Sneak Path Enabled Authentication for Memristive Crossbar Memories

Md. Badruddoja Majumder, Mesbah Uddin, and Garrett S. Rose
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville
Knoxville, Tennessee 37996 USA
Email: {mmajumde, muddin6, garose}@utk.edu

Jeyavijayan Rajendran
Department of Electrical Engineering
The University of Texas at Dallas
Richardson, Texas 75080 USA
Email: jv.ee@utdallas.edu

***Abstract*—The memristive crossbar has emerged as a promising candidate for future memory technologies. In this paper, we explore a sneak path enabled authentication method for memristive crossbar memory. We show that, due to data dependent sneak paths currents, memristive crossbar memory can inherently produce authenticating tags for stored data. A reserved row of memory cells is also used in the proposed protocol which is reconfigured randomly during every write to the memory. We perform a numerical probabilistic analysis for evaluating the security of the proposed protocol for different sizes of crossbar and generated tags. Security is measured in terms of minimum number of trials needed to get a collision among data tags. Simulation results can be extrapolated for predicting the security of the system with larger data tags. Since the memory itself with minimal additional circuitry is used as the tag generating function, the proposed protocol exhibits low overhead compared to more conventional methods.**

***Index Terms*—Memristor, authentication, tag, collision**

I. Introduction

The integrity of memory data is a major concern for secure processor design. Accuracy and performance of all applications running on a processor is highly dependent on the application code stored in memory. If a malicious attacker modifies the memory contents and this modification remains unnoticed, then it will cause all applications dependent on that data to fail. Therefore, authentication of memory data is an integral part of reliable and trusted computing systems.

A basic feature of any memory authentication protocol is storing a tag for data being written to the memory and later verifying the read data against it. A number of previous works on memory authentication use hash functions with a tree structure to generate authenticating tags [1]–[3]. Lu *et al.* proposed a Message Authentication Code (MAC) based tree structure (M-Tree) for ensuring integrity of software [4]. Yan *et al.* proposed a memory encryption and an authentication method based on a Galois Counter Mode (GCM) operation [5]. An Advanced Encryption Standard (AES) based signature generation method for memory authentication has been demonstrated in [6]. Hong *et al.* presented a tag design method for authenticating memory data in embedded sytems [7]. This design approach exhibits less overhead compared to that of prior techniques [5], [6]. Liu *et al.* addressed the security vulnerability of Hong *et al.*'s design and proposed a method to improve it by introducing random bit flip (RBF) and non-linear randomization technique [8]. However, most existing works on memory authentication are based on hash functions, AES block cipher or other cryptographic operations exhibiting high implementation overhead. While they are robust against many security threats, lightweight applications such as Internet of Things (IoT) devices would benefit from lightweight protocols.

In this paper, we propose a lightweight authentication protocol for memristive crossbar memory using a sneak path enabled tag generation scheme. The memristive crossbar is a promising candidate for future non-volatile memory due to its high integration capability and CMOS compatibility. The passive nature of memristors enables sneak path currents through unselected cells of the crossbar. This paper proposes a novel technique that detects unauthorized modifications in memristive memory by creating authenticating data tags based on these sneak path currents.

The remainder of the paper is organized as follows. In Section II, we briefly describe the memristor and memristive crossbar memory. Section III describes the simulation model considered for different analyses of the proposed method. The sneak path based tag generation method and impact of process and environmental variations are described in Section IV. Section V describes the proposed authentication protocol and Section VI provides security analysis of the protocol in terms of collision rate among data tags. Performance in terms of delay, power, and area is compared with an existing lightweight method in Section VII. Finally, concluding remarks are provided in Section VIII.

II. The memristors, Crossbar, and Sneak Paths

A memristor is a two terminal passive circuit element. The electrical resistance of a memristor at any instant depends on the past history of current flow through the device. From a high level, the memristor can be thought of as a switching device with resistance anywhere between a high resistance and a low resistance. On applying a bias voltage, the relative width of these two regions changes such that the overall

978-1-5090-5701-6/16/$31.00

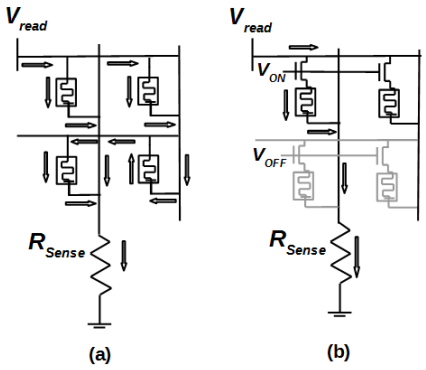

Fig. 1: (a) Sneak paths in a memristive crossbar (b) Eliminated Sneak paths in a 1 Transistor 1 Memristor (1T1M) crossbar

resistance also changes. A voltage applied with a positive polarity increases the region of low resistance state which causes the overall resistance of the memristor to decrease with time. Similarly, voltage applied with a negative polarity causes the overall resistance to increase with time. Commonly, the two extreme resistance states, high resistance state (HRS) and low resistance state (LRS), are fairly stable. Due to the non-volatility, once set to a particular resistance state, a memristor holds its state until a different bias is applied. Thus, a single memristor can work as a memory bit where the HRS and LRS represents logic '0' and '1', respectively.

A crossbar array of memristors shows great potential for applications requiring high density memories. Due to the passive nature of memristors, a voltage applied to a crossbar row does nothing to prevent current flow through unselected cells and contributes to the measured output. These current paths through unselected cells are known as sneak paths—a concern for correct readability of a memristive memory [9], [10]. Fig. 1(a) shows the current paths in a memristive crossbar through selected as well as unselected cells.

Researchers have proposed a number of techniques to control sneak paths in a memristive crossbar memory. Using one transistor and one memristor (1T1M), instead of a single memristor, for every memory cell in a crossbar is a common solution to sneak paths [11], [12]. Therefore, one can keep the transistors of unselected rows off and prevent sneak paths currents. Fig. 1(b) shows a 1T1M model of memristive crossbar where current flows are restricted through only the desired path by keeping all transistors in the unselected rows off. Thus, the 1T1M model allows the introduction and elimination of sneak paths by controlling the corresponding cell transistors.

III. Modeling of Memristor and Crossbar

Simulation of the memristive crossbar memory is done from two different aspects, one for analyzing the security and the other for evaluating implementation overhead. Security measurement of the proposed authentication system requires extensive statistical analysis. Most conventional SPICE simulators seem inconvenient in this regard due to long simulation times. We use a simplified model for crossbar memory using MATLAB for analyzing the security of the proposed system. Implementation cost in terms of area, delay, and power is measured in Cadence using a Verilog-A model for the memristor.

A. Verilog-A Model of Memristor

The memristor simulation model used here is based on a model for metal-oxide switching devices developed by McDonald *et al.* in [13]. This work considers only bipolar behavior, though the original model also provides non-polar and uni-polar behavior. Memristance is updated at each simulation time step based on the applied bias. For a positive bias, the memristance is updated by:

$$M(t_{i+1}) = M(t_i) - \frac{\Delta r \Delta t \left|V(t_{i+1})\right|}{t_{swp} V_p}, \tag{1}$$

while for a negative bias, the memristance is updated by:

$$M(t_{i+1}) = M(t_i) + \frac{\Delta r \Delta t \left|V(t_{i+1})\right|}{t_{swn} V_n}. \tag{2}$$

where M is the memristance, Δr is the absolute difference between LRS and HRS, Δt is the simulator time step-size, $V(t)$ is the applied voltage bias, V_p (V_n) is the threshold voltage for high to low (low to high) switching and t_{swp} (t_{swn}) is the required switching time with an applied bias of greater than the threshold level. Voltage below the threshold causes negligible change in the memristance level and therefore read voltage is chosen below threshold such that data remains reliably consistent between write cycles.

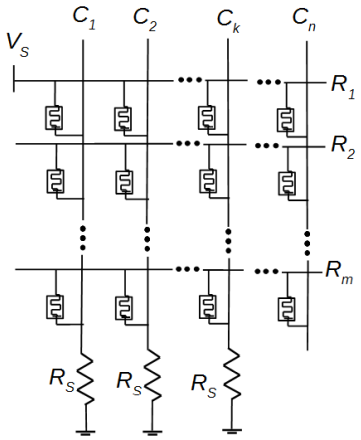

Fig. 2: Generalized $m \times n$ crossbar memory. Column C_1 through C_k are sampled with sense resistor R_S. Rest of the columns are un-sampled.

B. MATLAB Model of Crossbar

We develop a nodal-analysis based solution model for calculating sneak path currents corresponding to any data combinations of a crossbar memory. Let's consider a $m \times n$ crossbar memory of m rows, n columns as shown in Fig 2. k of the n columns are pulled down to ground, each with a sense resistance, R_s. Let's also consider the node voltages corresponding to crossbar rows are V_{R_1}, V_{R_2},...,V_{R_m}

and node voltages corresponding to crossbar columns are V_{C_1}, V_{C_2},...,V_{C_n}. Conductance of a memory cell connected between row R_i and column C_j is represented as G_{ij}. Applying Kirchoff's current law (KCL) in every node in the crossabar, $(m + n)$ nodal equations can be developed. The matrix representation of the nodal equations:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{V_R} \\ \mathbf{V_C} \end{bmatrix} = \mathbf{F} \tag{3}$$

Description of matrices used in equation 3 is given in Table I. By solving for the node voltages, current flowing through any column of the crossbar in the presence of sneak paths can be found. The solution for the node voltages is:

$$\begin{bmatrix} \mathbf{V_R} \\ \mathbf{V_C} \end{bmatrix} = \mathbf{inv}\left(\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \right) \times \mathbf{F} \tag{4}$$

TABLE I: Description of matrices used in Equation 3.

Matrix	Size	Specification
A	m x m	$A_{ij} = 0$; when $i \neq j$ $A_{ij} = \sum_{k=1}^{n} G_{ik}$; when $i = j$
B	m x n	$B_{ij} = G_{ij}$
C	n x m	$C_{ij} = G_{ji}$
D	n x n	$D_{ij} = 0$; when $i \neq j$ $D_{ij} = \sum_{k=1}^{n} G_{kj}$; when $i = j$
$\mathbf{V_R}$	m x 1	$V_{R_{ij}} = V_{R_i}$
$\mathbf{V_C}$	n x 1	$V_{C_{ij}} = V_{C_j}$
F	(m+n) x 1	$F_{ij} = V_s$; when $i = source\ row$ otherwise, $F_{ij} = 0$

IV. Data Tag Generation from Crossbar

In the presence of sneak paths, current flowing through the sampled columns is a function of every memory cell in the crossbar. This sneak path enabled data dependency when reading a crossbar memory can be leveraged as an authenticating tag. The voltage across the sense resistors of each sampled column is converted to a digital value with an Analog to Digital Converter (ADC). Readings from all sampled columns are concatenated to generate the overall tags from the crossbar memory. Fig. 3 shows a schematic for generating tags in a 1T1M crossbar memory. During tag generation, the applied voltage must be below the threshold so that it does not alter the resistance of the memristor. When the applied voltage is below the threshold, there is a negligible amount of change in the resistance and thus, any read induced reliability hazards are likewise negligible. Hence, the tag is expected to be stable for the relative frequency of read and write operations in a typical memory.

As a general property of any data authentication protocol, the size of the tag value should be large enough to provide an adequate level of security. In this sneak path based tag generation approach, tag size can be increased either by increasing the resolution of the ADC or increasing the number of columns sampled when reading the crossbar. As noise limits the resolution of the ADC, an effective way to increase tag size is to use multiple columns for generating authenticating tags.

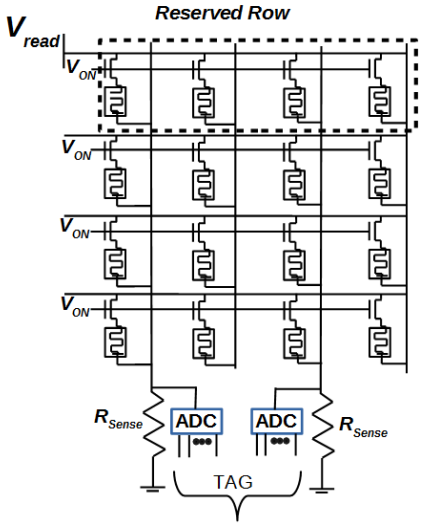

Fig. 3: Tag generation from a 4×4 crossbar memory using two sampling columns. All word lines are held high such that the 1T1M memory behaves as a crossbar with sneak paths. Encircled row with dotted rectangular box is the reserved row used for tag generation.

A. Concept of Reserved Row

The resistive network of a sneak path is such that any change in the memory cells of a selected row dominates over the effect of unselected cells. Therefore, the sneak path enabled tag generation process is biased towards cells in the selected row. A secure authentication system requires a tag generating scheme which is unbiased to all data bits. Otherwise, an attacker would target those specific bits for modifying memory contents. In the proposed method, memory cells in the selected row constitute reserved bits instead of being used as regular data bits. Fig. 3 illustrates a reserved row in a crossbar. During each tag generation phase, the reserved row is reconfigured with a random value. This concept of reserved row in the proposed authentication method serves two purposes that improves the security. First, it removes the bias of tag generation process towards any specific memory cells. Second, it improves the avalanche effect of the tag generation method. Avalanche effect is the property of changing the tag significantly due to a slight change in data. Under this reserved row technique, some additional reserved bits along with the regular data bits are changed during each new tag generation. Though the detailed analysis of avalanche effect is out of scope for this paper, the improvement can be discerned from the described concept of the reserved row.

B. Effect of Process and Environmental Variations

Process variation creates variability in the device parameters from one device to another. Intuitively, process variation does not have a detrimental impact over the tag generation method proposed. Rather, it *improves* the data dependency of sneak path currents and /it randomizes the distribution of data tags over all possible data combinations. Process variation

is accounted for in our MATLAB simulation model for the memristive crossbar by considering normal distributions for LRS and HRS.

Memristive devices also suffer from aging induced intra device variation. In the long run, it will worsen the functionality of the system which is common in any memristive applications. On the other hand, environmental variations have some impact over the reliability of tag generation. For example, due to temperature variation or noise in the supply voltage, a tag regenerated for authentication might deviate from the stored tag. However, the impact of environmental variations becomes significant only when the memristive array is large. This is another area worth investigating in the future.

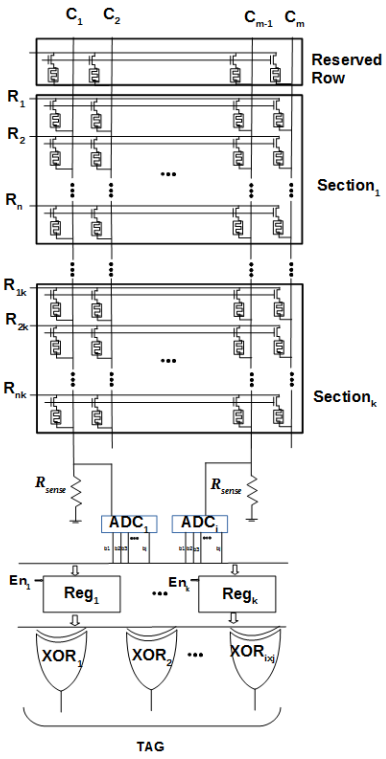

Fig. 4: Architecture of a memristive crossbar memory including additional circuitry necessary for the proposed authentication protocol.

V. Authentication Protocol

In this proposed authentication protocol, we assume the whole memory is divided into multiple sections. The sections are connected as shown in Fig. 4. Each memory section shares the same reserved row of memory cells used for generating the tags. Since the reserved row is shared, tag generation for each section is done sequentially. Another possible option might be using individual reserved rows for each section which costs extra memory cells. Though sequential tag generation needs more time for memory authentication compared to a parallel alternative, its usage of less extra memory outweighs this additional delay. Tags are generated and saved during the write operation. During read operations, tags are regenerated and verified against the previously saved tag. Different steps in these two phases of the proposed authentication protocol are as follows:

Write and Tag Generation:

1. Reserved row cells are configured with a random combinations of HRS and LRS.
2. Selected memory cells are written by keeping only the transistors in the selected row on.
3. During tag generation for each sections, cell transistors of only that section and the reserved row are kept on.
4. Generated tags from each sections are stored in registers.
5. Tags from all sections are combined with an XOR operation and saved to a secure storage to be used as a reference for authentication.

Secure storage refers to an on-chip dedicated memory that contains the authenticating tags generated from the insecure off chip main memory and only accessible by authorized users. This approach of storing tags is similar to the hash function based memory authentication described in [14].

Read and Authentication:

1. Tags are regenerated from the memory exactly the same way used for generation except the reserved row is kept as is.
2. Regenerated tags are verified against the reference tag saved during the last write operation.
3. If a mismatch is found between the two tags, an exception will be raised indicating an unauthorized modification. Otherwise, desired memory cells are read by keeping all cell transistors in the unselected rows turned off.

VI. Security Analysis

All memory authentication protocols rely on the tag generated from stored data to detect any unauthorized modification of the memory. An attacker's target is to circumvent the authentication phase by modifying memory contents in such a way that it produces exactly the same tag that was originally stored. Getting the same tag for two or more different data combinations is known as data collision which is an important security metric for a hash function or any other tag generating function used for data authentication [15]. The target of a secure authentication protocol is to maximize the required number of trials for getting a collision under brute force attack also known as birthday attack. Generally, in a data authentication protocol, security against the birthday attack is measured in terms of collision rate, which is the inverse of minimum number of trials for getting a collision.

We used our MATLAB model of crossbar memory for analyzing the security of the proposed authentication protocol. The transistors in the 1T1M memory cells are considered as ideal switches for simplicity. HRS and LRS of the memristor used in the simulation follow a normal distributions with mean $57M\Omega$, $58K\Omega$ and standard deviation 20% and 10%,

respectively. The read voltage applied to the reserved row for generating a new data tag is 600 mV and the value of sense resistance used is $58K\Omega$. The resolution considered for the ADC is $10mV$ and 4 bits are used for converting the analog voltage level to a digital value. Nominal device parameters for memristors used in the simulation are taken from [16].

As the sneak path memristive network acts as the tag generating function of this authentication protocol, it is highly dependent on the crossbar size. To evaluate the optimum size of the individual sections of crossbar memory corresponding to a particular tag size, we find the probability of collision under a particular trial of birthday attack for different sizes of crossbars and consider the crossbar with lowest probability of collision as the optimum one. Collision probability is measured according to Algorithm 1.

Input:
crossbar size, number of tag bits, number of trials, k;
number of sample space considered for numerical evaluation of collision probability, n;
Output:
probability of collision for k trials, $P(k)$;
for *i=1,2,...n* **do**
  $sum \leftarrow 0$;
  for *j=1,2,...k* **do**
    generate tag T_j for j^{th} random data combination;
    if $T_j = T_h$ *for any* $h = 1, 2, ..j-1$ **then**
      $sum \leftarrow sum + 1$;
      break;
    end
  end
end
$P(k) \leftarrow \frac{sum}{n}$;

Algorithm 1: Algorithm for finding collision probability

Analysis shown in Fig. 5 indicates that the collision probability for a particular number of attack trials decreases with the number of columns in the crossbar up to a certain point and increases with number of rows. The detailed analysis of this dependency of collision probability on crossbar size is out of scope of this paper. However, increase in number of columns causes the generated tags to more uniformly spread over different data combinations while increase in number of rows makes it biased towards some specific tags.

The collision rate of the proposed authentication system for different tag sizes is determined by finding the minimum number of trials that leads to a threshold collision probability of 0.5. Approximate collision rate for an ideal balanced tag generating function is found to be $2^{-\frac{r}{2}}$, where r is the tag size [17]. To measure the deviation from ideal scenario, data points gathered from our simulation are fitted with an equation $2^{-\alpha.r}$. The value of fitting parameter, α, is found to be approximately 0.40 from the best fit which leads to a collision rate of $2^{-\frac{r}{2.5}}$. This fitting equation can be used to predict the collision rate for higher tag sizes not explored in this paper by simulation. Comparing the fitting equation with the ideal one, it can be inferred that the proposed system requires approximately $2.5/2 = 1.25$ times as the number of tag bits required by a standard tag generating function to provide the same level of security. This comparison is also shown in Fig. 6. However, this overhead is significantly outweighed by the fact that in the proposed method, memory itself works as the tag generating function and hence requires very little additional circuitry for overall implementation. A comparison between two implementations for tag generation scheme providing the same level of security, using the proposed method and a conventional one, is presented in the next section.

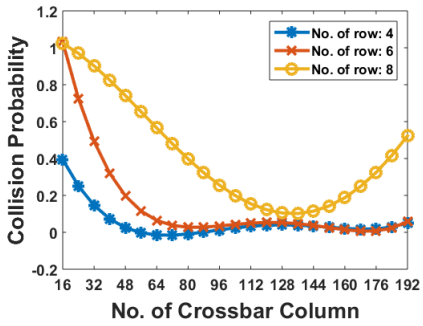

Fig. 5: Dependence of collision probability with respect to crossbar sizes for 32 bit data tags and 256 brute force trials.

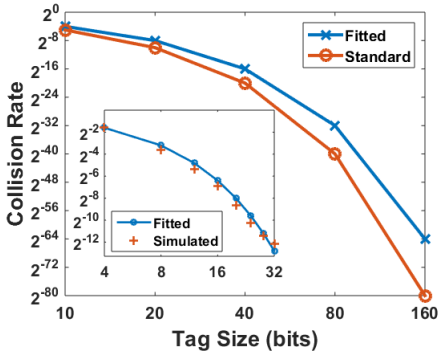

Fig. 6: Collision rate for different tag sizes. Simulated results and corresponding best fitting curve for collision rate with smaller tag sizes are shown in inset.

VII. Performance Analysis

Sneak path enabled memory authentication scheme proposed in this paper requires very little extra hardware because the crossbar memory here itself is used as the tag generating function. On the other hand, most of the conventional schemes use completely separate cryptographic modules to build the tag generator which results in significant amount of overhead.

As a point of comparison, Hong *et al.* proposed a tag generation approach for embedded system applications showing the superiority of that design to related previous works [7]. For a fair comparison, we designed and simulated tag generation protocols in 65 nm process for a 64-bit data with a 16-bit tag for Hong's design and 20-bit tag for our's approach. The 64

TABLE II: Comparison of implementation overhead between the authentication method proposed in this paper and that proposed by Hong *et al.* [7].

Overhead	Hong *et al.* [7]	Proposed
Energy (pJ)	53.66	16.59
Delay (No. of clock cycles)	8	4
Transistor count	13312	5448

bit crossbar memory is implemented using two 4×8 crossbar sections. In Hong's method, several shuffle and permutation rounds are used for tag generation which ends up with an XOR operation among different blocks. We made a very simple design for this with 4 rounds of shuffle, 1 round of rotation and an XOR operation. We also consider a more than average case for our design which involves writing 5 memory cells in the reserved row among a total of 8 before every new tag generation. Five 4-bit flash ADCs are used in our design, each requiring 364 transistors and consuming $\sim$52 μW.

Each memory cell of the considered crossbar memory is implemented with a series combination of a memristor and a NMOS select transistor. The switching time for the memristor model is considered as 10 ns. We used a time period of 10 ns for the clock used in both tag generation methods. The write operation is implemented with a two phase write method of the memristor memory [18]. In this two phase write method, all cells in a crossbar row are written with '0' at the first cycle. During the second cycle, only the desired cells are written with '1'. Using this writing method, our tag generation scheme needs only four cycles for the considered size of the memory where two cycles are used for writing the reserved row, two other cycles for generating tags by reading two sections of the crossbar sequentially.

The implementation cost for both approaches found from simulation results for both designs are given in Table II. There is more than $3\times$ improvement in the energy consumption, $2\times$ improvement in the speed and more than $2\times$ improvement in the transistor count for the proposed tag generation process. Though the results may vary for specific implementations, evidently the overhead will be significantly less for the proposed method where little additional hardware is used for tag generating function.

VIII. Conclusion

In this paper, we propose a lightweight memory authentication protocol for memristive crossbar memory. The proposed system is able to provide nearly the same level of security as a standard tag generating function does. This is validated by simulation for a tag size up to 32 bits and predicted for larger tags based on the trend of simulation results. The most notable feature of this proposed protocol is that it utilizes sneak path currents of memristive crossbar for authentication purposes and as a result, needs less than half extra hardware and energy compared to conventional schemes.

IX. Acknowledgment

This paper is partially based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0301. The authors would like to thank Musabbir Adnan and Gangotree Chakma of University of Tennessee for interesting discussions related to this topic.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, July 2013, pp. 156–160.

[11] S. Kim, H. Y. Jeong, S. K. Kim, S.-Y. Choi, and K. J. Lee, "Flexible memristive memory array on plastic substrates," *Nano letters*, vol. 11, no. 12, pp. 5438–5442, 2011.

[12] H. Manem and G. S. Rose, "A read-monitored write circuit for 1t1m multi-level memristor memories," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 2938–2941.

[13] N. R. McDonald, S. M. Bishop, B. D. Briggs, J. E. Van Nostrand, and N. C. Cady, "Influence of the plasma oxidation power on the switching properties of Al/Cu_xO/Cu memristive devices," *Solid-State Electronics*, vol. 78, pp. 46–50, December 2012.

[14] R. Elbaz, D. Champagne, C. Gebotys, R. B. Lee, N. Potlapally, and L. Torres, "Hardware mechanisms for memory authentication: A survey of existing techniques and engines," in *Transactions on Computational Science IV*. Springer, 2009, pp. 1–22.

[15] B. Alomair and R. Poovendran, "Efficient authentication for mobile and pervasive computing," in *International Conference on Information and Communications Security*. Springer, 2010, pp. 186–202.

[16] 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.

[17] 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.

[18] 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.