Approximation Attacks on Strong PUFs

Junye Shi *Student Member, IEEE*, Yang Lu *Student Member, IEEE*, Jiliang Zhang, *Senior Member, IEEE*

***Abstract*—Physical unclonable function (PUF) is a promising lightweight hardware security primitive for resource-constrained systems. It can generate a large number of challenge-response pairs (CRPs) for device authentication based on process variations. However, attackers can collect the CRPs to build a machine learning (ML) model with high prediction accuracy for the PUF. Recently, a lot of ML-resistant PUF structures have been proposed, e.g., a multiplexer-based PUF (MPUF) was introduced to resist ML attacks and its two variants (rMPUF and cMPUF) were further proposed to resist reliability-based and cryptanalysis modeling attacks, respectively. In this paper, we propose a general framework for ML attacks on strong PUFs, then based on the framework, we present two novel modeling attacks, named *logical approximation* and *global approximation*, that use artificial neural network (ANN) to characterize the nonlinear structure of MPUF, rMPUF, cMPUF and XOR Arbiter PUF. Logical approximation method uses linear functions to approximate logical operations and builds a precise soft model based on the combination of logical gates in the PUF. Global approximation method uses the function *sinc* with filtering characteristics to fit the mapping relationship between the challenge and response. Experimental results show that the proposed two approximation attacks can successfully model the (n, k)-MPUF (k = 3, 4), (n, k)-rMPUF (k = 2, 3), cMPUF (k = 4, 5) and l-XOR Arbiter PUF (l = 3, 4, 5) (n = 32, 64) with the average accuracies of 96.85%, 95.33%, 94.52% and 96.26%, respectively.**

***Index Terms*—Physical Unclonable Function, Machine Learning, Artificial Neural Network, Modeling Attacks, Cryptanalysis.**

I. Introduction

THE Internet of things (IoT) is a network that connects ubiquitous devices and terminals including various sensors, smart devices, mechanical and electronic components through wired or wireless networks [1]. According to *IoT Analytics* estimates [2], there are about 7 billion connected IoT devices at the end of 2018. Most of those devices are connected via short range protocols (WPAN or WLAN). The number of total connected IoT devices is expected to reach 22 billion by 2025. *IoT Security Market Report 2017-2022* [3] pointed out that the total spending of IoT security was 703 million dollars in 2017, reached 1,001 million dollars in 2018 and was estimated at about 1,439 million dollars in 2019. Security issues have governed the sustainable development of the IoT. Device authentication is a key technology to address IoT security issues. Traditional device authentications are based on high-complexity encryption/decryption algorithms and high-cost key storage technologies. However, in many IoT applications, resources like CPU, memory, and battery power are limited. Therefore, lightweight solutions are urgent.

Physical unclonable function (PUF) [4] is an alternative lightweight solution for various applications such as IP protection [5], software security [6], anti-overbuilding [7] and key-sharing [8]. PUFs can extract unique secret keys from the physical characteristics of the chip, or generate a large number of CRPs for device authentication based on physical interaction which is extremely hard to reproduce using a challenge and response procedure. Entropy of the physical interaction is derived from physical random variations caused by manufacturing process, which improves the ability to resist physical attacks and has the potential to be more efficient in IoT since costly physical protection measures can be avoided [9]. Since the PUF was first introduced in 2002 [10], a lot of PUF structures have been proposed. Current PUFs can be classed into the weak PUFs [11], [12], [13], [14] and strong PUFs [15], [16], [17], [18], [19] according to the number of CRPs. Weak PUF has a very limited CRPs space, which can be used as unique keys for encryption systems. Strong PUF has an exponential number of unique CRPs and a complex input-output relationship, which is suitable for lightweight device authentication. However, strong PUFs are vulnerable to machine learning (ML) attacks [20], [21], [22]. Attackers can collect a subset of CRPs from a strong PUF instance and use ML algorithms to build a mathematical model to predict response for arbitrary challenge with a high accuracy. Once the mapping relationship between the challenge and response is modeled, PUF is no longer unclonable.

In order to resist ML attacks, a large number of defenses have been proposed [19], [23], [24], [12]. Recently, multiplexer-based PUF (MPUF) and its two variants rMPUF and cMPUF [17] were proposed to prevent advanced ML attacks. However, in this paper, we analyze the structural characteristics of MPUFs in detail and prove that they can still be broken. Additionally, we propose a general framework for ML attacks on strong PUFs and present two advanced approximation attacks based on the framework, named logical approximation and global approximation. The logical approximation uses linear functions to approximate basic logical operations including AND, OR, and NOT, then build an artificial neural network (ANN) model for the logical structure of MPUF. The global approximation uses the function sinc(x) with filtering characteristics to model the mapping relationship between the input and output of MUX. The two approximation attacks model the MPUF and its variants from the local and global aspects, respectively, which provides a general framework to

This work was supported by the National Natural Science Foundation of China under Grant No. 61874042 and 61602107, 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. Shi, Y. Lu and J. Zhang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

conduct modeling attacks on strong PUFs: for a PUF with a simple logical gate combination in the structure, the logical operation can be approximated to a simpler linear equation; for a PUF with highly complex internal logical relationship, a mathematical function can be used to describe the mapping relationship between the challenge and response. The main contributions are as follows.

1) We analyze the working principle and structural characteristics of MPUF family, and reevaluate their security. In addition, we present accurate mathematical models for them.
2) We propose a general framework for ML attacks on strong PUFs. Based on the framework, we present two new ANN-based ML attacks, logical approximation and global approximation, to model the MPUF family and XOR Arbiter PUF.
3) Experimental results show that MPUF, rMPUF, cMPUF and XOR Arbiter PUF can be modeled by the proposed new ML attacks, and the average modeling accuracies are 96.85%, 95.33%, 94.52% and 96.26%, respectively.

The source code to reproduce the experiment is available online at http://hardwaresecurity.cn/AttackPUFcode.zip

The rest of this paper is organized as follows. Section II introduces ANN and modeling attacks on the Arbiter PUF and XOR Arbiter PUF. Section III describes the security and mathematical analysis for MPUF and its variants. Section IV demonstrates the ML attacks on MPUF, rMPUF, cMPUF and XOR Arbiter PUF. The experimental results and analysis are given in Section V. Related work is elaborated in Section VI. Finally, a conclusion is made in Section VII.

II. Preliminaries

In this section, we will describe the ANN which is used to model MPUF family and introduce the modeling attack on Arbiter PUF and XOR Arbiter PUF.

A. ANN

ANN is a computing system vaguely inspired by the biological neural networks that constitute animal brains [25]. It uses the knowledge of network topology as the theoretical basis to simulate the processing of complex information in a human brain. ANN consists of a large number of interconnected neurons. Each neuron represents a specific output function called activation function. The connection between two neurons is called weight that is equivalent to the memory of the neural network. For each neuron, input vectors are weighted, added, biased, and input to an activation function to generate an output. During the training of ANN, neurons update weights and biases based on the feedback function of the prediction error derived from the training set, and finally obtain a global optimal solution. ANN has high fault tolerance, self-adaptation, and parallel processing capabilities, hence it is suitable for modeling complex nonlinear PUF structures. A strong motivation to use the ANN-based modeling attack is that a feed-forward network with at least a single hidden layer is able to approximate any function with an activation function when the number of neurons is sufficient [26]. Based on this universal approximation theorem, we propose two modeling attacks, named logical approximation attack and global approximation attack. Logical approximation attack is to approximate a logical operation in the circuit with a linear function, and then use a two-layer neural network based on the linear function to replace the corresponding logical gate. Each individual subnetwork is connected based on the combination of logical gates, and a large neural network is constructed to characterize the entire PUF logical structure. Global approximation attack is to determine a continuous function that can satisfy the mapping relationship between the challenge and response, and organize the neural network structure to approximate this function. More details will be introduced in section IV.

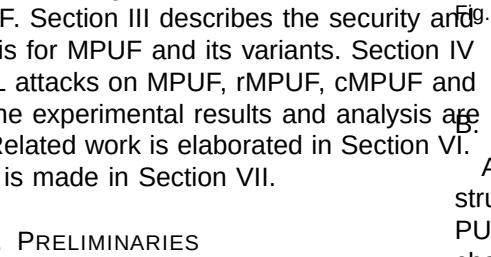

Fig. 1. The structure of Arbiter PUF.

B. Arbiter PUF

Arbiter PUF is one of classic strong PUFs [15]. The structure of Arbiter PUF is shown in Fig. 1. An Arbiter PUF consists of two parallel n-stage multiplexer (MUX) chains. The challenges $C (= c_1; c_2; c_3 ::: c_n)$ are the selection inputs of MUX chains. Two MUXes of the same stage share one selection input which determines whether the signal is transmitted in parallel or across. Therefore, the challenges divide the circuit into two independent propagation paths. A step signal T is given at the input side, after the propagation on two paths, an arbiter (D flip-flop) will receive two signals. 1-bit response $r \in \{0; 1\}$ is generated according to the order of the two signals arriving at the arbiter. If the upper signal arrives at D side first, the response is 1, otherwise it is 0. In general, Arbiter PUF exploits the delay difference between two paths to generate a response.

The functionality of Arbiter PUF can be represented by an additive linear delay model [20]. The total delay can be obtained by summating the propagation delay in each stage, which can be expressed as

$$\Delta = \vec{w}^{T} \cdot \vec{\Phi}; \quad (1)$$

where $\vec{\Phi}$ is a feature vector that represents the propagation delay of each MUX in the Arbiter PUF, and $\vec{w}$ is a function of n-bit challenge C:

$$\vec{\Phi}(C) = (\Phi_1(C); \Phi_2(C); \Phi_3(C); ::: ; \Phi_n(C); 1)^T; \quad (2)$$

where $\Phi_j(C) = \prod_{i=j}^{n} (1 - C_i); j = 1; 2; 3; ::: ; k$. If $\Delta > 0$, the output of Arbiter PUF r will be 1, otherwise, r will be

TABLE I
LIST OF PARAMETERS

Symbol	Description
n	Number of challenges
k	Number of MUX stages
(n,k)-MPUF	An MPUF with n-bit challenge and k selection inputs
2^k 1 MUX	A MUX with k selection inputs, 2^k data inputs and 1-bit output
c	Challenges of Arbiter PUF as well as MPUF
o	1-bit output of 2 1 MUX
O	1-bit output of MPUF
A_i^d	The i-th Arbiter PUF connected to the MUX data input
A_i^s	The i-th Arbiter PUF connected to the MUX selection input
R_i^d	Response of the i-th Arbiter PUF connected to the MUX data input
R_i^s	Response of the i-th Arbiter PUF connected to the MUX selection input
R^d	The set of all data inputs
R^s	The set of all selection inputs

0. For convenience, we use $t \in \{-1, 1\}$ to represent the output:

$$t = \mathrm{sgn}(\Delta) = \mathrm{sgn}(\vec{\omega}^{T} \cdot \vec{\Phi}) \quad (3)$$

An ML algorithm is used to learn $\vec{\omega}$ that shows the practical delay vector of physical PUF architecture. It is verified that the delay vector $\vec{\omega}$ of each stage in the Arbiter PUF can be learned within limited CRPs. The general assumption is that such ML attack requires less than 10% of the total CRPs [27].

C. XOR Arbiter PUF

In order to resist ML attacks, non-linear components such as XOR gates are introduced into the PUF structure. For example, in the XOR Arbiter PUF (XOR PUF for short) [28], the responses of multiple Arbiter PUFs are XORed as the final response. XOR gates can increase the complexity of PUF structure significantly, but it can still be modeled with a high accuracy by advanced ML attacks.

An XOR PUF with l individual Arbiter PUFs is denoted by l-XOR PUF. The individual outputs of Arbiter PUF is denoted by $t_i \in \{-1, 1\}, i = 1, 2, \ldots, l$. The l-stage XOR gate is used to XOR all t_i to get the final response $t_{XOR} = \prod_{i=1}^{l} (t_i)$. Assuming that all Arbiter PUFs share the same challenges and each Arbiter PUF has a unique delay feature vector $\vec{\omega}$, t_{XOR} can be expressed as

$$t_{XOR} = \mathrm{sgn}\left(\prod_{i=1}^{l} \vec{\omega}^{T} \cdot \vec{\Phi}\right) \quad (4)$$

It is well-known that environmental factors such as supply voltage and temperature have a negative influence on the reliability of Arbiter PUF [29]. An l-XOR PUF aggregates the unreliability of l Arbiter PUFs so that the reliability decreases exponentially with the number of XORs [30]. In [31], Becker proposed a reliability-based ML attack on XOR PUF, which will be discussed in detail in Section VI.

Fig. 2. The structure of n-$(, k)$-MPUF [17].

III. SECURITY AND MATHEMATICAL ANALYSIS FOR MPUF

In this section, we will introduce the MPUF and its two variants, rMPUF and cMPUF [17], then reevaluate their security. Finally, the detailed mathematical analysis for MPUFs will be given.

A. Security Analysis for MPUFs

1) Basic MPUF

As shown in Fig. 2, an n-$(, k)$-MPUF consists of a 2^k 1 MUX and $2^k + k$ Arbiter PUFs. The data inputs and selection inputs of the 2^k 1 MUX are the responses generated by different Arbiter PUF instances with the same challenge. With the selection inputs, the 2^k 1 MUX selects one of the data inputs as the final output. The parameters of MPUF are shown in Table I. According to R^s, the 2^k 1 MUX selects 1-bit response from R^d as the output. In this case, R^s and R^d are generated from independent Arbiter PUF instances. In the obfuscated MPUF circuit, attackers can only collect the challenges but cannot get the responses from the underlying Arbiter PUF directly, i.e., attackers cannot access the inputs of the 2^k 1 MUX directly to distinguish internal selection inputs and data inputs. Hence, it is difficult for attackers to model MPUF.

In [31], Becker proposed a novel reliability-based ML attack which exploits the reliability information of CRPs to model the XOR PUF. In an (n, k)-MPUF, 2^k 1 2 1 MUXes can be used to replace the 2^k 1 MUX, and the MPUF structure is divided into k stages. For example, when $k = 3$, the structure of n-$(, 3)$-MPUF is shown in Fig. 3(a). In the i th $(i = 1, 2, 3, \ldots, k)$ stage, 2^{i-1} MUXes will select half of the data inputs according to $R_i^{s\ 1}$ so that all selection inputs are inseparably linked to the response reliability of MPUF. Attacks can be conducted with the reliability information to build a highly accurate model for MPUF [17]. Therefore, MPUF is vulnerable to the reliability-based ML attack.

2) rMPUF

In [17], a robust variant rMPUF is proposed to resist the reliability-based ML attack, and the structure of an n-$(, 3)$-rMPUF is shown in Fig. 3(b). Compared with MPUF, rMPUF

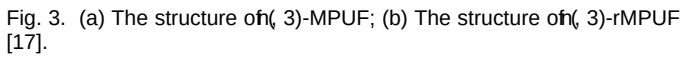

Fig. 3. (a) The structure of $(n, 3)$-MPUF; (b) The structure of $(n, 3)$-rMPUF [17].

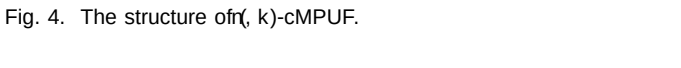

Fig. 4. The structure of (n, k)-cMPUF.

increases the number of selection inputs from 0 to $2^k - 1$, and all the selection inputs for each $2^k - 1$ MUX are generated by different Arbiter PUFs. For an (n, k)-rMPUF, the data inputs are R_i^d $(i = 0; 1; 2; :::; 2^k - 1)$ and selection inputs are R_i^s $(i = 0; 1; 2; :::; 2^k - 2)$. Hence, the number of Arbiter PUF in a (n, k)-rMPUF is $2^{k+1} - 1$, and the attack complexity increases exponentially with k. The overhead of rMPUF should not be too high. Considering the hardware overhead, $k = 3$ is taken as the preferred value in practice [17]. It is claimed that $(n, 3)$-rMPUF has better reliability and stronger resistance against modeling attack than 10-XOR PUF [17]. However, the overhead of $(n, 3)$-rMPUF is 50% more than 10-XOR PUF. In addition, this paper proves that $(n, 3)$-rMPUF can still be modeled with a high accuracy. Therefore, $(n, 3)$-rMPUF not only incurs high overhead but also shows low security.

3) cMPUF

As shown in Fig. 4, cMPUF is another MPUF variant proposed to resist linear cryptanalysis (LC) [17]. Compared with MPUF, cMPUF changes the data inputs and introduces inverters to make half of the data inputs complement of the others.

LC is a mathematical method used to find a linear approximation on the PUF for predicting the response for an arbitrary challenge. Modeling attacks based on the linear approximation is called linear approximation attack. Adding nonlinear components such as XOR gates into the PUF structure can resist the linear approximation attack [32], [28]. In cMPUF, inverters are introduced to the structure so that the data inputs of each $2^k - 1$ MUX in Stage-1 are the response of Arbiter PUF and its inversion. Therefore, the function of each $2^k - 1$ MUX in the first stage is an XOR gate, and the output of $2^k - 1$ MUX can be expressed as

$$o = ((!\, s)\&\, d)\,|\,(s\&(!\, d)) = s \text{ XOR } d; \tag{5}$$

where d is the 1-bit response of Arbiter PUF, s is the selection input, '!' denotes NOT, '&' denotes AND, and '|' denotes OR. Adding an inverter is equivalent to adding an XOR gate into the $2^k - 1$ MUX, which increases the nonlinearity of PUF structure greatly and makes cMPUF resistant to linear approximation attacks.

Fig. 5. Function $\sin c()$ and (m).

B. Mathematical Analysis for MPUF

In an MPUF, $2^k - 1$ MUX can be decomposed into several $2^k - 1$ MUXes. The decomposed structure of (3)-MPUF is shown in Fig. 3(a). Each $2^k - 1$ MUX is composed of NOT, AND and OR gates and can be implemented with the basic logical operations

$$o = (d_0\&(!\, s))\,|\,(d_1\&s); \tag{6}$$

where d_0 and d_1 are the upper and lower data inputs of $2^k - 1$ MUX, respectively; s is the selection input. The basic logical gates can be approximated by the following functions.

$$!a = 1 \quad a; \tag{7}$$

$$a \text{ AND } b \quad f_{AND} = (20 \quad a + 20 \quad b \quad 30); \tag{8}$$

Fig. 6. The framework of approximation attacks on PUFs.

$$a \text{ OR } b \quad f_{OR} = \quad (20 \quad a + 20 \quad b \quad 10); \tag{9}$$

$$\begin{aligned} a \quad b &= (a\&(!b))\,|\,((!a)\&b) \\ f_{XOR} &= f_{OR}(f_{AND}(a; 1 \quad b); f_{AND}(1 \quad a; b)); \end{aligned} \tag{10}$$

where $(x) = \text{sigmoid}(x) = 1=(1+e^{\ x})$, which is a common activation function used in ML algorithms. The $2^{\ } \quad 1$ MUX can be approximated with the above functions by substituting Eqn. (7)-(9) into Eqn. (6). The output of $2^{\ } \quad 1$ MUXes in each stage (except for the last stage) is used as the input of another $2^{\ } \quad 1$ MUX in the next stage. By this way, a complete neural network can be constructed.

As analyzed above, the MUX can be decomposed into multiple AND, OR and NOT logical gates and each gate can be approximated by a mathematical equation, which is the approximation on MUX from the local perspective. In addition, the mapping relationship between the input and output of MUX can be approximated from the global perspective and learned by the back propagation-based (BP) algorithm.

Firstly, the binary selection input R^s is converted into a decimal number S:

$$S = \sum_{i=0}^{k \quad 1} 2^{i} \quad R_i^s: \tag{11}$$

And the selected data input becomes R_S^d.

Secondly, we use a discrete function $g(m)$ to describe the behavior of MUX. Here, m is the serial number of data inputs R_m^d in the MUX ($m = 0; 1; 2; :::; k \quad 1$). We can define that at the selected point (denoted as the impulse point), $g(S) = 1$; at other points, $g(!S) = 0$:

$$g(m) = \begin{matrix} 1 \ (m = S) \\ 0 \ (m! = S) \end{matrix}: \tag{12}$$

Then a normalized discrete delta function (m) is used to replace $g(m)$, $m \ 2 \ (\ 1 \ ; +1 \ )$:

$$(m) = \begin{matrix} 1 \ (m = 0) \\ 0 \ (m! = 0) \end{matrix}: \tag{13}$$

The schematic representation of (m) is shown in Fig 5. The final output of MUX O can be expressed as the accumulated value of all $2^k \quad 1$ points:

$$O = \sum_{i=0}^{2^k \quad 1} R_i^d \quad (i \quad S): \tag{14}$$

Eqn. (14) describes the mapping relationship between inputs and outputs of MPUF, but it cannot be directly applied to the BP algorithms because (m) is a discrete function. Therefore, we need to find a continuous approximation function to replace (m), which should satisfy two characteristics of (m): 1) the value of the function is 1 at the impulse point and 0 at the rest points; 2) the function is symmetrical with respect to the impulse point. The normalized sampling function $\text{sinc}(x)$ exactly satisfies these characteristics. As shown in Fig. 5, (x) can be regarded as $\text{sinc}(x)$ sampled at integral points, and $\text{sinc}(x)$ can satisfy the characteristics of (m). $\text{sinc}(x)$ can be expressed as

$$\text{sinc}(x) = \sin(\quad x)=(\quad x): \tag{15}$$

Then replace (m) with $\text{sinc}(x)$ in Eqn. (14), the output O of MPUF with k selection inputs can be approximated as

$$O \quad \sum_{i=0}^{2^k \quad 1} R_i^d \quad \text{sinc}(i \quad S): \tag{16}$$

As shown in Fig. 5, the main lobe of $\text{sinc}(x)$ decays rapidly, and the value of the side lobes is small. Hence, the values of the unselected points (points except for the impulse point) are close to zero. Function $\text{sinc}(x)$ is equivalent to a filter that filters out all the unselected points. By substituting all points into Eqn. (16), the accumulated value will be very close to the value of the impulse point because the values of unselected points are all close to 0.

IV. Approximation Attacks

In this section, we will introduce a general modeling framework for ML attacks on strong PUFs and propose two approximation attacks based on this framework in detail.

A. Modeling Framework

An overview of the proposed approximation attacks is shown in Fig. 6, which provides a general framework for implementing approximation attacks on strong PUFs. The basic idea of approximation attacks is to find an approximation function to fit the mapping relationship between the challenge and response. In the framework, two approximation attacks, logical approximation and global approximation, are proposed to model strong PUFs from the local and global aspects, relatively.

First, we analyze the PUF structure to determine whether it is a composition of logical gates. If yes, the logical approximation is preferred.

Logical approximation:

- Decompose the PUF structure into basic logical gates (AND, NOT and OR).
- Approximate the function of logical gates according to Eqn. (7)-(9).
- Substitute parameters (e.g., the inputs of PUF) into the logical gate approximation functions.
- Compose all approximation functions based on the wiring of each individual logical gate in the circuit and obtain a global approximation function for the PUF.

However, when the logical structure is too complex or cannot be decomposed into basic logical gates, the global approximation can be used as an alternative.

Global approximation:

- Analyze and extract the features from the mapping relationship between the input and output of PUF structure.
- Find a mathematical function with the similar features to approximate the mapping relationship.
- Transform the mathematical function to the global approximation function.

Once the global approximation function for the PUF is obtained, an ML model can be built. Finally, ML algorithms are used to train the model.

Based on the framework, we will conduct two attacks on MPUF family, named *logical approximation* and *filter-based global approximation*. In addition, to demonstrate the universality of the proposed framework, the approximation attack on the XOR PUF is conducted for comparison.

B. Approximation Attacks on Basic MPUF

1) Logical Approximation

Based on the analysis of MUX in Section III, we select the first 2^1 MUX in MPUF to build a logical approximation model. Since $(d_0 \& s)$ and $(d_1 \& (!s))$ in Eqn. (6) are impossible to be true at the same time, we replace the logical operation OR with the add operation. The subsequent experiments verify that using the add operation can simplify the complexity of network and get a better result than OR. Based on Eqn. (7), Eqn. (8) and Eqn. (6), the linear approximation equation of 2^1 MUX can be expressed as

$$o \approx (20d_0 + 20s - 30) + (20d_1 - 20s - 10). \quad (17)$$

As shown in Fig. 7, the neural network model for 2^1 MUX can be designed based on Eqn. (17). The neurons in the input layer represent the bias 1, variables d_0, d_1 and s, respectively. In the hidden layer, the two neurons are activated by two $\sigma(x)$ functions in Eqn. (17), and the output layer has only one neuron producing the output of 2^1 MUX. Then we construct the networks of other 2^1 MUXes with the same way and connect them based on the wiring of logical gates in the circuit to get the entire network of $(2^k; k)$-MPUF.

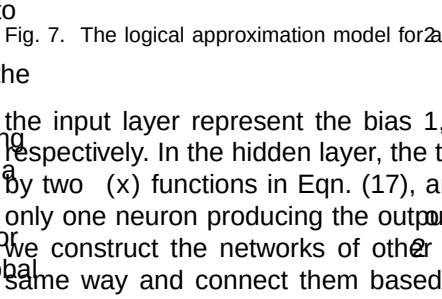

Fig. 7. The logical approximation model for a 2^1 MUX.

2) Filter-based Global Approximation

As discussed in section III, the function $\mathrm{sinc}(x)$ (with low-pass filtering characteristics) can describe the selection behavior of MUX. Therefore, we propose to use $\mathrm{sinc}(x)$ (as activation function to construct the mapping relationship between the input and output of MUX. Since Arbiter PUF is the basic component of MPUF, the additive linear delay model for Arbiter PUF is used to model the MPUF. For convenience, we use $t_{MPUF} \in \{-1, 1\}$ to represent the output of MPUF $O \in \{0, 1\}$. According to Eqn. (3), Eqn. (14) and Eqn. (16), the linear delay model of the MPUF is

$$t_{MPUF} = \mathrm{sgn}\Big[\sum_{i=0}^{2^k-1} (!d_i^{t} - d_i^{t})(i - S)\Big], \quad (18)$$

and the approximation model is

$$t_{MPUF} \approx \mathrm{sgn}\Big[\Big(\sum_{i=0}^{2^k-1} (!d_i^{t} - d_i^{t})\,\mathrm{sinc}(i - S)\Big] = \mathrm{sgn}[f(!; S; k)], \quad (19)$$

where $\mathrm{sgn}(!d_i^{t} - d_i^{t})$ denotes the output of A_i^d. The decimal number S of selection inputs can be expressed as

$$S = \sum_{i=0}^{k-1} 2^i R_i^s \approx \sum_{i=0}^{k-1} 2^i (!s_i^{t} - s_i^{t}). \quad (20)$$

The designed filter-based ANN model based on the approximation functions is shown in Fig. 8. The neurons in the input layer correspond to the selection inputs R^s. In the conversion layer, R^s is converted into the decimal number S. The filter layer contains 2^k neurons with the activation function $\mathrm{sinc}(x)$(

Fig. 8. The filter-based ANN model.

TABLE II
THE TRUTH TABLE FOR SELECTION INPUTS AND FINAL OUTPUT

Selection Inputs R_S							Output
R_6^s	R_5^s	R_4^s	R_3^s	R_2^s	R_1^s	R_0^s	O
1	1	1					R_7^d
1	1	0					R_6^d
1	0		1				R_5^d
1	0		0				R_4^d
0				1	1		R_3^d
0				1	0		R_2^d
0				0		1	R_1^d
0				0		0	R_0^d

The output layer generates the final output. The selection process of the filter-based model is similar to using a filter to remove noisy points.

C. Approximation Attacks on rMPUF

The rMPUF is proved to have enhanced security against the reliability-based attack because more Arbiter PUF instances are added as additional selection inputs. As is shown in Fig. 3, an (n, k)-rMPUF consists of k-stage of 2^1 MUXes. The selection inputs of 2^1 MUXes are generated by independent Arbiter PUFs. Compared with MPUF, the logical structure of rMPUF remains unchanged and hence the logical approximation model for rMPUF is the same as MPUF. We only need to modify the initial value of neurons in the input layer to model the rMPUF.

For the filter-based attack, the ANN model need to be redesigned due to the increase of selection inputs. The linear delay model of rMPUF is similar to MPUF (see Eqn. (18)), the only difference is that S has changed due to the increase of selection inputs in rMPUF. In order to get the expression of S, we summarize the truth table in Table II for the new mapping relationship between the selection inputs R_S and output O in the (n, 3)-rMPUF. In table II, the final output O is only determined by a part of selection inputs. The remaining selection inputs (denoted by X in table II) have no effect on O. For example, when $O = R_0^d$, only R_6^s, R_2^s and R_0^s contribute to the selection of MUX. In this case, the decimal number S_{true} of the real selection inputs can be expressed as

$$S_{true}(0) = 4 \quad R_6^s + 2 \quad R_2^s + R_0^s: \tag{21}$$

For the (n, k)-rMPUF, Eqn. (21) can be generalized to a total of 2^k different expressions $S_{true}(i); i = 0; 1; 2; :::; 2^k \quad 2$ by querying the truth table. The decimal number S of (n, k)-rMPUF can be expressed as

$$\begin{aligned} S = & \sum_{i=0}^{2^k \ 2} i \quad (S_{true}(i) \quad i) \\ & \sum_{i=0}^{2^k \ 2} i \quad sinc(S_{true}(i) \quad i): \end{aligned} \tag{22}$$

Finally, the linear delay model of rMPUF can be obtained by substituting S into Eqn. (18).

D. Approximation Attacks on cMPUF

For the cMPUF, the data inputs of each 2^1 MUXes in stage-1 are the response of Arbiter PUF and its inversion and hence these 2^1 MUXes become XOR gates. The XOR gates increase the nonlinearity of cMPUF greatly so that it is difficult to find a linear approximation for cMPUF. However, the filter-based method can model cMPUF successfully. In the cMPUF, there is a special bit R_0^s: if $R_0^s = 0$, all 2^1 MUXes in stage-1 will select the responses of Arbiter PUFs; if $R_0^s = 1$, all 2^1 MUXes will select the inverters's outputs. Therefore, we can

decompose the (n, k)-cMPUF into an $(n, k-1)$-MPUF and an Arbiter PUF A_0^s. The linear delay model for cMPUF is

$$t_{cMPUF} = sgn[f(!; S; k-1) \cdot !_0^t + s_0^t]; \quad (23)$$

where $sgn(!_0^t \cdot d_0^t + d_0^s)$ denotes the output of A_0^s, and $f(!; S; k-1)$ determines the output of $(n, k-1)$-MPUF.

E. Approximation Attacks on XOR PUF

XOR PUF uses XOR gates to connect individual Arbiter PUFs. Based on the framework in Fig. 6, the XOR PUF can be modeled from global or local perspective. In Section II-C, we have introduced a modeling attack on XOR PUF from the global perspective. From the local perspective, l-XOR PUF contains l XOR operations, and each operation can be approximated by Eqn. (10). Therefore, a logical approximation attack can be conducted on XOR PUF, and the approximation process can be expressed as

$$\begin{aligned} f_{XOR}(2) &= R_1 \cdot R_2; \\ f_{XOR}(3) &= f_{XOR}(2) \cdot R_3; \\ &\vdots \\ f_{XOR}(l) &= f_{XOR}(l-1) \cdot R_l; \end{aligned} \quad (24)$$

where $R_1; R_2; R_3; :::; R_l$ are the responses of Arbiter PUFs.

V. Experimental results and analysis

In this section we will describe the experimental setup, simulation process, and data collection, then present the experimental results and scalability analysis on approximation attacks.

A. Experimental Setup and Data Collection

Similar to existing works about ML modeling attacks on PUFs [17], [31], we reproduce the simulation experiments on MPUF, rMPUF, cMPUF and XOR PUF in Python. The Python 3.6.4 programming language and the TensorFlow 1.11.0 neural network toolkit are used to conduct ML attacks. All experiments are conducted on the Intel i3-8100 CPU@3.6 GHz, GeForce GTX 1060 GPU and 8G RAM.

In the simulation, all delay components in Arbiter PUFs are independent identically distributed and follow the Gaussian distribution $N(10; \sigma_0 = 0{:}05)$. In order to simulate the temperature and supply voltage variations, we add five-level additive noises in the delay components. Assuming that all noises follow $N(0; \sigma_{noise})$ and take the ratio $\sigma_{noise} : \sigma_0 = \gamma$, each delay component follows $N(10; \sigma_0 + \sigma_{noise})$. If $\gamma = 0$, the delay component is noise free and the PUF is 100% reliable. Here, $\gamma = 0.0125$, 0.05, 0.1, 0.15 and 0.2 were used to evaluate the reliability of PUFs and verify the effectiveness of proposed approximation attacks under different noise conditions. $1{:}5 \cdot 10^5$ CRPs are collected with different k and n. (n, k)-MPUF (k = 3, 4), (n, k)-rMPUF (k = 2, 3), (n, k)-cMPUF (k = 4, 5) and l-XOR PUF (l = 3, 4, 5) are simulated with different challenge size n = 32, 64. In the experiment, we conduct approximation attacks under different CRPs and test the average convergence accuracy with 10 times measuring.

TABLE III
Modeling accuracies under noise free conditions

PUF	k(l)	n	CRPs	Approximation Attack	Accuracy(%)
MPUF	3	32	$1{:}5 \cdot 10^4$	Filter-based	97.06
				Logical approximation	97.83
		64	$2 \cdot 10^4$	Filter-based	96.51
				Logical approximation	97.04
	4	32	$5 \cdot 10^4$	Filter-based	96.67
				Logical approximation	96.96
		64	$7 \cdot 10^4$	Filter-based	95.84
				Logical approximation	96.89
rMPUF	2	32	$2 \cdot 10^4$	Filter-based	97.23
				Logical approximation	98.84
		64	$3 \cdot 10^4$	Filter-based	96.61
				Logical approximation	97.79
	3	32	$7 \cdot 10^4$	Filter-based	91.23
				Logical approximation	96.46
		64	$8 \cdot 10^4$	Filter-based	90.43
				Logical approximation	96.04
cMPUF	4	32	$2{:}5 \cdot 10^4$	Filter-based	96.78
		64	$4 \cdot 10^4$		95.31
	5	32	$6 \cdot 10^4$		93.85
		64	$8 \cdot 10^4$		92.16
XOR PUF	3	32	$1{:}5 \cdot 10^4$	Logical approximation	97.98
		64	$2 \cdot 10^4$		97.12
	4	32	$5 \cdot 10^4$		96.82
		64	$6{:}8 \cdot 10^4$		96.02
	5	32	$3{:}5 \cdot 10^5$		95.10
		64	$5 \cdot 10^5$		94.53

B. Approximation Attacks and Results Analysis

This section evaluates the effectiveness of proposed logical approximation attacks and filter-based global approximation attacks. In the experiments, 80% of the CRPs are used as the training set, and the rest 20% are used as the testing set. There are some suggestions in the ML training:

1) The initialization parameters in the model should be biased. We suggest to use a small Gaussian random number as the initial value of parameters, which can prevent training failure due to the symmetry of network.
2) It is recommended to use the RProp [33] and gradient descent algorithm [34] together to train ANN models. In the experiment, if there is only one algorithm used for training, the network does not converge well. Hence, the RProp algorithm is used first, then the gradient descent algorithm instead after the accuracy reaches a threshold. Finally, the loss will steadily drop and the fluctuation can be small enough to be considered convergent.
3) As the training result may fall into local optimum value, it is suggested to restart the learning process to obtain a global optimization result.

The results of approximation attacks under noise free conditions are shown in Table III. For MPUF, both logical approximation and filter-based attacks can model the MPUF structure with high accuracies, and the prediction accuracies are all over 95% regardless of k = 3 or 4. For rMPUF, the accuracies of both models are over 90%. For the filter-based attack, since rMPUF's ANN structure is too complicated, our ML strategy is not effective as expected. Although the prediction accuracy can reach 97.23% at k = 2, it is reduced to 90.43% when k = 3. For the logical approximation attack, the accuracies are higher, and the highest accuracy can even reach 96.46% at k = 3. For cMPUF, the filter-based approximation can model cMPUF successfully, and the prediction accuracy is up to 96.78%. For 3-XOR PUF, 4-XOR PUF and 5-XOR PUF, the proposed

TABLE IV
RELIABILITY OF MPUFS AND XOR PUF UNDER DIFFERENT NOISE LEVELS

PUF	k(l)	n	noise				
			0.0125	0.05	0.1	0.15	0.2
MPUF	3	32	99.06	96.28	92.83	89.88	86.72
		64	98.91	96.08	92.64	89.38	85.80
	4	32	98.93	95.60	91.56	87.12	84.42
		64	98.80	95.54	91.13	87.01	83.92
rMPUF	2	32	99.24	96.88	93.82	90.95	88.84
		64	99.22	96.84	93.57	90.72	88.13
	3	32	99.02	96.23	92.33	89.20	86.42
		64	99.01	96.11	92.25	89.00	86.08
cMPUF	4	32	98.50	94.76	90.36	86.12	81.93
		64	98.48	94.36	90.24	85.97	81.83
	5	32	98.42	94.03	89.13	83.14	80.34
		64	98.27	94.01	89.06	82.86	79.78
XOR PUF	3	32	98.76	95.47	91.73	87.65	83.62
		64	98.65	95.33	90.81	87.33	82.05
	4	32	98.41	94.09	88.50	83.15	79.02
		64	98.34	93.58	87.91	82.53	78.23
	5	32	98.15	92.29	85.94	80.26	75.23
		64	98.06	91.96	85.82	79.84	73.70

TABLE V
MODELING ACCURACIES UNDER DIFFERENT NOISE CONDITIONS

PUF	Approximation Attack	k(l)	n	noise				
				0.0125	0.05	0.1	0.15	0.2
MPUF	Logical approximation	3	32	96.93	95.58	92.08	88.79	86.18
			64	96.35	95.03	91.02	87.58	85.36
		4	32	95.85	94.75	89.03	86.46	80.55
			64	92.35	90.06	88.34	85.13	79.14
rMPUF	Logical approximation	2	32	98.78	92.43	92.13	88.13	87.16
			64	97.41	92.24	91.1	87.84	86.43
		3	32	96.24	92.13	86.73	84.67	80.13
			64	95.72	91.91	80.35	79.46	78.84
cMPUF	Filter-based	4	32	95.78	91.57	88.67	81.46	77.81
			64	93.12	91.14	88.14	81.18	77.68
		5	32	93.43	90.58	86.09	80.35	76.74
			64	91.55	89.73	84.33	80.03	72.45
XOR PUF	Logical approximation	3	32	96.51	93.45	89.88	83.15	80.13
			64	96.01	92.78	88.45	82.75	79.54
		4	32	95.47	91.24	86.51	80.98	76.78
			64	94.72	90.75	85.12	80.01	75.89
		5	32	94.27	88.24	82.21	77.98	70.78
			64	93.75	87.75	81.72	76.41	69.89

logical approximation attacks have the average accuracies of 97.76%, 96.55% and 94.92%, respectively. Therefore, the experimental results show that MPUF, rMPUF cMPUF and XOR PUF are all vulnerable to the proposed ML attacks.

To test the impact of noise on our proposed approximation attacks, we also conduct experiments under different noise levels. The prediction accuracies for $n = 32$ and 64 are shown in Table V. In addition, we also reevaluate the reliability of MPUF family and XOR PUF under different noise levels, because reliability is the theoretical maximum of the modeling accuracy. The results are shown in Table IV. Compared with MPUF and rMPUF, the reliability of cMPUF is more susceptible to noise due to the XOR existing in its structure. Furthermore, we find that the reliability of $(k, k+1)$-cMPUF is very close to that of k-XOR PUF. When $= 0.2$, the reliabilities of cMPUF and XOR PUF even drop to nearly 80%. However, even under different noise levels, the prediction accuracies of MPUFs and XOR PUF are close to the reliability, which demonstrates that our proposed approximation attacks can resist the impact of noises.

The attack on MPUF in [17] is based on the reliability of the selection input Arbiter PUF. The output of MPUF can be determined by modeling the selection input with the reliability based attack and data input with LR algorithm individually. In this paper, instead of following the two-step strategy proposed in [17], we proposed two approximation attacks to model the MUX structure from the global and local aspects. In the proposed two approximation attacks, the accuracy of a single Arbiter PUF in MPUF does not contribute to the modeling of MPUF. Take then, 3)-MPUF as an example, assuming the selection input R_1^s and R_2^s are determined, then we substitute the value of R_2^s and R_1^s into the MPUF model (see Eqn. (19)). In the ideal case, the remaining selection input R_0^s can be correctly predicted. However, R_0^s can also be the false value, resulting in a mistaken S and a mistaken output of MPUF. Similar analysis on rMPUF and cMPUF can get the same conclusion. In addition, to test the accuracy of each individual Arbiter PUF in the approximation attack, we substitute all the linear delay models of Arbiter PUFs trained by our ML algorithms into the MUX structure. If these Arbiter PUF models are accurate suf ciently, the modeling accuracy of MPUF output will be close to 100%. However, our experimental results show that the accuracy is low, which demonstrates that the proposed attack does not successfully model all Arbiter PUFs in the MPUF because our strategy is to model the MUX structure instead of individual Arbiter PUF in the MPUF.

In addition, we also conduct a detailed experiment to prove the effectiveness of the approximation attack on the MUX structure. Assume that all selection and data inputs in an MPUF are known (modeling accuracies are 100%), we apply these determined inputs to the MPUF model (see Eqn. (18)), and the nal modeling accuracy of MPUF can reach 99.9%. Furthermore, the experimental results show that the prediction accuracies of the approximation attack still can reach 99.9% (see Table VI) when only the selection input is determined. In addition, each predicted response of data input Arbiter PUF R_i^d can also match the true one because the selection input Arbiter PUFs can be modeled accurately based on Eqn. (19). Assuming that R^s is uniform, the corresponding S will follow a uniform distribution in $[0; 2^k - 1]$, and all the data input Arbiter PUFs can be modeled with enough CRPs. Take k, 3)-MPUF for example, the experimental results are shown in Table VI.

C. MPUF Scale and Scalability Analysis

In practical applications, security, reliability and overhead are the important metrics to evaluate PUFs. In order to resist modeling attacks, obfuscation techniques such as structural non-linearization [18], [35] and CRP obfuscation [16], [36], [37] have been proposed. However, the higher security brought by the obfuscation may incur high overhead and decrease the reliability of PUF responses. Therefore, we need to nd an optimal point among these metrics.

In this section, we will discuss the overhead and reliability of MPUF. As shown in Fig. 2, Fig. 3 and Fig. 4, the number of Arbiter PUFs in an (n, k)-MPUF, (n, k)-rMPUF and (n, k)-cMPUF is $2^k + k$, $2^{k+1} - 1$ and $2^k - 1 + k$, respectively. The overhead of MPUF is determined by the number of Arbiter PUFs and hence increases exponentially with k. In addition,

TABLE VI
MODELING ACCURACIES OF $(n, 3)$-MPUF WITH DETERMINED S

n	Acc. of Single Arbiter PUF (%)								Average Acc. of Arbiter PUF (%)	Acc. of prediction (%)
	A_0^d	A_1^d	A_2^d	A_3^d	A_4^d	A_5^d	A_6^d	A_7^d		
32	99.86	99.85	99.73	99.76	99.85	99.87	99.77	99.83	99.81	99.89
64	99.57	99.71	99.79	99.74	99.68	99.79	99.77	99.62	99.71	99.78
128	99.23	99.44	99.51	99.41	99.19	99.50	99.44	99.18	99.36	99.48

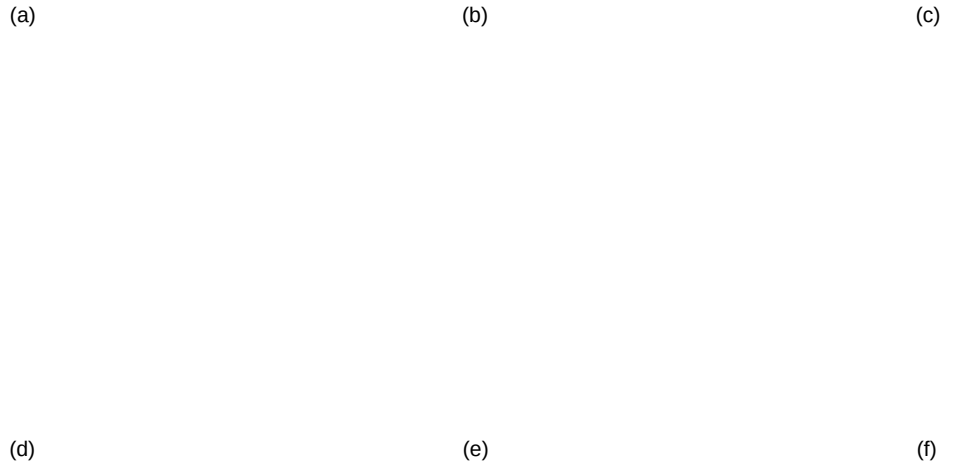

Fig. 9. (a) Double logarithmic plot of prediction error rate ε on the ratio of N_{Arb_CRP} and the dimension of $\dim(\ ) = n+1$, (b) Double logarithmic plot of prediction error rate ε on the ratio of N_{MPUF_CRP} and parameter size $(n+1)\ (2^k + k)$, (c) Double logarithmic plot of prediction error rate ε on the ratio of N_{rMPUF_CRP} and parameter size $(n+1)\ (2^{k+1} - 1)$, (d) Double logarithmic plot of prediction error rate ε on the ratio of N_{cMPUF_CRP} and parameter size $(n+1)\ (2^{k-1} + k)$, (e) Number of iterations in the logical approximation algorithm until convergence occurs, plotted in dependence of the training set size N_{MPUF_CRP}, (f) Number of iterations in the filter-based algorithm until convergence occurs, plotted in dependence of the training set size N_{MPUF_CRP}.

we have reevaluated the reliability of MPUF, cMPUF and rMPUF under different noise levels. As shown in Table IV, the reliability of MPUF family decreases with the increasing of k and σ, e.g., when $\sigma = 0.2$, the reliability of $(n, 5)$-cMPUF is 79.78%. In this paper, we take $k = 3, 4$ for MPUF, $k = 2, 3$ for rMPUF, and $k = 4, 5$ for cMPUF as the optimum to balance the overhead and reliability.

Additionally, we can get the number of required CRPs in the training and derive the computational complexity by the scalability analysis [20] on our proposed ML algorithms. In [20], Rührmair et al. have proved that the minimal number of CRPs (N_{CRP}) to model an n-stage Arbiter PUF should obey the relation

$$N_{Arb\_CRP} = O(n/\varepsilon); \tag{25}$$

where n is the challenge size, and ε is the prediction error rate. The above relation is confirmed by our experiments in Fig. 9 (a) which shows the double logarithmic plot of ε on the ratio of training CRPs and the dimension of $\dim(\ )$ (see Section II-B, $\dim(\ ) = n+1$). A first-degree polynomial equation is used to fit the data points in the plot. Based on the fitting parameters, an empirical formula of N_{Arb_CRP} is concluded as

$$N_{Arb\_CRP} \approx 0.5\ (n+1)/\varepsilon. \tag{26}$$

For an l-XOR PUF, the required number of CRPs in the training set N_{XOR_CRP} is related to the scale of XOR PUF. In the modeling of XOR PUF, the total dimension of feature vectors is $(n+1)^l$. Hence, N_{XOR_CRP} scales exponentially with the number of Arbiter PUFs:

$$N_{XOR\_CRP} \approx O((n+1)^l). \tag{27}$$

For an (n, k)-MPUF, the number of deployed Arbiter PUF is $2^k + k$. Hence, the number of required CRPs for MPUF

N_{MPUF_CRP} should obey

$$N_{MPUF\_CRP} \quad O((n+1) \quad (2^k + k)=\text{"}): \qquad (28)$$

Similarly, we can get the required CRPs for an (n, k)-rMPUF and (n, k)-cMPUF:

$$N_{rMPUF\_CRP} \quad O((n+1) \quad (2^{k+1} \quad 1)=\text{"}); \qquad (29)$$

$$N_{cMPUF\_CRP} \quad O((n+1) \quad (2^{k \ 1} + k)=\text{"}): \qquad (30)$$

As shown in Fig. 9 (b), (c) and (d), the experimental results have proven these relations.

The computational complexity of ML algorithms can be determined by the training time and estimated by the total number of basic operations N_{BOP}. The N_{BOP} is determined by: (1) the number of iterations in the optimization procedure before the training converges; (2) a time complexity function determines the number of elementary calculations in ML algorithms. In what follows, the two factors are explained in detail.

Based on our experimental data, we can estimate the relation between the number of iterations and N_{CRP}. As shown in Fig. 9 (e) and (f), the number of iterations is proportional to $\log(N_{CRP})$ and hence has the complexity of $O(\log(N_{CRP}))$.

As discussed in Section III and IV, our proposed two ML algorithms use approximation functions to approximate the nonlinear mapping relationship of MPUF. As shown in Eqn. (17) – (23), all approximation functions only contain matrix additions and constant calculations. Assuming a constant calculation is an elementary operation that obeys the complexity of $O(1)$, the total complexity can be measured by the number of elementary operations. For the logical approximation modeling (see Eqn. (17)), the training for MPUF is an iterative process for calculating $(x) = \text{sigmoid}(x)$, which can be seen as an elementary operation that has the complexity of $O(1)$. Therefore, the total complexity can be estimated by the number of calculations for (x). In the iteration process, the total number of calculations for (x) in an (n, k)-MPUF is the number of MUXes $2^k \quad 1$, as well as the rMPUF and cMPUF. Therefore, the complexity of logical approximation should obey

$$C_{LA} = O((2^k \quad 1) \quad N_{CRP} \quad \log(N_{CRP})): \qquad (31)$$

For the lter-based modeling on MPUF, the t_{MPUF} and S is calculated by the two summation formulas Eqn. (19) and Eqn. (20). Hence, the number of elementary operations in an (n, k)-MPUF is the total number of summations $2^k + k$. In addition, for the (n, k)-rMPUF and (n, k)-cMPUF, the number of summations is $2^{k+1} \quad 1$ and $2^{k \ 1} + k$, respectively. Therefore, the complexity of lter-based modeling can be expressed as

$$C_{FB\_MPUF} = O((2^k + k) \quad N_{CRP} \quad \log(N_{CRP})); \qquad (32)$$

$$C_{FB\_rMPUF} = O((2^{k+1} \quad 1) \quad N_{CRP} \quad \log(N_{CRP})); \qquad (33)$$

$$C_{FB\_cMPUF} = O((2^{k \ 1} + k) \quad N_{CRP} \quad \log(N_{CRP})): \qquad (34)$$

For the logical approximation attack on l-XOR PUF, the calculation for l-stage XOR is an iterative process (see Eqn. (24)). Hence the complexity can be computed by the number of iterations. In an l-XOR PUF, the number of iterations is l and the computational complexity of XOR PUF is

$$C_{LA\_XOR} = O(l \quad N_{CRP} \quad \log(N_{CRP})): \qquad (35)$$

TABLE VII
THE MINIMUM NUMBER OF CRPS FOR MPUF AND XOR PUF

PUF	k(l)	Modeling Attack	Minimal CRPs (64-bit)
MPUF	3	approximation attack	1:5 10^4
	4	approximation attack	3:2 10^4
rMPUF	2	approximation attack	1:8 10^4
	3	approximation attack	4:4 10^4
cMPUF	4	approximation attack	2:1 10^4
	5	approximation attack	4:2 10^4
XOR PUF	4	approximation attack	2:2 10^4
	5	approximation attack	6:3 10^4
	4	RProp [30]	1:0 10^4
	5	RProp [30]	4:5 10^4

D. Comparative Analysis

Different from the traditional ML attacks (e.g. RProp [20], [30]), the mathematical model of approximation attack is not 100% precise and there is a deviation between the true model and approximation model. As the number of XOR increases, the deviation between the true model and the approximation model will continue to accumulate. Table VII shows the minimum number of CRPs required to model MPUF and XOR PUF. We can see that the minimum number of CRPs for the approximation attack is higher because the ML algorithm used in approximation attacks is ANN, which requires more CRPs than the LR used in the RProp-based attack. However, the approximation attack has stronger robustness and fault tolerant to noise. It can be seen from Table V that the modeling accuracy of the approximation attack under different levels of noise is very close to its theoretical maximum (i.e., reliability). Therefore, the approximation attack is more effective in the real world with different noises.

For the comparation of modeling dif culty between (n, k)-MPUF and l-XOR PUF, if the stage number k and l are the same, the overhead of MPUF is larger. Hence the number of CRPs required to model the MPUF is higher. If the number of Arbiter PUF deployed in MPUF and XOR PUF is close, e.g. (64, 3)-MPUF and 64-bit 11-XOR PUF, MPUF is easier to model. This is because the response of each Arbiter PUF in the XOR PUF contributes to the nal output while only one data input in the MPUF is selected as the output.

VI. RELATED WORK

In many IoT applications, resources such as CPU, memory, and battery power are limited and cannot support classic cryptography security solutions. As a new lightweight security primitive, the security of strong PUF has attracted much attention in academia and industry. In 2004, Lim et al [29] demonstrated that ML attacks are a huge threat to strong PUFs for the rst time. Arbiter PUF, which was claimed to be physically unclonable, can be cloned successfully when attackers collect only a small number of CRPs. For a 64 64-bit Arbiter PUF, the modeling accuracy is over 95% by collecting

only 650 pairs of CRPs. Since then, various ML attacks on strong PUFs have been developed [32], [20], [38].

The current ML attacks for strong PUFs can be classed into three types: 1) ML attacks at software level. The complexity of ML attacks increases exponentially with the PUF scale or the number of nonlinear logical structures in the PUF [20], [38]. Therefore, the time required for modeling will be out of the acceptable range; 2) side-channel attacks at the hardware level. Side-channel attacks utilize information leakage such as time consumption, power consumption, electromagnetic radiation and fault injection during the operation of hardware circuit to attack cryptographic devices. Common side-channel attacks include power side-channel [39], time side-channel [40], electromagnetic [41], differential fault analysis [42], and photonic emission analysis [43]. In particular, photonic emission analysis utilizes the photonic emission information from the backside of the chip to physically characterize Arbiter PUF [43]. However, a simple side channel attack sometimes is dif cult to build a high accuracy model for PUFs; 3) hybrid attacks based on side-channel and ML. To reduce the modeling time and enhance the attack ability, hybrid attacks exploit the side-channel information to assist ML algorithms for modeling PUFs [44].

The hybrid attack that combines power consumption analysis with ML [31], [45] is one of the most effective attacks. In [45], Rührmair *et al.* proposed a differential power analysis (DPA)-based side-channel attack which can differentiate the subtle changes by comparing the power tracing before and after generating the response to extracte the power consumption of the XOR gate and convert it into an identi able form related to the response. In [31], Becker proposed a hybrid attack that combines the correlation power analysis (CPA) and covariance matrix adaptation Evolutionary Strategy (CMA-ES) to conduct the modeling attack with the power correlation coef cient as the tness function. CPA is a more practical attack on PUF than DPA. However, hybrid attacks suffer several issues. Take the power side-channel attack for example, noise has a great effect on the accuracy of modeling PUF. The relationship between the number of power tracing required for DPA attacks n and the signal-to-noise ratio (SNR) is $n \propto 1/\mathrm{SNR}$, which indicates that the reduction in SNR linearly increases the number of power tracings required for a DPA attack [46]. Therefore, noise is an important factor to increase the dif culty of DPA attacks. In addition, since the power consumption of PUF is extremely small and the measurement of side-channel information requires special equipment, it is dif cult to extract the power consumption information of the PUF module from the measured power tracing of the chip under the interference of noise with reasonable cost.

The reliability-based attack [31] is another hybrid attack that combines response reliability with ML. Reliability is used to assess the stability of PUF responses in different environments. Ideally, the responses should be stable with the same challenge under multiple measurements. The reliability-based attack uses the reliability information of the response instead of CRPs to model the PUF. In [31], Becker proposed a reliability-based attack on XOR-PUF. The key idea is to perform repeated measurements for the same challenge and observe the stable and ipped response bits. Then the obtained reliability information is used to conduct an ML attack based on CMA-ES. In [42], Delvaux *et al.* observed that the delay difference for a speci c challenge of an Arbiter PUF is proportional to the unreliability of the corresponding response bit if the environmental conditions are stable. Based on this observation, attackers analyze the measured reliability: if a response for a given challenge is unstable, it is likely that the corresponding delay difference is close to zero; if a response bit has a high reliability for a given challenge, it is likely that the delay difference is large. Then attackers compute the Pearson correlation coef cient [47] between reliability and hypothetical reliability, and utilize it as the tness function in the CMA-ES to select the ttest delay vector $\mathbf{l}^T$ of each Arbiter PUF. Here, a divide-and-conquer strategy is used to attack each Arbiter PUF individually while the unreliability introduced by the other Arbiter PUFs is seen as noises. Compared with other ML attacks, the divide-and-conquer strategy results in only a linear increase in the required number of CRPs for increasing numbers of XORs [31]. Recently, some defense methods are proposed to prevent against the reliability-based attack. Take rMPUF [17] for example, not every bit of selection input in rMPUF contributes to the response generation compared with MPUF, which means that the correlation between the response reliability and the input is weakened. Therefore, the required CRPs for modeling rMPUF is greatly increased, so that it is dif cult for attackers to conduct a reliability-based attack. In addition, we can eliminate the way for attackers to obtain reliability information. If the challenges are generated by both the tag and the veri er, attackers cannot send the same challenge twice and observe the reliability of the responses. Once the reliability information is not available, it is dif cult to conduct the reliability-based attack successfully.

VII. Conclusion

As a new hardware security primitive, the security of PUFs must be evaluated rigorously before putting into practice. This paper proposes a general framework for conducting ML attacks on strong PUFs. Two approximation attacks based on the framework, named logical approximation method and lter-based global approximation, are proposed to build the soft models with high accuracy for MPUF and its two variants, rMPUF and cMPUF. In addition, we conduct the logical approximation attacks on XOR PUF for comparison. Security analysis and experimental results prove that the MPUF family and XOR PUF are vulnerable to the approximation attacks, and the modeling accuracies are high under different noise levels.

Acknowledgement

The authors would like to thank Mr. Chen Li and Mr. Yehao Kong for editing the paper. J. Shi and Y. Lu contributed equally.

References

[1] Wikipedia, “Internet of things,” https://en.wikipedia.org/wiki/Internet_of_things.

[2] K. L. Lueth, "IoT 2018 in Review: The 10 Most Relevant IoT Developments of the Year," https://iot-analytics.com/iot-2018-in-review/.

[3] IoT_Analytics, "IoT Security Market Report 2017-2022," https://iot-analytics.com/product/iot-security-market-report-2017-22/.

[4] J. Zhang, G. Qu, Y. 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, Jul 2014.

[5] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, "A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-per-Device Licensing," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1137–1150, 2016.

[6] J. Zhang, B. Qi, Z. Qin, and G. Qu, "HCIC: Hardware-Assisted Control-Flow Integrity Checking," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 458–471, feb 2019.

[7] J. Zhang, "A Practical Logic Obfuscation Technique for Hardware Security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 1193–1197, mar 2016.

[8] J. Zhang and G. Qu, "Physical unclonable function-based key sharing via machine learning for iot security," *IEEE Transactions on Industrial Electronics*, 2019, Doi:10.1109/TIE.2019.2938462.

[9] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and RO Sum PUFs via environmental changes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1701–1713, 2014.

[10] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, sep 2002.

[11] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly PUF protecting IP on every FPGA," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 67–70.

[12] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight Secure PUFs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, no. 1, nov 2008, pp. 670–673.

[13] P. Tuyls, G.-J. Schrijen, B. Škorić, J. Van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 369–383.

[14] J. Zhang, X. Tan, Y. Zhang, W. Wang, and Z. Qin, "Frequency Offset-Based Ring Oscillator Physical Unclonable Function," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 711–721, oct 2018.

[15] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 148–160.

[16] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe, "PUF-FSM: A Controlled Strong PUF," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 1104–1108, 2018.

[17] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen, "A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 403–417, 2018.

[18] A. Vijayakumar and S. Kundu, "A Novel Modeling Attack Resistant PUF Design Based On Non-linear Voltage Transfer Characteristics," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 653–658.

[19] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring PUF: A new architecture for strong physical unclonable functions," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*. IEEE, 2011, pp. 134–141.

[20] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 237–249.

[21] D. P. Sahoo, P. H. Nguyen, D. Mukhopadhyay, and R. S. Chakraborty, "A Case of Lightweight PUF Constructions: Cryptanalysis and Machine Learning Attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1334–1343, 2015.

[22] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *2012 IEEE international workshop on Information forensics and security (WIFS)*. IEEE, 2012, pp. 37–42.

[23] J. Miao, M. Li, S. Roy, Y. Ma, and B. Yu, "SD-PUF: Spliced Digital Physical Unclonable Function," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 927–940, 2018.

[24] L. Liu, H. Huang, and S. Hu, "Lorenz Chaotic System-Based Carbon Nanotube Physical Unclonable Functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1408–1421, 2018.

[25] Wikipedia, "Artificial neural network," https://en.wikipedia.org/wiki/Artificial_neural_network.

[26] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[27] M. D. M. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, 2016.

[28] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual conference on Design automation - DAC '07*, 2007, p. 9.

[29] J. Lee, Daihyun Lim, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, 2004, pp. 176–179.

[30] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2015, pp. 17–31.

[31] G. T. Becker, "The gap between promise and reality: On the insecurity of XOR arbiter PUFs," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 535–555.

[32] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *2008 IEEE International Test Conference*. IEEE, 2008, pp. 1–10.

[33] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proceedings of the IEEE international conference on neural networks*, vol. 1993. San Francisco, 1993, pp. 586–591.

[34] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016.

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

[36] Y. Jing, H. Yu, and X. Li, "RPUF: Physical Unclonable Function with Randomized Challenge to resist modeling attack," in *IEEE Asian Hardware-oriented Security and Trust*, 2017, pp. 1–6.

[37] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 37–49, 2014.

[38] U. Ruhrmair, J. Solter, 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.

[39] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[40] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Annual International Cryptology Conference*, 1996, pp. 104–113.

[41] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive EM attack on FPGA RO PUFs and countermeasures," in *Proceedings of the Workshop on Embedded Systems Security*, 2011, pp. 1–9.

[42] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 137–142.

[43] S. Tajik, E. Dietz, S. Frohmann, H. Dittrich, D. Nedospasov, C. Helfmeier, J.-P. Seifert, C. Boit, and H.-W. Hübers, "Photonic Side-channel Analysis of Arbiter PUFs," *Journal of Cryptology*, vol. 30, no. 2, pp. 550–571, 2017.

[44] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined Modeling and Side Channel Attacks on Strong PUFs." *IACR Cryptology ePrint Archive*, p. 632, 2013.

[45] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2014, pp. 476–492.

[46] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008, vol. 31.

[47] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

Junye Shi is currently working toward his B.E. degree in the College of Computer Science and Electronic Engineering, Hunan University. His research interests include hardware security, physical unclonable function and artificial intelligence (AI) security.

Yang Lu is currently working toward his B.E. degree in the College of Computer Science and Electronic Engineering, Hunan University. His research interests include hardware security, and machine learning.

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. Since 2017, he has joined Hunan University. His current research interests include hardware/hardware-assisted security, artificial intelligence security, and emerging technologies.

Prof. Zhang is a recipient of the Hu-Xiang Youth Talent, and 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, and 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.