CSI:Rowhammer

Closing the Case of Half-Double and Beyond

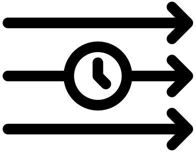
BlackHat Europe 2022

7th December 2022

Andreas Kogler
Graz University of Technology

Jonas Juffinger
Graz University of Technology

Motivation - Rowhammer



Since 2014

Numerous **exploits**

Mitigations
ineffektive

Something
fundamentally
better?

What if we could transparently correct arbitrary bitflips if we know that the OS can reconstruct the data from a different source?

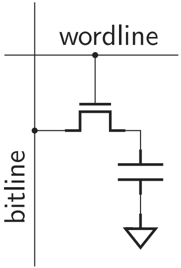
Introduction

Dynamic Random Access Memory (DRAM)

- **Large**, **cheap** and **energy efficient**

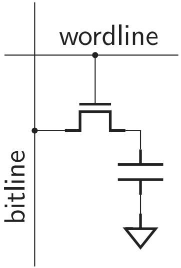
Dynamic Random Access Memory (DRAM)

- **Large**, **cheap** and **energy efficient**
- Cell: one transistor & one capacitor



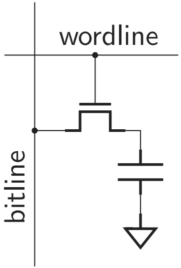
Dynamic Random Access Memory (DRAM)

- **Large**, **cheap** and **energy efficient**
- Cell: one transistor & one capacitor
- Capacitor looses charge over time



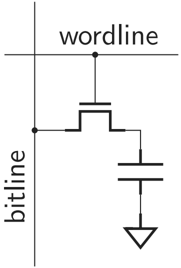
Dynamic Random Access Memory (DRAM)

- **Large**, **cheap** and **energy efficient**
- Cell: one transistor & one capacitor
- Capacitor looses charge over time
- Periodic refreshes **required**



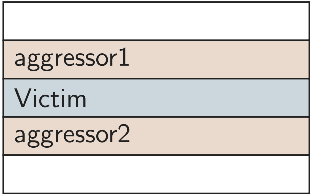
Dynamic Random Access Memory (DRAM)

- **Large**, **cheap** and **energy efficient**
- Cell: one transistor & one capacitor
- Capacitor looses charge over time
- Periodic refreshes **required**
- Organized in **rows**



Rowhammer

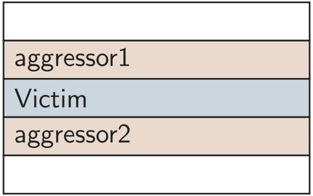
- Hardware fault of the DRAM [3]




```
for (i = 0; i < N; ++i) {
  *aggressor1;
  *aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```

Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows




```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

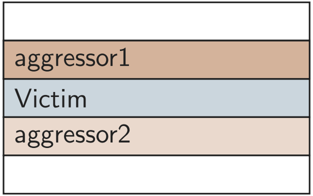
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows


```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows



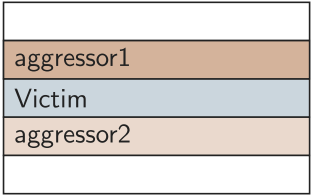

```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows


```
for (i = 0; i < N; ++i) {
  *aggressor1;
  *aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows




```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

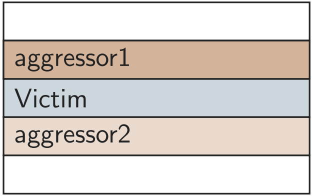
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows


```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows



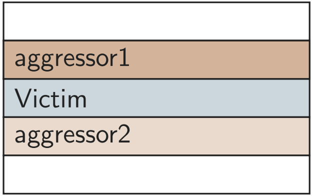

```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows


```
for (i = 0; i < N; ++i) {
  *aggressor1;
  *aggressor2;
  flush(aggressor1);
  flush(aggressor2);
}
```

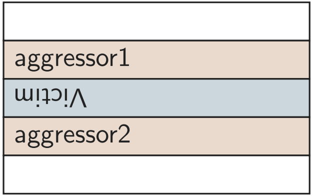
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows



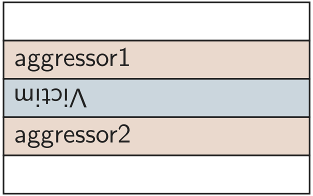

```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows




```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

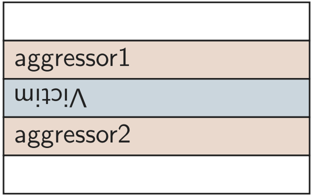
- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows
- Worse with every new DRAM generation




```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

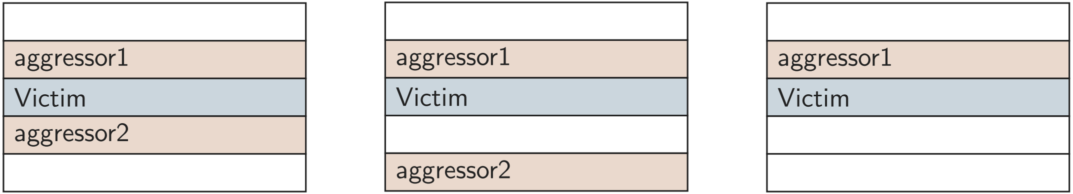
Rowhammer

- Hardware fault of the DRAM [3]
- Frequent accesses flip bits in neighboring rows
- Worse with every new DRAM generation
- Enables attacks, many countermeasures proposed




```
for (i = 0; i < N; ++i) {
    *aggressor1;
    *aggressor2;
    flush(aggressor1);
    flush(aggressor2);
}
```

Rowhammer Patterns



Mitigations focus on the characteristics of Rowhammer

Characteristics 1: Flips are Infrequent

Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes

Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy

Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch **low** number of flips in HW

Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
    - Additional hardware redundancy
    - Catch **low** number of flips in HW
- Increase the **refresh rate**

Characteristics 1: Flips are Infrequent

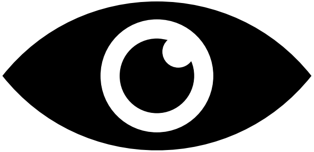
- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch **low** number of flips in HW
- Increase the **refresh rate**
  - Reduce the attackers time window

Characteristics 1: Flips are Infrequent

- **ECC** Error Correcting Codes
  - Additional hardware redundancy
  - Catch **low** number of flips in HW
- Increase the **refresh rate**
  - Reduce the attackers time window
  - Default 64*ms*

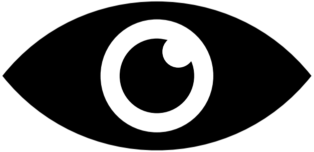
Characteristics 2: Attacks are Detectable

Characteristics 2: Attacks are Detectable



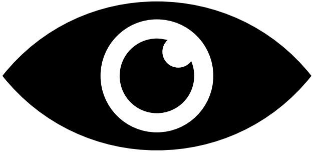
- Use Performance Monitoring Counters (PMCs)

Characteristics 2: Attacks are Detectable



- Use Performance Monitoring Counters (PMCs)
  - Monitor DRAM activity

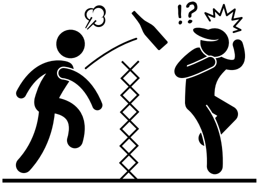
Characteristics 2: Attacks are Detectable



- Use Performance Monitoring Counters (PMCs)
  - Monitor DRAM activity
- Stop **suspicious** applications

Characteristics 3: Flips happen at Distance 1

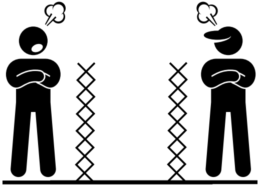
Characteristics 3: Flips happen at Distance 1



- Flips only happen in **neighboring** rows
- Mitigations based row layout
  - Guard rows
  - Targeted Row Refresh

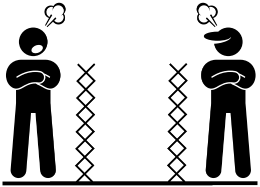
Guard Rows

- Allocate **guard rows**



Agressor
Guard
Victim
Guard
Agressor

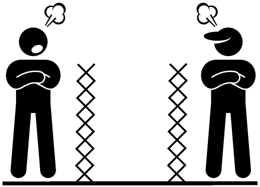
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

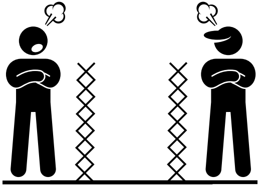
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

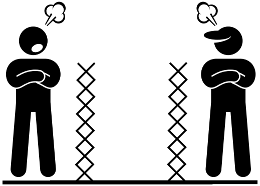
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

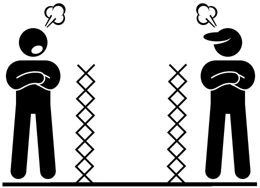
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

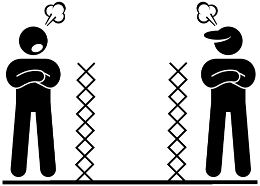
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

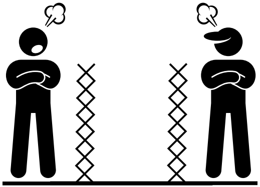
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

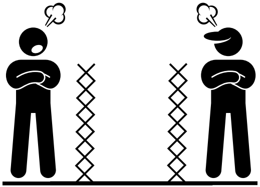
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

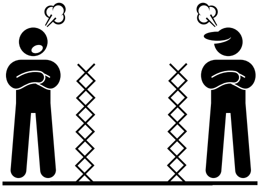
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

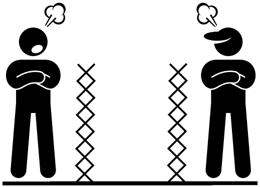
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

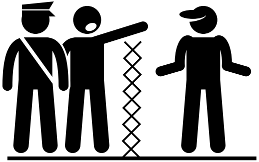
Guard Rows



- Allocate **guard rows**
- Can not reach victim

Agressor
Guard
Victim
Guard
Agressor

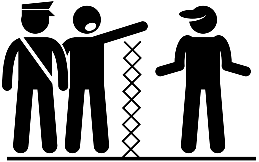
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_{+})$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_{-})$

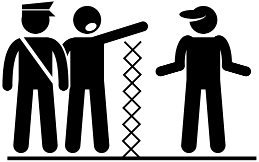
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_{+})$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_{-})$

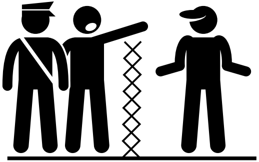
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

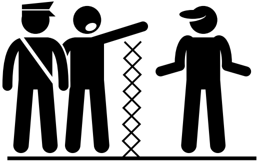
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

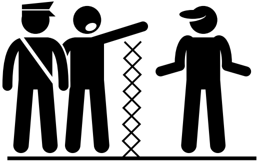
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

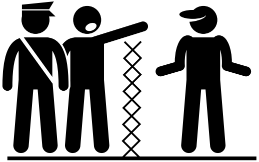
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_{+})$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_{-})$

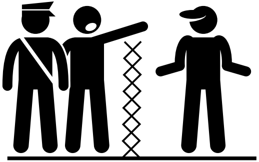
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor $(\mathcal{N}_+)$
Victim $(\mathcal{V})$
Near Aggressor $(\mathcal{N}_-)$

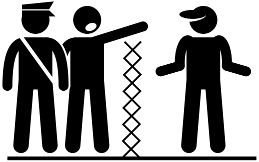
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

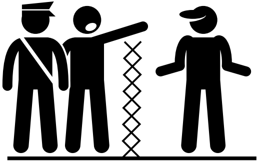
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

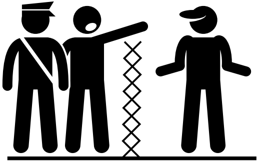
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

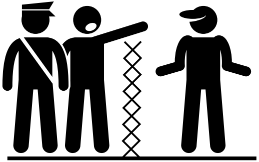
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

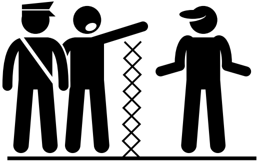
Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$

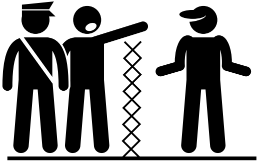
Target Row Refresh (TRR)



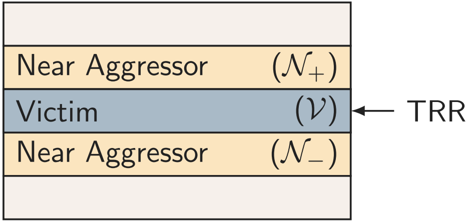
- Additional victim refreshes
- Standardized in LPDDR4x

Near Aggressor	$(\mathcal{N}_{+})$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_{-})$

Target Row Refresh (TRR)



- Additional victim refreshes
- Standardized in LPDDR4x



Let's start the cat and mouse game!

The Problem with Rowhammer Mitigations

The Problem with Rowhammer Countermeasures

- Bit Flips are infrequent - ECC, Increased Refresh Rate

- ~~Bit Flips are infrequent - ECC, Increased Refresh Rate~~ [3]

The Problem with Rowhammer Countermeasures

- ~~Bit Flips are infrequent - ECC, Increased Refresh Rate~~ [3]
- Detectable with Performance Counters - ANVIL, HexPADS

The Problem with Rowhammer Countermeasures

- ~~Bit Flips are infrequent - ECC, Increased Refresh Rate~~ [3]
- ~~Detectable with Performance Counters - ANVIL, HexPADS~~ [2]

The Problem with Rowhammer Countermeasures

- ~~Bit Flips are infrequent - ECC, Increased Refresh Rate~~ [3]
- ~~Detectable with Performance Counters - ANVIL, HexPADS~~ [2]
- Hammer Distance is 1 - TRR, ZebRAM, B-CATT

The Problem with Rowhammer Countermeasures

- ~~Bit Flips are infrequent - ECC, Increased Refresh Rate~~ [3]
- ~~Detectable with Performance Counters - ANVIL, HexPADS~~ [2]
- ~~Hammer Distance is 1 - TRR, ZebRAM, B-CATT~~ [4]

Would perfect TRR fix Rowhammer?

No

The Half-Double Effect

Distance 1

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-1**
  - $(\mathcal{N}_+ \to \mathcal{N}_-)^\infty$
  - *Classic* double-sided Rowhammer

Distance 1

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-1**
  - $(\mathcal{N}_+ \rightarrow \mathcal{N}_-)^\infty$
  - *Classic* double-sided Rowhammer
- **First** flip after:
  - 18 000 hammers in 1.2 *ms*

Distance 1

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-1**
  - $(\mathcal{N}_+ \rightarrow \mathcal{N}_-)^\infty$
  - *Classic* double-sided Rowhammer
- **First** flip after:
  - 18 000 hammers in 1.2 *ms*
  - ✓ **Within** the refresh window
  - ✗ **Mitigated** by TRR

Distance 2

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-2**
  - $(\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\infty$
  - Distance two double-sided Rowhammer

Distance 2

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-2**
  - $(\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\infty$
  - Distance two double-sided Rowhammer
- **First** flip after:
  - 4 000 000 hammers in 270 *ms*

Distance 2

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Distance-2**
  - $(\mathcal{F}_+ \to \mathcal{F}_-)^\infty$
  - Distance two double-sided Rowhammer
- **First** flip after:
  - 4 000 000 hammers in 270 *ms*
  - ✗ **Not** within the refresh windows

Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Half-Double**
  - $((\mathcal{F}_+ \to \mathcal{F}_-)^\beta \to \mathcal{N}_+ \to \mathcal{N}_-)^\infty$
  - **Many** distance-2 accesses with a **few** distance-1 accesses

Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Half-Double**
  - $((\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\beta \rightarrow \mathcal{N}_+ \rightarrow \mathcal{N}_-)^\infty$
  - **Many** distance-2 accesses with a **few** distance-1 accesses
- **First** flip after:
  - 296 960 hammers in 20 *ms*
  - Dilution $\beta = 57$ (5120 distance-1 accesses)

Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Half-Double**
  - $((\mathcal{F}_+ \to \mathcal{F}_-)^\beta \to \mathcal{N}_+ \to \mathcal{N}_-)^\infty$
  - **Many** distance-2 accesses with a **few** distance-1 accesses
- **First** flip after:
  - 296 960 hammers in 20 *ms*
  - Dilution $\beta = 57$ (5120 distance-1 accesses)
  - ✓ **Within** the refresh window

Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Half-Double**
  - $((\mathcal{F}_+ \rightarrow \mathcal{F}_-)^\beta \rightarrow \mathcal{N}_+ \rightarrow \mathcal{N}_-)^\infty$
  - **Many** distance-2 accesses with a **few** distance-1 accesses
- **First** flip after:
  - 296 960 hammers in 20 *ms*
  - Dilution $\beta = 57$ (5120 distance-1 accesses)
  - ✓ **Within** the refresh window
  - ✓ **Assisted** by TRR

Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

- **Half-Double**
  - $((\mathcal{F}_+ \to \mathcal{F}_-)^\beta \to \mathcal{N}_+ \to \mathcal{N}_-)^\infty$
  - **Many** distance-2 accesses with a **few** distance-1 accesses
- **First** flip after:
  - 296 960 hammers in 20 *ms*
  - Dilution $\beta = 57$ (5120 distance-1 accesses)
  - ✓ **Within** the refresh window
  - ✓ **Assisted** by TRR

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

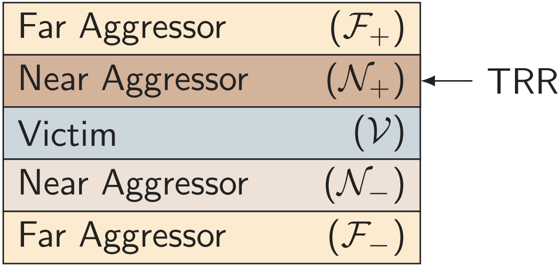
TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

TRR Assisted Half-Double



TRR Assisted Half-Double

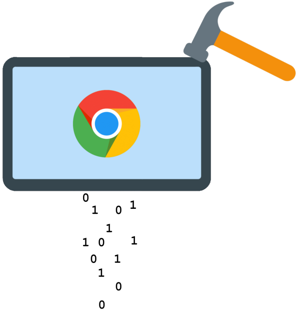
Far Aggressor	$(\mathcal{F}_+)$	
Near Aggressor	$(\mathcal{N}_+)$	
Victim	$(\mathcal{V})$	
Near Aggressor	$(\mathcal{N}_-)$	$\leftarrow$ TRR
Far Aggressor	$(\mathcal{F}_-)$	

TRR Assisted Half-Double

Far Aggressor	$(\mathcal{F}_+)$
Near Aggressor	$(\mathcal{N}_+)$
Victim	$(\mathcal{V})$
Near Aggressor	$(\mathcal{N}_-)$
Far Aggressor	$(\mathcal{F}_-)$

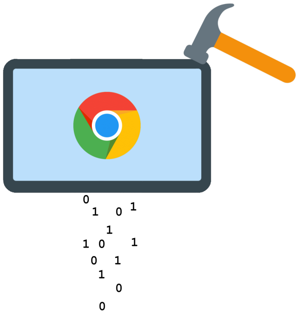
Exploitable in the Wild?

End-to-End Exploit – Overview



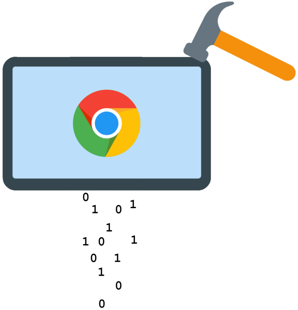
- Target PFN in Page Table Entry [6]

End-to-End Exploit – Overview



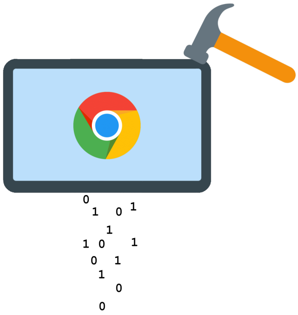
- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory

End-to-End Exploit – Overview



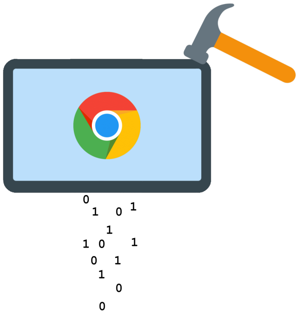
- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory
- **C2**: Alternative to Memory Templating

End-to-End Exploit – Overview



- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory
- **C2**: Alternative to Memory Templating
- **C3**: Memory Massaging

End-to-End Exploit – Overview



- Target PFN in Page Table Entry [6]
- **C1**: Allocation of Contiguous Memory
- **C2**: Alternative to Memory Templating
- **C3**: Memory Massaging
- **C4**: Bit-Flip Verification

C4 – Bit-Flip Verification

- Corrupt page table entries can kill the attacker process

C4 – Bit-Flip Verification

- Corrupt page table entries can kill the attacker process

```
if ( /*misprediction*/ ) {
    access(probe + (*ptr & 1));
}
if ( is_cached(probe) ) {
    // ptr[0-4] valid
}
```

- **Verify** if address save to access
- **Spectre** gadget

C4 – Bit-Flip Verification

- Corrupt page table entries can kill the attacker process

```
if ( /*misprediction*/ ) {
    access(probe + (*ptr & 1));
}
if ( is_cached(probe) ) {
    // ptr[0-4] valid
}
```

- **Verify** if address save to access
- **Spectre** gadget
- Cached → accessible

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

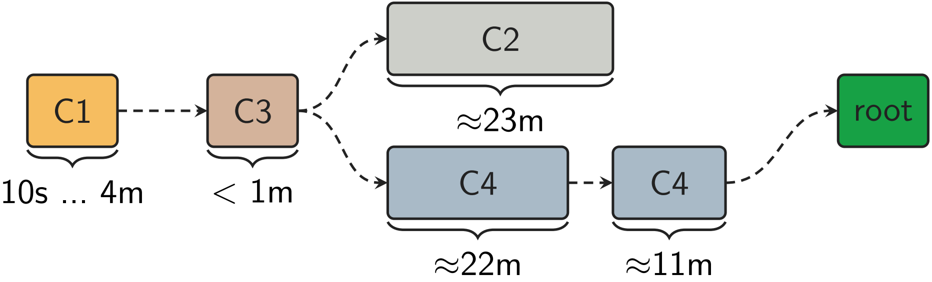
C4 – Bit-Flip Verification

- Corrupt page table entries can kill the attacker process

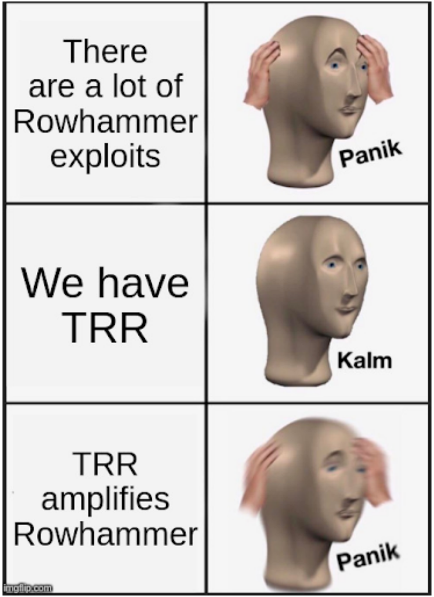
```
if ( /*misprediction*/ ) {
    access(probe + (*ptr & 1));
}
if ( is_cached(probe) ) {
    // ptr[0-4] valid
}
```

- **Verify** if address save to access
- **Spectre** gadget
- Cached → accessible
- **Suppresses** corruption faults

End-to-End Exploit - Timings



Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)


There are a lot of Rowhammer exploits
Panik
We have TRR
Kalm
TRR amplifies Rowhammer
Panik
imgflip.com

Rowhammer is still not fixed

~~ECC~~

~~ChipKill~~

~~TRR~~

~~PARA~~

~~PRA~~

~~Refresh Rate~~

~~MASCAT~~

~~HexPADS~~

~~ARMOR~~

~~NO OOM~~

~~B-CATT~~

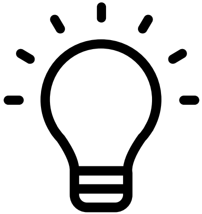
~~G-CATT~~

~~ZebRAM~~

~~MemGuard~~

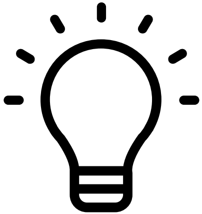
Rethinking Rowhammer Mitigations

General approach to data integrity protection

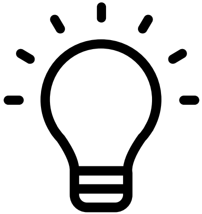


- **General** approach to data integrity protection

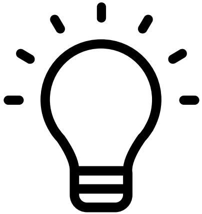
Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)



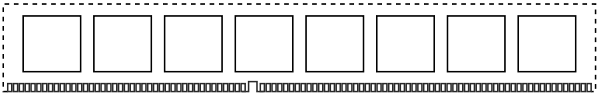
- **General** approach to data integrity protection
- Detect all data integrity failures with a MAC

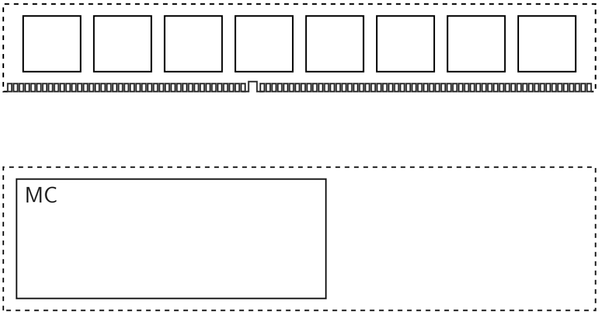


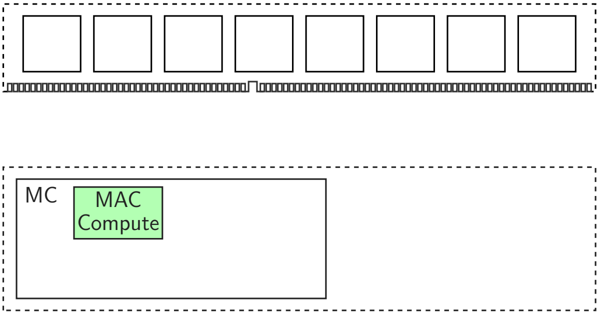
- **General** approach to data integrity protection
- Detect all data integrity failures with a MAC
- Best effort correction

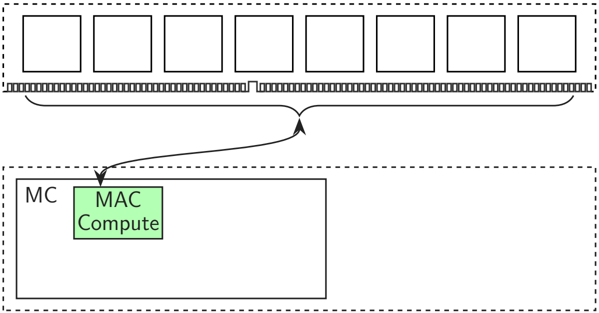


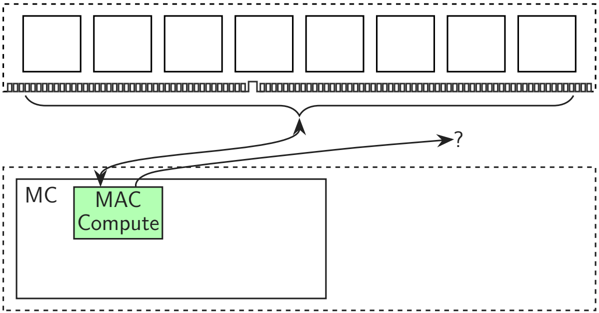
- **General** approach to data integrity protection
- Detect all data integrity failures with a MAC
- Best effort correction
- All Rowhammer attacks are DoS in the **worst case**

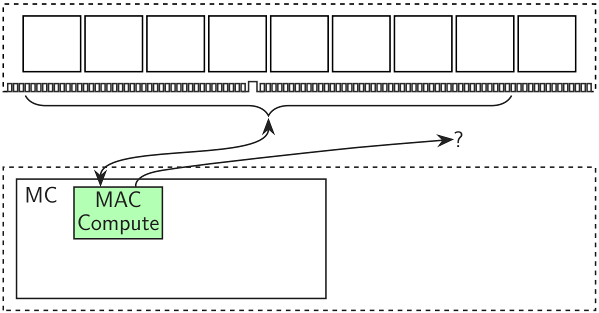


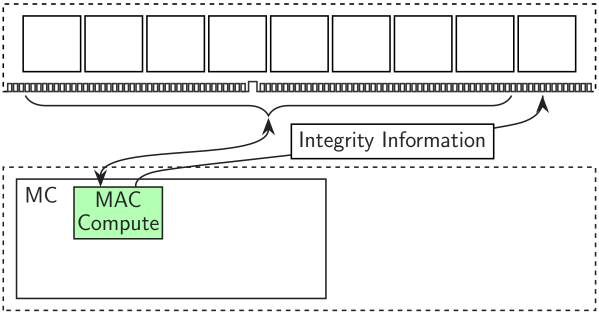

MC


MC
MAC
Compute

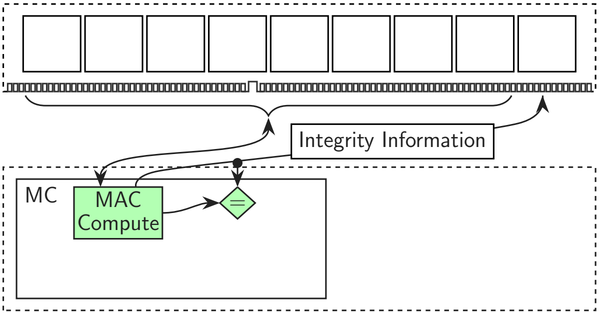

MC
MAC
Compute

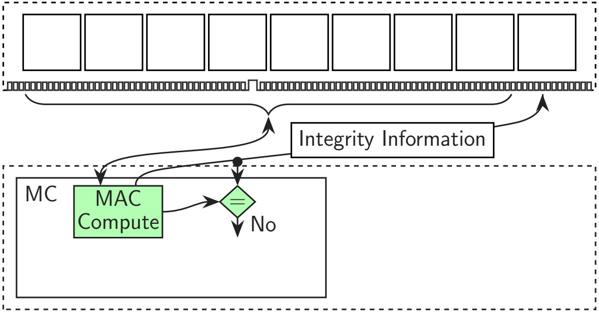

?
MC
MAC
Compute

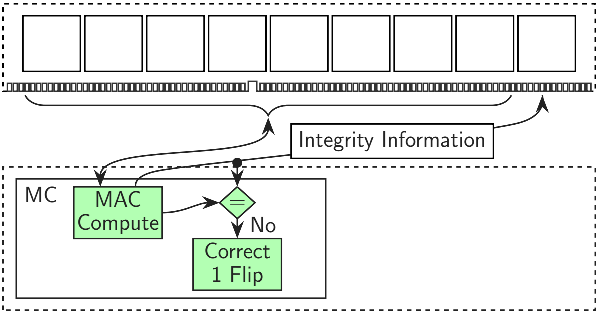

?
MC
MAC
Compute

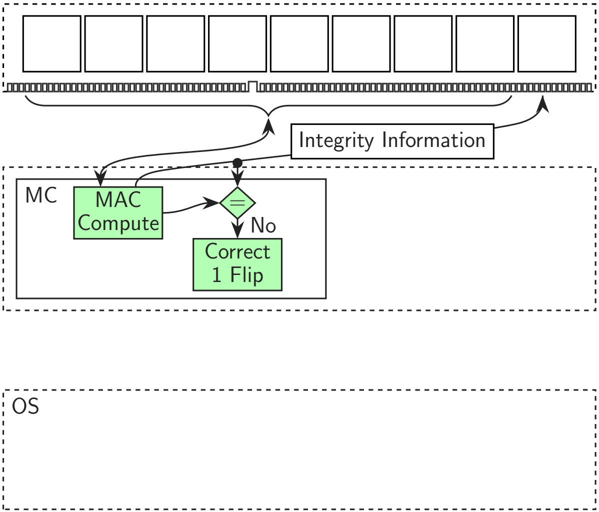


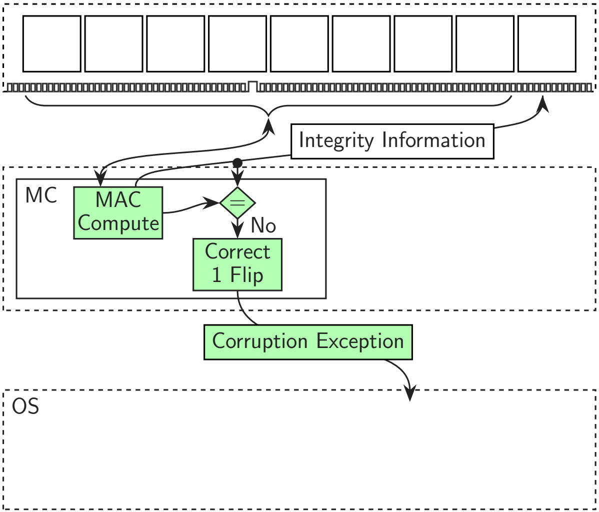
Integrity Information
MC
MAC
Compute


Integrity Information
MC
MAC
Compute
=

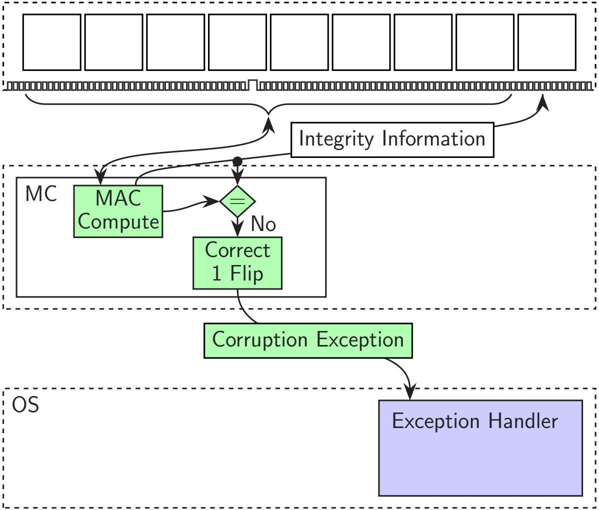

Integrity Information
MC
MAC
Compute
=
No

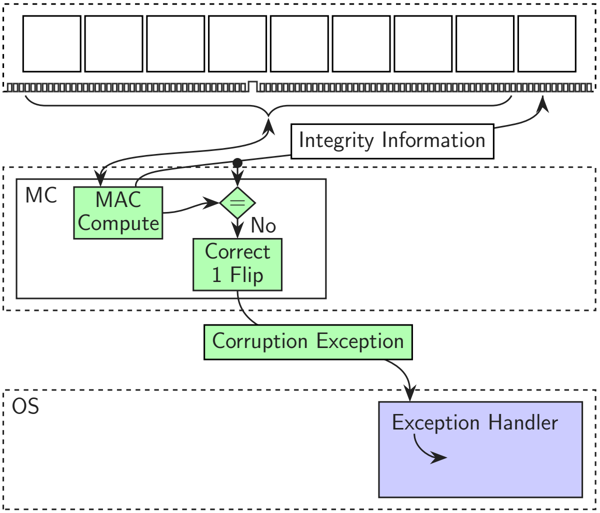

Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip

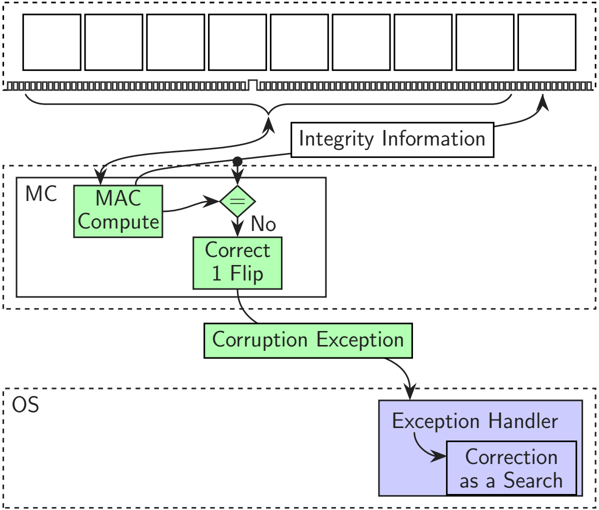

Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
OS



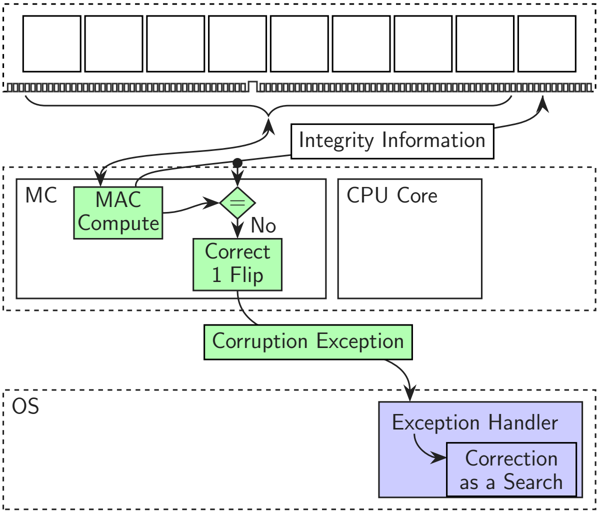
Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

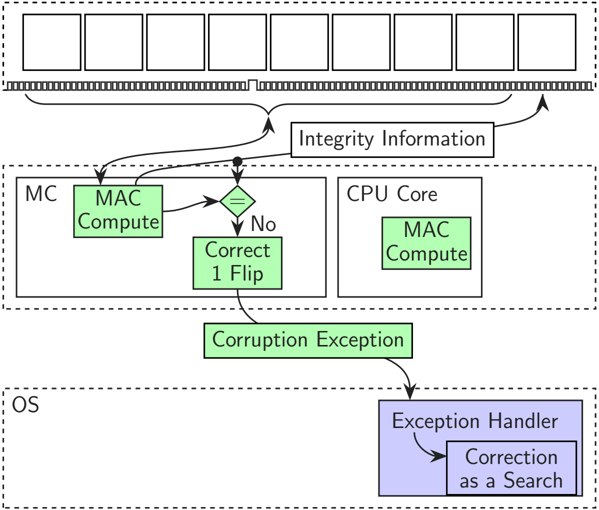

Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
Corruption Exception
OS
Exception Handler

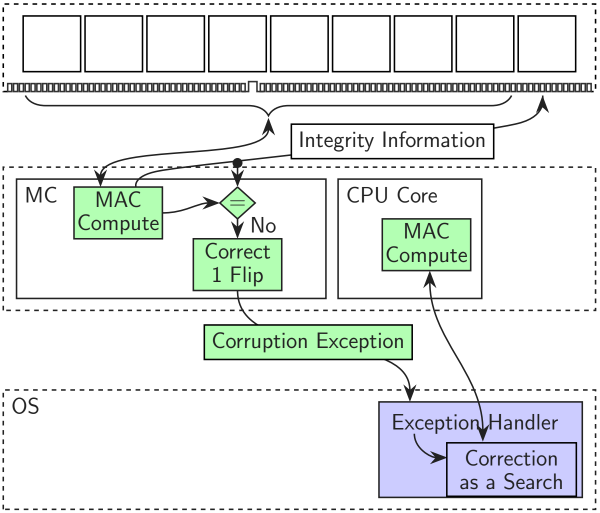

Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
Corruption Exception
OS
Exception Handler



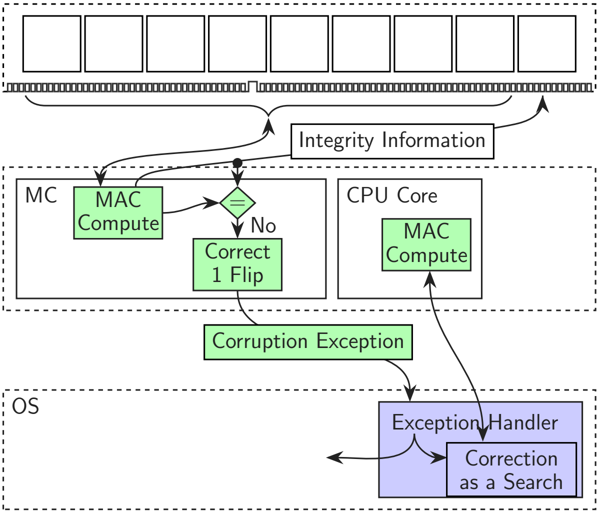
Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
Corruption Exception
OS
Exception Handler
Correction as a Search


Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
Corruption Exception
OS
Exception Handler
Correction
as a Search

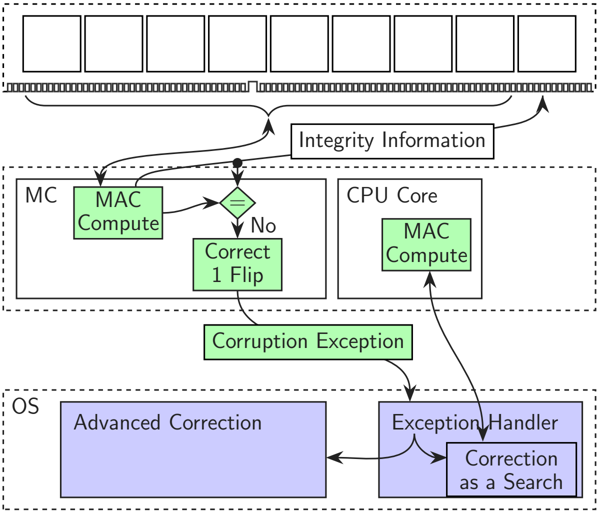

Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Corruption Exception
OS
Exception Handler
Correction as a Search

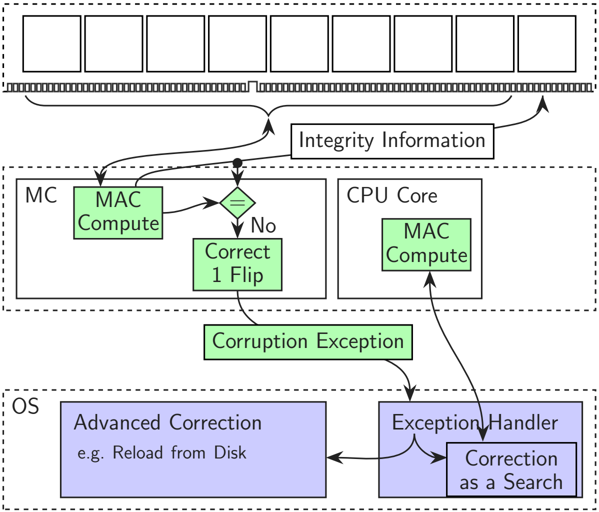


Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Corruption Exception
OS
Exception Handler
Correction as a Search

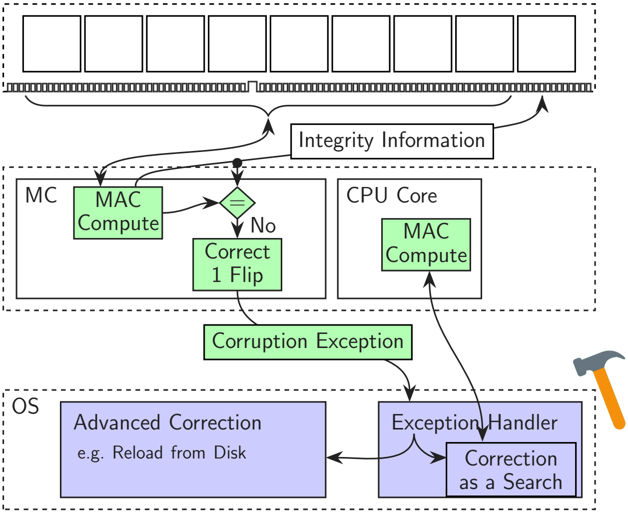


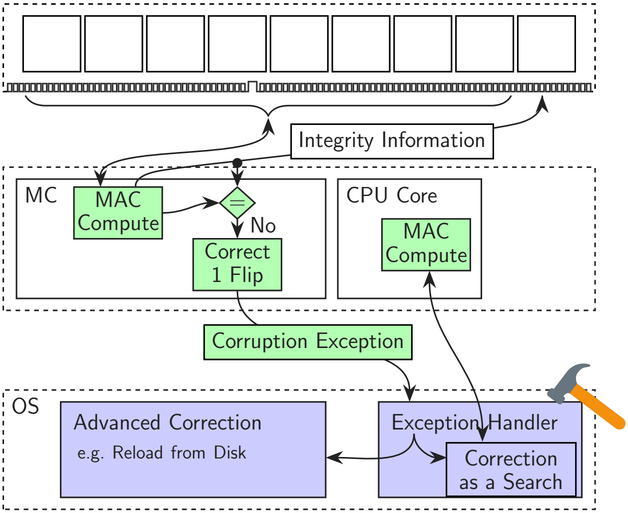
Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
MAC
Compute
Corruption Exception
OS
Exception Handler
Correction
as a Search


Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Corruption Exception
OS
Advanced Correction
Exception Handler
Correction as a Search

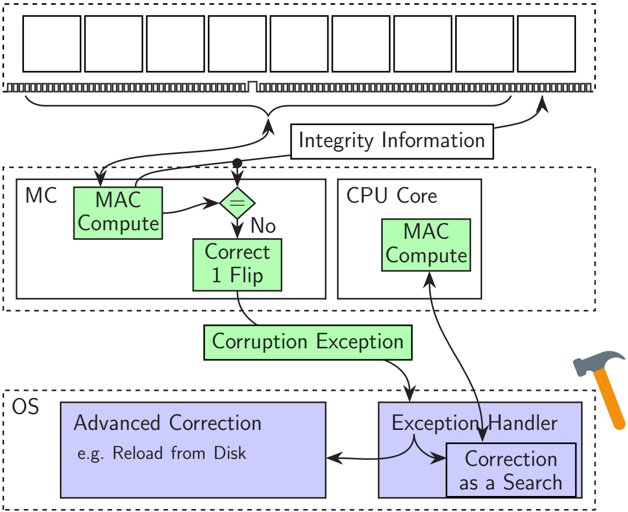


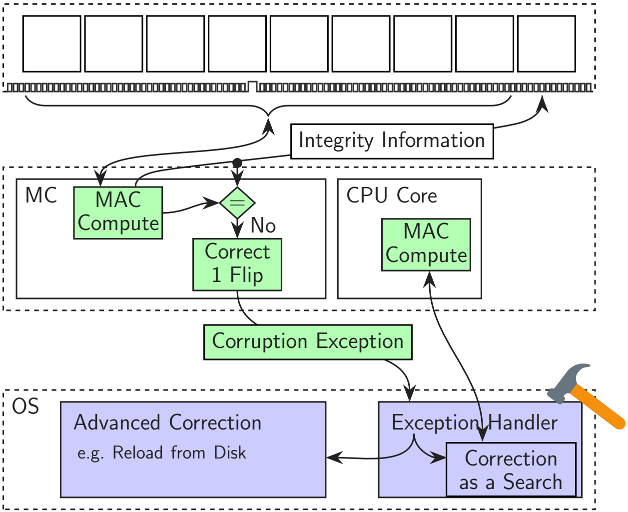
Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
MAC
Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction
as a Search

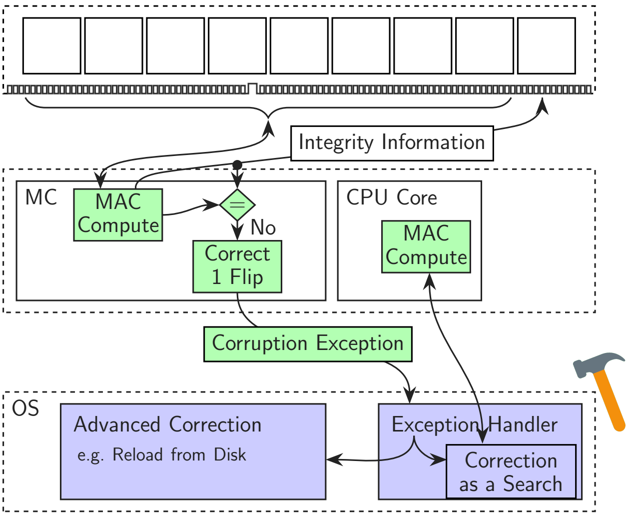

Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
MAC
Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction
as a Search

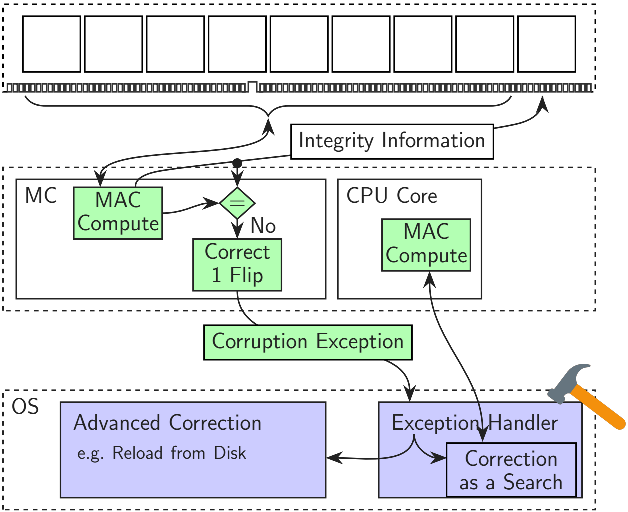


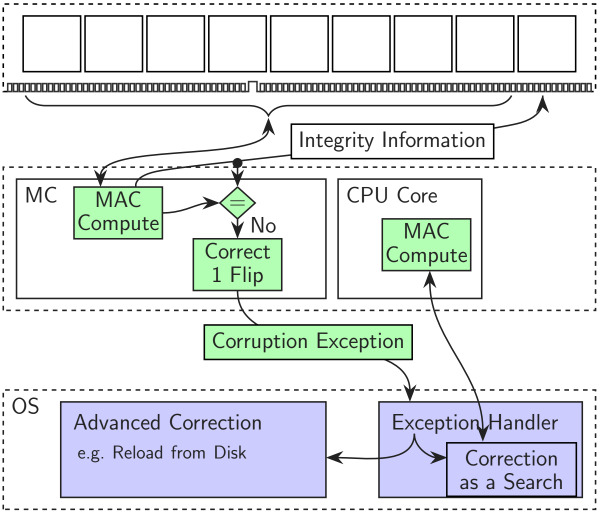
Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
MAC
Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction
as a Search


Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
MAC
Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction
as a Search

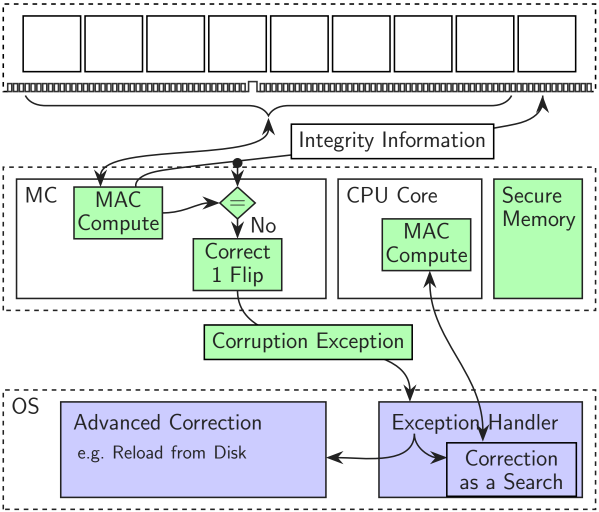

Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search

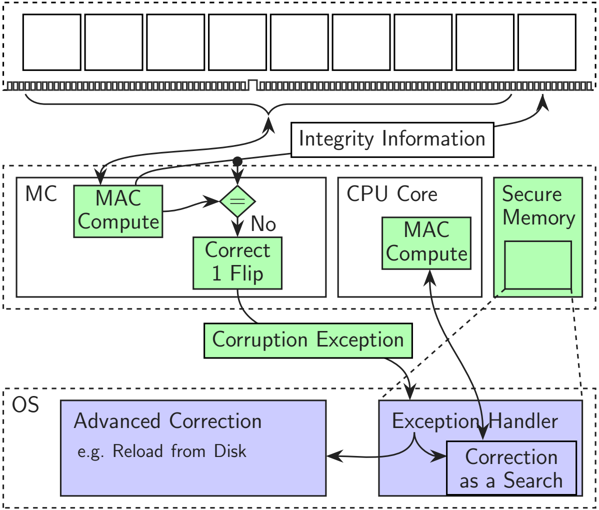

Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search

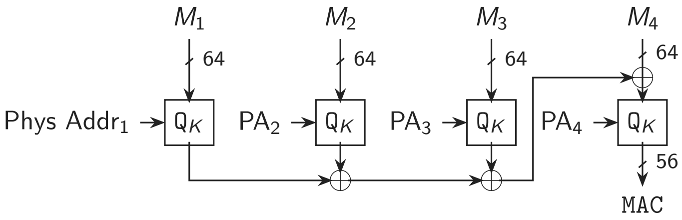

Integrity Information
MC
MAC
Compute
=
No
Correct
1 Flip
CPU Core
MAC
Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction
as a Search



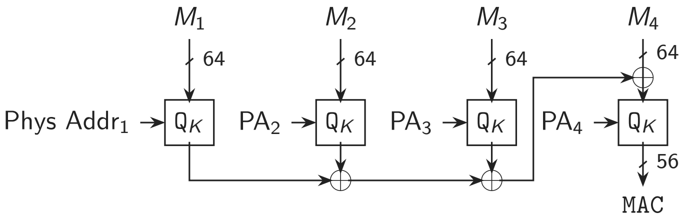
Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search


Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Secure Memory
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search


Integrity Information
MC
MAC Compute
=
No
Correct 1 Flip
CPU Core
MAC Compute
Secure Memory
Corruption Exception
OS
Advanced Correction
e.g. Reload from Disk
Exception Handler
Correction as a Search

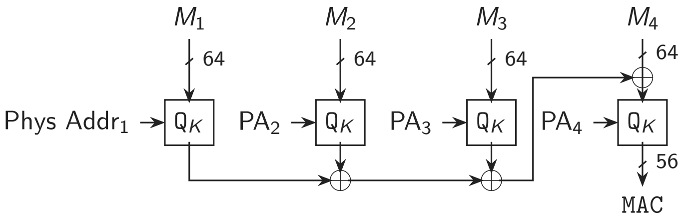

M_1
64
M_2
64
M_3
64
M_4
64
Phys Addr$_1$
Q_K
PA_2
Q_K
PA_3
Q_K
PA_4
Q_K
56
MAC

CSI:Rowhammer – MAC Design



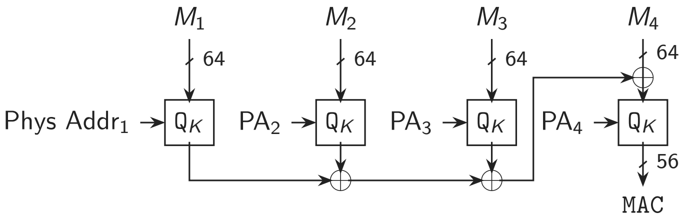
- Pipelined PMAC construction

CSI:Rowhammer – MAC Design



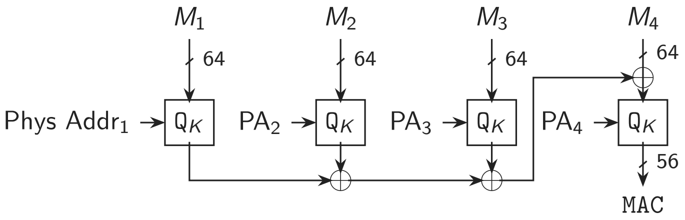
- Pipelined PMAC construction
- QARMA$_5$-64-σ_0 block cipher [1]

CSI:Rowhammer – MAC Design



- Pipelined PMAC construction
- QARMA$_5$-64-σ_0 block cipher [1]
- 256-bit data 5.13 ns

CSI:Rowhammer – MAC Design

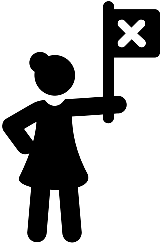


- Pipelined PMAC construction
- QARMA$_5$-64-σ_0 block cipher [1]
- 256-bit data 5.13 ns
- 512-bit data 6.60 ns

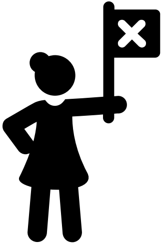
Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

Error Correction in Software

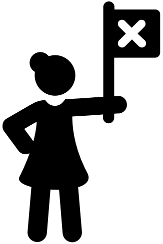
CSI:Rowhammer – Corruption Exception



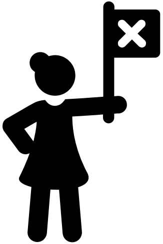
CSI:Rowhammer – Corruption Exception



- Physical address in CR2

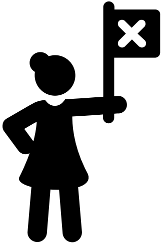


- Physical address in CR2
- **Load** corrupted data with `csi_load`



- Physical address in CR2
- **Load** corrupted data with `csi_load`
- **Compute** MAC with `csi_mac`

CSI:Rowhammer – Corruption Exception



- Physical address in CR2
- **Load** corrupted data with `csi_load`
- **Compute** MAC with `csi_mac`
- **Write** back corrected data with `csi_xchg`

MACs cannot correct bit flips

- MACs **cannot** correct bit flips

- MACs **cannot** correct bit flips
- Brute force search with approximate equality

CSI:Rowhammer – Correction as a Search

- MACs **cannot** correct bit flips
- Brute force search with approximate equality

  0010110100101101 $\rightarrow$ 01011010

- MACs **cannot** correct bit flips
- Brute force search with approximate equality

  0010110100101101 → 01011010

  0010110100101101 → 01010010 ✓

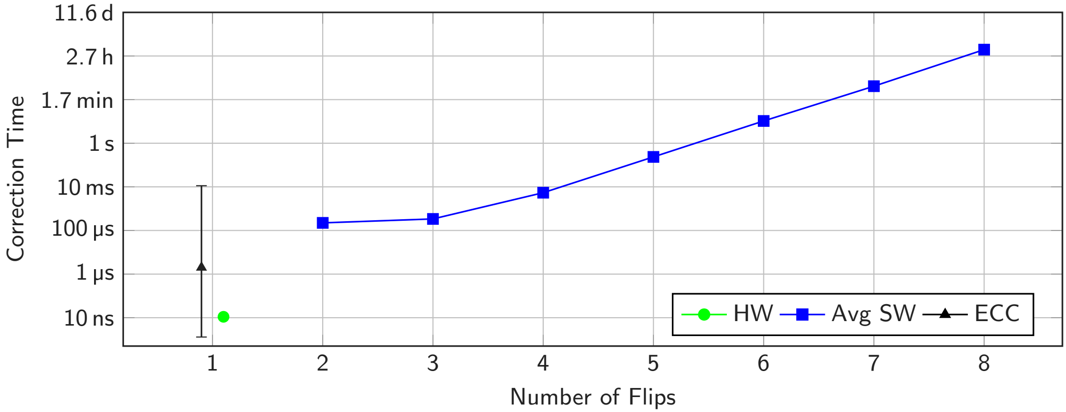
CSI:Rowhammer – Correction as a Search

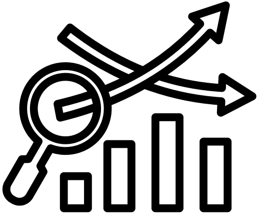
- MACs **cannot** correct bit flips
- Brute force search with approximate equality

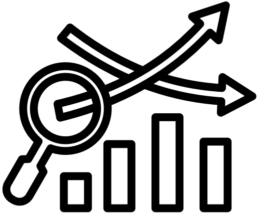
  0010110100101101 → 01011010

  0010110100101101 → 01010010 ✓
- Parity bits to shrink search space

CSI:Rowhammer – Correction Time

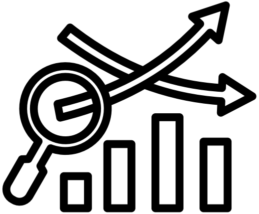






- Implemented CSI:Rowhammer in **gem5**

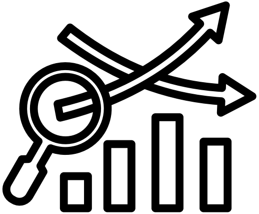
CSI:Rowhammer – Evaluation



- Implemented CSI:Rowhammer in **gem5**
- Modified **Linux** kernel

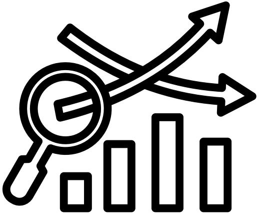
Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

CSI:Rowhammer – Evaluation



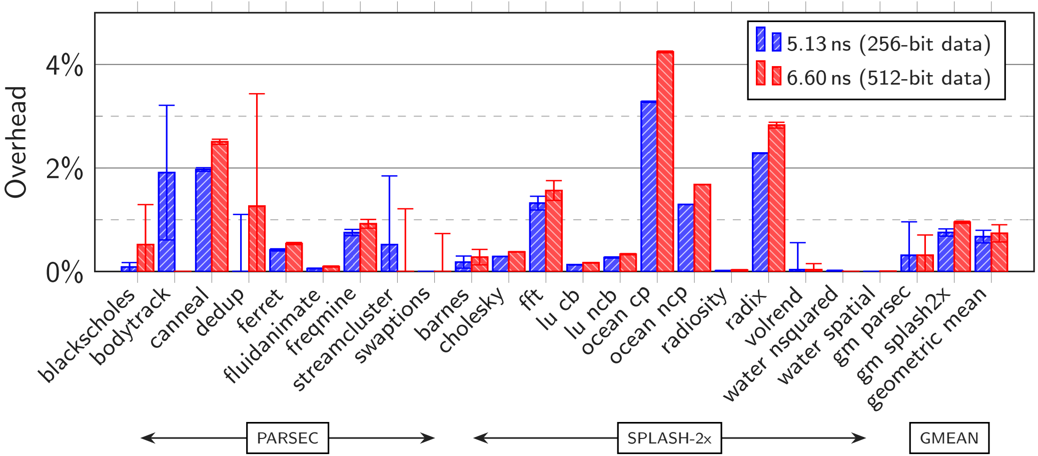
- Implemented CSI:Rowhammer in **gem5**
- Modified **Linux** kernel
- Evaluated correct functionality

CSI:Rowhammer – Evaluation



- Implemented CSI:Rowhammer in **gem5**
- Modified **Linux** kernel
- Evaluated correct functionality
- Evaluated performance overhead

CSI:Rowhammer – Performance Overhead



Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

- **Approximate Equality**

CSI:Rowhammer – Security Evaluation

- **Approximate Equality**

Data Flips	$\log_2(p)$	d	MAC Strength
5	26.0	3	41.2
6	31.5	2	45.4
7	38.8	1	50.2
8	42.4	0	56.0

CSI:Rowhammer – Security Evaluation

- **Approximate Equality**
- Silent Data Corruption rate less than once per 10^9 billion years.

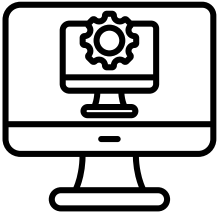
Data Flips	$\log_2(p)$	d	MAC Strength
5	26.0	3	41.2
6	31.5	2	45.4
7	38.8	1	50.2
8	42.4	0	56.0

CSI:Rowhammer – Security Evaluation

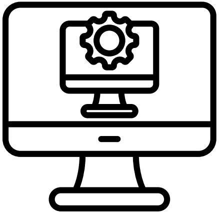
- **Approximate Equality**
- Silent Data Corruption rate less than once per 10^9 billion years.
- Rowhammer second preimage after one year: $9.75 \cdot 10^{-5}\%$

Data Flips	$\log_2(p)$	d	MAC Strength
5	26.0	3	41.2
6	31.5	2	45.4
7	38.8	1	50.2
8	42.4	0	56.0

CSI:Rowhammer – Virtual Environments

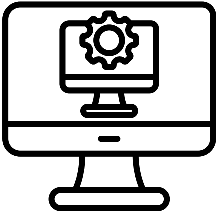


CSI:Rowhammer – Virtual Environments



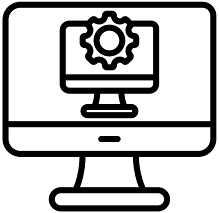
- Only **host support** required

CSI:Rowhammer – Virtual Environments



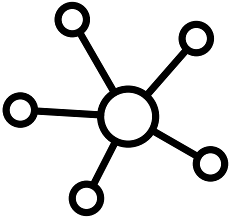
- Only **host support** required
- Guest support optional

Andreas Kogler (@0xhilbert) Jonas Juffinger (@notimaginary_)

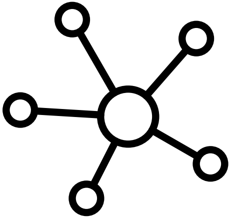


- Only **host support** required
- Guest support optional
- Makes upgrading easy

The Possibilities

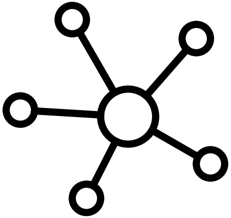


Correction in **software** opens up many new possibilities



Correction in **software** opens up many new possibilities

- Data correction as a service (DCaaS)



Correction in **software** opens up many new possibilities

- Data correction as a service (DCaaS)
- Correction by ChatGPT^{TM}

CSI:Rowhammer – Correction by ChatGPT

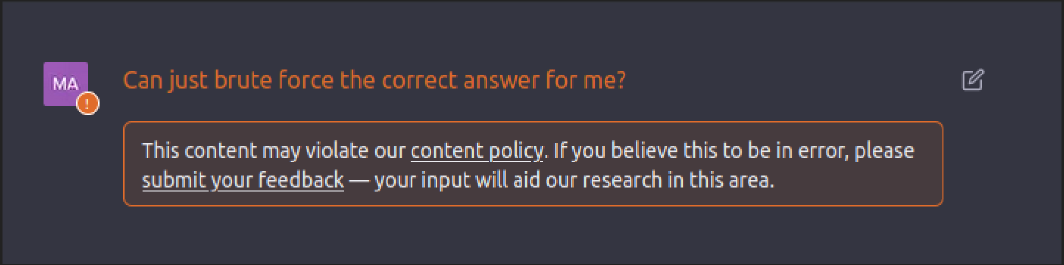
MA

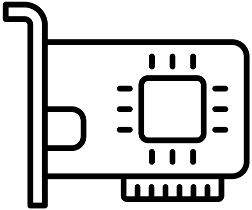
Find the error in "Testmessoge" so that is has the following SHA512 hash "9167bf29600fea2cb1ba4e21e2f311bf5c2641f0cd5cb5bec948177264ff78e2bc346216427b75e75fe67e842ef3085ac303b8111a37adeaf977b2b269ada636"

CSI:Rowhammer – Correction by ChatGPT

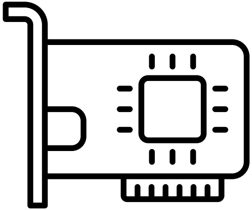
It is not possible for the string "Testmessoge" to have the SHA512 hash "9167bf29600fea2cb1ba4e21e2f311bf5c2641f0cd5cb5bec948177264ff78e2bc346216427b75e75fe67e842ef3085ac303b8111a37adeaf977b2b269ada636". This is because the SHA512 hash function is a one-way function, meaning that it is not possible to determine the input that produced a given hash. Therefore, it is not possible to determine the error in the input string "Testmessoge" based on the given hash value.

CSI:Rowhammer – Correction by ChatGPT



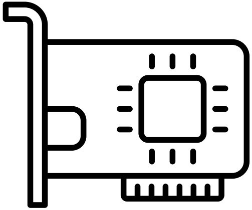


Hardware acceleration



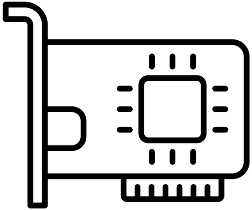
Hardware acceleration

- Ice Lake SP Xeon Server die size: 628 mm^2



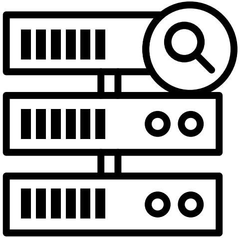
Hardware acceleration

- Ice Lake SP Xeon Server die size: $628\,\mathrm{mm}^2$
- Fits roughly 500 000 QARMA blocks

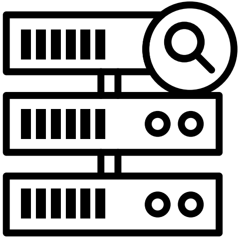


Hardware acceleration

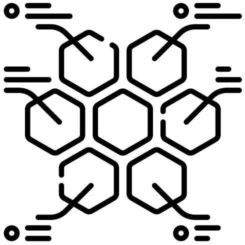
- Ice Lake SP Xeon Server die size: 628 mm^2
- Fits roughly 500 000 QARMA blocks
- 8 bit flips correction: 5.44 h $\rightarrow$ 40 ms



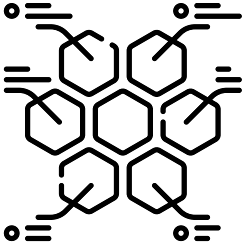
- Multi-bit flips not happening often



- Multi-bit flips not happening often
- **Specialized** data correction nodes

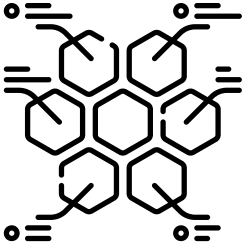


Kernel data with **known** structures



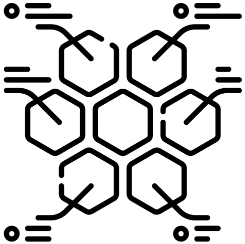
Kernel data with **known** structures

- Kernel code



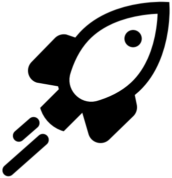
Kernel data with **known** structures

- Kernel code
- Page tables

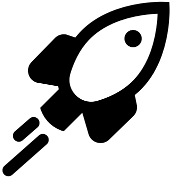


Kernel data with **known** structures

- Kernel code
- Page tables
- Task structs



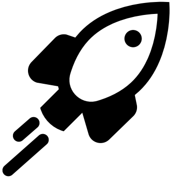
Performance impact caused by MAC computation



Performance impact caused by MAC computation

- Transient execution with unverified data

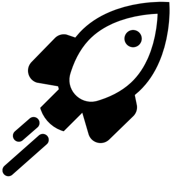
CSI:Rowhammer – Performance Improvements



Performance impact caused by MAC computation

- Transient execution with unverified data
- Can be made **secure** [7, 5]

CSI:Rowhammer – Performance Improvements



Performance impact caused by MAC computation

- Transient execution with unverified data
- Can be made **secure** [7, 5]

→ No performance impact

Final Remarks

- Rowhammer mitigations assuming characteristics are broken.

Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.

Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
- **Open Source**

  https://github.com/IAIK/halfdouble

  https://github.com/IAIK/csirowhammer

Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
- **Open Source**

  `https://github.com/IAIK/halfdouble`

  `https://github.com/IAIK/csirowhammer`
- **Many more details** in the papers

Final Remarks

- Rowhammer mitigations assuming characteristics are broken.
- Software-based correction opens up a lot of flexibility.
- **Open Source**

  `https://github.com/IAIK/halfdouble`

  `https://github.com/IAIK/csirowhammer`
- **Many more details** in the papers

Half-Double Rowhammer

CSI:Rowhammer

[1] Roberto Avanzi. The QARMA Block Cipher Family: Almost MDS Matrices Over Rings With Zero Divisors, Nearly Symmetric Even-Mansour Constructions With Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes. In: IACR Transactions on Symmetric Cryptology 2017.1 (2017), pp. 4–44.

[2] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. Another Flip in the Wall of Rowhammer Defenses. In: S&P. 2018.

[3] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, D607onghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In: ACM SIGARCH Computer Architecture News 42.3 (2014), pp. 361–372.

[4] **Andreas Kogler**, Jonas Juffinger, Salman Qazi, Yoongu Kim, Moritz Lipp, Nicolas Boichat, Eric Shiu, Mattias Nissler, and Daniel Gruss. Half-Double: Hammering From the Next Row Over. In: In major revision at USENIX Security Symposium. 2022.

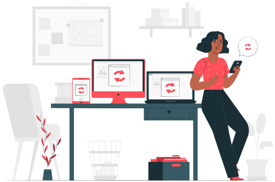
[5] Tamara Silbergleit Lehman, Andrew D. Hilton, and Benjamin C. Lee. PoisonIvy: Safe speculation for secure memory. In: MICRO. 2016.

[6] Mark Seaborn and Thomas Dullien. Test DRAM for bit flips caused by the rowhammer problem. 2015. URL: https://github.com/google/rowhammer-test.

[7] Weidong Shi and Hsien-Hsin S. Lee. Authentication Control Point and Its Implications For Secure Processor Design. In: MICRO. 2006.

Additonal Slides

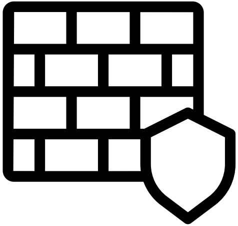
Affected Devices



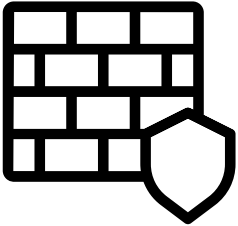
- Tested **13** DIMMs & devices
- **2** DIMMs affected
  - FPGA analysis
  - Exact numbers
- **5** out of **7** mobile devices affected
  - Reversed addressing
  - Unprivileged flush
  - Uncachable memory (10x)

Affected Devices - Flip Numbers

System	RAM	$N_{Hammers}$	$UC_{0 \to 1}$	$UC_{1 \to 0}$	$Flush_{0 \to 1}$	$Flush_{1 \to 0}$
$Chromebook_1$	LPDDR4x	23 274	27	40	2	5
$Chromebook_2$	LPDDR4x	23 586	235	2379	12	101
OnePlus 5T	LPDDR4x	25 687	2	30	1	24
Pixel 3	LPDDR4x	32 921	11	5	0	0
HTC U11	LPDDR4x	21 840	-	-	3	17

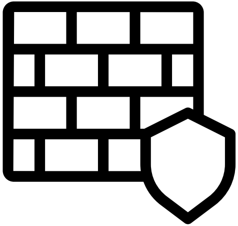


Protect exception handler against bit flips



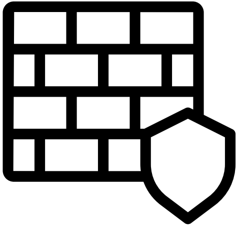
Protect exception handler against bit flips

- 1 Page IDT



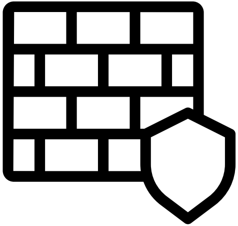
Protect exception handler against bit flips

- 1 Page IDT
- 1 Page GDT



Protect exception handler against bit flips

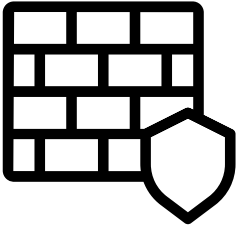
- 1 Page IDT
- 1 Page GDT
- 2 Pages exception handler + correction



Protect exception handler against bit flips

- 1 Page IDT
- 1 Page GDT
- 2 Pages exception handler + correction

→ Page table entries unprotected



Protect exception handler against bit flips

- 1 Page IDT
- 1 Page GDT
- 2 Pages exception handler + correction

→ Page table entries unprotected

- 4 lockable TLB entries

Two conditions

1. Bit must always be in register
2. Bit must be unique for every process

→ **CR3** register

CSI:Rowhammer – Nesting Detection

```
corruption_exception_handler() {
  if (has_nested_bit_set(CR3)) {
    enable_interrupts();
    error_correction_as_a_search(corruption_address);
  } else {
    set_nested_bit(CR3);
    enable_interrupts();
    advanced_error_correction(corruption_address);
    clear_nested_bit(CR3);
  }
}
```

1. Out of Order CPU core
2. MAC computation and check
3. New instructions
4. Modified Linux performing the correction as a search