
→
more (or any) information leakage bandwidth

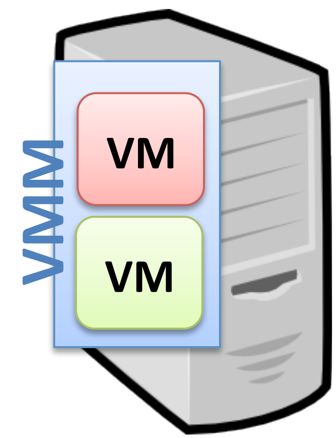
Background: Xen CPU Scheduler



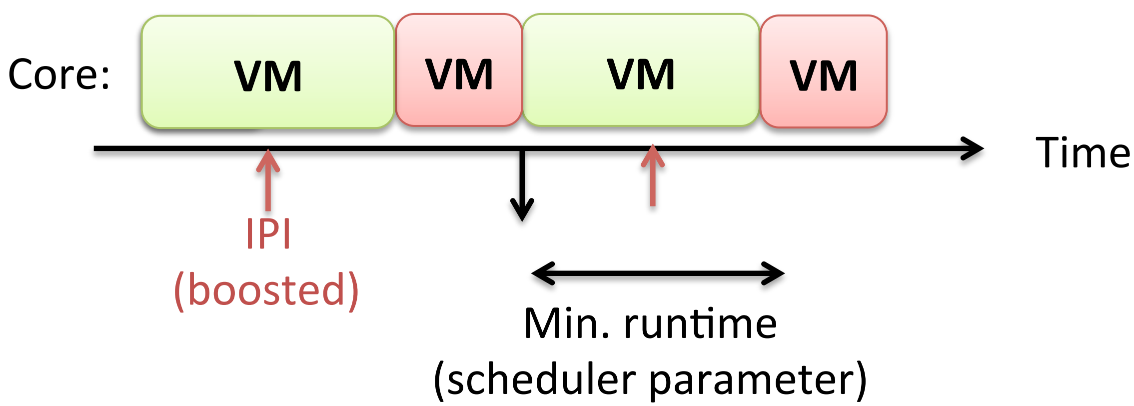
Soft-Isolation: Minimum Runtime Guarantee

Under Zhang et al. attack setting:





Under Minimum RunTime (MRT) guarantee:



Introduced in Xen and Linux for *performance improvement for batch VMs*

What about security properties?

Evaluation of MRT

1. Does MRT make existing side-channels harder?

2. What is the scope of security against side-channels for all victims?

3. How much performance overhead for latency-sensitive applications with MRT?

Experiment Setting

Machine Configuration:

Machine	Intel Xeon E5645, 2.4GHz, 6 cores, single package
Memory Hierarchy	Private 32KB L1 (I- and D-Cache), 256KB unified L2, 12MB shared L3 & 16GB DDR3 RAM.

Xen Configuration:

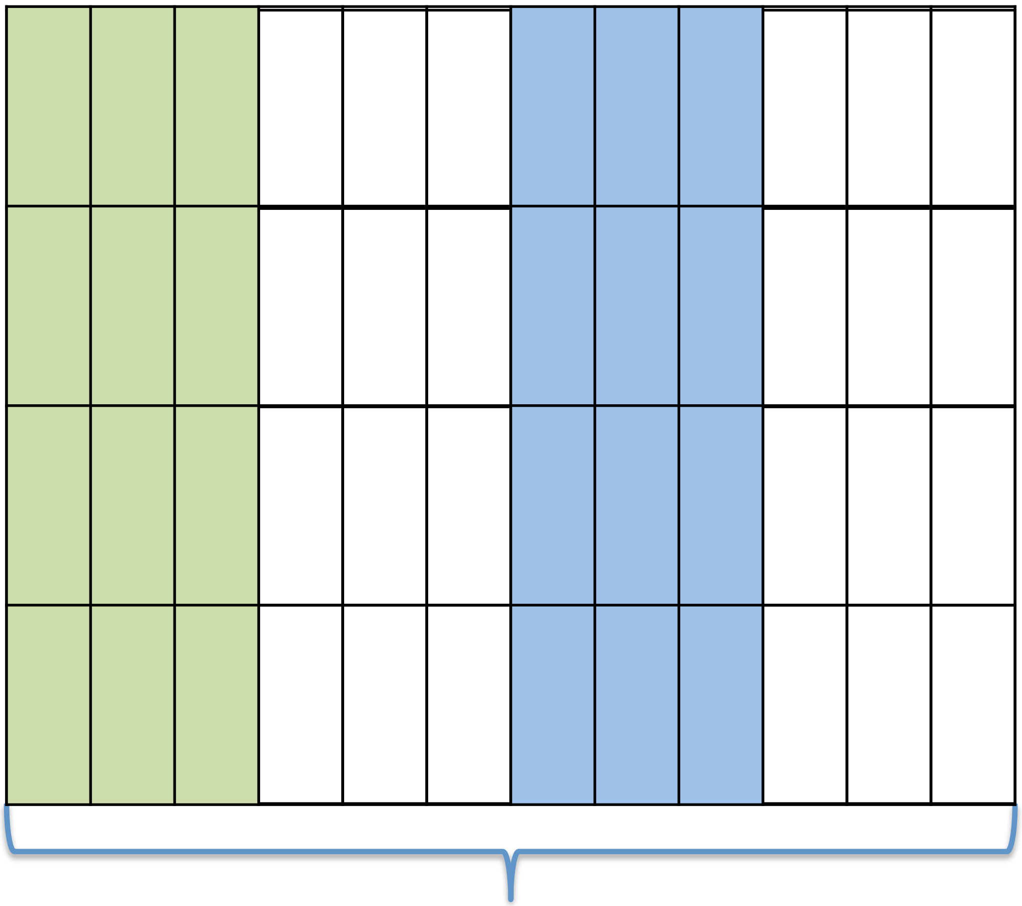
Xen Version	4.2.1
Scheduler	Credit Scheduler 1
Configuration (Non-work conserving)	40% cap on DomU VCPUs with equal weight
# VMs	6
# VCPUs per VM	2

Similar to setting used by Zhang et al.

Prime-Probe Timing Profile

A Sample Side-channel Victim

L1 I-Cache



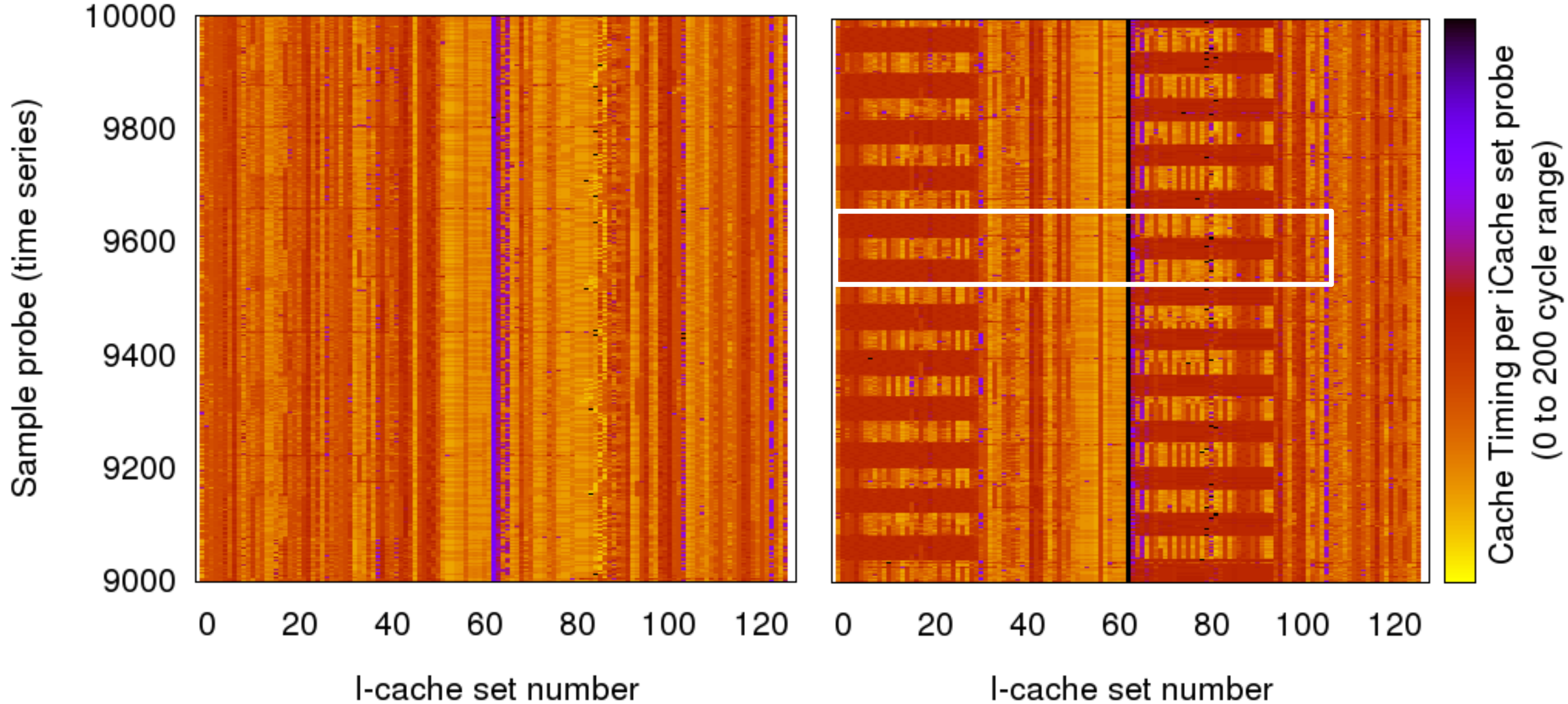
Cache sets

Victim Pseudo Code

```
if subset(secret)= X
then
  for( sometime )
  do
    instr. in green
  endfor
fi
if subset(secret)= Y
then
  for( sometime )
  do
    instr. in blue
  endfor
fi
```

For simplicity:
`secret = XYXY…`

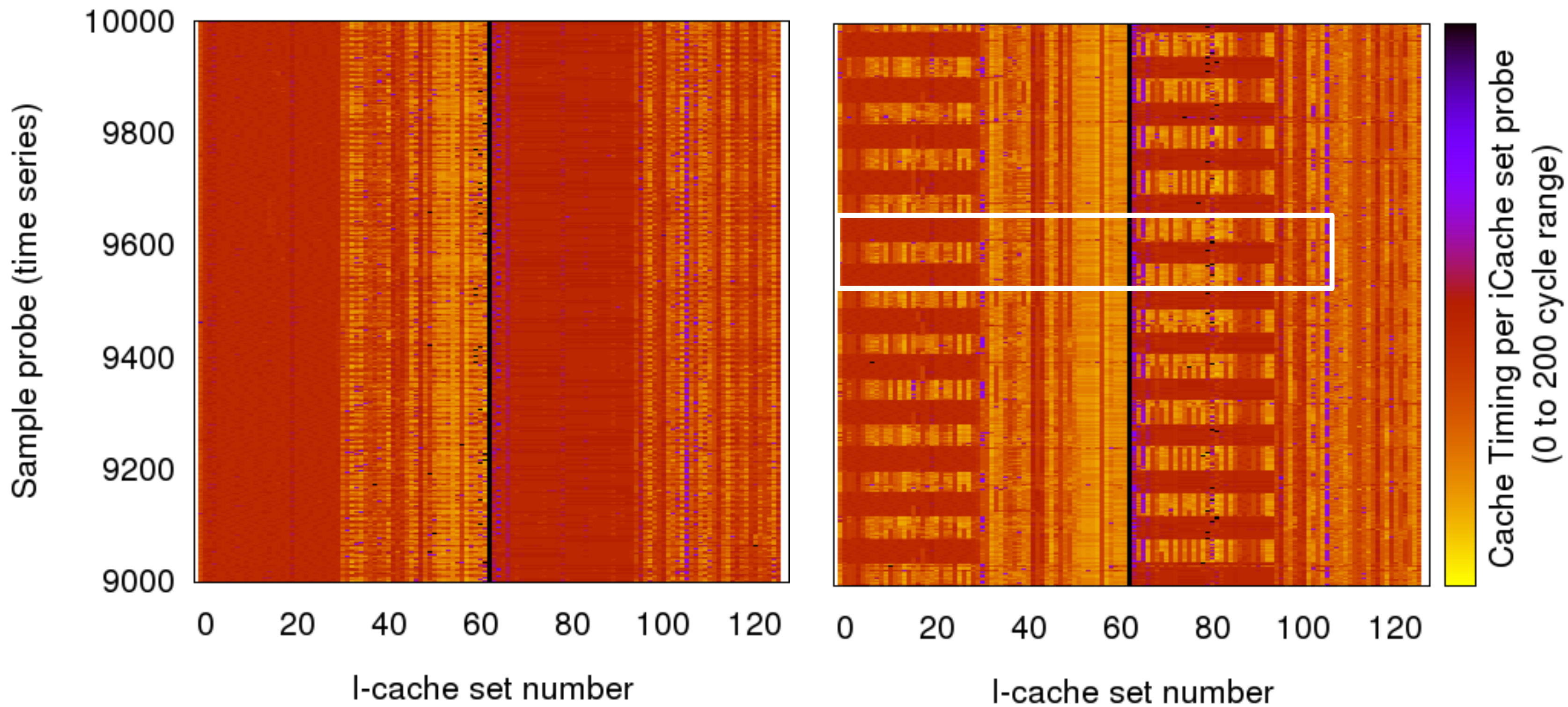
Prime-Probe Timing Profile



Idle Victim VM

Model Victim VM Under **Zero-MRT**

Prime-Probe Timing Profile

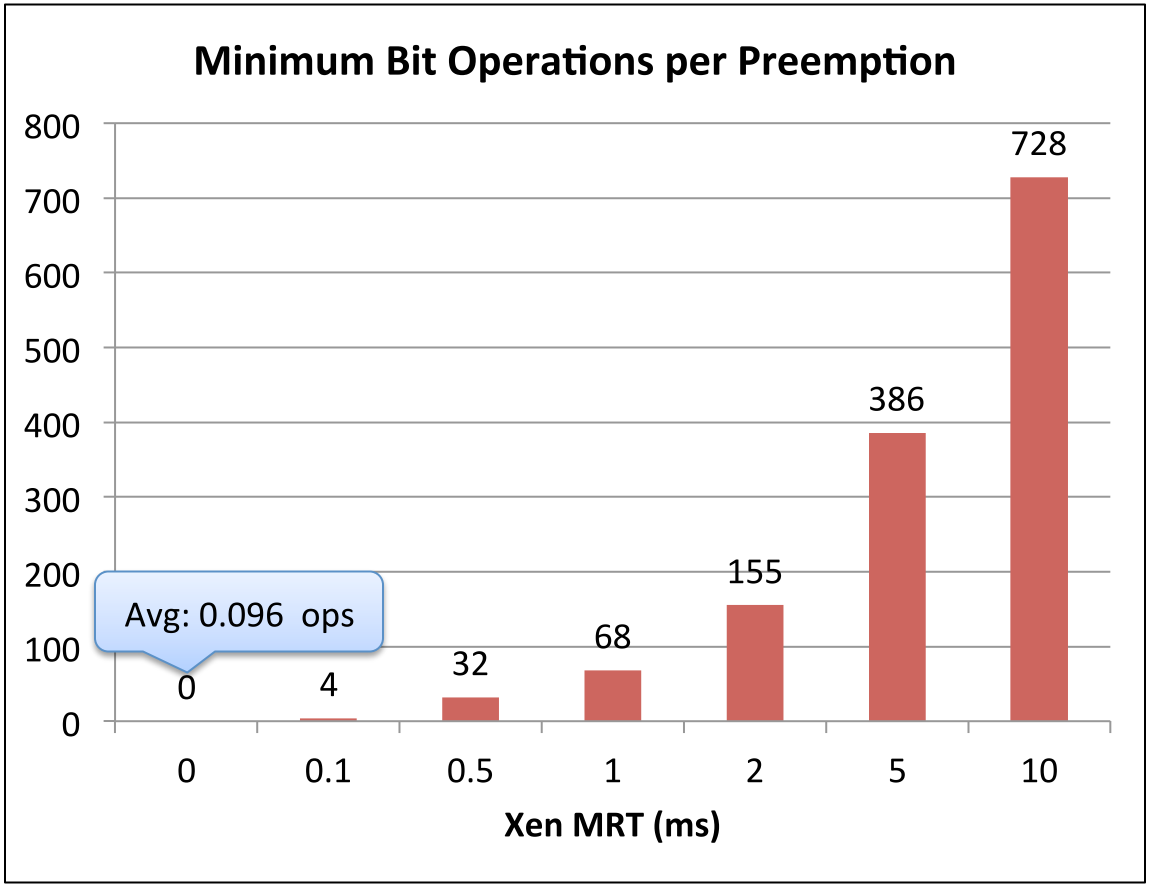


Model Victim VM Under **1ms MRT**

Model Victim VM Under **Zero-MRT**

A simple scheduler mechanism → known attacks are harder

Elgamal Victim: Information Leakage



Elgamal Side-channel rely on consecutive redundant observations for noise-reduction

Security Limitations of MRT

1. Slower victims could still leak!
2. Only applicable to sub-class of side-channels, and to virtualized setting,
3. Interactive VMs that voluntarily relinquish the CPU
   are still vulnerable!

Modular Exponentiation Algorithm:

```
SQUAREMULT(x, e, N):
Let e_n , ..., e_1 be the bits of e
y ← 1
for i = n down to 1 do
    y ← SQUARE(y)
    y ← MODREDUCE(y, N)
    if e_i = 1 then
        y ← MULT(y, x)
        y ← MODREDUCE(y, N)
    end if
end for
return y
```

Per-core Shared State-Cleansing

Performance Evaluation

1. What is the overhead of turning on MRT?
   - A 0.3% improvement for batch workloads
   - On average 4% and at worst 7% overhead on 95th percentile latency
2. What is the overhead of cleansing on *latency sensitive* real-world applications?
   - adds a overhead of 10μs for latency sensitive workloads,
   - At worst a 80-100μs on 95th percentile latency

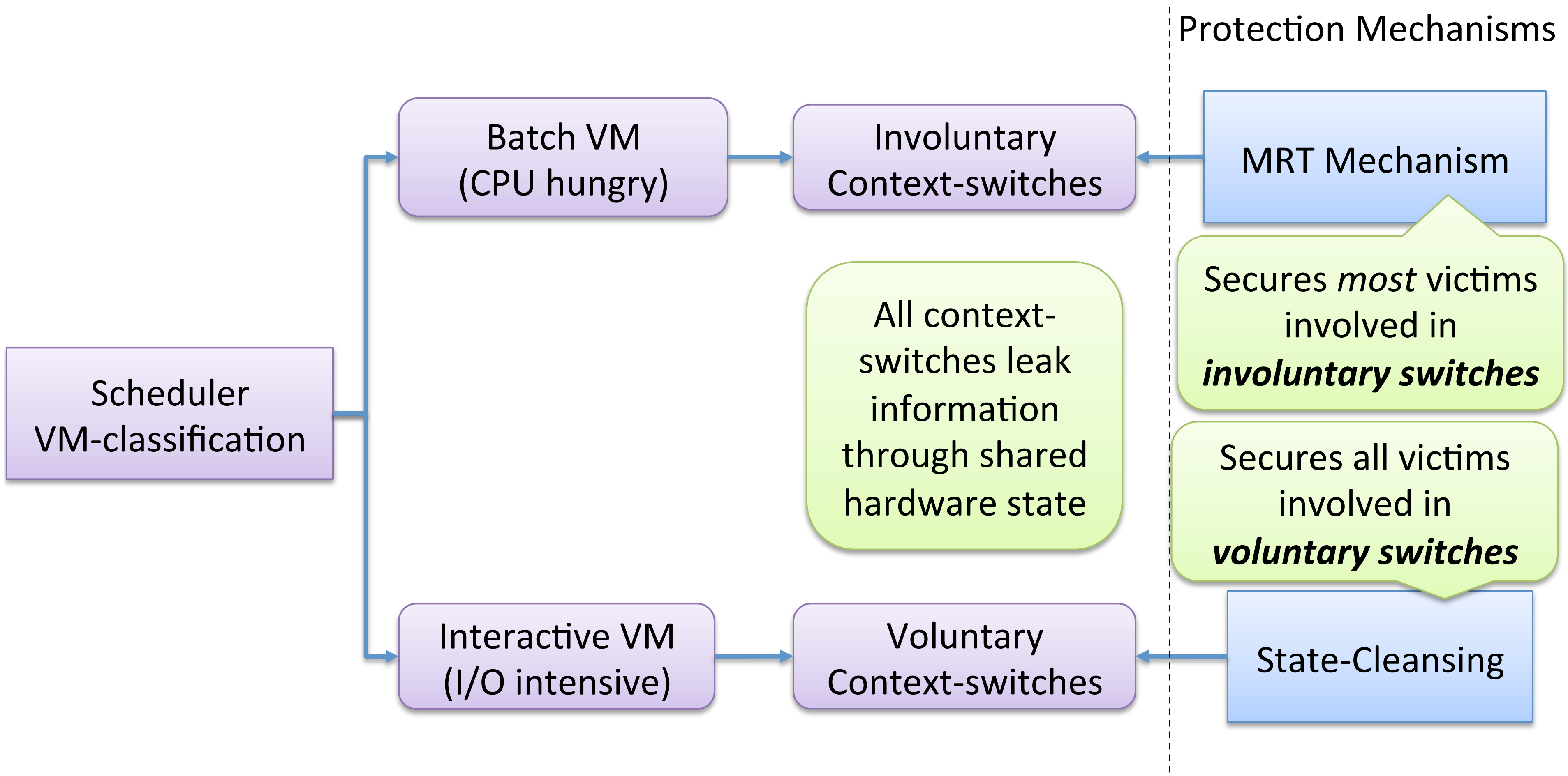
Conclusion

- Current state-of-the-art CPU schedulers *do not account for malicious users,*
- *First-of-its-kind* security analysis of schedulers
- Introduce new design paradigm: *soft-isolation*

Future work

- Model preemption-driven side-channels and estimate theoretical strength of MRT mechanism
- MRT-like mechanism for other system-level shared resources.

A Simple, Secure Scheduler Design



Related Work

1. Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds." In CCS '09.
2. Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. "Cross-VM side channels and their use to extract private keys." In CCS'12.
3. Venkatanathan Varadarajan, Thawan Kooburat, Benjamin Farley, Thomas Ristenpart, and Michael M. Swift. "Resource-freeing attacks: improve your cloud performance (at your neighbor's expense)." In CCS'12

Questions?

Contact: venkatv@cs.wisc.edu