PUFs Deep Attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs

Mahmoud Khalafalla
Department of Electrical and Computer Engineering
University Of Waterloo
Waterloo, Canada
mkhalafa@uwaterloo.ca

Catherine Gebotys
Department of Electrical and Computer Engineering
University Of Waterloo
Waterloo, Canada
cgebotys@uwaterloo.ca

***Abstract*—In the past decade and a half, physical unclonable functions (PUFs) have been introduced as a promising cryptographic primitive for hardware security applications. Since then, the race between proposing new complex PUF architectures and new attack schemes to break their security has been ongoing. Although modeling attacks using conventional machine learning techniques were successful against many PUFs, there are still some delay-based PUF architectures which remain unbroken against such attacks, such as the double arbiter PUFs. These stronger complex PUFs have the potential to be a promising candidate for key generation and authentication applications. This paper presents an in-depth analysis of modeling attack using deep learning (DL) techniques against double arbiter PUFs (DAPUFs). Unlike more conventional machine learning techniques such as logistic regression and support vector machines, DL results show enhanced prediction accuracy of the attacked PUFs, thus pushing up the boundaries of modeling attacks to break more complex architectures. The attack on 3-1 DAPUFs has improved accuracy of over 86% (compared to previous research achieving a maximum of 76%) and the 4-1 DAPUFs accuracy ranges between 71%-81.5% (compared to previous research of maximum 63%). This research is crucial for analyzing security of existing and future PUF architectures, confirming that as DL computations become more widely accessible, designers will need to hide the PUFs CRP relationship from attackers.**

***Index Terms*—Hardware security, PUFs, Modeling attacks, Machine learning, Deep learning**

I. Introduction

Physically unclonable functions (PUFs) are considered a promising cryptographic primitives that can be used in storing important data in a device in a way which is resistant to physical attacks. Many applications of PUFs have been proposed including key generation, challenge-handshake authentication, licensing, etc. PUFs have been further classified into weak (such as powered on state of SRAM) and strong (such as arbiter PUFs) categories according to the space size of the challenge response pairs (CRPs). PUFs are currently being used in Industry [1] as well as in academia where questions of attack and design remain open. Interestingly, PUF structures have also been proposed as a solution to thwart reverse engineering of circuits [2].

Arbiter PUFs (APUFs) have been widely proposed as a vehicle for delay-based PUFs in the literature due to their large challenge space (for example compared to other delay-based PUFs, such as ring oscillator PUFs and memory-based Butterfly PUFs) [3]. Large challenge space is a desirable feature for strong PUFs, avoiding attacks which over time can defeat the PUF by collecting all challenge response pairs. Moreover, having so many CRPs is crucial for key generation applications that require many sessions with different keys per chip through its lifetime. Many new proposed PUF variants are higher level structures based on APUFs such as XOR APUF, double APUF (DAPUF), feed forward APUF and controlled APUF. DAPUF was specially proposed in [4] and [5] to overcome the routing constraints on FPGA chips and complicate the relationship between input challenges and responses with minimum hardware overhead. In a typical DAPUF design, using N APUFs leads to a response that is the output of N(N-1)-input XOR. Hence, DAPUFs showed more resistance against modeling attacks using conventional machine learning (ML) techniques and its security could not be broken as reported in previous research [4], [6].
The contributions of this paper are as follows:

1) Successfully modeling 2-1 DAPUFs using conventional ML techniques with CRPs collected from a real DAPUF hardware implementation.
2) The first reported successful modeling attacks against 3-1 DAPUFs using deep learning (DL) techniques which shows it performs better than conventional ML algorithms.
3) Suggest DL techniques to enhance the modeling accuracy of 4-1 DAPUF compared to previous research [6].

The rest of the paper is organized as follows. Section 2 provides a review of DAPUFs architectures and modeling attacks. Section 3 describes in details the modeling techniques and computing resources used in our experiments. Obtained results are provided in Section 4 and followed by a discussion and conclusion in Section 5.

978-3-9819263-2-3/DATE19/©2019 EDAA

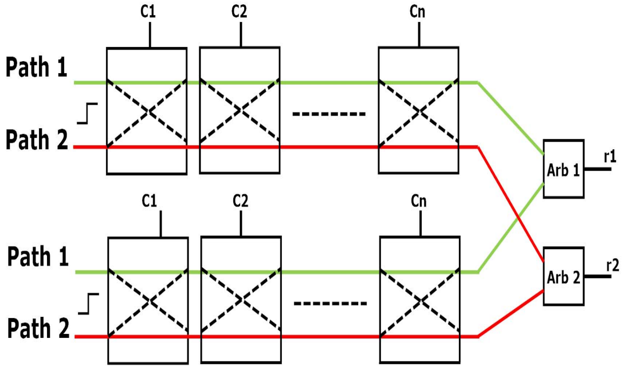


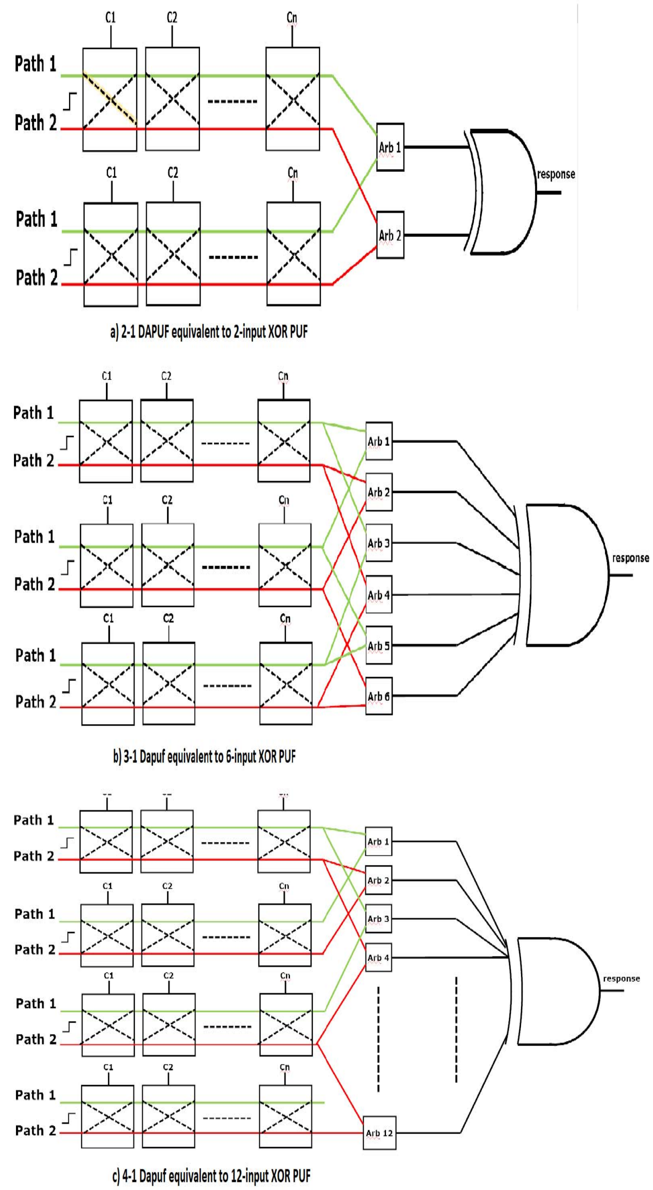
Fig. 1. Double Arbiter PUF Architecture

II. DAPUFs ARCHITECTURES AND MODELING ATTACKS

DAPUFs were introduced in [4] and [5] to overcome routing constraints imposed by FPGAs when implementing delay-based PUFs. The main idea of this approach is to use two identical APUFs placed adjacent to each other to eliminate routing differences. Every delay path has identical routing delays compared to the matching path in the second PUF. Hence, PUF response is determined by the matching delay paths race of the DAPUFs as shown in Fig. 1 The DAPUF design is further extended to make it more complex and harder to model by introducing XOR operation between PUF responses. As mentioned earlier, DAPUFs designs use minimum hardware overhead and can produce $O(N^2)$ input XOR response using $O(N)$ APUFs. For example, Fig. 2 shows the 2-1 DAPUF, 3-1 DAPUF, and 4-1 DAPUF are equivalent to 2-input XOR APUF, 6-input XOR APUF, 12-input XOR APUF respectively.

Numerical Modeling attacks against PUFs were one of the earliest attacking techniques proposed in literature [7], [8], [9], [10], [11]. Given a set of CRPs and using machine learning techniques like Support Vector Machines (SVM), Logistic Regression (LR), and Evolution Strategy (ES), it is possible to accurately predict the PUF outcome for the whole challenge-response space. Many variants based on APUF were proposed in literature to thwart modeling attacks, yet PUFs security against such type of attacks remains questionable.

Modeling attacks using conventional ML techniques against DAPUFs and equivalent XOR PUFs have been reported in literature. For example, Machida, T., et al [4] showed that LR attacks against DAPUFs were not successful. Furthermore, modeling accuracy was in range 56% - 80% for 2-1 DAPUF, $\sim$ 56% for 3-1 and 4-1 DAPUFs. However, their training sample was small (1K CRPs) and did not provide sufficient evidence of DAPUFs resistance. On the other hand, Yashiro, R., et al. [6] provided a more comprehensive analysis using 50K CRPs and their attacks were executed using SVM and DL. Their reported results showed 2-1 DAPUF could be successfully modeled using DL techniques with accuracy 90% and $\sim$ 85% using SVM. However, it was not clear how many PUF instances were under attack. Additionally, their results showed that DL and SVM could not successfully model 3-1 and 4-1 DAPUFs with modeling accuracy is 68% and 62% respectively.



a) 2-1 DAPUF equivalent to 2-input XOR PUF

b) 3-1 Dapuf equivalent to 6-input XOR PUF

c) 4-1 Dapuf equivalent to 12-input XOR PUF

Fig. 2. N-1 DAPUFs Architecture, a) 2-1 DAPUF , b) 3-1 DAPUF, c) 4-1 DAPUF.

Finally, modeling attacks against XOR PUFs were reported successfully in literature. However, these attacks were executed against maximum 6-input XOR PUFs using real PUF implementations and 200K CRPs [8] and 9-input XOR PUFs using simulated PUFs and 350K CRPs [10]. Therefore, the security performance of 3-1 DAPUF was tested by applying the same modeling technique used in [8] against 6-input XOR PUF. The poor results based on this test justifies shifting our focus to exploit DL techniques to break 3-1 DAPUF security.

III. Methodology and Experimental Setup

This section discusses in details the mathematical models, network architectures and the environment setup used to attack the DAPUFs.

A. The Mathematical Model of DAPUFs

As previously discussed, the DAPUF response is the output of an XOR. Therefore, the model used to attack DAPUFs is the same mathematical model used for XOR arbiter PUFs [8]. Consequently, The output response of a DAPUF is the multiplication of every single path response as shown in equation 1.

$$Txor = \prod_{i=1}^{l} sign(\vec{w_i^t}\vec{\phi_i}) \tag{1}$$

The $(\vec{w_i^t})$ is the delay difference vector of a single APUF and responsible for encoding delay difference at every stage [7], whereas $\vec{\phi_i}$ is the feature vector that represent the impact of every stage delay difference on the overall PUF response. If (δ) is the delay difference between the upper and lower path at a certain stage then its impact will be (+) or (-) depending on how many times paths will go straight or crossed after this stage. Hence, it is a function of input challenge bits and is a string of (1) and (-1) as depicted in equation 2 [7]. Note that 'k' represents the number of stages, 'l' is the stage position in APUF and 'C_i' is the challenge bit value at the i_{th} position.

$$\vec{\phi}(\vec{C}) = (\phi^1(\vec{C}), ...\phi^k(\vec{C}), 1), \phi^1(\vec{C}) = \prod_{i=l}^{k}(1 - 2C_i) \tag{2}$$

The final value of Txor is either (1) or (-1) representing (1) or (0) responses respectively. Hence, modeling the DAPUF response can be considered as a binary classification problem, which can be solved by conventional machine learning techniques (e.g. LR, SVM). Equations 3 and 4 are derived from equation 1 to calculate the decision boundary needed to classify the PUF response. Equation 3 is doing the classification by determining a non-linear decision boundary, which requires $l \times (k+1)$ parameters (l is the number of XOR inputs and k is the number of PUF stages). Whereas equation 4 takes a further step and calculates the linear decision boundary by applying outer product among the wights and features of all XOR input PUFs. The latter approach requires $(k + 1)^l$ parameters for every single CRP, which scales exponentially when increasing the XOR inputs. Please note that 'l' represents the number of XOR inputs in equations 1,3, and 4.

$$Txor = sign(\prod_{i=1}^{l}(\vec{w_i^t}\vec{\phi_i})) \tag{3}$$

$$Txor = sign(\bigotimes_{i=1}^{l} \vec{w_i^t} \bigotimes_{i=1}^{l} \vec{\phi_i^t}) \tag{4}$$

The 64 stage 2-1 DAPUF is similar to 64 stage 2-input XOR PUF, thus the number of parameters for every single CRP is $(64+1)^2$ = 4225. Due to this relatively low number of parameters, building a model for the 2-1 DAPUF using logistic regression is a reasonable choice and was adopted in this work. However, deep learning techniques were used in attacking 3-1 and 4-1 DAPUFs because the number of parameters scales exponentially (65^6 in case of 3-1 DAPUF and 65^{12} in case of 4-1 DAPUF). Furthermore, attacks against 3-1 and 4-1 DAPUFs were invoked using LR with non-linear decision boundary as a proof that conventional machine learning techniques are not successful in breaking larger DAPUFs. This is in contrast to the same technique performance against similar XOR-PUFs, which were accurately modeled using non-linear decision boundary in previous research [8] [10].

B. The Architectures of The Deep Neural Networks

Many deep neural networks architectures and configurations have been investigated in the experiments invoked to find the network that can successfully learn the complex relationships of 3-1 and 4-1 DAPUF architecture. The main idea is to build a network that emulates the DAPUF architecture, which makes the training process easier for the network to learn the non-linear relationships among different stages of DAPUF from the same/different paths.

The first architecture is illustrated in Fig. 3 where every path handles one APUF of the six PUFs contributing to 3-1 DAPUF output. The final result equals the six PUFs multiplication and the output of the probability of the response being '1' or '0'. All layers are fully connected because convolutional layers did not produce good results on this problem and could not capture all the relationships among different stages. Furthermore, the number of neurons in every layer is 2000 after running many experiments to tune the network. Although after using 65 × 6 = 390 neurons results were enhanced but the best accuracy performance obtained when using 2000 neurons in every layer. Also note that for 4-1 DAPUF the network has 12 branches instead of six.

The second architecture is shown in Fig. 4 and it is 12 fully connected (FC) layers for 3-1 DAPUF and the number is increased for the case of 4-1 DAPUF to be 18 layers. At the end of the network there is a dropout layer to avoid over-fitting problems and train a more generalized model. The second architecture introduced nearly the same accuracy results while taking nearly 40% less training time. Hence, the reported results in the next section are based on the second architecture. Furthermore, both architectures used adaptive moment estimation (Adam) optimization algorithm proposed by Kingma, D. and Ba, J. [12] because experiments showed that it converges faster and introduces better accuracy results than the normal gradient decent algorithm.

C. Hardware and Software Experimental Setup

DAPUF architectures were implemented on Mojo V3 development boards, each containing a Spartan 6 XC6SLX9 FPGA [13] (45nm process technology). Real CRPs were

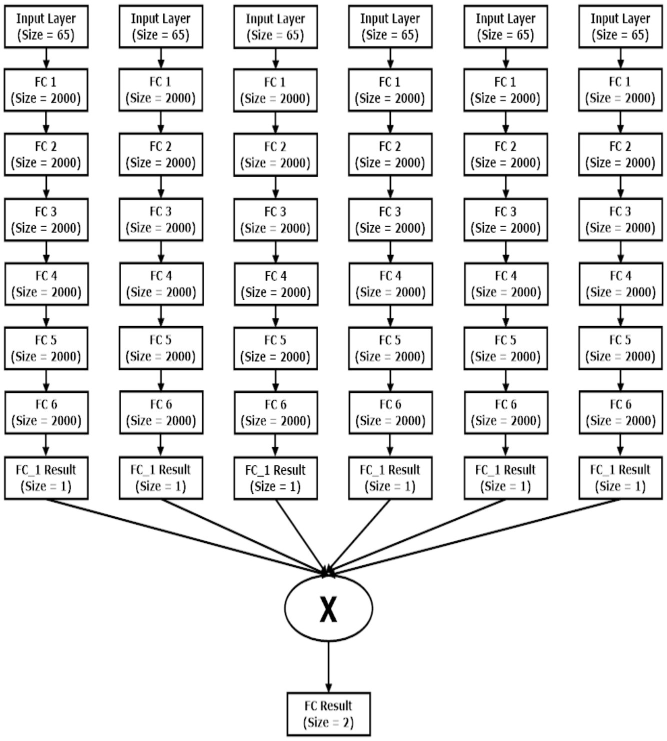


Fig. 3. N-Branch Fully Connected with Multiplication Deep Neural Network

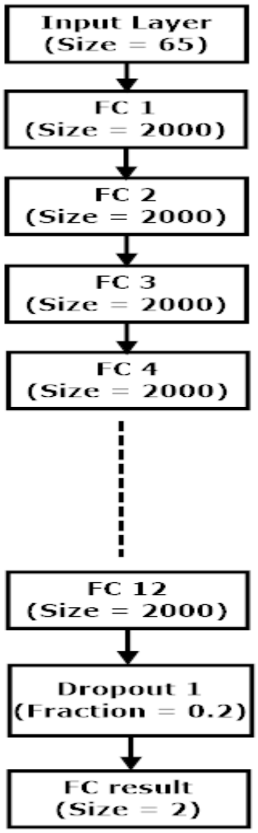


Fig. 4. 1-Branch Fully Connected Deep Neural Network

TABLE I
THE RANDOMNESS OF 2-1 DAPUFS UNDER ATTACK.

	Chip-A	Chip-B	Chip-C
Response Randomness	43.9%	40.2%	57.3%

collected from the boards and all readings were performed under normal voltage and temperature conditions. Most previous research does not disclose how CRPs are generated apart from some which use LFSRs [14] and there are not any known publicly available CRPs. Hence, the experiments used a random CRP set generated using linear-feedback shift register (LFSR) as in [8]. The only difference is that LFSR is implemented on chip. Finally, The Xilinx ISE Design Suite 14.7, Xilinx PlanAhead 14.7, and Xilinx FPGA editor are utilized to implement, manually place and route the DAPUF designs.

Logistic regression attacks were executed on Intel 8th Gen I7-8250 CPU with 16GB RAM of memory. The code for LR with non-linear decision boundary is implemented by Ruhrmair, U. et al [7] and is available online [15]. Their implementation with RProp optimization algorithm was successful in attacking up to 6-input XOR PUFs. Hence, it was used to attack 3-1 DAPUF to prove The PUF resistance against conventional ML techniques and justify the need for deep learning algorithms.

Deep learning attacks were executed on a Nvidia GeForce GTX 1080 Ti GPU card with 11GB RAM and worth around 800$. The code implementation for network architectures, training, and evaluation was done using Tensorflow v1.2. The CRP set was generated using the same HW LFSR. However, a set of 20M CRPs were collected to have enough samples to train the massive deep neural networks.

IV. EMPIRICAL RESULTS

This section shows in details the statistical properties of the PUFs under attack and the obtained results of the modeling attacks against 2-1, 3-1, and 4-1 DAPUFs.

A. LR Attacks Against 2-1 DAPUFs

Three 64 stages 2-1 DAPUFs were implemented on three Mojo V3 boards using the same bit configuration for all FPGA chips. The inter-chip hamming distance is 45% and randomness of every PUF response (percentage of '1' responses) is shown in Table 1. As discussed before, the CRP set size is 1M and all accuracy results shown are using a test set size = 1M - training set size.

Fig. 5 shows the modeling attack results against the three 2-1 DAPUF instances. The maximum accuracy performance using 100K CRPs for training is 93.4%, 81.7% , and 82.4% for Chip A, B, and C respectively. The graph shows that at training size equal to 20K CRPs, accuracy could reach $\sim$ 80% - 90%. By comparing these results with what was reported in [4] by Machida, T. et al (2-1 DAPUF accuracy 56%, 69%,

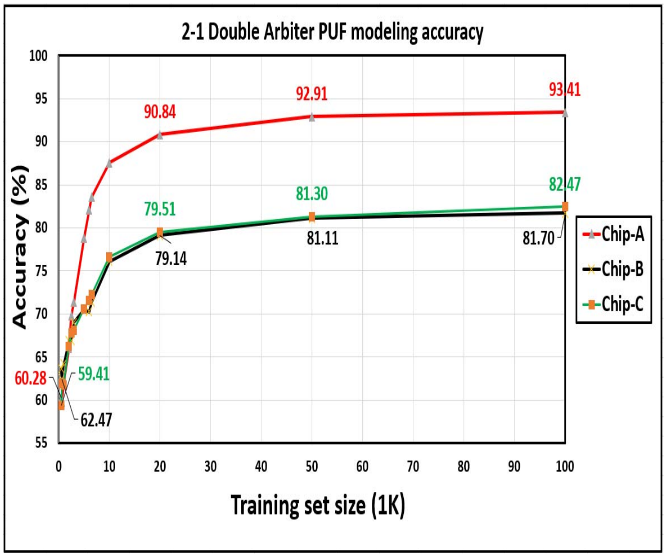


Fig. 5. LR with Linear Decision Boundary Attack Results on 2-1 DAPUF

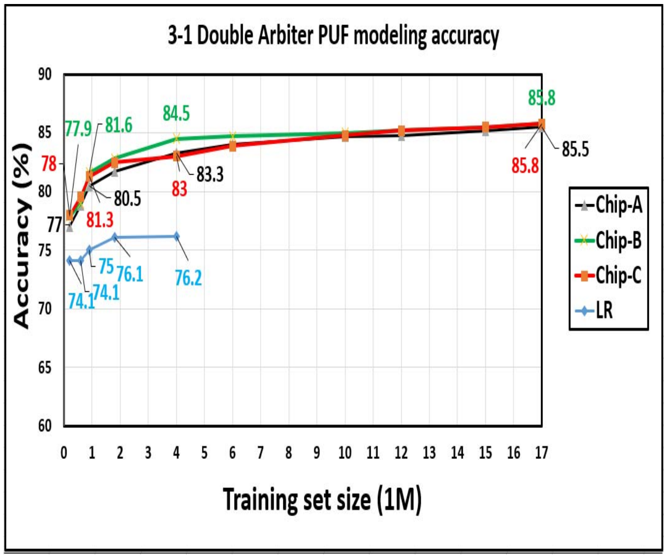


Fig. 6. LR and Deep Learning Attacks Results on 3-1 DAPUF

TABLE II
THE RANDOMNESS OF 3-1 DAPUFS UNDER ATTACK.

	Chip-A	Chip-B	Chip-C
Response Randomness	54.4%	46.7%	47.9%

TABLE III
THE RANDOMNESS OF 4-1 DAPUFS UNDER ATTACK.

	Chip-A	Chip-B	Chip-C
Response Randomness	48.7%	40.9%	51.8%

80%), It is obvious that 2-1 DAPUFs are not secure against LR with linear decision boundary modeling attacks. Furthermore, Yashiro, R. et al reported a successful modeling attack on 2-1 DAPUF using deep learning with 90% accuracy using 40K CRPs [6]. However, obtained results show that using LR with linear decision boundary can achieve better accuracy with half the number of CRPs and less cost because no need to use deep learning techniques and GPUs.

B. LR and Deep Learning Attacks against 3-1 DAPUFs

Similar to the previous experiments, three 64 stage 3-1 DAPUFs were implemented on three Mojo V3 boards. The inter-chip hamming distance is $\sim$ 39% and the randomness of every PUF response is shown in Table 2. The maximum training set size is 17M and The training/test set ratios is 90/10%. LR attacks were stopped after using 4M training set because it took a long time and increasing the training data did not seem to help the model to converge.

Fig. 6 shows the modeling attack results against 3-1 DAPUFs. The maximum accuracy performance using 17M CRPs for training is $\sim$ 86%. Moreover, The models could reach at least 83% accuracy using only 4M CRPs for training. On the other hand, the LR with non-linear decision boundary failed to converge and achieve more than 76% accuracy and took a long time because it is run on CPU. Typically, in all cases with huge training set it took $\sim$ 1hr-2hrs to reach 80% accuracy then within 3-5 hrs to reach the maximum accuracy. Note that this time is taken while training on one GPU. Furthermore, results show that the LR with non-linear decision boundary and the deep learning architectures used in this work are more successful than the conventional ML and DL techniques used in [4] and [6] respectively. Their reported attack results show a modeling accuracy of 56% - 68% using LR, SVM and DL techniques.

C. Deep Learning Attacks against 4-1 DAPUFs

The same experiments were run on three instances of 4-1 DAPUFs on three different Mojo V3 boards. The inter-chip hamming distance among the three instances is $\sim$ 50.9% and the randomness of every PUF response is shown in Table 3. Similarly, The maximum training set size is 17M and The training/test set ratios is 90/10%.

The modeling attack results against 4-1 DAPUFs are shown in Fig. 7. The maximum accuracy performance using 17M CRPs for training is 71.3%, 81.5%, and 73.2% for chip A, B, and C respectively. The results show that models reach 70%-80% accuracy at training set size of only 6M CRPs. The increase of training set size from 6M to 17M causes modeling accuracy to rise only 1%-3%. Hence, after a specific training set size, attackers should consider doing architectural modifications in their networks to strengthen the model accuracy instead of depending only on increasing the training set size. Although results show that the deep learning architectures used to attack 4-1 DAPUFs were not as successful as those used against 3-1 DAPUFs, but they still achieve better accuracy performance than the conventional ML and DL techniques used in [4] and [6] respectively. Their reported attack results show a modeling accuracy of 56% - 63% using LR, SVM and DL techniques. Hence, this work presents a step forward in

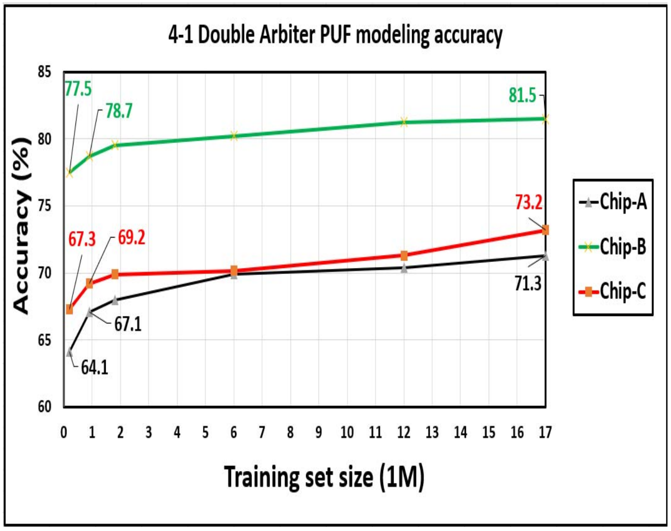


Fig. 7. Deep Learning Attacks Results on 4-1 DAPUF

fully modeling 4-1 DAPUFs and investigates more efficient techniques against PUFs with complex architectures.

V. Discussion and Conclusions

In this paper, successful modeling attacks have been executed against different DAPUFs architectures. Obtained results show that 2-1 DAPUF can be modeled using LR with linear decision boundary. Moreover, it achieves better accuracy (93.4%) using less CRPs and cheaper computing resources than ML & DL attacks reported in previous research [6]. Additionally, results proved that DL techniques can successfully model 3-1 DAPUF and achieved better accuracy than ML and DL attacks reported in literature (86% Vs. 68%) [4], [6]. Furthermore, experiments showed that same LR technique that was successful against equivalent XOR PUFs failed to achieve the same performance with 3-1 DAPUF. Hence, using DL techniques is justified to achieve better modeling accuracy. Although obtained results of 4-1 DAPUF attacks did not show the same success but the DL network used achieved better accuracy than previous research (71.3%-81.5% Vs. 63%) [6]. Other DL architectures and techniques should be investigated to achieve better results on 4-1 DAPUFs.

One concern that might rise is the practicality of modeling attacks using DL techniques, which requires huge set of CRPS for training. However, experiments were run on Mojo V3 boards running at 50MHZ frequency (clock period = 20ns) and the delay time of implemented DAPUFs ranges between 32-45ns. This means that reading one response with eased timing constraints can takes less than 100ns, thus collecting 20M CRPs is a matter of several minutes. Furthermore, with the AI hype and the race to introduce more efficient hardware, it is easy and even cheaper to use many GPU cards in parallel using cloud computing services and train huge CRP sets in parallel.

Finally, this work draws attention to the fact that corresponding challenge and response bits should not be revealed no matter how complex the PUF architecture is, since attacks are possible using the available DL techniques and hardware can be successfully attacked and broken. Furthermore, the ability to model multi-input XOR PUFs may enhance other hybrid attack schemes, which inject faults to disable part of the XOR inputs. Consequently, if the available modeling attacks can successfully model XOR PUFs with larger number of inputs, then the fault injections needed will be minimized and equipment used will be cheaper (no need to accurately disable one input at a time). Hence, hiding the challenge response relationship should be the first priority for PUF designers before introducing new complex architectures. This research was supported in part by grants from NSERC and XtremeEDA.

References

[1] N. 2013, *NXP Strengthens SmartMX2 Security Chips With PUF Anti-Cloning Technology*, 3rd ed., NXP, 2 2013.

[2] D. Yamamoto, M. Takenaka, K. Sakiyama, and N. Torii, *A Technique Using PUFs for Protecting Circuit Layout Designs against Reverse Engineering*. Cham: Springer International Publishing, 2014, pp. 158–173. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-09843-2_13

[3] L. D., "Extracting secret keys from integrated circuits," Master's thesis, MIT, 2004.

[4] M. I. Takanori Machida, Dai Yamamoto and K. Sakiyama, "A new arbiter PUF for enhancing unpredictability on FPGA," *The Scientific World Journal*, 2015.

[5] M. I. T. Machida, D. Yamamoto and K. Sakiyama, "A new mode of operation for arbiter PUF to improve uniqueness on FPGA," *Federated Conference on Computer Science and Information Systems (FedCSIS '14)*, 2014.

[6] R. Yashiro, T. Machida, M. Iwamoto, and K. Sakiyama, "Deep-learning-based security evaluation on authentication systems using arbiter PUF and its variants," in *Advances in Information and Computer Security*, K. Ogawa and K. Yoshioka, Eds. Cham: Springer International Publishing, 2016, pp. 267–285.

[7] U. Rührmair, F. Sehnke, J. S ölter, G. Dror, S. Devadas, and J. ü. Schmidhuber, "Modeling attacks on physical unclonable functions," *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*, p. 237, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866307.1866335

[8] U. Ruhrmair, J. Solter, 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," *Trans. Info. For. Sec.*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1109/TIFS.2013.2279798

[9] G. Hospodar, R. Maes, and I. 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)*, Dec 2012, pp. 37–42.

[10] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *Radio Frequency Identification*, S. Mangard and P. Schaumont, Eds. Cham: Springer International Publishing, 2015, pp. 17–31.

[11] G. T. Becker, "The gap between promise and reality: On the insecurity of xor arbiter pufs," in *CHES*. Springer, 2015, pp. 535–555. [Online]. Available: https://www.iacr.org/archive/ches2015/92930517/92930517.pdf

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 12 2014.

[13] Xilinx, *Spartan-6 FPGA Configurable Logic Block User Guide (UG384)*, Xilinx.

[14] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *2010 IEEE International Workshop on Information Forensics and Security*, Dec 2010, pp. 1–6.

[15] "LR with non-linear decision boundary and RProp optimization," http://www.pcp.in.tum.de/code/lr.zip, accessed: 2018-08-30.