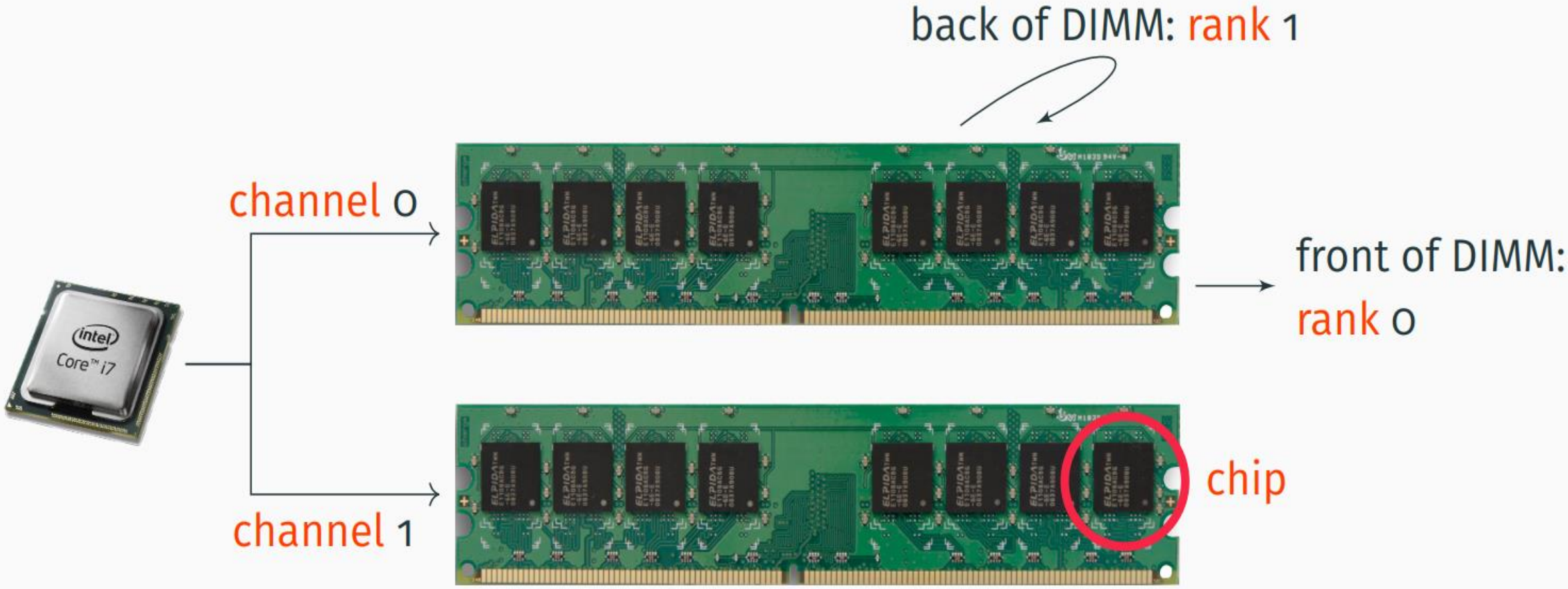
ATTACKS AT DRAM
CS665 - 2019

DIMM


back of DIMM: rank 1
channel 0
front of DIMM:
rank 0
chip
channel 1

Chip – Banks - Rows

chip

bank 0

row 0
row 1
row 2
...
row 32767

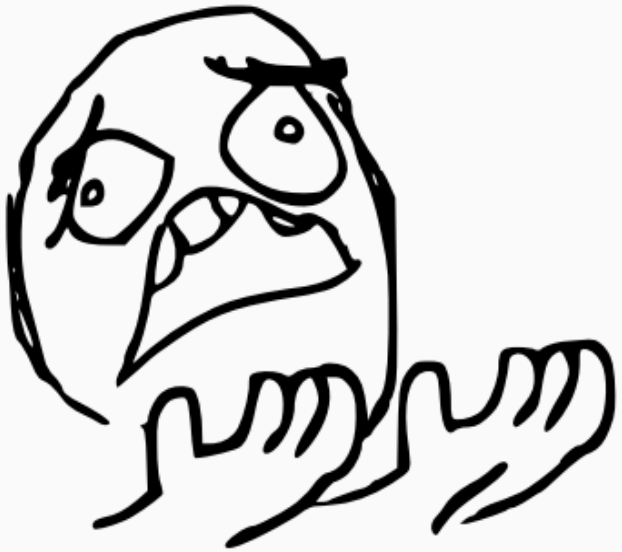
row buffer

64k cells
1 capacitor,
1 transitor each

How to attack ?

- The smallest unit of physical memory is one page
- Pages are usually $4\,\mathrm{kB}$
- DRAM rows are usually $8\,\mathrm{kB}$
- We need the victim's address and our address in the same row

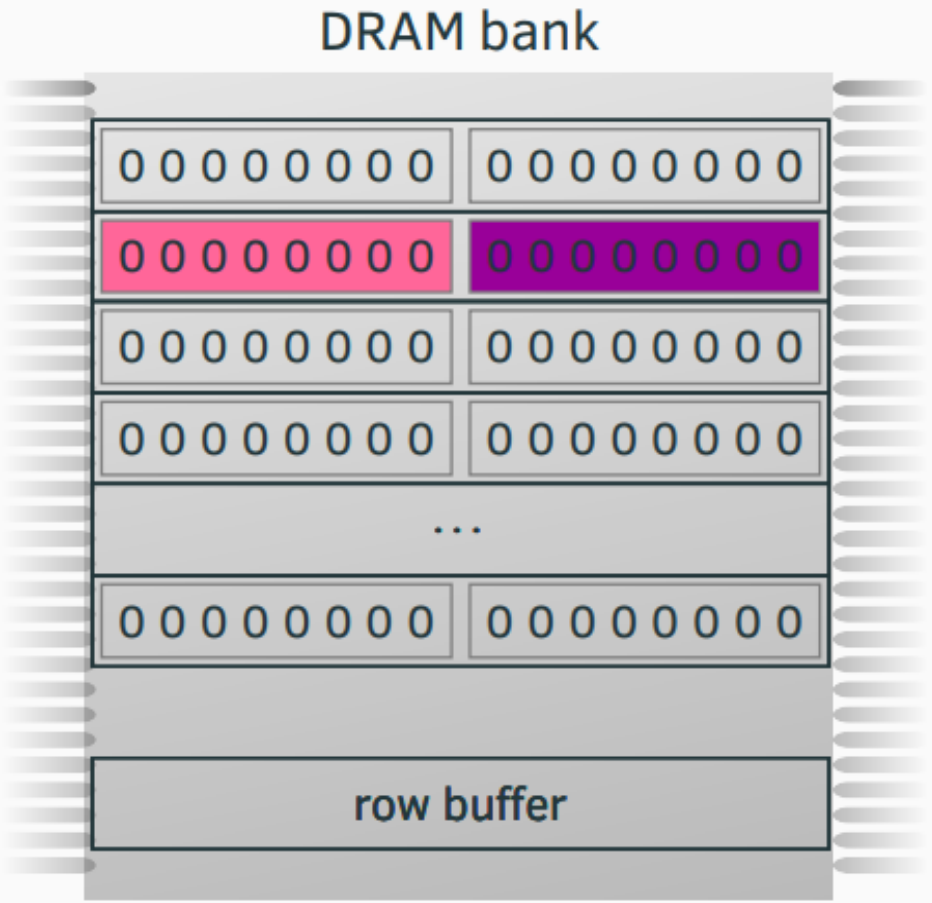
Aha !!



- If you say that **two pages** share one row you are not wrong...
- ...but not right either
- Why?

Shared Row [USENIX sec. ‘16]

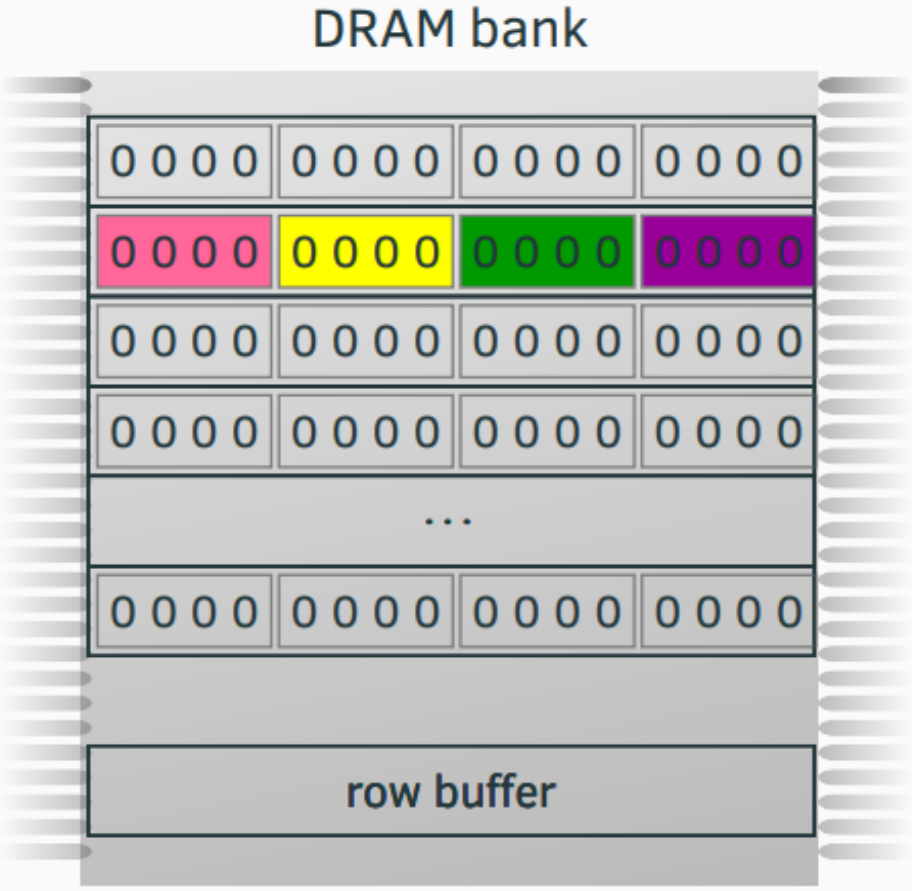
Sandy Bridge /w 1 DIMM



2 pages per row

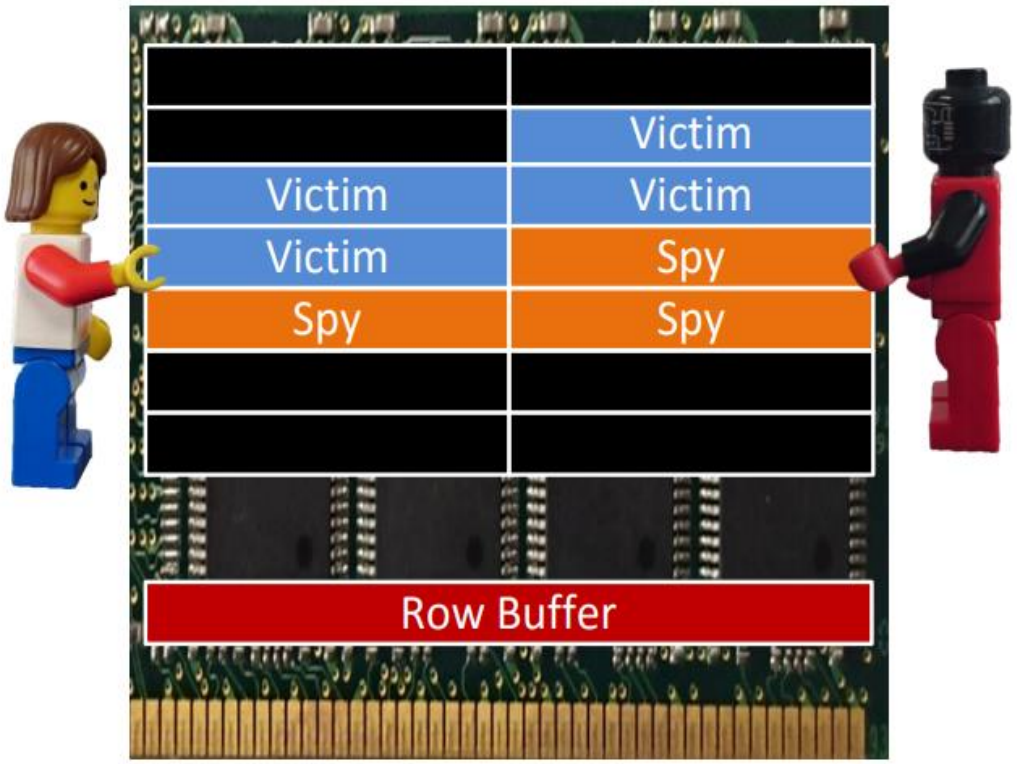
Shared Row [USENIX sec. ‘16]

Ivy Bridge /w 2 DIMM



4 pages per row

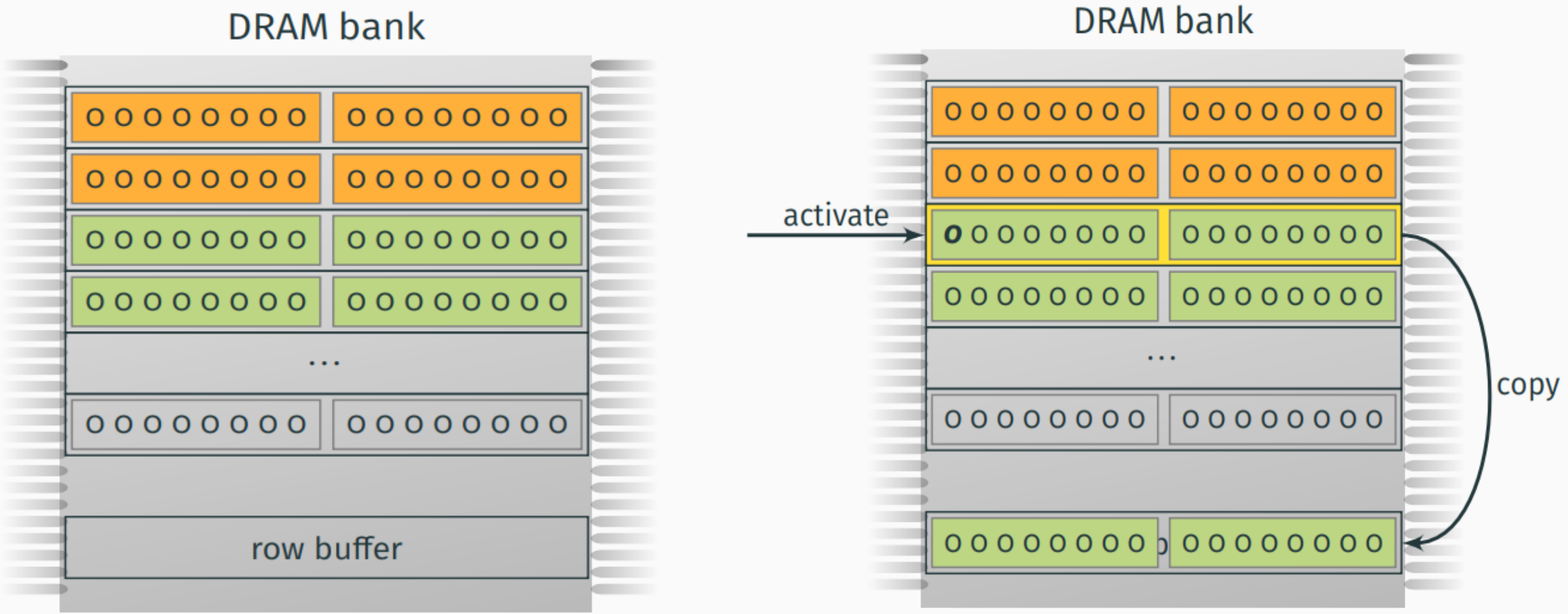
Side-Channel Attack [USENIX sec. '16]



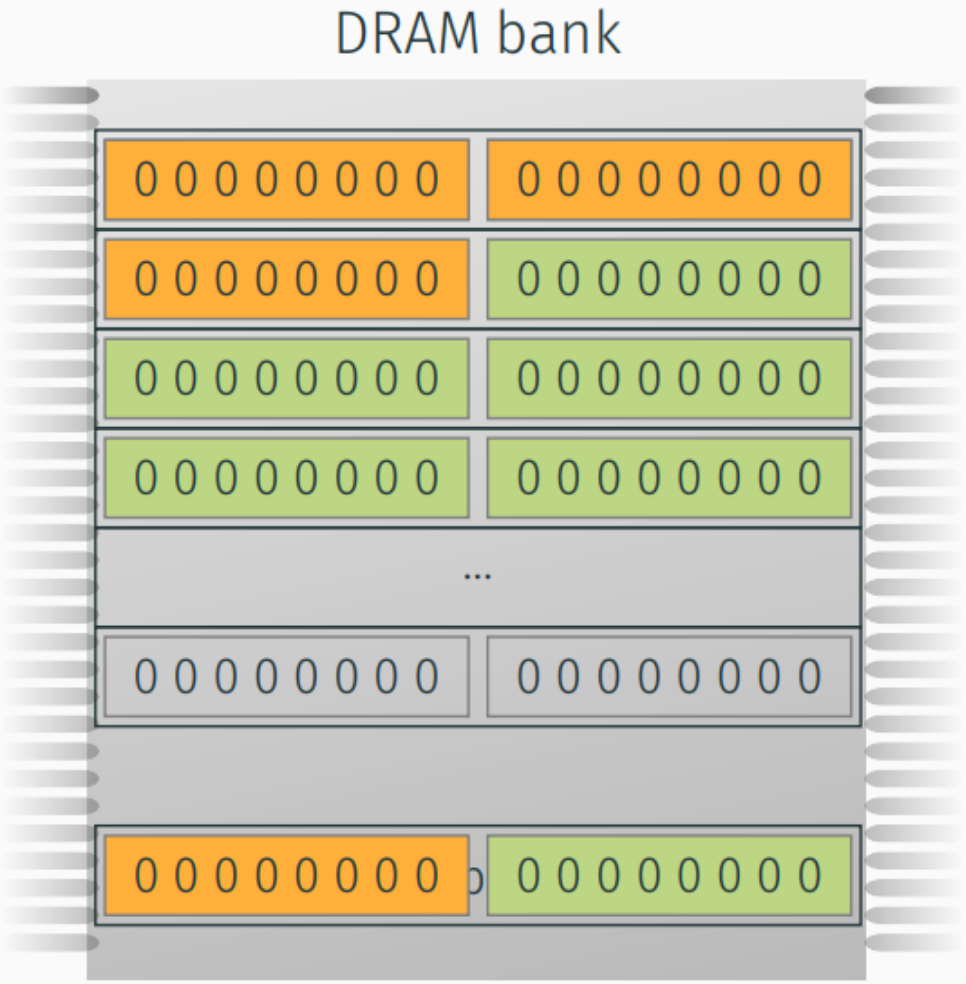
Side-channel attacks

Row-hits: Fast access
Conflicts: Slow access

Row-Buffer



Side-Channel Attack



spy and victim share a row *i*

case #1

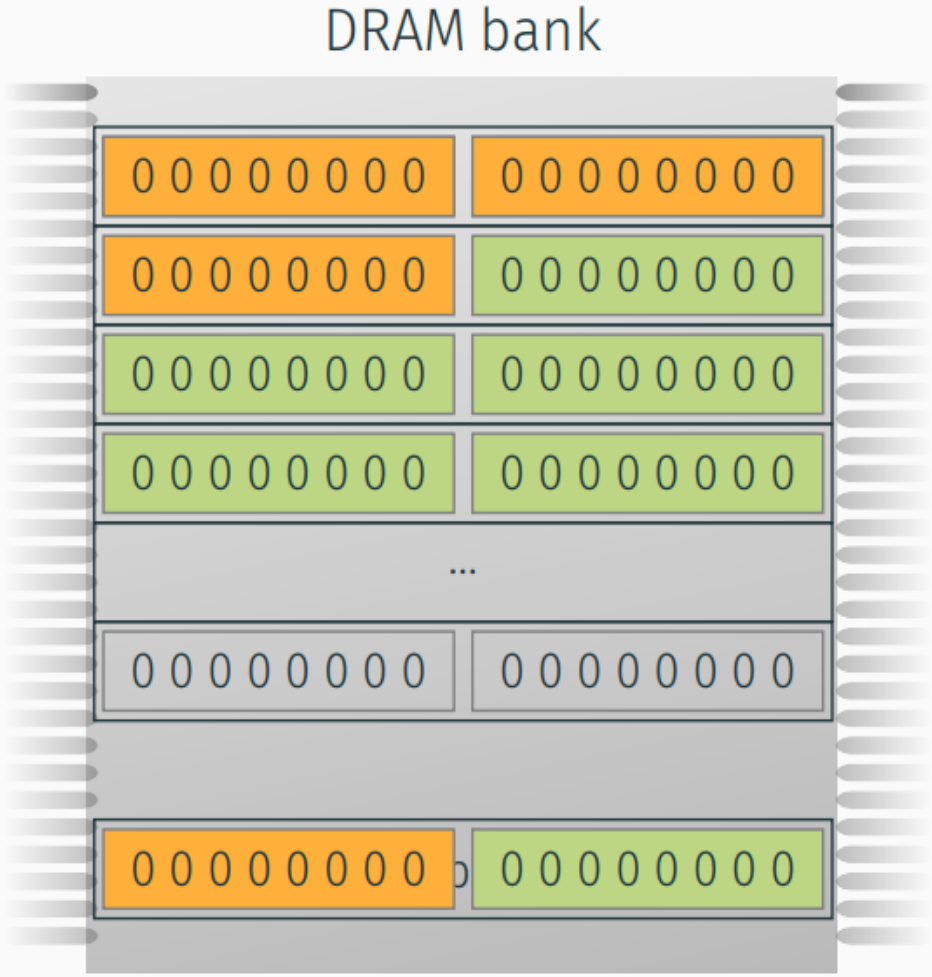
spy accesses row $j \neq i$, copy to row buffer

victim accesses row *i*, copy to row buffer

spy accesses row *i*, no copy

→ **fast**

Side-Channel Attack [USENIX sec. ‘16]



spy and victim share a row *i*

case #2

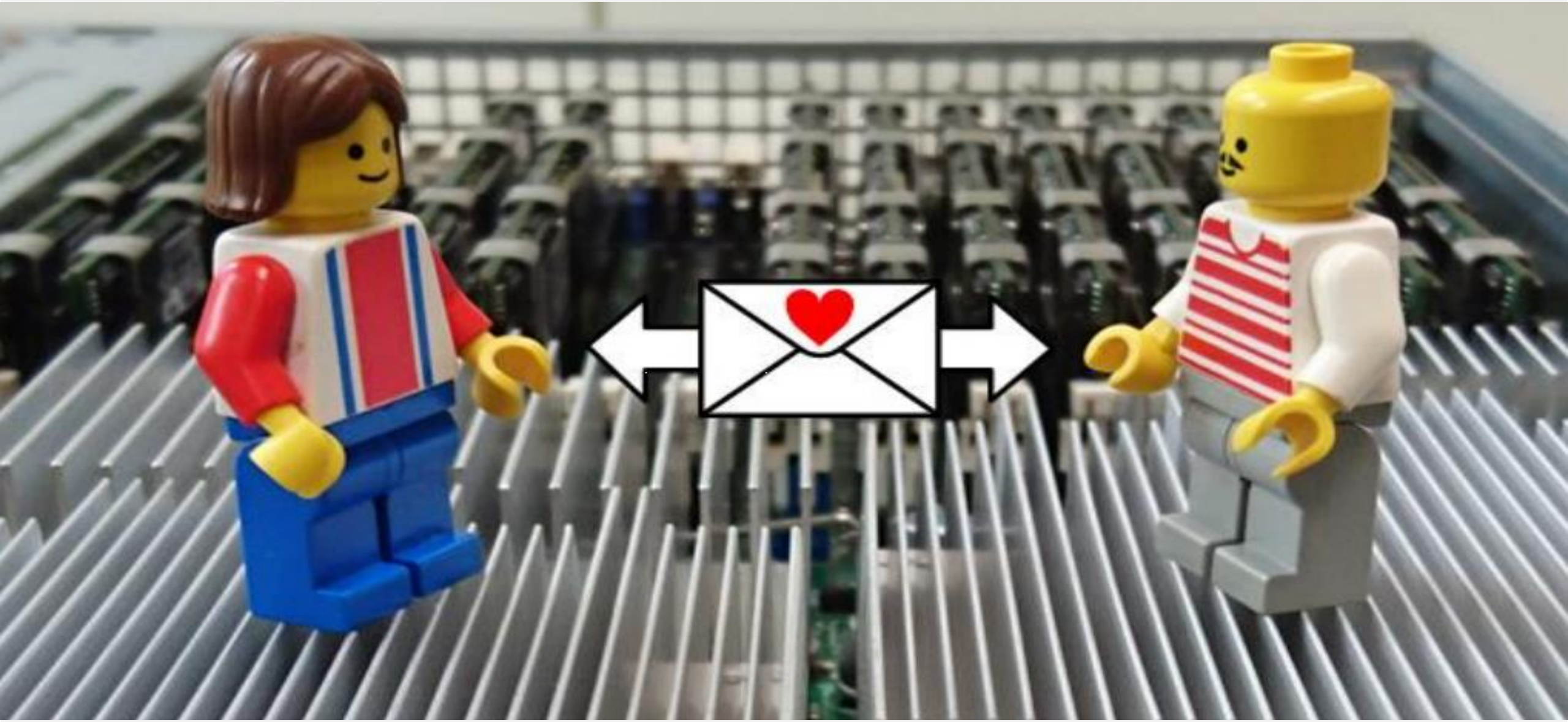
spy accesses row $j \neq i$, copy to row buffer

no victim access on row *i*

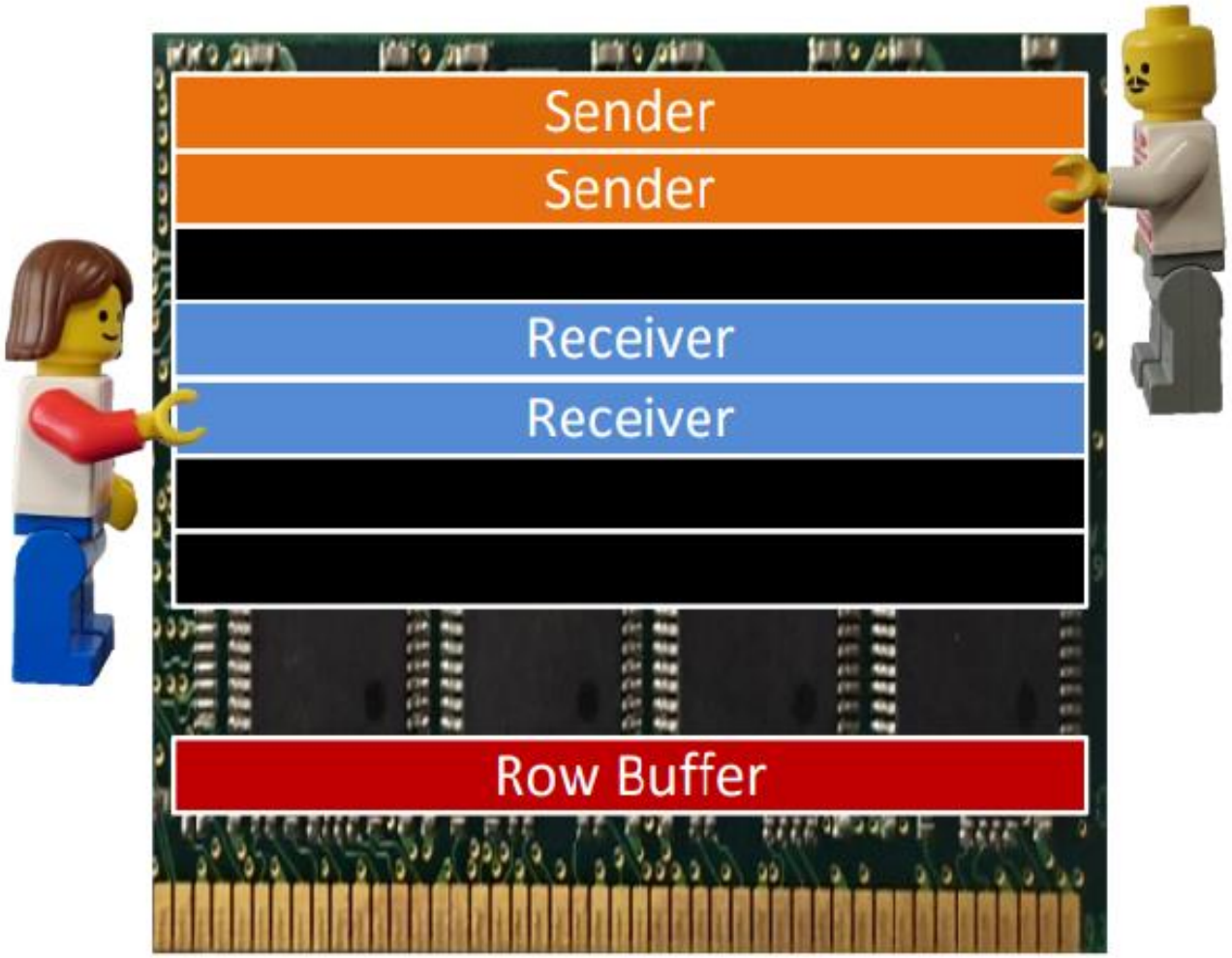
spy accesses row *i*, copy to row buffer

→ **slow**

Covert Channel with Romeo and Juliet

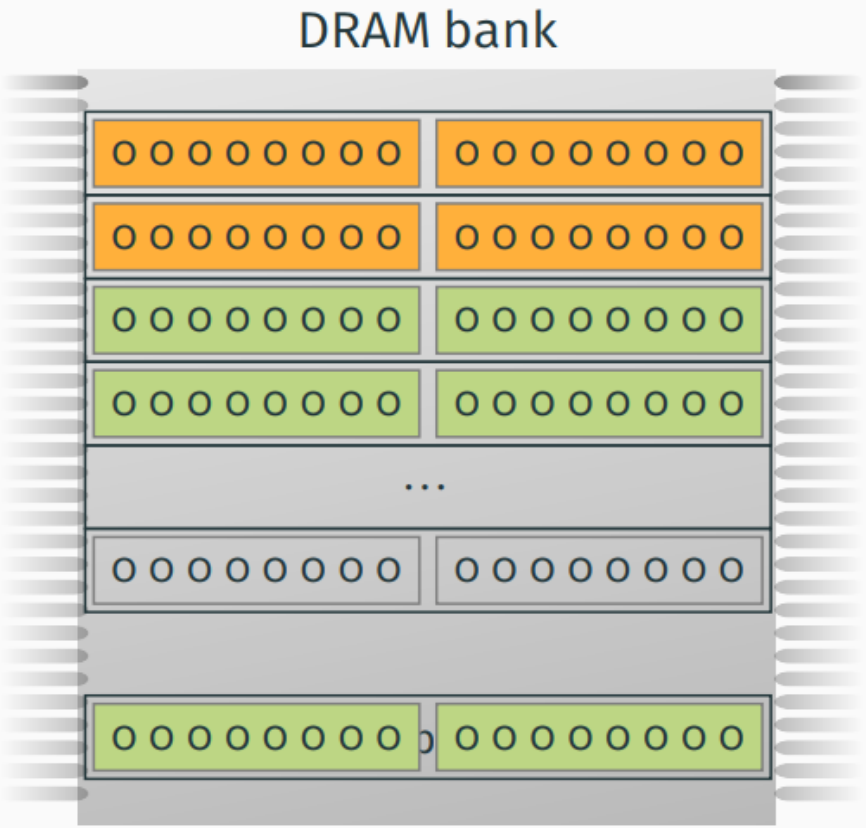


Covert-Channel Attack [USENIX sec. ‘16]



Covert-channel attacks

Covert-Channel Attack [USENIX sec. '16]



sender and receiver agree on one bank

receiver continuously accesses a row *i*

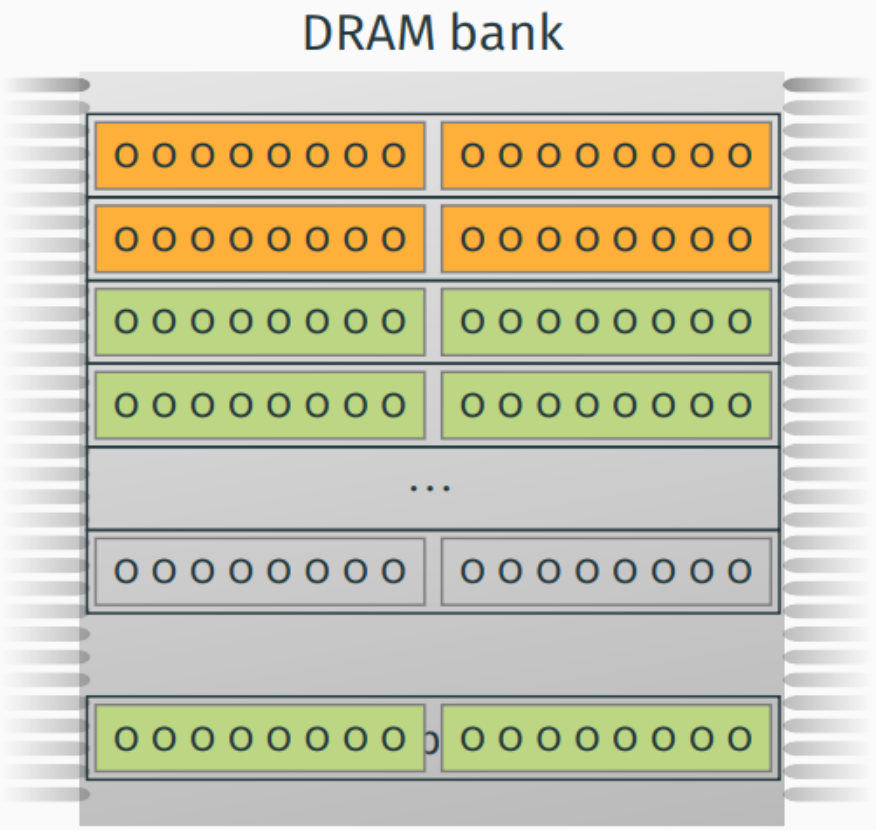
case #1: sender transmits 1

sender accesses row $j \neq i$

next receiver access $\rightarrow$ copy row buffer

$\rightarrow$ **slow**

Covert-Channel Attack [USENIX sec. '16]



sender and receiver agree on one bank

receiver continuously accesses a row *i*

case #2: sender transmits 0

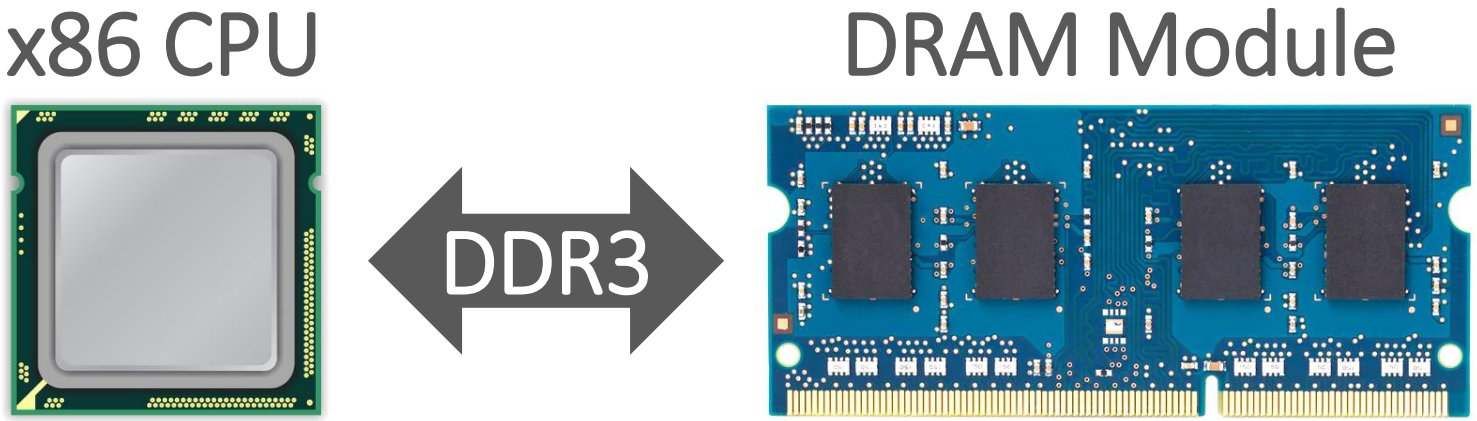
sender does nothing

next receiver access → already in buffer

→ **fast**

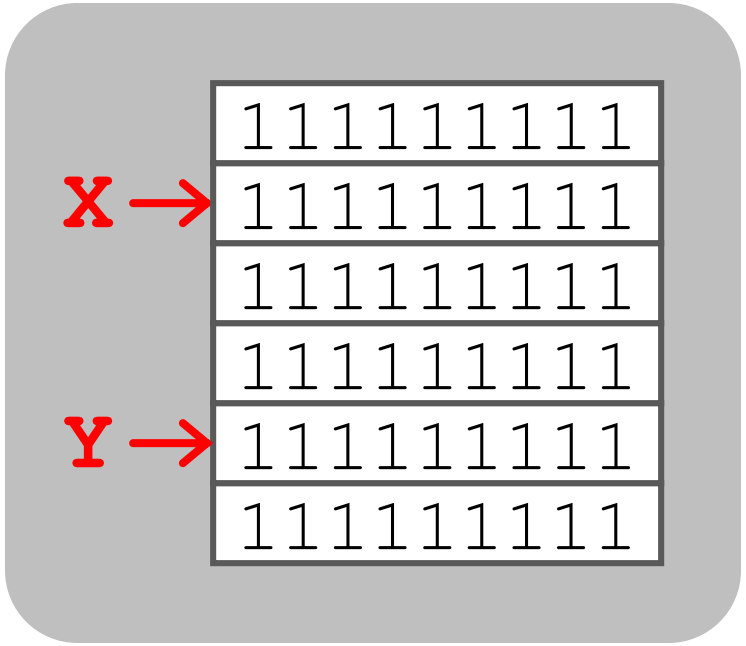
https://github.com/google/rowhammer-test

How to Induce Errors [ISCA '14]



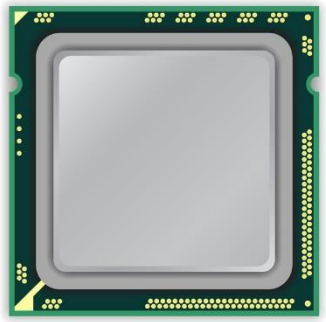
1. Avoid ***cache hits***
   - Flush **X** from cache

2. Avoid ***row hits*** to **X**
   - Read **Y** in another row



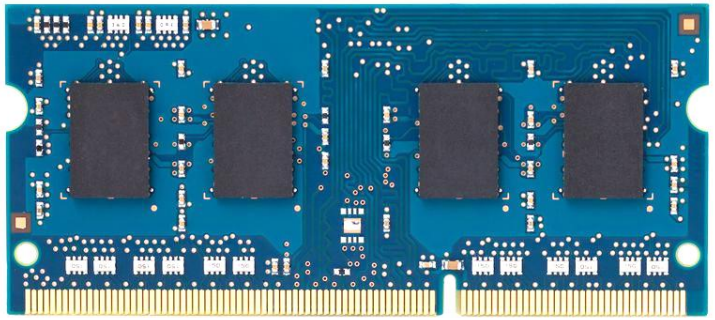
How to Induce Errors

x86 CPU

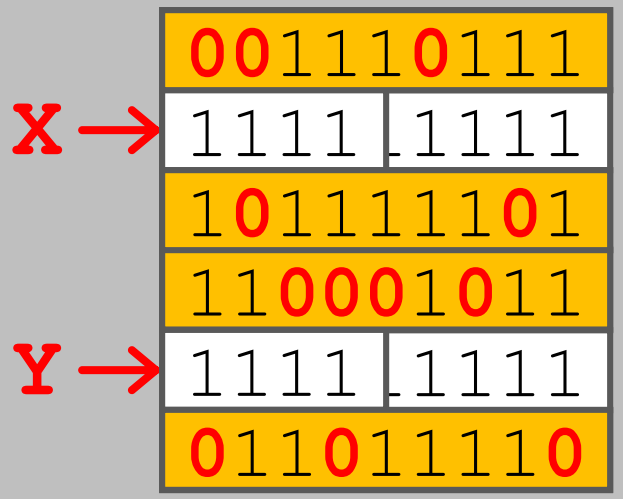


DDR3

DRAM Module



```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



Number of Disturbance Errors

CPU Architecture	Errors	Access-Rate
Intel Haswell (2013)	22.9K	12.3M/sec
Intel Ivy Bridge (2012)	20.7K	11.7M/sec
Intel Sandy Bridge (2011)	16.1K	11.6M/sec
AMD Piledriver (2012)	59	6.1M/sec

- *In a more controlled environment, as many as ten million disturbance errors*
- *Disturbance errors are a serious reliability issue*

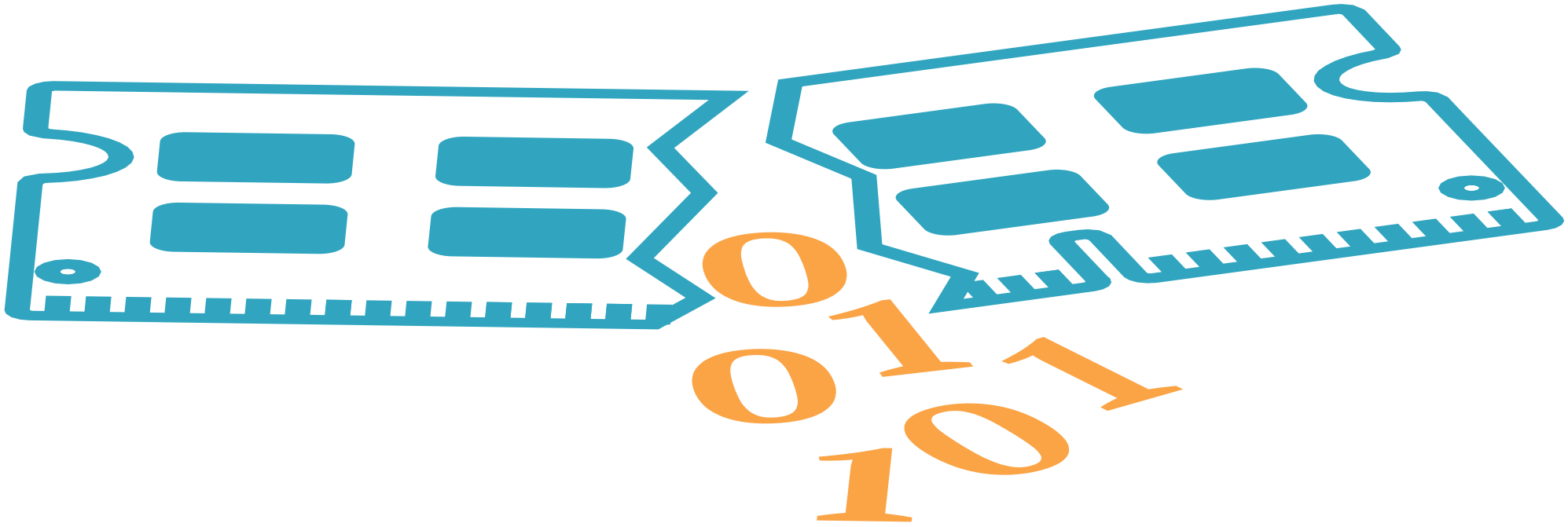
Naive Solutions

1. *Throttle accesses to same row*
   - Limit access-interval: $\geq 500\text{ns}$
   - Limit number of accesses: $\leq 128\text{K}$ $(=64\text{ms}/500\text{ns})$

2. *Refresh more frequently*
   - Shorten refresh-interval by $\sim 7\text{x}$

Both naive solutions introduce significant overhead in performance and power

Reading Assignment: RAM Bleed (https://rambleed.com/)



Reading Bits in Memory Without Accessing Them