RAMBleed
Reading Bits in Memory Without Accessing Them

Andrew Kwong [§] Daniel Genkin [§] Daniel Gruss [‡] Yuval Yarom [†]

[§] University of Michigan [‡] Graz University of Technology
[†] University of Adelaide and Data61

In Proceedings of the
41st Annual IEEE Symposium on Security & Privacy ***(S&P)***, Oakland, CA, May 2020

Presented by: Arno Esterhammer

Slide Credit: Onur Mutlu, Andrew Kwong

Executive Summary

- RAMBleed
  - Based on Rowhammer, formerly used to write bits
  - Paper shows how to read bits using Rowhammer
- How?
  - Find flippable bits in memory
  - Layout victim data as desired
  - Hammer rows & Infer bits of the secret
- Even ECC memory is affected

Outline

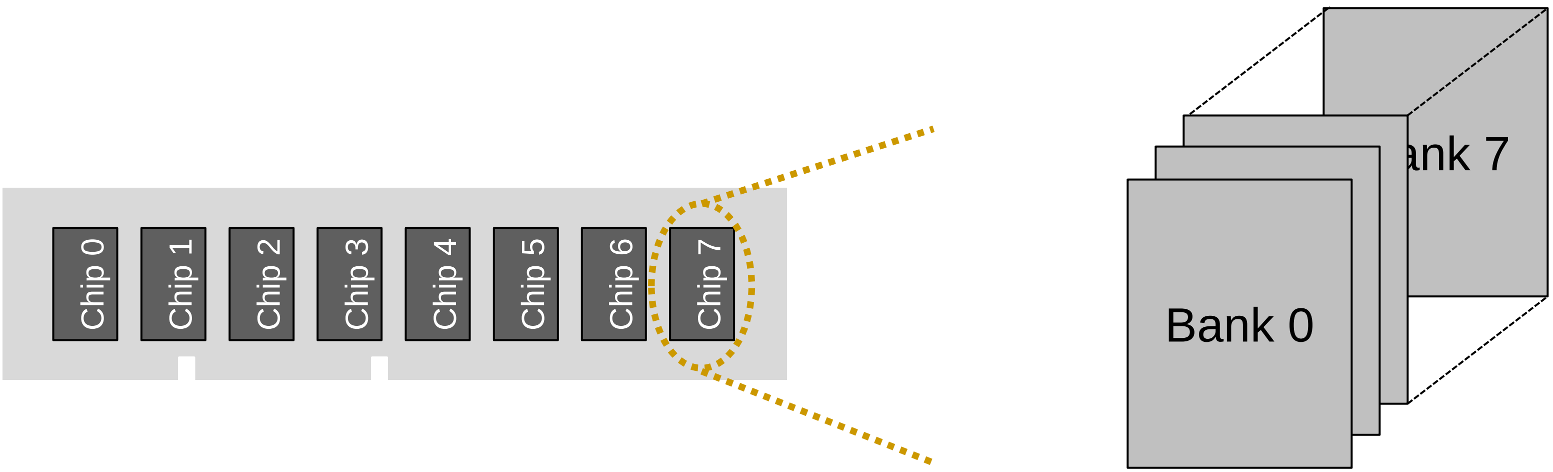
- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation

- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

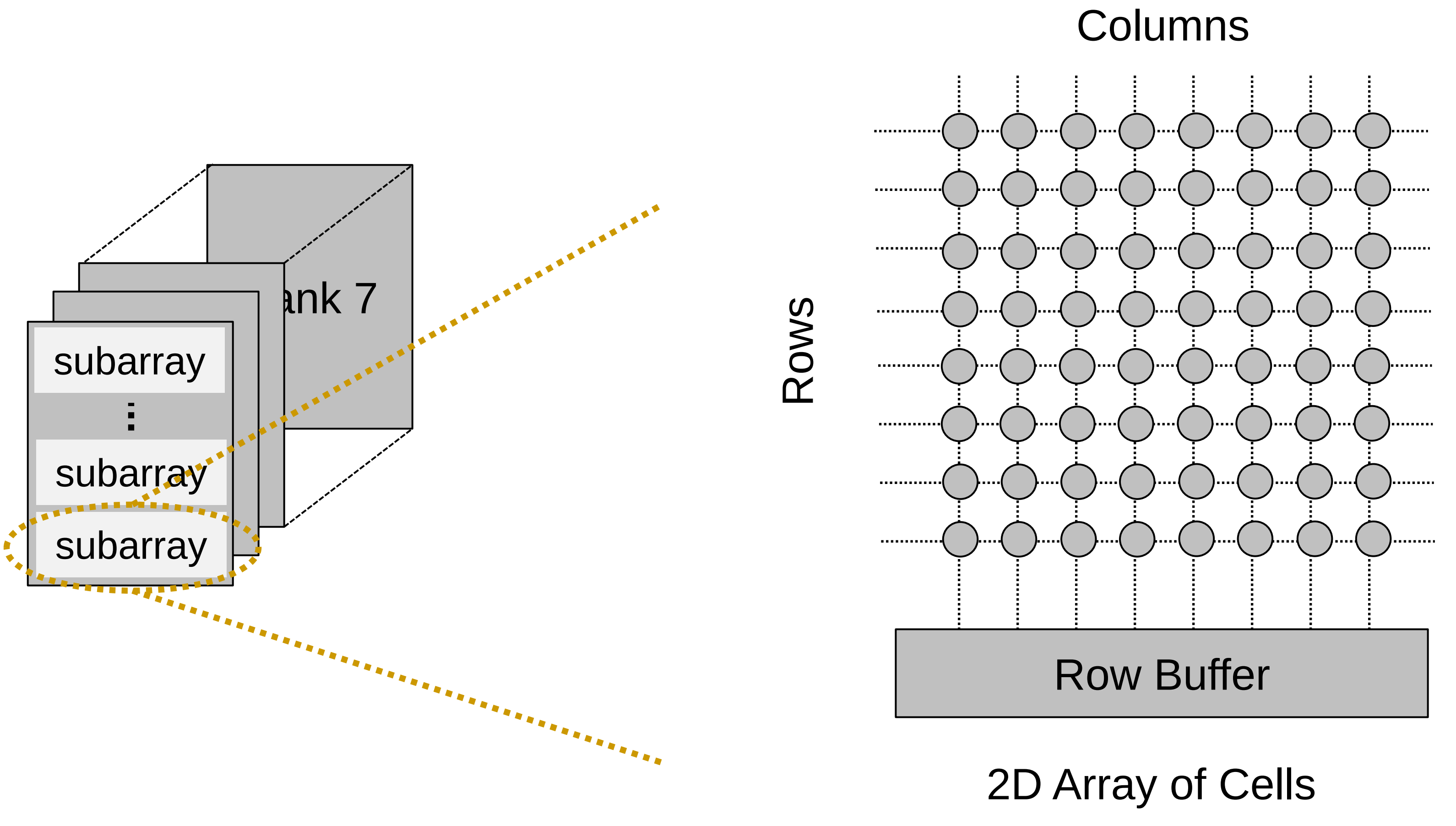
Outline

- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation
- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

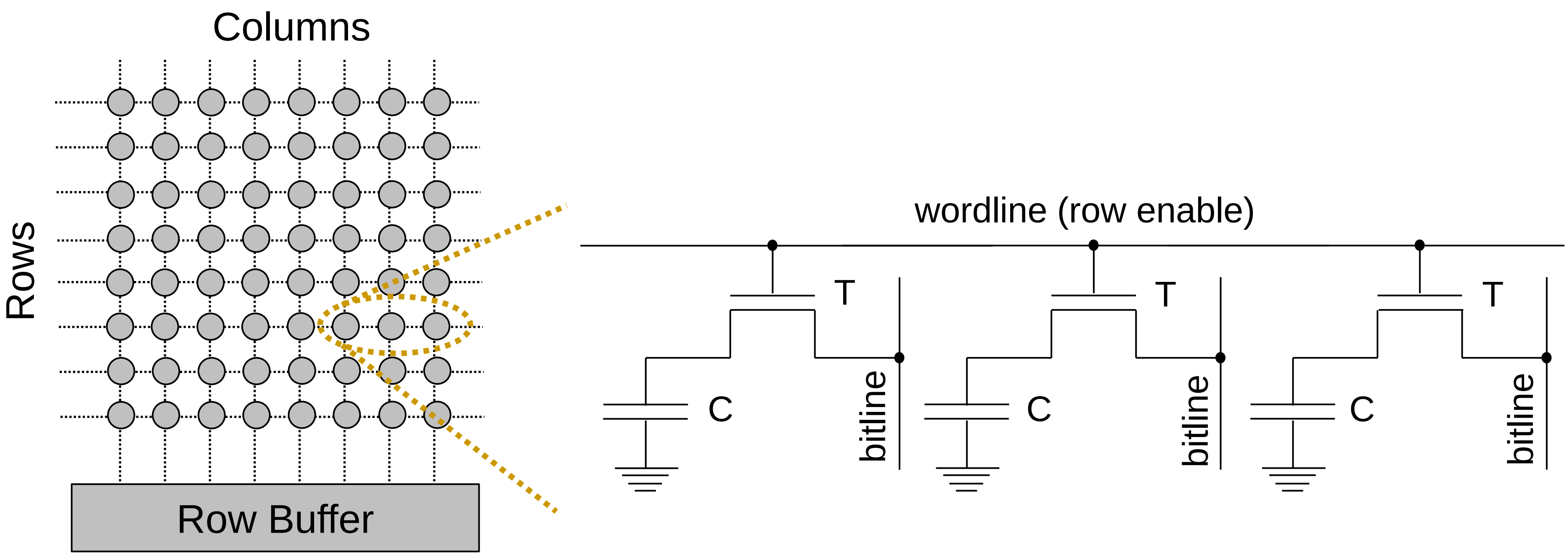
Recap: DRAM



Recap: DRAM


Bank 7
subarray
subarray
subarray
Columns
Rows
Row Buffer
2D Array of Cells

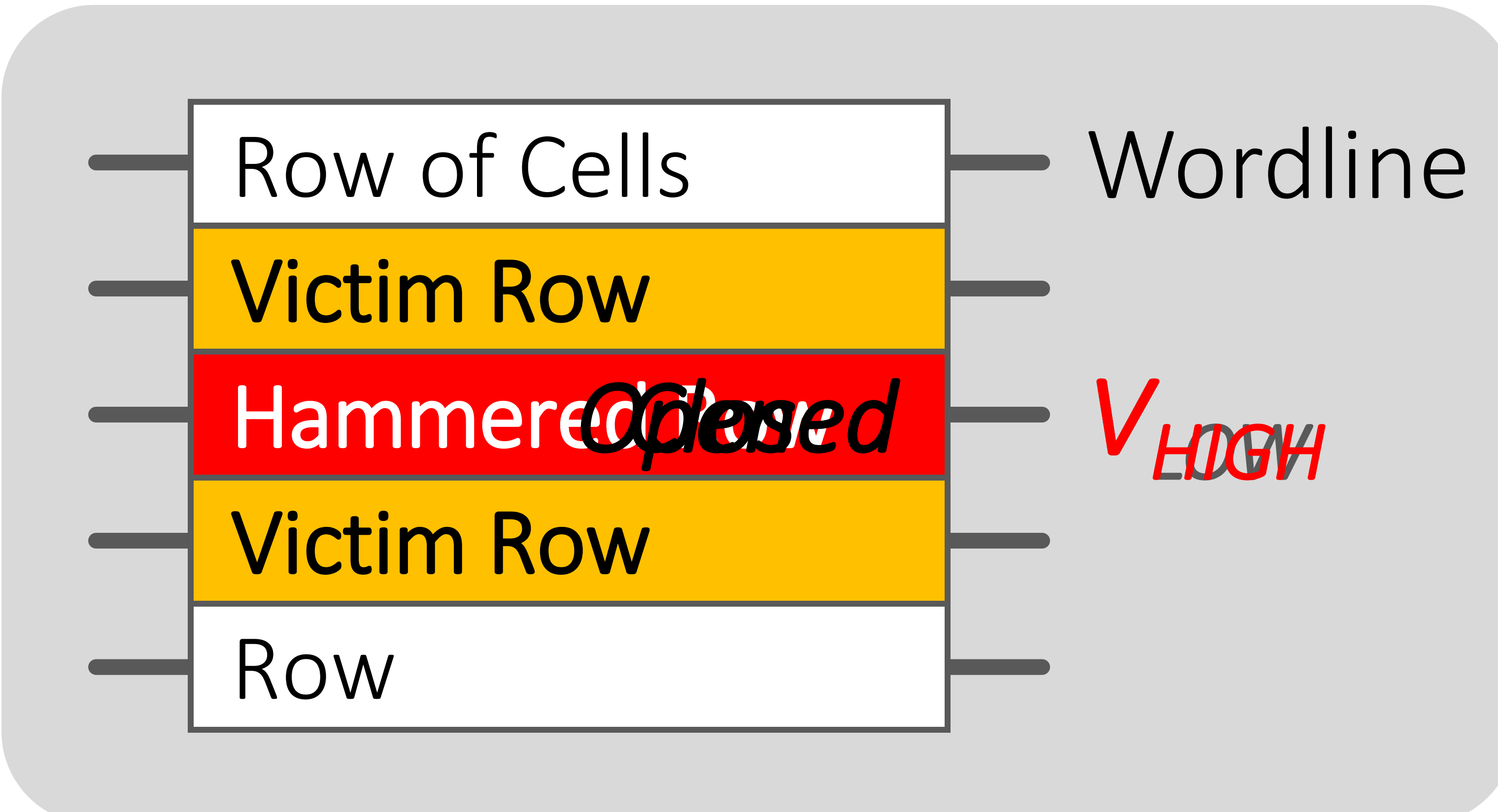
Recap: DRAM



Columns
Rows
Row Buffer
wordline (row enable)
T
C
bitline
T
C
bitline
T
C
bitline

Recap: Rowhammer

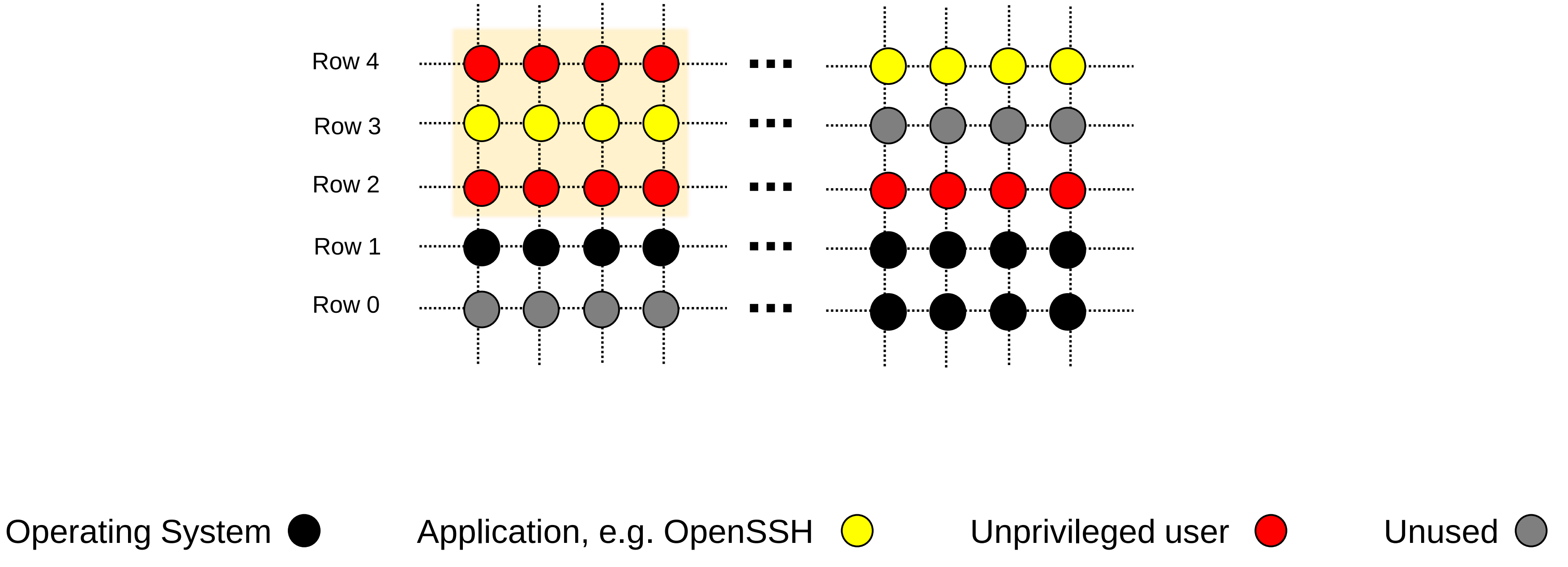
- Disturbance errors due to repeatedly reading same row



Animation: [Onur Mutlu, Presentation on RowHammer]

Problem

- DRAM is a highly shared resource
- Note: different security domains located in neighboring rows
- In combination with Rowhammer poses security risk



Goal of the Paper

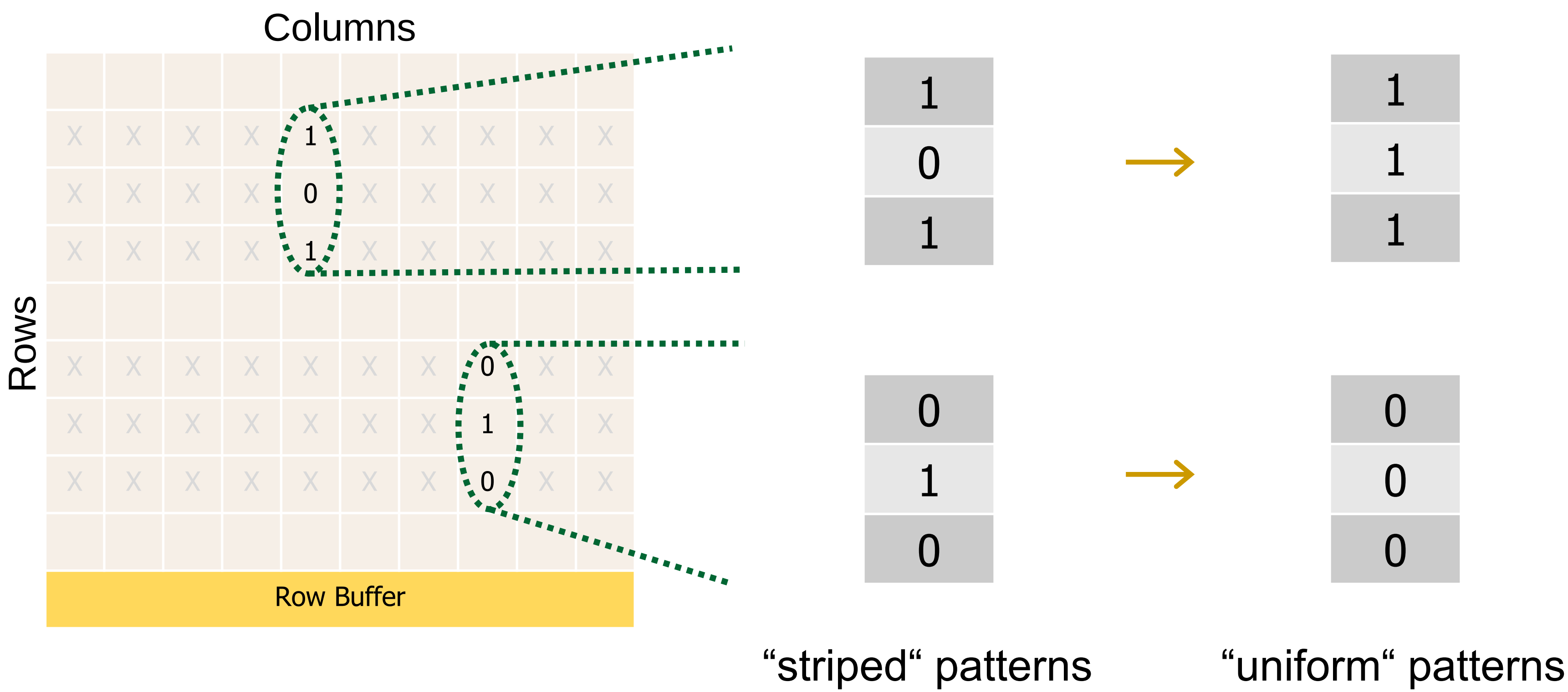
- Use Rowhammer to read secret data
- How?
  - Find memory locations vulnerable to bitflips
  - Intelligently place victim data inside memory
  - Hammer rows & Infer bits of the secret
- Results
  - End-to-End attack on Open SSH Server
    - Desktop machine (without ECC)
    - Server machine (with ECC)

Outline

- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation
- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

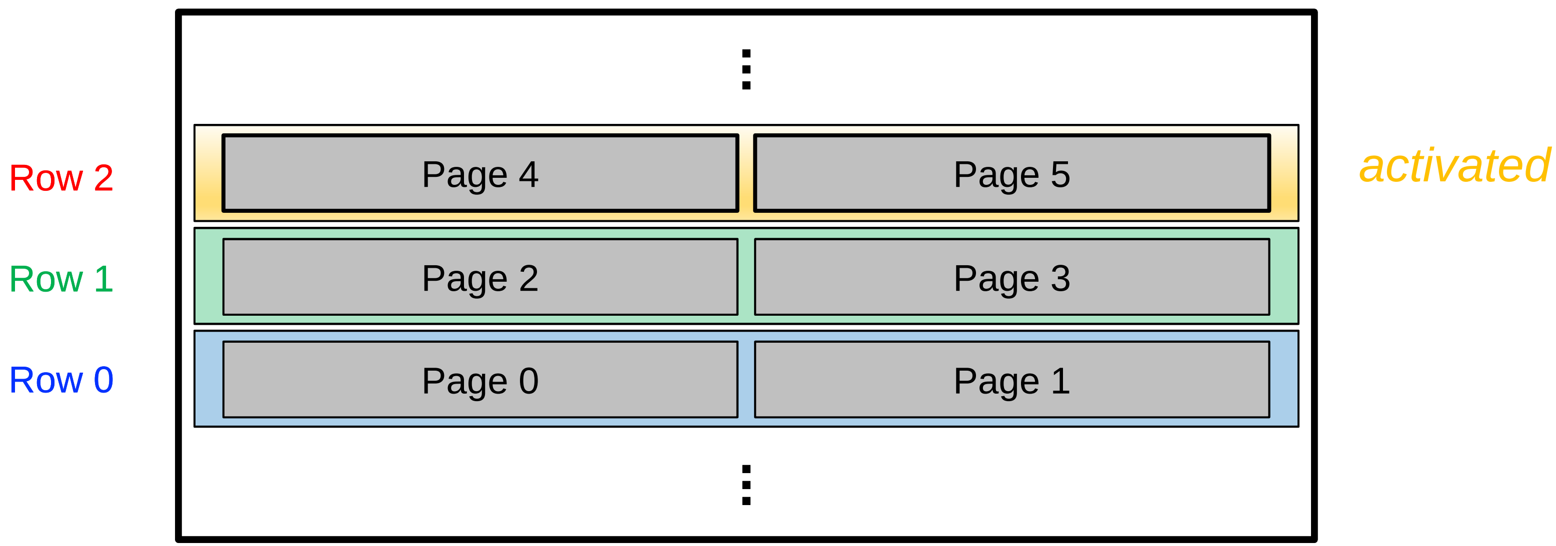
Observation

- Bit-flips in Rowhammer
  - Dependent on orientation of bit (i.e. 1 to 0 or 0 to 1)
  - also depend on neighboring bits!



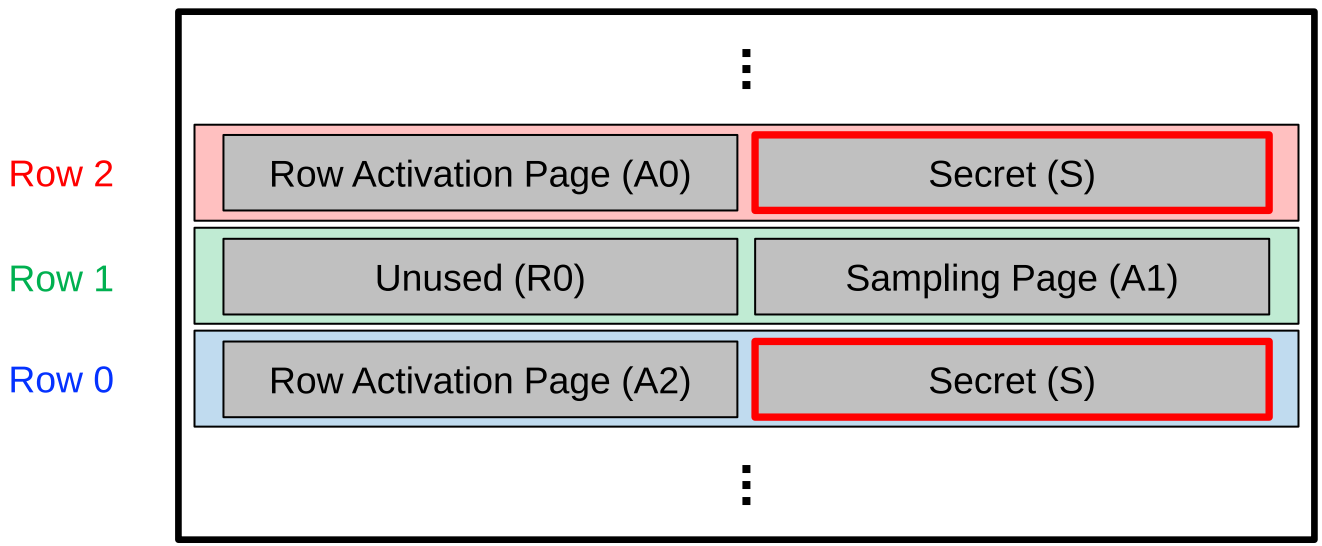
Observation

- DRAM banks operate on resolution of a row
  - typically 8KB
- 2 pages per row
- Access to one page → activates another page

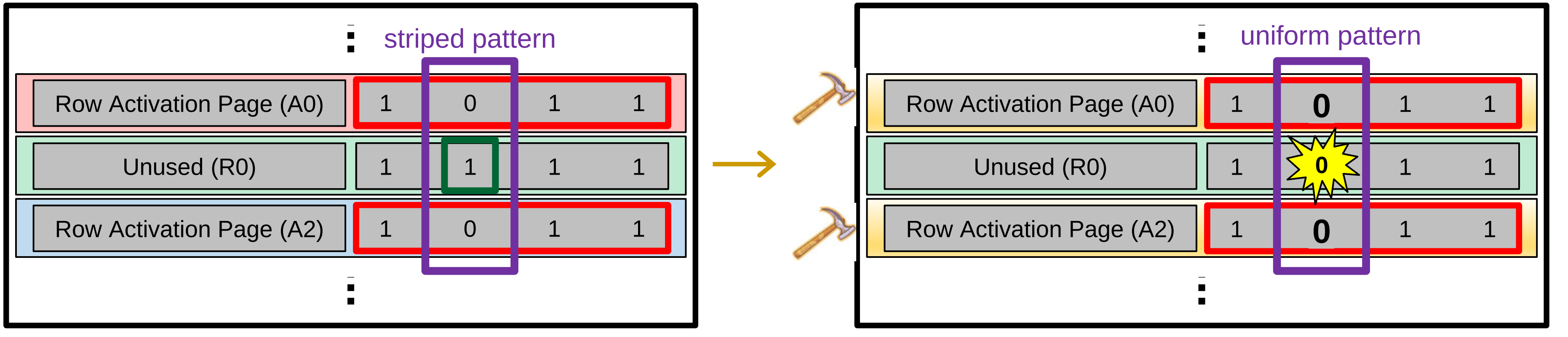


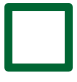
Idea – Combining these Observations

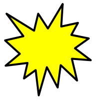
- Layout the memory in the following way
  - Sampling Page in between two identical copies of Secret
  - Activation of A0 and A2 also triggers copies of S
  - Thereby hammering A1
- No access permissions needed for pages S



Example: Inferring Bits (1)

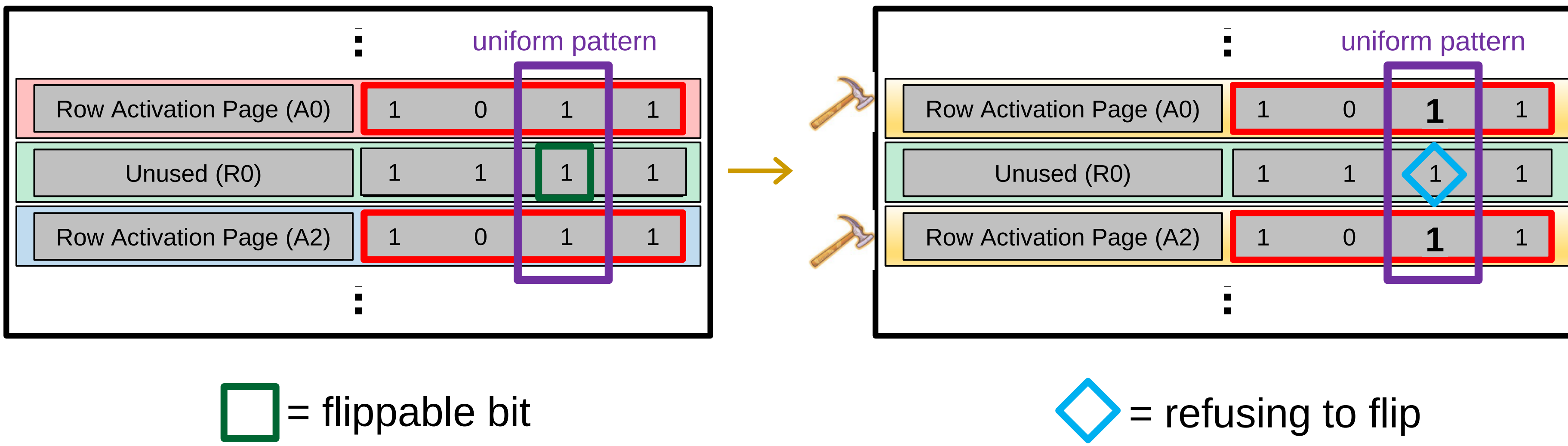






… infer that bit of Secret was 0 at this location

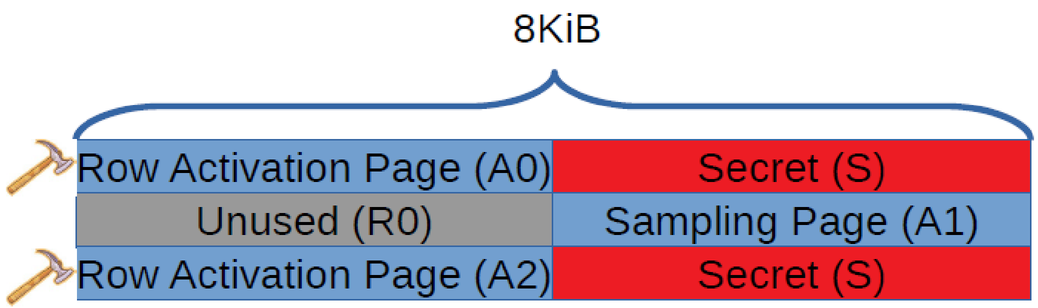
Example: Inferring Bits (2)



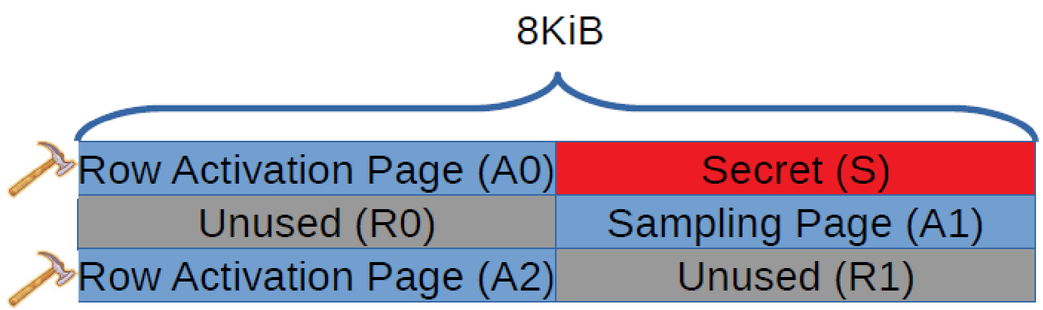
… infer that bit of Secret was 1 at this location

Types of RAMBleed

- 2 types presented in the paper



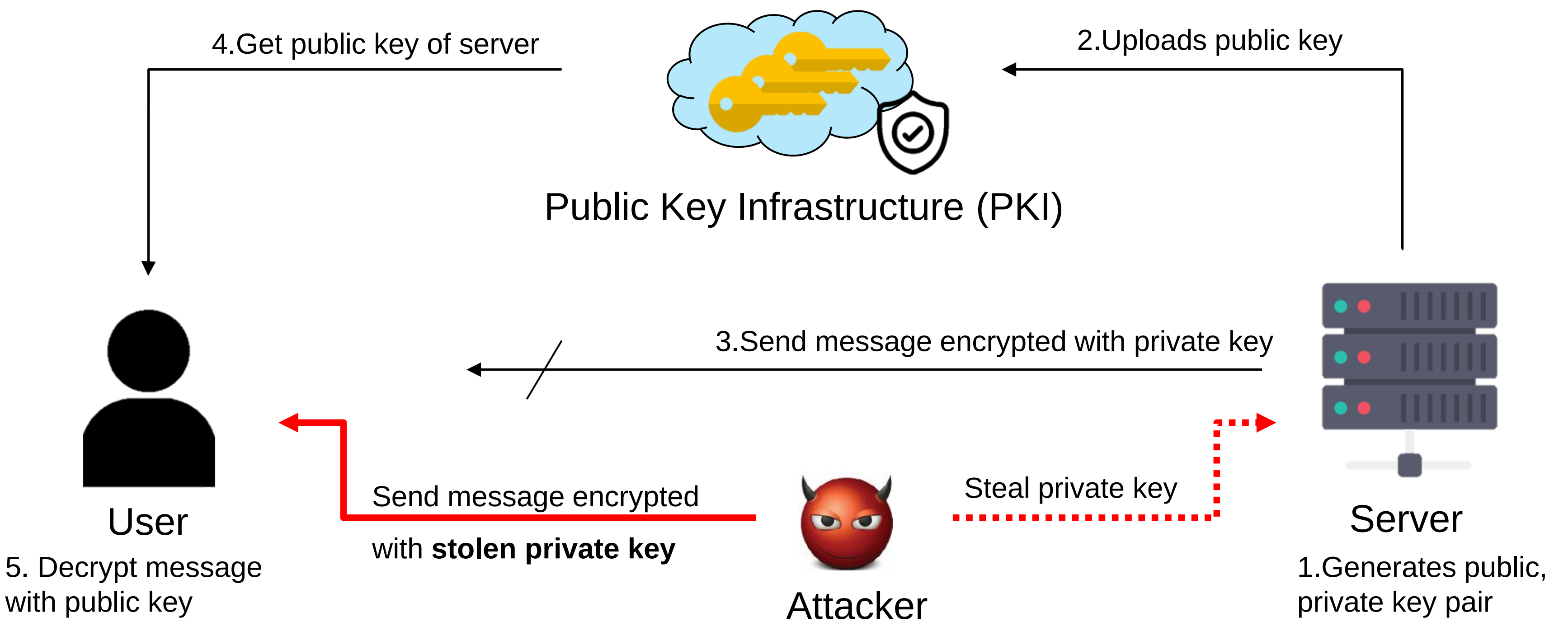
Double-sided RAMBleed



Single-sided RAMBleed

SSH (Secure Shell)

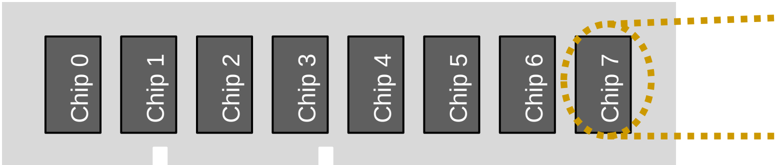
- cryptographic network protocol
- Uses RSA crypto system
  - Public key, Private key
- used for authentication (signing)

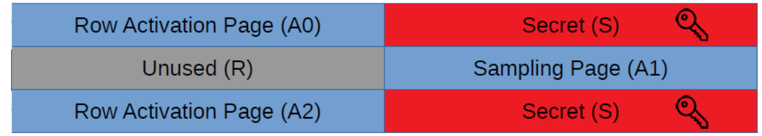


Outline

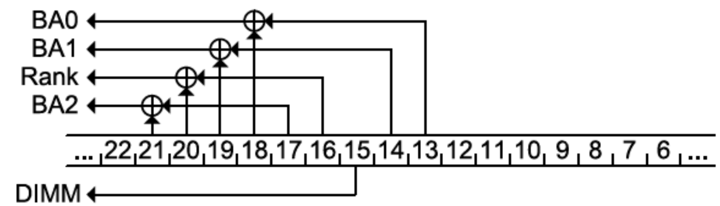
- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation
- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

Reversing the Mapping

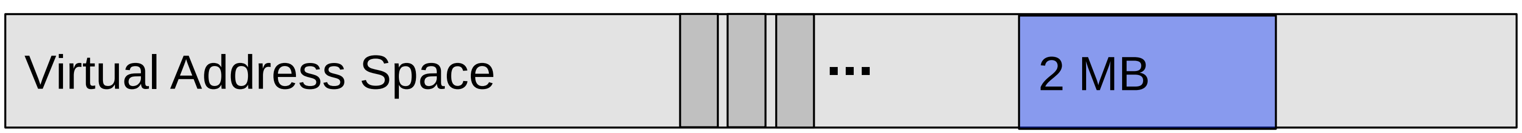




- DRAMA by Pessl et al
  - Able to reverse lower 22 bits of physical address
  - Need 2MB of contiguous physical memory



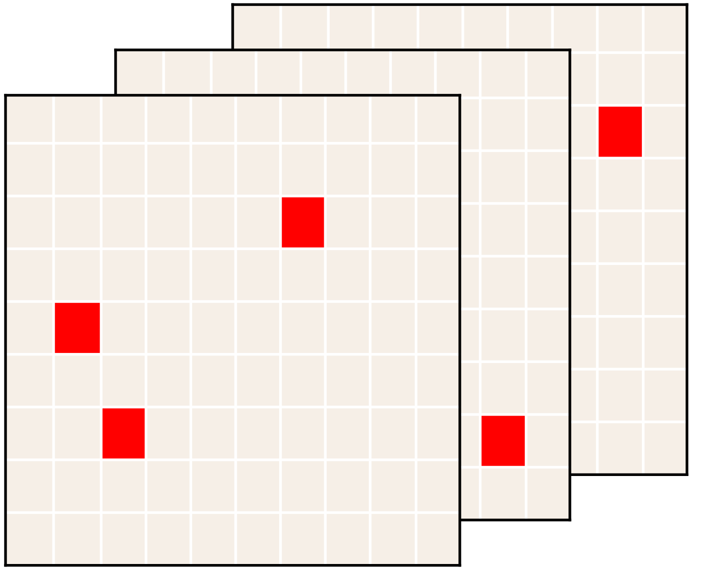
- Exhaust Small blocks of Linux Buddy Allocator
  - Until bigger blocks are served
- /proc/pagetypeinfo
  - to track available blocks



Memory Templating

- Scan the memory
  - Search for bits than can be flipped

- Take 3 consecutive rows and hammer
  - Remember for later, if a flip is observed



Frame Feng Shui

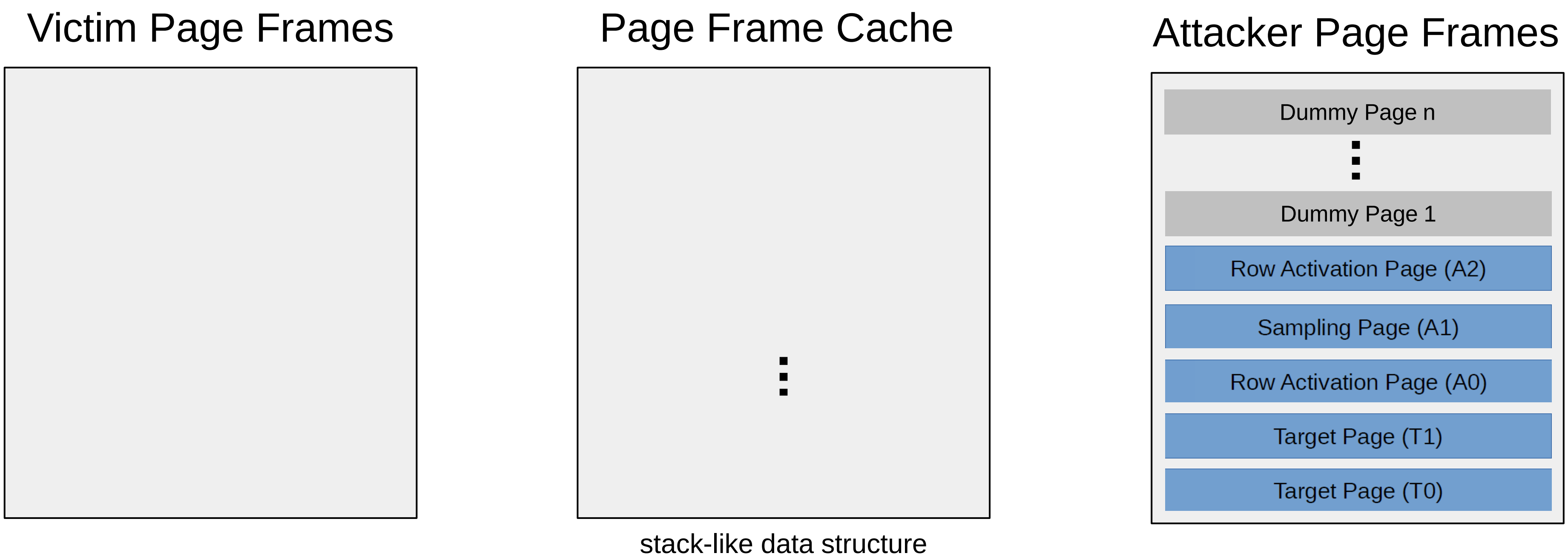
- Placing victim pages in desired physical location
- Exploiting Linux Page Frame Cache
  - Frames stored in FILO manner
  - i.e. returns most recently deallocated page on request
- Done in 3 phases
  - 1. Dummy allocations → allocate n pages (n = #pages before secret)
  - 2. Deallocation → choose target page & unmap it, unmap n pages from step 1
  - 3. Triggering the victim → e.g. by initiating an SSH connection
- Now secret is at the intended page
- Hammer until enough bits are recovered
  - ~66% bits suffice for SSH keys

Frame Feng Shui - Visualization

Victim Pseudo Code

```
alloc( d1 )
...
alloc( dn )
alloc( secret )
```

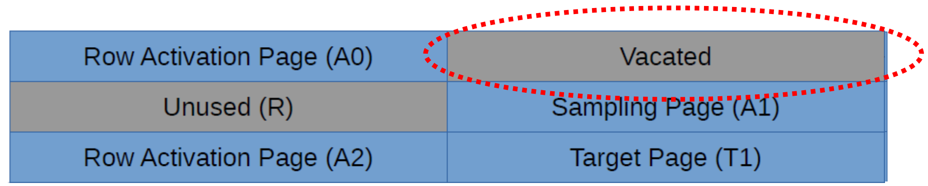
Row Activation Page (A0)	Target Page (T0)
Unused (R)	Sampling Page (A1)
Row Activation Page (A2)	Target Page (T1)

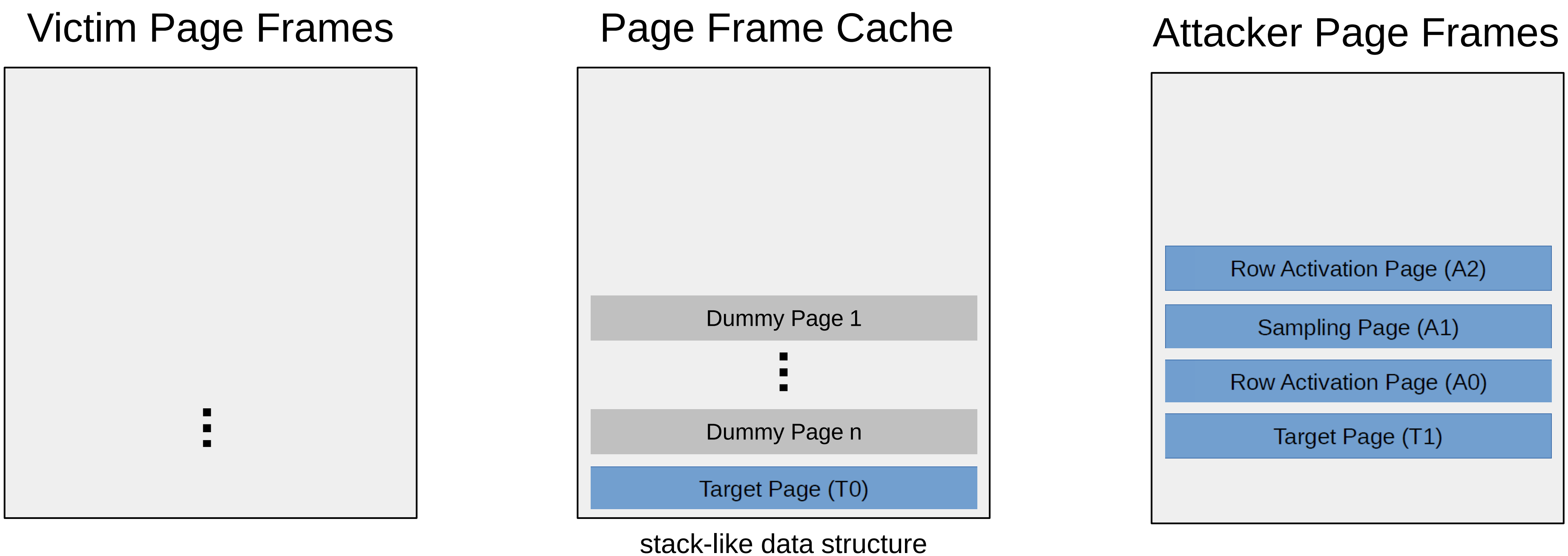


Frame Feng Shui - Visualization

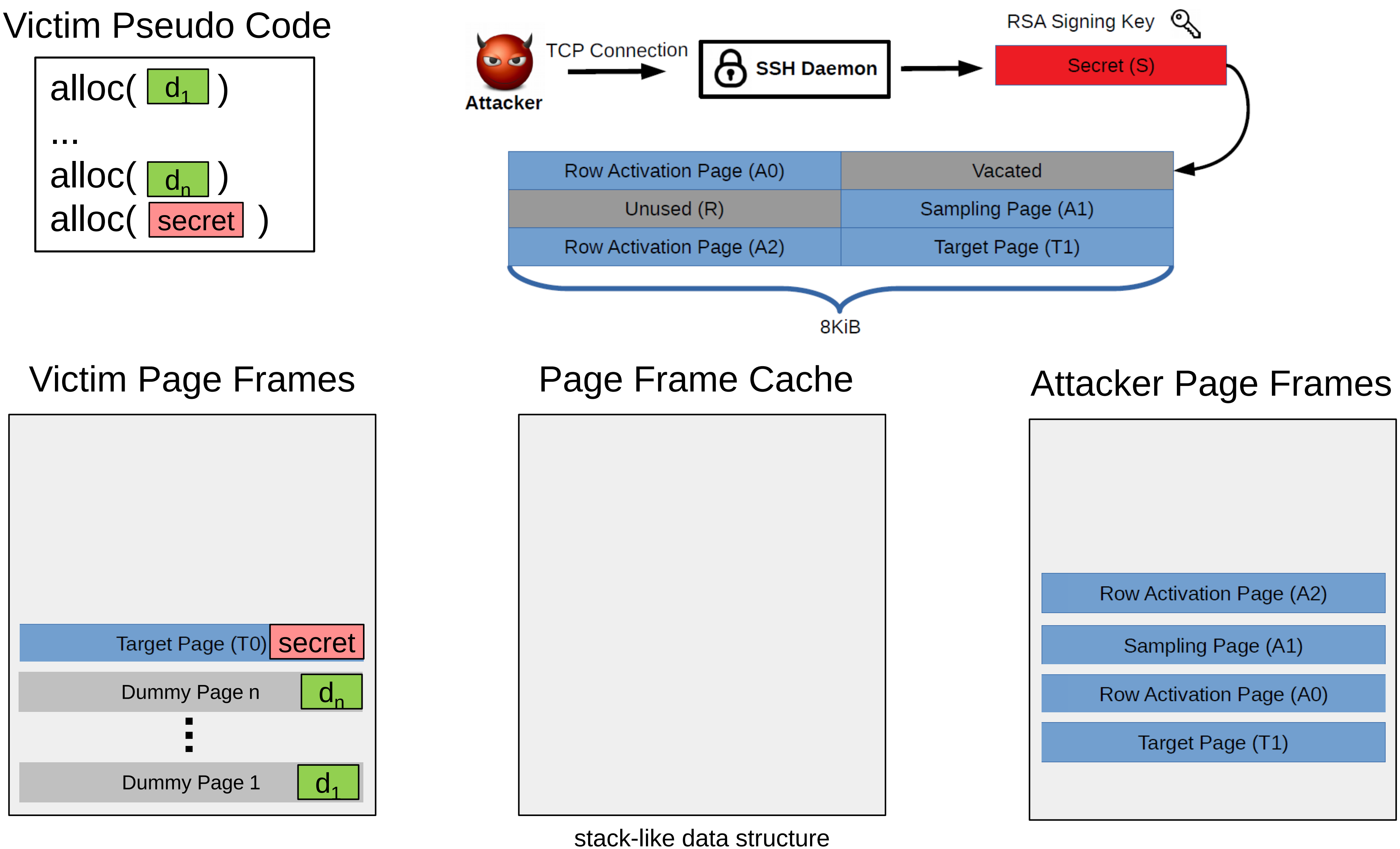
Victim Pseudo Code

alloc(d_1)
...
alloc(d_n)
alloc(secret)





Frame Feng Shui - Visualization



Attack (Summary)

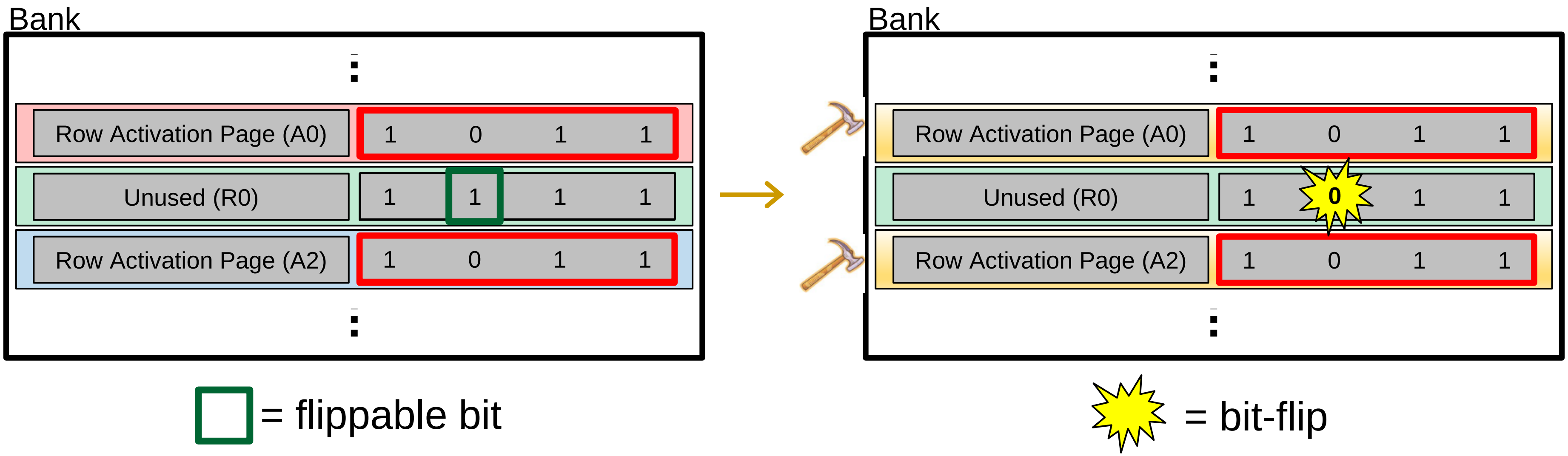
- Find flippable bits
  - Reverse engineer the mapping (virt. → phys. → DIMM add.)
  - Memory Templating
- Layout Memory by (ab-)using
  - Linux Buddy Allocator
  - Linux Page Frame Cache → Frame Feng Shui
- Hammer & Infer bits
- As soon as enough bits could be retrieved
  - Makes use of redundancy present in SSH-keys
  - Use a variant of Heninger-Shacham Technique to obtain full SSH-key

ECC memory

- ECC (Error-Correcting-Codes)
- Used in server machines to ensure data integrity
- Originally to correct rare bit-flips by cosmic radiation
- Usually only able to correct 1 error and detect 2 errors (SECDED)
- Corrected when read

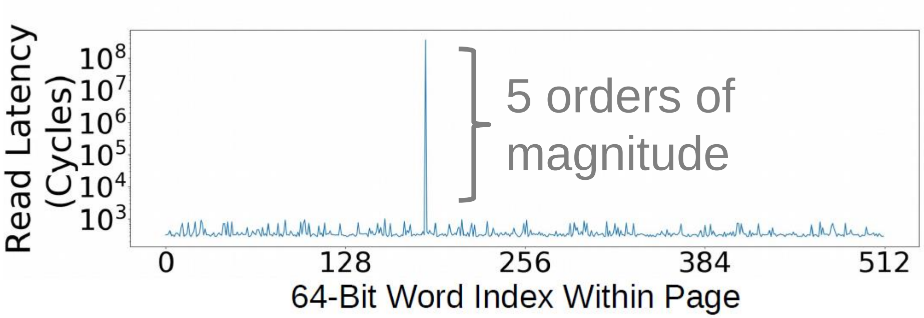
Example: Inferring Bits on ECC Memory

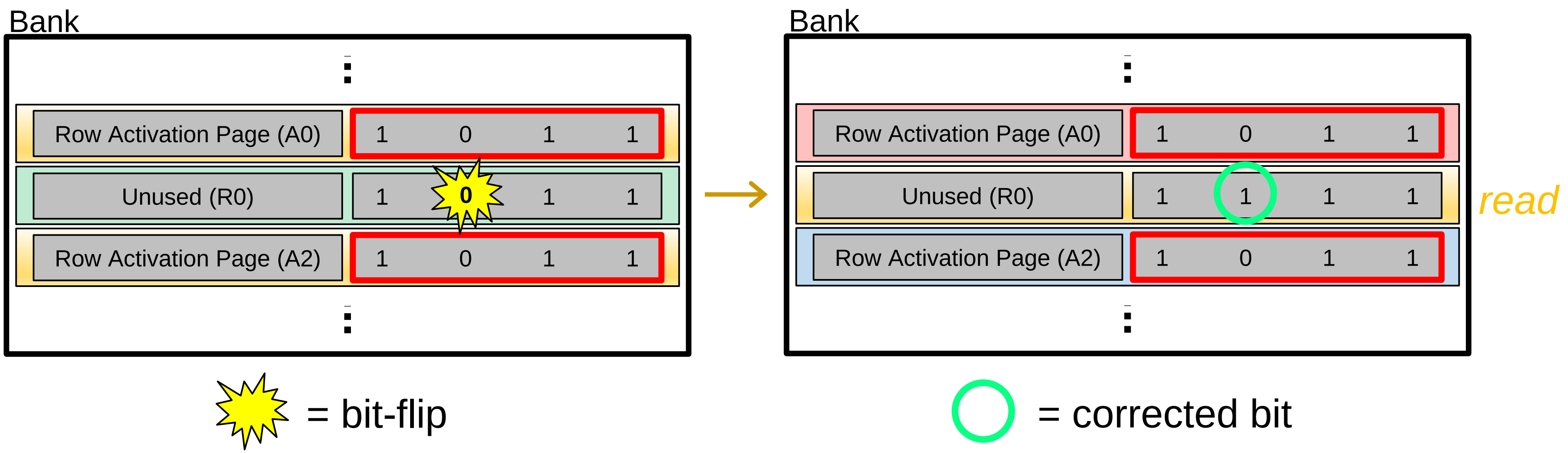
- After hammering bit flip occurs



Example: Inferring Bits on ECC Memory

- After hammering bit flip occurs
- But gets correct when reading
- Takes 100.000s of cycles to correct → observable





Outline

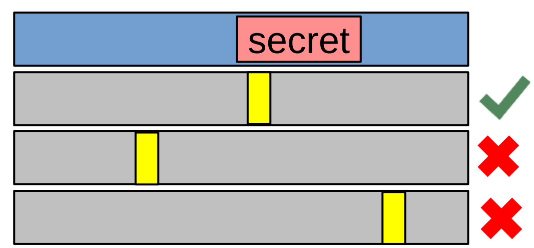
- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation
- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

Results (1)

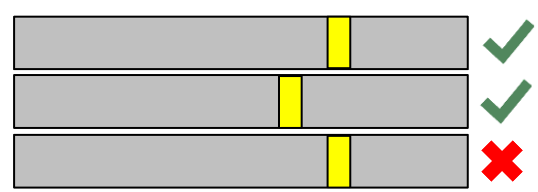
- Two experiments
  - Desktop machine (without ECC)
  - Server machine (with ECC)
- Online Phase
  - Need to read from ~4200 usable bits
  - Reading at 0.31 bits/second
  - With 82% accuracy on desktop machine (73% on server)
- Almost 4h to obtain the full key

Results (2)

- Memory templating
  - 84k bits (empirically chosen) → 4200 usable bits
  - 41 bitflips/min
  - 34h to find 84k flips
- Usable bits
  - 3/16 of bits are at position of secret key → ~15750 bits



  - Get rid of duplicate locations → ~4200 useful bits



Executive Summary

- RAMBleed
  - Based on Rowhammer, formerly used to write bits (breach for integrity)
  - Paper shows how to read bits using Rowhammer i.e. it breaks confidentiality
- How?
  - Find flippable bits in memory
  - Layout victim data as desired
  - Hammer rows & Infer secret
- Even ECC memory is affected

Outline

- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation
- Strengths & Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

Strengths of the Paper

- New interesting way of using Rowhammer
  - Use it as a read side channel
- Proof of Concept by a realistic Example
  - description of End-to-End attack
  - On commonly used software (Ubuntu + OpenSSH)
- Contribution
  - Combines findings of lots of prior works
  - And extended it to obtain new attack

Weaknesses/Limitations of the Paper

- Were prior Rowhammer exploits not also a way of breaking confidentiality?
- Are servers that susceptible to that attack?
  - Might be hard to predict the scheduling of threads
- Victim needs to be operating very predictably (e.g. #pages allocated before secret, …)
- Limited to secret data which has redundancy in it
- Modest Bit-Rate for reading bits
- Does not consider multi-processor setup

Outline

- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation
- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

Thoughts and Ideas/Discussion Starters

- Ways of mitigating RAMBleed?

- Possible Mitigations
  - HW
    - Increasing Refresh Intervals
    - TRR (Target Row Refresh) proposed by vendors
    - PARA (Probabilistic Adjacent Row Activation)
  - SW
    - Encryption = e.g. enclaves in SGX
    - 0-ing out data
    - Probabilistic Memory Allocator

Thoughts and Ideas/Discussion Starters

- Compilation of data to less susceptible bit-patterns?
- Is it necessary to isolate different security domains?

Outline

- Background, Problem, Goal
- Novelty, Key Approaches and Ideas
- Mechanisms
- Key Results: Methodology and Evaluation

- Strengths and Weaknesses
- Thoughts and Ideas / Discussion Starters
- Takeaways

Conflicting Trends

- Challenges for DRAM
  - Capacity
    - More capacitors on less space
    - Disturbance between them increases
  - Power Consumption
    - More Capacity → increases energy for refresh
    - Less Energy for refresh by intelligent refresh (RAIDR, …)
  - …

- These trends worsen the breach posed by Rowhammer and in turn RAMBleed

Extensions & Follow-Up Work

- Can this idea be improved s.t. higher bit-rates can be achieved?

- Can this idea be evaluated on other OSs, HW?

- Where to solve the problem?
  - HW level?
    - Ways to speed up ECC memory? (e.g. on-die ECC)

  - Involve higher levels in abstraction hierarchy?
    - E.g. better mapping from virtual to physical address space
    - Solutions specifically tailored to RAMBleed

RAMBleed
Reading Bits in Memory Without Accessing Them

Andrew Kwong [§] Daniel Genkin [§] Daniel Gruss [‡] Yuval Yarom [†]

[§] University of Michigan [‡] Graz University of Technology
[†] University of Adelaide and Data61

In Proceedings of the
41st Annual IEEE Symposium on Security & Privacy (***S&P***), Oakland, CA, May 2020

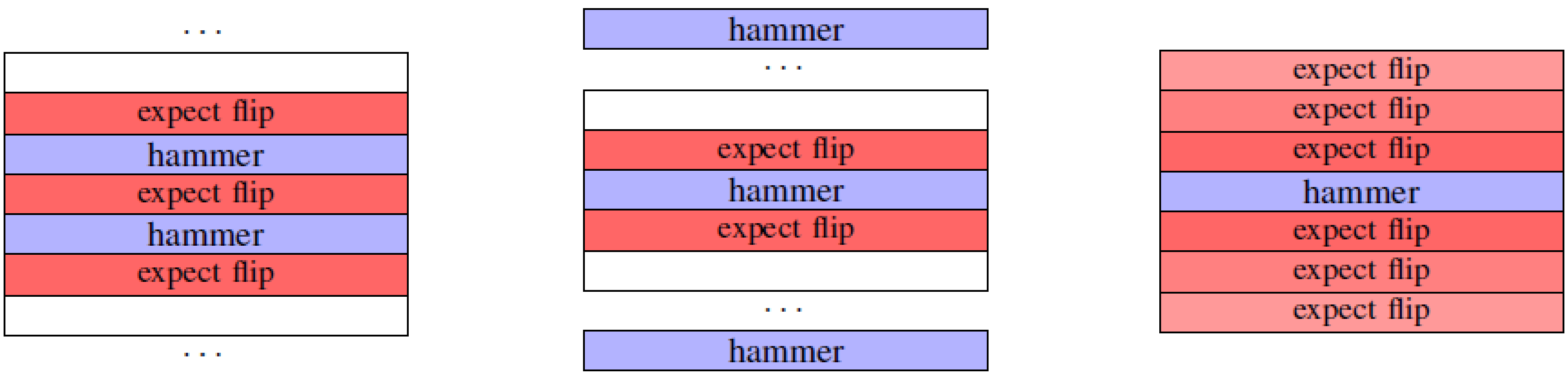
Presented by: Arno Esterhammer

Slide Credit: Onur Mutlu, Andrew Kwong

Backup Slides

Recap: Rowhammer

- 3 types of Rowhammer*



Double-sided RH　　Single-sided RH　　One-location RH

Figure: Paper on RAMBleed

* Daniel Gruss et al, Another Flip in the Wall of Rowhammer Defenses

Potential Problem: Memory Scrambling

- Memory Scrambling
  - To mitigate cold boot attacks
  - Avoid circuit damage due to resonant frequency
- Is not a problem for RAMBleed because striped patterns stay striped patterns even after scrambling [15]
- PRNG seed stays the same until machine is up

Linux Buddy Allocator (LBA)

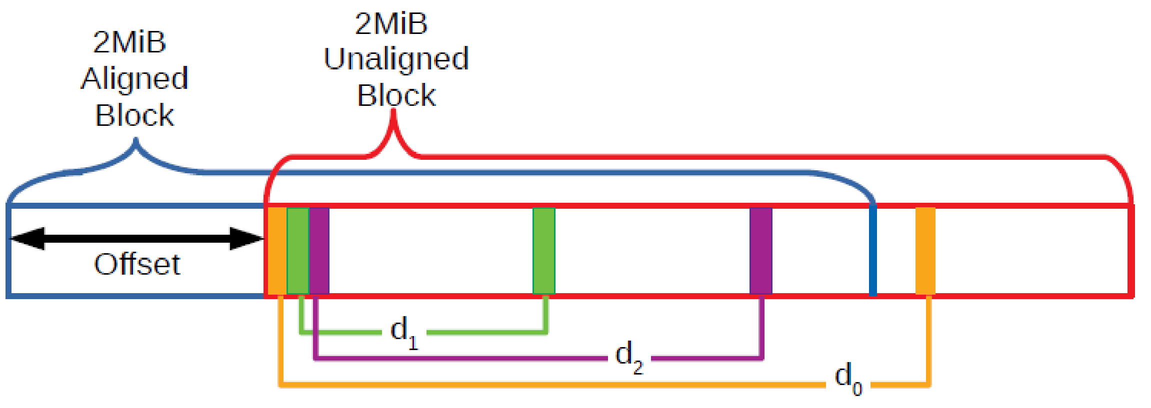
- Kernel stores memory in physically consecutive blocks
  - Arranged by order: nth order = 4096 * 2^n bytes
- Kernel maintains free-lists for blocks from order 0-10
- LBA tries to serve requests using smallest available blocks
  - If not possible → split the next smallest one into two "buddy" halves
- User space requests only allows order 0 requests
  - E.g. 16 KiB → LBA treats as 4 requests

Memory Massaging

- Need to get 2MiB of phys. contiguous memory
- Phase 1 → exhaust small blocks
  - Until less than 2MiB of free space is available in order <10
- 2 Requests for 2MiB
  - LBA needs to split order 10 block (=4MiB)
- After 1st request
  - there is more than 2MiB left of the split order 10 block
- The 2nd request results in phys. contiguous memory
  - Because the next 2 MiB are served in-order

Reversing Physical Address Bits

- Need to find out physical addresses of same-bank pages
- 2 MiB block from 2nd request might not be aligned on 2MiBs
- Use row-buffer timing side channel to find out offset
- Distance pattern uniquely identifies offset



Evaluation Environment

- Non-ECC Setup
  - HP Prodesk 600, Ubuntu 18.04, i5-4570 CPU
  - 2 Axiom DDR3 4GB 1333 MHz non-ECC DIMMs

- ECC Setup
  - Supermicro X10SLL-F motherboard
  - BIOD version 3.0a
  - With Intel Xeon E3-1270 v3 CPU
  - 2 Kingston 8GB 1333Mhz ECC DIMMs