Feature Extraction Driven Modeling Attack Against Double Arbiter PUF and Its Evaluation

Susumu Matsumi
Meijo University
1-501 Shiogamaguchi, Tempaku-ku
Nagoya Aichi, Japan
+81-52-832-1151
150441129@ccalumni.meijo-u.ac.jp

Yusuke Nozaki
Meijo University
1-501 Shiogamaguchi, Tempaku-ku
Nagoya Aichi, Japan
+81-52-832-1151
143430019@ccalumni.meijo-u.ac.jp

Masaya Yoshikawa
Meijo University
1-501 Shiogamaguchi, Tempaku-ku
Nagoya Aichi, Japan
+81-52-832-1151
dpa_cpa@yahoo.co.jp

ABSTRACT
Many imitations of electronic components exist in the market. The PUF has attracted attention as countermeasures against these imitations. The 2-1 DAPUF is one of the PUFs which is suitable for FPGA implementation. However, it is reported that some PUFs are vulnerable to modeling attacks using feature extraction. Regarding the effectiveness of feature extraction, it has not been evaluated in the modeling attack against 2-1 DAPUF. This study evaluated the effectiveness of feature extraction by simulation and FPGA implementation. The results showed that the feature extraction was effective for modeling attacks against 2-1 DAPUF.

CCS Concepts
Security and Privacy → Security in Hardware

Keywords
Authentication of Electronic Devices; Physical Unclonable Function; Machine Learning

1. INTRODUCTION
While cloud computing has been widely diffused, it is important to ensure the authentication of electronic devices connecting to networks. Regarding authentication technology of electronic devices, physical unclonable function (PUF) [1] has attracted attention. PUF uses the variation of semiconductor manufacturing for authentication. Several types of PUFs have been reported [2]. An arbiter PUF [3] and a 2-1 double arbiter PUF (2-1 DAPUF) [4], which use the variation of signal propagation delay, are typical PUF architectures. In particular, the 2-1 DAPUF has better performances than the arbiter PUF in the field programming gate array (FPGA) implementation.

However, the vulnerability of modeling attacks for arbiter PUF and 2-1 DAPUF has been reported [3, 5]. Modeling attacks build the PUF model by machine learning techniques. Then, modeling attacks predict the output of PUF, that is, these attacks can duplicate the function of the PUFs for authentication. For the modeling attack for 2-1 DAPUF, paper of [5] have performed the attack using the feature extraction method. However, the effectiveness of the modeling method by using the feature extraction is unclear because the comparison experiments are not performed. To evaluate the security of PUF against modeling attacks, it is important to verify the effectiveness of the feature extraction in 2-1 DAPUF. Therefore, this study evaluates the effectiveness of the feature extraction by experiments using simulation and FPGA.

AICCC '18, December 21–23, 2018, Tokyo, Japan

ACM ISBN 978-1-4503-6623-6/18/12...$15.00
DOI: https://doi.org/10.1145/3299819.3299835

2. Physical Unclonable Function
The PUF is a one-way function which is difficult to physically replicate. The PUF generates an output called a response from an input called a challenge. Since PUF generates responses using random manufacturing variation, different PUFs generate different responses. This property is called uniqueness.

2.1 Arbiter PUF
The arbiter PUF is one of the most popular PUFs. Figure 1 shows the outline of the arbiter PUF. The arbiter PUF consists of a selector chain and an arbiter. The selector chain consists of two equal length paths and *n* selector units. Also, the arbiter is a component for determination of output.

The arbiter PUF generates a response by using the difference between propagation delay times of two paths when two trigger

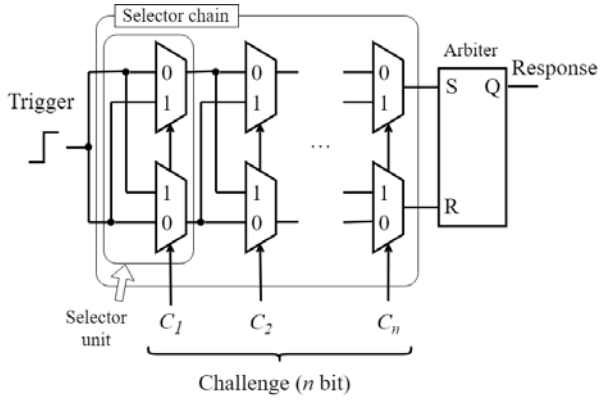

Figure 1. Outline of the arbiter PUF.

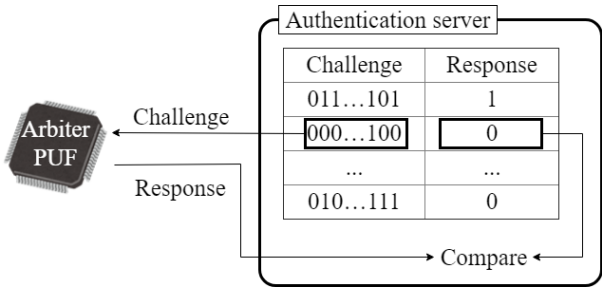

Figure 2. Authentication with arbiter PUF.

signals are input to each path at the same time. Each selector propagates the signal in parallel or crossing according to the challenge, and then two signals arrive at the arbiter. The arbiter determines which of the two paths has propagated the signal earlier and outputs a 1-bit response. Although the paths are designed to have the same length, their propagation delay times differ due to manufacturing variations. This means that the propagation delay time of the PUF is different from other PUFs. Therefore, responses are different for each arbiter PUF against the same challenge.

Figure 2 shows the overview of authentication using the arbiter PUF. As preparation, acquired challenge and response pairs (CRPs) are stored in the authentication server. In authentication, the authentication server sends a challenge to the arbiter PUF. Then, the arbiter PUF outputs a response against the received challenge and sends the response to the server. The server authenticates by comparing the received response with the registered response in the database.

2.2 Double Arbiter PUF

The Double Arbiter PUF (DAPUF) was proposed in [6]. The DAPUF is an improved arbiter PUF for FPGA implementation. Actually, the arbiter PUF implemented on FPGA has low uniqueness [7, 8]; therefore, available challenges for authentication are limited. The DAPUF can improve the uniqueness for the FPGA implementation by implementing multiple selector chains. The DAPUF, which has several selector chains, compares the propagation delays of the same paths of each

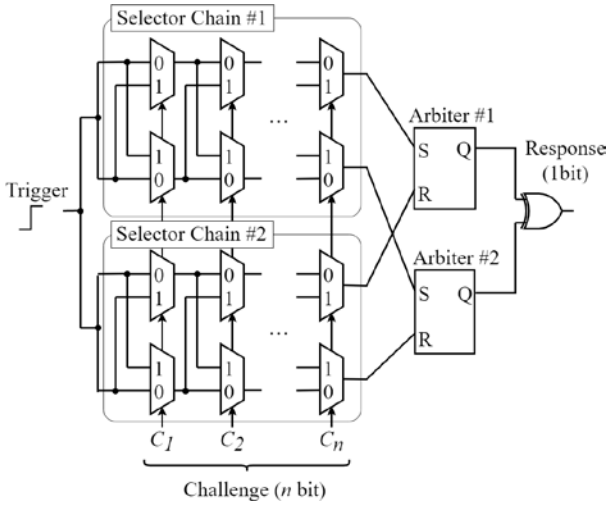

Figure 3. Outline of the 2-1 Double arbiter PUF.

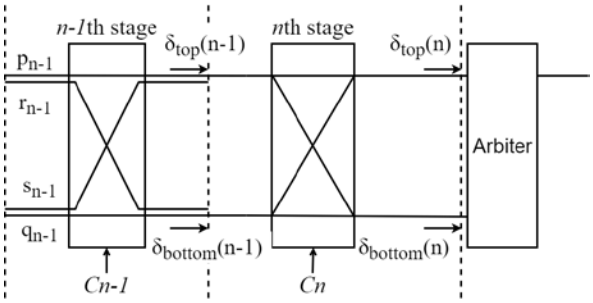

Figure 4. The delay of the arbiter PUF.

selector chain. Several DAPUF, including 2-1 DAPUF, and 3-1 DAPUF have been proposed in [4]. This study covered 2-1 DAPUF.

Figure 3 shows the structure of the 2-1 DAPUF. The 2-1 DAPUF consists of two selector chains and two arbiters and generates a response by an XOR operation of the output of the two arbiters.

3. Modeling Attack

3.1 Modeling Attack for Arbiter PUF

Modeling attacks for the arbiter PUF have been proposed in papers [3, 9]. The modeling attacks estimate the internal delay of the arbiter PUF by machine learning and create a model of the arbiter PUF. Then, the model can predict the response against challenge.

Figure 4 shows the delay of the arbiter PUF. The delay of the signal through the selector chain is the sum of the delays of each stage. $\delta_{top}(i)$ ($\delta_{bottom}(i)$) means the delay of the top (bottom) path from the start of the first stage to the end of the *i*th stage. As shown in Figure 4, the delays of each path of the *i*th stage selector are defined as p_i, q_i, r_i, and s_i. $\delta_{top}(i+1)$ and $\delta_{bottom}(i+1)$ are derived from p_i, q_i, r_i, s_i, $\delta_{top}(i)$, $\delta_{bottom}(i)$, and challenge *C*.

$$\boldsymbol{\delta_{top}(i+1)} = \frac{\mathbf{1}+\boldsymbol{b_{i+1}}}{\mathbf{2}}\left(\boldsymbol{p_{i+1}}+\boldsymbol{\delta_{top}}(\mathbf{i})\right) + \frac{\mathbf{1}-\boldsymbol{b_{i+1}}}{\mathbf{2}}\left(\boldsymbol{s_{i+1}}+\boldsymbol{\delta_{bottom}}(\mathbf{i})\right) \#(1)$$

$$\boldsymbol{\delta_{bottom}(i+1)} = \frac{\mathbf{1}+\boldsymbol{b_{i+1}}}{\mathbf{2}}\left(\boldsymbol{q_{i+1}}+\boldsymbol{\delta_{bottom}}(\mathbf{i})\right) + \frac{\mathbf{1}-\boldsymbol{b_{i+1}}}{\mathbf{2}}\left(\boldsymbol{r_{i+1}}+\boldsymbol{\delta_{top}}(\mathbf{i})\right) \#(2)$$

where $b_i = 1 - 2C_i$.

Then, the difference between $\delta_{top}(i)$ and $\delta_{bottom}(i)$ is defined as $\Delta(i)$. By formulae (1) and (2), $\Delta(i+1)$ is derived by

$$\Delta(\mathbf{i}+\mathbf{1}) = \boldsymbol{b_{i+1}}\cdot\Delta(\boldsymbol{i}) + \boldsymbol{\alpha_{i+1}}\boldsymbol{b_{i+1}} + \boldsymbol{\beta_{i+1}}, \#(3)$$

where

$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}$$

$$\beta_i = \frac{p_i - q_i - r_i + s_i}{2}.$$

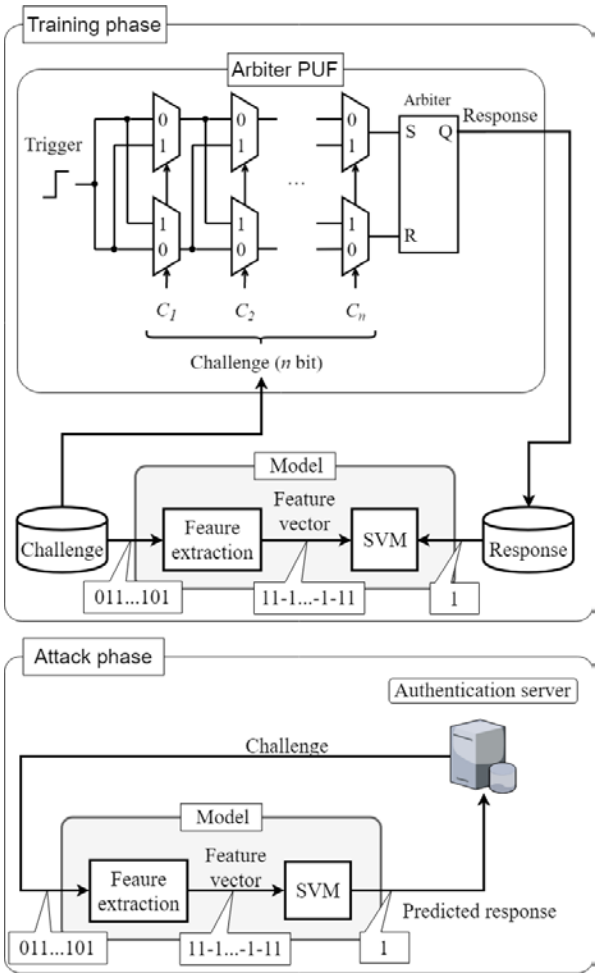

Figure 5. The flow of modeling attack for the arbiter PUF.

Then, the feature vector $\boldsymbol{\Phi}$ is defined as

$$\boldsymbol{\Phi_i} = \prod_{k=i+1}^{n} \boldsymbol{b_k} = \prod_{k=i+1}^{n} (\mathbf{1} - \mathbf{2}\boldsymbol{C_i}), \#(4)$$

where $\Phi_{n+1} = 1$.

Actually, converting a challenge to a feature vector is called feature extraction in [5].

Here, the parameter vector **w** is defined as

$$\boldsymbol{w_i} = \begin{cases} \boldsymbol{\alpha_1}, & \boldsymbol{if\ i = 1} \\ \boldsymbol{\alpha_i + \beta_{i-1}}, & \boldsymbol{if\ i = 2, \ldots, n}. \\ \boldsymbol{\beta_n}, & \boldsymbol{if\ i = n + 1} \end{cases} \#(5)$$

Finally, by formulae (4) and (5), $\Delta(n)$ is expressed as follows:

$$\Delta(n) = \mathbf{w}^{T} \cdot \boldsymbol{\Phi}$$

$\Delta(n)$ represents the delay time difference between the two signals in the arbiter. When $\Delta(n) < 0$, the response is 1, and when $\Delta(n) > 0$, the response is 0.

Figure 5 shows the flow of the modeling attack. As shown in figure 5, in the learning phase, a model is learned using CRPs of the arbiter PUF. First, challenge in CRPs are converted to feature vector by feature extraction. Second, The SVM is learned using feature vectors and responses. In the attack phase, the model impersonates the arbiter PUF by predicting the response from the challenge.

3.2 Modeling Attack for DAPUF

In the modeling attack for DAPUF [5, 10], the feature extraction method is used similarly in the attack for the arbiter PUF. Figure 6 shows modeling attacks against 2-1 DAPUF. As shown in figure 6, machine learning of model is performed using CRPs acquired from 2-1 DAPUF. At this time, the challenge is converted to the feature vector by the feature extraction (see formula (4)). Next, machine learning is performed using both feature vectors and responses. Finally, in the attack phase, the responses of 2-1 DAPUF is predicted by using the model. However, the effectiveness of the modeling method by using the feature extraction is unclear because the comparison experiments are not performed in paper [5].

4. EXPERIMENTS

In order to evaluate the effectiveness of feature extraction in modeling attacks, comparative experiments using 2-1 DAPUF on simulation and on FPGA were performed.

4.1 Experimental Environment

4.1.1 Simulation

In experiments by simulation, 32 and 64 stages 2-1 DAPUFs was created. Figure 7 shows 2-1 DAPUF on simulation. Simulation of 2-1 DAPUF was executed by assigning a virtual delay to each path of each stage. In order to reproduce manufacturing variations,

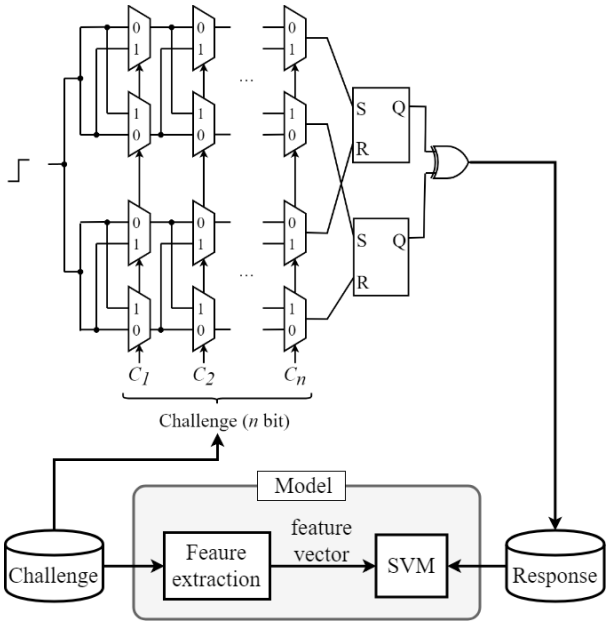

Figure 6. The modeling attack for the 2-1 DAPUF.

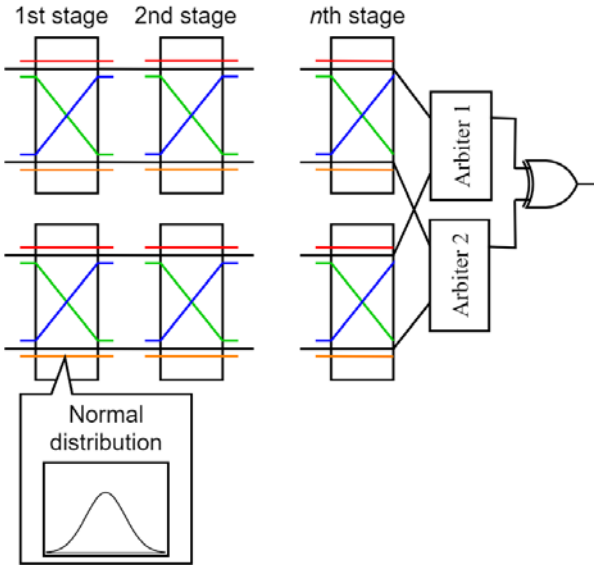

Figure 7. 2-1 DAPUF on simulation.

each delay was randomly determined according to the normal distribution with mean 1 and standard deviation 0.005.

4.1.2 FPGA

In experiments using FPGA, the 32 and 64 stages 2-1 DPUF was implemented. The evaluation board was used for the experiment. Figure 9 and Table 1 show the experimental environment. The 2-1 DAPUF was designed by using Verilog HDL and Xilinx ISE DesignSuite14.7. It was implemented into an FPGA Virtex-5 XC5VLX30 on a SASEBO-GII. The 32 stages 2-1 DPUF was placed on the fixed area from SLICE_X14Y77 to SLICE_X17Y44, and 64 stages 2-1 DAPUF was placed on the fixed area from SLICE_X14Y77 to SLICE_X17Y12 by using

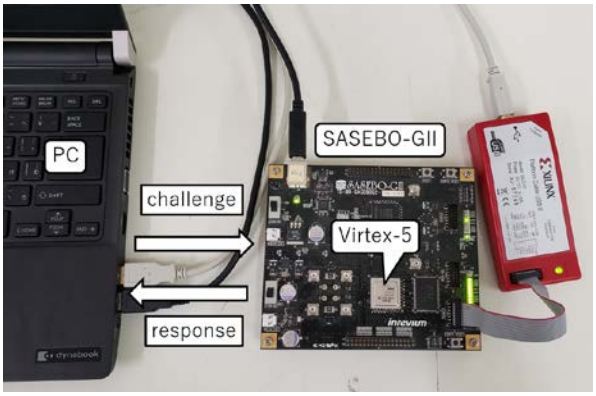

Figure 8. Evaluation system.

Table 1. Experiment Condition

PUF	2-1 DAPUF
# of stages	32 and 64
Evaluation board	SASEBO-GII
FPGA	Virtex-5 XC5VLX30
Development tool	Xilinx ISE Design Suite 14.7
Implementation tool	Xilinx PlanAhead v14.1

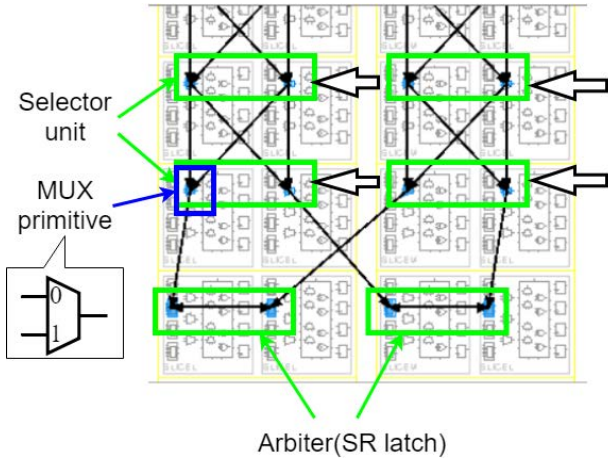

Figure 9. Outline of the implementation.

Xilinx PlanAhead v14.7. Figure 9 shows the outline of the implementation. Selectors are implemented by using MUX primitive. The selectors are arranged at regular intervals.

4.2 Experimental Method

Experiments consist of data acquisition phase, learning phase and test phase. Figure 10 shows the experimental method. In data acquisition phase, 60,000 types of *n*-bit challenge, which was randomly generated, is inputted to the 2-1 DAPUF, as shown in figure 10 (a). Then, 60,000 responses are acquired. The 50,000 CRPs is used as training CRPs and the 10,000 CRPs is used as test CRPs. Next, the learning phase generates learning model by training CRPs, as shown in figure 10 (b). At this time, experiments generate two models: model with feature extraction and model without feature extraction. The test phase, shown in Figure 10 (c), evaluates the accuracy of each model. For the evaluation, the model predicts the responses from the challenges of test CRPs. Then, the accuracy is calculated by comparing the predicted response and the response of test CRPs.

4.3 Experimental Results

Figures 11-14 show the experimental results. The vertical axis of figures shows accuracy and the horizontal axis of those shows the number of training CRPs. As shown in Figures 11-14, the model with feature extraction in 32 and 64 stages has high accuracy larger than the model without feature extraction in case of 50,000 CRPs on simulation and FPGA. Therefore, feature extraction can improve the accuracy of response prediction in modeling attack for 2-1 DAPUF.

5. Conclusion

This study evaluated the modeling attack using feature extraction on 2-1 DAPUF. The validity of the feature extraction was verified by performing experiments using not only simulation but also FPGA implementation.

The experiments indicate the modeling attack, which is optimized for arbiter PUF, is not suitable for 2-1 DAPUF. The future work includes the development of the optimal modeling method for 2-1 DAPUF to improve the prediction ratio of responses.

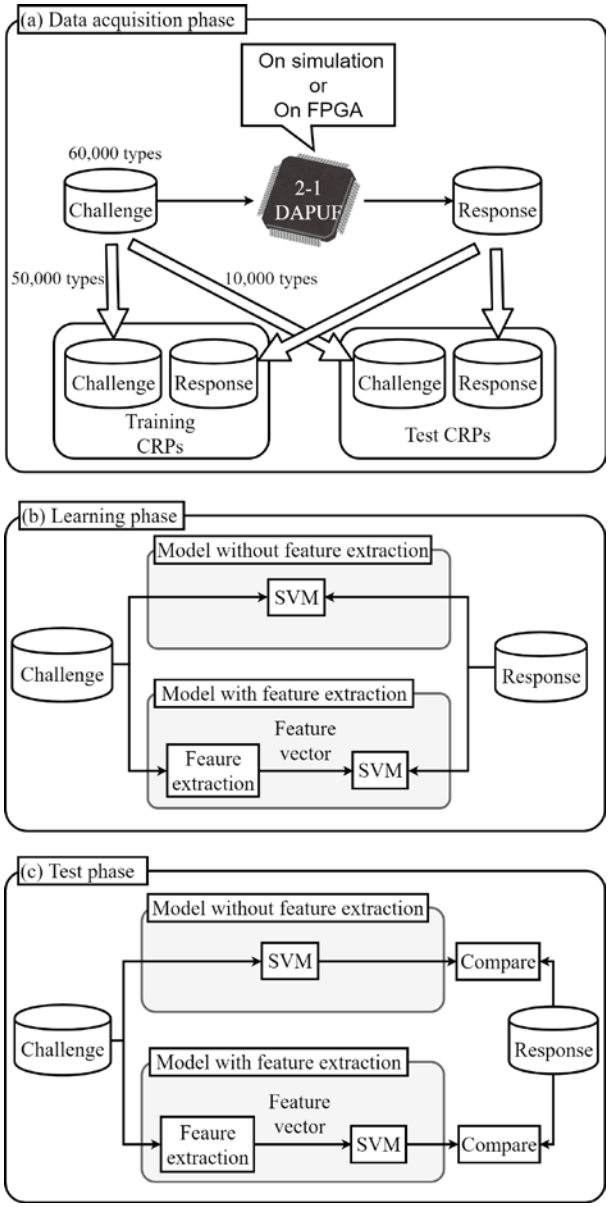

Figure 10. Experimental method.

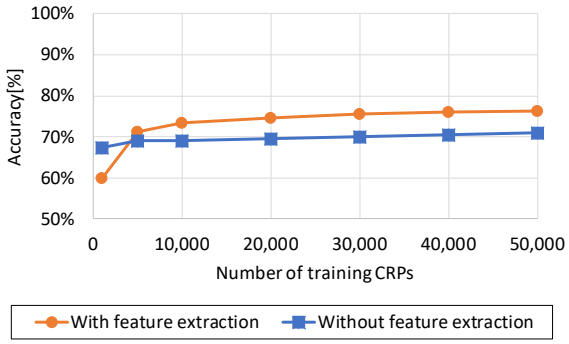

Figure 11. Experimental results for 32 stages 2-1 DAPUF on simulation.

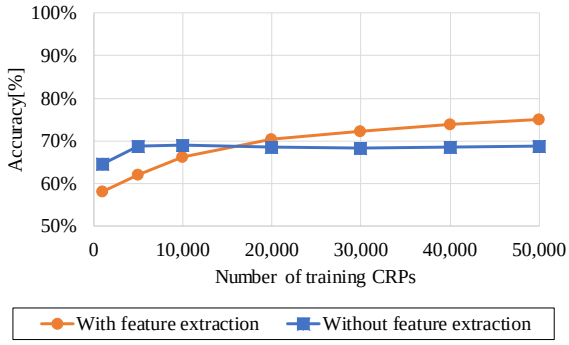

Figure 12. Experimental results for 64 stages 2-1 DAPUF on simulation.

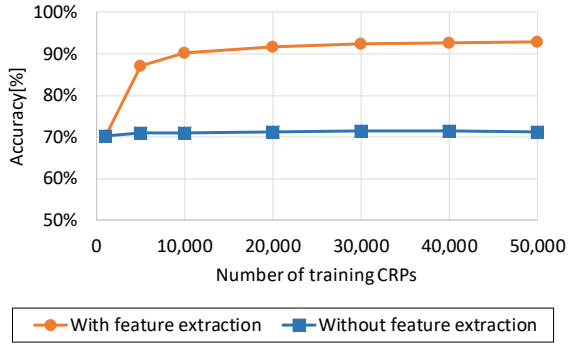

Figure 13. Experimental results for 32 stages 2-1 DAPUF on FPGA.

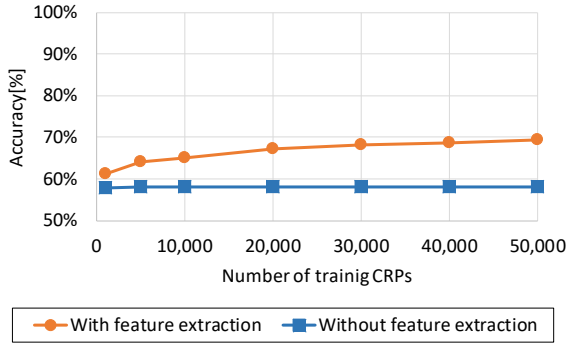

Figure 14. Experimental results for 64 stages 2-1 DAPUF on FPGA.

REFERENCES

[1] Pappu, R., Recht, B., Taylor, J., and Gershenfeld, N. 2001. Physical one-way functions. *Science* 297, 5589 (Sep. 2002), 2026-2030. DOI= http://dx.doi.org/10.1126/science.1074376.

[2] Maes, R., and Verbauwhede, I. 2010. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security* (Oct. 2010) Springer, Berlin, Heidelberg, 3-37. DOI= https://doi.org/10.1007/978-3-642-14452-3_1.

[3] Lim, D., Lee, J. W., Gassend, B., Suh, G. E., Van Dijk, M., and Devadas, S. 2005. Extracting secret keys from integrated circuits. *IEEE Trans. Very Large Scale Integration Systems* 13,10 (Oct. 2005), 1200-1205. DOI= https://doi.org/10.1109/TVLSI.2005.859470.

[4] Machida, T., Yamamoto, D., Iwamoto, M., and Sakiyama, K. 2014. A new mode of operation for arbiter PUF to improve uniqueness on FPGA. In *Proceedings of the Federated Conference on Computer Science and Information Systems* (Warsaw, Poland, September, 07 – 10, 2014). IEEE. 871–878. DOI= https://doi.org/10.15439/2014F140.

[5] Machida, T., Yamamoto, D., Iwamoto, M., and Sakiyama, K. 2015. A new arbiter PUF for enhancing unpredictability on FPGA. *The Scientific World Journal*, 2015. DOI= http://dx.doi.org/10.1155/2015/864812.

[6] Machida T., Yamamoto D., Iwamoto M., and Sakiyama, K., 2014. A study on uniqueness of arbiter PUF implemented on FPGA. in *Proceedings of the 31st Symposium on Cryptography and Information Security* (Kagoshima, Japan January, 21-24, 2014). SCIS ’14. 2014 (Japanese).

[7] Maiti, A., Gunreddy, V., and Schaumont, P. 2013. *A systematic method to evaluate and compare the performance of physical unclonable functions*. Embedded systems design with FPGAs 245-267. Springer, New York, NY. DOI= https://doi.org/10.1007/978-1-4614-1362-2_11.

[8] Hori, Y., Kang, H., Katashita, T., Satoh, A., Kawamura, S., and Kobara, K. 2014. Evaluation of physical unclonable functions for 28-nm process field-programmable gate arrays. *Journal of Information Processing*. J-STAGE, 22 (2014), 2, 344-356. DOI= https://doi.org/10.2197/ipsjjip.22.344.

[9] Yashiro, R., Machida, T., Iwamoto, M., and Sakiyama, K. 2016. Deep-learning-based security evaluation on authentication systems using arbiter PUF and its variants. *Advances in Information and Computer Security*. IWSEC 2016. Springer, Cham. 267-285. DOI= https://doi.org/10.1007/978-3-319-44524-3_16.

[10] Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., and Schmidhuber, J. 2010. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security* (Chicago, Illinois October 04 - 08, 2010). CCS `10. ACM, New York, NY, 237-249. DOI= https://doi.org/10.1145/1866307.1866335.