Machine Learning Attack and Defense on Voltage Over-scaling-based Lightweight Authentication

Jiliang Zhang, Member, IEEE, Haihan Su, Gang Qu, Senior Member, IEEE

Abstract—It is a challenging task to deploy lightweight security protocols in resource-constrained IoT applications. A hardware-oriented lightweight authentication protocol based on device signature generated during voltage over-scaling (VOS) was recently proposed to address this issue. VOS-based authentication employs the computation unit such as adders to generate the process variation dependent error which is combined with secret keys to create a two-factor authentication protocol. In this paper, machine learning (ML)-based modeling attacks to break such authentication is presented. We also propose a challenge self-obfuscation structure (CSoS) which employs previous challenges combined with keys or random numbers to obfuscate the current challenge for the VOS-based authentication to resist ML attacks. Experimental results show that ANN, RNN and CMA-ES can clone the challenge-response behavior of VOS-based authentication with up to 99.65% prediction accuracy, while the prediction accuracy is less than 51.2% after deploying our proposed ML resilient technique. In addition, our proposed CSoS also shows good obfuscation ability for strong PUFs. Experimental results show that the modeling accuracies are below 54% when $10^6$ CRPs are collected to model the CSoS-based Arbiter PUF with ML attacks such as LR, SVM, ANN, RNN and CMA-ES.

I. INTRODUCTION

The Internet of Things (IoT) is a novel networking paradigm which connects a variety of things or objects to the Internet through sensor technology, radio frequency identification (RFID), communication technology, computer networks and database technology [1]. According to the IHS forecast [2], the IoT market will grow from an installed base of 15.4 billion devices in 2015 to 30.7 billion devices in 2020 and 75.4 billion in 2025. With the increasing of IoT devices, security issues have attracted much attention. For example, in 2016, America suffered the largest DDoS attack in history [3]. The cyberattack that brought down much of America internet was caused by the Mirai botnet, which is a worm-like family of malware that infected IoT devices and corralled them into a DDoS botnet [4]. Therefore, secure and efficient defenses need to be deployed for IoT devices.

Secret key storage and device authentication are two key technologies for IoT security. Traditional key generation and authentication techniques are based on the classical cryptography, which requires expensive secret key storage and high-complexity cryptographic algorithms. In many IoT applications, resources like CPU, memory, and battery power are limited and cannot afford the classic cryptographic security solutions. Therefore, lightweight solutions for IoT security are urgent.

Physical unclonable functions (PUFs) [5], [6] and recently proposed voltage over-scaling (VOS) based authentication [7] are two emerged lightweight security primitives for IoT device authentication.

PUFs use a random factor caused by process variations in the manufacturing process to generate unclonable responses for challenges to authenticate devices. Since the PUF has been introduced [9], it has attracted much attention as a low-cost alternative solution for key generation and device authentication, and hence many different PUF structures have been proposed. PUFs can be broadly categorized into strong PUFs [10]–[14] and weak PUFs [15]–[18]. A weak PUF produces a small amount of stable CRPs that can be used as unique keys or seeds for traditional encryption systems. SRAM PUF [15] and ring oscillator (RO) PUF [16] are typical weak PUFs. Arbiter PUF [11] is a typical strong PUF, the circuit structure is shown in Fig. 1. Strong PUFs are based on their high entropy content and can provide a huge number of unique CRPs to authenticate the device. However, the current strong PUFs are vulnerable to machine learning attacks that attackers can collect a certain number of CRPs to model the PUF easily. For the traditional Arbiter PUF, a cloned model can be built with the prediction accuracy above 95% after only collecting 650 CRPs, which means the cloned model can exhibit the similar challenge-response behavior to the original PUF [19].

Compared with the PUFs, the VOS-based authentication has two advantages [7]: 1) lower power consumption; 2) no additional hardware required. Therefore, the VOS-based authentication is more suitable for resource-constrained IoT applications. VOS is a common power reduction technology...
and can be used for approximate computing [20]. The calculation unit of digital circuits can generate correct results for all inputs under the normal operating voltage, but calculation errors may occur in VOS [21]. Meanwhile, the errors generated by the computing unit in VOS are related to the manufacturing process variation and hence can be used as hardware fingerprints for device authentication. Recently, Arafin, Gao and Qu [7] proposed to use such errors generated by the computing unit in VOS as the device signatures and designed a two-factor authentication protocol, named voltage over-scaling-based lightweight authentication (VOLtA).

This paper proves that the VOLtA is vulnerable to machine learning (ML) attacks. We first report the ML attacks on VOLtA in [19]. In this article, 1) we elaborate the details of ML attacks on VOLtA; 2) In order to resist ML attacks, a challenge self-obfuscation structure (CSoS) is proposed against ML attacks for VOLtA, and it is a general obfuscation method that also can be used to secure Strong PUFs; 3) we verify the effectiveness of proposed ML attacks and defense by HSpice platform using the FreePDK 45nm libraries. The main contributions of this paper are as follows.

1) We reevaluate the security of VOLtA. For the first time, we demonstrate that ML attacks such as artificial neural network (ANN), recurrent neural network (RNN) and covariance matrix adaptation evolution strategy (CMA-ES) can break VOLtA successfully. Especially, the prediction accuracy of RNN is as high as 99.65%.

2) We propose a CSoS-based ML resistant authentication protocol that reduces the prediction accuracy of modeling to less than 51.2%.

3) The VOS-based two-factor authentication scheme requires a very long key to encrypt the output, which incurs unacceptable key storage overhead. Our proposed CSoS-based ML resistant authentication protocol eliminates such weakness.

4) CSoS is not only efficient for VOLtA, but also can be deployed for strong PUFs and exhibits good obfuscation ability. After deploying the CSoS, the modeling accuracy for an Arbiter PUF is below 54% with LR, SVM, ANN, RNN and CMA-ES when 10^6 CRPs are collected.

5) CSoS uses the previous challenges combined with keys or random numbers to obfuscate the current challenge without changing the structure of the authentication circuit such as VOS-adders and PUFs. Therefore, it will not affect the uniqueness and reliability.

The rest of this paper is organized as follows. Section II introduces some related definitions, concepts and terminologies. Section III gives a detailed security analysis for VOLtA and ML attack methods. The CSoS-based ML attacks resistant authentication is elaborated in Section IV. The detailed experimental results are reported in Section V. Finally, we give the conclusion in Section VI.

II. PRELIMINARIES

This section will introduce the principle of generating calculation errors in the VOS circuit and the ML algorithms which are used to model the VOLtA.

A. Voltage Over-scaling

In digital signal processing systems, the power consumption $P$ is given by:

$$P = C_L V_{dd}^2 f_s$$  \hspace{1cm} (1)

where $V_{dd}$ is the supply voltage; $C_L$ is the effective switching capacitance; $f_s$ is the clock frequency of circuit [21]. According to Eqn. (1), the power consumption $P$ decreases with the operating voltage $V_{dd}$. Some techniques employ this feature to reduce the power consumption of circuit, such as multiple supply voltages [22], variable voltage scaling [23] and retiming technique [24]. The circuit delay $\tau_d$ is given by:

$$\tau_d = \frac{C_L V_{dd}}{\beta(V_{dd} - V_t)^\alpha}$$  \hspace{1cm} (2)

where $\alpha$ is the velocity saturation index, $\beta$ is the gate trans-conductance and $V_t$ is the device threshold voltage [7]. We can see from the Eqn. (1) and (2) that power consumption will decrease quadratically and the delay will increase dramatically with the decreasing of supply voltage [25]. With the correct timing constraints, the circuit produces correct outputs for all inputs. However, when the operating voltage is lowered, the timing variations may incur calculation errors. In the approximate computing, the computing unit performs high-bit calculations in the normal voltage and calculates low-bits in VOS to generate approximate results and significantly reduces the power consumption [20, 25, 27]. Furthermore, the errors produced by the process variation are random and can be reproduced by the original device but difficult to be cloned. Therefore, the errors can be used as the hardware fingerprints to authenticate the devices.

B. Computing Errors

As a common computing unit in digital circuits, ripple carry adder (RCA) has the potential to preserve process variation related artifacts [7]. The principle of errors caused by the circuit delay is described in Fig. 2. The gate circuit and truth table of a full-adder (FA) are shown in Fig. 2(a) and Fig. 2(b), respectively. Fig. 2(c) gives the process of generating computing errors, where FA1 is a simplified diagram of Fig. 2(a). For ease of exposition, we assume that the red numbers marked in Fig. 2(a) are the signal transmission delays of the logic gates, and there is no delay in FA2. In Fig. 2(c), when the clock period of the input signal is ‘10’, the first clock period is as follows.

- At time $t = 0$, the input pulse signal $\{C_{in1}, A_1, B_1, A_2, B_2\} = \{1, 1, 0, 0, 0\}$;
- At time $t = 10$, since the delay $D_y = 6 + 5 > 10$ at the $y$-input of OR gate, the signal ‘1’ is not transmitted to $y$-input, hence the signal at $y$-input is still ‘0’. The $x$-input of OR gate delay $D_x = 7 < 10$, the signal ‘0’ is transmitted to $x$-input successfully, and thus the $C_{out1}$-output of OR gate is ‘0’. The output $\{S_1, S_2, C_{out2}\} = \{0, 0, 0\} \neq \{0, 1, 0\}$, the first clock period is over. The second clock period is as follows.
A machine learning model, logistic regression has multiple inputs, which is a binary classification problem. LR is a fast binary classifier that is obtained by inputting the high accuracy [19], [28]. Usually, LR uses the sigmoid function as the activation function.

VONs are related to the current input and the previous inputs. At time $t = 10$, the input pulse signal $\{C_{in1}, A_1, B_1, A_2, B_2\} = \{0, 0, 0, 1, 1\}$; At time $t = 20$, since $D_y = 6 + 5 < 20$, the signal ‘1’ of the first clock period is transmitting in $C_{out1}$, and thus the output $\{S_1, S_2, C_{out2}\} = \{0, 1, 1\} \neq \{0, 0, 1\}$.

As discussed above, the errors produced by the adder in VOS are related to the current input and the previous inputs.

C. Machine Learning

1) Logistic Regression (LR)

In the device authentication, the response bit is ‘0’ or ‘1’, which is a binary classification problem. LR is a fast binary classification algorithm used in machine learning. As a binary classification model, logistic regression has multiple inputs, such as feature vector $X = (x_1, x_2, ..., x_n)$, and the output $Y$ is obtained by inputting $X$ into the classifier. The formula of the classifier is $Y = g(w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n)$. Usually, LR uses the sigmoid function $g(z) = 1/(1 + e^{-z})$ to make $Y$ close to 0 or 1. Arbiter PUFs can be modeled by LR with the high accuracy [19], [28].

2) Support Vector Machines (SVM)

SVM [29] can perform binary classification by mapping known training instances into a higher-dimensional space. The goal of SVM training is to find the most suitable separation hyperplane and solve the nonlinear classification tasks that cannot be linearly separated in the original space. The separation hyperplane should keep the maximum distance from all vectors of different classifications as much as possible. The vector with the smallest distance to the separation hyperplane is called the support vector. The separation hyperplane is constructed by the two parallel hyperplanes with support vectors of different classifications. The distance between the hyperplanes is called the margin. The key of constructing a good SVM is to maximize the margin while minimizing classification errors and the whole process is regulated by the regularization coefficient $\lambda$. In well-trained SVMs, kernel functions are often used to solve the problem of support vector selection and classification difficulties. There are three frequently-used kernel functions: 1) linear: $K(w, z) = z^T w$ (only solves linearly separable problems); 2) radial basis function (RBF): $K(w, z) = \exp((-\|w - z\|^2)/\sigma^2)$; 3) multi-layer perception (MLP): $K(w, z) = \tanh(\alpha^T w + \beta)$. Training a good SVM classifier always requires to adjust regularization coefficient $\lambda$, $\sigma^2$ (RBF) or $(\alpha, \beta)$ (MLP).

3) Artificial Neural Network (ANN)

ANN is interconnected by computational nodes called neurons, which has the adaptive capability. In other words, ANN can adjust the weight parameters utilizing the prepared training set to fit the required function. The universal approximation theorem [30] shows that if a pre-feedback neural network has a linear output layer and at least one hidden layer with an activation function such as sigmoid, it can fit any function with high accuracy as long as there are enough neurons. The simplest neural network comprises of a layer with several neurons, called a single layer perceptron (SLP) [31]. For each neuron, all input vectors are weighted, added, biased, and applied to an activation function to generate an output. In the SLP training process, the neuron updates its weights and bias according to the linear feedback function of the training set prediction error. When the prediction accuracy or iterations of trained model reaches the predetermined value, the training process is terminated. This paper uses a simple 2-layer neural network structure to model the logic gates and the obfuscation mechanism with invariable key, and employs a 3-layer ANN (160 nodes in the first layer, 40 nodes in the second layer and 8 nodes in the third layer) to model VOLTAT. In addition, we use sigmoid as the activation function.

4) Recurrent Neural Network (RNN)

RNN is mainly used to deal with sequence data. In the traditional neural network model, from the input layer through the hidden layer to the output layer, the layers are fully connected and the nodes in the same layer are unconnected. However, such simple neural network structure is difficult to handle sequence data. For example, in natural language processing, it is not enough to comprehend a sentence by understanding its each word. Neural networks are required to process the sequence of these words. The previous input in the sequence will affect the current output, while the
network needs to recall the previous information and apply it to the current output calculation. Therefore, the nodes in the same hidden layer are connected, and the input of the hidden layer includes the input layer and the previous hidden layer. Theoretically, RNN can cope with any length sequence data. However, in order to reduce the complexity, the current output is usually related to the current input and the previous several inputs.

Fig. 3 shows a typical RNN structure. The previous input is forwarded to the next hidden layer through the previous hidden layer. When the n-bit Ripple Carry Adder is modeled, the carry bit from the previous full-adder will be used as the input of next full-adder. In the VOLtA, the current output is related to the previous and current inputs. Therefore, RNN can model n-RCA and VOLtA with the extremely high modeling accuracy. We will discuss the modeling attacks in detail in Section III.

5) Evolutionary Strategies (ES)

ES [32], [33] is a gradient-free stochastic optimization algorithm with invariance under some transformations, parallel scalability and efficient theoretical analysis. It is appropriate for medium-scale complex optimization problems. ES constantly searches for a normal distribution by iterations. Usually, the normal distribution of iterations is written as $N(m, \sigma^2, C)$. $m$ represents the mean of the central position of the distribution; $\sigma$ represents the step size parameter; $C$ represents the covariance matrix. The essence of the ES algorithm is to adjust these three parameters to obtain the best possible search results. How to adjust the step parameters and covariance matrix has a very important impact on the convergence rate of the ES algorithm. The basic idea of adjusting the parameters is to mutate in the direction of the probability of generating a satisfactory solution. The covariance matrix adaptation evolution strategy (CMA-ES) is a global optimization algorithm developed on the basis of evolution strategy (ES) [32]. It combines the reliability and globality of ES with the adaptiveness of covariance matrices, and can solve complex multiple peak optimization problems. Currently, CMA-ES has attracted much attention in the optimization field [33]. In addition, CMA-ES algorithm does not use gradient information in the optimization process. Therefore, as long as the attack model is established, CMA-ES can also effectively attack VOLtA.

III. SECURITY ANALYSIS AND MODELING ATTACKS ON VOLtA

This section will introduce the VOLtA and analyze its security in detail, and finally the several ML algorithms are proposed to model VOLtA.

A. VOLtA

VOLtA is a two-factor authentication scheme, where two factors include a secret key $K$ and the adder that generates errors in VOS (VOS-adder). The authentication protocol is illustrated in Fig. 4. Assume that Alice is the server and Bob is the device that carries an adder. The authentication protocol is divided into two phases. In the registration phase, Bob has an adder and a key $K = (k_1, k_2)$, Alice has a key $K = (k_1, k_2)$ and the model $M$ of Bob’s adder.

is the function of adder in VOS and $distance(L, L')$ can be measured by common distance measurement functions such as Hamming distance or Euclidean distance.

B. Security Analysis for VOLtA

In VOLtA, devices must carry the adder and the correct key $K$, otherwise the authentication would be failure. However, the constant key has low obfuscation ability. In addition, the VOS-adder is vulnerable to ML attacks. Therefore, VOLtA suffers the security issues which will be discussed below.

1) Security Analysis of Constant Key

As shown in Fig. 2(a), the inputs of the full-adder are $\{A_1, B_1, C_{in1}\}$, and the outputs are $\{S_1, C_{out1}\}$. Assume that the key $k_1$ is input to $A_1$ and the random challenge $C$ is input to $B_1$. For 1-bit calculation, the input $A_1$ is unchanged because $k_1$ is constant. We can see from Fig. 2(b), if $A_1 = 0$, then $S_1 = B_1 \oplus C_{in1}$ and $C_{out1} = B_1 \& C_{in1}$; if $A_1 = 1$, then $S_1 = !(B_1 \oplus C_{in1})$ and $C_{out1} = B_1 | C_{in1}$. The full-adder only implements the function of two logic gates after using the constant key $k_1$, which does not increase the difficulty of modeling authentication protocol. We need to model a full-adder without the constant key $k_1$. When the constant key $k_1$ is used, we only need to model the combination of two logic gates. Besides, the VOLtA uses the key $k_2$ to obfuscate the output. In what follows, we will further discuss the obfuscation effectiveness of the key $k_2$.

Assume that $R = L \oplus k_2$, for 1-bit calculation, if $k_2 = 0$, then $R = L$; if $k_2 = 1$, then $R = !L$, which shows that when the output is obfuscated by the constant key, the $i$-th bit output is always unchanged or flipped. For instance, when the adder calculates 4 times, the outputs are $L_{1-4} = \{1011_2, 0011_2, \ldots, 1010_2\}$, the key $k_2 = \{11, 02, \ldots, 18\}$, and the responses $R_{1-4} = \{0100_2, 0011_2, \ldots, 0110_2\}$ after using the XOR obfuscation. Obviously, when the $i$-th bit key $k_{2,i} = 1$, the $i$-th bit response is inverted such as the underlined parts
Adder in VOS

Fig. 5. A calculation example of VOLtA. In (a), the challenge $C$ and the key $k_1$ are calculated by the VOS-adder to generate $L$, then $L$ and the key $k_2$ are XORed to generate response $R$. An example of the computing process is given in (b), in which the red numbers indicate the computing errors. The response example of 5 times authentication is shown in (c). We call the data in red box as horizontal data, and the data in blue box as vertical data.

of $R_{1\sim4}$; when the $i$-th bit key $k_{2,i} = 0$, the $i$-th bit response remains unchanged. We just need to establish a ML model for the $i$-th bit output to implement similar functions.

As analyzed above, the defenses that use constant keys to obfuscate the output is unable to resist ML attacks.

2) Complexity of Challenge-response Mapping

The VOLtA employs the CRPs to authenticate devices. The mapping of challenge-response (CR) depends on the calculation errors generated by a VOS-adder. As long as the effective and enough CRPs are collected, ML algorithms can be modeled using valid CRPs with high prediction accuracy. The mapping of challenge-response (CR) depends on the calculation errors generated by a VOS-adder. As long as the effective and enough CRPs are collected, ML algorithms can be modeled using valid CRPs with high prediction accuracy. However, from the perspective of vertical data, the key used by the $i$-th byte of $C$ is the same for each time, and the calculation of the data in the blue box (see Fig. 5(b) and Fig. 5(c)) uses the same key. Therefore, we can use the data in the blue box to model the operation of its corresponding byte, and the VOLtA can be modeled using valid CRPs with high prediction accuracy.

C. Modeling Attacks on VOLtA

As analyzed above, we need to model the logic gates first. The common logic gates include NOT gate, AND gate, OR gate and XOR gate, where the XOR gate is linearly inseparable and hence it is often used to encrypt information in cryptography. However, the XOR can be implemented by other logic operations. For example,

$$a \oplus b = (a \& b) \oplus (\lnot a \& b)$$ (3)

where ‘!’ is NOT, ‘&’ is AND, ‘|’ is OR and ‘⊕’ is XOR. Besides, NOT, AND, OR and XOR can be approximated as:

$$a \& b \approx f_{\text{and}}(a, b) = \text{sigmoid}(20 \ast a + 20 \ast b - 30)$$ (5)

$$a | b \approx f_{\text{or}}(a, b) = \text{sigmoid}(20 \ast a + 20 \ast b - 10)$$ (6)

$$a \oplus b \approx f_{\text{xor}}(a, b) = f_{\text{or}}(f_{\text{and}}(a, 1 - b), f_{\text{and}}(1 - a, b))$$ (7)

where $\text{sigmoid}(x) = 1/(1 + e^{-x})$, which is a common activation function in the neural network. Substituting Eqns. (4), (5) and (6) into Eqn. (3), the approximate Eqn. (7) for XOR can be obtained. Based on this, we design the neural network structure shown in Fig. 5(c) to model the XOR gate, where $x_1 \approx a \& b$, $x_2 \approx a \& b$ and $y \approx x_1 | x_2$. To model the required functions, we expand the number of neurons in the hidden layer to 10, and set the edges with random weight parameters to model any logic gate; when the obfuscation mechanism which employs the constant key is modeled, the weight of edges is set to the red numbers in Fig. 5(c) and neuron $b$ is set to a random parameter.

The attack model of VOLtA is shown in Fig. 6. Since the current output in VOLtA is related to the current input and the previous input, the input of the model is adjusted to learn the effective mapping between input and output. As shown in Fig. 6(a), the current input is combined with the previous input to create the actual input $X_t = \{x_{t-(m-1)}, \ldots, x_{t-2}, x_{t-1}, x_t\}$, where $m$ denotes the number of input bytes, $x_t$ denotes the timing input, and $x_{t-m,i}$ denotes the $i$-th bit of $t - m$ timing input. We use the vertical data to model the VOLtA. The neural network model of 8-bit Ripple Carry Adder (8-RCA) is shown in Fig. 6(b), the i-th bit of 8-RCA is input to $M_i$, and $M_{i-1}$ serves as the input of $M_i$, which is a typical RNN structure. The XOR obfuscation mechanism is shown in Fig. 6(c). All weight parameters $W$ are random numbers that need to be adjusted.

IV. CHALLENGE SELF-OBFUSCATION STRUCTURE

To resist ML attacks, this paper proposes a challenge self-obfuscation structure (CSoS) against ML attacks. This section will introduce the CSoS and the CSoS-based authentication protocol for VOLtA in detail. In addition, the hardware implementation and security analysis of CSoS for VOLtA and Arbiter PUF will be introduced.

A. The CSoS

The errors generated by the VOS-adder are related to input timing, and the current output is determined by the current input and the previous input. If the correlation among inputs is enhanced or the input is obfuscated, ML modeling attacks would be difficult.

The key idea of CSoS is to combine the previous input with secret keys and random numbers to generate dynamic new keys, and exploit the new keys to obfuscate the current input. The 8-RCA is used as an example, assume that the challenge $C = \{c_1, c_2, \ldots, c_8\}$, the keys are $k_1$ and $k_2$, and the obfuscated challenge $C' = \{c'_1, c'_2, \ldots, c'_8\}$ can be expressed as:
Alice trains an adder with the XOR operation. The length of 8 bits, and $k_2$ is used to obfuscate the intermediate calculation values $G = \{g_1, g_2, \ldots, g_t\}$. For instance, if $k_2 = 10100101$ and $k_{2,i}$ denotes the $i$-th bit of $k_2$. The obfuscation process of CSoS is shown in Fig. 7 where the connection between $c_i$ and $g_i$ indicates XOR, i.e., $g_3$ is connected with $\{c_1, c_3\}$ to indicate that $g_3 = c_1 \oplus c_3$ and $g_4$ is connected with $\{c_1, c_3, c_4\}$ to indicate that $g_4 = c_1 \oplus c_3 \oplus c_4$. Since the attackers do not know the $k_1$ and $k_2$, it is impossible to collect the relevant information of obfuscated challenge $C'$. In the authentication, the obfuscated challenge $C'$ will be transmitted as the real challenge to the adder for calculation. Attackers can only collect the challenge $C$ and the response corresponding to $C'$. In this case, the attackers cannot collect valid CRPs for modeling attacks.

**B. The CSoS-based Authentication Protocol**

We propose a CSoS-based ML attacks resistant authentication protocol for VOLtA. The key $K$ and the VOS-adder are used to authenticate devices. The key $K$ consists of three different keys $k_1$, $k_2$, and $k_3$, where $k_1$ and $k_2$ are used to obfuscate the challenge in CSoS, and $k_3$ has two functions: 1) used as an input of the adder; 2) encrypting the output of adder with the XOR operation. The length of $k_1$ and $k_3$ are 8 bits, and $k_2$ can be any length (in this paper, $k_2$ is set to 8 bits). As shown in Fig. 8 the authentication protocol includes registration and authentication:

**Registration**

i. Alice and Bob obtain the secret key $K = \{k_1, k_2, k_3\}$ through key sharing or other similar methods;

ii. Alice randomly generates an input bitstream $X = \{x_1, x_2, \ldots, x_\omega\}$, where $\omega$ is the number of bytes of $X$,
then sends $X$ to Bob;

iii. Bob adds $x_i$ and $k_3$ using VOS-adder to generate an output bitstream $Y = \{y_1, y_2, ..., y_n\}$, and sends $Y$ to Alice;

iv. Alice uses $X$ and $Y$ to train the adder model of Bob.

**Authentication**

i. Alice generates a random challenge $C = \{c_1, c_2, ..., c_n\}$, and sends it to Bob;

ii. Bob employs CSOs to obfuscate challenge $C$ to get the challenge $C' = \{c'_1, c'_2, ..., c'_n\}$, and adds $c'_1$ and $k_3$ using VOS-adder, then XORs the calculation result and $k_3$ to obtain the response $R = \{r_1, r_2, ..., r_n\}$, and finally $R$ is sent to Alice;

iii. Alice obtains the obfuscated challenge $C'$ through CSOs and $C$, then employs the model $M$ and $k_3$ to generate the response $R'$;

iv. Alice calculates the Hamming distance $HD(R, R')$ between $R$ and $R'$. If the $HD(R, R')$ is greater than the threshold condition, the authentication fails.

**C. Hardware Implementation**

In Eqn. 8, $g_i$ need to be stored temporarily in the calculation. Therefore, we design the input cache structure (ICS), as shown in Fig. 9(a), which consists of some latches and multiplexers (MUXs). A NOR-type latch is used to store 1-bit $g_i$ and the truth table is given in Table I. When $S = R = 0$, the circuit remains in its original state; when $S = 0, R = 1$, regardless of the state of $Q$ and $\bar{Q}$, there will be $Q = 1, \bar{Q} = 0$; when $S = 1, R = 0$, regardless of the state of $Q$ and $\bar{Q}$, there will be $Q = 0, \bar{Q} = 1$. It is worth noting that $S = R = 1$ cannot be employed as an input signal.

![Fig. 9. (a) The 1-bit input cache structure (ICS). (b) An example of ICS.

| $S$ | $R$ | $Q$ | $\bar{Q}$ | $Q'$ | Function |
|-----|-----|-----|----------|------|----------|
| 0   | 0   | 0   | 1        | 0    | Hold     |
| 0   | 0   | 1   | 0        | 1    | Hold     |
| 0   | 1   | 0   | 1        | 1    | Set to 1 |
| 0   | 1   | 1   | 0        | 1    | Set to 1 |
| 1   | 0   | 1   | 0        | 0    | Set to 0 |
| 1   | 0   | 0   | 1        | 0    | Set to 0 |
| 1   | 1   | 0   | 1        | 0    | –        |
| 1   | 1   | 1   | 0        | 0    | –        |

1-bit input cache structure (ICS) is shown in Fig. 9(a), we take the $j$-th bit of $g_i$ as an example $g_{i,j}$, the ICS includes three operations:

- **Read operation (Ro):** The NOR-type latch keeps latching state and outputs $tp$ before calculating $g_{i,j}$, so that $g_{i,j} = c_{i,j} \oplus tp$.

- **Write operation (Wo):** After calculating $g_{i,j}$, the NOR-type latch is released from the latching state and then $g_{i,j}$ is written into the NOR-type latch, i.e., $tp = g_{i,j}$.

- **Hold operation (Ho):** The NOR-type latch holds latching state and outputs $tp$ throughout, $g_{i,j} = c_{i,j} \oplus tp$, and $tp$ keeps unchanged until the next operation is performed.

The read, write and hold operations are controlled by a signal based on the key $k_2$. We assume $k_2 = 10100101$, and the control signals of ICS are 10100101 10100101 ... 10100101. If the control signal is ‘1’, ICS performs the read and write operation; if the control signal is ‘0’, ICS executes the hold operation. We use the NOR-type latch combined with three MUXs to implement these operations. Fig. 9(b) gives a instance of storing $g_i$. Assuming that the single signal duration of $k_2$ is $T$, the ‘1’ port of $MUX_1$ is a periodic signal $ps$ with a period of $T$, and ‘0’ port is a low level signal. In the first half of time $T_1$, $k_2 = 1$, $c_{i,j} = 1, ps$ is connected to the circuit and transferred to $R$; in the second half of time $T_1$, $S = 0, R = 1$, there is $Q(tp) = g_{i,j} = 1$. In addition, when executing a single write operation, we use $MUX_3$ to prevent the updated value of $tp$ from affecting the value of $g_i$ again. Similarly, in the time period $T_3$, $g_{i,j} = 0$ is updated to $Q(tp)$. In this way, we get $G = \{g_1, g_2, ..., g_n\}$ which is obfuscated by the key $k_2$. In the obfuscation process, the CSOs just combines the previous input with keys to obfuscate the current input, and hence does not affect the original uniqueness and reliability of circuit.

As shown in Fig. 9(c), the CSOs proposed in this paper consists of the ICS, the control box and some XOR gates. The key generator is used to generate the key $k_1$ and $k_2$. 

![Fig. 10. (a) The hardware implementation of CSOs. (b) The CSOs for an 8-RCA in VOS. (c) The WCSoS for a 64-bit Arbiter PUF. (d) The TCSoS for a 64-bit Arbiter PUF.](image-url)
for obfuscation. It can be implemented using Weak PUF and True Random Number Generator (TRNG), named Weak PUF-based CSoS (WCSoS) and TRNG-based CSoS (TCSoS) respectively. Fig. [10](b) gives the deployment of CSoS in VOLTA, which is corresponding to Section IV.B. It is worth noting that the CSoS is a universal obfuscation method and hence can also be used for Strong PUFs. In Fig. [10](c) and [10](d), a classic Strong PUF, 64-bit Arbiter PUF, is used as an example to deploy WCSoS and TCSoS. In the WCSoS, $k_1$ and $k_2$ are different keys generated by Weak PUF. In the TCSoS, $k_1$ and $k_2$ are random numbers generated by TRNG. In order to reduce the complexity of authentication, we make $k_1$ equal to $k_2$. Moreover, the TCSoS does not require the key storage and has higher security. For example, if the number of bits in the TRNG is $T_{num} = 4$ and TRNG(4) = 1010, $k_1 = k_2 = \{1010\ 1010\ ...\ 1010\}$ has a total of 64 bits. In authentication, $64 \times 64$-bit challenges are input to the device in the time series to generate 64-bit responses which are sent to the server. Then the server needs to enumerate all the possibilities of $k_1$ and $k_2$ and verifies these responses one by one to authenticate the device (the number of possibilities is $2^{2T_{num}}$).

D. Security Analysis

In this study, we assume that the server is trustworthy and can get the keys and the cloned model stored in the server.

1) Key Security: Our proposed CSoS combines the previous input with the secret keys or random numbers to obfuscate the current input. In the TRNG-based CSoS for Arbiter PUF, if attackers know the cloned model of Arbiter PUF, they can enumerate all the random numbers to clone the authentication protocol. However, the cloned model of Arbiter PUF is securely stored in the server and hence will not be leaked. In the weak PUF-based CSoS, key generator on the device can be implemented with the weak PUF [6], [34], [35]. If attackers get the secret keys, the authentication protocol would be broken. Side-channel attacks are powerful noninvasive attacks that exploit the leakage of physical information when the encryption algorithm is being executed on a system [37]. Several side-channel attacks on weak PUFs have been reported within the past couple of years [38], [39], and most of the authors have pointed out potential countermeasures to their proposed attacks. We don’t propose any solution to prevent side-channel attacks on weak PUFs because it is beyond the scope of this article.

2) Brute Force Attacks: Attackers enumerate the keys and build multiple models to attack. In the weak PUF-based CSoS for VOLTA, assume that the keys $k_1$ and $k_3$ are 8 bits, $k_2$ is $x$ bits, the number of models that attackers need to build to pass the authentication is $2^{(16+x)}$ which is increased exponentially with the increasing of $x$. In the TRNG-based CSoS, the CSoS uses the TRNG to generate keys $k_1$ and $k_2$ ($k_1 = k_2$) to improve security, which only increases the computational overhead of server in authentication. In this case, the number of models that the attackers need to establish is related to the number of collected CRPs. The attackers need to select effective training set in massive data and build an efficient model. Therefore, it is impossible for attackers to clone the CSoS-based authentication by brute-force attacks.

3) Learning-based Attacks: Attackers try to collect large amounts of data to conduct ML attacks. The function of Arbiter PUF can be represented by an additive linear delay model, and the mathematical model of the Arbiter PUF is described in [11], [28]. In this model, we can define the final delay difference $\Delta$ between the upper and the lower path (see Fig. [1]) as:

$$\Delta = \Omega \cdot \Phi(C)$$

where $\Omega = \{\omega^1, \omega^2, ..., \omega^n, \omega^{n+1}\}$, the dimensions of $\Omega$ and $\Phi$ are both $n + 1$. The parameter vector $\Omega$ represents the delay of each stage in an Arbiter PUF; the eigenvector $\Phi(C) = (\phi^1(c), ..., \phi^n(c), 1)^T$ represents a function with the $n$-bit challenge, while $\phi^l(\cdot)$ is a function that can be represented by

$$\phi^l(c) = \prod_{j=l}^{n} (1 - 2c_j), l = 1, ..., n$$

The vector $\Omega$ determines a separate hyperplane in all the eigenvectors by $\Omega \cdot \Phi(C) = 0$. Any challenges have their vectors $\Phi(C)$ located on one side of the hyperplane produce $\Delta < 0$, and on the other side produce $\Delta > 0$. Note that there is non-linear relationship between the challenge $C = (c_1, c_2, ..., c_n)$ and delay difference $\Delta$, but the feature vector $\Phi(C) = (\phi^1(c), ..., \phi^n(c), 1)$ is linearly related to $\Delta$. This makes the application of ML very effective [19], [28], [36].

However, in the CSoS-based Arbiter PUF, the $i$-th timing challenge $C_i = (c_1^i, c_2^i, ..., c_n^i)$, and the final delay difference $\Delta$ can be represented as:

$$\Delta = \Omega \cdot \Phi(C_i')$$

where $\Phi(C_i') = (\phi^1(c_i^1), ..., \phi^n(c_i^n), 1)$ is a feature vector, and

$$\phi^l(c_i^l) = \prod_{j=l}^{n} (1 - 2c_{i,j}^l), l = 1, ..., n$$

according to Eqn. (8),(9) and (10),

$$c_{i,j}^l = k_{1,j} \oplus f(c_{1,j}, k_{2,1}) \oplus f(c_{2,j}, k_{2,2}) \oplus ...$$
$$\oplus f(c_{i-1,j}, k_{2,i-1}) \oplus c_{i,j}$$
$$= \text{Pre}f(x_{i,j} \oplus c_{i,j})$$

where $c_{i,j}$ represents the $i$-th timing and $j$-th bit of challenge. $x \oplus y$ can be expressed by Eqn. (16)

$$x \oplus y = x + y - 2x \cdot y$$

Therefore, the Eqn. (14) for CSoS can be represented as
\[ \phi_i(c'_i) = \prod_{j=1}^{n} (1 - 2c'_{i,j}) \]
\[ = \prod_{j=1}^{n} (1 - 2(\text{Prefix}x_{i,j} + c_{i,j} - 2\text{Prefix}x_{i,j} \cdot c_{i,j})) \]
\[ = \prod_{j=1}^{n} (1 - 2c_{i,j})(1 - 2\text{Prefix}x_{i,j}) \]
\[ = \prod_{j=1}^{n} (1 - 2c_{i,j}) \cdot \prod_{j=1}^{n} (1 - 2\text{Prefix}x_{i,j}) \]

We can see from Eqn. (17), the challenges in \( i \)-th timing are obfuscated by keys and previous challenges (\( \text{Prefix}x_{i,j} \)) in the CSoS. Even if the challenges are same, the generated obfuscated challenges may be different due to the different previous challenges. Furthermore, some previous challenges are hidden by keys and not used to obfuscate the current challenge. In our experiments, RNN fails to attack the CSoS without knowing which previous challenges are used. Therefore, it is difficult for attackers to model it with ML methods due to the high complexity of the obfuscated CRP mapping.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup and Data Collection

We have reproduced the simulation experiments for a 8-RCA circuit in [7] and performed simulations in the HSpice platform using the FreePDK 45nm libraries [40]. The python 3.6.4 programming language and the tensorflow 1.6.0 neural platform using the FreePDK 45nm libraries [40]. The python 3.6.4 programming language and the tensorflow 1.6.0 neural platform using the FreePDK 45nm libraries [40]. The python 3.6.4 programming language and the tensorflow 1.6.0 neural platform using the FreePDK 45nm libraries [40]. The python 3.6.4 programming language and the tensorflow 1.6.0 neural platform using the FreePDK 45nm libraries [40].

We modify the threshold voltages of the NMOS and PMOS models in the FreePDK 45nm libraries to simulate process variations based on the Gaussian Distribution \( \pm 7\% \). The circuit netlist for the 8-RCA is designed by using the modified NMOS and PMOS models at random, and then the circuit simulation is implemented in HSpice, where the simulation temperature is 25°C. We collect the challenge-response pairs (CRPs) generated randomly by this 8-RCA to perform modeling attacks. In addition, we get the first 18 bytes of random challenge \( C \) as vertical data (for the definition of vertical data, see Section III). To get massive vertical data more efficiently, the vertical data is arranged as a bit-stream for collection. In this bit-stream, the signal LOW is maintained for a period of time after completing the 18 bytes calculation, and then the 18 bytes computing is performed again, while massive vertical data can be produced by such a loop.

We use Hspice to simulate 20,000 CRPs for CSoS. We also carry out simulation experiments on WCSoS and TCSoS Arbiter PUF. In our simulation, the delay of the multiplexer of Arbiter PUF is generated by Gaussian Distribution, which follows the well-established linear additive delay model for PUFs [10], [11]. In addition, we simulate the TRNG function with the \texttt{random.randint()} function in Python. \( 10^6 \) CRPs are simulated in the Arbiter PUF experiments.

B. Attacks

ANN, RNN and CMA-ES are used to evaluate the effectiveness of modeling attacks, and RNN is used to attack the VOLtA and the no-key-VOLtA (VOLtA without keys). 20,000 CRPs for VOLtA and no-key-VOLtA are simulated by using HSpice. ML models are trained by using 10,000 CRPs and the rest of 10,000 CRPs are used as the testing set.

1) ML Attacks VOLtA: In the VOLtA, the current output of adder is related to the current input and the previous input. Therefore, the single input consists of multiple bytes, which is recorded as the input \( X_t = \{ x_{t-(m-1)}, \ldots, x_{t-2}, x_{t-1}, x_t \} \). The single output is 1-byte representing the current output of adder. Fig. [11] shows the modeling accuracies of RNN on VOLtA with different input bytes using 10,000 CRPs. We use the Hamming distance to evaluate the modeling accuracy. We can see from Fig. [11] that when \( m = 1 \), only the current input is used as the training input, the prediction accuracy of RNN is only 91.54\%. With the increasing of \( m \), the modeling accuracy is further increased. The prediction accuracy reaches the highest 99.65\% at \( m = 10 \). Therefore, we take \( m = 10 \) to conduct the following experiments.

The results of ML attacks on VOLtA and no-key-VOLtA are shown in Fig. [12]. When the RNN is used to attack the no-key-VOLtA, we collect two inputs of the adder as the challenge. When 500 CRPs are collected, the modeling accuracy of RNN model is more than 90\%; when 10,000 CRPs are collected, the prediction accuracy is up to 99.52\%. Therefore, the no-key-VOLtA is vulnerable to ML attacks. Next, we use ANN,
RNN and CMA-ES to attack VOLtA. Since the output and one input have been obfuscated by the key in VOLtA, we only collect one input of adder as the challenge and the obfuscated output as the response. When 5,000 CRPs are collected, the modeling accuracies of ML attacks reach more than 95%; when collecting 10,000 CRPs, the prediction accuracy of RNN is up to 99.65%. Therefore, the modeling accuracy of RNN for VOLtA is just slightly higher than the no-key-VOLtA. In fact, the adder performs an approximate addition operation in VOS, where response \( R = k_2 \oplus \text{adder}(C, X) \), if \( X \) is an input, attackers can guess \( k_2 \) according to large amounts of \( C, X \) and \( R \); if \( X \) is \( k_1 \), it will reduce the complexity of the model but increase the model security. Besides, the attackers need to collect vertical data to attack VOLtA, which requires to collect more data and consumes more time.

2) **VOLtA Reliability:** The intra Hamming distance (intra HD) of the responses is used to evaluate the reliability of VOLtA. We can see from Fig. 13 that the intra HD is around 0.47% when the temperature decreases from 25°C to 23°C, and it is about 0.62% when the temperature increases from 25°C to 27°C. The prediction accuracy of RNN is 99.65%, while the error generated by the RNN is only 0.35% (see the red dotted line in Fig. 13), which is less than the error caused by \( \pm 2°C \). Unfortunately, the setting of threshold in VOLtA must consider the influence of temperature and other factors on the reliability. When the threshold is determined, the ML models can reach the threshold condition as well. Therefore, the VOLtA is vulnerable to ML modeling attacks.

C. **Defenses**

1) **CSoS for VOLtA:** The effectiveness of CSoS-based ML attacks resistant authentication is evaluated. As shown in Fig. 14(a), we set the input byte \( m = 2, 6, 10, 14, 18 \); the training set is from 50 to 10,000. RNN is used to verify the effectiveness of the proposed protocol, in which the prediction accuracy selects the maximum during training. From the experimental results, we can see that even if the training set or \( m \) is increased, the modeling accuracy is still between 50% and 51.2%. The relationship between the iterations and the modeling accuracy of ML methods is shown in Fig. 14(b). We can see that with the increasing of iterations, the prediction accuracies of ML methods are oscillating around 50.1%. Therefore, the proposed CSoS-based authentication exhibits good resistance to learning-based attacks.

2) **CSoS for Arbiter PUF:** Due to the limited number of CRPs that Hspice can collect, it is impossible to verify in VOLtA whether CSoS can still maintain high resistance to machine learning algorithms in larger data sets. For this reason, under a large data set for CSoS-based Arbiter PUF, we have evaluated the influence of ML attacks. We simulated \( 10^6 \) CRPs to conduct this part of the experiment. Rührmann et al. [41] demonstrates that modeling attacks can work both on simulated and silicon data, and the only difference is the case that the results on simulated data are noise free. However, by using more CRPs in the training stage, results from the real silicon could achieve the same accuracy rate (e.g., 99%)
compare to the simulated data. Furthermore, LR, SVM, ANN, RNN and CMA-ES are used to model WCSoS Arbiter PUF and TCSoS Arbiter PUF. The experimental results are shown in Fig. 15 and Fig. 16.

As shown in Fig. 15, we use ML to attack Arbiter PUF without deploying the obfuscation mechanism. When 5,000 CRPs are collected, the modeling accuracies of ML algorithms are more than 95%; When $10^6$ CRPs are collected, LR can achieve 99.87% modeling accuracy. Obviously, the Arbiter PUF without deploying the defense mechanism can be broken by ML algorithms easily. When ML methods are utilized to model WCSoS Arbiter PUF, the modeling accuracy did not increase significantly as the training set growing. Even if $10^6$ CRPs are collected, the accuracy is still below 54%, which shows that CSOS still maintains good anti-modeling ability under the massive data set. In Fig. 16 we compare the WCSoS Arbiter PUF and TCSoS Arbiter PUF modeling attacks, where a 4-bit TRNG is used in TCSoS. Experimental results that both TCSoS and WCSoS show good resistance to ML attacks.

TCSoS has high flexibility to deploy different levels of TRNG based on its own security requirements and affordable computing power. As shown in Fig. 17 we use LR to model the 64-bit TCSoS Arbiter PUF with TRNG bits $T_{num} = 0, 1, 2, 4, 8, 16$ ($T_{num} = 0$ means TCSoS is not deployed), when $T_{num} = 1$ and 2, the modeling accuracy of LR can reach 74.93% and 60.83% respectively, which does not meet the security requirements; when $T_{num} = 4$ and 8, even if collecting $10^6$ CRPs, the modeling accuracy of LR is still below 54%. It is worth mentioning that when $T_{num} = 16$, LR only has a modeling accuracy of 51.63%. However, the computational cost of server authentication at this time will be $2^{16} = 65,536$ times more than normal conditions. Therefore, $T_{num} = 4$ or 8 is the empirical value we recommended in the actual deployment.

Next, we verify the effectiveness of TCSoS ($T_{num} = 4$) for Arbiter PUF with different stage sizes. As shown in Fig. 18 regardless of the stage size of Arbiter PUF, the modeling accuracy of ML for TCSoS has been reduced and finally stabilized around 54%. Hence, TCSoS provides good obfuscation ability for Arbiter PUFs with different stage sizes. Moreover, we also verify the effectiveness of different ML algorithms on TCSoS-based Arbiter PUF with different $T_{num}$ and $bit_{num}$ (stage size). Table II gives detail experimental data which demonstrate that TCSoS can effectively obfuscate the mapping relationship of CRPs in Arbiter PUF with different stage sizes and shows good resistance to several ML attack methods.

### VI. CONCLUSION

In this paper, we have reevaluated the security of the VOS-based authentication protocol and implemented several high-accuracy ML modeling attacks on VOLTA. Experimental results show that the VOLTA is vulnerable to ML attacks, and the prediction accuracy of RNN is up to 99.65%. To
resist the ML attacks on the VOLtA, this paper proposes a novel challenge self-obfuscation structure (CSoS), which lowers the prediction accuracy of ML on the VOLtA to 51.2%. Furthermore, our proposed CSoS exhibits good obfuscation ability for both VOLtA and strong PUFs. We collect 10^6 CRPs of an Arbitr PUF deployed with CSoS and modeled it using LR, SVM, ANN, RNN and CMA-ES. The experimental results show that modeling accuracy is reduced to 54%.

REFERENCES

[1] Wikipedia, “Internet of things,” [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things
[2] “Smart Summit Asia: Identifying Key Technology Drivers for Wider Adoption of Connected Solutions,” [Online]. Available: https://technology.ihts.org/S87648, 2017.
[3] “DDoS attack that disrupted internet was largest of its kind in history, experts say,” [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, 2016.
[4] M. Antonakakis et al., “Understanding the Mirai Botnet This paper is included in the Proceedings of the Understanding the Mirai Botnet,” USENIX Secur., 2017.
[5] U. Rührmair and D. E. Holcomb, “PUFs at a glance,” in Design, Automation and Test in Europe (DATE), 2014, pp. 1-6.
[6] J. L. Zhang, G. Qu, Y. Q. Lv, and Q. Zhou, “A survey on silicon PUFs and recent advances in ring oscillator PUFs,” J. Comput. Sci. Technol., vol. 29, no. 4, pp. 664-678, 2014.
[7] M. T. Arafin, M. Gao, and G. Qu, “VOLtA: Voltage Over-scaling Based Lightweight Authentication for IoT Applications,” 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, pp. 336-341, 2017.
[8] H. Su and J. Zhang, “Machine Learning Attacks on Voltage Over-scaling-based Lightweight Authentication,” Asian Hardware Oriented Security and Trust Symposium, 2018.
[9] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, “Physical one-way functions,” Science, vol. 297, no.5859, pp.2026-2030, Sep. 2002.
[10] J. W. Lee, Daihyun Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “A technique to build a secret key in integrated circuits for identification and authentication applications,” in 2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525), pp. 176-179.
[11] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “Extracting Keys from Integrated Circuits,” IEEE Trans. Very Large Scale Integr. Syst., vol. 13, no. 10, pp. 1200-1205, 2005.
[12] A. Vijayakumar and S. Kundu, “A Novel Modeling Attack Resistant PUF Design based on Non-linear Voltage Transfer Characteristics,” in Design, Automation And Test in Europe (DATE), 2015, 2015, pp. 653-658.
[13] M. Majzoobi, F. Koushanfar, and M. Potkonjak, “Lightweight secure PUFs,” IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD, vol. 1, no. 1, pp. 670-673, 2009.
[14] D. P. Sahoo, S. Saha, D. Mukhopadhyay, R. S. Chakraborty, and H. Kapoor, “Composite PUF: A new design paradigm for Physically Unclonable Functions on FPGA,” Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust. HOST 2014, pp. 50-55, 2014.
[15] D. E. Holcomb, W. P. Burleson, and K. Fu, “Initial SRAM state as a fingerprint and source of true random numbers for RFID tags,” Proc. Conf. RFID Secur., vol. 58, no. 9, pp. 1-12, 2007.
[16] G. E. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” in 44th ACM/IEEE Design Automation Conference, pp.9-14, 2007.
[17] P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and B. Becker, “Read-Proof Hardware from Protective Coatings,” Cryptogr. Hardware. Embed. Syst., pp. 369-383, 2006.
[18] M. Sauer, P. Riaoli, L. Feiten, B. Becker, U. Rührmair, and I. Polian, “Sensitized path PUF: A lightweight embedded physical unclonable function,” Proc. Des. Autom. Test Eur., pp. 680-685, 2017.
[19] J. Zhang and L. Wan, “CMOS: Dynamic Multi-key Obfuscation Structure for Strong PUFs,” 2018.
[20] A. Agarwal, K. Roy, and A. Raghunathan, “MACACO: Modeling and analysis of circuits for approximate computing,” in IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD, pp. 667-673, 2011.