

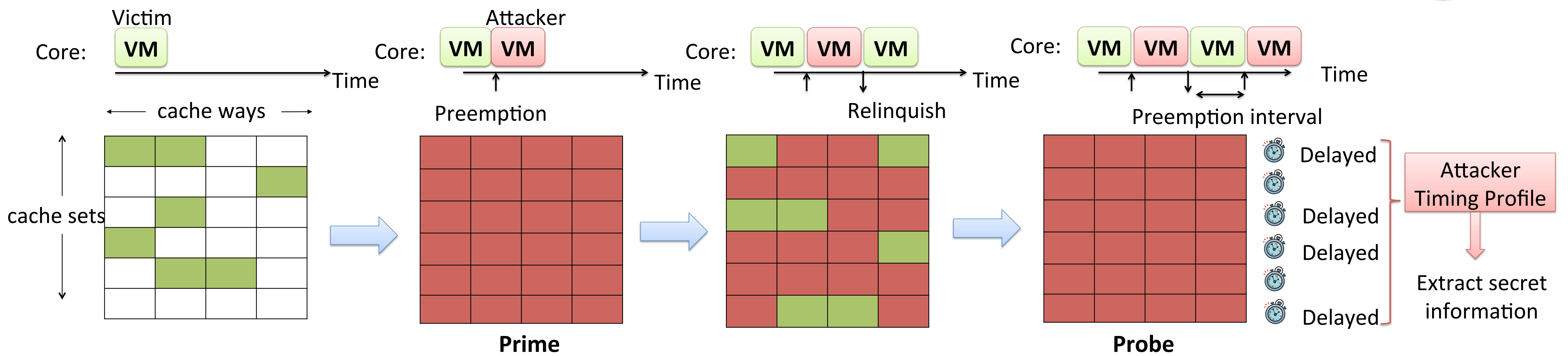
Venkatanathan Varadarajan, Thomas Ristenpart and Michael M. Swift  
University of Wisconsin-Madison

## Problem: Cross-VM Side-Channels

### Multi-Tenancy in Public IaaS Clouds

Two customers sharing the same physical host managed by a hypervisor is common place.

EC2, Azure, Rackspace, etc.



### An Example Control-flow Side-channel

- Libcrypt* implements a square multiply algorithm used by many cryptographic functions: *RSA*, *Elgamal*, etc.
- Leaks secrets via L-cache usage,
- Zhang et al. [CCS'12] in a lab setting extracted a 2048 bit secret key in 6 hours.

```
SQUAREMULT(x, e, N):
  Let en, ..., e1 be the bits of e ← Secret
  y ← 1
  for i = n down to 1 do
    y ← SQUARE(y)
    y ← MODREDUCE(y, N)
    if ei = 1 then
      y ← MULT(y, x)
      y ← MODREDUCE(y, N)
    end if
  end for
  return y
```

Executes when secret bit is either 0 or 1  
Executes only when secret key bit is 1

All context-switches leak information via shared hardware state

### Requirements for Successful Side-Channels:

- Shared hardware,
- Access to a high-precision timer, and
- Ability to preempt.

### Prior Defenses

- Resource Partitioning or Hard isolation:**  
Problems: low utilization, high service cost.
- Specialized Hardware:**  
Problems: high cost, non-commodity.
- Reduce Resolution of Timers:**  
Problems: Loss of feature or high overhead.

## Solution: Soft-Isolation of VMs

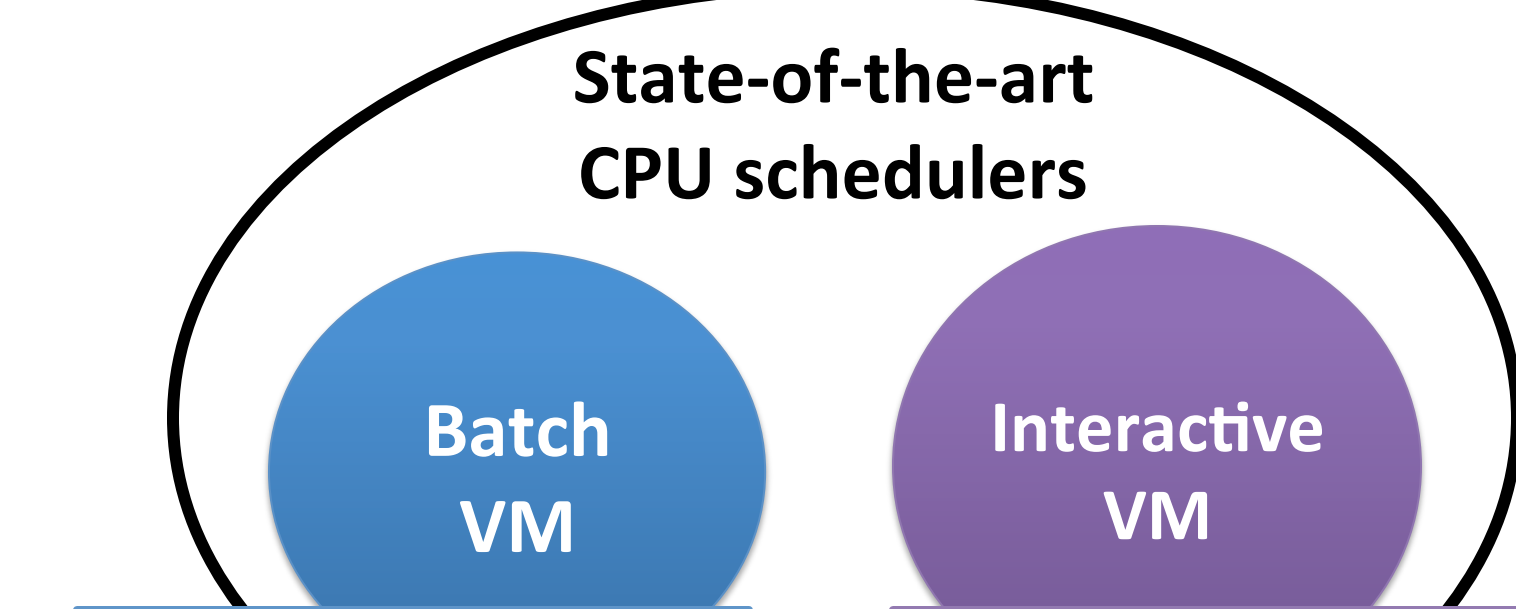
### Goals:

- Reduce risk of sharing,
- Monotonically improve security,
- Low performance overhead.

### Simple, Secure Scheduler Design

Two protection mechanisms:

- MRT mechanism** for securing *batch* VMs that are involved in *involuntary* context switches.
- Per-core State-Cleansing** for securing *interactive* VMs involved in *voluntary* context switches with *runtime* < *MRT*.



### Protection II: State-Cleansing

- Interactive VMs voluntarily give up CPU
- Not protected by MRT mechanism
- But, not performance sensitive to per-core state

### State-cleansing Routine:

```
000 <L13-0xd>:
  0: 8b 08      mov    (%rax),%ecx
  2: 85 c9      test  %ecx,%ecx
  4: 74 07      je     d <L13>
  6: 8b 08      mov    (%rax),%ecx
  8: 88 4d ff  mov    %c1,-0x1(%rbp)
  b: eb 05      jmp   12 <L14>

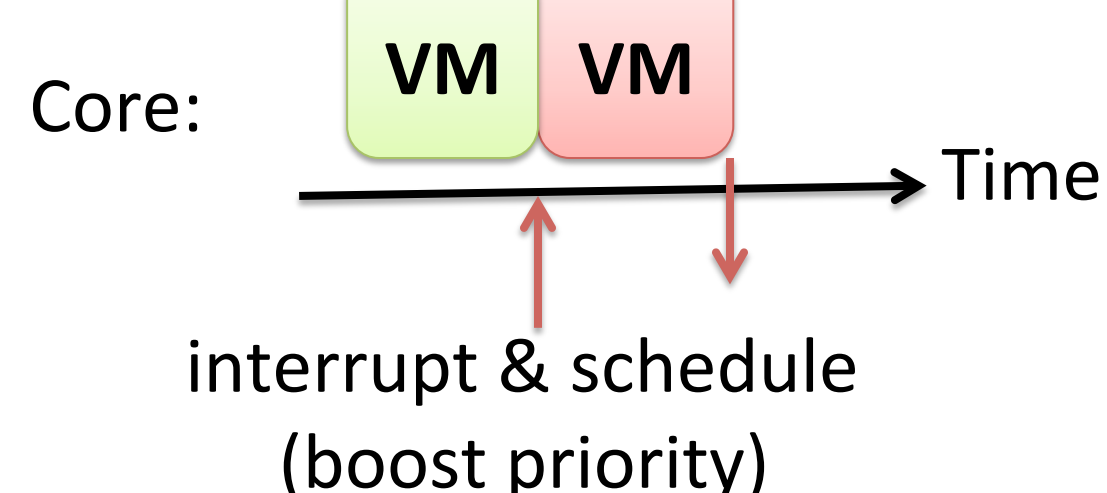
00d <L13>:
  d: 8b 08      mov    (%rax),%ecx
  f: 88 4d ff  mov    %c1,-0x1(%rbp)

012 <L14>:
  12: 48 8b 40 08 mov    0x8(%rax),%rax
  17: e9 e5 1f 00 jmpq  <next way in set>
```

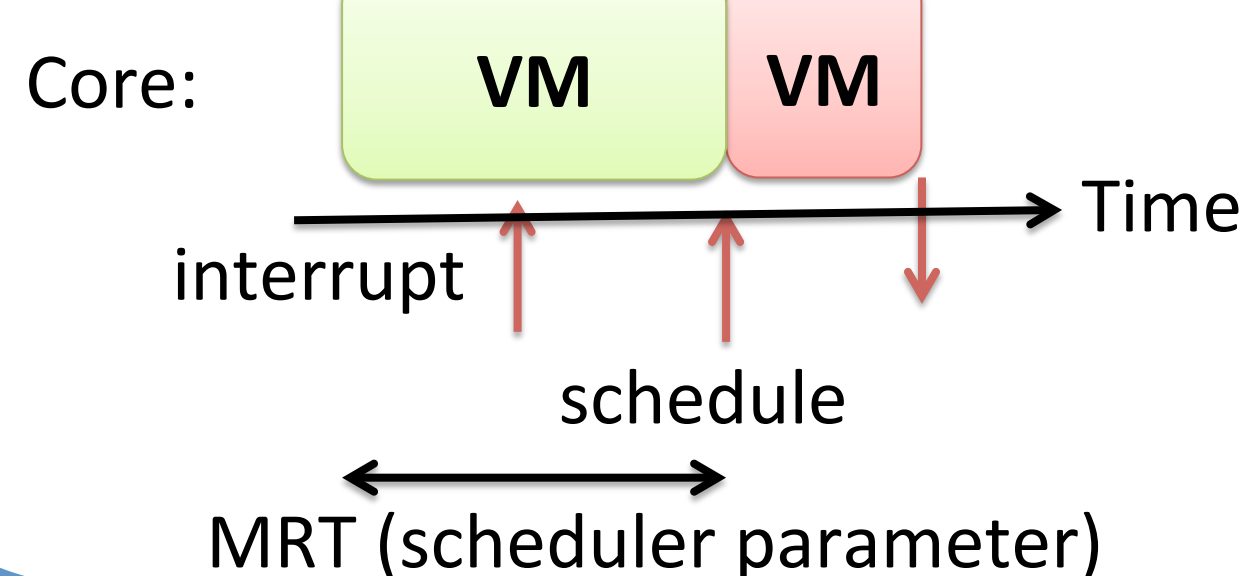
Branch predictor & data-cache access  
Move to next data-value in next way/set

### Protection I: MRT Mechanism

#### Existing Scenario



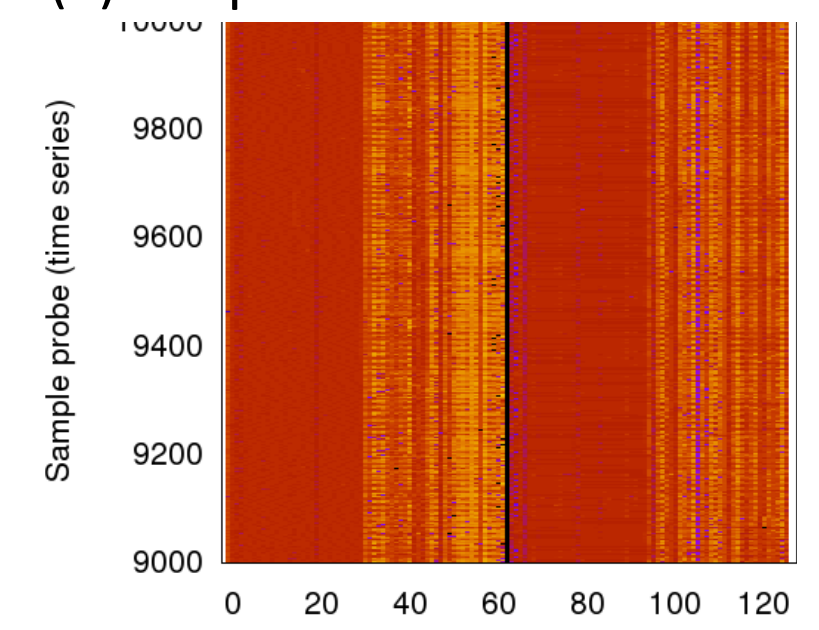
#### Minimum RunTime (MRT) Guarantee



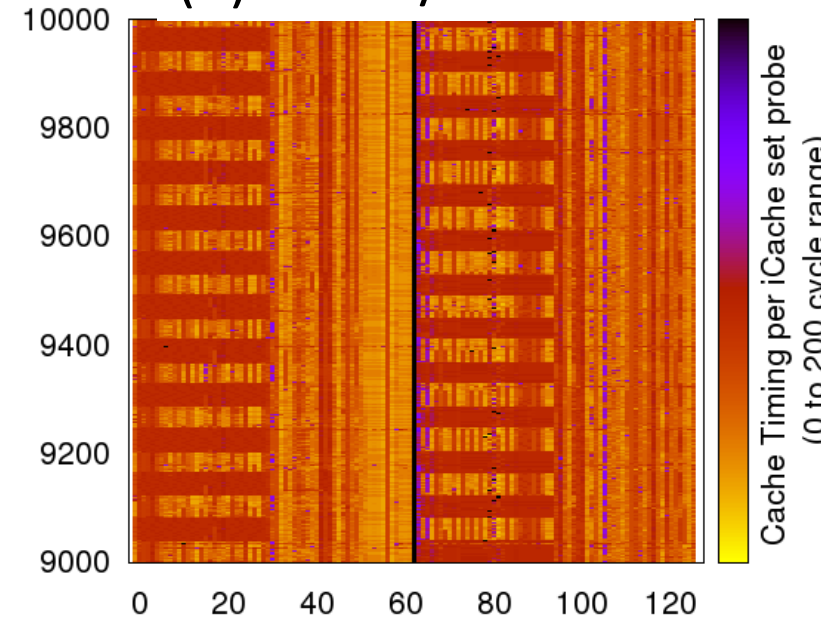
### Makes existing attacks harder

```
if subset(secret)= X then
  for( sometime )
    do
      instr. in 1/4th i-cache
    endfor
  fi
if subset(secret)= Y then
  for( sometime )
    do
      instr. in another 1/4th
    endfor
  fi
```

(a) Simple Victim Pseudocode



(b) Idle w/ No MRT



(d) 1ms MRT w/ victim

(c) No MRT w/ victim

### Results

#### Security

- 5ms MRT increases the mod-exp. bit operations per preemption to 386 from 0.096 bit ops with no MRT
- Overhead of MRT of 5ms:**
- 0.3% improvement for batch workloads,
- On average 4% and at worst 7% overhead on 95<sup>th</sup> percentile latency.

#### Overhead of State-Cleansing:

- An average overhead of 10-20 $\mu$ s,
- A worst case overhead of 80-100 $\mu$ s.