ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.10, 1–6

DRAM architecture for efficient data lifetime management

Yongjun Lee[1,2a)], **Yunkeuk Kim**[1], **Jinkyu Jeong**[1], **and Jae W. Lee**[3b)]

[1] *College of Information and Communication Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan, Suwon, Gyeonggi-do 16419, Republic of Korea*

[2] *Memory Division, Samsung Electronics Company, DSR, Samsung-ro, Giheung, Yongin, Gyeonggi-do 17113, Republic of Korea*

[3] *Department of Computer Science and Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea*

a) *yongjunlee@skku.edu*

b) *jaewlee@snu.ac.kr*

Abstract: Many applications compute on sensitive data, such as confidential user information. Even if these applications are terminated, sensitive data often persist in the main memory indefinitely until the deallocated pages are overwritten by OS. The conventional software-only solution of zeroing pages at deallocation generates a significant amount of bursty memory traffic to slow down other processes running concurrently. To address this, we propose Secure DRAM, a novel DRAM architecture that enables low-cost, secure deallocation of physical page frames. By preventing access to unallocated DRAM pages and not refreshing them, Secure DRAM effectively closes the window of vulnerability with minimal performance overhead.

Keywords: memory architecture, data management, security, data zeroing

Classification: Integrated circuits

References

[1] J. A. Halderman, *et al.*: "Lest we remember: Cold-boot attacks on encryption keys," Commun. ACM **52** (2009) 91 (DOI: 10.1145/1506409.1506429).

[2] Y. Wu, *et al.*: "A secure light-weight public auditing scheme in cloud computing with potentially malicious third party auditor," IEICE Trans. on Information and Systems **E99.D** (2016) 2638 (DOI: 10.1587/transinf.2016EDL8079).

[3] R. K. Venkatesan, *et al.*: "Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM," IEEE High Performance Computer Architecture (HPCA) (2006) (DOI: 10.1109/HPCA.2006.1598122).

[4] T. Vidas: "Volatile memory acquisition via warm boot memory survivability," IEEE 43rd Hawaii International Conference on System Sciences (2010) (DOI: 10.1109/HICSS.2010.439).

[5] S. Liu, *et al.*: "Flikker: Saving DRAM refresh-power through critical data partitioning," ACM SIGPLAN Not. **46** (2011) 213 (DOI: 10.1145/1961295.

DOI: 10.1587/elex.14.20170309
Received March 29, 2017
Accepted April 12, 2017
Publicized April 26, 2017
Copyedited May 25, 2017

1950391).
[6] J. Liu, *et al.*: "RAIDR: Retention-aware intelligent DRAM refresh," ACM SIGARCH Computer Architecture News **40** (2012) 1 (DOI: 10.1145/2366231.2337161).
[7] S. Lee, *et al.*: "Stealing webpages rendered on your browser by exploiting GPU vulnerabilities," IEEE Symposium on Security and Privacy (SP) (2014) (DOI: 10.1109/SP.2014.9).
[8] F. Kong, *et al.*: in *Multimedia and Ubiquitous Engineering* (Springer, Netherlands, 2013) 211.
[9] J. L. Henning: "SPEC CPU 2006 benchmark descriptions," ACM SIGARCH Computer Architecture News **34** (2006) 1 (DOI: 10.1145/1186736.1186737).
[10] Haiku Project. https://github.com/haiku/haiku.

1 Introduction

Ensuring security and privacy is a first-class design objective in modern computing systems. Many applications require access to sensitive data (e.g., confidential documents) during execution, which are stored in the main memory in plain text. Even if these applications are terminated or the system is rebooted, these data often persist indefinitely–until they are explicitly cleared by OS or overwritten by another application. This creates a window of vulnerability, which spans from the last valid access to the data to the reallocation of the page frame they belong to. Recently, researchers have demonstrated the feasibility to exploit such a vulnerability to leak sensitive data via cold/warm rebooting [1, 2] or memory dump [3, 4].

Although modern OSes zero deallocated memory pages to prevent such attacks, it is done either lazily (i.e., at next allocation) or asynchronously (i.e., by a separate zeroing thread), hence failing to eliminate this window of vulnerability. Zeroing generates a significant amount of bursty memory traffic to degrade throughput and increase performance variability of other applications running concurrently. In addition, it is demonstrated that a class of hardware-assisted attacks cannot be prevented by software-only solutions [1].

In this paper, we advocate a hardware-based solution to overcome these limitations and propose Secure DRAM, a novel DRAM architecture that enables low-cost, secure deallocation of physical page frames. Secure DRAM is a DRAM device-based solution to prevent illegal access to deallocated DRAM pages. Secure DRAM augments the conventional DRAM device with two capabilities: access control and data clearing. With these capabilities Secure DRAM can effectively close the window of vulnerability while incurring only a 0.1% throughput degradation for other co-scheduled processes on a four-core machine running multi-programmed workloads composed of SPEC CPU 2006 programs. This compares favorably to a software-only solution providing the same level of security, which suffers an up to 11.2% performance degradation with a geometric mean of 5.1%.

2 Memory attacks

Although a DRAM cell requires periodic refreshes to retain its value, the capacitor of a cell stores changes for a long time even if it is not refreshed [3, 5, 6].

DOI: 10.1587/elex.14.20170309
Received March 29, 2017
Accepted April 12, 2017
Publicized April 26, 2017
Copyedited May 25, 2017

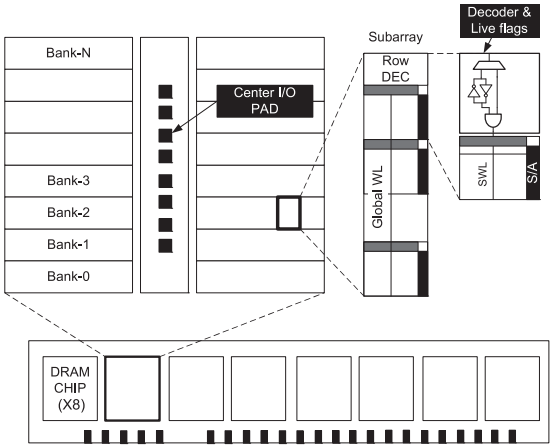

Fig. 1. Secure DRAM architecture with 4 K-bit row and a live flag

Venkatesan et al. find that about 90% of DRAM pages in modern DRAM devices retain data for up to 32 seconds. At a low temperature, retention time increases even further to make some pages keep their values much longer.

Exploiting this vulnerability, several memory attacks based on cold and warm booting have recently been presented [1, 4]. Halderman et al. demonstrate the feasibility of malicious (or forensic) acquisition of usable full-system memory images. Since DRAM devices retain data for tens of seconds and even longer with simple cooling techniques, they successfully extract cryptographic keys from memory images by using cold reboot.

Another class of memory attacks aim to acquire data from deallocated pages while the system is running [2]. This attack is feasible because an application's data are retained in memory even if the application is terminated until the memory pages are overwritten by another process. Lee et al. demonstrate that web pages can be reconstructed by acquisition of deallocated pages [7]. Kong et al. present a method to recover messages from a web mail client via memory dumps [8]. Thus, we propose Secure DRAM, a novel DRAM device architecture, to prevent such attacks with minimal performance and area overhead.

3 Secure DRAM architecture

Fig. 1 shows the organization of Secure DRAM. The double in-line memory module (DIMM) has multiple DRAM devices operating in tandem to send and receive data over memory channel. Each DRAM chip has multiple banks internally, and each bank contains multiple sub-arrays. Secure DRAM modifies the DRAM sub-array architecture for fast clearing of a DRAM page, hence effectively preventing malicious access to deallocated pages. Secure DRAM adds a live flag to each DRAM row, and the wordline is gated with this flag like in a similar way to a row repair method. We introduce a new DRAM command to set or reset the value of live flag, which takes a row address (like an activate (ACT) command). Whenever this command is received, the row address is decoded to select the specific flag register associated with this row and set its value. This flag indicates whether the corresponding page is currently allocated or not, which is explicitly managed by OS. When a DRAM chip is powered up, all live flags in the chip are

DOI: 10.1587/elex.14.20170309
Received March 29, 2017
Accepted April 12, 2017
Publicized April 26, 2017
Copyedited May 25, 2017

initialized to zero by a power-up sequence. This enables us to thwart a class of hardware-assisted attacks [1]. In addition, adding the live flag to a row decoder does not increase either activate-to-read delay (tRCD) or activate-to-precharge delay (tRAS) because the flag is already set before an activate command is received and not on the critical path of activation.

The hardware cost of Secure DRAM is minimal. The flag registers are placed in the row decoder. Assuming a 4 KB DRAM page size and X8 pins per DRAM chip, one-bit register (6 transistors) and one AND gate (6 transistors) are required per 4 K-bit memory cells, which increase the transistor count of the cell array by only 0.3%. The address decoder and other control signals are shared with the cell array. In addition, assuming a sense amplifier uses 40 transistors, adding an AND gate to its output increases the transistor count by 15%. Since the sense amplifier occupies 4∼5% of the total chip area, the area overhead is estimated to be 0.6∼0.75%.

4 Secure DRAM operations

Fig. 2 shows the operation of Secure DRAM when a page is allocated or deallocated. We assume that DRAM row buffer size (usually 4–16 KB) and OS page size are aligned. The live flag is set to one (live) when this physical page is allocated by OS and to zero (dead) when deallocated. To (re)set this flag, we can either introduce a new DRAM command or encode it in a high-order bit in the address bus. The live flag also gates the output of local sense amplifiers. When the page is live (Fig. 2(a)), normal output values will be sent to the I/O. When the page is dead (Fig. 2(b)), the output value will be gated to zero, hence preventing illegal access to the deallocated page.

When a physical page frame is freed, the OS deallocates the corresponding DRAM page by resetting the live bit (taking one DRAM command). Since the wordline of the DRAM page is disabled, refresh operations are not performed on this row. Thus, the DRAM cells in this row will eventually lose their values to be effectively cleared once retention time has elapsed. When a physical page frame is allocated, the OS sets the live bit of corresponding DRAM page to one. Meanwhile, the remaining charges in the DRAM cells of the page are discharged via bitline discharge scheme for clearing un-leaked DRAM cell. Therefore, any remaining data will be cleared regardless of how frequently the live flag is updated. Note that un-leaked DRAM cells are discharged before the page is activated, and that the activation time (tRCD) is not affected by the setting of the live flag.

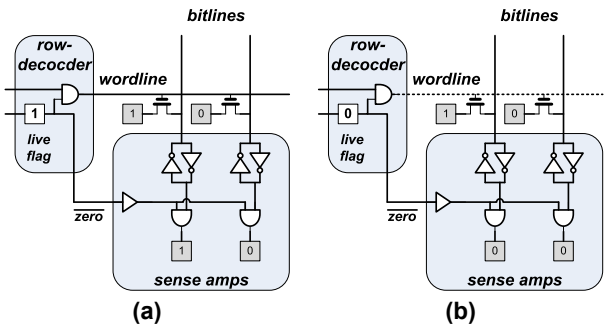

Fig. 2. Secure DRAM operations: (a) when live flag is one (live); (b) when live flag is zero (dead)

DOI: 10.1587/elex.14.20170309
Received March 29, 2017
Accepted April 12, 2017
Publicized April 26, 2017
Copyedited May 25, 2017

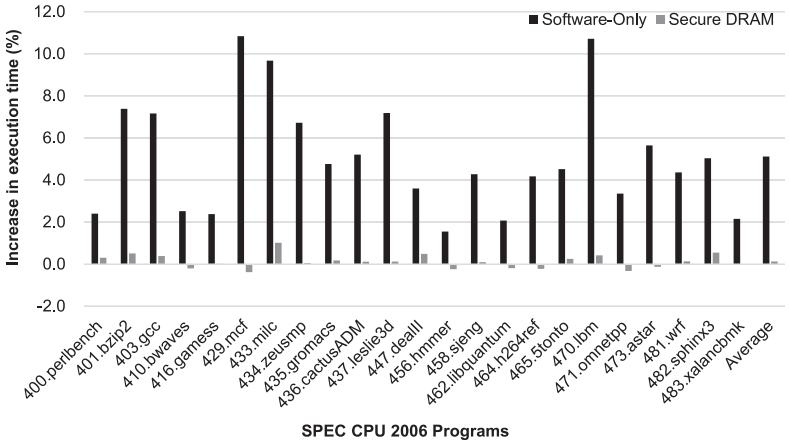

Fig. 3. Increase in execution time of the foreground SPEC CPU 2006 program due to zeroing in background

5 Evaluation

The performance of Secure DRAM is evaluated on Intel's 3.2 GHz Sandy Bridge quad-core system (i5-3470) with 4 GB memory, running Linux kernel version 3.9.1. To model the overhead of a software-only approach, we insert a zeroing function call into two kernel functions, `free_pages()` and `alloc_pages()`, which are invoked by OS at page frame deallocation and reallocation, respectively. For Secure DRAM we generate one memory request to reset the live bit of the DRAM page being freed. All execution times are normalized to the baseline with no protection.

Fig. 3 compares the execution time of the foreground SPEC CPU 2006 program [9] with three copies of *459.GemsFDTD*, a memory-intensive program with large resident set size (RSS) running in background. This quantifies the performance cost of providing security by zeroing page frames at deallocation. While the execution time is degraded by up to 11.2% (with an average of 5.1%) with zeroing in software, Secure DRAM incurs only negligible performance degradation (some 0.1%), which is generated by the OS calling `free_pages()` and `alloc_pages()` to set or reset the live flag bit whenever a physical page frame is allocated or deallocated.

Fig. 4 shows the distribution of execution time over 200 runs for *470.lbm*. With Secure DRAM, the execution time overhead is reduced from 11.2% (software-only) to 0.1%. Moreover, Secure DRAM also reduces the standard deviation of execution

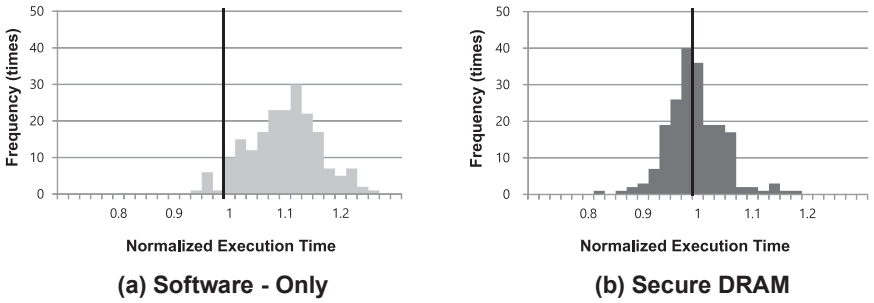

Fig. 4. Normalized execution distribution of 470.lbm out of 200 runs: (a) Software-only; (b) Secure DRAM

DOI: 10.1587/elex.14.20170309
Received March 29, 2017
Accepted April 12, 2017
Publicized April 26, 2017
Copyedited May 25, 2017

time by 20.1%. Fig. 5 shows the average execution time over 100 runs each case for forkbench [10] which is a (de)allocation-intensive application. Evaluation results show an average of 12.8% slowdown for the software-only solution but no slowdown for Secure DRAM.

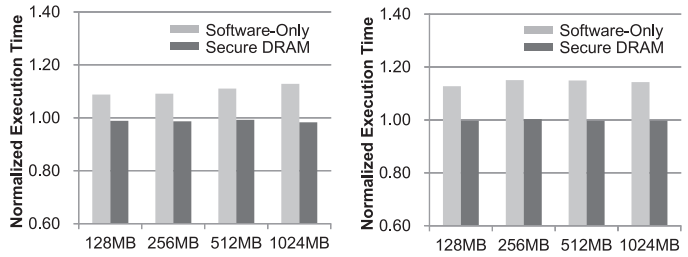

Fig. 5. Normalized execution time of forkbench: (a) with varying allocation sizes; (b) with varying number of forks

6 Conclusion

We propose Secure DRAM, a novel DRAM architecture to alleviate performance overhead of software-only solutions for clearing deallocated pages and prevent hardware-assisted memory attacks. Secure DRAM disables the wordline of a deallocated page and discharges bitline before page allocation. Secure DRAM incurs negligible performance overhead for clearing deallocated pages and minimal hardware cost. By preventing illegal access to a deallocated page, Secure DRAM effectively closes the window of vulnerability, which would otherwise be exploited by various memory attacks.

Acknowledgments

This research was supported by the National Research Foundation of Korea (NRF) grants funded by the Ministry of Science, ICT & Future Planning (MSIP) (NRF-2014R1A1A1005894 and NRF-2017R1C1B2007273). Jae W. Lee is the corresponding author.

DOI: 10.1587/elex.14.20170309
Received March 29, 2017
Accepted April 12, 2017
Publicized April 26, 2017
Copyedited May 25, 2017