Experimental Study of Component-Differentially-Challenged XOR PUFs as Security Primitives for Internet-of-Things

Khalid T. Mursi[1, 2] and Yu Zhuang[1]
[1] Department of Computer Science, Texas Tech University, Lubbock, Texas, USA
[2] College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia
Email: {khalid.mursi, yu.zhuang}@ttu.edu; kmursi@uj.edu.sa

***Abstract*—**Security is critically important for Internet-of-Things, but existing cryptographic protocols are not lightweight enough for resource-constrained IoT devices. Implementable with simplistic circuits and operable with shallow power, physical unclonable functions (PUFs) leverage small but unavoidable physical variations of the circuit to produce unique responses for individual PUF instances, rendering themselves good candidates as security primitives for IoT devices. Component-differentially-challenged XOR PUFs (CDC XPUFs) are among the PUFs which were shown to be highly secure to machine learning modeling attacks. However, no study of implementation and experimentation has been carried out. In this paper, we report our implementations of CDC XPUFs on FPGAs and experimental studies of the essential properties of CDC XPUFs.

Index Terms—Resource-constrained IoT, IoT security, XOR PUF, FPGA

I. Introduction

Internet of Things (IoT) is gaining momentum with the fast increases of communication-capable devices and applications involving such devices. Along with the explosive growth of IoT devices, communication security become a critically important issue. Many IoT devices are resource-constrained and operate under a limited supply of power, calling for lightweight security mechanisms of low power consumption.

Popular cryptographic key-based security protocols are not lightweight in demanding resources, as pointed out by [1], [2]. In addition, all key-based protocols require storing the keys in non-volatile memories, and all stored data can be exposed by side-channel attacks [3]-[8], especially when attackers are within close physical distances, which are very likely the case for many IoT devices due to their physical closeness to the crowd.

Physical Unclonable Functions (PUFs), introduced in [9], [10], provide a potential alternative to classical cryptography mechanisms. PUFs harvest the integrated circuits' internal variations to produce unique responses from different PUF instances even if they have exactly the same circuit design and going through the same production process by the same manufacturer. Such inner variations are the chips' fingerprints, which are utilized by PUFs to produce device-dependent responses to the same challenge inputs. Due to their capability to give different responses from different PUF instances even when supplied with the same inputs, PUFs can be used in authentication protocols, to generate secret keys [11], [12], or as the source for True Random Number Generators (TRNG) [1]. Also, implementable with thousands of transistors and operable with extremely low power, PUFs are high suitable for resource-constrained devices.

Before choosing a PUF for a security application, it is essential to have a proper examination of all important properties of the PUF to see if the PUF can deliver desired functionalities to meet the peculiar needs of the application. Must-examines include PUF fingerprint property (Sec. III.A), reliability (Sec. III.B), PUF entropy property (Sec. III.C), and resistance to machine learning attacks [13]-[17]. Information on such properties of a PUF is useful for security application developers to make a decision on the selection of a PUF for an application.

Machine learning (ML) attacks can be launched towards a PUF when an attacker has eavesdropped adequate communication data. During a machine learning attack, the attacker sniffs the ongoing packets on-air between a PUF and its trusted partner, and the challenge-response-pairs (CRPs) used between the PUF and its partner for authentication can be collected as training data to build an ML model for predicting the response of the PUF to any future input challenge. Studies [18], [1] showed that component-differentially-challenged XOR PUFs (CDC XPUFs) is one of the most secure PUFs against machine learning attacks. However, to the best of our knowledge, there has been no study on other important properties of the CDC XPUF. The high attack-resistance of CDC XPUFs makes them potential candidates to be adopted for secure applications, and it is thus important to have a good examination of the other essential properties of the CDC XPUF.

The remainder of this paper is organized as follows. Section II describes the arbiter PUF, the XOR PUFs, and CDC XPUFs. Section III presents the evaluation metrics used in this work to evaluate the CDC XPUFs and XOR PUFs. Section IV presents the implementation and

Manuscript received April 5, 2020; revised September 1, 2020.
doi:10.12720/jcm.15.10.714-721

experimentation of the PUFs on FPGAs and the results and discussions of our experimental studies, and Section V gives the conclusion.

II. Background Information on PUFs

A CDC XPUF, similar to an XOR PUF, consists of multiple arbiter PUFs whose outputs are XORed to produce the final response of the CDC XPUF. A CDC XPUF differs with an XOR PUF only in that different component arbiter PUFs of a CDC XPUF receive different challenges while all component arbiter PUFs of an XOR PUF receive the same challenge. In this section, we briefly describe the arbiter PUF, the XOR PUF, and the CDC XPUF.

A. The Arbiter PUF

Fig. 1 is an illustration of an arbiter PUF [10], [19]. A *k*-bit arbiter PUF consists of *k* stages, where each stage has two 2-to-1 multiplexers (MUXs). Signal enters the arbiter PUF from stage 1 and splits into two signals, which propagate through two paths determined the challenge bits to the multiplexers at each stage and finally reach the D flip-flop acting as an arbiter to decide if the signal on the top path or lower path arrived first. If the top path signal arrives earlier, the D flip-flop returns 1; otherwise, it returns 0.

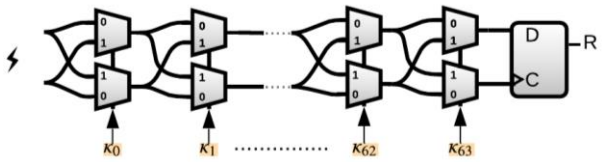


Fig. 1. An arbiter-PUF with 64-bit of challenges.

The response of an arbiter is determined by the difference between the signal delays of the two paths, and the delays at the final stage before entering the arbiter is the result of the accumulation of delays at all stages the two signals pass through. This observation leads to what is called the additive delays model, from which a linear classification model for the challenge-response relation of arbiter PUF [13] is established. With the linear classification model, when an adequate number of challenge-response pairs (CRPs) are obtained by attackers through eavesdropping or other means, an arbiter PUF can be "cloned" though mathematical modeling [15], [20] to produce the same responses the PUF would give, thus enabling the development of malicious software that uses such modeling to reproduce the response of the arbiter PUF.

B. The XOR Arbiter PUF

Due to arbiter PUFs' weak resistance to ML modeling attacks, a new PUF was proposed in [11] with increased nonlinearity by XORing a multiple arbiter PUFs to produce the final response. This type of PUF is known as the XOR arbiter PUF or XOR PUF for short. Fig. 2 illustrates a 3-XOR PUF. An *n*-XOR PUF consists of *n* component arbiter PUFs (streams) in which the responses of all *n* component arbiter PUFs are XORed to produce one response. Note that an XOR PUF's component arbiter PUFs are fed using the same challenge.

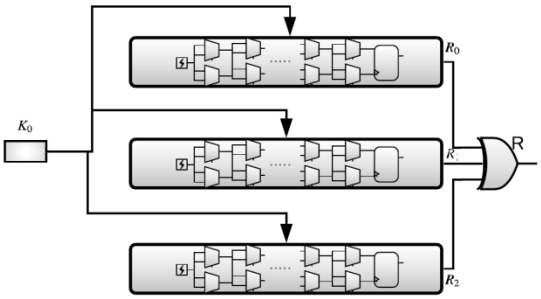


Fig. 2. A 3-XOR PUF, which consists of three arbiter PUFs whose outputs are XORed to generate the final response of the XOR PUF.

Studies [20]-[23] showed that XOR PUFs are more resistant to modeling attacks than arbiter PUFs. When equipped with mutual authentication like lockdown scheme [1] to eliminate open-access interface, XOR PUFs of 64 or more stages with nine or more component arbiter PUFs, all modeling attacks developed so far were not able to reproduce the responses of an XOR PUF with adequate accuracy. Nevertheless, for 64-bit XOR PUFs with eight or fewer component arbiter PUFs, studies [20-24] showed that there are attack methods that can predict the responses of such PUFs with prediction accuracy around 98%. Also reported in [25] were comprehensive tests on the XOR PUF using 1 trillion 32nm hardware CRPs with the conclusion that 64-bit 10-XOR PUFs are secure and stable PUFs. However, extending the number of streams and challenge length will raise the cost and power consumption of a PUF, which is an important issue for resource-constrained IoT devices [26] like inexpensive RFID tags.

C. The CDC XPUF

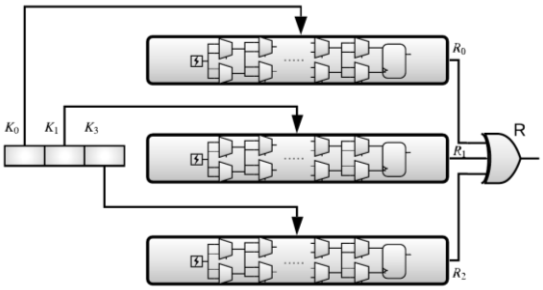


Fig. 3. A 3-CDC XPUF, which consists of three arbiter PUFs XORed to produce the final response. In CDC XPUF, different component arbiter PUFs receive different challenges.

CDC XPUF differs from XOR PUF only in that its component arbiter PUFs receive different challenges while the XOR PUF receives the same challenge for all its component arbiter PUFs. Studies [1, 18, 27] have found that applying different challenges to different components of an XOR PUF improves the resilience of the PUF against machine learning attacks, and existing modeling-based attack method for 64-bit CDC XPUFs with four or more components can attain success attack rate lower than 90% even if using 1 or 2 million CRPs

(see [18, Fig. 3, 6]) and can attain probably only 0.1% success rate if using only 30 thousand CRPs (see [18, Table. II]). These studies showed that CDC XPUFs is a potentially good candidate in terms of security performance. Thus, it is worthwhile to see if they are also good at other properties for real applications.

III. Performance Evaluation Metrics

Before confide on using a PUF as a reliable source for authentication, this PUF design needs to pass specific performance tests [28]-[30]. In this section, we list the evaluation metrics that we used to judge PUF designs in terms of performance.

A. Measuring the PUF Fingerprint Property

One of the major properties, when the PUF was introduced in 2002, is its ability to produce different responses from different devices when the same challenge is given. This property was later known as chip "fingerprint" [2] or uniqueness. In order to evaluate the CDC XPUF uniqueness, we generated CRPs from different FPGAs to the same set of challenges. We used the uniqueness that proposed by Hori *et al.* [3] and Maiti *et al.* [33], [34].

1) Maiti's Uniqueness: Maiti's uniqueness (*MU*) introduced in *[4]* evaluates how different the output of a PUF compared to other outputs of other devices using Hamming Distance (*HD*). The uniqueness of responses of *N* devices of the k_{th} challenge is calculated as follow:

$$MU_k = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\frac{\mathrm{HD}(ID_i,\ ID_j)}{L} \right), \quad (1)$$

where *L* is the number of bits in a response set, which is 128-bit for our experiment, and the *ID* is the response set. Equation 1 applies a bitwise HD for responses' bits of the i_{th} device with its corresponding bits, of the same challenge, in the j_{th} chip.

2) Hori's Uniqueness: On the other hand, Hori's uniqueness (HU), introduced in *[3]*, calculates the uniqueness of responses of the same PUF design and challenges but from different devices. Hori's uniqueness is defined as follows:

$$HU_k = \frac{4}{N_r \times N^2} \sum_{i=1}^{N_r} \sum_{j,m=1, j \neq m}^{N} \left(b_{j,i} \oplus b_{m,i} \right), \quad (2)$$

where *N* is the number of chips, N_r is the response length, and $b_{j,i}$ and $b_{k,i}$ are the i_{th} response bit from the j_{th} and m_{th} PUF instances, respectively.

Both *HU* in equation (2) and *MU* in (1) serve the same purpose but only differ in the normalization value. *MU* and *HU* need to be calculated for every challenge in the dataset, say *K* challenges, and then averaged as follows:

$$MU = \frac{1}{K} \sum_{i=1}^{K} MU_k \ \ or \ \ HU = \frac{1}{K} \sum_{i=1}^{K} HU_k . \quad (3)$$

B. Measuring the PUF Reliability

PUF outputs are expected to be persistent. However, many conditions and circumstances may affect the design reliability, such as aging, heating, and the voltage level of the input signals. To measure how correct are PUF's outputs, we used two metrics introduce by Hori et al. in [3], which they called them Steadiness and Correctness.

1) Correctness: The Correctness, introduced in *[3]*, is the study of response reliability when a PUF design is subjected to outward conditions. Correctness uses Fractional Hamming Distance (FHD) in order to record the changes that were applied to the response among iterations. For each iteration, the voltage level of the signal will differ as long as the device temperature. Correctness is defined as follows:

$$C = 1 - \frac{2}{N_c \times N_a} \sum_{k=1}^{N_c} \sum_{j=1}^{N_a} (b_k \oplus b_{k,j}), \quad (4)$$

where N_c is the total number of challenges, *k* is the challenge ID, N_a is the total number of repeated responses of a challenge, and *b* is a response bit.

2) Steadiness: An ideal PUF is expected to output the same response when it is given a single challenge on the same chip. Thus, the study of a PUF reliability in terms of the design output is needed. The steadiness (*S*), as defined in equation 5, measures if the response bit *b* has changed or lasts the same amongst N_a times of the same challenges. The steadiness is defined as follows:

$$S = 1 + \frac{1}{N_c} \sum_{k=1}^{N_c} \log_2 max \left\{ \frac{\sum_{j=1}^{N_a} b_{k,j}}{N_a}, 1 - \frac{\sum_{j=1}^{N_a} b_{k,j}}{N_a} \right\}, \quad (5)$$

where N_c denotes the total number of challenges, and *k* is a single challenge of the j_{th} response bit.

C. Measuring the PUF Entropy

In terms of reaching the unpredictability of PUF's outputs, each design is expected to produce different responses when given different challenges. If PUF's outputs are biased to 0 or 1, it will be easy for the attacker to guess the responses with a more prediction rate. Thus, evaluating PUFs responses in terms of randomness and diffuseness, introduced in [3], is essential for guaranteeing the needed authentication and keys complexity levels when using PUFs in security applications.

1) Randomness: Randomness is the study of PUF response balance in terms of 0's and 1's. The ideal PUF is expected to produce responses that have a balance of 0's and 1's by 50% for each when inputting different challenges to the same chip. We can calculate the frequency of *1*'s in responses of a chip for different challenges as follows:

$$p = \frac{1}{N_r} \sum_{i=1}^{N_r} (b_i), \quad (6)$$

where N_r is the total number of responses, and b_i is the i_{th} response bit. Then, randomness (*H*) can be calculated as follows:

$$H = -\log_2 max(p, 1-p). \quad (7)$$

2) Diffuseness: Diffuseness is the study of how dispersed the responses are among each other when

given different challenges on the same device. Diffuseness is important for showing the disparity of responses which form a device identity. Fractional Hamming Distance is used for all responses pairs to sum the diffuseness as follows:

$$D = \frac{4}{K^2 \times L} \sum_{l=1}^{L} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} (b_{i,l} \oplus b_{j,l}), \quad (8)$$

where L is the number of bits is a response and K is the number of responses.

IV. Experimental Setup and Results

A. Generating Silicon CRPs from FPGAs

In our experiments, we programmed the different CDC XPUFs and XOR PUFs on Xilinx Artix®-7 FPGA 28-nm, which includes a configurable MicroBlaze CPU. The experiments were replicated three times on three chips to allow capturing and evaluating the responses' uniqueness. VHSIC Hardware Description Language (VHDL) was used to construct the CDC XPUFs and XOR PUFs designs on Xilinx Vivado 15.4 HL Design Edition. Each design consists of sub-components such as MUXs, arbiters, and an XOR gate, which they are connected to form a CDC XPUF or XOR PUF.

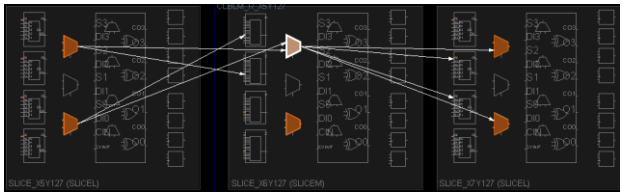


Fig. 4. Illustration of a synthesized CDC XPUF after horizontal placement was applied using TCL

The CDC XPUFs placement in the hardware was done horizontally using Tool Command Language (TCL), as shown in Fig. 4. For all designs, the placement starts from slice X0Y100 on the device to contain only two MUXs, top and bottom. Then, we iterate the placement's command for each stage to slice X32Y100, which is the last location for finishing the writing of the first half of the first stream. The second half of the same stream will be placed right above the first half, which starts from location X0Y101 and ends in X32Y101. Similarly, we place the other components, streams 2, 3, etc, above the current APUF until all streams are installed. Continuously, we place the arbiter of each stram right after the last slice of the component's second half, on slice X33Y101 for the above example. Finally, all arbiters will input to one XOR gate, and the output of the XOR gate represents the PUF's response.

For interacting with the PUFs, we use Xilinx SDK to transfer the random challenges to the CDC XPUFs and receive the corresponding response of each written challenge. The code of the Xilinx SDK was written using *C* language.

For generating the CRPs, eight AXI General Purpose Input/Output (GPIO) interfaces were used as follows: one GPIO for submitting the initial traveling signals, one for receiving the output response, and six for feeding the CDC XPUF with the generated challenges. Each GPIO can handle 32 + 32 bits, and our maximum design, 6-CDC XPUF 64-bit, needs eight GPIOs, for instance. For creating a more complex CDC XPUF designs such as 6-CDC XPUF 128-bit, we need to implement 14 GPIOs to handle the needed operations for this design. The required GPIOs for the 6-CDC XPUF 128-bit are one GPIO for the initial signals, one GPIO for receiving the response, and 12 GPIOs for inputting the challenges into the CDC XPUF. The reason behind the necessity for this number of GPIOs for implementing the 6-CDC XPUF 128-bit design is we need one GPIOs for each stream in which we input (32+32)||(32+32) bits of challenges, 6 × 128 bits as total challenges input. This complexity can be avoided, as explained in [18], by using the Permutation-Based PUF, which as resistance as the CDC XPUF against ML attacks. On the other hand, XOR PUF implementation is much easier than implementing the CDC XPUF. We need only three GPIOs for creating any XOR PUF design of 64-bit. Those GPIOs are one for the signals, one for the response, and one for the challenge since this challenge will input in all streams as is.

Our challenges were generated using the Pseudo-Random Number Generator (PRNG) as follows:

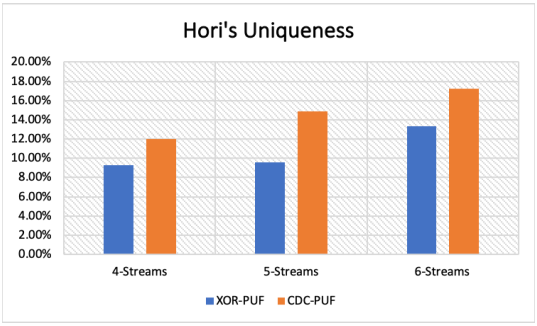
$$C_{n+1} = (a \times C_n + g) \, mod \, m, \quad (9)$$

where C is the sequence of the generated random number, a is a multiplier, g is a given constant, and m is 2^K where K is the number of stages. To speed up the data transfer between the PUFs and computer, AXI Universal Asynchronous Receiver Transmitter (UART) was installed with baud rate equals to 230,400 bits/second. Finally, Tera Term, which is a terminal emulator program, used for printing and saving the device's outputs.

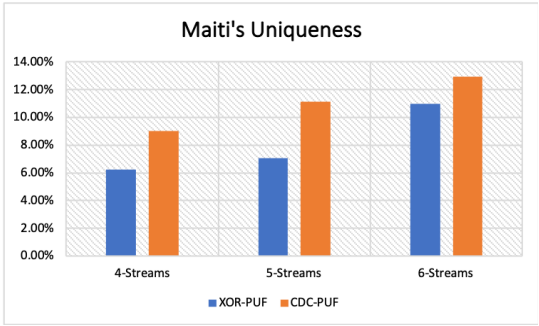
We have implemented a different number of components of CDC XPUF and XOR PUF ranged from 4 to 6 components and stages length of 64-bit. For the performance evaluation CRPs, we generated up-to 16,384 challenges × 16 iterations × 128 response length × 3 devices × 2 designs, CDC and XOR PUFs. The repetition of the CRPs is needed during the performance evaluation to study the related metrics such as randomness, steadiness, correctness, and diffuseness. The used silicon CRPs of this study were generated at ambient temperature of approximately 26 ℃, and core voltage set to 1.0V using the built-in chips' resistor.

B. Results of the Performance Evaluation

Fig. 5, Fig. 6, and Fig. 7 show the performance evaluation results of the state-of-the-art CDC XPUF based on the explained evaluation metrics in the previous section. As shown in the figures, we evaluated the 4, 5, and 6 CDC XPUFs with 64-bit of challenge length and compared them with their equivalents structure in XOR PUFs. In all bar graphs, the blue bars represent the performance evaluation of the XOR PUFs, while the orange bars are assigned for the CDC XPUFs.

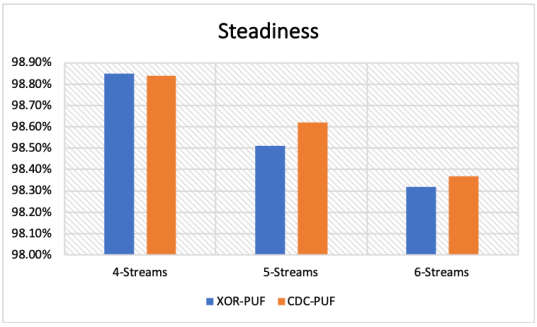


(A) Uniquiness of XOR PUFs and CDC XPUFs(Hori)

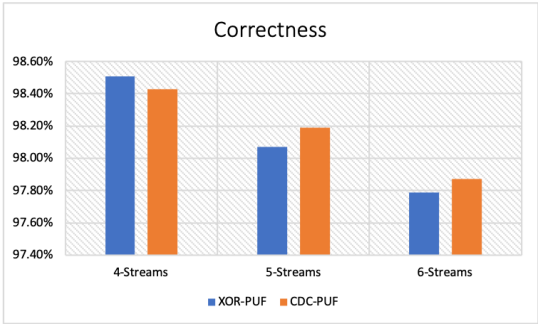


(B) Uniquiness of XOR PUFs and CDC XPUFs (Maiti)

Fig. 5. Experimental data in PUF fingerprint metrics. The ideal value of Hori's equation is 100% while Maiti's is 50%.

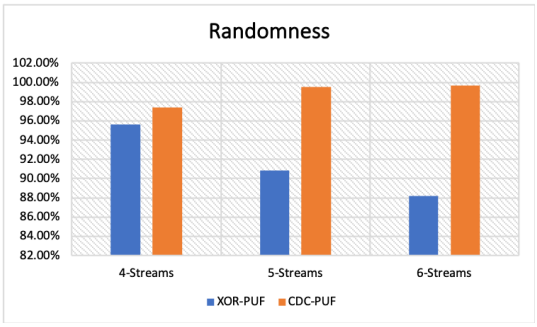


(A) Steadiness of XOR PUFs and CDC XPUFs

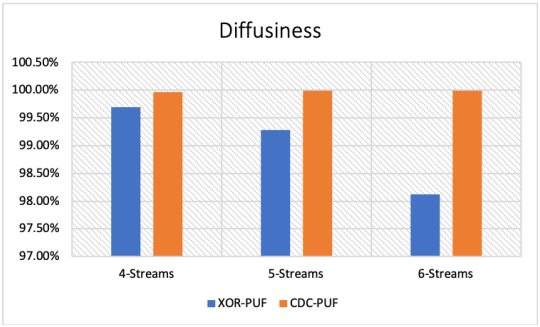


(B) Correctness of XOR PUFs and CDC XPUFs

Fig. 6. Reliability of CDC XPUF (Steadiness and Correctness). The ideal value of the Steadiness and Correctness is 100%.



(A) Randomness of XOR PUFs and CDC XPUFs



(B) Diffusiness of XOR PUFs and CDC XPUFs

Fig. 7. Experimental data in PUF entropy metrics, including Randomness and Diffuseness. The ideal value of the Randomness and Diffuseness is 100%.

1) Uniqueness Results: Fig. 5 demonstrates the uniqueness results of the CDC XPUF and XOR PUF, which were calculated using Hori's and Maiti's calculations, sub-figures (A) and (B) respectively. Both equations serve the same purpose but differ in normalization value. The ideal value of the Hori's calculation is 100%, while 50% for Maiti's equation. Overall it is clear that the CDC XPUF shows a better uniqueness compared to the XOR PUF, but it is still considered a low score for the highest one in sub-figure (A), 17%. Nevertheless, when considering a vast range of possible challenges such as $2^{64\times6}$ for the 6-CDC XPUF 64-bit, 17% of possible unique CRPs is relatively large and enough to confide the CDC XPUF in security applications, especially when comparing this number to the 6-XOR PUF 64-bit which has a much lower range of possible CRPs, ranged from 0 to 2^{64}. Among all the results of Fig. 5's sub-figures, the effect of increasing the number of components is clear. PUFs' uniqueness increases as the PUFs' number of streams raise.

2) Steadiness and Correctness Results: Fig. 6 illustrates the results of measuring the CDC XPUF reliability, the difference of responses of the same challenge from the same chip. The steadiness bar-graph, sub-figure (A), is shown on the left side of the figure while the correctness, sub-figures (B), is on the right side. The CDC XPUF shows a high steadiness between 98% and 99% of responses when repeating the responses of the same signal voltage and device temperature. Also, when comparing the CDC XPUFs steadiness to the XOR-PUFs, CDC XPUFs show a slightly better steadiness for all of the recorded components. On the other hand, when the device experience outer conditions, such as the change of the chip temperature, the correctness is slightly affected as shown in the correctness bars. The CDC XPUFs and XOR PUFs' correctness is gradually decreased when we increase the PUF's number of streams. Nevertheless, the overall steadiness and correctness of the CDC XPUF score better results compared to the XOR PUF except for a small number of components such as the 4-CDC XPUF.

3) Randomness and Diffuseness Results: Fig. 7 represents the results of calculating the difference of the CDC XPUFs' responses of different challenges in the same device, equations (7) and (8). Sub-figure (A) shows the results of the randomness, while the right sub-figure (B) is for the diffuseness. It can be found from the figure that CDC XPUFs have better randomness than XOR PUFs. Also, in CDC XPUFs, the randomness is enhanced as we increase the number of components while it is affected negatively in the XOR PUFs. The randomness scores for CDC XPUF are ranged from 97.5% to almost 100%, while the XOR PUF's randomness scores are between 95% and 88% approximately. On the other hand, diffuseness scores good results for both CDC XPUFs and XOR PUFs ranged from 98% to 100%. However, as the results of the randomness, the XOR PUF scores gradually decrease as we increase the number of streams.

V. Conclusion

In this paper, we have examined the CDC XPUFs' performance in various properties and compared them with the XOR PUF using evaluation metrics, including randomness, diffuseness, steadiness, correctness, and uniqueness. The CDC XPUF was shown to be highly secure against ML attacks in several studies [1, 18, 27], but there has been no study that evaluates the CDC XPUF in terms of other important properties needed before real applications, and this paper is filling in this void by implementing CDC XPUFs on FPGAs and carrying out experiments with more than 600 million CRPs out of three Xilinx Artix®-7 FPGAs.

Experimental study shows that when compared with XOR PUFs of the same circuit complexity in terms of the number of stages and number of component arbiter PUFs, CDC XPUFs have similar results in reliability metrics, but have higher values in PUF fingerprint metrics and entropy metrics. PUF fingerprint metrics are indicators of how many different devices such PUFs can represent or identify while PUF entropy metrics are indicators for how many different keys or identification numbers a PUF can generate. Since the challenge to a CDC *k*-XPUF if of *k* times the length of a *k*-XOR PUF of the same number of stages, CDC XPUF can uniquely represent trillions of trillions times more devices than an XOR PUF of the same stages can represent, assuming the same uniqueness rate, since a 64-stage CDC 4-XPUF even with 9% Maiti Uniqueness can represent close to O(0.09 devices while a 64-stage 4-XOR PUF with the perfect 50% Maiti Uniqueness can represent no more than O(0.5 devices. Similarly, it is true for the number of different keys a CDC XPUF can generate when compared with the number of different keys an XOR PUF of perfect PUF entropy metrics can generate. Thus, the experimental results show that the experimentally examined propertied of CDC XPUFs are up to the tasks of real IoT applications.

Acknowledgement

Khalid T. Mursi is supported by University of Jeddah (UJ) and the Saudi Arabian Cultural Mission (SACM). This research was supported partially by the National Science Foundation under Grant No. CNS-1526055.

This paper is a further development of the work [40]

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

This work is done by Dr. Yu Zhuang (Y.Z.) and Khalid T. Mursi (K.M.) as follows: Methodology, K.M and Y.Z.; software-performance evaluation programming, K.M.; software-CRPs generator programming, Y.Z. and K.M.; validation, Y.Z; Hardware-generating silicon CRPs, K.M.; Writing—original draft, K.M. Writing—reviewing and editing, K.M. and Y.Z.; All authors have read and agreed to the published version of the manuscript.

References

[1] M. D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, pp. 146-159, 2016.

[2] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, pp. 1126-1141, 2014.

[3] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+ Flush: a fast and stealthy cache attack," in *Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016.

[4] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," *Smartcard*, vol. 99, pp. 9-20, 1999.

[5] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of AES," in *Proc. Cryptographers' Track at the RSA Conference*, 2006.

[6] U. Rührmair and D. E. Holcomb, "PUFs at a glance," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014.

[7] S. P. Skorobogatov, "Semi-invasive attacks: A new approach to hardware security analysis," 2005.

[8] Y. Yarom and K. Falkner, "FLUSH+ RELOAD: A high resolution, low noise, L3 cache side-channel attack," in *Proc. 23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014.

[9] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conference on Computer and Communications Security*, 2002.

[10] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. 18th Annual Computer Security Applications Conference*, 2002.

[11] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE Design Automation Conference*, 2007.

[12] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *Proc. IEEE Symposium on Security and Privacy Workshops*, 2012.

[13] D. Lim, "Extracting secret keys from integrated circuits in Master thesis," *Massachusetts Institute of Technology*, 2004.

[14] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. V. Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, pp. 1200-1205, 2005.

[15] M. Majzoobi, F. Koushanfar and M. Potkonjak, "Testing techniques for hardware security," in *Proc. IEEE International Test Conference*, 2008.

[16] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *Proc. International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 2015.

[17] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Proc. Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*, 2004.

[18] N. Wisiol, G. T. Becker, M. Margraf, T. A. A. Soroceanu, J. Tobisch, and B. Zengin, "Breaking the lightweight secure PUF: Understanding the relation of input transformations and machine learning resistance," in *Proc. International Conference on Smart Card Research and Advanced Applications*, 2019.

[19] B. Gassend, D. Lim, D. Clarke, M. V. Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, pp. 1077-1098, 2004.

[20] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM conference on Computer and Communications Security*, 2010.

[21] 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, pp. 1876-1891, 2013.

[22] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A machine learning-based security vulnerability study on XOR PUFs for resource-constraint internet of things," in *Proc. IEEE International Congress on Internet of Things (ICIOT 2018)*, 2018.

[23] N. Wisiol, C. Graebnitz, M. Margraf, M. Oswald, T. Soroceanu, and B. Zengin, "Why attackers lose: Design and security analysis of arbitrarily large XOR arbiter PUFs," *Journal of Cryptographic Engineering*, 2019.

[24] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *WIFS*, 2012.

[25] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and reliable XOR arbiter PUF design: An experimental study based on 1 trillion challenge response pair measurements," in *Proceedings 54th Annual Design Automation Conference 2017*, 2017.

[26] R. Want, "The magic of RFID," *Queue*, vol. 2, pp. 40-48, 2004.

[27] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A subspace pre-learning approach to fast high-accuracy machine learning of large XOR PUFs with component-differential challenges," in *Proc. IEEE International Conference on Big Data (Big Data)*, 2018.

[28] K. T. Mursi, Y. Zhuang, M. S. Alkatheiri, and A. O. Aseeri, "Extensive examination of XOR arbiter PUFs as security primitives for resource-constrained IoT devices," in *Proc. 17th International Conference on Privacy, Security and Trust (PST)*, 2019.

[29] Y. Hori, H. Kang, T. Katashita, A. Satoh, S. Kawamura, and K. Kobara, "Evaluation of physical unclonable functions for 28-nm process field-programmable gate arrays," *Journal of Information Processing*, vol. 22, pp. 344-356, 2014.

[30] M. S. Alkatheiri, Y. Zhuang, M. Korobkov, and A. R. Sangi, "An experimental study of the state-of-the-art PUFs implemented on FPGAs," in Proc. *IEEE Conference on Dependable and Secure Computing*, 2017.

[31] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, pp. 1198-1210, 2008.

[32] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs," in *Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, 2010.

[33] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *Proc. IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010.

[34] 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*, Springer, 2013, pp. 245-267.

[35] G. T. Becker, "The gap between promise and reality: On the insecurity of XOR arbiter PUFs," in *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, 2015.

[36] G. T. Becker, "On the pitfalls of using arbiter-PUFs as building blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1295-1307, 2015.

[37] M. S. Alkatheiri and Y. Zhuang, "Towards fast and accurate machine learning attacks of feed-forward arbiter

PUFs," in *Proc. IEEE Conference on Dependable and Secure Computing*, 2017.
[38] H. M. Allam and A. A. Chaudhri, "Internet of things: Extracting latest challenges and solutions," *Journal of Communications*, vol. 12, no. 9, 2017.
[39] W. Abbass, *et al*., "Assessing the internet of things security risks," *Journal of Communications*, vol. 14, no. 10, 2019.
[40] K. T. Mursi and Y. Zhuang, "Experimental examination of component-differentially-challenged XOR PUF circuits," in *Proc. 4th International Conference on Circuits, Systems and Devices (ICCSD)*, 2020 (forthcoming).

Khalid T. Mursi was born in Jeddah, Saudi Arabia. He received the B.S. degree from the King Abdulaziz University (KAU), Jeddah, in 2011, in Information Technology, and the M.S. degree from the Texas Tech University (TTU), Lubbock, TX, USA, in 2018, in Computer Science. He is currently pursuing the Ph.D. degree with the Department of Computer Science, TTU. His research interests include IoT security, and Machine Learning attacks.

Dr. Yu Zhuang received his PhD in Computer Science and PhD in Mathematics both in 2000 at Louisiana State University. He was a visiting assistant professor at the computer science department of Illinois Institute of Technology from April to July of 2001, and has been with Texas Tech computer science department since September 2001. Dr. Zhuang's research interests include IoT security, high-dimensional data modeling and mining, high performance scientific computing.