Lightweight Obfuscation Techniques for Modeling Attacks Resistant PUFs

Mohd Syafiq Mispan*†, Basel Halak†, Mark Zwolinski†
†Electronics and Computer Science, University of Southampton, United Kingdom
e-mail: {msm1g14,bh9,mz@ecs.soton.ac.uk}
*Faculty of Engineering Technology, Technical University of Malaysia Malacca, Malaysia

***Abstract*—Building lightweight security for low-cost pervasive devices is a major challenge considering the design requirements of a small footprint and low power consumption. Physical Unclonable Functions (PUFs) have emerged as a promising technology to provide a low-cost authentication for such devices. By exploiting intrinsic manufacturing process variations, PUFs are able to generate unique and apparently random chip identifiers. Strong-PUFs represent a variant of PUFs that have been suggested for lightweight authentication applications. Unfortunately, many of the Strong-PUFs have been shown to be susceptible to modelling attacks (i.e., using machine learning techniques) in which an adversary has access to challenge and response pairs. In this study, we propose an obfuscation technique during post-processing of Strong-PUF responses to increase the resilience against machine learning attacks. We conduct machine learning experiments using Support Vector Machines and Artificial Neural Networks on two Strong-PUFs: a 32-bit Arbiter-PUF and a 2-XOR 32-bit Arbiter-PUF. The predictability of the 32-bit Arbiter-PUF is reduced to $\approx$ 70% by using an obfuscation technique. Combining the obfuscation technique with 2-XOR 32-bit Arbiter-PUF helps to reduce the predictability to $\approx$ 64%. More reduction in predictability has been observed in an XOR Arbiter-PUF because this PUF architecture has a good uniformity. The area overhead with an obfuscation technique consumes only 788 and 1080 gate equivalents for the 32-bit Arbiter-PUF and 2-XOR 32-bit Arbiter-PUF, respectively.**

***Keywords*—Physical Unclonable Function (PUF); Arbiter-PUF; Machine-Learning**

I. Introduction

Low-cost pervasive devices such as Radio-Frequency Identification Devices (RFIDs) and wireless sensor nodes are the foundations for building the next generation of ubiquitous networks or the so-called Internet of Things (IoTs). Examples of applications include secure access, health and social-security cards, electronic passports, smart meters and smart homes. The requirements of small footprints and low power consumption for these resource-constraint pervasive devices introduce a significant challenge to providing fundamental security services, such as authentication and identification. It has been a practice to store a binary vector (secret key) on the device to provide an authentication mechanism. On-chip non-volatile memory (NVM) is required to store the secret key. This may be too costly for pervasive devices. Besides, storing a secret key requires physical security measures such as hiding a secret key in a complex chip layout which also brings additional cost.

Silicon Physically Unclonable Functions [1], or PUFs, have been proposed as a low-cost solution for authenticating pervasive devices. PUFs exploit the inherent manufacturing process variations to map a set of challenges to a set of responses, uniquely and randomly. Several works, [2], [3], [4], [5], have demonstrated the suitability of PUFs to provide a lightweight authentication protocol for area and energy constrained platforms, such as RFIDs and IoT nodes. The Arbiter-PUF is one of the PUFs that has been suggested for lightweight authentication applications, [6]. However, due to the lack of complexity in the challenge to response mapping, it has been shown that the Arbiter-PUF and its derivatives are susceptible to modelling attacks resulting from machine learning (ML) techniques, [7]. The Controlled-PUF has been proposed as one technique to make the underlying PUF (e.g. Arbiter-PUF) robust against an ML-attack by using hash functions to process the logic of challenge and response, [8]. However, hash functions consume thousands of logic gates, which is too costly for low-cost pervasive devices that typically require fewer than 1000 gates, [3]. In this paper, our goal is to reduce the cost of implementation but still provide security against an ML-attack. The main contributions of this work are:

1) We show that using an obfuscation technique in the post-processing of the PUF's responses can increase the resilience against an ML-attack.
2) We also show that a 50% uniformity is the desired property for PUFs and combined with the obfuscation technique can further increase the resilience to an ML-attack.

The rest of the paper is organized as follows. Section II discusses the literature relating to ML-attacks and the architecture of Arbiter-PUFs. Section III describes challenge-response pair (CRP) generation, machine learning algorithms and the attacker model used in our work. The analysis of an ML-attack is presented in Section IV. Finally, conclusions are drawn in Section V.

II. Background

A. Related Work

ML-attack resistance describes the complexity of the challenge to response mapping for a particular PUF. An ML-attack is most applicable to Strong-PUFs, [7]. As defined in [9], Strong-PUFs are a type of PUF that are able to generate an exponential number of challenge-response pairs (CRPs), as in the Arbiter-PUF, [6]. However, the Arbiter-PUF can be easily modelled by ML due to the linear addition of the inherent delay values, [6]. Several PUFs have been derived from the Arbiter-PUF to introduce a non-linearity into the mapping function of

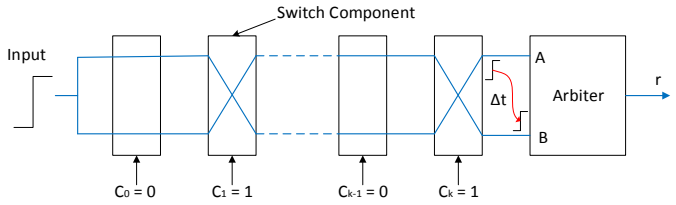
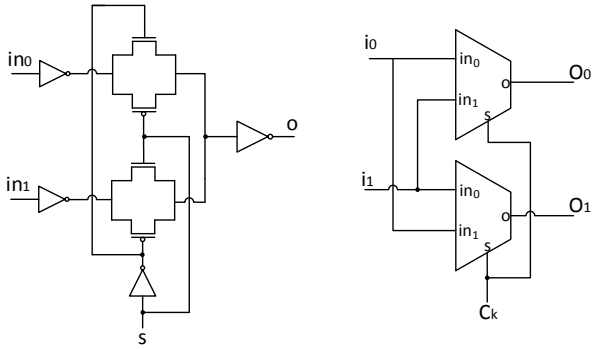


Fig. 1: Arbiter-PUF



(a) (b)

Fig. 2: Switching element circuit; (a) 2-to-1 MUX, (b) Switch component

the CRPs, such as the Feed-forward Arbiter-PUF, [10], XOR-PUF, [11] and Lightweight Secure PUFs, [2]. Although these Arbiter-PUF's derivatives increase the ML-attack resistance, ML techniques are still able to model them with high accuracy, [7]. Elsewhere, Gassend *et al*, [8], proposed a Controlled-PUF which uses a one-way hash function to increase the mapping complexity of the CRPs in a Strong-PUF. The challenges are pre-processed by the hash function before being applied to the Strong-PUF and the responses of the Strong-PUF are post-processed by the hash function before being output by the Controlled-PUF. For any attacker that only has access to the interface of the device, the Controlled-PUF successfully disables an ML-attack. Nevertheless, a one-way hash function is too costly for low-cost pervasive devices.

A few works focus on how to increase the resilience against an ML-attack by controlling either the challenges or responses during the authentication process. Rostami *et al*, [4], proposed a sub-string matching technique in which only a subset of PUF response strings is sent to the verifier during authentication. Elsewhere, Gao *et al*, [5] proposed an Obfuscated-PUF (OB-PUF) in which a partial challenge is sent by the verifier to the OB-PUF (i.e., the prover). Subsequently, within an OB-PUF, a partial challenge is padded with a random pattern generated by a random number generator (RNG) to make up a full-length challenge. Generally, both works, [4], [5], use the same technique by only exposing a subset of either challenges or responses. However, this might increase the authentication time to run the matching algorithm, as well as the area, on the verifier side. One might argue, however, that the area is not a concern since the verifier has always been assumed to be resource rich. In our work, we propose an obfuscation technique to post-process the responses of Strong-PUFs. We use the Arbiter-PUF and XOR Arbiter-PUF as test cases. Both PUFs will be described in the next sections.

B. Arbiter-PUF and Model Description

The Arbiter-PUF [6] consists of k stages; each stage is composed of two 2-to-1 multiplexers as shown in Figs. 1 and 2. A rising pulse at an input propagates through two nominally identical delay paths. The paths for the input pulse are controlled by the the switching elements, which are set by the bits of the challenge. For c_k=0, the paths go straight through, while for c_k=1 they are crossed. Because of manufacturing variations, however, there is a delay difference Δt between the paths. An arbiter at the end generates a response '0' or '1' depending on the difference in arrival times. In our simulation, a set-reset (SR) latch has been used as the arbiter block. The functionality of the Arbiter-PUF can be described by an additive linear model, [6], [7]. The total delays of both paths are modelled as the sum of the delays in each stage (switch components) depending on the challenge C $(c_1, c_2 ... c_k)$. The final delay difference Δt between the two paths in a k-bit Arbiter-PUF can be expressed as:

$$\Delta t = \vec{w}^T \vec{\Phi} \tag{1}$$

where parameter $\vec{w}$ is the delay-determined vector and $\vec{\Phi}$ is the feature vector. Both parameters are functions of the applied k-bit challenge with dimension $k+1$. As described in [5], we denote $\delta_i^{1/0}$ as the delay in stage i for the crossed (1) and uncrossed (0), respectively. Hence, δ_i^1 is the delay of stage i when $c_i = 1$, while δ_i^0 is the delay of stage i when $c_i = 0$. Then

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

where $w^1 = \frac{\delta_1^0 - \delta_1^1}{2}$, $w^i = \frac{\delta_{i-1}^0 + \delta_{i-1}^1 + \delta_i^0 - \delta_i^1}{2}$ for all $i = 2, ..., k$, and $w^{k+1} = \frac{\delta_k^0 + \delta_k^1}{2}$. Furthermore,

$$\vec{\Phi}(C) = (\Phi^1(C), ..., \Phi^k(C), 1)^T \tag{3}$$

where $\vec{\Phi}^j(C) = \prod_{i=j}^{k}(1 - 2c_i)$ for $j = 1, ..., k$

From (2), the vector $\vec{w}$ encodes the delay in each stage of the Arbiter-PUF and via $\vec{w}^T \vec{\Phi} = 0$ determines the separating hyperplane in the space of all feature vectors, $\vec{\Phi}$. The delay difference, Δt, is the inner product of $\vec{w}$ and $\vec{\Phi}$. If $\Delta t > 0$, the response bit is '1', otherwise, the response bit is '0'. Determination of this hyperplane allows prediction of the PUF.

C. XOR Arbiter-PUFs

The l-XOR Arbiter-PUF employs l individual Arbiter-PUFs in parallel, each with k stages. The same k-bit challenge is applied to all of them, and their individual outputs are XORed to produce a final response, t_{XOR}, [11]. The XOR function helps to make the uniformity of the Arbiter-PUF close to the ideal value of 50%. The uniformity is a measure of the proportion of 0's and 1's in the response bits of a PUF and it is calculated using the Hamming Weight (HW), [12]. For the

2-XOR Arbiter-PUF, the probability of the t_{XOR} output being '1' is given as:

$$P(t_{XOR} = 1) = P(X \cup Y) - P(X \cap Y) \quad (4)$$
$$= P(X) + P(Y) - 2P(X)P(Y) \quad (5)$$

where $P(X)$ and $P(Y)$ are the probabilities of the two Arbiter-PUF outputs being one, respectively. We found that the average uniformity of Arbiter-PUFs is around 30% to 35%. Based on (5), as l or the number of XORs increases, the uniformity become very close to the ideal value of 50%. Because of the importance of the uniformity metric in improving the resilience to an ML-attack, we also considered the XOR-Arbiter-PUF here. We used a 2-XOR Arbiter-PUF in combination with our obfuscation technique.

III. Methodology

In this section, we briefly discuss the simulation setup for CRP generation, the ML algorithms, and the threat model used in this work.

A. CRP Generation

For CRP generation, a 32-bit Arbiter-PUF has been implemented in a low-κ 65-nm technology node and simulated using the BSIM4 (V4.5) transistor model with a nominal supply voltage of 1.2V and a room temperature of 25^{o}C. Intrinsic variations such as effective length, effective width, oxide thickness and threshold voltage are modelled in Monte Carlo simulations using the built-in fabrication standard statistical variation (3σ variations) in the technology design kit. A total of 32000 CRPs have been generated for ML-attack analysis.

B. Machine Learning

In our study, we have employed two machine learning techniques: Support Vector Machine (SVM) and Artificial Neural Network (ANN). SVM has been previously used for testing the modelling-attack resistance of Strong-PUFs, [6], [7], [13]. An SVM classifies the binary response of a Strong-PUF by finding the best hyperplane that separates all data points of one class (response 1's) from those of the other class (response 0's). We performed a *5-fold* cross-validation to determine the SVM's best kernel setting (linear, radial basis function or polynomial) for a particular set of data. Once the best kernel is determined, we incrementally increased the amount of training data, extracted each of the SVM models, applied the test data that is not part of the training data and computed the prediction accuracy.

We also considered an ANN since it is known to have the capability of modelling highly non-linear systems. An ANN is a system formed by interconnected computing nodes called neurons. The simplest ANN has only one hidden layer of neurons. In each neuron, all input vector values are weighted, summed, biased and applied to an activation function to generate an output. During training, an error resulting from the difference between a predicted and observed value is propagated back through the network and the neuron's weight and bias are adjusted and updated. The training process stops when the prediction error reaches a predefined value or a predetermined number of epochs is completed. Based on our ML experiments, resilient back-propagation has been chosen as the best training algorithm considering the prediction accuracy and fast convergence time; this is consistent with the explanation elsewhere, [14]. Both the SVM and ANN are implemented in MATLAB.

C. Threat Model

An adversary may have several motives for attacking deployed low-end devices, such as gaining secure access or an electronic passport, through RFID, or to report fake data in smart metering. In order to achieve one of these goals, we assume that the attacker is restricted to non-invasive CRP measurement. The attacker only has access to the interface of the device, and can apply a polynomial number of challenges to the device and then collect the corresponding responses. Further, the attacker tries to derive a numerical model from the CRPs data by using ML techniques, as described in Section III-B.

IV. Analysis

A. PUF Modelling Attack

As explained in Section II-A, a Controlled-PUF uses a secure one-way hash function to break the relationship between challenges and responses. With our threat model (Section III-C), a Controlled-PUF successfully disables a modelling attack since it is known to be a hard problem to invert a one-way function. However, a one-way hash function consumes area and is power-hungry. To demonstrate this, we compiled readily-available SHA-1 and SHA-256 Verilog code, [15], using Synopsys Design Compiler. The area and power for SHA-256 are 12980 gate equivalents (GE) and 1.688mW, respectively. The area and power for SHA-1 are 9567 GE and 1.256mW, respectively. Clearly, one-way hash functions are too costly for low-cost pervasive devices, which typically require fewer than 1000 gates, [3].

Nevertheless, one important thing to be learned from SHA-1 and SHA-256 is that the hashing of input and output is exaggerated by an iterated compression function and obfuscation of their internal states, [16]. Learning from this, we could construct a simple technique of obfuscation for post-processing of a PUF's responses as shown in Fig. 3. L is the length of the PUF response string in bits and n is the length of the bit string that gets combined. In our study, $L = 32$ and $n = 8$.

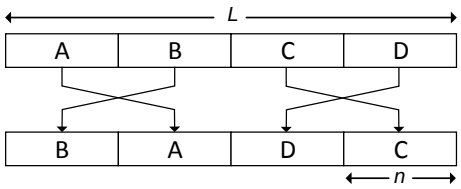
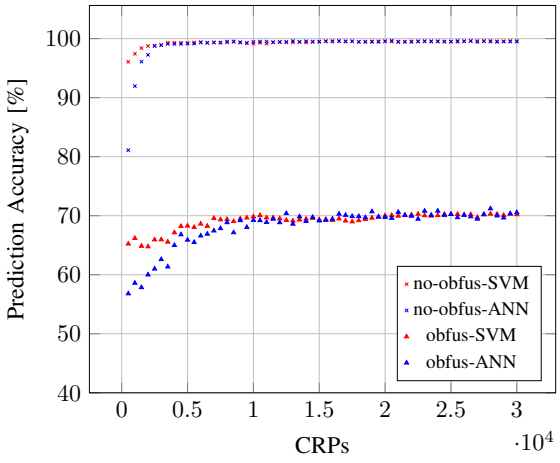
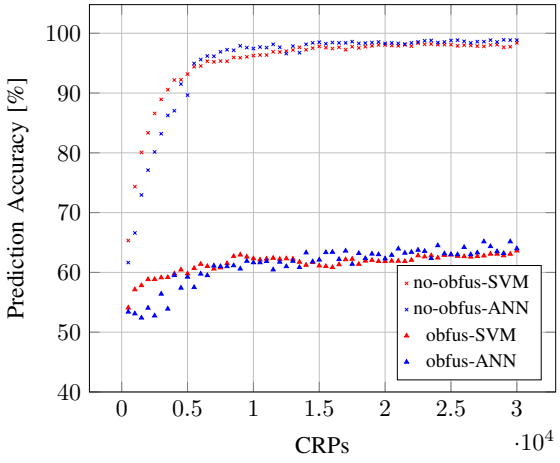


Fig. 3: Obfuscation of PUF responses

We applied the obfuscation technique of Fig. 3 to a 32-bit Arbiter-PUF and a 2-XOR 32-bit Arbiter-PUF. We evaluated the resiliency of this technique against ML-attack by using an SVM and ANN. Figure 4 shows the results of an ML-attack



(a) 32-bit Arbiter-PUF



(b) 2-XOR 32-bit Arbiter-PUF

Fig. 4: ML-attack for 32-bit Arbiter-PUF and 2-XOR 32-bit Arbiter-PUF using SVM and ANN

using both ML techniques. For a comparison, the ML-attack without obfuscation is also shown in Fig. 4. As can be seen from Fig. 4a, without an obfuscation technique, the Arbiter-PUF can be modelled easily with a prediction accuracy of 99% at $\approx$ 30000 CRPs. The resilience against an ML-attack is better for the 2-XOR Arbiter-PUF but still can be predicted with high accuracy, as shown in Fig. 4b. As discussed in Section II-C, the XOR Arbiter-PUF has a uniformity close to the ideal value of 50%, hence, increasing the randomness and unpredictability. Although the 2-XOR Arbiter-PUF can still be predicted in this case, it is interesting to note that uniformity is indeed an important metric for a PUF.

With an obfuscation technique, the prediction accuracy is reduced dramatically to $\approx$ 70% at 30000 CRPs for the Arbiter-PUF. Combining the obfuscation technique with the 2-XOR Arbiter-PUF helps to further reduce the prediction accuracy to $\approx$ 64% at 30000 CRPs. Here, we can see that a good uniformity caused by the XOR function helps to increase the resilience to the ML-attack by about 6%. From (1), without obfuscation, the challenges can be easily classified by generating the response '0' or '1' via a separating hyperplane $\vec{w}^T\vec{\Phi} = 0$. However, the obfuscation technique introduces a non-linear function which increases the complexity of the challenge to response mapping. Hence, the obfuscated response can be expressed as:

$$\Delta t_{obfus} = f(\vec{w}^T\vec{\Phi}) \tag{6}$$

Overall, an obfuscation technique helps to increase the resilience to ML-attacks. From an ML perspective, we note that for all the test cases in Fig. 4, the ANN learning curve is slower at the beginning given a small set of training data. However, as the training data increases, the ANN is better than SVM especially for the XOR Arbiter-PUF which proves that the ANN is suitable for highly non-linear systems.

B. Impact of L and/or n Variations on ML-attack

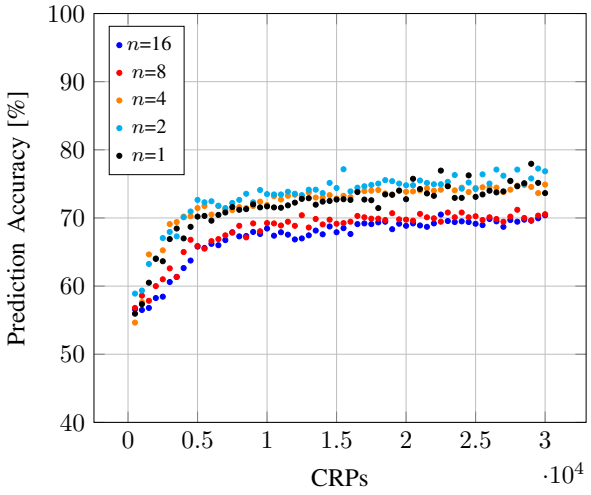
So far we have discussed only one set of permutations (see Fig. 3) for $L = 32$ and $n = 8$. In our study, the values of L and n in Fig. 3 are powers of two. Therefore, we have evaluated the impact of an ML-attack using an ANN on a 32-bit Arbiter-PUF with $L = 32$ and $L = 64$ for a set of $n = \{32, 16, 8, 4, 2, 1\}$ and the obfuscation technique shown in Fig. 3. As can be seen in Fig. 5, as n reduces the probability of prediction is higher for both $L = 32$ and $L = 64$ because some of the challenges to response mappings are essentially not obfuscated. A similar pattern has been observed for the 2-XOR 32-bit Arbiter-PUF as n reduces.

C. Impact of Other Permutations on ML-attack

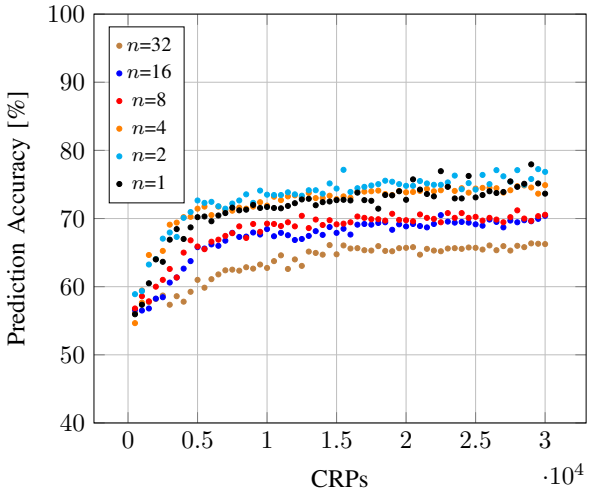
All of the above analysis has only considered an obfuscation technique with the permutations shown in Fig. 3. As discussed in Section IV-A, with $L = 32$ and $n = 8$, the total number of permutations can be calculated from (7) which gives a total of 23 permutations.

$$Total\_Permutations = \left(\frac{L}{n}\right)! - 1 \tag{7}$$

An ML-attack for all 23 possible permutations, using the ANN algorithm for the 32-bit Arbiter-PUF and the 2-XOR 32-bit Arbiter-PUF is shown in Fig. 6. It is clear that different permutations have different susceptibilities to an ML-attack which causes the spread of the prediction accuracy as the number of CRPs for ML training increases. However, from our analysis, we found no distinct pattern between different permutations and the probability of being random once the L-bit response is permuted or obfuscated is as observed in Section IV-B. From (6), it might be the case that different permutations induce a different level of non-linearity into (1) which causes a different susceptibility to an ML-attack. We also observed the spread of all 23 possible permutations for $L = 64$ and $n = 16$ as the number of CRPs for ML training increases. However, the best and the worst prediction accuracy for $L = 64$ and $L = 32$ do not occur on a similar set of permutations. Nevertheless, one interesting observation is that the prediction accuracy of a permutation, as in Fig. 3, is always at the middle point of the observed spread for $L = 64$ and $L = 32$ (i.e., compare Fig. 4 and Fig. 6). This indicates that the prediction accuracy of this permutation could be a benchmark if we have to choose a permutation from a very large number of possible permutations for use as the response obfuscation.

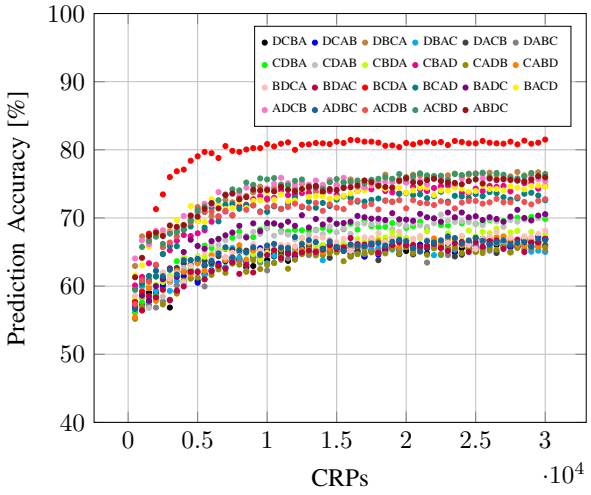


(a) $L = 32$

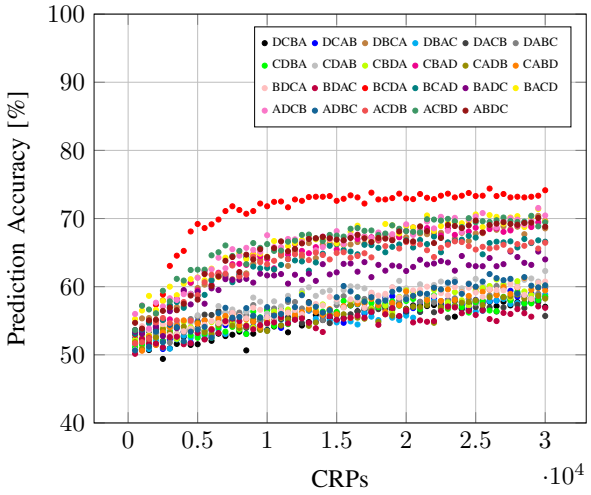


(b) $L = 64$

Fig. 5: ML-attack using ANN for 32-bit Arbiter-PUF with varying n.



(a) 32-bit Arbiter-PUF



(b) 2-XOR 32-bit Arbiter-PUF

Fig. 6: ML-attack using ANN of all possible permutations with $L = 32$ and $n = 8$

It is desirable to increase the possible number of permutations to decrease the susceptibility to a random guess of a challenge to response mapping. To achieve that, from (7) $\binom{L}{n} \geq 16$, which gives at least a total of 10^{13} permutations.

D. Power, Area and Predictability Comparison

In Section IV-A, we have presented an obfuscation technique. From the hardware perspective, this technique can be implemented with serial-in-parallel-out (SIPO) and parallel-in-parallel-out (PIPO) registers. Fig. 7 shows the main functional unit blocks (FUBs) within a typical low-cost pervasive device, which are a 32-bit linear feedback shift register (LFSR), a 32-bit Arbiter-PUF or 2-XOR 32-bit Arbiter-PUF, a 32-bit SIPO and a 32-bit PIPO. The LFSR is used to generate L challenges to apply to the PUF to generate an L-bit response string which is stored in an L-bit SIPO shift register. Once the L-bit response string has been generated, it is moved to the L-bit PIPO where the connectivity between SIPO and PIPO has been obfuscated according to Fig. 3. An obfuscated response in PIPO is sent to the verifier for authentication. As mentioned in Section IV-A, in our study, L has been set to 32. Therefore, the area and power for the FUBs have been estimated based on a 32-bit response string using Synopsys Design Compiler. If one has to increase the size of L, it only impacts the size of the SIPO and PIPO registers.

As reported in [3], low-cost pervasive devices typically require fewer than 1000 gate equivalents. From Table I, the estimated area and the prediction accuracy for a 32-bit Arbiter-PUF and a 2-XOR 32-bit Arbiter-PUF with an obfuscation technique indicate that this technique can provide secure and lightweight authentication of low-cost pervasive devices. Compared to an XOR Arbiter-PUF without an obfuscation, they have a reasonably low area overhead but there the responses can be predicted with high accuracy. An OB-PUF proposed

TABLE I: Comparison of Area, Power and Prediction Accuracy

Type	Area [GE]	Power [mW]	CRPs	Prediction Accuracy [%]
32-bit Arbiter-PUF + proposed obfuscation	788	0.155	3×10^4	70.6
2-XOR 32-bit Arbiter-PUF + proposed obfuscation	1080	0.197	3×10^4	64.0
2-XOR 32-bit Arbiter-PUF [11]	918	0.163	3×10^4	98.9
3-XOR 32-bit Arbiter-PUF [11]	1210	0.205	3×10^4	98.4
Lightweight OB-PUF [5]	NR	NR	2×10^4	63.27
Controlled-PUF [8]	26251	3.42	NA	unpredictable [9]

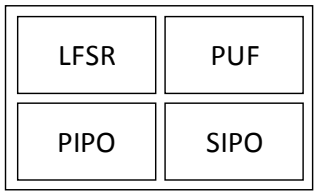


Fig. 7: Functional unit blocks within low cost pervasive devices

in [5] achieves approximately similar prediction accuracy as the 2-XOR 32-bit Arbiter-PUF with an obfuscation technique at 20000 CRPs. Nevertheless, the area and power of OB-PUF were not reported. With our attacker model as explained in Section III-C, Controlled-PUF which uses a one-way hash function makes the input-output relation of PUF unpredictable and therefore, disables the ML-attack [9]. For a comparison, the area and power for Controlled-PUF are included in Table I which was calculated from the architecture proposed in [8], but without considering the error correction code (ECC), and with the 32-bit Arbiter-PUF as the underlying PUF and SHA-256 as the hash function. Clearly, our obfuscation technique has much less area – it is more than 24$\times$ smaller.

V. Conclusion

Providing security such as authentication for low-cost pervasive devices is a significant challenge due to the small footprint and low power consumption. Strong-PUFs are a promising technology to provide low-cost authentication for pervasive devices. However, the susceptibility to ML-attack is still a major concern. In this paper, we have proposed a simple obfuscation technique for post-processing of Strong-PUF responses to increase the resilience to an ML-attack. Using this technique, we are able to reduce the predictability of Arbiter-PUF responses to $\approx$ 70%. Combining the obfuscation technique with a PUF that has a good uniformity (i.e., close to an ideal value of 50%), such as the XOR Arbiter-PUF, helps to further reduce the predictability to $\approx$ 64%. Our obfuscation technique consumes only 788 and 1080 GEs when implemented with the 32-bit Arbiter-PUF and the 2-XOR 32-bit Arbiter-PUF, respectively. Hence, this technique is suitable for lightweight security devices.

VI. Acknowledgments

The authors would like to thank Ministry of Education Malaysia (MOE) and Technical University of Malaysia Malacca (UTeM) for financial support.

References

[1] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *ACM Conference on Computer and Communications Security*, 2002, pp. 148–160.

[2] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 670–673.

[3] E. Oztürk, G. Hammouri, and B. Sunar, "Towards robust low cost authentication for pervasive devices," in *IEEE International Conference on Pervasive Computing and Communications*, 2008, pp. 170–178.

[4] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, pp. 37–49, 2014.

[5] Y. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavehei, D. Abbott, and D. C. Ranasinghe, "Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices," in *IEEE International Conference on Pervasive Computing and Communication Workshops*, 2016, pp. 1–6.

[6] D. Lim, "Extracting secret keys from integrated circuits," MSc. Thesis, Massachusetts Institute of Technology, 2004.

[7] U. Ruhrmair and J. Solter, "PUF modeling attacks: An introduction and overview," in *Design, Automation & Test in Europe Conference & Exhibition*, 2014, pp. 1–6.

[8] B. Gassend, M. van Dijk, D. Clarke, E. Torlak, and S. Devadas, "Controlled physical random functions and applications," *ACM Transactions on Information and System Security*, vol. 10, no. 4, pp. 15:1 –15:22, 2008.

[9] 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 Forensic and Security*, vol. 8, pp. 1876–1891, 2013.

[10] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. V. 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.

[11] G. E. Suh and S. Devadas, "Physical Unclonable Functions for device authentication and secret key generation," in *ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[12] M. S. Mispan, B. Halak, and M. Zwolinski, "NBTI aging evaluation of PUF-based differential architectures," in *IEEE International Symposium on On-Line Testing and Robust System Design*, 2016, pp. 103–108.

[13] M. S. Mispan, B. Halak, Z. Chen, and M. Zwolinski, "TCO-PUF: A subthreshold physical unclonable function," in *IEEE PRIME*, 2015, pp. 105–108.

[14] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *IEEE International Workshop on Information Forensics and Security*, 2012, pp. 37–42.

[15] J. Strömbergson, "sha256," https://github.com/secworks/sha256, 2013.

[16] National Institute of Standards and Technology (NIST), *FIPS PUB 180-4: Secure Hash Standard (SHS)*, August 2015. [Online]. Available: http://www.itl.nist.gov/fipspubs/