Machine Learning-based Vulnerability Study of Interpose PUFs as Security Primitives for IoT Networks

Bipana Thapaliya
Department of Computer Science
Texas Tech University
Lubbock, Texas, USA
bipana.thapaliya@ttu.edu

Khalid T. Mursi
Department of Cyber Security,
College of Computer Science and Engineering
University of Jeddah
Jeddah, Saudi Arabia
kmursi@uj.edu.sa

Yu Zhuang
Department of Computer Science
Texas Tech University
Lubbock, Texas, USA
yu.zhuang@ttu.edu

2021 IEEE International Conference on Networking, Architecture and Storage (NAS) | 978-1-7281-7744-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/NAS51552.2021.9605405

***Abstract*—Security is of importance for communication networks, and many network nodes, like sensors and IoT devices, are resource-constrained. Physical Unclonable Functions (PUFs) leverage physical variations of the integrated circuits to produce responses unique to individual circuits and have the potential for delivering security for low-cost networks. But before a PUF can be adopted for security applications, all security vulnerabilities must be discovered. Recently, a new PUF known as Interpose PUF (IPUF) was proposed, which was tested to be secure against reliability-based modeling attacks and machine learning attacks when the attacked IPUF is of small size. A recent study showed IPUFs succumbed to a divide-and-conquer attack, and the attack method requires the position of the interpose bit known to the attacker, a condition that can be easily obfuscated by using a random interpose position. Thus, large IPUFs may still remain secure against all known modeling attacks if the interpose position is unknown to attackers. In this paper, we present a new modeling attack method of IPUFs using multilayer neural networks, and the attack method requires no knowledge of the interpose position. Our attack was tested on simulated IPUFs and silicon IPUFs implemented on FPGAs, and the results showed that many IPUFs which were resilient against existing attacks cannot withstand our new attack method, revealing a new vulnerability of IPUFs by re-defining the boundary between secure and insecure regions in the IPUF parameter space.**

***Index Terms*—physical unclonable function, Interpose PUF, machine learning, neural network, FPGA**

I. Introduction

THE enormous quantity of communication-capable devices have created vast and diverse groups of networks and security are of critical importance for the communications among them. But many network nodes are resource-constrained, like sensors and IoT devices, and cannot accommodate conventional cryptographic protocols which are not lightweight, as pointed out by [6], [17].

Physical Unclonable Functions(PUFs) leverage small physical variations of the integrated circuits to produce responses unique to individual circuits and hence are not reproducible even by their manufacturers. In addition, being implementable with only thousands of transistors, PUFs incur very low fabrication costs and require very low operational power. Thus, PUFs are good candidates as hardware primitives for implementing security protocols for resource-constrained network nodes.

While physically unclonable, PUFs can be "cloned" by mathematical or machine learning models, where an attacking model can be built using a sufficient number of challenge-response pairs (CRPs) the attacker has accumulated by eavesdropping on communications between the PUF and its trusted partners. Thus, before a PUF design is adopted for security applications, all vulnerabilities of the PUF, including vulnerabilities to machine learning attacks, must be identified so that application developers can take application-level measures to prevent the exploitation of the vulnerabilities, or avoid PUFs whose vulnerabilities cannot be eliminated at the application level.

Recently, a new PUF design known as Interpose PUF (IPUF) [11] was proposed which consists of two XOR PUFs with the output of one XOR PUF interposed to one of the challenge bits of the second XOR PUF. In the study [11], IPUFs were tested to withstand the reliability-based modeling attack [5] as well as one of the most powerful machine learning methods for attacking PUFs. A later attack study [14] on multiple PUFs using neural network (NN) with the ReLU activation functions revealed that IPUFs consisting of small component XOR PUFs are vulnerable to NN-based attacks. A more recent attack method [16] was able to crack much large IPUFs, and it targets IPUF whose interpose bit is the middle bit of the second XOR PUF. While easily adaptable for IPUFs with the interpose bit at any other position, the attack method [16] requires the knowledge of the position of the interpose bit. Thus, IPUFs whose interpose bit positions are randomly chosen remain secure against the machine learning attack of [16].

In this paper, we present a method for attacking IPUFs without requiring knowledge of the position of the interpose

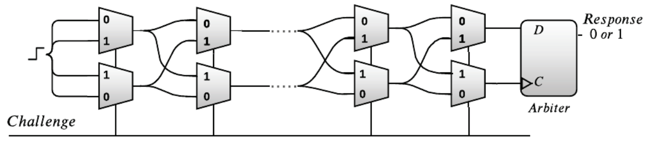

Fig. 1: An arbiter PUF

bit. Our attack method employs a multilayer neural network with the hyperbolic tangent ($tanh$) as the activation function for internal hidden layers. There have been many PUF attack studies that used neural networks, including [1]–[4], [7], [9], [10], [14]. The neural networks used in most of these studies employ the ReLU activation function for internal hidden layers. However, in a recent study [10], the hyperbolic tangent function was used as the activation function for modeling XOR PUFs and led to substantially better learning power in tests carried out in the study [10], contrary to the popular belief. In this paper, using a neural network with the $tanh$ activation function, our attack method which does not require the knowledge of the interpose bit position has also turned out to be highly powerful in modeling the behavior of the IPUF (details in Sec. 3). Tests on both simulated and FPGA-implemented PUFs have shown that our attack method can break large IPUFs including IPUFs consisting of two 8-XOR PUFs as well as IPUFs consisting of an Arbiter PUF and a 9-XOR PUF, revealing a vulnerability of IPUFs not discovered earlier.

The remaining of the paper is organized as follows. Section 2 gives a brief description of PUFs and Interpose PUFs. Section 3 contains the neural network-based attack method, and Section 4 presents the experimental study. Section 5 gives the conclusion.

II. PUFs and Interpose PUFs

Physical Unclonable Functions (PUFs) are increasingly recognized as potential candidates for implementing security protocols for resource-constrained communication networks. The reason behind that recognition has tow-folds. First, traditional cryptographic protocols require secret keys stored in secure non-volatile memories. But many devices in communication networks are within close physical distances and hence any data stored on such devices, including secret keys, are susceptible to invasive or side-channel attacks which are usually more effective within close distances. Second, traditional cryptographic protocols are not lightweight enough as pointed out by studies [6], [17], leaving them unsuitable for many resource-constrained communication devices.

PUFs utilize small variations in integrated circuits (ICs) to produce responses unique for individual circuits. These variations are undesirable for conventional ICs, but they are hardware fingerprints that can be utilized to prevent physical cloning even by the IC manufacturers. With these variations, PUFs do not have to store secret keys but make use of the variations to produce circuit-specific responses, rendering them good candidates as hardware primitives for security protocols.

An interpose PUF (IPUF) consists of two XOR PUFs, with the output of the first XOR PUF interposed as a challenge bit into the second XOR PUFs. To make discussion in later sections easier, in this section, we briefly describe the circuit structures of the XOR PUF and the Arbiter PUF.

A. The Arbiter PUF

An n-stage Arbiter PUF, or APUF for short, consists of n pairs of 2-to-1 multiplexers (see Figure 1), where the two multiplexers at the same stage receive the same challenge bit. The two signals pass through gates of all stages of the paths, and slightly different delays are incurred when signals pass through different gates. An arbiter, usually implemented by a D-latch, determines the final output depending on which signal arrives first. For instance, if the top path arrives first, the output is 1, otherwise is 0. The challenge bit values at all stages determine the paths, and consequently the delays, of the signals, leading to a total of 2^n possible paths.

The response and the challenge of an n-stage arbiter PUF satisfy the additive delay model [8], which stipulates that the elapsed times of the two signals arriving at the arbiter are the summations of the delays incurred at all stages of the PUF. Based on the additive delay model, the response of an arbiter satisfies

$$r = sgn\Big(\sum_{i=1}^{n+1} \phi(i)u(i)\Big), \tag{1}$$

where ϕ's are transformed challenge [8] given by

$$\phi(i) = (2c_i - 1)\,(2c_{i+1} - 1)\cdots(2c_n - 1) \tag{2}$$

with c_i being the challenge bit at stage i, u's being parameters quantifying gate delays at different stages, $sgn(\cdot)$ the sign function, and r the response of the arbiter PUF..

The challenge transform (2) turns the relationship between challenge and the response of an arbiter PUF into a linearly separable classification problem, where the hyperplane represented by the equation $\sum_{i=1}^{n+1} \phi(i)u(i) = 0$, resulting from setting to 0 the term inside the $sgn(\cdot)$ function in Eq. (1), is the separating hyperplane in the n-dimensional space of transformed challenges that partition the space into two regions, with 1 as the responses of all challenges in one region and 0 the responses for challenges in the other region. For machine learning attack method that has already obtained adequate number of CRPs through eavesdropping of communications or other means, the attack method uses the obtained CRPs to train a machine learning model so that the trained model can accurately predict the responses of other challenges.

B. The XOR PUF

The linearly separable classification model eq. (1) describing the relationship between the response r and transformed challenge ϕ makes arbiter PUFs easy victims of machine learning attacks [13]. XOR PUFs were introduced to add non-linearity to the relationship between challenge and response.

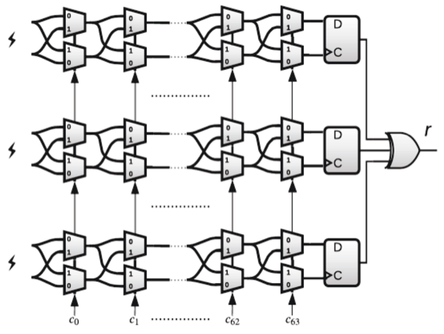

Fig. 2: Illustration of an XOR PUF

An XOR PUF consists of k multiple APUFs. As illustrated in Fig. 2, the k-XOR arbiter PUF uses k arbiter PUFs as components, where all of the k arbiter PUFs use the same challenge c as the challenge input. The responses of all individual arbiter PUFs are XORed together to produce the final response r for the corresponding input challenge.

With increased nonlinearity for the response as a function of the challenge, XOR PUFs are more resistant to machine learning attacks than APUFs. But the reliability-based modeling attack [5] can still easily crack XOR PUFs, where a reliability-based attack applies the same challenges repeatedly to the XOR PUF under attack and the responses can help the machine learning attack method identify where the separating planes of classification problem that represents the XOR PUF. The reliability-based attack can be defended when a PUF-embedded device employs the lockdown mutual authentication protocol [17] to prevent attackers from applying the same challenges repeatedly.

In addition to vulnerability to the reliability-based attack, XOR PUF has also succumbed to a conventional machine learning attack as showed by a recent study [10], which employed a multilayer neural network with the hyperbolic tangent function as the activation function for internal hidden layers. The machine learning method in [10] was successful in attacking XOR PUFs up to 9-XOR PUFs.

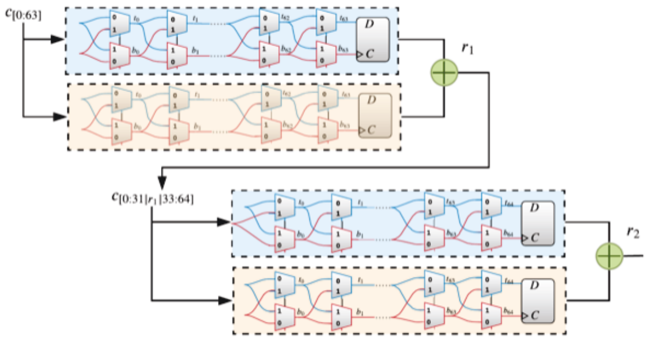

Fig. 3: Illustration of (2,2)-IPUF. The output of the first 2-XOR PUF interposed as a challenge bit into the second XOR PUF.

C. The Interpose PUF

An (x, y)-IPUF consists of an x-XOR PUF and a y-XOR PUF, where the output of the x-XOR PUF is interposed as a challenge bit into the y-XOR PUF. With the interpose bit added as a challenge bit to the second XOR PUF, the y-XOR PUF has exactly one more stage than the x-XOR PUF. See Fig. 3 for an illustration of a (2,2)-IPUF.

The IPUF was shown [11] to be secure against both the reliability-based modeling attack [5] and a successful machine learning method for attacking PUFs. A later attack study [14] revealed that IPUFs consisting of small component XOR PUFs are vulnerable to the attack method that uses neural networks with the ReLU activation functions. A more recent attack method [16] was able to crack much large IPUFs, including IPUFs consisting of a 1-XOR PUF and a 9-XOR PUF as well as IPUFs consisting two 8-XOR PUFs.

III. The Machine Learning Attack Method

One of the early studies on modeling or machine learning attacks on PUFs was reported in a 2010 paper [13], in which Logistic Regression with Resilient Propagation (RProp LR) was shown to be a fast and broadly applicable attack method among methods reported in [13]. The LR used alone to attack IPUFs was not successful, as shown in the work [11] where the IPUF was introduced.

Recently, a new attack [16] that employs LR in a divide-and-conquer method for attacking IPUFs, resulting in successful attacks for large IPUFs including $(8, 8)$-IPUFs and $(1, 9)$-IPUFs. But the required CRPs are very large, 300 million for attacking $(8, 8)$-IPUFs and 750 million for attacking $(1, 9)$-IPUFs, respectively. In addition, the attack method in [16] targets IPUF whose interpose bit is the middle bit of the second XOR PUF. While easily adaptable for IPUF with the interpose bit at any other position, the attack method [16] requires the knowledge of the position of the interpose bit. Thus, IPUFs whose interpose bit positions are randomly chosen remain secure to the machine learning attack of [16].

In 2012, a two-layer neural network [7] was used to attack 2-XOR PUFs, with four neurons for a hidden layer and one neuron for the other layer. In 2018 [4], a neural network with three ReLU-based hidden layers and one sigmoid output layer was used to attack XOR PUFs, and successfully attacked up to 8-XOR PUFs with an efficiency much faster than earlier attack methods on similar XOR PUFs. But when applied to IPUFs, ReLU-based neural networks can break only small IPUFs as supported by a 2019 study [14], and our own trials showed that ReLU-based neural networks were not able to break IPUFs consisting of two k-XOR PUFs with $k > 4$ in our trials.

In a recent study [10], a multilayer neural network with the hyperbolic tangent function $(tanh)$ as the activation function was employed to attack XOR PUFs, showing substantially higher learning power than ReLU-based neural networks. Though it is believed that $tanh$ sometimes suffers from vanishing gradient problem (which made it less popular after

TABLE I: Specification of the Neural Network

Hyperparameters	Description
No. of Neurons on hidden layers	Varies with IPUF size
Hidden layer activation function	Hyperbolic tangent
Output layer activation function	Sigmoid
Optimizer	Adam
Learning rate	Adaptive
Bias Initializer	Zeros
Weight Initializer	Xavier Normal
Loss function	Binary Cross Entropy

the introduction of ReLU), it seems that during the training of the networks for attacking PUFs, $tanh$ works substantially better than ReLU. And the vanishing gradient problem has not appeared in the PUF attack study [10] as well as in this study. Additionally, as mentioned by Mursi et. al. [10], using $tanh$ as the activation function also led to requiring fewer network layers than those with ReLU as the activation function. For these reasons and for the reason that an IPUF consists of two XOR PUFs, in this paper, we have decided to try the $tanh$-based neural network.

With the activation function chosen, the neural network we use for modeling an n-stage (x, y)-IPUF has the following structure of layers (also see Fig. 4 for illustration):

- The input layer of (n) challenge bits $c_1, c_2, \cdots, c_n$,
- the challenge transform layer that transforms challenge bits from input layer to ϕ's according to (2),
- three layers of the densely connected neural network whose weights for all neurons are to be trained, and
- the single-bit output layer to produce the output that models the response of the IPUF.

Other parameters of the neural network are listed in Table I.

IV. Experimental Studies

A. The Experiment Set-up

1) Simulated CRPs: Firstly, for generating the simulated data, we used the pypuf library published by Wisiol et. al. [15] in prior work on IPUF attacks [16] so that we can have a fair comparison of the attacking results. The software can be found at: pypuf.rtfd.io. We implemented the framework in Python and generated the data for the attack. Firstly, we generated 5 samples of data using different PUF instances for each IPUF using the interpose position as 32. Secondly, we generated 3 samples for each IPUF type using different PUF instances by selecting interpose bit below 32, where the interpose bit is randomly chosen. Similarly by randomly selecting interpose bit above 32, thus leading to a total of 11 samples for each IPUF type. The simulation is based on a linear additive delay model, and the arbiter delays are drawn independently using a Gaussian distribution. We generated a different number of CRPs for each IPUF type for 64-bit instance, the number of CRPs were increased depending upon the complexity of IPUF, summing up to 230 million CRPs for all our experiments, with x ranging from 1 to 7, and y ranging from 5 to 7 in a typical (x, y)-IPUF.

2) Silicon CRPs: To study the security vulnerability of IPUFs on real data, we programmed IPUFs on Artix®-7 FPGAs using the Xilinx Vivado design suite which is a software suite produced by Xilinx Inc. for synthesis and analysis of HDL designs. For our experiment, Vivado HL Webpack edition 2018.3 was used along with Xilinx SDK. VHSIC Hardware Description Language(VHDL) was used to build IPUF designs. Each IPUF, consisting of upper XOR PUF and lower XOR PUF, was placed vertically on the chip using Tool Command Language (TCL). Firstly, we placed the MUXes for both upper and lower XOR PUFs. And while generating the CRPs, we first pass the challenges only to the upper XOR PUF and wait for its response before the signal races through the second XOR PUF. Once the response from the upper XOR PUF is received, we interpose the response bit in the middle of the challenges of the second XOR PUF. The response received from the second XOR PUF was recorded as the final response for our CRPs.

To speed-up the CRPs transformation, AXI Universal Asynchronous Receiver Transmitter (UART) with a baud rate of 230,400 bits/second, was used. Thus, the time required for generating a million CRPs was only 10 minutes. Finally, the Xilinx SDK was utilized to program the input/output workflow behavior of the CRPs generation from the chips. The experiments were done on three 28-nm test chips. We generated the CRPs from these 3 different devices. The number of CRPs was different for each IPUF type depending upon the complexity of the IPUF, for example, 5M for $(5, 5)$-IPUF and 15M for $(1, 9)$-IPUF, sufficient enough to have multiple samples for each of them leading to a total of 15 samples using 3 different devices for each IPUF type. The number of CRPs for all the IPUF types and all samples collectively led us to have 210 million CRPs, in total. The silicon CRPs were generated at an ambient temperature of approximately 23°C, and core voltage was set to 1.0V using the built-in chips resistor. Even though the incurred noise during the generation process affects the model training, we were able to break the IPUFs on silicon CRPs with high accuracies as shown in the next section.

3) The Machine Learning Attack Tests: The proposed deep learning model was written using Python 3.7 with PyTorch [12] machine learning library. The experiments were carried out on a Windows machine with 16 GB of main memory with 1.99 GHz, Intel Core i7 processor. Each of the experiments was conducted without any external parallel execution from our side.

For each of the reported attacks, we dedicate 10K CRPs as a test set and 5K CRPs as a validation set. The number of CRPs for the test set and validation set remains the same for all the experiments. The modeling accuracies mentioned in the next section is the prediction accuracy evaluated using this test set and is completely fresh data that the model has never seen before. Please note that the CRPs mentioned in

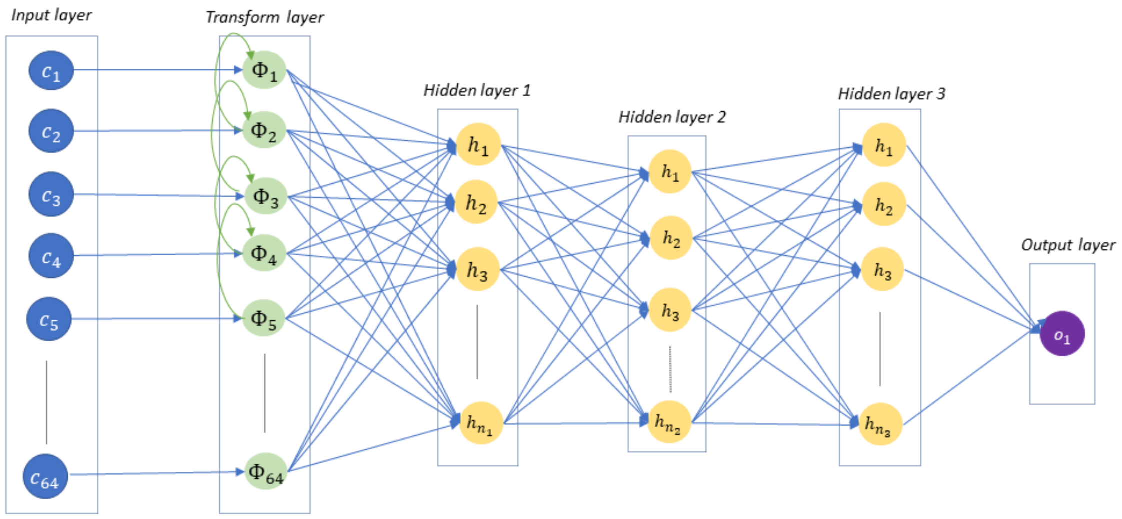

Fig. 4: The deep neural network architecture used for the proposed method. The no. of neurons in each hidden layer varies from 16 to 128 depending upon the complexity of the IPUF.

Table III does not include the test set and validation set CRPs and are solely used for training. The training was performed by implementing the method discussed in the previous section with the specifications as mentioned in table I and architecture as given in fig. 4.

B. The Experiment Results

We have listed our experimental results for the simulated CRPs, CRPs generated by using simulator from Wisiol et. al. [16] under table II. The x and y under the column "(x, y)-IPUF" represents IPUF with x number of components for the upper or the first XOR PUF layer and y number of components for the lower or the second XOR PUF layer in the IPUF. "Avg. Pr. Acc" is the average prediction accuracy from all the samples. We generated a different number of CRPs depending upon the complexity of the IPUF, starting from 500K and gradually increasing it. The minimum number of CRPs that resulted maximum accuracy and minimum loss was recorded to generate other samples.

The column "Method" represents the comparison of our method against the others. As mentioned in section 1, there are only two prior works on the IPUF attack until the time this paper was published: one from Santikellur et. al. [14] and the other from Wisiol et. al. [16]. The work by Santikellur et. al. includes the result only for IPUF up to components x=4 and y=4. However, in this work, we focus on larger IPUFs since smaller IPUFs are easier to attack. Thus, we compare our method with Wisiol et. al. [16] only and is represented as method "A". Our method is represented as method "B". It is important to note that the divide-and-conquer method used in Wisiol et. al. approach (method "A") was implemented based on the assumption that the attacker already knows the interposed bit position, whereas our model has no such assumption. Therefore, we tested our model using different interpose positions. All the results stated from method "A" in table II uses interpose bit position as 32, which is known to the attacker. For the proposed method, different interpose positions are used; 5 samples generated by using interpose position as 32, 3 samples by using randomly selected interpose position below 32, and 3 samples using randomly selected interpose position above 32, all unknown to the attacker, and the average of those 11 samples is reported.

The CRPs used for testing the model is not included in the training CRPs listed under table II, so is with method "A". Method "A" employs parallel computing using 4 cores for the larger experiments ($(7, 7)$-IPUF and above), therefore the training times of the method was multiplied by 4 so that the timing becomes comparable with our method. We do not include the column for success rate as we were able to receive a 100% success rate for all our samples; success is defined as final prediction accuracy as 85% and above. The average prediction accuracy (Avg. Pr. Acc.) for method "A" for the IPUFs $(5, 5)$ to $(7, 7)$ is left blank because the accuracy for those IPUFs in the paper [16] is the average accuracy only of the successful test samples, and since the success rate is not 100%, there is no enough information to calculate the average accuracy of all samples (successful and unsuccessful).

As the value of y in an (x, y)-IPUF increases, we observed that the required training CRPs and the training time also increase, along with the required number of hidden units in each layer. We also altered the batch size for the experiment as the number of CRPs increases. The stated results clearly indicate the efficiency of our model as we achieve 100% success rates by using no more than 6 million CRPs even for higher IPUFs like $(1, 7)$-IPUF and $(7, 7)$-IPUF. We were able to break the IPUFs with CRP sizes less than those needed by method "A" except for $(1, 5)$-IPUF, and our method incurs much shorter training times. For instance, the time required for breaking $(7, 7)$-IPUF was at maximum a 3 hours compared to 68 hours required by method "A". The results also indicate that our model is suitable for IPUFs with any interpose position as we successfully attacked the given IPUFs for randomly chosen interpose positions below or above 32.

Similarly, table III shows experimental results for the silicon CRPs, CRPs generated by PUFs implemented on FPGAs for different component IPUFs using different CRPs. An additional column named "FPGA" has been added to this table to denote the 3 FPGA devices that were used to test our model. We also list the best prediction accuracy represented as "Best Pr. Acc." and the worst prediction accuracy denoted as "Worst Pr. Acc." achieved out of all the samples. Since we are the first ones to try the deep learning-based approach on the Silicon CRPs for IPUF, there is no record for the comparison. Therefore, we state only our results. The result stated for each device in the table is calculated out of the 5 samples generated from that particular FPGA device. In our experimental results for silicon data, we achieve above 90% accuracy for all the sizes and complexities of the IPUF. Since the silicon data has noise, it was difficult to cross 95% training accuracy for the (x, y)-IPUF with $y > 5$. However, we did not increase the number of CRPs further as it was evident enough that it was unsafe given the prediction accuracy had reached 90% or above in all the cases for silicon CRPs.

The observed accuracy and time clearly state the efficiency of the proposed model. The CRPs and the training time received for even the highly complex $(1, 9)$-IPUF was at a maximum of 4 hours with only 3 million CRPs using our method. Therefore, the results obtained by the proposed method using very less CRPs and training time show that even larger IPUFs are not secure enough to withstand machine learning attacks.

V. Conclusion

The advent of the IoT era calls for new security mechanisms to address the peculiar challenges posed by IoT networks. Physical Unclonable Functions offer a potential solution for many IoT devices' resource-constrainedness and physical proximity to the crowds. But any PUF design must be thoroughly examined to identify all its vulnerabilities before being used in real applications. Interpose PUF was believed to be a highly secure PUF design that can withstand both reliability-based modeling attacks and machine learning attacks. In this study, we have investigated attacks on IPUFs based on multilayer neural networks which use the hyperbolic tangent function as the activation function for hidden layers, and the attack method has shown to have cracked large IPUFs with random interpose bit position, which existing attack methods failed to break, revealing a new security vulnerability of the Interpose PUF to network security application developers.

VI. Acknowledgment

The research was supported in part by the National Science Foundation under Grant No. CNS-1526055 and Grant No. OAC-2103563. We would also like to thank the anonymous reviewers whose comments/suggestions helped improve this paper.

References

[1] Meznah A Alamro, Khalid T Mursi, Yu Zhuang, Ahmad O Aseeri, and Mohammed Saeed Alkatheiri. Robustness and unpredictability for double arbiter pufs on silicon data: Performance evaluation and modeling accuracy. *Electronics*, 9(5):870, 2020.

[2] Meznah A Alamro, Yu Zhuang, Ahmad O Aseeri, and Mohammed Saeed Alkatheiri. Examination of double arbiter pufs on security against machine learning attacks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3165–3171. IEEE, 2019.

[3] Mohammed Saeed Alkatheiri and Yu Zhuang. Towards fast and accurate machine learning attacks of feed-forward arbiter pufs. In *2017 IEEE Conference on Dependable and Secure Computing*, pages 181–187. IEEE, 2017.

[4] Ahmad O Aseeri, Yu Zhuang, and Mohammed Saeed 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)*, pages 49–56. IEEE, 2018.

[5] Georg T Becker. The gap between promise and reality: On the insecurity of xor arbiter pufs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 535–555. Springer, 2015.

[6] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[7] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability. In *2012 IEEE international workshop on Information forensics and security (WIFS)*, pages 37–42. IEEE, 2012.

[8] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas 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)*, pages 176–179. IEEE, 2004.

[9] Khalid T Mursi, Bipana Thapaliya, and Yu Zhuang. A hybrid-optimizer-enhanced neural network method for the security vulnerability study of multiplexer arbiter pufs. In *Journal of Physics: Conference Series*, volume 1729, page 012010. IOP Publishing, 2021.

[10] Khalid T Mursi, Bipana Thapaliya, Yu Zhuang, Ahmad O Aseeri, and Mohammed Saeed Alkatheiri. A fast deep learning method for security vulnerability study of xor pufs. *Electronics*, 9(10):1715, 2020.

[11] Phuong Ha Nguyen, Durga Prasad Sahoo, Chenglu Jin, Kaleel Mahmood, Ulrich Rührmair, and Marten van Dijk. The interpose puf: Secure puf design against state-of-the-art machine learning attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 243–290, 2019.

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[13] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249, 2010.

[14] Pranesh Santikellur, Aritra Bhattacharyay, and Rajat Subhra Chakraborty. Deep learning based model building attacks on arbiter puf compositions. *IACR Cryptol. ePrint Arch.*, 2019:566, 2019.

[15] Nils Wisiol, Christoph Gräbnitz, Christopher Mühl, Benjamin Zengin, Tudor Soroceanu, Niklas Pirnay, and Khalid T. Mursi. pypuf: Cryptanalysis of Physically Unclonable Functions, 2021.

[16] Nils Wisiol, Christopher Mühl, Niklas Pirnay, Phuong Ha Nguyen, Marian Margraf, Jean-Pierre Seifert, Marten van Dijk, and Ulrich Rührmair. Splitting the interpose puf: A novel modeling attack strategy. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 97–120, 2020.

[17] Meng-Day Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A lockdown technique to prevent machine learning on pufs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, 2016.

TABLE II: Prediction accuracy and training time of our attack method using simulated CRPs

Stages	(x,y)-IPUF	Method	CRPs	Avg. Pr. Acc.	Training time
64-bit	1,5	A	500K	>95%	9.14 m
		B	1M	99%	2 m to 28 m
	1,6	A	2M	>95%	1.48 h
		B	2M	96%	12 m to 42 m
	1,7	A	20M	>95%	20.07 h
		B	6M	96%	45 m to 2 h 22 m
	5,5	A	1M	-	14.59 m
		B	1M	88%	10 m to 1 h 5 m
	6,6	A	5M	-	2.50 h
		B	4M	88%	1 h 8 m to 2 h 25 m
	7,7	A	40M	-	68.84 h
		B	6M	87%	1 h 50 m to 3 h

”A” refers to the method by Wisiol et. al. [16] which requires interpose position known to the attacker, and ”B” refers to the proposed method which allows random interpose position unknown to the attacker.
Avg. Pr. Acc. not available for Method “A” for (5,5), (6,6), and (7,7) IPUFs.

TABLE III: Prediction accuracy and training time of our attack method using silicon CRPs

Stages	**(x,y)-IPUF**	**FPGA**	**CRPs**	**Best Pr. Acc.**	**Avg. Pr. Acc.**	**Worst Pr. Acc.**	**Training time**
64-bit	5,5	Dev1	800K	98%	97%	96%	20 m to 40 m
		Dev2	800K	97%	97%	97%	15 m to 45 m
		Dev3	800K	98%	97%	97%	15 m to 40 m
	6,6	Dev1	1M	96%	95%	95%	39 m to 1 hr 25 m
		Dev2	1M	96%	95%	95%	35 m to 1 h 15 m
		Dev3	1M	97%	96%	96%	30 m to 1 h 30 m
	7,7	Dev1	2M	95%	95%	95%	35 m to 1 h 20 m
		Dev2	2M	96%	95%	95%	48 m to 1 h 22 m
		Dev3	2M	96%	95%	95%	40 m to 1 h 30 m
	8,8	Dev1	2M	94%	94%	94%	41 m to 1 h 10 m
		Dev2	2M	94%	94%	94%	43 m to 56 m
		Dev3	2M	95%	94%	94%	45 m to 1 h 20 m
	1,6	Dev1	1M	95%	95%	95%	24 m to 48 m
		Dev2	1M	95%	95%	95%	45 m to 1 h
		Dev3	1M	95%	94%	94%	25 m to 55 m
	1,7	Dev1	2M	95%	94%	94%	34 m to 4 h
		Dev2	2M	95%	94%	93%	40 m to 2 h 30 m
		Dev3	2M	95%	94%	94%	45 m to 3 h
	1,8	Dev1	2M	94%	94%	94%	31 m to 1 h 45 m
		Dev2	2M	94%	93%	93%	44 m to 2 h
		Dev3	2M	94%	93%	93%	40 m to 2 h 30 m
	1,9	Dev1	3M	93%	93%	93%	1 h 15 m to 7 h
		Dev2	3M	93%	92%	92%	55 m to 2 h 30 m
		Dev3	3M	93%	92%	92%	1 h 30 m to 4 h