A Phase Change Memory as a Secure Main Memory

André Seznec

A Phase Change Memory as a Secure Main Memory

André Seznec
Centre de Recherche INRIA Rennes Bretagne-Atlantique
Campus de Beaulieu, 35042 Rennes Cedex, France
seznec@irisa.fr

Abstract—Phase change memory (PCM) technology appears as more scalable than DRAM technology. As PCM exhibits access time slightly longer but in the same range as DRAMs, several recent studies have proposed to use PCMs for designing main memory systems. Unfortunately PCM technology suffers from a limited write endurance; typically each memory cell can be only be written a large but still limited number of times (10^7 to 10^9 writes are reported for current technology). Till now, research proposals have essentially focused their attention on designing memory systems that will survive to the average behavior of conventional applications. However PCM memory systems should be designed to survive worst-case applications, i.e., malicious attacks targeting the physical destruction of the memory through overwriting a limited number of memory cells.
In this paper, we propose the design of a secure PCM-based main memory that would by construction survive to overwrite attacks.

1 Introduction

Phase change memory (PCM) technology [5] appears as a promising technology for designing main memory in future computer systems [2], [7], [4], [3]. PCM presents advantages over DRAMs in terms of static energy consumption as well as integration scalability for future technologies generations; for instance, [4] anticipates a 4X higher memory density with PCM than with DRAM. Since PCM presents read access time in the same range as DRAMs, PCM has been recently considered as an alternative for designing main memory systems [2], [7], [4], [3]. Unfortunately, PCM suffers from a limited write endurance, i.e., a PCM memory cell can only support a limited number of writes and exceeding this limit might impair its correct functioning. The reported write endurances for PCM memory vary between 10^7 and 10^9 writes on a single cell. Such a limited endurance has been recognized as a major issue for the design of PCM-based main memory systems. Several propositions [2], [7], [4], [3] have been made to allow a PCM main memory to survive the anticipated lifetime of a computer system, i.e., 10 to 20 years, in the context of general applications.

At the exception of [3], these studies completely ignore the security breach that the limited write endurance of PCM components would create in a main memory. Using PCM components for main memory would create an opportunity to a malicious attacker to physically destroy the main memory through a very simple program overwriting the same memory cells again and again. The potential attack is particularly simple to mount. It can be run by any user without any execution privilege. Qureshi et al [3] show that their Region Based Start Gap scheme would survive a few months to a naive overwrite attack consisting in constantly overwriting the same physical memory address. However, in [6], we show that the Region Based Start Gap (RBSG) scheme considered in [3] would not survive more than a few days to a slightly more complex attack based on the birthday paradox. Morever the RBSG scheme a RBSG scheme supporting page mode would even be less endurant to an overwrite attack.

In this paper, we propose the design of a secure main PCM memory. In order to prevent a malicious user to overwrite some memory cells, the physical memory address (PA) manipulated by the computer system is not the same as the PCM memory address (PCMA) as proposed in [3]. PCMA is made invisible from the rest of the computer system. The PCM memory controller is in charge of the PA-to-PCMA translation . Hiding PCMA alone does not prevent a malicious user to blindly overwrite some PCM memory blocks. Therefore in the secure PCM-based main memory, PA-to-PCMA translation is continuously modified through a random process. This prevents a malicious user to overwrite some PCM memory words, it also uniformizes the write pressure on the overall memory for every possible type of workloads. For implementing the PA-to-PCMA translation, the PCM memory controller implements a translation table and needs an efficient random number generator. As an example, for write endurance in the 32M range, our study shows that associating a single translation table entry with a 4K memory blocks region should be sufficient. Provided one extra write per 8 program-generated writes, our scheme would resist an overwrite attack for 62 % of the expected total memory lifetime. However, endurance to overwrite attacks is obtained at the cost of some performance decrease on applications limited by the main memory bandwidth since one extra memory block read and one extra memory block write is generated every eight memory block writes. The security also limits the number of possible program-generated writes on the memory to 8/9 th of the total number of possible writes on the memory.

Manuscript submitted: 30-Nov-2009. Manuscript accepted: 21-Dec-2009. Final manuscript received: 18-Jan-2010

Posted to the IEEE & CSDL on 2/2/2010
DOI 10.1109/L-CA.2010.2 1556-6056/10/$26.00 © 2010 Published by the IEEE Computer Society

2 A SECURE PCM-BASED MAIN MEMORY

2.1 Security principles

2.1.1 Invisible PA-to-PCMA translation is required

If a malicious attacker knows the PA-to-PCMA translation then for a given PCM memory block B, he/she is able to figure out the address of the physical memory block that is mapped on B. If the PA-to-PCMA translation is made invisible from the outside of the PCM memory then the attacker can not retrieve the address of the physical memory block mapped on a given memory block.

2.1.2 PA-to-PCMA translations must dynamically change

Our analysis of the RBSG scheme [6] has shown that, in order to resist a birthday paradox attack, the PA-to-PCMA translation of any physical block B has to be modified with a frequency largely higher than one time every W_max *possible* writes on B. The PA-to-PCMA translation changes should be completely unpredictable from the outside of the PCM memory; in particular there should be no restrictions on the new translation.

2.2 Principles of a practical secure PCM-based main memory

2.2.1 PA-to-PCMA translation

In the secure PCM-based main memory we propose, the PA-to-PCMA translation is performed by the PCM-memory controller through the use of a translation table. For a physical memory block B, the address of the corresponding PCM block is computed from an entry read in the translation table and the address B. The PA-to-PCMA translation must perform a one-to-one address translation from the physical address space to PCM address space.

The simplest mapping would be to associate a translation table entry with each physical memory block and ensuring that the translation is a one-to-one block mapping. Such one-to-one block mapping appears as unpractical. Instead, we associate a single translation table entry with a region of R contiguous memory blocks; for instance if 4K contiguous memory blocks are mapped by a single entry, 64K entries are sufficient to map 16 GBytes. Such a single translation for a large region was already proposed in the context of wear leveling for conventional applications in [7].

2.2.2 PA-to-PCMA region address translation

Initializing at boot time the translation table T with a one-to-one region mapping is unpractical. Instead of such an initialization, we assume that at initialization time, the translation table T is initialized with only zeros, but that some computation is performed at run-time in addition to the read of the translation table. If memory regions are numbered from 0 to N-1, the translation is performed as follows: region **B** in physical memory is mapped onto region **(T(B).address xor B xor R_init)**, where **R_init** is a random number generated at initialization time. Note that, at initialization time,T (B) is null and that the PA-to-PCMA region address translation is a one-to-one mapping.

2.2.3 PA-to-PCMA region displacement translation

The use of a single entry to map a complete region of the physical memory could lead to a possible overwrite attack on trying to write a specific block in all the regions, for instance the first block. In order to avoid such an attack, the displacement in the region is also translated. Physical memory block **X** in region **B** is mapped onto block **(T(B).disp xor X xor D_init)** in region **(T(B).address xor B xor R_init)**. As **R_init**, **D_init** is a random number generated at initialization time.

2.2.4 Dynamically changing PA-to-PCMA translation

In order to avoid blind overwrite attacks, the PA-to-PCMA translation must be continuously modified. More precisely, only writes represent an issue. Therefore, the PA-to-PCMA translation modification is triggered randomly and only on memory writes.

This random triggering is particularly important: As an example, if the PA-to-PCMA translation change occurs periodically on writes, for instance every 10 writes, an attacker could repeat the sequence of nine consecutive writes on physical block B, one write on physical block C. Physical block C moves in the PCM memory, but block B remains on the same PCM memory location that can be easily overwritten.

2.2.5 How to modify PA-to-PCMA translation

Modifying the PA-to-PCMA translation for a physical region B is implementing through swapping the translations for two physical regions. This guarantees that the PA-to-PCMA translation remains a one-to-one mapping. The region swapping induces the modification of two entries in the translation table. A random physical region B′ is chosen and the PA-to-PCMA translations of B and B′ are exchanged, i.e.,
T(B).address := old(T(B′).address) xor B′ xor B and **T(B′).address := old(T(B).address) xor B′ xor B**. At the same time, displacement translations inside blocks B and B′ are also modified,: **T(B).disp: =old(T(B)).disp xor RAND** and **T(B′).disp: =old(T(B′)).disp xor RAND** where RAND is randomly selected.

The two memory regions in the PCM memory have to read and swapped accordingly. Randomly swapping memory regions has been already proposed in the context of wear leveling for flash memories [1].

2.2.6 Frequency of PA-to-PCMA translation modifications

The cost of a PA-to-PCMA translation modification is proportional to the size R of a region in the memory. The two swapped regions have to be read and rewritten, i.e., a PA-to-PCMA translation modification induces 2 R memory block reads and 2 R memory block writes.

Therefore, the frequency of the address translation modification should be chosen in order to maintain the total overhead to a reasonable level. In this study, we arbitrarily estimate that inducing in average one extra write on the PCM memory per 8 effective writes

would be acceptable. That is, in average one out of 16 R physical memory block writes can trigger a PA-to-PCMA address translation modification. Therefore on receiving a write on a physical memory block, the modification of its PA-to-PCMA translation is randomly triggered with probability $\frac{1}{16R}$.

2.3 Putting all together in the memory controller

The design of a secure PCM-based main memory leads to several constraints inside the memory controller.

2.3.1 Write endurance and region size

The principles above in Section 2.2 lead to the design of a PCM-based main memory on which an overwrite attack would only be able to consecutively write the same memory block in average 16R times before the physical block is moved in another PCM memory block. In practice, a write attack could succeed in significantly reducing the lifetime of the memory, if 16R is not small with respect to the write endurance of the cells.

We run simulations of an overwrite attack on a 16 GBytes PCM memory i.e., 2^{26} 256-byte blocks. Regions of respectively 64K and 4K memory blocks were considered. If the memory features a write endurance of $W_{max} = 2^E$ writes, then the theoretical write endurance of a uniformly accessed PCM memory is 2^{26+E}.

With a write endurance of only 8 Megawrites (2^{23}) per cell, using 64K memory blocks per region is not an option: some memory blocks would be destroyed by a brute force overwrite attack in less than a billion (2^{30}) writes. With 4K memory blocks regions, the PCM memory would be able to support an attack consisting of up to 38 % of the theoretical 2^{49} writes.

If the write endurance is 32 Megawrites per cell (2^{25}) then these respective ratios become 7.4 % for 64K memory blocks regions, and 62 % for 4K memory blocks region for a theoretical maximum of 88.88 % since in average one extra block write is triggered for 8 physical memory writes.

If the write endurance is 256 Megawrites per cell (2^{30}) then these ratios become 52 % and 79% for 64K and 4K memory block regions respectively.

Therefore, if the technology is able to ensure write endurance in the hundreds of millions range then even very large regions could be considered for PA-to-PCM translations.

2.3.2 Memory controller constraints

The secure PCM main memory would need to integrate extra hardware in the memory controller to implement the secure PA-to-PCMA translation.

2.3.3 Memory storage volume

The storage volume of the PCM memory controller is a major issue. The main component is the translation table that features an entry per memory region. For a 16 GB memory, the use of regions of 4K 256-bytes blocks would lead to 64K entries, each entry featuring the address of region (14 bits) and a displacement in the region (12 bits) i.e. a total of 26 bits. The total storage cost of the translation table would be 52Kbytes. If a 64K blocks region was used then the size of the PCM translation table would only be 3.25 Kbytes.

2.3.4 Swapping memory regions logic

The memory controller has to handle the important function of swapping two memory regions on a PA-to-PCMA translation change. This induces a large number of memory reads and writes. An atomic swap of the two memory regions would stop the normal read and write accesses by the computer system. This would be unacceptable.

Therefore the memory controller must feature logic to interleave blocks swapping with the normal flow of reads and writes from the computer system. The logic must be able to handle the case where a normal write is overwriting a block belonging to one of the memory regions being swapped. Moreover this normal flow of writes may randomly trigger new region swaps; the memory controller should be able to buffer these swaps.

The priority on writes must be dynamically adapted in order to maintain a limited number of regions waiting for swaps, for example at most 8 swaps. As an example, we tested a policy randomly splitting the write priority to 1/4th for region swapping and 3/4th for normal write flow when less than 4 region swaps are waiting and one half for region swapping and one half for normal write flow when 4 or more region swaps. On an experiment on 2^{40} writes and assuming a continuous saturated write flow from the computer system, there was never more than 8 waiting region swaps.

2.3.5 Extra PA-to-PCMA translation latency

The extra access time to main memory due to PA-to-PCMA translation is essentially due to the read of the PA-to-PCMA translation table. This table will be implemented as a SRAM table in the memory controller. For a 16 GB memory using 4K 256-byte blocks region, the read access time of a 52Kbytes SRAM memory would in the range of 2-4 processor cycles and would be marginal compared with the overall main memory access time.

2.3.6 The random number generator

Our secure PCM-based main memory will be able to resist to an overwrite attack if no one is able to follow or reconstruct the PA-to-PCMA translation process. Our proposal heavily relies on a random number generator. The security of our proposal also depends of the security of this random number generator.

One can remark that the output of the random number generation used in our memory controller cannot be directly observed from the outside of the PCM memory. Therefore different possible schemes could be implemented ranging from a true hardware random generator to a simpler algorithmic pseudo-random number generator personalized with a huge key at manufacturing time.

2.4 Write Endurance for conventional applications

The security of our proposed secure PCM main memory is ensured by 1) the invisibility of the access on the PCM

memory from outside the memory, and 2) the random distribution of the accesses through the PA-to-PCMA address translation change: under an overwrite attack, a given physical block will change PA-to-PCMA address translation after a number of writes that is comparatively small with the write endurance of the PCM memory cell.

For a conventional application, for a given physical memory region, the frequency of the PA-to-PCMA address translation changes measured in writes on the region is the same as under an overwrite attack, but the writes are distributed over the whole region. Therefore, for each block, the number of writes is very small compared with the endurance of the PCM memory cell. The write endurance of the overall memory system is therefore very close to its maximum theoretical endurance

2.5 A secure PCM main memory might become practical in a few years

2.5.1 Economic feasibility of PCM main memory

If within a few years, the write endurance per cell on PCM components reach 256 M writes then it would become feasible to build 16 GBytes (or larger) memory using comparatively very small PA-to-PCMA translation table (for instance, using 64K blocks memory regions): as mentioned above, the secure PCM memory would be able to survive to an overwrite attack at a full 4 GBytes/s write bandwidth for 52 % of an expected lifetime of 32 years.

2.5.2 Page mode is compatible with security

If PCM memories are used as main memory then a page mode would be interesting as on current DRAM to limit the access latency and increase bandwidth when the memory read requests exhibit high spatial locality. Our PA-to-PCMA translation scheme is compatible with such page mode since regions are large enough to accommodate large page, even split across several PCM components.

2.5.3 Limiting extra write traffic overhead

In this study, we have considered that the overwrite traffic associated with PA-to-PCMA translation modification could be as large as 1extra PCM block write per 8 physical memory block writes. This overhead can be reduced by decreasing the probability of triggering a PA-to-PCMA translation. This would reduce the total endurance of the system to the overwrite attack, but may still remain acceptable if the cell write endurance is large.

For instance, for a 16GBytes memory, if the extra PCM write traffic is limited to 1 extra PCM block write per 32 physical memory writes i.e., a 3.1% write overhead, the cell endurance is 256 Megawrites and 64K blocks memory regions are used then the secure PCM memory still survives a full 4 GBs/s bandwidth overwrite attack for 19 % of its 32 years expected lifetime, i.e., about 6 years.

3 Conclusion

If the promises of the PCM technology are fulfilled then it might become economically feasible to build a main memory from PCM memory component in the next few years. Such a PCM-based main memory will particularly be attractive due to its very low static energy consumption. However to consider such a memory for an industry product , the PCM based memory would have to be able to resist to software overwrite attacks targeting its physical destruction.

In this paper, we have proposed a first secure PCM based main memory that will resist to overwrite attacks . By hiding the effective PCM memory address from the rest of the computer system and continuously and randomly moving the physical memory blocks in PCM memory, overwrite attacks are made impossible. The proposed PA-to-PCMA translation scheme uniformizes and randomizes the write flow on PCM memory for malicious overwrite attacks as well as conventional non malicious applications. Our scheme requires some hardware overhead in the memory controller (essentially a PA-to-PCMA translation table, the memory region swapping logic and a random number generator). But our scheme allows the overall PCM main memory to resist to an overwrite attack for a very significant fraction of its expected lifetime, e.g. 62 % for 4Kblocks regions and 32M write endurance per cell.

The scheme presented in this paper induces a significant extra write traffic (1 extra write per 8 effective writes). This consumes some useful memory write bandwidth and may reduce performance on memory intensive applications. A future study will address this issue for conventional (i.e., non-malicious attack) applications.

4 Acknowledgments

This work was partially supported by the European Commission in the context of the SARC integrated project #27648 (FP6).

References

[1] A. Ben-Aroya and S. Toledo. Competitive analysis of flash-memory algorithms. In *Proceedings of the 14th Annual European Symposium on Algorithms, Springer Lecture Notes in Computer Science, Volume 4168/2006*, pages 100–111, sept. 2006.

[2] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting phase change memory as a scalable dram alternative. In *ISCA*, pages 2–13, 2009.

[3] M. K. Qureshi, J. Karidis, V. Srinivasan, M. Franceschini, L. Lastras, and B. Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *Micro*, dec 2009.

[4] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. In *ISCA*, pages 24–33, 2009.

[5] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam. Phase-change random access memory: a scalable technology. *IBM J. Res. Dev.*, 52(4):465–479, 2008.

[6] A. Seznec. Towards Phase Change Memory as a Secure Main Memory. Research Report RR-7088, INRIA, 2009. also to be presented at the WEST workshop in conjunction with HPCA 2010, January 2010.

[7] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. A durable and energy efficient main memory using phase change memory technology. In *ISCA*, pages 14–23, 2009.