Mitigating Power Attacks through Fine-Grained Instruction Reordering

Y. Chen*, A. Hajiabadi*, R. Poussier, A. Diavastos, S. Bhasin, and T. E. Carlson
ACM Transactions on Architecture and Code Optimization (TACO), 2024.

arXiv:2107.11336v1 [cs.CR] 23 Jul 2021

Learning Presentation 01

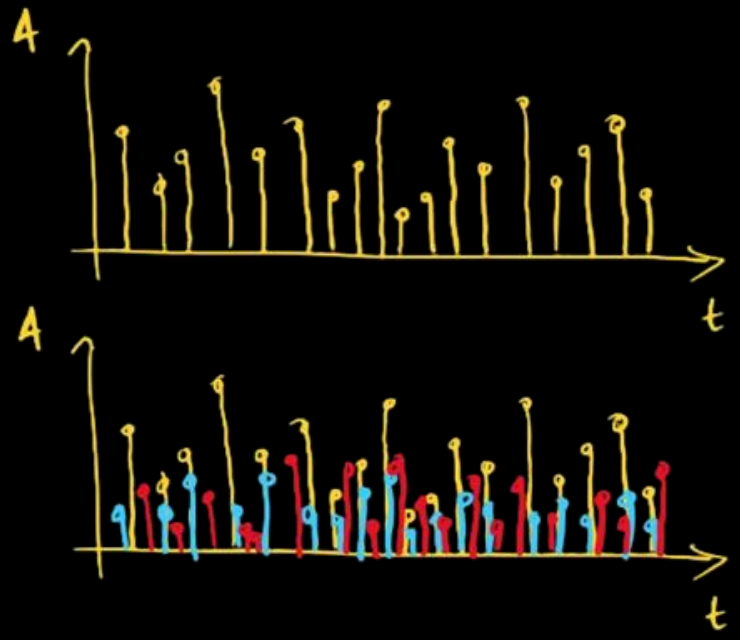
Karthik B K <ns24z023@smail.iitm.ac.in>

Logistics

- All blue slides are personal thoughts and observations
- This presentation is going to be a "night"mare for all those who do not like dark mode :)

Threat Model and Countermeasures

- Adversary can exploit the physical characteristics to leak sensitive information
- Adversary has physical access to the device under test with full control
- Adversary may have access to the device with full control
- Countermeasures classified into two:
  - Masking
    - Splits sensitive information
    - Only effective in the presence of noise
  - Hiding
    - Lowers the SNR
    - Noise in amplitude and time
      + Clock jitter
      + Parallel execution
  - Hiding provides low SNR for masking to be effective



Contributions

- Secret hiding by dynamic instruction scheduling
  - Evaluated on a GP OOO CPU based on 3rd gen BOOM
- Three different security evaluation techniques
  - Basic
    - Standard CPA with 10 M attack traces
  - Educated
    - Time integration prior to performing CPA
  - Advanced
    - Adversary can profile leakages – complete control
- Overheads – 1.1% perf, 0,7% area

OOO – Effects on Power Attacks

- Non-linear execution paths
- Variability in timing and power consumption
  - CPA relies on consistent power profiles
- Less predictable cache access patterns
- Few instructions – predictable power consumption
- Authors attack AES128 on baseline versions
  - io, ooo – 500, 1800 power traces

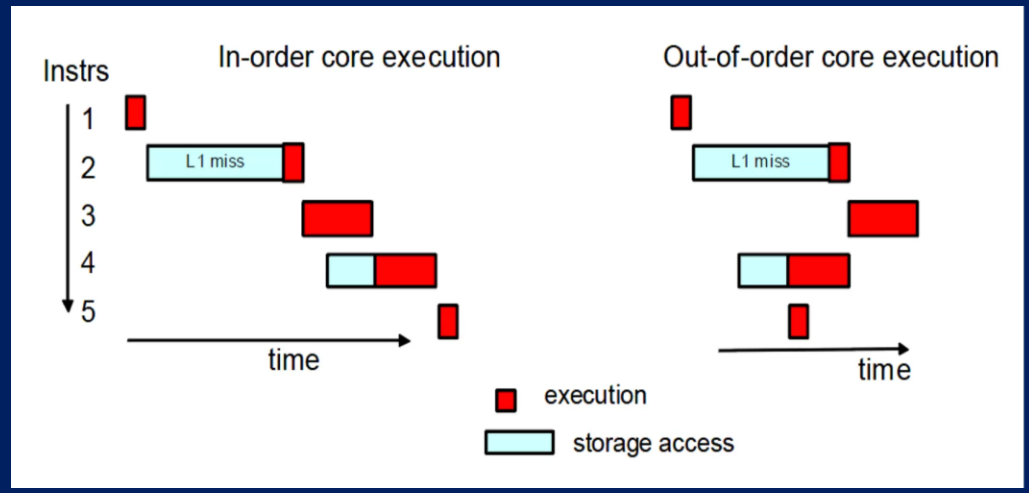
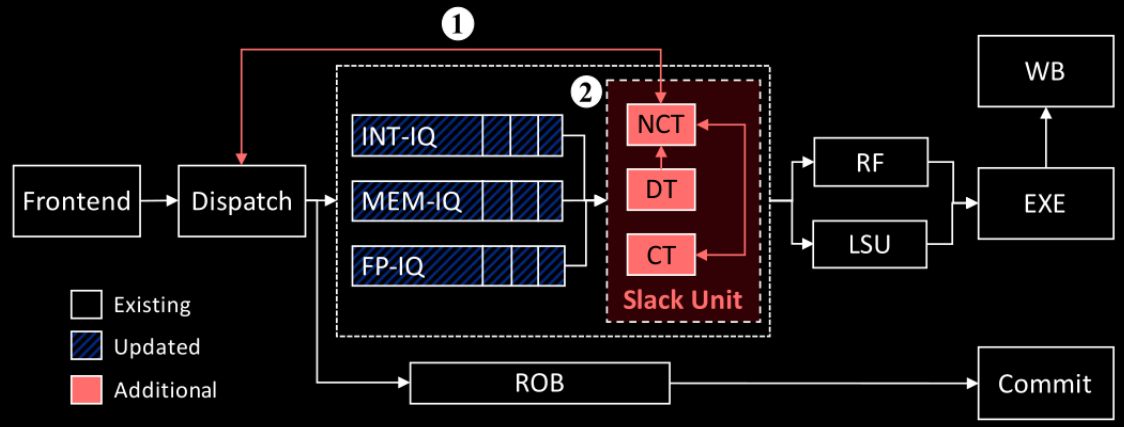


image stolen from the internet.

PARADISE uArch

- New Slack Unit (SU)
- Slack unit talks to sched. and IQs
- Slack recorded on first issue
- "Delay" is in number of cycles. Not ns.
- That would be significantly harder ?
- Steps to inject delay:
  1. Slack unit is looked up using the current D-IP. Hit results in a delay being injected.
  2. Slack unit is notified of the applied delay after successful delayed-dispatch.
- Slack unit has 3 SA structures using LRU repl.
  1. DT – Destination Table
  2. NCT – Non-Critical Table
  3. CT – Critical Table



Destination Table (DT)

PC (12 bits)	Non-critical PC offset (8 bits)
0x1310	192
0x1320	4
0x1330	8
...	...

Critical Table (CT)

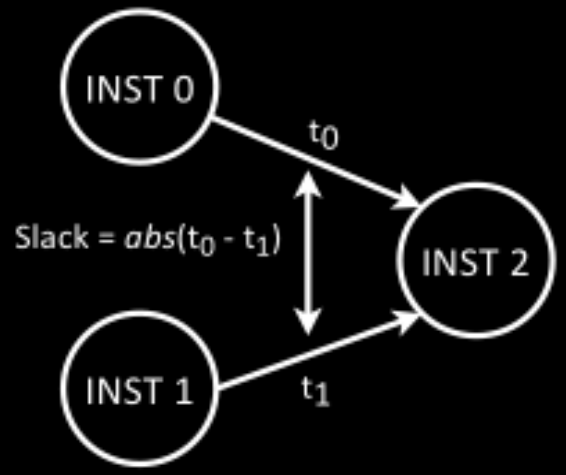
PC (12 bits)
0x1400
0x1410
0x1420
...

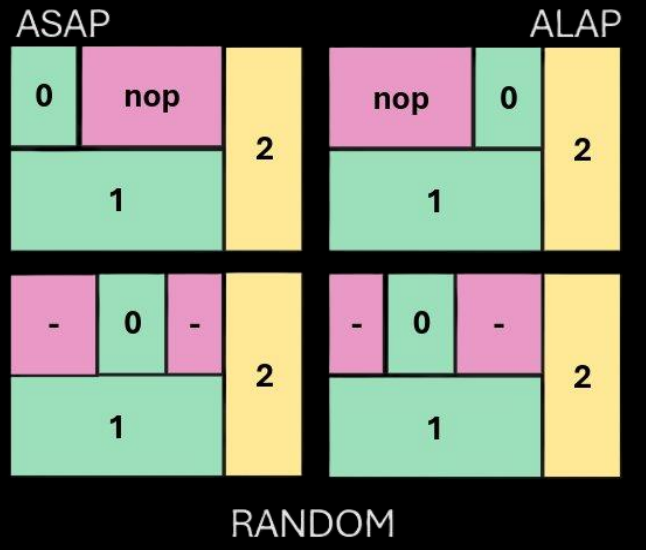
Non-Critical Table (NCT)

PC (12 bits)	Slack (5 bits)	Stable (1 bit)
0x1210	10	0
0x1220	5	1
0x1230	0	1
...	...	...

Critical Path of Execution and Slack Unit

- Operands of INST 2 are available at different times
  - Later is critical
- Slack – time difference in operand availability
- Tracking the criticality allows delayed issue





Slack Unit (contd.)

- DT holds the dependent inst. and the non-critical producer
  - Allocated upon issuing the D-IP
- NCT holds the NC producer and their total slack
  - Also captures stability (consistent non-criticality) as bool
- CT holds the D-IP of critical insts.
  - One may be critical and non-critical at two separate occurrences
- Delay randomisation uses an LFSR (on a GF)
  - Treated as a black-box for this presentation.

Critical Table (CT)

PC (12 bits)
0x1400
0x1410
0x1420
...

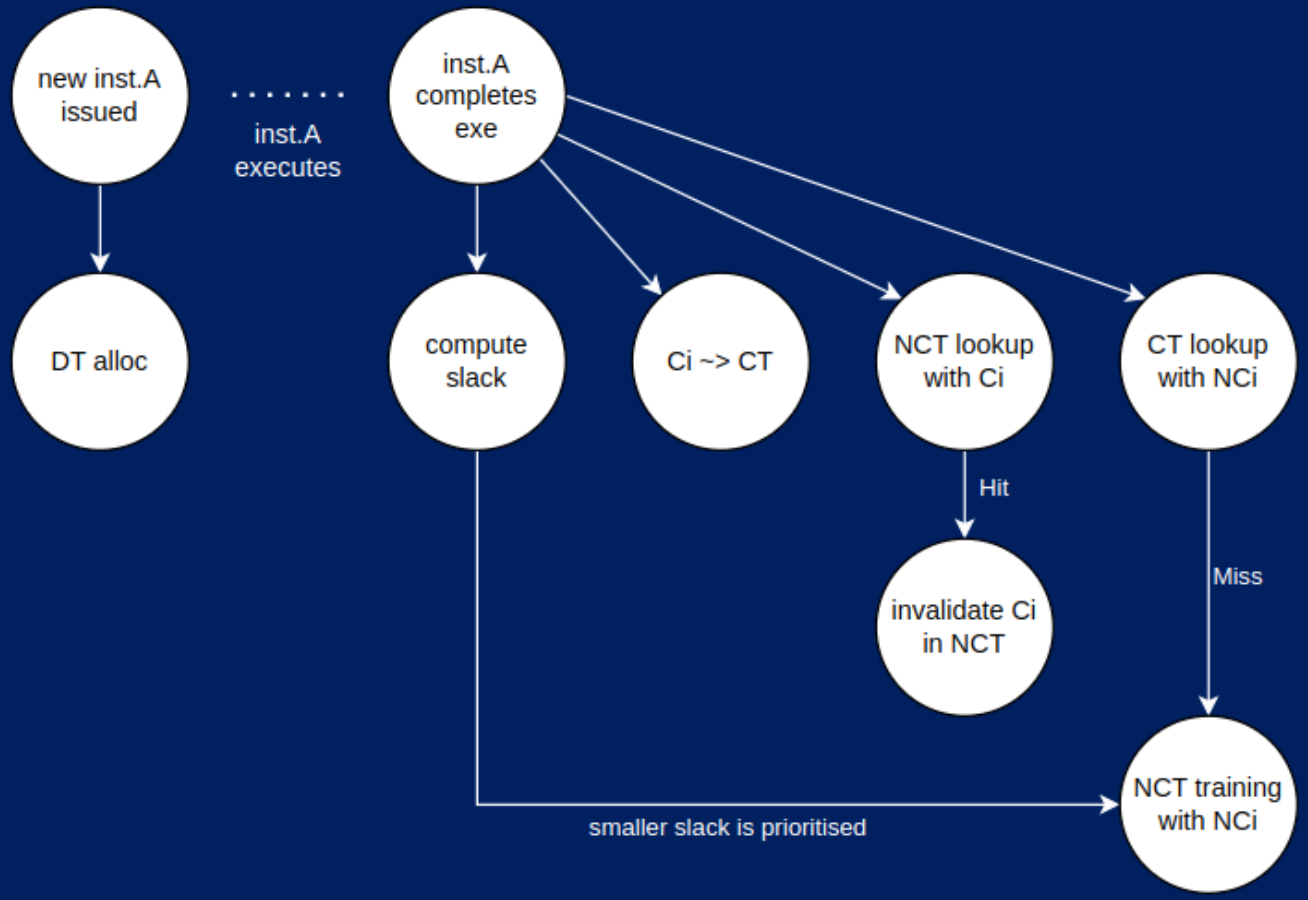
Destination Table (DT)

PC (12 bits)	Non-critical PC offset (8 bits)
0x1310	192
0x1320	4
0x1330	8
...	...

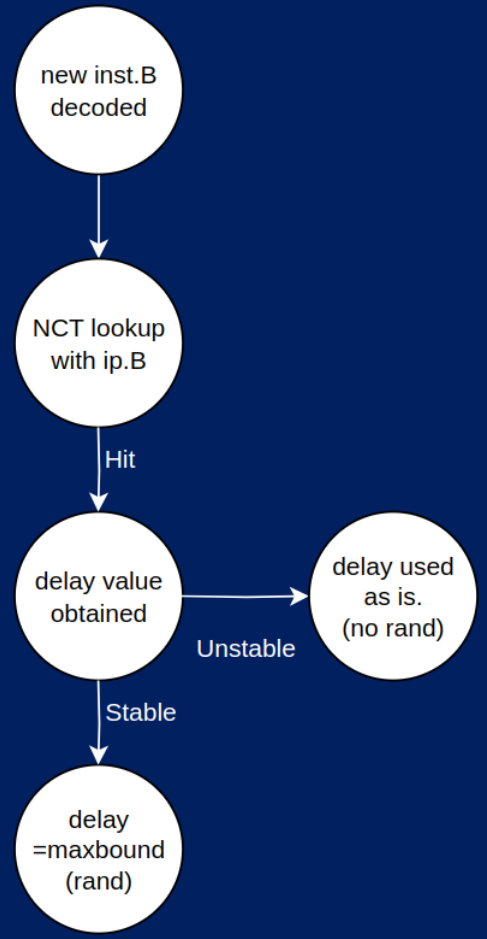
Non-Critical Table (NCT)

PC (12 bits)	Slack (5 bits)	Stable (1 bit)
0x1210	10	0
0x1220	5	1
0x1230	0	1
...	...	...

Criticality, Slack Detection, and Delay Injection



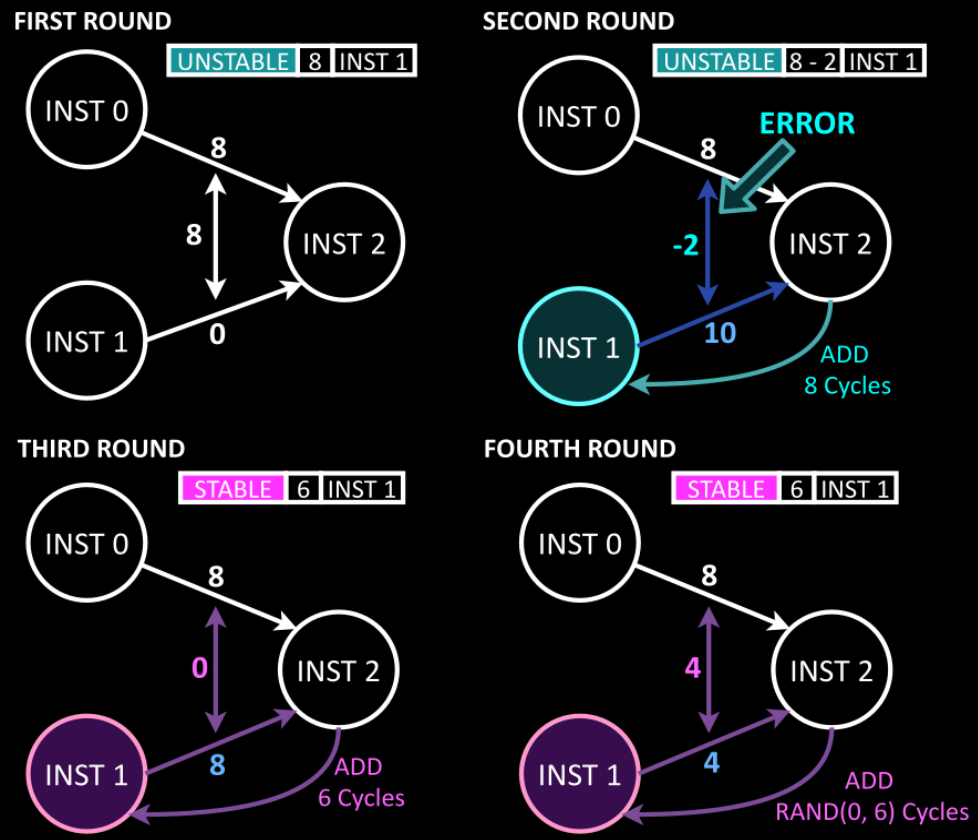
Training



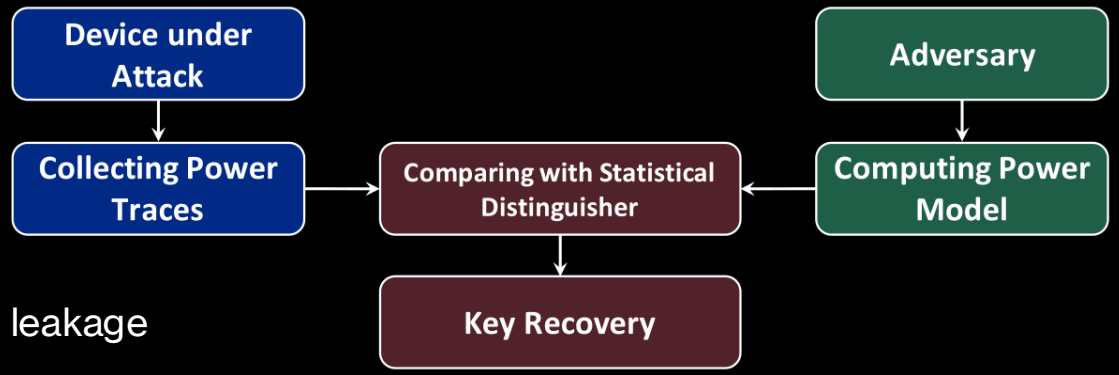
Consumption

Delay Injection (ex)

1. Completion time for INST 0, 1 noted when INST 2 issued
   + Slack = 8 cycles
   + INST 0 is critical, likewise INST 1
   + Unstable
2. INST 1 unstable – inject delay as is
   + Possible back-pressure
   + slack_upd = slack_old – slack_new
3. INST 1 still unstable – inject slack_upd
   + No error => marked stable
4. INST 1 stable – slack range(0, 6) randomized



Security Evaluation



- Information leakage through power side-channels
  - First order – in the statistical mean
  - Second order – in the statistical variance/co-variance
- TVLA for first order leakage assessment
  - Secure masking countermeasure is applicable only for first-order leakage
- Pass/fail T-test for hiding style of countermeasures must be avoided
- Correlation Power Analysis
  - Univariate – one sample at a time
  - Multivariate – combines multiple samples prior to correlation – sub-optimal
    - Template attacks are more effective in this case
- Very simple changes (jitter, misalignment, power balancing) will rended CPA hard to succeed
  - Failed CPA under sub-optimal assumptions != a sound security evaluation

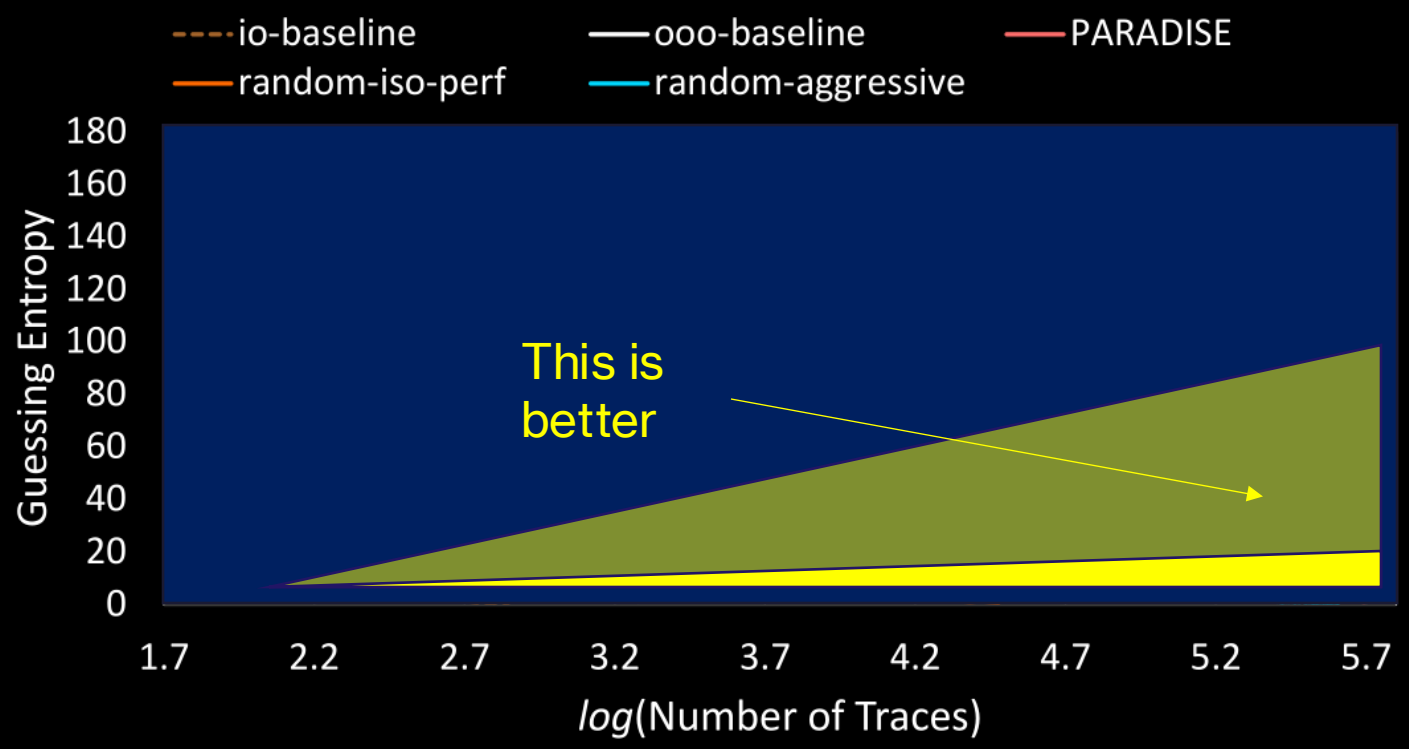
Proposed SE Methodology

- On GP-OOO-CPU
  1. AES128 with 2000 PTs.
  2. u-benchmarks by Chipyard.
  3. Subset of SPEC CPU2017
- Types of adversaries
  1. Basic – simple applies standard CPA, HW model on s-box output
  2. Educated – integrates leakage over different time samples
  3. Advanced – uses a profiling/training set for a multivariate template attack
     - Also uses profiled PCA for dimensionality reduction
- Provides a confident lower-bound on security (assuming the current threat model)
- Data Collection and Hypothetical Power Model -
  1. Two sets of 1M traces each.
     - First, fixed key (unknown), random PT.
     - Second, random key and PT (known)
  2. HammingWeight(rdvalue)
  3. No bus/memory value since that is much larger in aplitude
     - 33x higher for IRF
     - 61x higher for IRF+FRF
- Evaluated on -
  1. io-baseline – unprotected in-order (Berkeley Rocket)
  2. ooo-baseline – unprotected out-of-order (SonicBOOM)
  3. PARADISE – SonicBOOM + Slack Unit
  4. random-iso-perf – SonicBOOM + random delay (upto 8 cycles with 5% probability)
  5. random-iso-security – SonicBOOM + random delay (upto 8 cycles with 20% probability)
  6. random-aggressive – SonicBOOM + random delay (upto 8 cycles with 100% probability - naive)
- Simulation Setup (Synopsys)
  1. dc, vcs for synth + GLS
  2. PrimePower for switching activity

More Logistics

Needs to meet a threshold.

Lower is better

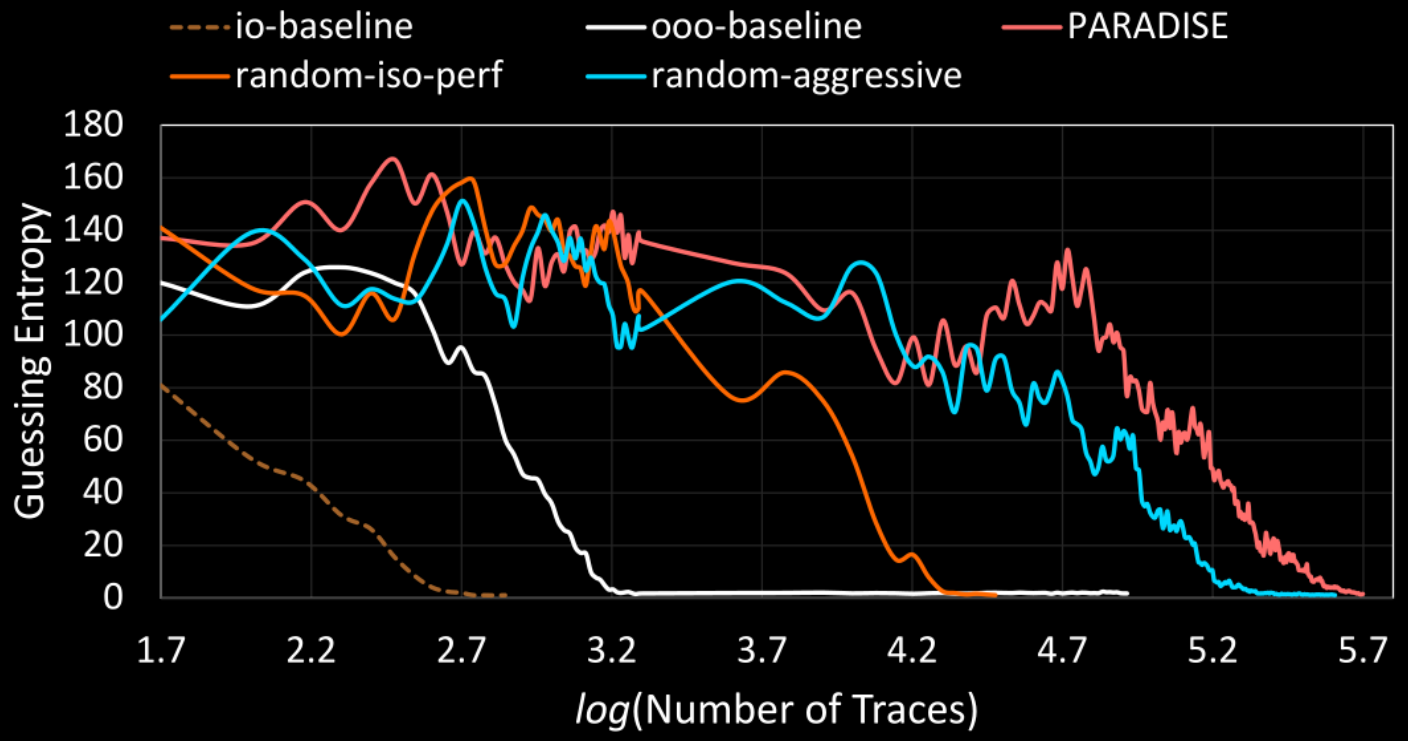


Higher the better

Basic SE

- Basic CPA
- GE=0 means correct guess was ranked highest
- random-aggressive not better than paradise
  - Only descynchronization
  - No rand in reg content (?)

Variant	Number of Traces	Security Improvement
io-baseline	500	-
ooo-baseline	1800	1x
Paradise	470k	261x
random-iso-perf	22k	12x
random-aggressive	220k	122x

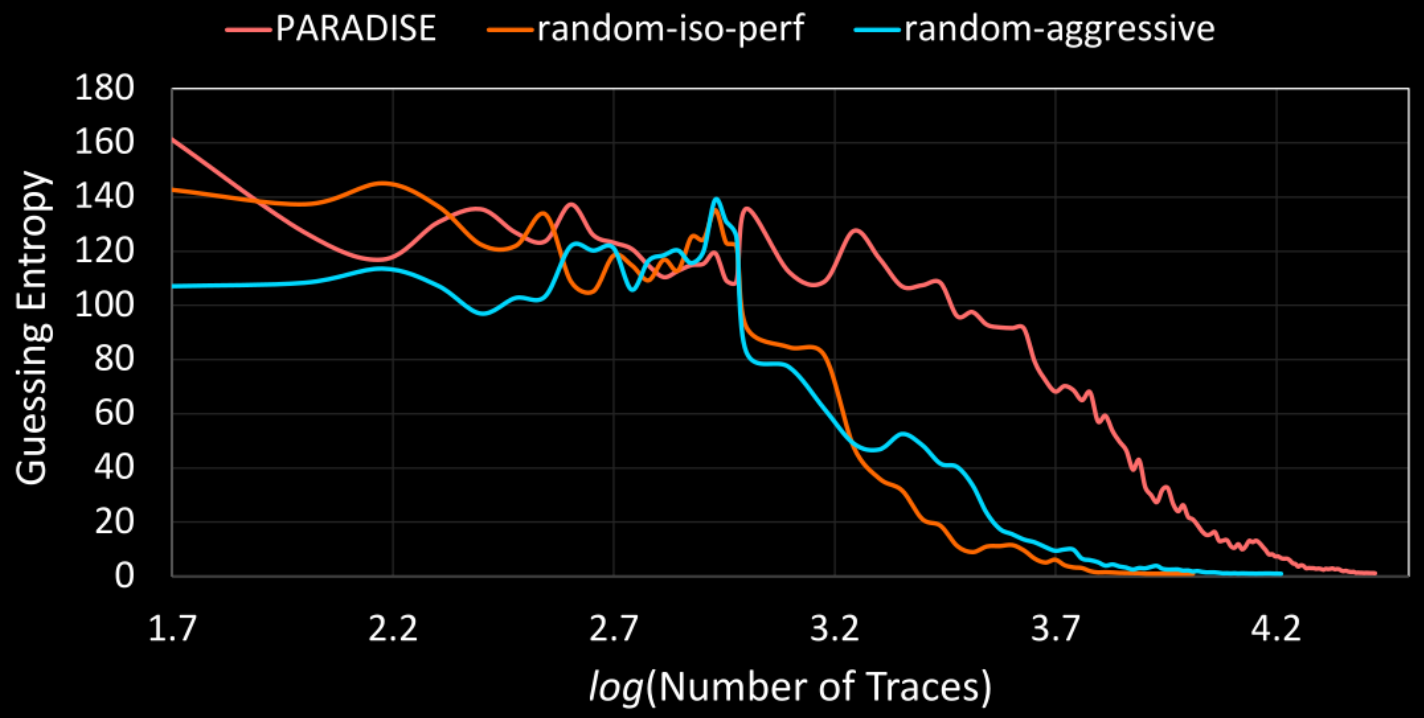


Educated SE

- Time integ: N=20, 50, 100, 150, 200
  - Consecutive power samples
  - Best results shown

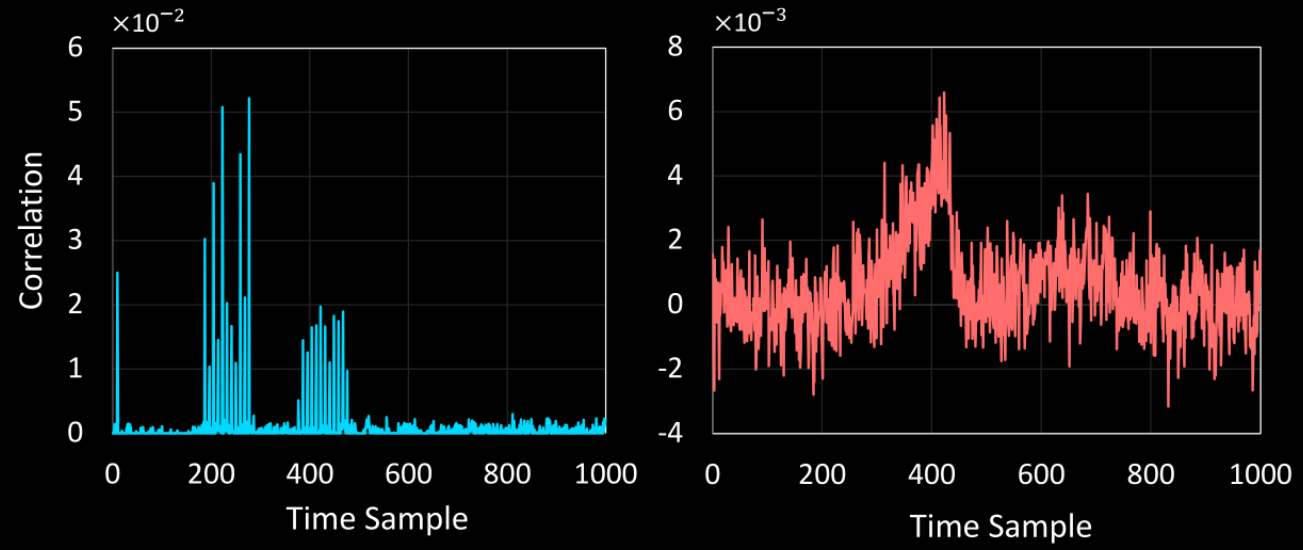
Emphasis on danger of sub-optimal attack strategies

Variant	Number of Traces	Security Improvement
Paradise	22.25k	12x
random-iso-perf	21k	11.5x
random-aggressive	20k	11x

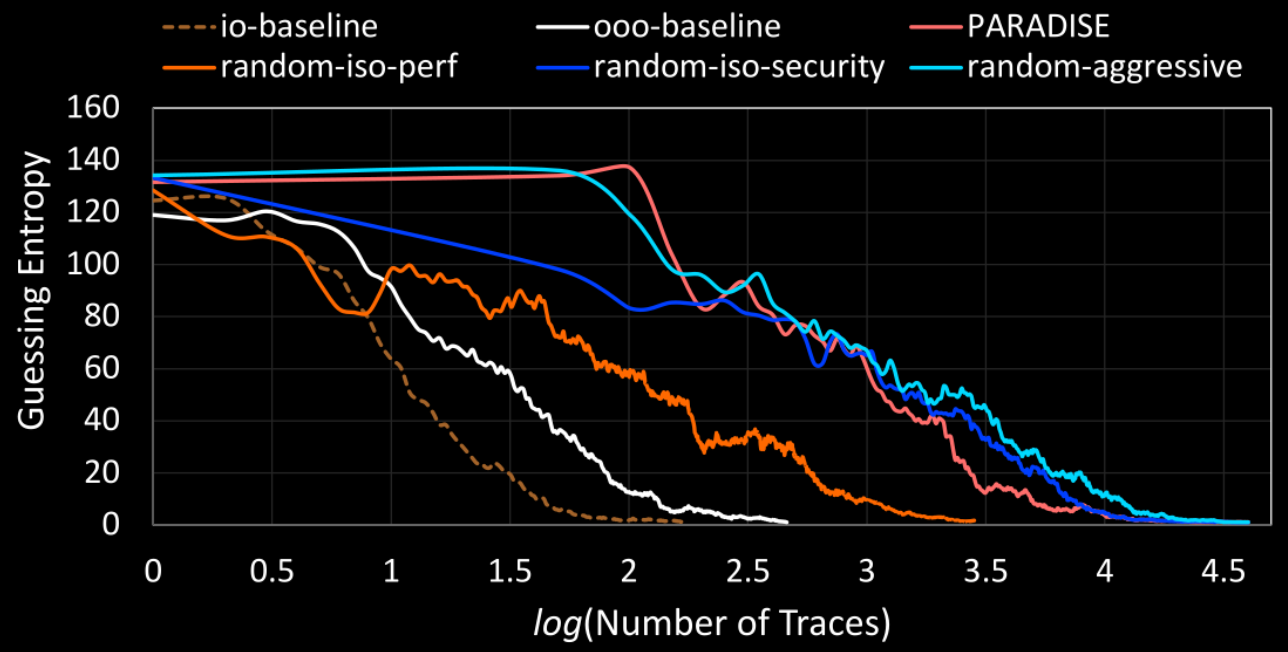


Advanced SE

- Adversary can profile leakages -- sample-efficient
- Correlation in time for ooo-baseline (L) and random-aggressive (R) with profiled CPA
- Delay injection results in a Gaussian dist. Instead of clear peaks.

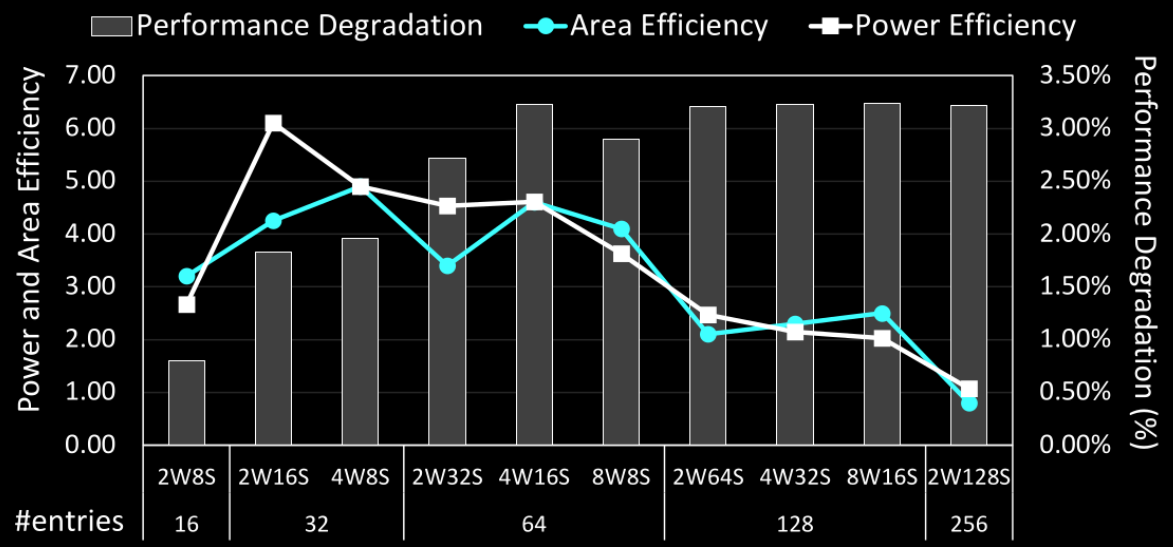


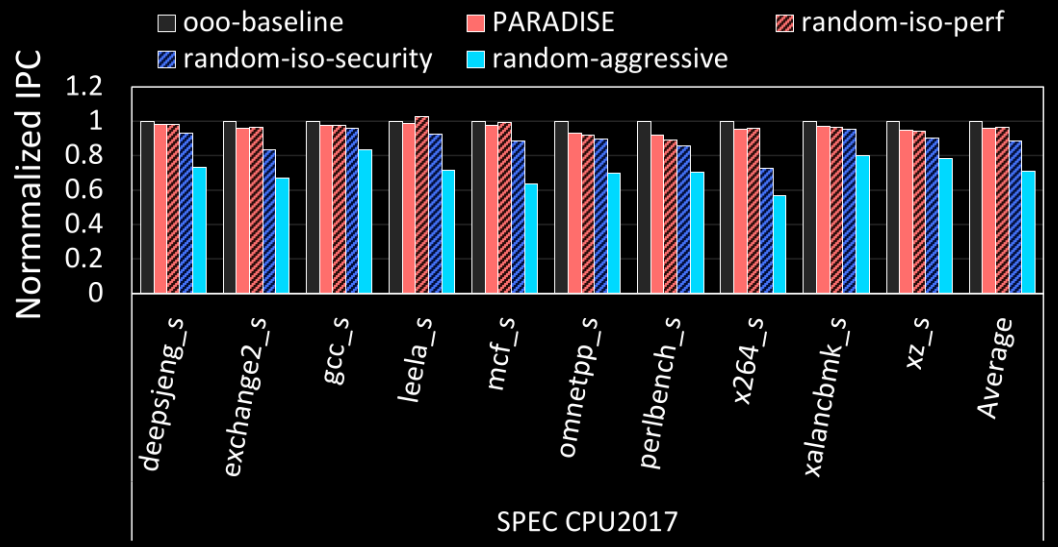
Variant	Number of Traces	Security Improvement
io-baseline	125	-
ooo-baseline	400	1x
Paradise	13.5k	34x
random-iso-perf	2.2k	5.5x
random-aggressive	15k	38x

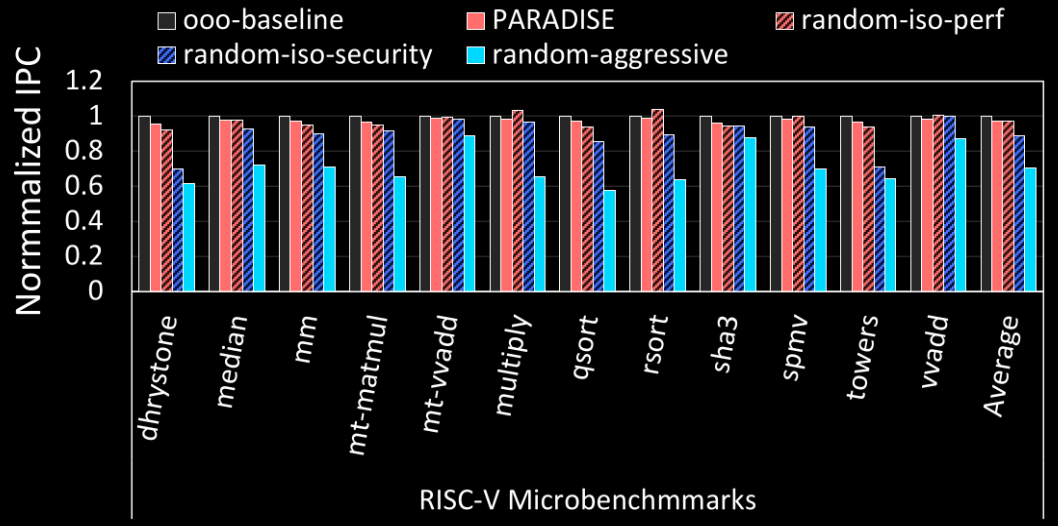


Performance Analysis

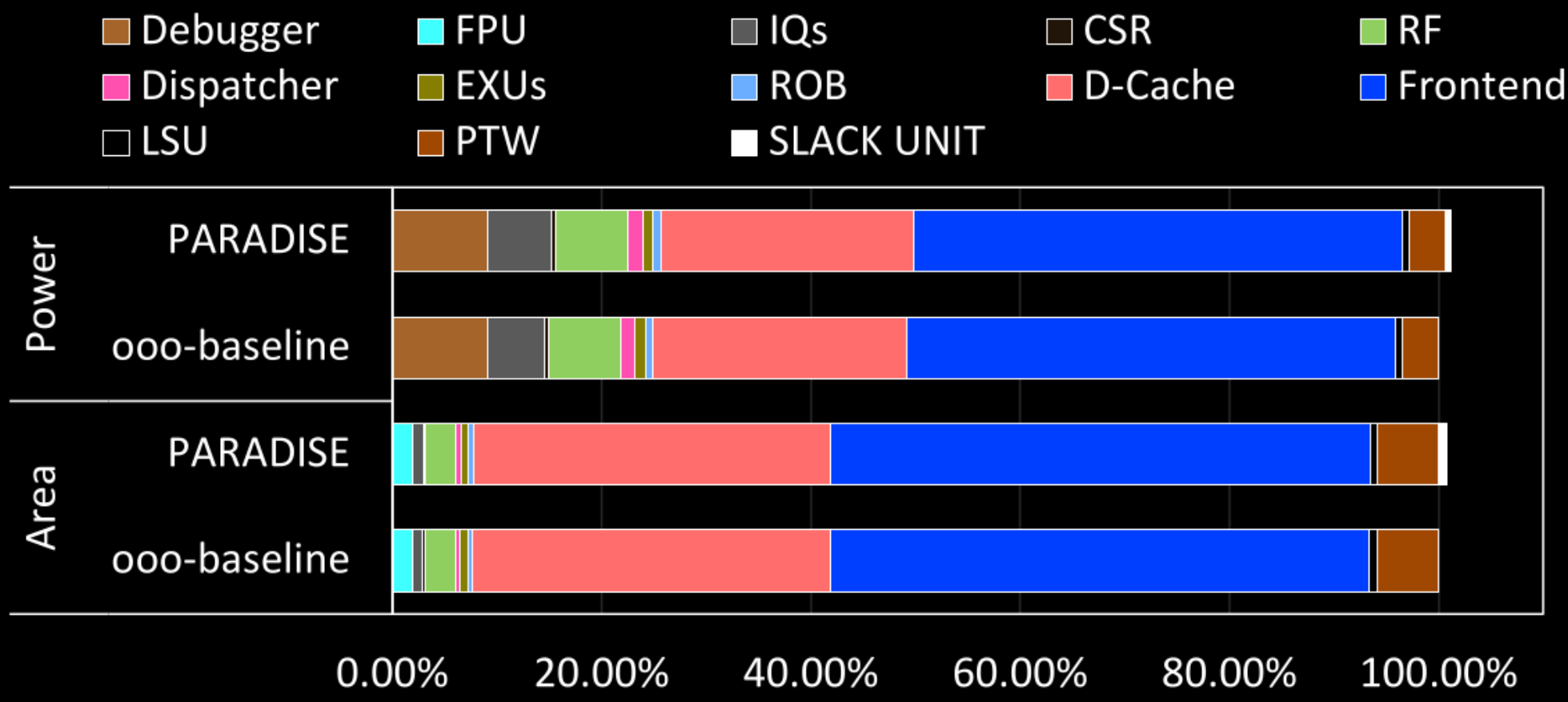
- eff.pwr = overhead.perf/overhead.pwr
- eff.area = overhead.perf/overhead.area
- Average overhead.perf = 2.6% (max=4.8%)
- Max overhead.pwr = 1.1%
- overhead.area = 0.7%
- 4w 16s SU







Power and Area Overheads



Related Works

Paper	Hardware Agnostic	Algorithm Agnostic	No Re-compile	Design	Overheads*			Security Evaluation*			Technique
					Area	Power	Performance	Basic	Educated	Advanced	
WDDL [24]			✓	VLSI	200%	300%	300%	128×	-	-	Power Balancing
IVR [25]		✓	✓	VLSI	100%	100%	0%	‡	-	-	Voltage Regulation
False-Key [50]			✓	VLSI	3%	0%	2%	187×	-	-	Gate-Level Masking
ASNI [17]		✓	✓	VLSI	60%	68%	0%	1000×	1000×	-	Noise Injection
Blinking [1]	✓	✓	✓	VLSI/SW	-	-	270%	10 – 100×	10 – 100×	-	Power Hiding
PARAM [7]	✓	✓	✓	µarch	∼20%	-	-	‡	-	-	Data Obfuscation
ARDPE [20]		✓	✓	µarch	7.23%	-	3.4%	4000×	-	-	Data Randomization
RIJID [3]	✓	✓		µarch/SW	2%	27%	30%	‡	-	-	Random Code Injection
Block Shuffler [10]	✓	✓		µarch/SW	2%	1.5%	0.7%	‡	-	-	Coarse Instr. Shuffling
random-iso-perf	✓	✓	✓	µarch	∼0%	0.8%	3.8%	12×	12×	5.5×	Fine Instr. Re-ordering
random-iso-security	✓	✓	✓	µarch	∼0%	0.4%	11%	-	-	34×	Fine Instr. Re-ordering
random-aggressive	✓	✓	✓	µarch	∼0%	0%	29%	122×	11×	38×	Fine Instr. Re-ordering
PARADISE (this work)	✓	✓	✓	µarch	0.7%	1.1%	3.7%	261×	12×	34×	Fine Instr. Re-ordering

PARADISE improves security against power analysis attacks by 34x to 261x with power and area overheads of 1.1% and 0.7%, respectively. Moreover, the system achieves performance within 96%, on average, of the ooo-baseline unprotected processor.

This slide is left blank intentionally.