Towards Fast and Accurate Machine Learning Attacks of Feed-Forward Arbiter PUFs

Mohammed Saeed Alkatheiri*†, Yu Zhuang ‡
*‡Department of Computer Science, Texas Tech University, Lubbock, Texas, USA
Email: *mohammed.alkatheiri@ttu.edu, ‡ yu.zhuang@ttu.edu
†Faculty of Computing and Information Technology, University of Jeddah, Jeddah, Saudi Arabia
Email:†msalkatheri@uj.edu.sa

***Abstract*—Utilizing integrated circuits' manufacturing variations to produce responses unique for individual devises, physical unclonable functions (PUFs) are not reproducible even by PUF device manufacturers. However, many PUFs have been reported to be "mathematically reproducible" by machine learning-based modeling methods. The feed-forward arbiter PUFs are among the PUFs which have showed strength [1], [2] against machine learning modeling unless large computation time is used in machine learning process and the feed-forward loops are of a special type. In this paper, we develop a signal delay model for the feed-forward arbiter PUFs, through which efficient and accurate machine learning of the PUF's essential features is made possible. Experimental results show that the new model has led to high accuracy and high efficiency for the prediction of the responses of the PUFs with any type of feed-forward loops, and the high prediction accuracy was measured in terms of average prediction rate over all tested all cases. The high efficiency and high accuracy prediction of responses reported in this paper has revealed a weakness of the feed-forward arbiter PUFs that can be potentially utilized by response-prediction-based malicious software.**

***Keywords*—Arbiter PUF, Feed-Forward Arbiter PUF, Machine Learning Attack, Multilayer Neural Network.**

I. Introduction

Authentication is essential for securing the transactions between electronic devices. The traditional approach of using secret keys stored in non-volatile memory for implementing authentication is, however, subject to invasive attacks, which can read out the stored secret keys. Physical unclonable functions (PUFs) utilize the manufacturing variations in integrated circuits to produce responses unique for individual PUF devises, and are not reproducible even by the PUF manufacturer, or anyone else, who has obtained the PUF circuit design, hence possessing great potential as primitives for implementing secure mechanisms.

While physically unreproducible, PUFs are reported [1] to be "mathematically reproducible" by machine learning-based modeling methods which can predict the responses of PUFs accurately. The mathematical reproducibility allows attackers to develop software to mimic the behavior of PUF-embedded devices for counterfeit products. An important component of work in security is to discover all possible insecure risks. Such information on security vulnerabilities will be useful for PUFs design researchers for developing new PUFs to overcome these existing vulnerabilities, as well as for PUF-embedded secure application developers who can avoid some PUFs' particular vulnerabilities the application cannot eliminate at software level, while adopt other PUFs with different security risks the application can take care of with software techniques.

Machine learning-based response predictions were proposed in [3], [4]. The first major comprehensive work was carried out by Rhrmair, et al. [1], in which they developed models for signal delays on the PUF circuit paths, with parameters in the signal delay models to be determined. Machine learning methods were used to find values for the parameters in the signal delay models, and Support Vector Machines (SVMs), Evolution Strategies (ES), and their in-house logistic regression (LR) incorporated with resilient propagation (RProp) are the machine learning methods employed in [1]. The PUFs studied in the work [1] include the Arbiter PUF, the XOR Arbiter PUF, the Lightweight Secure Arbiter PUF, and the Feed-forward Arbiter PUF. Based on a linear additive model developed in earlier work [3]–[5] for Arbiter PUFs, Rhrmair et al. [1] predicted the responses of 64-bit and 128-bit Arbiter PUFs using SVM, ES and RProp-LR, and prediction accuracy over 95% was attained by the RProp-LR method, all in less than 2.5 seconds. Then the linear additive model for Arbiter PUFs was utilized for developing models for the XOR Arbiter PUF, the Lightweight Secure Arbiter PUF, and the Feed-forward Arbiter PUF, and the best machine learning results were obtained with prediction accuracy in upper 90% and training times for the machine learning processing vary significantly with different complexities of the PUFs. For XOR Arbiter PUFs and Lightweight Secure Arbiter PUFs with five or more XORs, the reported training times [1] are over two hours with some PUFs taking substantially longer times, and the training times for Feed-forward Arbiter PUFs with six or more feed-forward loops are well over 20 hours.

After the seminal work of Rhrmair, et al. [1], there have been many successful studies on PUF modeling, including the work of Hospodar et al. [6], the work of Tobisch and Becker [7], and the work of Xu et al. [8]. In [7], Tobisch and Becker developed parallel code based on the RProp-LR machine method with code optimized in speed and memory usage, and applied it to the same model for XOR arbiter PUFs Rhrmair et al. used in [1], and reduced machine learning times in training for 6-XOR 64-bit Arbiter PUFs and 5-XOR 128-bit Arbiter PUFs to minutes, and reduced machine learning

978-1-5090-5569-2/17/$31.00 2017 IEEE

times to about 1 and 6.5 hours, respectively, for 7-XOR 64-bit Arbiter PUFs and 6-XOR 128-bit Arbiter PUFs. In the work [6], Hospodar, Maes, and Verbauwhede used SVM and Artificial Neural Network (ANN) for modeling Arbiter PUFs and XOR Arbiter PUFs implemented in CMOS. For the XOR Arbiter PUFs studied in [6], a two-layered ANN method was used with four neurons and one neuron respectively at the two ANN layers and achieved near 90% prediction accuracy. Reported in [8], Xu et al. developed a model for the difference between gate pull-up and pull-down strengths for Bistable Ring (BR) PUFs and for the twisted BR PUFs introduced by Schuster and Hesselbarth [9], and used SVM to predict the responses of BR PUFs and XOR-ed BR PUFs.

Feed-forward arbiter PUFs (FF APUFs) are among the groups of PUFs which have showed their strength [1], [2] against machine learning modeling unless large computation time is used for machine learning process. The study reported in [1] showed that it took over 27 hours to machine learn 64-bit 6-loop FF APUFs, and the more recent study reported in [2] show that it took over 3 hours to machine learn 128-bit 6-loop FF APUFs. To the best our knowledge, among all machine learning-based mathematical clonability studies, feed-forward Arbiter PUFs, lightweight secure PUFs, and XOR Arbiter PUFs (with large number of XORs) are the only groups of PUFs which have exhibited high resistance against machine learning attacks, unless long computation time is used for the machine learning process. Thus, any of the three groups of PUFs are good candidates for potential security applications as secure hardware primitives.

We choose to study feed-forward arbiter PUFs in this paper since among the three groups of attack-resistant PUFs mentioned above, FF APUFs have the lowest circuit complexity in terms of number of transistors needed for implementation, with an n-stage FF APUF consisting of about one-sixth of the transistors of an n-stage 6-XOR PUF and much less compared with an n-stage lightweight secure PUF. Thus, for resource-constraint secure applications, FF APUFs are potentially of higher value, and hence are more likely to be adopted for implementing various secure applications. It is thus of importance to investigate if there is any undiscovered vulnerability of a PUF of potentially high adoptability.

In this paper, based on the linear additive model [3] for Arbiter PUFs, we develop a new model for feed-forward Arbiter PUFs, which enables fast and accurate modeling of FF APUFs using machine learning methods. The new model overcomes a difficulty that the existing model possesses for machine learning in terms of the differentiability with respect to essential PUF features that affect the PUF responses.

We carried out experimental studies to examine the new model by applying three-layered neural network to both the existing model and our new model. While the neural network methods converge in only a few seconds to a few minutes for both the existing model and our model, the response prediction accuracy of the existing modeling is fairly low and our modeling produced substantial higher prediction accuracy, providing strong support for our new model.

The rest of the paper is organized as follows. Next section describes the proposed modeling attack on Feed-Forward Arbiter PUFs. Section III gives a description of Machine Learning algorithm that used for breaking FF Arbiter PUFs. The experimental result and discussion are given in Section IV. Finally, we conclude the paper in Section V.

II. The Model for Feed-Forward Arbiter PUF

Our model for the signal delay of Feed-Forward Arbiter PUFs is based on the linear additive model [3] for the arbiter PUF (see Fig. 1). The response r of an n-stage Arbiter PUF with challenge bits $(c_1, c_2, \ldots, c_n)$ can be modeled by

$$r = sgn(\phi(1)\omega(1) + \ldots + \phi(n)\omega(n) + v(n)) \qquad (1)$$

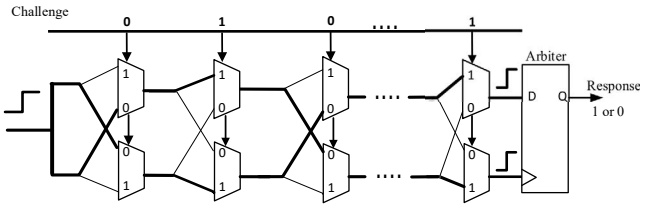


Fig. 1. Arbiter PUF.

where $\phi(i) = (2c_i - 1)(2c_{i+1} - 1)\ldots(2c_n - 1)$, $v(n)$ and $\omega(i)$'s are parameters quantifying the difference between signal delays of the two paths at the i-th stage, and $sgn(\cdot)$ is the sign function.

A machine learning-based "mathematical cloning" of the arbiter PUF is a process with which values of the feature vector W, i.e. values of feature variables $w(i)$'s are estimated by training the delay model (1) using a machine learning method on a set of challenge-response pairs. After the parameters in the vector W are accurately estimated, the response of the arbiter PUF to any given challenge $(c_1, c_2, \ldots, c_n)$ can be produced. The term inside the $sgn(\cdot)$ function is linear with respect to the PUF feature variables $w(i)$'s, and a "mathematical cloning" is hence to identify the hyperplane that separates the high-dimensional space into two subspaces with $r = -1$ in one subspace and $r = 1$ in the other subspace. Experimental results in [1] and [2] show that 64-bit arbiter PUFs can be broken under 0.6 second of training time and 128-bit arbiter PUFs can be broken under 2.5 seconds.

A feed-forward arbiter PUF resembles an arbiter PUF but with some of the challenge input bits receiving the outputs of feed-forward arbiters, and each of the feed-forward arbiters takes two output bits of an earlier stage of the arbiter PUF. Fig. 2 is an illustration of a feed-forward arbiter PUF with one feed-forward loop.

For a one-loop feed-forward arbiter PUF with n stages of multiplexers, if the loop starts at stage i_1 and ends at stage i_2, then the response of the feed-forward arbiter PUF can be modeled by

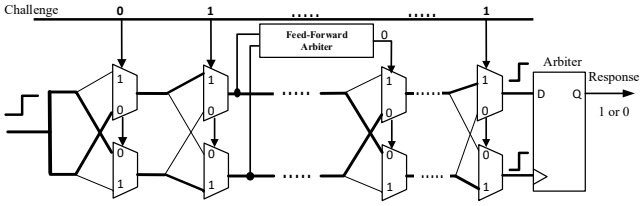


Fig. 2. A Feed-Forward Arbiter PUF with one loop.

$$r = sgn(v(n) + \sum_{i \neq i2} \phi(i)\omega(i) + \phi(i_2)\omega(i_2)) \quad (2)$$

where

$$\begin{aligned}
&\phi(n) = 2c_n - 1 \text{ if } i_2 \neq n,\\
&\phi(i) = (2c_i - 1)\phi(i+1) \text{ for } i \neq i_2,\\
&\phi(i_2) = \text{sgn}\left(v(i_1+1) + \sum_{i=1}^{i_1} \phi_{ff}(i)\omega(i)\right), \quad (3)\\
&\phi_{ff}(i_2) = \phi(i_2),\\
&\phi_{ff}(i) = (2c_i - 1)\phi_{ff}(i+1) \text{ for } i < i_2.
\end{aligned}$$

The only work known to us on machine learning-based mathematical cloning of feed-forward arbiter PUF was reported in [1] and a later journal version [2]. Though there is no model explicitly given in [1] or [2], we believe model (2) is the same model used in [1], [2] since it is stated in both [1] and [2] that "The ... dependence makes ... model of FF Arb-PUFs no longer differentiable", consistent with model (2) that the term inside the $sgn()$ function is not differentiable with respect to feature variables $w(i)$. Reported in [1], the best machine learning method used over 27 hours of training time to break feed-forward arbiter PUFs with 6, 7, or 8 FF loops, with the prediction accuracy reaching over 95% for the best trial among 40 trials. Reported in [2], the same machine learning method employed in [1] took over 3 hours to break 128-bit feed-forward arbiter PUFs with 6 or more loops. In studies reported in both [1] and [2], the method can break only PUFs with all feed-forward loops of the same loop length.

The non-differentiability of the term inside the $sgn()$ function in model (2) makes it difficult for any machine learning method to estimate the values of the feature variables $w(i)$. Since model (1) for arbiter PUFs can be easily broken by a machine learning method, we propose the following model for the one-loop feed-forward arbiter PUF with the loop starting at stage i_1 and ending at stage i_2:

$$\begin{aligned}
r &= sgn\left(v(n) + \sum_{i \neq i_2} \phi(i)\omega(i) + \omega(i_2)\right) \text{ or,}\\
r &= \text{sgn}\left(v(n) + \sum_{i \neq i_2} \phi(i)\omega(i) - \omega(i_2)\right) \quad (4)
\end{aligned}$$

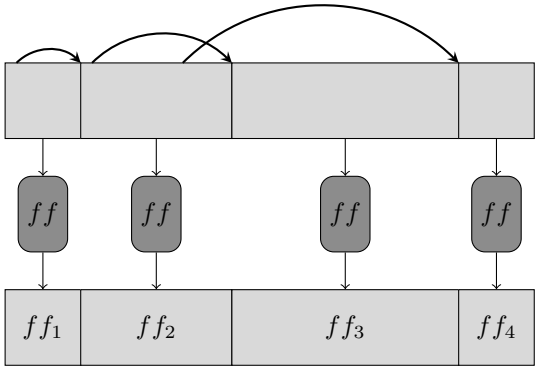


Fig. 3. Example of the feature extraction process for a 3-FF Arbiter PUF.

Model (4) means that only one of the two equalities describes the relationship between the response and the challenge bits correctly. A machine learning process for this model is to use each equality to estimate the values of feature variables over a set of challenge-response pairs, and the equality which takes less iterations to convergence in the machine learning process is chosen as the correct equality of the model.

With model (4), a one-loop feed-forward arbiter PUF can be broken with no more than twice the training time for breaking a same-size arbiter PUF. But for a feed-forward arbiter PUF with k loops, the model will consist of 2^k equalities with only one equality correctly describing the relationship between the response and the challenge bits. The 2^k equalities are due to the two possible choices among 0 and 1 for the challenge bit at each of the loop-end stages, and a total of k challenge bits have 2^k possible combination. Thus, a k-loop feed-forward arbiter PUF will take a machine learning method 2^k times the time the same method break a same-size arbiter PUF. Even with 2^k equalities to determine, a 7-loop PUF will take only 128 times the time that is needed for breaking an arbiter PUF, which is expected to be far faster than the fastest existing method can do. In addition, model (4) can handle any types of loops, loops with overlaps or non-overlap, and loops of different lengths. The fastest machine learning method used in [1] and [2] for their models can handle only PUFs with feed-forward loops of the same length, and two other methods employed in [1] and [2] can handle PUFs with only one loop or two non-overlapping loops. Fig. 3 illustrates the behavior of model (4) for a 3-FF Arbiter PUF. The arrows above the challenge indicate the starting and ending point of each feedforward loop. The final feature vector is the result of concatenating the individual feature vectors from each maximal challenge block without the ending feed forward loop.

III. The Machine Learning Method

Multi-layered neural network has been widely used in machine learning applications, and is one of the recognized methods for high learning power. To examine our proposed model (4) and to perform comparative studies with the existing model (2), we decide to employ neural network methods.

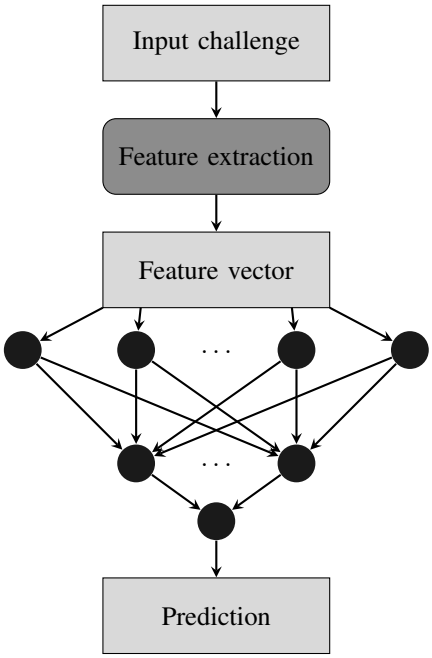


Fig. 4. Pipeline to break FF Arbiter PUF using MLP.

The existing model (2) has non-differentiable terms, but the sigmoid functions used in neural network methods can approximate non-differentiable, even discontinuous functions through nonlinear combinations of sigmoid functions. In particular, the traditional version of the neural network named Multilayer Perceptrons (MLP) [10] has been reported to be able to learn any binary function using hidden preceptrons, even with a single hidden layer of units. In our study we decide to apply MLP to the non-differentiable existing model as well as our model.

In the experimental studies presented in Section IV, we employ an MLP method with the following algorithmic structure. For a n-stage FF APUF with k feed-forward loops (i.e. the PUF has n stages of multiplexers, among which k stages receive input from internal feed-forward loops), the neural network consists of three layers of neurons, with $n-k$ neurons at the input layer, one neuron at the output layer, and 2^{k+1} neurons at the internal hidden layer. The number of neurons at the input and output layers, respectively, correspond to the input challenge bits and the output bit. The number of internal neurons is chosen after we tried several different values for the existing model (2) and found the number of 2^{k+1} neurons is the best for several trials on average. For each PUF, the same MLP method is applied to our model (4) and the existing model (2). The structure of the neural network method employed is illustrated in Fig. 4.

IV. Experiments and Discussion

In this section, we describe the experimental setting used for validating the proposed contributions. The source code is available for reproducibility. The source code was implemented in Python using the Machine Learning library scikit-learn [11]. For the simulated FF Arbiter PUF, we designed an artificial data generator with delays drawn from a Gausiann distribution with mean 300 and standard deviation 40. All the experiments were executed 100 times with randomly generated Arbiter PUFs.

Let's define C as the vector of training/test challenges, where $f(c)$ is the output of the predictive model given the challenge $c \in C$, and y_c is the ground-truth label of that challenge (i.e. the actual label). In order to assess the performance of each model, the following metrics were considered:

- Accuracy (Acc) (see (5)).
- F1-score (F1) (see (6)).
- Area Under the ROC Curve (AUC).
- False Acceptance Rate (FAR) (see (12)).
- False Recognition Rate (FRR) (see (13)).
- Equal Error Rate (EER) (see (14)).
- Time in seconds (Time).

These metrics cover a wide set of variables traditionally used by Machine Learning and PUF community. Accuracy defined as the fraction of observations correctly predicted by the model. While accuracy is a good indicator of model performance on hard (discrete) decisions, especially when the dataset is balanced, other metrics like AUC allow a more stable and sound analysis of the model behavior. Moreover, AUC is able to discriminate models regarding their prediction confidence, being able to distinguish between models with the same binary decisions but with different confidence levels.The Area Under the ROC Curve (AUC) is a good way to assess the relative confidence of the model per observation [12]. The F1-score besides being robust on unbalanced scenarios is able to provide a good estimate on the two types of errors. F1-score is the harmonic mean of precision and recall, being able to provide a good measure of the trade-off between both error types.

On the other hand, FAR, FRR and EER are frequently used in the biometrics community. The first two metrics describe one type of error, false negatives and false positives respectively. The latter, is the mean of the two errors.

All metrics were normalized to 0-100% in the tables to be expressed in percentages.

$$Acc(y,f) = \frac{|\{y_c = f(c) | c \in C\}|}{|C|} \tag{5}$$

$$F1(y,f) = 2\frac{Precision(y,f) \cdot Recall(y,f)}{Precision(y,f) + Recall(y,f)} \tag{6}$$

$$Precision(y,f) = \frac{TP(y,f)}{TP(y,f) + FP(y,f)} \tag{7}$$

$$Recall(y,f) = \frac{TP(y,f)}{TP(y,f) + FN(y,f)} \tag{8}$$

TABLE I
PERFORMANCE ACHIEVED NEURAL NETWORKS ON FF-ARBITER PUF.

Stages	FF-loops	Challenges	Features	Acc	F1	AUC	FAR	FRR	EER	Time
64	1	10k	Model (2)	76.9380	75.2473	87.7516	10.9707	12.0913	11.5310	4.9618
			Model (4)	**94.2400**	**93.0782**	**98.3860**	**3.6587**	**3.7013**	**3.6800**	6.2090
	2	20k	Model (2)	88.9507	90.2855	96.9067	5.5447	5.5047	5.5247	5.8968
			Model (4)	**90.3880**	**91.5796**	**97.6214**	**5.0300**	**4.5820**	**4.8060**	5.4319
	3	20k	Model (2)	85.8133	86.5879	94.5011	7.1267	7.0600	7.0933	8.5547
			Model (4)	**91.0133**	**91.5116**	**97.4364**	**4.5540**	**4.4327**	**4.4933**	8.5037
	4	200k	Model (2)	86.9842	88.3420	94.9425	6.6003	6.4155	6.5079	128.7664
			Model (4)	**89.0941**	**90.2333**	**96.4063**	**5.5599**	**5.3460**	**5.4530**	93.3633
	5	200k	Model (2)	80.2700	80.6559	89.8886	9.9280	9.8020	9.8650	199.8713
			Model (4)	**87.1367**	**87.3106**	**94.9460**	**6.1400**	**6.7233**	**6.4317**	117.0242
	6	200k	Model (2)	81.4510	80.2503	90.6701	9.2340	9.3150	9.2745	356.1145
			Model (4)	**86.8660**	**85.7388**	**94.9044**	**5.6250**	**7.5090**	**6.5670**	374.7134
128	1	20k	Model (2)	90.3593	90.4805	97.8942	4.7960	4.8447	4.8203	4.1466
			Model (4)	**95.0387**	**95.0983**	**99.2549**	**2.4280**	**2.5333**	**2.4807**	5.4022
	2	80k	Model (2)	93.5593	93.6249	98.9123	3.1905	3.2502	3.2203	17.0049
			Model (4)	**96.0915**	**96.1357**	**99.5563**	**1.9920**	**1.9165**	**1.9543**	15.4731
	3	100k	Model (2)	76.3428	75.9526	86.0464	11.8346	11.8226	11.8286	23.6495
			Model (4)	**92.4511**	**92.3312**	**98.2216**	**3.8576**	**3.6913**	**3.7745**	27.1399
	4	200k	Model (2)	93.6031	93.4303	98.5981	3.1333	3.2636	3.1984	81.6413
			Model (4)	**95.3891**	**95.2709**	**99.2798**	**2.3048**	**2.3061**	**2.3054**	54.1264
	5	200k	Model (2)	81.9340	82.1104	91.3108	8.9740	9.0920	9.0330	86.4605
			Model (4)	**86.9780**	**87.0523**	**94.9604**	**6.2460**	**6.7760**	**6.5110**	81.6782
	6	200k	Model (2)	68.2183	65.8453	76.3937	14.9857	16.7960	15.8908	215.0679
			Model (4)	**84.0047**	**83.9372**	**90.1570**	**9.9147**	**9.2777**	**9.5962**	324.0204

$$TP(y,f) = |\{y_c = 1 \ \wedge \ f(c) = 1 | c \in C\}| \quad (9)$$

$$FP(y,f) = |\{y_c = 0 \ \wedge \ f(c) = 1 | c \in C\}| \quad (10)$$

$$FN(y,f) = |\{y_c = 1 \ \wedge \ f(c) = 0 | c \in C\}| \quad (11)$$

$$FAR(y,f) = \frac{FN(y,f)}{|\{y_c = 0 \ \vee \ f(c) = 0 | c \in C\}} \quad (12)$$

$$FRR(y,f) = \frac{FP(y,f)}{|\{y_c = 1 \ \vee \ f(c) = 1 | c \in C\}} \quad (13)$$

$$EER(y,f) = \frac{FAR(y,f) + FRR(y,f)}{2} \quad (14)$$

Table I shows the results of the models on the FF Arbiter PUF using Multilayer Perceptron machine learning method. From the data in Table I, it can be seen that both models are able to break the FF Arbiter PUF in a few seconds but with different accuracies. Note that the bold fond in Table I indicates higher prediction accuracies measured with different metrics. It can be seen that whether measured in the straightforward metric accuracy (Acc) or other metrics, our new model produced much higher prediction accuracy. The data in Table I also show that for fixed number of training data (corresponding to rows with challenges fixed at 200k), as the number of FF-loops increases, it is increasingly difficult for model (2) to achieve good performance while the proposed model achieve significantly higher accuracy.

In order to examine the predictive power of our model and its comparative performance against the existing model (2) as the amount of training data increases, we carried out experiments by varying the amount of training data, and measured the prediction accuracy (the Acc metric). We plot the accuracies predicted by the two models in Fig. 5 for different numbers of feed-forward loops. In the plot, the vertical axis is the accuracy and the horizontal axis is the number of challenge-response pairs in log_{10} scale. The plotted curves show that our new model have significantly higher accuracy in all cases.

V. CONCLUSIONS

In this work, we propose a model for Feed-Forward Arbiter PUFs, and employ a Multilayer Perceptrons machine learning method with adaptation to determine the values of the PUF features. Our contributions can be summarized as follows:

- A new model that captures the nature of FF Arbiter PUF, and
- the discovery of a problem-tailored multilayer perceptrons as high-performance machine learning tool for PUF attack studies.

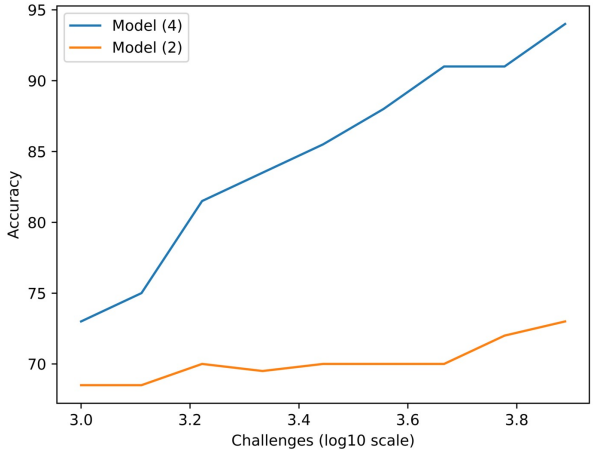
Our contributions have been validated on a set of FF Arbiter PUFs diverse architectures in terms of number of stages (64 and 128-bits), number of FF-loops (1-6), and the properties of FF loops (overlapping and non-overlapping). The approach explored in this work is able to break feed-forward Arbiter PUFs with high accuracy and high efficiency, providing secure application developers useful information on a new vulnerability of FF Arbiter PUFs while also pointing out one of the possibilities of increasing attack resistance by increasing the stages to a larger number.

ACKNOWLEDGMENT

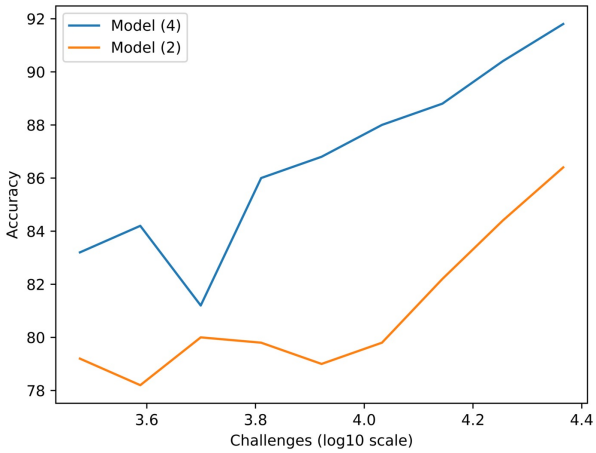
The research was supported in part by National Science Foundation under Grant No. CNS-1526055.

REFERENCES

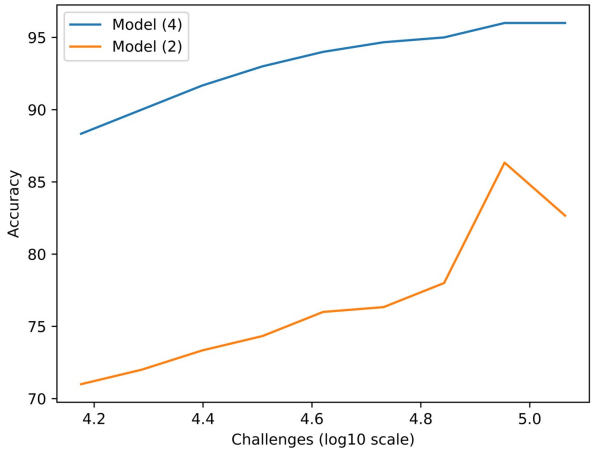
[1] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 237–249.



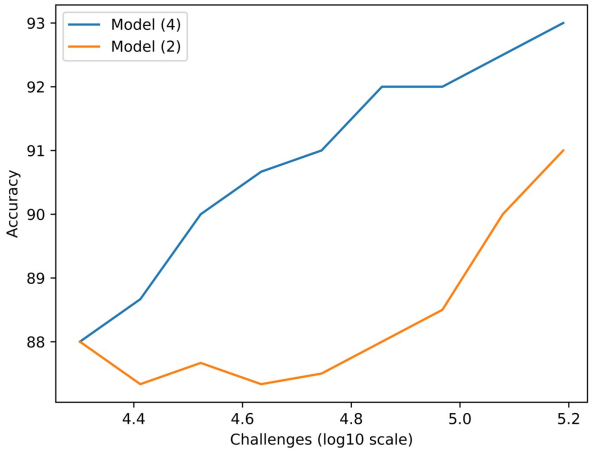
(a) $k = 1$



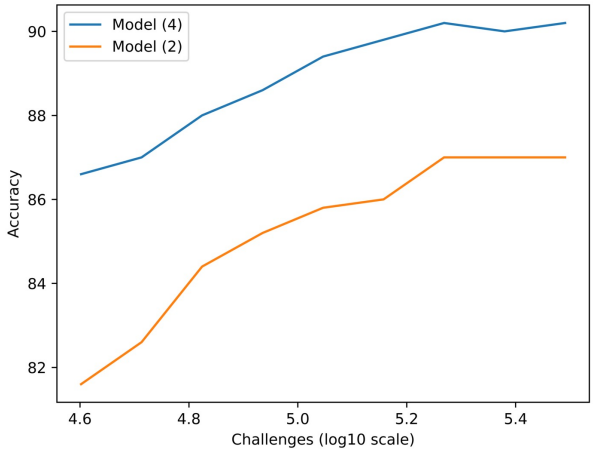
(b) $k = 2$



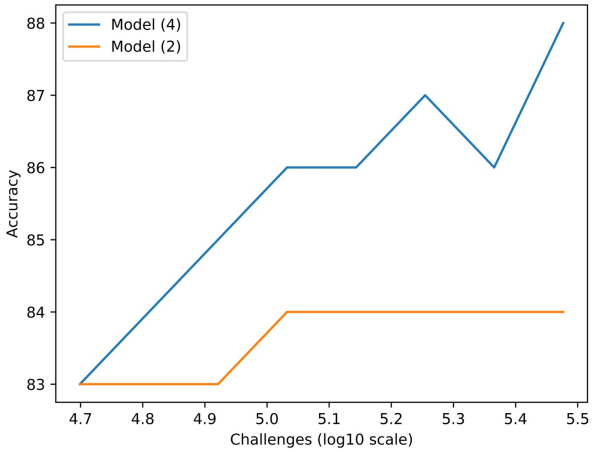
(c) $k = 3$



(d) $k = 4$



(e) $k = 5$



(f) $k = 6$

Fig. 5. Learning curves of model (2) and model (4) for k-FF PUFs. The vertical Y label represents the average accuracy score (%).

[2] 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.
[3] D. Lim, J. W. Lee, B. Gassend, G. E. 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.
[4] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Test Conference, 2008. ITC 2008. IEEE International*. IEEE, 2008, pp. 1–10.
[5] 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.
[6] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability," in *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*. IEEE, 2012, pp. 37–42.
[7] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on pufs with application to noise bifurcation," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2015, pp. 17–31.
[8] X. Xu, U. Rührmair, D. E. Holcomb, and W. Burleson, "Security evaluation and enhancement of bistable ring pufs," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2015, pp. 3–16.
[9] D. Schuster and R. Hesselbarth, "Evaluation of bistable ring pufs using single layer neural networks," in *International Conference on Trust and Trustworthy Computing*. Springer, 2014, pp. 101–109.
[10] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14, pp. 2627–2636, 1998.
[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
[12] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.