CycPUF: Cyclic Physical Unclonable Function

Michael Dominguez
Computer Engineering & Computer Science Department
California State University Long Beach
Long Beach, CA, USA
michael.dominguez01@student.csulb.edu

Amin Rezaei
Computer Engineering & Computer Science Department
California State University Long Beach
Long Beach, CA, USA
amin.rezaei@csulb.edu

***Abstract*—Physical Unclonable Functions (PUFs) leverage manufacturing process imperfections that cause propagation delay discrepancies for the signals traveling along these paths. While PUFs can be used for device authentication and chip-specific key generation, strong PUFs have been shown to be vulnerable to machine learning modeling attacks. Although there is an impression that combinational circuits must be designed without any loops, cyclic combinational circuits have been shown to increase design security against hardware intellectual property theft. In this paper, we introduce feedback signals into traditional delay-based PUF designs such as arbiter PUF, ring oscillator PUF, and butterfly PUF to give them a wider range of possible output behaviors and thus an edge against modeling attacks. Based on our analysis, cyclic PUFs produce responses that can be binary, steady-state, oscillating, or pseudo-random under fixed challenges. The proposed cyclic PUFs are implemented in field programmable gate arrays, and their power and area overhead, in addition to functional metrics, are reported compared with their traditional counterparts. The security gain of the proposed cyclic PUFs is also shown against state-of-the-art attacks.**

***Index Terms*—Hardware Security Primitives; Physical Unclonable Function; Cyclic Combinational Circuit; Modeling Attack**

I. Introduction

With an ever-increasing amount of sensitive data stored in electronic devices and the growing involvement of third-party manufacturing foundries, design automation companies, and testing facilities, hardware security is becoming an increasingly important issue in today's digital world. Physical Unclonable Functions (PUFs) [1] are hardware security primitives that ideally generate a unique device-dependent response to a particular input stimulus, such as a challenge or an interrogation signal. These responses are based on the inherent and uncontrollable physical variations that occur during the manufacturing process of Integrated Circuits (ICs). A weak PUF supports a small number of Challenge-Response Pairs (CRPs) with a polynomial function of the PUF size, while a strong PUF scales in such a way that it can support a very large number of CRPs with exponential growth with respect to the PUF size. In recent years, the use of PUFs in hardware security problems has gained popularity due to their ability to provide a unique identity for every chip [2], generate unpredictable hardware-based keys [3], [4], and facilitate privacy-friendly authentication protocols [5]. PUFs have been extensively studied in academia and have found their way into commercial applications [6]–[8], making them a promising solution for hardware security problems such as watermarking [9] and logic locking [10]–[13].

One of the main challenges in implementing PUFs is their vulnerability to modeling attacks [14]–[17]. In such attacks, adversaries usually use Machine Learning (ML) techniques to model the PUF based on a limited number of observed CRPs and predict its responses to new challenges, which poses a significant threat to the security of PUF-based systems. While researchers have developed hard-to-model PUF designs [18]–[21], their complexity imposes a high design overhead and has a negative effect on PUF functional metrics such as uniqueness, uniformity, and reliability.

Contrary to conventional wisdom, which believes that combinational circuits must be free of feedback loops, cyclic combinational circuits have been proposed for high-speed and low-power designs [22], [23], and their application in logic locking has been studied [24], [25] to improve security against oracle-guided attacks [26]. Inspired by these seminal works, in this paper, we propose CycPUF, a Cyclic Physical Unclonable Function framework, to safeguard against modeling attacks and enhance the PUF functional metrics with reasonable overhead. Our contributions are as follows:

- Proposing three delay-based CycPUFs and thoroughly analyzing their output behavior;
- Comparing CycPUFs functional metrics and overhead with their acyclic counterparts;
- Showcasing the security gain of CycPUFs against ML-based and hybrid modeling attacks.

A. Related Works

There are various types of traditional delay-based PUFs, such as Arbiter PUF (APUF) [27], Ring Oscillator PUF (ROPUF) [28], and Butterfly PUF (BPUF) [29]. Each type of PUF has its own unique properties and strengths, and the selection of the PUF category depends on the intended application and security requirements. For instance, ROPUFs rely on the frequency variations of multiple oscillators, while APUFs rely on the propagation delays of identical multiplexers [30]. A unified PUF that generalizes various existing PUFs has also been modeled [31].

One of the main attacks targeting the unpredictability of strong PUFs is the ML-based modeling attack, which has been shown to require only a linear or log-linear number of known CRPs to achieve high prediction accuracy on unknown CRPs [32]. An attacker may use the constructed PUF model to attack the same PUF instance they collected the CRPs from, or they

979-8-3503-4859-0/DATE24/© 2024 EDAA

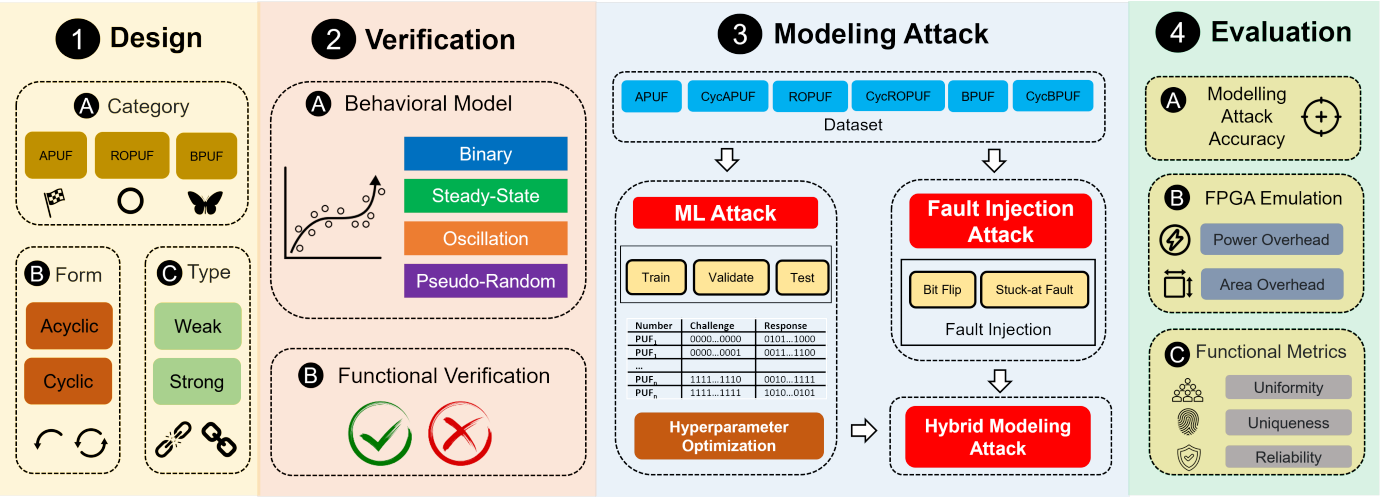


Fig. 1: CycPUF framework

may also use a sophisticated enough modeling attack to attack multiple instances of a PUF design [33]. It is even demonstrated that it is feasible to model not just delay-based PUFs but memory-based memristor PUFs [34] with high prediction rates [35]. We have witnessed significant advancements in defense strategies aimed at bolstering the resilience of PUFs against ML-based attacks. However, as time progresses, we expect ML to become more and more sophisticated, meaning that down the line, these PUF designs may also fall victim to modeling susceptibility.

A deception authentication protocol has been proposed by deceiving the adversary to use a training set dominated by invalid responses [18]. However, finding such a set for each instance of the device is a tedious task for the designer. While Interpose PUF (IPUF) [21] has been proposed as an anti-modeling primitive replacement for APUF, it was successfully modeled using a divide-and-conquer-based attack methodology [36]. In addition, structural unpredictability is exploited to reconfigure a conventional PUF into a noisy PUF and make it inherently unreliable and hence hard to model [19]. The downside of such an approach is the need to look for reliable CRPs or embed an Error Correction Code (ECC) module to make some pre-selected CRPs reliable. Linear-Feedback Shift Registers (LFSR) have been utilized for the purpose of obfuscating the CRPs of the proposed PUFs [37], [38], as well as using auxiliary PUFs to accomplish a similar goal [20]. Further, to hide the direct relationship between CRPs, a dual-mode PUF is introduced using a feedback structure to perform bitwise XOR of the challenge and its response and use the result as input to challenge the PUF again [39]. The method, however, seems to double the size of the original PUF.

B. Threat Model

We assume that the attacker has physical access to the IC and can apply a polynomial number of challenges to the PUF module to collect the corresponding responses. With the measured CRPs of the PUF in hand, the adversary tries to build a numerical model of the PUF using ML algorithms. In addition, the adversary may leverage physical access and knowledge of the PUF design to carefully inject faults to alter the PUF's functionality. The introduced faults, such as bit flips or stuck-at faults, may cause the PUF to generate erroneous responses that deviate from its anticipated behavior.

II. Preliminaries

The critical functional metrics needed to assess the functionality of a PUF are discussed here [40]. The Hamming Distance (HD) between two responses is defined as follows:

$$HD(R_1, R_2) = \sum_{i=1}^{n} (R_1[i] \oplus R_2[i]) \tag{1}$$

where R_1 and R_2 are two responses, and n is the number of bits in the response vector.

In addition, the Hamming Weight (HW) of one response is the sum of existing '1's in the response vector, as follows:

$$HW(R) = \sum_{i=1}^{n} R[i] \tag{2}$$

A. Uniqueness

Different PUF instances of the same design shall ideally generate different responses under the same challenge. This is what separates one PUF from another. Uniqueness is the metric used to highlight the differences between different PUF instances, which can be defined as the normalized inter-chip HD as follows:

$$Uniqueness = \left(\frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n}\right) \times 100\% \tag{3}$$

in which for any pair of different PUF instances, PUF_i and PUF_j, the responses produced by the involved PUFs are denoted by R_i and R_j respectively, and the number of PUF instances involved is given by the number k. The ideal uniqueness a PUF design can achieve is 50%.

B. Reliability

Ideally, PUFs would be able to operate in non-ideal conditions and still produce the same response for the same challenge that the PUF gets fed. Reliability is a metric used to identify a PUF's ability to reproduce the same response under the same challenge but under different operating conditions, such as different temperatures, altitudes, and supply voltages. Here, the intra-chip HD is used to measure reliability as follows:

$$Reliability = (1 - \frac{1}{s}\sum_{i=1}^{s}\frac{HD(R, R_i^*)}{n}) \times 100\% \quad (4)$$

where R is the n-bit reference response from a single PUF operating in standard conditions, and R_i^* is the response under the same challenge, collected under different operating conditions, and s is the total number of samples collected from varying the conditions of operation. A truly reliable PUF will achieve a reliability of 100%. This means that under varying operating conditions, the PUF can produce the same response under the same challenge.

C. Uniformity

The uniformity of a PUF can be measured as the normalized HW of all the responses produced by the PUF. The formula for uniformity is given as follows:

$$Uniformity = (\frac{1}{m}\sum_{i=1}^{m}\frac{HW(R_i)}{n}) \times 100\% \quad (5)$$

in which m is the number of responses produced by the PUF. The ideal uniformity a PUF design can achieve is 50%.

III. Cyclic PUF

The proposed CycPUF framework is depicted in Fig. 1.

1 While combinational feedback loops are generally not preferred when designing circuits, modifying acyclic PUF designs with feedback signals lends itself to interesting behaviors, which we shall refer to as *response modes*, that can prove useful in hardware security. The main idea behind constructing a CycPUF is to take a few bits from the response vector of a delay-based PUF and route them back into the challenge vector. Specifically, we choose three traditional PUF designs (i.e., APUF, ROPUF, and BPUF), select some random response bits and then XOR each with a random challenge bit, and route the output of the XOR gates back into the challenge input of PUFs. This allows us to hold the challenge vector constant while the response vector remains free to change at will. In this case, the users can choose between different PUF categories (i.e., APUF, ROPUF, and BPUF), PUF form (i.e., acyclic or cyclic), and PUF type (i.e., weak or strong) based on their needs.

2 Based on our analysis, the output of CycPUF can behave in four different response modes under a constant challenge.

1) Binary: In the binary response mode (shown in Fig. 2) the CycPUF behaves the same as its acyclic counterpart. Under a fixed challenge, the CycPUF will generate a fixed response depending on what delay variations appear in their symmetric paths for the duration that this challenge is applied.

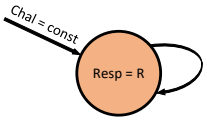


Fig. 2: CycPUF binary response mode

2) Steady-State: In the steady-state response mode (shown in Fig. 3), we see that inputting a fixed challenge will eventually produce a fixed response for the duration that the challenge is held constant. However, the fixed response may take some time to stabilize. This is to say that for some time, the CycPUF, while in this response mode, may produce different outputs before finally settling on a fixed response. This *cool-down* period distinguishes this behavior from a PUF's usual acyclic behavior since, in this scenario, the CycPUF response needs no additional helper code or hardware to maintain its steady-state response mode.

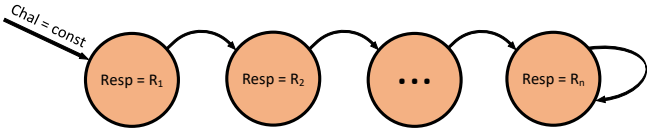


Fig. 3: CycPUF steady-state response mode

3) Oscillating: Oscillating outputs are also possible when using CycPUF (shown in Fig. 4). This oscillating response output can have varying periods for its output. It may take some intermediate responses before going back to a previously seen response. Following the same trend as the steady-state response mode's situation, we may also see a cool-down period before any oscillating outputs can be observed. When a fixed challenge is applied, the CycPUF may begin to produce its oscillating output after some time.

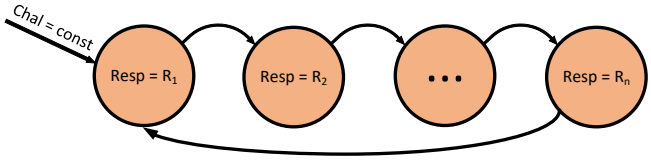


Fig. 4: CycPUF oscillating response mode

4) Pseudo-Random: The pseudo-random response mode is the one where applying a fixed challenge to the CycPUF produces response vectors with no discernible pattern (shown in Fig. 5). The steady-state and oscillating response modes may also produce seemingly random response vectors before finally collapsing to their appropriate behavior. However, the pseudo-random response mode does not follow a specific pattern, and it continuously generates separate responses for the duration of time that the challenge is held constant.

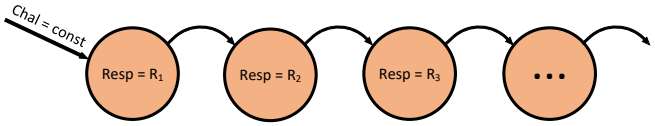


Fig. 5: CycPUF pseudo-random response mode

By cascading the delays of a PUF, the PUF is allowed to take on new possibilities that were otherwise thought to be unreliable. This gives rise to the term Challenge-Response Modes (CRMs), where instead of pairing up a single challenge with a single response, we now match a single challenge to the set of response vectors that the constant challenge may produce, where they will follow the behavior outlined in the given mode. Pairing a challenge in a CRM with each response vector in the set creates CRP-equivalents, where it should be noted that the challenges across these CRPs will overlap with each other.

Take the steady-state response mode, for example. In this CRM, we see that the output produces a response pattern to a fixed challenge rather than a single response vector. For modeling attacks that expect an acyclic, reliable response from a PUF, the introduction of the steady-state as a CRM would be able to poison the data that a ML algorithm can gather and render the attack unsuccessful. Take the oscillating response mode as another example. Since specific patterns will be repeated, this CRM can be a potential case for device authentication in a way that checking only a single challenge and storing the sequences of responses allows us to validate the authenticity of an IC later on. Last but not least, the pseudo-random CRM can be utilized to generate pseudo-random IC-specific keys by checking only one challenge vector.

❸ After designing and verifying acyclic and cyclic PUFs, we apply a ML-based model-building attack [15] to test the security of the traditional PUFs and CycPUFs. While the attack is primarily designed for APUFs, it can be easily extended to other delay-based PUFs and, in our case, with some modification, to CycPUFs since the training set is based on a polynomial number of observed CRPs. In addition, we mimic the behavior of the fault attack on PUFs [41] and insert stuck-at and bit-flip faults into the PUFs, create a faulty dataset, and run a hybrid modeling attack.

❹ Finally, we evaluate the proposed CycPUF framework by reporting the accuracy of the modeling attack and then emulating the PUFs' behavior on FPGA boards. It helps us practically measure the power and area overheads as well as extract the mentioned functional metrics in Section II.

IV. Experimental Results

We wrote a Python script to generate cyclic APUF, ROPUF, and BPUF in Verilog format based on the user-input challenge and response size and the number of feedback signals. Then, we created different CycPUFs and validated their Register Transfer Level (RTL) and post-implementation behavior via testbenches. After that, we evaluated the overhead, security, and functional metrics for the created CycPUFs and compared them with their acyclic counterparts. For running simulations, we used the Xilinx Vivado HL WebPack tool version 16.4 and implemented the design on the Nexys A7 FPGA board.

TABLE I: Modeling attack results

PUF Design	Challenge Size	# of Training CRPs	Model Accuracy
APUF	64	500,000	99.38%
CycAPUF	64	703,340	59.49%
Faulty CycAPUF	64	500,000	76.27%
ROPUF	64	500,000	78.00%
CycROPUF	64	1,048,576	48.74%
Faulty CycROPUF	64	642,603	50.02%
BPUF	64	500,000	83.32%
CycBPUF	64	938,420	54.76%
Faulty CycBPUF	64	582,645	61.44%

A. Security Evaluation

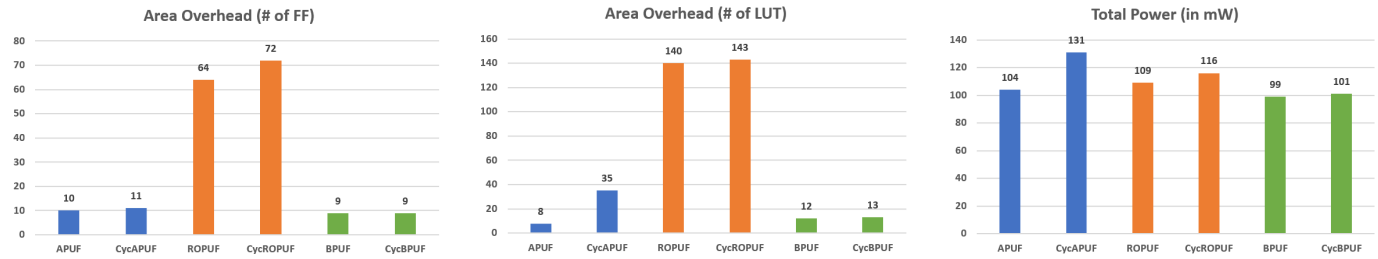
For security evaluation, we generated different strong PUFs with 64-bit challenge vectors and single-bit responses. This was done for compatibility between the PUF designs and the ML-based modeling attack used [15]. Each CycPUF design had a different number of feedback paths: 4 for the CycAPUF, 16 for the CycROPUF, and 12 for the CycBPUF. The CRP set was divided into two parts: the training set consisted of 80% of the total CRPs, while the test set consisted of the remaining 20%. We also ran a fault-injection modeling attack [41] by injecting a mix of stuck-at-0, stuck-at-1, and bit flip faults into the CycPUF designs and rerunning the ML-based modeling attack on the faulty CycPUFs. We injected 2 faults into the CycAPUF, 11 into the CycROPUF, and 7 into the CycBPUF. For the number of training CRPs, the acyclic PUFs all have 500,000, while the CycPUFs and faulty CycPUFs may possess a higher number of CRP-equivalents since, due to the cyclic behavior of the CycPUFs, they may generate multiple responses despite the challenge vectors being held constant.

The attack results are shown in Table I. Please note that the higher the model accuracy, the more successful the attack is at modeling the PUF. Comparing the modeling accuracy of the acyclic PUFs with the CycPUFs, we see that there are clear improvements in resistance against modeling attacks by CycPUFs. The CycAPUF and APUF had the most impressive discrepancies between each other, producing a difference in modeling accuracy of 40%. The CycROPUF and ROPUF had a model accuracy difference of 30%, while the CycBPUF and BPUF ended with a model accuracy difference of 28.5%. While injecting faults and running a hybrid attack can improve the modeling accuracy of the ML-based attack on CycPUFs, they still show better resistance compared with non-faulty acyclic ones.

B. Overhead Measurement

We generated weak cyclic and acyclic PUFs with 4-bit challenge and 4-bit response sizes and implemented them on the Nexys A7 FPGA board. The post-implementation overhead results are shown in Fig. 6.

For area overhead, we calculated the number of Look-Up Tables (LUTs) and Flip-Flops (FFs) utilized in the FPGA for



(a) (b) (c)

Fig. 6: Overhead analysis of CycPUFs and acyclic PUFs with 4-bit challenge and 4-bit response sizes (a) Number of FFs (b) Number of LUTs (c) Total power consumption

each PUF shown in Fig. 6a and Fig. 6b. It can be seen that CycAPUF exhibits significantly higher hardware complexity, with an approximate 3x increase in the number of LUTs and a 10% increase in the number of FFs compared to APUF. CycROPUF in contrast, approximately shows a 2% increase in LUTs and a 12.5% increase in FFs compared to ROPUF. CycBPUF maintains a relatively modest hardware overhead with an 8.3% increase in LUTs and no increase in FFs. In addition, CycPUFs tend to exhibit slightly higher power consumption compared to acyclic PUFs. As can be seen in Fig. 6c , the power consumption increase is 26%, 6.5% and 2% for CycAPUF, CycROPUF, and CycBPUF compared with their acyclic counterparts. Further, under the same challenge and response sizes, BPUF and CycBPUF generally consume less power than the others. The selection of a PUF design should carefully consider hardware overhead and power consumption trade-offs in relation to the specific constraints and requirements of the intended application.

C. Functional Metrics Analysis

Referring back to CycPUFs output behavior of Figs. 3, 4, and 5 in Section III, trying to apply the outlined functional metrics of Formulas 3, 4, and 5 in Section II would not be immediately possible since under a fix challenge, there may be a set of responses rather than one response for CycPUFs. Instead, we shall define a modification to these metrics so that we are able to compare the proposed CycPUFs to their acyclic counterparts in a meaningful way. We know that under a constant challenge, any response bit in the response vector can be at a logic '1' or a logic '0'. Averaging the number of cycles for which each bit is high or low allows us to apply the functional metrics to get an insight into how the CycPUFs will perform.

Let c be the number of clock cycles that a challenge is applied for, $R^i : i \in \{1, 2, ..., c\}$ be a response vector among the set of response vectors that appear during that time, and $r^i_j \in R^i : j \in \{1, 2, ..., n\}$ be a response bit in response vector R^i. Then, the Average Bit Value (ABV) can be defined as follows:

$$ABV(j) = \frac{1}{c} \sum_{i=1}^{c} r^i_j \tag{6}$$

TABLE II: PUF functional metrics results

PUF Design	Uniqueness	Uniformity	Reliability
APUF	7.55%	55.42%	99.77%
CycAPUF	47.10%	47.30%	98.34%
ROPUF	44.26%	53.81%	99.05%
CycROPUF	50.68%	52.12%	95.18%
BPUF	11.48%	55.04%	97.45%
CycBPUF	53.05%	46.67%	98.62%

If the resulting bit is greater than or equal to 0.5, then we say that on average we will observe a logic '1' from the output. Likewise, if the resulting bit is less than 0.5, we will observe a logic '0'. From here, we can apply ABV to the three previously defined PUF metrics in Section II.

The PUF functional metrics were computed and compared for the acyclic and cyclic PUFs using the same setup as in Section IV-B. The results are shown in Table II. As can be seen, while CycPUFs in general outperform their acyclic counterparts in uniqueness, they inherit the reasonable uniformity and reliability of their acyclic versions. As other papers have also previously highlighted [15], [40], [42], we confirmed that the FPGA-based APUF implementation exhibits relatively poor uniqueness. The intriguing fact is that CycAPUF fixes this issue and raises uniqueness to practically perfect levels. Comparing Tables I and II, it can be observed that improvement in the uniqueness metric has a positive effect on improving resistance against modeling attacks.

V. Conclusion

In this paper, we introduced CycPUF, a novel lightweight PUF generation framework featuring strong resistance against ML-based and hybrid modeling attacks while keeping up the PUF functional metrics. CycPUFs use response feedback loops to introduce the notion of challenge-response modes and poison the data an adversary might use for model training, introducing inaccurate correlations by erratically adjusting the response vector in every cycle. This improves the functional metrics of conventional PUF designs at the same time. The attack results confirm that CycPUFs achieve a high level of unpredictability against ML-based and hybrid modeling attacks. Additionally, CycPUFs were able to attain near-ideal levels of uniqueness, uniformity, and reliability. This work offers researchers new perspectives on approaching hardware security

problems with cyclic combinational circuits and looking into developing synthesis-friendly toolkits for them.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Award No. 2245247.

REFERENCES

[1] Y. Gao, S. F. Al-Sarawi, and D. Abbott, "Physical unclonable functions," *Nature Electronics*, vol. 3, pp. 81–91, 2020.

[2] W. Che, F. Saqib, and J. Plusquellic, "Puf-based authentication," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 337–344.

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

[4] R. Ueno, K. Kazumori, and N. Homma, "Rejection sampling schemes for extracting uniform distribution from biased pufs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 86–128, 2020.

[5] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, and M. Yung, "End-to-end design of a puf-based privacy preserving authentication protocol," in *Cryptographic Hardware and Embedded Systems (CHES)*, 2015, pp. 556–576.

[6] "Intrinsic id," https://www.intrinsic-id.com/, accessed: 2023-09-07.

[7] "Secure ic," https://www.secure-ic.com/, accessed: 2023-09-07.

[8] "Fungible inc." https://www.fungible.com/, accessed: 2023-09-07.

[9] S. Khalfaoui, J. Leneutre, A. Villard, I. Gazeau, J. Ma, J.-L. Danger, and P. Urien, "Water-puf: An insider threat resistant puf enrollment protocol based on machine learning watermarking," in *IEEE 20th International Symposium on Network Computing and Applications (NCA)*, 2021, pp. 1–10.

[10] A. Rezaei, A. Hedayatipour, H. Sayadi, M. Aliasgari, and H. Zhou, "Global attack and remedy on ic-specific logic encryption," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2022, pp. 145–148.

[11] A. Rezaei, R. Afsharmazayejani, and J. Maynard, "Evaluating the security of efpga-based redaction algorithms," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022.

[12] J. Maynard and A. Rezaei, "Dk lock: Dual key logic locking against oracle-guided attacks," in *International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–7.

[13] Y. Aghamohammadi and A. Rezaei, "Cola: Convolutional neural network model for secure low overhead logic locking assignment," in *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, 2023, pp. 339–344.

[14] S. Su, M. Zhu, H. Wang, B. Yang, and L. Liu, "A survey on the security of pufs," *Journal of Physics: Conference Series*, vol. 1993, no. 1, 2021.

[15] P. Santikellur, A. Bhattacharyay, and R. S. Chakraborty, "Deep learning based model building attacks on arbiter puf compositions," Cryptology ePrint Archive, Paper 2019/566, 2019.

[16] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong puf: Possible or a pipe dream?" in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 19–24.

[17] M. S. Mispan, B. Halak, and M. Zwolinski, "A survey on the susceptibility of pufs to invasive, semi-invasive and noninvasive attacks: Challenges and opportunities for future directions," *Journal of Circuits, Systems and Computers*, vol. 30, no. 11, 2021.

[18] C. Gu, C.-H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing lightweight-puf-based authentication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1183–1196, 2021.

[19] A. Wang, W. Tan, Y. Wen, and Y. Lao, "Nopuf: A novel puf design framework toward modeling attack resistant pufs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2508–2521, 2021.

[20] M. Ebrahimabadi, M. Younis, W. Lalouani, and N. Karimi, "A novel modeling-attack resilient arbiter-puf design," in *International Conference on VLSI Design and International Conference on Embedded Systems (VLSID)*, 2021, pp. 123–128.

[21] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Ruhrmair, and M. van Dijk, "The interpose puf: Secure puf design against state-of-the-art machine learning attacks," Cryptology ePrint Archive, Paper 2018/350.

[22] S. Malik, "Analysis of cyclic combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 7, pp. 950–956, 1994.

[23] M. D. Riedel and J. Bruck, "The synthesis of cyclic combinational circuits," in *ACM/IEEE Design Automation Conference (DAC)*, 2003, pp. 163–168.

[24] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 85–90.

[25] A. Rezaei, Y. Li, Y. Shen, S. Kong, , and H. Zhou, "Cycsat-unresolvable cyclic logic encryption using unreachable states," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2019, pp. 358–363.

[26] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[27] S. Hemavathy and V. S. K. Bhaaskaran, "Arbiter puf—a review of design, composition, and security aspects," *IEEE Access*, vol. 11, pp. 33 979–34 004, 2023.

[28] S. R. Sahoo, S. Kumar, and K. Mahapatra, "A novel ropuf for hardware security," in *International Symposium on VLSI Design and Test (VDAT)*, 2015, pp. 1–2.

[29] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly puf protecting ip on every fpga," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 67–70.

[30] Y. Wang, C. Wang, C. Gu, Y. Cui, M. O'Neill, and W. Liu, "Theoretical analysis of delay-based pufs and design strategies for improvement," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[31] F. Armknecht, D. Moriyama, A.-R. Sadeghi, and M. Yung, "Towards a unified security model for physically unclonable functions," in *Topics in Cryptology - CT-RSA*, 2016, pp. 271–287.

[32] U. Ruhrmair, F. Sehnke, J. Solter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *ACM Conference on Computer and Communications Security (CCS)*, 2010, pp. 237–249.

[33] T. Kroeger, W. Cheng, S. Guilley, J.-L. Danger, and N. Karimi, "Assessment and mitigation of power side-channel-based cross-puf attacks on arbiter-pufs and their derivatives," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 2, pp. 187–200, 2022.

[34] P. Koeberl, U. Kocabas, and A.-R. Sadeghi, "Memristor pufs: A new generation of memory-based physically unclonable functions," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 428–431.

[35] S. Zeitouni, E. Stapf, H. Fereidooni, and A.-R. Sadeghi, "On the security of strong memristor-based physically unclonable functions," in *ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[36] N. Wisiol, C. Muhl, N. Pirnay, P. H. Nguyen, M. Margraf, J.-P. Seifert, M. van Dijk, and U. Ruhrmair, "Splitting the interpose puf: A novel modeling attack strategy," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 97–120, 2020.

[37] Z. Chang, S. Shi, B. Song, W. Fan, and Y. Wang, "Modeling attack resistant arbiter puf with time-variant obfuscation scheme," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2021, pp. 60–63.

[38] L. Wu, Y. Hu, K. Zhang, W. Li, X. Xu, and W. Chang, "Flam-puf: A response–feedback-based lightweight anti-machine-learning-attack puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4433–4444, 2022.

[39] Q. Wang, M. Gao, and G. Qu, "A machine learning attack resistant dual-mode puf," in *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, p. 177–182.

[40] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds., 2013, pp. 245–267.

[41] S. Tajik, H. Lohrke, F. Ganji, J.-P. Seifert, and C. Boit, "Laser fault attack on physically unclonable functions," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2015, pp. 85–96.

[42] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen, "A multiplexer-based arbiter puf composition with enhanced reliability and security," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 403–417, 2018.