

PUF the Magic DRAmGON:

Persistent Storage in Volatile Memory using Rowhammer PUF

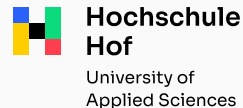
Emiliia Gelóczy^{1,*}, Martin Heckel^{2,*},
Stefan Katzenbeisser¹, and Florian Adamsky²

September 16, 2026

¹ University of Passau

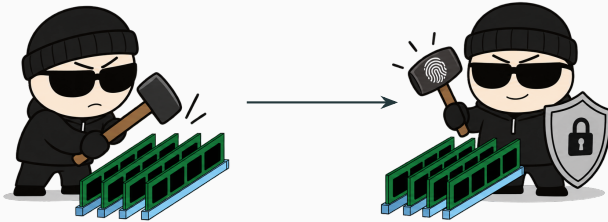
² Hof University of Applied Sciences

* These authors contributed equally to this work.

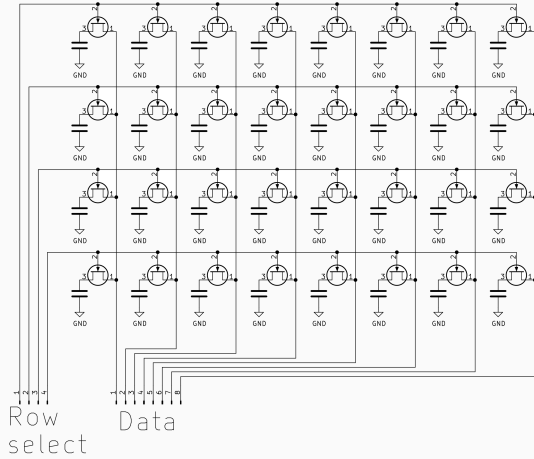


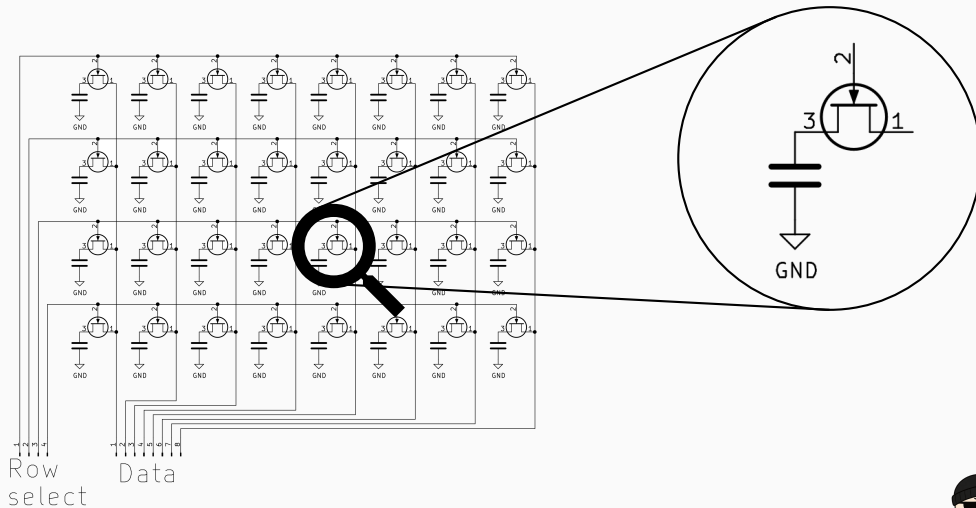




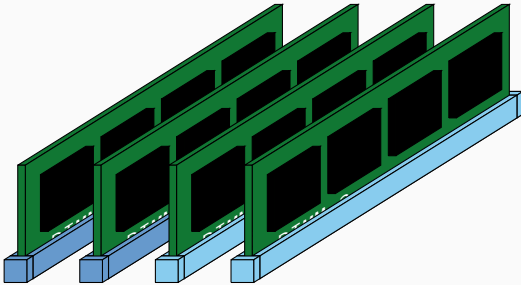




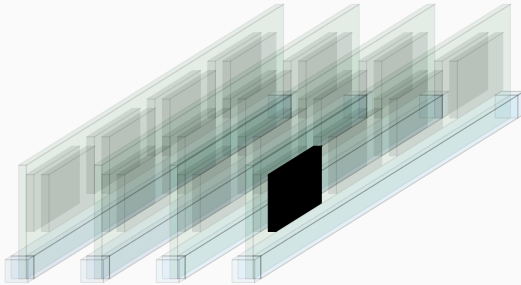


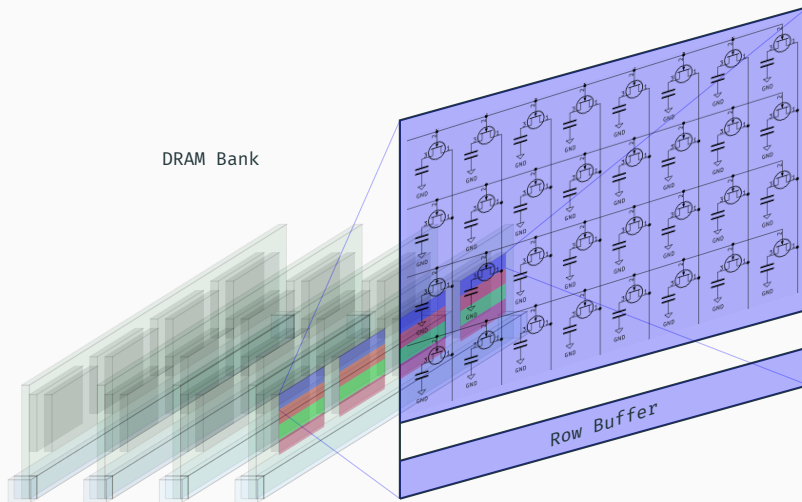


System DRAM

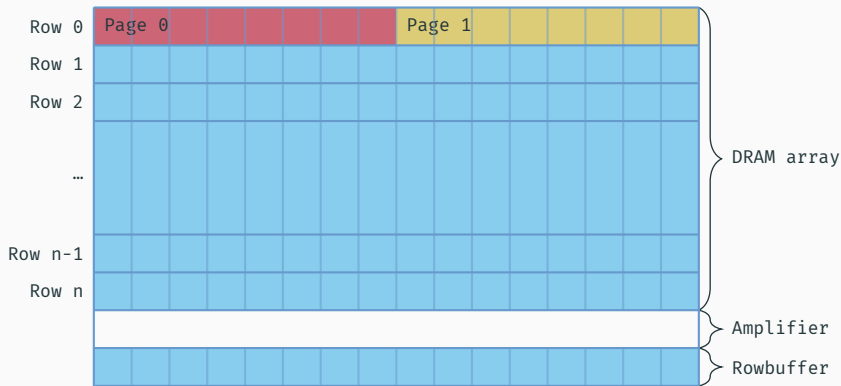


DRAM Chip





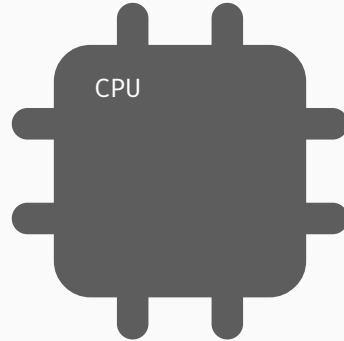
Structure within a DRAM Bank



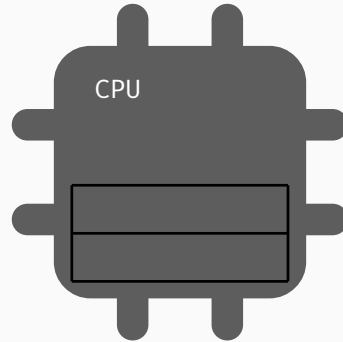
This example shows the Sandy Bridge microarchitecture



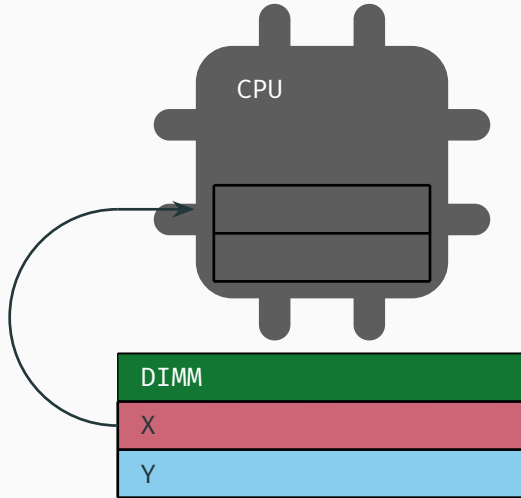
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



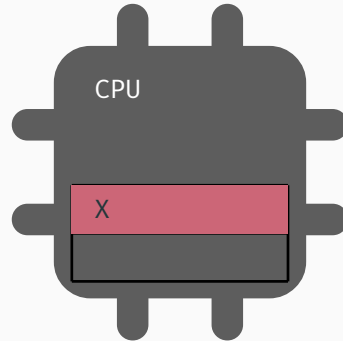
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



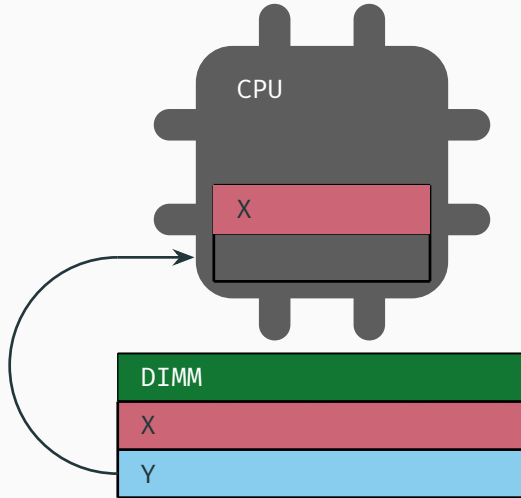
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



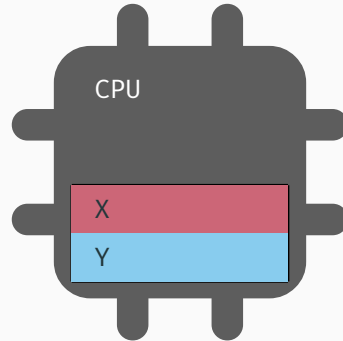
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



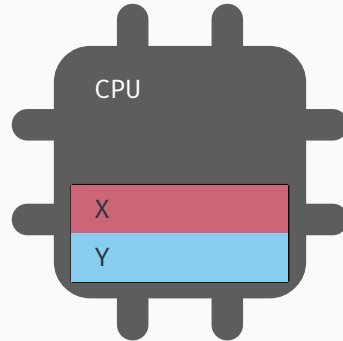
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



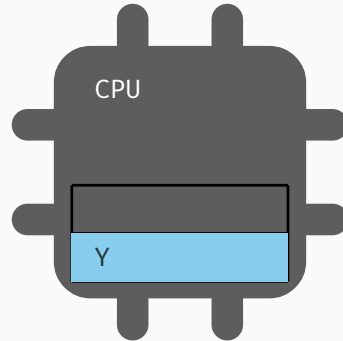
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



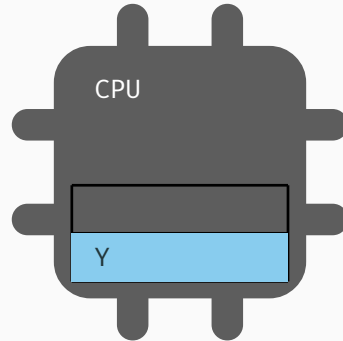
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```



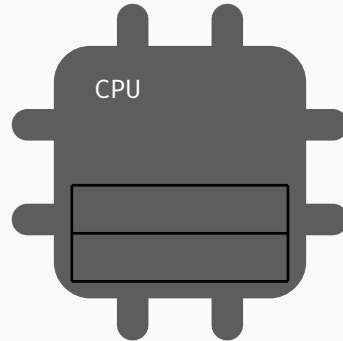
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```

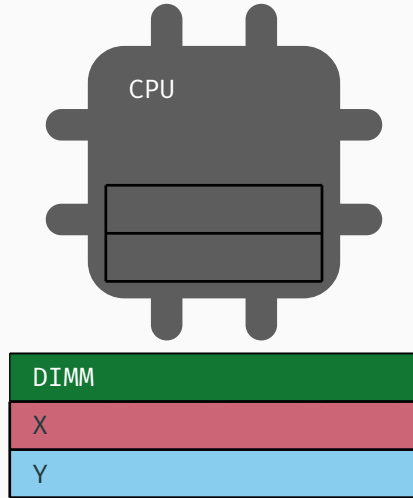


```
1  hammer:  
2    mov eax, X  
3    mov ebx, Y  
4    clflush X  
5    clflush Y  
6    jmp hammer
```



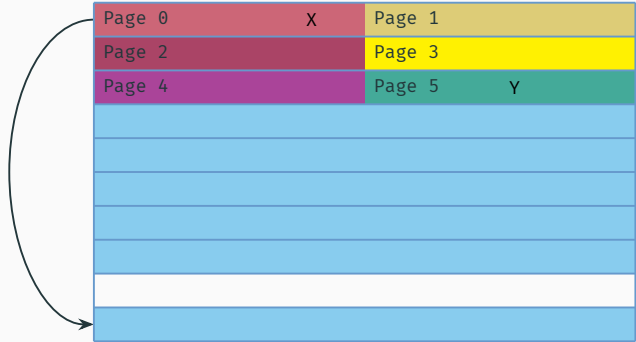
```
1  hammer:  
2  mov eax, X  
3  mov ebx, Y  
4  clflush X  
5  clflush Y  
6  jmp hammer
```

Source code from Kim et al. [1]




```

1  hammer:
2  mov eax, X
3  mov ebx, Y
4  clflush X
5  clflush Y
6  jmp hammer
    
```

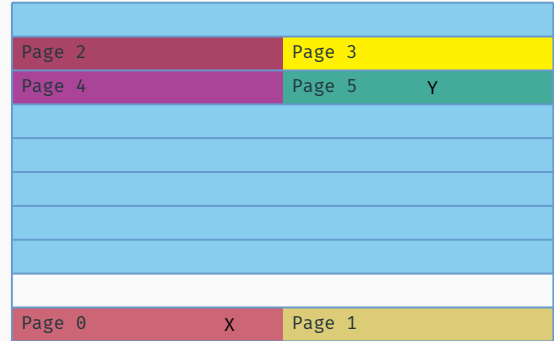


Source code from Kim et al. [1]



```

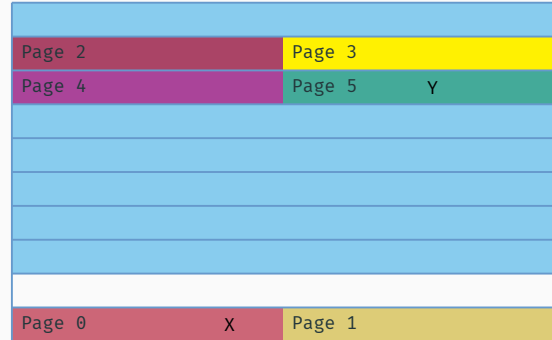
1  hammer:
2  mov eax, X
3  mov ebx, Y
4  clflush X
5  clflush Y
6  jmp hammer
    
```



Source code from Kim et al. [1]



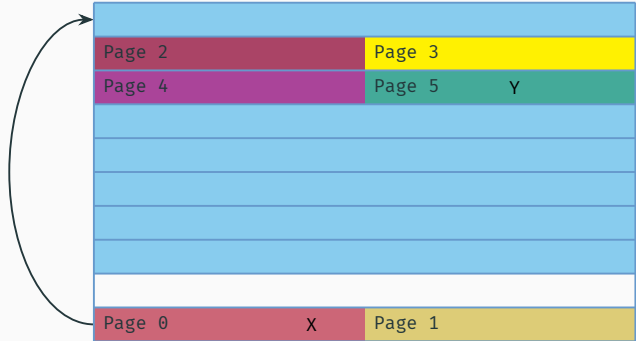
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]



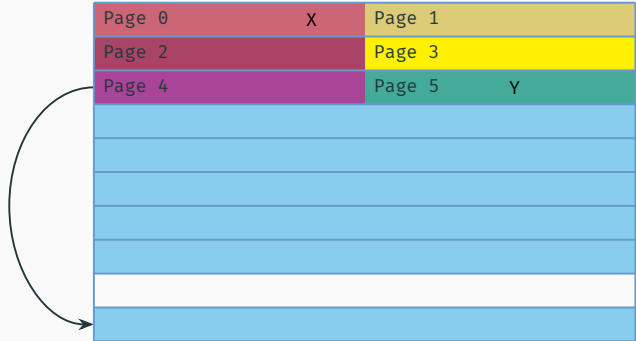
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]



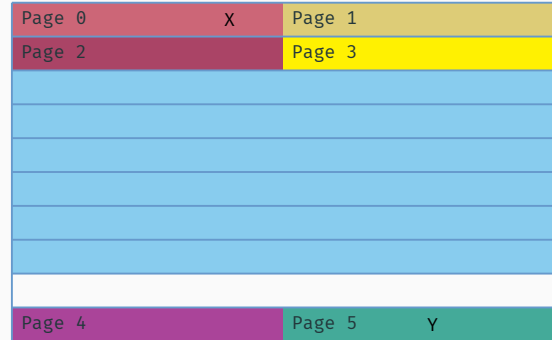
```
1  hammer:  
2    mov eax, X  
3    mov ebx, Y  
4    clflush X  
5    clflush Y  
6    jmp hammer
```



Source code from Kim et al. [1]



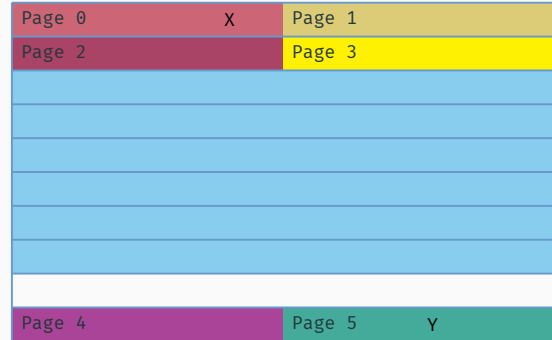
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]



```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



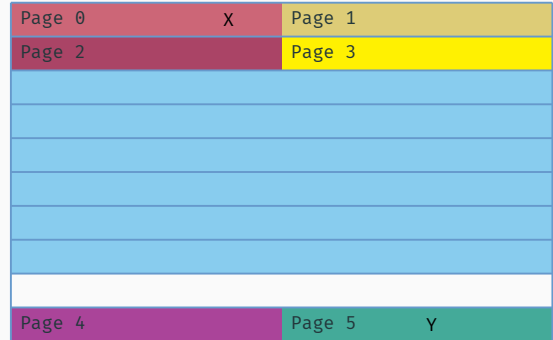
Source code from Kim et al. [1]



```

1  hammer:
2      mov eax, X
3      mov ebx, Y
4      cflush X
5      cflush Y
6      jmp hammer

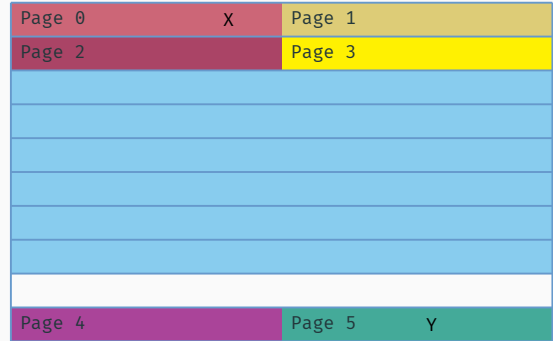
```



Source code from Kim et al. [1]



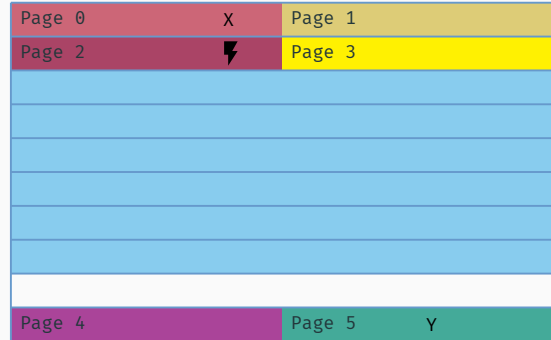
```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]



```
1  hammer:
2    mov eax, X
3    mov ebx, Y
4    clflush X
5    clflush Y
6    jmp hammer
```



Source code from Kim et al. [1]



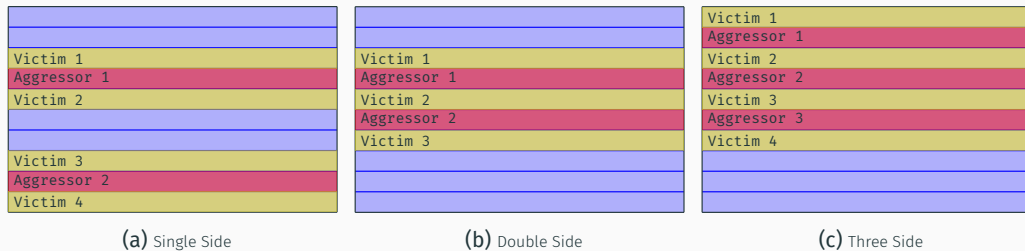


Figure 1: Examples of Rowhammer Patterns



Offensive technique?

Victim 1
Aggressor
Victim 2
Victim 3
Aggressor
Victim 4

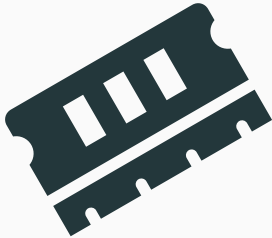


Defensive technique!

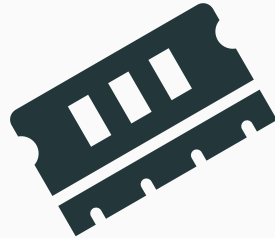
Victim 1
Aggressor
Victim 2
Victim 3
Aggressor
Victim 4



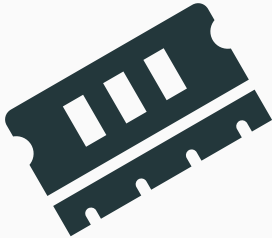
Device 1



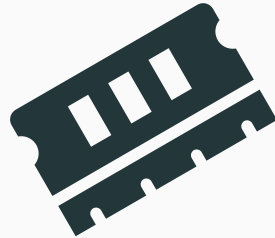
Device 2



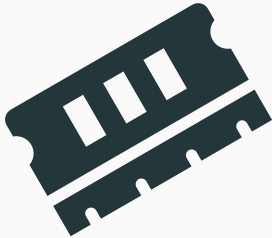
Device 1



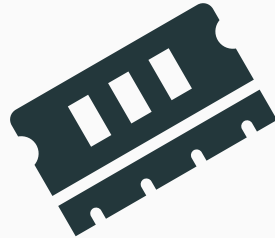
Device 2



Device 1



Device 2



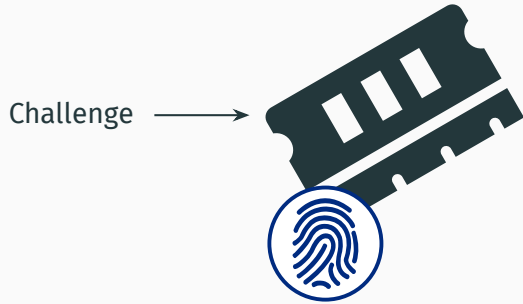
Device 1



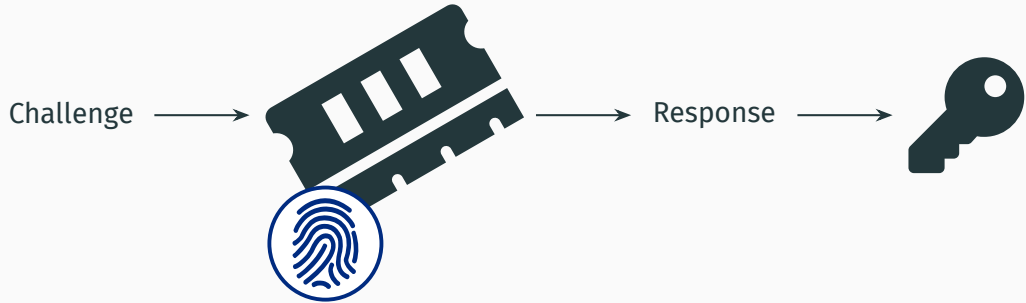
Device 2

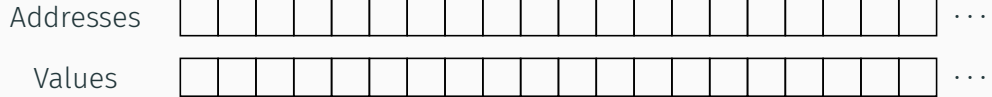












Addresses



Values



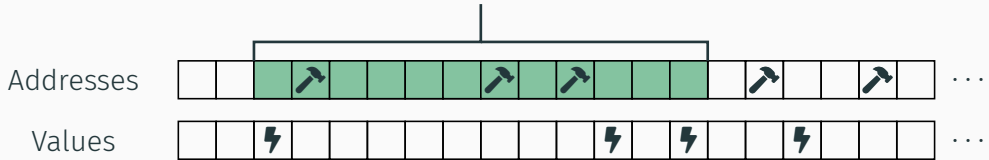
Addresses

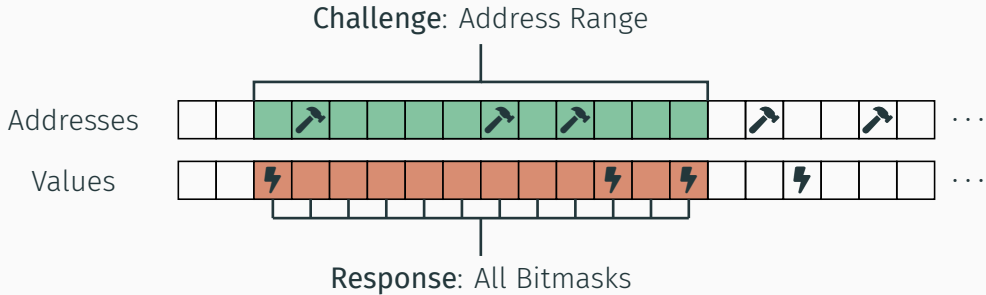


Values



Challenge: Address Range

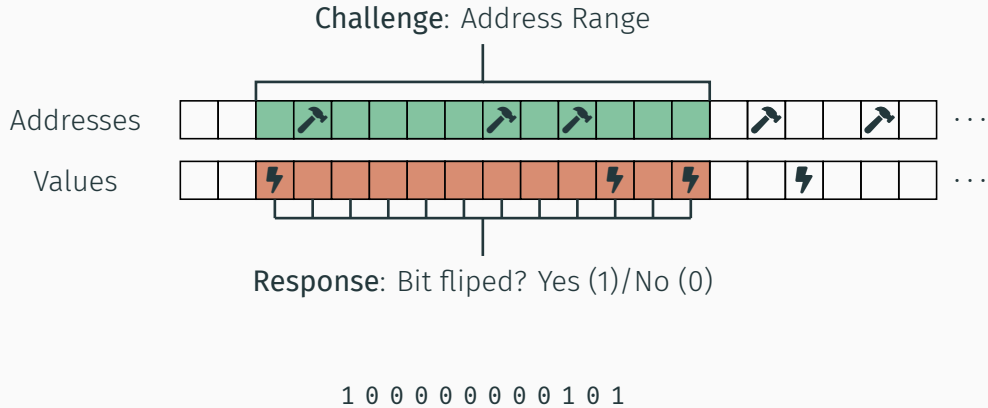




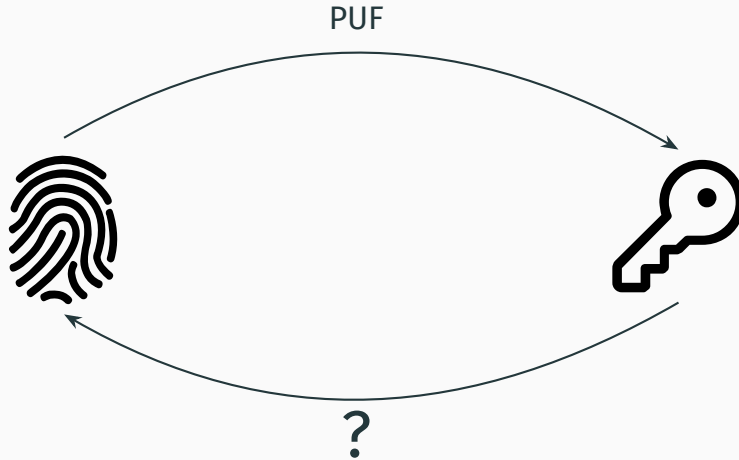
```

01000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00010000 00000000 00000100
    
```









Key to encode:

1	0	0	1
---	---	---	---



Key to encode:

1	0	0	1
---	---	---	---

Memory:

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8	6
---	---



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8	6
---	---



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8	6	0
---	---	---



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8	6	0
---	---	---



Key to encode:

1	0	0	1
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Challenge:

8	6	0	2
---	---	---	---



Challenge:

8	6	0	2
---	---	---	---



Challenge:

8	6	0	2
---	---	---	---

Memory:

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:

1



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:

1



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡					⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	---	--	--

Decoded Key:

1	0
---	---



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:

1	0
---	---



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:

1	0	0
---	---	---



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:

1	0	0
---	---	---



Challenge:

8	6	0	2
---	---	---	---

Memory:

		⚡						⚡			⚡						⚡		
--	--	---	--	--	--	--	--	---	--	--	---	--	--	--	--	--	---	--	--

Decoded Key:

1	0	0	1
---	---	---	---



- Bit flips are sparse, offset distribution could leak information on the key



- Bit flips are sparse, offset distribution could leak information on the key
 - ↳ Select no-flip addresses following the distribution of flip addresses (Monte Carlo Simulation)



- Bit flips are sparse, offset distribution could leak information on the key
 - ↳ Select no-flip addresses following the distribution of flip addresses (Monte Carlo Simulation)
- Stability of bit flips over multiple runs



- Bit flips are sparse, offset distribution could leak information on the key
 - ↳ Select no-flip addresses following the distribution of flip addresses (Monte Carlo Simulation)
- Stability of bit flips over multiple runs
 - ↳ Perform multiple runs and select most stable locations



- Bit flips are sparse, offset distribution could leak information on the key
 - ↳ Select no-flip addresses following the distribution of flip addresses (Monte Carlo Simulation)
- Stability of bit flips over multiple runs
 - ↳ Perform multiple runs and select most stable locations
- Required time to trigger bit flips (38.54 min for a full sweep)

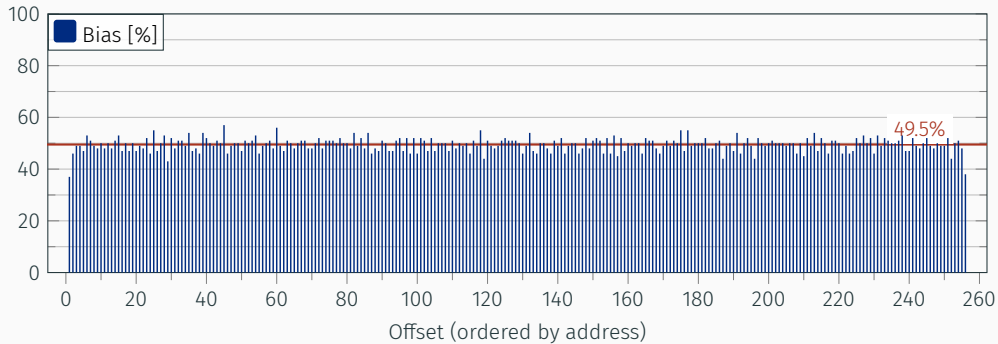


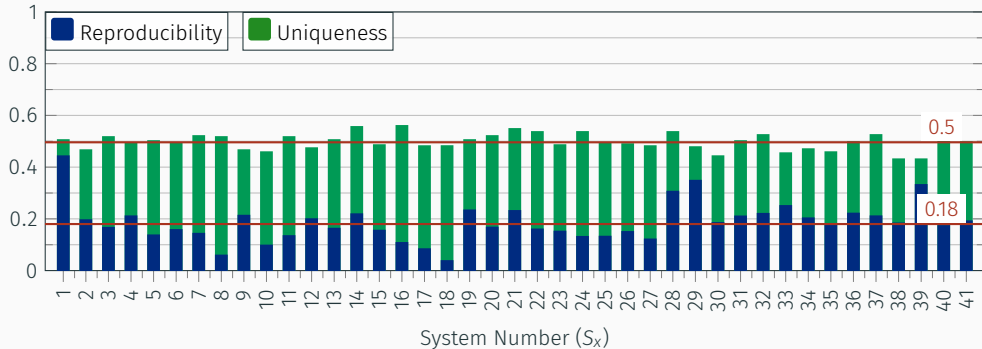
- Bit flips are sparse, offset distribution could leak information on the key
 - ↳ Select no-flip addresses following the distribution of flip addresses (Monte Carlo Simulation)
- Stability of bit flips over multiple runs
 - ↳ Perform multiple runs and select most stable locations
- Required time to trigger bit flips (38.54 min for a full sweep)
 - ↳ In theory: Disable refreshes for specific memory ranges used to encode keys (not done in this paper)



DRaMGoN - Experimental Evaluation









DRAmGON is:

- The **first** approach to use **volatile memory as permanent storage** based on PUF
- Based on the **not typical** application of Rowhammer as defence mechanism



DRAMGON is:

- The **first** approach to use **volatile memory as permanent storage** based on PUF
- Based on the **not typical** application of Rowhammer as defence mechanism

Apart from it, we also:

- Introduce **5 novel** RH-PUF constructions
- Show that most of the tested systems are susceptible to Rowhammer

PUF the Magic DRAmGON:

Persistent Storage in Volatile Memory using Rowhammer PUF

Emiliia Gelóczy^{1,*}, Martin Heckel^{2,*},
Stefan Katzenbeisser¹, and Florian Adamsky²

September 16, 2026

¹ University of Passau

² Hof University of Applied Sciences

* These authors contributed equally to this work.



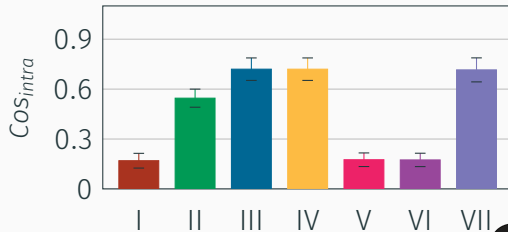
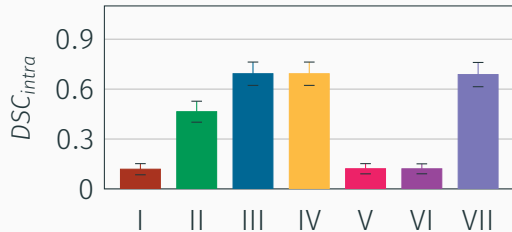
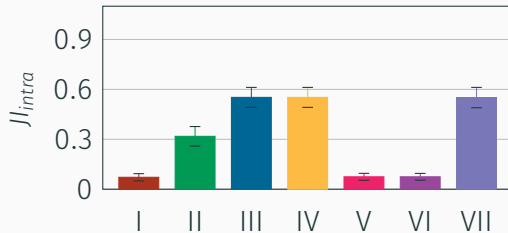
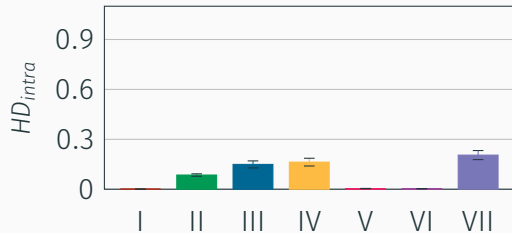
 Emiliia Gelóczy



 Martin Heckel

- [1] Yoongu Kim et al. “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors”. In: *ISCA*. 2014.
- [2] André Schaller et al. “Intrinsic rowhammer PUFs: Leveraging the rowhammer effect for improved security”. In: *HOST*. 2017.

Evaluation of RH-PUF Constructions: Reliability



Evaluation of RH-PUF Constructions: Uniqueness, Uniformity, Entropy

