USENIX Security Symposium 2026

USENIX

# Exploiting Hidden Resource Contention in Selective Speculation Defenses

**Xiaoyu Cheng**, Fei Tong, Zhenyu Lei, Fang Jiang,
Zhe Zhou, Guang Cheng, Trevor E. Carlson


東南大學
SOUTHEAST UNIVERSITY

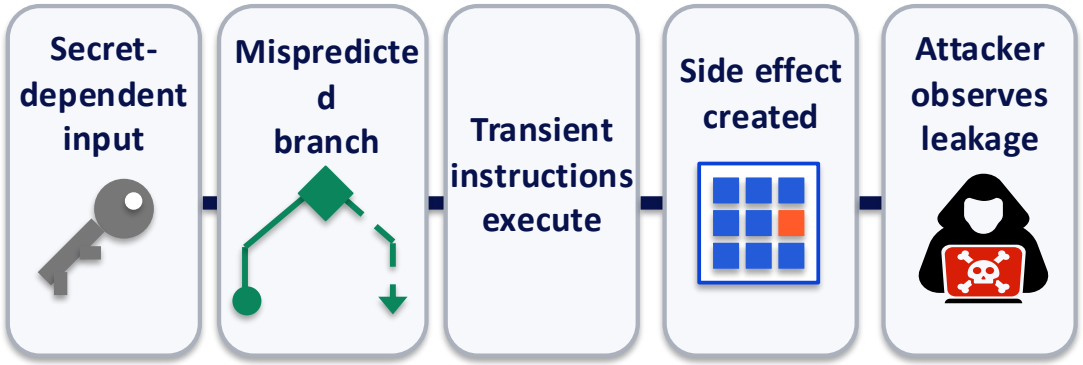
NUS
National University
of Singapore

USENIX

# What selective speculation tries to do

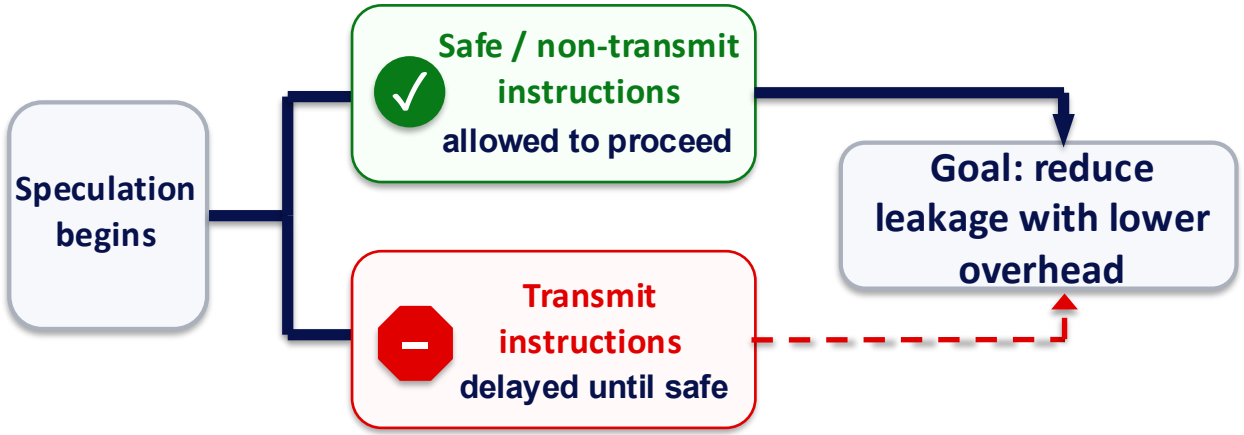
[NDA, MICRO'19] [SpecShield, PACT'19] [STT, MICRO'19] [SPT, MICRO'21] [DOLMA, USENIX'21] [SpecTerminator, TACO'23]

## Without protection




All transient effects are dangerous.

## Selective speculation



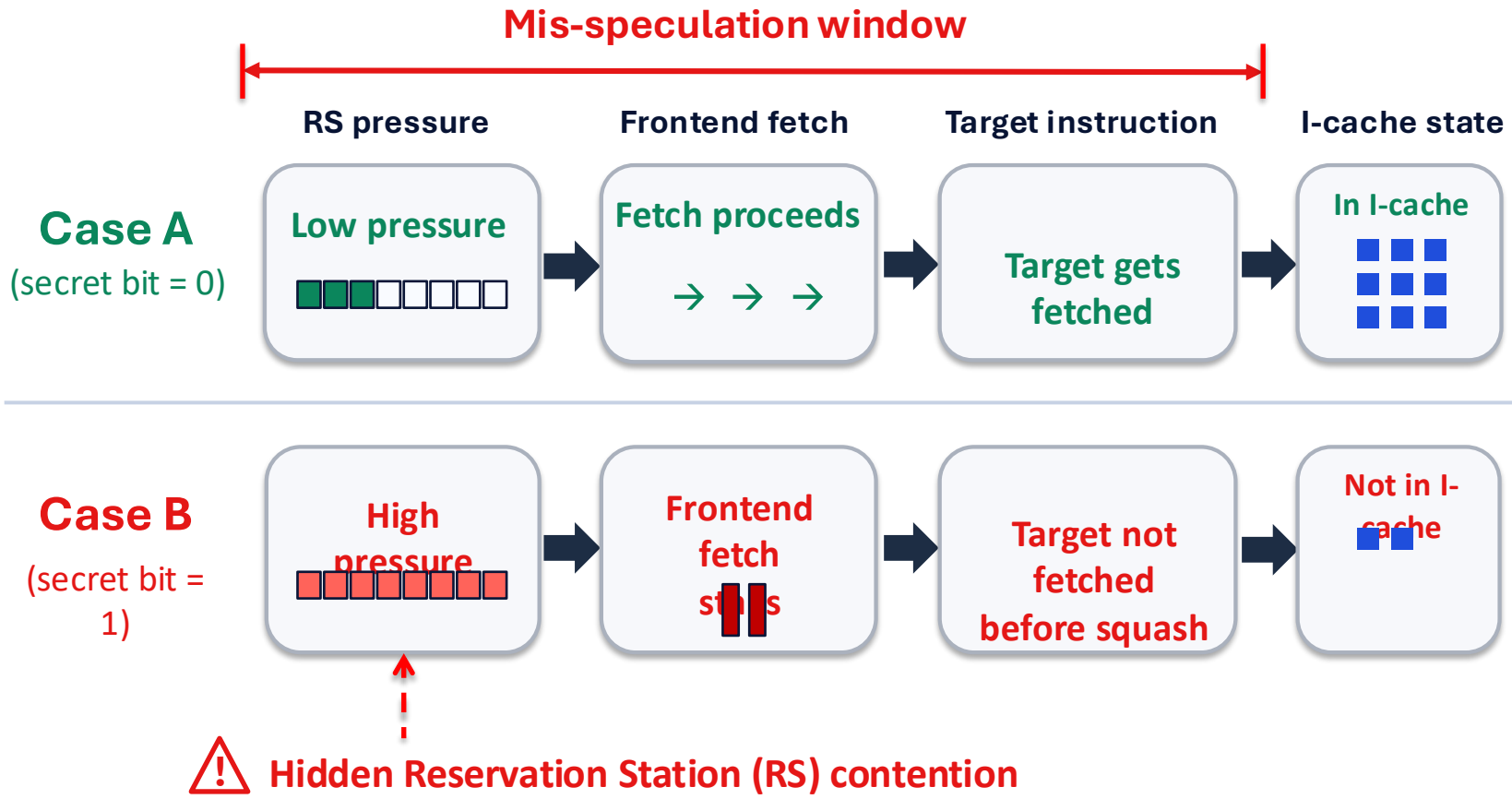

**Delay** only the instructions that could transmit secret-dependent side effects.

**Optimization opportunity**

If we can identify transmit instructions precisely, we can keep performance high.

# Why optimized selective speculation is still vulnerable

- **Goal of selective speculation:**
  keep speculation fast while blocking leakage.
- **Optimization trend:**
  rely on “safe” assumptions to avoid delaying everything.
- **Motivating question:**
  can these “safe” effects still become observable?




**Key observation:** Overlooked speculative effects can create secret-dependent RS pressure, leaving an observable I-cache footprint.

USENIX

# What optimized selective speculation assumes

Goal: keep speculation fast, while delaying only μops that can transmit secrets.

1 **Assumption 1** [STT, MICRO'19] [SPT, MICRO'21] [DOLMA, USENIX'21]

**Transmission starts after issue**

**Misses** dispatch-phase effects such as operand-dependent μop expansion.

2 **Assumption 2** [STT, MICRO'19] [SPT, MICRO'21] [DOLMA, USENIX'21]

**Control leakage = redirect-based**

**Misses** predicated secret-dependent selection without fetch redirect.

3 **Assumption 3** [DOLMA, USENIX'21] [SpecTerminator, TACO'23]

**Contention is only younger → older**

**Misses** bidirectional contention: older speculative μops can still throttle younger fetch.

USENIX

# Transmit taxonomy misses pre-issue footprint

## STT (instruction-level)

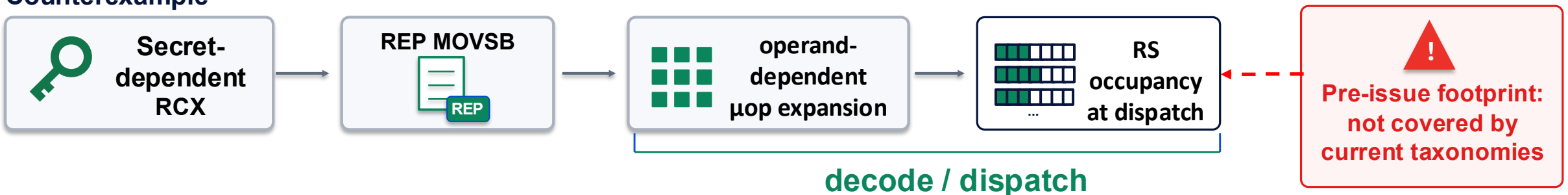
“ Transmit instructions are those whose **execution creates** operand-dependent resource usage that can reveal the operand. ”

## DOLMA (µop-level)

“ Transmit µops are speculative µops whose operand values are transmitted **during transient execution** via timing variations. ”

⚠ **Shared assumption: transmit effects start after issue / during execution**

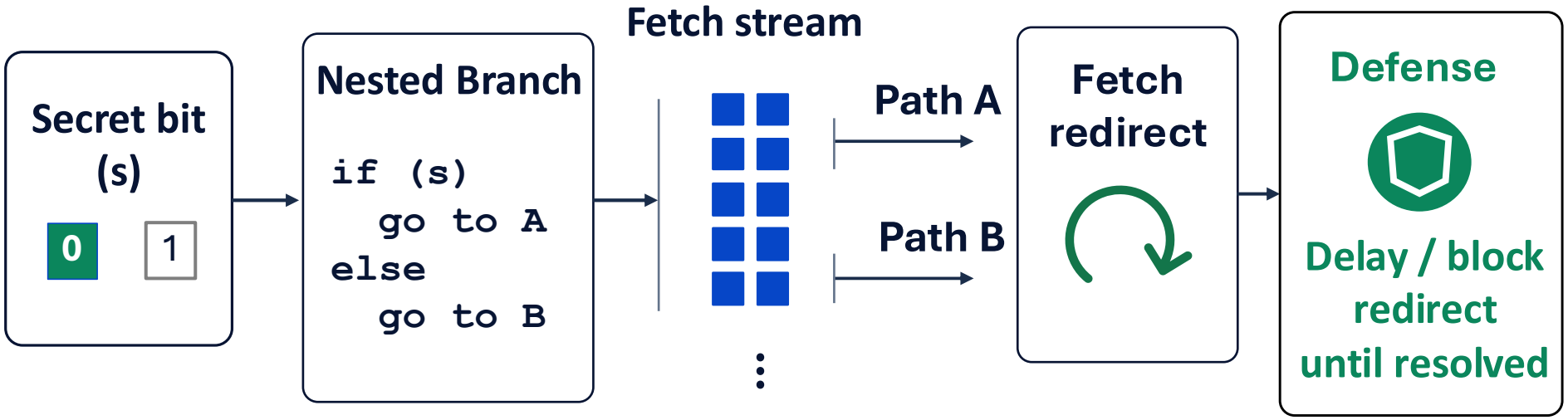

**Counterexample**



# Delay-until-resolution misses predication

Redirect-based protection blocks branch divergence, but not secret-dependent value selection through predication.

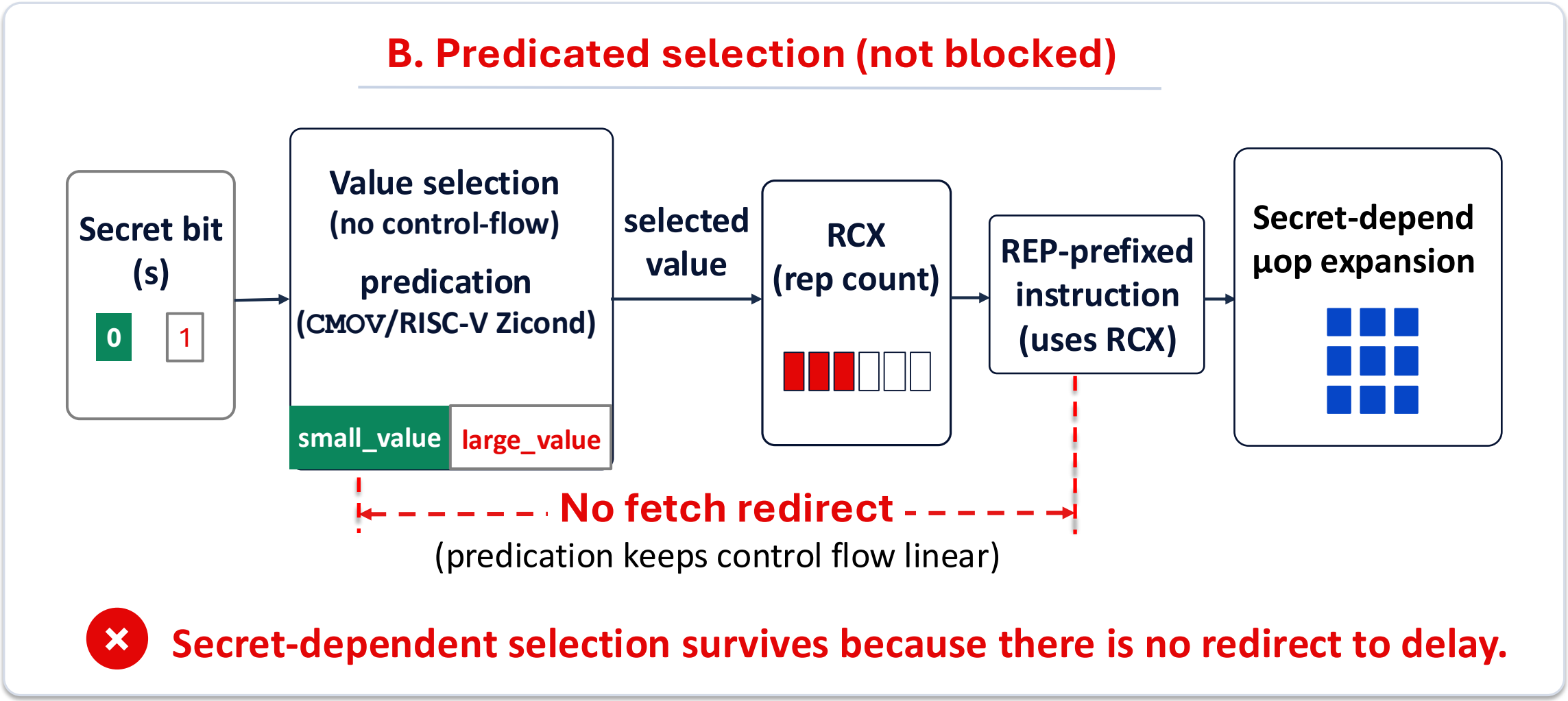
## A. Branch-based selection (blocked)




Delay-until-resolution keeps speculation on one path.

# Delay-until-resolution misses predication

Redirect-based protection blocks branch divergence, but not secret-dependent value selection through predication.



# Delay-until-resolution misses predication

Redirect-based protection blocks branch divergence, but not secret-dependent value selection through predication.

## Gadget 1: Predication + µop Expansion

**Example pseudocode**

```
if (condition){
CMP secret, 0
CMOVNE large, RCX
REP MOVSB
target inst
}
```

**small_value in RCX** → **Low RS pressure (RS sparse)**

**large_value in RCX** → **High RS pressure (RS crowded)**

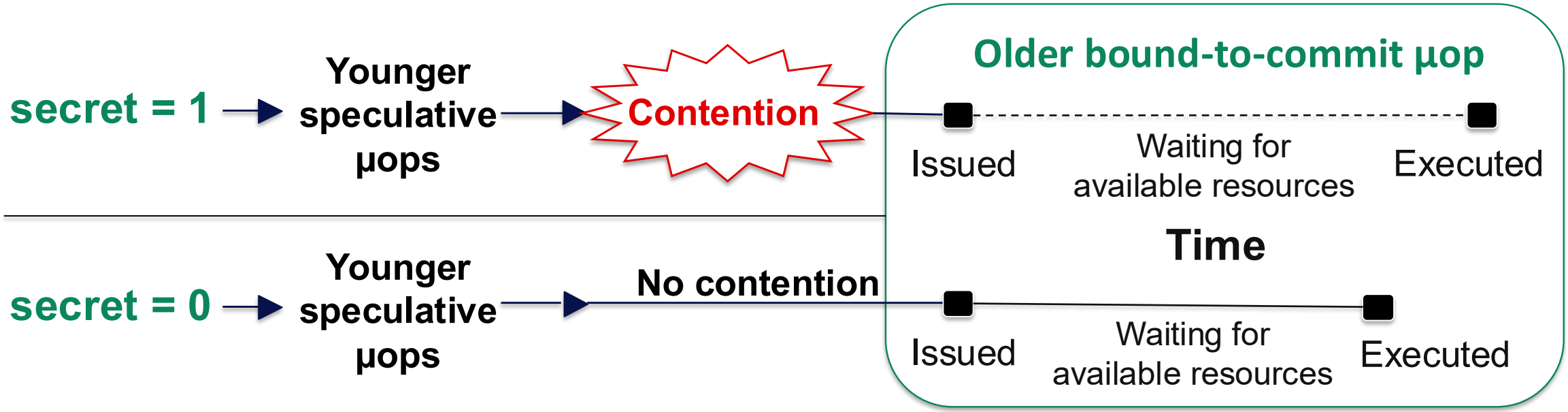
**Predication is the secret-injection step** - - - - → **REP MOVSB turns that value into RS pressure**

USENIX

# Older-µop-first allocation is not enough

Older-first allocation blocks younger-to-older contention, but not older-to-younger fetch effects.

## Why older-first is used

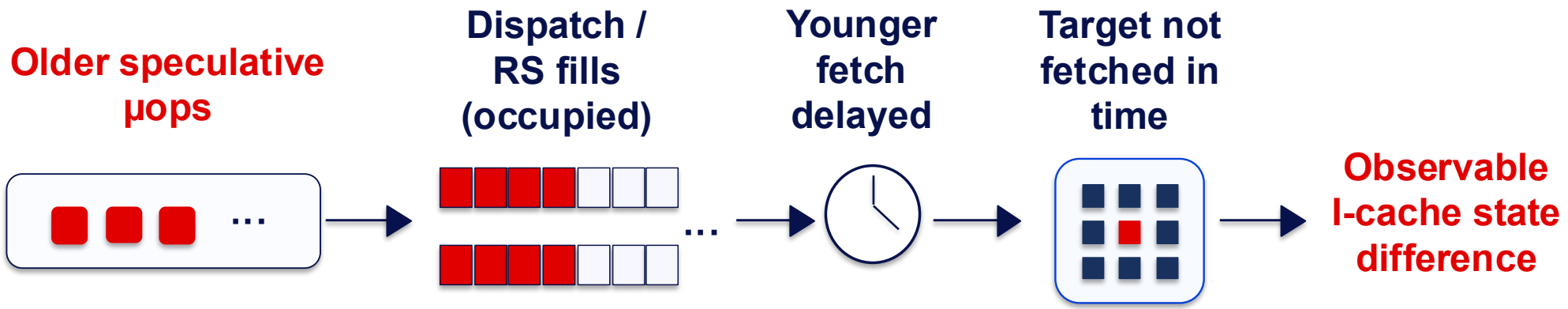



Older-first allocation aims to prevent unsafe younger → older contention from affecting older bound-to-commit instructions.

# Older-µop-first allocation is not enough

Older-first allocation blocks younger-to-older contention, but not older-to-younger fetch effects.

**Blind spot: contention is not one-way**

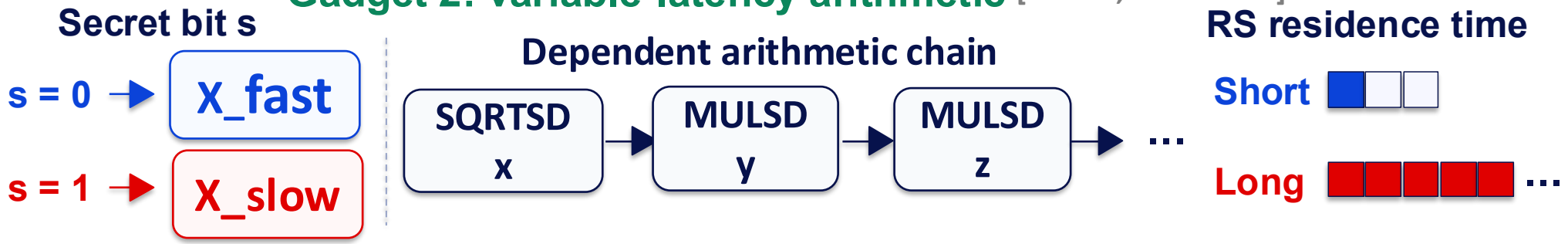



**Even if younger→older contention is prevented, older speculative µops can still create observable younger-fetch effects.**

# Older-μop-first allocation is not enough

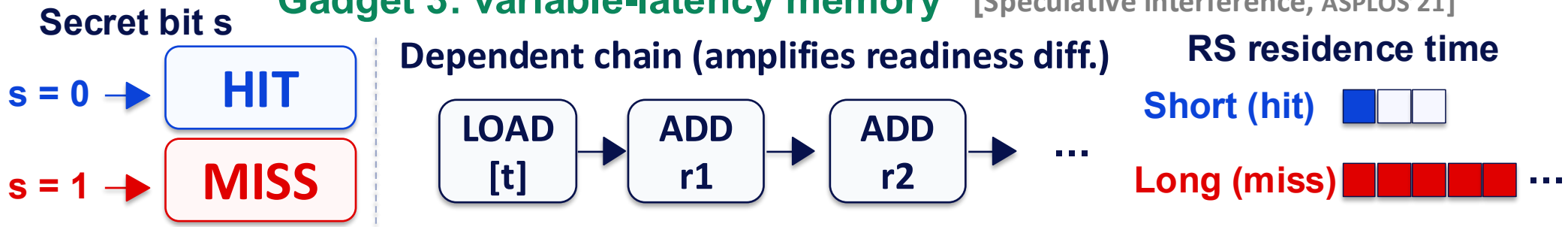
Older-first allocation blocks younger-to-older contention, but not older-to-younger fetch effects.

## Gadget 2: variable-latency arithmetic [USLH, USENIX'23]




*Older speculative arithmetic can create measurable RS pressure.*

## Gadget 3: variable-latency memory [Speculative interference, ASPLOS'21]




*Cache-hit / memory-readiness differences can also leak.*

USENIX

# How we evaluate the attack surface

## 1 Defense bypass in gem5

**? Can gadgets recover secret bits under selective-speculation defenses?**

- instantiate Spectre-v1-style gadgets
- x86 and RISC-V models
- validate STT and DOLMA bypassing

## 2 Real CPU validation

**? Do the RS-contention effects appear on real hardware?**

- Intel Raptor Cove and AMD Zen
- validate μop-expansion-induced RS pressure

## 3 Realistic gadget discovery

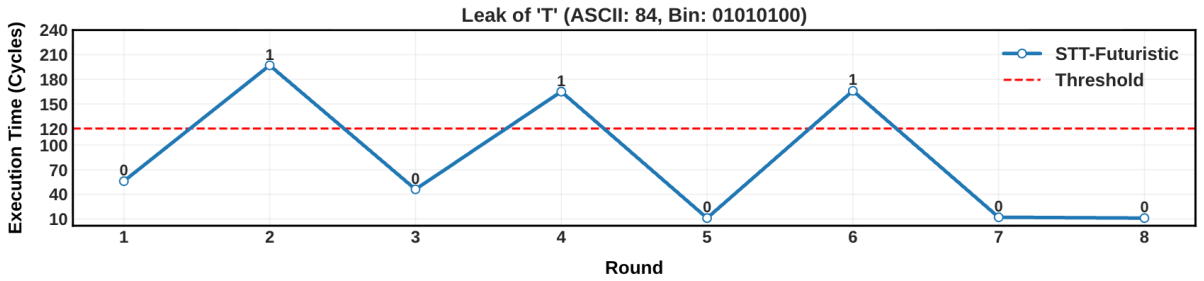
**? Can we find realistic gadget structure in real software?**

- LLVM-based pass identifies RS-contention-relevant code patterns
- `example target: libsodium ristretto255_frombytes`

RESULTS

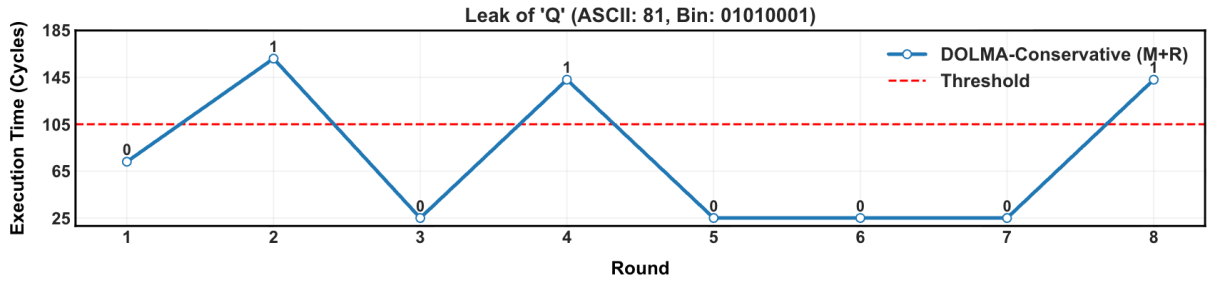
# Bypasses are observable in gem5

## 1 STT Futuristic: Gadget 1



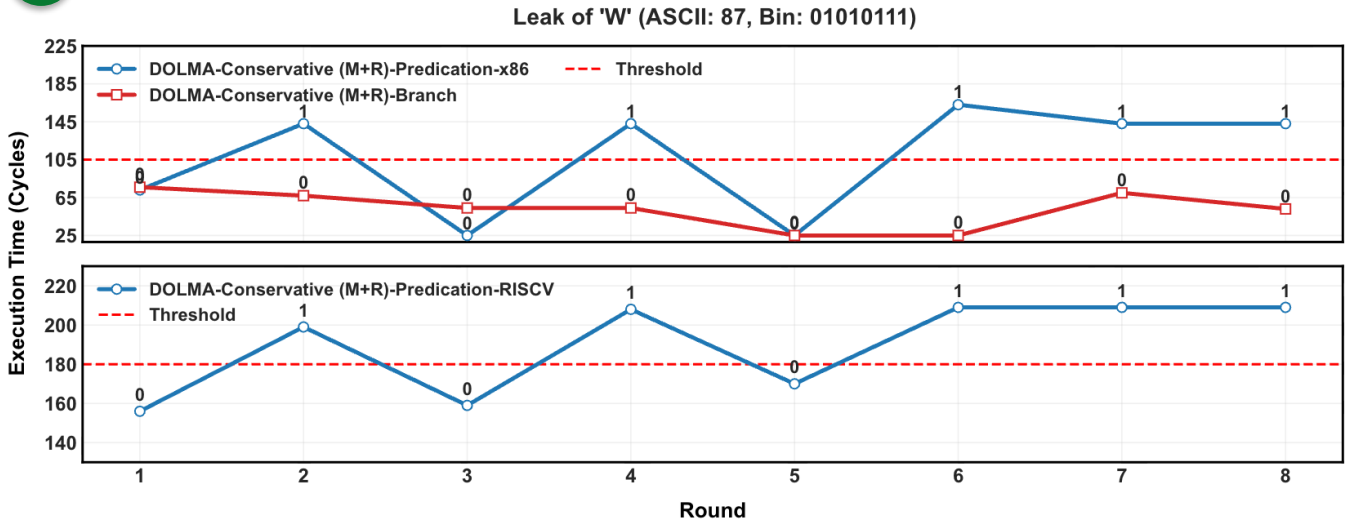

**Predication + REP MOVSB recovers an 8-bit secret under STT.**

## 2 DOLMA Conservative (M+R): Gadget 3




**Older-µop-first does not prevent the memory-based RS-contention gadget.**

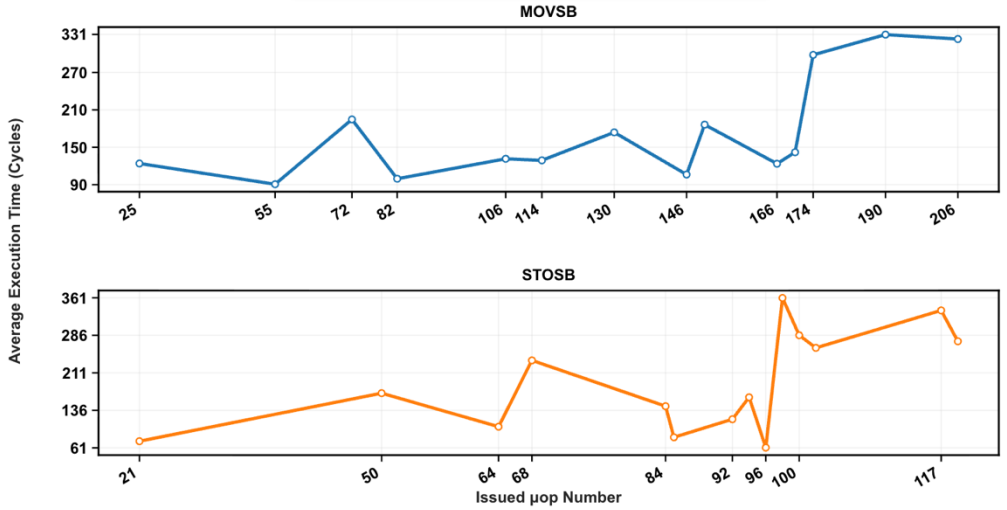
## 3 Branch vs predication



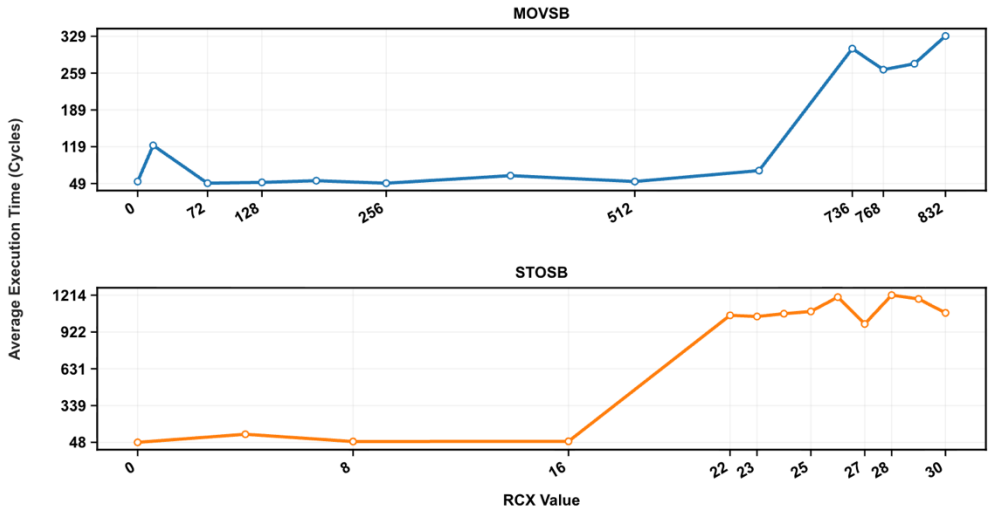

| | | |
|---|---|---|
| Branch-based selector (x86 & RISC-V) | All timings below threshold | Blocked ✗ |
| Predicated selector (x86) (e.g., `CMOV`) | Timings cross threshold | Succeeds ✓ |
| Predicated selector (RISC-V) (e.g. `czero`) | Timings cross threshold | Succeeds ✓ |

# Real-World RS-Contention Effects

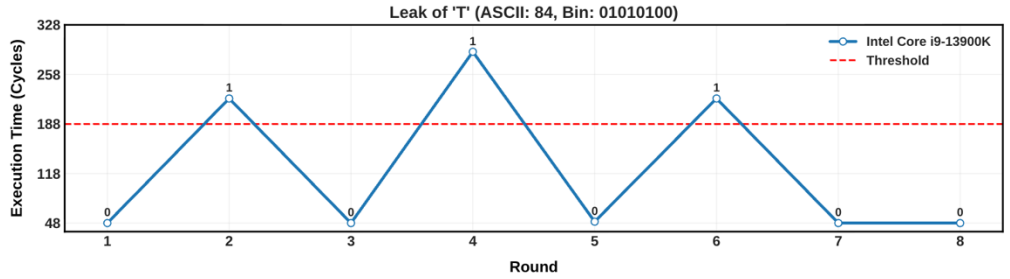
## Intel Core i9-13900K




## AMD Ryzen 7 5800H




## Secret bits recovery



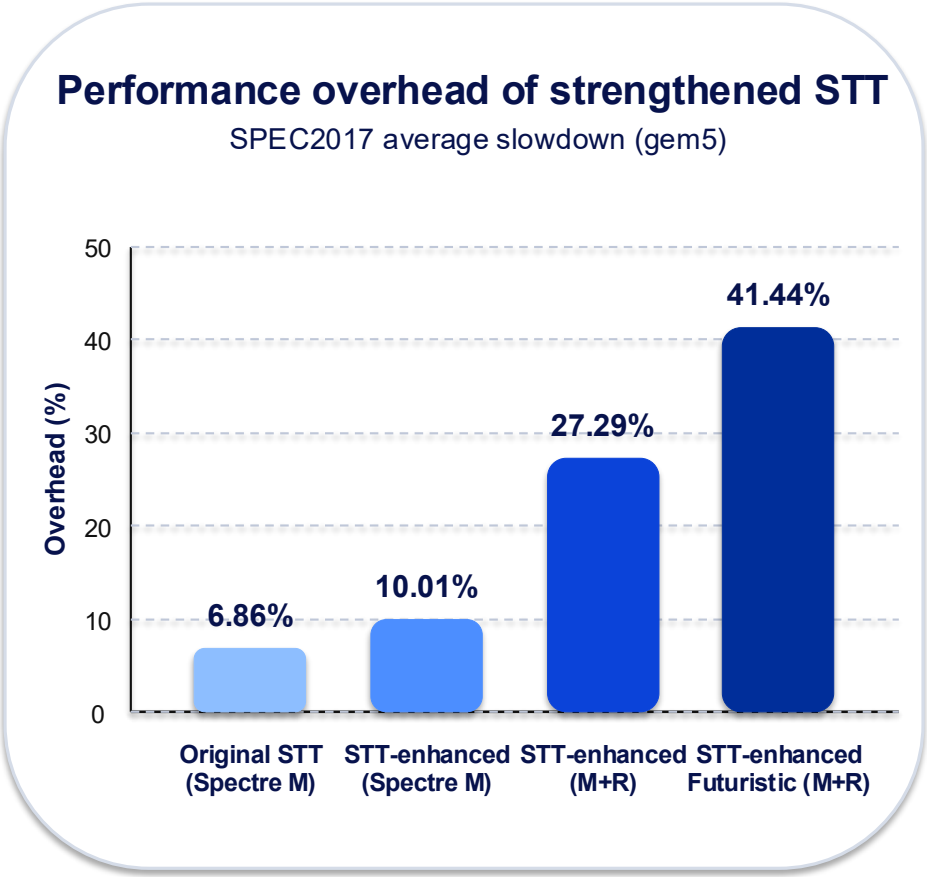

## LLVM pass: search results

| | OpenSSL 1.1.1q | Libgcrypt 1.11.0 | libsodium 1.0.20 | OpenBLAS 0.3.0 | musl 1.2.5 | zlib 1.3.1 | wolfssl 5.8.4 |
|---|---|---|---|---|---|---|---|
| variable-latency memory | 52 | 77 | 46 | 24 | 26 | 5 | 15 |
| REP-prefixed instruction | 28 | 13 | 0 | 0 | 10 | 9 | 10 |

# Strengthened STT closes the exposed gaps

Three limitations and corresponding fixes

| Lim. | Assumption | Bypass | Defense extension / fix |
|---|---|---|---|
| I | Transmit effects arise only after issue. | Operand-dependent µop expansion leaks through RS contention. | • Refine transmit taxonomy.<br>• Block dispatch of tainted operand-dependent microcode-expanding instructions. |
| II | Control leakage is captured by fetch redirection. | Predication injects secrets into operands without redirecting fetch. | • Delay issue of tainted predicated instructions. |
| III | Older-µop-first allocation is sufficient. | Older and younger µops can still contend bidirectionally. | • Continue delaying issue of arithmetic and memory-access instructions. |




**New transmit taxonomy** A speculative instruction is a transmit instruction if operand-dependent variation in its speculative behavior can induce attacker-observable microarchitectural state changes.

USENIX

# Thanks !

**Questions?** xiaoyu_cheng@seu.edu.cn

**Code & Artifact** Available at:

SOUTHEAST UNIVERSITY

NUS National University of Singapore