BINGHAMTON UNIVERSITY
State University of New York

A Predictive Model for Cache-Based Side Channels in Multicore and Multithreaded Microprocessors

Leonid Domnitser, Nael Abu-Ghazaleh and Dmitry Ponomarev
Department of Computer Science
SUNY-Binghamton
{lenny, nael, dima}@cs.binghamton.edu

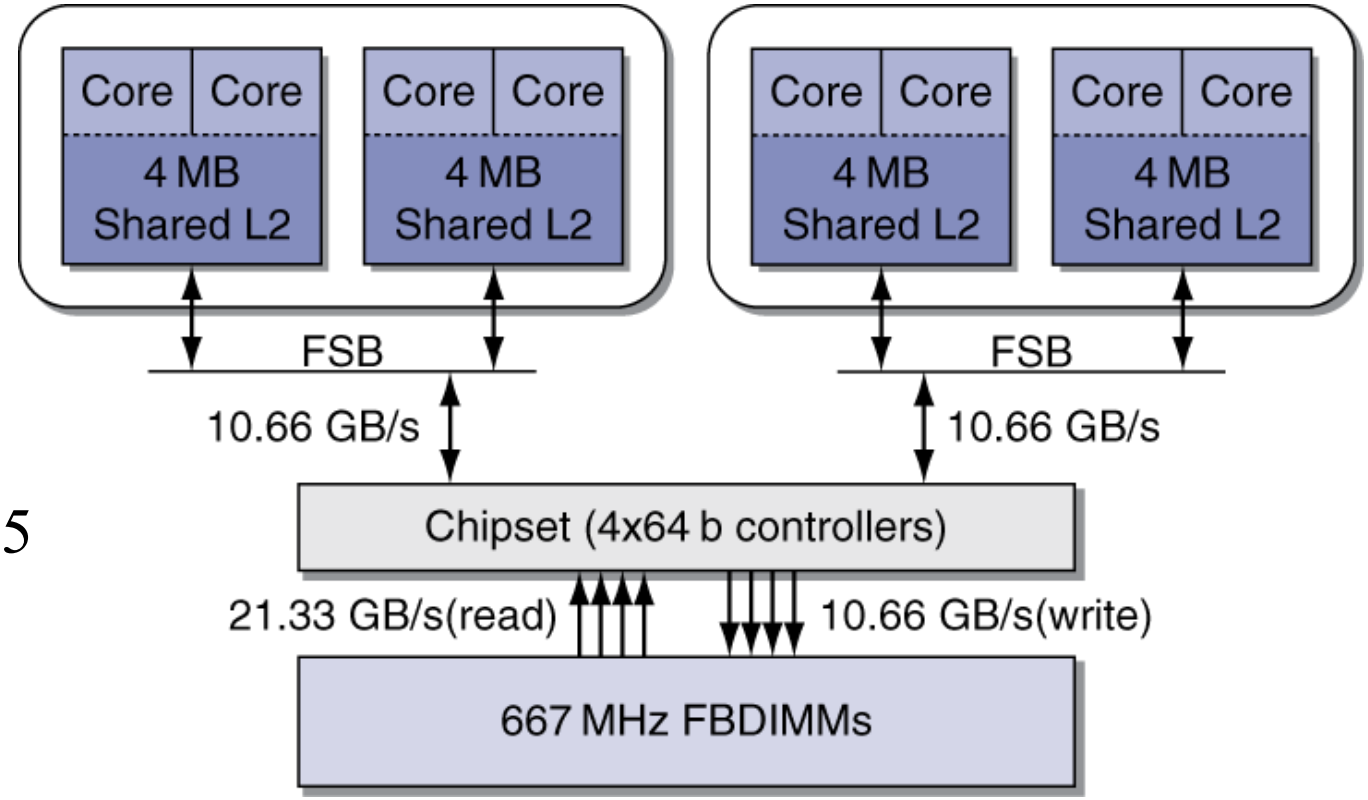
BINGHAMTON UNIVERSITY
State University of New York

Multi-cores-->Many-cores

- Moore's law coming to an end
  - Power wall; ILP wall; memory wall
  - “End of lazy-boy programming era”
- Multi-cores offer a way out
  - New Moore's law: 2x number of cores every 1.5 years
- New security vulnerabilities arise due to resource sharing
  - Side-Channel Attacks and Denial of Service Attacks

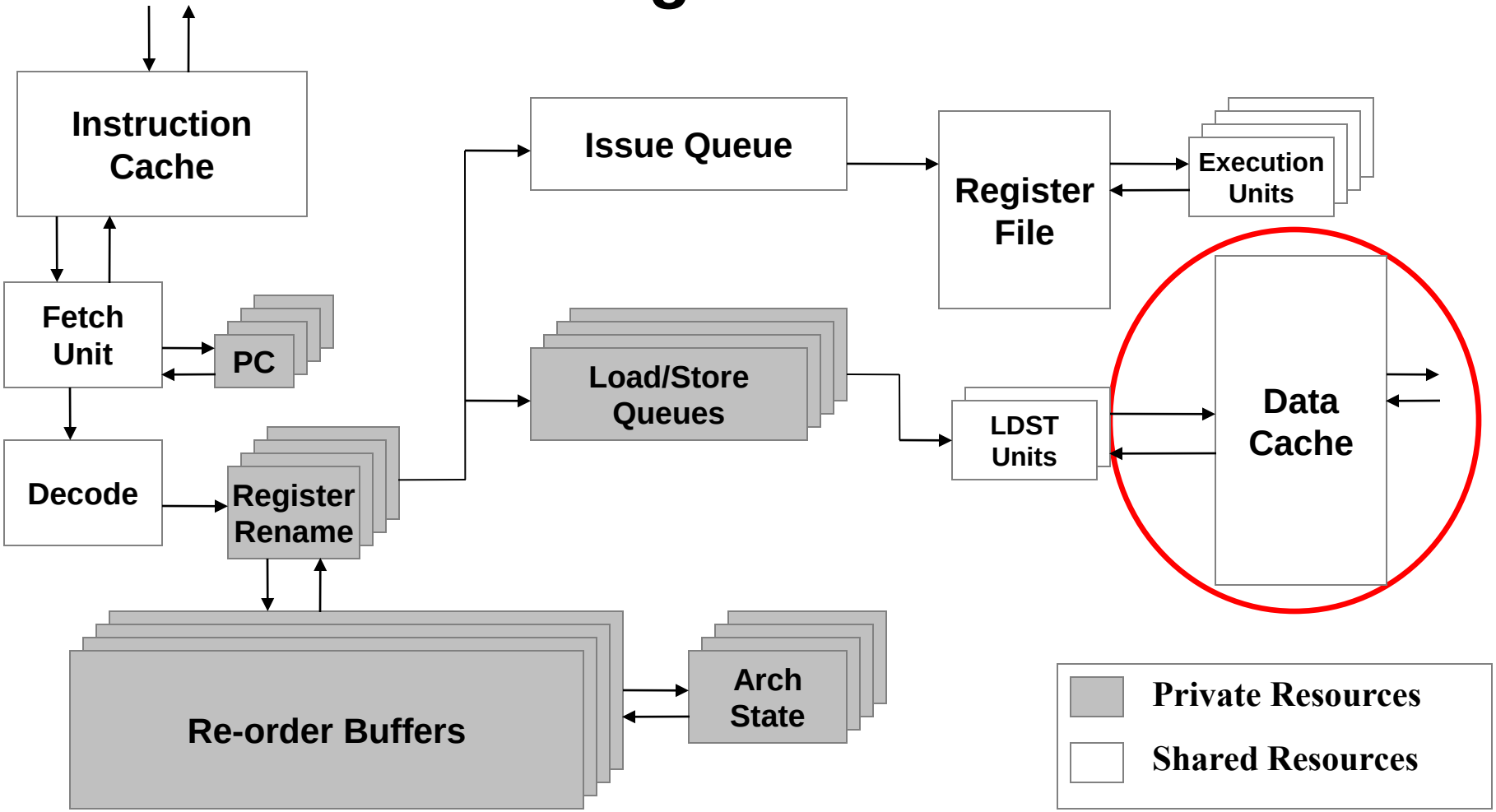
Last-level Cache Sharing on Multicores (Intel Xeon)

2 quad-core Intel Xeon e5345 (Clovertown)



BINGHAMTON UNIVERSITY
State University of New York

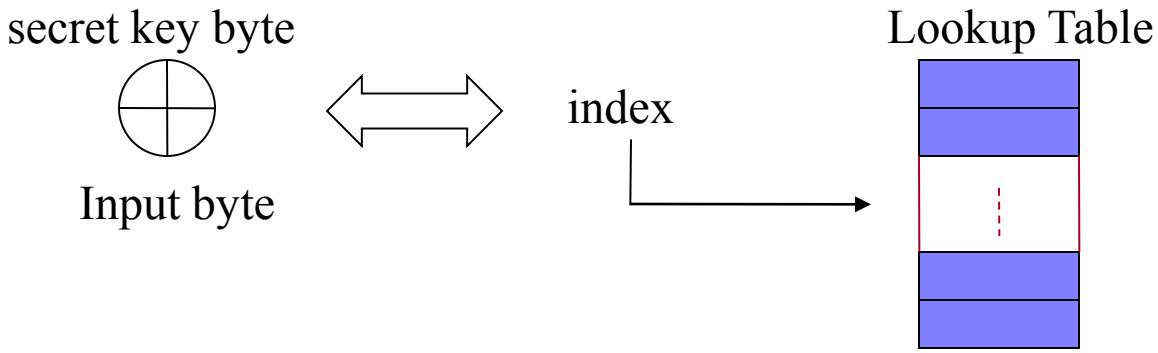
L1 Cache Sharing in SMT Processor



BINGHAMTON UNIVERSITY
State University of New York

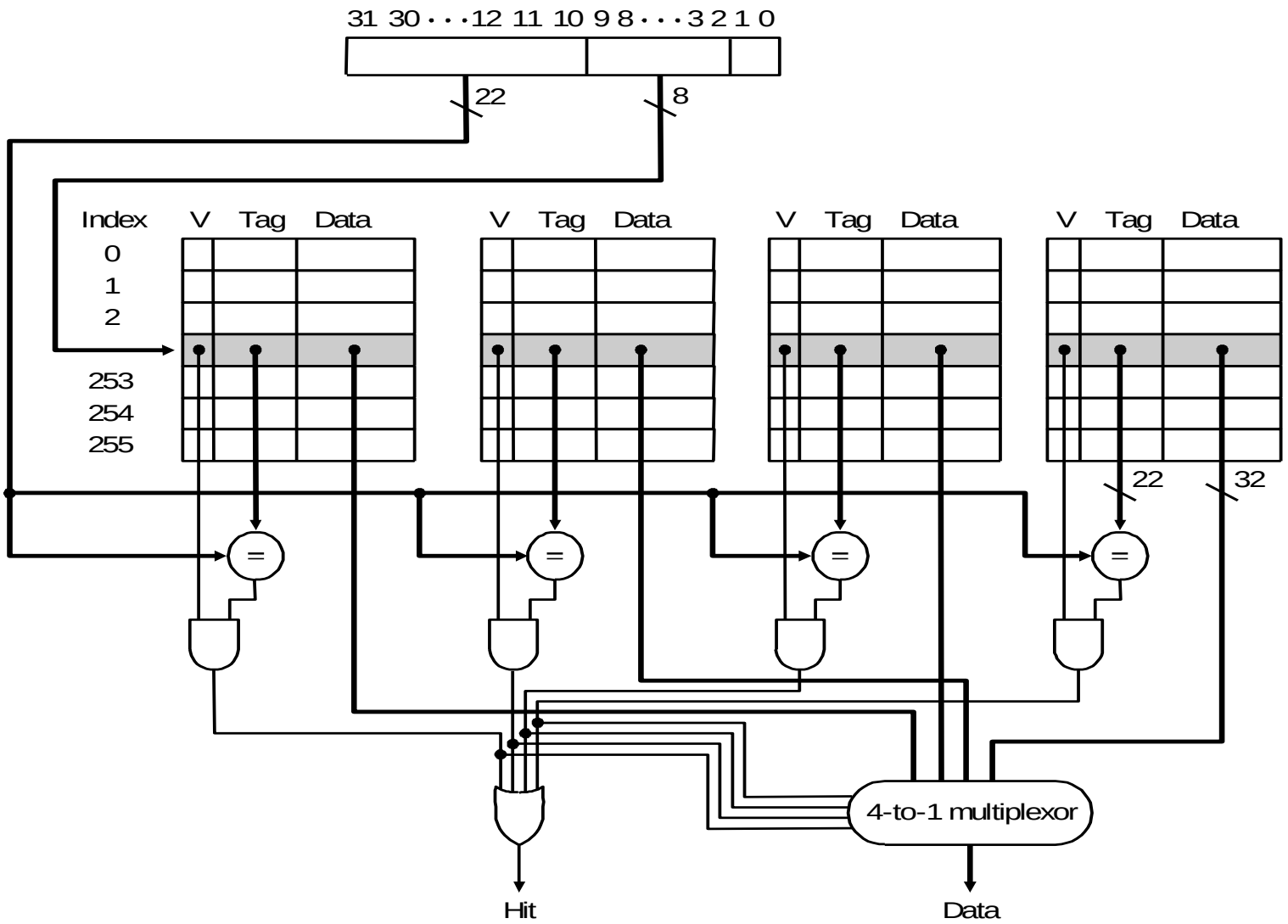
Advanced Encryption Standard (AES)

- One of the most popular algorithms in symmetric key cryptography
  - 16-byte input (plaintext)
  - 16-byte output (ciphertext)
  - 16-byte secret key (for standard 128-bit encryption)
  - several rounds of 16 XOR operations and 16 table lookups

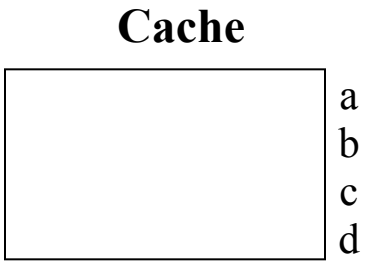


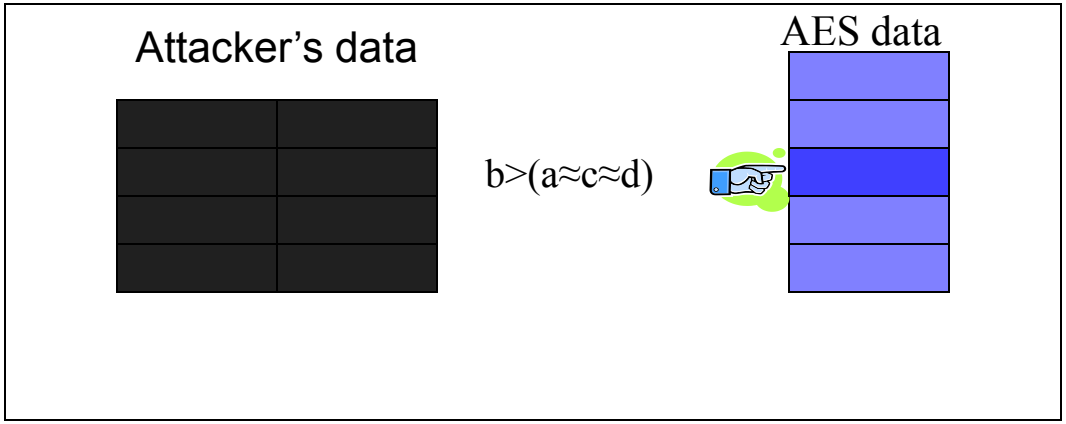
BINGHAMTON UNIVERSITY
State University of New York

Set-Associative Caches



Attack Example





Can exploit knowledge of the cache replacement policy to optimize attack

Cache-Based Side Channel Attacks

- An attacker and a victim process (e.g. AES) run together using a shared cache
- Access-Driven Attack:
  - Attacker occupies the cache, evicting victim's data
  - When victim accesses cache, attacker's data is evicted
  - By timing its accesses, attacker can detect intervening accesses by the victim
- Time-Driven Attack
  - Attacker fills the cache
  - Times victim's execution for various inputs
  - Performs correlation analysis

Simple Attack Code Example

BINGHAMTON UNIVERSITY
State University of New York

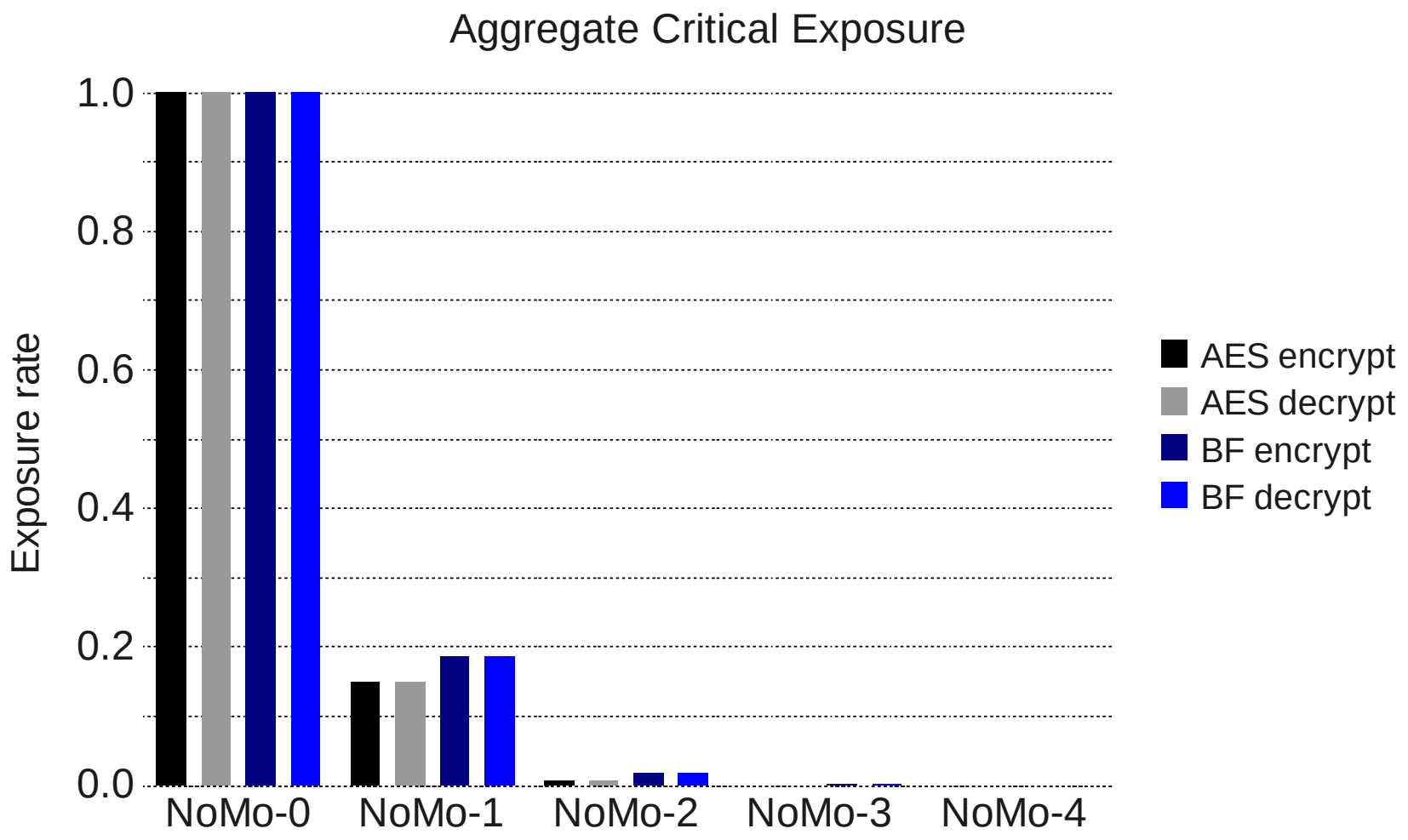
```
#define ASSOC 8
#define NSETS 128
#define LINESIZE 32
#define ARRAYSIZE (ASSOC*NSETS*LINESIZE/sizeof(int))

static int the_array[ARRAYSIZE]
int fine_grain_timer(); //implemented as inline assembler

void time_cache() {
   register int i, time, x;
   for(i = 0; i < ARRAYSIZE; i++) {
      time = fine_grain_timer();
      x = the_array[i];
      time = fine_grain_timer() - time;
      the_array[i] = time;
   }
}
```

Existing Solutions

- Avoid using pre-computed tables – too slow
- Close the channel:
  - Lock critical data in the cache (Lee, ISCA 07)
    - Impacts performance
  - Randomize the victim selection (Lee, MICRO'08)
  - Both are highly complex, impact performance and require OS/ISA support
- Constrain the channel:
  - NoMo: dynamic partitioning of the cache that sets some ways of the cache to be exclusive to each thread
  - Still leaks a small amount of accesses

BINGHAMTON UNIVERSITY
State University of New York

Aggregate Exposure of Critical Data



Contribution and Motivation

- Predictive mathematical model for access leakage for access based side channel attacks
- Why? Vulnerability is a function of information leaked through the side-channel
  - Estimate how vulnerable current algorithms and caches to attack
  - Estimate how effective imperfect solutions that constrain rather than close the side-channel are

BINGHAMTON UNIVERSITY
State University of New York

Model Parameters

A	event of an access
α	number of memory accesses
C	event of critical access
D	event of detected access
m	number of lines used in a set
N	number of sets
s	cache set number
T_a	time between repeat accesses by attacker
T_v	time between repeat accesses by victim
w	cache associativity

BINGHAMTON UNIVERSITY
State University of New York

Leakage Prediction Model

- P(D|C) is the probability that the cache access is detected by the attacker given that this access is critical
- P(D) is the probability that the access is detected
- P(C) is the probability that the access is critical

- Our model computes P(C|D) as follows:
- P(D|C) = [P(C|D)*P(D)] / P(C)

Leakage Prediction Model

BINGHAMTON UNIVERSITY
State University of New York

- Estimating P(C)
  - Average fraction of critical accesses – constant for a given program.
  - Can be estimated through static analysis or profiling
- Estimating P(D)
  - Number of detected accesses out of the total number of accesses
  - 100% for the perfect attacker. Less in reality as some accesses are hidden by cache hits.
- Estimating P(C|D)
  - Need to filter out the noise due to non-critical accesses

Estimating P(D)

$$P(D_s) = \frac{m_s \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_s}$$

Estimating P(C|D)

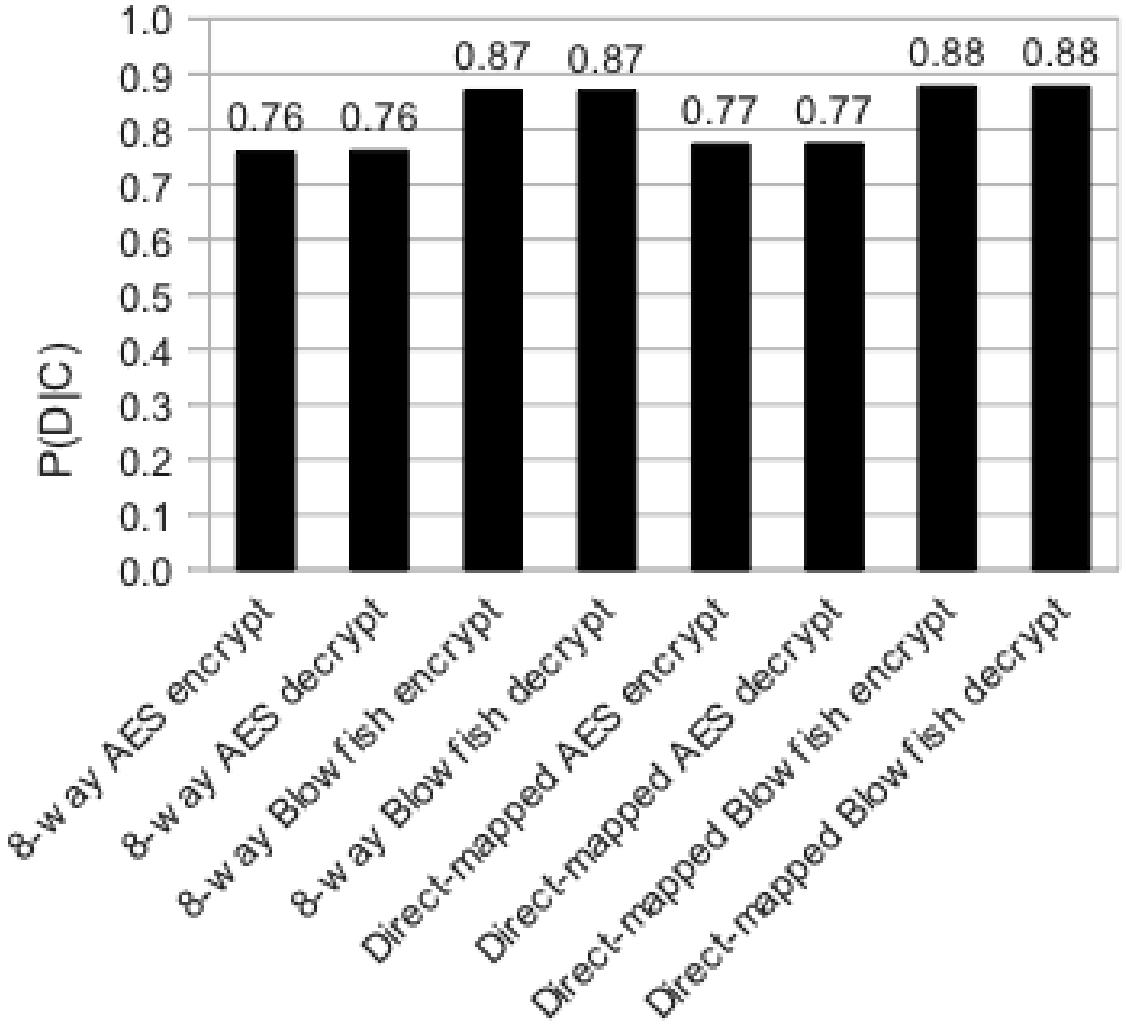
$$P(C_S|D_S) = \frac{\left(\frac{m_{C,S} \cdot \min\left(\frac{w}{m_S}, 1\right)}{\alpha_S}\right)}{\left(\frac{m_S \cdot \min\left(\frac{w}{m_S}, 1\right)}{\alpha_S}\right)} = \frac{m_{C,S}}{m_S}$$

BINGHAMTON UNIVERSITY
State University of New York

Estimating P(D|C)

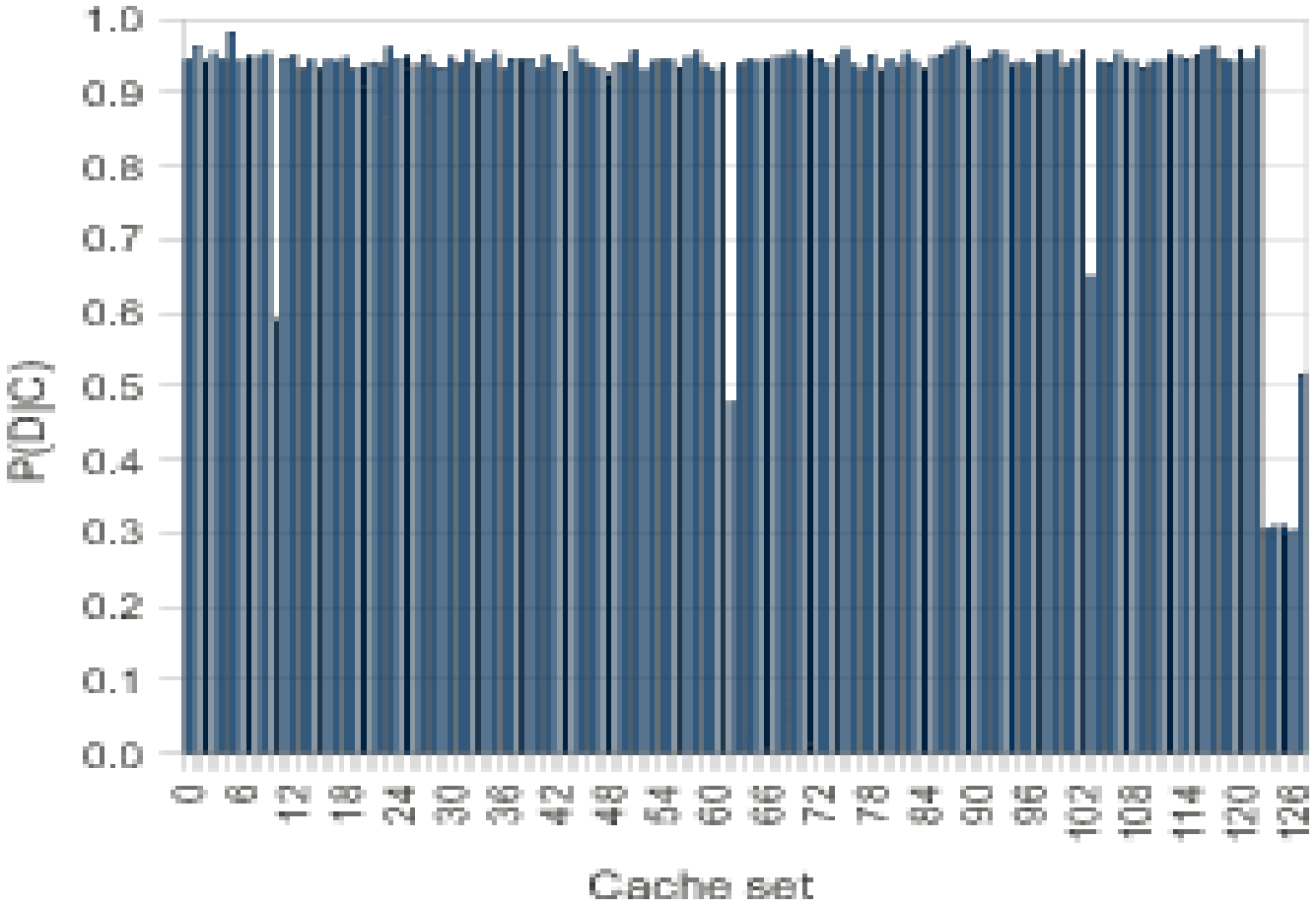
$$P(D|C) = \frac{\sum_{s=0}^{N} \begin{cases} 0 & : \alpha_{C,s} = 0 \\ \frac{m_{C,s} \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_{C,s}} & : otherwise \end{cases}}{\sum_{s=0}^{N} \begin{cases} 0 : \alpha_{C,s} = 0 \\ 1 : otherwise \end{cases}}$$

Aggregate Leakage Predicted by Model



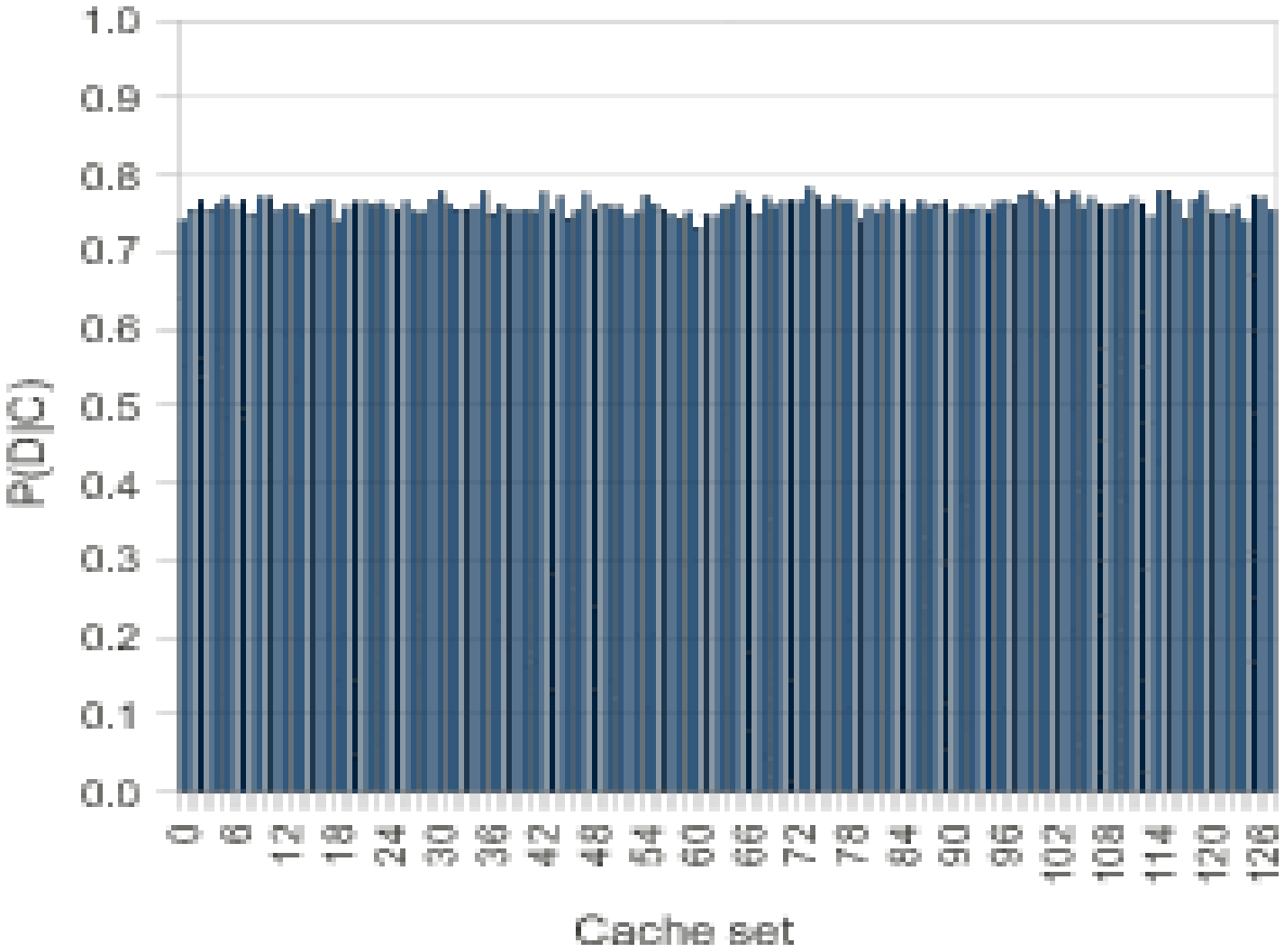
Predicted Leakge Per Set, Blowfish 8-way set associative cache

BINGHAMTON UNIVERSITY
State University of New York



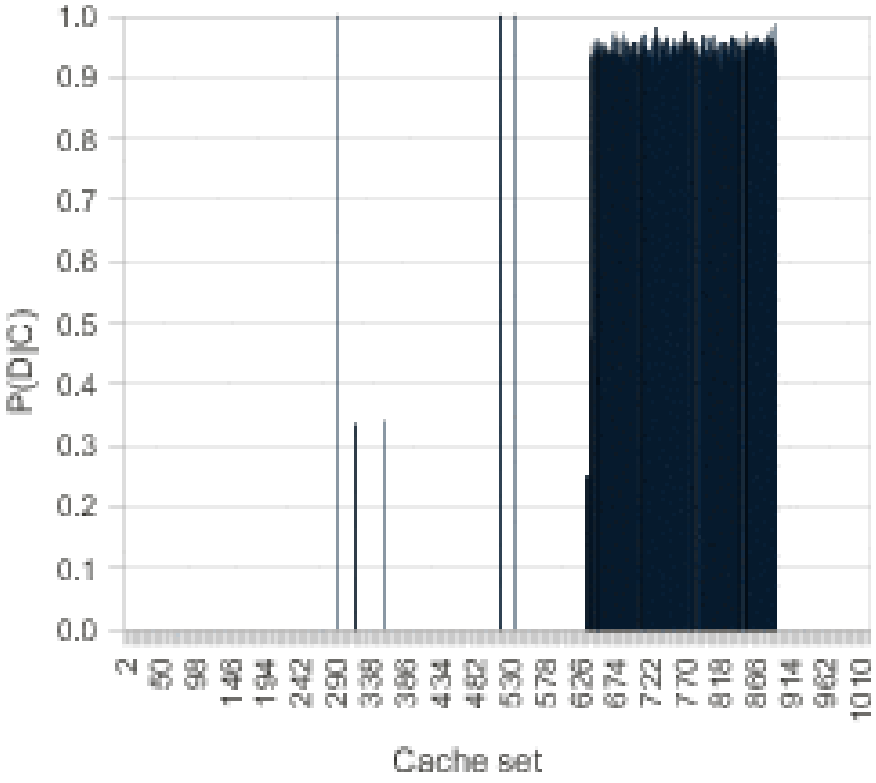
Predicted leakage Per Set, AES 8-way set associative cache

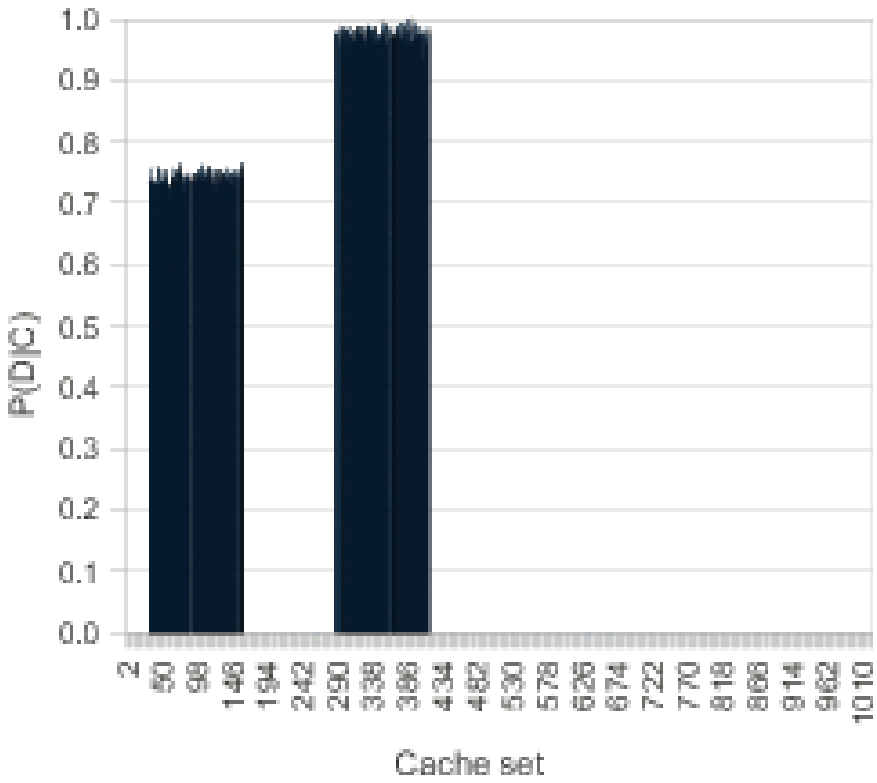
BINGHAMTON UNIVERSITY
State University of New York



Predicted leakage Per Set, Blowfish, AES, Direct Mapped Cache

BINGHAMTON UNIVERSITY
State University of New York





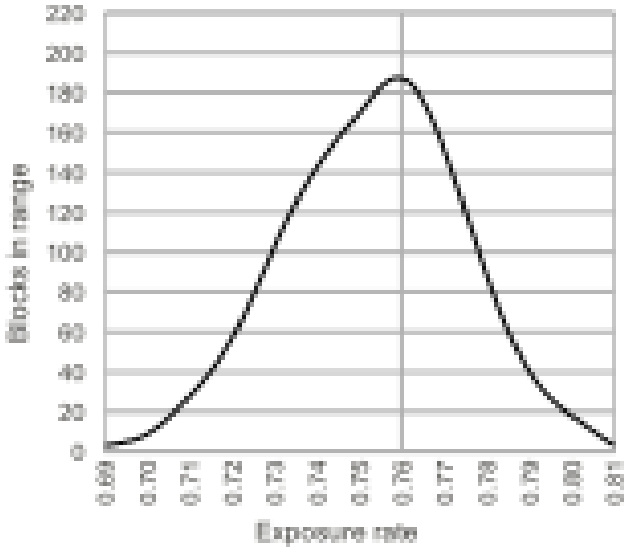
BINGHAMTON UNIVERSITY
State University of New York

Model Validation Methodology

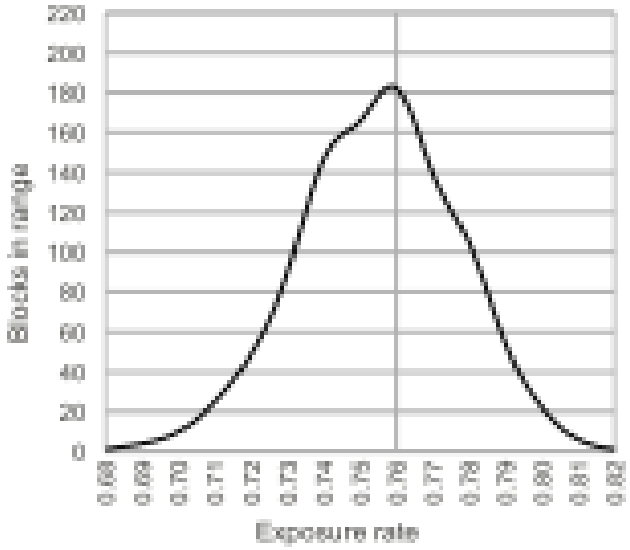
- We used M-Sim-3.0 cycle accurate simulator (multithreaded and Multicores derivative of Simplescalar) developed at SUNY Binghamton
  - http://www.cs.binghamton.edu/~msim
- Evaluated leakage for AES and Blowfish encryption/decryption
- Ran security benchmarks for blocks of randomly generated input
- Implemented the attacker as a separate thread and ran it alongside the crypto processes
- Assumed that the attacker is able to synchronize at the block encryption boundaries (i.e. It fills the cache after each block encryption and checks the cache after the encryption).

BINGHAMTON UNIVERSITY
State University of New York

Model Validation: Per Block Leakage in AES



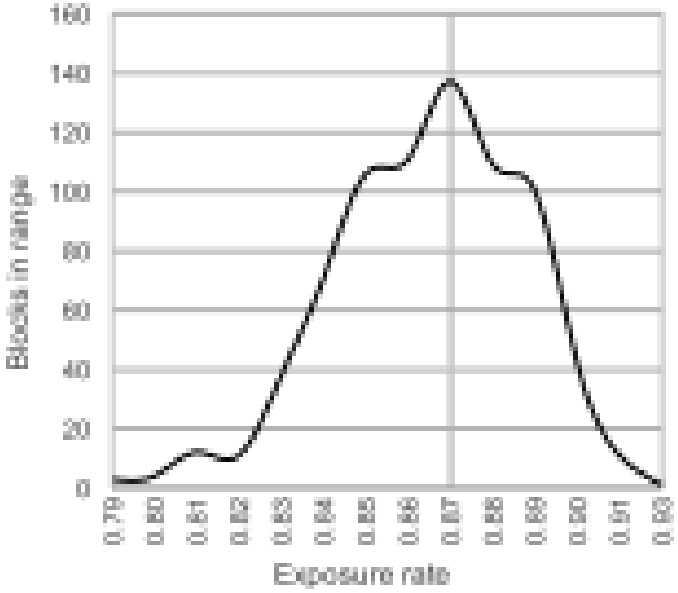
(a) AES enc., 8-way



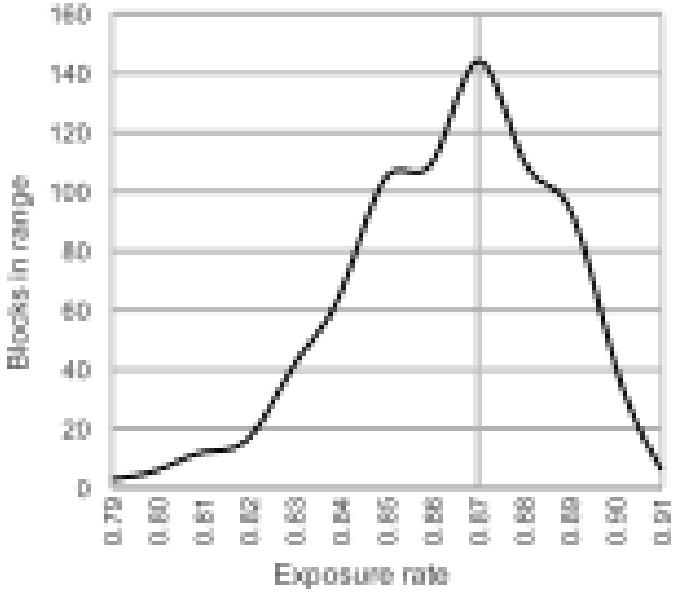
(b) AES dec., 8-way

BINGHAMTON UNIVERSITY
State University of New York

Model Validation: Blowfish



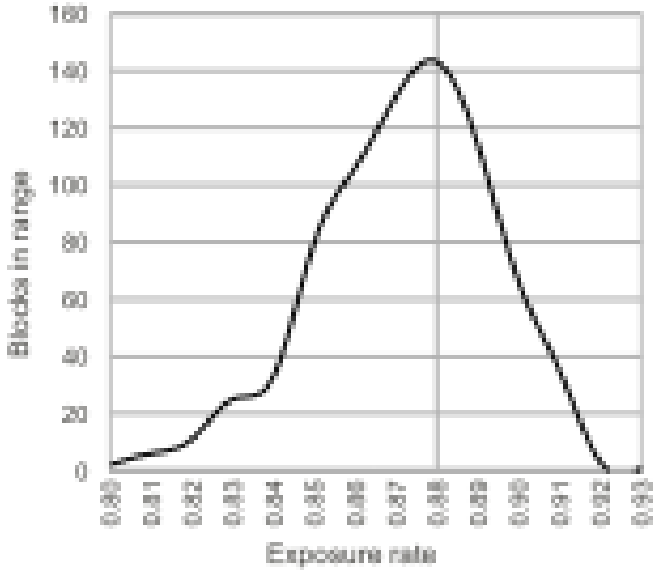
(e) Blowfish enc., 8-way



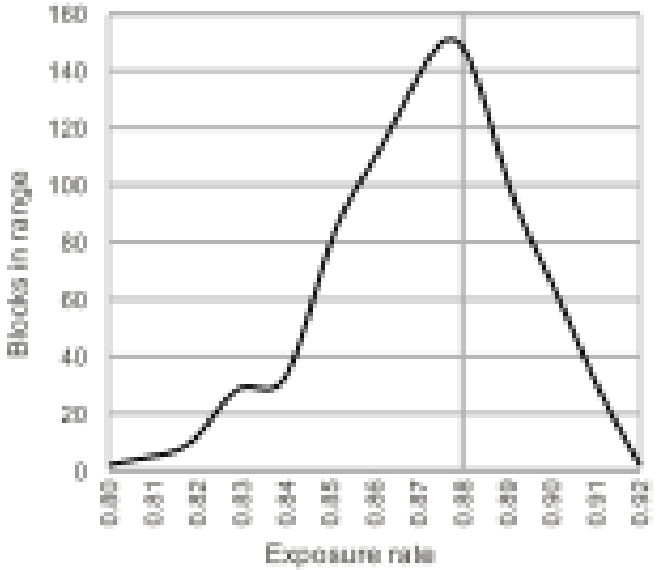
(f) Blowfish dec., 8-way

BINGHAMTON UNIVERSITY
State University of New York

Model Validation: Direct-Mapped Cache



(g) Blowfish enc., direct-mapped



(h) Blowfish dec., direct-mapped

Conclusions and Future Work

- We developed a model for information leakage for access based cache side channel attacks
  - An important attack with the emergence of multi-core and multi-threaded architectures
- Model can capture overall accesses in an application or any subset of them (e.g., specific sets)
- The model was validated against a cycle accurate simulator
- Future work: model the impact of defenses that reduce the leakage
- Future work: translate leakage pattern into a measure of difficulty of breaking the key

BINGHAMTON UNIVERSITY
State University of New York

Thank you!

Questions?

BINGHAMTON UNIVERSITY
State University of New York

Спасибо большое

какие-нибудь вопросы?

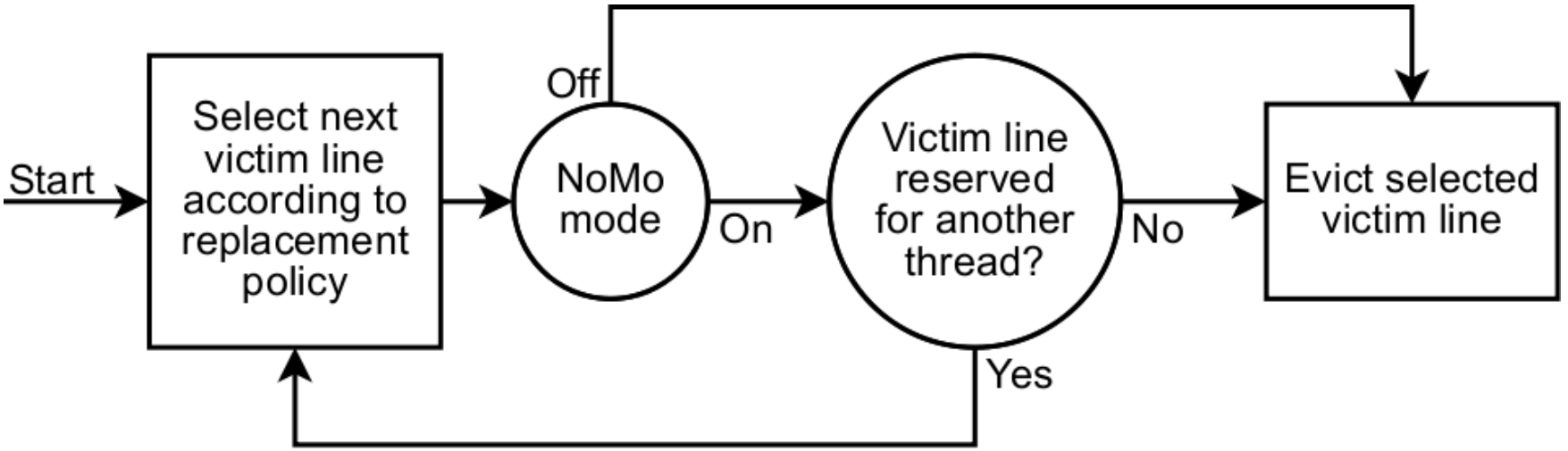
BINGHAMTON UNIVERSITY
State University of New York

Example of Partial Side Channel Closure: NoMo Caches

- Key idea: An application sharing cache cannot use all lines in a set
- NoMo invariant: For an *N*-way cache, a thread can use at most $N - Y$ lines
  - Y – NoMo degree
  - Essentially, we reserve Y cache ways for each co-executing application and dynamically share the rest
  - If Y=N/2, we have static non-overlapping cache partitioning
  - Implementation is very simple – just need to check the reservation bits at the time of replacement

BINGHAMTON UNIVERSITY
State University of New York

NoMo Replacement Logic



BINGHAMTON UNIVERSITY
State University of New York

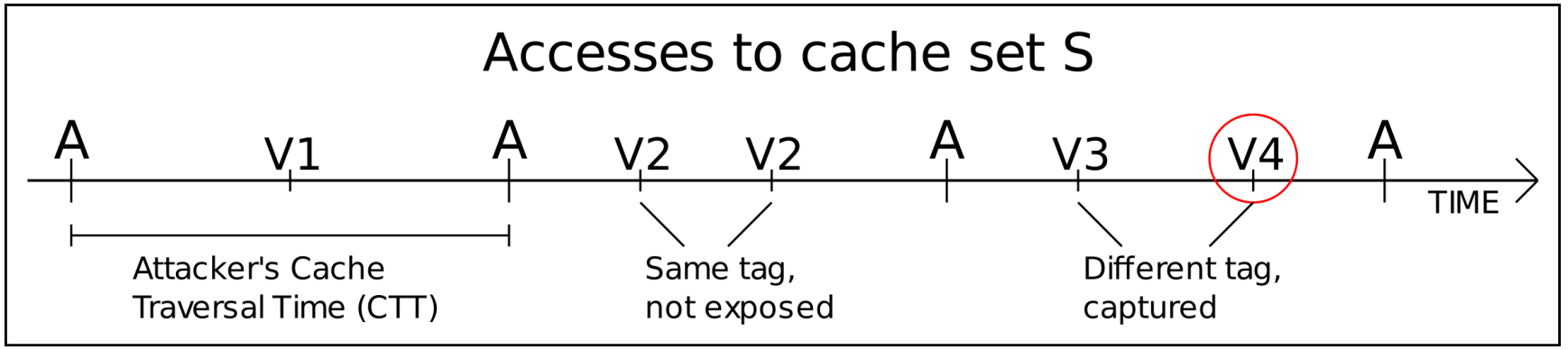
NoMo example for an 8-way cache

Shared way usage							
F:1	H:1	R:2		Q:2		K:1	
	A:1				P:1		
G:1	B:1			J:1		***N:1***	D:1
M:1	L:1	T:2	S:2	I:1	***U:2***	O:1	E:1

- Showing 4 lines of an 8-way cache with NoMo-2
- *X*:*N* means data *X* from thread *N*

BINGHAMTON UNIVERSITY
State University of New York

Why Does NoMo Work?



- Victim's accesses become visible to attacker only if the victim has accesses outside of its allocated partition between two cache fills by the attacker.

- In this example: NoMo-1

BINGHAMTON UNIVERSITY
State University of New York

Evaluation Methodology

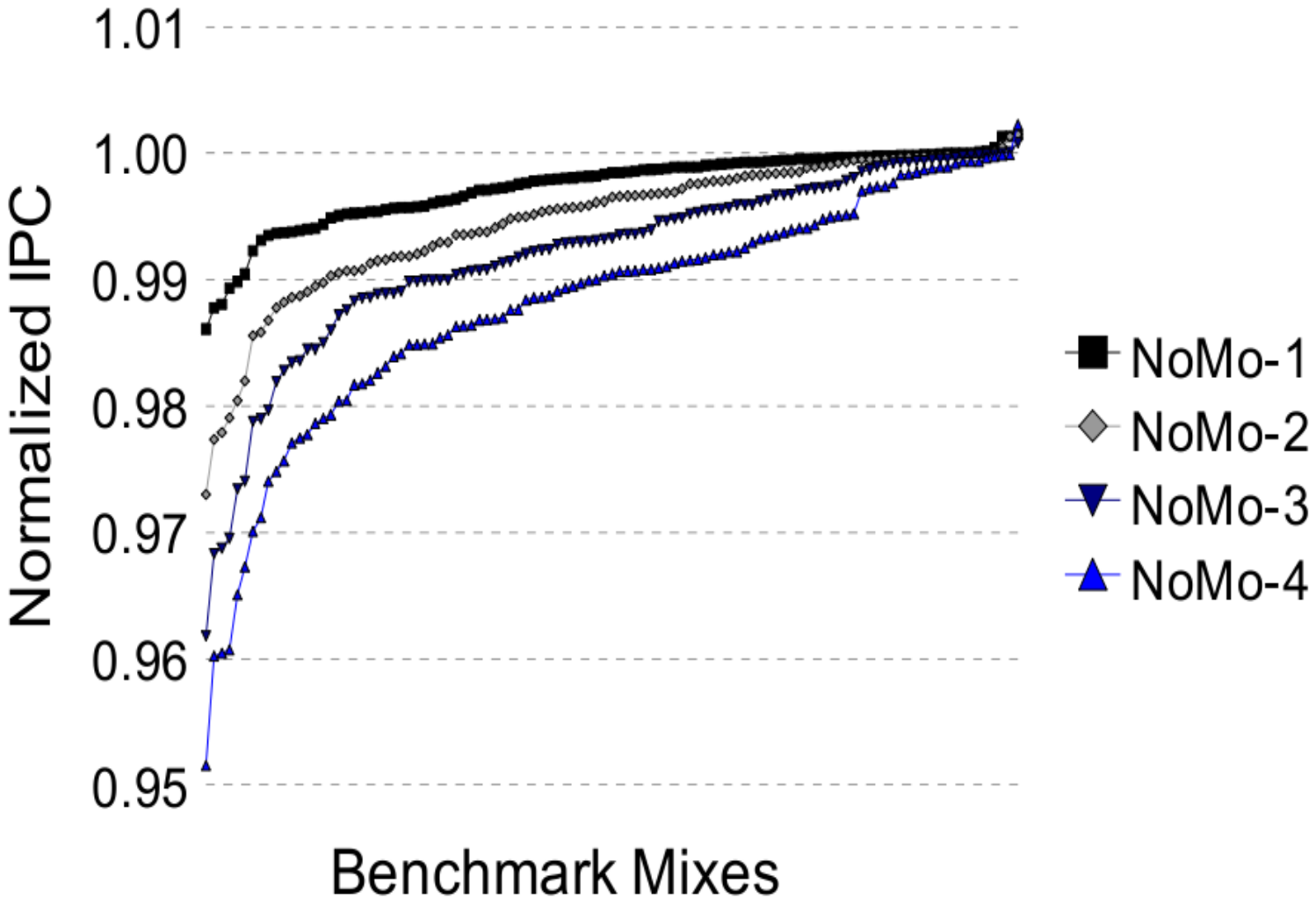
- We used M-Sim-3.0 cycle accurate simulator (multithreaded and Multicores derivative of Simplescalar) developed at SUNY Binghamton
  - http://www.cs.binghamton.edu/~msim
- Evaluated security for AES and Blowfish encryption/decryption
- Ran security benchmarks for 3M blocks of randomly generated input
- Implemented the attacker as a separate thread and ran it alongside the crypto processes
- Assumed that the attacker is able to synchronize at the block encryption boundaries (i.e. It fills the cache after each block encryption and checks the cache after the encryption)
- Evaluated performance on a set of SPEC 2006 Benchmarks. Used Pin-based trace-driven simulator with Pintools.

BINGHAMTON UNIVERSITY
State University of New York

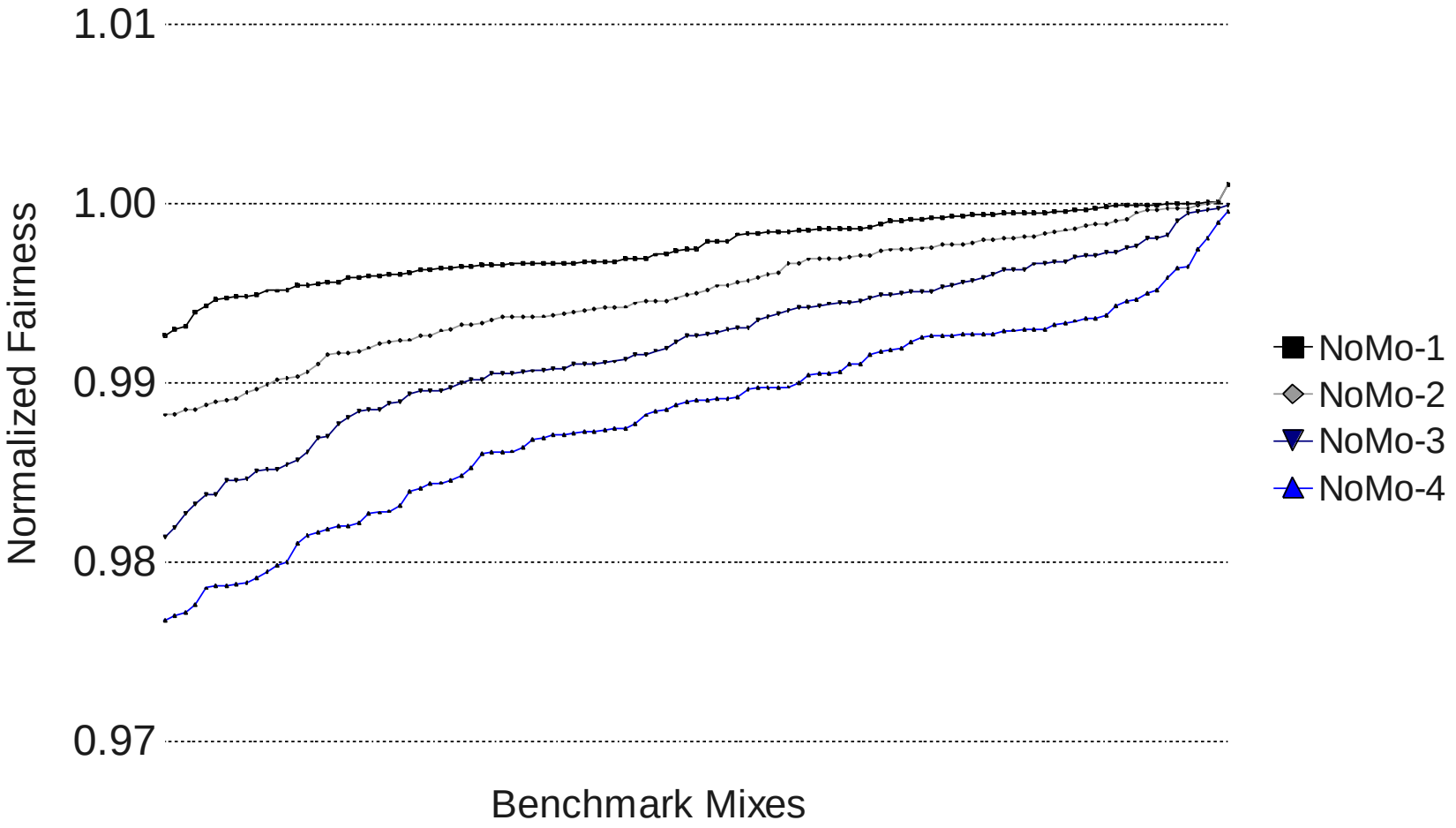
Sets with Critical Exposure

	AES enc.	AES dec.	BF enc.	BF dec.
NoMo-0	128	128	128	128
NoMo-1	128	128	128	128
NoMo-2	10	14	22	22
NoMo-3	0	0	1	1
NoMo-4	0	0	0	0

Impact on IPC Throughput (105 2-threaded SPEC 2006 workloads simulated)



Impact on Fair Throughput (105 2-threaded SPEC 2006 workloads simulated)



NoMo Design Summary

- Practical and low-overhead hardware-only design for defeating access-driven cache-based side channel attacks
- Can easily adjust security-performance trade-offs by manipulating degree of NoMo
- Can support unrestricted cache usage in single-threaded mode
- Performance impact is very low in all cases
- No OS or ISA support required

BINGHAMTON UNIVERSITY
State University of New York

NoMo Results Summary (for an 8-way L1 cache)

- **NoMo-4 (static partitioning):** complete application isolation with 1.2% average (5% max) performance and fairness impact on SPEC 2006 benchmarks
- **NoMo-3:** No side channel for AES, and 0.07% critical leakage for Blowfish. 0.8% average(4% max) performance impact on SPEC 2006 benchmarks
- **NoMo-2:** Leaks 0.6% of critical accesses for AES and 1.6% for Blowfish. 0.5% average (3% max) performance impact on SPEC 2006 benchmarks
- **NoMo-1:** Leaks 15% of critical accesses for AES and 18% for Blowfish. 0.3% average (2% max) performance impact on SPEC 2006 benchmarks