Securing PUFs against ML Modeling Attacks via an Efficient Challenge-Response Approach

Mieszko Ferens
Dept. of Electronic Systems
Aalborg University
Copenhagen, Denmark
mjfm@es.aau.dk

Edlira Dushku
Dept. of Electronic Systems
Aalborg University
Copenhagen, Denmark
edu@es.aau.dk

Sokol Kosta
Dept. of Electronic Systems
Aalborg University
Copenhagen, Denmark
sok@es.aau.dk

***Abstract*—Physical Unclonable Functions (PUFs) are lightweight security primitives capable of providing functionalities such as device authentication and identification. Such lightweight solutions are particularly important for small resource-constrained devices that cannot support many of the standard security mechanisms like e.g., TPMs. Even though PUFs are constructed to be unpredictable and unclonable, they have been susceptible to Machine Learning (ML) modeling attacks. Mitigation of these attacks typically requires additional hardware, causing potential deviation from the lightweight nature of low-end embedded devices. In this paper, we analyze the technical details that lead to the success of the previous ML modeling attacks, and utilize these findings to devise a novel challenge-response approach that improves PUF's security, more specifically the 4-XOR and 5-XOR PUFs, without additional hardware requirements. Our experimental results show that the proposed approach reduces modeling accuracies of state-of-the-art ML attacks by 10-15%, lowering the success rate of attacks significantly while remaining practical in the implementation.**

***Index Terms*—Arbiter PUF, XOR PUF, modeling attacks, CRP, hardware security, IoT**

I. Introduction

Resource-constrained devices are integral to critical infrastructures such as healthcare, transportation, or industrial systems, and protecting their communications is of utmost importance for preventing major consequences of cyberattacks. In the majority of PCs, laptops, and servers, the basic security-related functions can be provided by a trusted computing base, e.g., a standardized hardware module called Trusted Platform Module (TPM) [1]. However, in low-end embedded devices with limited resources, the presence of such a hardware component is often infeasible [2].

Aimed at providing lightweight security features, Physical Unclonable Functions (PUFs) have been proposed as a promising solution with low hardware overhead [3]. A PUF is a physical circuit that takes an input called *challenge* and gives an output called *response*. By exploiting the unavoidable manufacturing variations of Integrated Circuits (ICs), different PUFs can give different outputs to the same input. The first design proposed in 2002 was the Arbiter PUF (APUF) [2], which made use of the variations in wire delays and could be employed for applications such as lightweight device authentication [3]. Since then, many other PUF designs have been proposed, like the Feed-Forward PUF [3], the XOR PUF [4], and the Interpose PUF [5], among many others (not necessarily based on wire delays). All these designs are part of a PUF class called *Strong PUFs* that provide a large number of unique challenges, as opposed to *Weak PUFs*, which are more limited in amount [6].

In a standard paradigm, PUFs communicate with a server that possesses a dataset of Challenge-Response Pairs (CRPs), collected only once after manufacturing and used to authenticate the PUFs. Since CRPs are never reused, and physical invasive attacks are impossible due to physical operational characteristics being changed in such event, impersonating a PUF is supposedly not possible.

Nevertheless, many papers have shown that PUFs are vulnerable to Machine Learning (ML) modeling attacks [7]–[13]. By collecting previously used CRPs it is possible to create a PUF clone which can then impersonate the original. To mitigate this, additional security hardware is required, which contradicts the minimal lightweight design of resource-constraint devices.

Contributions. In this paper, we investigate the technical details that lead to the success of state-of-the-art ML modeling attacks against PUFs. To the best of our knowledge, all previous works on PUFs assume that, after manufacturing, CRPs are **randomly selected** without repetition. We question this critical assumption and investigate whether **selectively** choosing challenges can lead to more resilient systems. Our findings show that the successful modeling attacks seen in literature on current PUFs are not only due to how PUFs are designed, but also to how the CRPs are selected. Even though this paper focuses on state-of-the-art XOR PUFs, the proposed method is agnostic from the underlying implementation details of any APUF-based design. To this end, our main contributions can be summarized as follows:

- We propose a novel approach that enhances and improves PUFs' security by relying on selective CRPs. To achieve this, we present three methods that aim to prevent ML modeling attacks without requiring additional hardware overhead.

This work is supported by the IoTalentum project, funded by the European Union Horizon 2020 research and innovation program within the framework of Marie Skłodowska-Curie Actions ITN-ETN with grant number 953442.

- Our experimental results demonstrate that the proposed technique prevents modeling attacks from being successful. In particular, we show that the state-of-the-art ML modeling attacks are less effective when selective CRPs are used.

Outline. The rest of this paper is organized as follows. In Section II, we provide an analytical explanation of how an Arbiter PUF and its derivatives (specifically the XOR PUF) work, and how they can be modeled by ML. Section III shows an overview of the related work for attack models and defense methods that are applied to APUFs and derivatives. In Section IV, we describe our tested methods to prevent modeling attacks. Sections V and VI show the experimental setup and empirical data to support our claims. In Section VII, we discuss the applicability of our methods. Finally, Section VIII concludes the paper.

II. Background

A. Arbiter-based PUFs

By far the most popular PUF design in recent years has been the Arbiter PUF (APUF), mostly used as a basic building block on which many other PUF implementations are based on. An APUF, as shown in Fig. 1, is a circuit comprised of a sequence of n stages (n-bits) with an arbiter block at the end.

Each stage controls whether the inputs are passed to the outputs straight or crossed, creating a pair of paths. The random manufacturing variations of the IC randomize the delays of the wires approximating a Gaussian distribution [3]. By sending a signal and comparing the two unique path delays with the arbiter block, either 0 or 1 is given as the output[1]:

$$r = sign(\Delta) = sign(\vec{w}^T \vec{\Phi}) \tag{1}$$

This is referred to as the additive linear delay model, originally defined in [3] and later adopted for further research [6]. The vector $\vec{w}$ defines the delay parameters of each stage accounting for all previous stages. Thus for n stages it has $n+1$ elements.

$$\vec{w} = (w^1, ..., w^{n+1}) \tag{2}$$

where $w^1 = \frac{\delta_1^0 - \delta_1^1}{2}$, $w^{n+1} = \frac{\delta_n^0 + \delta_n^1}{2}$, and

$$w^i = \frac{\delta_{i-1}^0 + \delta_{i-1}^1 + \delta_i^0 - \delta_i^1}{2}; \forall i \in [2, n] \tag{3}$$

with $\delta_n^{0,1}$ denoting the sum of delays of both the straight (0) and crossed (1) paths up to and including stage n.

The vector $\vec{\Phi}$ is a function of the challenge $\vec{C}$ that is applied to the APUF.

$$\vec{\Phi}(\vec{C}) = (\Phi^1(\vec{C}), ..., \Phi^n(\vec{C}), 1) \tag{4}$$

where $\Phi^i(\vec{C}) = \prod_{i=1}^{n}(1 - 2b_i)$, with b_i being the i-th selector bit of challenge $\vec{C}$.

[1]Note that the output is remapped from $(0, 1)$ to $(1, -1)$ for convenience

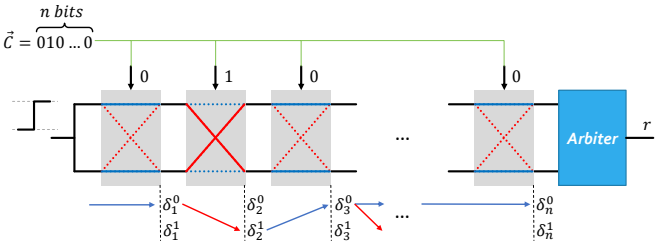


Fig. 1. Schematic view of an Arbiter PUF.

The additive linear delay model can be extended to PUF designs that include multiple independent APUFs, like the k-XOR PUF:

$$r' = sign(\prod_{l=1}^{k} \Delta_l) = sign(\prod_{l=1}^{k} \vec{w}_l^T \vec{\Phi}_l) \tag{5}$$

which involves k vectors $\vec{w}_1, ..., \vec{w}_k$ and $\vec{\Phi}_1, ..., \vec{\Phi}_k$.

B. Modeling Attacks

The models shown in (1) and (5) posses a separating hyperplane at $\vec{w}^T\vec{\Phi} = 0$ and $\prod_{l=1}^{k} \vec{w}_l^T \vec{\Phi}_l = 0$, respectively. The side of the hyperplane on which a specific challenge is located (through $\vec{\Phi}(\vec{C})$) determines the response. By feeding ML algorithms with CRPs they can obtain a function that approximates a decision boundary $f = \vec{w}^T\vec{\Phi}'$. If $\vec{\Phi}'$ is close enough to the real PUF's $\vec{\Phi}$, then the algorithm can successfully replicate it. Next we describe the algorithms used in this paper, motivated by the state-of-the-art (see Section III).

a) Logistic Regression: Logistic Regression (LR), in its application to PUF modeling, sees each challenge $\vec{C}$ having an assigned probability $p(\vec{C}, r \mid \vec{w})$ to generate an output r of -1 or 1, defined as

$$p(\vec{C}, r \mid \vec{w}) = \sigma(rf) = (1 + e^{-rf})^{-1} \tag{6}$$

where $f(\vec{w}) = 0$ determines a decision boundary of equal output probabilities. We refer the reader to [13] for the exact details of the implementation used in this paper.

b) Multi-Layer Perceptron: This algorithm is based on the use of a set of layers of neurons. The first layer is called the input layer, where the outputs of the neurons simply take the values of the inputs. Following the input layer, the hidden layers process the information by applying a certain operation to the input values. This is performed by each neuron individually by having its output for a single input be determined by a weight factor w, a bias b, and an activation function $g(x)$. Since in an MLP the neural network is densely connected, i.e., the output of a neuron in one layer is passed to all neurons in the next, the n-th neuron's output is determined by the sum of all its inputs X^n:

$$o^n = \sum_{i=1}^{X^n} g(w_i^n x_i^n + b^n) \tag{7}$$

The final layer in a PUF modeling problem is a single neuron with the same processing shown in (7), but possibly with a

TABLE I
ML TECHNIQUES APPLICABLE TO CLONING XOR PUFS

Paper	Year	Perspective	Method	Overhead	Result
Ruhmair et al. [7], [8]	2010	Attack	LR and ES	-	64-bit 6-XOR PUF broken
Paral et al. [14]	2011	Defense	PMKG	Low	Challenges partially unavailable to attackers
Majzoobi et al. [15]	2012	Defense	Slender PUF protocol	Medium	Challenges fully unavailable to attackers
Becker et al. [9]	2015	Attack	Reliability CMA-ES	-	128-bit 16-XOR PUF broken
Yu et al. [16]	2016	Defense	Lockdown protocol	Medium	Attackers unable to directly query the PUF
Alkatheiri et al. [10]	2017	Attack	MLP	-	Feed-Forward PUF broken
Aseeri et al. [11]	2018	Attack	MLP	-	64-bit 8-XOR PUF broken
Nguyen et al. [5]	2019	Defense	Interpose PUF	Low	Prevents reliability CMA-ES attack
Mursi et al. [12]	2020	Attack	Improved MLP	-	64-bit 9-XOR PUF broken
Wisiol et al. [13]	2022	Attack	ECP-TRN, Improved LR and MLP	-	Optimized LR attack
Wu et al. [17]	2022	Defense	FLAM-PUF	Very low	Obfuscates challenges using a Galois LFSR
This work	**2023**	**Defense**	**Selective CRPs**	**None**	**Prevents the success of LR and MLP attacks**

different activation function. We refer the reader to [12] for the exact details of the implementation used in this paper.

III. RELATED WORK

In this section, we summarize the related work on i) ML modeling attacks that deal with XOR PUFs, and ii) defense methods that aim to improve PUFs' security.

The first extensive analysis for the security of APUFs and derivatives was performed by Ruhmair et al. [7]. It showed that APUFs and many of their derivatives are clonable with ML algorithms, including Logistic Regression (LR) and Evolutionary Strategies (ES). Later, the work in [8] extended this research to real PUFs and not just simulated ones. In [9], Becker uncovered that the reliability of PUFs can be exploited to more efficiently train a ML algorithm like the Covariance Matrix Adaptation (CMA) ES. Additionally, in recent years, the use of Artificial Neural Networks (ANN) with the Multi-Layer Perceptron (MLP) algorithm has proven to be a useful tool to model more complex PUFs than ever before (Alkatheiri et al. [10] and Aseeri et al. [11]). Following this, Mursi et al. [12] improved the configuration of the MLP. Finally, Wisiol et al. [13] used the improved MLP and compared its performance with a new improved LR attack. The last two attacks mark the current best performing ML algorithms for modeling XOR PUFs, each better than the other depending on the size of the PUF.

In parallel, to improve the security of PUFs, Paral et al. [14] devised a Pattern Matching Key Generation (PMKG) scheme, where challenges are not sent in the clear when server-PUF communication is taking place via some additional control logic. Similarly, Majzoobi et al. [15] proposed the Slender PUF protocol, which uses a PRNG and TRNG to have both server and PUF agree on challenges without the need to send them in the clear. Finally, Yu et al. [16] proposed the Lockdown protocol designed to keep a PUF from answering queries from non-authentic servers by means of a PRNG. While not dealing with improved designs in this paper, we choose to mention the works by Nguyen et al. [5], since the Interpose PUF prevents the reliability CMA-ES attack that we do not test in this paper, and Wu et al. [17], due to the FLAM-PUF focusing on lightweight implementation with an APUF at its core. The key differentiating factor of our work from the latter ones is that our technique does not incur in any additional overhead. Moreover, our proposal is compatible with the others, as well as applicable to any APUF-based design (like the Interpose PUF).

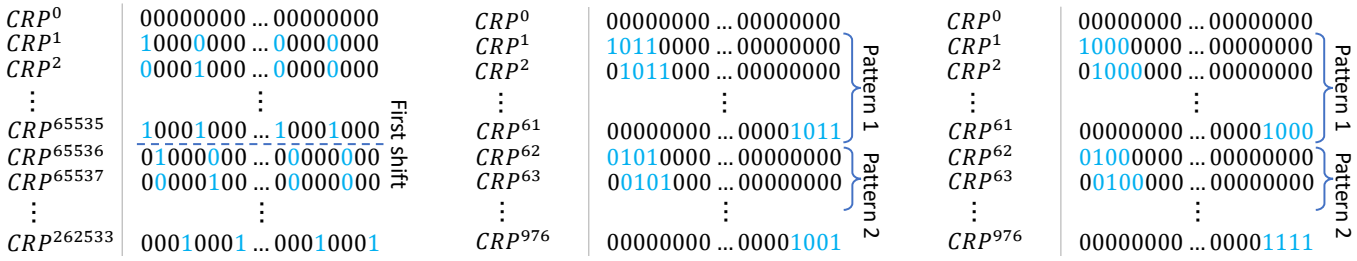
Table I gives an overview of the aforementioned works used to model XOR PUFs and the defense methods in chronological order.

IV. OUR PROPOSAL: SELECTIVE CRPS

Our proposal for improving PUFs' security against ML modeling attacks is based on induced data scarcity. To achieve this, we carefully select and craft a CRP set where challenges do not use all their bits simultaneously, i.e., many bits are set to a default value of either 0 or 1, and only begin to be used after a large number of CRPs. In this way, many of the delay parameters of the underlying PUF model, as described in (3), are hidden and impossible to learn unless a very numerous subset of CRPs is collected. By gradually introducing the unused bits in the CRP set, an attacker will have to keep collecting data for their model to learn the new bits that are suddenly used for the first time. Based on this, we propose three methods of creating a selective CRP dataset. Fig. 2 shows these methods with all actively used bits in each challenge marked.

A. Binary-coded with Padding CRPs (BP)

The first method we propose, called BP, is based on using a binary table for generating challenges, where the challenges are taken from the table in order. This introduces new values for previously unused bits at a rate of 2^c challenges, where c is the most significant bit in the challenge that has been flipped once. Since this becomes too slow for large values of c, we only use 16 out of the 64 bits of the challenge, filling between them with 0s (as shown in Fig. 2(a)). After that, a circular shift is applied three times to generate more CRPs and use all bits. The main issue is the low number of generated unique CRPs available to the server to operate the PUF for its whole lifespan.



(a) BP (b) RSP ($p = 4$) (c) BSP ($p = 4$)

Fig. 2. Challenge sets without elimination of repeats.

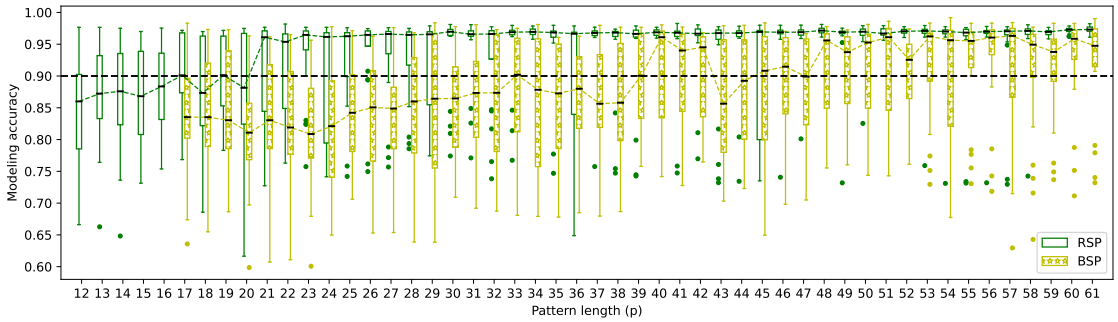


Fig. 3. Effect of pattern length (p) on modeling accuracies of improved LR attacks on 4-XOR PUFs with 80000 CRPs of RSP and BSP. For RSP a value of $p = 20$ is the highest that significantly improves security, while for BSP a higher value is possible.

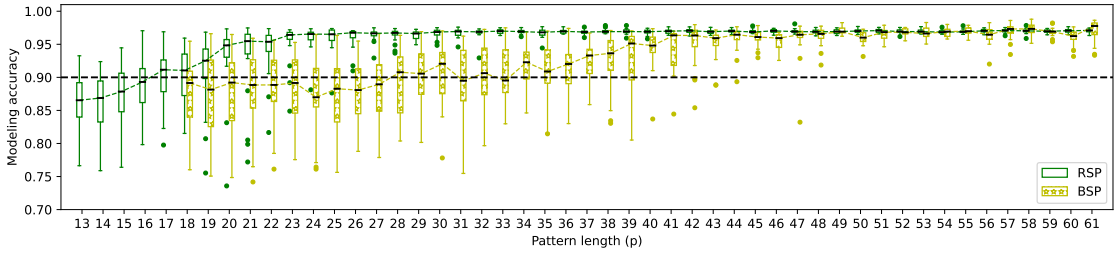


Fig. 4. Effect of pattern length (p) on modeling accuracies of improved MLP attacks on 5-XOR PUFs with 160000 CRPs of RSP and BSP. For RSP a value of $p = 16$ is the highest that significantly improves security, while for BSP a higher value is possible.

B. Random Shifted Pattern CRPs (RSP)

For our second method, called RSP, we propose a selective CRP dataset based on a random sequence of bits similar to the traditional challenges. The idea is to generate random patterns of p-bits that are then shifted through all bits of the full challenge, leaving all unused bits at a value of 0 (as shown in Fig. 2(b)). As a final step, all repeated challenges must be removed. The main advantage of this method is that as long as p is large enough, we can generate an exponentially large number of unique CRPs for the server to use. However, the use of randomly selected bits can lead to lower data scarcity.

C. Binary-coded Shifted Pattern CRPs (BSP)

Our final proposed method, called BSP, combines the previous two methods to address their shortcomings. This method is very similar to RSP, with the difference that it uses a binary table instead of generating random patterns. Fig. 2(c) depicts an example of the generated CRP dataset. Once again, the final step is to remove all repeated challenges. Similarly to RSP, if p is large enough, we can guarantee that the server will be provided with enough unique CRPs, but since we are not using random patterns we stipulate that data scarcity will be increased.

V. Experimental Setup

To evaluate the effectiveness of our approach against ML modeling attacks without introducing additional hardware assumptions, we conduct experiments on 20 simulated instances of 64-bit 4-XOR and 5-XOR PUFs with reliable responses (no noise). Our choice of PUFs is motivated by the observation that the improved LR attack is the most efficient state-of-the-art attack (i.e., least number of CRPs required) for k-XOR PUFs when $k \leq 4$ [13]. However, when $k \geq 5$, the improved MLP performs better. Since our primary goal is to prove that PUFs can remain lightweight and secure against ML modeling

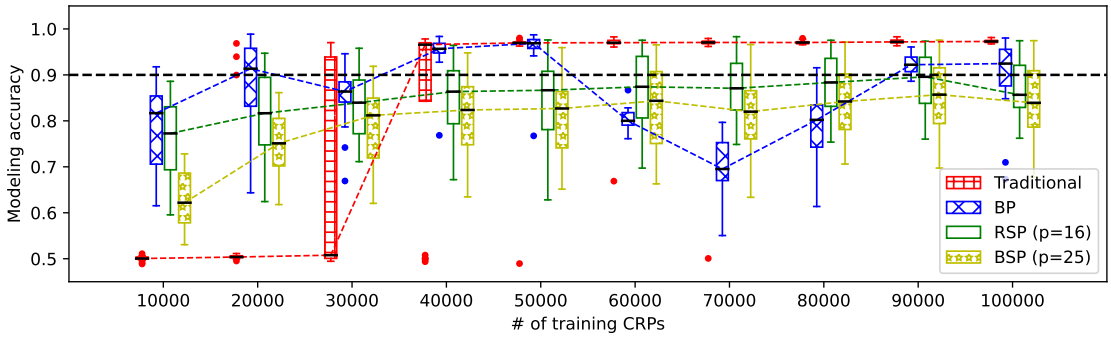


Fig. 5. Modeling accuracy of the improved LR attack on 4-XOR PUFs using traditional and selective CRPs. It can be seen as BP creates a drop in the modeling accuracy when the first of the three circular shifts occurs, while RSP and BSP provide lower modeling accuracies consistently.

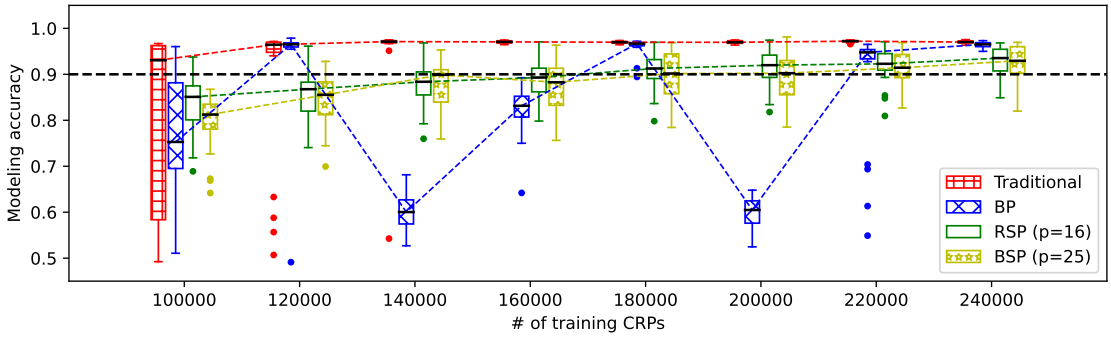


Fig. 6. Modeling accuracy of the improved MLP attack on 5-XOR PUFs using traditional and selective CRPs. It can be seen as BP creates drops in the modeling accuracy when the second and third of the three circular shifts occur, while RSP and BSP provide lower modeling accuracies consistently.

attacks, increasing k is not in our interest, although it is a way to add security if the hardware overhead can be tolerated.

For our experimental setup, we use Python 3.8 on a laptop with single thread processing. To simulate the XOR PUF instances, we use the pypuf library [18] with random seeds from 0 to 19. The attacks studied in this paper are integrated in the pypuf library, implemented with Keras over Tensorflow 2.4. We review, tested, and validated the code for these attacks before performing our experiments. For the generation of CRPs, since pypuf does not guarantee that random generated challenges are not repeated, we add this feature on our own. To test the modeling accuracies, the CRP dataset in each experiment is extended with 10000 CRPs not used for training. For replicability, the code used to generate our results is made available[2].

VI. Results

We first need to pick a value of the pattern length p for both our RSP and BSP methods. Fig. 3 and Fig. 4 show how different values affect the modeling accuracies of the improved LR and MLP attacks when enough CRPs are available for these attacks to yield accurate models if traditional CRPs are used. When traditional CRPs are used, our results differ from the ones reported in [13] as can be seen in Fig. 5 and Fig. 6. We find that the CRPs required for a 100% success rate of the improved LR attack on a 4-XOR PUF is 80000, as opposed to 30000. Similarly, the improved MLP attack requires 160000 CRPs, not 200000, to consistently model a 5-XOR PUF. Note that this does not change the fact that the LR attack is better than the MLP attack for $k \leq 4$, and viceversa. As depicted in Fig. 3 and Fig. 4, lower values for p are better for modeling complexity, but they come at the cost of total number of unique CRPs that can be generated. Therefore, for RSP we pick the highest possible value ($p = 16$) that retains a median modeling accuracy for attacks under 90% in both scenarios. For BSP, however, we can choose a higher value of $p = 25$. Since it provides more that enough unique CRPs (see Section VII) going higher is counterproductive.

Fig. 5 and Fig. 6 illustrate the results of using our proposed methods compared to the traditional CRPs considered in previous works. Regarding our proposed methods, we observe a significant change in the modeling accuracies across all our experiments. Starting with BP, it is clear that since 16 out of the 64 bits of the challenge are used at a time, the CRPs required to model the PUF are lower than normal. However, when the shifts occur, the modeling accuracy drops severely ($30 - 40\%$ lower), making attacks unsuccessful until more training data is added. For RSP, a pattern length of $p = 16$ consistently increases the training data that is required to successfully attack the PUFs. However, $p = 16$ only generates

[2]https://github.com/AAU-Edge-Computing-and-Networking/PUF-Selective-CRPs

$2^p(64-(p+1)) < 3211264$ CRPs. Combining the ideas from the previous two methods, BSP gives us the best results. It makes attacks require considerably more training data, but without limiting the available unique CRPs as much.

VII. Discussion of viability

In this section, we argue why our final method (BSP) is realistic and practical when real-world PUFs are involved. As discussed in [2], [3], [19], given a system of up to millions of devices, the number of CRPs required to perform a single authentication query does not exceed 400. Assuming that each device communicates with the server every $5-6$ minutes, it burns through around 100000 CRPs per day. This means that if we can generate at least a billion unique CRPs, the system could operate for over 30 years. Taking our proposed BSP, using $p = 25$ is enough as we can generate up to $2^p(64-(p+1)) \approx 1.3$ billion CRPs.

On a separate note, most papers consider the attacker to be able to either directly query the PUF for CRPs or listen to the PUF-server CRP exchange. It is also considered that an attacker can obtain the training data very quickly (e.g., 350000 CRPs per minute [9]) given no control mechanism to prevent it. However, if the attacker is limited to the CRPs used by the server, then it would take around a day to get 100000 CRPs as per our previous discussion. We believe the former method of obtaining CRPs can be unviable for a system like PUF-based RFID tags [9] that, if stolen, get reported and purged from the system. Only if the RFID tags can be queried physically by a reader setup by the attacker without raising suspicion, the long data collection time could be avoided. Seemingly the passive collection scheme is far more likely, and any security measures against man-in-the-middle attacks would simply need to detect the intrusion before the attacker collects enough data.

VIII. Conclusion

This paper presents three methods that can effectively improve the security of XOR PUFs against ML modeling attacks. In particular, the proposed methods enable the selective creation of challenges for CRPs, leaking less information about the PUFs underlying model. Our results show that PUF instances that are known to be vulnerable to improved LR and MLP attacks are actually more resilient that previously reported if traditional (random) CRPs are not used.

The three methods provide certain advantages regarding the modeling complexity due to induced data scarcity. While they pose the challenge of limiting unique CRPs, we argue that our third method (BSP) can provide sufficient CRPs to be practical in real-world applications. Additionally, our proposed methods are compatible with other techniques and APUF-based designs. As such, the main conclusion to this paper is that known to be broken PUF designs should be re-evaluated with selective CRPs to investigate whether they are actually insecure. To this end, our future work will focus on expanding our experiments to unreliable and silicon generated CRPs, as well as, other types of attacks (e.g., reliability-based attack [9]) and other PUF designs (e.g., Interpose PUF [5]).

References

[1] W. Arthur and D. Challener, *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*, 1st ed. USA: Apress, 2015.

[2] 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*, ser. CCS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 148–160.

[3] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

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

[5] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose puf: Secure puf design against state-of-the-art machine learning attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 4, p. 243–290, 2019.

[6] U. Rührmair and J. Sölter, "Puf modeling attacks: An introduction and overview," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–6.

[7] 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*, ser. CCS '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 237–249.

[8] 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.

[9] G. T. Becker, "The gap between promise and reality: On the insecurity of xor arbiter pufs," in *Cryptographic Hardware and Embedded Systems – CHES 2015*, T. Güneysu and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 535–555.

[10] M. S. Alkatheiri and Y. Zhuang, "Towards fast and accurate machine learning attacks of feed-forward arbiter pufs," in *2017 IEEE Conference on Dependable and Secure Computing*, 2017, pp. 181–187.

[11] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A machine learning-based security vulnerability study on xor pufs for resource-constraint internet of things," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, 2018, pp. 49–56.

[12] K. T. Mursi, B. Thapaliya, Y. Zhuang, A. O. Aseeri, and M. S. Alkatheiri, "A fast deep learning method for security vulnerability study of xor pufs," *Electronics*, vol. 9, no. 10, 2020.

[13] N. Wisiol, B. Thapaliya, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural network modeling attacks on arbiter-puf-based designs," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2719–2731, 2022.

[14] Z. Paral and S. Devadas, "Reliable and efficient puf-based key generation using pattern matching," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2011, pp. 128–133.

[15] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender puf protocol: A lightweight, robust, and secure authentication by substring matching," in *2012 IEEE Symposium on Security and Privacy Workshops*, 2012, pp. 33–44.

[16] M.-D. 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.

[17] L. Wu, Y. Hu, K. Zhang, W. Li, X. Xu, and W. Chang, "Flam-puf: A response–feedback-based lightweight anti-machine-learning-attack puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4433–4444, 2022.

[18] N. Wisiol, C. Gräbnitz, C. Mühl, B. Zengin, T. Soroceanu, N. Pirnay, K. T. Mursi, and A. Baliuka, "pypuf: Cryptanalysis of Physically Unclonable Functions," 2021.

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