Pin Tumbler Lock: A Shift based Encryption Mechanism for Racetrack Memory

Hongbin Zhang[1], Chao Zhang[2], Xian Zhang[2], Guangyu Sun[2], and Jiwu Shu[1]
[1]*Department of Computer Science and Technology, Tsinghua University, 100084, China*
[1]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China*
[2]*Center for Energy-efficient Computing and Applications, Peking University, Beijing, 100871, China*
(zhanghb10@mails.tsinghua.edu.cn, {zhang.chao, zhang.xian, gsun}@pku.edu.cn, shujw@tsinghua.edu.cn)

Abstract— As various non-volatile memory (NVM) technologies have been adopted in different levels of memory hierarchy, the security issue of protecting information retained in NVM after power-off has become a new challenge, which results in extensive research on data encryption for NVM. Previous encryption approaches, however, have some limitations, such as high design complexity and non-trivial timing and energy overhead. Recently, an emerging NVM called racetrack memory (RM) has been widely investigated because of its advantages of ultra-high storage density and fast read/write speed. Besides these well-known advantages, we observe that the tape-like structure of RM cell and its unique shift operation can also be leveraged to facilitate NVM data encryption. Base on this observation, we propose an efficient shift based mechanism, named Pin Tumbler Lock (PTL), which completes encryption and decryption by shifting racetracks in several nanoseconds. Experimental results demonstrate that our design can achieve the same security strength of AES-128 with 3.1% performance overhead and 3.7% energy overhead and 1.56% storage cost and 1.6% area cost.

I. Introduction

Due to the well-known problem of Memory Wall, computer architects are searching out for alternatives of traditional SRAM/DRAM memory technologies to bridge the increasing gap between computing throughput and bandwidth of memory hierarchy. To this end, various non-volatile memory(NVM) technologies, such as PCM, RRAM, STT-RAM and FeRAM[11], have been extensively investigated because they have advantages of high storage density, low standby power, and fast access speed. In order to improve storage density, many digital product manufacturers are using NVM as embedded memory in cell phones, digital camera, pad and laptops etc. With widely adoption of these NVMs in different levels of memory hierarchy, a new security vulnerability has emerged. Simply speaking, information is retained in NVM long after the system is powered off. It enables an attacker with physical access to the system to retrieve sensitive information.

To overcome this problem, various encryption approaches for emerging NVMs have been proposed recently. There are several approaches that directly use AES to encrypt data before being stored in memory[5]. However, the AES process on critical path induces non-trivial timing overhead. The i-NVMM method encrypts the main memory incrementally according to the facts that the working set of application is much smaller than its resident set[9]. And only non-working part of resident set is encrypted to mitigate timing overhead. However, a critical problem of this mechanism is that sensitive information in working set of the application is not protected. Zhang *et al* proposed a PAD-XOR method to encrypt the main memory, which can provide run-time protection to all data in main memory with moderate timing and power overhead[14]. But this scheme focuses on PCM based main memory and it introduces extra sub-PAD tables for encryption.

Recently, a new type of emerging NVM, racetrack memory (RM) has attracted attention of researchers. Previous research has demonstrated that it can achieve ultra-high density by integrating multiple domains in a tape-like namowire as demonstrated in Figure 1. In addition, RM provides fast read/write access speed comparable to SRAM and can be used as cache and memory. As a type of NVM, RM also faces the challenge of data encryption. However, the encryption timing overhead of existing approaches is much higher than normal data access latency. Fortunately we observe that the tape-like structure of RM cell and its unique shift operation can also be leveraged to facilitate NVM data encryption. Base on this observation, we propose an efficient shift based mechanism, named Pin Tumbler Lock (PTL), which completes encryption and decryption by shifting racetracks in several nanoseconds. At the same time, the sufficient security strength is achieved with moderate design overhead.

The major contribution of this can be summarized as follow,

- To the best of knowledge, this is the first work that leverages the RM structure and shifting operations for NVM data encryption.
- Both encryption and decryption is completed with 4-stage Feistel Network and shift operations. AES software or hardware is avoided so that both design complexity and overhead is significantly reduced.
- A clear attack model is presented and we prove that our scheme can achieve the same or even higher security strength as prior works using AES algorithm.
- Our encryption mechanism is compatible with RM design for different levels of a memory hierarchy.
- Comprehensive evaluation is provided to show that our work achieves less timing/energy/storage overhead than existing approaches.

978-1-4673-9569-4/16/$31.00 ©2016 IEEE

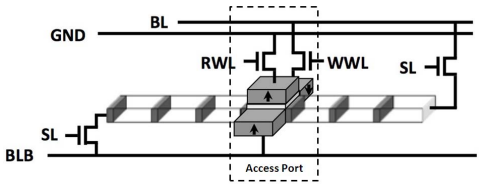

Fig. 1. Cell of Racetrack Memory

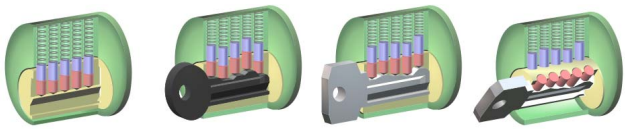

Fig. 2. Pin Tumbler Lock. (a) With no key. (b) With wrong key. (c) With right key. (d) Unlocked.[1]

II. Background and Motivation

A. Racetrack Memory background

Racetrack memory is an emerging non-volatile memory based on spintronic domain wall technology. A racetrack memory array contains multiple tape-like cells and their access ports. A racetrack cell is illustrated in Figure 1. It is a magnetic nanowire, containing multiple domains isolated by domain walls. The magnetization direction (arrows) of a domain is programmed to store either bit 1 or bit 0. The port can only read/write the domain aligned to it. Thus, besides read and write operations, a unique shift operation is introduced to move a domain to an access port before being accessed.

Read operation can be performed by comparing the resistance between Bit-line(BL) and Bit-line-bar(BLB), after turning on read-word-line (RWL). Write operation can be performed by selectively turning on write-word-line (WWL). Shift operations are based on a phenomenon called spin-momentum transfer caused by shift current. The shift current pulse is supplied by the transistors attached to the ends of the stripe, controlled by shift-line (SL). All domain walls move in the same direction with the same speed [15].

B. Pin Tumbler Lock

Pin tumbler lock is widely used in our daily life[1]. Like Figure 2 shows, without a key in the lock, the driver pins (blue) are pushed downwards, preventing the plug (yellow) from rotating. When an incorrect key is inserted into the lock, the key pins (red) and driver pins (blue) do not align with the shear line; therefore, it does not allow the plug (yellow) to rotate. Only the correct key can drive the pins and align the gaps between the key pins (red) and driver pins (blue) align with the edge of the plug(yellow), and unlock the lock through rotating.

Normally, a data block is distributed on multiple racetrack stripes to enable multiple bit access in parallel. As shown in Figure 3, a 128-bit data block is stored on 128 stripes. In a conventional RM design, all bits of the same block are stored in the same position of each stripe, as shown in Figure 3(a). Thus, we just need to apply the same shift command to each stripe for each read/write operation. However, after all data are stored,

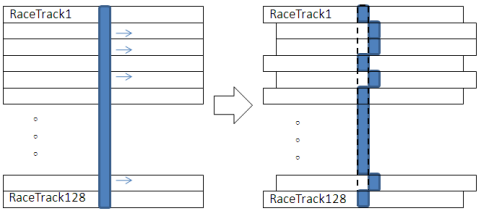

Fig. 3. Demonstration of shift based encryption scheme.(a)Before encryption.(b)After encryption.

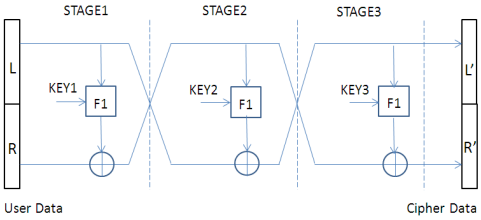

Fig. 4. Three-stage Feistel Network.

if we apply variant shift steps to these stripes on purpose, as shown in Figure 3(b), these bits are hidden in stripes just like using a pin tumbler lock. Without knowing the shift pattern of each stripe, the attacker cannot access the data directly. Like Figure 3 showed. Just like the Pin tumbler lock is locked, only the right shift key can restore the data.

Since shift operations across multiple racetracks can be operated in a few cycles in parallel, the encryption can be operated in ultrashort latency than software based methods or existing hardware based approaches. If shift currents of different amplitudes are applied to different row according to the linear current-velocity relationship of shift operation, all shifts can even be done in one cycle[10]. Which means the encryption can be completed with higher efficiency than conventional methods. With appropriate design, the shift based encryption method can provide security strength no less than AES-128 with less time and power consuming.

C. Random Number Generator

The random number generator(RNG) plays an important role in modern encryption system. RNG is a type of circuit which can generate random numbers using the physical random source (e.g. the electric noise[8]) or cryptography algorithm(e.g. the Feistel Network[6, 14, 7]). For the Feistel Network(FN), it can also transform a data into a pseudo random number and the transformation is invertible. In addition, with enough number of FN rounds, an adversary cannot distinguish the output of FN from the truly random numbers without acquiring enough plain-cipher queries[6]. Figure 4 shows the logic for a three stage FN. FN is widely used including in the Data Encryption Standard (DES).

With only shift based encryption, the schematic is not semantically security and data with strong patterns will be easily decrypted. For example, data with all ”0” or all ”1” will be the same after being shifted. So, before the shift operation, we use a four stage FN to transform data into random numbers.

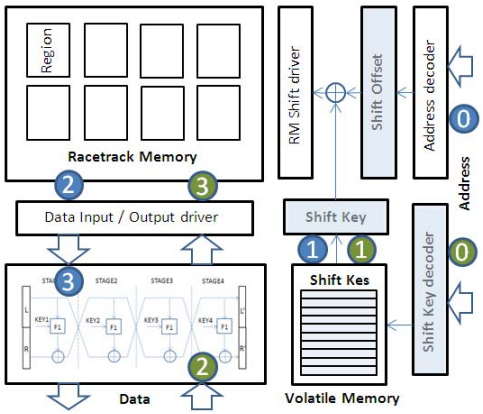

Fig. 5. The structure of secure racetrack on-chip memory.

III. PTL Encryption Mechanism Design

In this section, the detailed shift based encryption mechanism will be discussed. In general, the architecture of PTL encryption is shown in Figure 5. It is composed of racetrack on-chip memory, volatile memory, FN encryption module and other peripheral circuits. The racetrack on-chip memory is divided into regions, each region has a shift key which is stored in volatile memory. The volatile memory can be SRAM, DRAM. All the shift keys are generated from randomizer when the system is initialized. When the system is powered off, shift keys will disappear and the user data on the RM is protected. The keys of FN are also stored in volatile memory. As FN needs only several keys, the storage cost can be ignored. The other part of the architecture will be introduced in latter sections.

A. Attack Model

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 RM as a normal user. These data can be used in later attack.
- Racetrack memory can be physically obtained by attacker either during runtime execution or after power-off.
- Racetrack memory can be plugged into an attack system which can scan out all cipher-data for attack.

With these conditions, the goal of the attacker is to find out sensitive plain-data from cipher-data retained in RM memory using statistical or computing methods. It can be proved that statistical method cannot work. The details can be found in section IV-E. The basic flow of computation attack is as follows,

- **Step1** The attacker write specific data into racetrack memory in specific physical address.
- **Step2** The attacker physically get the racetrack memory after the system is powered-off.
- **Step3** The attacker scan all the cipher-data from the racetrack memory and crack the plain-data through statistical or mathematical method. Detailed mathematical description can be found in section IV-E.

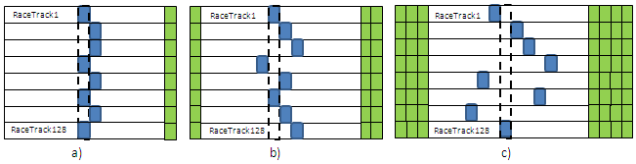

Fig. 6. (a)Region with 128bit shift key(key-width is 1). (b)Region with 256bit shift key(key-width is 2). (c)Region with 384bit shift key(key-width is 3).

B. Encryption Region and Shift Key

Many researchers has made sufficient work on RM and a quantitative modeling is introduced[12]. Our design is proposed based on this model. We define an encryption region as a basic unit which explain how the shift based encryption scheme works. As shown in figure 2, the encryption region has 128 racetracks and each racetrack has 64 bit. This region has 1KB capacity in total and data is stored in stripe across multiple racetracks. Each stripe has 16bytes which is defined an encryption word. The plain data will be encrypted or decrypted when its racetracks shift forward or back. We define the shift pattern as shift key and the key length of each racetrack is key-width. If key-width is 1, the racetrack has two shift option: standby or shift one bit to right. So the total possibility of cipher data is 2^{128}.

Shift keys are created by Randomizer as system is initialized. Shift key determines how racetrack shift its domain walls during data encryption. With different key-width, the region has different key length and security strength. Like the Figure 6 shows, the second region has shift key with 256 bits and each racetrack has 2 of it(key-width is 2), then it has 4 shift options: shift one bit to left, standby, shift one bit to right, shift two bits to right. The third region has shift key with 384 bits and each racetrack has 3 of it(key-width is 3), then it has 8 shift options: from shifting 3 bits to left, to shifting 4 bits to right. The yellow part of the region is redundant space to accommodate the bits shifted out of the racetrack, which will be discussed in section IV-D.

C. Redundant Domain Wall

The RM region needs extra redundant domain walls to contain the bits shifted out. For the region which has 128bit key, it need one stripe redundant bits on right. Like figure 3 shows. For the region which has 256-bit key, it need three stripe redundant bits, one of them on left and two of them on right. The overhead of redundant bits will be mentioned in section V.

What should be pointed out is, the redundant bits should be filled will data randomly during the system initialization. Because if these bits are filled with data with regular pattern when they leave factory, such as all ”0” or all ”1”, the attacker has chance to crack out the shift key by the pattern of redundant data after encryption.

D. Read and Write operation

As shown in Figure 5. All encryption components are included in memory controller. In order to improve the performance of encryption and decryption, the access of shift key and

shift operation are all performed in logic firmware. When data is read from or written into the memory, the key address and the data address are decoded at the same time. Detailed Initialization, read, write and shutdown procedures are described as follows,

- **Initialization** During the initialization when system is power-on, shift keys of all regions and global FN keys are generated from randomizer and stored in volatile memory. The redundant domain walls are writen with random data. And all the regions are locked with shift keys.
- **Read operation** Step1. The region shift back using shift key and then read the cipher text out. Step2. Input the cipher data to FN and get the plain data. Step3. Shift forward racetracks using shift key.
- **Write operation** Step1. The region shift back using shift key. At the same time, input plain data to FN and get cipher data. Step2. Write the cipher text into the region. Step3. Shift forward racetracks using shift key.
- **Normal shutdown** When the system is powered off normally, user has two options:

  Option1. If user wants to keep data in RM for later use, the stored shift keys and FN keys should be encrypted using AES algorithm and stored in RM too. But note that this encryption is not the responsibility of our work.

  Option2. If the user do not want to keep data in NVM, no extra operations are needed. All keys in volatile memory disappear after power-off.

E. Security Proof

In this section, we will prove the equivalent security strength of our design to that of AES-128. In brief words, we will prove the following three statements: (1) our design is immune to active attack (2) our design is immune to statistical attack (3) our design achieve equivalent security strength of AES-128 under computational attack.

As mentioned above, the attacker can scan the memory content after power-off, in which he has inserted "flag data" to perform the active attack. Note that all the shift-keys in the volatile memory has disappeared. The attacker cannot infer the shift states of the regions containing users' data. In other words, the shift-key of every region are "isolated" in our design because of the independence of process to generate shift-key. In addition, the attacker cannot break the ciphertext by matching. For example, he can compare the ciphertext of his inserted data and user's target data. If the ciphertexts are the same, we can conclude they are the same data. This is infeasible because even for the same data in different addresses, they are encrypted as different ciphertexts.

Now the attacker has to focus on the content of the target region. We remark that the attacker cannot break our design by statistical attack. Since our plaintext and shift states are both random and independent, every bit in the ciphertext is random and independent. For simplicity, let us focus on the first two bits $C_{1,i}, C_{2,i}$ in the i-th column of the region. We denote the original plaintext as $\{P_{ij}\}$. In the ciphertext, there are four combinations of $C_{1,i}, C_{2,i}$: $(C_{1,i}, C_{2,i}) = (P_{1,i}, P_{2,i}), (P_{1,i-1}, P_{2,i-1}), (P_{1,i-1}, P_{2,1}), (P_{1,1}, P_{2,i-1})$. Note that when $i = 1$, $P_{j,i-1}$ denotes the waste bit which is random. According to our assumption, $P_{i,j}$ and $P_{i',j'}$ are random and independent. Thus, for all the four combinations, $C_{1,i}, C_{2,i}$ are random and independent. It is quite similar to prove that this property holds for any two bits in the ciphertext.

As a consequence, the attacker has to use computational attack. To restore one region to the states before the encryption, the attacker should at least make 2^k trials, where k represents the number of stripes in the region. This is because that without enough query pairs, the attacker cannot distinguish the ciphertext from the truly random number when the round of Feistel network is larger than 4[6]. To obtain the query pairs, the attacker has to try to restore the shift states. Otherwise, using brute force to break the Feistel Network will cost more computation (i.e. 2^{r*k} trials, r stands for the round of Feistel network). In conclusion, 2^k trials are needed to break our design, which means the security strength for our design is 2^k . If we set the number of stripes in one region more than 128, our design can achieve equivalent security strength as that of AES-128.

IV. Experimental results

In this section, the experimental setup is first introduced. Then, we compare our design with other approaches. At last, we provide a detailed sensitivity analysis with various configurations, including region size, key length. Under different configuration, the storage cost, energy cost and performance of our encryption mechanism are discussed.

A. Experiment setup

We evaluate the PTL encryption mechanism with a full system cycle accurate simulator gem5[3]. With Gem5, many new style memory can be simulated using its parameters. We simulate racetrack on-chip memory by modifying the read, write and shift latency and power consumption parameters[13]. The detailed configurations of evaluation system is described in Table 1.

For workload, we select 13 workloads from Parsec3 benchmarks[2]. The benchmark programm domain include financial analysis, computer vision, physical modeling, future media, content-based search, etc., which representative main aspect of computer application at present.

We make the evaluation using a 128M RM on-chip memory as L3 cache. It is composed of basic region, which has 128 racetracks and each racetrack has 64 bits. The key-width is 1 and key length is 128bit. We compared results of four experiments with different encryption designs, in respect of performance, energy, storage and area overhead. The baseline is a system without data encryption (labeled as None Enc). The second system encrypts data with AES-128 through software (labeled as AES-Enc)[4]. The third system employs PAD-XOR based encryption scheme in Zhang's work (labeled as

TABLE I
THE DETAILED CONFIGURATIONS OF EVALUATION SYSTEM.

Unit	Configurations
CPU	4 single Alpha cores, 2GHz, 1-way issue
L1	split I/D, 32KB/32KB, 2-way, 64B,LRU, private, R/W: 1/1-cycle, 0.074/0.074nJ, 23.4mW
L2	1MB shared by 2 cores, 4-way, 64B, LRU, R/W lat.: 7/7-cycle, R/W E: 0.407/0.386-nJ, 681.5mW
L3	16-way, 64B, shared, LRU, RM,128MB, R/W/S:24/24/4-cycle, 0.956/0.952/1.331-nJ, 948.4mW
Mem.	Dual Channel DDR3, 1600MHz, 100-cycle, 38.10nJ, 12.8GB/s.

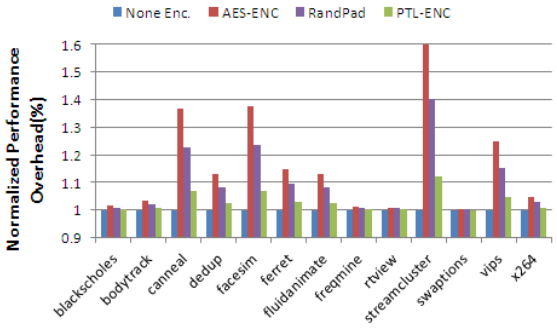

Fig. 7. Normalized Performance Overhead.

RandPad)[14]. The forth system employs our PTL based encryption scheme, together with a 4-stage FN. Note that we do not include i-NVMM approach because it cannot achieve a run-time protection to all data in memory.

B. Comparison of Experimental Results

1) Performance Evaluation: We compare the performance overhead caused by different encryption scheme in Figure 7. All results are normalized to baseline without using any encryption techniques. Obviously, all the encryption scheme have impact on latency of read and write.

For the case of AES, many benchmarks has obvious performance overhead. Especially the ones which have intensive memory access such as *canneal, facesim, streamcluster* and *vips* degrade the performance heavily, the highest one reach up to 64%. The average performance overhead is 16.7%. For the case of RandPad, the situation is better and the highest overhead is 40%. The average is 10.4%. For the case of PTL, it is obvious that it has little performance overhead that no one exceeds 12%. The average overhead is 3.1%.

The PTL has evident better performance than AES and RandPad because PTL complete the encryption and decryption operation in just several FN transformation and shift cycles, other than calculating cipher data using large amount of cycles(eg. AES use 32 cycles[4, 5]).

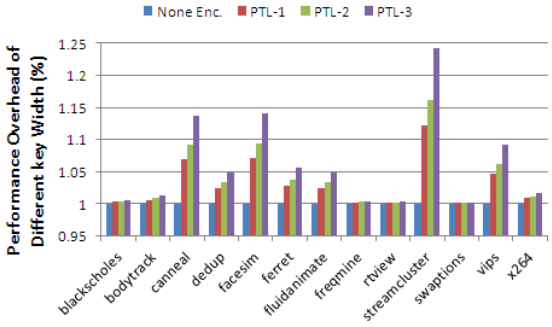

Fig. 8. Normalized Performance Overhead of Different PTL key-width.

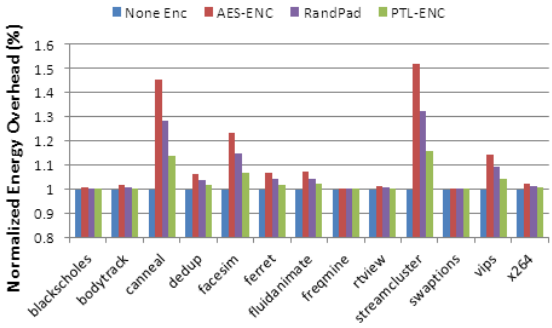

Fig. 9. Normalized Energy Overhead.

2) Performance Evaluation of Key-width: AS we mention in section III-A, the region can use longer shift key to strengthen their security. In Figure 8, PTL-1, PTL-2, PTL-3 mean that key-width is 1,2,3 respectively. And so their key length is 128, 256, 384 respectively.

It obvious that, the longer the key is, the more the programm will degrade the performance. It is easy to understand that the longer the shift keys each racetrack has, the more racetrack length it need to shift during encryption and decryption operation. For some programm which has intensive memory access, the performance degrade is more obvious. On average, the performance overhead are 3.1%, 4.1% and 6.2% for PTL-1, PTL-2 and PTL-3 respectively, which is moderate and acceptable to users.

What should be pointed out is, this evaluation is based on the hypothesis that multiple key bits in one racetrack is shifted one bye one in every single cycle. As is mentioned before, if shift currents of different amplitude are applied to each racetrack according to the linear current-velocity relationship of shift operation, all shifts can be done in one cycle[10]. If this mechanism is adopted, the performance of PTL has further space to enhance.

3) Energy Evaluation: We compare energy overhead caused by encryption in Figure 9. All results are normalized to baseline without using any encryption techniques. It is obvious that AES scheme has most heavy energy overhead, RandPad are moderate and PTL has the least energy overhead.

AES has the most heavy energy cost, several memory intensive benchmarks such as canneal and streamcluster are most

TABLE II
THE STORAGE AND AREA COST EVALUATION FOR 128MB CACHE.

Region Size	Key Num.	Key width	Key length	Key Storage	Storage Cost(%)	Redun. Area	Area Cost(%)
1KB	128K	1	128	2MB	1.56	2MB	1.6
2KB	64K	1	256	2MB	1.56	2MB	1.6
4KB	32K	1	512	2MB	1.56	2MB	1.6
1KB	128K	2	256	4MB	3.12	6MB	4.7
2KB	64K	2	512	4MB	3.12	6MB	4.7
4KB	32K	2	1024	4MB	3.12	6MB	4.7
1KB	128K	3	384	6MB	4.68	14MB	10.9
2KB	64K	3	768	6MB	4.68	14MB	10.9
4KB	32K	3	1536	6MB	4.68	14MB	10.9

TABLE III
THE STORAGE AND AREA COST EVALUATION FOR 4GB MAIN MEMORY.

Region Size	Key Num.	Key width	Key length	Key Storage	Storage Cost(%)	Redun. Area	Area Cost(%)
1KB	4M	1	128	64MB	1.56	64MB	1.6
2KB	2M	1	256	64MB	1.56	64MB	1.6
4KB	1M	1	512	64MB	1.56	64MB	1.6
1KB	4M	2	256	128MB	3.12	192MB	4.7
2KB	2M	2	512	128MB	3.12	192MB	4.7
4KB	1M	2	1024	128MB	3.12	192MB	4.7
1KB	4M	3	384	192MB	4.68	448MB	10.9
2KB	2M	3	768	192MB	4.68	448MB	10.9
4KB	1M	3	1536	192MB	4.68	448MB	10.9

obvious ones. The highest is around 50%. On average, the energy cost overhead of AES is 13%. For the case of RandPad, the situation is better. On average, the energy cost overhead of RandPad is 7.8%. PTL is the best one. The average energy cost is 3.7%.

4) Storage Cost Evaluation: RM can be used in each hierarchy of storage structure. Though we use RM based cache as a case study, our PTL encryption method can be applied to other memory architectures such as main memory or even storage. So, we evaluate the key's storage cost of 128MB cache and 4GB RM main memory respectively. We compose different regions with different number of racetracks and each racetrack has 64 bit. As all the regions in cache or memory has the same FN 4-stage keys, we neglect the storage cost of these 4 keys. The result is like Table 2 and Table 3 show.

For both 128MB cache and 4GB main memory, we find that with the same key-width, different region size has the same volatile storage cost. With the same region size, larger key-width means longer key length, which means larger volatile storage cost. For the region has 128bit key length, which is proved in section 3 that has the same security strength with AES-128, has the storage cost of 1.56%. Which is acceptable in modern system.

5) Redundant Area Cost Evaluation: We also evaluate the area cost caused by redundant bits which accommodate the shifted bits in racetrack, for both 128MB cache and 4GB main memory respectively. The result like Table 2 and Table 3 show.

For both 128MB cache and 4GB main memory, we find that with the same key-width, different region size has the same redundant bits ratio. Under the same region size, the larger key-width the region has, the larger the area cost is. For the 128bit key, the area cost of 1.6%. Which is acceptable.

V. CONCLUSIONS

Racetrack memory is extractive because of its high density and comparable read/write speed with SRAM, and non-volatility. However, one obstacle of using racetrack as cache or memory is the security vulnerability that data retain in memory long after the power off. We propose a shift based encryption scheme named PTL, which is proved has security strength not less than AES-128. Experimental results show that PTL is effective to protect whole data online or off line, accompanied with acceptable performance, storage and energy cost, outbalance other conventional algorithms.

VI. ACKNOWLEDGEMENTS

This paper is supported by National Natural Science Foundation of China (No. 61572045). This work is also supported by National High Technology Research and Development Program of China (Grant No. 2013AA013201). The paper corresponding author is Jiwu Shu.

REFERENCES

[1] $https://en.wikipedia.org/wiki/Pin\_tumbler\_lock$.

[2] C. Bienia, S. Kumar, J. P. Singh, et al. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages TR–811–08. Princeton University, Tech. Rep., 2008.

[3] N. Binkert, B. Beckmann, G. Black, et al. The gem5 simulator. In *SIGARCH Comput. Archit*, pages 39(2):1–7, 2011.

[4] T. Kgil, L. Falk, and T. Mudge. Chiplock: support for secure microarchitectures. In *Acm Sigarch Computer Architecture News*, pages 134 – 143. ACM, 2005.

[5] J. Kong and H. Zhou. Improving privacy and lifetime of pcm-based main memory. In *In Dependable Systems and Networks (DSN)*, page 333C342. IEEE/IFIP, 2010.

[6] J. Patarin. Luby-rackoff: 7 rounds are enough for 2 n (1- ε) security. In *Advances in Cryptology-CRYPTO 2003*, pages 513–529. Springer, 2003.

[7] 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*, pages 14–23. ACM, 2009.

[8] L. H. H. S. Seong N H, Dong H W. Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *International Symposium on Computer Architecture*, pages 383–394. ACM, 2010.

[9] Y. S. Siddhartha Chhabra. i-nvmm: A secure non-volatile main memory system with icremental encryption. In *The 39th International Symposium on Computer Architecture (ISCA)*, pages 177–188. ACM, 2011.

[10] T. L. Ssp P, Hayashi M. Magnetic domain-wall racetrack memory. In *Science*, pages 320(5873):190–192, 2008.

[11] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Sai, M. Jiang, Y. Yan, and H. Yang. A 3us wakeup time nonvolatile processor based on ferroelectric flip-flops. In *European Solid-State Circuits Conference (ESSCRIC)*, pages 149–152. IEEE, 2012.

[12] C. Zhang, G. Sun, W. Zhang, et al. Quantitative modeling of racetrack memory, a tradeoff among area, performance, and power. In *Proceedings of 20th Asia and South Pacific Design Automation Conference (ASP-DAC 2015)*, pages 100–105, 2015.

[13] C. Zhang, G. Sun, X. Zhang, et al. Hi-fi playback: Tolerating position errors in shift operations of racetrack memory. In *Proceedings of the 42nd ACM/IEEE International Symposium on Computer Architecture (ISCA 2015)*, pages 694–706. IEEE Press, 2015.

[14] X. Zhang, C. Zhang, G. Sun, J. Di, and T. Zhang. An efficient run-time encryption scheme for non-volatile main memory. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, page 24. IEEE Press, 2013.

[15] Y. Zhang, W. S. Zhao, D. Ravelosona, et al. Perpendicular-magnetic-anisotropy cofeb racetrack memory. In *Journal of Applied Physics*, page 093 925C093 925C5, 2012.