Voltage Over-scaling-based Lightweight Authentication for IoT Security

Jiliang Zhang, *Senior Member, IEEE,* Chaoqun Shen, Haihan Su, Md Tanvir Arafin, Gang Qu, *Fellow, 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 accuracy is below 54% when 10^6 challenge-response pairs (CRPs) are collected to model the CSoS-based Arbiter PUF with ML attacks based on LR, SVM, ANN, RNN, and CMA-ES.**

I. Introduction

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 increase of IoT devices, security issues have attracted much attention. For example, in 2016, the United States suffered the largest DDoS attack in history from Mirai botnet [3]. The Mirai botnet attack brought down much of the Internet infrastructure of the United States. Mirai is a worm-like family of malware that infected IoT devices and corralled them into a DDoS botnet. 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 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 [4], [5].

Physical unclonable functions (PUFs) [6] and recently proposed voltage over-scaling (VOS) based authentication [7] are two emerged lightweight security primitives for IoT device authentication [8]. PUFs use a random factor caused by process variations in the manufacturing process to generate unclonable responses for challenges to authenticate devices. VOS is a common power reduction technology and can be used for approximate computing [9]. 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 [10]. 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. We recently proposed using such errors generated by the computing unit in VOS as the device signatures and designed a two-factor authentication protocol [7]. Compared with the PUFs, the VOS-based authentication has two advantages: 1) lower power consumption; 2) no additional hardware is required for its implementation. Therefore, VOS-based authentication is more suitable for resource-constrained IoT applications. However, we prove that the VOLtA is vulnerable to machine learning (ML) attacks [11].

This paper is the extension of our previous conference papers [7], [11]. 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) the CSoS-based authentication protocol is proposed; 4) we verify the effectiveness of proposed ML attacks and defense strategies using HSpice simulations. Many new experimental results (Figures 13-18, Table III-V) are added for current submission. The main contributions of this paper are as follows.

1) We reevaluate the security of VOLtA. We demonstrate that ML attacks such as an artificial neural network (ANN), recurrent neural network (RNN), and covariance matrix adaptation evolution strategy (CMA-ES)

This work is supported in part by the National Natural Science Foundation of China under Grant No. U20A20202 and 61874042, the Hunan Natural Science Foundation for Distinguished Young Scholars under Grant No. 2020JJ2010, the Hu-Xiang Youth Talent Program under Grant No. 2018RS3041, the Key Research and Development Program of Hunan Province under Grant No. 2019GK2082. (*Corresponding author: Jiliang Zhang*)

J. Zhang, C. Shen and H. Su are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

M. T. Arafin is with Electrical and Computer Engineering Department, Morgan State University, Baltimore, Maryland 21251, USA (e-mail: mdtanvirarafin@morgan.edu)

G. Qu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: gangqu@umd.edu).

TABLE I
LIST OF PARAMETERS

Symbol	Description
X	The set of challenges used for VOLtA
Y	The set of responses used for VOLtA
C	The set of challenges used for adder, CSoS, PUF
R	The set of responses used for adder, CSoS, PUF
K	The set of keys used for VOLtA, CSoS
t	The timing of challenges
m	The number of input bytes
M	Trained model
w	The number of data collected by the server for training to obtain the soft model
G	The intermediate calculation values of CSoS
tp	The iterative part of CSoS
T_{num}	The number of TRNG bits

can break VOLtA successfully. Especially, the prediction accuracy of RNN is up to 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. The 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 source code to reproduce the experiment is available online at http://hardwaresecurity.cn/CSoS-Code.zip

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 some terminologies and concepts used in this paper. The symbols and terminology used in this paper are shown in Table I.

A. Physical unclonable functions

Over the last decades, many different PUF structures have been proposed and can be broadly categorized into strong PUFs [12]–[15] and weak PUFs [16]–[18]. A weak PUF, such as an SRAM PUF [16] or a ring oscillator PUF [17], produces a small amount of stable challenge-response pairs (CRPs) that can be used as unique keys or seeds for traditional encryption systems. On the other hand, strong PUFs, such as an Arbiter PUFs [13], can provide a huge number of unique CRPs to authenticate the device. However, the current strong PUFs are vulnerable to ML attacks that attackers can collect a certain number of CRPs from modeling the PUF easily [19]–[23]. For example, Delvaux attacked five authentication protocols based on PolyPUF, OB-PUF, RPUF, LHS-PUF, and PUF-FSM [22]. Shi et al. [23] proposed approximate attacks that can model nonlinear strong PUFs with high accuracies.

In recent years, many defenses have been proposed to resist ML attacks [24]. These defenses can be classed into structural non-linearization, and CRP obfuscation [25]. The structural non-linearization adds nonlinear elements to the PUF structure to resist ML attacks [12], [26]. However, the use of nonlinear elements also dramatically decreases the reliability of PUF. CRP obfuscation can hide the mapping of CRPs to prevent attackers from collecting valid CRPs to model strong PUFs. Several obfuscation methods have been proposed and can be classified into three categories: XOR gates [13], hash functions [27], [28] and random bits [29], [30]. However, some heavy obfuscation structures (e.g., hash function) are added to the PUF structure for obfuscation, which incurs high hardware overhead. In addition, the hardware overhead of error correction on the responses would be high [20].

B. Voltage Over-scaling

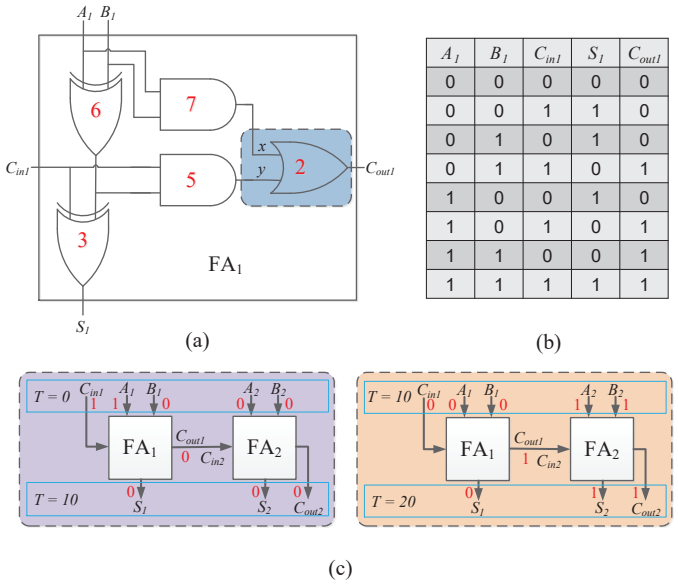
In digital signal processing systems, the power consumption P is given by:

$$P = C_L V_{dd}^2 f_s \tag{1}$$

where V_{dd} is the supply voltage; C_L is the effective switching capacitance; f_s is the clock frequency of circuit [10]. 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 [31], variable voltage scaling [32] and retiming technique [33]. The circuit delay τ_d is given by:

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

where α is the velocity saturation index, β is the gate transconductance, and V_t is the device threshold voltage [7]. We can see from Eqn. (1) and (2) that power consumption will decrease quadratically and the delay will increase dramatically with the lowering of supply voltage [34]. With the correct timing constraints, the circuit produces correct outputs for all inputs. However, when the operating voltage is lowered, the timing violations may incur calculation errors. In 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 [9], [35]. Furthermore, the errors produced by the process variation are random and can be reproduced by the original device but difficult to clone. Therefore, the errors can be used as the hardware fingerprints to authenticate the devices.



A_1	B_1	C_{in1}	S_1	C_{out1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig. 1. An example of computing error. (a) The gate circuit of a full-adder. (b) The truth table of the full-adder. (c) The generation process of computing errors. A n-bit RCA is connected by n full-adders.

C. 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. 1. The gate circuit and truth table of a full-adder (FA) are shown in Fig. 1(a) and Fig. 1(b), respectively. Fig. 1(c) gives the process of generating computing errors, where FA$_1$ is a simplified diagram of Fig. 1(a). For ease of exposition, we assume that the red numbers marked in Fig. 1(a) are the signal transmission delays of the logic gates, and there is no delay in FA$_2$. In Fig. 1(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.

- 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 $t = 20$, the delay since $D_y = 6 + 5 < 20$. Since the period of the signal is 10, 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.

D. 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 $sigmoid$ $g(z) = 1/(1 + e^{-z})$ to make Y close to 0 or 1. Arbiter PUFs can be modeled by LR with the high prediction accuracy [36].

2) *Support Vector Machines (SVM)*

SVM [37] 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 to constructing a good SVM is to maximize the margin while minimizing classification errors, and the whole process is regulated by the regularization coefficient.

3) *Artificial Neural Network (ANN)*

ANN is interconnected by computational nodes called neurons, which has adaptive capability. In other words, ANN can adjust the weight parameters utilizing the prepared training set to fit the required function. The simplest neural network comprises a layer with several neurons, called a single-layer perceptron (SLP) [38]. All input vectors are weighted, added, biased, and applied to an activation function to generate an output for each neuron. 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 the 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 keys, 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 VOLtA. 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 a 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 each word. Neural networks need 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

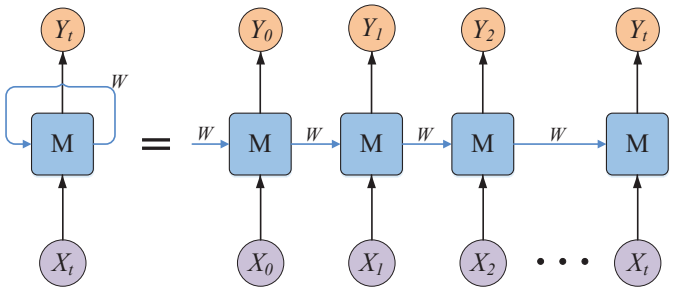


Fig. 2. The structure of recurrent neural network

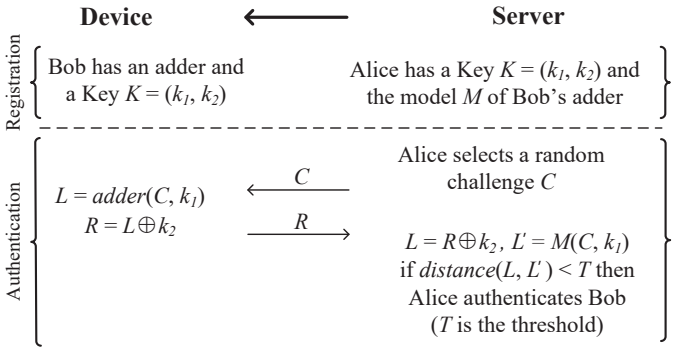


Fig. 3. The voltage over-scaling-based lightweight authentication protocol [7], where $adder()$ is the function of adder in VOS and $distance(L, L')$ can be measured by common distance measurement functions such as Hamming distance or Euclidean distance.

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.

Fig. 2 shows a typical RNN structure. In n-RCA and VOLtA, the current output is related to the previous and current inputs. Therefore, RNN can model n-RCA and VOLtA with high modeling accuracy. We will discuss the modeling attacks in detail in Section III.

5) *Evolutionary Strategies (ES)*

ES [39] is a gradient-free stochastic optimization algorithm with invariance under some transformations, parallel scalability, and sufficient theoretical analysis. ES constantly searches for a normal distribution by iterations. It is appropriate for medium-scale complex optimization problems. The covariance matrix adaptation evolution strategy (CMA-ES) is a global optimization algorithm developed on the basis of evolution strategy (ES) [39]. It combines the reliability and globality of ES with the adaptiveness of covariance matrices, and can solve complex multiple peak optimization problems. 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, several ML algorithms are proposed to model VOLtA.

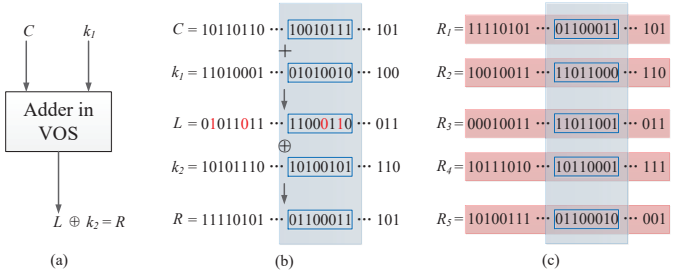


Fig. 4. A calculation example of VOLtA. In (a), the challenge C and the key k_1 are calculated by the VOS-adder to generate L, then L and the key k_2 are XORed to generate response R. An example of the computing process is given in (b), in which the red numbers indicate the computing errors. The response example of 5 times authentication is shown in (c). We call the data in the red box horizontal data, and the data in the blue box vertical data which represents the challenge of the same position at different timings.

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. 3. 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 and the adder model M of Bob. In the authentication phase, 1) Alice generates a random challenge C and sends it to Bob; 2) Bob calculates $L = adder(C, k_1)$ using the VOS-adder, then computes $R = L \oplus k_2$, and sends R to Alice; 3) Alice calculates $L = R \oplus k_2$ and $L' = M(C, k_1)$. If the difference between L and L' meets the threshold condition, Alice authenticates Bob.

In VOLtA [7], some images are used as challenges, assume that the length of the random challenge C is $8{\times}n$ bits, the K is $16{\times}n$ bits (k_1 and k_2 are both $8{\times}n$ bits), which incurs unacceptable key storage overhead. For example, if a $52{\times}40$ pixels image is used as the challenge for authentication, the required key K will be $16{\times}52{\times}40$ = 33,280 bits. The proposed CSoS-based ML resistant authentication protocol in this paper eliminates such weakness.

B. Security Analysis for VOLtA

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

1) *Security Analysis of Constant Key*

As shown in Fig. 1(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. 1(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

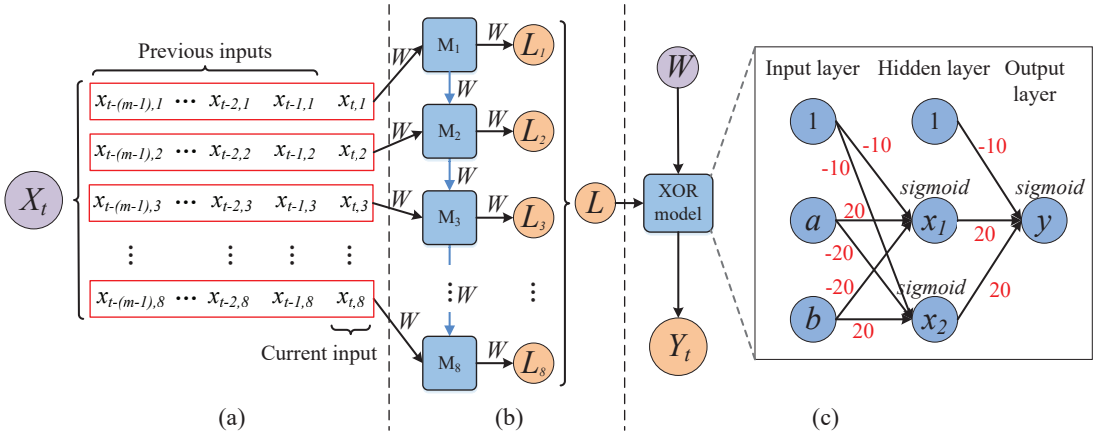


Fig. 5. The attack model of VOLtA.

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 of challenge, 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 $L_{1\sim4} = \{1011_1, 0011_2, ..., 1010_8\}$, the key $k_2 = \{1_1, 0_2, ..., 1_8\}$, and the responses $R_{1\sim4} = \{\underline{0100}_1, 0011_2, ..., \underline{0101}_8\}$ 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 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 cannot resist ML attacks.

2) *Complexity of Challenge-response Mapping*

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 effective and enough CRPs are collected, ML algorithms can model the VOS-adder to simulate its CR behavior. In what follows, we discuss the complexity of CR mapping.

An example calculation for VOLtA is illustrated in Fig. 4. The adder performs each addition and XOR operation with the corresponding k_1 and k_2. Therefore, the horizontal data are obfuscated by different keys so that horizontal data cannot be used to train the model with high 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. 4(b) and Fig. 4(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,

$$a \oplus b = (a\&!b)|(!a\&b) \tag{3}$$

where '!' is NOT, '&' is AND, '|' is OR and '$\oplus$' is XOR. Besides, NOT, AND, OR, and XOR can be approximated as:

$$!a = 1 - a \tag{4}$$

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

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

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

where $sigmoid(x) = 1/(1 + e^{-x})$, which is a common activation function in the neural network. Substituting Eqn. (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 that 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. 5. 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. 5(a), the current input is combined with the previous input to create the actual input $X_t = \{x_{t-(m-1)}, ..., x_{t-2}, x_{t-1}, x_t\}$, where m denotes the number of input bytes, x_t denotes t timing input, and $x_{t-m,i}$ denotes the i-th bit of $t-m$ timing input. The horizontal data of X_t is the value of the same bit at different input timing, which will be obfuscated by the same keys, and the vertical data is the value of the input at the same

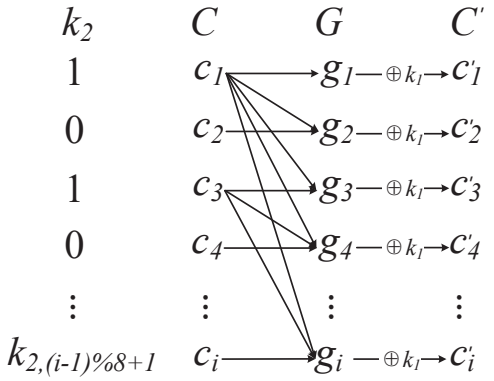


Fig. 6. The obfuscation process of CSoS

timing, which will be obfuscated by different keys. Therefore, the data in the red box is obfuscated with the same key, which is the "vertical data" mentioned in Section III.B.2), so we use it for modeling. Fig. 5(b) shows the neural network model of 8-RCA. In this model, the challenge obfuscated by the i-th key and the output of model M_{i-1} is the input of model M_i, which is a typical RNN structure. Fig. 5(c) is the XOR obfuscation mechanism described earlier. Weight parameters W are random numbers that need to be adjusted, and it does not mean that all parameters W are equal.

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. Additionally, 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. For an 8-RCA, assume that the challenge is $C = \{c_1, c_2, ..., c_t\}$, the keys are k_1 and k_2, and the obfuscated challenge is $C' = \{c'_1, c'_2, ..., c'_t\}$. Then, we can write

$$c'_i = k_1 \oplus g_i \tag{8}$$

$$g_i = f(c_1, k_{2,1}) \oplus f(c_2, k_{2,2}) \oplus ... \oplus f(c_{i-1}, k_{2,(i-2)\%8+1}) \oplus c_i \tag{9}$$

$$f(x, y) = \begin{cases} x, & \text{if } y = 1 \\ 00...00, & \text{if } y = 0 \end{cases} \tag{10}$$

In Eqn. (9), an 8-bit key k_2 is used to obfuscate the intermediate calculation values $G = \{g_1, g_2, ..., g_t\}$. For instance, if $k_2 = 10100101$ ($k_{2,i}$ denotes the i-th bit of k_2). The obfuscation process of CSoS is shown in Fig. 6, where C to G represent Eqn. (9), and G to C' represent Eqn. (8). When calculating g_i, if $k_{2,j} = 1$ ($j < i$), then c_j is selected to be XORed with c_i. For example, g_4, where $k_{2,1} = 1$, $k_{2,2} = 0$, $k_{2,3} = 1$, so c_1 and c_3 is chosen to be XORed with c_4 (j = 1,2,3, $i = 4$). The connection between c_i and g_i in the figure indicates XOR, i.e., the connection between g_4 and $\{c_1, c_3, c_4\}$ represents $g_4 = c_1 \oplus c_3 \oplus c_4$, which is consistent with the situation in Eqn. (9). Since the attackers do not know the k_1 and k_2, it is impossible to collect the relevant information of the obfuscated challenge C'. During 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.

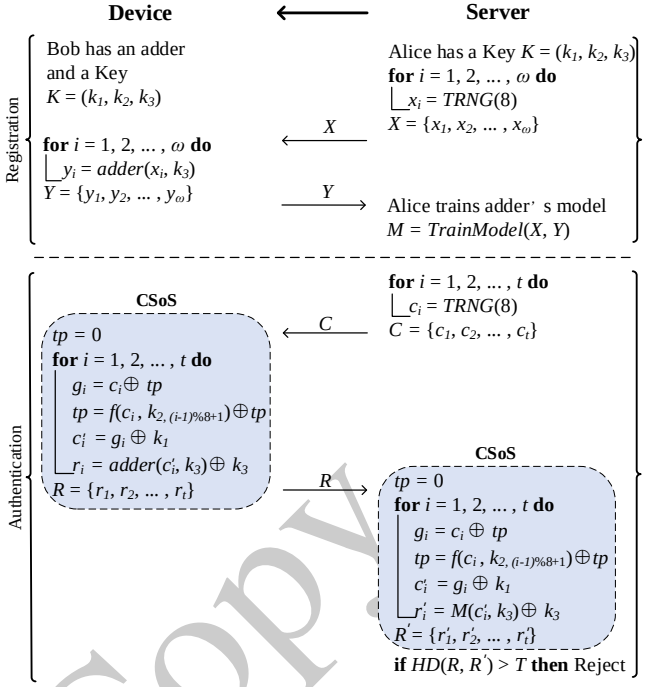


Fig. 7. The CSoS-based ML attacks resistant authentication protocol.

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) it is used as an input of the adder; 2) it encrypts 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. 7, the authentication protocol includes registration and authentication. To express the obfuscation mechanism in time sequence, Eqn. (8) - (10) are re-expressed in a "for" loop. At each time sequence, $f(c_i, k_{2,(i-1)\%8+1})$ is iteratively XORed first and then XORed with c_i, so g_i is expressed as $g_i = c_i \oplus tp$, $tp = f(c_i, k_{2,(i-1)\%8+1}) \oplus tp$. Next,

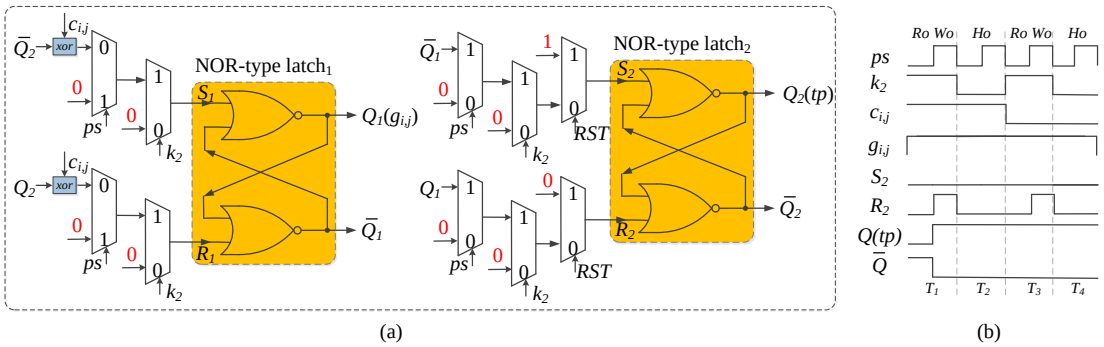


Fig. 8. (a) The 1-bit input cache structure (ICS), ps is a periodic signal and RST is used for circuit initialization. (b) An example of ICS.

we will introduce the registration and authentication process in detail.

Registration

i. Alice and Bob obtain the secret key $K = \{k_1, k_2, k_3\}$;
ii. Alice randomly generates an input bitstream $X = \{x_1, x_2, ..., x_\omega\}$, where ω 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 $Y = \{y_1, y_2, ..., y_\omega\}$, 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 $C = \{c_1, c_2, ..., c_t\}$, and sends it to Bob;
ii. Bob employs CSoS to obfuscate challenge C to get the challenge $C' = \{c'_1, c'_2, ..., c'_t\}$, and adds c'_i and k_3 using VOS-adder, then XORs the calculation result and k_3 to obtain the response $R = \{r_1, r_2, ..., r_t\}$, 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 attacker does not know the adder model, the $HD(R, R')$ is greater than the threshold condition, and the authentication fails.

C. Hardware Implementation

In Eqn. (8), g_i need to be stored temporarily in the calculation for iterative obfuscation. Therefore, we design the input cache structure (ICS), as shown in Fig. 8(a), which consists of some latches and multiplexers (MUXs). NOR-type latch$_1$ is used to store 1-bit g_i and NOR-type latch$_2$ is used to store tp. The truth table is shown in Table II. When $S = R = 0$, the circuit remains in its original state; when $S = 0$ and $R = 1$, regardless of the state of Q and $\overline{Q}$, there will be $Q = 1$ and $\overline{Q} = 0$; when $S = 1$ and $R = 0$, regardless of the state of Q and $\overline{Q}$, there will be $Q = 0$ and $\overline{Q} = 1$. It is worth noting that $S = R = 1$ cannot be employed as an input signal.

1-bit ICS is shown in Fig. 8(a). We take the j-th bit $g_{i,j}$ of g_i as an example, the ICS includes three operations:

TABLE II
THE TRUTH TABLE FOR NOR-TYPE LATCH

S	R	Q	$\overline{Q}$	Q'	$Functiong$
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	0	1	0	Set to 0
1	0	1	0	0	Set to 0
1	1	0	1	–	–
1	1	1	0	–	–

- **Read operation (Ro):** NOR-type latch$_2$ keeps latching state and outputs tp. NOR-type latch1 obfuscates the intermediate calculated value according to k_2, and $g_{i,j} = c_{i,j} \oplus tp$ can be obtained.
- **Write operation (Wo):** After calculating $g_{i,j}$, NOR-type latch$_1$ keeps latching state and stores $g_{i,j}$. At the same time, NOR-type latch$_2$ is released from the latching state and then $g_{i,j}$ is written into NOR-type latch$_2$, i.e., $tp = g_{i,j}$.
- **Hold operation (Ho):** NOR-type latch$_1$ and NOR-type latch$_2$ hold latching while keeping $g_{i,j}$ and tp unchanged until the next operation is performed. So far, a obfuscation iteration is completed.

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. At the beginning of a round of encryption, $tp = 0$ when the first bit is encrypted, therefore the RST signal is set to '1', and then the RST signal is set to 0 to make tp work normally. We use two NOR-type latches combined with some MUXs to implement these operations. Fig. 8(b) gives a instance of storing g_i. Assuming that the single signal duration of k_2 is T. We first set tp to '0' by RST. In the first half of time T_1, NOR-type latch$_1$ works for read operation, where $c_{i,j} = 1, ps = 0$ and $k_2 = 1$. It will choose the upper route in the first MUX ($ps = 0$) and also chooses the upper route in the second MUX ($k_2 = 1$), then $Q_1(g_{i,j}) = c_{i,j} \oplus tp = 1$. In the second half of time T_1, NOR-type latch$_2$ works for write operation, where $ps = 1, k_2 = 1$ and $RST = 0$. It will choose the upper route in the first MUX ($ps = 0$) and in the second

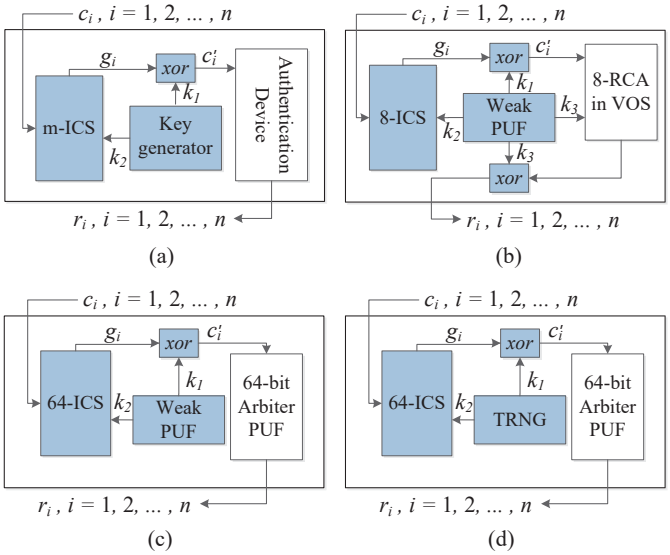


Fig. 9. (a) The hardware implementation of CSoS. (b) The CSoS for an 8-RCA in VOS. (c) The WCSoS for a 64-bit Arbiter PUF. (d) The TCSoS for a 64-bit Arbiter PUF.

MUX ($k_2 = 1$), and choose the lower route in the third MUX ($RST = 0$), then $S_2 = 0, R_2 = 1$ and $Q_2(tp) = g_{i,j} = 1$. Repeating this operation, we get $G = \{g_1, g_2, ..., g_t\}$ 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(a), the CSoS proposed in this paper consists of structures with low hardware overhead, such as the ICS, the key generator, and some XOR gates. The key generator is used to generate the key k_1 and k_2 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. 9(b) gives the deployment of CSoS in VOLtA, which corresponds 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. 9(c) and 9(d), the classic Strong PUF, 64-bit Arbiter PUF, is taken as an example to deploy WCSoS and TCSoS. In the WCSoS, k_1 and k_2 are different keys generated by the Weak PUF. In the TCSoS, k_1 and k_2 are random numbers generated by the TRNG, where e-fuse technology is used. In order to reduce the complexity of authentication, we make k_1 equal to k_2. Moreover, TCSoS has higher security. For example, if the number of bits in the TRNG is $T_{num} = 4$ and TRNG(4) = 1010, then $k_1 = k_2 = \{\underline{1010}\ \underline{1010}\ ...\ \underline{1010}\}$ has a total of 64 bits. In authentication, 64×64-bit challenges are input to the device in the time series to generate responses, which are sent to the server. Then the server enumerates all the possibilities of k_1 and k_2 and verifies these responses to authenticate the device (the number of possibilities is $2^{T_{num}}$).

D. Security Analysis

In this paper, i) we assume that the server is trusted. In this case, the attacker cannot obtain the key and clone model stored in the server; ii) the attacker can only collect the data transmitted between the server and the device in the protocol, namely C and R.

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. 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 [40]. Several side-channel attacks on weak PUFs have been reported within the past couple of years [41], [42], 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) Replay Attacks: One of advantages for TCSoS is the obfuscation key can be updated. Therefore, it is meaningless to guess the key for a round of process, only WCSoS is discussed here. As shown in Fig. 6, the attacker sets the challenge c_i of a time sequence to a non-zero challenge, and sets the challenges c_j ($i \neq j$) of all other time sequences to zero challenges. If $k_{2,i}$ = 1, when $1 \leq s < i$, g_s = 0; when $i \leq s \leq t$, $g_s = c_i$. If $k_{2,i}$ = 0, when $s = i$, $g_s = c_i$; when $s \neq i$, g_s = 0. This makes r_i only have two modes: if $g_s = c_i$, then $r_s = adder(g_s \oplus k_1, k_3) \oplus k_3$; if g_s = 0, then $r_s = adder(k_1, k_3) \oplus k_3$. Therefore, the attacker can infer g and then recover k_2 based on the composition of r. However, k_1 and k_3 cannot be inferred in a similar way because k_1 and k_3 have no connection with challenges. Most important of all, the characteristic of setting the challenge of a time sequence to non-zero and the challenges of other time sequences to zero challenges is obvious and hence can be easily detected. If such special challenge mode is detected, the system will reject the authentication.

4) 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 [13], [36]. In this model, we can define the final delay difference Δ between the upper and the lower path as:

$$\Delta = \Omega \cdot \Phi(C) \tag{11}$$

where $\Omega = \{\omega^1, \omega^2, ..., \omega^n, \omega^{n+1}\}$, the dimensions of Ω and Φ are both $n+1$. The parameter vector Ω 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

$$\phi^l(c) = \prod_{j=l}^{n}(1 - 2c_j), l = 1, ..., n \tag{12}$$

The vector Ω 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 nonlinear relationship between the challenge $C = (c_1, c_2, ..., c_n)$ and delay difference Δ, but the feature vector $\Phi(C) = (\phi^1(c), ..., \phi^n(c), 1)$ is linearly related to Δ. This makes the application of ML very effective [36].

However, in the CSoS-based Arbiter PUF, the i-th timing challenge $C'_i = (c'_{i,1}, c'_{i,2}, ..., c'_{i,n})$, and the final delay difference Δ can be represented as:

$$\Delta = \Omega \cdot \Phi(C'_i) \tag{13}$$

where $\Phi(C'_i) = (\phi^1(c'_i), ..., \phi^n(c'_i), 1)$ is a feature vector, and

$$\phi^l(c'_i) = \prod_{j=l}^{n}(1 - 2c'_{i,j}), l = 1, ..., n \tag{14}$$

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

$$\begin{aligned} c'_{i,j} &= k_{1,j} \oplus f(c_{1,j}, k_{2,1}) \oplus f(c_{2,j}, k_{2,2}) \oplus ... \\ &\quad \oplus f(c_{i-1,j}, k_{2,i-1}) \oplus c_{i,j} \\ &= Prefix_{i,j} \oplus c_{i,j} \end{aligned} \tag{15}$$

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

$$x \oplus y = x + y - 2x \cdot y \tag{16}$$

Therefore, Eqn. (14) for CSoS can be represented as

$$\begin{aligned} \phi^l(c'_i) &= \prod_{j=l}^{n}(1 - 2c'_{i,j}) \\ &= \prod_{j=l}^{n}(1 - 2(Prefix_{i,j} + c_{i,j} - 2Prefix_{i,j} \cdot c_{i,j})) \\ &= \prod_{j=l}^{n}(1 - 2c_{i,j})(1 - 2Prefix_{i,j}) \\ &= \prod_{j=l}^{n}(1 - 2c_{i,j}) \cdot \prod_{j=l}^{n}(1 - 2Prefix_{i,j}) \end{aligned} \tag{17}$$

We can see from Eqn. (17), the challenges in i-th timing are obfuscated by keys and previous challenges ($Prefix_{i,j}$) 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 CRPs mapping.

TABLE III
PARAMETERS USED FOR SIMULATIONS

Parameter Name	Value(s)
Supply voltage (V_{DD})	0.4V/0.45V/1V
NMOS threshold voltage	0.322±0.02415V
PMOS threshold voltage	-0.302±0.02265V
Operating temperature	25 deg. C
Clock Period	1ns

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 [43]. The python 3.6.4 programming language and the TensorFlow 1.6.0 neural network toolkit are used to conduct modeling attacks. All experiments are conducted on the Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz, 8G RAM and GeForce GT 720 GPU.

We use Hspice to simulate 20,000 CRPs for CSoS. As shown in Table III, to simulate process variations, we modify the threshold voltages of the NMOS and PMOS models in the FreePDK 45nm libraries based on the Gaussian Distribution $\pm 7.5\%$. 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. The CRPs are generated randomly by an 8-RCA to perform modeling attacks.

We also carry out simulation experiments on WCSoS and TCSoS Arbiter PUF. In our simulation, the delay of the multiplexer segment of Arbiter PUF is generated by Gaussian Distribution, which follows the well-established linear additive delay model for PUFs [13]. In addition, we simulate the TRNG function with the $random.randint()$ function in Python. 10^6 CRPs are simulated in the Arbiter PUF experiments. LR, SVM, ANN, RNN and CMA-ES attacks are used in the experiment. When the prediction accuracy of the model is not improved for a long time, the model training stops. In particular, CMA-ES will delay one day on this basis.

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 on VOLtA: In 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 $X_t = \{x_{t-(m-1)}, ..., x_{t-2}, x_{t-1}, x_t\}$.

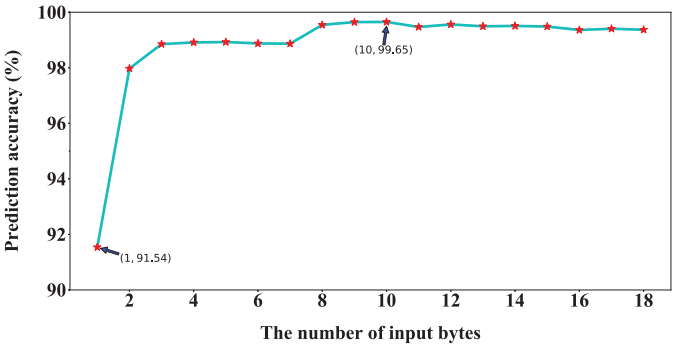


Fig. 10. Modeling accuracies of RNN on VOLtA with different numbers of input elements using 10,000 CRPs.

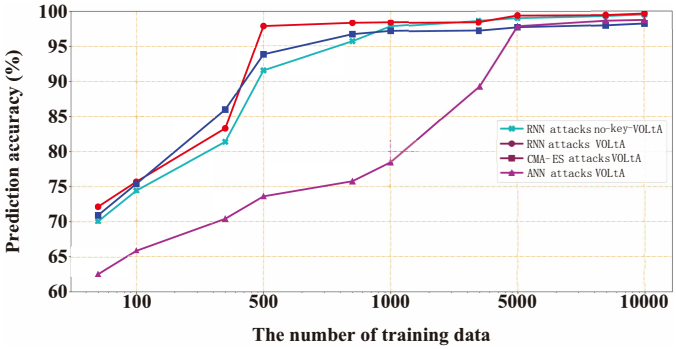


Fig. 11. Modeling accuracies for VOLtA and no-key-VOLtA using 10,000 CRPs.

The single output is 1-byte representing the current output of adder. Fig. 10 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. 10 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. 11. 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 key in VOLtA has obfuscated the output and one input, we only collect one input of the adder as the challenge and the obfuscated output as the response. When 5,000 CRPs are collected, the modeling accuracy of ML attacks reaches 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. The adder performs an approximate addition operation in VOS, where response $R = k_2 \oplus adder(C, X)$, 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.

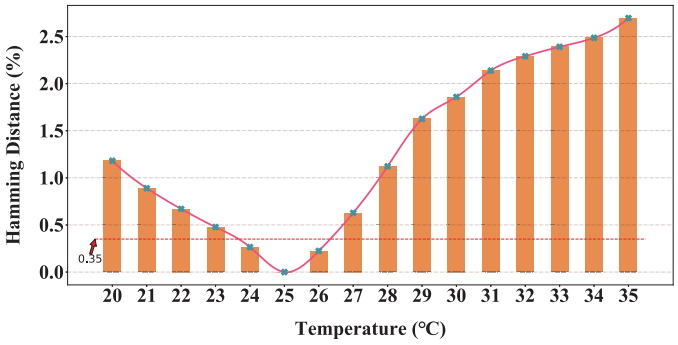


Fig. 12. Reliability impacted by temperature variation (nominal temperature is 25°C).

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. 12 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. 12), which is less than the error caused by ±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. 13(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. 13(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. Rührmair et al. [44] 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,

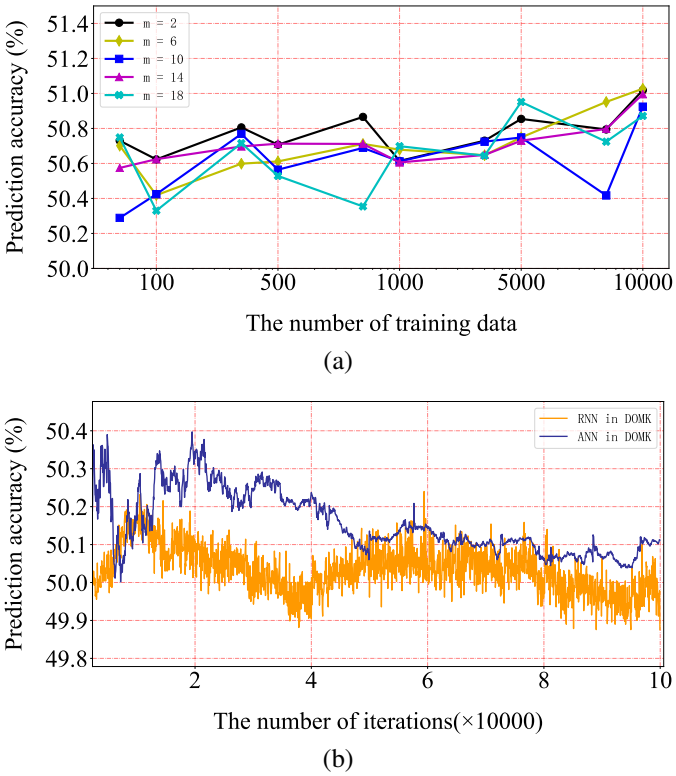


Fig. 13. The effectiveness of CSoS-based ML attacks resistant authentication.

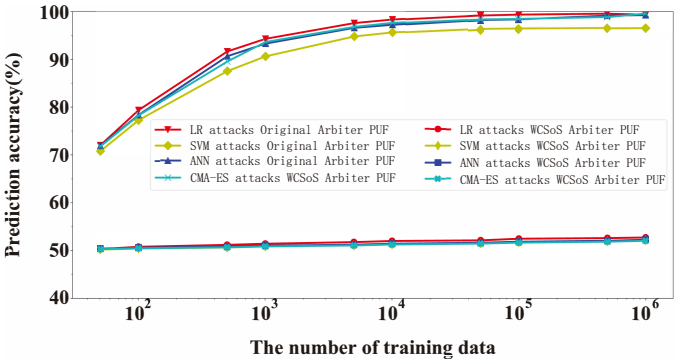


Fig. 14. Modeling accuracies on the 64-bit Original and WCSoS Arbiter PUF using 1 million CRPs.

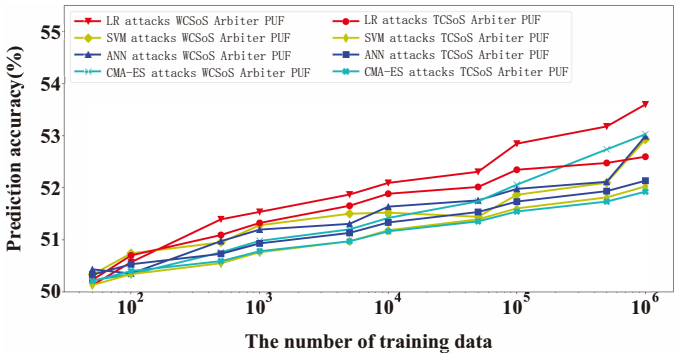


Fig. 15. Modeling accuracies on the 64-bit WCSoS and TCSoS Arbiter PUF using 1 million CRPs.

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. 14 and Fig. 15.

As shown in Fig. 14, 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. 15, 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. 16, 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. 17, 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). The experimental data in Table IV shows that the proposed TCSoS exhibits high ability against several ML attacks and hence it can effectively obfuscate the mapping relationship of PUF CRPs with different stage sizes. Considering the number of IoT devices and the cost of authentication, we recommend that the length of challenges and responses is 128. In authentication, usually about 3% of errors are allowed, but as can be seen from these data, the prediction accuracy of models (including RNN) after deploying TCSoS with $T_{num} = 8, 16$ is about 52%, which obviously cannot meet this standard.

Moreover, approximate attacks [23] is a general framework for ML attacks on strong PUF. In CSoS scheme, the ICS, key generator and XOR unit are used to obfuscate the input.

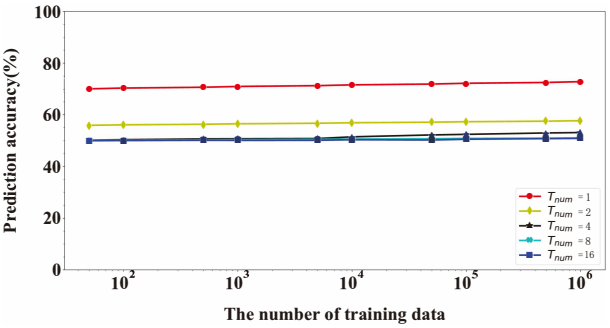


Fig. 16. LR attack results on 64-bit TCSoS Arbiter PUF with different number of T_{num} (T_{num} is the bit number of TRNG).

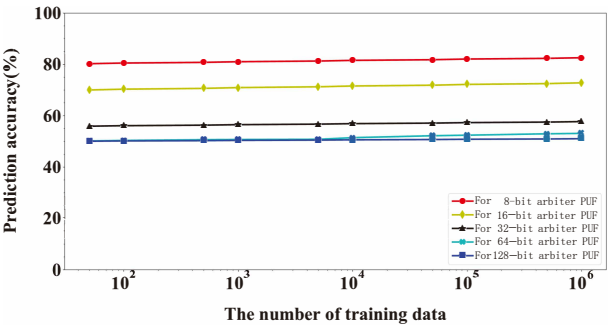


Fig. 17. LR attack results on TCSoS Arbiter PUF with different number of bit_{num} (bit_{num} is the stage size of Arbiter PUF).

TABLE IV
MODELING ACCURACIES ON TCSOS ARBITER PUF WITH DIFFERENT NUMBER OF T_{num} AND bit_{num} USING 10^5 CRPS.

T_{num}	MLs	Bit_{num}			
		16	32	64	128
0	LR	99.99%	99.99%	99.96%	99.95%
	SVM	99.99%	99.97%	99.83%	99.89%
	ANN	99.99%	99.99%	99.96%	99.98%
	CMA-ES	99.99%	99.99%	99.99%	99.99%
1	LR	75.06%	73.23%	79.61%	74.77%
	SVM	71.87%	74.32%	72.39%	75.25%
	ANN	78.83%	72.06%	71.67%	80.54%
	RNN	76.96%	77.23%	76.79%	73.36%
	CMA-ES	79.36%	76.07%	78.26%	73.60%
2	LR	59.96%	61.23%	63.39%	65.03%
	SVM	56.51%	66.32%	58.11%	64.05%
	ANN	61.33%	58.74%	57.70%	60.22%
	RNN	58.73%	64.54%	61.96%	67.14%
	CMA-ES	58.28%	56.23%	63.31%	62.26%
4	LR	52.23%	52.33%	52.09%	53.02%
	SVM	52.05%	51.81%	52.93%	52.23%
	ANN	53.07%	53.16%	52.68%	51.99%
	RNN	52.18%	51.87%	52.59%	52.83%
	CMA-ES	53.06%	52.30%	51.96%	52.30%
8	LR	51.73%	52.49%	52.03%	52.38%
	SVM	51.57%	52.81%	51.77%	52.49%
	ANN	52.13%	52.16%	51.69%	51.37%
	RNN	52.13%	51.80%	51.97%	52.43%
	CMA-ES	52.09%	51.27%	52.36%	51.97%
16	LR	51.99%	52.37%	52.51%	51.69%
	SVM	52.26%	51.59%	51.72%	52.50%
	ANN	51.87%	52.32%	51.92%	52.42%
	RNN	52.32%	52.29%	51.87%	52.69%
	CMA-ES	52.28%	51.98%	52.31%	51.72%

The ICS is composed of latches and multiplexers, and the key generator can be Weak PUF or TRNG. Since the CSoS cannot be decomposed into basic logical gates, the logical approximation cannot work efficiently. In addition, we have obfuscated the different timing inputs based on the value of k_2, the obfuscation scheme is difficult to be approximated by a function, so the global approximation is no longer effective. In [23], the authors recommend to deploy an appropriate defense mechanism on the PUF, such as CRP obfuscation. Our proposed CSoS is such an obfuscation mechanism and hence approximation attacks are difficult to break it.

TABLE V
OVERHEAD OF AUTHENTICATION CIRCUITS IN THE DESIGN (64 BIT)

Authentication circuits	No. of LUTs	No. of Flip-Flops	Power(W)
Arbiter PUF	128	1	0.102
3-XOR PUF	385	3	0.387
TCSoS Arbiter PUF	257	130	0.431

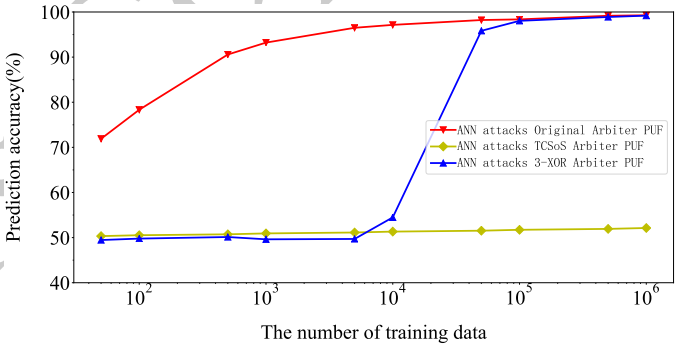


Fig. 18. LR attack results on TCSoS Arbiter PUF with different number of bit_{num} (bit_{num} is the stage size of Arbiter PUF).

D. Hardware Overhead

We implement 64-bit Arbiter PUF [13], 64-bit 3-XOR PUF [45] and the TCSoS for 64-bit Arbiter PUF on Zynq-7000 xc7z010clg400-1 FPGAs to show the hardware efficiency.

As shown in Table V, a 64-bit Arbiter PUF composed of multiplexer and D flip-flop consumes 128 LUTs and 1 Flip-Flop. A 64-bit 3-XOR PUF with three Arbiter PUFs composed of multiplexer, D flip-flop and XOR gate consumes 385 LUTs and 3 Flip-Flops. A 64-bit TCSoS Arbiter PUF composed of multiplexer, D flip-flop, XOR gate and TRNG consumes 257 LUTs and 130 Flip-Flops. It can be observed that the hardware overhead of our proposed 64-bit TCSoS Arbiter PUF on LUTs is less than 64-bit 3-XOR PUF, the hardware overhead on Flip-Flops is greater than 64-bit 3-XOR PUF. Therefore, their hardware overhead is comparable. At the same time, we also evaluated the power consumption. It can be seen that the power consumption of the proposed structure is a little larger than 3-XOR PUF. However, we used ANN to model the three PUFs, when 10^6 CRPs are collected, ANN can achieve 99.27%, 99.19% and 52.13% modeling accuracy, respectively. The experimental results are shown in Fig. 18. Obviously, our

proposed TCSoS Arbiter PUF uses less hardware resources to enhance anti-modeling effects. In addition, XOR PUF will become increasingly unreliable as the number of Arbiter PUFs increases. TCSoS uses the previous challenges combined with random numbers to obfuscate the current challenge without changing the structure of the authentication circuit. Therefore, it will not affect the uniqueness and reliability.

The experimental results in this paper show that the proposed CSoS incurs low hardware and power overhead. Therefore, it is a lightweight solution to meet the authentication requirements of IoT.

VI. Conclusion

In this paper, we carefully analyzed the working principle of VOS and illustrated how to use the errors generated by the computing unit as the hardware fingerprints. Then, we analyzed the relationship between the inputs and outputs in VOLtA's work process. Moreover, we reevaluated the security of VOLtA by modeling logic gates 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, this paper proposes a novel challenge self-obfuscation structure, named CSoS, which includes WCSoS and TCSoS. CSoS-based ML attacks resistant authentication protocol can lower the prediction accuracy of ML on the VOLtA to 51.2%. In addition, we also designed the input cache structure (ICS) to complete the obfuscation of CSoS. Furthermore, we collect 10^6 CRPs of an Arbiter 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%. Our proposed CSoS exhibits high obfuscation ability for both VOLtA and strong PUFs with lower hardware overhead and power consumption.

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.ihs.com/587648, 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] S. Garg, K. Kaur, G. Kaddoum, and K. R. Choo, "Toward Secure and Provable Authentication for Internet of Things: Realizing Industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4598-4606, 2020.

[5] C. Miranda, G. Kaddoum, E. Bou-Harb, S. Garg, and K. Kaur, "A Collaborative Security Framework for Software-Defined Wireless Sensor Networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2602- 2615, 2020.

[6] J. Zhang, G. Qu, Y. Q. Lv, and Q. Zhou, "A survey on silicon PUFs and recent advances in ring oscillator PUFs," *Journal of Computer Science and Technology*, 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," in *Asia and South Pacific Design Automation Conference*, pp. 336-341, 2017.

[8] J. Zhang, and G. Qu, "Physical Unclonable Function-based Key-Sharing via Machine Learning for IoT Security," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 7025-7033, 2020.

[9] R. Venkatesan, 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*, pp. 667-673, 2011.

[10] J. N. Chen, and J. H. Hu, "Energy-Efficient Digital Signal Processing via Voltage-Overscaling-Based Residue Number System," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 7, pp. 1322-1332, 2013.

[11] H. Su, and J. Zhang, "Machine Learning Attacks on Voltage Over-scaling-based Lightweight Authentication," in *Asian Hardware Oriented Security and Trust Symposium*, pp. 50-55, 2018.

[12] A. Vijayakumar and S. Kundu, "A Novel Modeling Attack Resistant PUF Design based on nonlinear Voltage Transfer Characteristics," in *Design, Automation And Test in Europe*, 2015, pp. 653-658.

[13] Daihyun Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[14] G. S. Lee, G. Kim, K. Kwak, D. S. Jeong, and H. Ju , "Enhanced Reconfigurable Physical Unclonable Function Based on Stochastic Nature of Multilevel Cell RRAM," *IEEE Transactions on Electron Devices*, vol. 66, no. 4, pp. 1717-1721, 2019.

[15] D. Li, and K. Yang, "A 562F2 Physically Unclonable Function with a Zero-Overhead Stabilization Scheme," in *IEEE International Solid-State Circuits Conference*, pp. 400-402, 2019.

[16] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proceedings of the Conference on RFID Security*, pp. 1-12, 2007.

[17] G. E. Suh, and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *ACM/IEEE Design Automation Conference*, pp.9-14, 2007.

[18] M. Sauer, P. Raiola, L. Feiten, B. Becker, U. Rührmair, and I. Polian, "Sensitized path PUF: A lightweight embedded physical unclonable function," in *Design, Automation And Test in Europe*, pp. 680-685, 2017.

[19] N. Wisiol, G. T. Becker, M. Margraf, T. A. A. Soroceanu, J. Tobisch, and B. Zengin, "Breaking the lightweight secure PUF: Understanding the relation of input transformations and machine learning resistance," *Smart Card Research and Advanced Applications*, Springer International Publishing, pp. 40-54, 2020.

[20] G. T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs," *Cryptographic Hardware and Embedded Systems*, Springer Berlin Heidelberg, pp. 535–555, 2015.

[21] F. Ganji, "On the Learnability of Physically Unclonable Functions," Springer International Publishing, 2018.

[22] J. Delvaux, "Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2043-2058, 2019.

[23] J. Shi, Y. Lu, and J. Zhang, "Approximation Attacks on Strong PUFs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2138-2151, 2020.

[24] J. Zhang, C. Shen, "Set-based Obfuscation for Strong PUFs against Machine Learning Attacks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1-13, 2020.

[25] X. Xu, and J. Zhang, "Rethinking FPGA Security in the New Era of Artificial Intelligence," in *International Symposium on Quality Electronic Design*, pp. 46-51, 2020.

[26] R. Kumar, and W. Burleson, "On design of a highly secure PUF based on nonlinear current mirrors," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 38-43, 2014.

[27] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Annual Computer Security Applications Conference*, pp. 149-160, 2002.

[28] G. T. Becker, "On the Pitfalls of Using Arbiter-PUFs as Building Blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1295-1307, 2015.

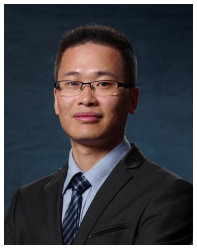
[29] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching," in *IEEE Symposium on Security and Privacy Workshops*, pp. 33-44, 2012.

[30] M. Yu, D. M'Raïhi, I. Verbauwhede, and S. Devadas, "A noise bifurcation architecture for linear additive physical functions," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 124-129, 2014.

[31] F. Z. Rokhani, and G. E. Sobelman, "Low-power bus transform coding for multilevel signals," in *IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 1272-1275, 2006.

[32] V. Gutnik, and A. P. Chandrakasan, "Embedded power supply for low-power DSP," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 425-435, 1997.

[33] N. Chabini and W. Wolf, "Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 6, pp. 573-589, 2004.
[34] R. Liu, and K. K. Parhi, "Power reduction in frequency-selective FIR filters under voltage overscaling," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 343-356, 2011.
[35] H. Li, J. Hu, and J. Chen, "A novel low-power filter design via reduced-precision redundancy for voltage overscaling applications," in *IEEE Global Communications Conference*, pp. 3282-3287, 2013.
[36] U. Rührmair, F. Sehnke, J. Selter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *ACM Conference on Computer and Communications Security*, pp. 237-249, 2010.
[37] C. M. Bishop, "Pattern Recognition and Machine Learning (Information Science and Statistics)," *Springer-Verlag*, 2006.
[38] F. Rosenblatt, "The Perceptron - A Perceiving and Recognizing Automaton," *Math. Stat*, 1957.
[39] N. Hansen, "The CMA Evolution Strategy: A Tutorial," arXiv:1604.00772v1, 2016.
[40] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *Annual International Cryptology Conference on Advances in Cryptology*, Springer-Verlag, pp. 104-113, 1996.
[41] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-Channel Analysis of PUFs and Fuzzy Extractors," in *International Conference on Trust and Trustworthy Computing*, pp. 33-47, 2011.
[42] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl, "Localized electromagnetic analysis of RO PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust* , pp. 19-24, 2013.
[43] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An Open-Source Variation-Aware Design Kit," in *IEEE International Conference on Microelectronic Systems Education*, pp. 173-174, 2007.
[44] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "PUF Modeling Attacks on Simulated and Silicon Data," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876-1891, 2013.
[45] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and Reliable XOR Arbiter PUF Design: An Experimental Study based on 1 Trillion Challenge Response Pair Measurements," *Design Automation Conference*, pp. 1-7, 2017.



Jiliang Zhang received the Ph.D. degree in Computer Science and Technology from Hunan University, Changsha, China in 2015. From 2013 to 2014, he worked as a Research Scholar at the Maryland Embedded Systems and Hardware Security Lab, University of Maryland, College Park. From 2015 to 2017, he was an Associate Professor with Northeastern University, China. He is currently a Professor at Hunan University, China. His current research interests include hardware/hardware-assisted security, artificial intelligence security and privacy protection. He has authored more than 50 technical papers in leading journals and conferences such as IEEE TIFS, TCAD, TVLSI, ACM TODAES, ACM TRETS, TNNLS, TCASII, TMSCS, DAC and FCCM.

Prof. Zhang is a recipient of the best paper nominations in several conferences. He has been serving on the technical program committees of many international conferences such as ASP-DAC, FPT, GLSVLSI, ISQED and AsianHOST. He is serving as a steering member for Hardware Security Forum of China and Editorial Board member for International Journal of Cognitive Computing in Engineering. He is a senior member of IEEE and a Guest Editor of the Journal of Information Security and Applications and Journal of Low Power Electronics and Applications. He severed as a reviewer for dozens of IEEE transactions and international conferences.

Chaoqun Shen is currently pursuing the Ph.D. degree with Hunan University, China. Her current research interests include microarchitecture and hardware security.

Haihan Su received the master degree from College of Information Science and Engineering, Hunan University, in 2020. His research interests include hardware security and AI security.

Md Tanvir Arafin received his Ph.D. from Department of Electrical and Computer Engineering at the University of Maryland, College Park, in 2018. He is currently an Assistant Professor in the Electrical and Computer Engineering Department, Morgan State University. His research interests include semiconductor physics, integrated circuits, and embedded security of microelectronic devices.

Gang Qu (Fellow, IEEE) received the B.S. and M.S. degrees in mathematics from the University of Science and Technology of China, in 1992 and 1994, respectively, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2000. Upon graduation, he joined the University of Maryland at College Park, where he is currently a professor in the Department of Electrical and Computer Engineering and Institute for Systems Research. He is also a member of the Maryland Cybersecurity Center and the Maryland Energy Research Center. Dr. Qu is the director of Maryland Embedded Systems and Hardware Security Lab and the Wireless Sensors Laboratory.

His primary research interests are in the area of embedded systems and VLSI (Very Large Scale Integration) CAD (Computer Aided Design) with focus on low power system design and hardware related security and trust. He studies optimization and combinatorial problems and applies his theoretical discovery to applications in VLSI CAD, wireless sensor network, bioinformatics, and cybersecurity. Dr. Qu has received many awards for his academic achievements, teaching, and service to the research community. He is serving as associate editor for the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, ACM Transactions on Design Automation of Electronic Systems, IEEE Embedded Systems Letters and Integration, the VLSI Journal.