SECRET: Smartly EnCRypted Energy Efficient Non-Volatile Memories

Shivam Swami Joydeep Rakshit Kartik Mohanram

Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh

shs173@pitt.edu jor73@pitt.edu kartik.mohanram@gmail.com

ABSTRACT

Data persistence in emerging non-volatile memories (NVMs) poses a multitude of security vulnerabilities, motivating main memory encryption for data security. However, practical encryption algorithms demonstrate strong diffusion characteristics that increase cell flips, resulting in increased write energy/latency and reduced lifetime of NVMs. State-of-the-art security solutions have focused on reducing the encryption penalty (increased write energy/latency and reduced memory lifetime) in single-level cell (SLC) NVMs; however, the realization of low encryption penalty solutions for multi-/triple-level cell (MLC/TLC) secure NVMs remains an open area of research. This work synergistically integrates zero-based partial writes with XOR-based energy masking to realize Smartly EnCRypted Energy efficient, i.e., SECRET MLC/TLC NVMs, without compromising the security of the underlying encryption technique. Our simulations on an MLC (TLC) resistive RAM (RRAM) architecture across SPEC CPU2006 workloads demonstrate that for 6.25% (7.84%) memory overhead, SECRET reduces write energy by 80% (63%), latency by 37% (49%), and improves memory lifetime by 63% (56%) over conventional advanced encryption standard-based (AES-based) counter mode encryption.

Categories and Subject Descriptors: B.3.2 [Hardware]: Primary Memory

Keywords: Non-volatile Memories, Encryption, Security

1. INTRODUCTION

The high power consumption and poor potential for technology scaling below 22nm of DRAM [1] has spurred research in emerging resistance-class non-volatile memories (NVMs) such as phase change memory (PCM) and resistive RAM (RRAM) [2–4]. Resistance-class NVMs store data by altering the resistance of the storage material. Due to a large separation between the lowest and the highest resistance state, both PCM and RRAM NVMs support multi-level/triple-level cell (MLC/TLC) operation, which is the ability to store 2/3 logical bits per physical cell. Whereas PCM/RRAM NVMs have emerged as promising DRAM replacements, data persistence in PCM/RRAM NVMs poses many security vulnerabilities that must be addressed prior to commercialization of these advanced NVM technologies [5–8].

A broad class of memory encryption techniques have been proposed in the literature [5–11] to address security vulnerabilities in NVMs. Encryption algorithms demonstrate strong diffusion characteristics that ensure that a single bit change in the plaintext (i.e.,

unencrypted data) results in several bit changes in the ciphertext (i.e., encrypted data). Due to strong diffusion characteristics, the theoretical average cell flips per write for encrypted MLC (TLC) NVM is 0.75 (0.875) (discussed in section 2.3). This renders cell flip reduction techniques like data comparison write (DCW, i.e., classical read-modify-write) [12] and flip-n-write (FNW) [13] ineffective in practice, increasing write energy/latency and reducing lifetime of NVMs.

Initial efforts to lower encryption penalty in NVMs, e.g. [6], performed memory-side (i.e., encryption unit is placed in memory) incremental encryption, called i-NVMM, by keeping only the frequently accessed memory pages in unencrypted form in the memory and securing rest of the memory in encrypted form. However, i-NVMM suffers from security vulnerabilities due to partial memory encryption and unencrypted off-chip communications. More recently, processor-side encryption techniques like block level encryption (BLE) [7] and dual counter encryption (DEUCE) [8] have been proposed to address the security vulnerabilities of i-NVMM by placing the encryption unit in the secure processor. Additionally, in order to reduce the encryption penalty, these techniques encrypt only the modified words in a cache line and allow the unmodified words to remain in their last encrypted state (word-level read-modify-write). However, without exception, state-of-the-art security solutions address encryption penalty (increased write energy/latency and reduced lifetime) in single-level cell (SLC) NVMs; the realization of low encryption penalty solutions for MLC/TLC secure NVMs remains an open area of research.

In this work, we synergistically integrate smart encryption with XOR-based energy masking to realize Smartly EnCRypted Energy efficient, i.e., SECRET MLC/TLC NVMs, without compromising the security of the underlying encryption technique. Smart encryption integrates word-level re-encryption and zero-based partial writes in order to reduce memory writes. Word-level re-encryption modifies classical counter mode encryption—which is used at cache line granularity—by allocating separate counters for each word, allowing data encryption at granularities smaller than a cache line. Zero-based partial writes leverages the fact that a significant fraction of the plaintext written to the memory is zero for real-life workloads (e.g., SPEC CPU2006 [14]) [15]. We use a one-bit zero-flag per word to track zero-words (i.e., words with only zeros) in a cache line and maintain zero-words in their last encrypted states, saving the write overhead of re-encrypting zero-words. Following smart encryption, we perform write optimization by filtering the encrypted words (i.e., the ciphertext) through energy masks. For an overhead of one bit (energy-flag) per word, XOR-based energy masks transform high energy states in the ciphertext into low energy states, reducing the overall write energy of the cache line. Both the zero-flag and the energy-flag are stored in encrypted state in the memory in order to ensure the security of the data.

SECRET is evaluated on MLC/TLC RRAM architectures using the NVMain [16] memory simulator on memory traces from the SPEC CPU2006 benchmark suite [14]. We consider advanced encryption standard-based (AES-based) counter mode encryption as our baseline. SECRET is compared to state-of-the-art encryption

This research was supported by NSF Award CCF-1217738.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '16, June 05–09, 2016, Austin, TX, USA

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4236-0/16/06...\$15.00

DOI: <http://dx.doi.org/10.1145/2897937.2898087>

Figure 1: This figure depicts counter mode encryption, which uses a counter, line address, and secret key to generate a one-time pad (OTP). During encryption (decryption), the OTP is XORed with the plaintext (ciphertext) to generate the ciphertext (plaintext).

techniques, namely BLE and DEUCE, which perform cache line encryption at a granularity of 128 bits and 16 bits, respectively. Whereas BLE and DEUCE rely exclusively on read-modify-write to prevent full re-encryption of a cache line, SECRET integrates read-modify-write with zero-based partial writes to further reduce full re-encryption of a cache line, and employs XOR-based energy masking to lower write energy. Simulations on MLC RRAM show that BLE, DEUCE, and SECRET reduce write energy (latency) by 40% (23%), 40% (17%), and 80% (37%) over the baseline. Our lifetime evaluations show that BLE, DEUCE, and SECRET improve MLC RRAM lifetime by 35%, 36%, and 63% over the baseline. Furthermore, for TLC RRAM, BLE, DEUCE, and SECRET reduce write energy (latency) by 33% (31%), 40% (23%), and 63% (49%), respectively, over the baseline. The lifetime improvements for TLC RRAM from BLE, DEUCE, and SECRET are 18%, 24%, and 56%, respectively, over the baseline.

The rest of the paper is organized as follows. Section 2 provides background for security in NVMs and builds the motivation for SECRET. Section 3 describes the theory and architecture of SECRET. Section 4 presents results and Section 5 is a conclusion.

2. BACKGROUND AND MOTIVATION

This section covers security issues in NVMs and explains how memory encryption is used to safeguard NVMs from various attacks. Furthermore, we discuss the adverse effects of encryption on write energy, latency, and lifetime of MLC/TLC NVMs, which motivates optimization of the existing encryption techniques.

2.1 Security of NVMs

The most common security vulnerabilities in NVMs take the form of stolen DIMM attacks and bus snooping attacks [6–8]. In the stolen DIMM attack, the attacker has physical access to the NVM DIMM, enabling them to stream data from the DIMM. Data persistence of NVMs exposes data in the plaintext to attackers on power down. A similar problem has been addressed in disk storage through encryption, motivating encryption of NVMs. In the bus snooping attack, the attacker can acquire data by monitoring unsecured off-chip communications. Again, it is widely accepted that attacks can be thwarted by implementing data encryption in the secure processor¹.

Data encryption in NVMs is generally achieved by applying a block cipher (e.g., AES) to the plaintext to transform it into the ciphertext using a secret key. However, direct encryption of data introduces latency of the encryption/decryption unit on the performance critical read path of main memory. Recent work [7, 8, 10, 11] has advocated use of counter mode encryption as a zero latency scheme for memory encryption. SECRET leverages and extends the capabilities of counter mode encryption to ensure data security without the high encryption penalty (discussed in sec. 2.3).

2.2 Counter mode encryption

In counter mode encryption, a block cipher is used to encrypt a seed, with a secret key (stored on the processor), to produce a

¹We do not consider bus tampering attacks in our attack model. These attacks can be prevented by implementing memory integrity verification schemes [11, 17], which are orthogonal to SECRET.

Figure 2: (a) Average cell flips per write (b) geometric mean of write energy (normalized to unencrypted memory) for SPEC CPU2006 benchmarks for SLC/MLC/TLC RRAM. Average cell flips per write for encrypted SLC/MLC/TLC NVMs is 0.5/0.75/0.875. High average cell flips increase write energy/latency and reduce lifetime of NVMs. In comparison to unencrypted SLC/MLC/TLC NVM, energy of encrypted SLC/MLC/TLC NVM is $6\times/8.6\times/8.9\times$.

pad. This pad is bitwise XORed with the plaintext to generate the ciphertext. During decryption, the same pad is XORed with the ciphertext to obtain the plaintext. If the seed is available to the processor during a cache miss, it can overlap the pad generation with off-chip fetch, thereby hiding the encryption latency.

The spatial and temporal exclusivity of the pad is critical for the security of counter mode encryption [18], which implies that the pads should be unique for different memory blocks, and also for a particular block over time. These unique pads, namely one-time pads (OTPs), are generated from unique seeds that have two components: (i) line address of the memory block (ensures spatial exclusivity) and (ii) a counter value which is incremented on each write (ensures temporal exclusivity). Previous work [18] has shown that the counter values can be stored in the plaintext because the attacker still needs the secret key to regenerate the pad. Fig. 1 illustrates counter mode encryption. During a write, the line address and the counter value of the memory block is used to generate an OTP. The plaintext data is XORed with this OTP to produce the ciphertext, which is dispatched to main memory. During a read, the line address of the memory block being read is available to the processor, and assuming that the counter is available in a counter cache, the pad generation proceeds in parallel to the data fetch.

Following general practice in memory security research [5–8, 17] we assume a secure on-chip domain (e.g., processor, registers, cache, etc) in which we implement SECRET, and consider all off-chip resources (e.g., DIMM) and communications as unsecure.

2.3 Encryption penalty of MLC/TLC NVMs

Encryption algorithms demonstrate strong diffusion characteristics that ensure that a single bit change in the plaintext results in several bit changes in the ciphertext. Due to strong diffusion characteristics, encryption renders cell-level write reduction techniques like FNW [13] and cell-level DCW [12] ineffective in practice, increasing the average cell flips per write operation. As shown in Fig. 2(a), average cell flips per write in encrypted SLC NVM is 0.5, and it increases to 0.75 (0.875) in encrypted MLC (TLC) NVM. High average cell flips per write results in increased write energy/latency and reduced memory lifetime.

Furthermore, writing to an MLC/TLC requires configuring the cell into a target resistance range by using state-of-the-art program-and-verify (P&V) [19–21]. Typically, P&V requires several iterations of read-verify-write to bring an MLC/TLC to the desired state. The main drawback of P&V is that programming an MLC/TLC to intermediate states consumes more energy/latency in comparison to programming it to the terminal states. Table 1 shows that the energy/latency required to configure an RRAM TLC to states 3 and 4 are $10\times$ in comparison to the terminal states. Fig. 2(b) illustrates the impact of encryption on write energy of MLC/TLC RRAM NVMs. Our simulations on SPEC CPU2006 benchmarks demonstrate that on average, write energy of encrypted MLC (TLC) NVMs is $8.6\times$ ($8.9\times$) in comparison to unencrypted MLC (TLC) NVMs.

State	0	1	2	3	4	5	6	7
T _{wc} (ns)	15.2	46.8	98.3	143	150	101	52.7	12.1
Energy (pJ)	2.0	6.7	19.3	35.1	35.6	19.6	6.1	1.5

Table 1: Latency/energy for TLC RRAM program-and-verify [3, 21]

In summary, memory encryption is indispensable to protect NVMs against security vulnerabilities, but it comes at the cost of increased energy/latency and reduced lifetime. The goal of this work is to develop a secure MLC/TLC NVM system, with low encryption penalty. Without loss of generality, we consider RRAM as the representative NVM in this work; the solutions developed here are applicable to other NVM technologies.

3. SECRET

In this section, we describe SECRET, a low hardware, low memory overhead scheme which significantly reduces the cell flips, decreases the write energy/latency, and improves the lifetime of MLC/TLC NVMs without compromising its security.

3.1 Smart encryption

Without exception, counter mode encryption, as discussed in section 2.2, ensures data security during the write-back of a cache line to main memory by re-encrypting all the words in a cache line with a new OTP. However, this introduces a high encryption penalty of increased cell flips leading to increased write energy/latency and reduced lifetime of NVM. We propose smart encryption to reduce the encryption penalty of counter mode encryption as follows:

- Whereas classical counter mode encryption uses a single counter for an entire cache line, we allocate separate counters for each word, allowing word-level re-encryption at granularities smaller than a cache line.
- We propose zero-based partial encryption, where zero-words (i.e., words with only zeros) are maintained in their last encrypted states, saving the write overhead of re-encrypting zero-words.

During the write-back operation of a cache line, the majority of the words remain unmodified [7, 8]. Smart encryption preserves these words in their previous encrypted states, without compromising the security of the data. In this work, we refer to this as word-level data comparison write (DCW). We perform word-level DCW to prevent undesired cell flips by blocking the re-encryption of the unmodified words. To perform word-level DCW, we allocate a separate local counter to each word in a cache line, which is updated only when that particular word is modified during a write-back. We also maintain a global counter for the entire cache line. During a write-back, each modified word is re-encrypted with an OTP, which is generated using a counter value obtained from the concatenation of the global counter and the updated local counter for that word, whereas the unmodified words are left unchanged.

The local counter (LC) associated with any word overflows when the corresponding word observes higher number of writes than what the local counter can handle (LC_{max}). For e.g., a 2-bit local counter can handle at most ($2^2=$) 4 writes of the corresponding word before it overflows. Whenever a local counter in a cache line overflows, it compromises security because of the pad-reuse. To avert this situation and preserve the security in the event of a local counter overflow, smart encryption takes the following measure: all the local counters are reset to 0, the global counter is incremented by 1, and the entire cache line is re-encrypted using the new global and local counters. Thus, smart encryption prevents the re-encryption of unchanged non-zero-words without compromising the security of the data in memory.

To further reduce the encryption penalty of increased cell flips, we leverage the fact that a significant fraction ($\approx 60\%$ of the bytes) of the plaintext written to the memory is zero [15], and prevent

Figure 3: Smart encryption, which prevents the re-encryption of unmodified words (word 2) and zero-words (word 3). Each word is allocated a local counter that facilitates smart encryption, and is updated only when the word is modified. The zero-flags track zero-words in a cache line; zero-flag = 1 for word 3 in the figure.

the re-encryption of this zero data. We provision a single flag-bit for each word in a cache line, which records the status (i.e., zero/non-zero) of the word. We set the zero-flag of a word to 1 whenever a zero is written to it and leave the word in its previous encrypted state in main memory during write-back. Note that zero-flags are written to the memory in encrypted state, which prevents an attacker from learning if the word written was zero (discussed in depth in Sec. 3.2). Since a majority of data during a write-back is zero, preventing zero-word re-encryption further assists in reducing cell flips due to encryption.

Fig. 3 illustrates smart encryption. Without loss of generality, we assume a 24-bit cache line, with a word size of 8 bits and a local counter size of 2 bits. The first two lines show the previous plaintext, and the corresponding ciphertext that was written-back to main memory. The initial states of the local counters for words 1–3 (left to right) are 1, 3, and 2 respectively. The zero-flags of all the words are set to 0 initially. When the new data (plaintext) arrives, we observe that word 2 has the same data, whereas word 3 has all zeros. Hence, the words are left in their respective previous encrypted state (unchanged ciphertext) in the main memory. The local counter for word 2 is unchanged; however, the local counter of word 3 (zero-word) is updated (refer Sec. 3.3) and the zero-flag for word 3 is set to 1. Only word 1 is modified to a new non-zero value; hence its local counter is updated, and the new ciphertext corresponding to the updated counter is written to main memory. Thus, smart encryption successfully prevents re-encryption of unmodified/zero-words, thereby reducing cell flips and resulting in lower write energy/latency as well as improved lifetime.

3.2 Energy masks

In order to ensure the security of the ciphertext, encryption algorithms introduce a high degree of confusion in the ciphertext [9]. Confusion refers to making the relationship between the secret key and the ciphertext as complex and as involved as possible. Thus, instead of long runs of zeros or ones (which are often seen in the plaintext), a large number of 01/10 pairs is seen in the ciphertext. Since programming an MLC into 01/10 states consumes $3-4\times$ more energy in comparison to 00/11 states [3, 21], writing the ciphertext to memory is highly energy intensive in practice.

Whereas smart encryption is effective in reducing cell flips and therefore the energy of the unchanged and the zero-words, words which change are re-encrypted and written to the memory. To reduce the write energy of these re-encrypted words, SECRET incorporates post-encryption write optimization by filtering the words through energy masks. For every n -bit word (ciphertext), SECRET uses an n -bit full energy mask with $n/2$ pairs of 01 (high energy state), which are XORed with the ciphertext to transform the high

Figure 4: Transition from high to low and low to high energy states using XOR operation. Energy values are from [3, 21].

Figure 5: **Energy masking module to select the lowest energy mask for a word.** The mask which gives the minimum number of high energy states is selected. Note that for parallel operation, a separate energy masking module is present for each word.

energy states in the ciphertext into low energy states (i.e., states 00 or 11). Fig. 4 depicts the XOR-based state transition of high energy states to low energy states and vice versa. Note that either state 01 or 10 can be used in the energy mask; we use 01 for our analysis. Additionally, SECRET uses an n -bit half energy mask with the upper $n/2$ bits as 00 and the lower $n/2$ bits as $n/4$ pairs of 01. The half energy mask covers the cases where high energy states are concentrated in the lower half of the word and not uniformly distributed.

Each word (in the ciphertext form) in a cache line is XORed in parallel with a full and a half energy mask. Thereafter, the write energy of the full-mask word, half-mask word, and unmasked word are compared to identify the best mask for each word, as shown in Fig. 5. Continuing our example from Fig. 3, we illustrate the concept of energy masking in Fig. 6. Word 1 is already in a low energy state, hence it is left unmodified. Word 2 uses a full mask and word 3 uses a half mask to reduce the ciphertext into a low energy ciphertext (LEC), which is finally written to the NVM. The energy mask and zero-word are encoded in a 2-bit flag for each word as follows: 00 – no flag, 01 – zero word, 10 – half energy mask, and 11 – full energy mask. Note that if the zero-flag is set for a word, then the energy mask information for that word is not required for decryption. The combined memory overhead for smart encryption and energy masking is 6.25% per cache line (refer Sec. 3.4).

Figure 6: **This figure depicts the energy masks for 3 different words from Fig. 3.** Each word is XORed with an energy mask such that the overall write energy of the word is minimized. Note that if the zero-flag for a word is set, the energy mask for that word is not recorded in the flag-bits. The low energy ciphertext (LEC) obtained from application of the energy mask gets written on the NVM.

3.3 Flag-bit encryption

This work considers a 512-bit cache line consisting of eight 64-bit words. Thus, size of our meta-data (i.e., zero-flags and energy-flags) is 16 bits (2 bits/word). For ensuring security of the data, the meta-data is also encrypted and stored along with the cache line in the memory. Since 16-bit encryption is susceptible to attacks, we extend the size of the meta-data to 66 bits by including 5 error-correcting pointers (ECP) [22], each of 10 bits (for a 512-bit cache line). An ECP uses a pointer to point to a failed cell and also stores the correct value for that cell. Note that the use of ECPs for endurance enhancement of NVMs is common practice [21, 22]. Hence, SECRET ensures the security of the meta-data for no overhead beyond the flag-bits, by incorporating already provisioned ECP bits along with the meta-data to form a secure word.

Figure 7: **This figure depicts the architecture for write (yellow) and read (green) data-paths of SECRET.** In the write path, the zero detection module and implements the zero-based partial write, whereas the energy masking module implements energy masking. In the read path, the energy unmasking module generates the true ciphertext from the low energy ciphertext, and the zero-flag-bits determine the zero-words during decryption.

The meta-data encryption is achieved by XORing the meta-data with an OTP generated from the line address and the concatenation of global and all the local counters. To generate a unique OTP on every write, the local counters are updated even when a zero-word is written to the memory. For example, if a cache line is written with only zero-words, then only the meta-data (zero-flag-bits) change. In this scenario, if the local counters are not updated, the OTP generated for the meta-data will be same as the OTP for the previous write-back (since other parameters like line address and key remain unchanged). This leads to a pad-reuse problem, which compromises the security of our scheme. Hence, the local counters of modified words (both zero/non-zero) are updated, but the zero-words are left in the previous encrypted state in the memory. This ensures the prevention of cell flips due to zero-words without compromising the security of SECRET due to pad-reuse.

3.4 SECRET: Architectural design

SECRET requires modifications to the write and read data-paths to realize smart encryption and energy masking. In this section, we describe each in detail, and finally evaluate the hardware overhead of a practical implementation.

Write operation: Fig. 7 depicts the modified write data-path circuitry for SECRET. Writes result in (i) smart encryption followed by (ii) energy masking. Smart encryption performs a read-modify-write scheme at the word level, identifying the unmodified words, which are subsequently not written back to the NVM module (main memory). The modified words, referred to as the plaintext in Fig. 7, are then examined to identify the zero-words, which are also not written back to the NVM module. This is achieved by the zero detection module, which eliminates zero-word writes. The non-zero-words are then encrypted by XORing with an OTP. Parallel AES units generate all the possible combinations of one-time pads (OTPs) corresponding to all possible states of the local counter ($2^2 = 4$ in this example), to avoid the latency of serial generation of OTPs. The line address (LA) and the sum of the global counter (GC) and individual local counters (LC) are provided as inputs to the AES units, which generate the corresponding OTPs. The appropriate OTP for a word is selected from the combinations based on the updated local counter bits of the modified word. The ciphertext is then passed through the energy masking module (EMM). The EMM transforms the ciphertext to a low energy ciphertext that is written back to the NVM module.

Read operation: The modified read data-path to implement SECRET is also illustrated in Fig. 7. During a read, SECRET reverses the transformations that it applies to the data on a write operation. After the data is fetched from the NVM module, the meta-data (zero-flags, energy-flags, ECP) is first decrypted (not shown in the figure). The data, which is in the low energy ciphertext form, is transformed to the true ciphertext by passing it through the energy unmasking module (EUM). The EUM generates the true ciphertext by XORing the low energy ciphertext with the appropriate energy mask (no, half, full), which is selected on the basis of the energy-flag for that word. The true ciphertext is then decrypted by XORing it with the OTP corresponding to its local counter. However, if the zero-flag-bit of a word is set, a zero-word is forwarded as the plaintext to the cache.

Hardware overhead: SECRET has memory overhead associated with storage of the (i) local counters, (ii) zero-flag-bits, and (iii) energy-flags. For a 512-bit cache line with 64-bit words, each word has a 2-bit local counter and 2 bits of combined zero-flag and energy-flag, i.e., an overhead of 6.25%. SECRET requires 4 additional AES units over conventional AES-based counter mode encryption. A highly optimized and energy-efficient hardware implementation of AES for the 22nm node is reported in [23]. The AES unit overhead is 8360 gates with an area of $\approx 0.02\text{mm}^2$. The standard Intel i7 (22nm) die size is 160mm^2 [24], therefore, the AES unit area overhead (0.0125%) of SECRET is negligible in practice.

4. EVALUATION AND RESULTS

We perform trace-based simulations on an MLC/TLC RRAM architecture using NVMain [16] on both integer and floating-point workloads from the SPEC CPU2006 [14] benchmark suite. NVMain is a cycle accurate main memory simulator designed to simulate emerging non-volatile memories at the architectural level. We configure NVMain with 8GB main memory, organized as a single channel, with 1 rank, and eight x8 devices/rank. The memory controller performs first-ready first-come-first-serve scheduling, with open page policy. We configure NVMain with energy/latency parameters provided in [3, 21]. The authors in [10, 11] have shown that by using an on-chip counter cache, OTP generation can be overlapped with the reading of ciphertext from memory, thus, eliminating the encryption latency from the read path. Following past research [7, 8], we assume a 100% counter cache hit rate for all the schemes evaluated in this work. Further, we perform memory lifetime evaluation using an in-house simulator that operates at the page level with a page size of 4KB. Along [21], we assume perfect wear leveling and a mean cell lifetime of 10^8 writes until failure.

State-of-the-art in NVM security: BLE [7] and DEUCE [8] are state-of-the-art counter mode encryption techniques that perform word-level DCW to reduce the number of re-encrypted words in a cache line. BLE splits a cache line into four 128-bit words and assigns a 2-bit local counter to each word. When a local counter reaches its terminal count, the entire cache line is re-encrypted. In contrast, DEUCE splits a cache line into 32 16-bit words and maintains a leading and a trailing counter for the entire cache line. After a pre-determined number of writes (epoch), the entire cache line is re-encrypted. While the leading counter is incremented on every write, the trailing counter is fixed and incremented to the leading counter's value after every 32 writes (epoch interval). Due to its design, DEUCE requires that the words that are modified once in a given epoch interval be re-encrypted on every write during that epoch. In contrast, SECRET only encrypts non-zero modifies words, and incorporates XOR-based energy masking to reduce memory writes of encrypted data. We compare write energy, la-

NVM	Encryption technique	Energy reduction	Latency reduction	Lifetime improvement	Memory overhead
MLC RRAM	BLE [7]	40%	23%	35%	1.56%
	DEUCE [8]	40%	17%	36%	6.25%
	SECRET	80%	37%	63%	6.25%
TLC RRAM	BLE [7]	33%	31%	18%	1.56%
	DEUCE [8]	40%	23%	24%	6.25%
	SECRET	63%	49%	56%	7.84%

Table 2: Summary of energy, latency, lifetime improvements, and memory overhead of BLE, DEUCE, and SECRET over AES-based counter-mode encryption.

tency, and memory lifetime of SECRET with BLE and DEUCE. We consider AES-based counter mode encryption, which re-encrypts the entire cache line on every write, as our baseline.

Summary: Table 2 summarizes the energy, latency, and lifetime results for BLE, DEUCE, and SECRET, normalized to the baseline. Whereas BLE and DEUCE rely exclusively on read-modify-write to reduce memory writes, SECRET uses zero-based partial writes along with word-level DCW to reduce memory writes, and energy masks to lower the energy overhead of write operations.

4.1 MLC RRAM NVM

We first present our results for energy, latency, and lifetime, for an MLC RRAM architecture. Note that an MLC stores 2 logical bits per cell; a 64-bit word is stored in 32 physical MLCs.

Energy and latency: As shown in the Fig. 8, BLE, DEUCE, and SECRET reduce write energy by 27%, 38%, and 82%, respectively, over the baseline. While energy reductions for BLE (DEUCE) result from 128-bit (16-bit) word-level DCW only, energy reductions for SECRET result from 3 levels of optimization – 64-bit word-level DCW, zero-based partial writes, and energy masks. Latency reductions for BLE, DEUCE, and SECRET over the baseline are 23%, 17%, and 37%, respectively. It is interesting to note that for benchmarks such as perlbench and bzip2, the latency reductions for SECRET are lower than DEUCE. This occurs when a particular word in a cache line changes on every write, resulting in a rollover of its local counter. A counter rollover precipitates the re-encryption of the entire cache line, increasing memory latency.

Figure 8: (a) Write energy and (b) latency of BLE, DEUCE, and SECRET (normalized to baseline) evaluated using SPEC CPU2006 benchmarks [14] on NVMain [16]. For MLC RRAM, BLE, DEUCE, and SECRET reduce write energy (latency) by 40% (23%), 40% (17%), and 80% (37%), respectively, over baseline AES-based counter mode encryption.

Memory lifetime: Figure 9 reports the improvements in memory lifetime of MLC RRAM NVM using BLE, DEUCE, and SECRET, normalized to the baseline. As seen from the figure, there is no improvement in lifetime from BLE/DEUCE for the milc benchmark because the milc benchmark consists of a large number of zeros in the plaintext. Since SECRET eliminates zero encryption in the plaintext, only SECRET increases lifetime for the milc bench-

Figure 9: **Memory lifetime (normalized to AES-based counter mode encryption) for MLC RRAM using BLE, DEUCE, and SECRET encryption architectures.** Lifetime evaluated using an in-house simulator, using average cell lifetime of 10^8 writes until failure [21].

mark. On average, BLE, DEUCE, and SECRET improve memory lifetime by 35%, 36%, and 63%, respectively. SECRET improves memory lifetime by reducing the number of programmed cells, thereby reducing the wear rate of memory.

4.2 TLC RRAM NVM

We also evaluate energy, latency, and lifetime of SECRET for TLC RRAM NVM. We partition a 512-bit cache line into nine 51-bit words and one 53-bit word for smart encryption and energy masking. A 51-bit (53-bit) word is stored in 17 (18) TLCs (3 logical bits per physical TLC). Note that although we use a 51/53-bit word for implementing SECRET, we do not modify the instruction set architecture (ISA) of the underlying system. Similar to MLC NVM, we used two energy masks (a full mask and a half mask) with repeated state 3 (011), since it is one of the highest energy/latency consuming states in a TLC (refer Table 1).

Energy and latency: Fig. 10 compares the write energy and latency of TLC RRAM NVM using BLE, DEUCE, and SECRET. SECRET reduces energy (latency) by 63% (49%) over the baseline. BLE and DEUCE reduce write energy (latency) by 40% (23%) and 40% (17%), respectively, over the baseline.

Figure 10: (a) Write energy and (b) latency of BLE, DEUCE, and SECRET (normalized to AES-based counter mode encryption) evaluated on SPEC CPU2006 benchmarks [14] using NVMain [16]. For TLC RRAM, BLE, DEUCE, and SECRET reduce energy (latency) by 33% (31%), 40% (23%), and 63% (49%), respectively.

Memory lifetime: Figure 11 reports memory lifetime of TLC RRAM using BLE, DEUCE, and SECRET, normalized to the baseline. As seen from the figure, lifetime improvements are maximum for SECRET (56%), followed by DEUCE (24%) and BLE (18%).

Figure 11: **Memory lifetime (normalized to AES-based counter mode encryption) for TLC RRAM using BLE, DEUCE, and SECRET.** Lifetime evaluated using an in-house simulator, assuming average cell lifetime of 10^8 writes until failure [21].

5. CONCLUSIONS

Memory encryption, which is required to address security vulnerabilities in NVMs, increases NVM write energy/latency and degrades NVM lifetime. This problem is exacerbated in MLC/TLC NVMs, due to the prevalence of high energy (latency) states. SECRET is the first paper to target encryption penalty reduction for secure MLC/TLC NVMs. SECRET integrates smart encryption with XOR-based energy masking to realize secure MLC/TLC NVMs with low write energy/latency and improved lifetime. Smart encryption prevents re-encryption of unmodified or zero-words during a write-back, thereby preventing unnecessary cell flips and reducing write energy. For modified non-zero-words, energy masking further optimizes the write operation by applying XOR-based masks to transform a high energy ciphertext into a low energy ciphertext. SECRET outperforms state-of-the-art NVM encryption solutions, with the lowest write energy and latency, as well as the highest lifetime.

6. REFERENCES

- [1] *International Technology Roadmap for Semiconductors*, 2011.
- [2] B. C. Lee *et al.*, “Architecting phase change memory as a scalable DRAM alternative,” in *Proc. Intl. Symposium on Computer Architecture*, 2009.
- [3] C. Xu *et al.*, “Understanding the trade-offs in multi-level cell ReRAM memory design,” in *Proc. Design Automation Conference*, 2013.
- [4] P. M. Palangappa and K. Mohanram, “Flip-Mirror-Rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories,” in *Proc. Great Lakes Symposium on VLSI*, 2015.
- [5] W. Enck *et al.*, “Defending against attacks on main memory persistence,” in *Annual Computer Security Applications Conference*, 2008.
- [6] S. Chhabra and Y. Solihin, “i-NVMM: a secure non-volatile main memory system with incremental encryption,” in *Intl. Symposium on Computer Architecture*, 2011.
- [7] J. Kong and H. Zhou, “Improving privacy and lifetime of PCM-based main memory,” in *Proc. Intl. Conference on Dependable Systems and Networks*, 2010.
- [8] V. Young *et al.*, “DEUCE: Write-efficient encryption for non-volatile memories,” in *Proc. Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, 2015.
- [9] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.
- [10] J. Yang *et al.*, “Improving memory encryption performance in secure processors,” *IEEE Trans. Computers*, vol. 54, no. 5, 2005.
- [11] C. Yan *et al.*, “Improving cost, performance, and security of memory encryption and authentication,” in *Proc. Intl. Symposium on Computer Architecture*, 2006.
- [12] B.-D. Yang *et al.*, “A low power phase change random access memory using a data-comparison write scheme,” in *Proc. Intl. Symposium on Circuits and Systems*, 2007.
- [13] S. Cho and H. Lee, “Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance,” in *Proc. Intl. Symposium on Microarchitecture*, 2009.
- [14] J. L. Henning, “SPEC CPU2006 benchmark descriptions,” in *ACM SIGARCH Computer Architecture News*, 2006.
- [15] M. Ekman and P. Stenström, “A robust main-memory compression scheme,” in *Intl. Symposium on Computer Architecture*, 2005.
- [16] M. Poremba and Y. Xie, “NVMain: An architectural-level main memory simulator for emerging non-volatile memories,” in *Proc. Computer Society Annual Symposium on VLSI*, 2012.
- [17] G. E. Suh *et al.*, “Efficient memory integrity verification and encryption for secure processors,” in *Proc. Intl. Symposium on Microarchitecture*, 2003.
- [18] H. Lipmaa *et al.*, “CTR-mode encryption,” in *NIST Workshop on Modes of Operation*, 2000.
- [19] F. Bedeschi *et al.*, “A multi-level cell bipolar-selected phase change memory,” in *Proc. Intl. Solid-State Circuits Conference*, 2008.
- [20] P. M. Palangappa and K. Mohanram, “CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM,” in *Proc. Intl. Symposium on High Performance Computer Architecture*, 2016.
- [21] D. Niu *et al.*, “Low power multi-level cell resistive memory design with incomplete data mapping,” in *Proc. Intl. Conference on Computer Design*, 2013.
- [22] S. Schechter *et al.*, “Use ECP, not ECC, for hard failures in resistive memories,” in *Proc. Intl. Symposium Computer Architecture*, 2010.
- [23] S. Mathew *et al.*, “340mV–1.1V, 289Gbps/W, 2090-gate nanoAES hardware accelerator with area-optimized encrypt/decrypt GF $(2^4)^2$ polynomials in 22nm tri-Gate CMOS,” *Solid-State Circuits*, vol. 50, no. 4, 2015.
- [24] <http://ark.intel.com/products>.