SQUIP and Why We Need To Study Processors Like Nature

Stefan Gast, Daniel Gruss

2023-05-11

Graz University of Technology

Side Channels are Everywhere


How did this song go again?
Please don't...
Side Channels are everywhere ...
imgflip.com

Side Channels Through the Years

- 1996: Runtime could leak crypto keys
- 2004: Runtime does leak crypto keys
- 2006: Cache leaks crypto keys
- 2009: Cache leaks info on function calls in other programs
- 2011: Cache leaks info on co-located VMs
- 2013: Cache breaks (K)ASLR
- 2014: Cache flushing allows Rowhammering in software
- 2014: Cache leaks highly accurate function call traces
- 2015: Attacks from JavaScript
- 2015: Cache-based partial keylogger
- 2015: Cache eviction $\rightarrow$ Rowhammer in JavaScript
- 2016: Mobile and non-Mobile devices affected alike
- 2017: Video streaming through a cache side channel

What about Mitigations?

Google Scholar "provable" "side channel" mitigation

Artikel Ungefähr 3 070 Ergebnisse (**0,06** Sek.)

Beliebige Zeit
Seit 2023
Seit 2022
Seit 2019
Zeitraum wählen...

Nach Relevanz sortieren
Nach Datum sortieren

Beliebige Sprache
Seiten auf Deutsch

Alle Typen
Übersichtsarbeiten

☐ Patente einschließen
☑ Zitate einschließen

Alert erstellen

Verifiable **side-channel** security of cryptographic implementations: constant-time MEE-CBC [PDF] inesctec.pt
JB Almeida, M Barbosa, G Barthe... - Fast Software Encryption ..., 2016 - Springer
... methodologies effectively **mitigate side-channel** leakage; for ... protocols before the emergence of **provable** security. On the ... we bring the mathematical guarantees of **provable** security to ...
☆ Speichern 99 Zitieren Zitiert von: 66 Ähnliche Artikel Alle 15 Versionen

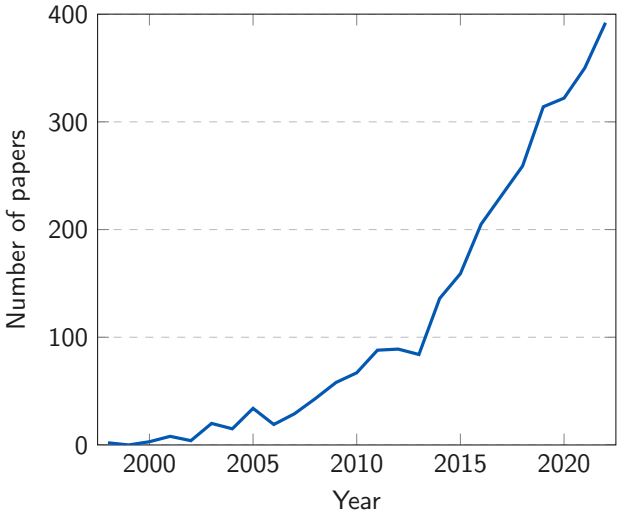
[PDF] An efficient **mitigation** method for timing side channels on the web [PDF] researchgate.net
S Schinzel - ... Workshop on Constructive **Side-Channel** Analysis ..., 2011 - researchgate.net
... a **provable** security ... **side channel** attack, and secondly the performance impact on the system. The rest of this paper is organised as follows: In section 2, we describe timing **side channel** ...
☆ Speichern 99 Zitieren Zitiert von: 30 Ähnliche Artikel Alle 3 Versionen »

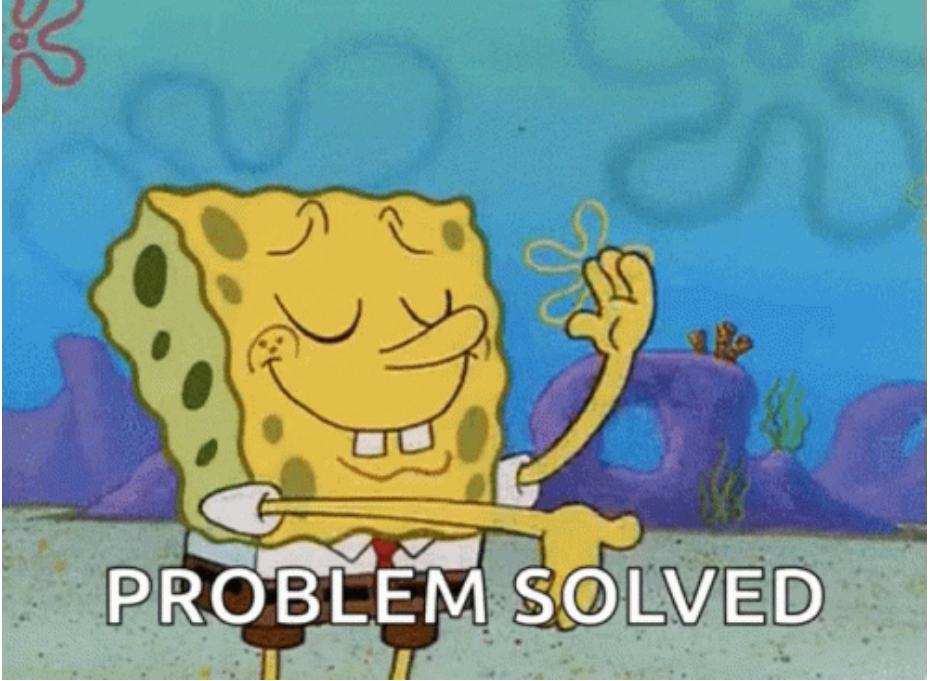
An approach for symmetric encryption against **side channel** attacks in **provable** security
W Li, D Gu - **Provable** Security: First International Conference ..., 2007 - Springer
... in **side channel** attacks) and IND-CPASCA (indistinguishability of chosen plaintext attacks and **side channel** ... -SCA by **reduction**, and IND-CPASCA is stronger than IND-CPA or UB-SCA. ...
☆ Speichern 99 Zitieren Zitiert von: 3 Ähnliche Artikel Alle 7 Versionen

Provable Secure Software Masking in the Real-World [PDF] iacr.org
A Beckers, L Wouters, B Gierlichs, B Preneel... - ... **Side-Channel** Analysis ..., 2022 - Springer
... several times while acquiring **side-channel** information. **Side-channel** information can come ... To **mitigate** these SCA attacks an implementer generally tries to break the relation between ...
☆ Speichern 99 Zitieren Zitiert von: 5 Ähnliche Artikel Alle 9 Versionen

Universal Exponentiation Algorithm A First Step towards ***Provable*** SPA-Resistance [PDF] psu.edu
C Clavier, M Joye - ... Hardware and Embedded Systems—CHES 2001 ..., 2001 - Springer
... We provide in this way a kind of **reduction**. Instead of carefully analyzing a specific ... has access to some **side-channel** information. Depending on the **side-channel** information and the ...

Provable Side-Channel Mitigations Through the Years




PROBLEM SOLVED

Google Scholar "side channel attacks"

Artikel Ungefähr 1 120 Ergebnisse (0,12 Sek.)

Beliebige Zeit
Seit 2023
Seit 2022
Seit 2019
Zeitraum wählen...

Nach Relevanz sortieren
Nach Datum sortieren

Beliebige Sprache
Seiten auf Deutsch

Alle Typen
Übersichtsarbeiten

Patente einschließen
Zitate einschließen

Alert erstellen

[PDF] It's all in your head (set): **Side-channel attacks** on ar/vr systems [PDF] usenix.org
Y Zhang, C Slocum, J Chen, N Abu-Ghazaleh - USENIX Security, 2023 - usenix.org
… This paper demonstrates that AR/VR systems are vulnerable to **sidechannel attacks** …
We develop a number of **side-channel attacks** targeting different types of private information…
☆ Speichern 99 Zitieren Zitiert von: 1 Alle 2 Versionen »

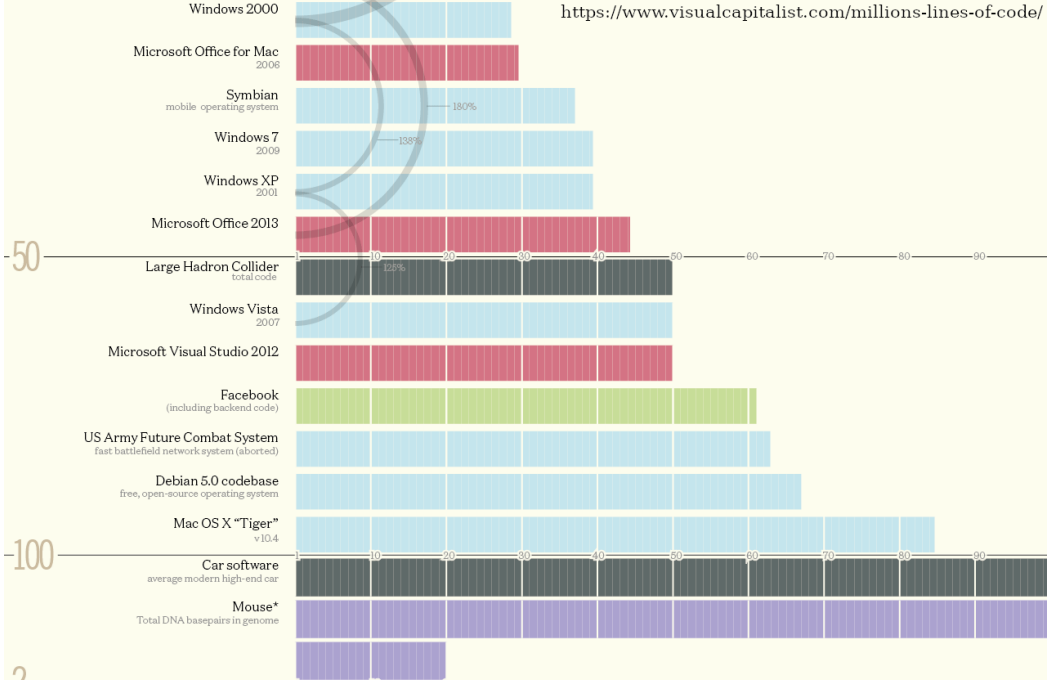
[PDF] **Side-Channel Attacks** on Optane Persistent Memory [PDF] usenix.org
S Liu, S Kanniwadi, M Schwarzl, A Kogler… - 32th USENIX Security …, 2023 - usenix.org
… persistent memory introduce new **side-channel attacks** that … The foundation of our **side-channel attacks** is a thorough … attack primitives for novel **sidechannel attacks** and covert channels, …
☆ Speichern 99 Zitieren Zitiert von: 2 Ähnliche Artikel Alle 8 Versionen »

[PDF] NVLeak: Off-Chip **Side-Channel Attacks** via Non-Volatile Memory Systems [PDF] usenix.org
Z Wang, M Taram, D Moghimi, S Swanson… - USENIX Security …, 2023 - usenix.org
… We study microarchitectural **side-channel attacks** and defenses on non-volatile RAM (NVRAM) …
Our results show that **side-channel attacks** exploiting NVRAM are practical and defeat …
☆ Speichern 99 Zitieren Zitiert von: 3 Ähnliche Artikel Alle 5 Versionen »

SoK: Deep learning-based physical side-channel analysis [PDF] acm.org
S Picek, G Perin, L Mariot, L Wu, L Batina - ACM Computing Surveys, 2023 - dl.acm.org
… Deep learning-based **side-channel attacks** entered the ield in … We irst dissect deep learning-based **side-channel attacks** … to be followed in deep learning-based **side-channel attacks**. …
☆ Speichern 99 Zitieren Zitiert von: 28 Ähnliche Artikel Alle 3 Versionen

Pushing the Limits of Generic **Side-Channel Attacks** on LWE-based KEMs-Parallel PC Oracle Attacks on Kyber KEM and Beyond [PDF] iacr.org
G Rajendran, P Ravi, JP D'Anvers, S Bhasin… - IACR Transactions on …, 2023 - tches.iacr.org
… -Checking (PC) oracle based **side-channel attacks** for Kyber KEM. These attacks operate in a … In this respect, we propose novel parallel PC oracle based **side-channel attacks**, which are …
☆ Speichern 99 Zitieren Zitiert von: 3 Ähnliche Artikel »

security


https://www.visualcapitalist.com/millions-lines-of-code/
Windows 2000
Microsoft Office for Mac
2008
Symbian
mobile operating system
180%
Windows 7
2009
138%
Windows XP
2001
Microsoft Office 2013
50
1
10
20
30
40
50
60
70
80
90
Large Hadron Collider
total code
128%
Windows Vista
2007
Microsoft Visual Studio 2012
Facebook
(including backend code)
US Army Future Combat System
fast battlefield network system (aborted)
Debian 5.0 codebase
free, open-source operating system
Mac OS X "Tiger"
v10.4
100
1
10
20
30
40
50
60
70
80
90
Car software
average modern high-end car
Mouse*
Total DNA basepairs in genome
2

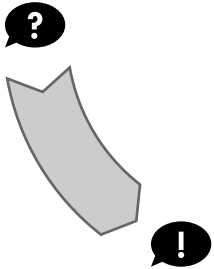
Herbert A. Simon

The Sciences of the Artificial

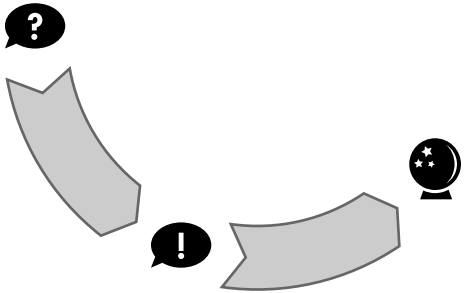
Third Edition

Artificial / Natural Science Methodology

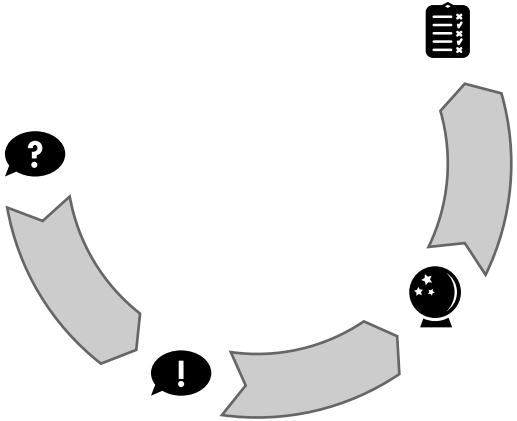
Artificial / Natural Science Methodology



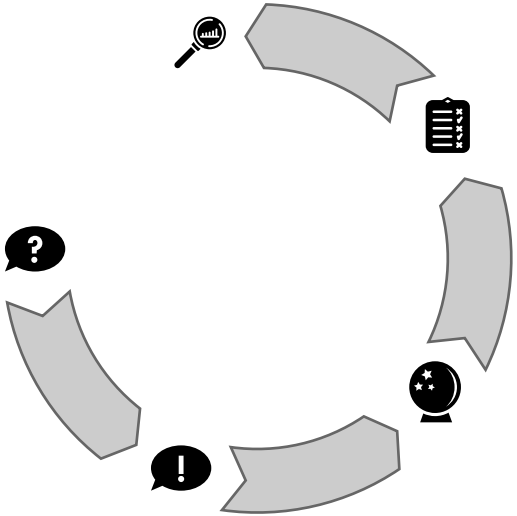
Artificial / Natural Science Methodology



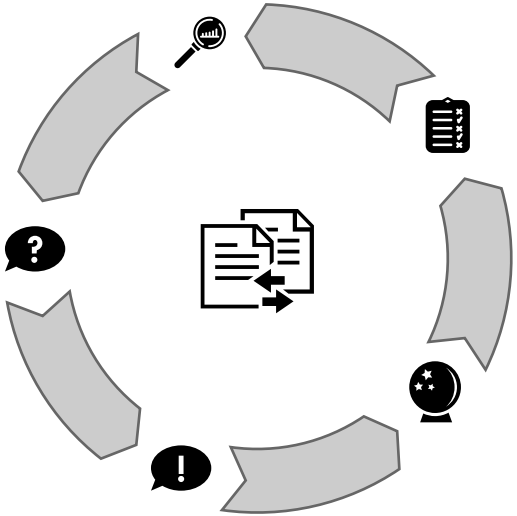
Artificial / Natural Science Methodology

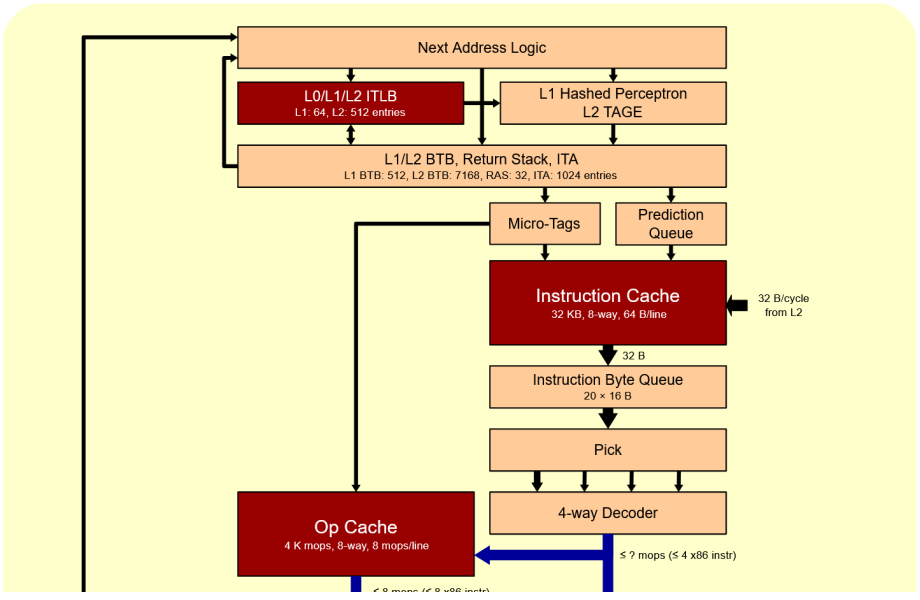


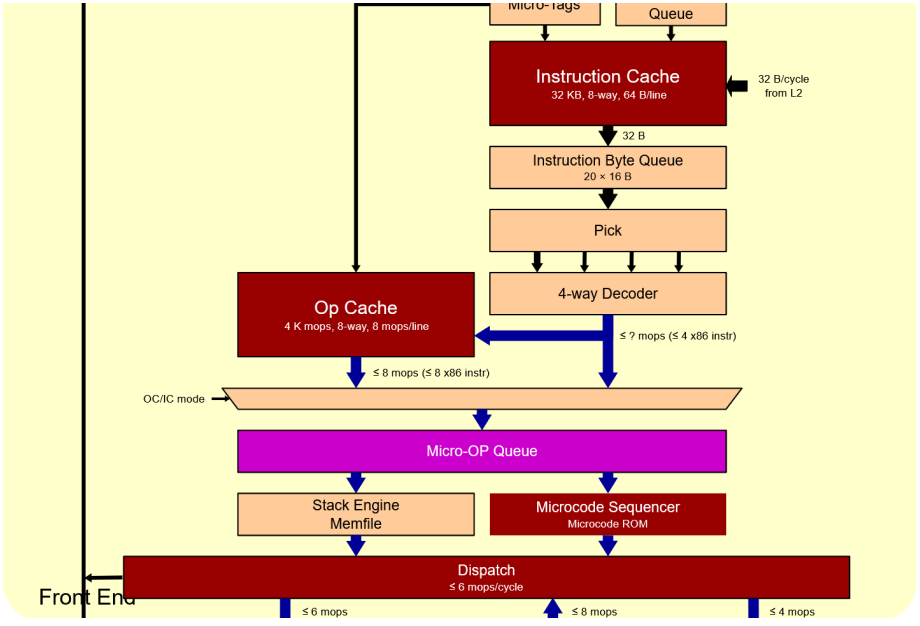
Artificial / Natural Science Methodology

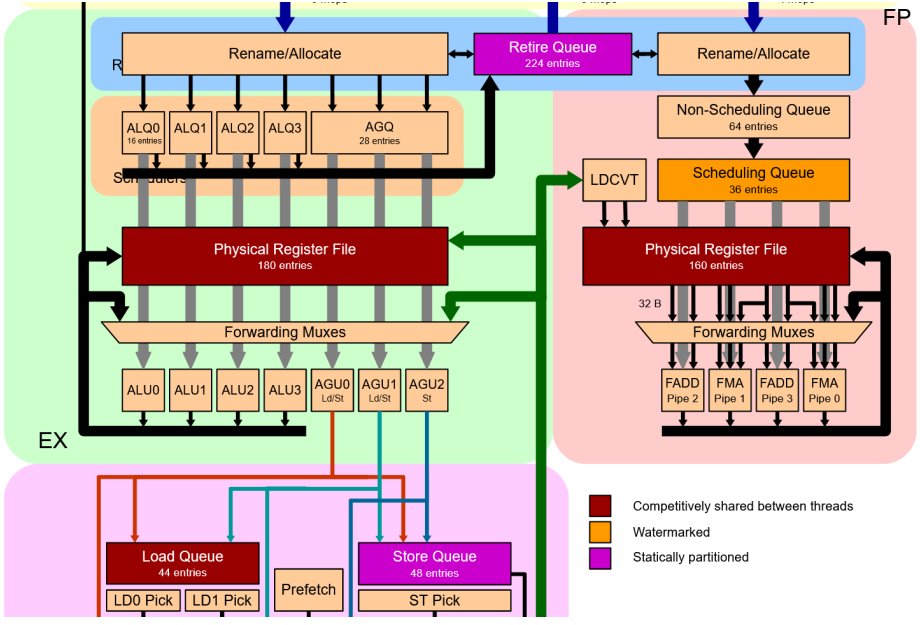


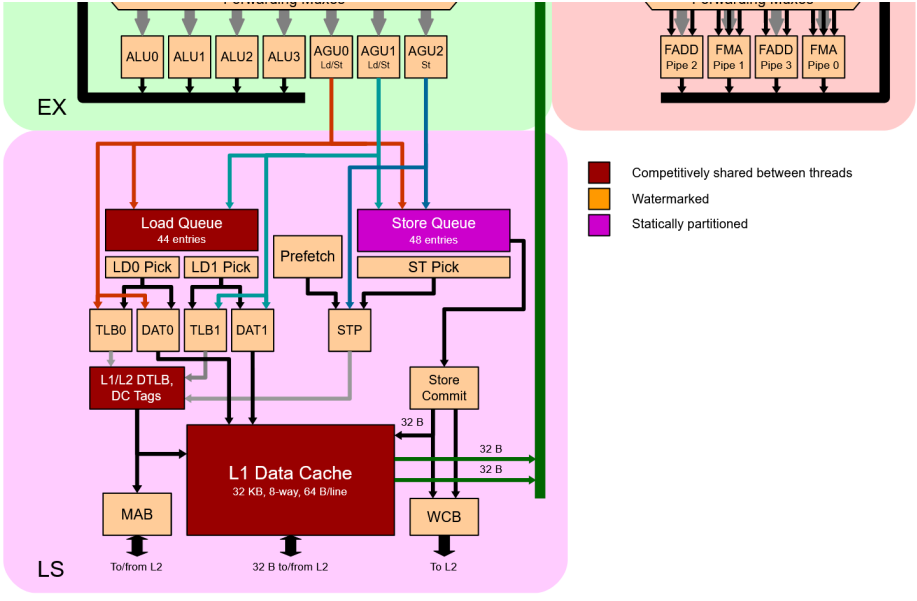
Artificial / Natural Science Methodology

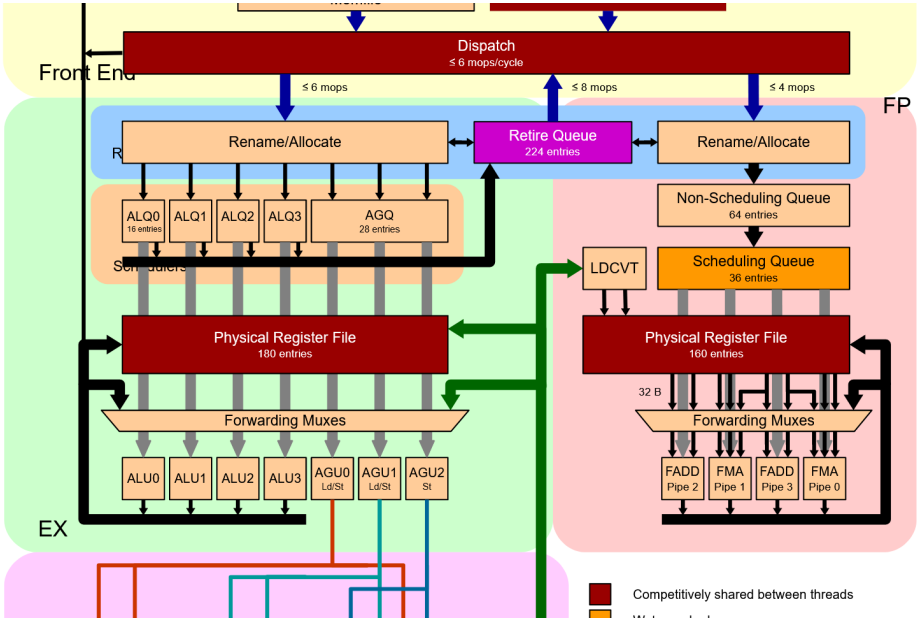



Next Address Logic
L0/L1/L2 ITLB
L1: 64, L2: 512 entries
L1 Hashed Perceptron
L2 TAGE
L1/L2 BTB, Return Stack, ITA
L1 BTB: 512, L2 BTB: 7168, RAS: 32, ITA: 1024 entries
Micro-Tags
Prediction Queue
Instruction Cache
32 KB, 8-way, 64 B/line
32 B/cycle from L2
32 B
Instruction Byte Queue
20 × 16 B
Pick
4-way Decoder
Op Cache
4 K mops, 8-way, 8 mops/line
≤ ? mops (≤ 4 x86 instr)

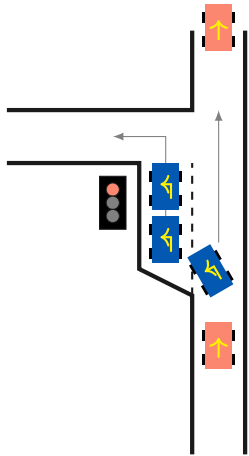

Micro-Tags
Queue
Instruction Cache
32 KB, 8-way, 64 B/line
32 B/cycle
from L2
32 B
Instruction Byte Queue
20 × 16 B
Pick
4-way Decoder
Op Cache
4 K mops, 8-way, 8 mops/line
≤ ? mops (≤ 4 x86 instr)
≤ 8 mops (≤ 8 x86 instr)
OC/IC mode
Micro-OP Queue
Stack Engine
Memfile
Microcode Sequencer
Microcode ROM
Dispatch
≤ 6 mops/cycle
Front End
≤ 6 mops
≤ 8 mops
≤ 4 mops


FP
Rename/Allocate
Retire Queue
224 entries
Rename/Allocate
Non-Scheduling Queue
64 entries
ALQ0
16 entries
ALQ1
ALQ2
ALQ3
AGQ
28 entries
LDCVT
Scheduling Queue
36 entries
Physical Register File
180 entries
Physical Register File
160 entries
32 B
Forwarding Muxes
Forwarding Muxes
ALU0
ALU1
ALU2
ALU3
AGU0
Ld/St
AGU1
Ld/St
AGU2
St
FADD
Pipe 2
FMA
Pipe 1
FADD
Pipe 3
FMA
Pipe 0
EX
Load Queue
44 entries
LD0 Pick
LD1 Pick
Prefetch
Store Queue
48 entries
ST Pick
Competitively shared between threads
Watermarked
Statically partitioned


EX
ALU0
ALU1
ALU2
ALU3
AGU0
Ld/St
AGU1
Ld/St
AGU2
St
FADD
Pipe 2
FMA
Pipe 1
FADD
Pipe 3
FMA
Pipe 0
Competitively shared between threads
Watermarked
Statically partitioned
Load Queue
44 entries
Store Queue
48 entries
Prefetch
LD0 Pick
LD1 Pick
ST Pick
TLB0
DAT0
TLB1
DAT1
STP
L1/L2 DTLB,
DC Tags
Store
Commit
32 B
32 B
32 B
L1 Data Cache
32 KB, 8-way, 64 B/line
MAB
WCB
LS
To/from L2
32 B to/from L2
To L2


Front End
Dispatch
≤ 6 mops/cycle
≤ 6 mops
≤ 8 mops
≤ 4 mops
FP
Rename/Allocate
Retire Queue
224 entries
Rename/Allocate
Non-Scheduling Queue
64 entries
ALQ0
16 entries
ALQ1
ALQ2
ALQ3
AGQ
28 entries
LDCVT
Scheduling Queue
36 entries
Physical Register File
180 entries
Physical Register File
160 entries
32 B
Forwarding Muxes
Forwarding Muxes
ALU0
ALU1
ALU2
ALU3
AGU0
Ld/St
AGU1
Ld/St
AGU2
St
FADD
Pipe 2
FMA
Pipe 1
FADD
Pipe 3
FMA
Pipe 0
EX
Competitively shared between threads

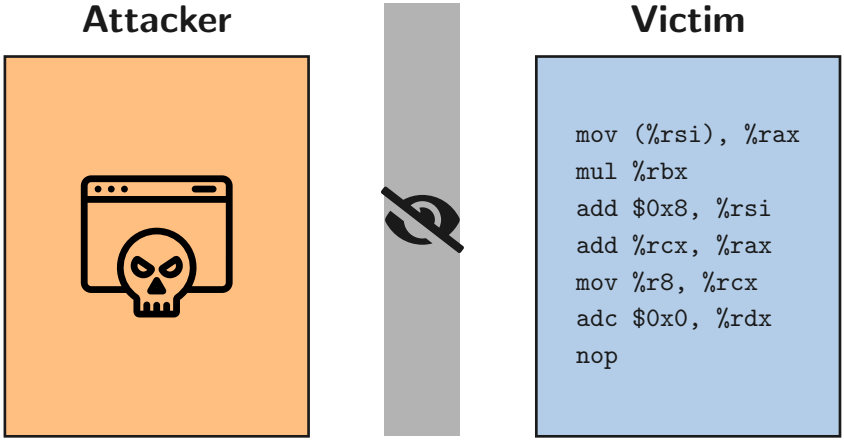
SQUIP:
Scheduler Queue Usage Interference Probing

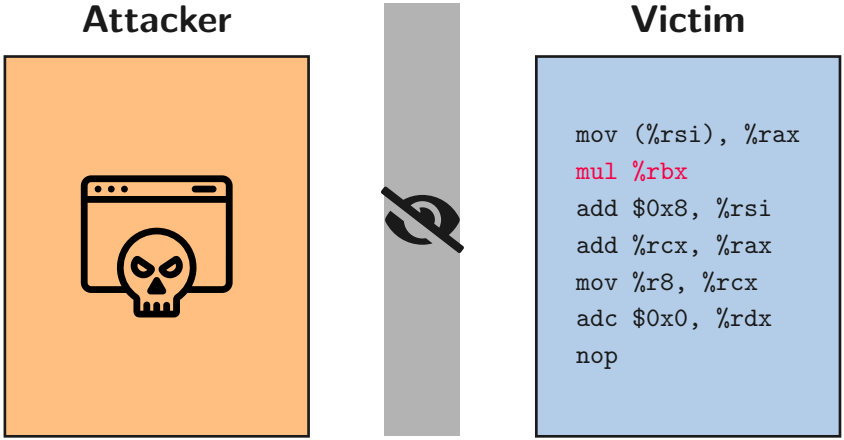


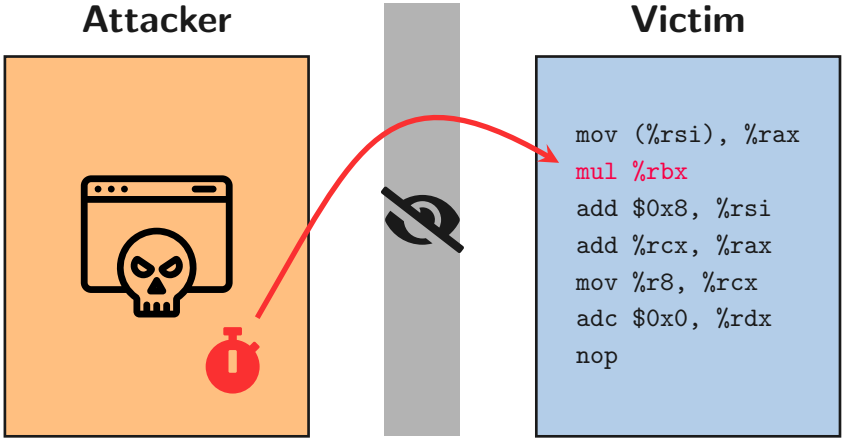
Attacker

Victim

```
mov (%rsi), %rax
mul %rbx
add $0x8, %rsi
add %rcx, %rax
mov %r8, %rcx
adc $0x0, %rdx
nop
```


Attacker
Victim
mov (%rsi), %rax
mul %rbx
add $0x8, %rsi
add %rcx, %rax
mov %r8, %rcx
adc $0x0, %rdx
nop


Attacker
Victim
mov (%rsi), %rax
mul %rbx
add $0x8, %rsi
add %rcx, %rax
mov %r8, %rcx
adc $0x0, %rdx
nop


Attacker
Victim
mov (%rsi), %rax
mul %rbx
add $0x8, %rsi
add %rcx, %rax
mov %r8, %rcx
adc $0x0, %rdx
nop

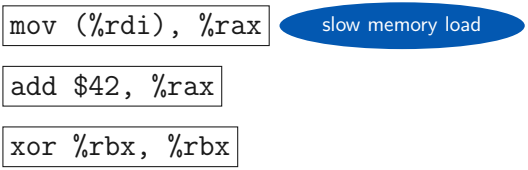
Out-of-Order Execution

```
mov (%rdi), %rax
```

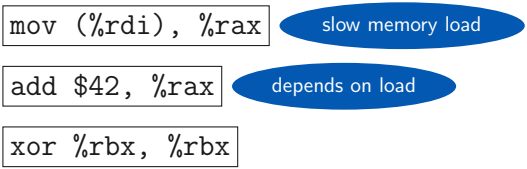
```
add $42, %rax
```

```
xor %rbx, %rbx
```

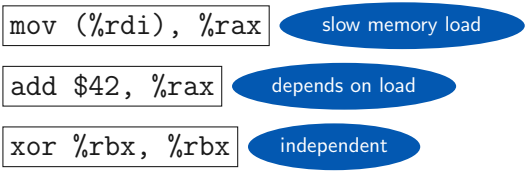
Out-of-Order Execution



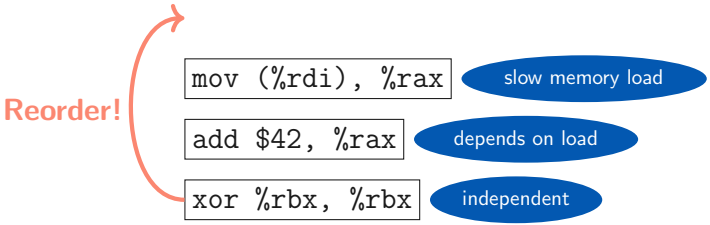
Out-of-Order Execution



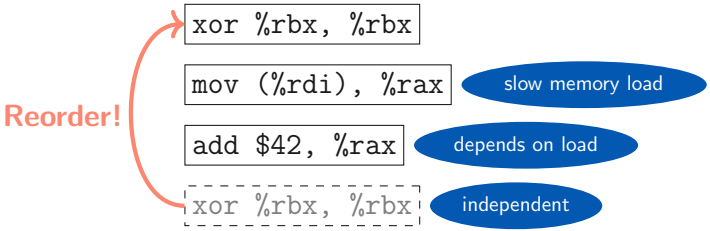
Out-of-Order Execution



Out-of-Order Execution



Out-of-Order Execution



How to Read the Clock

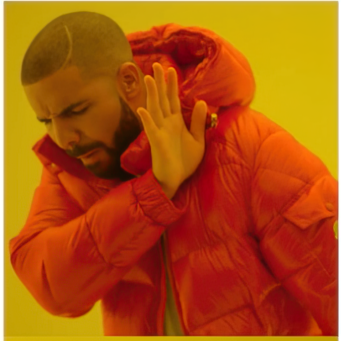
- `rdtsc`: Read time-stamp counter

How to Read the Clock

- `rdtsc`: Read time-stamp counter
- `rdpru`: Read `APERF` counter (more precise on AMD)

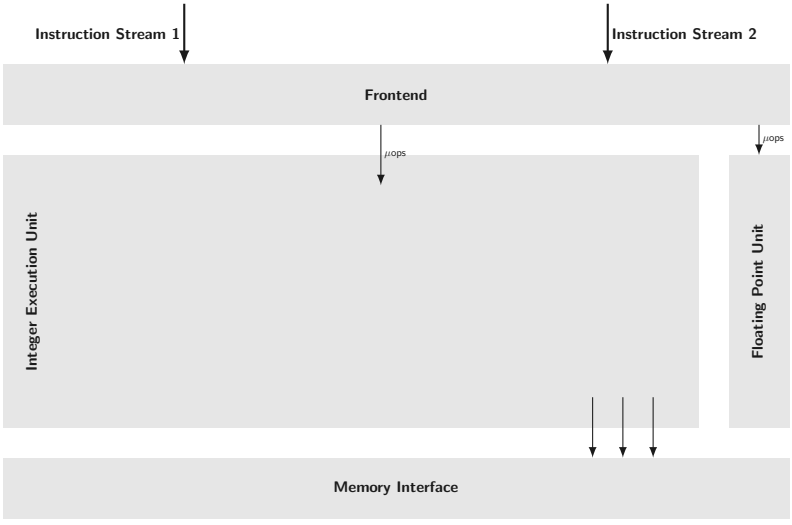
How to Read the Clock

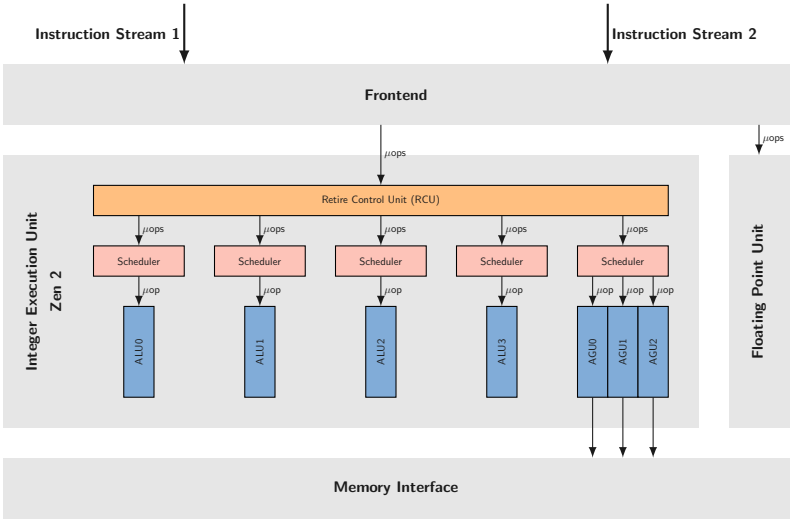
- `rdtsc`: Read time-stamp counter
- `rdpru`: Read `APERF` counter (more precise on AMD)
- Requires explicit serialization to prevent out-of-order execution

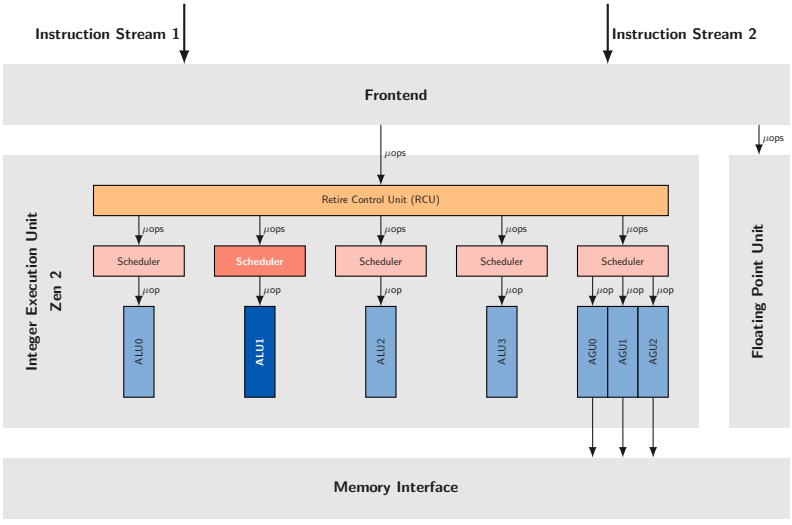


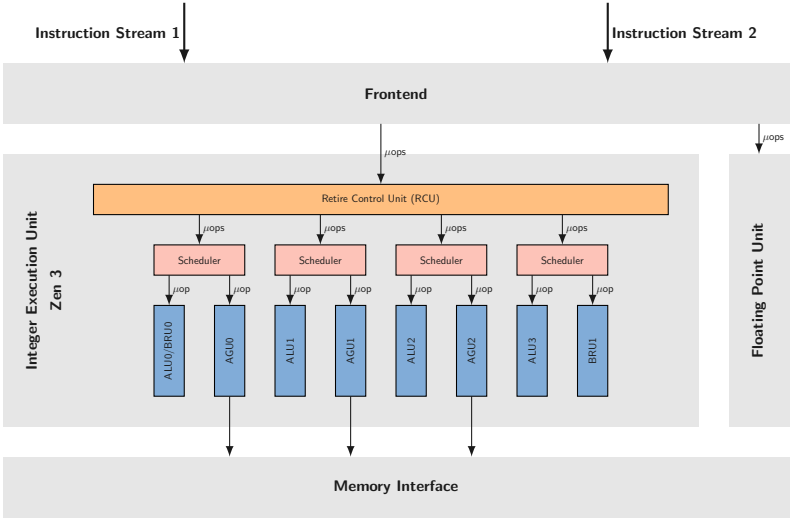
Adding serializing instructions around rdtsc

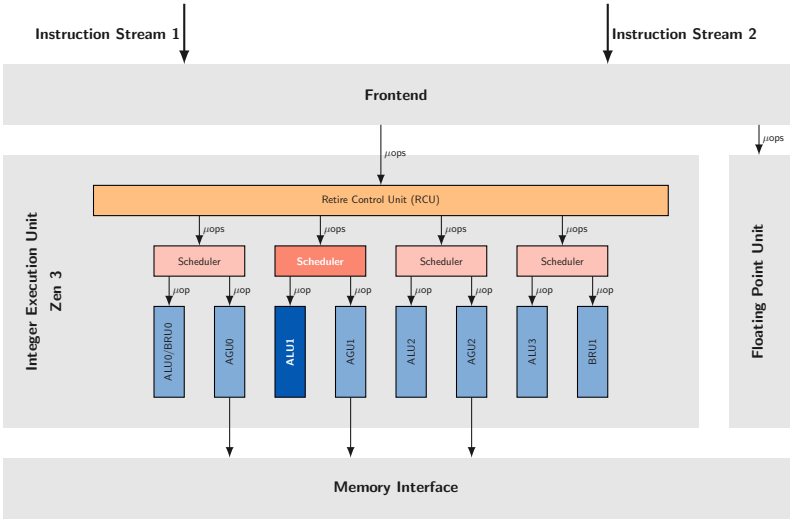
Exploiting out-of-order execution of rdtsc


Instruction Stream 1
Instruction Stream 2
Frontend
µops
µops
Integer Execution Unit
Floating Point Unit
Memory Interface

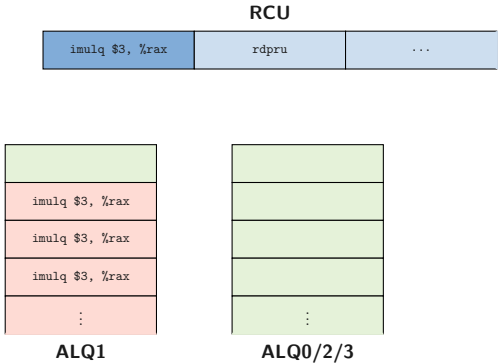

Instruction Stream 1
Instruction Stream 2
Frontend
µops
µops
Retire Control Unit (RCU)
Integer Execution Unit
Zen 2
µops
µops
µops
µops
µops
Scheduler
Scheduler
Scheduler
Scheduler
Scheduler
µop
µop
µop
µop
µop
µop
µop
ALU0
ALU1
ALU2
ALU3
AGU0
AGU1
AGU2
Floating Point Unit
Memory Interface


Instruction Stream 1
Instruction Stream 2
Frontend
µops
µops
Retire Control Unit (RCU)
Integer Execution Unit
Zen 2
µops
µops
µops
µops
µops
Scheduler
Scheduler
Scheduler
Scheduler
Scheduler
µop
µop
µop
µop
µop
µop
µop
ALU0
ALU1
ALU2
ALU3
AGU0
AGU1
AGU2
Floating Point Unit
Memory Interface

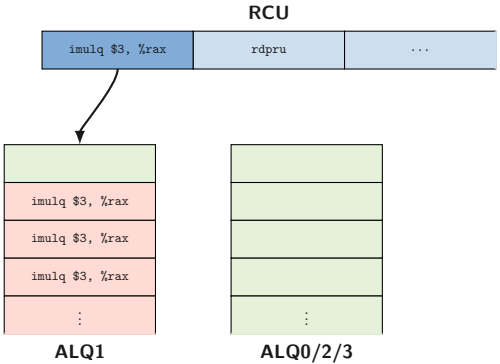

Instruction Stream 1
Instruction Stream 2
Frontend
µops
µops
Retire Control Unit (RCU)
Integer Execution Unit
Zen 3
µops
µops
µops
µops
Scheduler
Scheduler
Scheduler
Scheduler
µop
µop
µop
µop
µop
µop
µop
µop
ALU0/BRU0
AGU0
ALU1
AGU1
ALU2
AGU2
ALU3
BRU1
Floating Point Unit
Memory Interface


Instruction Stream 1
Instruction Stream 2
Frontend
µops
µops
Retire Control Unit (RCU)
Integer Execution Unit
Zen 3
µops
µops
µops
µops
Scheduler
Scheduler
Scheduler
Scheduler
µop
µop
µop
µop
µop
µop
µop
µop
ALU0/BRU0
AGU0
ALU1
AGU1
ALU2
AGU2
ALU3
BRU1
Floating Point Unit
Memory Interface

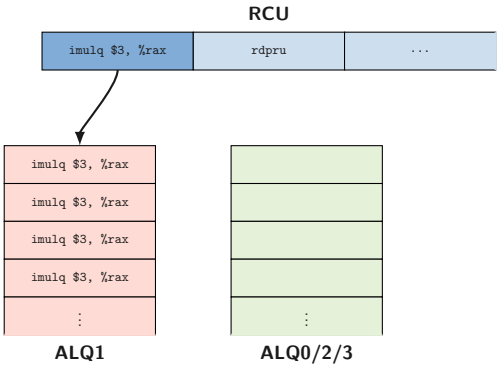
What if a scheduler queue is full?



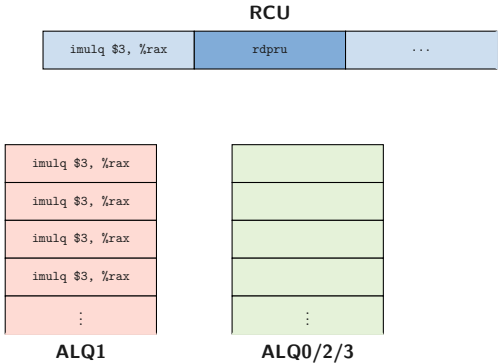
What if a scheduler queue is full?



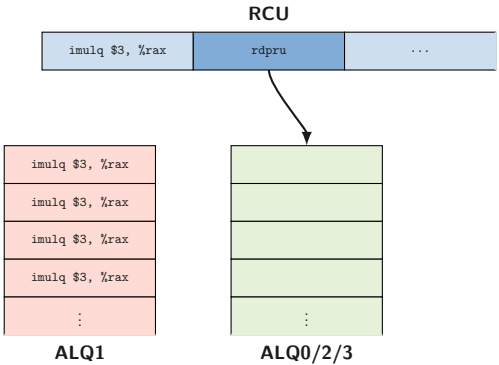
What if a scheduler queue is full?



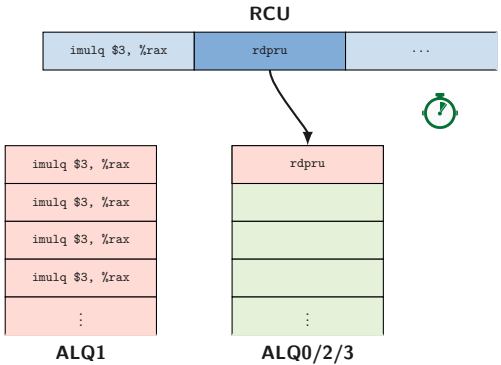
What if a scheduler queue is full?



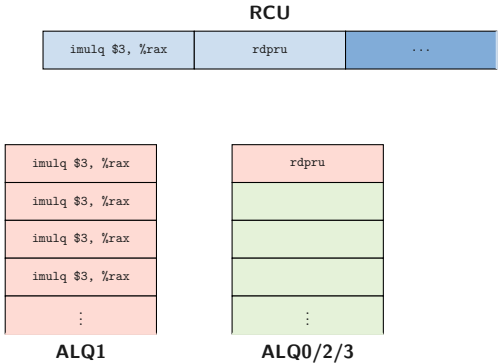
What if a scheduler queue is full?



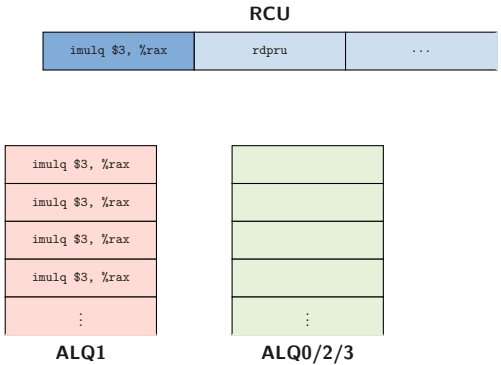
What if a scheduler queue is full?



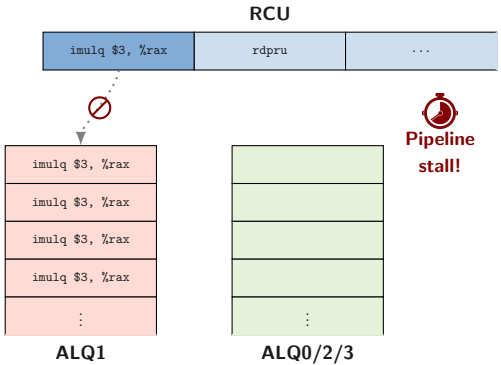
What if a scheduler queue is full?



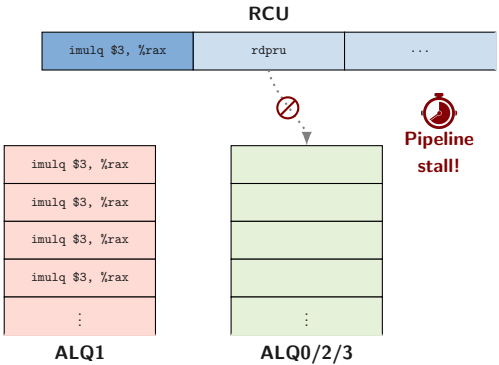
What if a scheduler queue is full?



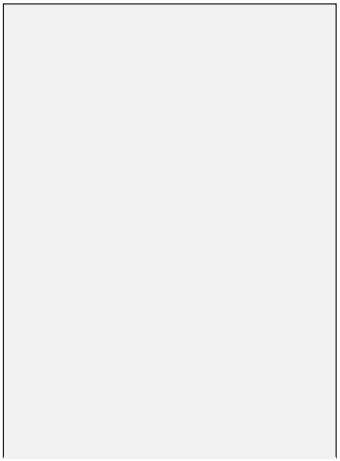
What if a scheduler queue is full?

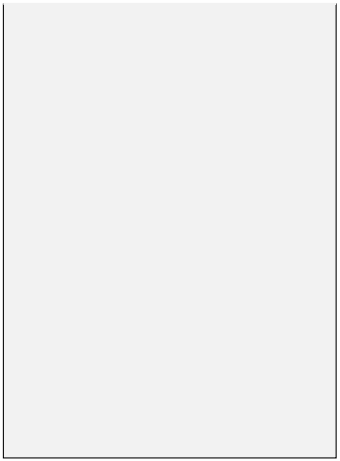


What if a scheduler queue is full?

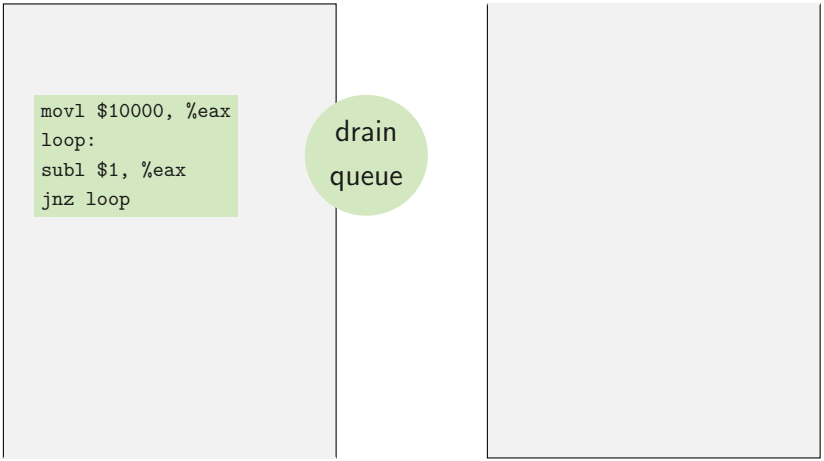


Let's code that!

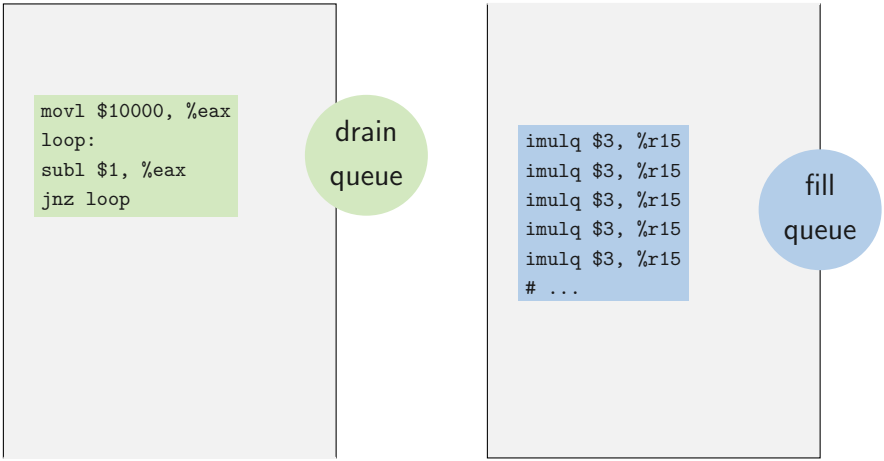




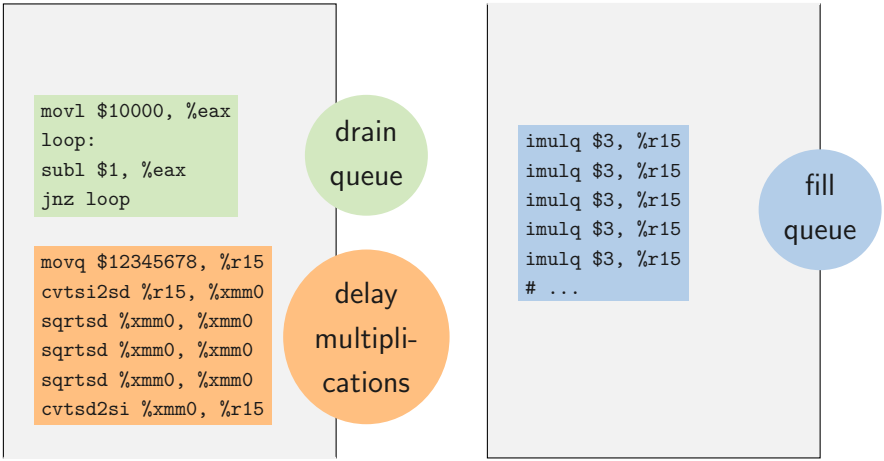
Let's code that!



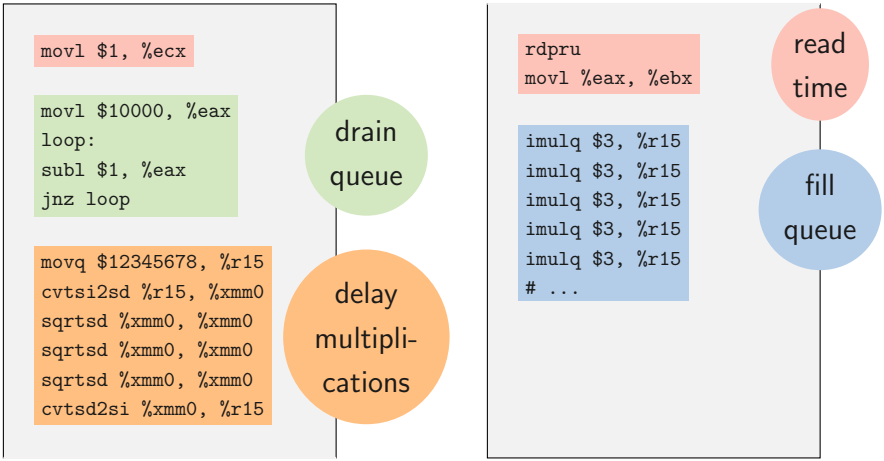
Let's code that!



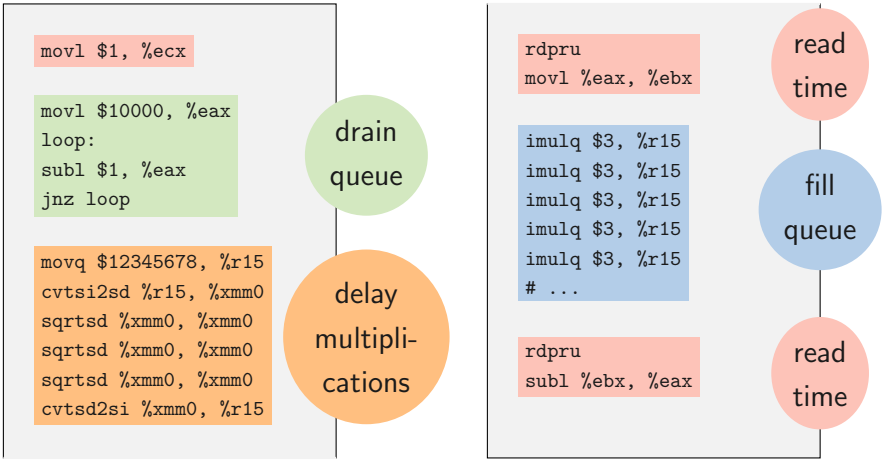
Let's code that!

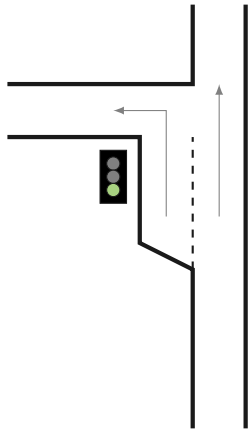


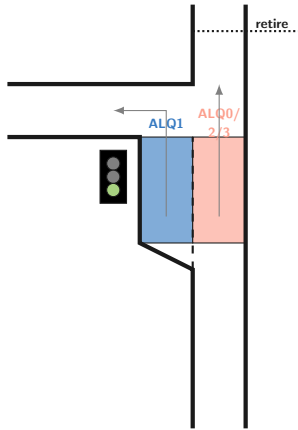
Let's code that!

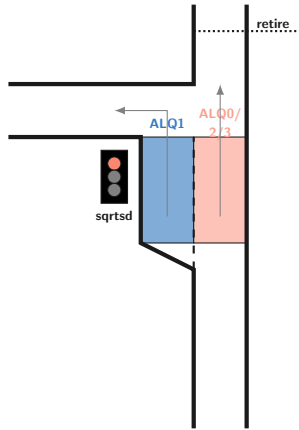


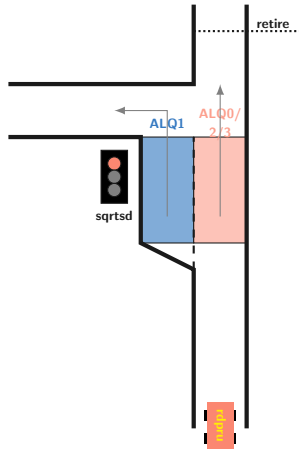
Let's code that!

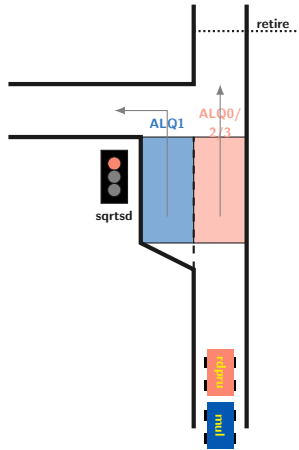


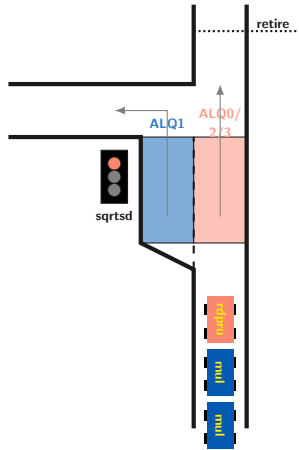


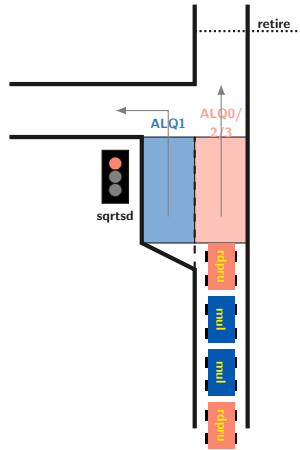

retire
ALQ1
ALQ0/
2/3

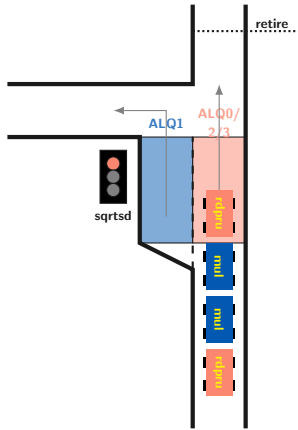

retire
ALQ1
ALQ0/
2/3
sqrtsd

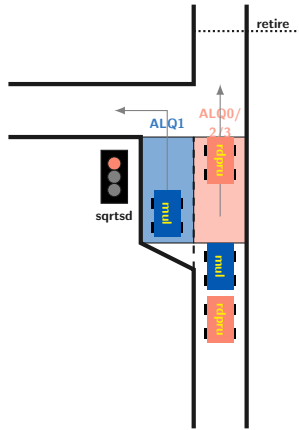

retire
ALQ1
ALQ0/
2/3
sqrtsd
rdpn

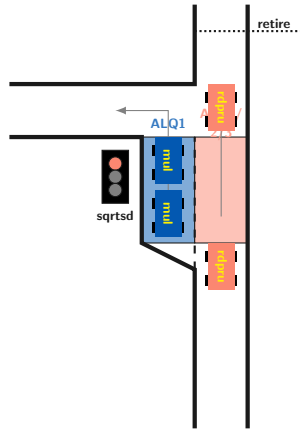

retire
ALQ1
ALQ0/
2/3
sqrtsd
rolpn
rnnl

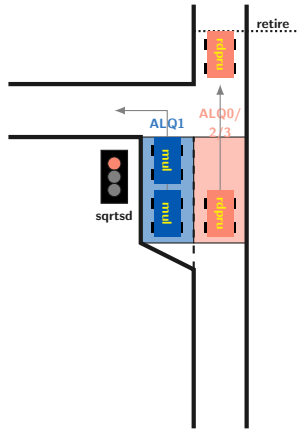

retire
ALQ1
ALQ0/
2/3
sqrtsd
rdpn
mul
mul

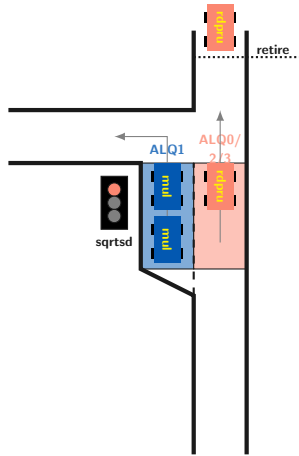

retire
ALQ1
ALQ0/
2/3
sqrtsd
rdpn
mul
mul
rdpn

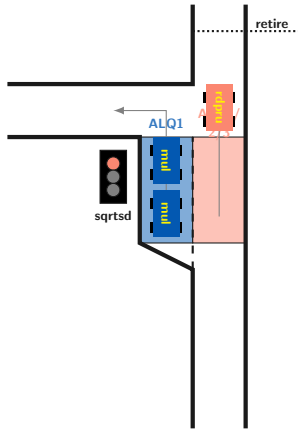

retire
ALQ1
ALQ0/
2/3
sqrtsd
mul
mul
mul

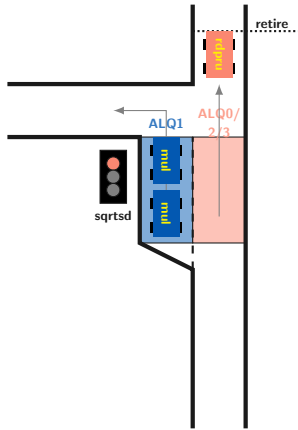

retire
ALQ1
ALQ0/
2/3
sqrtsd
mul
mul
rdpn
rdpn

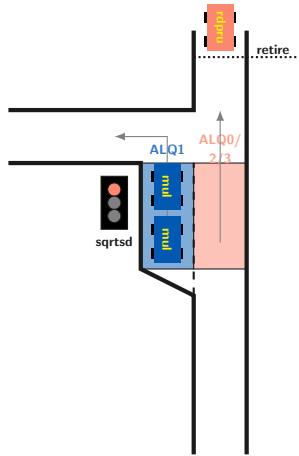

retire
ALQ1
nulpn
nul
nul
sqrtsd
nulpn

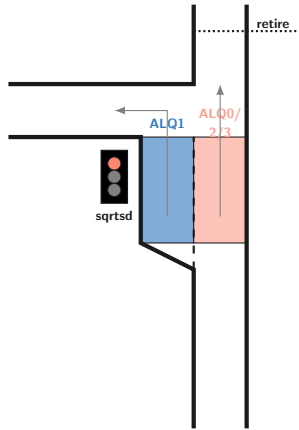

retire
rdpm
ALQ1
ALQ0/
2/3
mul
mul
rdpm
sqrtsd

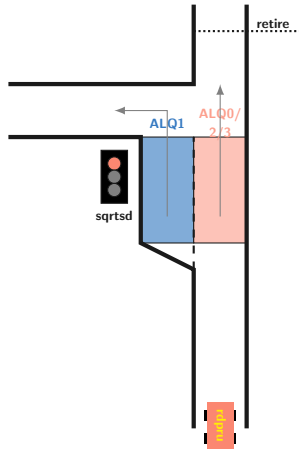

dqtp
retire
ALQ1
ALQ0/
2/3
mul
mul
rqtp
sqrtsd

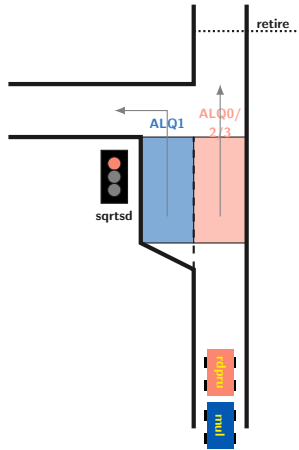

retire
ALQ1
rnbpn
rnuf
rnuf
sqrtsd

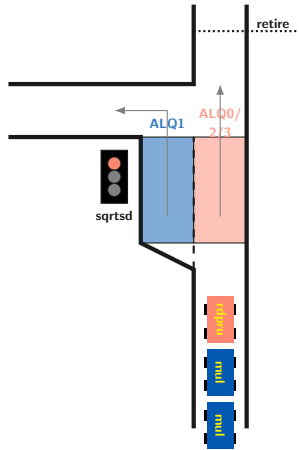

retire
mpn
ALQ1
ALQ0/
2/3
mul
mul
sqrtsd

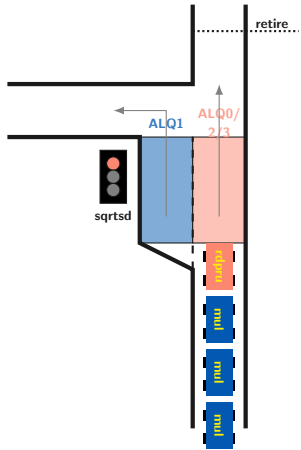

dpnp
retire
ALQ1
ALQ0/
2/3
mul
mul
sqrtsd

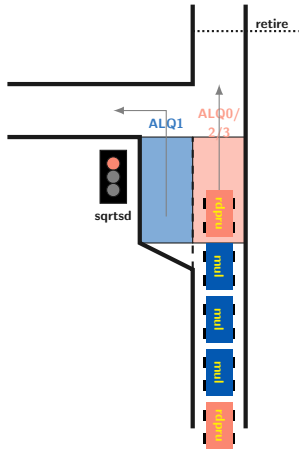

retire
ALQ1
ALQ0/
2/3
sqrtsd

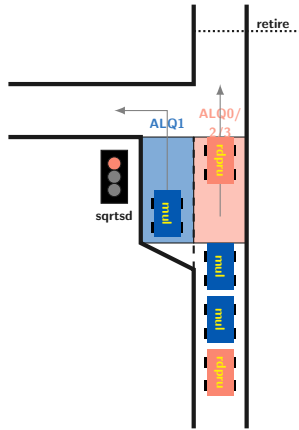

retire
ALQ1
ALQ0/
2/3
sqrtsd
rdpn

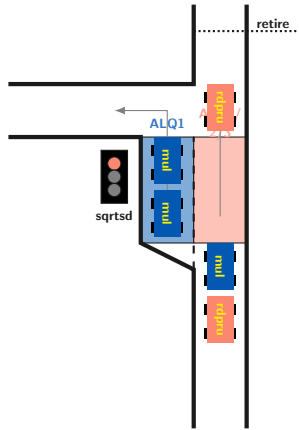

retire
ALQ1
ALQ0/
2/3
sqrtsd
rolpn
rnnl

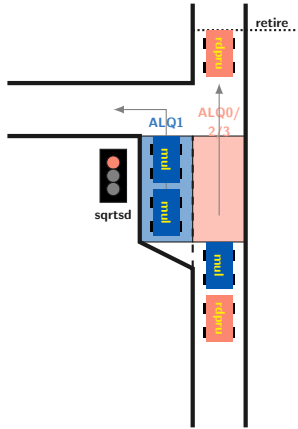

retire
ALQ1
ALQ0/
2/3
sqrtsd
rdpn
mul
mul

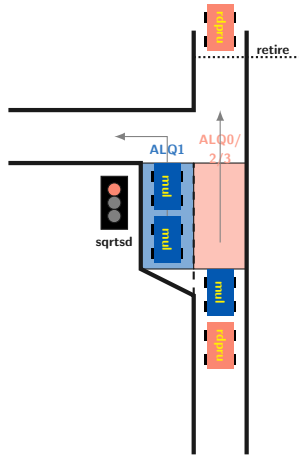

retire
ALQ1
ALQ0/
2/3
sqrtsd
subps
mul
mul
mul

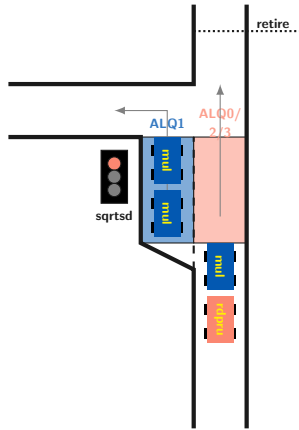

retire
ALQ1
ALQ0/
2/3
sqrtsd
rdpn
mul
mul
mul
rdpn


retire
ALQ1
ALQ0/
2/3
sqrtsd

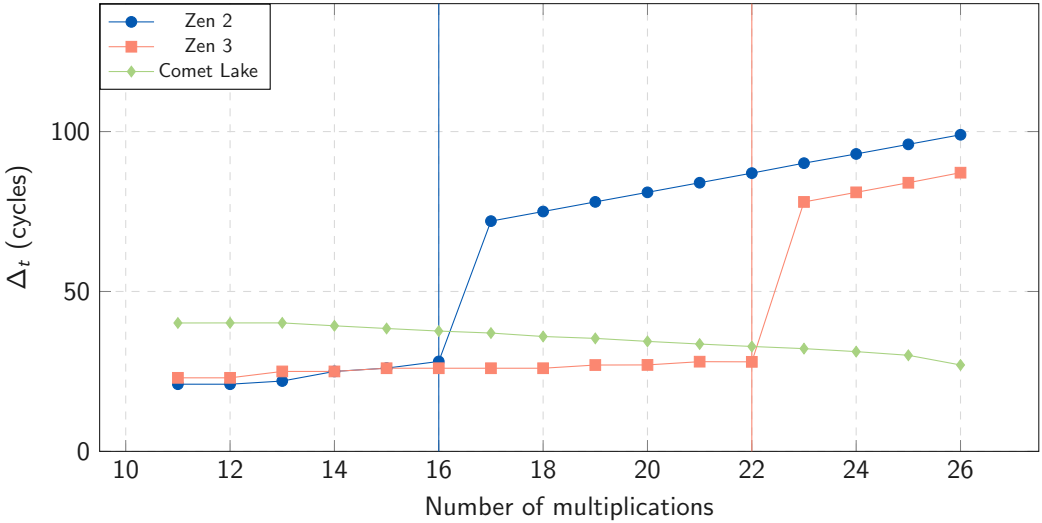

retire
ALQ1
sqrtsd


retire
mulp
ALQ1
ALQ0/
2/3
mul
mul
sqrtsd
mul
mulp


mulp
retire
ALQ1
ALQ0/
2/3
mul
mul
sqrtsd
mul
mulp


retire
ALQ1
ALQ0/
2/3
mul
mul
sqrtsd
mul
addps

Timing Differences on Zen 2, Zen 3 and Comet Lake



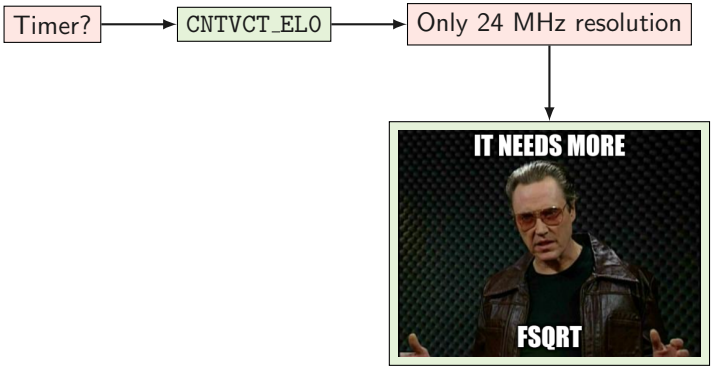
Adaption for the Apple M1

Timer?

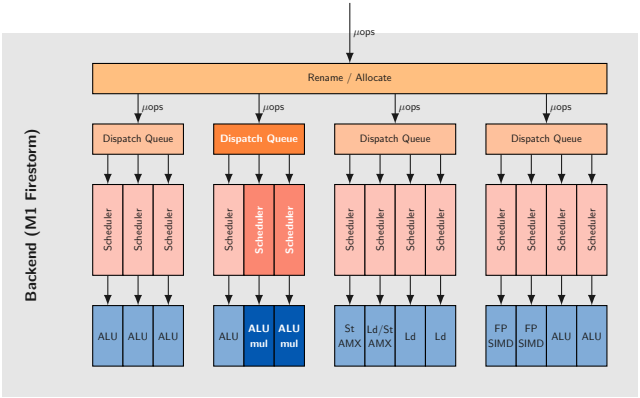
Adaption for the Apple M1

Adaption for the Apple M1

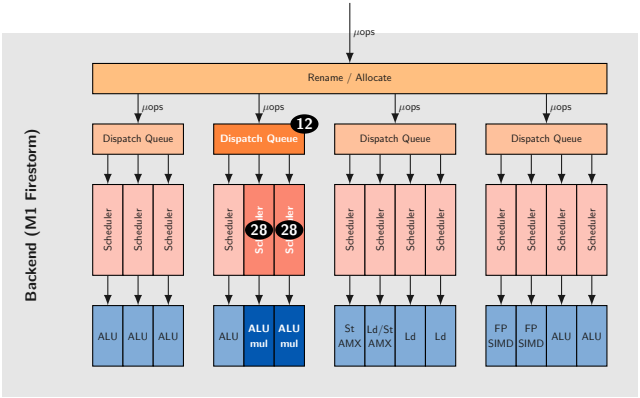
Adaption for the Apple M1



Scheduler Design of the Apple M1



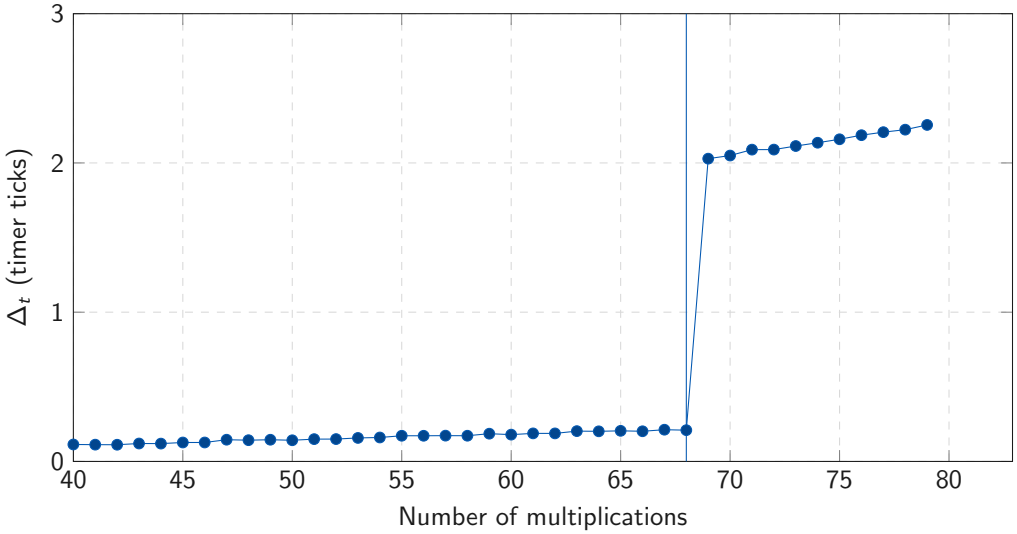
Scheduler Design of the Apple M1



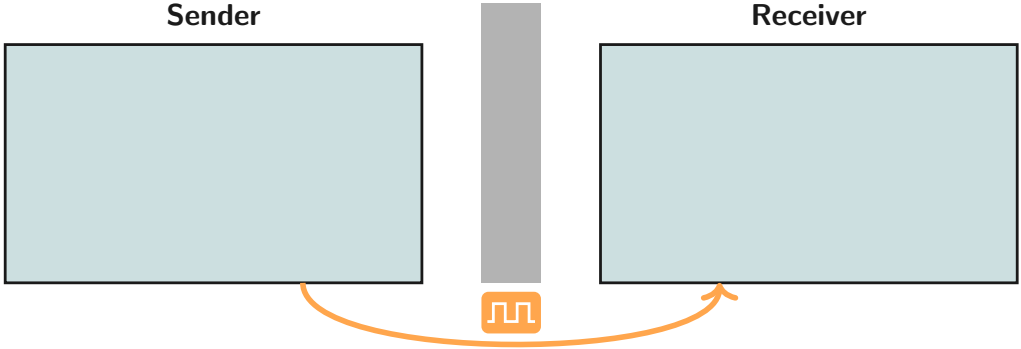
Expected capacity: $28 + 28 + 12 = 68$ entries

Does it work?

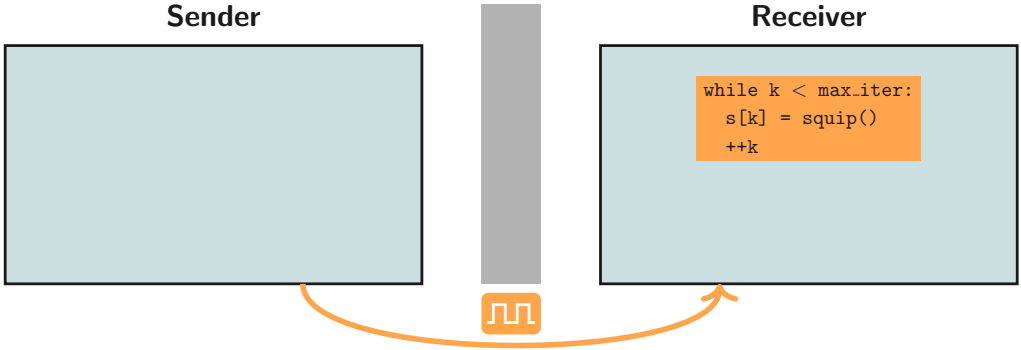
Does it work?


3
2
1
0
Δ_t (timer ticks)
40
45
50
55
60
65
70
75
80
Number of multiplications

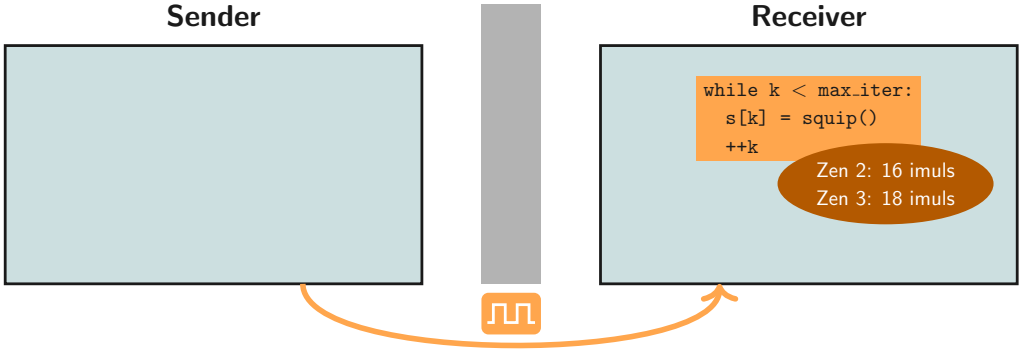
Observing Multiplications of the Sibling Thread



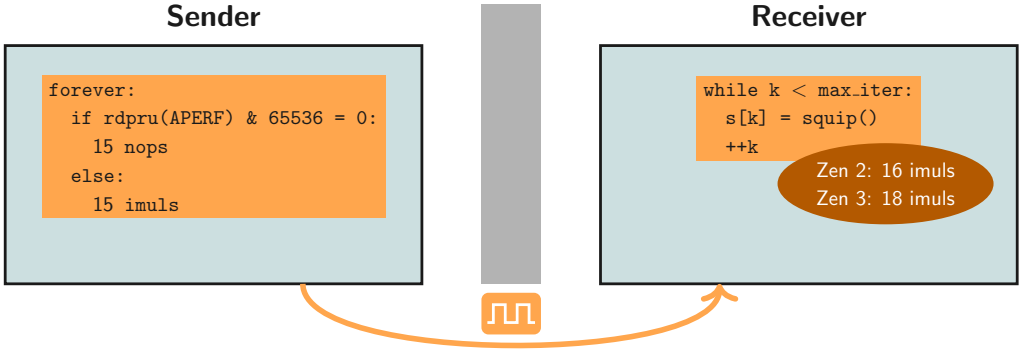
Observing Multiplications of the Sibling Thread



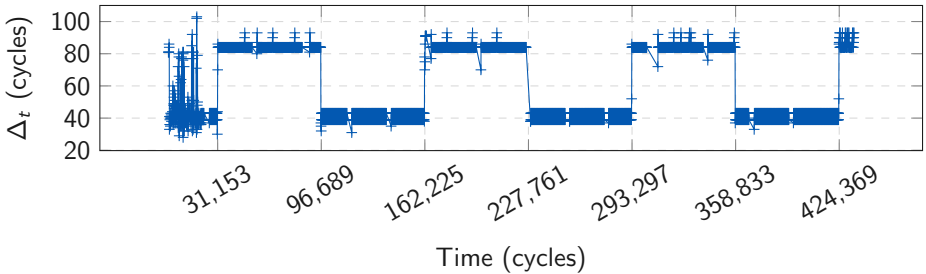
Observing Multiplications of the Sibling Thread



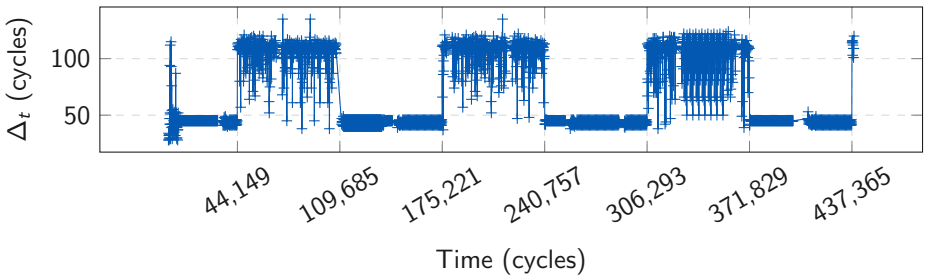
Observing Multiplications of the Sibling Thread



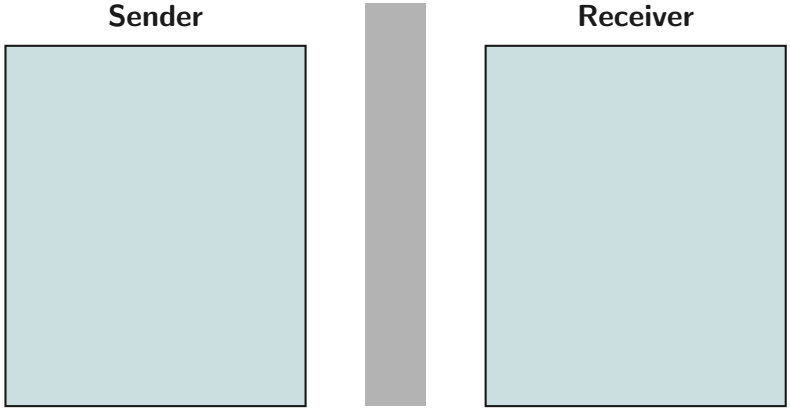
Observing Multiplications of the Sibling Thread (Zen 2)



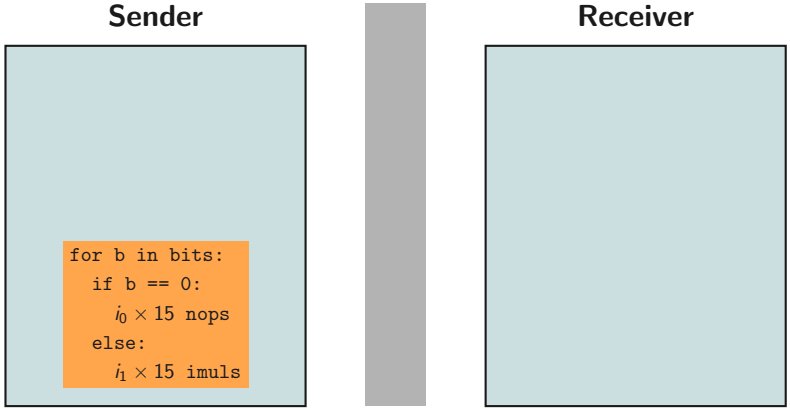
Observing Multiplications of the Sibling Thread (Zen 3)



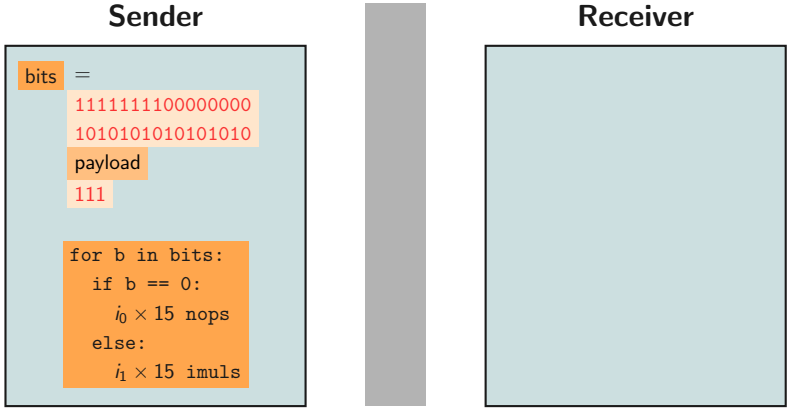
Covert Channel



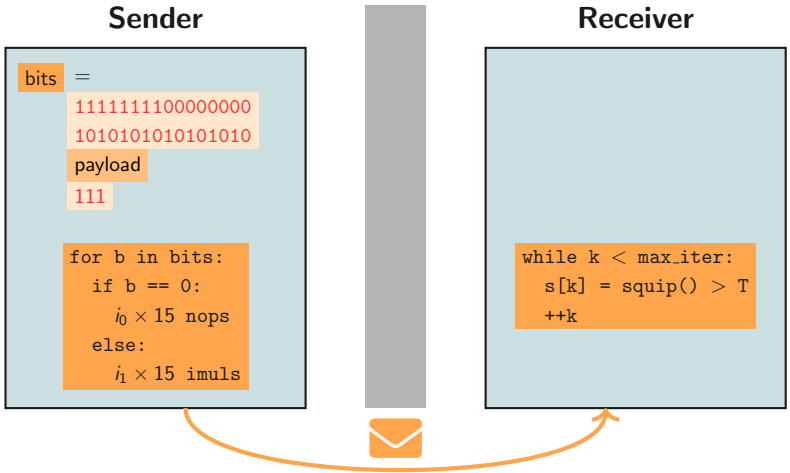
Covert Channel



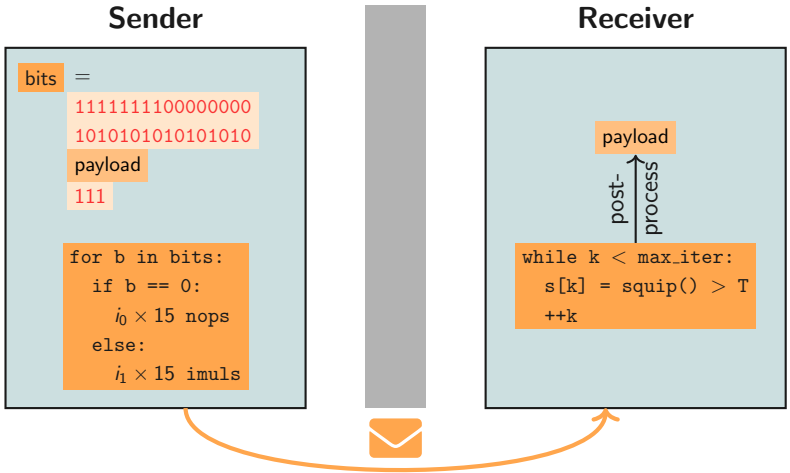
Covert Channel



Covert Channel



Covert Channel



Results

Scenario	CPU	Raw Tx Rate	Error Rate

10 000 random messages, each with 32 kbit payload

Results

Scenario	CPU	Raw Tx Rate	Error Rate
Cross-Process	Ryzen 7 3700X (Zen 2)	$2.195\,\mathrm{Mbit\,s^{-1}}$	0.71 %
	Ryzen 7 5800X (Zen 3)	$2.700\,\mathrm{Mbit\,s^{-1}}$	0.62 %

10 000 random messages, each with 32 kbit payload

Results

Scenario	CPU	Raw Tx Rate	Error Rate
Cross-Process	Ryzen 7 3700X (Zen 2)	$2.195\,\mathrm{Mbit\,s^{-1}}$	0.71 %
	Ryzen 7 5800X (Zen 3)	$2.700\,\mathrm{Mbit\,s^{-1}}$	0.62 %
Cross-VM	Ryzen 7 3700X (Zen 2)	$0.873\,\mathrm{Mbit\,s^{-1}}$	3.18 %
	Ryzen 7 5800X (Zen 3)	$0.892\,\mathrm{Mbit\,s^{-1}}$	0.75 %
	EPYC 7443 (Zen 3)	$0.874\,\mathrm{Mbit\,s^{-1}}$	0.96 %

10 000 random messages, each with 32 kbit payload

Results

Scenario	CPU	Raw Tx Rate	Error Rate
Cross-Process	Ryzen 7 3700X (Zen 2)	$2.195\,\mathrm{Mbit\,s^{-1}}$	0.71 %
	Ryzen 7 5800X (Zen 3)	$2.700\,\mathrm{Mbit\,s^{-1}}$	0.62 %
Cross-VM	Ryzen 7 3700X (Zen 2)	$0.873\,\mathrm{Mbit\,s^{-1}}$	3.18 %
	Ryzen 7 5800X (Zen 3)	$0.892\,\mathrm{Mbit\,s^{-1}}$	0.75 %
	EPYC 7443 (Zen 3)	$0.874\,\mathrm{Mbit\,s^{-1}}$	0.96 %
Cross-VM (SEV)	EPYC 7443 (Zen 3)	$0.873\,\mathrm{Mbit\,s^{-1}}$	1.47 %

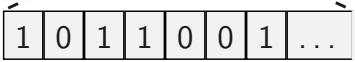
10 000 random messages, each with 32 kbit payload

Attacking RSA (Square+Multiply)

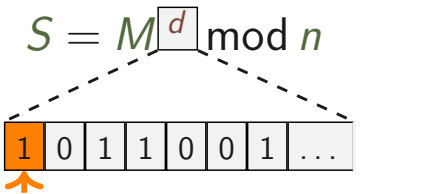
$$S = M^{\boxed{d}} \bmod n$$

Attacking RSA (Square+Multiply)

$$S = M^{d} \bmod n$$

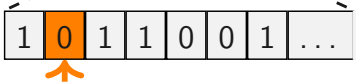


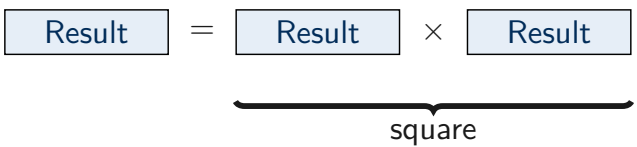
Attacking RSA (Square+Multiply)



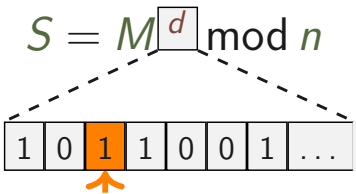
Attacking RSA (Square+Multiply)

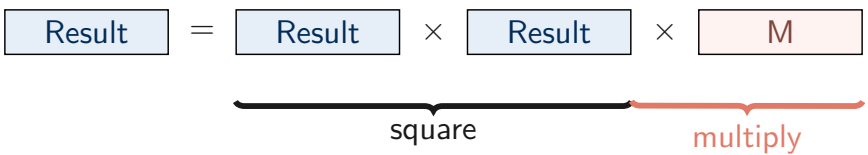
$$S = M^{d} \bmod n$$





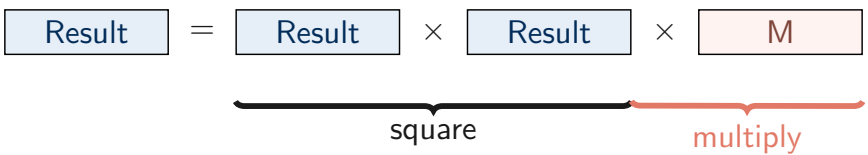
Attacking RSA (Square+Multiply)



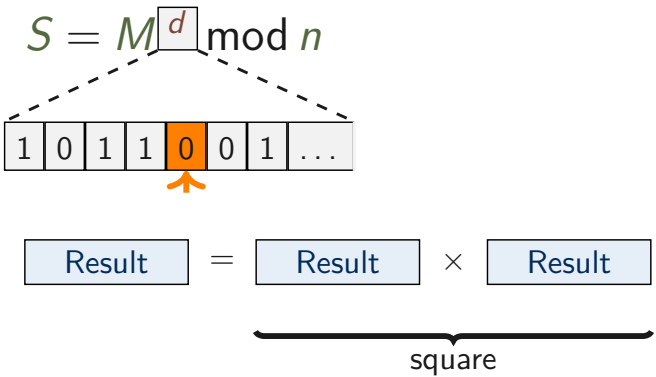


Attacking RSA (Square+Multiply)

$$S = M^{d} \bmod n$$

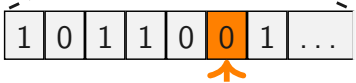


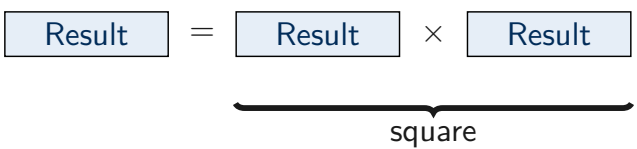
Attacking RSA (Square+Multiply)



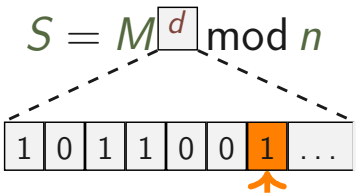
Attacking RSA (Square+Multiply)

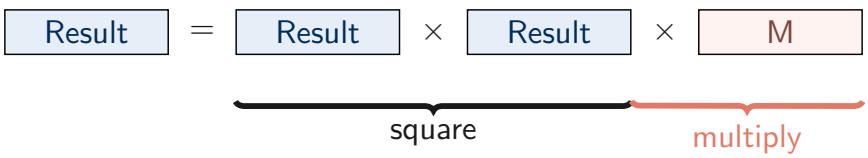
$$S = M^{d} \bmod n$$



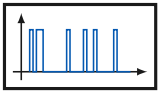


Attacking RSA (Square+Multiply)

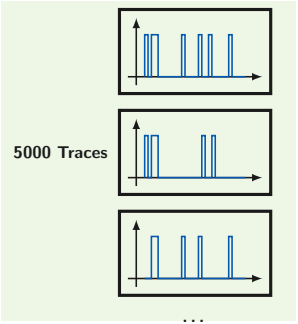




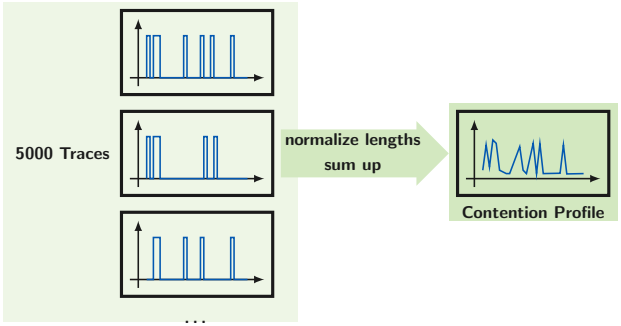
Contention Profiles



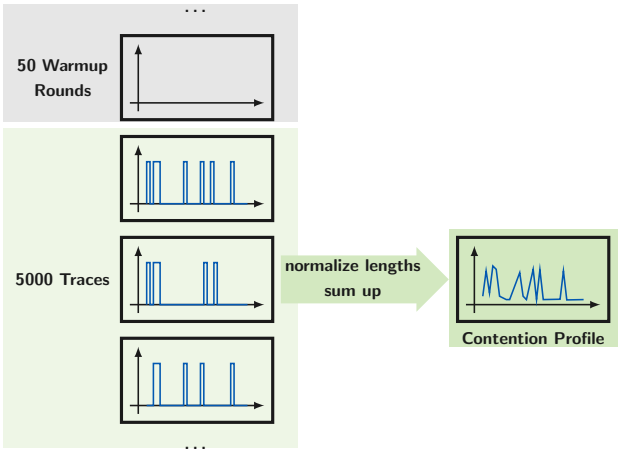
Contention Profiles



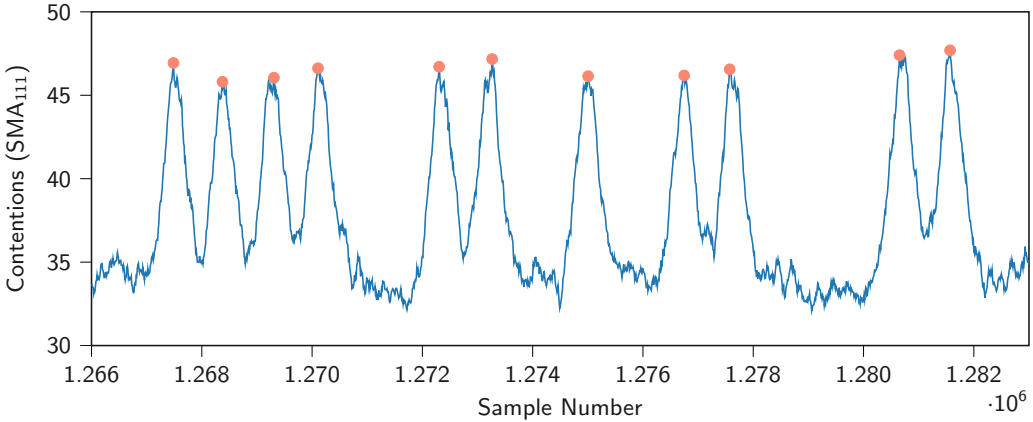
Contention Profiles



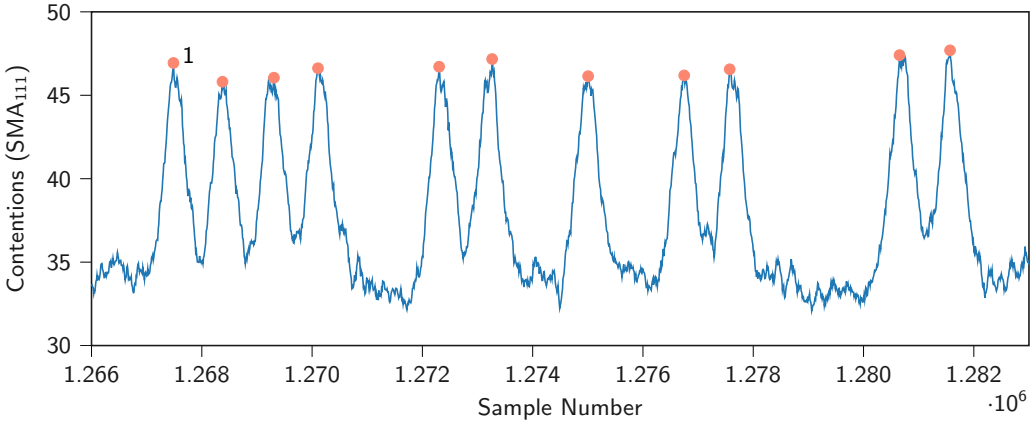
Contention Profiles



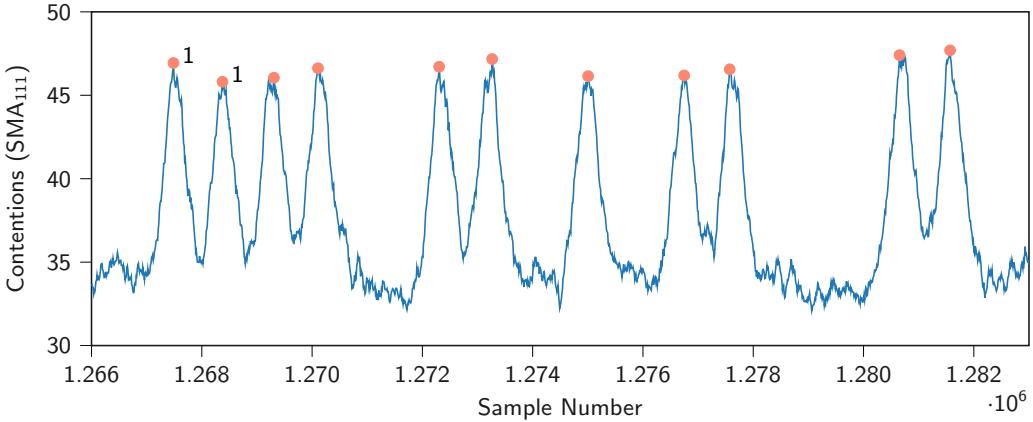
Recovering the Private Key



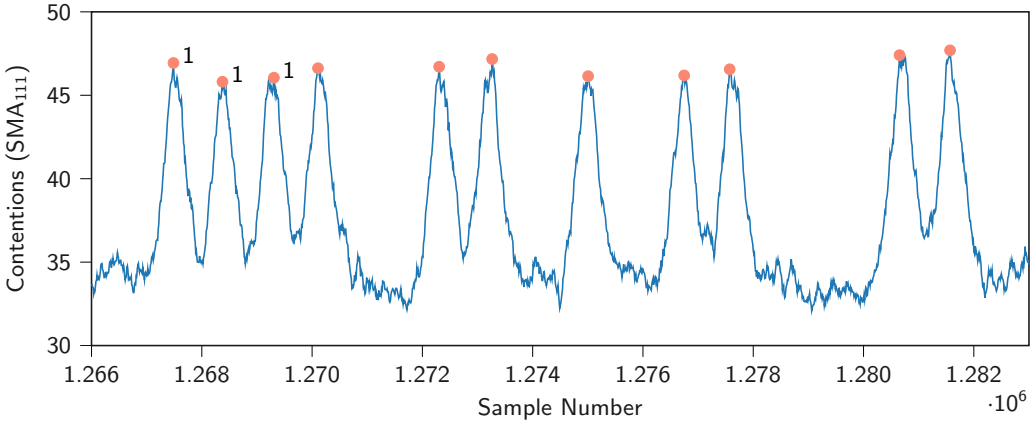
Recovering the Private Key



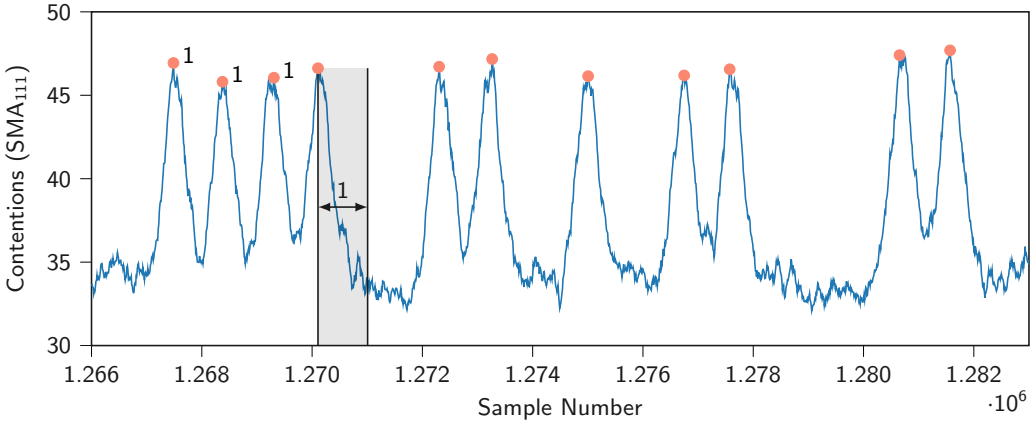
Recovering the Private Key



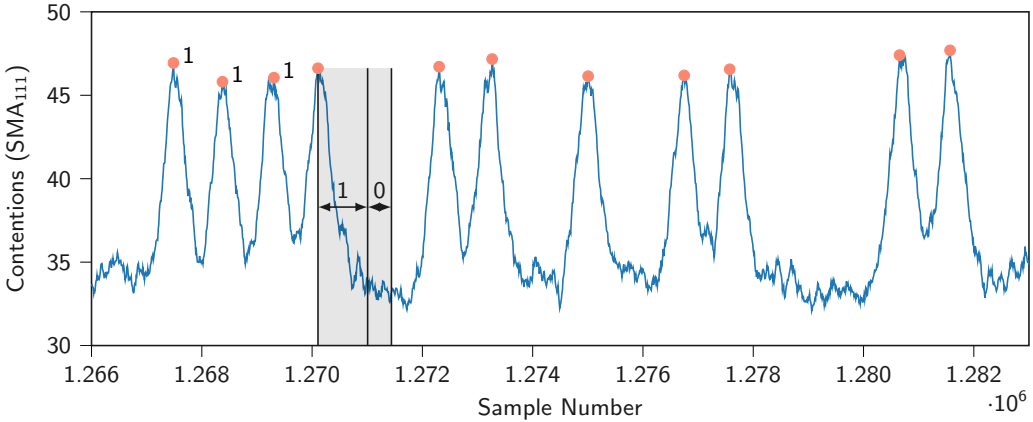
Recovering the Private Key



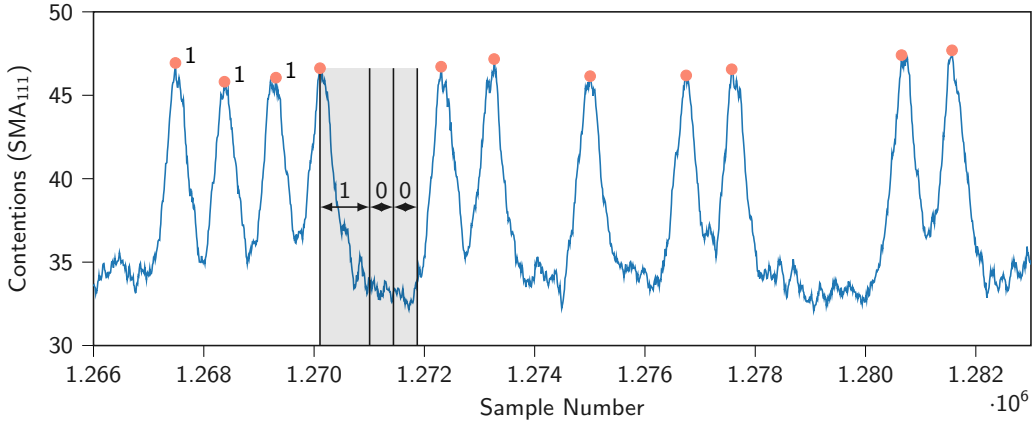
Recovering the Private Key



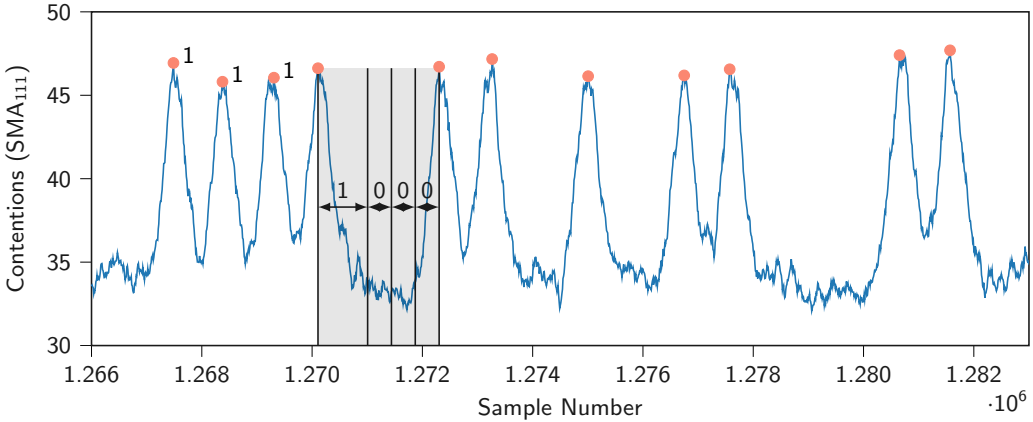
Recovering the Private Key



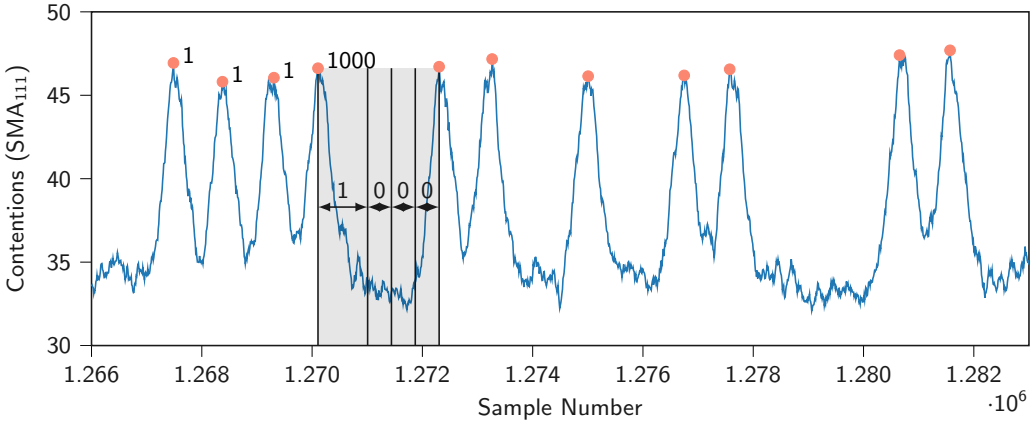
Recovering the Private Key



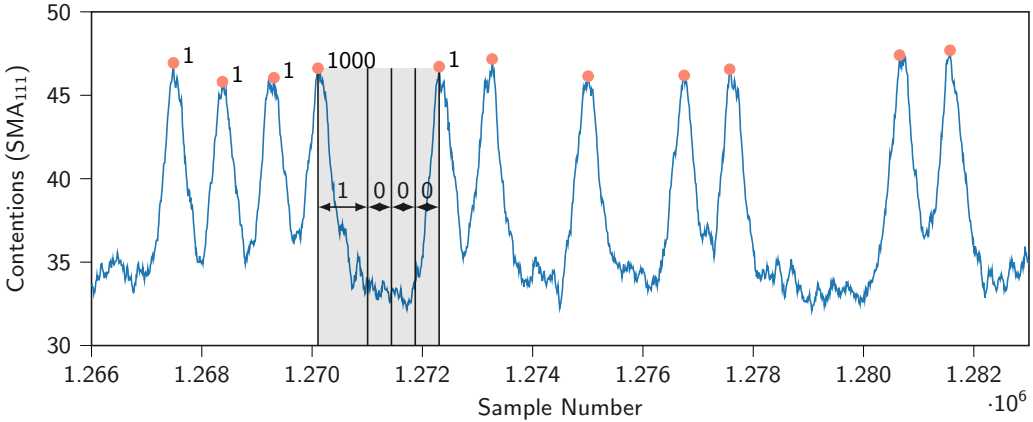
Recovering the Private Key



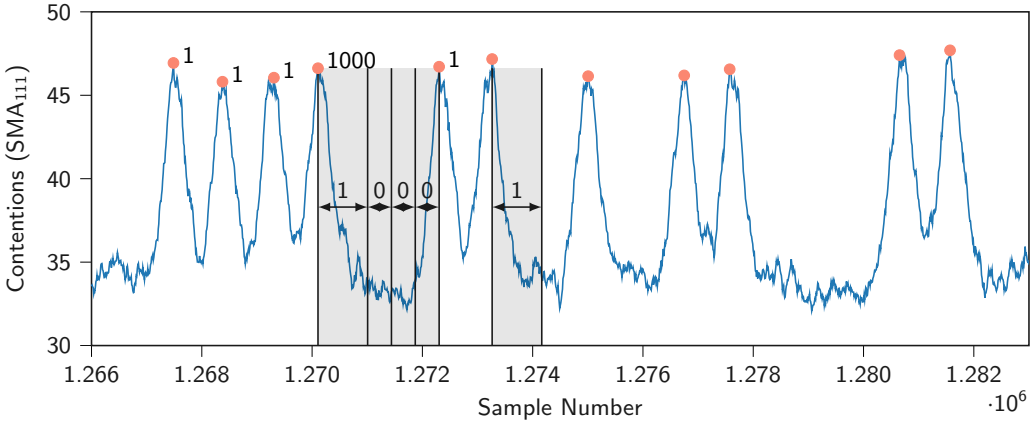
Recovering the Private Key



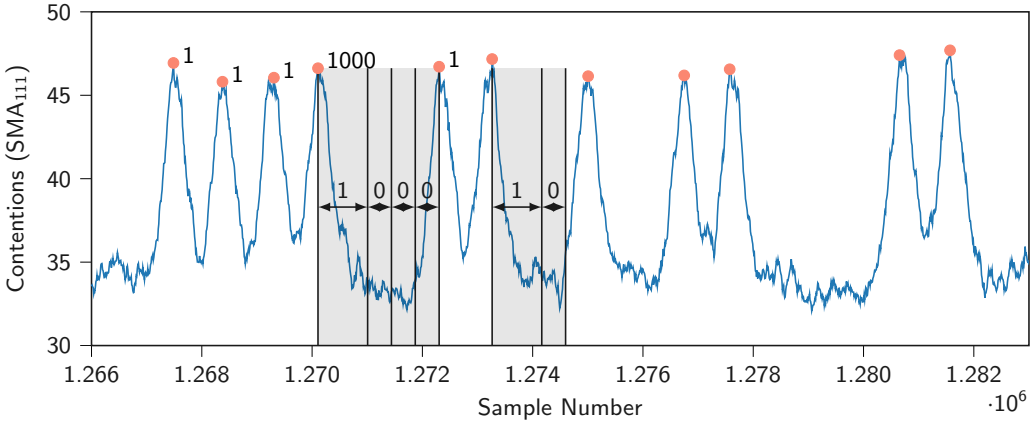
Recovering the Private Key



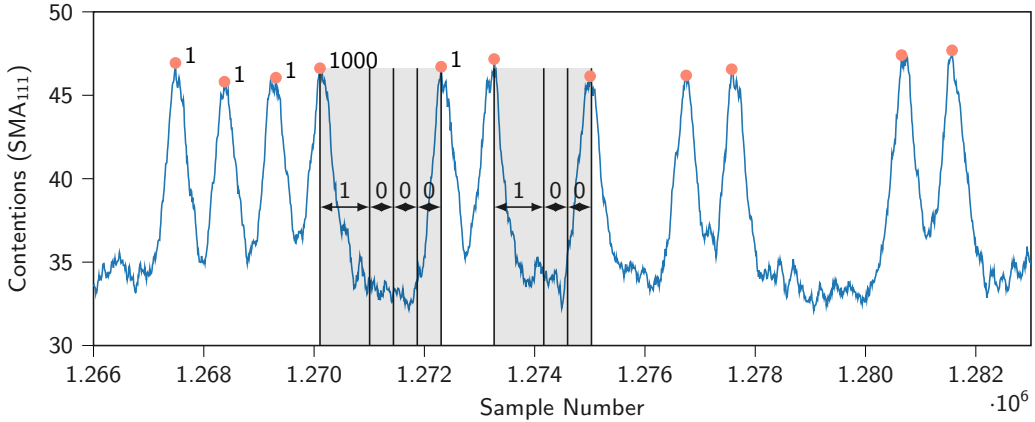
Recovering the Private Key



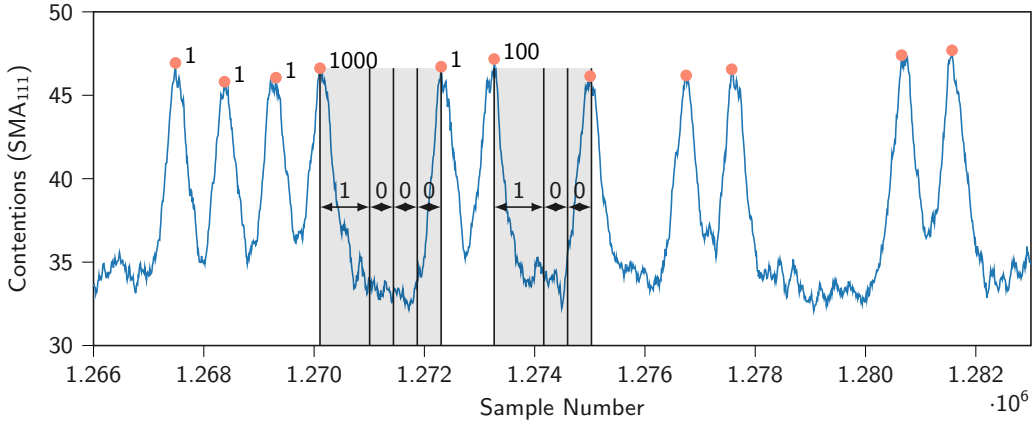
Recovering the Private Key



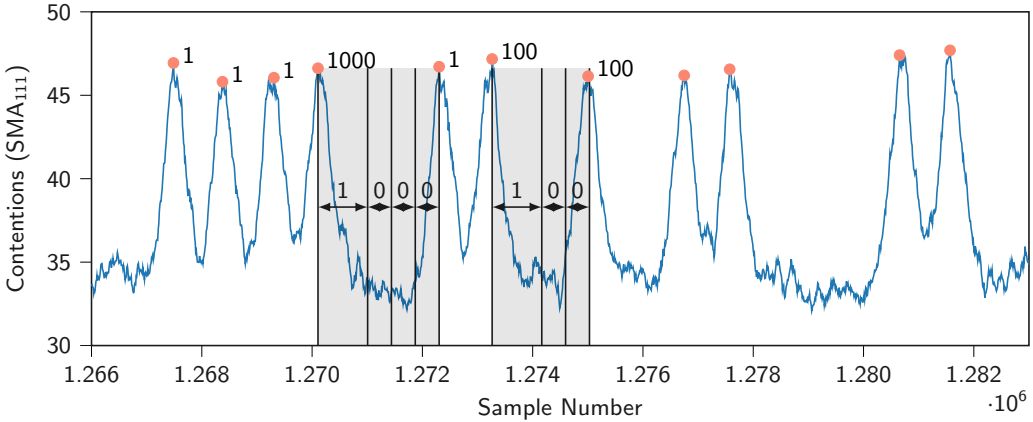
Recovering the Private Key



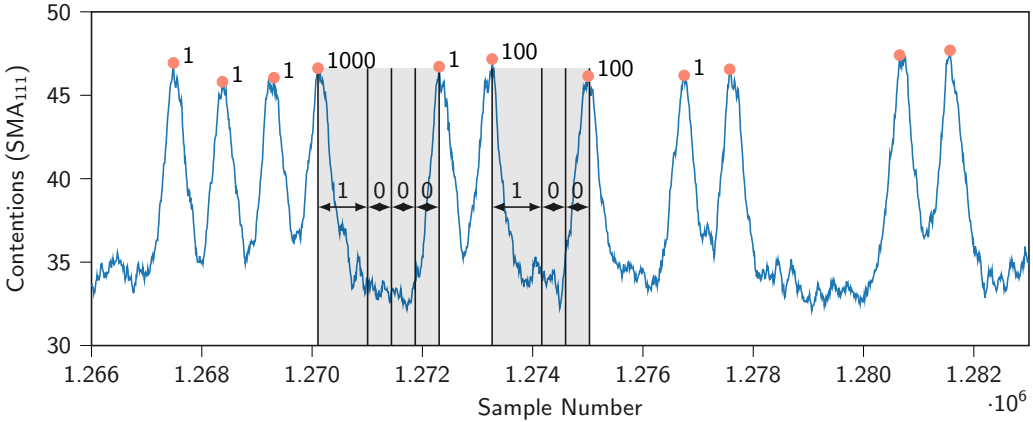
Recovering the Private Key



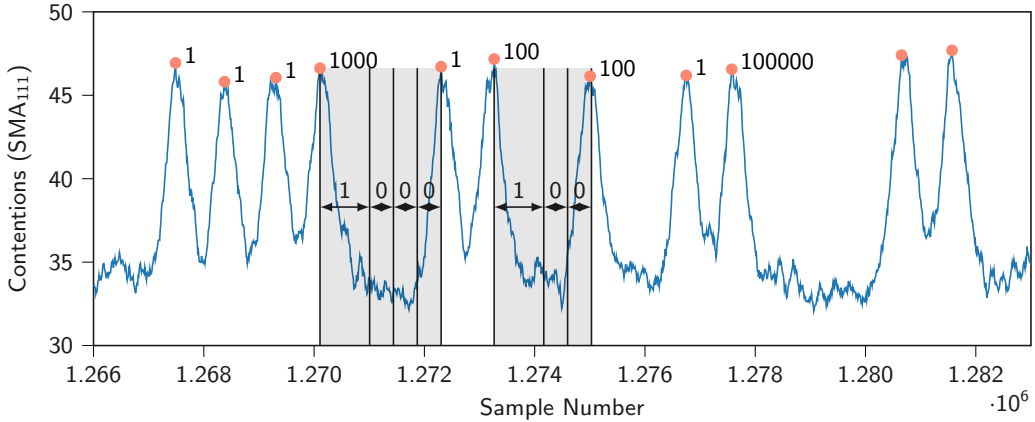
Recovering the Private Key



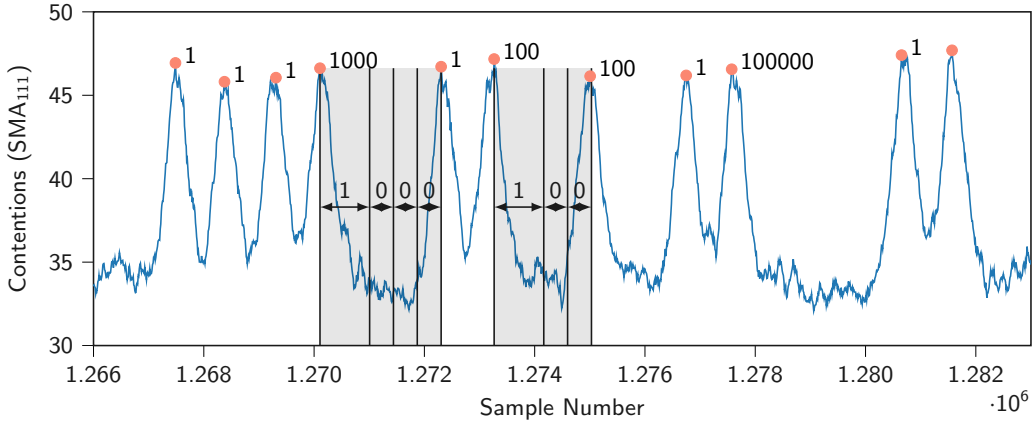
Recovering the Private Key



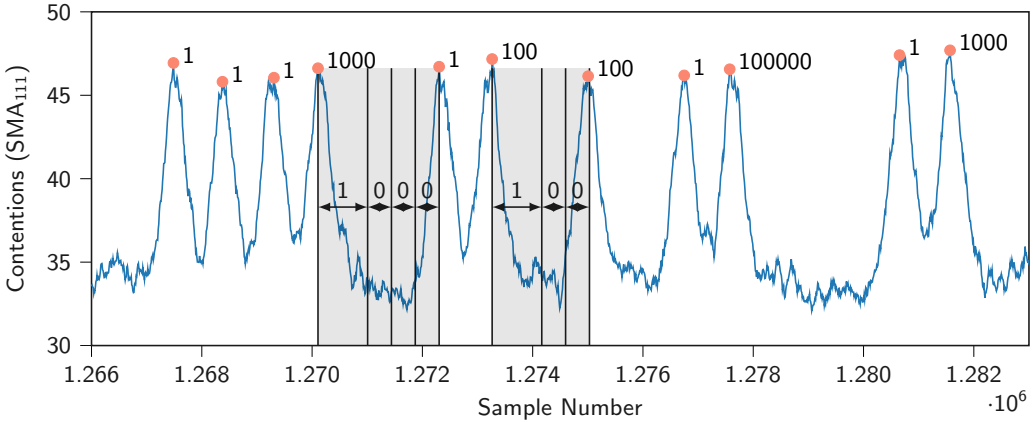
Recovering the Private Key



Recovering the Private Key



Recovering the Private Key



Evaluation Results

Scenario	Average Edit Distance	Error Rate	Recording Time

- 10 keys, generated with `openssl genrsa`
- CPU: AMD Ryzen 7 5800X (Zen 3)

Evaluation Results

Scenario	Average Edit Distance	Error Rate	Recording Time
Cross-Process	4.9 bit	0.12 %	41 min

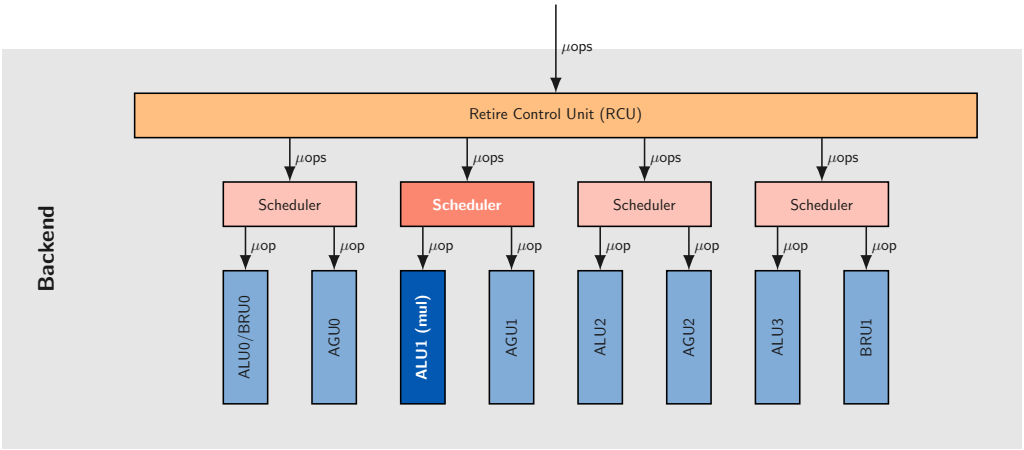
- 10 keys, generated with `openssl genrsa`
- CPU: AMD Ryzen 7 5800X (Zen 3)

Evaluation Results

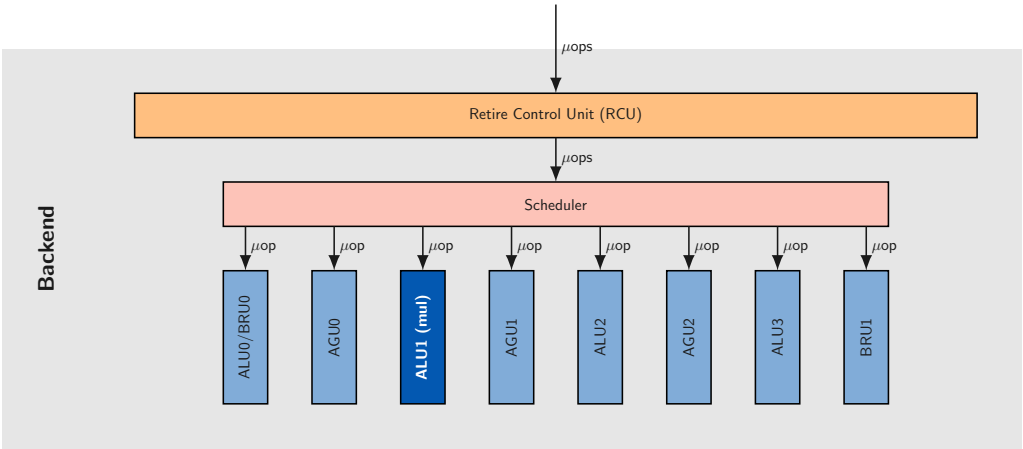
Scenario	Average Edit Distance	Error Rate	Recording Time
Cross-Process	4.9 bit	0.12 %	41 min
Cross-VM	17.8 bit	0.5 %	38 min

- 10 keys, generated with `openssl genrsa`
- CPU: AMD Ryzen 7 5800X (Zen 3)

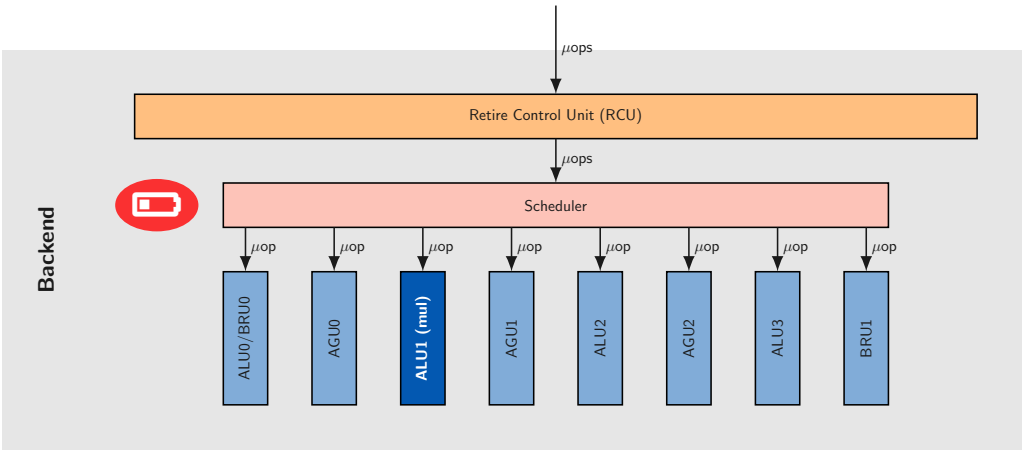
Countermeasures: Hardware



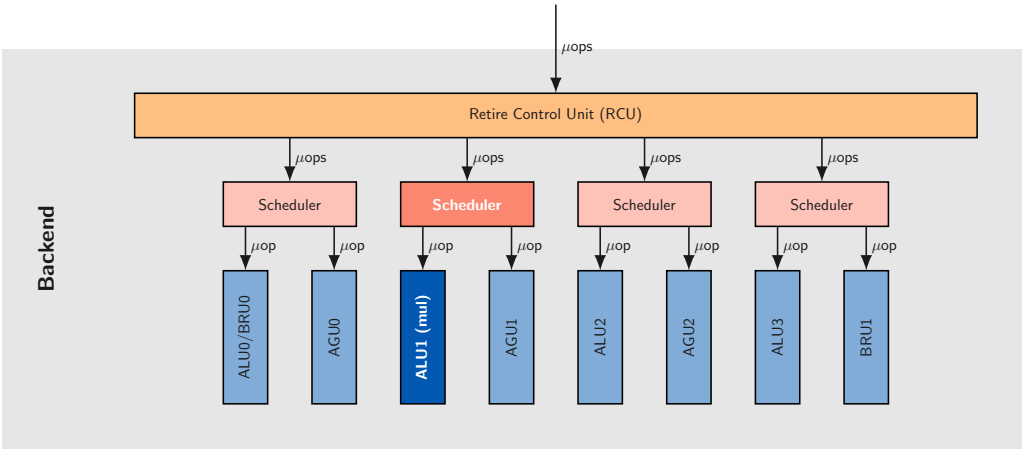
Countermeasures: Hardware



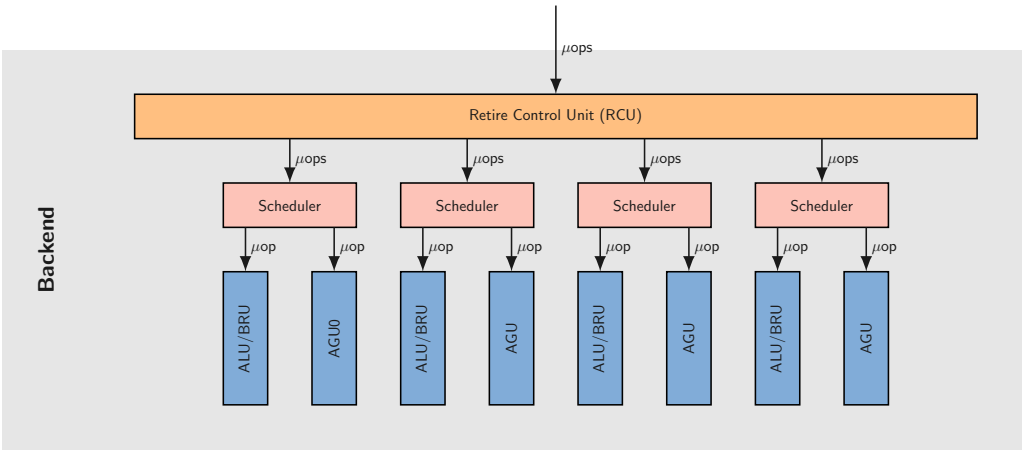
Countermeasures: Hardware



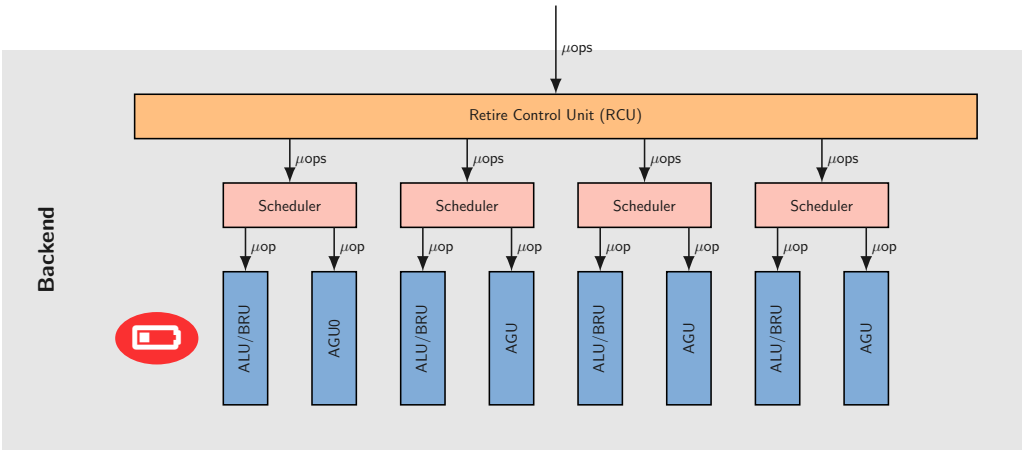
Countermeasures: Hardware



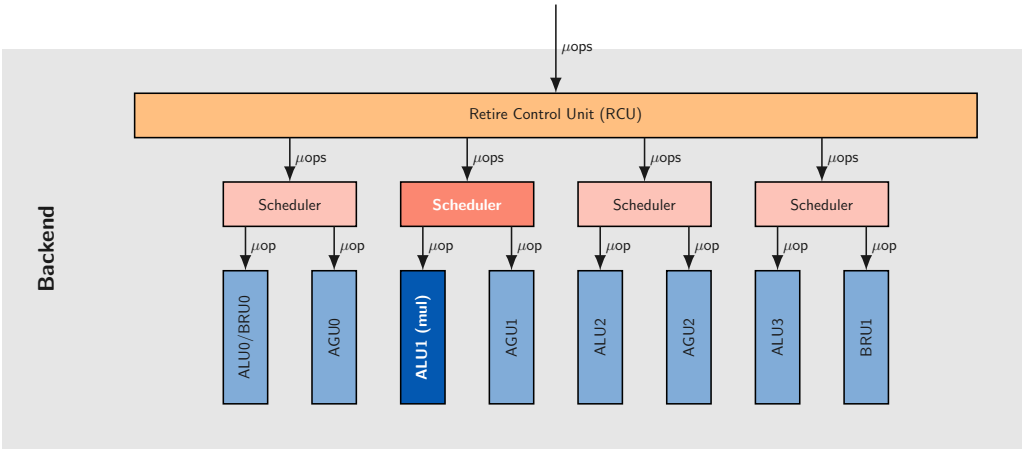
Countermeasures: Hardware



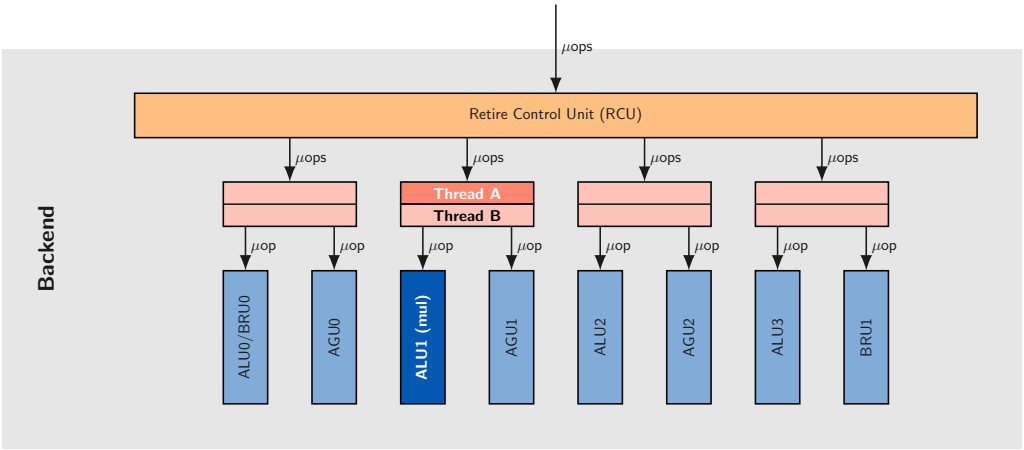
Countermeasures: Hardware



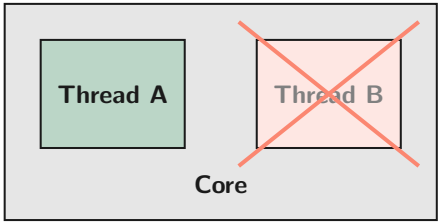
Countermeasures: Hardware



Countermeasures: Hardware

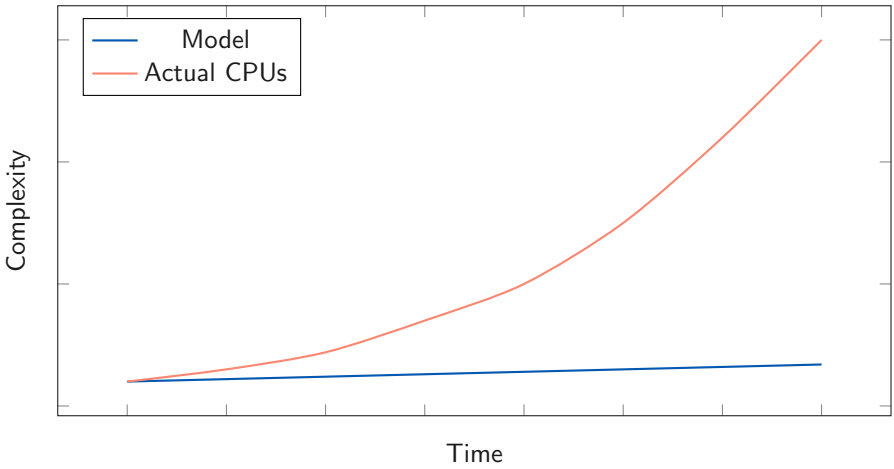


Countermeasures: Software



Side Channels are still Everywhere

Model vs Real-World Microarchitecture Complexity



Conclusion

Conclusion

Conclusion

Conclusion

- Closing a side channel increases security → not in vain, but ...

Conclusion

- Closing a side channel increases security $\rightarrow$ not in vain, but ...
- ... there is always another side channel

Conclusion

- Closing a side channel increases security $\rightarrow$ not in vain, but ...
- ... there is always another side channel
- Computers are complex and constantly become more complex

Conclusion

- Closing a side channel increases security → not in vain, but ...
- ... there is always another side channel
- Computers are complex and constantly become more complex

→ Keep studying processors to find the next one as early as possible and prevent exploitation