A Secure Integrity Checking System for Nanoelectronic Resistive RAM

Md Badruddoja Majumder, Md Sakib Hasan, Mesbah Uddin and Garrett Rose

IEEE Transaction on Very Large Scale Integration (VLSI) Systems, VOL. 27, No. 2, February 2019.

Citation Information (BibTex):


```
@ARTICLE{8527666,
author={M. B. {Majumder} and M. S. {Hasan} and M. {Uddin} and
G. S. {Rose}},
journal={IEEE Transactions on Very Large Scale Integration
(VLSI) Systems},
title={A Secure Integrity Checking System for Nanoelectronic
Resistive RAM},
year={2019},
volume={27},
number={2},
pages={416-429},
doi={10.1109/TVLSI.2018.2876693},
ISSN={1063-8210},
month={Feb},
}
```

A Secure Integrity Checking System for Nanoelectronic Resistive RAM

Md Badruddoja Majumder, *Student Member, IEEE,* Md Sakib Hasan, *Member, IEEE,* Mesbah Uddin, *Student Member, IEEE,* and Garrett S. Rose, *Member, IEEE*

***Abstract*—Recent advances in resistive random access memory (RRAM) as high density, low power, and faster memory systems drive the need for devising a more lightweight integrity checking system for RRAM. In this paper, we design a new tag generation system for integrity checking of RRAM. A single read operation to a crossbar RRAM in the presence of sneak path currents can output a tag for the memory data that can be used for integrity checking. An analytical approach to model such a tag generation process is described in this paper. Security results predicted by the analytical model provide various design options leading to an optimal system from the perspective of considered security properties. The proposed design is simulated to investigate and verify the security properties of the system for a number of optimal design options predicted by the analytical model. Reliability of the proposed system is also measured for varying conditions of device parameters, operating temperatures, load resistances and read voltage. Finally, the performance of the proposed system is compared against another existing lightweight tag generation method from the perspective of energy consumption, transistor count and delay.**

***Index Terms*—RRAM, memristor, integrity checking, tag, crossbar, uniformity, avalanche. diffusion, reliability.**

I. Introduction

Resistive random access memory (RRAM) is an emerging technology for next generation computer memory systems [1], [2]. Features like non-volatility, high density, and fabrication compatibility with existing technology makes RRAM an attractive candidate for replacing existing memory technologies. Due to their convincing features from different perspectives, RRAMs are being investigated for a number of applications including non-volatile memory for emerging computing system.

The reliability and security of any classical computing system largely depends on the integrity of data fetched from memory. Usually, the main memory of a computing system is built on a separate chip other than the processor itself and often is a target for attack. Therefore, increasing possibility of RRAM as future non-volatile main memory, demands an efficient design for its data integrity checking system.

In general, memory integrity checking protocols are based on generating tags from data to be stored in the memory which can be later used to verify data read from the memory. Hash and message authentication code (MAC) based protocols are common approaches to memory integrity verification [3]–[6]. Some researchers have proposed block cipher based protocols for memory authentication [7]–[9]. Hong *et al.* proposed a cost effective tag design (CETD) method for memory authentication targeting embedded system applications [10]. Liu *et al.* extended Hong *et al.*'s work in order to improve security [11].

Most existing works on memory integrity checking use a separate cryptographic module such as hash functions or block ciphers for tag generation. These cryptographic functions require computationally extensive mathematical operations to generate a tag from memory data and hence exhibit significant area, power and delay overhead. With a view to designing more lightweight integrity checking applications for RRAM, Majumder *et. al.* proposed a technique that leverages sneak path currents through the crossbar memory for tag generation and hence requires less overhead [12]. Security applications of RRAM using sneak path currents have also been proposed in several other works where it proved to be more efficient than conventional techniques for those applications [13], [14].

In this paper, we propose a new sneak path based tag generation and integrity checking method for nanoelectronic RRAM that builds on earlier work [12]. The proposed technique is presented in a generalized way which can be applied to any RRAM system though we consider here a specific memristor device for the sake of simulation.

The contributions of this paper include: i) development of an analytical model for tag generation from RRAM leveraging sneak path currents, ii) formulation of different design choices based on desired security goals, iii) security analysis from both analytical and experimental models of the system, and iv) detailed reliability analysis considering possible sources of variation.

The remainder of the paper is organized as follows. Section II provides some background regarding resistive switching devices including a model for an emerging resistive switching device, the memristor, and different memory architectures for RRAM. Section III presents the adversarial model and assumptions considered for the proposed design. Section IV describes the desired security goals for the proposed design. Section V and Section VI describes the proposed tag generation method and its analytical model, respectively. Section VII includes different design options and associated security. Section VIII includes the circuit design for the tag generation system. Detailed security analysis of the proposed method has been presented in Section IX. Reliability of the proposed design considering different possible sources of variation is analyzed in Section X. Performance comparisons between the

This work is partially based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0301.

M.B. Majumder, M.S. Hasan, M. Uddin, and G.S. Rose are with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, 37996. (email: {mmajumde, mhasan4, muddin, garose}@utk.edu)

proposed method and another method described in literature are provided in Section XI and the paper is concluded with Section XII.

II. RRAM Background

Emerging non-volatile memory (NVM) architectures are investigated more and more in recent years with a view to replacing conventional charge based volatile memory. NVM technology of emerging trends mainly include Phase Change Memory (PCM), Spin Torque Transfer RAM (STT-RAM) and Resistive switching RAM (RRAM). Despite different internal operating mechanisms, all of these technologies share some common features such as all being two terminal devices, switching between high and low resistive states also known as OFF and ON state, and transition between the ON and OFF state is acquired by applying an electrical excitation via voltage or current. In this paper, we consider memristor based RRAM technologies for designing an integrity checking system.

Many researchers independently reported resistive switching mechanisms using different metal insulator metal (MIM) device stacks [15], [16]. Transition metal oxides are mostly used as the insulator of these MIM structures. Due to the switching property between two stable resistive states, such devices have great potential as non volatile memory elements. A popular name used for such metal oxide switching devices is memristor which stands for "memory resistor" and well represents characteristics of metal oxide resistive switching devices. The term memristor was first theoretically proposed by Leon Chua as the fourth fundamental circuit element after resistor, capacitor and inductor [17]. Researchers of Hewlett Packard (HP) labs reported a metal oxide switching behavior as the first experimental characterization of the idea of a memristor [18].

A. Memristor Operation and Model

A memristor's current-voltage can be related by:

$$v(t) = M(t)i(t) = \frac{d\phi}{dq}i(t), \tag{1}$$

where $M(t)$ is the instantaneous resistance of the memristor, also known as memristance; $\phi(t)$ and $q(t)$ are flux linkage and charge, respectively.

A memristor can be switched between its low and high resistive states by applying an external voltage across the terminals of the device. The low resistive state or ON state is obtained by applying a positive voltage pulse having a magnitude above a certain threshold voltage and a certain minimum pulse width. Similarly, switching from low to high resistance state is obtained by applying a negative voltage pulse of a specific minimum pulse magnitude and width. Two resistive states of the memristor can be used as binary data where low and high resistance states correspond to 1 and 0, respectively.

The memristor model considered for the RRAM simulation in this paper is described in greater detail in [19]. According to this model, with an applied voltage greater than a threshold and a minimum switching time, the memristor can be switched between its ON and OFF states. When the applied voltage is less than the threshold, it results in a negligible change in the memristance level. Therefore, the applied voltage for reading a memristive memory element must be below its switching threshold level for a reliable and non-destructive read operation.

B. Memristor based RRAM Architecture

The crossbar structure for memristor based RRAM is a promising architecture for future memory technology. In the crossbar architecture, a memristor is placed at the crosspoint of each horizontal and vertical metal wire and thus can be used to form a very high density memory. However, the crossbar RRAM where only a single memristor is used for each memory cell suffers from current leakage through unselected cells. Conductive paths through unselected cells are called sneak paths. Sneak path currents are undesirable as they cause read and write disturbances in crossbar RRAM [20]. Due to sneak path currents when a voltage is applied to a particular memory cell for reading or writing, current also flows through the unselected cells. Thus, unselected cells are written undesirably in the write process. In the read process, unselected cells contribute to the reading of the selected cells and thus possibly can cause an error.

Researchers have proposed some modified architectures such as 1T1R and 1D1R for the crossbar RRAM to mitigate sneak path problems [21], [22]. In these architectures, a selector device is used along with the resistive switching device (memristor) in each memory cell so that current flow can be prevented through unselected cells. The 1T1R structure proposes a transistor whereas the 1D1R concept uses a diode as the selector device for each memory cell. Since 1D1R only applies to unipolar resistive switching devices, in this paper we consider 1T1R structure for the memristor based RRAM architecture shown in Fig. 1. Another interesting feature of the 1T1R RRAM is that it has the controllability of enabling sneak paths in a particular portion of the memory by keeping corresponding transistor lines turned on. This feature is useful for applications leveraging sneak path currents.

III. Assumptions and Adversarial model

Memory can be subject to unauthorized modification in different ways. Unauthorized modification attacks on memory can be classified into two categories: runtime attack and offline attack. A runtime attack primarily includes different software attacks where a malicious piece of code accesses the memory and manipulates the data. Programming languages such as C/C++ have very low level access to memory enabling malicious software that can access a portion of memory without authorization and modify memory contents. One example is the stack overflow attack where a malicious program writes data to the stack for a number of times in such a way that the stack overflows and an adjacent portion of memory is overwritten [23].

Another example of runtime memory attack is the Direct Memory Access (DMA) attack [24]. DMA is a special feature in a computer system that enables some special peripheral

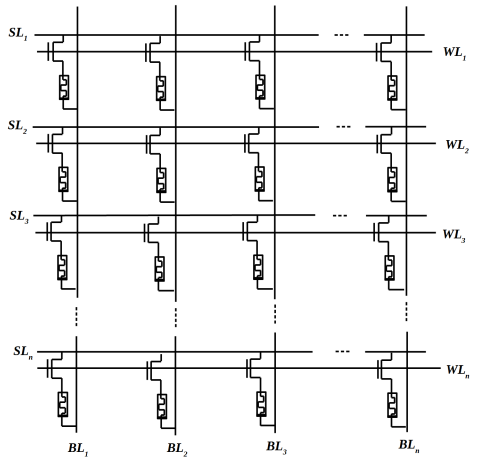

Fig. 1: 1T1R model for Memristor based Crossbar RRAM.

devices to access the memory bypassing the operating system's supervision. The motivation of this feature is to enable the I/O device to interact with memory while the processor can perform other tasks. However, the same feature provides an attacker the opportunity to modify memory content maliciously.

Offline attacks refer to modifying memory content using a different source while the memory is offline with the authorized processor. A relevant example is the Evil Maid Attack [25]. This is an attack where an adversary accesses a powered down computer and manipulates the boot-loader program stored in the non-volatile memory by connecting his own external device to the processor.

In this paper we consider traditional computer architecture for the memory system where the processor and memory are separate. The processor interacts with memory with a shared or separate address-data bus. For writing to and reading from memory, we assume that the processor-memory bus is free of attack. This assumption ensures that, whenever an authorized person sends data to write into the memory or requests to read from the memory, data can't be modified while transmitting through the data bus. However, an attacker can modify the data when it is available in the memory in different ways, such as those described earlier.

IV. Security Goals

For integrity verification of memory data, a tag is generated which represents the data in a compressed way. During every update to the memory, a new tag is generated and stored. During every read operation, the tag is regenerated from the data to be read and verified against the previously stored tag. The end goal for an adversary to attack this system is to manipulate data in such a way that it would generate the same tag as before the manipulation, also known as collision. A secure integrity verification system must be collision resistant which means that it should be computationally hard enough to find two different memory states resulting in the same tag. There are two types of collision resistance, targeted and untargeted collision resistance that a secure integrity checking system must have. These two types of collision resistance are explained below.

1) **Targeted Collision Resistance:**
   Targeted collision resistance is widely known as second preimage resistance in cryptography [26]. Given some data space, x where the tag of x_1 is t_1, it should be very difficult for an attacker to find another x_2 such that $x_2 \neq x_1$ and the tag of x_2 is also t_1. Ideally, a secure integrity verification system has a resistance of 2^{-N} against targeted collision where N is the number of bits in the tag.
2) **Untargeted Collision Resistance:**
   Untargeted collision refers to the actual collision resistance in a cryptographic hash function. It should be difficult for an attacker to find any two data sets x_1 and x_2 such that $x_1 \neq x_2$ and both produce the same tag. Ideally, a secure integrity verification system has a resistance of $2^{\frac{-N}{2}}$ against untargeted collision where N is the number of bits in the tag.

A targeted collision is more difficult to find for an adversary as compared to the untargeted one and therefore resistance against them are also known as strong and weak collision resistance, respectively [27]. In case of targeted collision an adversary can directly exploit the vulnerability by replacing the targeted second preimage of the data. On the other hand, if an adversary finds an untargeted collision, i.e. collision between two random data sets, she can not directly leverage that without any one of the data in the collision pair currently being in the memory.

The above two properties are the end security goals for a memory integrity verification system design. However, these goals are achieved through several properties of the tag generation: uniformity, the avalanche effect and diffusion.

1) **Uniformity:**
   The probability distribution of the tag should be uniform, i.e. all possible tags should be generated with equal probability for a random distribution of data. Without uniformity, some tags would be more likely than others and there would exist an increased chance for collision. Collision resistance for N -bit tag can be expressed as $2^{-\alpha N}$ where α is the indicator of uniformity of a tag distribution. Definition of α is provided in Eq. 31.
2) **Avalanche Effect:**
   Avalanche effect is a well known term used in cryptographic security first used by [28]. Given a small change in the data, it is desirable that 50% of all tag bits be flipped [29]. Without this property, given a small change in data, sample space for generated tag would be smaller. As a result, there would be increased chances of collision.
3) **Diffusion:**
   Diffusion refers to the property that each tag bit be dependent on or sensitive to all bits. Mathematically, for any change made in the memory every bit of the tag should flip with a chance of 50% [30]. This property is also known as the strict avalanche criteria.
   If the system includes the avalanche effect but lacks

diffusion, it might be possible for a small change in the data to affect only a small subset of the 50% bits that are flipped in the tag. In this way, the attack space is shortened and by observing a certain amount of tag generation from different data, it is possible for the attacker to figure out which set of the 50% bits have flipped.

V. Tag Generation Architecture

Let's consider a crossbar RRAM having m rows and n columns as shown in Fig. 2 (a). For tag generation, a non-destructive read voltage, V_R is applied to a row, and k columns are pulled down to ground with a load resistor, R_L in each column. When sneak paths are enabled (all select transistors are ON) in the crossbar RRAM, voltage across the load resistors collectively can be used as a tag, i.e. a compressed representation of the data stored in the entire crossbar. Tag generation architecture proposed in this paper brings major modifications in the basic sneak path current based tag generation architecture described in [12]. All unselected rows are shorted to a high impedance node and similarly, all unselected columns are shorted. Instead of multi bit conversion, each load voltage is converted to single bit which avoids using multi bit ADC hardware.

All memory cells in the crossbar shown in Fig. 2 are classified into four categories and marked accordingly. These 4 categories are: connected between (i) selected row and selected column, (ii) selected row and unselected column, (iii) unselected row and unselected column, and (iv) unselected row and selected column. We use R_0, R_1, R_2 and R_3 to refer to memory cells of these 4 categories, respectively. From the circuit shown in Fig. 2(b) it can easily be inferred that the selected cell resistance, $R_{0,1}$, $R_{0,2}$,..$R_{0,k}$ has significantly greater impact on load voltage and hence the tag, τ. When selected cells are ON, the dependency of generated tag for unselected memory cells is at a minimum. To reduce this bias condition we should keep R_0 always OFF during tag generation. Such a requirement implies that R_0 cannot be a part of the regular memory. However, this can be accomplished if we use a reserved crossbar row as the selected row for tag generation. Unlike the previous architecture, only the unselected cells in the reserved row are written randomly during every write operation and selected cells are kept off which helps improve security. Another modification in this architecture is that k columns used for tag generation are sampled randomly during every new write operation. This randomness helps achieve certain other security properties for integrity checking.

VI. Analytical Model

The crossbar network shown in Fig. 2(a) can be redrawn as in Fig. 2(b) by considering the equivalent resistance of each category. We simplify the equivalent circuit of the crossbar by considering every OFF resistance as an open circuit. This assumption is valid up to a certain level of ON/OFF ratio of the memory element. Since the selected cells $R_{0,1}, R_{0,2}, ..., R_{0,k}$ are always OFF during tag generation, we remove them from the equivalent circuit of Fig. 2(b) while analyzing it for tag generation. Using fundamental circuit analysis we can obtain the voltage across the load resistor in the i^{th} sample column, V_i.

$$V_i = \frac{V_R.G_i.R_L}{1 + (R_{1,eq} + R_{2,eq}).\sum_1^k G_i}, \tag{2}$$

where $G_i = \frac{1}{R_{3i,eq}+R_L}$.

From this, τ_i is the i^{th} bit of the tag which is found by converting V_i into binary value using a comparator with a reference voltage of $\frac{V_R}{2}$. We develop a model here to analyze the probability of each bit, τ_i of the tag being 1 (or 0) in relation to crossbar size, number of sampled columns and the load resistance value.

$$Pr(\tau_i = 1) = Pr(V_i \geq \frac{V_R}{2}). \tag{3}$$

Substituting for V_i, the inequality $V_i \geq \frac{V_R}{2}$ can be rewritten as:

$$\frac{V_R.G_i.R_L}{1 + (R_{1,eq} + R_{2,eq}).\sum_1^k G_i} \geq \frac{V_R}{2}, \tag{4}$$

which reduces to

$$R_{1,eq} + R_{2,eq} \leq \frac{2.G_i.R_L - 1}{\sum_{j=1}^k G_j}. \tag{5}$$

The equivalent resistance of each category of memory cell can be represented in terms of the ON resistance:

$$R_{1,eq} = \frac{R_{ON}}{N_1}, R_{2,eq} = \frac{R_{ON}}{N_2}, R_{3i,eq} = \frac{R_{ON}}{N_{3,i}}, \tag{6}$$

$$R_L = \frac{R_{ON}}{N_L}, \tag{7}$$

where N_L is a constant and N_1, N_2 and N_3 are random variables representing the number of ON cells in the selected row-unselected column, unselected row-selected column and i^{th} sampled column, respectively. The probability distribution of N_1, N_2 and N_3 can be calculated assuming a uniform distribution of memory data. Data distribution in memory depends on the memory instructions used by a program code. Different programs have different data distributions for memory usage. Every computer memory serves a large number of different programs and therefore we assume a uniform distribution for data in memory.

$$N_1 = \{x \epsilon Z : 0 \leq x \leq n - k\} \tag{8}$$

$$N_2 = \{x \epsilon Z : 0 \leq x \leq (m-1) * (n-k)\} \tag{9}$$

$$N_{3,i} = \{x \epsilon Z : 0 \leq x \leq m - 1\} \tag{10}$$

Eq. 5 can be rewritten in terms of N_L, N_1, N_2 and N_3.

$$\frac{1}{\frac{1}{N_1} + \frac{1}{N_2}} \geq (\frac{1}{\frac{1}{N_{3,i}} + \frac{1}{N_L}} + \sum_{j=1, j\neq i}^{k} \frac{1}{\frac{1}{N_{3,j}} + \frac{1}{N_L}}).(\frac{N_{3,i} + N_L}{N_{3,i} - N_L}) \tag{11}$$

Let X represent the random variable $\frac{1}{\frac{1}{N_1} + \frac{1}{N_2}}$ and f_X is the probability density function (P.D.F) of X. Similarly, Y represents the random variable $(\frac{1}{\frac{1}{N_{3,i}} + \frac{1}{N_L}} + \sum_{j=1, j\neq i}^{k} \frac{1}{\frac{1}{N_{3,j}} + \frac{1}{N_L}}).(\frac{N_{3,i} + N_L}{N_{3,i} - N_L})$ and f_Y is the P.D.F of Y.

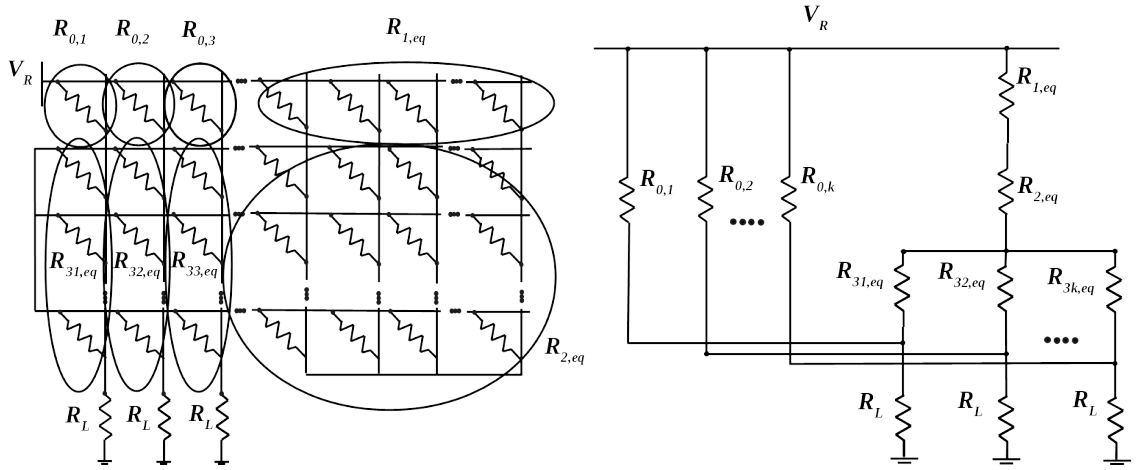

Fig. 2: (a)Multiple column sampling for shorted unselected row columns architecture of crossbar RRAM (b)Equivalent circuit of (a)

From the properties of random variables we get,

$$Pr(X > Y) = \int_{-\infty}^{+\infty} \int_{y=-\infty}^{x} f_X(x) f_Y(y) dx dy. \tag{12}$$

$$Pr(\tau_i = 1) = Pr(X > Y) = \int_{-\infty}^{+\infty} \int_{y=-\infty}^{x} f_X(x) f_Y(y) dx dy. \tag{13}$$

Now we find f_X and f_Y to calculate $Pr(X > Y)$ using Eq. 13.

A. Finding f_X

Random variable, X ranges from 0 to $\frac{(n-k)(m-1)}{m}$. However, the upper limit of X can be approximated as $n-k$ as $\frac{m-1}{m}$ approaches 1 considering a moderately large m. Now, $f_X(p-1 \leq X < p)$ represents the probability of X having a value between $p-1$ and p, where p is an integer between 1 and $n-k$. The condition $p-1 \leq X < p$ reduces to,

$$\frac{N_1 * (p-1)}{N1 - (p-1)} \leq N_2 < \frac{N_1 * p}{N_1 - p}. \tag{14}$$

The lower limit of N_2 in Eq. 14 must be less than or equal to the maximum value of N_2, i.e.

$$\frac{N_1 * (p-1)}{N1 - (p-1)} \leq N_{2,max}. \tag{15}$$

Eq. 15 reduces to,

$$N_1 \geq \frac{N_{2,max} * (p-1)}{N_{2,max} - (p-1)}, \tag{16}$$

and eventually,

$$N_1 \geq \frac{(m-1) * (n-k) * (p-1)}{(m-1) * (n-k) - (p-1)}. \tag{17}$$

Now, $f_X(p-1 \leq X < p)$ can be calculated as follows:

$$f_X(p-1 \leq X < p) = \sum_{i=r}^{n-k} Pr(N_1 = i) \sum_{u_{1,i}}^{u_{2,i}} Pr(N_2 = j), \tag{18}$$

where r is the minimum value of N_1 found from Eq. 17.

$$r = ceil(\frac{(n-k) * (m-1) * (p-1)}{(n-k) * (m-1) - (p-1)}), \tag{19}$$

$u_{1,i}$ and $u_{2,i}$ can be found from the ranges of N_2 given in Eq. 14.

$$u_{1,i} = ceil(\frac{i * (p-1)}{i - p + 1}) \tag{20}$$

$$u_{2,i} = \begin{cases} floor(\frac{i*p}{i-p}), \text{ if } floor(\frac{i*p}{i-p}) \leq (m-1) * (n-k) \\ (m-1) * (n-k), \text{ otherwise} \end{cases} \tag{21}$$

B. Finding f_Y

We recall that Y is a random variable which is a function of random variables $N_{3,1}$, $N_{3,2}$,..., $N_{3,k}$.

$$Y = (\frac{1}{\frac{1}{N_{3,i}} + \frac{1}{N_L}} + \sum_{j=1, j \neq i}^{k} \frac{1}{\frac{1}{N_{3,j}} + \frac{1}{N_L}}).(\frac{N_{3,i} + N_L}{N_{3,i} - N_L}) \tag{22}$$

Let $Y = (Y_1 + Y_2).Y_3$ where Y_1, Y_2 and Y_3 are individual random variables having probability distributions of f_{Y_1}, f_{Y_2} and f_{Y_3}, respectively. Here, Y_2 and Y_3 are functions of the same random variables $N_{3,i}$ which have binomial probability distribution functions of f_N representing the probability of a number of cells being ON in the i^{th} sampled column in the crossbar. The set definition of $N_{3,i}$ has been provided in Eq. 10 from which we can obtain f_N.

$$f_N(x) = \frac{\binom{m-1}{x}}{2^{m-1}} \tag{23}$$

From Eq. 22, Y_2 is a sum of Y_1 over (k-1) times. According to the central limit theorem, when n number of independent and identically distributed random variables drawn from a distribution having a mean, μ and variance, σ^2, are added together, the resulting random variables will follow a normal distribution with mean, $n\mu$ and variance $n\sigma^2$.

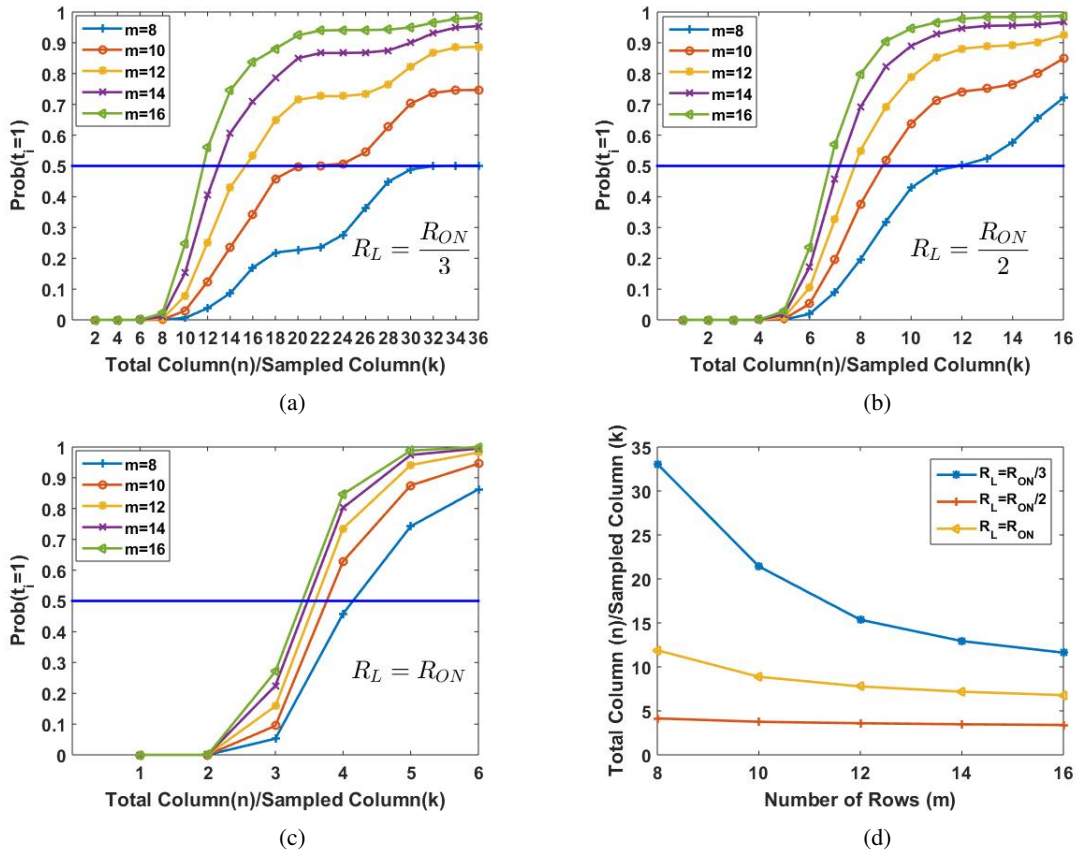

Fig. 3: (a)-(c):Probability of each tag bit being high with respect to the ratio of total columns and sampled columns for different row size of crossbar. Load resistance used in different plots (a) $\frac{R_{ON}}{3}$ (b) $\frac{R_{ON}}{2}$ (c) R_{ON}.(d): ratio of total columns and sampled columns with respect to number of crossbar rows required for achieving 0.5 probability of each tag bit being high

Let, $\mu = \text{mean}(f_{Y_1})$, $\sigma^2 = \text{variance}(f_{Y_1})$. According to the central limit theorem:

$$f_{Y_2}(x) = \frac{1}{\sqrt{(k-1)\mu}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right). \tag{24}$$

Now, using algebra of random variables we can obtain the P.D.F of random variables, Y from random variables Y_1, Y_2 and Y_3.

$$f_Y(z) = \int_{x=1}^{m-1} f_N(x) f_{Y_2}\left(\frac{z}{Y_3(x)} - Y_1(x)\right) \frac{1}{Y_3(x)} dY_1(x) \tag{25}$$

Since we have f_X and f_Y we can find the probability of each tag bit, $\tau_i = 1(= 0)$ using Eq. 13. Using the relation expressed in Eq. 13, we evaluate the probability of a tag bit being 1 (or 0) for different crossbar sizes and load resistance values which are illustrated in Fig. 3.

VII. Security based Design Choice

According to the analytical model of the proposed tag generation method, the probability of a tag bit being 1 (or 0) depends on the number of rows in the crossbar, ratio of number of total columns to sampled columns, and ratio of load resistance to ON resistance value of the crossbar. From the plots shown in Fig. 3, it can be seen that for a fixed load resistance the probability of a tag bit being 1 (or 0) increases with both the number of crossbar rows and ratio of total columns and sampled columns. Again, for a fixed number of row and fixed total column to sampled column ratio, the probability of a tag bit being 1 (or 0) increases with the load resistance. Another observation is that for a higher load resistance the curves are steeper, implying higher sensitivity to the ratio of total column and sampled column. Optimal design choices from the perspective of security are combinations of these parameters leading to a probability of 0.5 for each tag bit being 1 (or 0). We discuss here that this property is supported by a few additional assumptions and architectural considerations and ensures the desired security properties described earlier for the proposed integrity checking system of RRAM.

A. Uniformity

This property can be achieved if the generated tag value has a uniform probability distribution. For a uniform probability distribution of a k- bit tag, the probability of each tag:

$$Pr(\tau) = \frac{1}{2^k}. \tag{26}$$

In cases where the probability of each tag bit, τ_i being 1 (or 0) is independent of each other, the target value of τ_i is $\frac{1}{2}$ for

obtaining an uniform tag distribution. From Eq. 22, random variable Y for each sampled column has an independent term comprised of the number of ON cells in each sampled column, $N_{3,i}$. Therefore, the probability of each tag bit being 1 (or 0) expressed by the Eq. 13 is independent of each other except for some limiting size of crossbars for which the term $N_{3,i}$ in Eq. 22 becomes negligible as compared to N_L.

From the plots of Fig. 3, we can achieve this target value with different sets of combinations among the three design parameters: number of crossbar rows, ratio of total columns to sampled columns, ratio of load resistance to ON resistance. For a better demonstration of design choices we plot the ratio of total number of columns and number of sampled columns with respect to number of crossbar rows for different load resistance to ON resistance ratio corresponding to the 0.5 lines in each plot of Fig. 3.

B. Diffusion

Diffusion is the measure of sensitivity of each bit in the data to every bit in the generated tag. As we described earlier, for a secure integrity checking system every bit in the tag should be flipped with a probability of $\frac{1}{2}$ due to a 1-bit flip in the data.

As we described earlier, our tag generation architecture is such that during every new write operation to the memory, a new set of k columns is chosen randomly and used for sampling in tag generation phase. In addition, all unselected cells in the reserved row are reconfigured with random bit in every new write operation. These added randomness scrambles the data and essentially makes it random to the tag generator even if only 1 bit of the data is being flipped by an user.

For corner cases where the proportion of 0's and 1's in the memory data is extremely unbalanced, e.g. all 0's or all 1's, choosing a random set of k columns for a new tag generation corresponding to a small change in data contributes negligibly to the tag change. In this case, the only randomness that contributes to the tag change is that in the reserved row and may result in a deviation in the diffusion results from the general cases. However, this can be mitigated by adding more than one reserved row to strengthen the effect of randomness.

For general cases, let the tag generated from a random data be τ and it changes to τ' due to 1-bit flip in the data. $\Delta\tau$ is the binary distance between τ and τ'. Now, the probability of any tag bit being flipped:

$$Pr(\Delta\tau_i = 1) = Pr(\tau_i = 1)Pr(\tau_i' = 0) + Pr(\tau_i = 0)Pr(\tau_i' = 1) \tag{27}$$

By choosing appropriate design parameters suggested in Fig. 3 (d), we can achieve $Pr(\tau_i = 1) = Pr(\tau_i = 0) = \frac{1}{2}$ as we mentioned earlier. Due to the randomness in tag generation, the data becomes a random one even only a bit is flipped. As a result $Pr(\tau_i' = 1) = Pr(\tau_i' = 0) = \frac{1}{2}$. Plugging these values in Eq. 27 we get $Pr(\Delta\tau_i = 1) = \frac{1}{2}$.

C. Avalanche Effect

Avalanche effect refers to the property of the output being changed significantly due to a small change in the input. Quantitatively, on average, half of the total bits in the tag should flip due to a single bit flip in the data. In our tag generation method, we define the avalanche coefficient, σ_h as a measure of avalanche effect. Avalanche coefficient is calculated by estimating the expected value of hamming distance (HD) between a tag generated from a random data and the tag generated by flipping 1-bit of the data.

$$\text{HD} = \sum_{i=1}^{k} \Delta\tau_i \tag{28}$$

From the analysis of diffusion property in the sneak path based tag generation, we can achieve the property of each bit in the tag being flipped independently with a probability of $\frac{1}{2}$ due to only a single bit flipping in the input. However, corner cases affecting the diffusion property also impact the avalanche effect. For general cases, let f_h is the probability distribution function of hamming distance between two tags generated from two data separated by a hamming distance of 1. For a k-bit tag:

$$f_h(HD = x) = \frac{\binom{k}{x}}{2^k} \tag{29}$$

The expected value of HD is the desired value of avalanche effect.

$$E(x) = \sum_{x=0}^{k} x.f_h(x) = \frac{k}{2}; \tag{30}$$

VIII. Circuit Design and Operation

The tag generation method proposed in this paper utilizes sneak path enabled in-memory computing feature of RRAM. The same crossbar structure used for the memory can also be used to generate tags for integrity checking purposes with minimal additional control circuitry.

A. Design

We consider a 1T1R structure for the RRAM as shown in Fig. 4 which can mitigate the sneak path leakage problem associated with regular read and write operation of the memory. The memory has row and column control blocks for regular read-write operations. For tag generation purposes, separate row and column control units are multiplexed with regular ones. A trigger signal is used as the selector of the multiplexer which makes the memory switch between the regular read-write and tag generation phase. We already described that the row where read voltage is applied for tag generation is not a part of regular memory operation. Rather, it is reserved only for tag generation purposes. In the row control circuit for tag generation, the reserved row is connected with a read/write controller. All other rows are tied together to a common node.

Column control circuit has a sampling decoder unit which is used to reconfigure the sampled and unsampled columns for a tag generation instant. An analog de-multiplexer is used with every column going into tag generation column control unit. One output of each de-multiplexer is connected to a common node to implement the shorted unselected column structure as described in Section V. A read/write control unit is connected

with the other output of each de-multiplexer. A sampling decoder feed by a pseudo random number generator (PRNG) randomly selects a set of columns for sampling. PRNG is enabled by a voltage pulse derived from the regular write signal of the memory. Thus, the PRNG generates a new value during every new write operation and hold it until the next write. In read mode, each read/write control unit connects each column to a load which can be implemented using polysilicon resistor having low deviation in the resistance level [31]. Voltage across each load resistor is connected to an analog comparator and the output of each comparator is feed into a flip flop. Finally a $n \times k$ selector configured by the PRNG output selects k tag bits out of n flip flop outputs. A transistor control unit is used to control the transistors of every 1T1R memory cell in both regular read/write and tag generation mode of the RRAM.

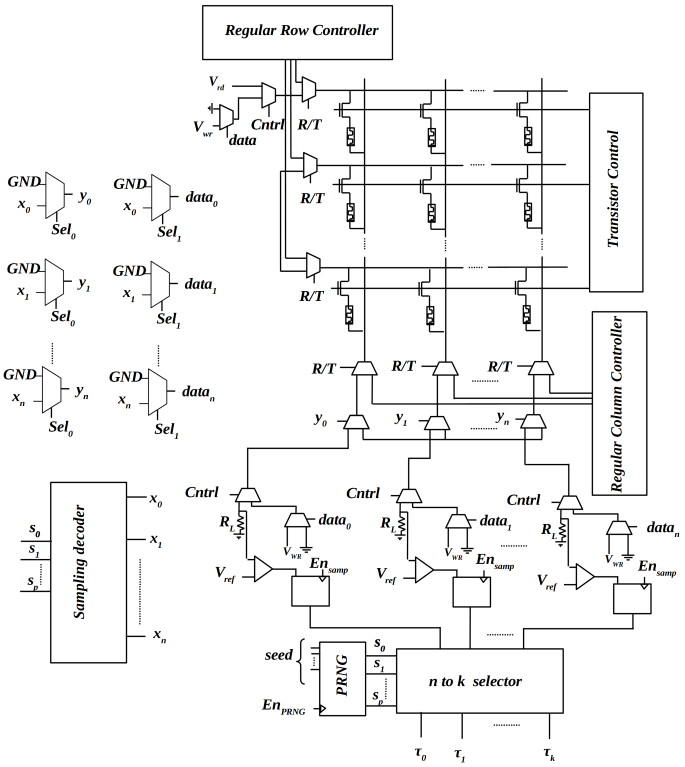

Fig. 4: Crossbar RRAM with peripheral control circuitry for tag generation and integrity checking

B. Operation

The tag generation phase of the proposed integrity checking method is divided into two steps.

The first step is configuring phase where the PRNG generates a random number and the sampling decoder decodes sampled and unsampled columns accordingly. In this step, bits in the reserved row corresponding to sampled columns are written with all 0's and those corresponding to unsampled columns are written with random 0's and 1's. This task is accomplished by a two phase write method used to write a whole row of resistive crossbar memory [32]. In the first phase of a two phase write method, the whole crossbar row is written with 0's. In the second step, only desired locations are written with 1's. A similar two phase write method is applied to write the reserved row which also leverages stochasticity in memristor switching [33], [34]. As we consider a binary memristor for the application proposed in this work, switching probability of the memristor depends on the write pulse width provided that the device exhibits stochasticity in its switching behavior. There is a particular range of write time where the memristor switches between the ON and OFF state stochastically with a probability of around 50%. In the first phase of reserved row write, all RRAM cells in the reserved row are reset with a regular write pulse ensuring 100% switching. Regular write pulse refers to the write voltage pulse having a sufficiently large pulse width which causes switching of a stochastic memristor with a probability of 1 . After that, a set pulse with pulse width in the stochastic region is applied to only the cells in the unsampled columns of reserved row. In the stochastic region of switching, a write pulse width can be found for which a stochastic binary memristor can be switched with a 50% chance. In this way, a random number is written into the reserved row of the crossbar RRAM in a much simpler way than the regular PRNG based write method. However, not all RRAM devices exhibit a stochastic switching characteristic. This stochastic switching based reserved row write method will only apply to the RRAM that has stochasticity in the switching characteristics. Otherwise, we can always use the traditional PRNG based write method as described in [12].

The second phase of tag generation is to read the memory through the sampled columns while all sneak paths in the RRAM are enabled. To generate the tag, the analog voltage across the load resistor of each column is converted to a binary bit using a comparator with a certain reference voltage. A sampling pulse stores each bit to the corresponding flip flop. Among n flip flop outputs the $n \times k$ selector selects only those outputs corresponding to the columns decoded by sampling decoder. This set of selected output is the generated tag from the RRAM data. Unlike the first phase, this phase of tag generation is triggered by an external signal. This signal comes from the processor interacting with the memory. Whenever the processor writes something to the memory, it also sends a tag generation signal to generate a tag from it and stores the generated tag for integrity checking. After the write operation, whenever the processor reads the memory, it checks the integrity of the memory data from any unauthorized modification. For verifying integrity, the processor sends the tag generation signal to the memory and regenerates the tag. If the regenerated tag matches the previously saved one, the data read from the memory can be treated as safe and unaltered. Otherwise, the integrity of data is violated and regarded as unsafe for using in the applications of computer dependent on it.

IX. Simulation Tests and Results

For empirical evaluation of the security properties, the proposed design is simulated in MATLAB using a model for the resistive crossbar structure described in [12]. We have

added temperature variability to the model for investigating the impact of temperature variation on the security properties of generated tags. We specifically consider a temperature coefficient of $0.004/^oC/$ and $-0.008/^oC$ for LRS and HRS variation of the memory cell, respectively [35]. The nominal value considered for the HRS and LRS of memristor is $57\,\mathrm{M}\Omega$ and $58\,\mathrm{k}\Omega$, respectively [36]. The read voltage ensuring a nondestructive read operation considered in the simulation is $600\,\mathrm{mV}$. To account for process variation, LRS and HRS are modeled with 10% and 20% variations, respectively. Security properties are tested using some algorithms developed based on the definition of these properties.

A. Uniformity Test

Uniformity is measured using a metric that refers to how balanced the probability distribution of tag generated from randomly distributed data. Tags are generated from a number (k) of randomly selected data combination. This set of tags are divided into a number (t) of equally spaced bins, b_1, b_2,.., b_t. Number of tags fallen into each bins are nb_1, nb_2,.., nb_t, respectively. In our experiment, we choose a k of 1000 and t of 256. Uniformity can be measured using the following equation developed using the balance metric described in [37].

$$uniformity, \alpha = log_t\Big[\frac{k^2}{{nb_1}^2 + {nb_2}^2 + ... + {nb_t}^2}\Big] \quad (31)$$

From the above equation, uniformity will be 1 when the distribution of tag is fully balanced, i.e. $nb_1 = nb_2 = ... = nb_t = \frac{k}{t}$. Based on the tag distribution found from our simulation results we can calculate the uniformity of the proposed design for integrity checking using Eq. 31.

B. Avalanche Test

For measuring avalanche effect, pairs of tag are generated from random data and one of its variant with a randomly selected bit flipped. A number (k) of such pairs are generated using simulation. Hamming distances are measured between tags in each pairs as a percentage of the total number of tag bits. Let, H_1, H_2,..., H_k are measured hamming distances between tag generated in each pair. The mean of resulted hamming distance distribution gives the measure of avalanche effect of our proposed tag generation method. This measure of avalanche effect is termed as avalanche coefficient (Av) for further reference in this paper.

$$Av = \frac{\sum_{i=1}^{k} H_i}{K} \quad (32)$$

C. Diffusion Test

For measuring diffusion we use the same measurements taken for avalanche test. However, unlike the avalanche test, in this test, our goal is to measure the likeliness of each bit being flipped. We calculate the bit distance of each tag bit in every pairs of generated tags. Diffusion is a property which evaluates each bit individually. Let, d_1, d_2,..., d_k are average bit distance of each bit across all tag pairs. If each of the bit individually flips with a probability of 50%, proposed tag generation method would then meet the diffusion properties also known as strict avalanche criteria. However, to quantify the property using a single numerical value we take the geometric mean for probability of bit flip across all bits in the tag. This quantification is termed as the diffusion coefficient ($diff$).

$$diff = \sqrt[k]{d_1.d_2....d_k} \quad (33)$$

TABLE I: Security results for optimal crossbar sizes with different values of load resistances. Tag size (number of sampled columns) considered in this analysis is 8 bits

R_{LOAD}	Crossbar Size	Uniformity	Avalanche	Diffusion
$\frac{R_{ON}}{3}$	8×264	0.9596	0.5035	0.4870
	10×176	0.9561	0.5011	0.4930
	12×120	0.9602	0.4983	0.5080
	14×104	0.9612	0.4946	0.4860
	16×92	0.9056	0.4948	0.4930
$\frac{R_{ON}}{2}$	8×96	0.9619	0.4983	0.4833
	10×72	0.9583	0.4913	0.4860
	12×60	0.8931	0.4853	0.4900
	14×58	0.7395	0.4945	0.4810
	16×55	0.6264	0.4989	0.4860
R_{ON}	8×34	0.7150	0.5009	0.4760
	10×32	0.5821	0.4951	0.4540
	12×30	0.4751	0.5052	0.4690
	14×29	0.3507	0.5145	0.4610
	16×28	0.3241	0.4931	0.4720

D. Results

From our analytical model of the proposed tag generation method from crossbar RRAM, we find the optimal crossbar size that leads to desired security properties. Fig. 3 shows optimal parameters for RRAM design that meets the security goals for integrity checking. We choose RRAM with those set of parameters for simulating the proposed tag generation method in order to generate security results.

An interesting observation we found from the analytical model of the proposed tag generation method is that security properties of this proposed system depends on the ratio of the number of total columns to number of sampled columns. As the voltage across each sampled column translates into a tag bit, the total number of sampled columns is essentially the tag size of the proposed integrity checking system. Due to the dependency of the security properties on relative size of crossbar columns with respect to tag size, optimal points shown in Fig. 3 are valid for any tag size. For the selection of crossbar memory sizes, we first choose a particular tag size. After that, we choose a particular crossbar row size and find the corresponding ratio of number of total column to sampled columns. Finally, we find the column size by multiplying the ratio with chosen tag size. Table I shows security results for the proposed tag generation method in room temperature (25^oC) for all of the optimal points suggested in Fig. 3(d) with a tag size (number of sampled columns) of 8. The same results also apply to larger tag size, where we have to scale up the number of total columns in the crossbar accordingly.

Results demonstrated in Table I indicate that avalanche effect and diffusion for our proposed tag generation method are close to their ideal values, 0.50 for all of the optimal points suggested in Fig. 3(d). Uniformity is also close to its ideal

TABLE II: Impact of temperature variation on the security properties of proposed tag generation method

Crossbar Size	Uniformity		Avalanche		Diffusion	
	0^0C	100^0C	0^0C	100^0C	0^0C	100^0C
8×264	0.9562	0.9584	0.4959	0.4988	0.5159	0.4854
10×176	0.9590	0.9681	0.4905	0.5008	0.4994	0.4904
12×120	0.9581	0.9583	0.4984	0.4946	0.4944	0.4931
14×104	0.9577	0.9584	0.4890	0.4996	0.4997	0.4873
16×92	0.8911	0.8963	0.4999	0.5012	0.4994	0.4899

TABLE III: Security results for optimal crossbar sizes with different values of load resistances. Tag size (number of sampled columns) considered in this analysis is 8 bits

Tag size	Crossbar Size	Uniformity	Avalanche	Diffusion
8	8×96	0.95	0.49	0.4850
16	8×192	0.95	0.49	0.48
32	8×384	0.96	0.51	0.475
64	8×768	0.96	0.48	0.50

value, 1 for a design having less number of rows in the crossbar. However, when the number of rows increase uniformity deviates from its ideal value. The rate at which uniformity decreases with the increase of crossbar rows increases for a higher R_L. For a load resistance of $\frac{R_{ON}}{3}$, the uniformity value is nearly ideal up to a row size of 14.

The analytical model for the proposed tag generation method can explain such deviation of security results from their optimal values. Every optimal point of Fig. 3(d) results in probability of 0.5 for each tag bit being high or low. However, as we already described in Section VI that for having a uniform probability distribution of generated tag, each tag bit being high (or low) must be 0.5 and independent of each other. The right hand side of the inequality in Eq. 11 consists of the generic term $N_{3,i}$ and N_L where, $N_{3,i}$ represents number of ON memory cells in i^{th} sampled column and N_L represents the ratio of of ON resistance (R_{ON}) and load resistance (R_L). The term $N_{3,i}$ is the source of independence of Eq. 11 across each sampled column. When the number of rows in the crossbar increases, the number of cells in a column increases too. As a result, number of ON memory cells in a sampled column also becomes higher on an average. At a particular row size the term $\frac{1}{N_{3,i}}$ becomes negligible compared to $\frac{1}{N_L}$ and hence Eq. 11 lacks independence across all sampled columns. In this case, in spite of having 0.5 probability for each tag bit being high or low, the overall tag is reduced to only a few combination and results in low uniformity. This condition can happen similarly when the ON resistance to load resistance ratio N_L decreases.

In order to investigate the impact of temperature variation we also evaluated the security properties for tag generation considering two temperatures 0^oC and 100^oC with $R_{LOAD} = \frac{R_{ON}}{3}$. The security results with temperature variation is shown in Table II. It can easily be seen that temperature variation has almost no impact on the security results within the considered range of variation.

In Table I and Table II, we demonstrated results for a tag size of 8 bits. Since the security results depend on the number of sampled columns relative to the total number of columns, the same results also apply to higher tag size with number of crossbar columns scaled up proportionately. We demonstrate scalability results in Table III for one of the optimal design configurations suggested in Fig. 3(d) by keeping the ratio of total column to sampled column constant while increasing the absolute size of sampled columns as well as total columns. For the results in Table III, We chose a crossbar with 8 rows, total columns $12\times$ of the sampled columns and a load resistance of $\frac{R_{ON}}{2}$. We select 4 different sampled column sizes (i.e. tag size) $8, 16, 32, 64$ for demonstrating that security results depend on the ratio of total and sampled column rather than depending on the absolute column size.

Simulation results presented in Table I validates the analytical model quite accurately. Design choices suggested by the analytical model which is shown in Fig. 3 yields a desired level of security for most cases. There are some cases where the security results deviate from the values suggested by the analytical model. These can also be explained based on the assumptions considered in the model. A designer can therefore make design choices guided by the analytical model with a sufficient level of confidence.

X. Variability Sources and Reliability Analysis

Reliability is one of the major characteristics of any usable system design. For the proposed integrity checking system, reliability refers to the ability of generating the same tag on every read operation between two consecutive write operations in the memory. It is desirable that after memory is written, generated tag be always same until the next write is performed. However, variability is a major concern for memristor based RRAM designs. A number of different sources contribute to the variability in RRAM based system design. For the proposed tag generation system we consider memristor's cycle to cycle variation, supply voltage fluctuation, temperature variation and load resistance variation as the sources of variation. Memristors also suffer from aging induced resistance drift over time. However this is a slow process and does not affect the reliability of tag over multiple reads between consecutive write operations.

A. Variability Sources

Memristor devices exhibit variations in their parameters across different set/reset cycles. Cycle to cycle variation potentially can be a reliability concern for any RRAM based system. However, one notable feature of the proposed tag generation method is that the proposed method only involves read operation. Usually a read operation is performed by applying a low voltage which prevents change in the resistance level of the memory cell. Therefore, an RRAM based system exhibits little variation in terms of memory cell resistance levels from one read cycle to another as compared to the variations between one write cycle to another.

Like most of the electronic devices, memristor is also temperature dependent. Many of its device parameters are temperature dependent. For example, OFF state resistance of a HfOx memristive devices increases and both set and reset voltages decreases with the increase of temperature [38]. Due to temperature variation, generated tag from the RRAM

TABLE IV: Reliability results of the proposed tag generation method for three different crossbar sizes.

R_{LOAD}	Crossbar Size	Reliability (%)
$R_{ON}/3$	12×120	83.11%
$R_{ON}/2$	8×96	91.44%
R_{ON}	8×34	83.11%

can be different during tag verification phase at every read. Therefore, temperature fluctuation can be one of the deterrents for reliability of the proposed tag generation method.

Another variation we consider in reliability analysis of the proposed design is fluctuations in read voltage applied for tag generation. All designs of voltage supply have some noises associated with it that cause the voltage level to fluctuate randomly around its nominal value According to the relationship between the load voltage and the read voltage given in Eq. 2, load voltage is directly proportional to the read voltage. Generated tag is affected therefore by the fluctuation in applied read voltage level.

Another source of variation in the proposed tag generation is the load resistance of each sampled column. From our analytical model of the proposed tag generation from RRAM, load resistance is one the crucial design parameters. It can be seen in Fig. 3 that a crossbar of particular size exhibits a huge difference in the distribution of the tag value for 3 different ratios between the load resistance and ON resistance value considered in the analysis. Due to variability in the load resistance value from its nominal value, output voltage across the load resistor also varies accordingly and becomes a potential factor for affecting the reliability of tag generation.

B. Reliability Test and Results

To investigate the reliability of the proposed tag generation method from RRAM, we simulated the proposed design of tag generation using Cadence Spectre simulator. In this reliability test experiment the memory is written with a random data and tag is generated for a number of times. For each of these tag generation instant, simulator chooses a random value for every source of variation within a range of their respective nominal values. We consider a cycle to cycle variation of 2%, a temperature range of $0°\mathrm{C}$ to $100°\mathrm{C}$, and 5% variation for load resistance of each sampled column [35]. 20 mV variation is considered for the read voltage used in the tag generation method [39]. The tag that is generated immediately after the write operation is the reference tag for integrity checking. Every read operation after this write and before the next write requires regeneration of the tag and verification whether regenerated tag matches with the reference tag. Reliability is evaluated as the percentage of tag regeneration during memory read that matches with the reference tag. Overall reliability is found by averaging over the reliability results for tags generated from different random data considered in the experiment. Table IV demonstrates reliability results for three different crossbar having the best security results corresponding to each of the load resistance value considered in Fig. 3. Results indicate that an optimal design from the perspective of security with a load resistance of half of the ON resistance is more reliable than designs with other two choices of load resistance.

Investigating the proposed tag generation architecture, it can be inferred that memory states resulting in load voltages around the threshold voltage of the comparator are less reliable for tag generation. These states are more susceptible to the variation in different parameters involved in the tag generation process. However, if these states are identified correctly, reliability of the system can be improved by applying standard error correction schemes used in other RRAM based systems, such as those considered for physically unclonable functions [40].

Detailed reliability model portraying the relationship between reliability value and design parameters is left as a future work. As another possible future aspect of this work, the analytical model can be modified to include variability of design parameters and thus can help choosing optimal designs from the perspective of both security and reliability.

XI. Overhead Analysis

In most of the existing works on memory integrity checking, tag generation is performed using computationally intensive cryptographic operations. On the other hand, the proposed method use a single memory read operation for tag generation and hence is expected to incur significant amount of resource savings.

For comparing the implementation cost of the proposed tag generation method with an existing one, we choose the tag generation method proposed by Hong. *et al.* as comparable to ours as it is also a lightweight one compared to more conventional methods [10]. There is another related work which addressed some of the security concerns of Hong. *et. al.*'s design and proposed some addition to the design for improving the security [11]. However, the security improvement comes with a cost of overhead. If the tag generation method proposed in this paper shows improvement over Hong *et. al.*'s design, the improvement would be even more compared to Liu *et. al.*'s method. Since this work is built on the basic idea of tag generation leveraging sneak path currents proposed in [12], overhead incurred by both systems are comparable. However, based on design considerations outlined in this paper, we are able to design a system which exhibits security properties close to the ideal value. On the other hand, the earlier work lacks this and requires 25% more tag bits to reach a similar level of security [12].

For the purpose of comparison, we implement a prototype for both our design and Hong *et al.*'s design with same configurations and technology node. We choose 64 bit data and 8 bit tag system for tag generation using both method and 65 nm CMOS technology is used for the implementation.

Prototype implementation of tag generation method proposed by Hong. *et. al.* includes random shuffling among data blocks and bit rotation by a random number. Data is divided into blocks of a bit length equals to the tag size. Final step of this tag generation method is the bit wise XOR among the output of each data block after shuffling and rotation. Prototype implementation of the tag generation method proposed

TABLE V: Comparison of implementation cost among different existing tag generation techniques

Overhead	Hong *et al.*	Yan *et al.*	Roger *et al.*	Proposed
Energy (pJ)	96	408	455	9.75
Delay (ns)	50	104	138	20
Transistor count	9358	15340	15340	3456

in this paper generates a tag of 8 bits from 64 bit data. The crossbar RRAM considered for this implementation consists of 5 rows and 16 columns. Due to the reserved row the actual data row is 4 and the number of total memory cells is 64. The tag generation circuit is implemented using the circuit described in Fig. 4.

Implementation cost for both designs are presented in Table V. Cost is evaluated from the perspective of average energy consumption in a tag generation cycle, transistor counts required for the circuit implementation and delay for completing tag generation. Comparisons with two other existing methods proposed by Yan *et al.* [8] and Roger *et al.* [9] are also presented in Table V evaluated based on the overhead comparison given in [10]. According to the Table V, the design proposed in this paper shows significant improvement over its comparable designs considered here. The improvement is almost $10\times$ for energy and $2.5\times$ for both transistor count and delay, over the design proposed by Hong *et al.* which exhibits least overhead among the three comparables. This improvement of our design in the implementation cost is achieved due to a number of factors. First, in our method, the main processing of data leading to tag generation requires a simple read operation whereas a number of shuffling, rotation and XOR operation is involved in the other method. Though some processing is also required in our design to facilitate the column sampling, it is only performed on the columns. In contrast, all operation is performed on each data in the other method and therefore requires a large amount of hardwares compared to our method. The improvement in delay is also noticeable in our design. The required time to generate a tag is simply the time required for a read operation and propagation delay of the sampling circuit. It must be noted that for the proposed tag generation method in this paper, an additional write is performed to the reserved row during every regular write operation. As we described earlier, this write is performed using a two phase write method and thus requires a delay equal to twice the write time for a RRAM memory. However, this extra overhead in delay is required only during the write operation to memory. For read operation from memory no new tag is generated and thus does not require the step of reserved row writing. Therefore, the tag generation method proposed in this paper still shows a significant improvement for the average case delay of tag generation.

Other comparables used in this comparison are implemented using well established CMOS technology and therefore are highly reliable in their operation. On the other hand, the proposed method is to be implemented for RRAM based memory and involves nanoelectronic devices, memristors, where reliability is always a concern. For the proposed system, reliability is at best 91% among the three test cases considered. Again, the proposed system is more efficient than other comparable methods in terms of implementation. Therefore, future work on reliability enhancement methods for the proposed system should be considered.

XII. Conclusion

Main contribution of this work includes the development of an analytical model and formulating design considerations for the sneak path current based tag generation mechanism inspired by an earlier work. We investigated the security properties of the system using circuit level simulation of the design and verified the analytical model. We also measured the reliability of the proposed method considering different possible sources of variations.

Acknowledgment

The authors would like to thank Dr. Jeyavijayan Rajendran of Texas A&M University for interesting discussions related to this topic.

References

[1] A. Makarov, V. Sverdlov, and S. Selberherr, "Emerging memory technologies: Trends, challenges, and modeling methods," *Microelectronics Reliability*, vol. 52, no. 4, pp. 628–634, 2012.

[2] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[3] R. C. Merkle, "Protocols for public key cryptosystems." in *IEEE Symposium on Security and privacy*, vol. 122, 1980.

[4] B. Gassend, G. E. Suh, D. Clarke, M. Van Dijk, and S. Devadas, "Caches and hash trees for efficient memory integrity verification," in *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*. IEEE, 2003, pp. 295–306.

[5] M. Haifeng, Chengjie, and G. Zhenguo, "Memory integrity protection method based on asymmetric hash tree," in *Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on*, Nov 2014, pp. 263–267.

[6] C. Lu, T. Zhang, W. Shi, and H.-H. S. Lee, "M-tree: a high efficiency security architecture for protecting integrity and privacy of software," *Journal of Parallel and Distributed Computing*, vol. 66, no. 9, pp. 1116–1128, 2006.

[7] R. Elbaz, L. Torres, G. Sassatelli, P. Guillemin, M. Bardouillet, and A. Martinez, "Transactions on computational science x," M. L. Gavrilova, C. J. K. Tan, and E. D. Moreno, Eds. Berlin, Heidelberg: Springer-Verlag, 2010, ch. Block-level Added Redundancy Explicit Authentication for Parallelized Encryption and Integrity Checking of Processor-memory Transactions, pp. 231–260. [Online]. Available: http://dl.acm.org/citation.cfm?id=1985581.1985591

[8] C. Yan, D. Englender, M. Prvulovic, B. Rogers, and Y. Solihin, "Improving cost, performance, and security of memory encryption and authentication," in *ACM SIGARCH Computer Architecture News*, vol. 34, no. 2. IEEE Computer Society, 2006, pp. 179–190.

[9] A. Rogers and A. Milenković, "Security extensions for integrity and confidentiality in embedded processors," *Microprocessors and Microsystems*, vol. 33, no. 5, pp. 398–414, 2009.

[10] M. Hong, H. Guo, and S. X. Hu, "A cost-effective tag design for memory data authentication in embedded systems," in *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems*. ACM, 2012, pp. 17–26.

[11] T. Liu, H. Guo, S. Parameswaran, and X. S. Hu, "Improving tag generation for memory data authentication in embedded processor systems," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2016, pp. 50–55.

[12] M. B. Majumder, M. Uddin, G. S. Rose, and J. Rajendran, "Sneak path enabled authentication for memristive crossbar memories," in *Hardware-Oriented Security and Trust (AsianHOST), IEEE Asian*. IEEE, 2016, pp. 1–6.

[13] S. Kannan, N. Karimi, O. Sinanoglu, and R. Karri, "Security vulnerabilities of emerging nonvolatile main memories and countermeasures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 1, pp. 2–15, 2015.

[14] M. T. Arafin and G. Qu, "Rram based lightweight user authentication," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2015, pp. 139–145.

[15] J. Gibbons and W. Beadle, "Switching properties of thin nio films," *Solid-State Electronics*, vol. 7, no. 11, pp. 785–790, 1964.

[16] J. Simmons, "Conduction in thin dielectric films," *Journal of Physics D: Applied Physics*, vol. 4, no. 5, p. 613, 1971.

[17] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[18] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, p. 80, 2008.

[19] S. Amer, S. Sayyaparaju, G. S. Rose, K. Beckmann, and N. C. Cady, "A practical hafnium-oxide memristor model suitable for circuit design and simulation," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*.

[20] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2013.

[21] H. Manem, J. Rajendran, and G. S. Rose, "Design considerations for multilevel cmos/nano memristive memory," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 8, no. 1, p. 6, 2012.

[22] M.-J. Lee, Y. Park, B.-S. Kang, S.-E. Ahn, C. Lee, K. Kim, W. Xianyu, G. Stefanovich, J.-H. Lee, S.-J. Chung *et al.*, "2-stack 1d-1r cross-point structure with oxide diodes as switch elements for high density resistance ram applications," in *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*. IEEE, 2007, pp. 771–774.

[23] C. Cowan, F. Wagle, C. Pu, S. Beattie, and J. Walpole, "Buffer overflows: Attacks and defenses for the vulnerability of the decade," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2. IEEE, 2000, pp. 119–129.

[24] T. Mller and F. C. Freiling, "A systematic assessment of the security of full disk encryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 491–503, Sept 2015.

[25] X. Ruan, *Platform Embedded Security Technology Revealed: Safeguarding the Future of Computing with Intel Embedded Security and Management Engine*, 1st ed. Berkely, CA, USA: Apress, 2014.

[26] G. Leurent, "Md4 is not one-way," in *International Workshop on Fast Software Encryption*. Springer, 2008, pp. 412–428.

[27] A. Abidi, B. Bouallegue, and F. Kahri, "Implementation of elliptic curve digital signature algorithm (ecdsa)," in *Computer & Information Technology (GSCIT), 2014 Global Summit on*. IEEE, 2014, pp. 1–6.

[28] H. Feistel, "Cryptography and computer privacy," *Scientific american*, vol. 228, no. 5, pp. 15–23, 1973.

[29] D. Xiao, X. Liao, and S. Deng, "One-way hash function construction based on the chaotic map with changeable-parameter," *Chaos, Solitons & Fractals*, vol. 24, no. 1, pp. 65–71, 2005.

[30] Y. M. Motara and B. Irwin, "Sha-1 and the strict avalanche criterion," in *Information Security for South Africa (ISSA), 2016*. IEEE, 2016, pp. 35–40.

[31] T. Lin, Y. Ho, and C. Su, "High-r poly resistance deviation improvement from suppressions of back-end mechanical stresses," *IEEE Transactions on Electron Devices*, vol. 64, no. 10, pp. 4233–4241, 2017.

[32] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the challenges of crossbar resistive memory architectures," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 476–488.

[33] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, 2013.

[34] P. Knag, W. Lu, and Z. Zhang, "A native stochastic computing architecture enabled by memristors," *IEEE Transactions on Nanotechnology*, vol. 13, no. 2, pp. 283–293, 2014.

[35] M. Uddin, M. Majumder, K. Beckmann, H. Manem, Z. Alamgir, N. C. Cady, and G. S. Rose, "Design considerations for memristive crossbar physical unclonable functions," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 1, p. 2, 2017.

[36] B. Mohammad, D. Homouz, and H. Elgabra, "Robust hybrid memristor-cmos memory: modeling and design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 11, pp. 2069–2079, 2013.

[37] M. Bellare and T. Kohno, "Hash function balance and its impact on birthday attacks," in *Advances in Cryptology EUROCRYPT 04, Lecture Notes in Computer Science*. Springer-Verlag, 2004, pp. 401–418.

[38] Z. Fang, H. Y. Yu, W. J. Liu, Z. R. Wang, X. A. Tran, B. Gao, and J. F. Kang, "Temperature instability of resistive switching on $hbox HfO_x$-based rram devices," *IEEE Electron Device Letters*, vol. 31, no. 5, pp. 476–478, May 2010.

[39] B. Sahu and G. A. Rincon-Mora, "An accurate, low-voltage, cmos switching power supply with adaptive on-time pulse-frequency modulation (pfm) control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 2, pp. 312–321, 2007.

[40] B. Colombier, L. Bossuet, V. Fischer, and D. Hély, "Key reconciliation protocols for error correction of silicon puf responses," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1988–2002, 2017.

Md Badruddoja Majumder received his B.Sc. (2013) in electrical and electronic engineering from Bangladesh University of Engineering and Technology in Dhaka, Bangladesh. He is currently working towards his Ph.D in electrical engineering at University of Tennessee, Knoxville, TN. His research interest includes secure nanoelectronic circuit design and security of emerging memory technology.

Md Sakib Hasan received his B.Sc. in electrical and electronic engineering from Bangladesh University of Engineering and Technology in Dhaka, Bangladesh in 2009 and the Ph.D. degree in electrical engineering from the University of Tennessee, Knoxville in 2017. His Ph.D. dissertation was on modeling and circuit implementation of SOI four gate transistors. He is currently working as a post-doctorate research associate in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. His research interests include semiconductor device modeling, VLSI circuit design, secure nanoelectronic circuit design and neuromorphic computing.

Mesbah Uddin received his B.Sc. (2013) in electrical and electronic engineering from Bangladesh University of Engineering and Technology in Dhaka, Bangladesh. Since August, 2015, he is working towards his Ph.D. in computer engineering at University of Tennessee, Knoxville, TN. His research interest includes hardware security, VLSI circuit design, emerging nanotechnology and neuromorphic computing.

Garrett S. Rose (S'98-M'06) received the B.S. degree in computer engineering from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, in 2001 and the M.S. and Ph.D. degrees in electrical engineering from the University of Virginia, Charlottesville, in 2003 and 2006, respectively. His Ph.D. dissertation was on the topic of circuit design methodologies for molecular electronic circuits and computing architectures.

Presently, he is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville where his work is focused on research in the areas of nanoelectronic circuit design, neuromorphic computing and hardware security. Prior to that, from June 2011 to July 2014, he was with the Air Force Research Laboratory, Information Directorate, Rome, NY. From August 2006 to May 2011, he was an Assistant Professor in the Department of Electrical and Computer Engineering at the Polytechnic Institute of New York University, Brooklyn, NY. From May 2004 to August 2005 he was with the MITRE Corporation, McLean, VA, involved in the design and simulation of nanoscale circuits and systems. His research interests include low-power circuits, system-on-chip design, trusted hardware, and developing VLSI design methodologies for novel nanoelectronic technologies.

Dr. Rose is a member of the Association of Computing Machinery, IEEE Circuits and Systems Society and IEEE Computer Society. He serves and has served on Technical Program Committees for several IEEE conferences (including ISVLSI, GLSVLSI, NANOARCH) and workshops in the area of VLSI design. In 2010, he was a guest editor for a special issue of the ACM Journal of Emerging Technologies in Computing Systems that presented key papers from the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH09). Since April 2014 he is an associate editor for IEEE Transactions on Nanotechnology.