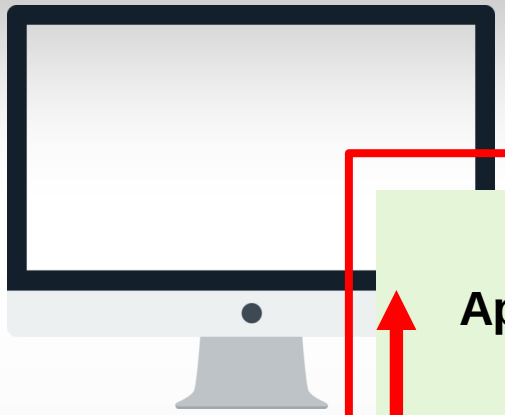
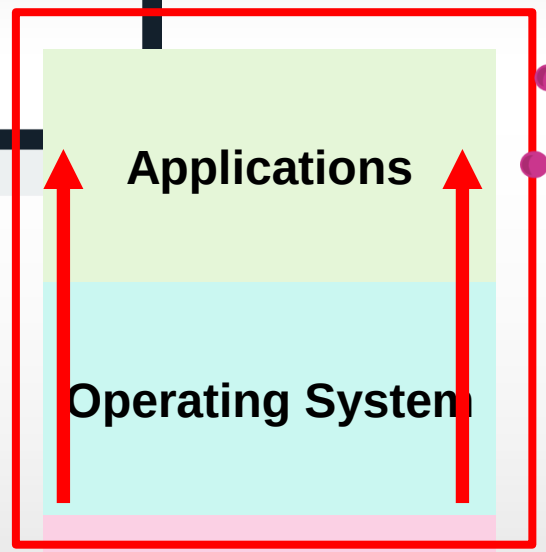
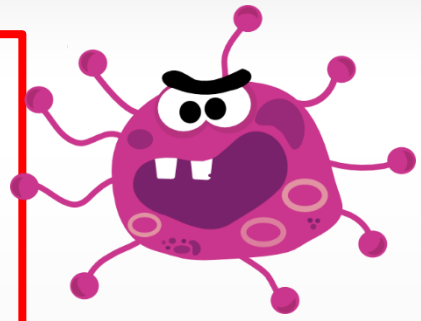
Applying Software Side-Channels to Hardware Vulnerabilities

Andrew Kwong

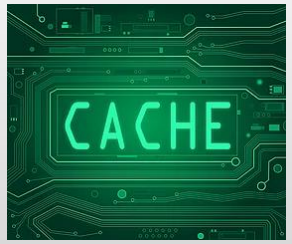
THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

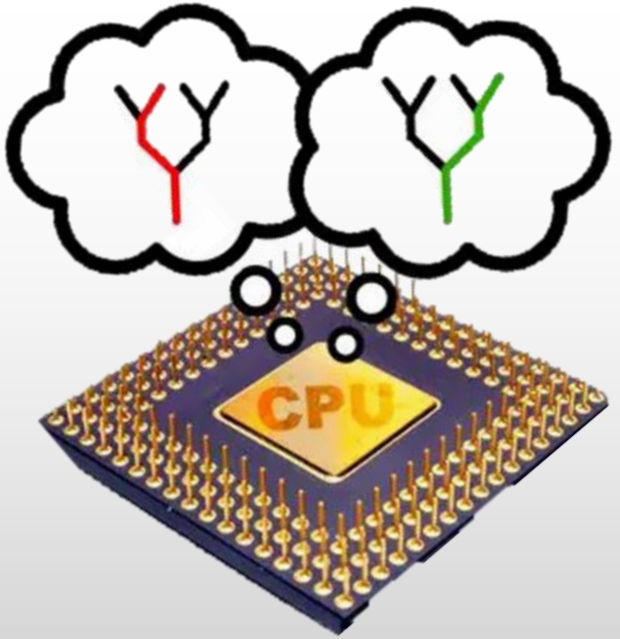




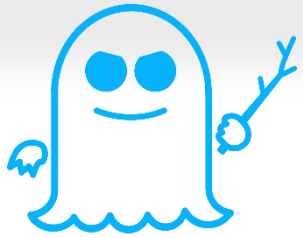


Malware





Hardware abstractions leak secrets to software!



SPECTRE

MELTDOWN

Forbes

Massive Intel Vulnerabilities Just Landed -- And Every PC User On The Planet May Need To Update

CNBC

Amazon, Microsoft, and Google respond to Intel chip vulnerability

CacheOut

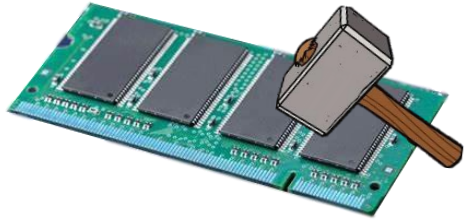
ars TECHNICA

I'M SURE THIS WON'T BE THE LAST SUCH PROBLEM —

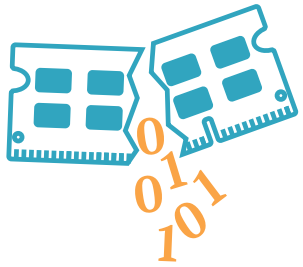
Intel's SGX blown wide open by, you guessed it, a speculative execution attack

WIRED

Forget Software—Now Hackers Are Exploiting Physics



Rowhammer



RAMBleed

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STOR

RAMBLEED —

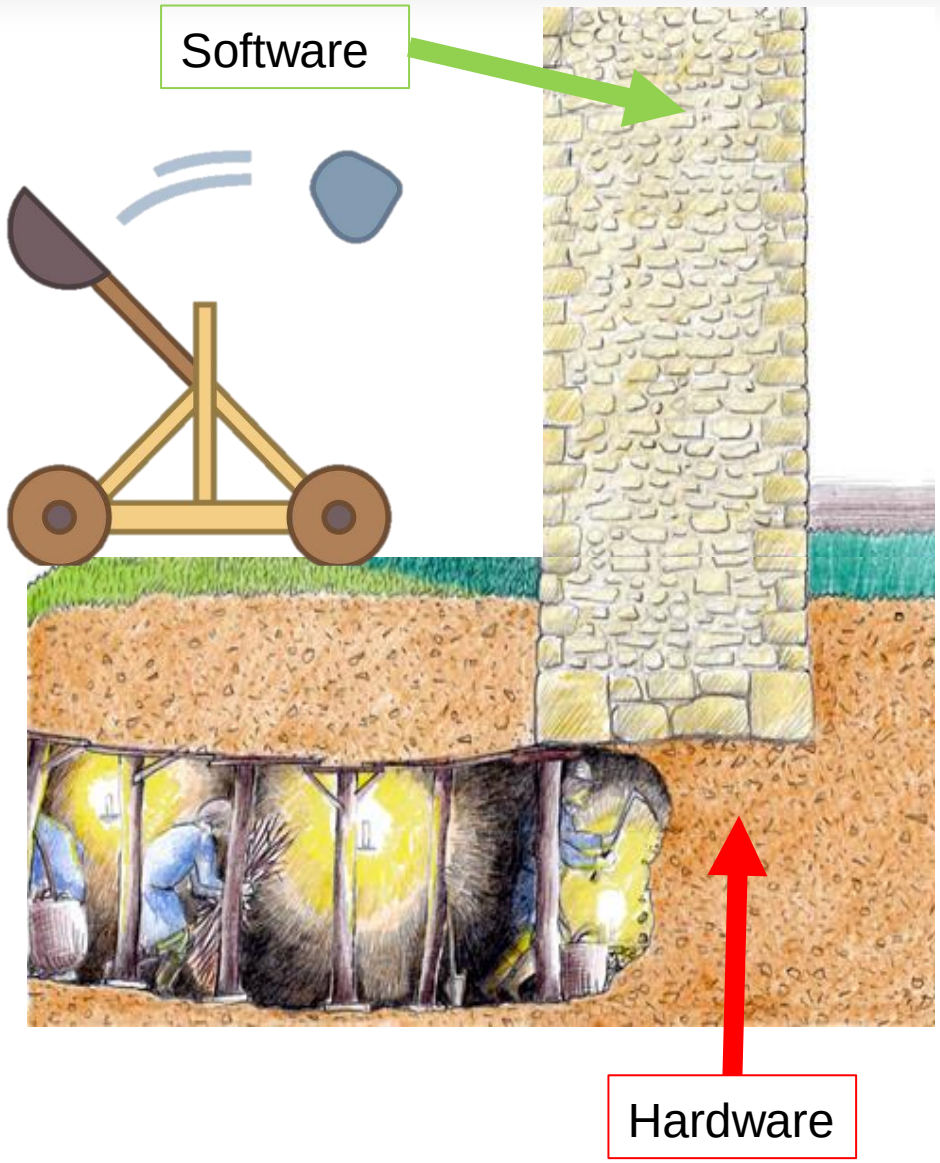
Researchers use Rowhammer bit flips to steal 2048-bit crypto key

RAMBleed side-channel attack works even when DRAM is protected by error-correcting code.

Implications of Side Channels

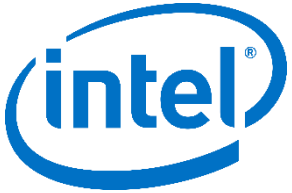
- Secure software systems must consider leakage from the hardware

- Very hard to do when we still don't even understand what these attacks are capable of



My Work

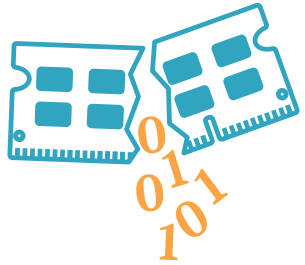
Crypto	Post-Quantum Crypto KEMs [CCS'22] Secure Entropy Generation [IEEE S&P'20] Chrome Password Check [USENIX Security'23]
Operating System/Architecture	CacheOut [IEEE S&P'21] OS Availability Attacks [IEEE S&P'18] Kernel Hammer [Current] SGAxe [Current]
DRAM	RAMBleed [IEEE S&P'20] Spectre+Rowhammer [IEEE S&P'22]
Analog	Acoustic Eavesdropping [IEEE S&P'19]



5

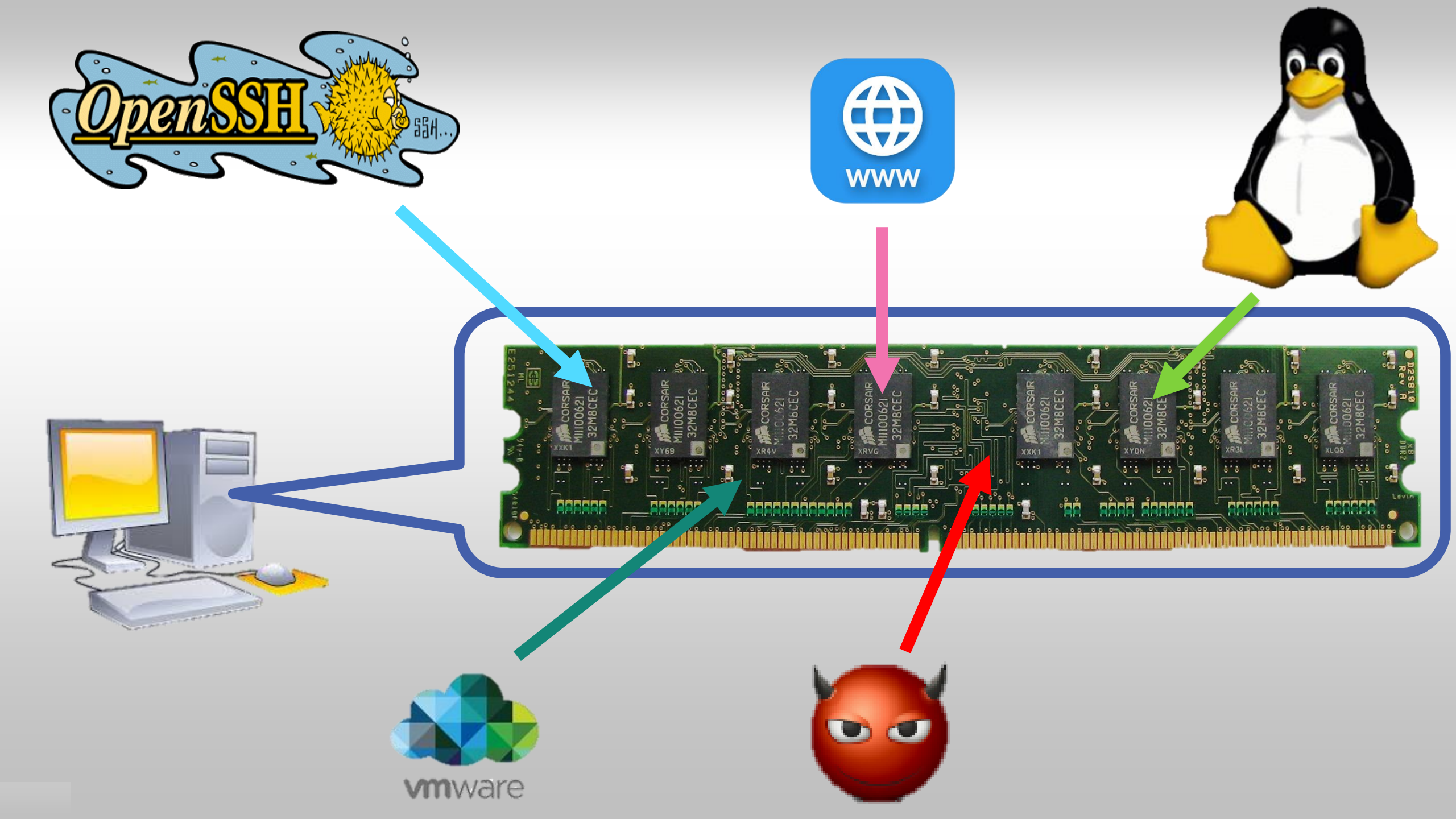
My Work

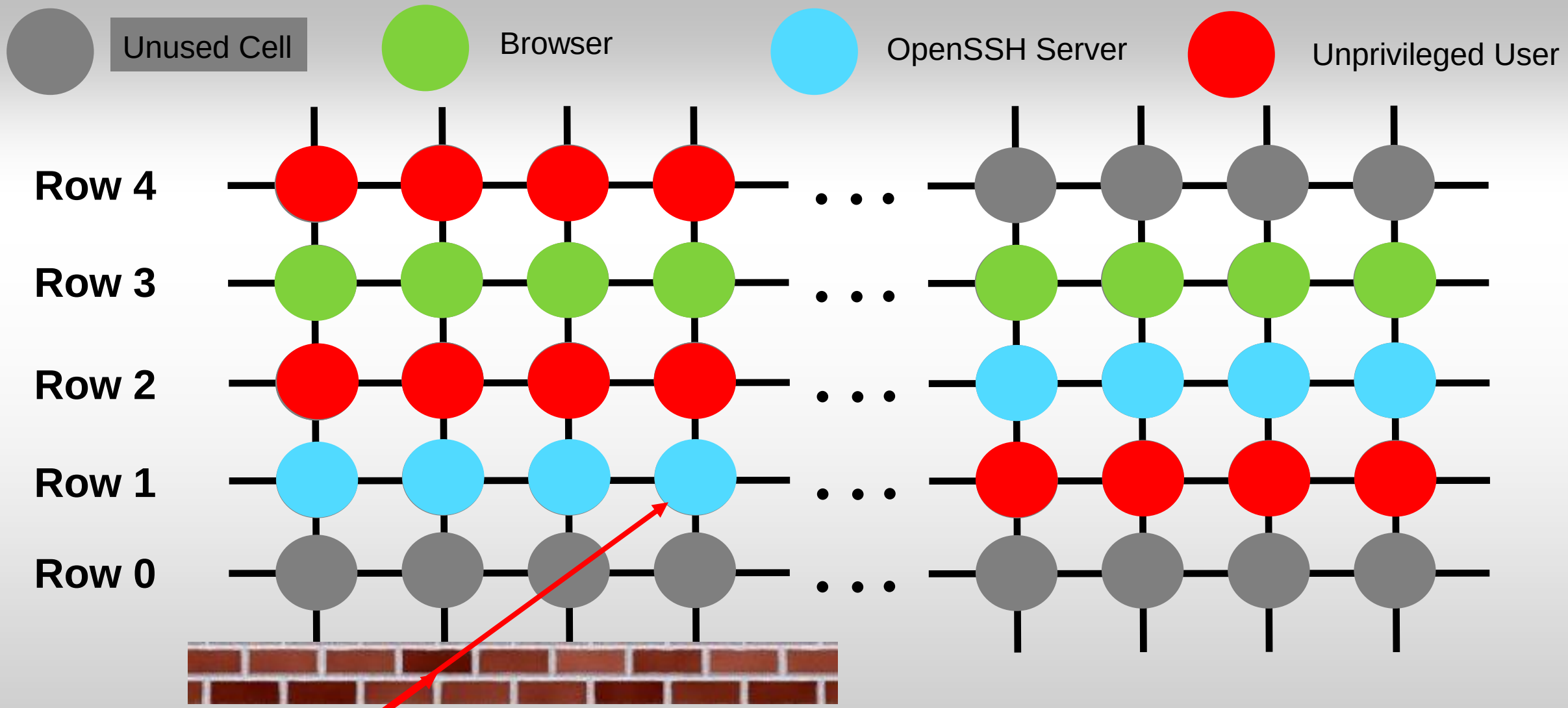
Crypto	Post-Quantum Crypto KEMs [CCS'22] Secure Entropy Generation [IEEE S&P'20] Chrome Password Check [USENIX Security'23]
Operating System/Architecture	CacheOut [IEEE S&P'21] OS Availability Attacks [IEEE S&P'18] Kernel Hammer [Current] SGAxe [Current]
DRAM	RAMBleed [IEEE S&P'20] Spectre+Rowhammer [IEEE S&P'22]
Analog	Acoustic Eavesdropping [IEEE S&P'19]



RAMBleed

6


OpenSSH
SSH...
www
vmware



- OS enforces isolation
- **Can we bypass OS and dump memory across these boundaries?**

ars TECHNICA

BIZ & IT | TECH | SCIENCE | POLICY | CARS | GAMING & CULTURE | STORE

RAMBLEED —

Researchers use Rowhammer bit flips to steal 2048-bit crypto key

RAMBleed side-channel attack works even when DRAM is protected by error-correcting code.

DAN GOODIN - 6/11/2019, 1:00 PM

REVYUH

TOP NEWS | POLITICS | BUSINESS | HARDWARE & GADGETS | LI

Home › Software › IT Security › RAMBleed: OpenSSH develops protection against sidechannel atta

IT Security

RAMBleed: OpenSSH develops protection against sidechannel attacks

By **Kamal Saini** - June 22, 2019

Log in | Sign up | Forums

Serverless

The Register®

Biting the hand that feeds IT

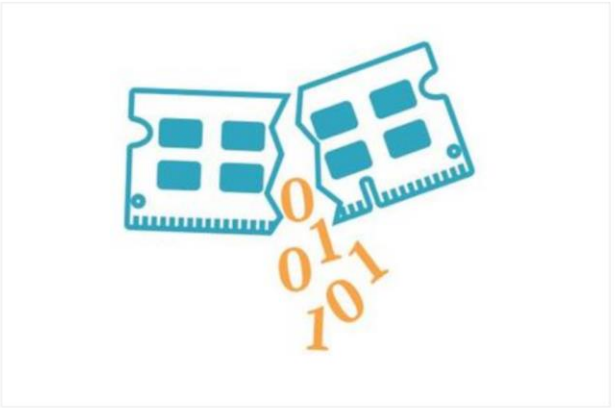
DATA CENTRE | SOFTWARE | SECURITY | DEVOPS | BUSINESS | PERSONAL TECH | SCIENCE | E

Security

RAMBleed picks up Rowhammer, smashes DRAM until it leaks apps' crypto-keys, passwords, other secrets

Boffins blast boards to boost bits

By Thomas Claburn in San Francisco 11 Jun 2019 at 22:26 31 SHARE ▼



ZDNet

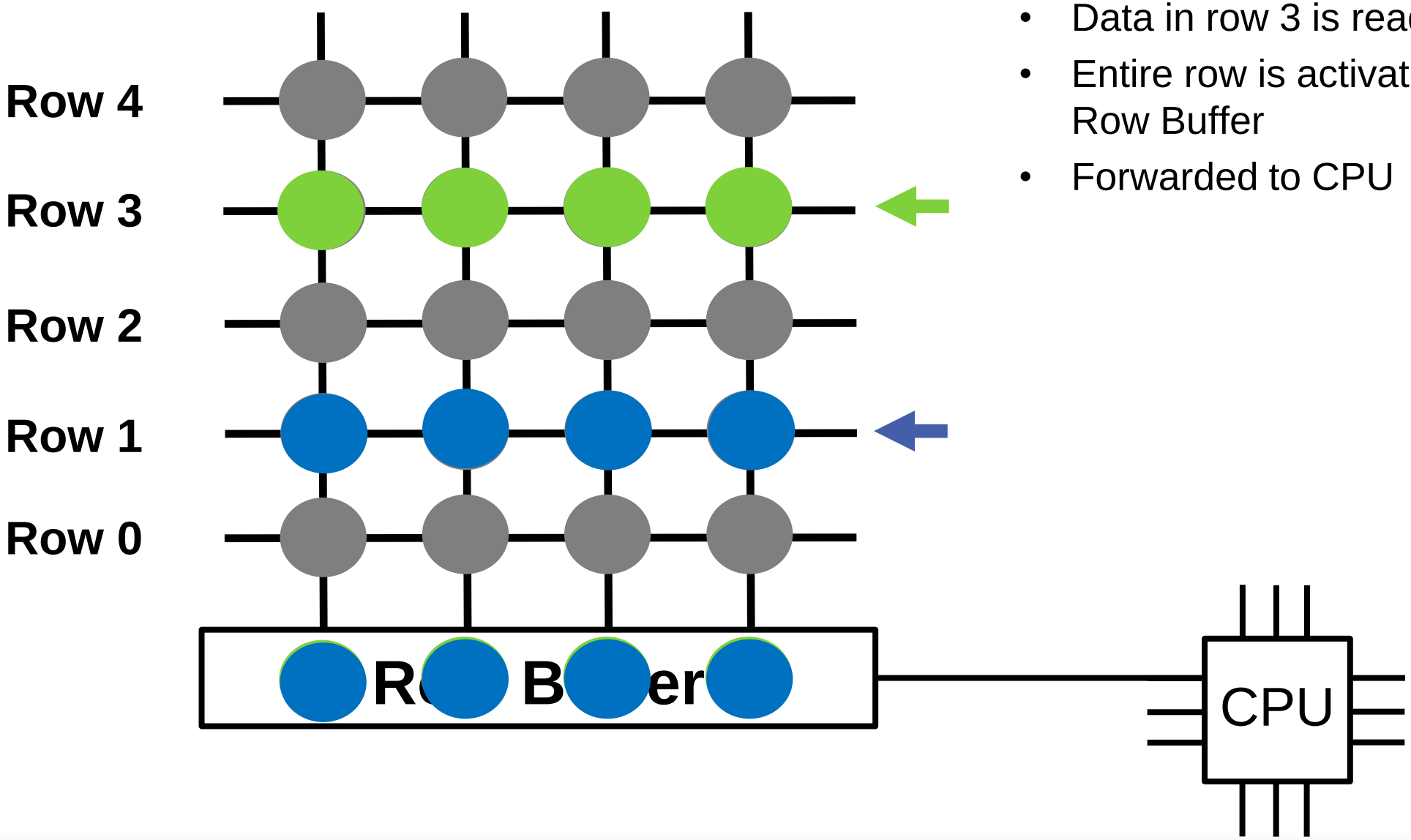
VIDEOS | 5G | WIND

'RAMBleed' Rowhammer attack can now steal data, not just alter it

Academics detail new Rowhammer attack named RAMBleed.

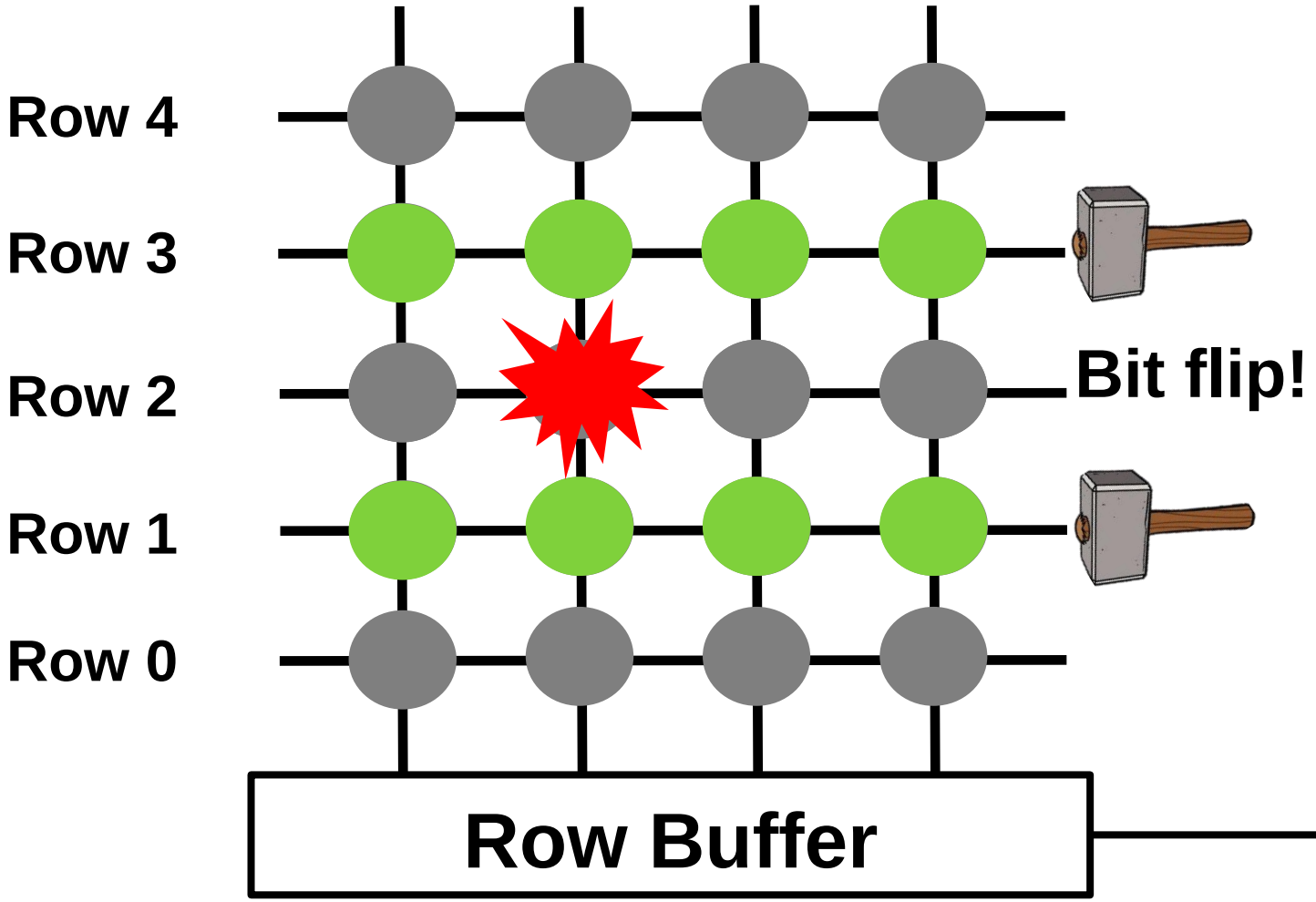
By Catalin Cimpanu for Zero Day | June 11, 2019 -- 17:00 GMT (10:00 PDT) | Topic: Security

How DRAM works

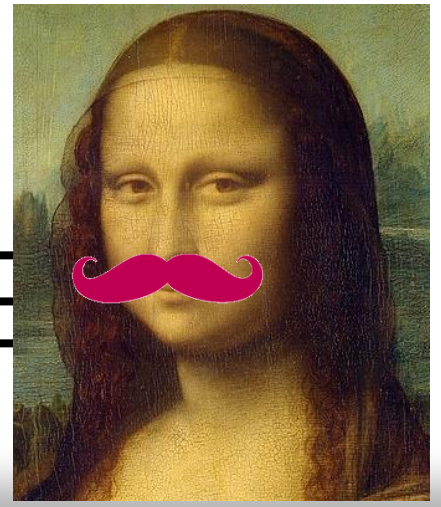


- Data in row 3 is read
- Entire row is activated and stored in Row Buffer
- Forwarded to CPU

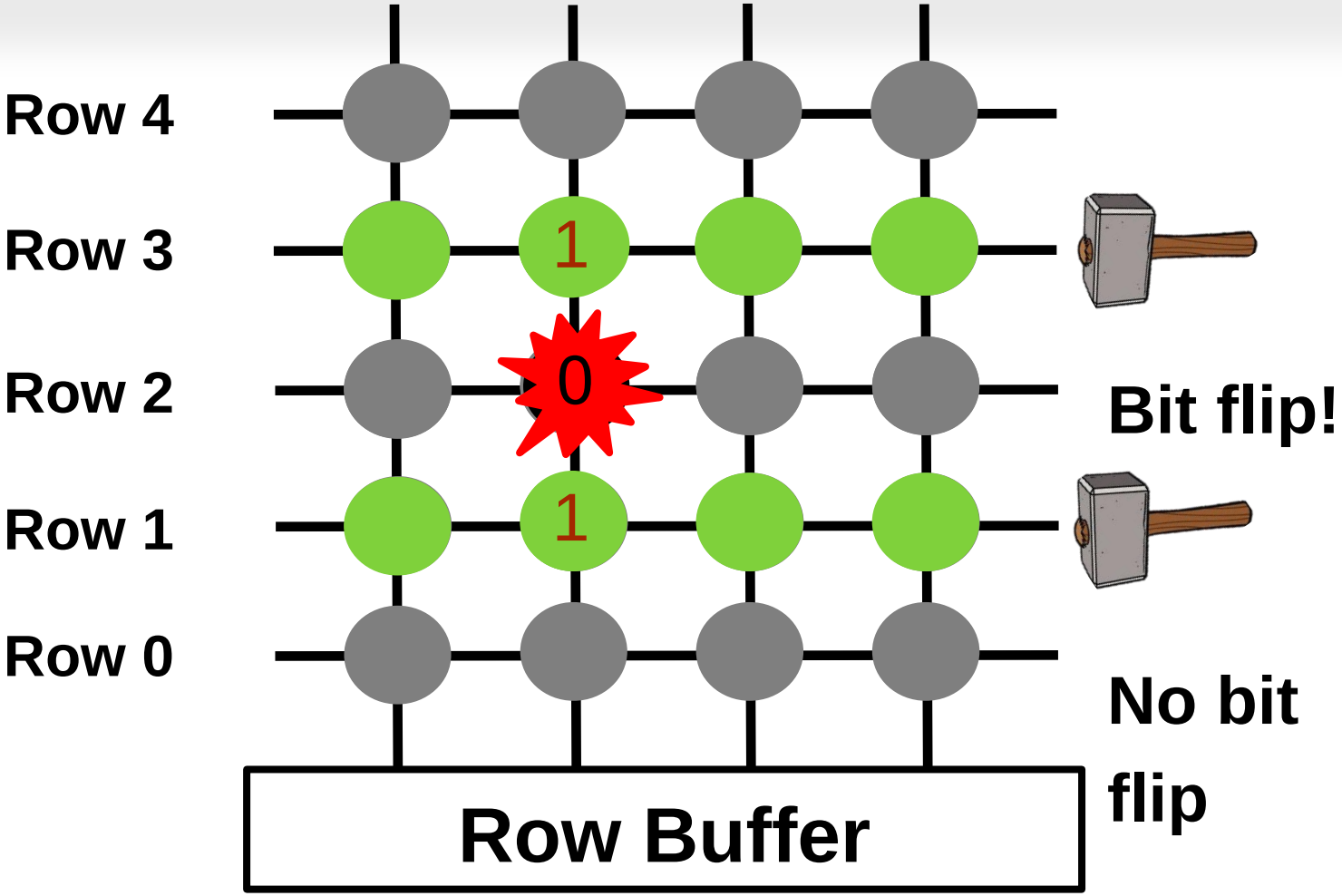
Rowhammer for Writing



- Activating a row drains charge from nearby capacitors
- Repeated activation of rows causes bit flips in nearby rows!
- Attacker that controls values in rows 1 and 3 writes to victim's memory in row 2
- Rowhammer can be used to **write** across security domains
- RAMBleed aims to **read** data from DRAM



Data Dependent Bit Flips



- Detecting bit flips in row 2 reveals the likely values in rows 1 and 3
- Data from rows 3 and 1 “bleed” into row 2
- How can we use this effect to read something useful

- Stripe Pattern: 0-1-0 = more likely to flip
- Uniform Pattern: 1-1-1 = less likely to flip

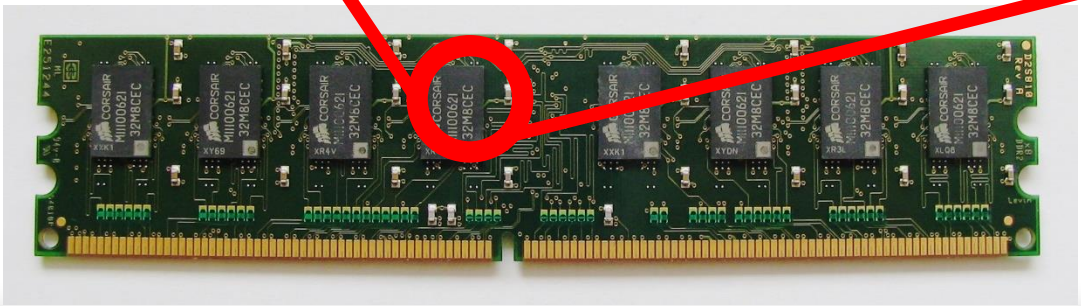
Memory Layout on DIMM

Attacker

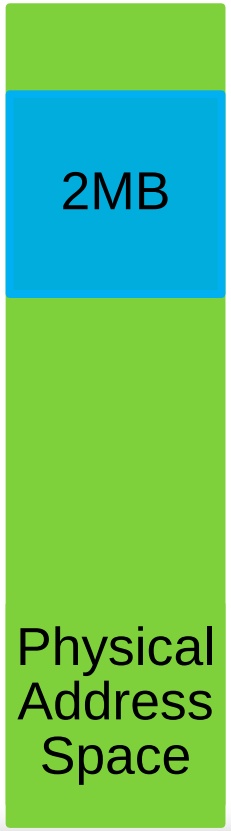
SSH Daemon

Row 2	Row Activation Page (A0)	Target Page (T0)
Row 1	Unused (R)	Sampling Page (A1)
Row 0	Row Activation Page (A2)	Target Page (T1)

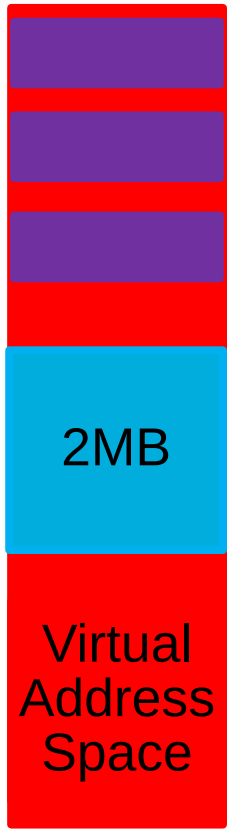
8KiB



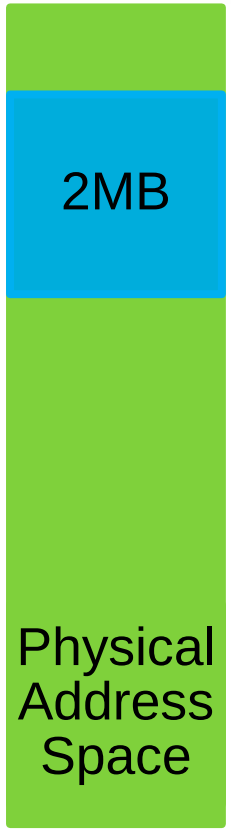
- How does an unprivileged user process locate and acquire this memory layout?
- Developed attack on Linux's memory allocator



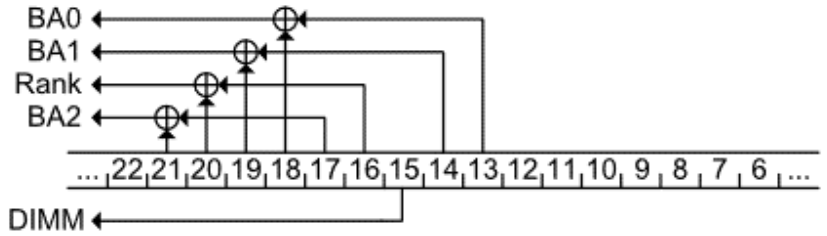
Memory Massaging

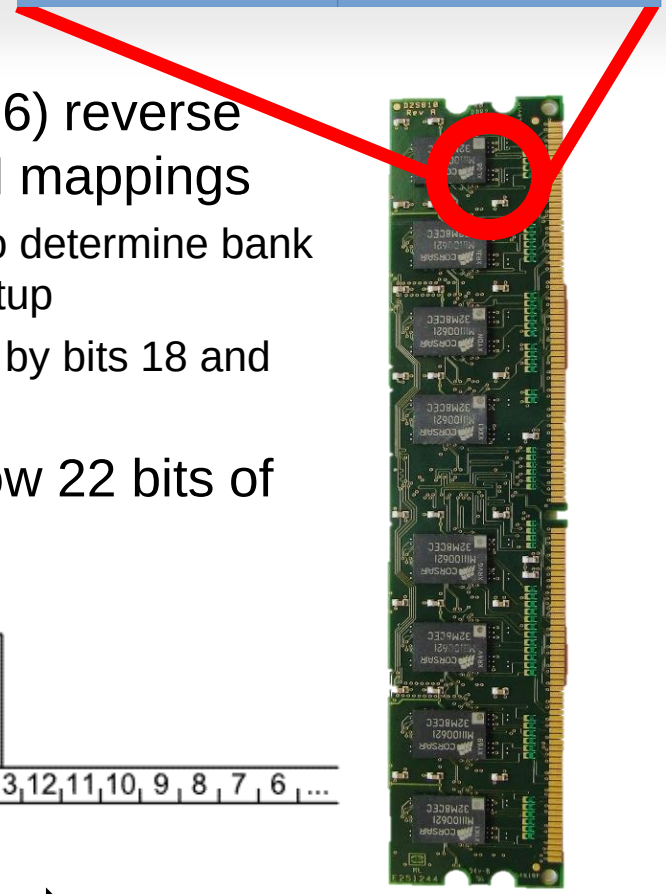


- Need to obtain 2MB physically sequential memory from unprivileged user
- /proc/pagetypeinfo
  - Lists number of available blocks of memory for each size
  - Was world readable at the time of publication



- DRAMA (Usenix`16) reverse engineered DRAM mappings
  - Uses low 22 bits to determine bank on our memory setup
  - Row index formed by bits 18 and above
- Suffices to learn low 22 bits of physical address



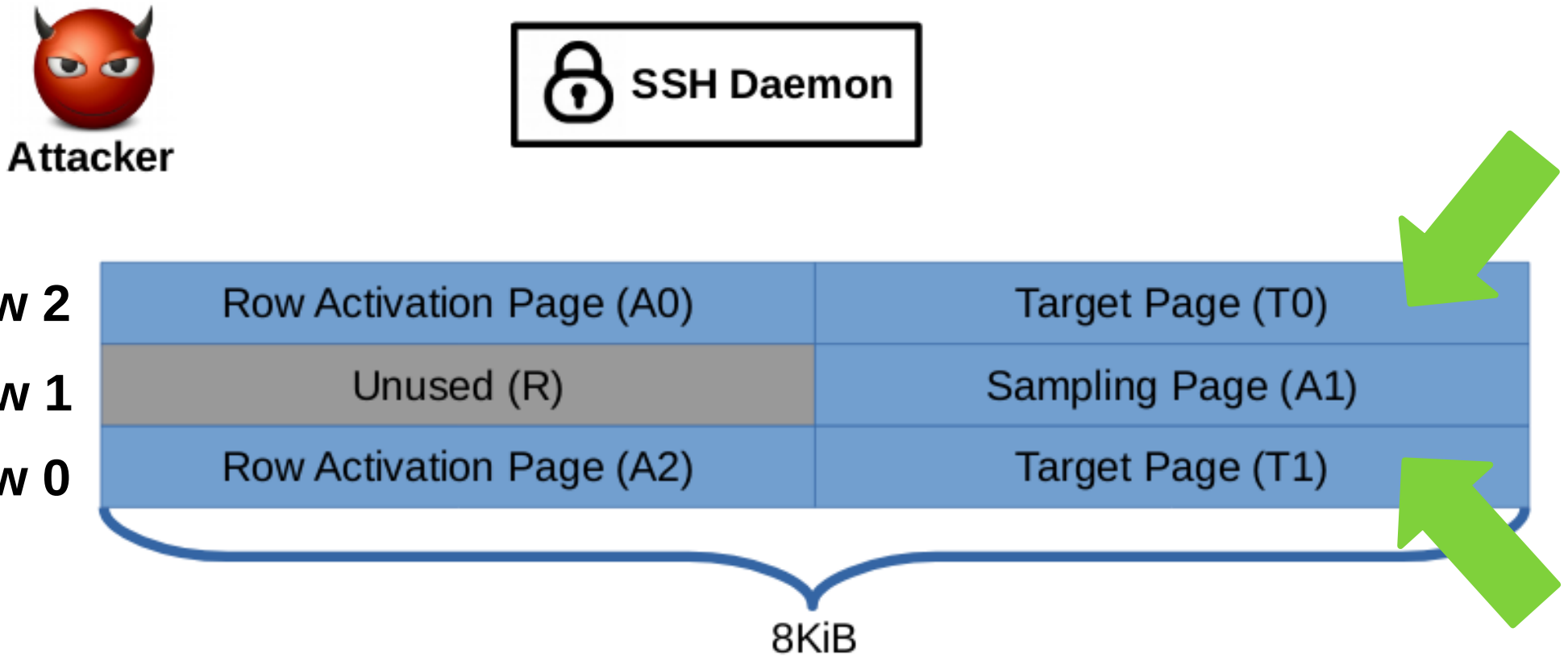


- Precise method for ining the bits is bed in the paper

Demonstrated that memory allocator state is security sensitive

Linux Restricted Access Afterwards

Memory Layout on DIMM



- Attacker's memory in desired locations:
- Want SSH server's RSA keys to land in T0 and T1
- Developed "Frame Feng Shui" to place victim's pages in frames of attacker's choosing

Frame Feng Shui

Row Activation Page (A0) | Target Page (T0)

Unused (R) | Sampling Page (A1)

Row Activation Page (A2) | Target Page (T1)

X

Y

Victim Pseudo Code

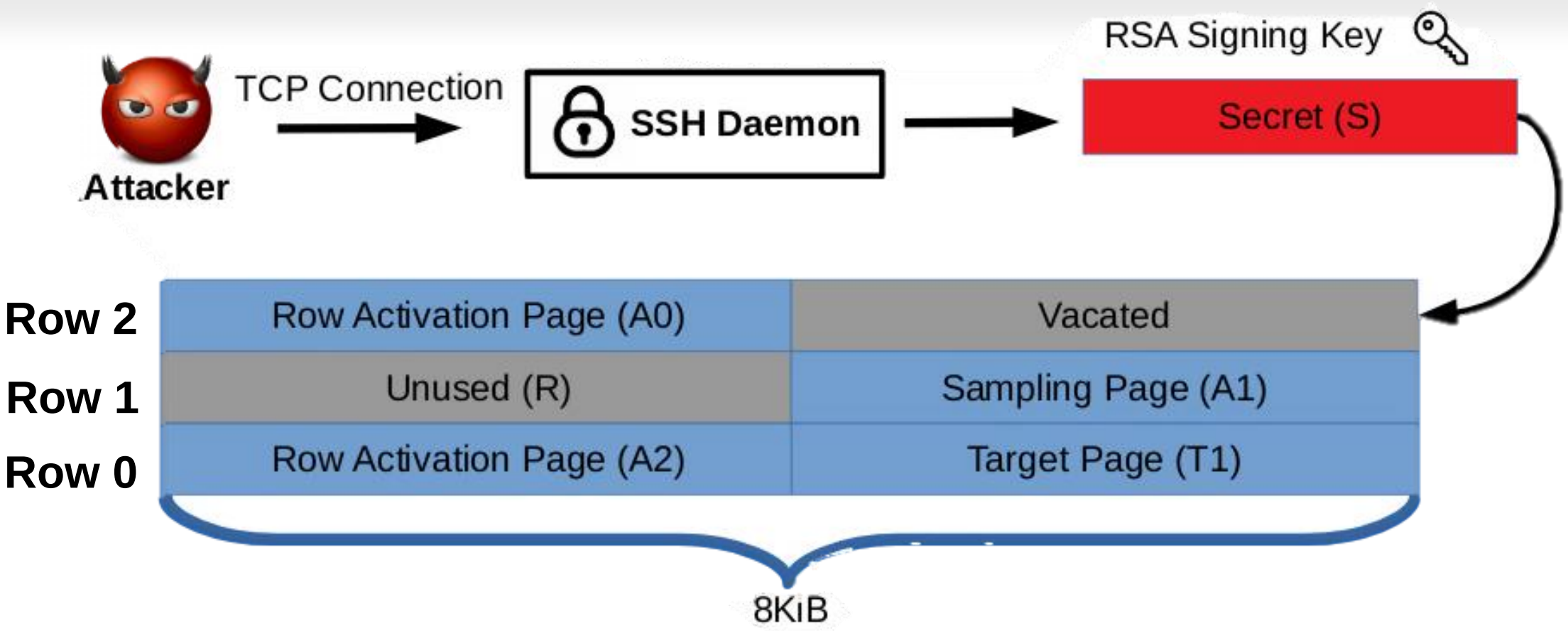
```
alloc(buffer0);
alloc(buffer1);

alloc(secret);
```

Attacker's Frames

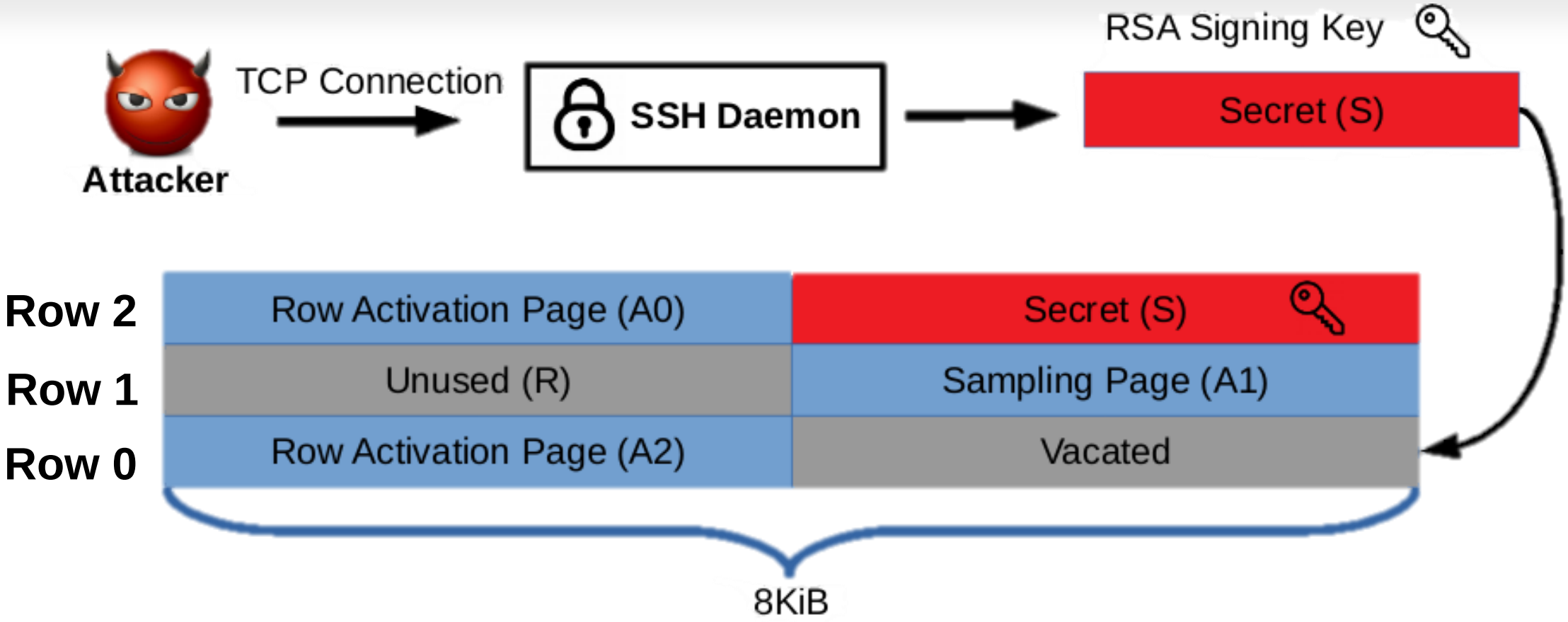
Page Frame Cache

Victim's Frames

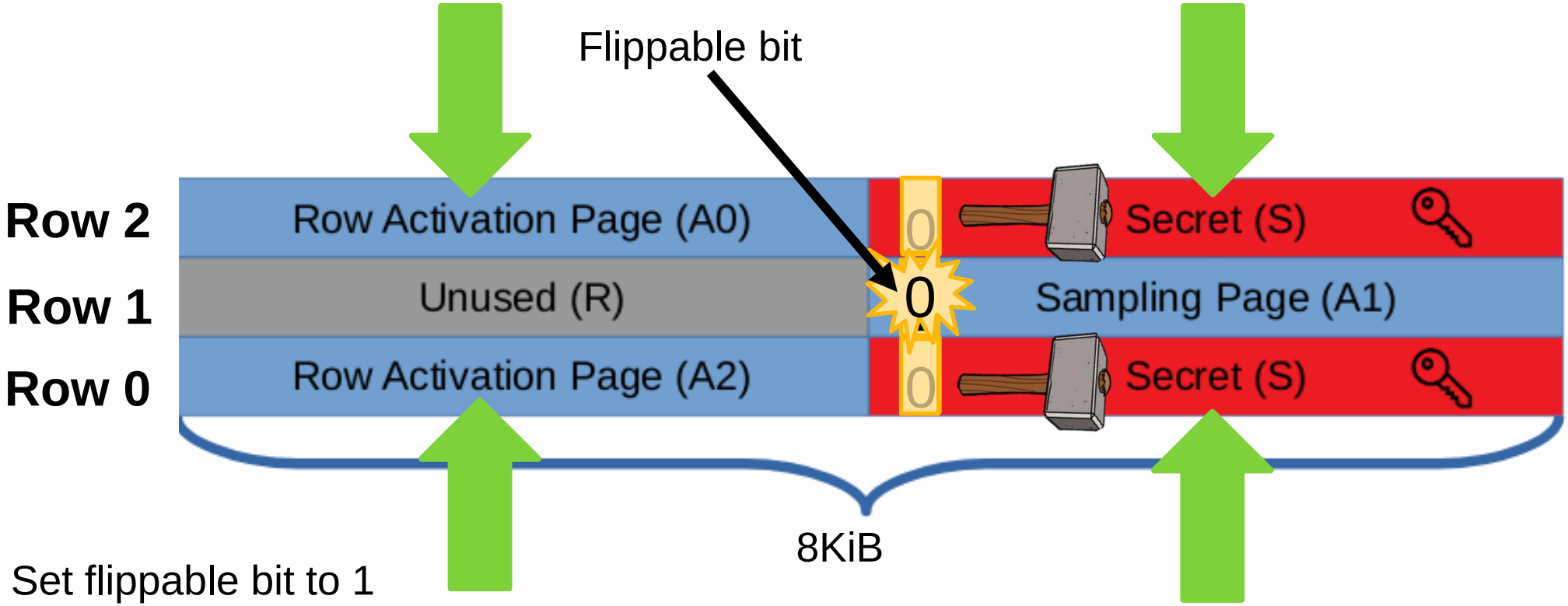
Stack



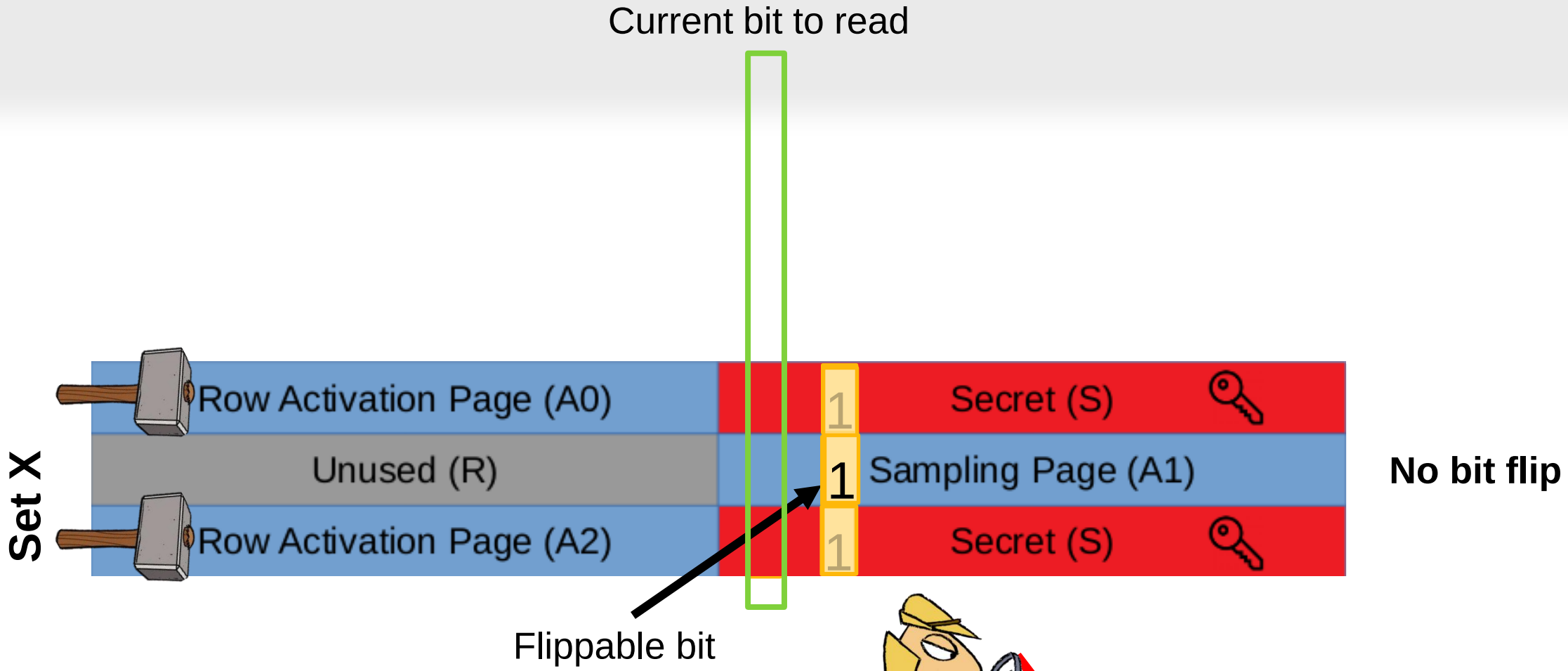
- Attacker’s releases T0
- Victim’s Key placed in desired location:



- Attacker's memory in desired locations: ✓
- Victim's Key placed in desired location: ✓



- Set flippable bit to 1
  - Try to read bit above and below
- If it flips, secret bit is 0!
  - Was a 0-1-0 stripe pattern before the flip
- Accessing data in A0 activates the cells in the Secret's page

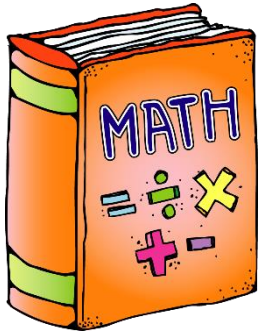


- Reading the next bit:
  - Search memory for a bit flip at desired offset
  - Repeat Frame Feng Shui, Hammer, and read another bit
- Repeat for all bits

Results

- Read 68% of a 2048 bit RSA key from OpenSSH server
  - Read from DRAM at 0.31 bits/second
  - 82% accuracy

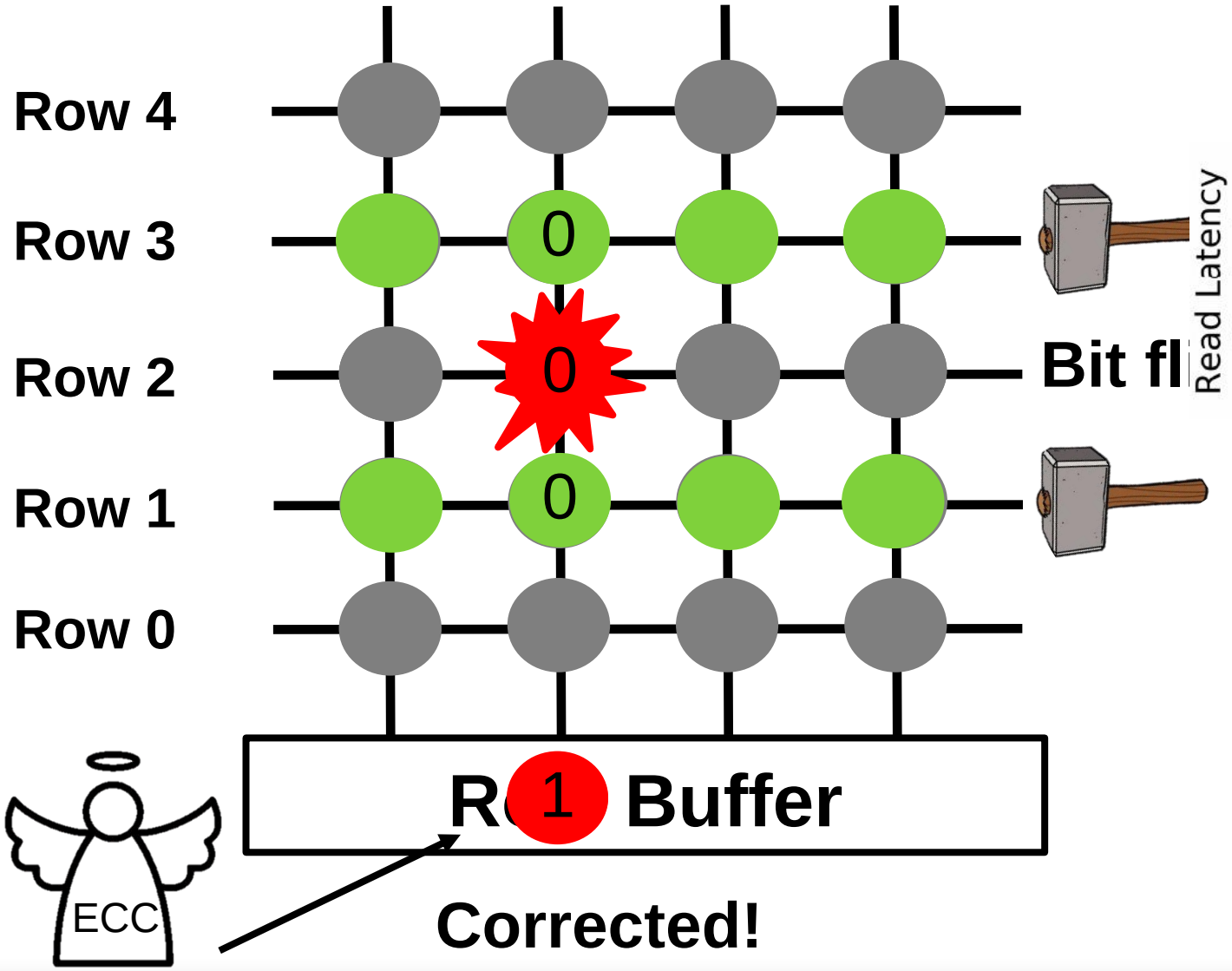
- Read out sufficient bits for full key recovery in a couple of hours
  - With some math tricks



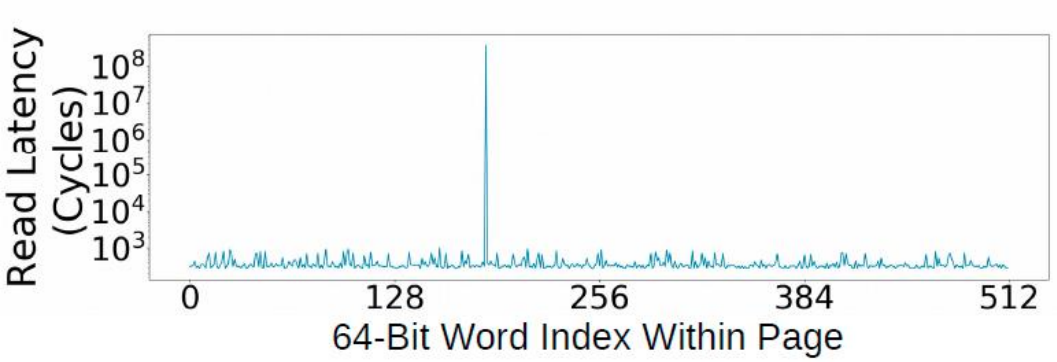
What About Servers?

?

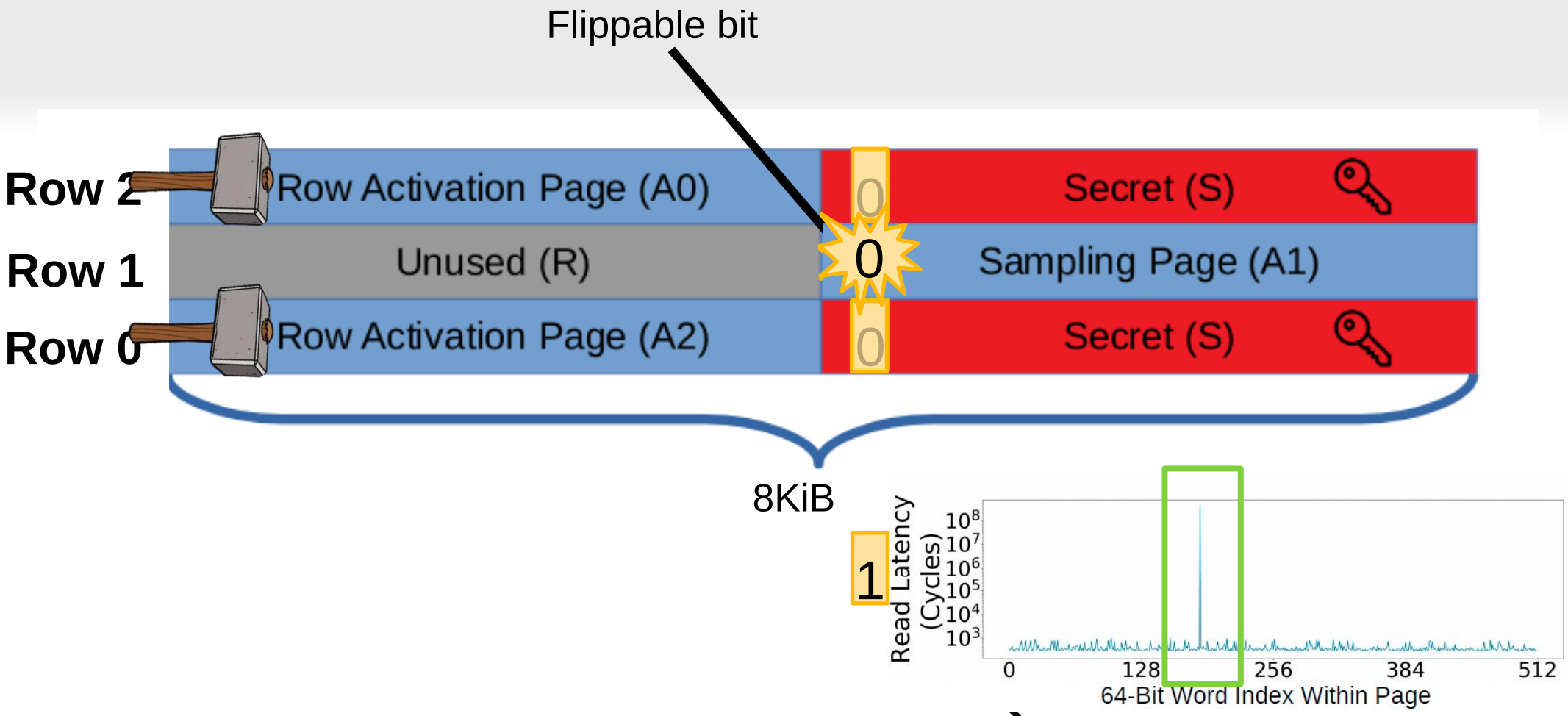
RAMBleed on ECC Memory



- Error-Correcting Code Memory:
  - Corrects corrupted data words when when read back



- Data words with errors have a **much** longer read latency
- RAMBleed only requires the detection of bit flips

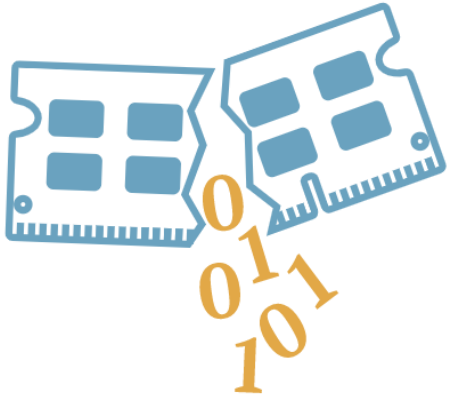


- Set flippable bit to 1
- Hammering flips it to zero
  - ECC corrects the bit flip so it always returns a 1
  - Large read latency indicates bit flip

Impact

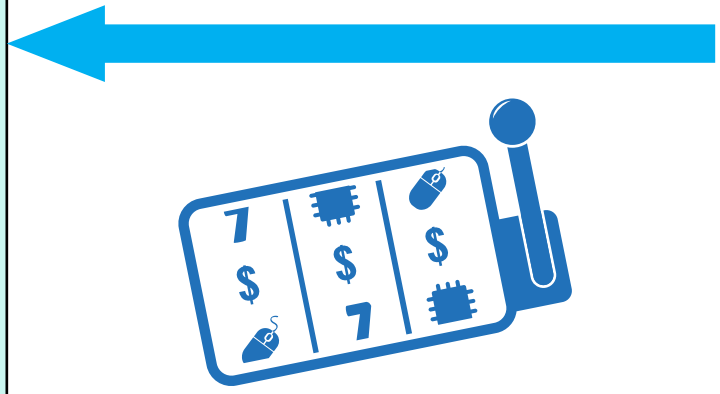
- Rowhammer no longer restricted to integrity
  - Confidentiality ✖
- Works against server ECC memory
- DRAM is shared by everything
  - Problem extends beyond crypto keys
- Mitigations deployed by OpenSSH and Linux

- Rowhammer discovered in 2014
  - Defenses built without even understanding that Rowhammer affects confidentiality!



My Work

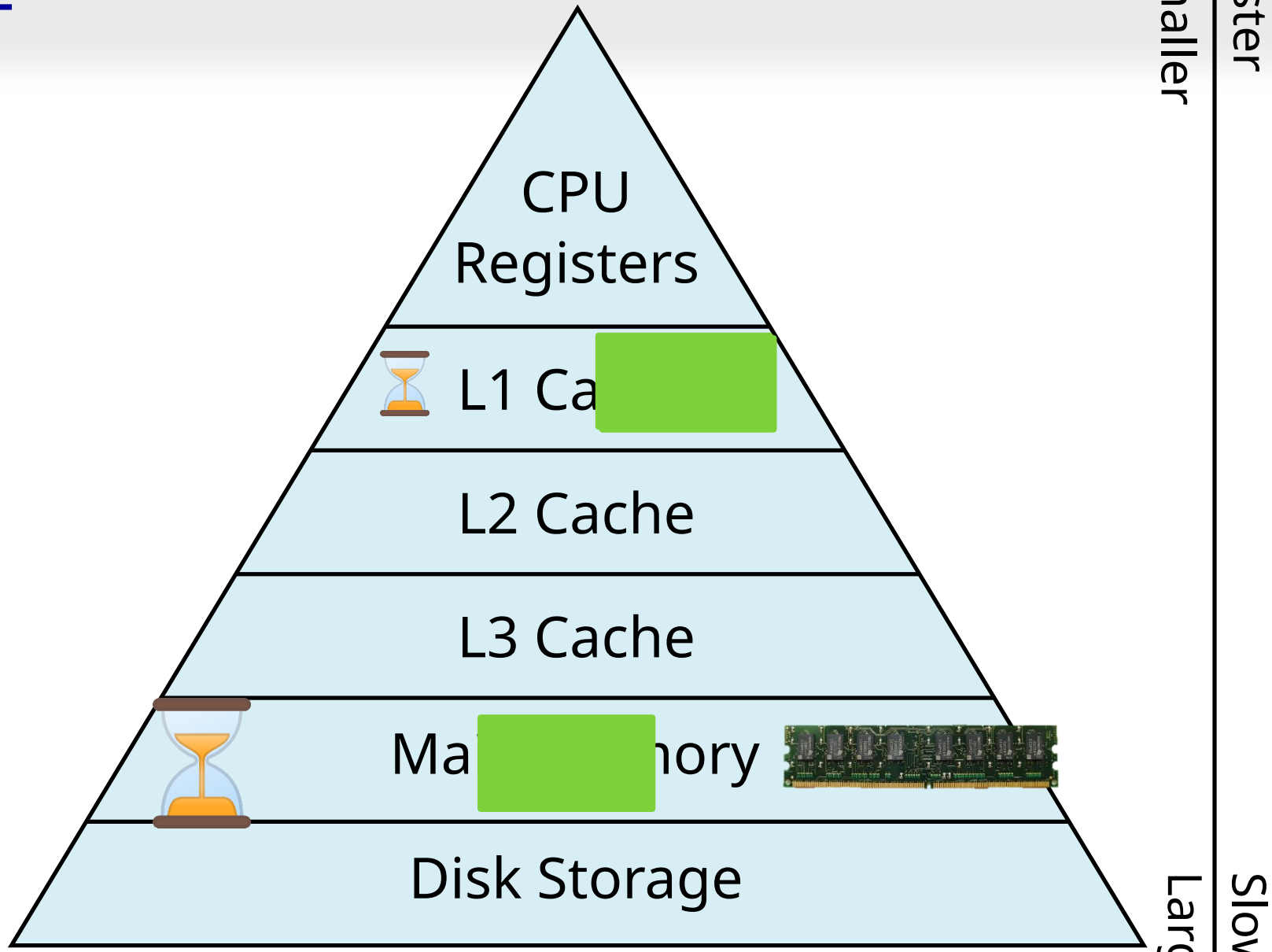
Crypto	Post-Quantum Crypto KEMs [CCS'22] Secure Entropy Generation [IEEE S&P'20] Chrome Password Check [USENIX Security'23]
Operating System/Architecture	CacheOut [IEEE S&P'21] OS Availability Attacks [IEEE S&P'18] Kernel Hammer [Current] SGAxe [Current]
DRAM	RAMBleed [IEEE S&P'20] Spectre+Rowhammer [IEEE S&P'22]
Analog	Acoustic Eavesdropping [IEEE S&P'19]



CacheOut

26

Caching 101



27

- Caches are a great performance optimization
- Shared across security domains
  - Kernel
  - Javascript running in browser
- Can also be a source of side-channel leakage!
  - Timing difference reveals presence in cache

Cache Side-Channel

Attacker

Cache Lines

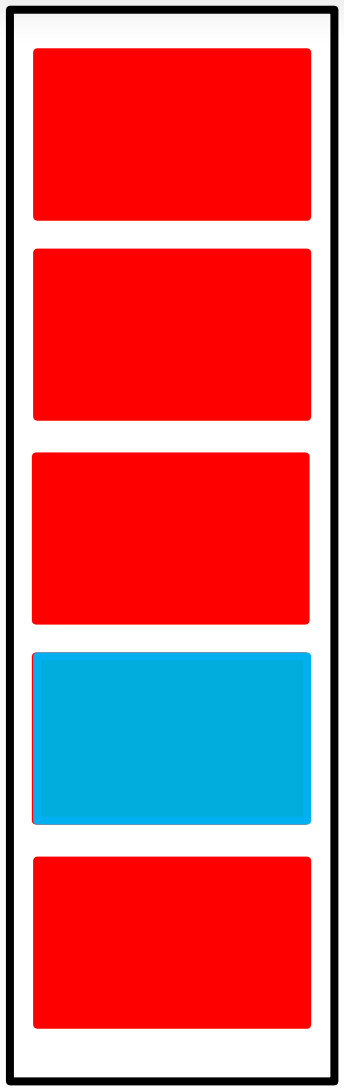
Read (Fast)

Read (Fast)

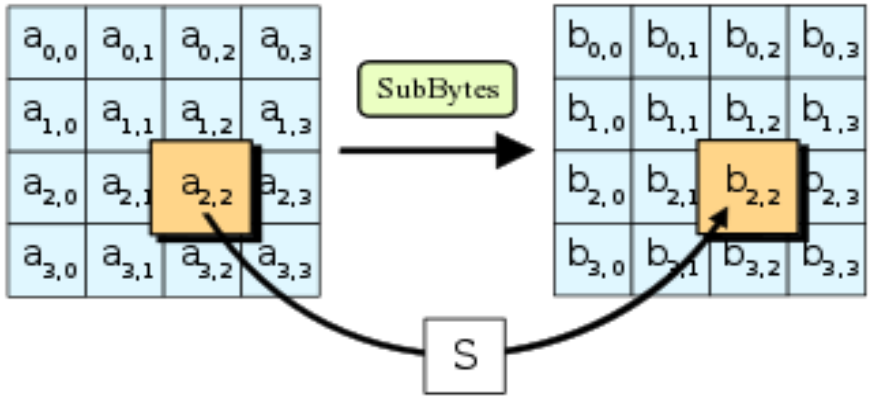
Read (Fast)

Read (Slow)

Read (Fast)



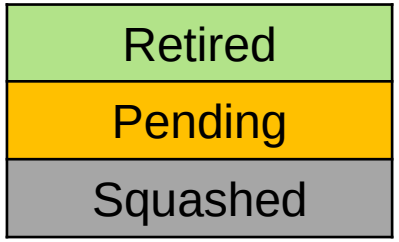
Victim



Attacker learns Victim's memory access pattern!

Speculative Execution

```
char probe[256*4096]
clflush(ptr);
secret = *ptr;
y = secret * 4096;
x = probe[y];
```



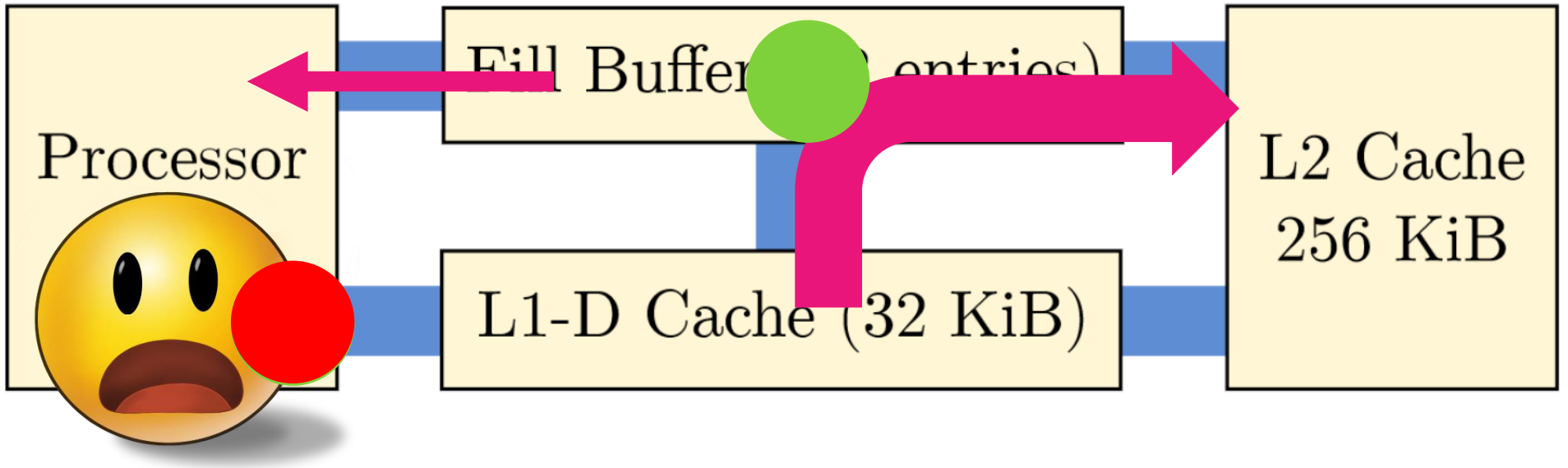
Speculative Execution

```
char probe[256*4096]
clflush(ptr);
secret = *ptr;
y = secret * 4096;
x = probe[y];
```

Speculatively Executed

Speculative Execution leaves footprints in the cache!

Retired
Pending
Squashed



Attacker speculatively reads across security domains!

Speculative Execution

```
char probe[256*4096]
clflush(ptr);
evict_to_lfb();
secret = *ptr;
y = secret * 64;
x = probe[y];
```

secret=3

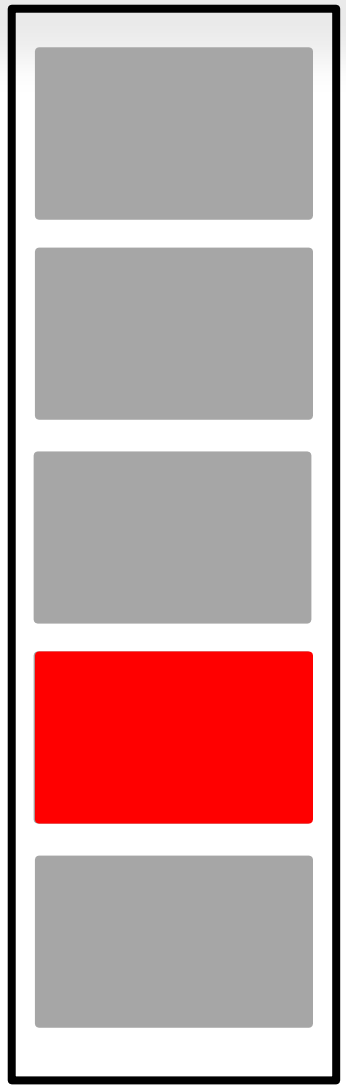
probe[0*64]

probe[1*64]

probe[2*64]

probe[3*64]

probe[4*64]



Retired
Pending
Squashed

Attacker learns secret byte!

CacheOut in the News

Intel promises fix after researchers reveal 'CacheOut' CPU flaws

New 'CacheOut' Attack Targets Intel CPUs

New 'CacheOut' attack targets Intel processors, with a fix arriving soon

SGAxe and CrossTalk flaws in Intel CPUs could enable attackers to steal data, researchers say

SGX hardware encryption technology was launched in 2015 with the Skylake microarchitecture

Plundering of crypto keys from ultrasecure SGX sends Intel scrambling again

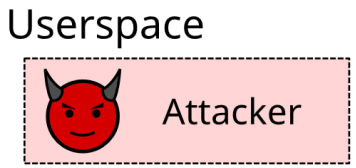
Researchers disclose new CacheOut attack that targets Intel processors

New 'CacheOut' Attack Leaks Data from Intel CPUs, VMs and SGX Enclave

CacheOut also made a big splash in the news

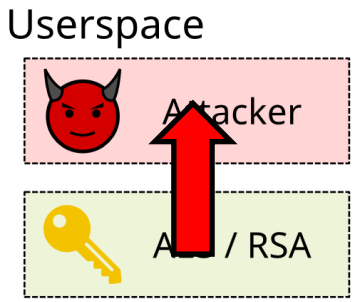
34

CacheOut Examples



35

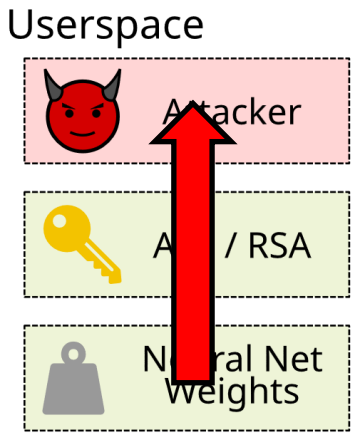
CacheOut Examples



AES and RSA keys

36

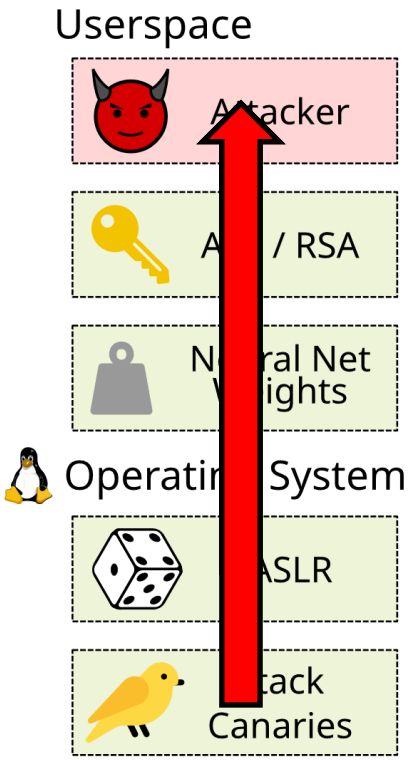
CacheOut Examples



Neural network weights

37

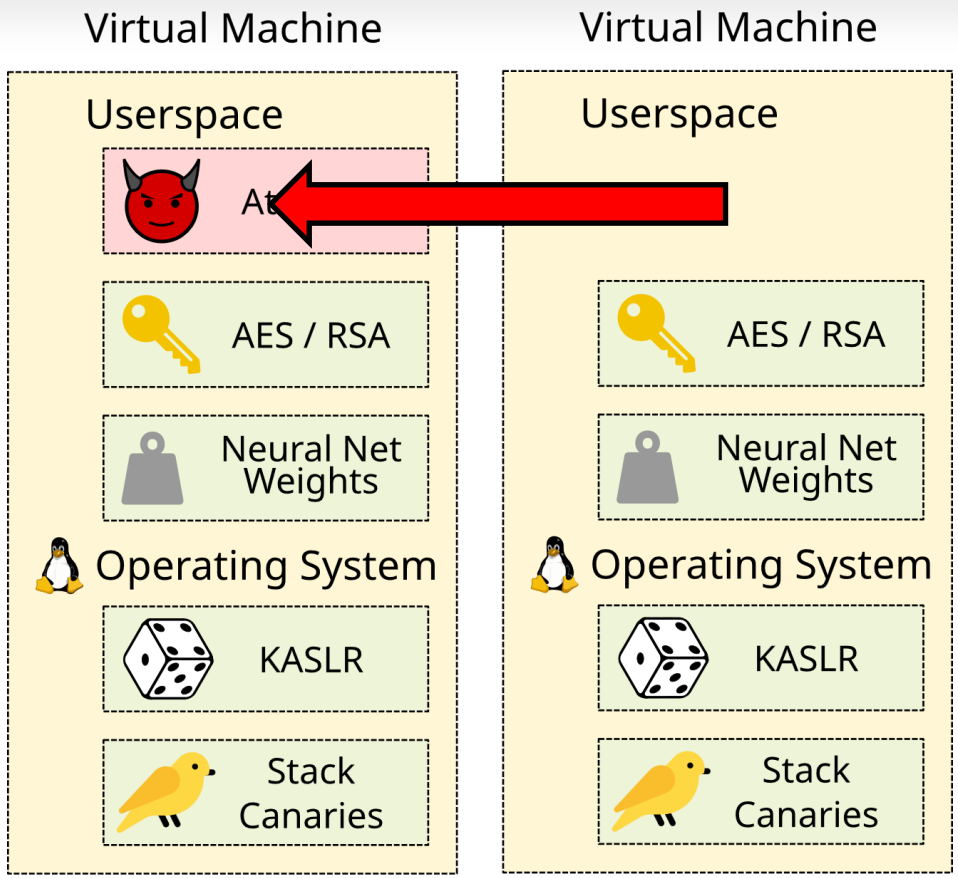
CacheOut Examples



KASLR and kernel stack canaries

38

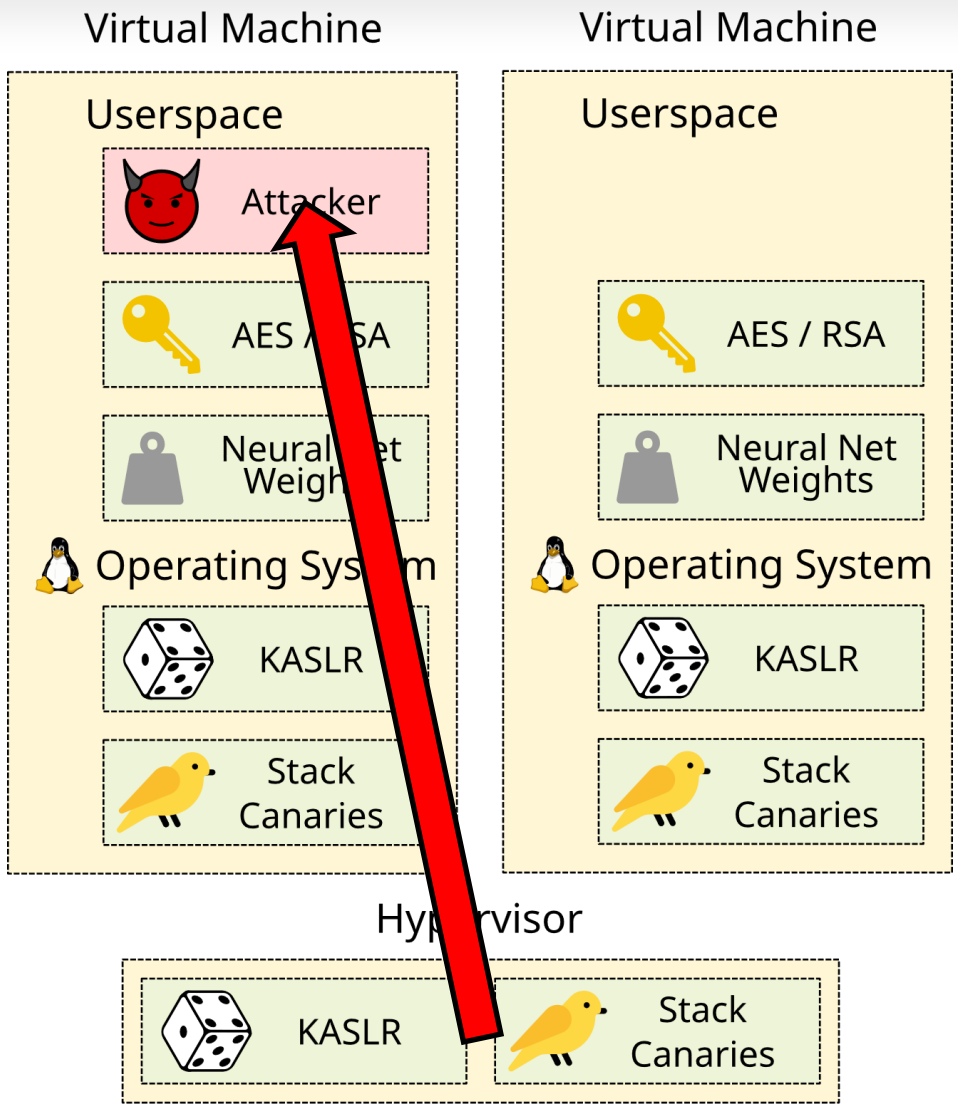
CacheOut Examples



This works across VMs

39

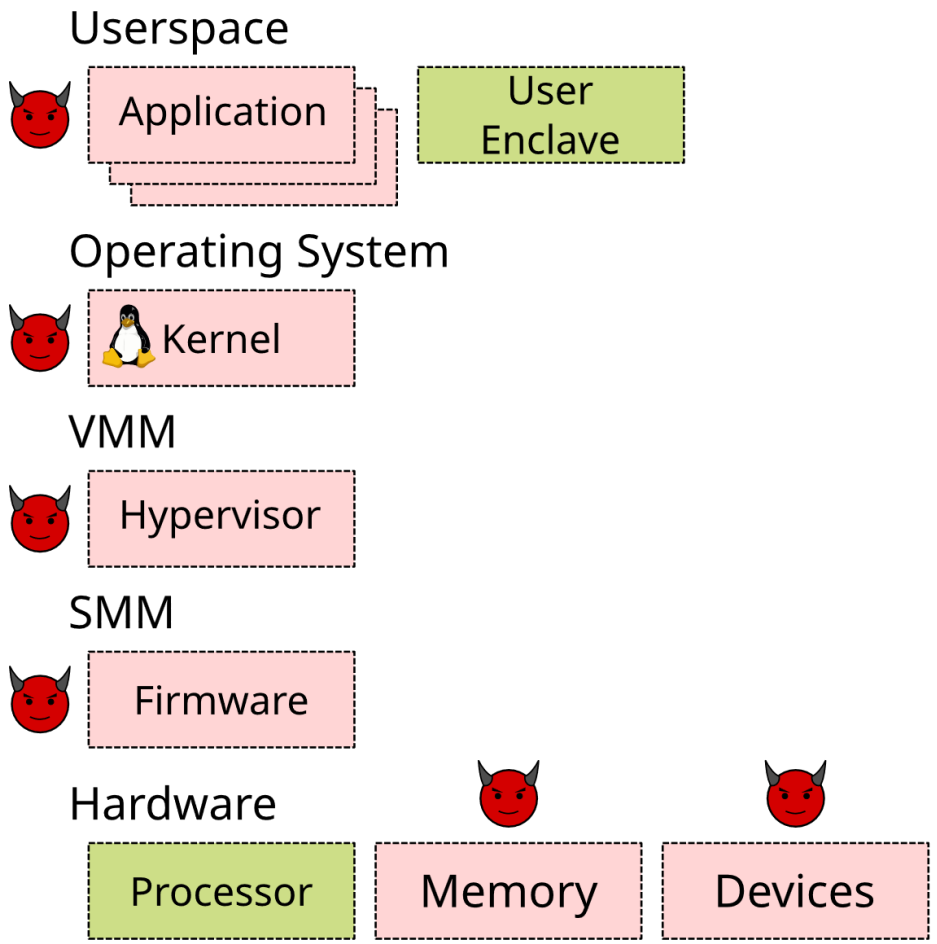
CacheOut Examples



We can even target the hypervisor!

40

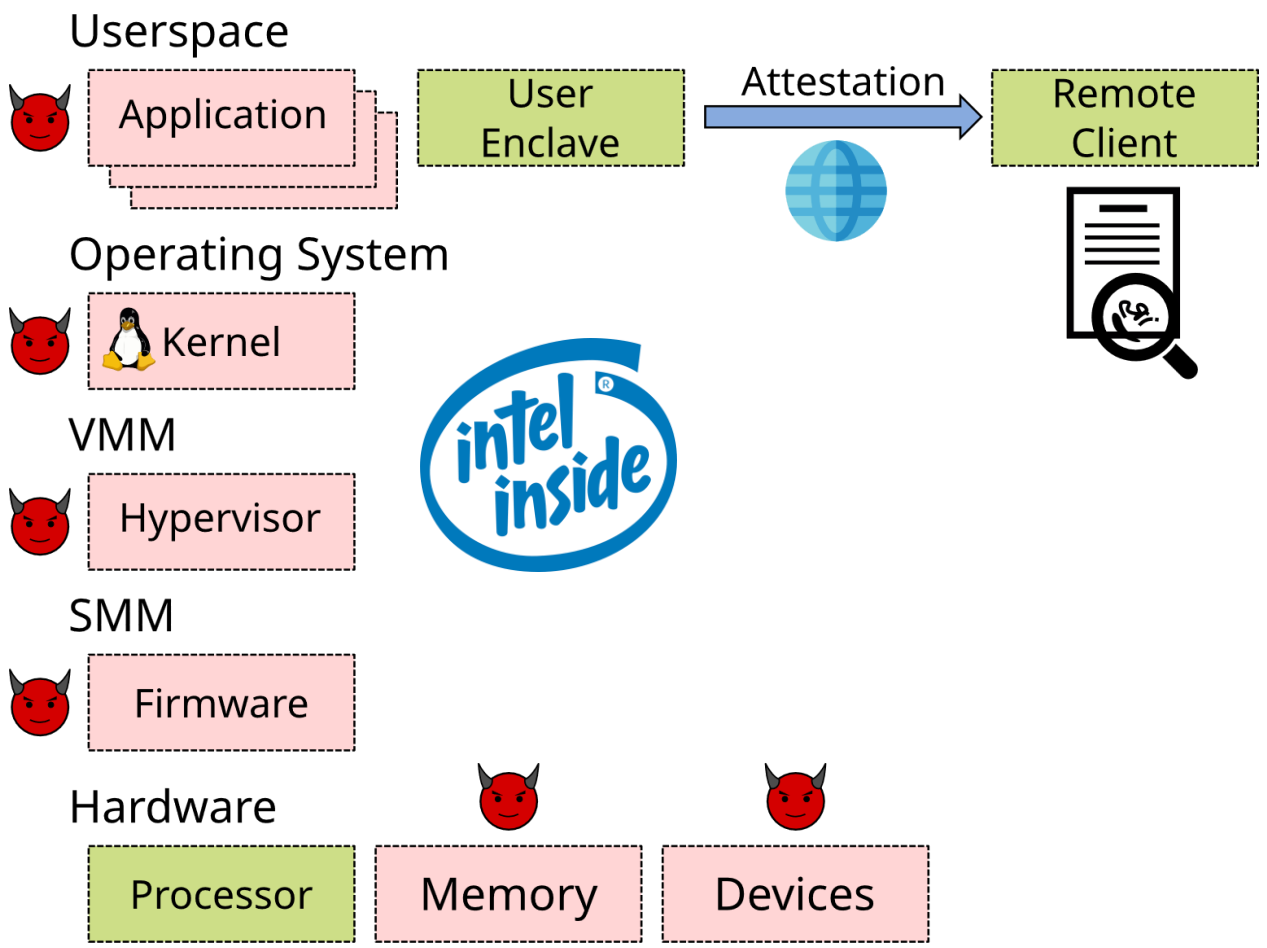
Intel Software Guard eXtensions



Intel SGX allows developers to partition code into enclaves

41

Intel Software Guard eXtensions



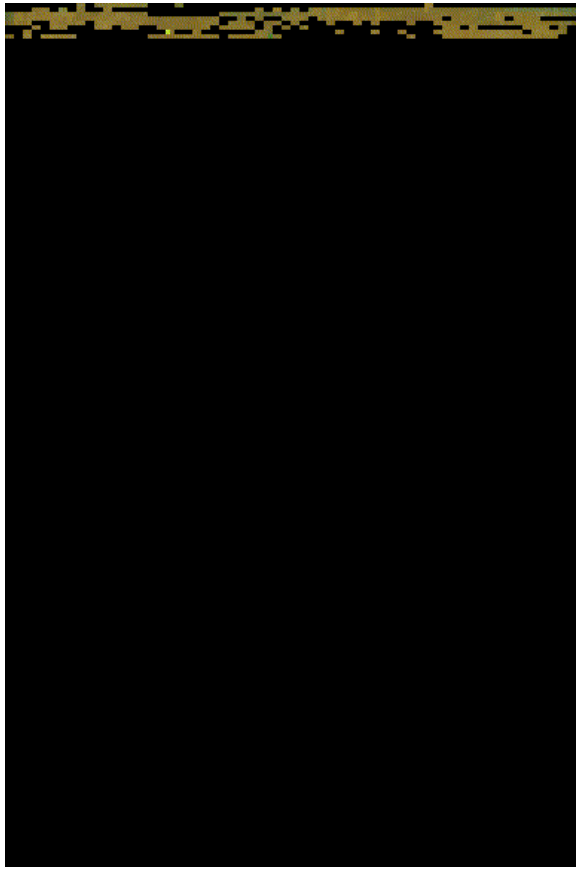
Remote Attestation proves system is genuine

42

Dumping Enclave Memory

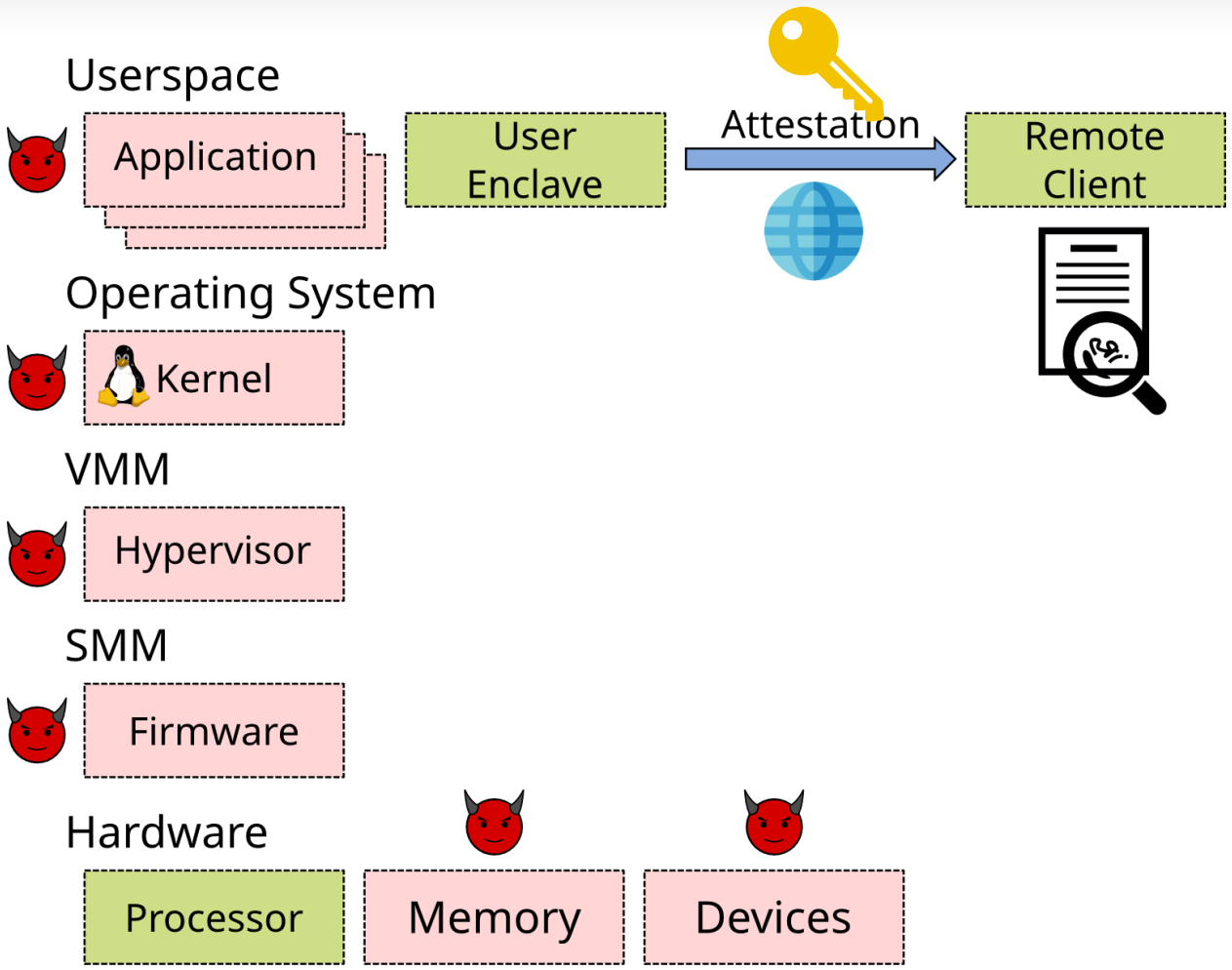
43

Dumping Enclave Memory



44

Intel Software Guard eXtensions



All trust relies on this key

45

Attestation Key

With access to the attestation key:

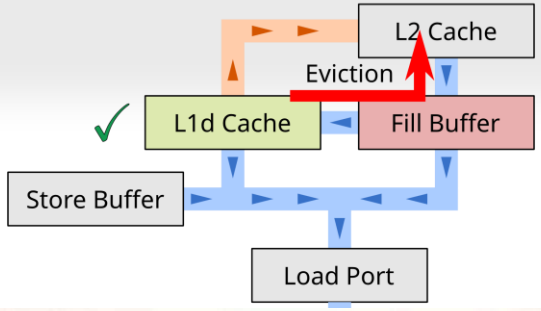
- You can fabricate and sign your own quotes
- Intel can't tell who signed that quote:
  - Enhanced Privacy ID (EPID) ensures **pseudonymity**
  - Hacker privacy guaranteed!
- A single compromised key erodes trust in the SGX ecosystem
- No need for an actual SGX machine to sign quotes:
  - Non-Intel machines can use SGX too

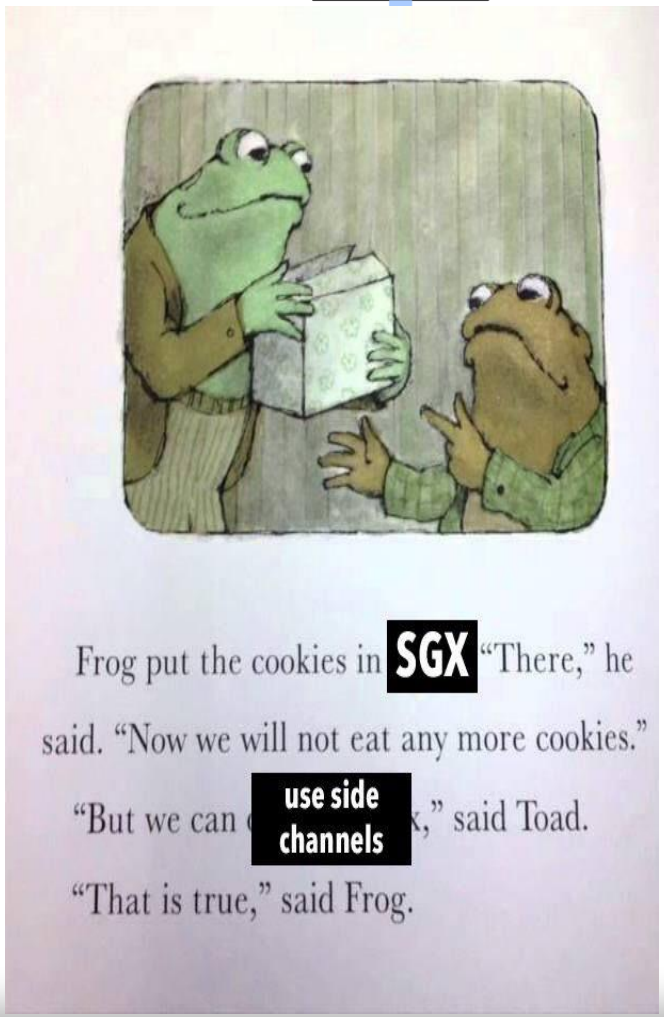
46

Impact

- New data path for eviction from the cache through the LFB
  - Leaked data across kernel, VMs, and even Intel SGX boundaries
- Microcode update to mitigate CacheOut
- Invalidate all SGX attestation keys

- Problem with speculation runs deep
  - Still works on meltdown proof architectures





My Work

Crypto	Post-Quantum Crypto KEMs [CCS'22] Secure Entropy Generation [IEEE S&P'20] Chrome Password Check [USENIX Security'23]
Operating System/Architecture	CacheOut [IEEE S&P'21] OS Availability Attacks [IEEE S&P'18] Kernel Hammer [Current] SGAxe [Current]
DRAM	RAMBleed [IEEE S&P'20] Spectre+Rowhammer [IEEE S&P'22]
Analog	Acoustic Eavesdropping [IEEE S&P'19]

RSA

- RSA is a public key crypto system

- The main operation is modular exponentiation, i.e. calculating

$$b^d \pmod{n}$$

- The exponent d is used for decryption and for digital signatures
  - d is secret!

Square and Multiply Exponentiation

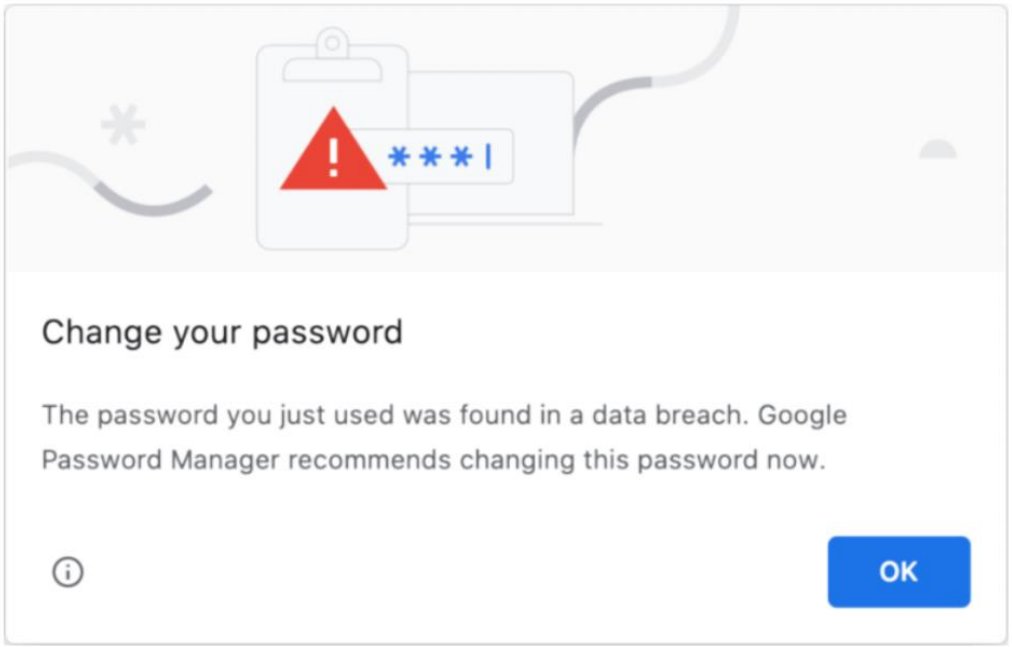
- Scans d from MSB to LSB
- For clear bits: square-reduce
- For set bits: square-reduce-multiply-reduce
- The sequence of operation reveals the secret exponent

$x \leftarrow 1$
for $i \leftarrow |e|\text{-}1$ **downto** 0 do
  $x \leftarrow x^2 \bmod n$
  if $(e_i = 1)$ **then**
    $x = xb \bmod n$
  endif
done
return x

Password Leak Detection

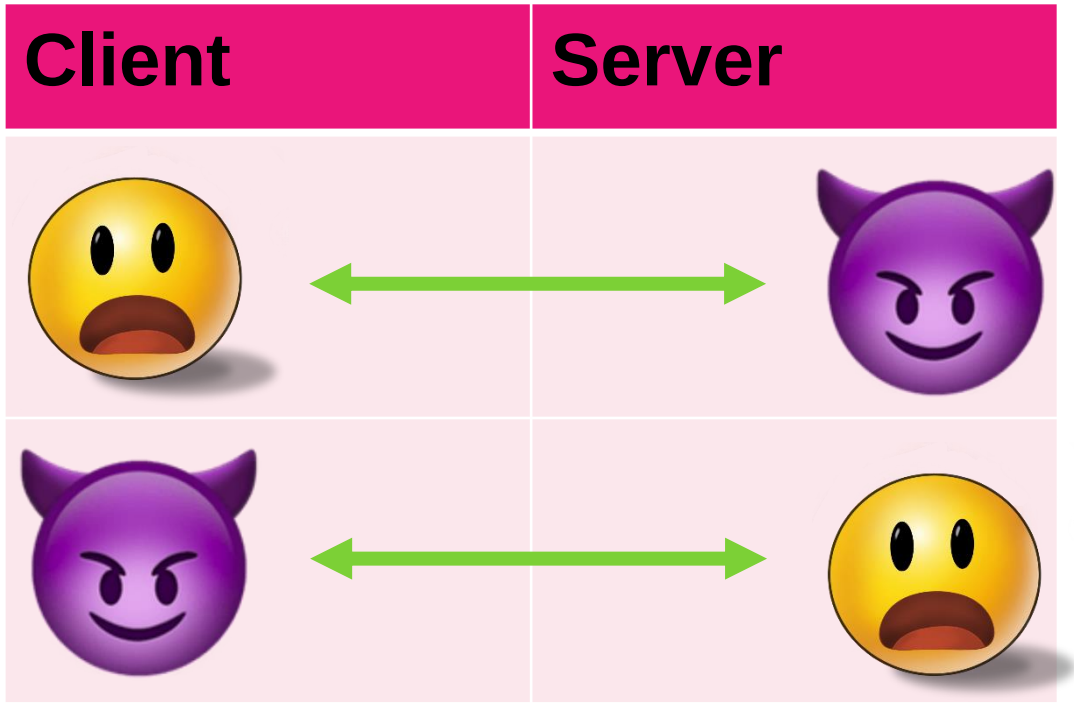
- Credential stuffing attacks
  - Reuse credentials on other services
  - ~7% credentials valid after compromise

- Chrome's Password Leak Detection
  - Checks input credentials for compromise on every login



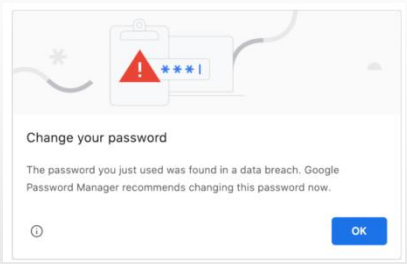
Password Leak Detection

- Privacy Preserving in both directions: assumes malicious client and malicious server
  - Server learns nothing about Client's credentials
  - Client learns nothing about the leaked credential database
- Developed custom protocol:
  - Anonymity sets
  - Memory-hard hashing
  - Private Set Intersection (PSI)
- Leaks at multiple points
  - Guess password on first attempt 80% of the time
  - Leaks to a malicious web page

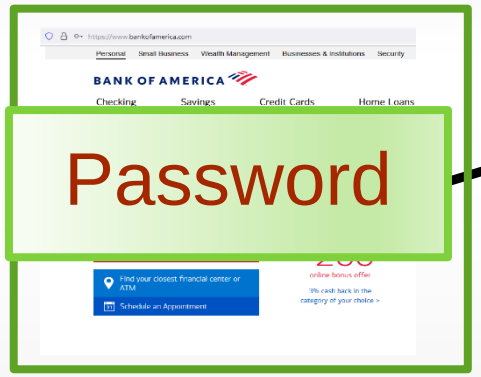


Chrome Browser

Google's Password Leak Server



Tab 1



Tab 2

JS/WASM

Bankofamerica.com

Evil.com

How Scrypt Leaks

- Scrypt is maximally memory-hard
  - Cost scales with memory, not CPU
  - Resistant against cracking attacks

```
1:  function SCRYPTROMIX(r, B, N)
2:      X ← B
3:      for i = 0 to N − 1 do                    ▷ Initialization Phase
4:          V[i] ← X
5:          X = scryptBlockMix(X)
6:      for i = 0 to N − 1 do                    ▷ Access Phase
7:          j = Integerify(X) mod N
8:          T = X ⊕ V[j]                         ▷ Input-Dependent Memory Access
9:          X = scryptBlockMix(T)
10:     return X
```

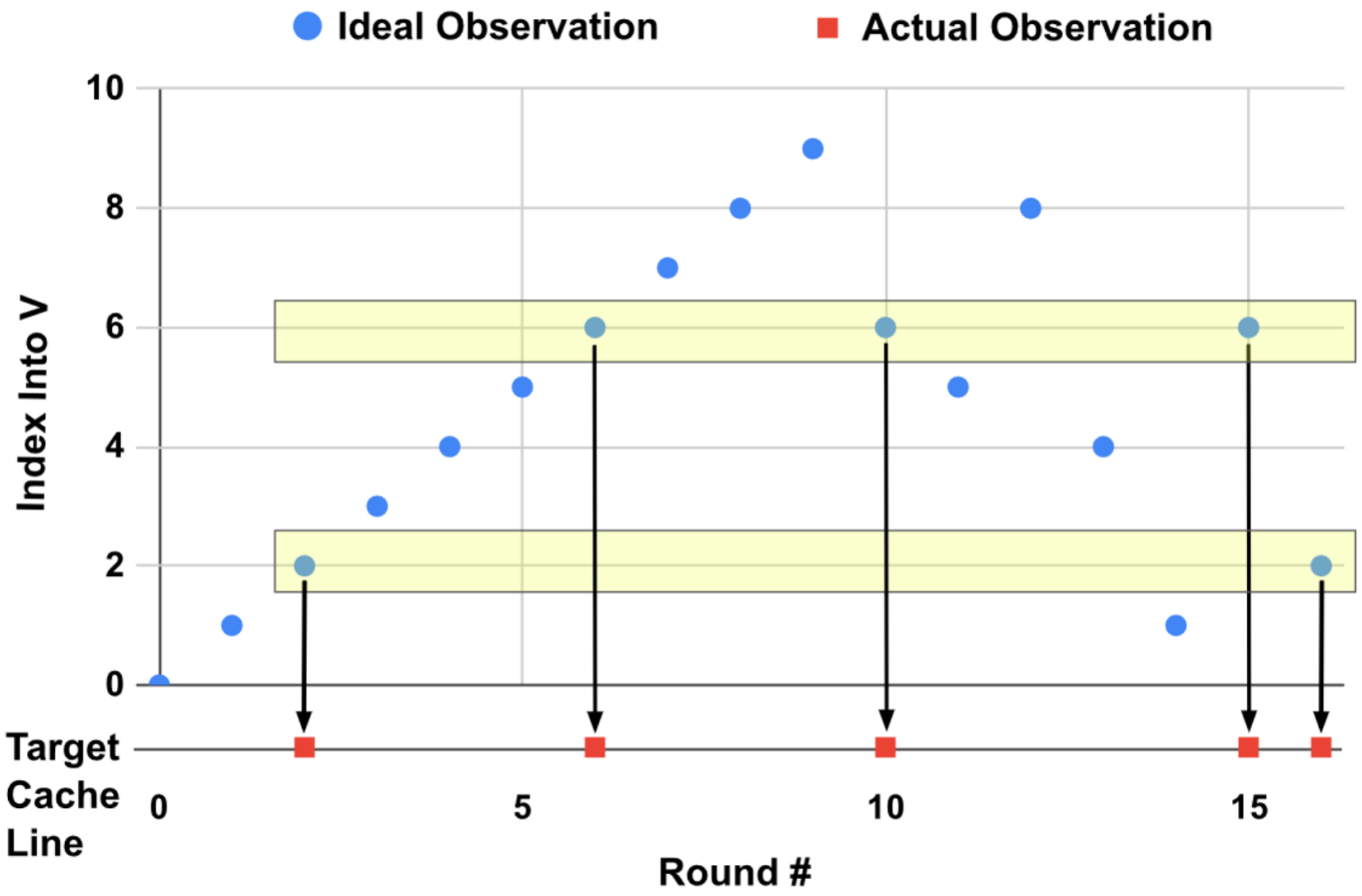
- Inherently non input-oblivious

Accesses into *V* *dependent on input*

55

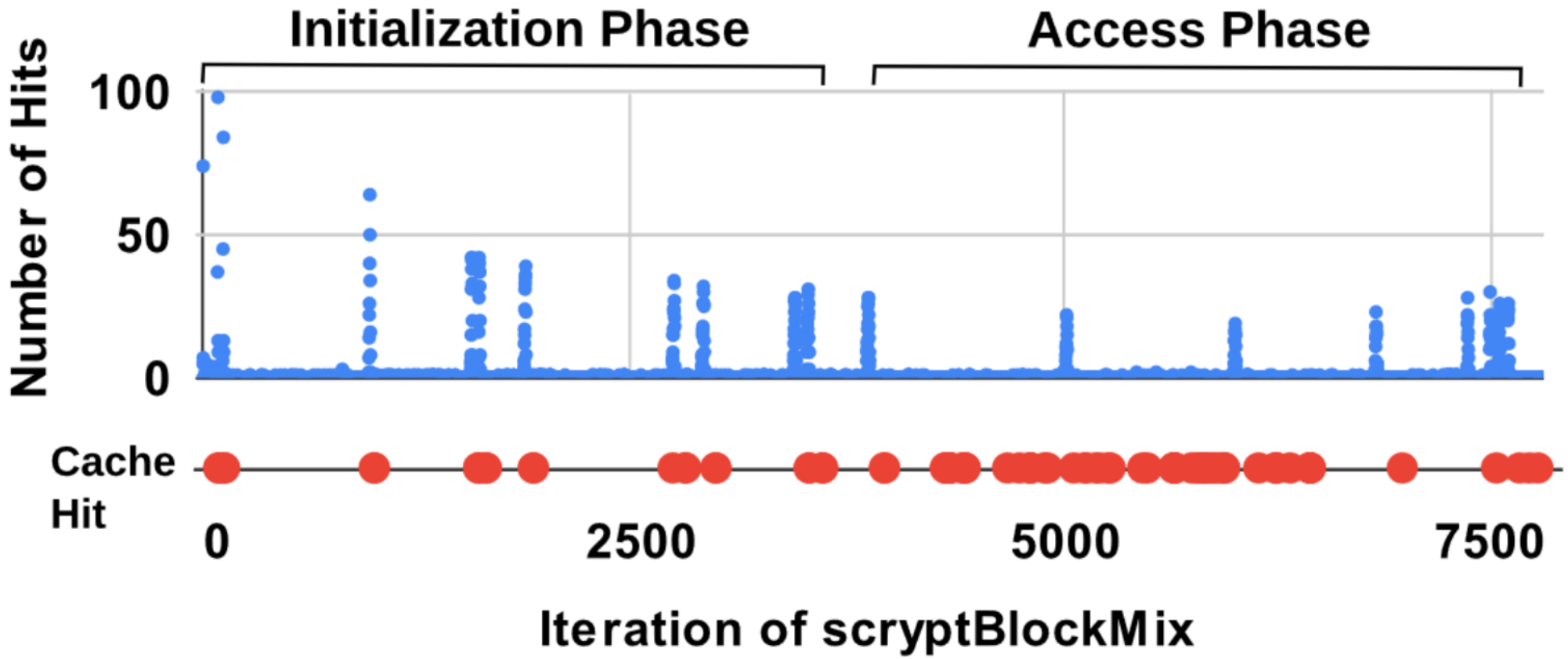
Accesses into *V*

- Reality: Prime+Probe gives very limited view into the accesses into *V*
  - only probing 1 cache-set
  - ~8 elements in *V* map to a single cache set



56

Access Patterns



- Top: 150 traces from custom scrypt implementation
- bottom: single trace (what we get in reality with Chrome)

57

Dictionary Attack

- Score every password in dictionary
  - Similarity of the resulting access pattern with that password as input

- Victim inputs randomly chosen password from Rockyou.txt
  - 14,341,564 plaintext passwords
  - Uniquely identified the password the majority of the time

Memory-hard hash functions are not always suitable for passwords

58

Fixing Chrome

- Worked with Google to fix it
- Constant-time:
  - Memory accesses, branches, and execution time cannot be dependent upon the client's credentials
- Scrypt is memory-hard
  - Inherently not input-oblivious
- Complex tradeoffs

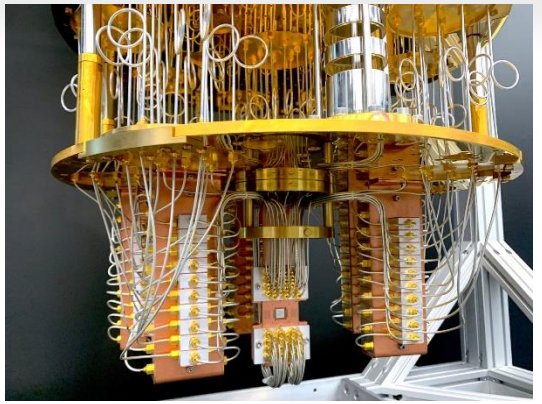
Impact

- Other browsers have also implemented their own password monitors
  - Safari, Firefox
  - Edge developed their own fully homomorphic encryption (FHE) based PSI

- New crypto needs to consider hardware security
  - Zero Knowledge Proofs (ZK)
  - Multi-Party Computation (MPC)
  - Post Quantum Cryptography (PQC)

Post Quantum Cryptography

- Quantum computers can breach many popular cryptosystems in use today
- NIST began standardization process in 2016
- We worked with NIST and examined *FrodoKEM*
  - Third round candidate
- Side-channels enable an **end-to-end key recovery** attack!
  - Bruteforce session keys in *2* minutes on a laptop*
  - Compromise is difficulty to detect, permanent



Best Paper Honorable Mention at CCS 2022

My Work

Crypto	Post-Quantum Crypto KEMs [CCS'22] Secure Entropy Generation [IEEE S&P'20] Chrome Password Check [USENIX Security'23]
Operating System/Architecture	CacheOut [IEEE S&P'21] OS Availability Attacks [IEEE S&P'18] Kernel Hammer [Current] SGAxe [Current]
DRAM	RAMBleed [IEEE S&P'20] Spectre+Rowhammer [IEEE S&P'22]
Analog	Acoustic Eavesdropping [IEEE S&P'19]

Questions?

62

Impact

- Post Quantum Crypto has unique considerations
  - Vulnerable to failure boosting attacks
  - Keys should be verifiable
  - Should be rotated

- Selected by BSI: “cryptographically suitable to protect confidential information on a long-term basis”
- ISO/IEC currently planning on standardizing some PQC algorithms
  - FrodoKEM is one of the three suggestions.
- Post Quantum Cryptography needs to consider leaky hardware