# Robustness Analysis of a Memristive Crossbar PUF Against Modeling Attacks

Mesbah Uddin, Md. Badruddoja Majumder, and Garrett S. Rose

IEEE Transactions on Nanotechnology, May 2017.



The online home for this paper may be found at: http://seneca.eecs.utk.edu/research.html.

Citation Information (BibTex):

```
@ARTICLE{TNANO:Uddin2017,
  author="M. Uddin and M. B. Majumder and G. S. Rose
  title="Robustness Analysis of a Memristive Crossbar PUF
      Against Modeling Attacks",
  journal="{IEEE} Transactions on Nanotechnology",
  month="May",
  year="2017",
  volume="16",
  pages="396-405",
  DOI="10.1109/TNANO.2017.2677882"
}
```

# Robustness Analysis of a Memristive Crossbar PUF Against Modeling Attacks

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

***Abstract*—In the greater context of computer security, hardware security issues such as integrated circuit counterfeiting, cloning, reverse engineering and piracy have emerged as critical issues due in part to an increasingly globalized supply chain. To help combat hardware security vulnerabilities, a wide range of security primitives have emerged in recent years. A popular example are physical unclonable functions (PUFs) which leverage process variations to provide unique signatures or fingerprints that can be used for authentication or secret key generation. Nanoelectronic technologies, such as the memristor technologies considered here, provide an excellent opportunity to engineer dense, energy-efficient PUF circuits with desirable statistical properties. Here we specifically focus on the design considerations of a memristive crossbar based PUF that generates response bits as a function of variable memristor switching time. In addition to describing the operation of the crossbar PUF, we also consider its resilience to two specific machine learning attacks, specifically through the use of linear regression and support vector machines. Two circuit design modifications for the crossbar PUF are provided to improve the resilience to machine learning attacks: XORing of response bits and internal column swapping. We show that the design modifications lead to a reduction in the likelihood of successful attack to about 50% (near ideal) even given 5,000 iterations for the attack itself. We also provide power estimates and performance considerations for the crossbar PUF based on three specific memristive material stacks: hafnium-oxide, tantalum-oxide and titanium-oxide.**

***Index Terms*—Physically unclonable function, PUF, memristor PUF, machine learning attack, modeling attack, memristor, RRAM, ReRAM, transition metal-oxide, emerging nanotechnology, hardware security.**

## I. Introduction

PHYSICAL unclonable functions (PUF) are promising security primitives useful for a number of security applications including integrated circuit (IC) piracy, cloning, and counterfeiting [1], [2]. A PUF can produce a variety of unique outputs when implemented on different devices corresponding to the same given input. The input and output of a PUF are also known as the challenge and response, respectively. PUFs implemented on different chips provide unique challenge-response combinations which can be used as the authenticating signatures for that chip. The PUF concept usually relies on exploiting random and uncontrollable physical entropy sources in a device that makes the response unique over various implementations [1].

Depending on the source of entropy, PUFs can be classified into two major types: i) delay based and ii) memory based. SRAM [3], memristor PUF (memristor as a memory element) [4], [5], STT MRAM PUF [6] are some examples of memory based PUF which leverage the uncertainty in the initial contents of a memory on start up. On the other hand, a delay based PUF utilizes path delay variations for generating unique challenge-response pairs (CRP).

As technologies continue to scale, the need has arisen for increasingly lightweight PUFs that leverage nano-scale components for lower area utilization and improved energy efficiency. Memristor based PUFs have been proposed as nano-scale security primitives. We use the word 'memristor' to mean RRAM, ReRAM or transition metal-oxide devices. Several works that focus on memristor-based PUF and memristor characteristics are presented in [7]–[13]. In a previous work, we presented a PUF designed using a dense crossbar array of memristors, the memristive crossbar PUF (XbarPUF) [14], [15]. The XbarPUF leverages the relative write-times of pairs of memristor-based circuits and thus does not require control of the specific minimum write-time of the devices. The crossbar is a very regular architecture that allows for a lightweight implementation of a PUF using nano-scale memristors.

Any PUF is potentially vulnerable to machine learning based modeling attacks [16]–[19]. Specifically, an adversary can collect a subset of all the challenge-response pairs of a PUF and can build a numerical model using a machine learning algorithm to predict the binary outcome of the response bit(s). Rührmair *et al.* showed in [16] that an arbiter PUF (APUF) [20] and its variants could be broken with more than 99% accuracy using algorithms like logistic regression and evolution strategies. While different variants of APUF could take more time to train, they are still breakable with an increased number of CRPs. In this work, we have collected a small subset of all possible CRPs of an XbarPUF and developed algorithms to perform machine learning based modeling attacks. Our implemented attack model is also found to be roughly 99% successful in predicting an XbarPUF outcome, although the modified XbarPUF architecture shows resistance against the same attack model with accuracy being around 50-60% only.

The contributions of this work include: (1) development of novel high level behavioral models of a memristor and an XbarPUF; (2) development of machine learning based attack models using logistic regression (LR) and support

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0301. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

The authors are with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, 37996 (e-mail: muddin6@vols.utk.edu; mmajumde@vols.utk.edu; garose@utk.edu).

vector machines (SVM); (3) evaluation of the robustness of XbarPUF against LR and SVM; (4) techniques for modifying the XbarPUF to improve resiliency against machine learning; (5) static power consumption estimates given the techniques considered.

We present some background on memristor and previous memristor XbarPUF works in section II-A and II-B. PUF resiliency against modeling attacks and the motivation behind this work are discussed in section II-C. Section III-A1 and III-A2 describe the development and verification of high-level behavioral models for memristors and XbarPUF. The two machine learning algorithms or models used in this work are described briefly in section III-B1 and III-B2. Section IV presents the robust modified XbarPUF architecture. Details for the experimental setup for this work are presented in section V while section VI contains results and detailed analyses. Finally, directions for future work and concluding remarks are presented in section VII.

## II. Background

### A. Memristor

CMOS has been the dominant integrated circuit technology for the last few decades. Performance improvement by scaling device parameters is the most notable part of this technology. As the feature size of CMOS technology has almost reached its limit, researchers are diving into alternative emerging nano-devices. The memristor is one of the more intriguing nano-devices under exploration for a large number of potential uses.

Chua first introduced the term memristor in 1971 by describing it as the fourth fundamental electrical component [21]. Devices fabricated by HP Labs were later demonstrated to be memristors in 2008 [22]. A memristor is a two terminal electrical device whose instantaneous resistance is dependent on its previous history. In a bipolar memristor, resistance can be switched between a high resistance state (HRS) and low resistance state (LRS) by applying an appropriate bias voltage. Switching also requires that the bias voltage be applied above a particular threshold level for at least some minimum amount of time. Threshold voltages and switching times could be different for HRS to LRS and LRS to HRS switching and are referred to as positive threshold voltage ($V_{tp}$), positive switching time ($t_{swp}$), negative threshold voltage ($V_{tn}$), and negative switching time ($t_{swn}$), respectively.

Memristor device characteristics are highly material and fabrication process dependent. Even memristors built with the same materials show significant variations in their switching characteristics due to process variations. Besides inter-die variations, a single memristor also shows variations from one switching cycle to another due to aging and other environmental disturbances. A number of materials have been proposed so far for memristive applications. In this paper, we have considered three different metal-oxide memristors: $HfO_x$, $TiO_x$ amd $TaO_x$ [15], [23], [24]. The parameter values used for our models of these representative memristive devices are listed in Table I.

It is important to note that the values in Table I are representative for each memristor device type based on a very wide range of possible device behaviors reported in the litereature. These are not the only possible representative parameter values for the corresponding materials. Device behavior will vary according to several physical characteristics not considered in detail here, including thickness, impurity concentrations, and interfacial properties of the electrodes, to name a few. Here we focus our attention on behavioral parameters in order to consider circuit-level performance across a range of potential memristive devices.

TABLE I
Switching Parameters for Metal-Oxide Memristors

| Parameter \ Devices (mean) | $HfO_x$ [15] | $TaO_x$ [24] | $TiO_x$ [23] | Parameter variance |
|---|---|---|---|---|
| HRS | 300KΩ | $10k$ | $2M$ | ±20% |
| LRS | 30KΩ | $2k$ | $500k$ | ±10% |
| $V_{tp}$ | 0.7V | 0.5V | 0.5V | ±10% |
| $V_{tn}$ | -1.0V | 0.5V | 0.5V | ±10% |
| $t_{swp}$ | 10ns | 105ps | 10ns | ±5% |
| $t_{swn}$ | 1$\mu$s | 120ps | 10ns | ±5% |

Nanoscale devices such as memristors often exhibit relatively high process variations contributing to die-to-die variations. This translates into large variation in device parameters [25], [26], [27]. The parameter variances assumed for the simulations in this work are listed in the rightmost column of Table I. These are actually worst case variations that we found for stable memristor devices noted in literature. Because of these high (and uncontrolled) die-to-die variations, memristors have high potential to be used in PUF circuits. Moreover, memristors have cycle-to-cycle variations too, resulting in different device parameters just for a single device. For this experiment, we have considered a flat 2% cycle-to-cycle variation for all memristor parameters listed in Table I.

### B. XbarPUF (Crossbar PUF)

A memristive crossbar PUF or the XbarPUF is a 2-D crossbar array of memristors originally proposed in [14] to be implemented as a strong PUF with potentially arbitrary number of challenge-response pairs derived from high variability in memristor device parameters. It was shown to be very light-weight in terms of both area and energy. A version based on the XORing of response bits was also presented in [15] where it was shown to improve reliability and robustness. Fig. 1 shows the schematic of the XORed XbarPUF.

There are three different phases of operation for an XbarPUF. First is RESET, during which a negative write voltage, $V_{wr}$, less than its negative threshold voltage, $V_{tn}$ is applied across each memristor for a sufficient amount of time to ensure that each memristor resets to their high resistance state (HRS). Second is the CHALLENGE cycle, depending on the challenge bit, either a high or zero voltage is applied to each row for a certain period of time. With a positive voltage greater than a memristors's positive threshold voltage ($V_{tp}$), a corresponding memristor starts to set to a low resistance state or LRS. Due to process variations, there would be switching time mismatches and resistance mismatches among the memristors which would give rise to different voltages at

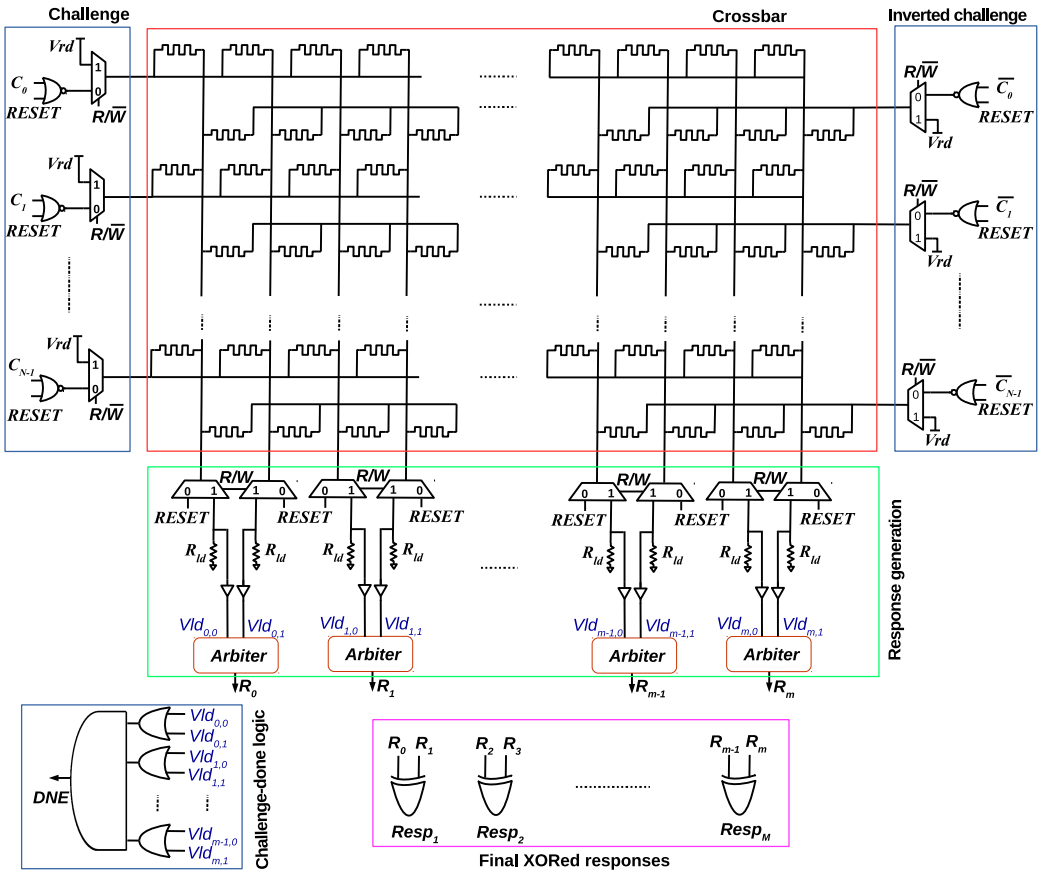


Fig. 1. Schematic of a $N \times M$ XORed crossbar-memristive PUF (crossbar architecture and peripheral read-write circuit) [15].

the load connected at the end of each crossbar column. In the READ phase, a small positive read voltage, $V_{rd}$ is applied and the load voltage between two adjacent columns are compared to generate the final response.

The modified XbarPUF described in [15] performs XOR operations on pairs of response bits to generate one response bit for each pair. This slightly increases the area and energy consumption of the PUF in terms of number of response bits but also increases the non-linearity of XbarPUF operation. Increased non-linearity improves the robustness against machine learning based modeling attack, as explored in this paper.

### C. Motivation

As stated, a PUF response results from a complex function dependent on internal process variations inherent to specific chip implementations. Therefore, it is very difficult and time consuming to predict a PUF response based only on the challenges provided. However, if the internal functionality of a PUF is not complex enough, then using a machine learning approach such as any classification algorithm, it is possible to model a PUF and predict the response with high accuracy. In [16], a modeling attack on an arbiter PUF was shown to achieve 99% accuracy with sufficient CRPs. A modified APUF with XORing, although requiring a higher number of CRPs, was also shown to be modeled very accurately. High modeling accuracy was achieved for an APUF due in part to the fact that an APUF can be expressed with a linear additive delay model and a linear classifier can be used for modeling purposes. Since a PUF response is binary, individual PUF responses can be modeled with a binary classifier. XbarPUF is very similar in behavior to the APUF, because like APUF, XbarPUF also consists of a number of delay elements, implemented by leveraging the memristors' switching rates. Thus, it is also expected that XbarPUF would be vulnerable against these machine learning based modeling attacks. Two of the most commonly used machine learning algorithms for classification purposes are logistic regression and support vector machine. Logistic regression is based on regression analysis and can be used as a linear binary classifier. It is widely used and very fast in nature. SVM is also very common for classification applications. The main objective of using SVM here is to classify non-linearly separable data more accurately by tweaking the SVM algorithm easily with different kernels.

We have created an equivalent behavioral model of a memristor and XbarPUF in MATLAB. It is much faster to simulate a high level behavioral model than simulating a large circuit at the transistor-level. Therefore, it's easy to construct large crossbar arrays and generate responses. However, it is essential that our MATLAB model imitates the actual transistor-level (Cadence Spectre) model accurately. Therefore, we have verified our MATLAB model by simulating several different circuits in Cadence to compare their performance and verifying that they behave similarly. In our previous work [15], the memristor crossbar size was only $4\times2$. It is impractical

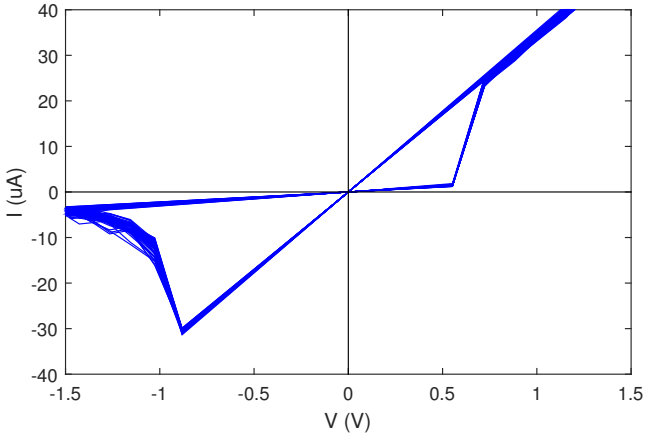


Fig. 2. Simulated IV characteristics of a metal-oxide ($HfO_x$) memristor for 200 set/reset cycles for a 100kHz clock.

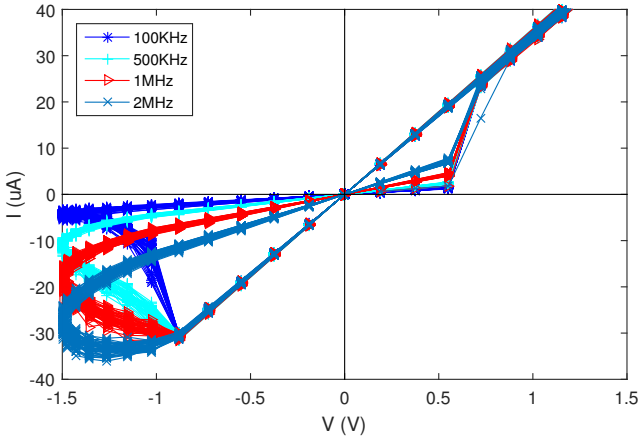


Fig. 3. Impact of frequency on the memristor IV characteristics.

to use such a small circuit for any kind of machine learning based modeling attacks as the total number of possible inputs is only $2^4$ = 16. Therefore, we have used XbarPUFs with larger numbers of CRPs so that only a small subset of all possible combinations of CRPs are used in our experiments.

## III. Methodology

### A. Development of High Level MATLAB Models

*1) High Level Memristor Model:* In prior work demonstrating the XbarPUF [15], we have used a Verilog-A model for a hafnium-oxide memristor adapted from a model presented first in [28]. For this work, we have also created an equivalent high level model of a memristor to use in MATLAB based simulations. At each simulation time step, memristance at the next step is calculated from the memristor's current state based on the time duration and magnitude of an applied voltage bias. Details of this switching operation can be found in [15]. As mentioned before, there are six primary parameters in our memristor model, specifically: HRS, LRS, $V_{tp}$, $V_{tn}$, $t_{swp}$, and $t_{swn}$. As necessary, parameter values are recalculated on each iteration using their mean and cycle-to-cycle variations for any particular memristor. Fig. 2 shows the IV characteristics of a memristor for 200 set and reset cycles. All parameter values are allowed to vary from one cycle to another, even for a single memristor. The various hysteresis loops in Fig. 2 demonstrate cycle-to-cycle variations for a single modeled device.

The choice of clock frequency is very important for reliable set and reset operations for any particular memristor. If the clock time period is less than either the set or reset switching time, then the memristor is not able to switch to HRS or LRS completely in that cycle and would be in an intermediate state. This relationship with increasing frequency, or decreasing time period is shown in Fig. 3. With increasing frequency, the hysteresis loop tends to be narrower indicating the memristor is not fully set to LRS or reset to HRS. In this particular example, the maximum of the two switching times is 1$\mu$s. Therefore, if the frequency is greater than 1/1$\mu$s or 1MHz, a smooth curve is observed in Fig. 3.

*2) High Level XbarPUF Model:* The memristor crossbar PUF circuit consists of two main components, a memristor crossbar array and peripheral read-write circuit. A crossbar is simply a 2-D array with memristors at each crosspoint. A matrix of size *row*$\times$*col* is created where each component is represented with a memristor 'class' in MATLAB. Using respective means and standard deviations for all the memristor parameters, and assuming a Gaussian distribution function, each memristor is assigned to different random initial values for its parameters. This is the step where die-to-die variations of memristors are mimicked in the simulation.

As mentioned, three phases of operations are performed on the memristor crossbar. A vector with its length equal to the number of rows in the crossbar is created to represent the voltage applied to each row. During RESET, all these voltages are equal to -$V_{wr}$. Each memristor is reset to HRS and, therefore, the effective memristance across an entire column is at its maximum value. The load resistance is chosen such that it has minimum drop during this reset period and maximum drop during the challenge. During the CHALLENGE phase, either a positive $V_{wr}$ or zero voltage is applied to each row depending on the state of the corresponding challenge bit. If the challenge bit is high, then memristors in that row start to set toward LRS and the voltage drop across the load starts to increase. Otherwise, any corresponding memristor is kept at HRS since the applied voltage is 0V. In the READ phase, all the memristors in a column are in parallel with the applied voltage $V_{rd}$. This effective column memristance is in series with the column's corresponding load and a simple voltage divider determines the load voltage. The load voltage from two side-by-side columns are compared to generate a response with '0' for one column and '1' for the other. Which column first hits the '1' state determines the column that "wins" the race, thus generating the corresponding response bit. For an XORed XbarPUF, two response bits (generated from 2 column pairs) are logically XORed to determine one final bit.

### B. Employed Machine Learning Attack Models

*1) Logistic Regression (LR) Model with Gradient Descent:* Logistic regression [29] is a statistical method for analyzing

a dataset and predicting a dichotomous outcome. Thus, it does the job of a binary classification algorithm but is more similar to a regression model. It analyzes a data set and assigns a probability to each data-point from which one of the two classes it belongs using a logit or signum function. It is particularly powerful for classifying data with a relatively small feature set and is very popular due to fast convergence and an easy-to-implement algorithm.

Gradient descent is often used with logistic regression to minimize errors in each iteration. Gradient descent is an iterative algorithm which helps find local minima of a function by taking steps proportional to the negative gradient at each iteration. In our experiment, we have developed a custom implementation of LR along with gradient descent as cost minimization function.

In our LR algorithm, we have set the number of features to be the same as the number of challenge bits plus one extra feature as bias. Initially, the weight of each feature is set to zero. It then works on the training dataset and assigns a '0' or '1' to each data point. For an $m \times n$ XbarPUF, we select feature set size of $k = m+1$, where '$m$' features representing '$m$' challenges, and 1 for bias. Therefore, the feature vector, $\Theta$ would be a function of m-bit challenge C. As in LR algorithm, we use sigmoid function ($h_\Theta(\mathbf{X})$) to evaluate the probability of each data point belonging to one of two classes. Equations 1-4 describe all relationships in this method. ($X$,$y$) denotes a single data-point, i.e. a challenge-response pair. The constant 1 at the beginning of X is the bias term.

$$\mathbf{X} = [1, C_1, C_2, \ldots, C_m, C_{x1}, C_{x2}, C_{x3}] \tag{1}$$

$$\mathbf{\Theta}(\mathbf{C}) = [\Theta_0, \Theta_1, \Theta_2, \ldots, \Theta_m, \Theta_{x1}, \Theta_{x2}, \Theta_{x3}]^T \tag{2}$$

$$h_\Theta(\mathbf{X}) = sigmoid(\mathbf{\Theta}^T\mathbf{X}) = \frac{1}{1 + e^{-\mathbf{\Theta}^T\mathbf{X}}} \tag{3}$$

$$\mathbf{R} = \begin{cases} 1, & \text{if } h_\Theta(\mathbf{X}) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Thus, $h_\Theta(\mathbf{X})$ = 0.5 or equivalently $\Theta^T\mathbf{X}$ = 0 is the separating hyperplane for classification. More features could be created using the inner-products of existing features. However, we are using LR model as a linear classifier or linear attack model due to its fast convergence time with less memory requirement and, therefore, we have tried to keep the model simple.

The cost or error of a single data point is defined as:

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{if y = 1} \\ -\log(1 - h_\theta(x)), & \text{if y = 0} \end{cases}. \tag{5}$$

An important characteristic of this logarithmic cost function is that it gives zero error when LR performs classification correctly, but produces a very high cost when there is a classification error. The shape of this cost function is also either monotonically increasing or decreasing on one side of the global minimum, thus avoiding any local minima.

The total cost of the classifier is just the sum of errors for each data point and is defined as:

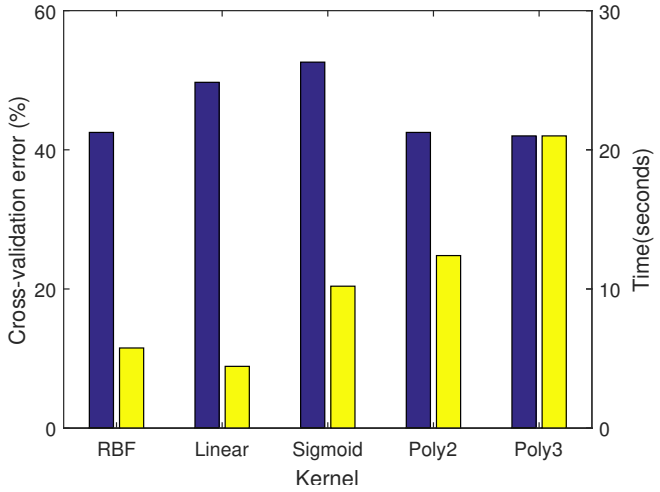


Fig. 4. Results of cross-validation of training data for different kernel functions of SVM.

$$J(\theta) = \frac{1}{k}\sum_{i=1}^{k} Cost(h_\theta(x^{(i)}, y^{(i)})) \tag{6}$$

where k = total number of features.

Minimization of this cost function $J(\theta)$ is then achieved by the gradient descent algorithm. On each iteration, all the feature parameters $\theta_j$ are simultaneously updated using this equation:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial_j}J(\theta) \tag{7}$$

This is done by partially differentiating the cost function for each $\theta_j$ to update their values, thereby achieving the direction for global minima of cost function. In this way, the gradient descent algorithm converges to a solution very fast.

*2) Support Vector Machine (SVM):* A support vector machine or SVM [30] is a supervised learning algorithm primarily used for classification. SVM attempts to find separating hyperplanes as decision boundaries to classify data into multiple sets. For example, a 2-D data set could be classified with a straight line, a 3-D data set with a plane and so on. However, using a kernel trick, different decision boundaries can be specified for data where the two classes are not linearly separable.

For this work, we have utilized the machine learning toolbox from MATLAB [31]. We have experimented with five different kernel functions, namely, Gaussian or RBF (radial basis function), linear, sigmoid and polynonimal kernel with order 2 and order 3. We have used a dataset containing 5000 data points or CRPs generated from the modified and robust XbarPUF architectures. Two-thirds of this dataset is used for training. To determine and choose the best kernel function we have worked with the training dataset separately. Specifically, we have divided up the training dataset into ten sets to perform k-fold cross-validation. Cross-validation with each kernel is iterated many times providing the results shown in Fig. 4. The cross-validation error rate is found to be similar for the RBF, poly2 and poly3 kernels. Fig. 4 also shows the average

TABLE II
COLUMN SWAPPING LOGIC IMPLEMENTED IN THIS DEMONSTRATION

| S1 S0 | y3 | y2 | y1 | y0 |
|---|---|---|---|---|
| 0 0 | x3 | x2 | x1 | x0 |
| 0 1 | x3 | x2 | x0 | x1 |
| 1 0 | x2 | x3 | x1 | x0 |
| 1 1 | x2 | x3 | x0 | x1 |

| S2 | z3 | z2 | z1 | z0 |
|---|---|---|---|---|
| 0 | y3 | y2 | y1 | y0 |
| 1 | y1 | y0 | y3 | y2 |

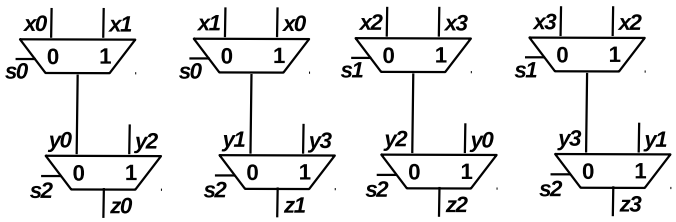


Fig. 5. An implementation of column swapping circuit.

elapsed time for one run of cross-validation for each kernel. Since RBF has much lower training time than poly2 and poly3 with accuracy being almost the same, we have chosen the RBF kernel for the rest of our experiment.

The main purpose of using SVM in our analysis is to determine how the XbarPUF model holds up against a non-linear classifier, especially when non-linear features such as XORing and column-swapping are added to the XbarPUF. Although, as expected, SVM with a non-linear decision boundary is slower than a linear classifier such as LR, SVM performs better for the modified XbarPUF architecture. It is also important to mention that we have used already implemented SVM functions from the machine learning toolbox of MATLAB, unlike LR where we developed custom implementations.

## IV. Modification of XbarPUF Architecture to Mitigate Machine Learning based Attacks

The XOR operation introduces additional non-linearity in a XbarPUF and thus, is assumed to improve robustness against machine learning attacks. This is found to be partially correct as described in later sections of this paper. However, it is also found that a machine learning algorithm with a carefully chosen non-linear decision boundary can help make accurate predictions when presented with a large number of CRPs. Therefore, we introduce a second method for adding further non-linearity to the XbarPUF. Specifically, a 'column swapping' or 'column shuffling' circuit can be added which connects any one of four upper crossbar columns to any of the four lower columns, such that each column is broken and recombined. The implemented shuffling logic is provided in Table II.

To ensure that any delay difference arising from the swapping or shuffling operation is minimal, we have used only a subset of all possible combinations of $2^4 = 16$. It is also essential that the delay for each path be the same such that this operation does not add any bias to the XbarPUF and the memristors remain the primary sources of entropy. Keeping these requirements in mind, we have implemented the column swapping operation as shown in Table II. A specific logic implementation using multiplexers is also shown in Fig. 5.

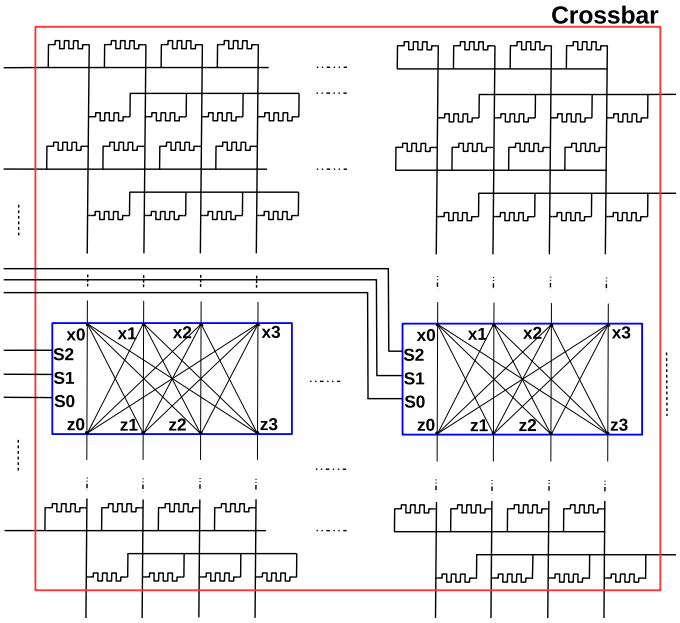


Fig. 6. XbarPUF with the inclusion of column swapping logic.

Three extra challenge bits are used as selectors for this column swapping circuit. Assuming a $M \times N$ crossbar in terms of rows and columns, a total of $N/4$ different column swapping circuit blocks would be required. The same three selector bits can be used or different bits can be used for different blocks as well as shown in Fig. 6. Increasing the number of challenge bits, in this case adding bits that do not control the memristance directly, also increases the complexity of the learning or modeling algorithm. In this demonstration, however, the crossbar has four columns and only one swapping circuit and overall three additional challenge bits are used.

## V. Experimental Setup

For this work, we have assumed a crossbar size of 32×2 in terms of challenge-response pairs, containing a total of 64×8 memristors. The purpose of this experiment is to collect a small subset of all the possible input combinations and use that small subset to perform a modeling attack on the XbarPUF. For most of the experiment, we have applied 5000 random challenges (out of possible $2^{32}$) and collected the 2-bit response for the XbarPUF without XORing and the 1-bit response for the XORed XbarPUF. The write and read voltages and clock frequency are chosen according to the specific parameter values of particular memristor considered. As an example, for a memristor with threshold voltages of 0.7V and -1V, we chose the write and read voltages to be 1.3V and 0.6V, respectively. For a set or reset switching time of 1$\mu$s, the clock period is chosen such that several cycles are needed for a memristor to switch from HRS to LRS or LRS to HRS, respectively.

Each set of memristor parameters is generated from a normal distribution using the same set of mean and variance for all devices. Afterward, a random challenge is applied and the response is generated for all four different types of XbarPUF: basic XbarPUF (no mitigation), XbarPUF with

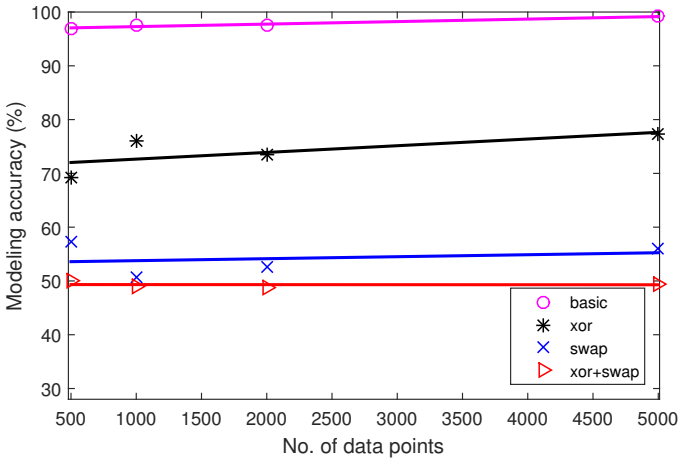


Fig. 7. Modeling accuracy of LR vs. the size of dataset.

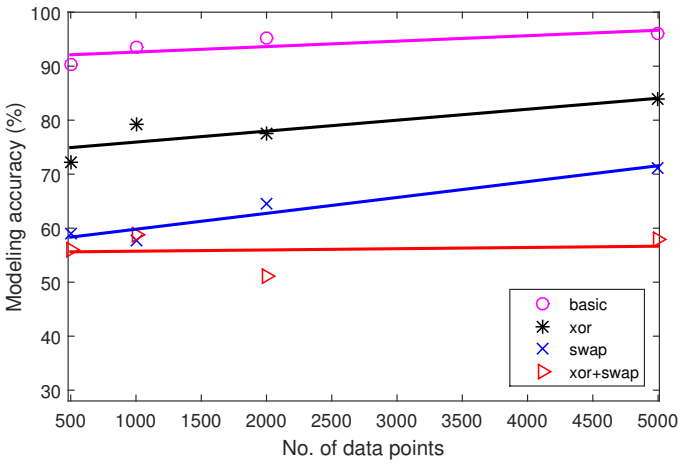


Fig. 8. Modeling accuracy of SVM vs. the size of dataset.

TABLE III
RESULTS FROM LOGISTIC REGRESSION BASED MODELING ATTACK

| XbarPUF | Training accuracy (%) | Testing accuracy (%) |
|---|---|---|
| No XORing or column swapping | 99.7 | 99.0 |
| XORing only | 77.0 | 77.4 |
| Column swapping only | 65.6 | 65.5 |
| Both column swapping and XORing | 54 | 49 |

TABLE IV
RESULTS FROM SVM (WITH RBF KERNEL) BASED MODELING ATTACK

| XbarPUF | Training accuracy (%) | Testing accuracy (%) |
|---|---|---|
| No XORing or column swapping | 99.8 | 96.8 |
| XORing only | 93.4 | 83.8 |
| Column swapping only | 86 | 80.5 |
| Both column swapping and XORing | 85.3 | 57.9 |

column swapping only, with XORing only and with both column swapping and XORing. As an example, for a column-swapping only XbarPUF, the size of the dataset would be 5000×37, where 37 = 32 challenge bits + 3 selector bits for column swapping logic + 2 response bits. Two-thirds of the dataset points are used for training while the remainder are used for testing. Logistic regression has the same number of features as the number of challenges with one additional feature as the bias. The SVM model uses the same number of features as the number of data points. By applying a kernel trick, the SVM model was fitted to represent non-linear datasets more accurately. In this experiment, we choose the RBF or radial basis function kernel as discussed earlier.

## VI. RESULTS AND DISCUSSIONS

The results from the logistic regression based modeling attack using 5000 data-points are shown in Table III. It is evident from the table that the basic memristor XbarPUF, that was first proposed in [14], is very susceptible to machine learning based modeling attacks. The numbers in the table show the average modeling accuracy for two response bits. This result was expected since the XbarPUF has a somewhat similar behavior as that of an APUF and the APUF was also shown to be modeled very successfully [16]. The addition of non-linearity to the XbarPUF architecture does increase modeling complexity, especially for a linear classifier, as is clearly indicated in Table III. Specifically, the modeling accuracy is lower for XbarPUF with XORing and column swapping, although column swapping shows better resilience to modeling attack than XORing. However, after we incorporate both of these two methods, modeling accuracy drops significantly and is very close (≈49.5%) to a random binary predictor of 50%. This shows the strength of the combined XORing+column swapping operation with XbarPUF against a robust classification algorithm.

The results from support vector machine (SVM) based modeling attacks are shown in Table IV. SVM also achieves very high accuracy for XbarPUF without any modification by XORing or column swapping. The accuracy is comparable to LR for basic XbarPUF. The main purpose of using SVM in our experiment is to achieve improved modeling accuracy when presented with a non-linear dataset. This is justified by comparing results from Table III with Table IV, which show SVM has higher accuracy when fed with a non-linearly separable dataset, i.e. with an XbarPUF with XORing or column swapping included. When both XORing and column swapping are included, the accuracy is still very low for the testing dataset. However, the training accuracy is much higher than testing accuracy in this particular case and indicates the presence of over-fitting. Since we desire a fair comparison among all the entries of Table IV, we have used the same configuration to generate all results despite any over-fitting. The best modeling accuracy of only around 58% suggests that XORing+column swapping is an effective circuit-level improvement to mitigate this modeling attack.

Further demonstrations of the effect of increasing data-points vs. the modeling accuracy for LR and SVM methods are shown in Fig. 7 and 8 for the four versions of XbarPUF: basic, with XORing only, with column swapping only and with both XORing and column swapping included. For the

TABLE V
POWER ESTIMATION OF A $HfO_x$ MEMRISTOR BASED XBARPUF FROM MATLAB AND CADENCE

| Crossbar size (chall×resp) | Average Power (mW) (MATLAB) | Average Power (mW) (Cadence Spectre) |
|---|---|---|
| 4×2 | 0.25 | 0.26 |
| 8×2 | 0.51 | 0.61 |
| 16×2 | 0.98 | 1.01 |
| 32×2 | 2.07 | 2.22 |

TABLE VI
POWER CONSUMPTION DURING DIFFERENT PHASES OF A 32×2 XBARPUF

| XbarPUF Phase | $HfO_x$ (mW) | $TiO_x$ (mW) | $TaO_x$ (mW) |
|---|---|---|---|
| RESET | 3.20 | 0.27 | 51.40 |
| CHALLENGE | 2.10 | 0.007 | 42.20 |
| READ | 0.60 | 0.014 | 1.10 |

LR method, increasing the data-points does not help much in improving the accuracy of modeling for any of the XbarPUF options with added non-linearity, as expected. This can be verified from the near-zero slope of the plots in Fig. 7. If we compare results of Table 1 from [16] for the LR method, we can see tiny change in accuracy over a much larger change in dataset size or in number of CRPs which further justifies our result. On the other hand, modeling accuracy with SVM, though small, increases linearly as the number of data-points increases. Training time is also much smaller (especially with LR) for simpler PUFs like XbarPUFs with higher prediction accuracy. The same case is noticeable for arbiter PUFs in [16]. Like arbiter PUF, XbarPUF also becomes more resilient against modeling after XORing is introduced [16]. However, both SVM and LR also fail to improve accuracy on the last version of XbarPUF, the XbarPUF with XORing+column swapping. With both algorithms, accuracy is near 50%, the accuracy of a pure random guess for a balanced binary output. Thus, it can be concluded that this method (XORing+column swapping) is very robust against machine learning attacks.

We also have evaluated the approximate power consumption for our XbarPUF to show its merit as a lightweight security primitive. Due to the presence of a constant resistive path from source to ground, the XbarPUF circuit has static power consumption. This static power consumption includes power from all the memristors and the CMOS transistors in series with the crossbar columns' current path. The peripheral digital logic gates comprising of CMOS also exhibit dynamic power dissipation. To verify the accuracy of our high-level XbarPUF power estimation model, we have shown results for power consumption estimation of $HfO_x$ memristor based XbarPUF from both high-level (MATLAB) and transistor-level (Cadence Spectre) simulations. Table V provides this result for four different crossbar sizes. Although we get slightly lower values from MATLAB since we have neglected dynamic power dissipation of the peripheral digital logic circuitry, the overall estimations are consistently close to Spectre results and follow the same trend.

Table VI provides the static power consumption during three different cycles of operation for a 32×2 XbarPUF for the three different types of memristors mentioned in Table I. Although the overall memristance is lower during both CHALLENGE and READ phases, the power consumption is highest during RESET since read voltage is almost half the write or reset voltage. It is worth noting that the average power values of Table V can be calculated from Table VI with the amount of time spending in one cycle of operation in mind. Our designed XbarPUF stays in RESET for half of the time of a sampling cycle and very small time (≈8%) in CHALLENGE phase and thus RESET and READ phase dominate the average static power consumption. As the XbarPUF scales up and the number of memristors increase, static power starts to dominate the overall power consumption of the PUF circuit.

The power values in Table VI are calculated for the data from Table I. As expected, power consumption is directly dependent on the values of HRS and LRS and read-write voltages which depend on the switching threshold voltages of the memristors used. The reason behind a larger power dissipation for the $TaO_x$ memristor based XbarPUF is due to the fact that the $TaO_x$ memristors have much lower HRS and LRS values than the other memristors considered. If HRS and LRS are very low, the equivalent memristance of a column drops and becomes comparable to the series PMOS and NMOS channel resistances from the peripheral circuitry. Thus, the MOSFETs also consume a good portion of the total power in the circuit with $TaO_x$ devices. If an XbarPUF is scaled up, the equivalent memristance drops further and thus $TaO_x$ memristor XbarPUF may not be operational for a larger number of rows, i.e. larger number of challenge bits. It is to be noted that read-write voltages for $HfO_x$ memristors are 1.3V and 0.6V, respectively, and 1.0V and 0.3V, respectively, for the other two memristors. If all three memristor circuits use the same read-write voltages, $TaO_x$ and $TiO_2$ would have even higher power consumption than those listed in Table VI.

We have already discussed the area of a XbarPUF in our previous works [14], [15]. Memristors require effectively zero area when compared against CMOS area. Specifically, with an increasing number of CRPs, the area of a XbarPUF increases linearly with a small slope while the area of a CMOS APUF or other CMOS PUF increases exponentially. The XbarPUF with XORing or column swapping requires slightly more area than the basic XbarPUF because of increased number of logic gates, but still requires much less area as compared to any CMOS PUF.

Figure 9 shows the relation of static power consumption with the number of crossbar rows and columns for a basic XbarPUF. As expected, the power consumption increases linearly with an increase in the number of response and challenge bits. The XbarPUF with $HfO_x$ and $TiO_x$ memristors have significantly less power consumption as compared to $TaO_x$ because of their relatively higher value of LRS and HRS. For the same reason their operating ranges in terms of number of challenge bits are larger too. The power consumption for XbarPUF (not shown) with column swapping would be slightly higher due to the series connection of the two multiplexers in each column. For the XORed XbarPUF, the power consumption would be double that of the basic

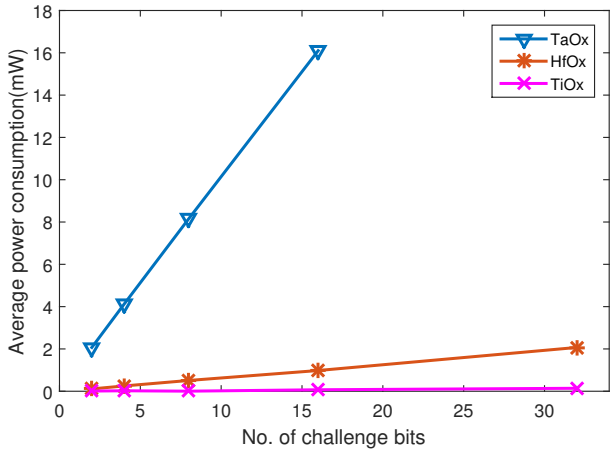


Fig. 9. Relationship between average static power consumption and size of the XbarPUF.

XbarPUF, as the XORed version has half as many response bits and such that the crossbar size is double for the same number of response bits.

Major security metrics for a PUF include uniqueness, uniformity, reliability and bit-aliasing [32]. Although we have not shown them here, we have evaluated and presented results for these metrics in our previous works [14], [15]. Those results show that the merit of PUFs built from emerging devices (e.g. memristors) is on par with that of CMOS PUFs in terms of security performance. Furthermore, the nanoelectronic security primitives show improvements in terms of area and power efficiency.

## VII. Conclusion and Future Work

The main focus of this work has been the evaluation of the robustness of a memristive crossbar PUF against machine learning based modeling attacks. To the best of our knowledge, this is the first demonstration that the basic XbarPUF is also prone to these machine learning attacks. Unlimited access to a PUF for a few seconds can potentially give away thousands of CRPs to build an attack model. In addition to understanding the vulnerability of the basic XbarPUF to machine learning, we have also provided mitigation techniques (i.e. XORing and column swapping) which dramatically reduce the accuracy of the machine learning algorithms considered to the point of essentially being equivalent to a random guess. However, in our attack model, we have assumed that an adversary does not have full knowledge as to which bits in a challenge set are used as selectors for the mixing or swapping logic circuit blocks. It is necessary to obfuscate those bits from the perspective of an attacker. Thus, our immediate focus is to work on the attack models and test strategies that work better with different types of PUFs.

Using nanoelectronic devices such as memristors for a PUF application has several potential advantages over traditional PUF architectures. Specifically, because of their very high integration density, it is much more difficult to have a successful probing attack on XbarPUFs or similar nanoelectronic PUFs. Another important reason behind using nanoelectronic devices for security is to provide lightweight security schemes. All of these advantages notwithstanding, a crossbar array consisting of many memristors can have significant power consumption if the HRS and LRS of the memristors are too low. Thus, it is important to choose the nanoelectronic devices carefully based on their range of operations and specific applications.

## Acknowledgment

The authors would like to thank Lok Kwong-Yan, Bryant Wysocki, Nathan McDonald and Jillian Hallak of the Air Force Research Laboratory for interesting discussions relating to this topic. The authors are also grateful to Gangotree Chakma, Musabbir Adnan and other members of their UTK research group for their valuable contribution in proofreading this work.

## References

[1] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *44th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2007, pp. 9–14.

[2] Z. Paral and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *IEEE Int. Symp. on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2011, pp. 128–133.

[3] J. Guajardo, G. Kumar, S. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *Proc. of the IEEE Int. Conf. on Field Programmable Logic and Applicat.*, August 2007, pp. 189–195.

[4] P. Koeberl, U. Kocabas, and A.-R. Sadeghi, "Memristor PUFs: A new generation of memory-based physically unclonable functions," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2013, pp. 428–431.

[5] O. Kavehei, C. Hosung, D. Ranasinghe, and S. Skafidas, "mrPUF: A Memristive Device based Physical Unclonable Function," *ArXiv e-prints*, Feb. 2013.

[6] L. Zhang, X. Fong, C.-H. Chang, Z. H. Kong, and K. Roy, "Highly reliable memory-based physical unclonable function using spin-transfer torque MRAM," in *IEEE Int. Symp. on Circuits and Syst. (ISCAS)*. IEEE, 2014, pp. 2169–2172.

[7] G. S. Rose, N. McDonald, L. Yan, and B. Wysocki, "A write-time based memristive PUF for hardware security applications," in *Proc. of the IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, November 2013, pp. 830–833.

[8] A. Mazady, H. Manem, M. Rahman, D. Forte, and M. Anwar, "Memristor PUF - a security primitive: Theory and experiment," *IEEE J. on Emerging and Selected Topics in Circuits and Syst.*, vol. 5, no. 8, pp. 222–229, June 2015.

[9] G. S. Rose, M. Uddin, and M. B. Majumder, "A designer's rationale for nanoelectronic hardware security primitives," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 194–199.

[10] A. Chen, "Utilizing the variability of resistive random access memory to implement reconfigurable physical unclonable functions," *IEEE Electron Device Lett.*, vol. 36, pp. 138–140, February 2015.

[11] H. Abunahla, B. Mohammad, and D. Homouz, "Effect of device, size, activation energy, temperature, and frequency on memristor switching time," in *26th Int. Conf. on Microelectronics (ICM)*, December 2014, pp. 60–63.

[12] P. Y. Chen, , Tempe, R. Fang, R. Liu, C. Chakrabarti, Y. Cao, and S. Yu, "Exploiting resistive cross-point array for compact design of physical unclonable function," in *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, May 2015, pp. 26–31.

[13] R. Liu, H. Wu, Y. Pang, H. Qian, and S. Yu, "Experimental characterization of physical unclonable function based on 1 kb resistive random access memory arrays," *IEEE Electron Device Lett.*, vol. 36, pp. 1380–1383, October 2015.

[14] G. Rose and C. Meade, "Performance analysis of a memristive crossbar PUF design," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[15] M. Uddin, M. B. Majumder, G. S. Rose, K. Beckmann, H. Manem, Z. Alamgir, and N. C. Cady, "Techniques for improved reliability in memristive crossbar PUF circuits," in *IEEE Comp. Society Annual Symp. on VLSI*, July 2016.

[16] U. Ruhrmair, J. Solter, F. Sehnke, and X. Xu, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. on Inform. Forensics and Security*, vol. 8, pp. 1876–1891, August 2013.

[17] D. Lim., "Extracting Secret Keys from Integrated Circuits. MSc thesis," Master's thesis, MIT, Massachesetts, USA, 2004.

[18] M.-D. M. Yu, D. MRaihi, R. Sowell, and S. Devadas, "Lightweight and secure puf key storage using limits of machine learning," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 358–373.

[19] X. Xu and W. Burleson, "Hybrid side-channel/machine-learning attacks on PUFs: A new threat?" in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[20] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas, "Design and implementation of the AEGIS single-chip secure processor using physical random functions," in *Proc. of the 32nd Annual Int. Symp. on Comput. Architecture*, 2005, pp. 25–36.

[21] L. O. Chua, "Memristor-the missing circuit element," *IEEE Trans. on Circuit Theory*, vol. 18, no. 5, pp. 507–519, September 1971.

[22] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008.

[23] G. Medeiros-Ribeiro, F. Perner, R. Carter, H. Abdalla, M. D. Pickett, and R. S. Williams, "Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution," *Nanotechnology*, vol. 22, no. 9, p. 095702, 2011.

[24] J. J. Yang, M. Zhang, J. P. Strachan, F. Miao, M. D. Pickett, R. D. Kelley, G. Medeiros-Ribeiro, and R. S. Williams, "High switching endurance in TaOx memristive devices," *Applied Physics Letters*, vol. 97, no. 23, p. 232102, 2010.

[25] P. Pouyan, E. Amat, and A. Rubio, "Reliability challenges in design of memristive memories," in *5th European Workshop on CMOS Variability (VARI)*, September 2014, pp. 1–6.

[26] 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 $HfO_2$ -based RRAM devices," *IEEE Electron Device Lett.*, vol. 31, pp. 476–478, May 2010.

[27] C. Walczyk, D. Walczyk, T. Schroeder, T. Bertaud, M. Sowinska, M. Lukosius, M. Fraschke, D. Wolansky, B. Tillack, E. Miranda, and C. Wenger, "Impact of temperature on the resistive switching behavior of embedded $HfO_2$ -based RRAM devices," *IEEE Trans. on Electron Devices*, vol. 58, pp. 3124–3131, September 2011.

[28] 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/$Cu_x$O/Cu memristive devices," *Solid-State Electronics*, vol. 78, pp. 46–50, December 2012.

[29] C. M. B. et. al., *Pattern recognition and machine learning*. New York, USA: Springer, 2006.

[30] T. Joachims, *Text categorization with Support Vector Machines: Learning with many relevant features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142.

[31] "Support Vector Machines for Binary Classification," https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html, 2014.

[32] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds. Springer New York, 2013, pp. 245–267.


**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.

**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.

**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.