Curious case of Rowhammer: Flipping Secret Exponent Bits using Timing Analysis

Sarani Bhattacharya and Debdeep Mukhopadhyay

Indian Institute of Technology Kharagpur

CHES 2016
August 19, 2016

Objective of Talk

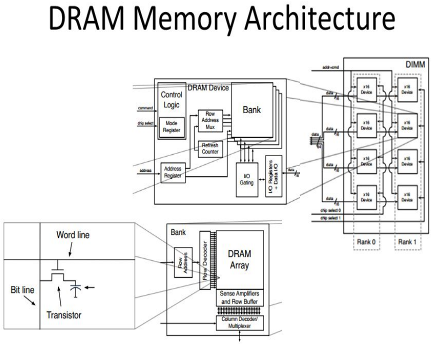
- Develop a methodology which combine timing analysis to perform the hammering in a controlled manner and to create bit flips in cryptographic keys which are stored in memory.

- Our Contributions:
  - We combine knowledge of reverse engineering of LLC slice and DRAM addressing with timing side-channel to determine the bank in which secret resides.
  - We precisely trigger rowhammer to address in the same bank as the secret.
  - This increases probability of bit flip in the secret exponent and the novelty of our work is that we provide series of steps to improve the controllability of fault induction.

Outline

DRAM Architecture

- DRAM is hierarchically composed of Channels, Rank and Banks.
- Each bank is a two-dimensional collection of cells having typically 2^{14} to 2^{17} rows and a row-buffer.
- Any row in a particular bank can only be read and written by involving the row-buffer.



The latency in DRAM access when two access request concurrently map to same channel, rank, bank but different row is termed as *row-buffer conflict*.

Rowhammer: DRAM vulnerability

- Repeated discharging and recharging of the cells of a row results in leakage of charge in the adjacent rows.
- If repeated enough times, typically before the automatic refresh in the adjacent rows, causes flipping of bits - phenomenon termed as Rowhammer.

```
Code-hammer
{
mov (X), %eax // read adrs X
mov (Y), %ebx // read adrs Y
clflush (X) // cacheflush X
clflush (Y) // cache flush Y
jmp Code-hammer
}
```

Cache Memory Architecture

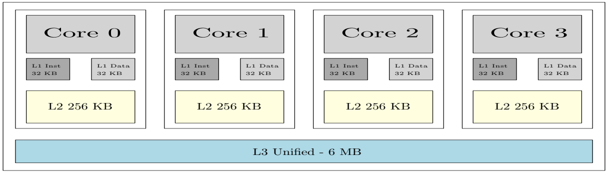


Figure: Cache Architecture in Intel Ivy Bridge [1]

- The slice addressing in modern processors is implemented computing a complex Hash function.
- Recently, reverse engineering of the LLC slice addressing function has been attempted [2, 3].
- The functions differ across different architectures.

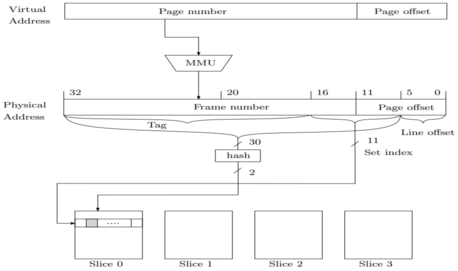


Figure: Cache Memory Indexing

Attack Model and Assumptions

The Adversary aims to:

To induce bit fault in the secret exponent of the public key exponentiation algorithm.

The challenges involved are:

- The secret resides in some location in the cache memory and also in some unknown location in the main memory.
- The attacker having user-level privileges in the system, does not have the knowledge of these locations in LLC and DRAM since these location are decided by mapping of physical address bits.

- In order to perform rowhammer on the secret exponent, the adversary first needs to identify the corresponding bank in DRAM in which the secret exponent resides.

- If adversary frequently queries the decryption oracle with valid ciphertexts, decryption process will perform exponentiation which access the secret exponent.
- But access requests are usually addressed from the cache memory itself since they result in a cache hit.

- This motivates the adversary to incorporate a spy process
  - which runs concurrent to the execution of the decryption algorithm,
  - uses timing analysis,
  - to identify the channel, rank and the bank where the secret gets mapped to.

Outline

Identifying the Eviction sets

- The adversary is oblivious of the virtual address space used by the decryption engine and thus involves a spy process which uses *Prime + Probe* cache access methodology to identify the target sets.
- The spy process targets the Last Level cache (LLC) since it is shared within all cores of the system.

1. Spy initially allocates a set of data elements and consults its own pagemap to obtain the corresponding physical addresses for each element.
2. The kernel allows userspace programs to access their own pagemap (`/proc/self/pagemap`) for all Linux kernels before version 4.0.

Cache Set collision with secret

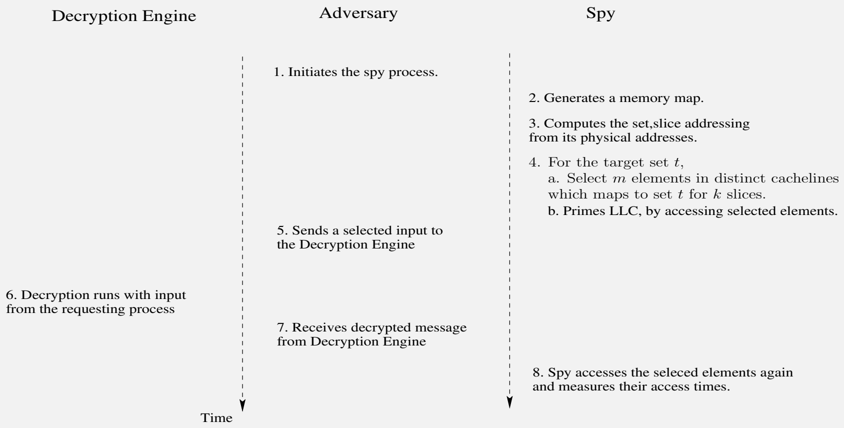

Decryption Engine
Adversary
Spy
1. Initiates the spy process.
2. Generates a memory map.
3. Computes the set,slice addressing from its physical addresses.
4. For the target set t,
a. Select m elements in distinct cachelines which maps to set t for k slices.
b. Primes LLC, by accessing selected elements.
5. Sends a selected input to the Decryption Engine
6. Decryption runs with input from the requesting process
7. Receives decrypted message from Decryption Engine
8. Spy accesses the seleced elements again and measures their access times.
Time

Figure: Steps to Determine Cache sets shared by Secret Exponent

Spy at work

If the target system is having k processor cores then LLC has k slices, each slice having c cache sets and each set being m way associative.

- If the cache line size is of b bytes, then least significant $log_2(b)$ bits of physical addresses are used as index within the cache line.
- $log_2(k)$ bits determine the cache set number.
- Because of associativity, m such cache lines having identical $log_2(k)$ bits reside in the same set.
- Hash function reverse engineered in [2, 3], is used to compute the slice in which a cache set gets mapped.

Selecting elements belonging to Eviction set

To precisely control the eviction of existing cache lines from set t, spy runs a selection algorithm to select an *eviction set* of $m * k$ elements belonging to each set t.

- Thus the selection algorithm selects elements of distinct cache lines for each of the k cache slices such that physical addresses maps to the same set t.
- In addition, each set of a slice is m way associative, the selection algorithm selects m elements corresponding to each k cache slice belonging to set t.
- The spy accesses each of these $m * k$ selected memory elements repeatedly to ensure that the cache replacement policy has evicted the existing cache lines.

Prime + Probe

1. The spy primes the target Set t and becomes idle.
2. The adversary sends the chosen ciphertext for decryption and waits till the decryption engine sends back the message.
3. If cache sets used by the decryption is same as spy, then the cache lines primed by the spy process gets evicted during the decryption.
4. Adversary signals the spy to start probing and timing measurements are noted.
5. Spy process accesses each of the selected m elements of eviction set t for all slices and time to access each of these elements are observed.
6. When the spy again accesses the same elements, if it takes longer time to access then we conclude that the cache set has been accessed by the decryption as well.

Outline

Identifying the target LLC slice

- Adversary identifies the target LLC slice by iteratively running *Prime + Probe* protocol separately for each k slices with the selected m elements for that particular slice.
- The timing observations while probing will show significant variation for a set of m elements which corresponds to the same slice where the secret maps.
- Thus we further refine the size of *eviction set* from $m * k$ to m elements.

Outline

Identifying the Target DRAM bank

- Concurrent accesses to different rows in the same DRAM bank results in row-buffer conflict and automatically leads to higher access time.
- The functions which decide the channel, rank and bank mapping from the physical addresses are not disclosed by the architecture manufacturers.
- Some recent works attempted the reverse engineering of these unknown mappings.

DRAM bank collision with secret

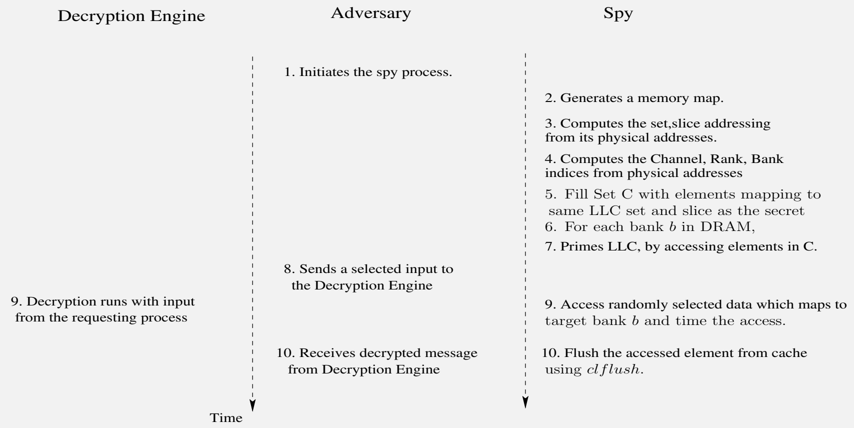

Decryption Engine
Adversary
Spy
1. Initiates the spy process.
2. Generates a memory map.
3. Computes the set,slice addressing from its physical addresses.
4. Computes the Channel, Rank, Bank indices from physical addresses
5. Fill Set C with elements mapping to same LLC set and slice as the secret
6. For each bank b in DRAM,
7. Primes LLC, by accessing elements in C.
8. Sends a selected input to the Decryption Engine
9. Decryption runs with input from the requesting process
9. Access randomly selected data which maps to target bank b and time the access.
10. Receives decrypted message from Decryption Engine
10. Flush the accessed element from cache using $clflush$.
Time

Figure: Steps to determine the DRAM bank in which secret maps to

Experimental Setup

1. We target an 1024 bit RSA implementation using square and multiply as the underlying exponentiation algorithm using `GNU-MP` big integer library (version number `2:5.0.2+dfsg-2`).
2. The experiments are performed on Intel Core i5-3470 processor of Intel Ivy Bridge micro-architecture running Ubuntu 12.04 LTS with the kernel version of 3.2.0-79-generic.
3. The Linux kernel version for our experimental setup being older than version 4.0, we did not require administrator privileges to perform the entire attack.

Determining cache set and slice

- The experiments being performed on RSA, the 1024 bit exponent resides in 2 cache lines each of 64 bytes.
- 11 bits of physical address from $b_6, b_7, \cdots b_{16}$ refer to the Last Level cache set.
- The functions used for slice selection are:
  $h_0 = b_{17} \oplus b_{18} \oplus b_{20} \oplus b_{22} \oplus b_{24} \oplus b_{25} \oplus b_{26} \oplus b_{27} \oplus b_{28} \oplus b_{30} \oplus b_{32}$
  $h_1 = b_{18} \oplus b_{19} \oplus b_{21} \oplus b_{23} \oplus b_{25} \oplus b_{27} \oplus b_{29} \oplus b_{30} \oplus b_{31} \oplus b_{32}$
- The host machine having 4 LLC slice for 4 cores selects the eviction set of $12 * 4 = 48$ data elements in distinct cache lines mapped to same set and all 4 slices.

Identifying the Cache Set

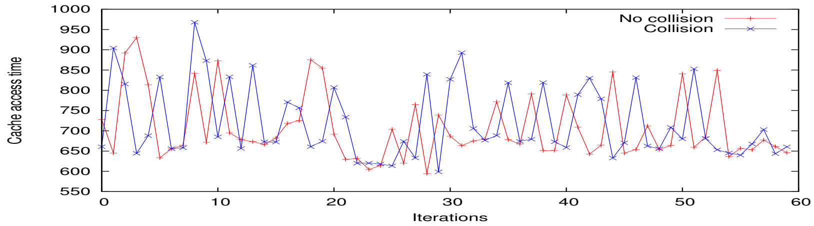
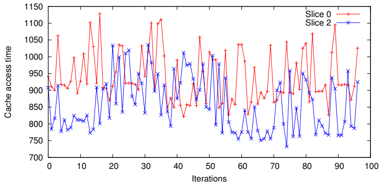


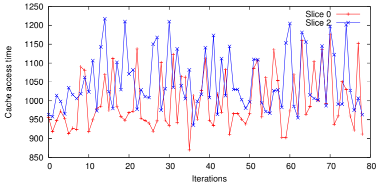
Figure: Timing Observations for Cache set collision

- The sets are chosen such that one of them is having a collision with the secret exponent and the other set does not have any collision.
- The average access time of the these two sets during the probe phase differs by approximately 80 clock cycles.

Identifying the LLC Slice



(a) Timing Observations during Probe phase when secret maps to slice 0

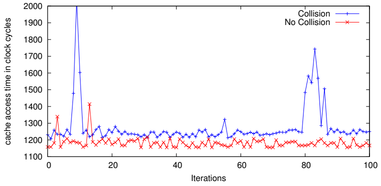


(b) Timing Observations during Probe phase when secret maps to slice 2

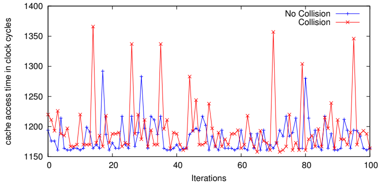
Figure: Timing Observations for LLC slice collision

- In Fig.(a), the secret is mapped to LLC slice 0, while in Fig.(b), the secret gets mapped to LLC slice 2.
- In both figures, access time for the cache slice where secret access collides is observed higher than the other slice belonging to same set but no cache collision.

Alternative Strategy determining target Set and slice



(a) Timing Observation of Cache set Collision from optimal eviction set



(b) Timing Observation of Cache slice Collision

Figure: Timing Observations for LLC set and slice collision

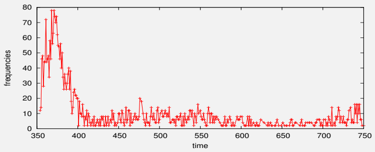
- In Fig.(a), timing observations obtained using optimal eviction set as described in [4].
- Fig.(b) uses the slice selection functions for a 4-core processor where the functions [3] are:

$$o_0 = b_6 \oplus b_{10} \oplus b_{12} \oplus b_{14} \oplus b_{16} \oplus b_{17} \oplus b_{18} \oplus b_{20} \oplus b_{22} \oplus b_{24} \oplus b_{25} \oplus b_{26} \oplus b_{27} \oplus b_{28} \oplus b_{30} \oplus b_{32} \oplus b_{33}$$

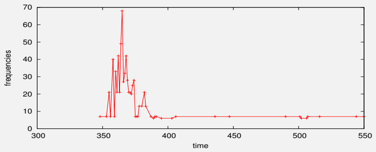
$$o_1 =$$

$$b_{07} \oplus b_{11} \oplus b_{13} \oplus b_{15} \oplus b_{17} \oplus b_{19} \oplus b_{20} \oplus b_{21} \oplus b_{22} \oplus b_{23} \oplus b_{24} \oplus b_{26} \oplus b_{28} \oplus b_{29} \oplus b_{31} \oplus b_{33} \oplus b_{34}$$

Identifying the DRAM Bank



(a) Timing Observations in clock cycles for DRAM bank collision



(b) Timing Observations in clock cycles of separate DRAM bank access

Figure: Timing Observations for Row-buffer collision during DRAM bank accesses

In our experimental setup, there exists 2 channel, 1 DIMM per channel, 2 ranks per DIMM, 8 banks per rank and 2^{14} rows per bank.

- The DRAM bank equations for Ivy Bridge [5] is decided by the physical address bits: $ba_0 = b_{14} \oplus b_{18}$, $ba_1 = b_{15} \oplus b_{19}$, $ba_2 = b_{17} \oplus b_{21}$,
- Rank is decided by $r = b_{16} \oplus b_{20}$ and the
- Channel is decided by, $C = b_7 \oplus b_8 \oplus b_9 \oplus b_{12} \oplus b_{13} \oplus b_{18} \oplus b_{19}$.
- The DRAM row index is decided by physical address bits $b_{18}, \cdots, b_{31}$.

Inducing Bit Flip using Rowhammer

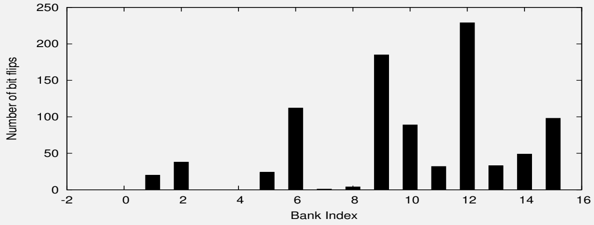


Figure: Number of bit flips observed in all banks of a single DIMM

- The secret exponent can sit in any of the rows in the target bank.
- We restricted our hammering attempts in the target bank.
- The fault attack in [6] requires a single faulty signature to retrieve the secret.
- Bit flip introduced in the secret exponent by the rowhammer in a specific bank can successfully reveal the secret by applying fault analysis techniques in [6].

Countermeasures

1. Probabilistic Adjacent Row Activation (PARA) [7]
2. Targeted Row Refresh (TRR) [8]
3. ANVIL [9]

Limitations and practicality of the attack

- We assume that the secret decryption exponent resides in a particular location of the DRAM and is not page-swapped by other running processes.
- Access to pagemap is assumed to be available at user privilege level since our setup has 3.2.0-79-generic version of Linux kernel.
- The attack would still be relevant in a cross-VM environment as in [10], where the users of the co-located VMs actually have the administrator privilege.
- The attack in its original form might be relevant in customized embedded system applications.

Summary

- We illustrate a combination of timing and fault analysis attack exploiting Rowhammer to induce bit flip in the secret.
- Experiments involving timing analysis shows significant variation and leads to the identification of LLC set and slices.
- Row-buffer collision has been exploited to identify the DRAM bank which holds the secret.
- Proposed attack finds most relevance in cross-VM setup, where co-located VMs share the same underlying hardware and thus root privileges are usually granted to the attack instance.

Thank You

Yuval Yarom and Katrina Falkner.
FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack.
In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 719–732. USENIX Association, 2014.

Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar.
Systematic reverse engineering of cache slice selection in intel processors.
In *2015 Euromicro Conference on Digital System Design, DSD 2015, Madeira, Portugal, August 26-28, 2015*, pages 629–636. IEEE Computer Society, 2015.

Clémentine Maurice, Nicolas Le Scouarnec, Christoph Neumann, Olivier Heen, and Aurélien Francillon.
Reverse engineering intel last-level cache complex addressing using performance counters.
In Herbert Bos, Fabian Monrose, and Gregory Blanc, editors, *Research in Attacks, Intrusions, and Defenses - 18th International Symposium, RAID 2015, Kyoto, Japan, November 2-4, 2015, Proceedings*, volume 9404 of *Lecture Notes in Computer Science*, pages 48–65. Springer, 2015.

Daniel Gruss, Clémentine Maurice, and Stefan Mangard.
Rowhammer.js: A remote software-induced fault attack in javascript.
CoRR, abs/1507.06955, 2015.

Peter Pessl, Daniel Gruss, Clémentine Maurice, and Stefan Mangard.
Reverse engineering intel DRAM addressing and exploitation.
CoRR, abs/1511.08756, 2015.

Dan Boneh, Richard A. DeMillo, and Richard J. Lipton.
On the importance of checking cryptographic protocols for faults (extended abstract).
In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu.
Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors.

In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*, pages 361–372. IEEE Computer Society, 2014.

Mark Seaborn, Thomas Dullien.
Exploiting the DRAM rowhammer bug to gain kernel privileges,http://googleprojectzero.blogspot.in/2015/03/exploiting-dram-rowhammer-bug-to-gain.html , 2015.

Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd M. Austin.
ANVIL: software-based protection against next-generation rowhammer attacks.
In Tom Conte and Yuanyuan Zhou, editors, *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '16, Atlanta, GA, USA, April 2-6, 2016*, pages 743–755. ACM, 2016.

Mehmet Sinan Inci, Berk Gülmezoglu, Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar.
Cache attacks enable bulk key recovery on the cloud.
IACR Cryptology ePrint Archive, 2016:596, 2016.