An Efficient Run-time Encryption Scheme for Non-volatile Main Memory

Xian Zhang, Chao Zhang, Guangyu Sun
Peking University
{zhang.xian, zhang.chao, gsun}@pku.edu.cn

Jia Di
University of Arkansas
jdi@uark.edu

Tao Zhang
Pennsylvania State University
zhangtao@cse.psu.edu

***Abstract*—Emerging non-volatile memories (NVMs) have been considered as promising alternatives of DRAM for future main memory design. The NVM main memory has advantages of low standby power, high density, and good scalability. Its non-volatility, however, induces a security design challenge that data retained in memory after power-off need to be protected from malicious attacks. Although several approaches have been proposed to solve this problem through data encryption, they have some limitations such as high design complexity and non-trivial timing/energy overhead. Moreover, these techniques decrease the lifetime of NVM main memory due to extra write operations caused by encryption. In order to overcome these limitations, we propose an efficient PAD-XOR based encryption scheme in this work. A novel PAD generator based on a randomizer and several sub-PAD tables is introduced. With the PAD generator, our encryption scheme can provide run-time data protection to all data in NVM memory with low timing and power overhead. In addition, the encryption process can co-operate with wear-leveling of NVM to reduce design complexity. More important, our encryption technique has no impact on lifetime because no extra writes are incurred. Experimental results demonstrate that, compared to prior approaches, our design can achieve the same security strength with substantial lower overhead in respect of timing, energy consumption, and design complexity.**

I. Introduction

Recently, smart phones and tablet computers become more and more popular. As the customer requirements keep increasing, the main memory capacity in these portable devices has increased significantly over recent years. For example, the main memory capacity of Intel's next generation tablet computer can be as large as 16GB [2]. Such a requirement makes the well-known problem of *Memory Wall* severer, especially for these portable devices with design constrains from power budget and footprint.

To attack *Memory Wall*, various emerging non-volatile memories (NVMs), such as Phase Change Memory (PCM), Resistive Random-access memory (ReRAM), and Spin-transfer Torque Random-access Memory (STTRAM), have been widely proposed to replace traditional SRAM/DRAM for future memory hierarchy design [14], [21], [16], [20], [8]. These NVMs have advantages of good scalability, low standby power consumption, high density, fast access speed, etc.

Among these NVMs, PCM is considered as a competitive alternative of DRAM. Prior research has demonstrated that PCM based NVM main memory can significantly improve memory energy-efficiency, compared to DRAM counterpart. On the other hand, researchers also pointed out that optimization techniques are required to overcome limitations of NVM main memory [18], [13], [14], [21], [16].

Most prior research focuses on improving write performance and lifetime of NVM main memory. Only a few of them address the security challenge caused by its non-volatility [4], [10]. Different from traditional DRAM memory, data stored in NVM main memory can be retained after power-off because of its non-volatility. On the one hand, it provides advance of being restored instantly when the system is powered on next time, but on the other hand, non-volatility makes these data more vulnerable to be attacked by malicious attacks. For example, an attacker can physically remove the main memory and extract sensitive information from it through a memory scanning [4]. This security problem is severer as mobile computing systems, such as tablets and smart phones, become more and more popular nowadays.

Recently, two approaches have been proposed to solve this security problem using data encryption. Chhabra *et al* proposed i-NVMM, which selectively encrypted data in NVM using AES algorithm [4]. In order to mitigate timing and energy overhead of encryption and decryption, i-NVMM only encrypted cold data that are not frequently accessed during runtime execution. The hot data is not protected until the execution is over or the system is powered down. One critical problem of this technique is that the hot data, which may be more sensitive than cold part, are not protected. For example, if the NVM memory is pulled out during run-time execution, the hot data are exposed directly to attackers. In order to fully protect all data in NVM memory, Kong *et al* introduced a counter-mode XOR based encryption [10]. Instead of encrypting data directly with AES, this technique calculates a crypto-PAD with AES for each memory line. Then, data is encrypted by XOR the crypto-PAD. However, the counter information used to calculate crypto-PAD has to be stored with data in NVM main memory for decryption. In addition, the overhead of AES algorithm cannot be avoided.

Besides non-trivial overhead, these approaches are not optimized for NVM main memory because of three reasons. First, these two approaches have a common limitation that the encryption process can harm the lifetime of NVM main memory due to extra write operations caused by encryption. The efficiency of write reduction techniques for NVM main memory is significantly degraded. Second, the wear-leveling techniques for NVM memory are never considered during encryption. Actually, the address re-mapping process in wear-leveling can also be considered as a type of data encryption,

This work is supported by the National Natural Science Foundation of China (No. 61202072), National High-tech R&D Program of China (No. 2013AA013201), and AMD Gift Grant.

978-1-4799-1400-5/13/$31.00 ©2013 IEEE

which can be leveraged to reduce design complexity of encryption. Third, the attack mode in previous mode is not well defined so that some encryption features are overemphasized. With a well-defined attacked mode, the requirements can be relaxed with the same protection strength. The detailed discussion of these issues can be found in Section II.

To overcome these limitations of prior work, we need to find an encryption scheme to satisfy following key requirements.

1) **Security** All data in memory must be encrypted during execution against potential runtime attacks.
2) **Overhead** Both design complexity and runtime timing/power overhead should be moderate.
3) **Compatibility** Encryption method should work well with other optimization techniques for NVM.

In order to achieve these design goals, we propose an efficient run-time XOR based encryption scheme in this work. The contribution of our design can be concluded as follows,

- Our encryption scheme can provide run-time protection for all data in memory.
- AES hardware is avoided so that both design complexity and timing/power overhead is significantly reduced.
- We present a clear attack model and prove that our scheme can achieve the same protection strength as prior approaches using AES algorithm.
- Our encryption scheme is totally compatible with partial write and redundant write techniques and there are not extra writes induced.
- Our encryption scheme can reuse hardware of wear-leveling component of NVM to reduce design complexity.
- We provide comprehensive evaluation by comparing different approaches. The results show that our method can outperform prior approaches, in respect of performance, power consumption, and lifetime of NVM memory.

The rest of this paper is organized as follows. Section II prepares preliminary knowledge of NVM main memory and XOR based encryption. We introduce our encryption scheme in Section III. Note that we move most detailed mathematical derivation to Appendix to make it easy to be understood. Comprehensive evaluation is presented in Section IV. We conclude our paper in Section V, followed by the Appendix section.

II. Preliminaries

In this section, we present related background to help understand our encryption scheme and corresponding discussion. It includes (1) write reduction techniques that can be affected by prior encryption approaches and wear-leveling techniques that can be leveraged by our encryption, (2) basic framework of PAD-XOR based encryption method.

A. Write Reduction and Wear-leveling Techniques

The limited write cycle is a well known problem of NVM main memory. For example, a PCM cell's write cycle is in the range of $10^6 \sim 10^8$ with different technologies [14], [16]. NVM main memory design may wear out within several months without any optimization. There has been extensive research on how to improve lifetime of NVM memory. These works can be categorized into either write reduction or wear leveling techniques, which are described as follows.

a) ***Write Reduction:*** The purpose of this method is try to reduce the number of updated bits to NVM main memory in each write operation. Lee *et al* proposed the method of partial write that only the cache lines written back from last level cache (LLC) are updated in a memory line [11]. In DCW method [21], old data in NVN main memory is first read out and compared with new data bit by bit. Then, only the modified bits are updated in the write operation. The Flip-N-write technique is proposed base on DCW method to further reduce number of updated bits by calculating hamming distance between write data and old data [5]. These techniques can effectively reduce write intensity up to 85% [21], [5].

Prior encryption techniques, however, can severely degrade efficiency of these techniques. It is because both these approaches are based on AES algorithm. It means that the whole memory line need to be re-encrypted even when only one bit is updated in a write operation. Corresponding evaluation can be found in Section IV. Note that some data encoding techniques are also proposed for write reduction [19]. The impact of encryption on these techniques is complicated to discuss quantitatively and is not included due to page limitation.

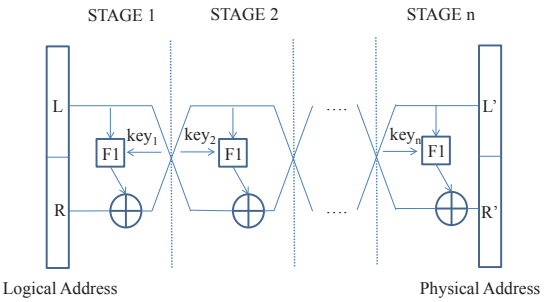

Fig. 1. An example of address mapping based on Feistel Network[14].

b) ***Wear Leveling:*** Even with write reduction techniques, some cells in NVM memory may wear out faster than the others due to non-uniform write intensity [21]. Thus, the wear leveling techniques are also necessary to balance write intensity. Various techniques, such as table based re-mapping, Start-gap, Security Refresh, etc., are proposed for wear leveling [21], [14], [16]. Although design details are different, the fundamental rule behind these techniques is the memory address re-mapping between *logic address* and *physical address*. In other words, memory addressing is periodically changed so that write intensity is uniformly distributed throughout the whole memory space. An example of address mapping based on *Feistel Network* is illustrated in Figure 1. Interestingly, these wear leveling techniques are also employed to protect NVM from the attack that try to wear out NVM memory through malicious repeat updates to same cells. In Section III-D, we show that the wear-leveling can be leveraged for data encryption in our scheme to reduce design complexity.

B. PAD-XOR based Encryption Method

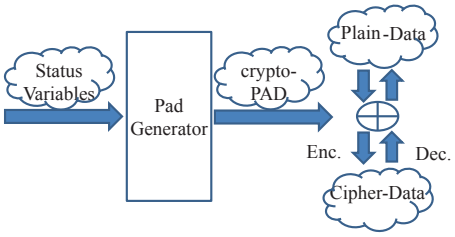

Fig. 2. Illustration of PAD-XOR based Encryption Framework.

The PAD-XOR based encryption method has been widely adopted in prior work [10], [15], [17]. The main advantage is that it can overlap the PAD generation process with memory fetching time to mitigate the encryption impact on performance. The illustration of such a framework is demonstrated in Figure 2. For encryption, some status variables, such as data address, are first used to generate crypto-PAD in the PAD generator. Then, the crypto-PAD XOR with *plain-data* (a.k.a. data to be encrypted) to get *cipher-data* (a.k.a encrypted data). The decryption is the reverse process of the encryption. The plain-data can be restored by XOR of cipher-data and the same crypto-PAD.

Kong's work is a encryption design of PAD-XOR based encryption method. However, this work is inefficient for several reasons. First, the crypto-PAD is generated with AES algorithm. Since the timing overhead of AES is really high [10], [4], the PAD generation delay may not be fully hidden by data fetching time. Second, a counter is used to provide status variables for encryption. Thus, the counting numbers used for encryption have to be stored with cipher-data in NVM memory for decryption. Despite of extra storage, the crypto-PAD of a memory line is changed with each write operation and induces extra updates. Third, as mentioned in last subsection, it can degrade efficiency of write reduction techniques and cannot reuse hardware of wear-leveling.

III. Encryption Scheme

In this section, we first declare a feasible attack model to NVM main memory and introduce the definition of security strength. Then, the detailed encryption scheme is introduced. Note that we move most mathematical derivation to Appendix to make it concise and easy to follow. At last, we introduce the architecture of a NVM main memory with our encryption component integrated. In addition, the combination of encryption and wear-leveling is presented.

A. Attack Model & Security Strength

In this work, a reasonable attack model is based on the following assumptions to facilitate the attack.

- The attacker has the right to write specific data into NVM main memory as a normal user. These data can be used in later attack.
- NVM main memory can be physically obtained by attacker either during runtime execution or after power-off.
- NVM main memory can be plugged into an attack system which can scan out all cipher-data for attack.
- Sensitive data are stored in the memory line, the logical and physical addresses of which are denoted as LA and PA, respectively. The corresponding crypto-PAD for encryption of sensitive data is denoted as CP.

With these conditions, the goal of the attacker is to find out sensitive plain-data from cipher-data retained in NVM memory using statistical or computing methods. First, it can be proved that statistical method cannot work. The details can be found in VI-B of Appendix. The basic flow of computation attack is as follows,

- **Step 1** The attacker reads cipher-data from the memory line with any logical address LA' and physical address PA'. The corresponding crypto-PAD for encryption of sensitive data is denoted as CP'.
- **Step 2** Obviously, it should be ensured that PA has been scanned by the attacker. In the worst case, the attacker needs to scan the whole NVM main memory.
- **Step 3** The next step is to remove the protection of crypto-PAD (CP') through computing to get the sensitive plain-data. Detailed mathematical description can be found in VI-B of Appendix.

Conventionally, an encryption mechanism is considered as broken when the possibility of retrieving sensitive plain-data is greater than a predefined threshold number Pr_{th}. This number is normally set as 0.9 in real cases [9]. In this work, we use total computation required to achieve this threshold probability to present the strength of security. Assume that the possibility of retrieving sensitive plain-data with one computation is Pr_0. The total computation required to break encryption can be approximated as $\frac{1}{Pr_0}$ [12]. For example, the computation of Kong's counter-mode approach is 2^{128}, using a 128-bit key in AES algorithm [12], [6].

B. Design of PAD Generator

In Kong's approach, the status variables needed for encryption and decryption (e.g. counter, logic page ID, etc.) have to be stored in NVM main memory with cipher-data. Despite of the storage overhead, the information is also retained in NVM and may be leveraged during an attack. One straightforward solution is to store all these information in volatile memory, such as SRAM or registers. However, the overhead for volatile memory is not trivial. For example, about 50MB SRAM is required for a 4G NVM main memory in Kong's approach. In order to solve this problem without inducing significant design overhead, we propose a novel design of PAD generator, as illustrated in Figure 3.

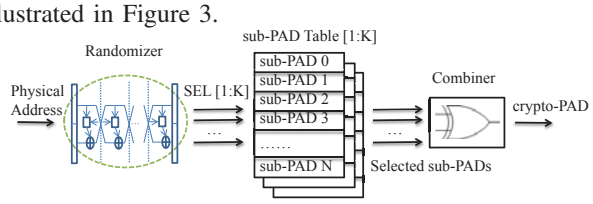

Fig. 3. Structure of PAD generator.

Our design of PAD generator is composed of three main components including *K sub-PAD Tables*, a *Randomizer*, and a *Combiner*. The functions of these components are introduced

as follows. In order to simplify description, we assume that there are 2^N memory lines in the whole NVM main memory and the size of each memory line is M-Byte.

sub-PAD Table This is the component storing information to generate crypto-PAD. Each table is made by volatile memory (e.g. SRAM) that contains $(N+1)$ binary sequences named as *sub-PADs* in this work. For the i_{th} sub-PAD table, these sub-PADs are denoted as $(S_0^i, S_1^i, \ldots, S_N^i)$. The size of each sub-PAD is equal to that of a memory line (a.k.a. M-Byte). The factor K is related to the security strength requested. The security strength can be increased with a larger K. However, the storage overhead is also increased at the same time. The optimization analysis is shown in Section IV-C. These sub-PADs are randomly generated when the system is initialized, and are **unknown** to any users. Note that there are various methods of generating these sequences. For example, we can use random number generator (RNG) [16], which can generate random numbers using machine noise.

Randomizer This component is used to generate selection signals used by sub-PAD tables. As shown in Figure 3, the input of randomizer is the physical address of a memory line containing requested data. The selection signal is generated individually for each sub-PAD table. For example, SEL_i is a N-bit binary number for i_{th} sub-PAD table. It is denoted as $(a_1^i, a_2^i, \ldots, a_{(N)}^i)$. Since the address is an N-bit binary sequence, the transformation from the address to SEL_i can be considered as a $N \rightarrow N$ mapping process. There are various realizations for such a randomizer. For example, the Feistel Network mentioned in Section II can be employed, as illustrated in Figure 1. Note that the randomizer also needs secret keys that are stored in volatile memory and are randomly generated at power-on.

Combiner This component is used to calculate the final crypto-PAD based on the sub-PADs selected by K selection signals. The calculation is described as follows,

$$CP = CP^1 \oplus CP^1 \oplus CP^2 \ldots \oplus CP^K \quad (1)$$

$$CP^i = S_0^i \oplus (a_1^i \cdot S_1^i) \oplus (a_2^i \cdot S_2^i) \oplus \ldots \oplus (a_N^i \cdot S_N^i) \quad (2)$$

, where we have $(i = 1, 2, .., K)$. Here, CP refers to the final crypto-PAD; CP^i refers to the XOR result of sub-PADs from i_{th} sub-PAD table.

We can find that the combiner is composed of an array of XOR gates. There is a design trade-off between performance and hardware overhead. The more XOR gates we have, the less latency we can achieve to generate the crypto-PAD. The hardware overhead is increased at the same time. Since the process of generating crypto-PAD can be overlapped with data fetching from NVM main memory, we can optimize the number of XOR gates to reduce hardware overhead without inducing timing overhead. The related analysis can be found in Section IV-C.

With this crypto-PAD generator, our encryption design can satisfy the three requirement listed in Section I. First, since both sub-PADs and mapping relationship are unknown to users, our design can achieve a high security strength easily. The detailed proof is introduced in Section VI. Second, the mapping process is faster than AES algorithm and storage overhead of sub-PAD table is trivial. The detailed analysis is presented in evaluation section. Third, the encryption process does not induce extra write intensity and has no impact on write reduction techniques. It is because the crypto-PAD corresponded to each physical address is fixed and the calculation of crypto-PAD from sub-PADs only involves XOR operations. Thus, the unmodified bits in write data are kept the same after encryption. More important, we can find that the function of randomizer in PAD generator is similar to that used in wear-leveling design [14], [16]. It provides the potential co-operation and hardware reuse between encryption and wear-leveling components in NVM main memory, as introduced in the next subsection

C. NVM Main Memory with Encryption

The architecture of NVM main memory is shown in Figure 4(a). Both wear-leveling and our encryption components are included in memory controller. Note that we hide other functional components of memory controller in the Figure to make it clear to read. The overlap between these two components represents hardware reuse. The operations of NVM main memory is described as follows,

- **Initialization** During the initialization power-on, sub-PADs are randomly generated and stored in sub-PAD table located in memory-controller. At the same time, both mapping relationships in wear-leveling components and encryption components are also initialized randomly (i.e. keys of wear-leveling and encryption components are randomly generated) .
- **Write operation**
  - **Step 1** When there is a write back from last level cache (LLC), the physical address of corresponding memory line is output from wear-leveling component and sent to the crypto-PAD generator.
  - **Step 2** After the crypto-PAD is generated based on SEL from sub-PAD selector, the cipher-data is calculated by an XOR operation between original write data and crypto-PAD.
  - **Step 3** The cipher-data is updated to NVM memory array with write reduction technique, such as DCW.
- **Read operation**
  - **Step 1** When there is a read request from LLC, the address of corresponding memory line is sent to sub-PAD selector. Physical address of the memory line is calculated through wear-leveling component at the same time.
  - **Step 2** The crypto-PAD is generated in parallel with data fetching process from NVM memory array. The original data is calculated by an XOR operation between cipher-data and crypto-PAD.
  - **Step 3** The original data is sent to LLC.
- **Normal Shutdown** When the system is powered off normally, user has two options:
  - **Option 1** When the user wants to keep data in NVM main memory for later restoration, we need to store

status of PAD generator, including sub-PAD tables, keys used for mapping in randomizer, etc. These information also need to protected in NVM main memory. For example, user can provide a password to encryption them before shutdown. Note that this encryption is not the responsibility of our encryption design.

  - **Option 2** If the user do not want to keep data in NVM main memory, no extra operations are needed. Those information in PAD generator disappear after power-off.

- **Wear Leveling** During the wear leveling process, a logic address can be re-mapped to a different physical address. This process is similar to that in a NVM main memory without encryption. The data in old physical address are read out then written into the new physical address. After using encryption, the only difference is that data are decrypted first in the read operation and encrypted in the write one. Obviously, the encryption/decryption process is transparent to the wear-leveling component.

D. Encryption Architecture

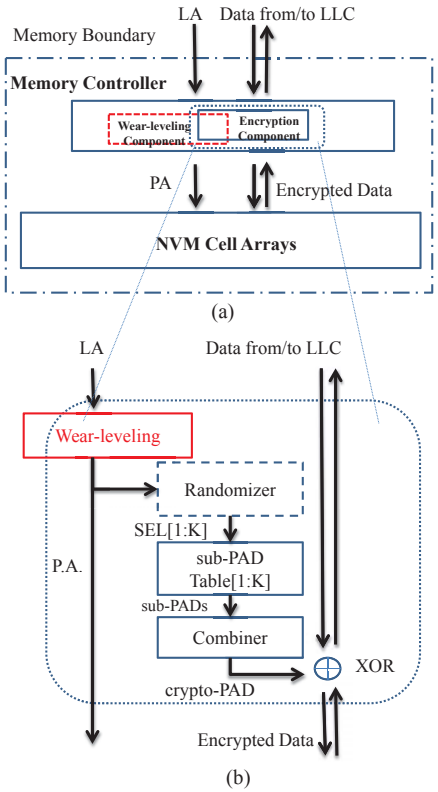

Fig. 4. Illustration of overall architecture.

The design details of encryption component is enlarged in Figure 4(b). We can find that the input of encryption component is physical address output from wear leveling component. Since the mapping between logic addresses and physical addresses in wear-leveling component is also unknown to users, it can help increase the computation needed to break the cipher-data. In fact, the mapping in wear-leveling component is similar to the function of randomizer in encryption component. Thus, encryption component can leverage the wear-leveling component to reduce the design complexity of its own randomizer. We provide two case studies with two popular wear-leveling techniques, namely Start-Gap and Security Refresh [14], [16].

- **Case for Start-Gap** Since the address mapping of Start-Gap is also based on Feistel Network, it can be directly used by randomizer. For example, a 3-stage Feistel Network is used in wear leveling of prior approach. The stage of Feistel Network used for randomizer can be reduced by three to achieve the same security strength.
- **Case for Security Refresh** When the Security Refresh is employed as wear leveling, the stage number of Feistel Network used to achieve the same security strength can be reduced by two. The detailed proof can be found in VI-B of Appendix.

IV. Evaluation

In this section, we first introduce experimental setup. Then, we compare our design using a fixed configuration with other approaches. At last, we provide a detailed sensitivity analysis with various configurations.

A. Experiment Setup

We evaluate performance with a full system cycle accurate simulator *gem5* [3]. The system is configured as one Intel *i5*-like processor, similar to those used in modern products[2]. A PCM based NVM main memory is employed for evaluation. The detailed parameters of processor and memory hierarchy are listed in Table I. We select 15 workloads from SPEC CPU 2006 benchmark suite for simulation on *gem5*. For each simulation, one billion instructions are executed. We assume that both PCM and CMOS technologies nodes are 45*nm*. In order to evaluate energy consumption, we record exact read operations and updated bits in each write operations to calculate the dynamic energy of NVM main memory. The static power of NVM main memory is extracted from the tool NVSim [7]. The dynamic and static energy of encryption logic are generated by synthesis tool and calibrated with prior approaches to provide a fair comparison [4], [10].

TABLE I
Detailed Simulation Setup.

Processor Configuration
2GHz, Issue Width:8, Fetch Width:8, INT/FP FUs:8/8 LD/ST: 24/24, Branch penalty: 6cycles ROB entries: 192, Fetch Q: 56, INT/FP registers: 128/128
Cache Configurations
DL1/IL1: 32/32KB, 2-way, 64B, R/W: 2/2-cycle, private L2: 1MB, 8-way, 64B, R/W: 10/10-cycle, share
Memory Configurations
4GB, PRAM, Security Refresh Wear leveling, 256Byte per line 1333MHz, 160/320-cycle ave. lat., Write Buffer: 16-entry

We compared results of four memory systems with different encryption designs, in respect of performance, lifetime, energy consumption, and hardware overhead. The baseline is a NVM main memory without using data encryption. The second system is the one that directly encrypts data with AES algorithm using 128-bit key (labeled as *Direct-AES*). The third NVM

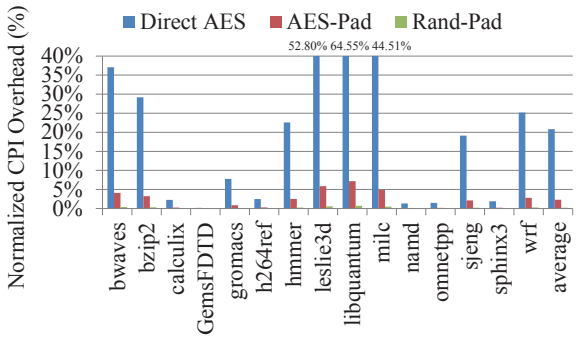

Fig. 5. Comparison of performance overhead with different designs.

main memory uses PAD-XOR based encryption method as introduced in Kong's work and is labeled as *AES-PAD*. The forth system employs our PAD-XOR based encryption scheme using a 9-stage Feistel Network as randomizer together with a single-level Security Refresh as the wear-leveling technique. It should be addressed that our design also works well with other wear-leveling techniques. The results of forth system is labeled with *Rand-PAD*. Note that we do not include results of i-NVMM approach because it cannot achieve a run-time protection to all data in NVM main memory.

B. Comparison of Experimental Results

All results in this section are based on configuration in Table I, In addition, four sub-PAD tables are used in PAD generator. We optimize PAD generator considering performance and area overhead (a.k.a. Case-2 in Subsection IV-C).

1) Performance Evaluation: We compare performance overhead caused by encryption in Figure 5. All results are normalized to baseline without using any encryption techniques. Obviously, memory encryption and decryption processes have impact on latency of read and write on the main memory.

For the case of Direct-AES, the performance is degraded as AES encryption and decryption latency are directly added to latency of write and read operations, respectively. For the case of *AES-PAD*, the performance overhead is moderate because both decryption and encryption latency can be partially hidden with data fetching and updating processes in read and write operations, respectively. However, due to long latency of AES algorithm (e.g. 190-cycle), the overhead cannot be fully covered. On average, the performance overhead is about 2%.

The results also demonstrate the performance overhead using our encryption method is trivial because we avoid using AES algorithm. The latency of generating crypto-PAD in our design can be fully hidden by data fetching and updating. The only extra latency comes from the XOR operation to generate cipher-data. Thus, the average overhead for all workloads is only 0.1%.

2) Lifetime Evaluation: The encryption algorithms like AES, involve data diffusion to keep data security. The data diffusion, however, leads to write amplification which significantly amplify write overhead. Since crypto-PAD for a physical address is fixed before attack, there are no extra write induced with our encryption design.

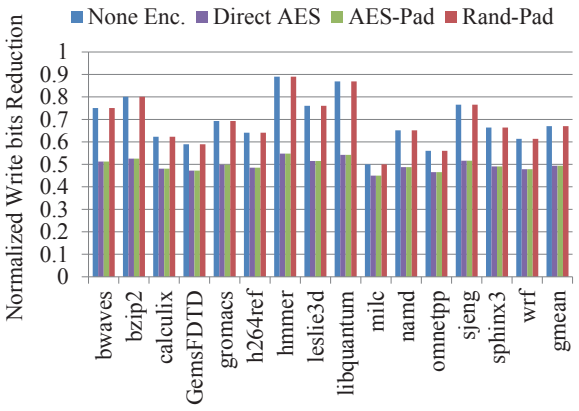

Fig. 6. Comparison of write reduction for different designs

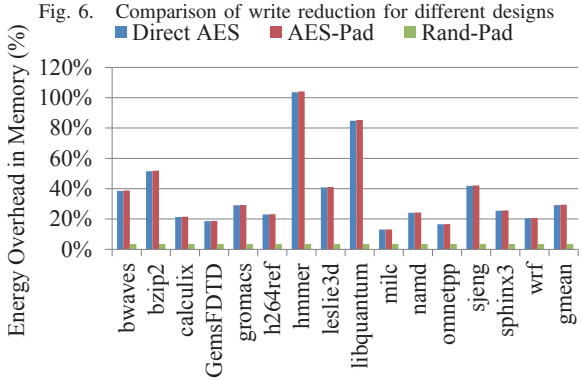

Fig. 7. Comparison of energy overhead for different designs.

We compare our work with Direct-AES and AES-PAD cases, when the DCW technique is employed for all workloads, as shown in Figure 6. The y-axis represents the normalized total number of write bits reduction. We also provide the case without using any encryption method as the baseline. The results prove that our encryption method does not incur extra write intensity. As a comparison, the number of write bits reduction is reduced by about 30% percent on average, when AES algorithm is used in encryption. Since the lifetime of NVM memory is only related to write intensity. We can conclude that using our encryption method can increase the lifetime of NVM memory by 67% compared to those approaches using AES algorithm.

3) Energy Evaluation: The system using our security scheme can also benefit from low energy consumption of our light-weight encryption process, compared to other approaches. In addition, the write energy reduction is reduced because there are no extra write induced with our encryption scheme. We conduct the experiment with configurations provided by previous works [4][1], and compare energy consumption for cases of Direct-AES encryption, AES-PAD based method, and our RAND-PAD. The total energy consumption including encryption and decryption energy is shown in Figure 7. The results show that total memory energy overhead is about 4% on average with our encryption method, compared to the baseline without using any encryption. For comparison, the average energy overhead of AES-PAD is about 29%.

4) Hardware Area Evaluation: The extra design overhead of our scheme is also lower than those of designs using

TABLE II
AREA AND STORAGE COST COMPARISON

	Direct AES	AES-Pad	Rand-Pad
Logic Part Transistors(K)	591.1	591.4	462.1
Extra Logic Area (mm^2)	2.0	2.0	1.5
Extra Memory Storage	8MB SRAM	65MB NVM	25.1KB SRAM

TABLE III
COMPARISON AMONG DIFFERENT DESIGN CONFIGURATIONS

Config.	Design	Storage Overhead (KB)	Area Overhead (mm^2)	Combiner Latency (cycle)	C-/P-Data Latency (cycle)
4G 256B	Case-1	25.1	6.2	10	1
	Case-2	25.1	1.5	39	1
	Case-3	25.1	1.1	51	5
32G 256B	Case-1	28.1	6.6	11	1
	Case-2	28.1	1.7	43	1
	Case-3	28.1	1.2	57	5
4G 1024B	Case-1	92.1	23.3	9	1
	Case-2	92.1	1.5	36	1
	Case-3	92.1	1.0	47	18
4G 4096B	Case-1	336.1	84.5	8	1
	Case-2	336.1	1.3	33	1
	Case-3	336.1	0.9	43	73

AES algorithms. To encrypt the NVM main memory of 4GB size, the extra storage for sub-PAD table is only about $25KB$ SRAM. The capacity needed for our scheme and relative mechanisms are list in Table II. The parameters used here are obtained from prior works [4], [1].

We evaluate overhead of logic part by calculating total numbers of transistor used. In addition, the area overhead is also estimated using synthesis tool. We compare these numbers in Table II. The total transistor numbers for cases of *Direct AES* and *AES-PAD* are similar (about $590K$). It means that most overhead comes from the realization of AES algorithm. For our encryption design, about $462K$ transistors are needed because we avoid using AES algorithm. Note that the whole memory line is encrypted in parallel in our design. The overhead can be further reduced if the XOR is completed in multiple steps. The latency, however, will be increased at the same time. The extra storage required are also compared in Table II. The storage overhead of encryptions of first two cases comes from internal storage required by AES. The overhead of our design, however, is significantly reduced because we only need to store the sub-PAD tables.

C. Sensitivity Analysis

In the last subsection, memory system configuration and design parameters of our encryption component are fixed for evaluation. Obviously, the metrics of security strength, performance, energy consumption, and area overhead can be affected by various memory system configurations and encryption design parameters. In order to provide the comprehensive evaluation for different scenarios, we analyze the sensitivity of design metrics for these design factors.

As shown in Table III, we discuss sensitivity of storage overhead, extra hardware design, and latency of generating crypto-PAD for different memory configurations and designs of PAD generator. We first introduce three possible PAD generator designs, which have different realizations of sub-PAD combiner and calculation of cipher-data.

- **Case-1** For the combiner design, each combination operation is processed with the granularity of a memory line size (M-Byte). It means that an operation like $S_0 \oplus S_1$ in Equation (2) is processed with $8M$ XOR gate at the same time. Similarly, the calculation of cipher-data is also processed with the same granularity. In other words, the XOR operation between crypto-PAD and plain-data is also processed with $8M$.
- **Case-2** The processing granularity of combination operations is equal to a word size (4-Byte). It means that an operation like $S_0 \oplus S_1$ is completed in multiple rounds ($\frac{M}{4}$). For the calculation of cipher-data, the process granularity is still kept as a memory line size.
- **Case-3** In this design, the processing granularity for both combination operation and calculation of cipher-data is set to a word. Compare to Case-2, the difference is that calculation of cipher-data is completed in $\frac{M}{4}$ rounds.

These three design styles have an impact on hardware overhead and performance, which are compared in Table III. A larger processing granularity requires more XOR gates, which result in more extra transistors. For example, the area overhead is increased from $1.1mm^2$ to $6.2mm^2$ when we use Case-1 design instead of Case-3 design, with a 4GB NVM memory and the memory line size set to $256B$. On the other hand, the latency of combining crypto-PAD is reduced to 11-cycle, and the latency of calculating cipher-data is reduced to 1-cycle.

The trade-off between performance and hardware overhead should be explored to obtain an optimized design. For a read operation, the combination of sub-PADs can be processed in parallel with data fetching. Since combination latency can be hidden by data fetching, the read performance is not improved after using design of Case-1 or Case-2. The latency of calculating plain-data, however, is added to that of a read operation. Thus, for a read operation, the design of Case-2 is preferred. The case for write operation is different because both latencies from combination of sub-PADs and calculation of cipher-data are added. Fortunately, the extra latency of encryption is trivial compared to that of a write operation. In addition, the write operation is not in the critical path of data access. Consequently, we select Case-2 as the optimal design, which is used for evaluation in previous subsection.

The impact of memory capacity and memory on design overhead is also compared in Table III. We can find that the storage overhead increases significantly with the size of a memory line. It is because the storage overhead mainly comes from the sub-PAD Table. Since the size of a sub-PAD is equal to that of a memory line, the storage overhead increases linearly with the memory line size. For the similar reason, the logic overhead increases at the same time with Case-1.

Figure 8 demonstrates the relationship between security strength and design factors including memory capacity, memory line size, and number of sub-PAD tables. Note that the log based z-axis represents the lower-bound of computation needed to break encryption. We have following two observations based on the results. First, the security strength is close related to the number of sub-PAD tables (a.k.a K in

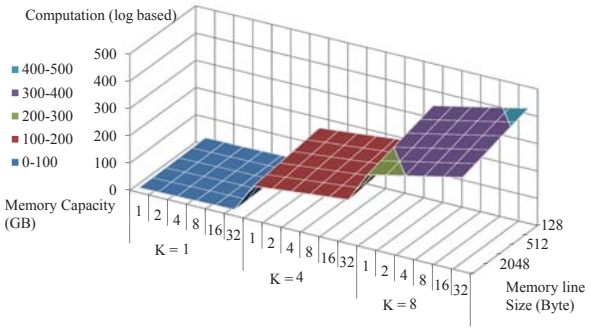

Fig. 8. Sensitivity analysis of security strength

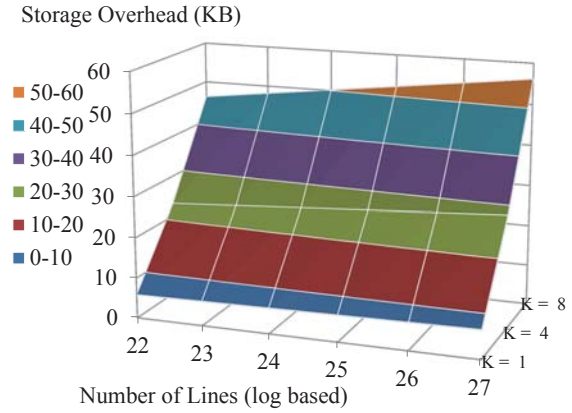

Fig. 9. Sensitivity analysis for storage overhead.

Equation (1)). We can find that the computation to break encryption increases exponentially as K increases from 1 to 4. The reason is because the space of mapping function of randomizer is increased exponentially with K. The results show that, in order to achieve a comparable computation to prior approach (e.g. 2^{128}), we should set the number of sub-PAD tables at least to 4. Second, the security strength improves as the memory capacity increases. It is because the space of mapping function of randomizer increases polynomially with the total number of memory lines. Since the number of memory lines increases with the memory capacity when the size of a memory line is fixed, computation required to break the encryption increases at the same time. For the same reason, the security strength is also improved when we decrease the size of a memory line with the memory capacity fixed.

There is a trade-off between security strength and storage overhead for encryption. Apparently, the storage overhead increases when we have more sub-PAD tables. In addition, it also increases when have more memory lines because the entry. The related results of storage overhead can be found in Figure 9. Fortunately, the storage overhead can be controlled within 60KB for different configurations. Consequently, the storage overhead is not a critical issue for our design. As a summary, we select the optimized encryption design with the following configuration. We choose Case-2 design for PAD generator. The number of sub-PAD tables is set to 4. The memory line size is equal to 256-Byte.

V. Conclusion

The security challenge caused by non-volatility of NVM becomes a critical obstacle for adoption of NVM main memory. Data encryption is necessary to protect NVM main memory from malicious attacks. However, the schemes directly use AES algorithms are inefficient because they are not optimized based on features of NVM main memory. The encryption proposed in this work can overcome the limitations in prior approaches with trivial overhead in respect of performance, energy consumption, and hardware area. More important, the lifetime of NVM main memory will not be affected.

References

[1] http://www.vlsitechnology.org/html/lib-densities.html.

[2] http://www.intel.com/idf/. 2013.

[3] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. 39(2):1–7, Aug. 2011.

[4] S. Chhabra and Y. Solihin. i-nvmm: a secure non-volatile main memory system with incremental encryption. In *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, pages 177–188, New York, NY, USA, 2011. ACM.

[5] S. Cho and H. Lee. Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 347–357, New York, NY, USA, 2009. ACM.

[6] W. Diffie and M. Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67(3):397–427, 1979.

[7] X. Dong, C. Xu, Y. Xie, and N. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(7):994–1007, 2012.

[8] C. Kim. Advances in memory technology. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 1105–1105. VLDB Endowment, 2006.

[9] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in CryptologyłCRYPTO96*, pages 104–113. Springer, 1996.

[10] J. Kong and H. Zhou. Improving privacy and lifetime of pcm-based main memory. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 333–342, 2010.

[11] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting phase change memory as a scalable dram alternative. *SIGARCH Comput. Archit. News*, 37(3):2–13, June 2009.

[12] J. Patarin. Luby-rackoff: 7 rounds are enough for $2^{n(1-\varepsilon)}$ security. In *Advances in Cryptology-CRYPTO 2003*, pages 513–529. Springer.

[13] M. Qureshi, M. Franceschini, and L. Lastras-Montano. Improving read performance of phase change memories via write cancellation and write pausing. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–11, 2010.

[14] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 14–23, New York, NY, USA, 2009. ACM.

[15] B. Rogers, S. Chhabra, M. Prvulovic, and Y. Solihin. Using address independent seed encryption and bonsai merkle trees to make secure processors os- and performance-friendly. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 183–196, Washington, DC, USA, 2007. IEEE Computer Society.

[16] N. H. Seong, D. H. Woo, and H.-H. S. Lee. Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pages 383–394, New York, NY, USA, 2010. ACM.

[17] W. Shi, H.-H. S. Lee, M. Ghosh, C. Lu, and A. Boldyreva. High efficiency counter mode security architecture via prediction and precomputation. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, pages 14–24, Washington, DC, USA, 2005. IEEE Computer Society.

[18] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A novel architecture of the 3d stacked mram l2 cache for cmps. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 239–249, 2009.

[19] G. Sun, D. Niu, J. Ouyang, and Y. Xie. A frequent-value based pram memory architecture. In *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, pages 211–216, 2011.

[20] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie. Hybrid cache architecture with disparate memory technologies. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 34–45, 2009.

[21] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 14–23, 2009.

VI. Appendix

All symbols used for derivation are listed in Table IV. To quantitatively analyze the security strength, we set up system configurations as follows, $N = 24, M = 256, K = 4$. We use a 24-bit-width 9-sage Festel Network in this case. Thus, we need 4 sets of 12-bit-width keys (totally 36 keys).

TABLE IV
SYMBOLS USED IN THIS WORK

Symbol	Description
N	Length of address
M	Size of a memory line (Byte)
K	Number of sub-PAD tables
LA	Logical address of sensitive data before attack
PA	Physical address corresponded to LA before attack
P_{LA}	Sensitive data at LA
C_{PA}	Retained cipher-data at PA before attack
SEL	Selection signal of sub-PAD tables for PA before attack
CP_{SEL}	Crypto-PAD generated by SEL before attack
SEL_i	Selection signal of the i_{th} sub-PAD table before attack
a^i_j	j_{th} bit of the SEL_i before attack
S^i_j	j_{th} sub-PAD from the i_{th} sub-PAD table before attack
$CP^i_{SEL_i}$	$= S^i_0 \oplus (a^i_1 \cdot S^i_1) \oplus (a^i_2 \cdot S^i_2) \oplus \ldots \oplus (a^i_N \cdot S^i_N)$
LA'	Logical address the attacker read from during attack
PA'	Physical address of LA during attack. (Note that $PA' = PA$)
$P'_{LA'}$	Data that attacker read out from LA' during attack
$C'_{PA'}$	Namely C_{PA}
SEL'	Selection signal of sub-PAD tables for PA' during attack
$CP'_{SEL'}$	Crypto-PAD generated from SEL' during attack
SEL'_i	Selection signal of the i_{th} sub-PAD table during attack
a'^i_j	j_{th} bit of the SEL'_i during attack
S'^i_j	j_{th} sub-PAD from the i_{th} sub-PAD table during attack
$CP'^j_{SEL'_i}$	$= S'^i_0 \oplus (a'^i_1 \cdot S'^i_1) \oplus (a'^i_2 \cdot S^i_2) \oplus \ldots \oplus (a'^i_N \cdot S'^i_N)$

During the attack described in subsection III-A, we have the following relationship among data and addresses. Note that the read and write operation must go through our encryption component. Thus, the decryption in read is similar to encryption in write. It prevents attacker directly getting the cipher-data:

$$P'_{LA'} = P_{LA'} \oplus CP_{SEL} \oplus CP'_{SEL'} \tag{3}$$

Since we have $K = 4$ for the sub-PAD table design, the crypto-PAD CP_{SEL} and $CP'_{SEL'}$ are described in following equations.

$$CP_{SEL} = CP^1_{SEL_1} \oplus CP^2_{SEL_2} \oplus CP^3_{SEL_3} \oplus CP^4_{SEL_4} \tag{4}$$

$$SEL = (SEL_1, SEL_2, SEL_3, SEL_4) \tag{5}$$

$$CP'_{SEL'} = CP'^1_{SEL'_1} \oplus CP'^2_{SEL'_2} \oplus CP'^3_{SEL'_3} \oplus CP'^4_{SEL'_4} \tag{6}$$

$$SEL' = (SEL'_1, SEL'_2, SEL'_3, SEL'_4) \tag{7}$$

A. Basic Properties

We first introduce two properties of our encryption design, which are used for derivation in the rest of this section.

Property 1: The value of each bit of crypto-PAD follows the Bernoulli distribution.

As shown in Equation (1) (2), assume that there are r sub-PADs selected from all sub-PAD tables to generate the crypto-PAD. For any bit of crypto-PAD, the possibility that it is equal to bit '1' is calculated as,

$$\sum_{i=0}^{r/2} \binom{r}{2i} (\frac{1}{2})^{2i} (\frac{1}{2})^{r-2i} = \frac{1}{2} \tag{8}$$

Property 2: We can assume that each crypto-PAD is spatially and temporally unique.

For spatially uniqueness, the proof by contradiction is provided as follows. Let CP_i and CP_j denote crypto-PADs for i_{th} and j_{th} memory lines ($i \neq j$), respectively. we can prove the possibility that $CP_i = CP_j$ is near zero. From Equation (2), we get

$$CP_i = S_{i_1} \oplus S_{i_2} \oplus \ldots \oplus S_{i_{r_1}} \tag{9}$$

$$CP_j = S_{j_1} \oplus S_{j_2} \oplus \ldots \oplus S_{j_{r_2}} \tag{10}$$

If $CP_i = CP_j$, we have

$$S_{i_1} \oplus S_{i_2} \oplus \ldots \oplus S_{i_{r_1}} \oplus S_{j_1} \oplus S_{j_2} \oplus \ldots \oplus S_{j_{r_2}} = 0 \tag{11}$$

Similar to proof of Property 1, the possibility that any bit of two crypto-PADs are equal to each other is

$$\sum_{k=0}^{(r_1+r_2)/2} \binom{r_1+r_2}{2k} (\frac{1}{2})^{2k} (\frac{1}{2})^{r_1+r_2-2k} = \frac{1}{2} \tag{12}$$

Since all bits of sub-PADs are independent from each other, the possibility that all bits of two crypto-PADs are equal can be calculated as $(\frac{1}{2})^{8\times M}$. Note that $8 \times M$ represents the total number of bits in each crypto-PAD (a.k.a. length of a memory line). This possibility is extremely small for a normal design of NVM main memory, since the length of a memory line is normally large than 256-Byte.

For temporal uniqueness, it means that the possibility that $CP_i = CP'_j$ is also near to zero. The proof is similar to that for spatial uniqueness.

B. Security Analysis

Based on the two properties in previous subsection, our crypto-PAD can hide partial information of plain-data so that the data read out by attacker do not have statistical correlation to be used for attack [17]. Thus, the break process can only be achieved through computing instead of using statistical attack.

For computational analysis, the attack is based on Equation (3) and the character of XOR operation. It means that

CP_{SEL} and $CP'_{SEL'}$ can only be eliminated by doing XOR among readout data [15]. According to Equation (4) and (6), CP^i or CP'^i $(i = 1,2,..,K)$ should be eliminated. In order to describe the computation attack, we introduce adjoint vector of SEL_i as

$$\overrightarrow{SEL_i} = [a_1^i, a_2^i, \ldots, a_N^i]^T \tag{13}$$

Then, the elimination of $CP^i_{SEL_i}$ is based on Theorem 1 and it's similar with $CP'^i_{SEL'_i}$.

Theorem 1: $CP^i_{SEL_i}$ can only be eliminated by XOR operation when the following two conditions are satisfied:

$$(\overrightarrow{SEL_i} + \sum_{j=1}^{r} \overrightarrow{SEL_i^j}) = 0 \ (mod 2) \tag{14}$$

$$r \ mod \ 2 = 1 \tag{15}$$

Proof: Based on Equation (14), each sub-PAD except S_0^i appears even times in the following expression

$$CP^i_{SEL_i} \oplus CP^i_{SEL_i^1} \oplus CP^i_{SEL_i^2} \ldots \oplus CP^i_{SEL_i^r} \tag{16}$$

Since we have $r \ mod \ 2 = 1$, S_0^i also appears even times in the expression. Thus, we have

$$CP^i_{SEL_i} \oplus CP^i_{SEL_i^1} \oplus CP^i_{SEL_i^2} \ldots \oplus CP^i_{SEL_i^r} = 0 \tag{17}$$

According to Property 2, we can quickly know that the the inverse theorem of above is true.

For instance, if $SEL_i = 6 = (1,1,0)$, $SEL_i^1 = 1 = (0,0,1)$, $SEL_i^2 = 4 = (1,0,0)$, $SEL_i^3 = 3 = (0,1,1)$, so $\overrightarrow{SEL_i} + \sum_{j=1}^{3} \overrightarrow{SEL_i^j} = 0 \ (mod 2)$, namely $[1,1,0]^T$ can be linearly expressed by $[0,0,1]^T, [1,0,0]^T, [0,1,1]^T$, thus $CP_6^i \oplus CP_1^i \oplus CP_4^i \oplus CP_3^i = 0$ and CP_6^i is eliminated.

There is one extreme case that we need to address. If we XOR all the readout data together, the pad will be eliminated because that their sum must be 0 and there are even number of memory lines. According to Thereon 1, the CP and CP' would be eliminated. The result is equal to the XOR of all plain-data before attack. If the attacker knows all other plain-data except the sensitive one to attack, the sensitive data can be obtained. However, it is impossible that the attacker get all other plain-data.

Based on Theorem 1, we can calculate the computation of breaking our encryption design under different configurations.

Conclusion 1: If the mapping from PA to SEL_i $(i = 1,2,..,K)$ is truly random, the lower bound of computation is 2^{7N}.

Proof: Because of the independence between $CP^{(')i}$ $(i = 1,2,..,K)$ (when $K = 4$, there are 8 in total), when using Thereon 1 to eliminate CP or CP', adjoint vectors of $SEL_i^{(')}i$ $(i = 1,2,..,K)$ are in different linear spaces. Based on Equation (3), to break the encryption, the adjoint vector of $[SEL, SEL']^T$ also needs to satisfy Equation (14). Since $SEL_i^{(')}$ is random and independent, this happens with a possibility of 2^{-7N}. So the computation to break encryption is at least 2^{7N}.

Conclusion 2: When the randomizer is implemented with an 11-stage Feistel Network (FN), the lower bound of computation is 2^{132}. It means that such a design can achieve the similar security strength as an approach based on AES with a 128-Bit key.

Proof: If we the attacker cannot differentiate the FN from an ideal randomizer, the computation is at least $2^{7N} = 2^{168} >> 2^{132}$. Otherwise, the attacker needs a specific amount of queries to distinguish FN from an ideal randomizer [12]. The query refers to a $< PA, \ SEL >$ pair. The queries are obtained based on the inverse theorem of Theorem 1.

In order to calculate the lower bound of computation, we relax the constraints for the attacker. We assume that the attacker can write data to the whole memory space. To simplify the discussion, we further assume attacker write all zeros to the whole memory space. Then, Equation (3) is changed to

$$P'_{LA'} = CP_{SEL} \oplus CP'_{SEL'} \tag{18}$$

Based on Equation (18), the computation to get one query is at least 2^{2KN}. This is because for $CP^i_{SEL_i}$ $(i = 1,2,..,K)$ or $CP'^j_{SEL'_j}$ $(j = 1,2,..,K)$. they are independent from each other and the computation to get one query for each is 2^N. Since we have $K = 4$, the total computation to get a query is 2^{192}, which is even larger than that (2^{132}) using brutal force to break FN directly.

So far, we provide security analysis without using any wear-leveling techniques. When the wear-leveling is induce, the input of FN (a.k.a physical address) is also unknown to any users. Thus, extra effort is needed to identify the PA correlated to LA' to attack. In other words, we can reduce the design complexity of randomizer to achieve the same level of security strength. As a consequence, we have the following conclusions.

Conclusion 3: When the randomizer is implemented with a 9-stage Feistel Network (FN) and a single-level Security Refresh is used for wear leveling, the lower bound of computation is 2^{132}.

Proof: With wear leveling, Equation (3) becomes:

$$P'_{LA'} = C'_{PA'} \oplus CP'_{SEL'} = P_{LA} \oplus CP_{SEL} \oplus CP'_{SEL'} \tag{19}$$

We simplify address mapping of Security Refresh as $PA = LA \oplus K_s$, and Equation (19) becomes

$$P'_{LA'} = P_{LA' \oplus K_s} \oplus CP_{SEL} \oplus CP'_{SEL'} \tag{20}$$

Similar to proof of conclusion 2, we need another computation of 2^N to identify the PA correlated to LA' to attack. Thus, we can reduce the randomizer to a 9-stage FN to achieve the similar security strength as an approach based on AES with a 128-Bit key. It is because the the computation to break a 9-stage FN is $2^{9\times 12} = 2^{108}$).