A Permutation Challenge Input Interface for Arbiter PUF Variants Against Machine Learning Attacks

Yu Zhuang
Computer Science Department
Texas Tech University
Lubbock, Texas 79409–3104
Email: yu.zhuang@ttu.edu

Gaoxiang Li
Computer Science Department
Texas Tech University
Lubbock, Texas 79409–3104
Email: gaoxiang.li@ttu.edu

Khalid T. Mursi
College of Computer Science & Engineering
University of Jeddah
Jeddah 21959, Saudi Arabia
Email: kmursi@uj.edu.sa

2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) | 978-1-6654-6605-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISVLSI54635.2022.00094

***Abstract*—Internet of Things (IoT) have broad and deep penetration into our society, and many of them are resource-constrained, calling for lightweight security protocols. Physical unclonable functions (PUFs) leverage physical variations of circuits to produce responses unique for individual devices, and hence are not reproducible even by their manufacturers. Implementable with simplistic circuits and operable with low energy, PUFs are promising candidates as security primitives for resource-constrained IoT devices. Arbiter PUF (APUF) and its variants are lightweight in resource requirements but suffer from vulnerability to machine learning attacks. To defend APUF variants against machine learning attacks, in this paper we investigate a challenge input interface, which incurs low overhead. Analytical and experimental studies were carried out, showing substantial improvement of resistance against machine learning attacks when a PUF is equipped with the interface, rendering interfaced APUF variants promising candidates for security critical applications.**

***Index Terms*—IoT security, physical unclonable function, machine learning attack, arbiter PUF variants**

I. Introduction

Many IoT devices are resource-constrained, and can support only a low level of operating power, calling for lightweight security protocols. Physical unclonable functions (PUFs) are an emerging class of hardware primitives for implementing security protocols. Small scale variations of integrated circuits exist in manufactured silicon chips. These variations are regarded as side effects for conventional circuits [1], [2], but they make each chip unique and can be exploited to prevent semiconductor re-fabrication. PUFs utilize these variations as hardware fingerprints to produce responses unique for individual PUF circuits [1]–[5], and hence are not reproducible even by the PUF manufacturers. Such features of PUFs present a potential solution for challenges that many IoT devices are facing. Implementable with simplistic circuits with only thousands of transistors, PUFs require low fabrication cost and consume very low operating power, rendering them potential candidates for resource-constrained IoT devices

Though physically unclonable, some PUFs are "mathematically clonable" in the sense that the responses of a PUF can be predicted accurately by machine-learning methods. Attackers can eavesdrop on the communications between a PUF and its trusted partner, and the challenges sent to a PUF and the responses from the PUF can be collected by attackers to train machine learning models. Such models are able to accurately predict the responses of the PUFs after the models are trained with sufficient challenge-response pairs (CRPs). Thus, mathematical clonability allows attackers to develop malicious software to impersonate the PUF.

The arbiter PUF (APUF) [6], [7] is highly lightweight but vulnerable to machine learning attacks. Efforts to improve its security resulted in many APUF variants including XOR PUFs (XPUFs) [2], LSPUFs, FF PUFs [6], [7], Interpose PUFs. But these PUFs still succumb to machine learning attacks [8], and the vulnerable ones include XPUFs [8]–[10], LSPUFs with 6 or fewer component arbiter PUFs [11], [12], FF PUFs [8], and Interpose PUFs [12]. Besides sophisticated PUFs, there are also protocols obfuscating challenges and/or responses, or hiding PUF responses but transmitting transformed responses [13]–[17]. These protocols incur hardware cost not very small compared with the APUF itself, e.g. TRNG and transistors to make use of TRNG, fuzzy extractor, or cryptographic cipher.

Observing that all successful machine learning attack methods employ a transform of the challenge into a "feature" vector that simplifies the relationship between the "feature" and the response, we analysed that machine learning attacks might be thwarted if an input interface can prevent attackers from obtaining the desirable "feature" vector. We developed a challenge input interface based on our analysis, and carried out experimental studies to examine the interfaced PUFs, which clearly showed the effectiveness of the interface.

II. Defensively Interfaced PUFs

An interface that would probably naturally occur to many people is the one that outputs the permuted challenge bits to PUF stages. Wisiol et al. [11] attacked 4-XOR PUFs with a master challenge and the challenge fed to each of the four component APUFs of the 4-XOR PUF is a permutation of the master challenge, and these XPUFs can be considered as equipped with permutation interfaces. Their attack study showed permutations led to improved security against machine learning attacks, but still there is a significant percentage of interfaced XPUFs succumbed to machine learning attacks.

The study of Wisiol et al. [11] led us to a careful consideration of guiding principles for designing permutation interfaces, and we figured that two ideal conditions for a set of interfaces to secure PUFs against machine learning attacks are:

2159-3477/22/$31.00 ©2022 IEEE
DOI 10.1109/ISVLSI54635.2022.00094

- each member of the set turns an interfaced PUF into one that is secure against the all machine learning attacks, and practically against the most powerfully machine learning attacks developed so far, and
- there are exponentially many members in the set.

The need for exponentially many permutations is to thwart exhaustive search attacks to find the permutation implemented on a PUF instance. To look for such a set of permutations, there is a big challenge. The set should have exponentially many interfaces and each interface can secure a PUF against the most powerful machine learning attacks developed so far, but it is not possibleto experimentally test every interface using existing machine learning attacks. But we have the following idea that could lead to a solution for this challenge.

An interfaced PUF is viewable as a classification to be machine learned. Due to the lack of full mathematical understanding of machine learning, in general it is difficult to know if there is a machine learning method that can attain adequate modeling accuracy for an interfaced PUF without applying the method to it. But there is hope. As pointed out by Princeton mathematics professor Weinan E, machine learning is function approximation [18]. While it is difficult to tell if an interfaced PUF can be accurately modelled without trying any method, it is generally true that the more nonlinear a function is the more difficult the function can be approximated. Thus, to estimate the potential of an interfaced PUF against machine learning before trying any machine learning method, we will choose an appropriate indicator for the level of nonlinearity of the classification's separating surface. We wish to comment that we are not looking for a metric for nonlinearity to measure how "far away" a function is from being linear because, first, we have seen no existing nonlinearity metric and, second, we also believe it not useful since different functions of different forms of nonlinearity make such a metric not practical if not impossible.

Seeing that the classification separation surface of a k-XOR PUF is representable by a k-th order multivariate polynomial [8], and realizing that higher-order polynomials are in general more difficult to be accurately approximated than lower order polynomials, and also realizing that the polynomial order is an indicator reasonably easy to estimate for some PUFs, our investigation is to focus on looking for

- a set of similar permutation interfaces with different interfaces in the set turning the APUF into classification problems with separation surfaces defined by similar polynomials of different orders, and
- there are exponentially many members in the set.

We are targeting interfaces that will resulting in similar polynomials, because we believe similar polynomials have similar properties and then the orders of the polynomials might be the distinguisher in their behaviours in resistance to machine learning, which would reasonably allow us to avoid doing experimental attacks of all exponentially many interfaces. With this line of thought, we propose the following set of permutation interfaces.

The Proposed Set of Permutation Interfaces

For n-stage PUFs with $n \geq 64$, and for $m \in \{3, 4, \cdots, n/3\}$, let $S_m = \{i_1, i_2, \cdots, i_m\}$ be a subset of $\{1, 2, \cdots, n\}$ with $i_1 < i_2 < \cdots < i_m$ satisfying $i_j + 1 < i_{j+1}$ for all $j = 1, 2, \cdots, m-1$, that is, any two integers in S_m are separated by a distance of 1 or larger. For a positive integer $k \leq m/2$, let $P^k_{S_m}$ be a permutation on $\{1, 2, \cdots, n\}$ such that

$$\begin{cases} P^k_{S_m}(i_j) = i_{(j+k)\%m} & \text{for } i_j \in S_m, \\ P^k_{S_m}(i) \ = i & \text{for } i \notin S_m, \end{cases} \tag{1}$$

where $x\%m = 1+(x \ mod \ m)$. The permutation $P^k_{S_m}$ is called the k-hop circular right-shift on S_m, and integers in S_m are called the non-stationary points of the permutation $P^k_{S_m}$ and other integers in $\{1, 2, \cdots, n\}$ are called stationary points.

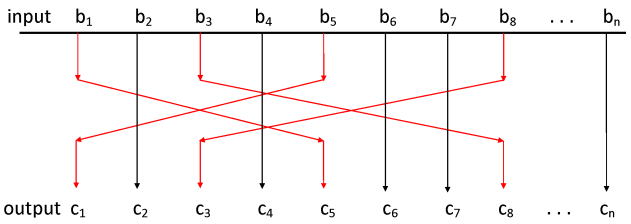

Fig. 1. A permutation interface with $S_m = \{1, 3, 5, 8\}$ and $k = 2$.

An illustration of an interface with $S_m = \{1, 3, 5, 8\}$ and $k = 2$ is given in Fig. 1. Below is a result revealing the nonlinear relationship between the response and input bits, with proof omitted due to page limit.

Theorem 1. An n-stage APUF equipped with the interface $P^k_{S_m}$ has a classification separation surface defined by an n-variable polynomial of an order between $(4k–2)$ to $(4k+1)$. An n-stage x-XOR PUF is equipped with the permutation interface $P^k_{S_m}$ has a classification defined by an n-variable polynomial of order $(4k-2)$ or higher.

From Theorem 1, one can see that if $m \geq 6$ and $3 \leq k \leq m/2$, the polynomial for an interfaced APUF is of at least 10-th order, and the polynomial for an interfaced 3-XOR PUF is of at least 30-th order. It can also be verified that the number of sets S_m that satisfy $10 \leq m \leq n/3$ and $i_j + 1 < i_{j+1}$ for all $j = 1, 2, \cdots, m-1$ grows exponentially with n. This can be verified by choosing $m = n/3$ and i_1 from $\{1, 2\}$, i_2 from $\{4, 5\}$, $\cdots$, i_j from $\{3j-2, 3j-1\}$ for $j = 1, 2, \cdots, m$, resulting in $2^m = 2^{n/3}$ permutations whose polynomials are of at least 10-th order for interfaced APUFs.

Theorem 1 and the subsequent discussion indicate the existence of exponentially many permutation interfaces, each of which turns an APUF into a classification with a separation surface defined by a high-order polynomial. We wish to emphasize that they do not necessarily guarantee the existence of exponentially many permutation interfaces capable of securing APUFs against the most powerful machine learning attacks. Though the order of a polynomial is an indicator, but exactly how large the values of permutation parameters m and k can

lead to secure PUFs have to be experimentally determined. Thus, in Sec. III, we will experimentally examine how well interfaced APUs perform against attacks with different values for interface parameters.

III. Experimental Studies of Interfaced PUFs

A. Machine Learning Methods for Attacking PUFs

Existing PUF attack studies [8]–[10], [12], [19] show that among various machine learning attack methods, neural network has the highest modeling power for capturing the behavior of a broad range of PUFs, and hence we have decided to use neural networks for our attack study.

After having tried multiple sets of parameters, including parameter sets used in studies [10], [19], the neural network we used in our attack study of PUFs and their interfaced counterparts has the following structure of layers with other neural network parameters listed in Table I.

- The input layer of n input bits $b_1, b_2, \cdots, b_n$;
- the second layer that transforms input layer to ϕ's according to $\phi(i) = (2b_i - 1)(2b_{i+1} - 1)\cdots(2b_n - 1)$;
- four hidden layers of the neurons whose weights for all neurons are to be trained, where the number of neurons at these layers is specified in Table I; and
- the last layer is the single-bit output layer.

TABLE I
Neural Network for attacking 64-stage PUFs

Parameter	Description
Optimizing Method	ADAM
Hidden Lyr. Actv. Fx.	tanh
Output Lyr. Actv. Fx.	Sigmoid
Learning Rate	Adaptive
Hidden Layers	4 hidden lyr. $(64, 32, 32, 64)$
Loss Function	Binary cross entropy
Mini-batch Size	100K
Kernel Initializer	Random Normal
Epoch	500
Early Stopping	Validation accuracy $\geq$ 98%

The attack method was implemented in Python using the TensorFlow library. The machine learning method for each PUF instance is run for up to 500 epochs with an early stopping when the training validation accuracy reaches 98%. The experiments used an 84-1-15 split, with 84% of CRP data for training, 1% of data for validation, and 15% of the data for testing the trained model.

B. Attack Study on Interfaced PUFs

We applied the interface to APUFs and XPUFs, all 64 stages. We chose a set of interfaces with different types of permutations as indicated by interface notations in Table II with the notations explained in Table III, where in the column titled "S_m Property", words "Pts separated" means that all points i_j in the set S_m satisfy $i_j + 1 < i_{j+1}$ for $j = 1, 2, \cdots, m-1$, and "All pts consecutive" means that $i_j + 1 = i_{j+1}$ for all $j = 1, 2, \cdots, m-1$.

TABLE II
Attack Results with/without Permutation Interface

PUF Type	Interface	CRPs	Avg. Acc.	Success Rate
Arbiter PUF	No interface	1 K	98%	100%
	$A_{5,1,1}$	10 M	72%	10%
	$A_{5,1,2}$	10 M	68%	0%
	$A_{6,1,1}$	10 M	62%	0%
	$A_{6,1,2}$	10 M	66%	0%
	$A_{6,1,3}$	10 M	67%	0%
	$A_{7,1,1}$	10 M	66%	0%
	$A_{7,1,2}$	10 M	65%	0%
	$A_{7,1,3}$	10 M	61%	0%
	$A_{8,1,3}$	10 M	67%	0%
	$A_{8,0,3}$	10 M	71%	20%
	$A_{9,0,3}$	10 M	75%	20%
3-XOR PUF	No interface	6 K	98%	100%
	$A_{3,1,1}$	10 M	74%	0%
	$A_{4,1,1}$	10 M	71%	0%
	$A_{4,1,2}$	10 M	72%	0%
	$A_{5,1,1}$	10 M	65%	0%
	$A_{5,1,2}$	10 M	69%	0%
	$A_{6,1,1}$	10 M	63%	0%
	$A_{6,1,2}$	10 M	66%	0%

For the **Success Rate**, an attack is considered a success if its prediction accuracy is 80% or higher. Lower success rate indicates higher security.

TABLE III
Permutation Interfaces used in Experimental Studies

Notation	N-stationry Pts	S_m Property	Permutation
$A_{m,1,k}$	m	Pts are separated	k-hop circular r-shift
$A_{m,0,k}$	m	All pts consecutive	k-hop circular r-shift

For each type of interfaces listed in Table II, i.e. each row in the table, 20 different instances of interface-PUF pairs were generated. In the generation of these interface-PUF instances, for each interface-PUF pair,

- a set of permutations was chosen to satisfy the specifications of the interfaces of the row with some parameters randomly chosen (e.g. the positions of non-stationary points but still satisfying the requirements of the interfaces listed on the table like consecutive or not) and some other parameters uniquely selected (e.g. m and k); and
- an instance of the PUF was generated to meet the specification of the PUF type, with some specification parameters randomly chosen (e.g. gate delays) and others are uniquely chosen (e.g. the XOR size of 3 for XOR PUFs).

In the experiments, from each instance of interface-PUF pair, 10 million random CRPs were generated using the simulation software [20]. In each attack of a PUF-interface instance, the number of CRPs listed in Table II is the total

number of CRPs used for the training, validation and testing, the Success Rate column lists the percentage of attacks that produced testing accuracy of at least 80%, and the Avg. Acc. column list the average testing accuracy of all attacks for the row, including both successful and unsuccessful attacks. An attack of at least 80% accuracy is considered a success because attacks with accuracy below 80% cannot successfully impersonate a PUF with PUFs' noise usually below 5%, and 80% is a reasonably high standard for examining the security of an interface since choosing 95% will make a lot more interfaces look secure.

The results in Table II clearly show the effectiveness of the interface. As indicated by the decrease of success rate for interfaces with separated non-stationary points, the interfaced PUFs are secure against attacks when the number of non-stationary points m and the number of hops k become high.

The permutation interface is the most effective for XOR PUFs, as indicated by 0% attack success rate. Theorem 1 reveals that the order of the representing polynomial of an interfaced x-XOR PUF is x times higher than that of an interfaced APUF, and the experimental data support the analysis. The data also showed that interfaces with consecutive non-stationary points are weak against attacks (Rows $A_{8,0,3}$ and $A_{9,0,3}$), which is consistent with Theorem 1.

For all tested PUFs, experimental data show that the number m needed to attain 0% success rate is less than $n/3$, with $n = 64$ for our tested PUFs. Then, the discussion following Theorem 1 suggests that there are exponentially many interfaces that can make all three types of PUFs secure against neural network attacks, the most powerful ML attack method as of now according to studies [10], [12], [19].

IV. Conclusion

Many IoT devices are resource-constrained and demand security mechanisms implementable with low cost and operable with low energy. Leveraging integrated circuits' internal variability as hardware fingerprints, PUFs have the potential as underlying primitives for implementing lightweight security protocols. The arbiter PUF and its variants are highly lightweight but have succumbed to machine learning attacks. This paper introduces a challenge interface to improve security against machine learning attacks. With the interface, analytical and experimental studies have shown that the strength of interfaced PUFs against machine learning attacks is substantially improved. The proposed interface incur a low resource overhead, and hence can maintain the lightweightness of the PUFs. With low resource footprint and substantial improvement of resistance to machine learning attacks, the proposed interface render itself a fitting companion of PUFs for delivering security for resource-constrained IoT devices.

Acknowledgment

The research was supported in part by the National Science Foundation under grant No. 2103563. We thank anonymous reviewers whose comments/suggestions improved this paper.

References

[1] U. Rührmair and D. E. Holcomb, "Pufs at a glance," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

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

[3] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Controlled physical random functions," in *18th Annual Computer Security Applications Conference, 2002. Proceedings.* IEEE, 2002, pp. 149–160.

[4] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[5] 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.

[6] 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.

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

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

[9] 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)*. IEEE, 2018, pp. 49–56.

[10] 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, p. 1715, 2020.

[11] N. Wisiol, G. Becker, M. Margraf, T. Soroceanu, J. Tobisch, and B. Zengin, "Breaking the lightweight secure puf: Understanding the relation of input transformations and machine learning resistance," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2019, pp. 40–54.

[12] P. Santikellur, A. Bhattacharyay, and R. S. Chakraborty, "Deep learning based model building attacks on arbiter puf compositions." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 566, 2019.

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

[14] J. Ye, Y. Hu, and X. Li, "Rpuf: Physical unclonable function with randomized challenge to resist modeling attack," in *IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE, 2016, pp. 1–6.

[15] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe, "Puf-fsm: A controlled strong puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 1104–1108, 2017.

[16] C. Herder, L. Ren, M. van Dijk, M. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Transactions on Dependenable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2017.

[17] E. I. Vatajelu, G. D. Natale, M. S. Mispan, and B. Halak, "On the encryption of the challenge in physically unclonable functions," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2019, pp. 115–120.

[18] W. E, "A mathematical perspective of machine learning: Machine learning from the viewpoint of numerical analysis in high dimension," in *SIAM Conference on Mathematics of Data Science*, 2020.

[19] N. Wisiol, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural-network-based modeling attacks on xor arbiter pufs revisited." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 555, 2021.

[20] N. Wisiol, C. Gräbnitz, C. Mühl, B. Zengin, T. Soroceanu, N. Pirnay, and K. T. Mursi, *pypuf: Cryptanalysis of Physically Unclonable Functions*, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.3901410.