HotStorage 2021

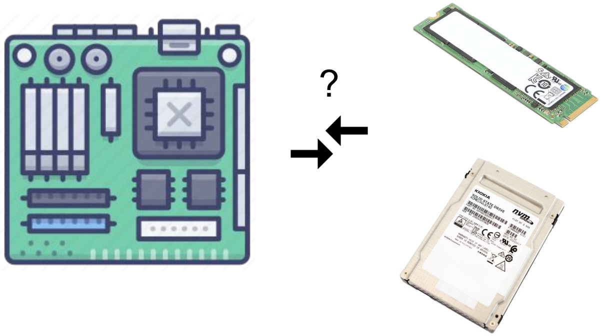
Rowhammering Storage Devices

Tao Zhang, Boris Pismenny, Donald E. Porter
Dan Tsafrir, Aviad Zuck

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL
TECHNION Israel Institute of Technology
vmware®

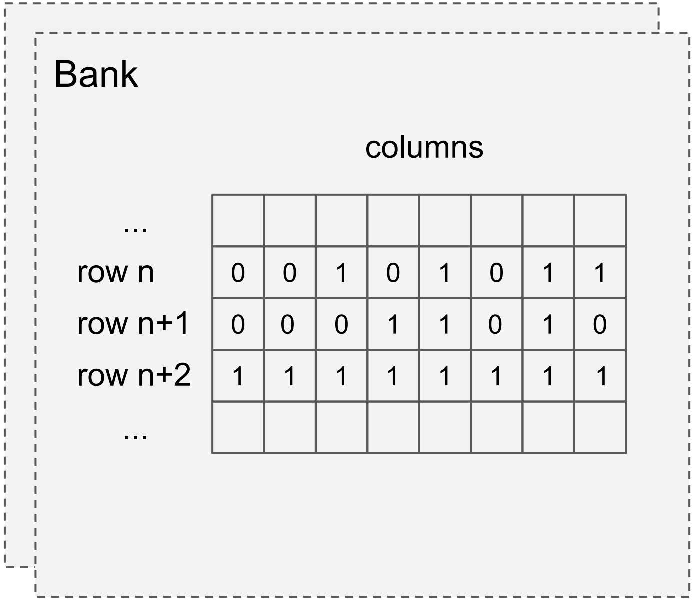
The Story...

Explore the feasibility of rowhammering the DRAM inside an SSD, using ***only standard storage commands***.



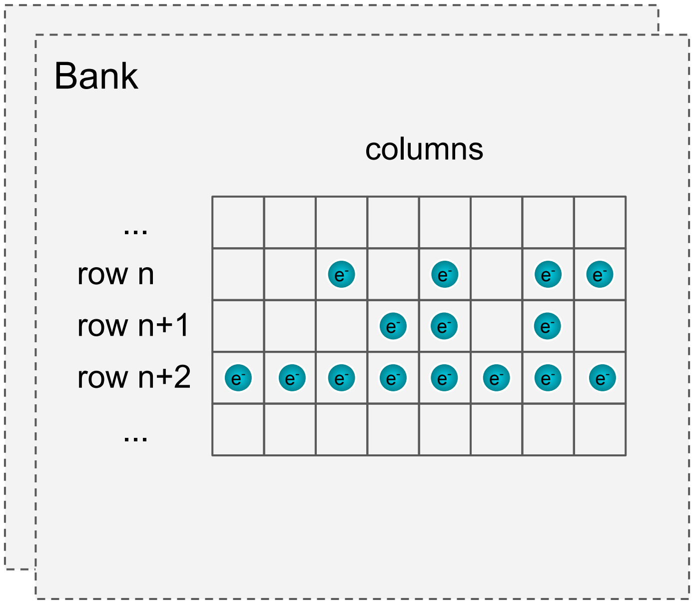
Understand whether the ***small embedded system within the device*** is vulnerable to the same rowhammer attack like a “big” system.

DRAM Overview



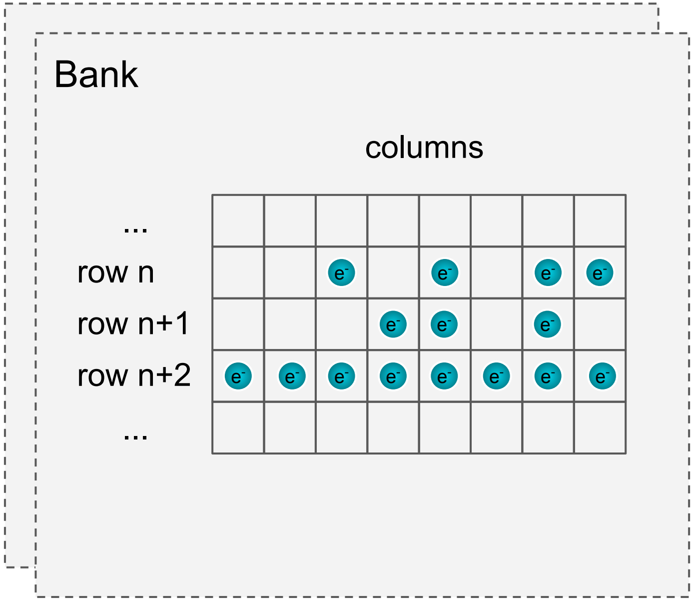
- DRAM cell array
  - Addressed by row/column

DRAM Overview



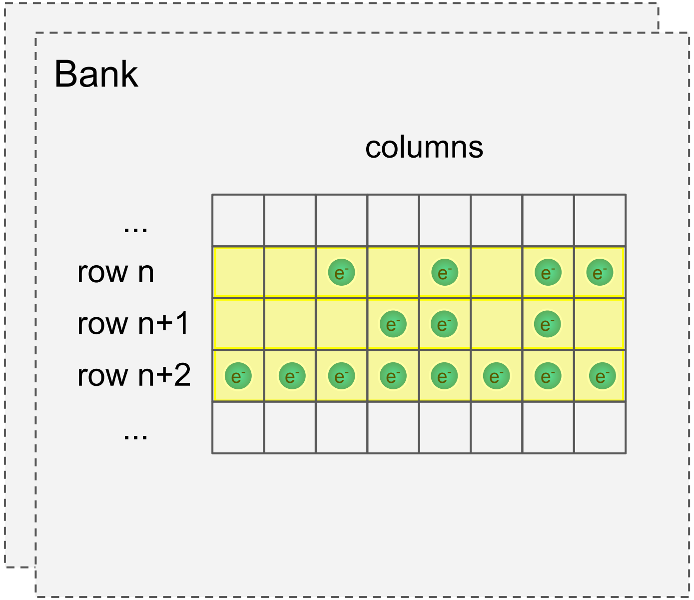
- DRAM cell array
  - Addressed by row/column
  - Bits stored as electric charges

DRAM Overview



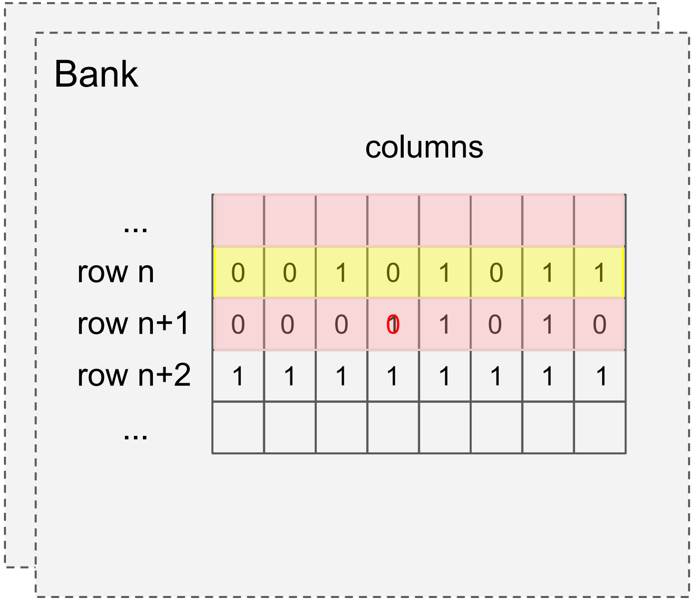
- DRAM cell array
  - Addressed by row/column
  - Bits stored as electric charges
  - Cells lose charge over time

DRAM Overview



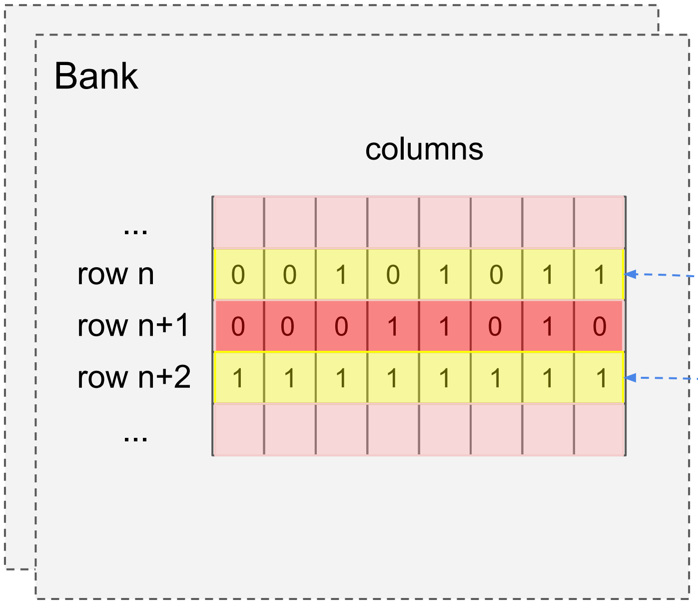
- DRAM cell array
  - Addressed by row/column
  - Bits stored as electric charges
  - Cells lose charge over time
    - Refresh rows in intervals (64ms)

Rowhammer attack



- Disturbance error
  - Aggressor row: Repeatedly open/close
  - Victim(adjacent) row: charge leaks faster
  - Leak faster than refresh, bitflip happens

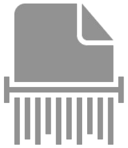
Rowhammer attack



- Disturbance error
  - Aggressor row: Repeatedly open/close
  - Victim(adjacent) row: charge leak faster
  - Leak faster than refresh, bitflip happens
- Double-sided rowhammer

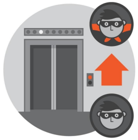
```
loop:
  mov (addr1), %eax
  mov (addr2), %ebx
  clflush (addr1)
  clflush (addr2)
  mfence
  jmp loop
```

Outcomes of Rowhammer Attack



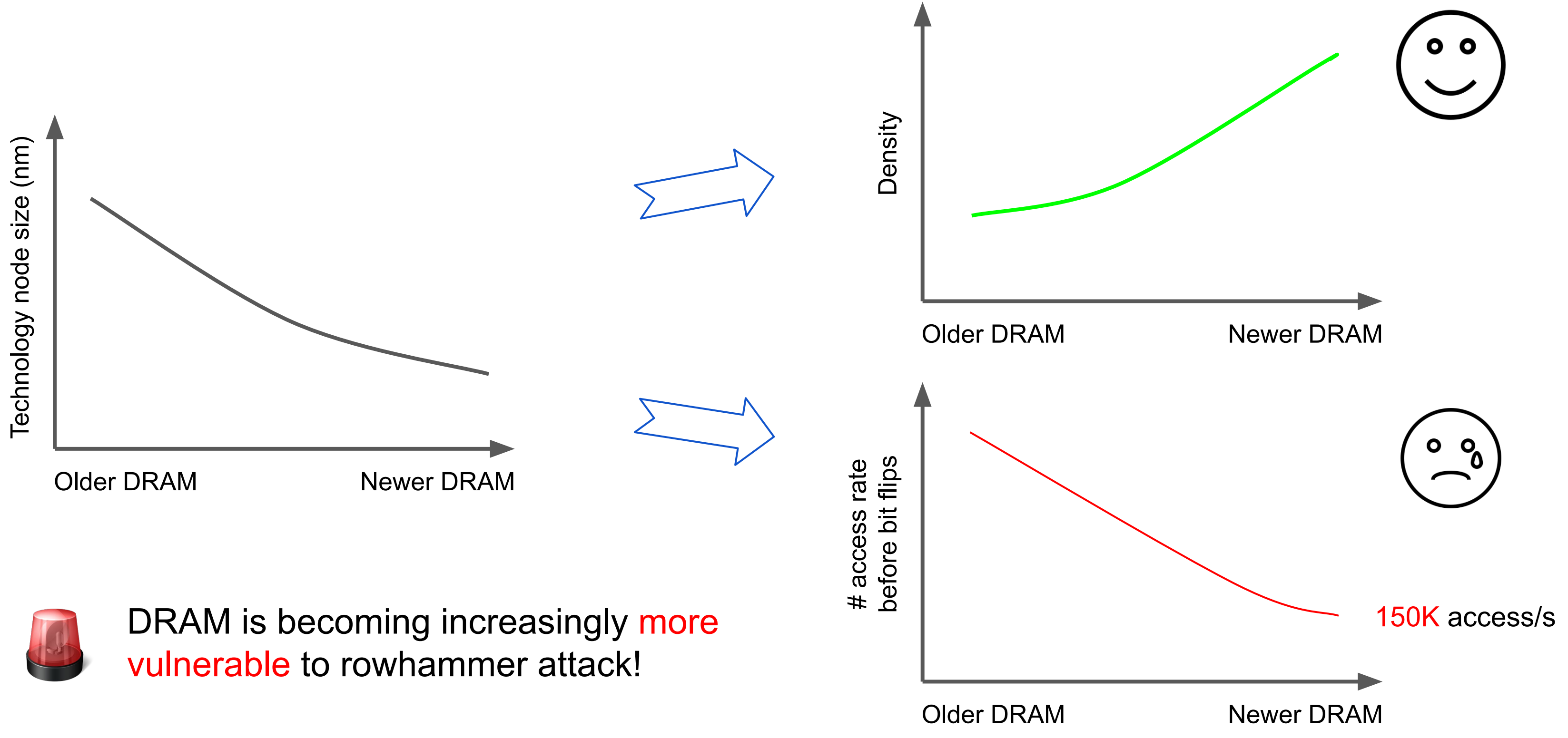
Data corruption

Information leak



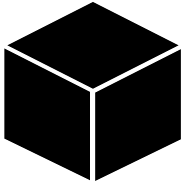
Privilege escalation

Trends in DRAM Technology

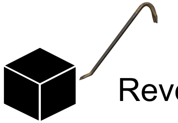


DRAM is becoming increasingly more vulnerable to rowhammer attack!

How to Rowhammer SSD?



SSD works as a black box, little internals known

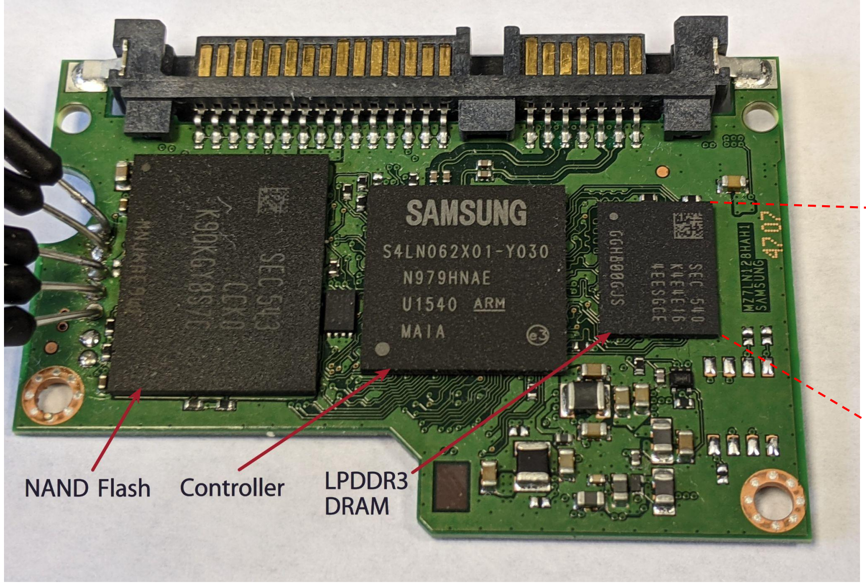


Reveal internals by reverse engineering

Not able to run rowhammer code directly

Find an indirect way for rowhammering

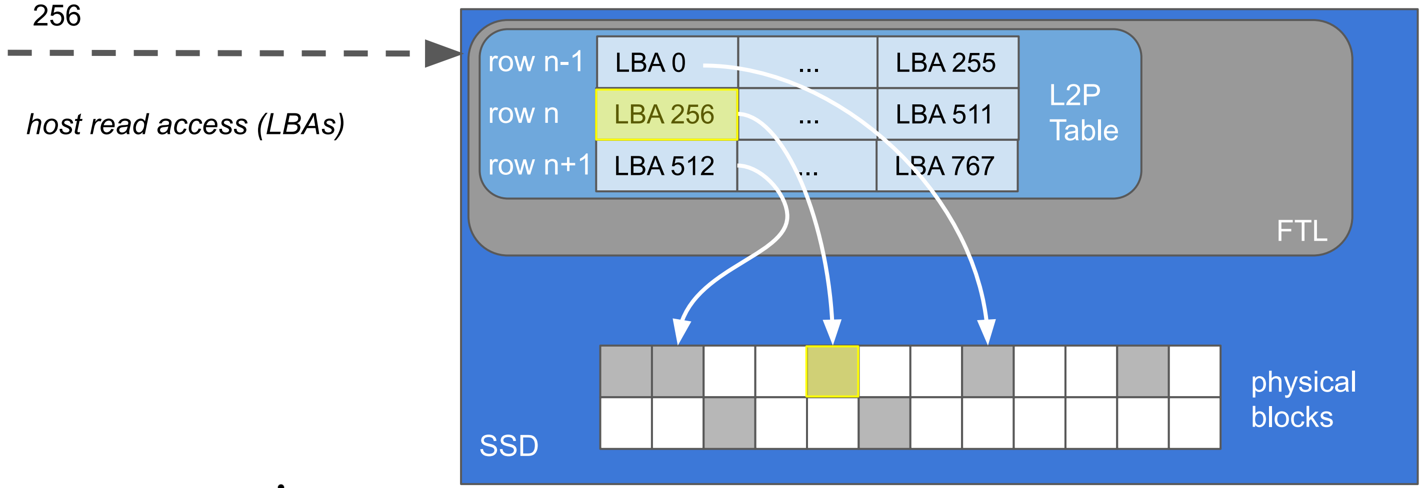
SSD Reverse Engineering



- SSD as a “computer system”
  - 3 core Cortex-R4 ARMv7 CPU
  - 512MiB LPDDR3 DRAM
  - 120GiB NAND flash
- SSD DRAM
  - FTL runtime code & data
  - Buffer for incoming I/O commands
  - Logical-to-physical mapping table (L2P)
    - Linear table, hash table
  - DRAM accesses are uncached

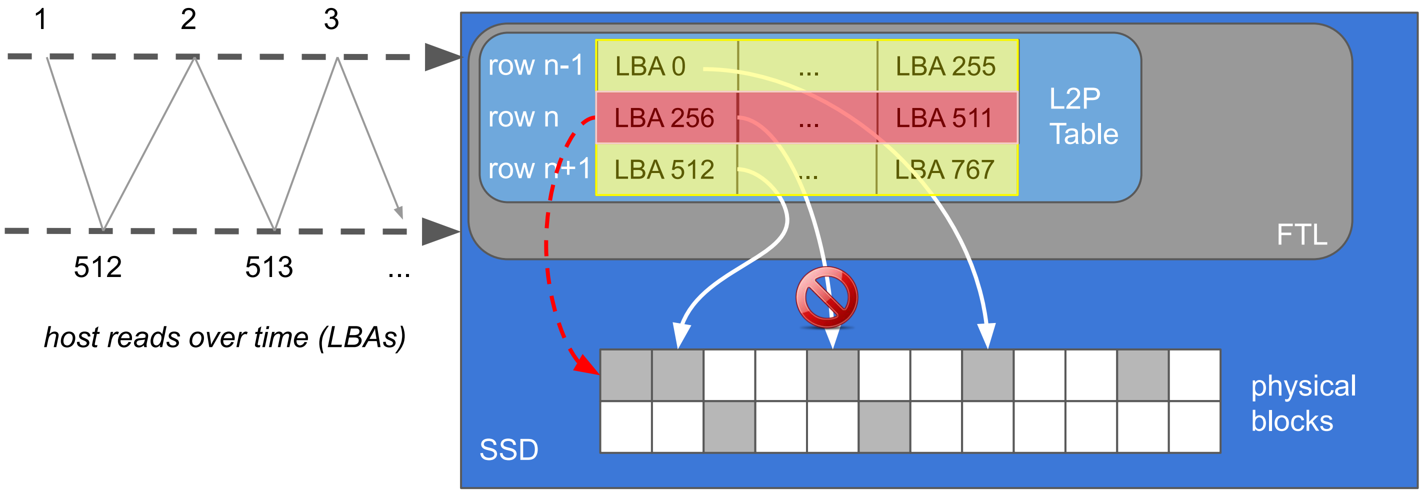
Can we rowhammer it?

L2P Table



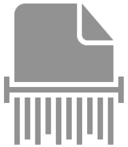
Rowhammer L2P by using SSD as intended (read)

Rowhammering the L2P Table



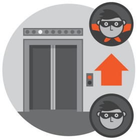
Latest PCIe 4.0 NVMe SSDs provide ~1.5M IOPS, compare to decreasing access rate for rowhammer (150K access/s)

Outcomes of Rowhammering SSD?



Data corruption

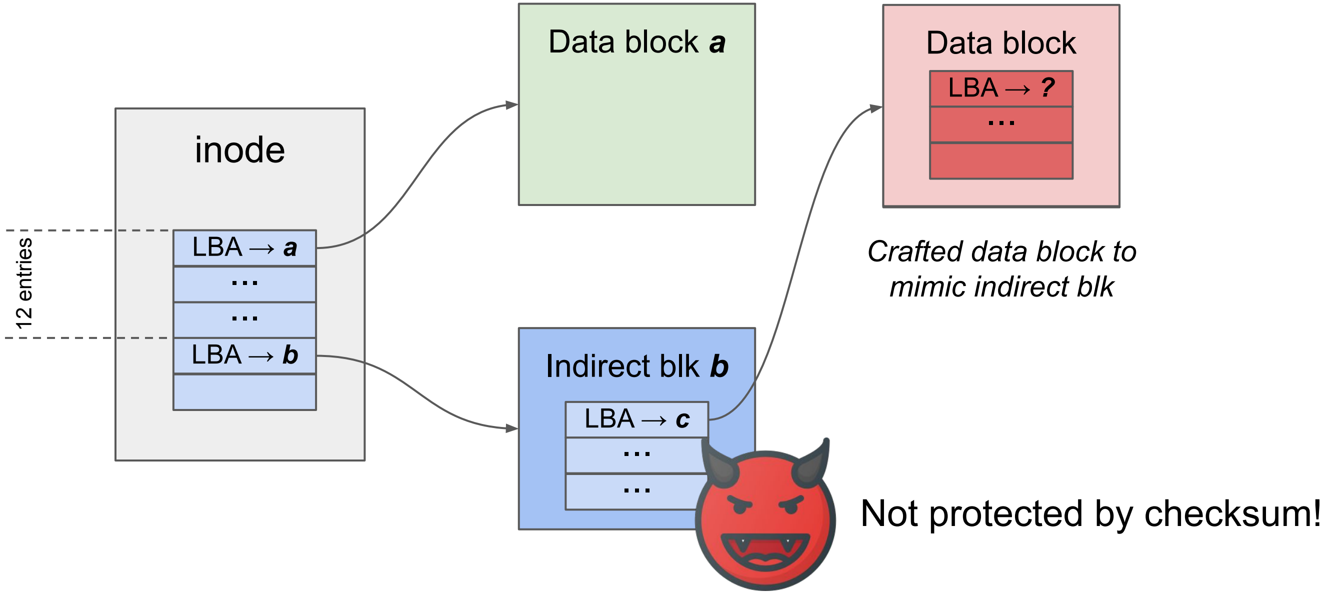
Information leak



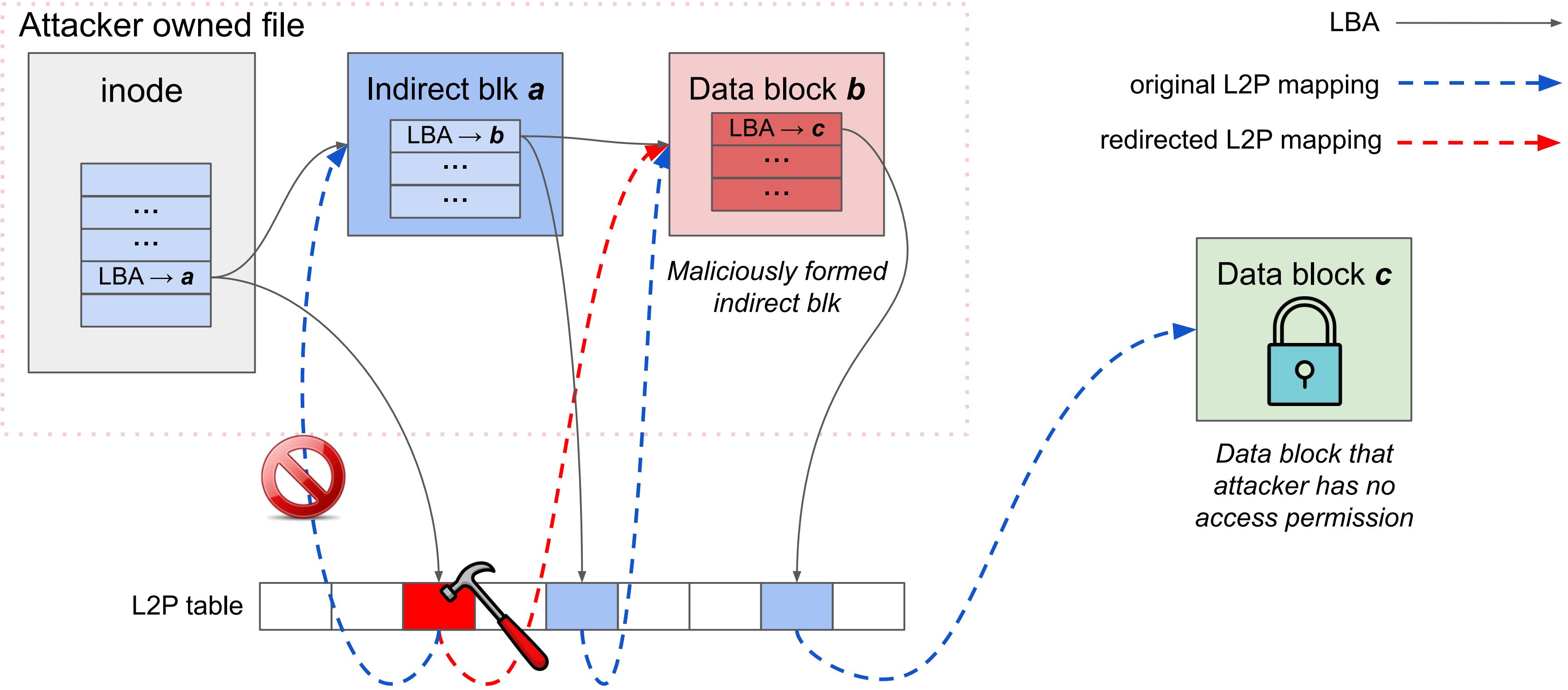
Privilege escalation

Need SSD-oriented exploits to turn bitflips into meaningful results.

Ext4 Direct/Indirect Block Addressing



Attacking the Ext4 Indirect Block



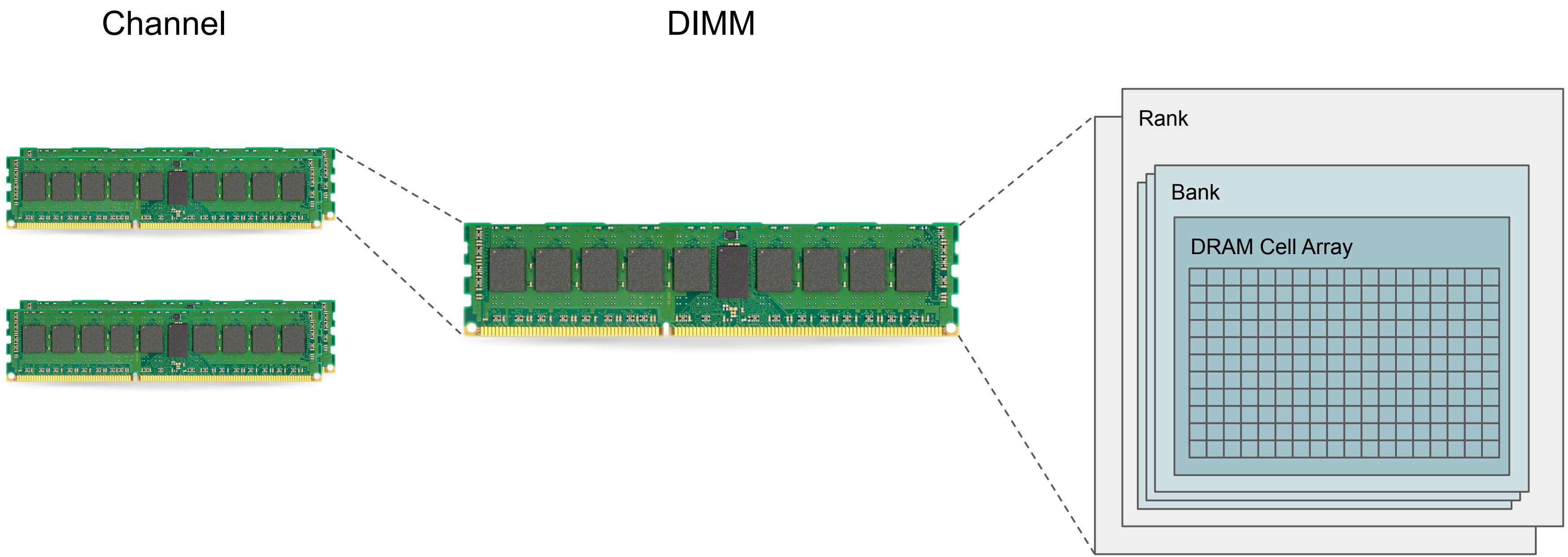
Conclusions

- Threats of rowhammer attack extended to a new dimension, storage devices
  - What about other attacks targeting host-side hardware?
  - What are possible mitigations?
  - Do we have a more principled solution?

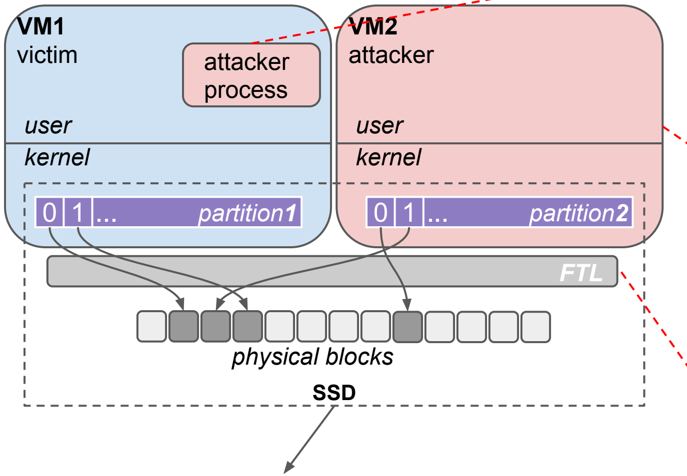
zhtao@cs.unc.edu

Discussions

*DRAM Overview



Attack Scenario: Cloud Server



Emulated with Intel SPDK

- L2P placed on rowhammer vulnerable uncached DRAM region
- Amplified L2P access to compensate relatively slow hardware

Unprivileged attacker process in victim system:

- Has normal access to owned files
  - read/write/create/delete

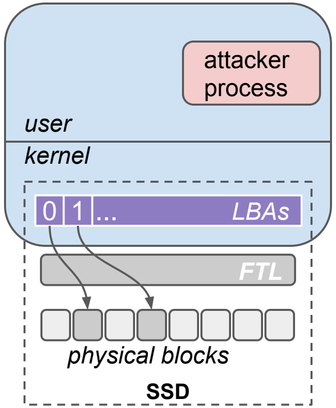
Attacker VM shares the same SSD with victim:

- Hardware pass-through to SSD partition
  - SRIOV or namespace

FTL manages L2P table and physical blocks, shared between VMs.

Accesses to empty LBAs (LBAs unwritten/TRIMed) are served faster

Attack Scenario: Single System



Attacker has a normal process in victim system

- Has unprivileged access to owned files
- Fast direct access to the underlying storage
  - `O_DIRECT` and libaio/io_uring

Attacker needs SSD that can serve read to non-empty LBAs fast enough

Mitigations

- Mitigations for host-side rowhammer attacks (e.g. ECC, TRR, cache)
  - Impact on performance, cost-efficiency, power-efficiency
  - Attacks circumventing these mitigations available
- Hardening the FTL
  - Stronger isolation between partitions/namespaces
  - Randomize FTL-internal structures
- Enforce block-level data integrity protection and encryption
  - Enforce extent tree addressing for Ext4
  - Can’t stop data corruption