An Efficient and Scalable Modeling Attack on Lightweight Secure Physically Unclonable Function

Phuong Ha Nguyen and Durga Prasad Sahoo, *Graduate Student Member, IEEE*

Abstract—The Lightweight Secure Physically Unclonable Function (LSPUF) was proposed as a secure composition of Arbiter PUFs with additional XOR based input and output networks. But later, researchers proposed a Machine Learning (ML) based modeling attack on x-XOR LSPUF, and they also empirically showed that pure ML based modeling is not computationally scalable if the parameter x of x-XOR LSPUF is larger than nine. Besides this pure computational attack using only challenge-response pairs (CRPs), there are other proposals for modeling attacks on LSPUF using timing and power side-channel information, reliability information and photonic side-channel information of an LSPUF instance. In this paper, we proposed another pure computational attack (i.e. without any side-channel information) on multibit output LSPUF variants using both cryptanalysis and ML techniques together. We, first, cryptanalyse the output network of LSPUF to reduce the computational efforts required by previously proposed pure ML based modeling of an x-XOR LSPUF. Specifically, we model an LSPUF instance, while its output bit is defined as x-XOR PUF, using the ML modeling of y-XOR PUF where $y < x$. From the computational complexity view point, our proposed modeling attack is efficient and scalable than previously proposed pure ML based modeling of LSPUFs with respect to both data and time complexities. We demonstrate the effectiveness of our proposed attack using the *Matlab* based simulation of LSPUFs and LSPUFs implemented on Xilinx Artix-7 Field Programmable Gate Arrays (FPGAs).

Index Terms—Cryptanalysis, hardware–intrinsic security, lightweight secure PUFs, logistic regression, machine learning, modeling attack, physically unclonable function (PUF).

1 Introduction

PHYSICALLY Unclonable Function (PUF) has emerged as a promising hardware security primitive, since first introduced in [1], [2]. The defining characteristic feature of a PUF is the *unpredictable* and *unclonable instance–specific input–output (or challenge–response) mapping*. The proposed usages of PUFs include diverse application scenarios, a few among which are: device identification and authentication [2], binding software to hardware platforms [3], secure storage of cryptographic secrets [4], secure protocol design [5]. A PUF hardware instance is operated through a challenge/response mechanism: for a given challenge, corresponding instance-specific response is generated that depends on the intrinsic and random physical properties of the embedding hardware [1], [2].

The main security vulnerability of PUFs is the so–called "modeling attacks" (MA) [6]–[8]. In these attacks, an adversary tries to build a mathematical model, potentially with high probability of success, of a given PUF instance using the challenge-response pairs (CRPs) and typically machine learning algorithms such as *Support Vector Machine* (SVM) and *Logistic Regression* (LR). Arbiter PUF [2] is mostly used as strong PUF design and it is vulnerable to modeling attacks.

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, INDIA–721302. E-mail:phuongha.ntu@gmail.com, dpsahoo@cse.iitkgp.ernet.in.

Past experience has shown that reaching an acceptable balance between design overhead and modeling attack resistance is extremely challenging for PUFs. As a potential solution, the *Lightweight Secure PUF* (LSPUF) was introduced in [9]. Structurally, a LSPUF consists of three layers: an input network, a PUF layer and an output network. The input and output networks are simple and lightweight logic networks; the PUF layer is based on a set of lightweight but insecure *Arbiter PUFs* (APUFs) [2]. The LSPUF achieves several improvements over a previously proposed single–output PUF having structural similarities with it, called the *XOR Arbiter PUF* (XOR PUF) [10], in which the PUF output is generated by simple XOR-ing of the constituent APUFs.

The LSPUF is considered as a landmark design in the existing PUF literature, because it is significantly lightweight, and has been shown (through consideration of the security of the input network, the PUF layer and output network) to be secure against various attacks (e.g. reverse engineering, machine learning and statistical modeling [6], [7], replay attack, and reconfigurability–specific vulnerabilities). In addition, LSPUF can have multiple outputs, which is considered a desirable property as this enhances the "extractable entropy" [11] of the PUF. Note that the critical feature responsible for the security of LSPUF is that the values at output of APUF layer are hidden from an adversary. If these values are exposed to an adversary for a given applied challenge, then since APUFs can be modeled extremely accurately

by standard machine learning techniques, the LSPUF can be easily and accurately modeled.

In [6], [12], [13], the modeling attack on LSPUF is discussed and it is also shown that the attack becomes computationally infeasible if the number of APUFs (say x) being XOR-ed to define a 1-bit LSPUF output is greater than six [6]. In [12], author shows that x-XOR PUF is vulnerable to modeling attack when $x < 10$ using parallel implementation of LR algorithm. Besides these pure computational attacks, various side-channel (power, timing and photonic side-channel information) assisted modeling attacks have also been proposed to enhance the effectiveness of such modeling attacks [14]–[17]. The attack presented in [16] is pure side-channel attack, i.e., it does not require any CRP. A modeling attack based on the reliability information of PUF instance and Evolution Strategy (ES) is also proposed in [15]. In [18], we exploited a property of LSPUF's output network to develop a cryptanalytic attack on any multibit output LSPUF, without focusing on modeling attack.

In this paper, we propose an efficient and pure computational modeling attack on multibit output LSPUF based on cryptanalysis of LSPUF output network and traditional LR based modeling of XOR PUF. In our previous work in [18], we proposed only the cryptanalysis of LSPUF by exploiting the weakness of its output network; *there was no complete modeling attack proposal*. In this work, we extend our cryptanalysis attack in [18] to a modeling attack of LSPUF, thus *reaching higher level of prediction accuracy*. Our proposed attack is purely computational, of low computational complexity, and does not require access to any side–channel information such as power profile. In this paper, we discuss the security analysis of multibit output LSPUF variants in the modeling attack, and all security claims and reported attack results are related to multibit output LSPUF variants. Specifically, our proposed attack is not applicable to single output LSPUFs.

1.1 Our Contributions

In brief, our contributions are as follows.

1) We propose a modeling attack based on our previously proposed cryptanalysis attack on LSPUF in [18]. In this modeling attack, we use y-XOR PUF modeling using LR as primitive operation to model a x-XOR LSPUF, where $y < x$. Thus, the time and data complexities of this attack is significantly less than that of the attack presented in CCS-2010 [6], and this attack is more scalable than the modeling attack presented in RFIDSec-2015 [12] even without parallel implementation of LR algorithm. Since the modeling accuracy of y-XOR PUF using LR is comparatively higher than x-XOR PUF ($x > 2$) due to less noise in the CRPs of y-XOR PUF, our proposed modeling attack is feasible even for x-XOR PUF with large x value.
2) We present an interesting observation regarding the LR based modeling attack of XOR PUF, exploited in our proposed model building technique, as: *modeling of an XOR PUF using LR [8] enables modeling of the individual constituent APUFs*, is called as "model separability." Since LSPUF's output bits are defined as XOR-ing of APUF outputs, this observed feature is also applicable to LSPUF modeling. Although LR based MA of XOR PUF has been discussed by several authors [6], [7], and mathematically analysed in detail [8], as far as we are aware, this is the first time this particular feature has been exploited for MA. As would be evident in Section 4.3, this feature has deep implications in reducing the complexity of the proposed attack. We consider this insight of "model separability" to be one of the major contributions of this paper.
3) In addition, we derive the necessary and sufficient conditions for multibit output LSPUF structure to enable accurate MA (cf. Section 4). We call those particular LSPUF variants susceptible to accurate MA as "fully modeled LSPUFs".
4) We demonstrate the proposed attack on the *Matlab* based simulated LSPUF, and implementation of LSPUF on a Field Programmable Gate Array (FPGA) platform. We provide results for 64-bit and 128-bit 9-XOR LSPUFs.

1.2 Organization of the Paper

The rest of paper is organized as follows. In Section 2, we provide the structural description, mathematical specification and notion of cryptographic security in the context of LSPUF, and then describe previous attempts of modeling LSPUFs. In Section 3, we explain various properties of LSPUF's output netwrok. Section 4 present the detailed security analysis of LSPUF in the context of modeling attack. Section 5 provides experimental results to validate our proposed attack. In Section 6, we compare the proposed attack with all existing attacks on LSPUFs. We conclude the paper in Section 7.

2 Preliminaries

2.1 Notations

We use following notation system in the rest of the paper. Lowercase and uppercase letters in bold font refer to a vector and a matrix, respectively, e.g. $\mathbf{a}$ is a vector while $\mathbf{E}$ denotes a matrix. A vector with m-components is represented as follows: $\mathbf{a} = (a_0, \ldots, a_i, \ldots, a_{m-1})^{\mathsf{T}}$, where a_i is ith component of vector $\mathbf{a}$. The normal lowercase letter denotes a scalar, e.g. n. The $\oplus$ stands for Exclusive-OR (XOR) operation or modulo-2 addition. Set is represented by calligraphic uppercase letter, e.g. set $\mathcal{D}$. The Hamming weight of a binary vector $\mathbf{a}$ by $\mathrm{HW}(\mathbf{a})$.

2.2 Lightweight Secure PUF (LSPUF)

Figure 1 shows the general architecture of a LSPUF [9], with n-bit challenge $\mathbf{c}$ and m-bit response $\mathbf{o} = (o_0, \ldots, o_{m-1})$. Structurally, the LSPUF consists of three main

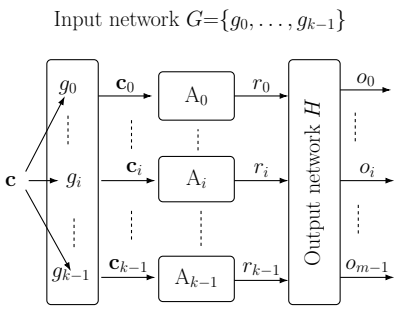

Fig. 1: The achitectural overview of Lightweight Secure PUF [19].

building blocks: (1) the input network $\{g_0, g_1, \ldots, g_{k-1}\}$ producing k n-bit input chunks $\mathbf{c}_0, \ldots, \mathbf{c}_{k-1}$ based on a single n-bit input $\mathbf{c}$ (e.g, $\mathbf{c}_i = g_i(\mathbf{c})$, where $i = 0, \ldots, k-1$); (2) k instances of n-bit APUFs [2] $\{A_0, A_1, \ldots, A_{k-1}\}$, and, (3) the output network producing m output bits $o_0, \ldots, o_{m-1}$ based on the k bits $r_0, \ldots, r_{k-1}$ produced as outputs of the k APUF instances. The output bit $o_i, i = 0, \ldots, m-1$ is defined as:

$$o_i = \bigoplus_{l=0}^{x-1} r_{((i+s+l) \mod k)}, \tag{1}$$

where $r_j = A_j(\mathbf{c}_j)$, $j = 0, \ldots, k-1$, and $x(< k)$ and s are structural parameters chosen by the designer which eventually determine the level of security offered by an LSPUF.

Note that it is possible to generate multiple output bits using a relatively small number of APUF outputs by the above scheme, a feature considered to be a major advantage for LSPUF. Also, the use of input network reduces correlation among the challenge-response pairs (CRPs) of APUF instances, which increases the security of LSPUF [19]. However, since the attack developed by us is agnostic of the structure of the input network, we would not be discussing it any further in this paper. In the context of MA, addition of the input network only modifies the "feature vectors" for the LR algorithm—it does not change the circuit parameters learned by the machine learning algorithm (cf. [8, Chapter 4] for details). In the rest of the paper, without loss of generality we assume the parameter $s = 0$ to facilitate the description of mathematical analysis; however, the proposed attack is effective even when $s \neq 0$. We would also assume $m > 1$, as mentioned in Section 1.

2.3 Security Notion of LSPUF

Definition 1 (Security Notion [1]). *A LSPUF instance with n-bit challenge* $\mathbf{c}$ *and m-bit response* $\mathbf{o}$ *is considered to be* secure *if and only if there is no* ideal attack *such that: (i) the probability of predicting m-bit response for a given challenge is greater than $\frac{1}{2^m}$; (ii) the time complexity of the attack is less than 2^n PUF queries, and, (iii) the data complexity, i.e., the number of Challenge-Response pairs (CRPs) required to execute the attack is less than 2^n.*

An *ideal attack* is referred to an attack where the noise in measurement setup is not taken into account. This assumption is reasonable because the security of a PUF should be inherent in its design, instead of being provided by the measurement noise. This assumption of ideality is not a major concern in our context, as through proper care in the design and measurement setup, the effect of noise resulting from experimental and design bias can be made arbitrarily small. Our proposed attack has the added advantage of being largely agnostic to measurement noise, as it requires only CRP values, rather than current or delay measurements.

2.4 Related Works

Although LSPUF was proposed as a secure PUF composition, a significant body of research has been done on its mathematical modeling in [6], [12]–[18] to show many security threats on LSPUF. Typically, all these modeling attacks (MA) can be categorized into following classes based on the type of information and techniques are employed.

1) **Pure ML.** In this kind of modeling attack, the adversary builds a model, based on the CRPs of LSPUF, for each output bit o_i of an LSPUF by using x-XOR PUF modeling approach as reported in [6], [14]. In case of LR based modeling, adversary needs comparatively more CRPs for LSPUF than simple XOR PUF modeling to achieve same level of modeling accuracy due to the presence of input network, and data complexity of attack in both cases increase exponentially with the increasing values of x and challenge size n. So, this attack does not explicitly employ the weakness in the output network of LSPUF. As reported in [6], pure ML (specifically LR) based modeling becomes computationally infeasible if LSPUF design parameter $x \geq 6$, in case of sequential implementation of LR algorithm. Later in [12], authors provided parallel implementation of LR algorithm, and they could build a model for 9-XOR 64-bit LSPUF. So, in literature we can find the modeling of 9-XOR 64-bit LSPUF, and beyond that pure ML modeling is not scalable due to exponential computational-overhead.

2) **Combined cryptanalysis and ML.** First cryptanalysis attack on LSPUF was proposed in [18] by us. In this attack, we have exploited the weakness of LSPUF's output network. This was the first attack on LSPUF using its output network explicitly. We called this attack *as the cryptanalysis instead of modeling because adversary cannot build a complete model of LSPUF*, but he can predict the m-bit response of LSPUF with probability significantly larger than $1/2^m$. Our proposed work in this paper is the complete modeling attack on LSPUF by using cryptanalysis approach proposed in [18].

3) **Combined Side-channel and ML.** In [14], authors employed the power and timing side-channel of LSPUF instance to derive information related to the responses of individual APUFs. They have used a modified version of LR based APUF modeling to incorporate the side-

channel information. In this attack, data complexity is linearly related to the value of parameter x.

4) **ML with Reliability Information.** In this attack [15], adversary's objective is to build a model for an XOR PUF by modeling the individual APUFs independently. An interesting fact is that individual APUF are modeled based on the output of XOR PUF. This attack does not employ the CRPs directly in modeling, instead the reliability information obtained from multiple evaluation of challenges is exploited. This attack is powerful, but a limiting point is that modeling requires a reasonable amount of unreliable challenges and multiple evaluations of those challenges. In practice, for example in PUF-based protocols, adversary might not have any control on challenge selection and number of times it is to be evaluated. Moreover, in XOR PUF design where error correction circuit or majority voting circuit is used to improve the reliability, reliability information based on the multiple evaluations is not directly accessible. This implies that this attack is only feasible to bare LSPUF design, and helpful for PUF designer to evaluate the security of LSPUF design.

5) **Photonic Emission analysis with Lattice Basis Reduction.** In this attack [16] of XOR PUF, authors used the concept of lattice basis reduction approach to find the delay parameters of an APUF model. For this attack, adversary needs to measure the delays of APUF trigger signals at the output of last switching stage. In this case, authors exploited the transient photonic emission from IC back side to measure the path delays. Although in this attack no extra circuit is required, it needs to open the IC back side to measure the photonic emission efficiently. This attack is feasible only when adversary has physical access to PUF device.

6) **Combined Fault Analysis and ML.** In this case, modeling of an XOR PUF is also performed by modeling individual APUFs. Modeling of individual APUF is performed by directly using the CRPs of APUF and traditional ML based modeling approach. To collect the CRPs of individual APUF, adversary introduces fault such that LSPUF output depends only on the target APUF. In [17], authors used the laser based fault insertion technique. To collect CRPs of ith APUF, laser gun can be used to modify Look Up Table (LUT) values (in FPGA) of other APUFs such that trigger signal does not reach to the clock input of arbiter circuit of corresponding APUF circuit. This attack approach can reduce the data complexity of attack, but PUF circuit is to be restarted x times to model individual APUFs of x-XOR LSPUF, which is an impractical assumption.

As mentioned earlier, in this paper our proposed modeling attack is purely computational with significantly low data and computational complexities (especially compared to [6], [12]) even if $x \geq 6$, and does not require access to any side–channel information. To the best of our knowledge, this is the first modeling attack based on the output network of multibit output LSPUF variants. In addition, we can model 9-XOR 128-bit LSPUF without parallel implementation of LR algorithm as in [12].

3 Analysis of LSPUF Output Network and Its Vulnerabilities

In this section, we present some important observations about the output network of a multibit output LSPUF variant ($m > 1$). Based on these observations, in next sections we show that all multibit output LSPUF variants do not satisfy the security notion described in Definition 1.

3.1 Linear Independence of LSPUF Outputs

Let us define the binary matrix $\mathbf{E} = \mathbf{E}_{m\times k} = \{e_{i,j}\}$, where entries $e_{i,j} = 1$ iff output bit o_i of LSPUF is dependent on r_j (i.e. the jth APUF output), $i = 0, \ldots, m-1$, $j = 0, \ldots, k-1$, and $e_{i,j} = 0$ otherwise. An example $\mathbf{E}$ matrix is shown in Table 1a, following Eq. (1), for the parameter set k =7, $m = 6$, $x = 6$ and $s = 0$. It should be evident that the rank of the matrix $\mathbf{E}$ is $m = 6$, implying the output bits $o_0, \ldots, o_{m-1}$ are linearly independent. We would get a similar observation for cases where $s \neq 0$. This observation is generalized in the following lemma:

Lemma 1. *For $m < k$, the output bits $o_0, \ldots, o_{m-1}$ of LSPUF are linearly independent.*

Proof. This fact follows from the nature of the matrix $\mathbf{E}$. We can apply *Gaussian Elimination* to show that the rank of the matrix $\mathbf{E}$ is m in a straightforward way. It implies that all actual outputs $o_0, \ldots, o_{m-1}$ are linearly independent. □

3.2 Virtual Output Bits and Its Properties

In order to develop our attack, we introduce the concept of specially constructed *virtual output bits*, by performing a suitable linear transformation on the output bits of LSPUF. From Eq. (1), we have the following observation which is important to develop our attack:

TABLE 1: Example of Matrix $\mathbf{E}$ and $\mathbf{H}$ for $k = 7$, $m = 6$, $x = 6$, $s = 0$.

(a) Matrix $\mathbf{E}$

	r_0	r_1	r_2	r_3	r_4	r_5	r_6
o_0	1 ($e_{0,0}$)	1 ($e_{0,1}$)	1 ($e_{0,2}$)	1 ($e_{0,3}$)	1 ($e_{0,4}$)	1 ($e_{0,5}$)	0 ($e_{0,6}$)
o_1	0 ($e_{1,0}$)	1 ($e_{1,1}$)	1 ($e_{1,2}$)	1 ($e_{1,3}$)	1 ($e_{1,4}$)	1 ($e_{1,5}$)	1 ($e_{1,6}$)
o_2	1 ($e_{2,0}$)	0 ($e_{2,1}$)	1 ($e_{2,2}$)	1 ($e_{2,3}$)	1 ($e_{2,4}$)	1 ($e_{2,5}$)	1 ($e_{2,6}$)
o_3	1 ($e_{3,0}$)	1 ($e_{3,1}$)	0 ($e_{3,2}$)	1 ($e_{3,3}$)	1 ($e_{3,4}$)	1 ($e_{3,5}$)	1 ($e_{3,6}$)
o_4	1 ($e_{4,0}$)	1 ($e_{4,1}$)	1 ($e_{4,2}$)	0 ($e_{4,3}$)	1 ($e_{4,4}$)	1 ($e_{4,5}$)	1 ($e_{4,6}$)
o_5	1 ($e_{5,0}$)	1 ($e_{5,1}$)	1 ($e_{5,2}$)	1 ($e_{5,3}$)	0 ($e_{5,4}$)	1 ($e_{5,5}$)	1 ($e_{5,6}$)

(b) Matrix $\mathbf{H}$

	o_0	o_1	o_2	o_3	o_4	o_5
v_0	1	1	0	0	0	0
v_1	0	1	1	0	0	0
v_2	0	0	1	1	0	0
v_3	0	0	0	1	1	0
v_4	0	0	0	0	1	1

Observation 1 ([18]).

$$o_i \oplus o_{i+1} = r_{((i+s) \mod k)} \oplus r_{((i+x+s) \mod k)}, \quad (2)$$

where $0 \leq i \leq m-2$.

Let us define the virtual output bits $v_0, \ldots, v_{m-2}$ as a linear transformation of the actual outputs $o_0, \ldots, o_{m-1}$ of LSPUF:

$$v_i = o_i \oplus o_{i+1} \text{ for } i = 0, \ldots, m-2. \quad (3)$$

Note that we provided the definition of virtual output bits $v_0, \ldots, v_{m-2}$ in [18] to enable a cryptanalysis attack on multibit output LSPUF variants. At that time, we could not define the *last virtual bit* v_{m-1} and it makes the reverse linear transformation of $o_0, \ldots, o_{m-1}$ from $v_0, \ldots, v_{m-2}$ infeasible. Hence, the work presented in [18] was concluded with a cryptanalysis attack, instead of modeling attack. In this paper, we provide two different definitions of *last virtual bit* v_{m-1}, and then we develop two different attack strategies to model multibit output LSPUF variants in Section 4.

To achieve reverse linear transformation, we have to ensure that virtual output bits $v_0, \ldots, v_{m-1}$ are linearly independent. Now, we show that $v_0, \ldots, v_{m-2}$ are linearly independent, and later we define v_{m-1} such that $v_0, \ldots, v_{m-1}$ are linearly independent. Let us define a binary matrix $\mathbf{H} = \mathbf{H}_{(m-1)\times m} = \{h_{i,j}\}$, where entries $h_{i,j} = 1$ iff virtual output bit v_i of LSPUF is dependent on o_j (i.e. the jth LSPUF output), $i = 0, \ldots, m-2$, $j = 0, \ldots, m-1$, and $h_{i,j} = 0$ otherwise. An example $\mathbf{H}$ matrix is shown in Table 1b for $m = 6$. It is observed that the rank of the matrix $\mathbf{H}$ is $m - 1 = 5$. This can be generalized as the following lemma:

Lemma 2. *The virtual output bits $v_0, \ldots, v_{m-2}$ are linearly independent.*

Proof. Since all actual outputs $o_0, \ldots, o_{m-1}$ are independent (see Lemma 1) and the rank of matrix $\mathbf{H}$ is $m-1$, all virtual outputs $v_0, \ldots, v_{m-2}$ are linearly independent. □

3.3 Reliability of Virtual Outputs

Interestingly, the virtual output bits are less noisy than actual output bits. For example, let two output bits, o_1 and o_2, of a LSPUF instance be defined as follows:

$$o_1 = r_1 \oplus r_2 \oplus r_3 \oplus r_4 \oplus r_5 \oplus r_6$$
$$o_2 = r_2 \oplus r_3 \oplus r_4 \oplus r_5 \oplus r_6 \oplus r_7.$$

Let bits $r_1, \ldots, r_7$ be noisy outputs of APUFs. Then, o_1 and o_2 are supposed to be very noisy because they depend on six noisy APUF outputs. However, if $v_1 = o_1 \oplus o_2$, then v_1 is significantly less noisy than either o_1 or o_2, because the noise in $r_2, \ldots, r_6$ appears twice (in o_1 and o_2). Thus, the repeated noisy PUF outputs cancel themselves out, and it makes the attack more robust in noisy environment. The impact of this property can be observed in Table 5 (in Section 5) obtained from the FPGA implementation of LSPUF.

4 Proposed Modeling Attack on LSPUF

4.1 Attack Overview

Our proposed attack consists of two following phases:

1) **Modeling Phase.** This is the beginning step of our proposed attack. Objective of this phase is to build a model, denoted as v_i^s, for each virtual bit $v_i, i = 0, \ldots, m-1$. The virtual output vector $\mathbf{v} = (v_o, \ldots, v_{m-1})^\mathsf{T}$ is defined from the actual output bits as $\mathbf{v} = \mathbf{M}\mathbf{o}^\mathsf{T}$, where $\mathbf{o} = (o_0, \ldots, o_{m-1})$ and $\mathbf{M}_{m\times m}$ is the linear transformation matrix defined based on definition of virtual outputs $v_0, \ldots, v_{m-1}$. The matrix $\mathbf{M}$ is similar to the matrix $\mathbf{H}$ (cf. Section 3.2 and Table 1b) with an additional row (last row) for virtual output v_{m-1}. This phase is summarized in Algorithm 1.
2) **Prediction Phase.** Objective of this step is to predict the response $\mathbf{o}$ to a given unknown challenge $\mathbf{c}$. We first predict the virtual output vector $\mathbf{v} = (v_o, \ldots, v_{m-1})$ corresponding to $\mathbf{c}$, where $v_i \leftarrow v_i^s(\mathbf{c})$. Subsequently, output $\mathbf{o}$ of LSPUF is computed as $\mathbf{o} = \mathbf{M}^{-1}\mathbf{v}^\mathsf{T}$. Algorithm 2 depicts the details of this phase.

Both the virtual output (say y-XOR PUF) and actual output (say x-XOR PUF) are represented by XOR operations, but in this case $y < x$. It results that ML based modeling of virtual outputs are more efficient (both in data and time complexity) than actual outputs. This is the main motivation of exploiting the models of virtual output bits in the development of attack, instead of the models of actual LSPUF's outputs. Now, we discuss two different attack strategies based on the definition of the last virtual bit v_{m-1}, but overall modeling steps are same in both the cases.

4.2 Attack Strategy-I

Here, we start with the definition of last virtual output bit v_{m-1}, as this will be the basis of the discussion in this section. Let $\mathbf{a} = (a_0, \ldots, a_{m-1})^\mathsf{T}$ with $a_0, \ldots, a_{m-1} \in \{0, 1\}$, be a set of binary coefficients, and then v_{m-1} is defined as:

$$v_{m-1} = a_0 o_0 \oplus \cdots \oplus a_{m-1} o_{m-1}. \quad (4)$$

Let us denote the number of APUFs outputs present in the expression for v_i (cf. Eqs. (3) and (4)) by $t_i, i = 0, \ldots, m-1$, and $\hat{t}$ is the threshold for MLMA resistant

Algorithm 1 Model building of an LSPUF instance

Input: A CRP set $\mathcal{S}$ of an LSPUF instance
Output: A model of the given LSPUF instance
1: Derive virtual output bits $(v_0, \ldots, v_{m-1})$ from the actual output bits $(o_0, \ldots, o_{m-1})$ of LSPUF for each CRP in $\mathcal{S}$
2: **for** $i = 0$ to $m-1$ **do**
3: Build a ML based model v_i^s for each virtual bit v_i w.r.t. the CRP set S

Algorithm 2 Response prediction of an LSPUF instance

Input: A challenge $\mathbf{c}$
Output: The predicted response $\mathbf{o}$ to challenge $\mathbf{c}$
1: **for** $i = 0$ to $m-1$ **do**
2: Compute $v_i \leftarrow v_i^s(\mathbf{c})$ { v_i^s is the model of ith virtual bit}
3: **for** $i = 0$ to $m-1$ **do**
4: Compute o_i from the $v_0, \ldots, v_{m-1}$
5: $\mathbf{o} \leftarrow (o_0, \ldots, o_{m-1})$

XOR PUF. For each virtual output v_i, if $t_i < \hat{t}$, then the model v_i^s of v_i can be built by using machine learning [6]. For each given challenge $\mathbf{c}$, we use these models to compute the values $v_0^s(\mathbf{c}), \ldots, v_{m-1}^s(\mathbf{c})$. Since the virtual output bits are obtained as a linear transformation of actual outputs, it is straightforward (by the reverse linear transformation) to calculate the values of the corresponding actual outputs for a given $\mathbf{c}$, after computing all $v_0^s(\mathbf{c}), \ldots, v_{m-1}^s(\mathbf{c})$ values. In this way the response $\mathbf{o} = (o_0, \ldots, o_{m-1})$ to a given challenge $\mathbf{c}$ can be predicted.

To perform the reverse linear transformation of actual outputs from virtual outputs, all the virtual outputs v_i must be linearly independent. Note that Lemma 2 states the linear independence of only the first $(m-1)$ components of $\mathbf{v}$. Hence, there should be no binary coefficient vector $\mathbf{b} = (b_0, \ldots, b_i, \ldots, b_{m-1})^\mathsf{T} \neq \mathbf{0}$ and $b_i \in \{0, 1\}$, such that the following equation is satisfied:

$$\mathbf{b}^\mathsf{T}\mathbf{v} = b_0(o_0 \oplus o_1) \oplus \cdots \oplus b_i(o_i \oplus o_{i+1}) \oplus \cdots \oplus b_{m-1}(\bigoplus_{j=0}^{m-1} a_j o_j) = 0. \quad (5)$$

Equation (5) can be re-written as:

$$(b_0 \oplus a_0 b_{m-1})o_0 \oplus \cdots \oplus (b_{i-1} \oplus b_i \oplus a_i b_{m-1})o_i \oplus \cdots \oplus (b_{m-2} \oplus a_{m-1}b_{m-1})o_{m-1} = 0 \quad (6)$$

Since all output bits $o_0, \ldots, o_{m-1}$ are linearly independent (cf. Lemma 1), Eq. (6) is equal to zero if and only if:

$$b_0 \oplus a_0 b_{m-1} = \cdots = b_{i-1} \oplus b_i \oplus a_i b_{m-1} = \cdots = b_{m-2} \oplus a_{m-1}b_{m-1} = 0. \quad (7)$$

Notice that Eqs. (2) and (3) together suggest that the model for each of the bits v_i where $i = 0, \ldots, m-2$ can be built by machine learning, as each of them depends on **only two APUF outputs.** However, the number of APUF outputs being XOR-ed to form v_{m-1} must be less than $\hat{t}$ to make it feasible to build a model for v_{m-1}.

Now we discuss the construction of vector $\mathbf{a}$ such that there exists the vector $\mathbf{b}$ for which Eq. (6) holds, or there is no vector $\mathbf{b}$ that satisfies Eq. (6). In other words, we study the construction of last virtual bit output. Since $v_i = o_i \oplus o_{i+1}$, $i \in [0, m-2]$, we have the following observation:

Observation 2. $o_{i_1} \oplus o_{i_2} = \bigoplus_{l=i_1}^{i_2-1} v_l$ *where* $0 \leq i_1 < i_2 \leq m-1$. *Note that if* $i_2 = i_1 + 1$, *then* $o_{i_1} \oplus o_{i_2} = v_{i_1}$ *by definition of virtual outputs.*

Next we prove two important theorems.

Theorem 1. (Sufficient condition for linear dependence of virtual outputs) *For a given* $\mathbf{a} = (a_0, \ldots, a_{m-1})^\mathsf{T}$, *virtual outputs* $v_0, \ldots, v_{m-1}$ *are* ***linearly dependent*** *if* $a_{i_1} = a_{i_2} = \cdots = a_{i_t} = 1$, $0 \leq i_1 < i_2 < \cdots < i_t \leq m-1$, $2 \leq t \leq m$, *and* t *is an* ***even*** *integer. In other words, if* $\mathrm{HW}(\mathbf{a})$ *is even, then virtual outputs* $v_0, \ldots, v_{m-1}$ *are* ***linearly dependent****.*

Proof. Let t be an even integer and $2 \leq t \leq m$. Since $a_{i_1} = a_{i_2} = \cdots = a_{i_t} = 1$, $v_{m-1} = o_{i_1} \oplus o_{i_2} \oplus \cdots \oplus o_{i_{t-1}} \oplus o_{i_t}$ $= (o_{i_1} \oplus o_{i_2}) \oplus \cdots \oplus (o_{i_{t-1}} \oplus o_{i_t})$. Based on Observation 2, v_{m-1} can be expressed as

$$v_{m-1} = (\oplus_{l=i_1}^{i_2-1} v_l) \oplus \cdots \oplus (\oplus_{l=i_{t-1}}^{i_t-1} v_l).$$

Thus, $v_0, \ldots, v_{m-1}$ are linearly dependent. □

Observation 3. *The number of* o_i *terms appearing in any expression of the form* $v_{i_1} \oplus v_{i_2} \oplus \cdots \oplus v_{i_t}, 0 \leq i_1 < \cdots < i_t \leq m-2$, *is even.*

This observation follows from the fact that the number of o_i terms presents in $v_{i_1} \oplus v_{i_2}$ or v_{i_1} is even according to Observation 1.

Theorem 2. (Sufficient condition for linear independence of virtual outputs) *For a given* $\mathbf{a} = (a_0, \ldots, a_{m-1})^\mathsf{T}$, *virtual outputs* $v_0, \ldots, v_{m-1}$ *are* ***linearly independent****, if* $a_{i_1} = a_{i_2} = \cdots = a_{i_t} = 1$, $0 \leq i_1 < i_2 < \cdots < i_t \leq m-1$, $1 \leq t \leq m$, *and* t *is an* ***odd*** *integer. In other words, if* $\mathrm{HW}(\mathbf{a})$ *is odd, then virtual outputs* $v_0, \ldots, v_{m-1}$ *are* ***linearly independent****.*

Proof. The proof is by contradiction. Assume that t is an odd integer and $v_{m-1} = o_{i_1} \oplus \cdots \oplus o_{i_t}$, but $v_0, \ldots, v_{m-1}$ are linearly dependent. Since $v_0, \ldots, v_{m-2}$ are linearly independent (cf. Lemma 2), b_{m-1} must be equal to 1 (cf. Eq. (5)). This implies that there exists a $\mathbf{b} = (b_0, \ldots, b_{m-1})^\mathsf{T}$ such that $b_{j_1} = b_{j_2} = \cdots = b_{j_q} = b_{m-1} = 1$ and $\mathbf{b}^\mathsf{T}\mathbf{v} = v_{j_1} \oplus \cdots \oplus v_{j_q} \oplus v_{m-1} = 0$, $0 \leq j_1 < \cdots < j_q \leq m-2$ and $1 \leq q \leq m-1$. This fact is equivalent to $v_{m-1} = v_{j_1} \oplus \cdots \oplus v_{j_q}$. According to Observation 3, the number of o_i terms present in v_{m-1} must be even, which implies that if $v_{m-1} = o_{i_1} \oplus \cdots \oplus o_{i_t}$, then t is even. But this contradicts the fact that t is odd. This completes the proof. □

Theorem 2 lets us construct correct values of vector $\mathbf{a}$, such that the virtual output bits are linearly independent. Among the 2^{m-1} guesses for $\mathbf{a}$ (corresponding to odd values of t), we choose that value of $\mathbf{a}$ which enables building a model for v_{m-1} with the smallest value of t_{m-1} (t_{m-1} is the number of APUFs outputs present in the expression of v_{m-1}).

Based on relationship between the LSPUF's parameters k, m and x, we have two important observations as described in Lemma 3 and Lemma 4.

Lemma 3. **(t_{m-1} in case of even k)** *If k and x are both even, with $m = k-1$ and $v_{m-1} = \bigoplus_{i=0}^{m-1} o_i$, v_{m-1} is dependent on x APUF outputs; otherwise, if x is odd, v_{m-1} is dependent on $t_{m-1} = k - x$ APUF outputs. For example, when k is even and $x = m = k-1$, $v_{m-1} = r_{k-2}$, i.e., v_{m-1} is dependent on only one APUF output.*

Lemma 4. **(t_{m-1} in case of odd k)** *If k is odd and x is even, $m = k-1$ and $v_{m-1} = \bigoplus_{i=0}^{m-2} o_i$, then v_{m-1} is dependent on only two APUF outputs; otherwise, if x is odd, v_{m-1} is dependent on $t_{m-1} = k-2$ APUF outputs.*

Due to the page limitation, we omit the proofs. The significance of these lemmas is that the adversary can have many special cases where the attack can be conducted without exhaustively searching for the right vector $\mathbf{a}$ in the entire space.

We summarize all analyzed results for the special cases for v_{m-1} in Table 2. If $t_{m-1} < \hat{t}$, a model v^s_{m-1} can be

TABLE 2: Expression of t_{m-1} for $m = k-1$

k	x	t_{m-1}
Even	Even	x
	Odd	$k-x$
Odd	Even	2
	Odd	$k-2$

built using LR based modeling of t_{m-1}-XOR PUF [7]. We call a LSPUF as **fully modeled** if the last bit v_{m-1} can be modeled by machine learning techniques, or in other words if $t_{m-1} < \hat{t}$. Otherwise, we consider the model v^s_{m-1} to be a random guess algorithm and its prediction accuracy is denoted as $p^+ = 1/2$. Note that, in this case, the efficiency of the attack is equivalent to that of the cryptanalysis attack reported in [18].

In next section, we show that the adversary can build a set of models of APUF outputs from models of virtual outputs, and we call this process *reaching the root of LSPUF security*. By using this fact, we develop an alternative attack strategy with a new definition for last virtual output bit v_{m-1}.

4.3 Attack Strategy-II

To make the explanation of attack strategy comprehensible, we recall the main idea of LR-based modeling attack on x-XORPUF, as presented in [8, Ch. 4]. In x-XOR PUF, the response r can be computed as follows:

$$r = \prod_{i=1}^{x} sgn(\mathbf{w}_i^{\mathsf{T}} \mathbf{\Phi}_i) = sgn(\prod_{i=1}^{x} \mathbf{w}_i^{\mathsf{T}} \mathbf{\Phi}_i),$$

where $\mathbf{w}_i$ and $\mathbf{\Phi}_i$ are weight vector and feature vector (or parity vector) of the ith APUF, respectively. The function $sgn(y) = 0$ if $y \leq 0$, otherwise $sgn(y) = 1$.

According to the LR algorithm, initially, the $\mathbf{w}$ is randomly generated, where $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_x)$. After that $\mathbf{w}$ is gradually updated based on the training CRPs that would approximate the behavior of of x-XOR PUF. It is evident that when $\mathbf{w}$ is constructed, all the sub-vectors $\mathbf{w}_i, i = 1, \ldots, x$ are built as well. In other words, *LR-based modeling attack constructs not only a mathematical model of x-XOR PUF, but also mathematical models of individual x APUFs.*

The following observation and lemma are crucial to develop the *Attack Strategy-II*:

Observation 4 (Separability property). *Each virtual output $v_i = o_i \oplus o_{i+1}$ for $i = 0, \ldots, m-2$ depends only on two APUFs r_i and $r_{((i+x) \mod k)}$. If the adversary can build a model v^s_i for v_i using the LR machine learning algorithm (cf. [8, Ch. 4, Eq. 4.12]), then (s)he can build two different models r^s_i and $r^s_{((i+x) \mod k)}$ corresponding to the ith and $((i+x) \mod k)$th APUFs, respectively. In other words, the models r_i and $r_{((i+x) \mod k)}$ are "separable".*

Lemma 5. *Based on Observation 4, for a LSPUF variant with m-bit outputs, adversary can build a set of APUF models $\mathcal{D} = \{r^s_0, r^s_1, \ldots, r^s_{m-2}, r^s_x, \ldots, r^s_{k-1}\}$.*

Proof. The lemma follows from Observation 4, since $v_i = r_i \oplus r_{((i+x) \mod k)}$, $i = 0, \ldots, m-2$. Note that if $m > x$ then all APUF models $r^s_0, \ldots, r^s_{k-1}$ are in set $\mathcal{D}$; otherwise the APUF models $r^s_{m-1}, \ldots, r^s_{x-1}$ are not in $\mathcal{D}$. □

Lemma 5 implies an important fact: models of all $(k+m-x-1)$ APUFs $r_0, r_1, \ldots, r_{m-2}, r_x, \ldots, r_{k-1}$ are available to adversary after successful modeling of virtual outputs $v_0, \ldots, v_{m-2}$, and only $(x-m+1)$ APUFs $r_{m-1}, \ldots, r_{x-1}$ are unknown to adversary. Thus, the security of LSPUF should be re-evaluated based on this fact.

4.3.1 Defining Last Virtual Bit v_{m-1}

From Eq. (3) and Table 1a, it follows that

$$v_{m-1} = \mathbf{a}^{\mathsf{T}}\mathbf{o} = \bigoplus_{i=0}^{m-1} a_i o_i = \bigoplus_{i=0}^{m-1} a_i (\bigoplus_{j=0}^{k-1} e_{i,j} r_j)$$
$$= \bigoplus_{j=0}^{k-1} (\bigoplus_{i=0}^{m-1} a_i e_{i,j}) r_j = \bigoplus_{j=0}^{k-1} d_j r_j,$$

where $d_j = \bigoplus_{i=0}^{m-1} a_i e_{i,j}$. It is evident that d_j is associated to the jth column in matrix $\mathbf{E}$, and in turn determines whether r_j is present in the expression for v_{m-1}, $j = 0, \ldots, k-1$ (r_j is only present in v_{m-1} if $d_j = 1$). In addition to the presence of r_j in v_{m-1}, it is also crucial whether r_j is in the set $\mathcal{D}$. Suppose $r_j \in \mathcal{D}$ and hence the adversary has a model of r_j. If now r_j is present in the expression of v_{m-1}, then adversary can reduce the computational burden of modeling v_{m-1} by simply ignoring r_j from the expression of v_{m-1}, as the model for r_j has been already recovered. This is equivalent to making $d_j = 0$ in the expression of v_{m-1}, and this leads to a simplified matrix $\mathbf{E}^m$. For example, Table 3a describes LSPUF variant $(k, m, x, s) = (7, 3, 6, 0)$ and the Table 3b presents the modified (simplified) matrix $\mathbf{E}^m$

TABLE 3: Matrix $\mathbf{E}$ and $\mathbf{E}^m$.

(a) Matrix $\mathbf{E}$

	r_0	r_1	r_2	r_3	r_4	r_5	r_6
o_0	1	1	1	1	1	1	0
o_1	0	1	1	1	1	1	1
o_2	1	0	1	1	1	1	1

(b) Modified Matrix $\mathbf{E}^m$

	r_0	r_1	r_2	r_3	r_4	r_5	r_6
o_0	0	0	1	1	1	1	0
o_1	0	0	1	1	1	1	0
o_2	0	0	1	1	1	1	0

modified after using the fact that the set $\mathcal{D} = \{r_0, r_1, r_6\}$. Formally, matrix $\mathbf{E^m}$ can be defined based on the matrix $\mathbf{E}$ as:

$$e^m_{i,j} = \begin{cases} 0 & \text{if } r_j \in \mathcal{D}, \\ e_{i,j} & \text{otherwise.} \end{cases} \tag{8}$$

For the matrix $\mathbf{E}^m$, we have the following important observation:

Observation 5. *It consists of following two cases:*

1) *If $m \leq x$, then*

$$e^m_{i,j} = \begin{cases} 1 & \forall i \in [0, m-1], j \in [m-1, x-1], \\ 0 & \textit{otherwise.} \end{cases} \tag{9}$$

2) *If $m > x$ (or $m-1 > x-1$), then $\forall i \in [0, m-1], j \in [0, k-1] : e^m_{i,j} = 0$.*

The main reasons for Observation 5 are as follows:

1) In case $m \leq x$, all the $(x-m+1)$ APUF outputs $r_j \notin \mathcal{D}$ *are necessarily present in every actual output* $o_0, \ldots, o_{m-1}$. This fact implies that all corresponding $e^m_{i,j}$ must be 1. All the APUF outputs $r_j \in \mathcal{D}$ which are present in each o_i will define the corresponding entry $e^m_{i,j} = 0$.
2) In case $m > x$, the set of $r_j \notin \mathcal{D}$ is empty and then all $e^m_{i,j}$ are equal to 0. In other words, all APUFs $r^s_0, \ldots, r^s_{k-1}$ are in $\mathcal{D}$ and then the adversary knows all the models $r^s_0, \ldots, r^s_{k-1}$.

Let us denote t_u as the number of APUFs present in the expression of v_{m-1}, but their models are not present in $\mathcal{D}$. Let us focus on the case: $m \leq x$. From Eq. (9), and since $d_j = \bigoplus_{i=0}^{m-1} a_i e_{i,j}$ and HW($\mathbf{a}$) is an odd number for any considered $\mathbf{a}$ (see Theorem 2), we must have $d_{m-1} = \cdots = d_{x-1} = 1$. In other words, for **any chosen value of vector a**, the expression for v_{m-1} will contain $t_u = (x-m+1)$ APUF outputs $r_{m-1}, \ldots, r_{x-1}$ which cannot be directly modeled by the adversary. Since all 2^{m-1} values of $\mathbf{a}$ are equivalent in terms of usage, we select $\mathbf{a} = (1, 0, \ldots, 0)$, which implies $v_{m-1} = o_0$. The significance of this fact is that for a given LSPUF variant, we can decide the vulnerability of that LSPUF to MLMA by comparing the value of $t_u = (x-m+1)$ with $\hat{t}$— **if $t_u < \hat{t}$, the LSPUF is vulnerable to MLMA.**

Now we consider the case: $m > x$ (cf. Observation 5). Since all $e^m_{i,j}$ are equal to 0 and $d_j = \bigoplus_{i=0}^{m-1} a_i e_{i,j}$, from the view of adversary, v_{m-1} consists of all known APUFs for any chosen value of $\mathbf{a}$. In other words, all vectors $\mathbf{a}$ are same to the adversary for executing attack. Hence, the adversary can choose $\mathbf{a} = (1, 0, \ldots, 0)$, which implies $v_{m-1} = o_0$.

We summarize the *Attack Strategy-II* of LSPUF in the following theorem:

Theorem 3. *Let (n, m, k, x, s) be a given LSPUF variant with $s = 0$ and $m > 1$, and let its virtual output bits are defined as:*

$$v_i = \begin{cases} o_i \oplus o_{i+1} & \textit{if } i = 0, \ldots, m-2 \\ o_0 & \textit{if } i = m-1. \end{cases} \tag{10}$$

*If the number of individually unmodeled APUF outputs in the expression for v_{m-1}, denoted by $t_u = (x-m+1)$, is less than $\hat{t}$, then the given LSPUF variant is considered as **"fully modeled"**. Otherwise, we say that the LSPUF variant can be only cryptanalyzed. It means that for an unknown challenge $\mathbf{c}$ the adversary can randomly guess v_{m-1}, and use the models of other virtual bits $v_0, \ldots, v_{m-2}$ to obtain the actual output $\mathbf{o}$ of LSPUF.*

4.3.2 Model Building of Last Virtual Bit v_{m-1}

Now, we discuss about the building of a model v^s_{m-1} for v_{m-1}. In the context of modeling attack on PUF, there are two main sources of noise:

1) noise in the CRPs of PUF used to train a model, and
2) degree of imperfection (inaccuracy) inherent in the model itself.

To simplify the explanation of our proposed attack strategy, we assume that there is *no noise in the CRPs of LSPUF, i.e., the reliability of LSPUF output is 100%*. In Section 5, we provide the details of the attack developed based on LSPUF implemented on FPGA, to demonstrate the impact of CRP-noise on the modeling accuracy.

As defined in Eq. (10),

$$v_{m-1} = o_0 = r_0 \oplus \cdots \oplus r_{m-2} \oplus r_{m-1} \oplus \cdots \oplus r_{x-1}$$

and $t_u = (x-m+1)$. Let p_r be the average prediction accuracy for the models $r^s_0, \ldots, r^s_{m-2}$, and p_v be the average prediction accuracy for the models $v^s_0, \ldots, v^s_{m-2}$. Since the adversary cannot access the values of $r_0, \ldots, r_{k-1}$, (s)he cannot estimate the modeling accuracy p_r for each model $r^s_i \in \mathcal{D}$. However, the p_r is always close to p_v because the models $r^s_i \in \mathcal{D}$ are obtained from the models $v^s_0, \ldots, v^s_{m-2}$. Since the adversary can build the models $v^s_0, \ldots, v^s_{m-2}$ and can access the outputs $v_0, \ldots, v_{m-2}$, (s)he can estimate p_v accurately. We consider this as a method to compute p_r in practice, i.e., $p_r \approx p_v$.

Let p^+ be the prediction accuracy for model v^s_{m-1}. For the sake of brevity, we provide Fig. 2 to summarize all the prediction accuracies presented in this section. Let us recall $\hat{t}$, which is the threshold for MLMA (LR is the used ML approach) resistant XOR PUF. Now we consider several cases in sequence.

4.3.2.1 **Case-I** $m \leq x - \hat{t} + 1$: In this case $t_u = (x-m+1) \geq \hat{t}$. This implies that we cannot build a model v^s_{m-1} of $v_{m-1} = o_0$ by using MLMA [6]. As a solution, we assign a random guess for v_{m-1} (or for o_0). Thus, the model v^s_{m-1} is simply a random guess algorithm and the prediction accuracy of v^s_{m-1} is $p^+ = 1/2$.

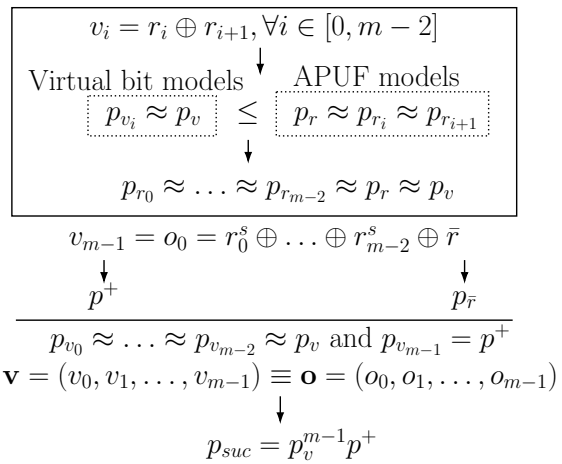

Fig. 2: All prediction accuracies considered in the modeling of v_{m-1} in Attack Strategy-II.

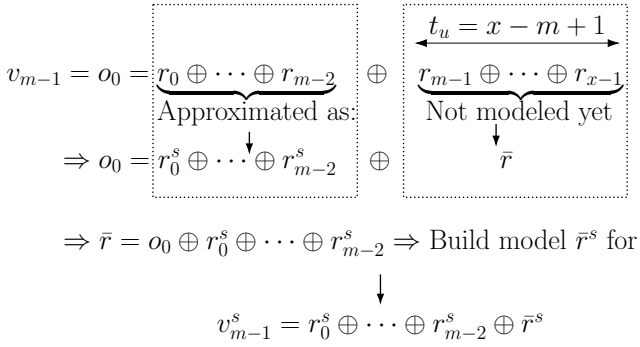

Fig. 3: Modeling of the last virtual bit v_{m-1}. The models $r_0^s, \ldots, r_{m-2}^s$ are obtained from the models of $v_0^s, \ldots, v_{m-2}^s$ according to the separability property (cf. Observation 4). Model v_{m-1}^s can be obtained as $v_{m-1}^s = r_0^s \oplus \cdots r_{m-2}^s \oplus \bar{r}^s$.

4.3.2.2 **Case-II** $(x+1-\hat{t}) < m < (x+1)$: In this case $0 < t_u < \hat{t}$. Since the $r_0^s, \ldots, r_{m-2}^s \in \mathcal{D}$, the adversary only needs to build the model for $\bar{r} = r_{m-1} \oplus \cdots \oplus r_{x-1}$. Since the adversary has no information on $r_{m-1}, \ldots, r_{x-1}$, (s)he has to compute $\bar{r} = o_0 \oplus r_0 \oplus \cdots \oplus r_{m-2}$ as described in Fig. 3. Let $\bar{r}^s$ be the model for $\bar{r}$, and it is constructed as follows:

- **Phase 1:** For a given challenge $\mathbf{c}$, the response $\bar{R}$ is computed as $\bar{R} = o_0(\mathbf{c}) \oplus r_0^s(\mathbf{c}) \oplus \cdots \oplus r_{m-2}^s(\mathbf{c})$. This step is repeated for sufficient number of challenges.
- **Phase 2:** After collecting a sufficiently large set of CRPs $\{(\mathbf{c}, \bar{R})\}$, the model $\bar{r}^s$ is built using MLMA [7].

The model v_{m-1}^s of v_{m-1} is then computed as:

$$v_{m-1}^s = r_0^s \oplus \cdots \oplus r_{m-2}^s \oplus \bar{r}^s. \tag{11}$$

Now, we discuss about the problems in construction of v_{m-1}^s. Note that for the sake of analysis, we assume that prediction accuracy of each of these models is $p_r < 1$. In addition, we assume that the reliability of all APUFs $r_0, \ldots, r_{k-1}$ are 100%, and all the actual outputs $o_0, \ldots, o_{m-1}$ are 100% reliable.

It is important to know the reliability or correctness of the constructed response $\bar{R}$ of $\bar{r}$ for a given challenge $\mathbf{c}$ which is used to build the model $\bar{r}^s$. This information is important to decide whether *the adversary can build the model $\bar{r}^s$ or not*. The reason is that if the reliability of constructed $\bar{R}$ is too less, then the adversary can not build an accurate model $\bar{r}^s$.

It is evident that for a given challenge $\mathbf{c}$, in case the reliability of o_0 is 100%, and if all the models $r_0^s, \ldots, r_{m-2}^s$ produce correct guesses, i.e., $r_i^s(\mathbf{c}) = r_i(\mathbf{c}), \forall i = 0, \ldots, m-2$, then the correct value $\bar{R}$ is produced, i.e., $\bar{R} = o_0(\mathbf{c}) \oplus r_0^s(\mathbf{c}) \oplus \cdots \oplus r_{m-2}^s(\mathbf{c}) = o_0(\mathbf{c}) \oplus r_0(\mathbf{c}) \oplus \cdots \oplus r_{m-2}(\mathbf{c}) = r_{m-1}(\mathbf{c}) \oplus \cdots \oplus r_{x-1}(\mathbf{c})$. This event happens with a probability $\epsilon_1 = p_r^{m-1}$.

Since the probability of the event $\forall i = 0, \ldots, m-2 : r_i^s(\mathbf{c}) = r_i(\mathbf{c})$ is $\epsilon_1 = p_r^{m-1}$ for a given challenge $\mathbf{c}$, then the probability that *there exists* $i \in [0, \ldots, m-2]$ *such that* $r_i^s(\mathbf{c}) \neq r_i(\mathbf{c})$ for a given challenge $\mathbf{c}$ is $\epsilon_2 = 1 - p_r^{m-1}$. Since $\bar{R}$ is computed as XOR of $r_0^s, \ldots, r_{m-2}^s$, the correct $\bar{R}$ can be produced if *there is an even number of models* r_i^s *such as* $r_i^s(\mathbf{c}) \neq r_i(\mathbf{c})$ and the probability of this event is $\epsilon_2/2 \approx \frac{1-p_r^{m-1}}{2}$ in general.

Hence, the probability of the event that a correct $\bar{R}$ can be produced for a given challenge $\mathbf{c}$ is

$$\epsilon_3 = \epsilon_1 + \frac{\epsilon_2}{2} = \frac{1}{2} + \frac{p_r^{m-1}}{2}. \tag{12}$$

This probability ϵ_3 places an important restriction on the correctness of CRPs computed in Phase 1. Based on the value of ϵ_3, the adversary can decide whether a model $\bar{r}^s$ can be built or not. For instance, if m becomes big enough to make $p_r^{m-1} \approx 0$, then $\epsilon_3 \approx 1/2$ and the adversary has no advantage from knowledge of the set $\mathcal{D}$ even if a very large number of CRPs $(\mathbf{c}, \bar{R})$ is collected. In other words, the adversary can use random guess algorithm as a model of v_{m-1} and the prediction accuracy of v_{m-1}^s is $p^+ = 1/2$.

Since the adversary cannot access the real output of $\bar{r}$, *the real prediction accuracy $p_{\bar{r}}$ of model $\bar{r}^s$ cannot be computed*. Actually, $p_{\bar{r}}$ can be computed as follows:

1) The adversary constructs $v_{m-1}^s = r_0^s \oplus \cdots \oplus r_{m-2}^s \oplus \bar{r}^s$.
2) The adversary collects $(\mathbf{c}, o_0)$ pairs to compute the prediction accuracy of model v_{m-1}^s which is denoted as p^+.
3) Since p^+ and p_r, $p_{\bar{r}}$ are connected as described in Eq. (13), the adversary can compute the *real* $p_{\bar{r}}$ now. Note that Eq. (13) is derived as explained for ϵ_3.

$$\begin{aligned} p^+ &= p_r^{m-1} p_{\bar{r}} + \frac{1 - p_r^{m-1} p_{\bar{r}}}{2} \\ &= \frac{1}{2} + \frac{p_r^{m-1} p_{\bar{r}}}{2} \approx \frac{1}{2} + \frac{p_v^{m-1} p_{\bar{r}}}{2}. \end{aligned} \tag{13}$$

4.3.2.3 **Case-III** $m \geq x+1$: In this case, $t_u \leq 0$ or all $r_0^s, \ldots, r_{x-1}^s$ are present in $\mathcal{D}$, and then the model of $v_{m-1}^s = r_0^s \oplus \cdots \oplus r_{x-1}^s$. As discussed previously, the prediction accuracy p^+ of v_{m-1}^s is $p^+ = 1/2 + \frac{p_r^x}{2}$.

4.4 The Relationship Between p^+, x and m in Practice

Theoretically, the prediction accuracy p^+ heavily depends on m in Case-II (cf. Section 4.3.2.2), and on x in Case-III (cf. Section 4.3.2.3). If we take a closer look at Case-II, we see that it corresponds to $m \leq x$. Thus, eventually, p^+ only depends on x. Since p^+ is drastically reduced when x increases, the designer may choose x big enough to prevent the modeling attack on v_{m-1}. Since all the APUF outputs $o_0, \ldots, o_{m-1}$ are not 100% reliable, the parameters x and m should be less than ten [7], [9], to make the reliability of LSPUF output $\mathbf{o}$ good enough (see Section 5 for the FPGA implementation results). Thus, increasing x (and m) is not a good practical solution to prevent the modeling attack on v_{m-1}. In other words, our proposed attack is feasible on any practical LSPUF variants.

4.5 Success Probability of the Proposed Attack

For the sake of simplicity, we assume that modeling accuracy values of all models $v^s_0, \ldots, v^s_{m-2}$ are same as p_v and the modeling accuracy of v^s_{m-1} is p^+. The attack is said to be successful if the adversary can correctly predict the entire output $\mathbf{o}$ of LSPUF for a given challenge $\mathbf{c}$. Let us denote the success probability of the proposed attack as p_{suc}, and it is computed by using Eq. (14).

$$p_{suc} = p_v^{m-1} p^+ \tag{14}$$

From the experimental result in Section 5, we observed that $p_v \in [0.92, 0.96]$ for FPGA-based LSPUF and $p_v \approx 0.99$ for simulated LSPUF. Even assuming $p^+ = \frac{1}{2}$, $p_{suc} \gg p_{rand} = \frac{1}{2^m}$. We have presented the experimental results of our proposed attack in Section 5. All the results described in Section 5 confirms the efficiency of our proposed attack, i.e., $p_{suc} \gg p_{rand} = \frac{1}{2^m}$.

4.6 Complexity of the Proposed Attack

4.6.1 Time Complexity

Let T_{ML} denote the average time to build each of the $v^s_0, \ldots, v^s_{m-2}$ models by using the LR technique [6], and T_{m-1} be the time required to build model v^s_{m-1}.

Note that we have proposed two different attack strategies, and they differ only in the definition of v_{m-1}. Here, we discuss the modeling complexity of the **Attack Strategy-II**, and complexity analysis for the **Attack Strategy-I** is almost similar.

In case of modeling of v_{m-1}, depending on the value of m and x, we have three cases. For Case-I, the adversary makes a random guess for v_{m-1}, and in Case-III, the adversary does not need to employ extra effort to build model v^s_{m-1} because (s)he can use set $\mathcal{D}$ for this purpose. So, in these cases $T_{m-1} = 0$. On the other hand, for Case-II, the adversary needs to build a model for v_{m-1} with the time complexity $T_{\text{ML}_{m-1}}$, i.e., $T_{m-1} = T_{\text{ML}_{m-1}}$.

Finally, let T_{I} be the time required (in **Prediction Phase**) to compute $\mathbf{o}$ to a given challenge $\mathbf{c}$ based on the output of models $v^s_0, \ldots, v^s_{m-1}$ by solving an $m \times m$ system of linear equations (modulo 2), which has complexity $\mathcal{O}(m^3)$. Then, the total time required is $T = (m-1) \times T_{\text{ML}} + T_{m-1} + T_{\text{I}}$. In practice, usually $m \ll n$, thus the time complexity of the attack is less than that of exhaustive search, i.e., $\mathcal{O}(2^n)$ (cf. Definition 1).

TABLE 4: Complexities of proposed attack

	v^s_{m-1} is a ML-based model	v^s_{m-1} is a random guess
Prediction accuracy	p^+	$1/2$
Data complexity N	$N = \max\{N_{\text{ML}}, N_{m-1}\}$	N_{ML}
Time complexity T	$(m-1) \times T_{\text{ML}} + T_{m-1} + T_{\text{I}}$	$(m-1)T_{\text{ML}} + T_{\text{I}}$
Attack accuracy p_{suc}	$p_v^{m-1} p^+$	$\frac{p_v^{m-1}}{2}$

4.6.2 Data Complexity

Let N_{ML} denote the average number of CRPs to build each of the $v^s_0, \ldots, v^s_{m-2}$ models by using the LR technique [6], and N_{m-1} be the number of CRPs required to build v^s_{m-1}. Let N denote the total data complexity. It is evident that $N = \max\{N_{\text{ML}}, N_{m-1}\}$ number of CRPs $(\mathbf{c}, \mathbf{o})$ are required to build all $\mathbf{v^s}$. Note that if v^s_{m-1} is considered as a random guess algorithm, then $N_{m-1} = 0$, and in practice $N \ll 2^n$ (cf. Section 5).

We summarize the results of complexity analysis in Table 4. Next we provide the experimental validation of the proposed attack strategies.

5 Experiment and Results

In this section, we validate the proposed attack strategies using both simulated and FPGA implemented 64-bit and 128-bit LSPUFs.

5.1 Experimental Setup

To validate the attack, we have implemented 64-bit and 128-bit LSPUFs on five Xilinx Artix-7 (XC7A100T) FPGAs. We have also simulated LSPUF to validate the proposed attack in noise-free scenario. Reliability is an important quality metric of PUF instances and it has a direct impact on the modeling accuracy. In case of simulated PUF instances, we have considered that the PUF instances are perfectly (100%) reliable. Whereas, in FPGA implementation, PUF instances are not perfectly reliable due to the dynamic noise generated during the PUF evaluation, e.g., variation in the supply voltage and ambient temperature. Table 5 reports the reliability of FPGA implemented LSPUF instances. Reported reliability is measured at normal operating condition (at room temperature with standard supply) based on the CRPs obtained from 11 different evaluations. So, reader can consider the reported reliability as the best case reliability, and we have demonstrated the proposed attack in this scenario.

Another important aspect is how to define the *golden* CRPs for a PUF instance, which is not perfectly reliable. We have used majority voting strategy, to yield golden

TABLE 5: Reliability (%) of actual output bits and virtual output bits of FPGA implemented 9-XOR LSPUF ($k = 10, m = 9, x = 9$)

$n^\dagger$	Type	Reliability (%)										
		$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$	o	v
64	o_i	97.08	97.10	97.10	97.10	97.05	97.11	97.06	97.07	97.08	96.11	
	v_i	99.31	99.30	99.29	99.34	99.34	99.33	99.37	99.35	99.66		96.50
128	o_i	96.87	96.86	96.88	96.87	96.81	96.82	96.82	96.90	96.88	95.81	
	v_i	99.27	99.26	99.24	99.32	99.38	99.38	99.29	99.22	99.58		96.24

† Challenge size of LSPUF in bits.
Note: Reliability is measured using 100000 CRPs. The o_i and v_i are actual and virtual output bits of LSPUF, respectively. Last two columns of the table represent the reliability of entire m-bit actual and virtual responses, respectively.

TABLE 6: Uniformity (%) of output bits of 9–XOR LSPUF ($k = 10$, $m = 9$, $x = 9$)

Platform	$n^\star$	Actual Output Bits (o_i)								
		o_0	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
Simulated	64	49.90	49.76	50.14	49.97	49.86	50.01	49.79	49.79	49.86
	128	49.91	49.90	49.93	50.13	49.98	50.25	49.91	50.05	50.19
FPGA	64	49.37	49.59	49.29	49.53	49.56	49.44	49.67	49.61	49.44
	128	49.11	49.24	49.26	49.18	49.58	49.58	49.51	49.30	49.31

* Challenge length of LSPUF in bits.
Note: Uniformity is measured using 100000 CRPs.

CRPs, over the CRPs obtained from 11 evaluations of LSPUF instance over 11 different time instants at normal operating condition. In addition, the uniformity of LSPUF is reported in Table 6. It shows that the distribution of 0's and 1's in the individual PUF output bits are reasonably uniform.

Now, we provide experimental results to demonstrate the effectiveness of the proposed attack on these reference LSPUF designs. To build the models of virtual output bits, we have used in-house *Matlab* code for implementing Logistic Regression (with Rprop+ optimization) based model of XOR PUF [8].

As discussed in Section 3.3, the reliability values of virtual outputs are significantly greater than that of actual outputs, and it can be observed in Table 5. From the reported results in Table 5, it is evident that the reliability values of $\mathbf{o}$ and each $o_i, i = 0, \ldots, m-1$ are poor when $x = 9$ and $m = 9$. It implies that the reliability of actual output bit o_i is sensitive to the parameter x. Thus, the value of x cannot be arbitrarily large in practice to make LSPUF robust against modeling attack, and based on our experimental results it can be $x < 10$.

Since reliability of o_i is not 100%, reliability of LSPUF output $\mathbf{o}$ depends on the parameter m. It is observed that $x = 9$ and $m = 9$ parameter setting produces a poor reliability value for $\mathbf{o}$. Hence, in order to achieve a good reliability for $\mathbf{o}$, m should not be large, e.g. $m < 10$. Subsequently, we show that the proposed attack is feasible for all practical instances of LSPUF. Now, we illustrate the attack for certain chosen implementations for LSPUF to make our validation process comprehensible to readers.

Although the reported results are for PUF instances mapped on a single FPGA board, we observed similar results for the PUFs mapped on the other boards, with the performance figures within $\pm 1\%$ of the reported results in each case.

5.2 Results of the Proposed Attack

5.2.1 Special Case of Attack Strategy-I

To demonstrate our modeling attack for special cases as described in Lemma 3 and Lemma 4, we have used an LSPUF variant with following parameter setting: $k = 10, x = 9, m = 9$. For this parameter setting, there is a closed form of $\mathbf{a} = (1, 1, 1, 1, 1, 1, 1, 1, 1)$, i.e., virtual bit v_{m-1} is the bitwise XOR of all actual output bits, and it results $v_{m-1} = r_{k-2}$ (i.e., $v_8 = r_8$). Table 7 shows the prediction accuracy values of model of the virtual bits and the LSPUF's output $\mathbf{o}$.

In Table 7, we have reported a practical accuracy measurement t_{test} which is computed using Eq. (15) over 50,000 CRPs.

$$t_{test} = \frac{\#\{\text{Correctly classified test CRP}\}}{\#\{\text{Test CRP}\}}. \qquad (15)$$

Specifically, the test CRPs are those which are not used in the training phase of modeling. A test CRP is said to be *correctly classified* if the predicted response of test challenge is same as its actual response. It is evident from the results that all reported modeling accuracy values (p_{suc} and t_{test}) in both the case of simulated and FPGA implemented LSPUFs are significantly greater than the accuracy of random guesses $p_{rand} = 0.19\%$. Note that p_{suc} and t_{test} are prediction accuracy values of entire LSPUF output $\mathbf{o}$, not for a output bit o_i. Due to the noise in CRPs of FPGA implemented LSPUF, the modeling accuracy values of both the virtual outputs and actual outputs are poorer than those of simulated LSPUF.

One point we want to mention here is that we have used *similar number of CRPs* in the modeling of both simulated and FPGA implemented LSPUF instances to maintain the similar complexity in the model building. But, it results in the reduction of prediction accuracy of model in case of FPGA implemented LSPUF, as CRPs are noisy due to reliability issue. If we use more CRPs in the modeling of FPGA implemented LSPUF, then we can achieve better prediction accuracy than the reported one. We have also followed this strategy for the following cases.

5.2.2 Case-I of Attack Strategy-II

Recall that Case-I of the Attack Strategy-II (cf. Section 4.3.2.1) is the scenario where the adversary cannot build a model v^s_{m-1} for virtual output bit v_{m-1} because $t_{m-1} = x - m + 1 \geq \hat{t}$. To demonstrate this case, we have implemented a LSPUF variant with parameters $k = 10, m = 2, x = 9$ and $\hat{t} = 7$, and result is presented in Table 8. In this context, the adversary uses random guess algorithm as a model of v_{m-1}. In this case, attack is not a modeling attack; it is a cryptanalysis attack, as we reported in [18]. Since p_{suc} and t_{test} are equivalent, in this case, we do not have t_{test} in Table 8. It is evident from result that $p_{suc} > p_{rand} = \frac{1}{2^2} \times 100\% = 25\%$.

TABLE 7: Special case of Attack Strategy-I: Modeling accuracy of 9–XOR LSPUF ($k = 10$, $m = 9$, $x = 9$)

Platform	$n^\dagger$	$N^\ddagger$	Virtual Output Bits (%)									p_{suc} (%)	t_{test}(%)	p_{rand} (%)
			p_{v_0}	p_{v_1}	p_{v_2}	p_{v_3}	p_{v_4}	p_{v_5}	p_{v_6}	p_{v_7}	p_{v_8}			
Simulated	64	25000	99.81	99.80	99.83	99.84	99.83	99.76	99.83	99.81	99.93	98.44	98.59	0.19
		30000	99.87	99.85	99.85	99.86	99.85	99.84	99.89	99.85	99.93	98.81	98.92	
	128	25000	99.62	99.65	99.59	99.62	99.61	99.61	99.59	99.64	99.84	96.82	97.10	
		30000	99.73	99.71	99.73	99.69	99.73	99.69	99.71	99.72	99.89	97.63	97.84	
FPGA	64	25000	92.95	92.75	92.87	94.11	94.48	94.62	94.45	93.13	95.90	56.83	69.89	
		30000	93.01	92.73	92.88	94.11	94.48	94.61	94.51	93.19	95.91	56.92	70.16	
	128	25000	92.37	92.60	92.58	93.64	95.29	95.41	93.88	92.68	95.69	56.13	68.33	
		30000	92.40	92.69	92.67	93.75	95.32	95.43	93.91	92.73	95.71	56.41	68.69	

† Challenge length of LSPUF in bits.
‡ Number of CRPs used to train a model for virtual output bits $v_0, \ldots, v_8$.

TABLE 8: Case-I: Modeling accuracy of 9–XOR LSPUF ($k = 10$, $m = 2$, $x = 9$)

Platform	$n^\star$	$N^\dagger$	p_{v_0} (%)	p_{suc} (%)	p_{rand} (%)
Simulated	64	25000	99.75	49.88	25.00
		30000	99.83	49.92	
	128	25000	99.50	49.75	
		30000	99.62	49.81	
FPGA	64	25000	93.00	46.50	
		30000	92.95	46.48	
	128	25000	92.28	46.14	
		30000	92.32	46.16	

⋆ Challenge length of LSPUF in bits.
† Number of CRPs used to train a model of v_0.

5.2.3 Case-II of Attack Strategy-II

To demonstrate the Case-II of Attack Strategy-II, we have used LSPUF variants with following parameter setting: $k = 10, x = 9, m = 6, 7, 8, 9$. Note that to simplify the experimental effort, we have only implemented LSPUF instance with $k = 10, x = 9, m = 9$ on FPGA, and we have collected the APUFs outputs along with LSPUF outputs. Other LSPUF instances with $m = 6, 7$, and 8 are derived from the collected CRPs of APUFs. But, in case of simulation, all LSPUF instances are implemented separately.

The results are described in Tables 9 and 10. In this case, the adversary needs to know ϵ_3 (see Eq. (12)) for deciding whether a model $\bar{r}^s$ can be constructed or not. To compute the ϵ_3, we need to know the p_r which is the average modeling accuracy of models $r_0^s, \ldots, r_{m-2}^s$. As mentioned in Section 4.3.2, p_r cannot be computed, but p_v (modeling accuracy of $v_i^s, i = 0, \ldots, m-1$) can be used as its approximation, i.e., each $p_{r_i} \approx p_{v_i}$, and they are shown in Table 9. In practice, to compute the noise ϵ_3, the reliability p_{o_0} of output o_0 should be taken into account. The reliability p_{o_0} (for FPGA implemented LSPUF) can be found in Table 5. Without loss of generality, since the $x = 9$ for all $m = 6, 7, 8$ and 9, we assume that p_{o_0} values for all LSPUF variants with $m = 6, 7, 8$ are equal to that of LSPUF variant with $m = 9$ which is described in Table 5. Then ϵ_3 can be computed as follows:

$$\epsilon_3 = \frac{1}{2} + \frac{p_{r_0} \times \cdots \times p_{r_{m-2}} \times p_{o_0}}{2}. \tag{16}$$

Note that the prediction accuracy p^+ of v_{m-1}^s is computed based on actual output o_0 and the model $v_{m-1}^s = r_0^s \oplus \cdots \oplus r_{m-2}^s \oplus \bar{r}^s$.

In Table 10, we have described the modeling accuracy values of $v_0^s, \ldots, v_{m-1}^s$ in term of t_{test} (cf. Eq. (15)) and p_{suc} (cf. Eq. (14)). The p_{rand} values are also provided for understanding the efficiency of the attack when it is compared with random guess algorithm. The model v_{m-1}^s is constructed as described in Section 4.3.2, and it is evident that if m is small, then the number of CRPs required (N_2) to build the model v_{m-1}^s is large. The reason is that building the model $\bar{r}^s$ becomes more difficult because the model $\bar{r}^s$ consists of many APUFs. However, it is evident that the p_{suc} and t_{test} are significantly greater than p_{rand}. This implies that multibit output LSPUF variants are not secure against our proposed modeling attack.

Note that in Tables 9 and 10, the modeling accuracy values for $p_{v_0}, \ldots, p_{v_{m-2}}$ are similar for all LSPUF variants with $m = 6, 7, 8, 9$, which have the same n. The reason is that we have the same set of virtual bit models to simplify the design and experiment efforts. Note that prediction accuracy in case of FPGA implementation is comparatively poor than simulation due to large amount of noise in derived CRPs used for building a model of v_{m-1}, but noise can be manage by increasing the amount of training CRPs at the cost of computational time.

5.2.4 Case-III of Attack Strategy-II

We focused on the LSPUF variant with $k = 10, m = 9, x = 8$. In this case, $m > x$ and then the $v_{m-1}^s = r_0^s \oplus \cdots \oplus r_{x-1}^s$. Specifically, all APUF models are in $\mathcal{D}$ and the adversary does not need to build a model for v_{m-1}, unlike case-II. The results are reported in Table 11. It is evident from Table 11 that p_{suc} and t_{test} values are significantly greater than p_{rand}.

6 Discussion

Now we provide a relative comparison between our proposed modeling attack on multibit output LSPUF variants with the previously proposed modeling attacks in [6], [12]–[18]. Compared to the pure MLMA in [6], [12], [13], our proposed attack requires a significantly less number of CRPs and time complexity to achieve the same level of prediction accuracy p_{suc}, as we have used the 2-XOR PUF modeling for $v_0, \ldots, v_{m-2}$ and $(x - m + 1)$-XOR PUF modeling for v_{m-1} instead of x-XOR PUF modeling. In [12], authors used the parallel implementation of LR algorithm to show that they can scale the modeling attack in [6] upto 9-XOR LSPUF. But, in our work we have used the traditional implementation of LR as used in [6]. Thus, if we employ the parallel implementation of LR in our modeling attack,

TABLE 9: Case-II: Modeling accuracy values of component APUFs of 9–XOR LSPUF ($k = 10$, $x = 9$) derived from the models of virtual outputs, reliability (p_{o_0}) of LSPUF's output bit o_0, ϵ_3 as the correctness of CRPs used for modeling v_{m-1}, and prediction accuracy p^+ of model v^s_{m-1}

Platform	$n^\dagger$	m	$N_1{}^\dagger$	Component APUFs (%)								$p_{o_0}(\%)$	$\epsilon_3(\%)$	$p^+(\%)$
				p_{r_0}	p_{r_1}	p_{r_2}	p_{r_3}	p_{r_4}	p_{r_5}	p_{r_6}	p_{r_7}			
Simulated	64	9	25000	99.77	99.75	99.75	99.78	99.78	99.76	99.79	99.80	100.00	98.18	98.80
		8	25000	99.77	99.75	99.75	99.78	99.78	99.76	99.79		100.00	98.38	98.43
		7	25000	99.77	99.75	99.75	99.78	99.78	99.76			100.00	98.59	98.18
		6	25000	99.77	99.75	99.75	99.78	99.78				100.00	98.83	94.11
	128	9	25000	99.50	99.53	99.45	99.50	99.48	99.48	99.46	99.52	100.00	96.00	97.50
		8	25000	99.50	99.53	99.45	99.50	99.48	99.48	99.46		100.00	96.46	97.01
		7	25000	99.50	99.53	99.45	99.50	99.48	99.48			100.00	96.46	96.01
		6	25000	99.50	99.53	99.45	99.50	99.48				100.00	97.50	94.19
FPGA	64	9	25000	93.00	92.66	92.86	94.07	94.36	94.54	94.40	93.13	97.08	78.66	76.90
		8	25000	93.00	92.66	92.86	94.07	94.36	94.54	94.40		97.08	80.74	75.10
		7	25000	93.00	92.66	92.86	94.07	94.36	94.54			97.08	82.57	73.39
		6	25000	93.00	92.66	92.86	94.07	94.36				97.08	84.43	73.92
	128	9	25000	92.28	92.55	92.55	93.61	95.21	95.30	93.75	92.62	96.87	78.21	76.81
		8	25000	92.28	92.55	92.55	93.61	95.21	95.30	93.75		96.87	80.48	74.40
		7	25000	92.28	92.55	92.55	93.61	95.21	95.30			96.87	82.51	71.30
		6	25000	92.28	92.55	92.55	93.61	95.21				96.87	84.11	70.31

* Challenge length of LSPUF in bits.
† Number of CRPs used to train a model for virtual output bits $v_0, \ldots, v_{m-2}$.

TABLE 10: Case-II: Modeling accuracy of 9–XOR LSPUF ($k = 10$, $x = 9$)

Platform	$n^\dagger$	m	$N_1{}^\dagger$	Virtual Output Bits (%)									$N_2{}^\ddagger$	p_{suc} (%)	t_{test} (%)	p_{rand} (%)
				p_{v_0}	p_{v_1}	p_{v_2}	p_{v_3}	p_{v_4}	p_{v_5}	p_{v_6}	p_{v_7}	p_{v_8}				
Simulated	64	9	25000	99.77	99.75	99.75	99.78	99.78	99.76	99.79	99.80	98.80	20000	96.91	98.24	0.19
		8	25000	99.77	99.75	99.75	99.78	99.78	99.76	99.79	98.43		30000	96.78	97.94	0.39
		7	25000	99.77	99.75	99.75	99.78	99.78	99.76	98.18			40000	96.83	97.77	0.78
		6	25000	99.77	99.75	99.75	99.78	99.78	94.11				60000	92.98	93.76	1.56
	128	9	25000	99.50	99.53	99.45	99.50	99.48	99.48	99.46	99.52	97.50	30000	93.51	96.38	0.19
		8	25000	99.50	99.53	99.45	99.50	99.48	99.48	99.46	97.01		45000	92.97	96.00	0.39
		7	25000	99.50	99.53	99.45	99.50	99.48	99.48	96.01			80000	92.99	95.14	0.78
		6	25000	99.50	99.53	99.45	99.50	99.48	94.19				120000	91.72	93.45	1.56
FPGA	64	9	25000	93.00	92.66	92.86	94.07	94.36	94.54	94.40	93.13	76.90	40000	44.34	68.16	0.19
		8	25000	93.00	92.66	92.86	94.07	94.36	94.54	94.40	75.10		50000	46.39	66.50	0.39
		7	25000	93.00	92.66	92.86	94.07	94.36	94.54	73.39			60000	48.56	65.19	0.78
		6	25000	93.00	92.66	92.86	94.07	94.36	73.92				80000	52.40	66.41	1.56
	128	9	25000	92.28	92.55	92.55	93.61	95.21	95.30	93.75	92.62	76.81	50000	44.10	66.40	0.19
		8	25000	92.28	92.55	92.55	93.61	95.21	95.30	93.75	74.40		60000	45.78	64.38	0.39
		7	25000	92.28	92.55	92.55	93.61	95.21	95.30	71.30			90000	47.03	61.80	0.78
		6	25000	92.28	92.55	92.55	93.61	95.21	70.31				120000	49.70	61.37	1.56

* Challenge length of LSPUF in bits.
† Number of CRPs used to train a model for virtual output bits $v_0, \ldots, v_{m-2}$.
‡ Number of CRPs used to train a model for virtual output bit v_{m-1}.
Note: t_{test} is computed using 50000 CRPs.

TABLE 11: Case-III: Modeling accuracy of 8–XOR LSPUF ($k = 10$, $m = 9$, $x = 8$)

Platform	$n^\dagger$	$N^\dagger$	Virtual Output Bits (%)									p_{suc} (%)	t_{test}(%)	p_{rand} (%)
			p_{v_0}	p_{v_1}	p_{v_2}	p_{v_3}	p_{v_4}	p_{v_5}	p_{v_6}	p_{v_7}	p_{v_8}			
Simulated	64	25000	99.73	99.77	99.80	99.75	99.19	99.79	99.78	99.77	99.31	97.33	98.46	0.19
		30000	99.74	99.81	99.83	99.79	99.40	99.84	99.78	99.78	99.43	98.12	99.08	
	128	25000	99.53	99.44	99.42	99.50	99.49	99.45	99.31	99.44	98.41	95.15	97.11	
		30000	99.65	99.57	99.46	99.62	99.51	99.54	99.32	99.56	98.62	96.30	97.23	
FPGA	64	25000	93.16	93.33	93.17	94.31	93.47	95.86	92.56	92.62	82.12	47.76	70.96	
		30000	93.15	93.33	93.19	94.35	93.48	95.88	92.57	92.57	82.45	48.15	71.21	
	128	25000	92.75	92.80	92.85	92.04	92.57	93.57	94.30	94.41	81.98	47.07	70.67	
		30000	92.80	92.90	92.87	92.07	92.59	95.71	94.35	94.46	82.10	47.42	70.89	

† Challenge length of LSPUF in bits.
† Number of CRPs used to train a model for virtual output bits $v_0, \ldots, v_7$.
Note: t_{test} is computed by using 50000 CRPs.

then our proposed attack will also be more scalable than its present degree of scalability.

Compared to [14]–[17], we do not require the physical access and physical tampering of PUF device. Moreover, we also do not need multiple evaluations of PUF for each challenge to obtain reliability information of PUF instance as used to develop attack in [15]. Typically, all these additional information are not accessible to the adversary in practical application of PUF like PUF-based protocols where the adversary can only eavesdrop the CRPs. Thus, our proposed attack is more feasible in practical scenarios.

In comparison with our previous work in [18], which is a cryptanalysis attack, our present work deals with the modeling attack on the multibit output LSPUF. The modeling attack results in the significantly higher prediction accuracy than that of cryptanalysis attack [18]. We have also reported that there are few LSPUF variants where modeling attack is not computationally feasible, but we can perform cryptanalysis attack. So, those LSPUF variants are comparatively less vulnerable than LSPUF variants that can be modeled.

Since our attack is developed by exploiting the property of output network of LSPUF with multibit outputs, this attack cannot be applied to the other PUF designs without this kind of network. However, from this work, it is evident that PUF designer should not use this type of output network to generate multibit outputs in any future PUF design. Besides the advantages and disadvantages of various existing attacks, development of all these attack methodologies is establishing a good platform for research on secure PUF designs.

7 Conclusion

In this paper, we have presented a comprehensive security analysis of multibit output LSPUF variants in the light of the combined cryptanalysis and machine learning based modeling attack. The effectiveness of our attack is validated using both simulated and FPGA implemented LSPUF instances. We successfully built the models for 9-XOR 64-bit and 128-bit LSPUFs, and the pure computational modeling of 9-XOR 128-bit LSPUFs is performed for first time in this paper. Based on our analysis, the multibit output LSPUF variants can be divided into two following classes: i) only cryptanalyzed class and ii) fully modeled class. The LSPUF variants belonging to cryptanalyzed class are still robust against modeling attacks; however, design parameter selection of these instances might be infeasible due to practical issues with reliability of the LSPUF implementations. Note that our proposed modeling attack can be applied only for multibit output LSPUF variants. However, the significance of this work is that we have revealed the weakness of LSPUF's output network although it is significantly lightweight and can efficiently generate multibit outputs. From this work, it is evident that lightweightness should not be the main design objective, as it might compromise the security of design. As we have seen in the literature that designing secure strong PUFs is a challenging task, we would not say that we break some PUF designs, rather this work should be treated as another way to analyze the security of a PUF design. As a consequence of this study, we might have good PUF design in future.

References

[1] R. S. Pappu, "Physical one-way functions," Ph.D. dissertation, Massachusetts Institute of Technology, March 2001.

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

[3] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *Proc. of IEEE HOST*, June 2008, pp. 67–70.

[4] M.-D. M. Yu, D. M'Raïhi, R. Sowell, and S. Devadas, "Lightweight and Secure PUF Key Storage Using Limits of Machine Learning," in *Proc. of 13th CHES* , vol. 6917. Springer Berlin / Heidelberg, 2011, pp. 358–373.

[5] C. Brzuska, M. Fischlin, H. Schrder, and S. Katzenbeisser, "Physically Uncloneable Functions in the Universal Composition Framework," in *Advances in Cryptology (CRYPTO)*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed. Springer Berlin / Heidelberg, 2011, vol. 6841, pp. 51–70.

[6] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. of 17th ACM CCS*. New York, NY, USA: ACM, 2010, pp. 237–249.

[7] 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.

[8] J. Sölter, "Cryptanalysis of Electrical PUFs via Machine Learning Algorithms," Master's thesis, Technische Universität München, 2009.

[9] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. of the 2008 IEEE/ACM ICCAD*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 670–673.

[10] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. of DAC*. New York, NY, USA: ACM Press, 2007, pp. 9–14.

[11] R. van den Berg, B. Skoric, and V. van der Leest, "Bias-based modeling and entropy analysis of PUFs," in *Proc. of the ACM TrustED*, 2013, pp. 13–20.

[12] J. Tobisch and G. T. Becker, "On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation," in *Proc. of RFIDsec*, 2015, pp. 17–31.

[13] F. Ganji, S. Tajik, and J. Seifert, "Why Attackers Win: On the Learnability of XOR Arbiter PUFs," in *Proc. of TRUST*, 2015, pp. 22–39.

[14] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. P. Burleson, "Efficient Power and Timing Side Channels for Physical Unclonable Functions," in *Proc. of 16th CHES*, 2014, pp. 476–492.

[15] G. T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs," in *Proc. of 17th CHES*, 2015.

[16] F. Ganji, J. Krämer, J. Seifert, and S. Tajik, "Lattice Basis Reduction Attack against Physically Unclonable Functions," in *Proc. of the 22nd ACM SIGSAC CCS*, 2015, pp. 1070–1080.

[17] S. Tajik, H. Lohrke, F. Ganji, J. P. Seifert, and C. Boit, "Laser Fault Attack on Physically Unclonable Functions," in *12th FTDC*, 2015.

[18] D. P. Sahoo, P. H. Nguyen, D. Mukhopadhyay, and R. S. Chakraborty, "A Case of Lightweight PUF Constructions: Cryptanalysis and Machine Learning Attacks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 43, no. 8, pp. 1334–1343, 2015.

[19] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for Design and Implementation of Secure Reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–33, 2009.

Phuong Ha Nguyen is a Postdoctoral Fellow in the Computer Science and Engineering Department of Indian Institute of Technology Kharagpur. He received a Ph.D. in Computer Engineering from Nanyang Technological University (Singapore) in 2013 and a Specialist Degree (Red Diplom) from Moscow State University (Russia) in 2008. He has work experience at Temasek Lab@NTU.

His research interests include: Cryptanalysis, Side Channel Analysis and Physical Unclonable Functions. He has 15 publications in international journals and conferences. He holds one U.S. patent based on his research work.

Durga Prasad Sahoo has been pursuing Ph.D. in Dept. of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India, since 2012. He received B.Sc., M.Sc., and M.Tech. in Computer Science from University of Calcutta in 2007, 2009, and 2011, respectively.

His research interests include Physically Unclonable Functions (PUFs), and Secure Embedded System Design.