Security Vulnerabilities of Emerging Nonvolatile Main Memories and Countermeasures

Sachhidh Kannan, *Member, IEEE*, Naghmeh Karimi, *Member, IEEE*, Ozgur Sinanoglu, *Member, IEEE*, and Ramesh Karri, *Senior Member, IEEE*

***Abstract*—Emerging nonvolatile memory devices such as phase change memories and memristors are replacing SRAM and DRAM. However, nonvolatile main memories (NVMM) are susceptible to probing attacks even when powered down. This way, they may compromise sensitive data such as passwords and keys that reside in the NVMM. To eliminate this vulnerability, we propose sneak-path encryption (SPE), a hardware intrinsic encryption technique for memristor-based NVMMs. SPE is instruction set architecture independent and has minimal impact on performance. SPE exploits the physical parameters, such as sneak-paths in crossbar memories, to encrypt the data stored in a memristor-based NVMM. SPE is resilient to a number of attacks that may be performed on NVMMs. We use a cycle accurate simulator to evaluate the performance impact of SPE-based NVMM and compare against other security techniques. SPE can secure an NVMM with a ∼1.3% performance overhead.**

***Index Terms*—Encryption, hardware security, memory security, memristor, RRAM.**

I. Introduction

According to the International Technology Roadmap for Semiconductors, emerging nonvolatile memories, such as phase change memories (PCM) [1] and metal-oxide memristors [2] are candidates for next-generation high-performance and high-density storage due to their non-volatility, low-power consumption, and support for multilevel cells (MLC), where multiple bits can be stored in a single cell [1]. Nonvolatile main memories (NVMM) built using PCM and memristor devices are ready replacements for flash memory, and are promising replacements for SRAM cache and DRAM main memory [3].

NVMMs provide high density due to their small cell size and their MLC capabilities. NVMMs are energy efficient, tolerant to power failure, and provide "instant-on" (the ability to suspend system operation when powered down and to resume the previous state on power-up). However, using NVMMs introduces security vulnerabilities. Sensitive data written to NVMM persists even when the system is powered down and an attacker with physical access to the system can probe the NVMM and extract valuable information.

We focus on memristor-based NVMM. For the sake of simplicity, in the following sections, we use NVMM to specifically refer to memristor-based NVMM. We designed a secure NVMM (SNVMM) that can protect against an attacker with physical access to the NVMM. The design of such a SNVMM has the following five main goals.

1) Preserve the instant-on benefit of NVMM.
2) Always keep data encrypted.
3) Have minimal impact on performance and area.
4) Be instruction set architecture (ISA) independent (i.e., work with any ISA such as X86, ARM, SPARC, and MIPS).
5) Do not modify main memory and processor.

A. Contributions

We propose sneak-path encryption (SPE) to secure a NVMM. SPE is a hardware intrinsic encryption algorithm designed for NVMMs that use MLC. SPE does not modify the processor architecture and is ISA-independent. SPE is orchestrated by an SPE control unit (SPECU) that resides between the NVMM and cache. SPE exploits sneak-paths (unintended and undesirable electrical paths within a circuit) inherent in a memory and physical parameters of the memory to encrypt the data stored in the NVMM. Data encrypted on one NVMM can only be decrypted on that NVMM. Copying the encrypted data from one NVMM to another and decrypting does not yield the original data. SPE has a low latency (∼97.5% less than block ciphers, but higher than stream ciphers) and small area overhead (less than both block and stream ciphers).

This paper is organized as follows. Section II describes prior studies in memory security. Section III-A describes the threat model. Section III-B introduces our proposed SNVMM architecture. SPE technique is presented in Section IV. The integer linear programming (ILP) technique used in SPE is detailed in Section V. The developed SNVMM architecture is analyzed for resilience in Section VI. Section VII evaluates the performance and area overhead of SNVMM. Section VIII provides a brief discussion of issues relating to SPE such as encryption latency and time to solve the ILP. It also describes the process to port SPE to memory technologies other than memristors. Section IX concludes this paper and discusses future directions.

Manuscript received April 3, 2014; revised August 7, 2014; accepted October 10, 2014. Date of publication November 11, 2014; date of current version December 17, 2014. This paper was recommended by Associate Editor Y. Chen.

S. Kannan and R. Karri are with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA (e-mail: skanna01@students.poly.edu).

N. Karimi was with the Polytechnic Institute of New York University, Brooklyn, NY 11201 USA. She is now with Rutgers University, Piscataway, NJ 08854 USA.

O. Sinanoglu is with the Department of Engineering, New York University, Abu Dhabi, U.A.E.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2369741

II. Related Work

One concern in NVMM security is that an attacker can exploit the limited write endurance to run an application that damages the memory through repeated writes [4]. Qureshi *et al*. [4] proposes a randomized start-gap wear leveling algorithm, which moves a line in the physical memory to a new location before it reaches its endurance limit.

Another security vulnerability is data persisting in plaintext (PT) form in the NVMM after the system is powered off. To deal with this vulnerability, a number of main memory encryption techniques have been proposed [5]. These techniques prevent data leakage by using a cryptographic engine embedded in the processor and modifying the processor ISA. Such techniques address the security vulnerability that this paper targets. In practice, self-contained solutions where security techniques are independent of processor's platforms and ISA are preferable [6].

References [6]–[9] propose self-contained techniques to SNVMMs. References [7] and [8] secure an NVMM using stream ciphers. Stream ciphers use a pseudorandom key to encrypt/decrypt data. They provide low latency encryption but require registers to store the pseudorandom key ($\sim$6.18 mm^2 in [6]). Stream ciphers are vulnerable to correlation attacks and distinguishing attacks and they are not considered as secure as block ciphers [10]. Zhang *et al.* [9] minimizes performance impact by parallelizing a PAD-XOR-based encryption method with a memory read. However, similar to stream ciphers, this scheme requires a large area overhead ($0.9 - 84.5$ mm^2 in [9]).

Although, block ciphers may be used for NVMM encryption, they have high performance overhead that further exacerbates the processor-memory bottleneck [6]. Chhabra and Solihin [6] propose i-NVMM, a block cipher-based encryption approach for NVMM. To reduce performance overhead associated with block ciphers, i-NVMM only encrypts inert pages (memory pages that, based on previous access patterns, are not likely to be reused in the near feature). Remaining pages are encrypted before system power down. In practice, workloads that are memory-intensive and access a large subset of their memory pages have few inert pages. Therefore, memory pages for these workloads will not be encrypted until after power down, leaving a large window of opportunity (14.6 s [6]) for the attacker to steal the data after power down.

III. SNVMM

A. Threat Model

We focus on three attacks that are specific to NVMM in this paper.

1) Attack 1: The attacker has physical access to the NVMM either during system operation or after power down. He can steal the NVMM to leak sensitive data using a brute force attack or a known-PT attack [11].

2) Attack 2: The attacker has access to the NVMM and the system. He can read from and write to the NVMM and determine any secret keys used to secure the NVMM using a chosen-PT [11] or an insertion attack [11].

3) Attack 3: The attacker has access to the NVMM during power down. During power down, there is a delay between the initiation of power down and the time when all data on the NVMM is encrypted. The attacker exploits this delay and leaks data from the NVMM [12].

4) Other Attacks (Not Targeted): NVMMs have limited write endurance. An attacker may "age" the memory by writing specific data patterns to the NVMM to accelerate aging and damage the memory. Studies have explored this attack and developed techniques to defend against such attacks [4]. Denial of service attacks corrupt the stored data, or force the system into an unstable state. Data may also be corrupted by environmental effects such as heat and gamma rays. Environmental effects can be mitigated by error-correction codes and/or physical shielding [8]. NVMMs are also susceptible to permanent damage by application of force. This can only be prevented by increased physical security.

B. SNVMM High-Level View

Fig. 1 shows the modification necessary to a memory read and write operation using a write-through cache with no-write allocation [13]. The gray blocks show the modifications to the typical cache read/write policy.

1) Read Operation: Data is read from NVMM on a cache miss. When data is requested from the NVMM, first, the required data block is located in the main memory. The required data blocks is the ciphertext (CT). The data block is moved to an intermediate memristor memory in the SPECU and sneak path decryption is performed. The PT data is read out and sent to the corresponding cache block.[1]

2) Write Operation: During the write operation, with a write-through cache policy, the PT is written to the cache and then to the intermediate memory. The data block is encrypted using SPE and then moved to the NVMM.

The architecture required to enable the proposed SPE and decryption process is shown in the following section.

C. SNVMM Architecture

Fig. 2(a) shows the SNVMM architecture with a typical two-level memory architecture. The processor and cache only operate on the unencrypted (PT) data. The SPECU is inserted between the NVMM and the L2 cache providing ISA-independent memory encryption. SPECU protects the NVMM by storing data in the NVMM in encrypted form (CT).

D. SPECU

A SPECU, shown in Fig. 2(b), uses SPE to encrypt data stored in NVMM. The SPECU consists of a volatile memory (to store the key), a pseudo-random number generator (PRNG), and an 8×8 intermediate memory (memristor-based crossbar) performs encryption/decryption using SPE. SPE uses sneak-paths in the intermediate memory (described in Section IV). Sneak-paths may corrupt data during the read/write operation

[1]In [14], we perform SPE directly on the NVMM without using an intermediate memory. However, as shown in Section VII this results in a larger performance impact.

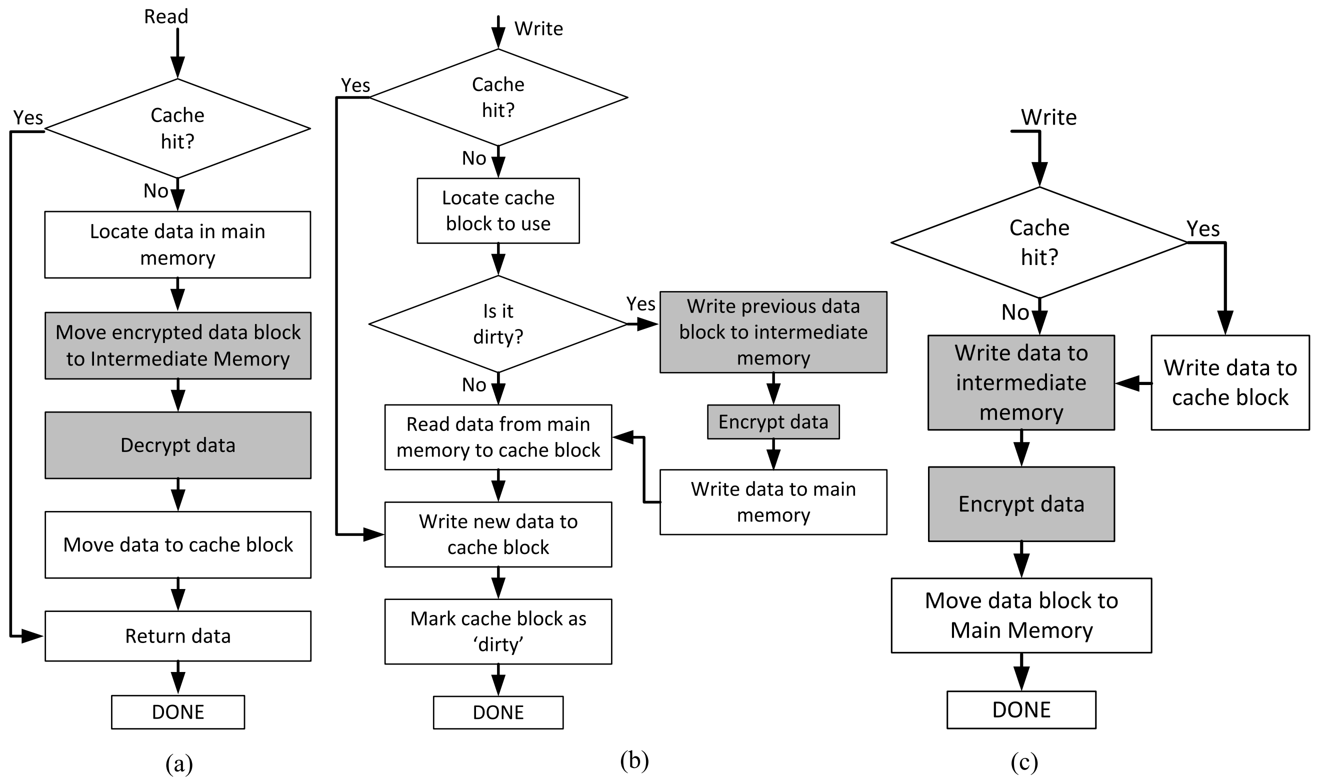

Read
Cache hit?
Yes
No
Locate data in main memory
Move encrypted data block to Intermediate Memory
Decrypt data
Move data to cache block
Return data
DONE
(a)
Write
Cache hit?
Yes
No
Locate cache block to use
Is it dirty?
Yes
No
Write previous data block to intermediate memory
Encrypt data
Write data to main memory
Read data from main memory to cache block
Write new data to cache block
Mark cache block as 'dirty'
DONE
(b)
Write
Cache hit?
Yes
No
Write data to intermediate memory
Write data to cache block
Encrypt data
Move data block to Main Memory
DONE
(c)

Fig. 1. Cache read and write policies when using SPE. (a) Cache read. (b) Write back. (c) Write through policy. White blocks represent typical cache operations, gray blocks represent additional steps performed with SPE.

of the NVMM. They are disabled when reading or writing to the NVMM. However, sneak-paths are reintroduced during the encryption and decryption operation. Hence, the NVMM is modified to introduce sneak-paths only on demand (described in Section IV-A). The SPECU works as follows.

1) Initialization: We assume that the computer has a trusted platform module (TPM) to ensure integrity of the platform [15]. During power-on, the TPM authenticates the NVMM and sends the key to the SPECU. The SPECU stores the key in volatile memory (either SRAM or DRAM). Hence, the key is lost during a power down and does not persist if the attacker steals the NVMM.

2) Encryption: During a memory write operation, the PT is first written to the intermediate memory. The PT is encrypted and then written to the NVMM. Encryption is performed by enabling sneak-paths. The PRNG generates a pseudorandom sequence (using the 88-bit key as a seed) that is mapped to specific voltage pulses and address locations (discussed in Section V). Each voltage pulse is applied to the corresponding address locations, in the presence of sneak-paths, encrypting the PT to generate the CT (details in Section IV). The CT is then written to the NVMM in the absence of sneak-paths.

3) Decryption: When the processor reads memory from the NVMM, the data needs to be decrypted. First, the CT is moved from the NVMM to the intermediate memory in the absence of sneak-paths. Decryption is performed using sneak-paths. On completing decryption, sneak-paths are disabled and the PT is sent to the processor. During decryption, the PRNG generates a pseudorandom sequence (using the 88-bit key as a seed) that is mapped to specific voltage pulses and address locations (determined in Section V). The sequence of address locations and voltage pulses are reversed and each voltage pulse is applied to the corresponding address location, decrypting the CT (details in Section IV).

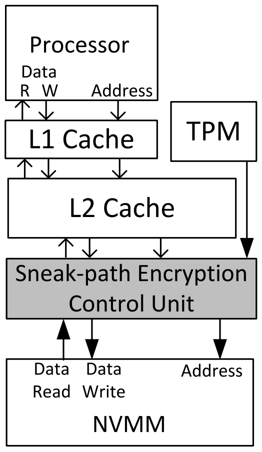
IV. SPE

We use a metal-oxide memristor-based intermediate crossbar to illustrate SPE. SPE can be adapted for use with other resistive RAM-based memories as well.

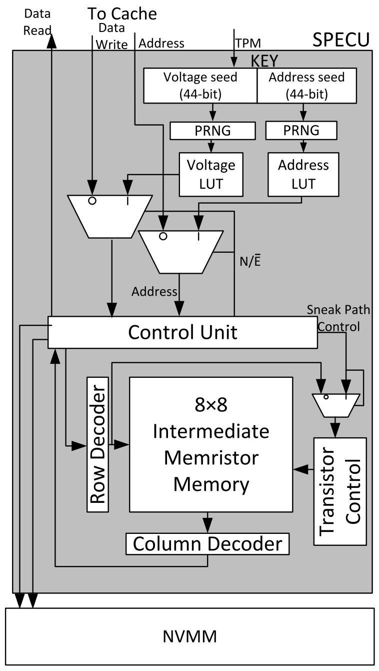
Fig. 3(a) shows the encryption/decryption for a 4×4 crossbar using a 10-bit key (the size of the key varies with that of the crossbar as shown in Section IV-D), which is the seed of the PRNG. The PRNG generates a sequence of four 10-bit numbers. The first 5-bits are mapped to a pulse width and voltage and the last 5-bits are mapped to an address in the intermediate memory. A voltage pulse is applied at each address location, and sneak-paths are enabled to encrypt the surrounding memory cells as well. The addressed cell is the point of encryption (PoE). The group of cells that experience resistance change by applying a voltage pulse at the PoE, with sneak-paths enabled, is called a polyomino.[2] To encrypt the PT, we use four voltage pulse-PoE pairs generated by the PRNG and look-up table. The set of PoEs (determined using ILP described in Section V) ensures that all memory cells are encrypted.

Fig. 3(a) shows the decryption operation of a 4×4 crossbar. The sequence in which the polyominos are decrypted is

[2]A polyomino is any plane geometric figure formed by joining one or more equal squares edge-to-edge.

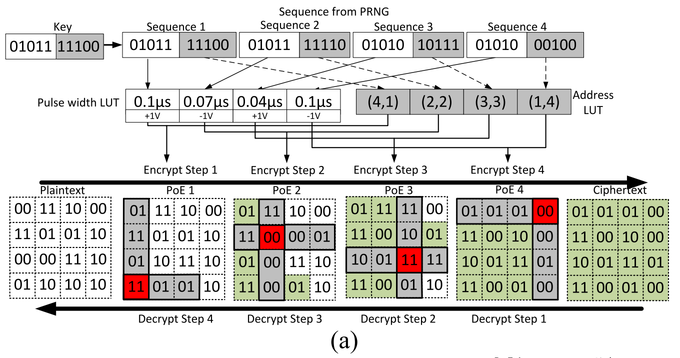


(a)

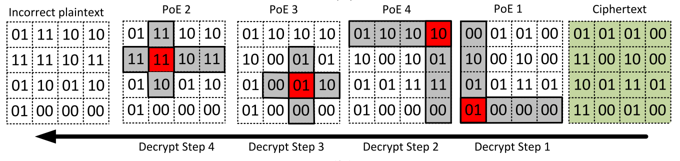


(b)

Fig. 2. (a) Proposed architecture of a SNVMM. (b) SPECU architecture for SPE.

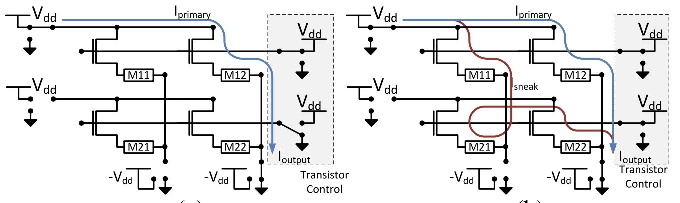


(a)



(b)

Fig. 3. (a) Encryption/decryption in a 4 × 4 crossbar. (Red: PoE, Gray: polyomino, Green: encrypted cells). (b) Attempted decryption using improper sequence of PoEs.



(a) (b)

Fig. 4. (a) 2×2 1T1M crossbar. I_{primary} flows through the addressed memory cell (M_{12}). Only transistors on the addressed row are turned on. (b) Sneak-paths in a 2 × 2 1T1M crossbar. Sneak-paths are introduced by turning on all transistors. A sneak-path current (I_{sneak}) flows through M_{11}, M_{21}, and M_{22}, corrupting the output current (I_{output}).

extremely important for proper decryption. In Fig. 3(b), we decrypt the CT using the same PoEs, but in an order different from encryption. Although the PoE locations used are the same as encryption, the shape of the polyomino will be different and depend on the data stored in the crossbar. Every time a voltage is applied at a PoE, the resistance of every memory cell within the polyomino changes. This in turn changes the shape of polyomino of the next PoE.

To study the effect of parametric variations on the encryption operation, we use a Monte Carlo analysis. We vary the wire resistance by ±5% and see that there is no change in the shape of the polyomino. Macro level changes to the device/crossbar parameters change the shape of the polyomino showing significant effect in the encryption.

A. Intermediate Memory

The intermediate memory is a metal-oxide memristor-based crossbar [16]. Each memristor cell is used as a four-level MLC (MLC-2) that stores two bits in a single cell. As shown in Fig. 4(a), there is a metal-oxide memristor in series with a transistor at the intersection of each pair of perpendicular wires (1T1M crossbar). In a 1T1M crossbar, the gates of all transistors in the same row are connected. When a memristor is addressed, all transistors on the addressed row are turned ON, allowing current to flow only through memristors in that row eliminating sneak-paths.

SPE relies on the existence of sneak-paths. As shown in Fig. 4(b), we modify the peripheral circuitry of the crossbar circuit to control sneak-path, specifically by turning on all transistors during encryption/decryption.

The memristor model used in our SPICE simulations is based on the nonlinear ionic drift model [17]. This model assumes asymmetric switching behavior and has been extensively used in the simulating memristors and memristor crossbars [17]. The nonlinear I-V characteristics of this model make it desirable for simulating memory and logic gates. The I-V relationship is given by

$$i(t) = \alpha^n \beta \ \sinh(\delta v(t)) + \chi \left[e^{\gamma v(t)} - 1\right]$$

$$\text{Window function: } f(w) = 1 - \left(\frac{2w}{D} - 1\right)^{2p} \qquad (1)$$

TABLE I
FITTING PARAMETERS AND CONSTANTS USED IN MODELING MEMRISTOR [17]

Fitting Parameters	Value
n	16
β	10
γ	4
δ	2
χ	0.01

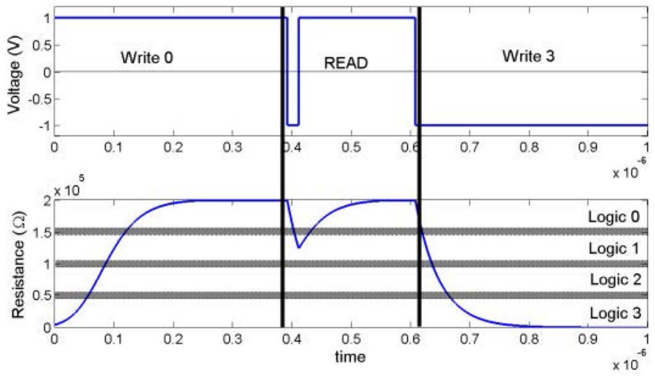


Fig. 5. Resistance distribution in a four-level (MLC-2) memristor cell. The gray regions are undefined regions. Memristors that have a resistance that lies in the undefined region may be erroneously read by the sense amplifier.

where β, γ, δ, and χ are experimental fitting parameters and n is a parameter that determines the influence of the state variable α on the current. The fitting parameters used in our model is presented in Table I.

As shown in Fig. 5, one can use intermediate resistance states of the memristor to represent multiple memory levels thereby enabling each memristor cell to store multiple bits of data.

B. Encryption

Applying a voltage pulse to an addressed memory cell, in the presence of sneak-paths, results in a voltage difference across adjacent cells. Thereby, their resistance changes. In Fig. 6, applying 1 V to the PoE shown in red, results in a voltage difference across some adjacent cells. Cells with a voltage less than the transistor threshold voltage (V_t) are not affected. The cells affected are unique to each PoE and are determined by the physical parameters of the crossbar and the data stored in each cell.

Applying a voltage at the PoE changes the resistance of the cells in the polyomino. Different PoEs with overlapping polyominos can be used to encrypt the entire memory. The user key includes the voltages applied at each PoE and the order in which they are applied.

Although SPE applies multiple pulses across each memristor, it has a negligible effect on the endurance of the memory cells since the resistance change is small compared to typical write operations [18].

C. Decryption

To decrypt the CT, an opposite pulse of equivalent magnitude must be applied at each PoE in the reverse order.

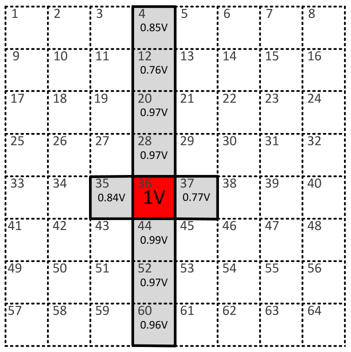


Fig. 6. Voltages across the polyomino (gray) memory cells when a 1 V pulse applied at PoE (red). Memory cells with a voltage $< V_t$ (white) are not affected.

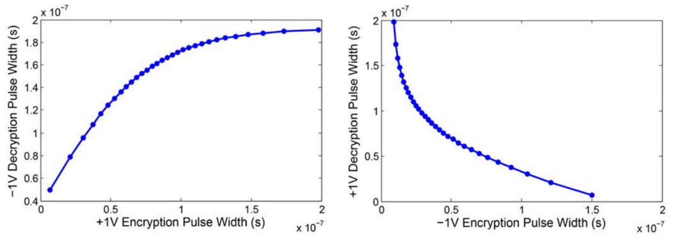


Fig. 7. Encryption pulse width and corresponding decryption pulse width assuming voltage of +1 V and −1 V.

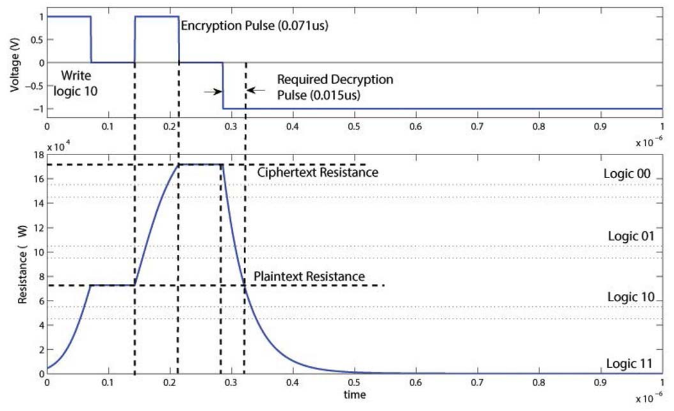


Fig. 8. Decryption of a memristor cell. Decryption pulse has a different pulse width compared to the encryption pulse due to the nonlinear memristor characteristics.

Memristors display hysteresis and a different pulse width is required during encryption and decryption. Fig. 7 shows the encryption pulse width and the corresponding pulse width required for decryption. For example, Fig. 8 shows the encryption and decryption process of a single memristor. The memristor first stores a logic 10 (PT). An encryption pulse of 1 V with a width of 0.071 μs is used to encrypt the memristor, increasing its resistance to 172 $k\Omega$ (logic 00). Due to the hysteretic nature of the memristor, the pulse width required to decrypt the memristor is of a different width. In Fig. 8, a -1V pulse that is 0.015 μs is required to properly decrypt the CT.

Decryption of memory cells that are encrypted using sneak-paths is slightly more complicated. The voltage across the

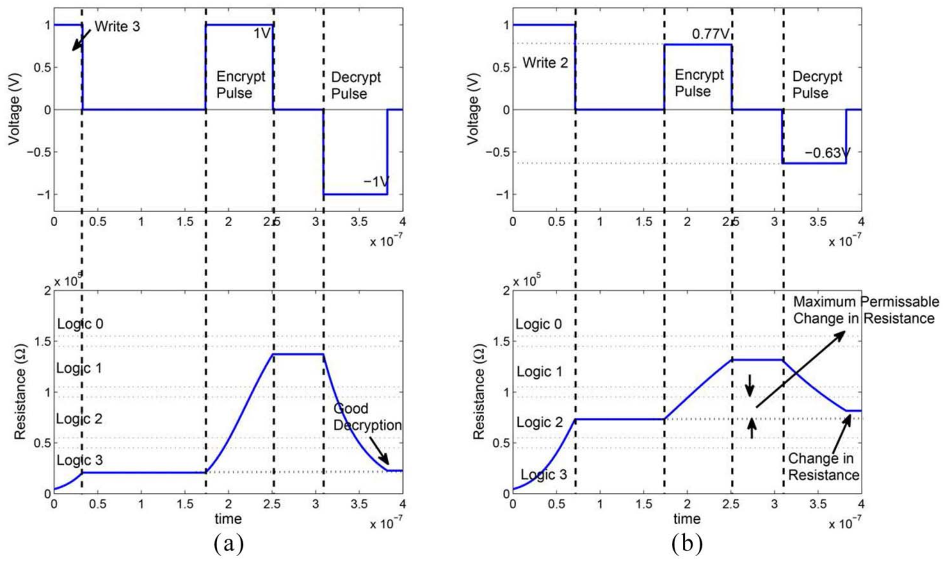


Fig. 9. (a) Encryption and decryption of addressed memory cell ($M_{i,j}$). (b) Encryption and decryption adjacent memory cell ($M_{i+1,j}$). Resistance after decryption is not necessarily same as resistance before encryption. The resistance bands for each logic level is determined based on worst case change in decrypted resistance.

cell during decryption will not necessarily be the same as the voltage applied during decryption. Let us assume that a memory cell $M_{i,j}$ at row i and column j is the PoE. The adjacent memory cell $M_{i+1,j}$ lies in the polyomino. Fig. 9 shows the encryption and decryption of both $M_{i,j}$ and $M_{i+1,j}$. As we can see in Fig. 9(a), $M_{i,j}$ has a PT of logic 3 and $M_{i+1,j}$ has a PT of logic 2. $M_{i,j}$ is encrypted using a 1 V voltage pulse. Due to sneak-paths, a resultant voltage of 0.77 V is applied across $M_{i+1,j}$ encrypting it. The pulse width of the decrypting pulse is determined based on the graphs in Fig. 7 (+1 V encryption). This pulse decrypts $M_{i,j}$ to the original PT resistance. However, $M_{i,j}$ has a voltage -0.63 V across it during decryption. This voltage is sufficient to return the resistance to the PT logic level (logic 2) but does not return to the PT resistance level. The net change in resistance due to encryption and decryption is represented by ΔR.

In order to ensure proper decryption, we assume that the initial resistance of a memristor is at the center of the resistance band of any logic level. This can be achieved by iterative read and write mechanisms such as a read-monitored write [19]. Hence, for proper decryption, the maximum allowable ΔR should be less than half the resistance band of the logic level. The worst case scenario occurs when the PoE is a low resistance state (logic 3) and all other memory cells are at a high resistance state (logic 0). Performing encryption followed by decryption in this scenario gives us a maximum ΔR of 23 $k\Omega$. Hence, the minimum resistance band for each logic level should be at least 46 $k\Omega$. This is a necessary operation in order to correctly perform decryption. The resistance band in our memristor model is 50 $k\Omega$.

D. Secret Key

The secret key of SPE is used to determine: 1) order of PoEs and 2) voltage applied at each PoE. The order of the PoEs and voltages are determined by a PRNG using the key as a seed value. The PRNG is seeded at the beginning of each encryption/decryption operation to ensure that the same sequence is generated every time. In Section VII, we show that for a 8×8 crossbar we require 16 PoEs to secure the PT stored in the intermediate crossbar. Hence, the permutation of any P_{64}^{16} cells can be represented by 44-bits. A typical memristor-based crossbar memory uses several different pulse widths to program the memory cells. We use the same pulse width generator to perform SPE. We assume that the pulse width generator is capable of producing 32 distinct pulse widths of either +1 V or -1 V. The sequence of 16 voltage/pulse width combinations can be generated by a PRNG with a 44-bit key. Hence, the minimum key size for a 8×8 crossbar is 88-bits. Each element of the generated sequence consists of 44-bits to represent the PoE location and 44-bits to represent the voltage. A secure PRNG with an increased key size [20] may be used to further improve security.

E. Analytical Model for SPE/Decryption

We address the memory cell located in mth row and nth column of each crossbar with location $i = (m-1)N + n$, where M and N are the number of rows and columns in the crossbar, respectively. Voltages are applied at P polyominos to encrypt the memory. We use an array A of binary variables, where $A_{i,j} = 1$ if and only if polyomino j affects cell i. R_i and R_i^+ represent the resistance of the cell before and after encryption, respectively

$$R_i^+ = R_i + \sum_{j=1}^{P} \left(A_{i,j}.V_{i,j}^*/I_{i,j}^* \right). \tag{2}$$

$V_{i,j}^*$ and $I_{i,j}^*$ are the voltage and current across memory cell i when a voltage is applied across PoE j. $V_{i,j}^*$ and $I_{i,j}^*$ depend on the physical parameters of the crossbar, PoE locations, and the R_i of surrounding memory cells.

Consider a crossbar model to evaluate SPE. We assume that there is no wire resistance, coupling effects, and effect from parametric variations. Assume that the voltage at each row is given by $V_1, V_2, ..., V_M$ and the voltage at each column is given by $V_1^{'}, V_2^{'}, ..., V_M^{'}$. The voltage across any memristor can be evaluated using

$$\left(\begin{array}{ccc:ccc} \sum_{a=1}^{N}\frac{1}{R_{(1,a)}} & \cdots & 0 & \frac{-1}{R_{(1,1)}} & \cdots & \frac{-1}{R_{(1,N)}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{a=1}^{N}\frac{1}{R_{(M,a)}} & \frac{-1}{R_{(M,1)}} & \cdots & \frac{-1}{R_{(M,N)}} \\ \hdashline \frac{-1}{R_{(1,1)}} & \cdots & \frac{-1}{R_{(M,1)}} & \sum_{a=1}^{M}\frac{1}{R_{(a,1)}} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{-1}{R_{(1,N)}} & \cdots & \frac{-1}{R_{(M,N)}} & 0 & \cdots & \sum_{a=1}^{M}\frac{1}{R_{(a,N)}} \end{array}\right)\left(\begin{array}{c} V_1 \\ V_2 \\ \vdots \\ V_M \\ \hdashline V_1^{'} \\ V_2^{'} \\ \vdots \\ V_N^{'} \end{array}\right) = \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right). \quad (3)$$

When a voltage $V_k - V_l^{'}$ is applied across the (k, l)th memory element, the voltage across any other memory element (m, n) is $(V_m - V_n^{'})$. Solving (3), we get

$$\left(V_m - V_n^{'}\right) = R_{(m,n)}\left[\sum_{p=1}^{n}\sum_{q=m}^{M} f((m,q),(p,n))\right]^{-1} \cdot \left(V_k - V_l^{'}\right) \quad (4)$$

where

$$f((m,q),(p,n)) = \left[R_{(m,q)} + R_{(p,n)}\right] + \sum_{a=m+1}^{p}\sum_{b=q}^{n-1}\frac{1}{f((a,q),(b,j))}.$$

V. Determining PoE Location

To determine the set of PoEs used in SPE we use ILP. We formulate our problem as an ILP and leverage ILP solvers such as the FICO optimization suite [21] to solve the problem optimally. In order to generate a set of PoE that can be used to encrypt the entire memory, there are two main criteria.

1) Ensure that every memory cell is covered by at least one polyomino.
2) Determine the minimum number of PoEs required for encryption.

Let us assume that the PoE is located in the mth row and nth column of each crossbar. The PoE address is given by $i = (m-1)N+n$, where M and N are the number of rows and columns in the crossbar, respectively. We use a matrix B of binary variables to track the PoE locations. A PoE address i is assigned to polyomino j if and only if $B_{i,j} = 1$. In addition, as mentioned in Section IV, applying a voltage at PoE i affects its neighboring memory cells. To consider this, we use a matrix A of binary variables. $A_{i,j} = 1$ if and only if polyomino j affects cell i.

As an example, consider the 8×4 crossbar shown in Fig. 10. In this figure, the red cells show two POEs. The yellow cells shown the polyomino with PoE $P1$ and the gray cells show the polyomino with PoE $P2$. For this example, the elements

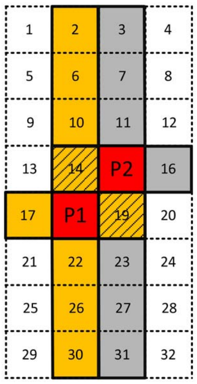


Fig. 10. Encryption of 8 × 4 crossbar. Red cells (P1, P2) are two different PoEs. Yellow cells are memory cells in polyomino P1. Gray squares are memory cells covered by polyomino P2. Numbers shown are the cell addresses (i).

of matrices A and B are as below

$$B_{i,1} = \begin{cases} 1 & \text{if } i \in 18; \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,2} = \begin{cases} 1 & \text{if } i \in 15; \\ 0 & \text{otherwise} \end{cases}$$

$$A_{i,1} = \begin{cases} 1 & \text{if } i \in 2, 6, 10, 14, 17, 18, 19, 22, 26, 30; \\ 0 & \text{otherwise} \end{cases}$$

$$A_{i,2} = \begin{cases} 1 & \text{if } i \in 3, 7, 11, 14, 15, 16, 19, 23, 27, 31; \\ 0 & \text{otherwise.} \end{cases}$$

The goal of our model is to minimize the number of PoEs (P) used to encrypt the NVMM. Hence, the objective function is given by

$$\text{Objective function} = \min(P). \quad (5)$$

The following equations provide the constraints to the ILP when determining the locations of the PoEs:

$$\forall\ 1 \le j \le P : \sum_{i=1}^{\text{MN}} B_{i,j} = 1 \quad (6)$$

$$\forall\ 1 \le i \le \text{MN} : \sum_{j=1}^{P} B_{i,j} \le 1. \quad (7)$$

Equation (6) dictates that each polyomino has one PoE. Equation (7) dictates that each memory cell can be used as a PoE at most once.

Each memory cell should be covered by at least one polyomino. However, a large number of overlapping polyominos could results in improper decryption (Section IV). Based on our memristor and crossbar models, we determine that each memory cell should be covered by at most two polyominos. This constraint is given by

$$\forall\ 1 \le i \le \text{MN} : 1 \le \sum_{j=1}^{P} A_{i,j} \le 2. \quad (8)$$

Equation (8) implies that each memory cell is affected by either one or two PoEs. In practice, it is preferred that each cell is covered by two or more polyominos to improve security (Section VI). However, this increases the

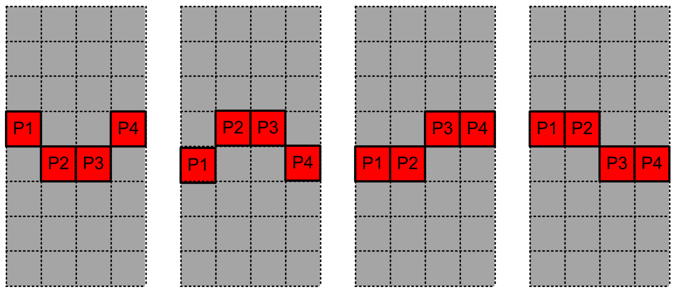


Fig. 11. Several possible PoE locations identified by the ILP on a 8 × 4 crossbar using the polyomino shown in Fig. 6. Each of these solutions cover all memory cells in the crossbar.

encryption/decryption latency. The trade-off between security and encryption/decryption latency is shown by

$$\sum_{i=1}^{\mathrm{MN}}\sum_{j=1}^{P} A_{i,j} > \mathrm{MN} + S \tag{9}$$

where S $(0 \le S \le (\mathrm{MN} - 1))$ is a tuning variable that allows us to trade-off between the number of PoE used in encryption and the number of memory cells covered by multiple polyominos. This provides a security-latency trade-off described in Section VI-A. The polyomino shown in Fig. 6 represents the smallest possible polyomino shape (determined using SPICE simulations of the crossbar). The following equation models the polyomino shown in Fig. 6:

$$A_{i,j} = B_{i+1,j} + B_{i-1,j} + \sum_{k=-4}^{4} B_{i-Nk,j} \tag{10}$$

where $i - Nk$ denotes the memory locations that reside in the same column but in ±4 rows from cell i. $i{+}1$ and $i{-}1$ represent the memory locations to the left and right of the PoE. Note that this equation should be customized for boundary cells based on the cell location. For example, in the 8 × 4 crossbar shown in Fig. 10, for a PoE at address $i = 20$, (10) should be revised as $A_{20,j} = B_{19,j} + \sum_{k=-4}^{4} B_{20-Nk,j}$

The ILP model has several nonunique solutions. Fig. 11 shows four possible PoE locations suggested by our ILP model for an 8 × 4 crossbar. Any one of these four solutions can be used to encrypt/decrypt the data stored in this crossbar. The multiplicity of solutions is an advantage for SPE, as an attacker with access to the ILP cannot exactly determine the PoE locations used during encryption (Section VI).

VI. Security Analysis

We analyze the security strength of SPE against attacks described in Section III-A. We model an 8 × 8 1T1M crossbar using HSPICE. Memristors are modeled using the TEAM model [22] in MATLAB and integrated with HSPICE. We use FICO ILP solver [21] to determine the PoE locations.

A. Randomness Tests

To study the randomness of SPE, nine sets of data were collected and analyzed using the NIST randomness test suite [23]. For all tests, 150 binary sequences (120 kB per sequence) were analyzed.

1) Key Avalanche: Avalanche effect measures the sensitivity of SPE to changes in input parameters (i.e., the key or PT). In key avalanche effect, the randomness is determined by a three-step process.

a) Given an 88-bit random key and an all-zero PT, the CT is determined.
b) With a fixed PT, the key is perturbed by flipping the ith bit, where $1 \le i \le 88$.
c) The CTs from the previous two steps are XORed to generate the data set.

2) PT Avalanche

a) Given an 88-bit random PT and an all-zero key, the CT is determined.
b) With a fixed key, the PT is perturbed by flipping the ith bit, where $1 \le i \le 128$.
c) The CTs from the previous two steps are XORed to generate the data set.

3) Hardware Avalanche: We consider another type of avalanche effect that is unique to SPE. We use an all-zero PT and an all-zero key. We then perturb the physical parameters (wire resistance, resistance range of memristor, etc.) of the crossbar beyond the parametric variations and study the effect on the CT. The physical parameters are perturbed from 5% to 10% in steps of 0.5%.

4) PT/CT Correlation: To study the correlation of PT/CT pairs, n PTs are encrypted using SPE. Given a random 88-bit key and n random PT blocks, a binary sequence is constructed by concatenating the results of applying the XOR operator on the PT and its corresponding CT.

5) Random PT/Key: To examine the randomness of CT (based on random PT and random 88-bit keys), a data set is constructed as a result of the concatenation of n CT blocks using n random PTs and a random 88-bit key.

6) Low Density PT: Two data sets are created based on low-density blocks used as PT. Each data set consists of n CT blocks. These CT blocks are formed from one all-zero PT block, n PT blocks of a single one and 127 zeroes, and 8001 PT blocks of two ones and 126 zeroes.

7) Low Density Key: Two data sets are created based on low-density blocks used as an 88-bit key. Each data set consists of n CT blocks. These CT blocks are formed from one all-zero key, 88 keys of a single one and 7 zeroes, and 3828 keys of two ones and 86 zeroes.

8) High Density PT: Two data sets are created based on high-density blocks used as PT. Each data set consists of n CT blocks computed using SPE. These CT blocks are formed from one all-one PT block, 128 PT blocks of a, and 8 128 PT blocks of two zeroes and 126 ones.

9) High Density Key: Two data sets are created based on high-density blocks used as PT. Each data set consists of n CT blocks computed using SPE. These CT blocks are formed from an all-one key, eight key blocks of a single zero and 7 ones, and 28 key blocks of two zeroes and 6 ones.

Table II shows the number of sequences that fails each NIST test [23]. With a significance level of 0.01, not more than five sequences are allowed to fail a test. SPE passes all NIST tests.

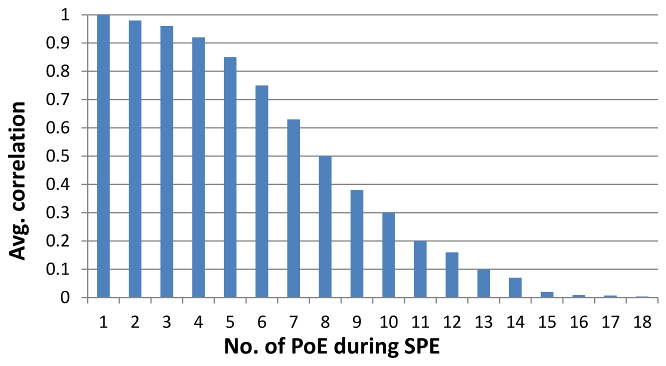


Fig. 12. Correlation of PT and CT versus the number of PoEs used in encryption increases. A correlation of 1 denotes that the PT and CT are strongly correlated and vulnerable to attacks. A correlation < 0.01 shows that there is no correlation [23].

TABLE II
NUMBER OF FAILED SEQUENCES (OUT OF 150) FOR EACH NIST RANDOMNESS TEST [23] APPLIED TO EACH DATA SET

Data Set →	Avalanche			PT/CT	Rnd.	Low Den.		High Den.	
Test ↓	Key	PT	h/w	corr.	PT/CT	Key	PT	Key	PT
F-mono	0	0	2	0	1	2	0	2	1
F-block	0	0	2	1	0	1	0	2	1
Runs	2	3	0	0	0	1	0	0	0
LroO	2	3	0	0	0	2	0	0	0
BMR	0	1	0	0	0	2	0	1	2
DFT	0	0	1	0	0	0	0	2	0
NOTM	0	0	0	0	1	0	3	0	1
OTM	0	0	0	2	0	0	2	0	0
Maurer	4	2	0	0	0	0	3	2	2
Lin. Com.	0	0	0	0	0	0	1	1	1
Ser. Com.	0	1	0	0	0	0	0	0	0
App. Ent	0	2	5	0	2	0	0	0	1
Cusums	0	0	0	0	1	0	0	2	1
Rnd. Ex.	2	2	3	0	0	3	2	1	3
REV	2	2	3	0	1	3	1	4	5

However, it needs to be noted that the randomness of SPE is dependent on the number of PoEs. Initial tests using SPE with fewer than 16 PoEs (for a 8×8 crossbar) fail a large number of tests. Randomness increases with a larger number of overlapping polyominos (S in 9 can be tuned to produce several results with a varying number of PoEs). Fig. 12 shows that the average correlation between PT and CT decreases as the number of number of PoE used in SPE increases. A correlation coefficient < 0.01 shows sufficient randomness of the SPE [23].

B. Defending Against Attack 1

The attacker steals the NVMM either during system operation or after power down. The key, which is stored in a volatile memory, is lost. Data may only be leaked using a brute force attack.

1) CT-Only/Brute Force Attack: SPE encrypts 64 bytes of data (cache block size) at a time. Four 8×8 crossbars are used to store 64-bytes of data. Using our ILP model, an 8×8 crossbar requires 16 PoEs to encrypt the entire memory. In SPE, the sequence in which voltages are applied to PoEs is critical for decrypting the data. Using a brute force attack, we have P_{16}^{64} possible PoE sequences. We assume 32 discrete

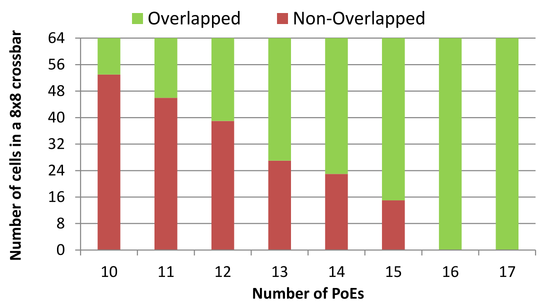


Fig. 13. Polyomino coverage in an 8×8 crossbar for various numbers of PoEs. Cells covered by a single polyomino (red) create security vulnerabilities.

pulses (16 pulse widths for both ± 1 V) that can be applied at each PoE, resulting in 32^{16} voltage combinations. Hence, a brute force attack takes $\sim 10^{32}$ years (100 ns per PoE) to determine the key. A similar attack on an AES block cipher takes $\sim 10^{38}$ years.

What if the attacker knows the ILP? Even if the attacker has access to the ILP, a brute force attack is required since the PoE sequence is unknown. The attacker has to check $16! \times 16^{16}$ combinations which takes $\sim 10^{19}$ years.

Data decryption can only be performed on the same SNVMM it was encrypted. This prevents parallelization of the brute force attack by creating several copies of the data. Also, a brute force attack may force the NVMM to reach its endurance limit [18], destroying the memristors, and any data stored in them.

2) Known-PT Attack: The attacker knows part of the PT and the corresponding CT, because of known structures such as file headers that are encrypted together with the unknown parts of the PT. Even if the attacker has access to a PT-CT pair and the PoE addresses, the shape of the polyomino and the pulse widths of the voltage applied at each PoE is unknown to the attacker. Based on the initial and final resistances of the memristors at the PoEs, the attacker can determine the applied voltage pulses. However, if the memory cell is encrypted by more than one overlapping polyominos, several possible pulse combinations (one at each PoE) can be applied to reach the final resistance.

Fig. 13 shows the correlation between the number of PoEs and the coverage of cells by polyominos. In this figure, the red bars show the cells covered by one polyomino and the green bar shows the cells covered by multiple polyominos. Cells covered by a single polyomino are vulnerable to a known-PT attack since the attacker can infer the applied voltage based on the initial and final resistance of the memristor cell. Cells covered by several polyominos are secure. The voltage pulses can not be inferred from the resistance of the memristors cells since multiple pulse combinations can result in the same resistance. The pulse generator used in SPE is capable of generating 32 different pulse widths, and each voltage pulse is applied across one of 16 different PoEs, which may be ordered in 16! possible sequences. The degree of complexity forces the attacker to resort to a brute force attack.

C. Defending Against Attack 2

1) Chosen-PT Attack: The attacker performs a chosen-PT attack by storing a PT in the NVMM and analyzing the corresponding CT. The attacker faces a challenge similar to a known-PT attack. The overlapping polyominos prevent the attacker from determining the voltage pulses applied. Even for an all-zero PT, the CT is sufficiently random (as seen in Section VI-A).

2) Insertion Attack: In an insertion attack, the attacker knows a CT-PT pair and has the capability to make the SPECU encrypt the same PT with the same key again, but with one bit of the PT (that is known to the attacker) inverted. The attacker performs a statistical analysis to determine the key. In Section VI-A, we apply NIST randomness tests [23] to data collected using PT avalanche and show that no correlation can be determined for use in an insertion attack.

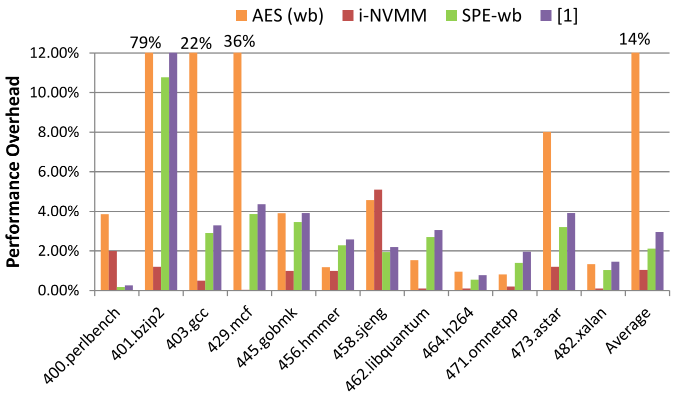
D. Defending Against Attack 3

1) Cold Boot Attack: During power down, there is a delay between the initiation of power down and the time when all data on the NVMM is secured [12]. The attacker may exploit this delay and leak data from the NVMM before it is completely secure. DRAM memory retains its data for up to 3.2 s after power down [12]. For an 8×8 crossbar SPE uses 16 write operations (one at each PoE) in order to encrypt one block of data. Each write operation takes around 100 ns and hence it takes 1600 ns to encrypt 64 bytes of data. On a power down, all data residing in the cache is written back into the NVMM and encrypted. It is extremely unlikely that the entire cache is written back to the memory, but even then it takes 32.7 ms (compared to 3.2 s in DRAM) to encrypt all the data stored on the 2 MB cache. Thus, the window of opportunity is very small for the attacker to exploit. However, if the attacker were to instantaneously cut power to the system the data stored in the intermediate memory may be unencrypted or partially encrypted and is vulnerable. In this case, the attacker has access to ¡ 64 bytes (cache block size) of data and the remaining 99.99% of the data in the NVMM is secure.

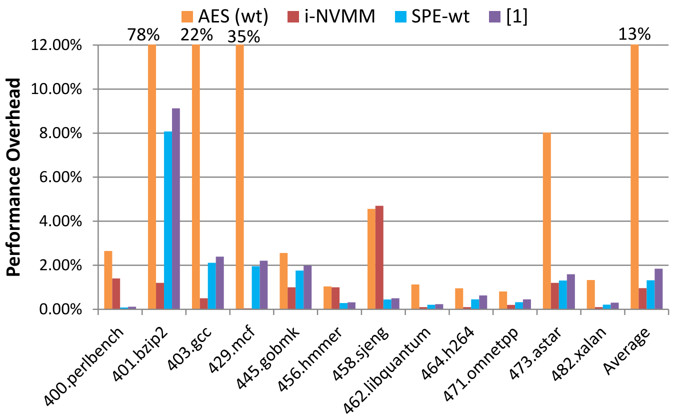
VII. Experimental Evaluation

We evaluated the performance impact of SPE using Zesto [24], a cycle accurate full-system simulator. We model a 3.2 GHz, single-threaded, four-issue out-of-order processor core. The private L1 instruction and data caches are eight-way set associative, 32 kB in size, and incur an access latency of four cycles. The shared L2 cache is 16-way set associative, 2 MB in size, and has 16 cycle access latency. All caches have 64-bit line size and use LRU replacement policy. We model the main memory as a single-rank, 800 MHz, 2 GB with eight devices (128 MB per device).

We simulate two versions of SPE with different cache write schemes [shown in Fig. 1(b) and (c)]. Both SPE with write-back (SPE-wb) and SPE with write-through (SPE-wt) are simulated. In SPE-wt, each read and write operation between NVMM and cache is delayed by 16 cycles for the encryption/decryption. In SPE-wb, the write operation is delayed by an additional 16 cycles only if the data in the cache is found to be dirty. We also evaluate the performance of AES block ciphers (without "least used" encryption [6]). We compare these results to stream ciphers [8], PAD-XOR [9], and



(a)



(b)

Fig. 14. Comparison of SPE to other encryption techniques [6], [8] using (a) write-back and (b) write-through cache read/write policies.

i-NVMM [6]. We also compare our results to the SPE architecture proposed in [14]. We use a number of SPEC CPU2006 benchmarks circuits [25]. All benchmarks are executed for 500 million instructions.

A. Performance Overhead

Fig. 14(a) shows the performance overhead of SPE compared to AES and i-NVMM [6] when using cache write-back policy. Fig. 14(b) shows the performance overhead using cache write-through. The performance impact of SPE-wb is around 2.1% compared to 1.3% of SPE-wt. i-NVMM has a 0.5% performance impact (for both write-back and write-through), however, it uses partial encryption (some parts of the memory are unencrypted at any given point of time). The bulk encryption using AES has an overhead of 13% and 14% for a write-back and write-through cache read/write policy, respectively. The SPE architecture we proposed in [14] performs SPE directly on the NVMM (i.e., no use of an intermediate memory). This architecture causes the NVMM to be inaccessible during encryption/decryption, resulting in a ~2.2% performance impact. Zhang *et al.* [9] has a low performance overhead 2% and is comparable to SPE-wb.

B. Partial Encryption

Fig. 15 shows the percentage of the encrypted memory at any given time. AES always ensures that 100% encryption but

TABLE III
COMPARISON OF SPE WITH AES BLOCK CIPHERS AND STREAM CIPHERS

	AES	i-NVMM [6]	Stream cipher [7]	PAD-XOR [9]	SPE [14]	SPE-wt	SPE-wb
Latency (cycles)	80	80	1	10-116	16-32	32	16
Avg. Performance Impact	14%	1%	0.4%	∼2%	1.5-2.9%	1.3%	2.1%
% Memory Secure	100%	73%	100%	100%	>99.4%	100%	100%
Area Overhead (mm^2)/Technology	8.0 (180nm)[a]	5.3 (N/A)[b]	6.18 (65nm)	0.9 - 84.5(45nm)	1.3 (65nm)	1.4 (65nm)	1.4 (65nm)

[a]Can be approximated to ∼$2.2mm^2$ in 65nm
[b]Technology not mentioned in paper [6]

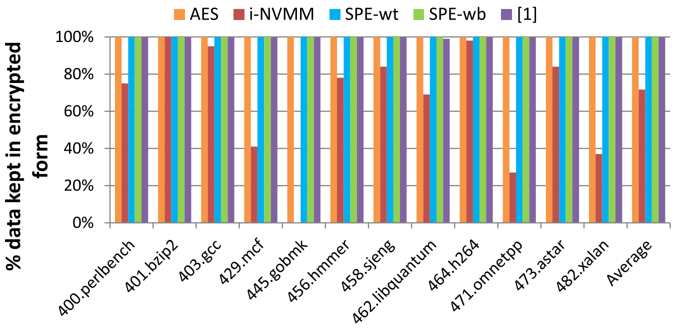


Fig. 15. Percentage of data in NVMM that is kept encrypted.

has a performance impact of 14%. i-NVMM leaves 73% of the NVMM unencrypted. SPE-wt, SPE-wb, and PAD-XOR [9] ensures that 100% of the NVMM is encrypted at all times.

As shown in Fig. 15, workloads that repeatedly access a single memory page such as bzip2 show significant performance impact in our encryption technique compared to i-NVMM. This occurs due to the fact that i-NVMM only encrypts pages that have not been accessed for a fixed duration. However, in applications that access data from several memory pages, such as sjeng, SPE is more efficient.

Table III summarizes different encryption techniques. AES has the largest performance impact of 14% arising from its large latency. i-NVMM reduces performance impact to 1% by using a "least used" encryption technique. In i-NVMM, on average 73% of the data is encrypted during run time making it vulnerable to Attacks 1 and 3. SPE has an area overhead of 1.4 mm^2. Stream ciphers and PAD-XOR [9] provide a low latency and high security (100%) encryption but suffers from an area overhead $\sim 5\times - 60\times$ of SPE. Stream ciphers are also vulnerable to correlation attacks, distinguishing attacks, etc., and are not considered as secure as block ciphers [10]. The SPE architecture we proposed in [14] performs SPE directly on the NVMM (i.e., no use of an intermediate memory) resulting in a lower area overhead at the cost of a larger performance impact.

VIII. DISCUSSION

A. Guard Band for Memristor

There are maximum/minimum limits to the resistance of a memristor and encryption/decryption at resistances close to these limits may result in errors. For example, applying a positive voltage pulse to a memristor that is at maximum resistance shows no change in resistance. This problem can be circumvented by using a "buffer" region in the memristor.

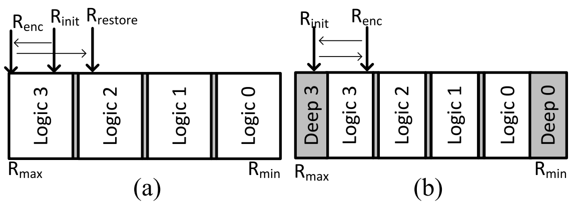


Fig. 16. (a) Erroneous decryption due to saturation. (b) Deep state limits the range of resistances, but allows decryption.

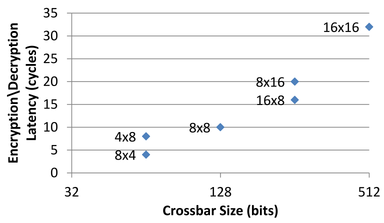


Fig. 17. Change in encryption/decryption latency with crossbar size. We assume that all cells are a four-level MLC capable of storing two-bits of data.

In Fig. 16(a), the memristor stores a logic 3 with a resistance R_{init}. A positive voltage pulse increases the resistance to R_{enc}. However, the resistance of the memristor is limited to R_{max} and will be erroneously restored. This effect can be avoided by introducing "Deep" states to the memory [26]. We reduce the range of resistance in the memristor that can be used during normal read/write operations. We introduce a range of resistances close to R_{max} and R_{min} that act as a buffer region and allow for proper restoration of resistances during decryption. Although for larger MLCs (larger number of bits in each cell) it is more difficult to introduce deep states into the memory cell, larger MLCs improve the security of SPE at the cost of more complex peripheral circuitry.

B. Encryption/Decryption Latency

Fig. 17 shows the encryption latency for various crossbar dimensions using the polyomino shown in Fig. 6. The encryption/decryption latency is the number of PoEs that is required to encrypt the data in the crossbar. As shown in Fig. 17, the encryption latency is directly affected by the crossbar size and the aspect ratio. We also see that crossbars with more rows than columns ($m > n$) have a lower encryption time than crossbars with the same capacity and ($m < n$). This is due to the vertical shape of the polyomino (Fig. 6). The encryption latency is dependent on the crossbar size and not the capacity of the crossbar. Hence, increasing the number of logic levels per cell improves latency. Nonvolatile memories such as

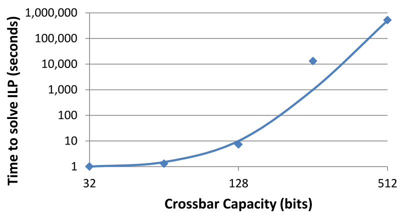


Fig. 18. Time taken to determine PoE locations using the ILP.

TABLE IV
FITTING PARAMETERS AND CONSTANTS USED IN MODELING PCM [30], [31]

Constants	Value
q	1.67×10^{-19}
τ_0	10^{-14}
T	$298K$
$(E_c - E_f)$	0.31
Δz	5×10^{-9}
N_T	16×10^{24}

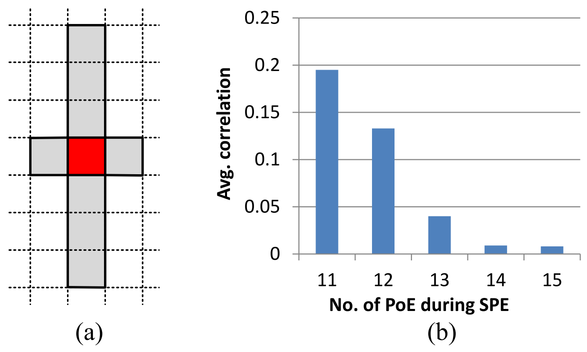


Fig. 19. (a) Smallest polyomino in PCM-based crossbar. Red squares is PoE and gray squares represent polyomino. (b) Average correlation between PT and CT as number of PoEs used in encryption increases. A correlation of 1 denotes that the PT and CT are strongly correlated and vulnerable to attacks. A correlation < 0.01 shows that there is no correlation [23].

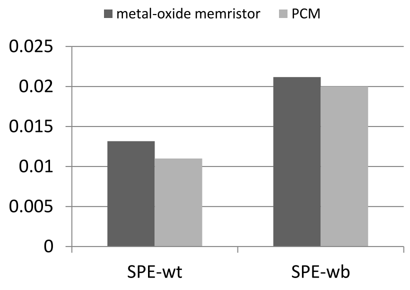


Fig. 20. Performance impact of using SPE on a PCM-based crossbar compared to SPE on a memristor-based crossbar.

memristors and PCM demonstrate asymmetric read and write. This means that the read and write processes are different from each other, in respect of timing, energy consumption, and even reliability. Designing for the worst case scenario may lead to reduced performance, power and reliability overhead. Hence, techniques such as [27] and [28] can be used along with SPE to maximize performance and reliability.

C. Time to Solve ILP

The ILP was run on a Intel Core i5 CPU at 3.1 GHz, with 8G RAM using the FICO Xpress optimization suite [21]. Fig. 18 shows the time taken to determine one set of PoEs that can be used to encrypt/decrypt the crossbar. The time increases exponentially with the size of the crossbar making it harder for the attacker to determine the PoEs used during SPE. The ILP generates a nonunique solution. An attacker who is trying to determine the set of PoEs used during decryption will have to solve the ILP several times in order to determine all possible solutions.

D. SPE With PCM

SPE can be used to perform encryption on any emerging memory technology. In this section, we adapt SPE for use with PCM [29]. We update out device model with the sub-threshold conduction and threshold switching PCM model presented in [30] and [31]. The I-V relationship is shown by

$$I(t) = 2q\frac{\pi r_{eff}^2}{\tau_0} N_T \Delta z . e^{\frac{-(E_c - E_f)}{kT}} . \sinh\left[\frac{q\Delta z V(t)}{2kTu_{Aeff}}\right] \quad (11)$$

where u_{Aeff} is a fitting parameter and q, $\tau 0$, T, $(E_c E_f)$, Δz, and N_T are physical constants. The constants are summarized in Table IV.

We first determine the size of the smallest possible polyomino using our SPICE framework V. The polyomino is shown in Fig. 19(a). Equation (10) of the ILP is updated to reflect the new polyomino. On solving the updated ILP equations, we see that the minimum number of PoE required to ensure that all memory cells are covered by at least one polyomino is 11. The number of PoE used during SPE is determined by studying the correlation between PT and CT as shown in Section VI-A. Fig. 19(b) shows the correlation between PT and CT. As we can see, the correlation drops below 0.01 (threshold for randomness [23]) for 14 PoE. Hence, 14 PoEs are required to satisfy randomness and to provide sufficient security. Fig. 20 shows the performance impact of using SPE in a PCM-based crossbar. SPE uses 14 PoEs in a PCM-based crossbar, compared to 16 in a memristor-based crossbar, resulting in a lower encryption latency. Hence, when using a write-through cache, a PCM-based crossbar has a 1.1% performance impact compared to 1.3% when using a memristor-based crossbar. Similarly, when using a write-back cache the performance impact is 2% and 2.1% for PCM-based and memristor-based crossbars, respectively.

E. Memristor Endurance

The limited endurance is a problem with NVMMs. A typical NVMM cells write endurance is in the range of $10^6 \sim 10^8$ depending on the technology used [4], [32]. Hence, extensive research has been performed to minimize wear out and improve lifetime of NVMMs. There are two techniques that can be used to improve lifetime of an NVMM.

1) *Write Reduction:* This method minimizes the number of updated bits in the NVMM for each write operation. A method of partial write where only the cache lines written back from last level cache is shown in [33]. In the

DCW method [34], data in NVMM is first read out and compared with new data bit-by-bit. Only the modified bits are updated in the write operation. These techniques can effectively enhance the lifetime by up to 85% [34].

2) *Wear Leveling:* Even with write reduction techniques, some cells in NVM memory may wear out faster than the others due to nonuniform write intensity [34]. Wear leveling techniques such as table-based remapping, start-gap, security-refresh, etc., have been proposed [4], [32], [34].

We assume that write reduction and wear leveling techniques are applied to the intermediate nonvolatile memory to improve its lifetime.

F. Temperature Effects on Memristor

In this paper, we assume that our memristors have the same characteristics independent of the operating temperature. In practice memristor characteristics are temperature-dependent and the switching speed and other physical parameters vary with temperature [35]. Hence, the temperature during decryption and encryption needs to be the same. In order to correctly perform SPE, temperature sensors can be used to ensure that the memory is within a certain temperature range during encryption and decryption eliminating the effect of temperature. Merkel and Kudithipudi [36] propose an architecture that enables each memristor in the memory die to be used both as a memory bit and a temperature sensor. We are considering the effect of temperature on SPE in our future paper.

IX. Conclusion

Although NVMM has recently attracted attention, various security concerns continue to arise over such devices. Sensitive data written to NVMM persists even when the system is powered down and can be leaked easily. We have presented SPE that exploits the physical parameters of the NVMM to secure the data. Data secured by SPE can only be decrypted on the same SNVMM used for encryption. SPE also has a low performance and area overhead. However, memristor devices are affected by real-world conditions such as temperature variations between encryption and decryption operations. The simulations provided in this paper serves as a proof of concept for SPE. Further exploration into the stability of SPE is an interesting direction for future research. Currently, there are no available models describing the effect of temperature on a memristor. However, memristors are still an emerging technology and such models will become available as the technology matures.

References

[1] J. Hutchby and M. Garner, "Assessment of the potential and maturity of selected emerging research memory technologies," in *Int. Technol. Roadmap Semicond.*, San Francisco, CA, USA, Jul. 2010, pp. 1–50 [Online]. Available: http://www.itrs.net

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and S. R. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[3] Micron. (2013, Feb.). *About Phase Change Memory*. [Online]. Available: http://www.micron.com/products/phase-change-memory

[4] M. K. Qureshi *et al.*, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. IEEE/ACM Int. Symp. Microarchit.*, New York, NY, USA, 2009, pp. 14–23.

[5] T. Gilmont, J. Legat, and J. J. Quisquater, "Enhancing the security in the memory management unit," in *Proc. EUROMICRO Conf.*, vol. 1. Milan, Italy, 1999, pp. 449–446.

[6] S. Chhabra and Y. Solihin, "i-NVMM: A secure non-volatile main memory system with incremental encryption," in *Proc. IEEE Int. Symp. Comput. Archit.*, San Jose, CA, USA, 2011, pp. 177–188.

[7] J. Yang, L. Gao, and Y. Zhang, "Improving memory encryption performance in secure processors," *IEEE Trans. Comput.*, vol. 54, no. 5, pp. 630–640, May 2005.

[8] J. Valamehr *et al.*, "Inspection resistant memory: Architectural support for security from physical examination," *SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 130–141, Jun. 2012.

[9] X. Zhang, C. Zhang, G. Sun, J. Di, and T. Zhang, "An efficient run-time encryption scheme for non-volatile main memory," in *Proc. Int. Conf. Compilers Archit. Synth. Embedded Syst.*, Montreal, QC, Canada, Oct. 2013, pp. 1–10.

[10] E. Zenner and M. Boesgaard, "How secure is secure? On message and IV lengths for synchronous stream ciphers," *Cryptico A/S Tech. Rep.*, May 2005.

[11] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. Chichester, U.K.: John Wiley, Apr. 2008.

[12] R. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, Austin, TX, USA, 2006, pp. 157–167.

[13] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam, The Netherlands: Elsevier Publishers, 2011.

[14] S. Kannan, N. Karimi, and O. Sinanoglu, "Secure memristor-based main memory," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2014, pp. 1–6.

[15] Z. Huanguo, Q. Zhongping, and Y. Qi, "Design and implementation of the TPM chip J3210," in *Proc. Trusted Infrastruct. Technol. Conf.*, Hubei, China, 2008, pp. 72–78.

[16] M. Ziegler and M. Stan, "Design and analysis of crossbar circuits for molecular nanoelectronics," in *Proc. IEEE Conf. Nanotechnol.*, Washington, DC, USA, 2002, pp. 323–327.

[17] C. Yakopcic, T. Taha, G. Subramanyam, R. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Lett.*, vol. 32, no. 10, pp. 1436–1438, Oct. 2011.

[18] J. J. Yang *et al.*, "High switching endurance in TaO_x memristive devices," *Appl. Phys. Lett.*, vol. 97, no. 23, pp. 232102–232103, 2010.

[19] H. Manem and G. S. Rose, "A read-monitored write circuit for 1T1M multi-level memristor memories," in *Proc. Int. Symp. Circuits Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 2938–2941.

[20] R. S. Katti and R. G. Kavasseri, "Secure pseudo-random bit sequence generation using coupled linear congruential generators," in *Proc. Int. Symp. Circuits Syst.*, Seattle, WA, USA, 2008, pp. 2929–2932.

[21] (2013, Feb.). *Fico Xpress Optimization Suite*. [Online]. Available: http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx

[22] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM—ThrEshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.

[23] A. Rukhin *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *Comput. Security Resour. Center Nat. Inst. Std. Technol.*, Tech. Rep. NIST SP—800-22 Rev 1a, Sep. 2010.

[24] G. H. Loh, S. Subramaniam, and Y. Xie, "Zesto: A cycle-level simulator for highly detailed microarchitecture exploration," in *Proc. IEEE Int. Conf. Perform. Anal. Softw. Syst.*, Boston, MA, USA, Apr. 2009, pp. 53–64.

[25] S. P. E. Corporation. (2011.). *SPEC2006 Benchmark v1.2*. [Online]. Available: http://www.spec.org/cpu2006/

[26] S. Kannan, J Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of crossbar-based non-volatile random access memories," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 413–426, May 2013.

[27] Y. Zhou, C. Zhang, G. Sun, K. Wang, and Y. Zhang, "Asymmetric-access aware optimization for STT-RAM caches with process variations," in *Proc. Int. Conf. Great Lakes Symp. Very Large Scale Integr. (VLSI)*, Paris, France, 2013, pp. 143–148.

[28] G. Sun, Y. Zhang, Y. Wang, and Y. Chen, "Improving energy efficiency of write-asymmetric memories by log style write," in *Proc. Int. Symp. Low Power Electron. Design*, Redondo Beach, CA, USA, 2012, pp. 173–178.

[29] S. Raoux *et al.*, "Phase-change random access memory: A scalable technology," *IBM J. Res. Develop.*, vol. 52, pp. 465–479, Jul. 2008.

[30] D. Ielmini and Y. Zhang, "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices," *J. Appl. Phys.*, vol. 102, no. 5, Sep. 2007, Art. ID 054517.

[31] D. Ielmini, "Unified physical modeling of reliability mechanisms and scaling perspective of phase change memory," *Curr. Appl. Phys.*, vol. 11, no. 2, pp. e85–e91, Mar. 2011.

[32] 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 *Proc. Int. Symp. Comput. Archit.*, Saint-Malo, France, Jun. 2010, pp. 383–394.

[33] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 2–13, Jun. 2009.

[34] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. Int. Symp. Comput. Archit.*, Austin, TX, USA, Jun. 2009, pp. 14–23.

[35] C. Walczyk *et al.*, "Impact of temperature on the resistive switching behavior of embedded HfO_2-based RRAM devices," *IEEE Trans. Electron Devices*, vol. 8, no. 9, pp. 3124–3131, Sep. 2011.

[36] C. Merkel and D. Kudithipudi, "Temperature sensing RRAM architecture for 3-D ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 4, pp. 878–887, Apr. 2014.

Sachhidh Kannan (M'10) received the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic Institute of New York University, Brooklyn, NY, USA, in 2009 and 2014, respectively.

He was with IBM Research Laboratory, Zurich, Switzerland, in 2012, where he was involved with characterization and modeling of phase-change memories. His current research interests include hardware security and implementation, applications of emerging memories, test techniques and security for emerging memory technologies, in particular memristors and phase-change memories. He has published over ten conference and journal papers at several international conferences and publications.

Mr. Kannan was the recipient of the IBM Great Minds Internship Award.

Naghmeh Karimi (M'03) received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the University of Tehran, Tehran, Iran, in 1997, 2002, and 2010, respectively. Her master's thesis was on testability enhancement at the register transfer level and her Ph.D. thesis was on concurrent error testing and reliability enhancement.

She was a Visiting Researcher at Yale University, New Haven, CT, USA. She joined as a Post-Doctoral Researcher at Duke University, Durham, NC, USA, for one year and also served as a Visiting Assistant Professor at New York University, Brooklyn, NY, USA, for two years. She is currently with the Electrical and Computer Engineering Department, Rutgers University, New Brunswick, NJ, USA, as a Visiting Assistant Professor. Her current research interests include design-for-testability, design-for-security, concurrent testing, fault tolerance, reliability enhancement, and hardware security.

Ozgur Sinanoglu (M'00) received the B.S. degrees in electrical and electronics engineering and in computer engineering, both from Bogazici University, Istanbul, Turkey, and the M.S. and Ph.D. degrees in computer science and engineering from the University of California, San Diego, San Diego, CA, USA, in 1999, 2001, and 2004, respectively.

He is an Associate Professor at the Electrical and Computer Engineering Department, New York University Abu Dhabi, Abu Dhabi, U.A.E. He was also with TI, Dallas, TX, USA, IBM, Madison, NY, USA, and Qualcomm, San Diego, CA, USA. His current research interests include design-for-test, design-for-security, and design-for-trust for VLSI circuits. He has published over 120 conference and journal papers and holds 15 issued and pending U.S. patents. He has given over a dozen tutorials on hardware security and test at various IEEE conferences.

Dr. Sinanoglu was the recipient of the IBM Ph.D. Fellowship Award twice, the Best Paper Awards of the IEEE VLSI Test Symposium 2011 and ACM Conference on Computer and Communication Security 2013.

Ramesh Karri (M'91) received the Ph.D. degree in computer science and engineering from the University of California, San Diego, San Diego, CA, USA.

He is a Professor at the Electrical and Computer Engineering Department, Polytechnic School of Engineering, New York University, Brooklyn, NY, USA. His current research interests include trustworthy ICs and processors, high assurance nanoscale IC architectures and systems, VLSI design and test, and interaction between security and reliability. He has published over 150 journal and conference publications in the above areas. He has authored two invited articles in the IEEE COMPUTER ON TRUSTWORTHY HARDWARE, an invited article on Digital Logic Design using memristors in the Proceedings of IEEE, and an invited article in the IEEE COMPUTER ON RELIABLE NANOSCALE SYSTEMS.

Prof. Karri was the recipient of the Humboldt Fellowship and the National Science Foundation CAREER Award. He is the Area Director of the NY State Center for Advanced Telecommunications Technologies, Polytechnic Institute of New York University (NYU), Brooklyn, NY, USA, for cyber security, a Hardware Security Lead at the Center for Research in Interdisciplinary Studies in Security and Privacy, a Co-Founder of the Trust-Hub, and organizes the annual red team blue team event at NYU, and the Embedded Systems Challenge. He also co-founded the IEEE/ACM Symposium on Nanoscale Architectures (NANOARCH). He is the Program Chair in 2012 and a General Chair in 2013 of the IEEE Symposium on Hardware Oriented Security and Trust (HOST). He is the Program Co-Chair in 2012 and a General Co-Chair in 2013 of the IEEE Symposium on Defect and Fault Tolerant Nano VLSI Systems. He is also the General Chair of the 2009 and 2013 NANOARCH. He is the General Co-Chair of the International Conference on Computer Design 2015, Radio Frequence Identification Security workshop 2015, and Conference on Security and Privacy in Wireless and Mobile Networks 2015. He serves on several program committees. He was the Co-Founder and served as the Chair of the IEEE Computer Society Technical Committee on NANOARCH. He was an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY from 2010 to 2014, the IEEE TRANSACTIONS ON CAD in 2014, the *ACM Journal of Emerging Computing Technologies* from 2007, and *ACM Transactions on Design Automation of Electronic Systems* from 2014. He is an IEEE Computer Society Distinguished Visitor from 2013. He is on the Executive Committee of the IEEE/ACM Design Automation Conference leading the Security@DAC initiative. He has organized/delivered invited tutorials on Hardware Security and Trust, including VLSI Test Symposium 2012, 2014, the International Conference on Computer Design 2012, the IEEE North Atlantic Test Workshop 2013, Design Automation and Test in Europe 2013, the IEEE International Test Conference 2014, the IEEE/ACM Design Automation Conferene, 2014, and the IEEE Latin-American Test Symposium 2014.