On the Evaluation of FPGA-based Physical Unclonable Functions

†Daniele Lombardi, †Mario Barbareschi, †Valentina Casola, ‡Elena Ioana Vatajelu, ‡Giorgio Di Natale

***Abstract*—Physical Unclonable Functions (PUFs) have emerged as a promising hardware security solution, offering robust capabilities for unique device identification and authentication. Their seamless integration, particularly in FPGA-based implementations, makes them especially appealing for resource-constrained devices. However, their widespread adoption is hindered by the inherent challenges of conducting systematic and fair evaluation that considers on-chip experimental data coming from a large population of devices. To address these challenges, we propose a novel evaluation-system for the automated characterization of FPGA-based PUFs. This system provides a comprehensive suite of tools that enable the efficient assessment and detailed analysis of key PUF properties, such as uniqueness and reliability, across a substantial number of physical devices. Furthermore, the system supports the whole lifecycle of PUF development, encompassing both the design and implementation of diverse PUF architectures. This allows end-users to explore, optimize, and seamlessly integrate PUF-based security features into their electronic systems and applications.**

***Index Terms*—Physical Unclonable Functions, Large-scale Characterization, Hardware Security, Field Programmable Gate Array.**

I. Introduction

In recent decades, Physical Unclonable Functions (PUFs) have gained increasing attention as a hardware security primitive for protecting systems, especially in applications using scarce resources devices, such as in the Internet of Things (IoT) domain [1]. Their inherent characteristics, including unclonability and anti-tamper properties, make them particularly well suited to address a variety of security challenges [2], as for example the mutual authentication [3], the dynamic-group secure-communication [4], and the remote attestation [5]. Among the various implementations of PUFs, those based on Field Programmable Gate Array (FPGA) are considered attractive due to the unique advantages of FPGA technology [6]. As a matter of fact, the latter offers a high degree of flexibility, allowing hardware configurations to be tailored precisely to specific security tasks without requiring costly custom silicon fabrication. Moreover, the opportunity to be easily reprogrammed enables the same hardware to be adapted, such that it turns able to accommodate evolving security requirements [7].

†Authors are with the Department of Electrical Engineering and Information Technologies, University of Naples Federico II. Email: {name.surname}@unina.it.

‡Authors are with University Grenoble Alpes, CNRS, Grenoble INP, Techniques of Informatics and Microelectronics for integrated systems Architecture Laboratory. Email: {name.surname}@univ-grenoble-alpes.fr.

†‡All authors contributed equally.

Nonetheless, although PUFs represent a sound solution, before considering an implementation as a viable trusted anchor to address security challenges, its actual effectiveness needs to be verified by respecting important evaluation criteria. First, while simulation methods can be exploited to get preliminary approximate results [8], large-scale on-chip experimentation is essential to benefit from more accurate and significant results [9]. In addition, a thorough evaluation involves characterizing the PUF on the main quality metrics proposed by the scientific community [10], [11]. Finally, a fair evaluation that also takes into account the comparison with the results of other PUF implementations must necessarily be conducted using the same experimental setup [12]. However, to the best of our knowledge, most of the existing PUF implementations are evaluated on a not significant number of devices. Moreover, existing solutions for large-scale characterization do not completely and efficiently address the aforementioned challenges, nor are they freely available to everyone.

In light of the above, in this work we present a methodology to design a generic evaluation-system for the large-scale characterization of any FPGA-based PUF. Our main original contributions can be summarized as follows:

1) to present an evaluation-system for the automatic characterization of PUFs based on FPGA;
2) to evaluate different PUF implementations, our solution is designed to provide a mechanism for easy integration of any FPGA-based PUF;
3) to facilitate the assessment on a large number of devices, our system enables modifications to the number of devices with a reduced expenditure of effort and complexity in management.
4) to conduct in-depth analysis, our evaluation-system allows to obtain data regarding all quality metrics, both by providing statistical indexes as mean and variance values, and by representing the distributions of values with charts. It also returns raw data for further later analysis, including information such as temperature, voltage and for some PUF implementations the oscillation frequencies.
5) to demonstrate the suitability of our solution, we present the results obtained on an actual implementation of the evaluation-system concerning 6 different PUFs known to the scientific community.
6) to encourage future research on novel PUF architectures, their large-scale characterization and their fair comparison against other PUFs, the implementation details of our proposal are freely available, along with the possibility to

use an actual realization of it.

The remainder of the paper is structured as follows: Section II provides the reader with an overview on PUFs, their main implementations, statistical metrics adopted to assess PUF quality, and, lastly, details on the characterization of PUFs in the literature; Section III outlines the general requirements for designing an evaluation-system for PUFs; Section IV describes our proposal of evaluation-system, focusing on both methodological design choices and on implementation details of a practical realization of it; Section V deals with the evaluation of the evaluation-system, both in terms of its performance and by testing it with different implementations of FPGA-based PUFs; finally, Section VI draws the conclusions of this work.

II. Technical Background on PUFs

PUFs represent a class of hardware security primitives that exploit the inherent physical variations in materials. Such variations are the result of uncontrollable factors that occurs during the manufacturing process, namely random nanoscale differences during silicon etching and in material deposition [13]. Such differences are not readily reproducible, thereby letting PUFs be immune against cloning or forgery. At their core, PUFs operate on the principle of challenge/response, so that by applying an input signal – the challenge – to the PUF, a unique and unpredictable output – the response – is then generated. Formally, PUF can be described by the following mathematical relation:

$$\theta : C \rightarrow R, C \in \{0,1\}^m, R \in \{0,1\}^n \tag{1}$$

where C is the challenge set and R is the response set. Challenge-Response Pairs (CRPs) are device-specific, meant that different devices produce different responses to the same challenge due to their inherent physical variations. In more recent literature, PUFs are categorized as extensive or confined [14], replacing the earlier terminology of strong and weak PUFs. An extensive PUF is capable of producing a large number of unique and independent CRPs, while a confined PUF generates a limited set of responses.

A. Metrics

Aiming at assessing a PUF design effectiveness, so far a number of quality metrics have been presented in the scientific literature [10], [11]. The most frequently employed are uniformity, uniqueness, reliability, and bit-aliasing. With reference to the notation outlined in Table I, a description of these metrics is provided below, along with their corresponding mathematical formulations.

1) Uniqueness: The Uniqueness measures the ability of a PUF to effectively discriminate between two distinct devices. Intuitively, given two devices, the greater the randomness of responses of devices to the same challenges, the better the uniqueness value. By considering $|K|$ different devices implementing the same PUF, a challenge $\bar{c}$, selected from the available set of challenges $\mathcal{C}$, and a response width of N bits, the Uniqueness can be defined as follows:

$$U(\bar{c}) = \frac{2}{|K| \cdot (|K|-1)} \sum_{i=1}^{|K|-1} \sum_{j=i+1}^{|K|} \frac{HD(\theta_i(\bar{c}), \theta_j(\bar{c}))}{N} \cdot 100\% \tag{2}$$

where HD denotes the Hamming Distance (HD). In the ideal case, the responses of two different devices are expected to have exactly $N/2$ different bits, resulting in an ideal HD of 50%, which corresponds to an ideal uniqueness value of 50%.

2) Reliability: Reliability metric estimates how stable are PUF responses sampled repetitively during time. Typically, these measures are taken under different environmental conditions, such as temperature, voltage, and aging. Hence, by querying a PUF M times with the same challenge $\bar{c}$, the desirable value of Reliability –corresponding to 1– occurs in the case where all M responses are identical to each other:

$$R(\bar{c}) = \frac{1}{M} \sum_{m=1}^{M} \frac{N - HD(\theta_d(\bar{c}), \theta_d^m(\bar{c}))}{N} \cdot 100\% \tag{3}$$

3) Uniformity: Uniformity gives an indicative value about the distribution of $'0'$ and $'1'$ bits in the PUF responses. Ideally, responses should have a uniform distribution, so about 50% of the bits being $'1'$ and 50% being $'0'$. This balancing ensures that the response has no biases that could be somehow exploited in an attack. By considering $bit(x,i)$, a function capable of extracting the $i-th$ bit from x, then Uniformity can be estimated as follows:

$$Uf(\bar{c}) = \frac{1}{N} \sum_{i=1}^{N} bit(\theta_d(\bar{c}), i) \cdot 100\% \tag{4}$$

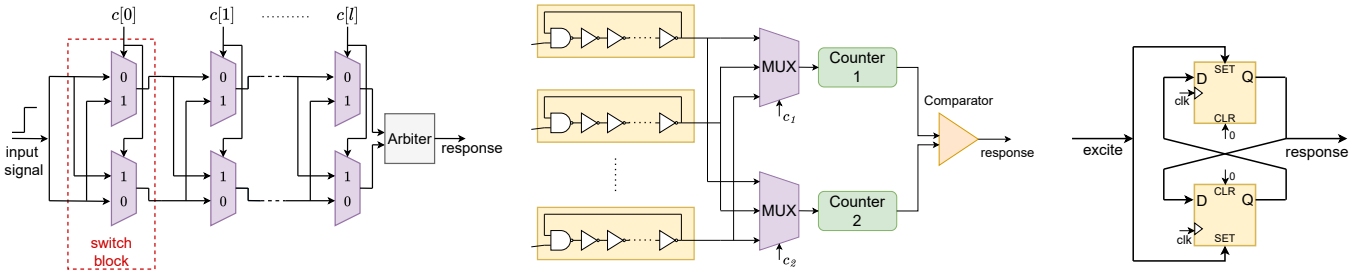
4) Bit-aliasing: Bit-aliasing assesses the similarity of responses to the same challenge between several instances of the PUF. In particular, the metric identifies whether certain homologous bits of responses may be affected by bias, meaning a tendency to assume either the value $'1'$ or $'0'$ more frequently. Given the $i-th$ bit, Bit-aliasing is calculated as:

$$BA(i) = \frac{1}{|K|} \sum_{k=1}^{|K|} bit(\theta_k(\bar{c}), i) \cdot 100\% \tag{5}$$

Beyond their descriptive function, the aforementioned metrics are essential because tightly linked to the security properties of PUF-based systems. Deviations from ideal values may result in reduced unpredictability, making PUF responses more susceptible to modeling or statistical attacks. For instance: low

TABLE I: Summary of notations.

$\theta_d(\cdot)$	PUF of Device d
$\theta_d^m(\cdot)$	m-th measurement of PUF on device d
$\mathcal{C}$	Set of challenges
$\mathcal{R}$	Set of responses of a traditional PUF
S	Set of CRPs of a PUF
K	Set of devices equipped with a PUF
N	Number of response bits
M	Number of PUF measurements
$HD(x,y)$	Hamming distance between x and y
$bit(x,y)$	returns value of y-th bit of string x



(a) (b) (c)

Fig. 1: Some examples of PUF deployable on FPGA: 1a) Arbiter-PUF; 1b) RO-PUF; and, 1c) Butterfly PUF cell.

uniformity may expose biased bit distributions; insufficient bit-aliasing can lead to predictable correlations across bit positions; and, low uniqueness could allow cross-device inference. Thus, achieving values close to the ideal 50% threshold for these metrics is not merely a matter of design quality, but a critical aspect in ensuring that PUF responses remain resistant to adversarial exploitation in cryptographic and authentication protocols.

B. Implementations

So far, in the scientific community, a variety of PUF implementations have emerged, each one exploiting a different physical phenomenon [1]. Among them, silicon PUFs, which leverage variations in integrated circuits, are the most popular because of their ease of integration into already existing devices [15]. They can be classified into two major families: delay-based PUFs, that exploit propagation delays of signals in symmetric paths, and memory-based PUFs that depend on the power-on states of memory cells. Let us now briefly overview main implementations of PUF that are designed to be implemented onto FPGA devices.

A example of delay-based PUF is the Arbiter PUF (A-PUF), which uses the difference in delay between signals on symmetric paths to produce unique responses [16]. A typical A-PUF circuit, represented in Figure 1a consists of a series of switch blocks, each driven by a different challenge capable of determining, by means of multiplexers, which path the input signal should take. The last stage entails an arbiter able to determine which of the two signals arrives first, thereby resolving each challenge into a binary response. Although A-PUFs are efficient in terms of silicon area and power consumption, they are susceptible to machine learning attacks, leading the proposal of several different variants [17]–[19]. Analogously, Ring-Oscillator PUFs (RO-PUFs) exploit the time period (or frequency) variations in oscillating circuits [20]. A ring oscillator typically consists of an odd number of inverters connected in a loop, generating a periodic signal whose frequency is highly sensitive to physical characteristics such as gate delay, wire length, and transistor threshold voltages. Due to the uniqueness of these characteristics across different integrated circuits, each device exhibits distinct oscillation patterns. In a typical RO-PUF design, represented in Figure 1b the relative frequency between two or more ring oscillators is compared to produce a single response bit. By selecting different pairs of oscillators and measuring their relative speed, a large number of CRPs can be generated. RO-PUFs offer a larger number of potential CRPs compared to A-PUF and exhibit resistance to environmental noise, yet they face limitations in power-constrained applications, where their power consumption can be an issue [15]. Conversely, memory-based PUFs include those PUFs exploiting the inherent randomness of memory-cell state when powered-on [21]. A well-known example is the Static Random Access Memory (SRAM)-PUF. Each SRAM cell is designed as two symmetric halves. Ideally, when powered-on, the memory circuit is initially in an unstable state that eventually goes toward either logic-0 or logic-1 configuration, with equal probability. However, when manufactured, unpredictable and unique nanoscale variations affects the symmetry, resulting in each cell exhibiting a preference for one of two memory states. The principal advantage of the latter is its compatibility with existing hardware, as it utilizes memory components that are already equipped in microcontrollers. Similarly, butterfly PUFs are based on the initialization of bistable circuits, which are typically designed with the use of inverters or cross-coupled Flip Flop (FF) [22], as depicted in Figure 1c. Upon initialization, the bistable circuit is brought into a metastable condition with no predefined logic state. Due to minute physical variations introduced during fabrication, the circuit resolves nondeterministically into one of two stable states, thereby generating a device-specific and reproducible response bit. The symmetric configuration of the latches resembles a butterfly pattern, hence the name. In contrast to SRAM PUF, Butterfly PUFs do not depend on existing memory cells, rather is implemented through specific designed circuits, thereby making them a versatile option in various hardware designs, including the FPGA technology.

C. Characterization

As aforementioned, there are different challenges to be faced when characterizing a PUF implementation. Table II provides a summary of some PUF proposals developed on FPGA devices that, at a glance, points out that not all quality metrics are necessarily always taken into account, and especially often the number of engaged devices is not adequately statistical meaningful. Some studies attempt to overcome the limited availability of physical devices by reusing the same PUF design and deploying it at multiple distinct physical locations within the same FPGA. This approach assumes that each PUF

TABLE II: Comparison among FPGA-PUF Implementations.

Category	Ref.	FPGA	Uniqueness	Uniformity	Reliability	Bit-aliasing	Area	Devices
Arbiter PUF	[24]	Artix-7	41.53%	54%	93% ∼ 97%	-	128 slices	11
	[25]	Artix-7	49.1%	50.3%	100%	-	150 slices	8
	[26]	Spartan-6	48%	-	-	-	224 slices	3+3+15
	[27]	Artix-7	15.15%	∼ 0.45%	98.97%	-	224 6-LUT	2
	[28]	Zynq 7000	45.2%	-	-	-	-	4
	[29]	Artix-7	40%	45% ∼ 50%	1.36% (BER)	-	407 slices	5
	[30]	Artix-7	41.53%	-	93% ∼ 97%	-	2816 slices	10
	[31]	Artix-7	17%	-	97.5%	-	-	3
	[32]	Artix-7	19.39	-	0.24% (BER)	-	-	n.s.
	[33]	Artix-7	49.8%	-	94.2%	-	395 6-LUT	1
	[18]	Artix-7	51.1%	-	100%	-	-	1
	[34]	Zynq 7000	49.3%	-	100%	-	-	1
	[19]	Altera	51.06%	50.18%	99.67%	-	-	16
RO PUF	[35]	Spartan-7	46.4%	-	99.68%	-	-	2
	[36]	Artix-7	56.1%	50.36%	98% ∼ 99%	-	-	50
	[37]	Pynq-Z2	39.1%	∼ 50%	98.9%	50.2%	7 LUT	20
	[38]	Kintex-7	50%	-	∼ 99%	-	365 slices	2
	[39]	Artix-7	47.40%	49.2%	-	49.19%	5 LUT	6
	[40]	Spartan-6	48.49%	50.99%	99.95%	-	210 LUT	40
	[41]	Artix-7	49%	50%	85.95%	-	626 LUT	3
Others	[22]	Virtex-5	∼ 50%	-	94%	-	130 slices	36
	[42]	Virtex-7	47.9%	-	96.4%	-	2 slices	6
	[43]	Spartan-3	49.2%	-	96.4%	-	2 slices	6
	[44]	Spartan-6	50.89%	49.41%	91.25%	-	348 slices	10
	[45]	Artix-7	37.4%	55.4%	98.9%	46.9%	17 slices	10

instance, although implemented on the same device, behaves as if it were located on a separate physical chip. However, this assumption may lead to inaccurate PUF characterization across different implementations. As a matter of fact, it is well known that different regions of an FPGA fabric can exhibit non-uniform behavior when implementing PUFs [23], which significantly impacts the evaluation of key metrics such as bit-aliasing.

Conversely, other studies in the scientific literature have concentrated on large-scale on-chip characterization of PUFs, aiming to evaluate their suitability for secure hardware applications. However, these works typically aim to characterize a specific type of PUF, not allowing a direct, easy and fair comparison among different PUF architectures. For instance, in [12], authors present a comprehensive analysis of oscillation-based PUFs, including Ring Oscillator (RO), Transient Effect Ring Oscillator (TERO), and Loop PUFs, implemented on 100 Digilent BASYS-3 development boards featuring Xilinx Artix-7 FPGA. The paper details different metrics such as uniqueness, reliability, and temperature sensitivity across a controlled range of $-22\ ^{\circ}C$ to $44\ ^{\circ}C$, providing the reader with valuable insights into PUFs behavior under different conditions. However, implementation details of the evaluation infrastructure are not specified, making it challenging to evaluate its effectiveness against other PUF implementations. Also in [9] authors present large-scale RO-PUF evaluation, deploying 125 Xilinx Spartan 3E FPGA devices in order to assess inter-die uniqueness, which achieves an average HD of 47.31%, and intra-die reliability of 0.86% under standard operating conditions. For data collection, the authors enlisted the Computer Engineering students of their department, who are required to buy such a development board for course-works and projects. After publicizing the technical details and the purpose of the experiment via email and webpage, data were collected from the students' FPGA in 8 different sessions, each 2 hours long, conducted in the departmental laboratory used by students for project works. Despite the large number of devices involved, unfortunately the paper makes no reference to any automation process for data acquisition. The authors of [46] advance the theoretical foundation by modeling the relationship between PUF uniqueness and min-entropy. By using a testbed of 184 different Digilent Basys3 boards, they demonstrate how high uniqueness correlates with robust entropy metrics in RO-PUFs. Although this study introduces a theoretical tool to support large-scale assessments, it focuses on a single PUF type and does not directly address the practical challenges of incorporating diverse PUF architectures in a testing infrastructure. Instead, the work [47] addresses the need for secure PUFs in resource-constrained IoT environments, by implementing and characterizing a TERO-PUF on both Xilinx Spartan 6 and Altera Cyclone V FPGAs leveraging on a testing infrastructure. Their evaluation spans critical quality metrics such as robustness to temperature and voltage fluctuations. Although the paper lacks of details about the exploited infrastructure, authors claim that they are able to enroll PUF from just 6 different boards at a time, which limits a proper automated evaluation, as well as the scalability, of the entire infrastructure. The work [48] further refines the characterization of RO-PUFs by exploring the impact of FPGA slice type, evaluation time, and temperature variations on 217 Xilinx Artix-7 based FPGA devices. The paper provides an extensive dataset that includes optimal evaluation times and temperature-dependent error rates, offering valuable insights about RO-PUF reliability under different environmental conditions. However, like previous works, authors focus solely on RO-PUFs and there is no effort to make evaluation infrastructure adaptable for different PUF architectures on a shared platform.

Finally, authors of [49] introduce an automated platform for large-scale data collection of SRAM-based PUFs on micro-

TABLE III: Comparison among large-scale characterization systems. C indicates the possibility of comparing multiple PUFs; D the availability of data; M is the maintainability property; A is the possibility to access the system. n.a. stands for not available; av. for available; n.s. not suitable.

Ref.	PUFs	FPGA	C.	D.	M.	A.
[12]	RO, TERO, Loop	yes	yes	n.a.	n.s.	no
[9]	RO	yes	no	n.a.	n.s.	no
[46]	RO	yes	no	n.a.	no easy	no
[47]	TERO	yes	no	n.a.	low	no
[48]	TERO	yes	no	n.a.	no easy	no
[49]	SRAM	no	no	av.	low	open
Us	**Any**	**yes**	**yes**	**av.**	**high**	**open**

controllers. Their setup, which involves 84 different STM32 microcontrollers, allows for the automated acquisition of SRAM-PUF data under various environmental conditions.The platform data, as well as the computation of all quality metrics, is publicly accessible. However, the interconnection of all boards in daisy-chain configuration limits the easy scalability and maintainability of involved devices.

III. PUF Evaluation

Before going into details of our proposal and taking into account limitations of current solutions presented in Section II-C and challenges presented in the introductory Section I, we attempt to list what are the general desirable requirements for the design of a system for the evaluation of PUFs. In particular, we detail the features that enable the evaluation-system to efficiently address the aforementioned challenges. We also present a reference system model that highlights the advantages of such an evaluation-system, later exploited to design our proposal.

A. General requirements

The tight connection between a PUF behavior and the physical phenomenon it exploits implies that any evaluation method not based on direct on-chip experimentation is inherently less accurate. Although PUF simulation methods are undoubtedly valuable for designing new PUF implementations [50], they cannot be reliably employed for systematic and accurate evaluations. A crucial element in this context is the availability of a significant number of devices for experimentation. As a matter of fact, certain quality metrics, such as uniqueness and bit-aliasing, are estimated using data collected from different devices. Having only a limited number of devices would potentially produce results that are not statistically relevant. Therefore, considering the necessity of relying on a large number of devices, fully or partially manual approaches to the enrollment of PUFs are strongly discouraged. Instead, fully automated solutions are recommended, as manual processes would entail multiple steps for each device, culminating in a significantly time-consuming task. Additionally, a desirable requirement is the possibility to add devices to the system while minimizing the effort involved, ideally through a plug-and-play operation. This would offer two significant advantages. On one hand, maintaining the infrastructure, e.g. in the case of failures, would become simpler and faster. On the other hand, the reduced complexity of device integration would facilitate the deployment of the entire infrastructure, without necessarily requiring advanced technical expertise. Finally, an evaluation-system should enable the comparison of multiple implementations. This would allow for a more critical and fair assessment of the various PUF implementations.

B. Reference Model

The evaluation-system described above can be effectively applied to the reference model we propose in Figure 2. Specifically, the activity concerning the proposal of a novel FPGA-based PUF can be easily separated into two distinct phases: design and evaluation. During the former, the PUF-designer is solely concerned with the design and implementation of the new PUF. This phase generally requires the use of FPGA design tools, which may also offer simulation capabilities enabling the check of some parts of the new design. In this way, the design phase can be accomplished following a totally hardwareless approach, by lowering the economic cost and time required to build the infrastructure necessary for quantitative performance evaluation. On the other hand, as for the evaluation phase, the evaluation-system allows the PUF-tester to run experiments on a large number of devices and to obtain the PUF quality metrics. Therefore, such results are a necessary feedback to the designer in order to have an estimate of the quality parameters of implemented solution. Moreover, the design phase must necessarily end by providing as output: the bitstream file to program the FPGAs of devices; and, the information to be provided to the tester in order to properly set the communication with PUFs on devices.

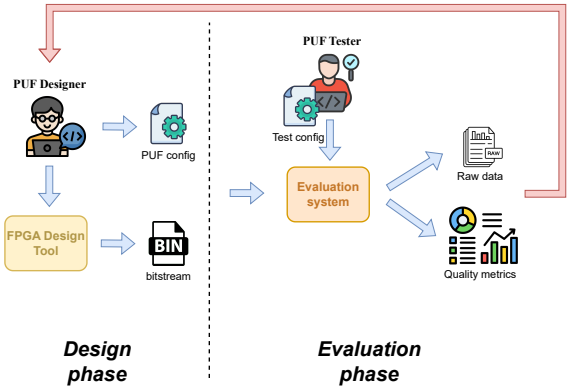


Fig. 2: Reference Model.

IV. PUFs Evaluation-System Proposal

Bearing in mind the section above, we present our proposal of evaluation-system for the assessment of FPGA-based PUFs. We first present the methodological choices of the design aiming to satisfy the mentioned-requirements, then we discuss a practical implementation of it.

A. Design

To better separate the responsibilities of each component and offer a single interface to the end-user of the evaluation-system, we opt for a closed layered style to be deployed on

a three-tier architecture, as shown in Figure 3. In this way, the end-user interacts with the system by means of a client application that allows for initiating experiments on PUFs and subsequently obtaining the corresponding results, both in terms of raw data and aggregated quality metrics – such as uniqueness, reliability, bit-aliasing and uniformity–. The server constitutes the core of the evaluation-system, since it comprises all of the logical processes required to handle client requests and to interact with devices of the underlying layer by means of an ad-hoc communication protocol. Finally, the device layer consists of a bank of devices, exposing both a CPU to execute the device-application – required to manage all operations related to experiments – and a FPGA part on which the PUF to test is synthesized. It is worth noting that CPU and FPGA do not necessarily need to be co-located on the same chip; instead, they can be hosted on separate devices interconnected with a generic communication channel.

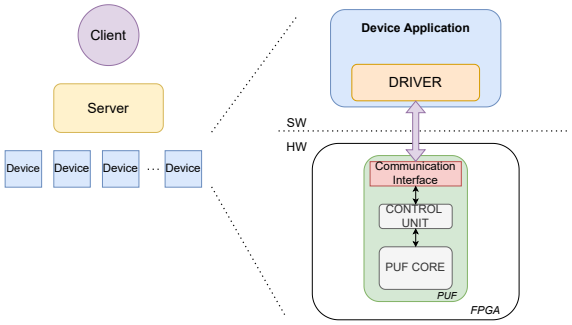


Fig. 3: Details on deployment architecture of the evaluation-system and on communication between PUF and device-application.

Moreover, aiming to address all the requirements for a proper and fair evaluation of PUFs, our proposal is designed to guarantee the following fundamental properties.

1) Maintainability: In order to handle a large number of devices, our proposal adopts several design strategies that favour its maintainability. First, the communication between the server and devices is performed by means a wireless channel. Consequently, when a new device needs to be added or a malfunctioning one requires replacement, the system maintainer is only required to load the appropriate device-application onto the device and provide the power supply. Accordingly, no additional connection for data exchange is required. Moreover, the maintainability is further supported by the possibility of reusing the same device-application for all the PUFs to be tested, without therefore needing to change it for each experiment. As a matter of fact, the application takes advantage of the dynamic reconfiguration capability of the FPGA to replace the PUF to assess. In particular, in order to be totally transparent to each kind of experiment, the application running on the devices entails the following application life-cycle:

1) Initialization, in which the application initializes its internal data structures and establishes the communication with the server;
2) Registration, useful for signalling to the server the presence of a new device, available for experiments;
3) Commands handling, accomplished with an endless loop, allows the application to handle requests from the server, such as: reprogramming of the FPGA (with related data to configure the PUF-driver); querying the PUF; shutdown operation.

Finally, in case of a new release of the device-application, the server automatically updates all devices, ensuring their logic is always working with the latest version available.

2) Compatibility: The ability to assess different PUFs with the same evaluation-system strictly depends on how well the logic for experiments used by the latter can be suitable for different types of PUF implementations. To this end, our evaluation-system uses a common and unified interface to enable the communication between the system and the PUF to be tested, as represented on the right side of Figure 3. Consequently, all FPGA-based PUFs that expose such interface can be integrated into the evaluation-system and hence tested. Although this may appear to be a limitation as it restricts the kind of PUFs to be tested – i.e. PUFs with different interfaces are not testable with the evaluation-system – it should be noted that the interface logic of an Intellectual Property core (IP-core) generally requires only minimal effort compared to the entire design and development process of a new PUF. Therefore, by conceptually decoupling the core logic of the PUF from the interface logic, the choice to use a common interface makes the system highly compatible with a multitude of disparate PUF implementations. Clearly, the only use of a common interface is not sufficient to guarantee the compatibility property; rather, the adoption of a common driver utilizing that interface is also necessary. Once again, despite the potential limitation, the integration of the communication control-logic within the IP-core represents a negligible part of that required for the new PUF.

3) Configurability: Full automation of experiments is achievable if the evaluation-system can be seamlessly adapted to each type of PUF under test. Ensuring only the compatibility property just described is clearly not sufficient in this regard, a configuration mechanism is necessary to make the system flexible and easy to use. To address this, the solution we propose ensures complete transparency concerning the PUF being tested through the use of three distinct configuration files: the bitstream file, the PUF configuration file, and the experiment configuration file. The first two files are produced as output of design phase of the new PUF, while the third one is written by the PUF tester, as depicted in Figure 2. The use of a configuration file for the PUF enables the isolation of all specific information utilized by the driver for the initialization and the communication with the PUF instance. Such information may include, but is not limited to, details regarding the enabling of the PUF, the challenge values, the response value and so on. In contrast, the configuration file for the experiments permits to tailor the type of queries in accordance with the specific implementation of the PUF. For instance, if a PUF involves a challenge value represented over a very large space – e.g. 256 bits –, it is unlikely to test that PUF over the entire challenge representation space,

due to the considerable time constraints involved. Accordingly, the evaluation-system allows the configuration of three distinct types of experiments: list, range, and random. In the first case, the PUF-tester is required to specify the precise list of challenges to be submitted to the PUF. In contrast, the range type enables the tester to define the representation range of the challenge, from which all individual values with a specified step are extracted. Meanwhile, the last type, i.e. random, allows the generation of a number n of random values within the representation range of the challenge, once the seed is specified. All the aforementioned types of experiments afford the tester to configure the desired number of repetitions, the interval between two successive PUF queries, and the interval between two consecutive repetitions of the same experiment.

B. Implementation

To validate correctness of the proposed evaluation-system and verify its versatility to manage different PUF architectures, we present a complete prototype implementation covering all layers of the system just described[1].

As already mentioned, the client-application acts as sole interface for the evaluation-system. It is developed in Python and does not require any particular requirements other than the installation of libraries specified in its requirements file. It allows communication with the server on both private and public networks. In particular, upon the user authentication, the application permits the following operation to be carried out: (i) launching experiments on a user-definable number of boards; (ii) obtaining raw data of previously performed experiments; (iii) computing PUF quality metrics; (iv) turning all devices off and on; (v) turning device cooling system off and on.

As for the server, it is hosted on a desktop machine, equipped with Intel Core i5-2400 CPU @ 3.10GHz; 4Gb of RAM memory and Ubuntu 24.04 LTS as operating system. All the logic part is developed with Django[2], a high-level python-based web-framework for web server development. In particular, the requests from clients and devices are handled through a Django's dedicated app. Django natively supports essential scalability features such as asynchronous request handling, load balancing, and horizontal scaling, making it suitable for deployments involving a large number of devices sending concurrent requests. In our context, these features allow the server to remain responsive even under high load conditions, such as those generated by large-scale PUF evaluation campaigns. Nginx[3], an open-source web server, is used as a reverse proxy for the private and public network communication of clients. Its lightweight architecture and high concurrency support further contribute to the scalability and robustness of the system. Moreover, to ensure adequate bandwidth for wireless data transmission in multi-device setups, the system leverages IEEE 802.11n [51] connectivity, which guarantees a theoretical throughput of up to 600 Mbps. This bandwidth is considered sufficient for our application scenario, given that the data packets exchanged by the system typically consist of only a few tens of bytes, with the exception of reconfiguration packets containing the new FPGA bitstream, which may reach up to a few MB in size. The MySQL database, which is officially supported by Django, is chosen for data persistence, while MySQL Workbench is used for the database modeling part. Figure 4 shows an example of all the information stored in the database for each query.

Given the requirements expressed in the previous sections, concerning the devices layer, the system utilizes 18 Avnet Ultra96v2 boards, equipped with Zynq UltraScale+ MPSoC ZU3EG A484 architecture, including 4 Arm Cortex-A53 MPCore cores up to 1.5 GHz and 70560 CLB Luts on the FPGA part. They are also equipped with an SD memory reader, used for the boot of the system, and a ATWILC3000 bluetooth Wi-Fi microchip for the connectivity. As for the software part, needing dynamic reprogramming of FPGA; memory access to peripheral devices; and, communication with wifi module, we opt for an operating system to simplify the development of such operations. Specifically, we use Petalinux 2020.2, a framework for creating linux-based embedded systems, that embodies the Linux FPGA Manager Driver. The FPGA reprogramming process is performed by loading the proper bitstream file – previously saved in `lib/firmware` –, in `/sys/class/fpga_manager/fpga0/firmware`, by specifying the full reprogramming as mode. It is important to note that any transient effects caused by this process do not affect PUF experiments. Indeed, the first request to access the PUF arrives several seconds after reprogramming has completed. This delay accounts for the completion of the following sequence of operations: the `pr_done` bit indicates the successful conclusion of reprogramming on the device-side; the device-application builds a response message to notify the server; upon receiving this message, the server is informed that the device is ready to perform the experiments; only then it can send the first PUF query request to the device. Regarding the device logic part, it is realized by an application written in C language, running on the operating system. Communication with PUFs is facilitated via the AXI4 Lite interface, whereas device memory access is enabled by a user-driver that relies on the `mmap` and `munmap` system calls. Temperature and voltage values are read using the sensors already on the FPGA of the board, through the PL SYSMON unit, provided by the AMD SYSMONE4 architecture.

The switching on and off of all boards, as well as of additional heatsinks, can be automatically driven by means of the MIKROE-3357 relais, controlled by the server.

V. Evaluation

The goal of this section is to illustrate that the functionalities of the evaluation-system are capable of answer to challenges outlined in Section I. To this aim, we present the results of experiments conducted on 6 different types of PUFs, as well as 2 variants of these. Some of the PUFs used for the experiments are publicly available designs, while the others are created from scratch. Since delay-based implementations are particularly sensitive to the area of FPGA in which they

[1]https://github.com/daniref/PUF-Evaluation-Platform.git
[2]https://www.djangoproject.com/
[3]https://nginx.org/

```
idcrp,idpuf,numrep,challenge,response,temperature,voltage,counter1,counter2,timestamp
841,0,0,00,00,26.42,0.841,006B41B5,006BE1AF,2024-10-22 10:14:31+00:00
```

Fig. 4: Example of query saved in the database.

TABLE IV: Tests report.

Test	Fans	Chal width	Type	Num Challenge	Num exps	Exp period [m]	Queries period [s]
0: RO-PUF	yes	8	Range [step=1]	256	200	5	2
1: 4x4 Arbiter [52]	yes	30	Random [seed=5678]	1000	50	5	2
2: Arbiter	yes	32	Random [seed=4123]	1000	50	5	2
3: RO-PUF v.2	yes	1	List	1	1000	0	15
4: Nolting [53]	yes	1	List	1	1000	0	10
5: Arb-Butterfly [54]	yes	32	Random [seed=4123]	800	800	5	5
6: 7xRO-PUF v.2	no	1	List	1	200	5	0
7: RO-PUF	no	8	Range [step=1]	256	200	5	2

are synthesized, the placement of their resources is done by means the hard macro of `vhdl` or by using ad-hoc `julia` scripts. It should be noted that the primary objective of the section, and in general of the paper, is not to realize the optimal design, but rather to show the potentials of the system in terms of quantitative results that can be employed: (i) to enhance the design of a PUF; (ii) to check the best placement of a PUF within the available area of the FPGA; (iii) to compare performances of different PUF solutions. Finally, we analyze the performance of the proposed evaluation-system.

A. Experiments

The Table IV shows a summary of the tests performed with the evaluation-system, along with all the parameters necessary to configure the tests.

For Test 0, we use a RO-PUF developed from scratch, having 128 challenge bits and 1 response bit. It consists of 2 different banks, each comprising 128 ring oscillators. The challenge submitted to the PUF drives the selection of the oscillator index within the bank to be compared. Ring oscillator comparison is always done for elements having the same bank index. That is, the ring-oscillator of index 4 in bank 1 can only be compared with its counterpart in bank 2. The frequency of the various oscillators is calculated using 2 counter registers, one for each bank, whose outputs feed a comparator that determines whether the PUF response is 0 or 1. At each query, the PUF, along with the single response bit, also provides values of the 32-bits counter registers.

With Test 1, we evaluate the A-PUF proposed in [52], with a challenge width of 30 bits and a 6-bits response. Each stage of the PUF consists of a switch block with 4 input and 4 output signals. The PUF relies on dynamically configuring the connections between these four inputs and four outputs, by controlling them with two input signals. Each combination of control signals selects a specific permutation of inputs to outputs. This mechanism is represented by a vector of permutations $\Pi = (\pi_{00}, \pi_{01}, \pi_{10}, \pi_{11})$, defining a unique mapping for a specific configuration of the control signals. This approach provides 576 possible configurations per switch block, enabling a high degree of reconfigurability. According to the authors, the reconfigurable nature of the switch blocks allows to update the PUF response, enhancing security by making modeling or cloning more challenging.

Test 2 exploits a classic A-PUF, developed from scratch. Each stage of the PUF consists of a pair of $2:1$ multiplexers, implemented by means the `LUT` primitive. The implementation allows for automatic configuration of the response and challenge width, in addition to the absolute position on the FPGA area. This specific test uses a configuration with 32-bit challenges and 8 bits for the response. A detail of this implementation is shown in Figure 5.

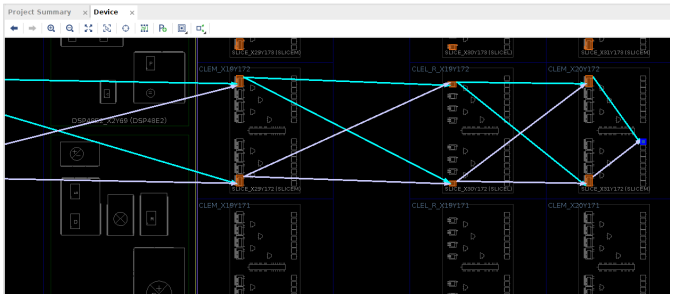


Fig. 5: A-PUF architecture details on the FPGA used for experiments.

Test 3 uses a slightly modified version of the PUF used in Test 0. Rather than returning only one bit per query, it returns the results of all possible comparisons between the two banks, i.e. 128 bits of response. Clearly because of the limitation of the AXI interface, this implementation cannot also provide the values of the counter registers for each response bit.

The PUF leveraged for the Test 4 provides 96-bit unique identifier based on the intrinsic manufacturing variations of individual FPGA chips [53]. Each bit of the ID is determined by an asynchronous PUF cell, which consists of a ring oscillator controlled by a latch. The ring oscillator, implemented with a simple feedback inverter, generates oscillations influenced by the specific physical characteristics of the FPGA, such as routing delays and semiconductor variations. When the PUF is enabled, the latch in each cell controls the oscillator by allowing it to oscillate temporarily before obtaining its state (0 or 1). To avoid interference between oscillators, a shift register activates cells sequentially, ensuring precise timing control.

In Test 5 a Hybrid Arbiter-Butterfly PUF is evaluated. Such PUF implementation combines two key elements: arbiter-based PUFs and butterfly cells, leveraging both timing path differences and metastable behavior to enhance the complexity

and unpredictability of the response. The PUF architecture is arranged in stages, each of which involves an arbiter, a butterfly cell followed by another arbiter. The arbiter components consist of a pairs of multiplexers that select different signal paths based on the challenge bits. The butterfly cells, on the other hand, are composed by cross-coupled flip-flops, introducing controlled metastability, where small variations in timing due to manufacturing discrepancies create unpredictable but repeatable states, further amplifying the entropy. The final stage employs D flip-flops to capture the state of the signals, resulting in a single bit of response. Such basic structure can be replicated to generate a response with multiple bits.

Test 6 employs the same PUF design as Test 0, but in contrast to the latter, the hardware design includes seven distinct PUFs implementation in disparate regions of the FPGA, as shown in Figure 6.

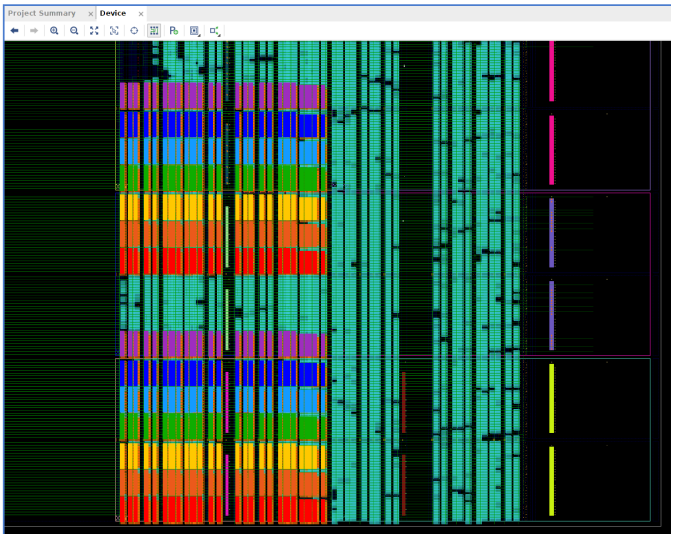


Fig. 6: 7 RO-PUFs implemented on the FPGA used for experiments.

All PUFs are stimulated in an identical manner, i.e. utilizing an identical set of challenges, aiming to determine which instance best performs.

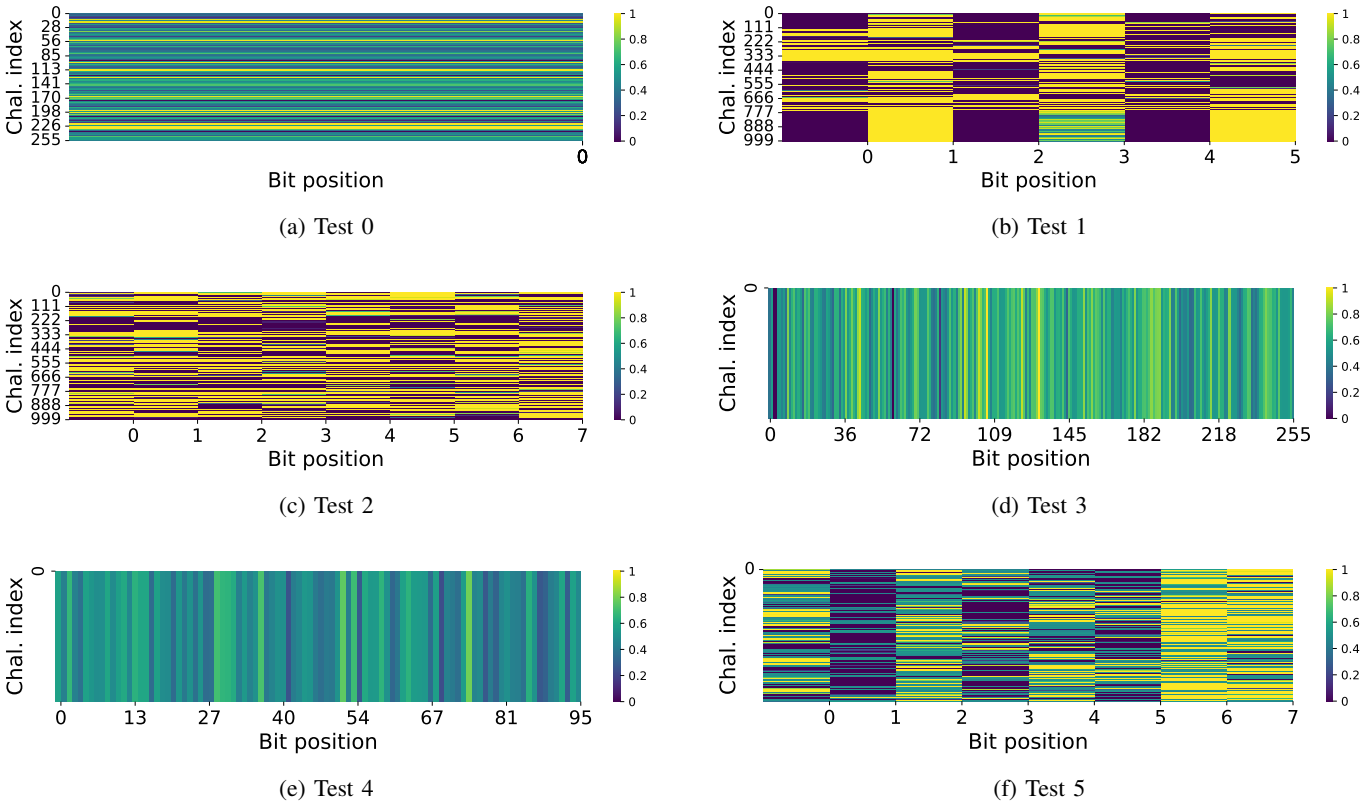
Finally, Test 7 replicates the configuration of Test 0 but is conducted under a different environmental temperature. This experiment is expressely designed not to provide an exhaustive analysis of all possible environmental factors, but rather to just demonstrate the capability of our evaluation-system to properly consider them. In particular, it allows to analyze the possible correlation between the temperature variation and the oscillation frequencies of ROs. Other factors, such as voltage fluctuations and device aging, are also intrinsically supported by the system, which allows precise control over the duration of an experiment and the acquisition of voltage measurements.

B. Results

Figure 7 shows the bit-aliasing values. As can be easily observed, the PUFs exhibiting good bit-aliasing values are those of tests 0, 3, and 4, respectively figures 7a, 7d and 7e. Except for a few bits where color bands tend toward yellow or purple, i.e. respectively 1 or 0 of bit-aliasing, most bits are represented by greenish coloring, indicating values close to 0.5. Tests 1 and 2, on the other hand, have poor bit-aliasing values, except for discrete values found in the bit of position 2 in Test 2 in correspondence with high values of challenge, see Figure 7c. The scarce quality of these two implementations is also evidenced by the uniqueness values reported in Table V. This confirms the known difficulty of being able to find perfectly symmetric paths for signals on FPGA, that can impact the responses solely because of their nanoscopic differences. Hence, these designs need to be further investigated and modified to achieve better metrics. As an example of the aforementioned, the PUF employed in Test 5, while fundamentally based on an Arbiter architecture, demonstrates improved performance due to the integration of a bistable cell, which enhances the entropy of the response, as can be clearly observed from the results in Table V and Figure 7f. Regarding reliability measure, the results are plotted in Figure 8. In all cases, the values are exceptionally high, except for Test 5 in Figure 8f. It is evident that in Tests 3 and 4, Figures 8d and 8e respectively, the standard deviation is zero, given that the PUF only accepts a single challenge value. Finally, Tables V shows the uniformity and uniqueness values of the tests just analyzed. Clearly, for Test 0 these metrics are not reported since the PUF response is represented on only one bit.

Regarding Test 6, reliability and uniqueness are given in Table VI, in terms of mean and standard deviation, while bit-aliasing and uniformity are respectively reported in Figures 10 and 11. The data show that, except for reliability – always consistently equals to 0.99 –, metrics can change by several percentage points, depending on the area of the FPGA chosen for the PUF placement.

Ultimately, analyzing data from Test 0 and Test 7, as expected, the operating temperature of the circuits affects the oscillation frequencies of the ring oscillators. In Figure 9, we report the heatmaps of devices 2 and 6, showing how the frequency difference between the ROs of block 1 and block 2 varies with temperature. Bits represented by highly saturated colors (red or blue) are those that exhibit greater resilience to temperature variations. Conversely, bits depicted in a grayish color are characterized by less pronounced frequency differences between block 1 and block 2, making them potentially more susceptible to instability. Specifically, subfigures 9a and 9b represent data of device 2, while subfigures 9c and 9d data of device 6. It is important to note that in all graphs, data are not available across the entire range of temperatures shown. The gaps are due to the experiments being conducted in a room at ambient temperature—first with the heatsinks attached to the boards active (subfigures of Figure 9 on the left), and subsequently with them inactive (subfigures on the right). As well as being visible graphically, further proof that the frequency variation fluctuates more at higher temperatures comes from the mean of the variances conducted on all frequency variations of bits. The average variance goes from 5.09 Mhz to 11.42 Mhz for device 2, while for device 6 it goes from 5.09 Mhz to 11.5 Mhz. These types of results can be



(a) Test 0

(b) Test 1

(c) Test 2

(d) Test 3

(e) Test 4

(f) Test 5

Fig. 7: Bit-aliasing results of Tests 0 to 5.

The x-axis shows the position of the bits in the responses, while the y-axis shows the index of the challenges sent to the PUFs. Finally, the colour scale indicates the resulting bit-aliasing value. Yellow means that, for a given challenge, the bit on all devices is always equal to 1, while purple is always equal to 0.

very useful when cross-referenced with those obtained from all other devices to aid in identifying a common subset of stable bits.

TABLE V: Uniformity and Uniqueness results of Tests 0 to 5.

	Uniformity		**Uniqueness**	
Test	Mean	Std	Mean	Std
Test 0	n.a.	n.a.	n.a.	n.a.
Test 1	0.433348	0.203739	0.028948	0.036607
Test 2	0.528323	0.316405	0.032599	0.040439
Test 3	0.603338	0.277523	0.480992	0.0
Test 4	0.484375	0.039045	0.513338	0.0
Test 5	0.497826	0.143418	0.232832	0.084725

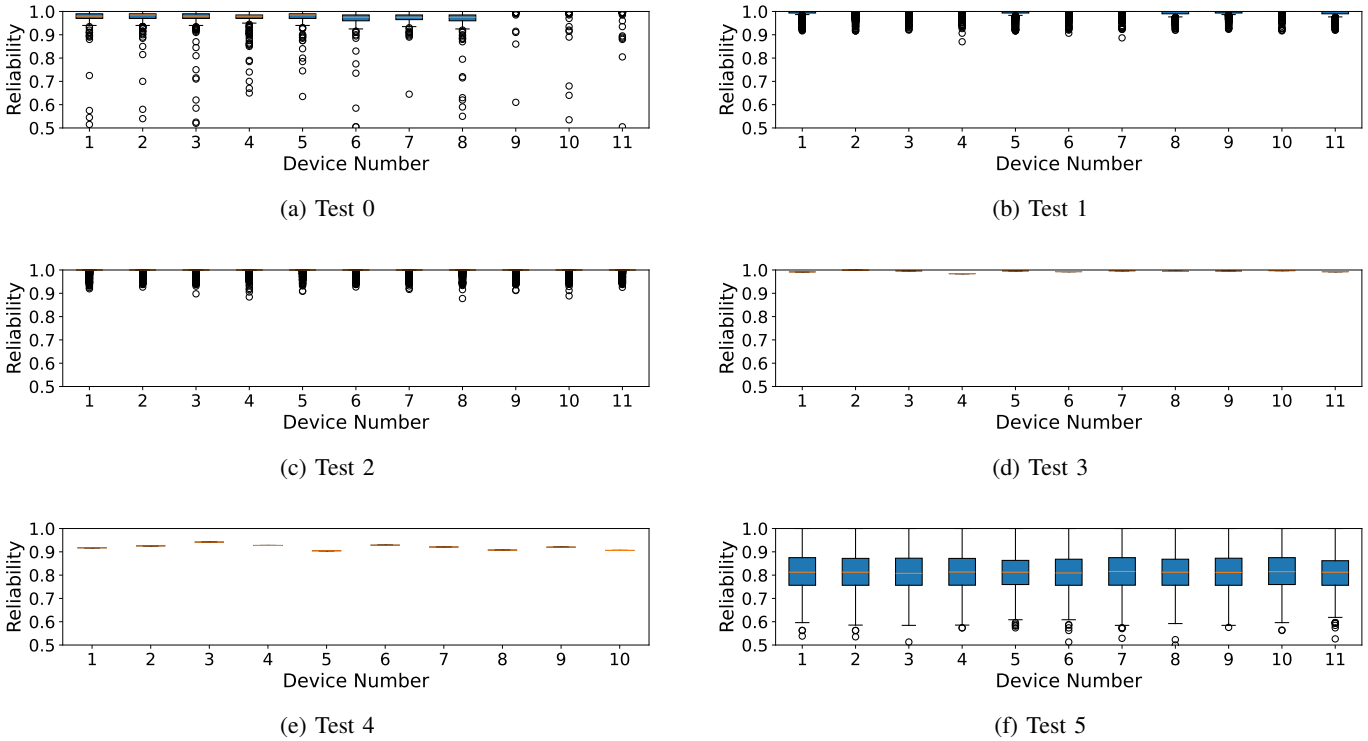
TABLE VI: Reliability and Uniqueness of Test 6.

	Reliability		**Uniqueness**	
idPuf	Mean	Std	Mean	Std
0	0.99	0.00	0.37	0.0
1	0.99	0.00	0.38	0.0
2	0.99	0.00	0.37	0.0
3	0.99	0.00	0.42	0.0
4	0.99	0.00	0.43	0.0
5	0.99	0.00	0.44	0.0
6	0.99	0.00	0.36	0.0

C. Analysis of System Performance

This section presents a performance analysis of the proposed evaluation-system, focusing on three main aspects: registration latency, communication throughput, and overall efficiency.

1) Device Registration Latency: To investigate the behaviour of the proposed system under realistic conditions and at larger scale than our physical testbed permits, we rely on an emulation campaign where multiple parallel instances of the device application execute on a separate Linux host. Each instance simulates a distinct device and communicates with the server over a commercial 802.11n Wi-Fi router, using the same UDP-based protocol and registration logic implemented on the actual physical device. Upon startup, each emulated device undergoes a full initialization sequence including network boot, IP assignment via DHCP, and registration to the server via multicast request. The server logs the timestamps of all registration events, allowing us to compute the elapsed time between device activation and successful registration acknowledgment. Preliminary results on 100 concurrent emulated devices, depicted in Figure 12, suggest that registration latencies remain within acceptable bounds under moderate load. These observations confirm the ability of the proposed system to support the parallel registration of multiple devices, without immediate performance degradation.



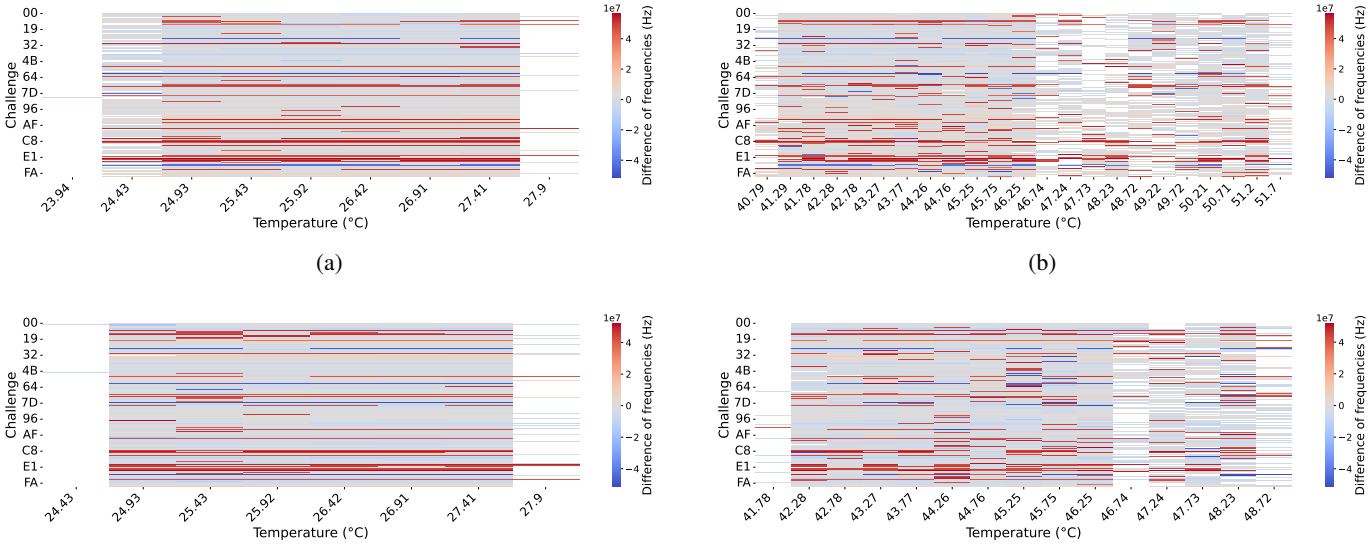
(a) Test 0 (b) Test 1

(c) Test 2 (d) Test 3

(e) Test 4 (f) Test 5

Fig. 8: Reliability results of Tests 0 to 5.

The x-axis shows the device used for the test, while the y-axis the value of reliability across the various challenges.



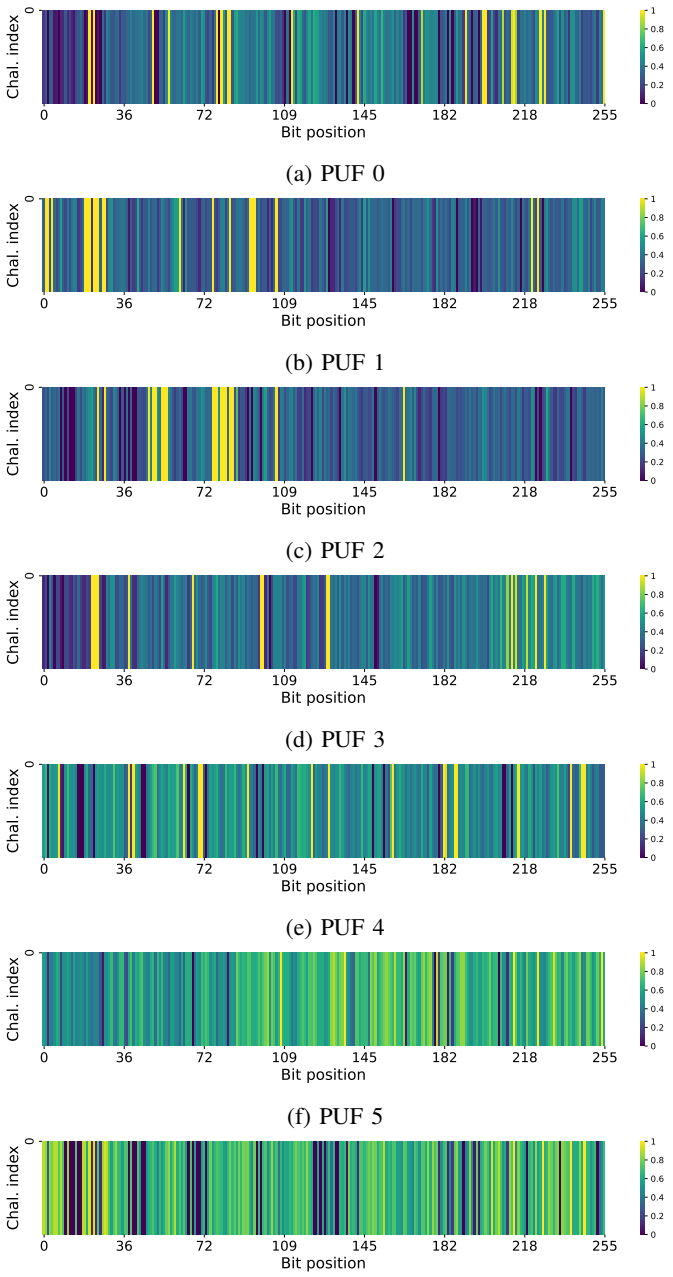
(a) (b)

(c) (d)

Fig. 9: Difference of ROs fequencies under temperature variation for devices 2 and 6, obtained in Test 6.

The y-axis shows the challenges, while on the x-axis there are the temperatures. The color represents the variation of frequency between bank 1 and bank 2 of RO-PUF. Subfigures 9a and 9b represent data of device 2, while 9c and 9d device 6.

2) Theoretical Communication Throughput: During standard operation, each device periodically sends UDP messages to the server containing raw PUF responses. Considering an average PUF response of 256 bits, our communication protocol entails a 57 bytes of useful data, excluding transport and network layer headers. Assuming a nominal 802.11n net throughput of 150 Mbps and a per-packet size of approximately 85 bytes – 57 bytes of payload + 8 bytes UDP header + 20 bytes IP header –, the maximum theoretical



(a) PUF 0

(b) PUF 1

(c) PUF 2

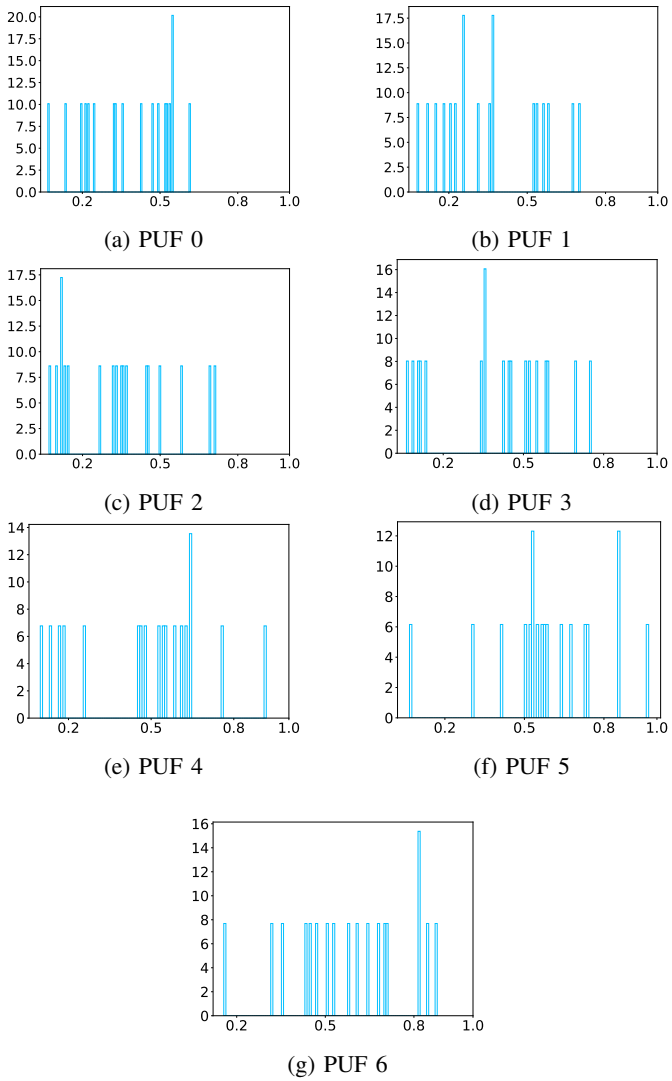
(d) PUF 3

(e) PUF 4

(f) PUF 5

(g) PUF 6

Fig. 10: Bit-aliasing results of PUFs 0 to 6 analyzed in Test 6.



(a) PUF 0

(b) PUF 1

(c) PUF 2

(d) PUF 3

(e) PUF 4

(f) PUF 5

(g) PUF 6

Fig. 11: Uniformity results of PUFs 0 to 6 analyzed in Test 6.

number of packets per second the channel can sustain is: $Th_{\max} = \frac{150{,}000{,}000\,\text{bit/s}}{85\times 8\,\text{bit}} \approx 220\,\text{packets/s}$. While this upper bound does not account for contention or processing delays, it provides a meaningful reference for evaluating system capacity. Notably, in practical scenarios involving delay-based PUFs, the response generation time on the device side can range from a few milliseconds up to nearly one second, depending on the specific implementation under test. Hence, sustaining an average query rate of one request per second per device is not only realistic, but also consistent with the intrinsic timing constraints of PUFs evaluation. This suggests that the estimated channel capacity is adequate to support large-scale experiments without becoming a limiting factor.

3) Evaluation Efficiency: To measure how efficiently the proposed solution responds to the challenges presented in Section I, it is useful to analytically estimate the amount of time T_r required for the realization of a new PUF. Considering performing experiments sequentially on N devices, such time can be represented as $T_e = T_D + N \cdot (T_S + T_t)$, where T_D is the time for the design of the new PUF; T_S the time to set the single experiment; and, T_t is the time required to execute the experiment. As experiments can be automatically and simultaneously launched on all devices, the proposed evaluation-system allows to reduce the overall time T_e at the cost of T_D, T_S and T_t.

VI. Conclusions and Final Remarks

A comprehensive evaluation of PUFs is often hampered by the difficulty of acquiring large datasets from different devices. As a direct consequence, we are witnessing new PUF proposals that are either evaluated on a limited amount of data or are slowed down by the impossibility of researchers

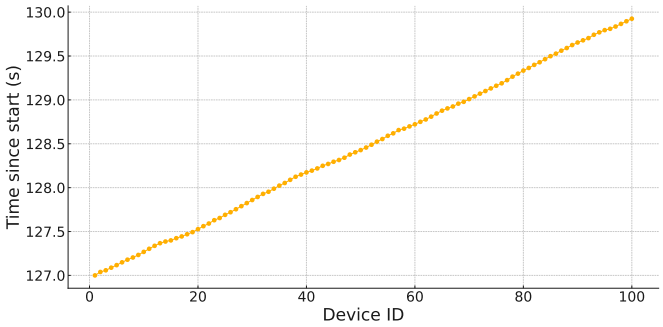


Fig. 12: Registration times of 100 emulated devices.

and developers to focus solely on implementation aspects of their novelty. To address this bottleneck which is stalling innovation in PUF-based hardware security, we present an innovative and flexible evaluation-system designed to automate the characterization of FPGA-based PUFs at scale. Our proposal introduces a groundbreaking approach to PUFs evaluation by combining modularity, automation, and ease of use. Its architecture enables seamless integration of a wide variety of PUF implementations, supporting users in evaluating critical performance metrics such as uniqueness, reliability, uniformity, and bit-aliasing across multiple devices. Moreover, since the system is designed to expose both raw and processed data, it is inherently suitable for integration with additional analysis pipelines. This includes, for example, the use of adversarial testing methodologies, such as machine learning-based modeling attacks, to assess the resilience of a PUF design against predictive models. Finally, its configurability allows researchers to design and conduct experiments tailored to specific PUF characteristics, thereby overcoming many of the limitations associated with current evaluating platforms. Looking ahead, we envision the integration of other devices to increase the current total number, also considering heterogeneous FPGA-hardware architectures along with PUFs beyond the FPGA domain, in order to further enhance the versatility of the system.

REFERENCES

[1] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on physical unclonable function (PUF)-based security solutions for Internet of Things," *Computer Networks*, vol. 183, p. 107593, Dec. 2020.

[2] P. Mall, R. Amin, A. K. Das, M. T. Leung, and K.-K. R. Choo, "PUF-Based Authentication and Key Agreement Protocols for IoT, WSNs, and Smart Grids: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8205–8228, Jun. 2022.

[3] M. Barbareschi, A. De Benedictis, and N. Mazzocca, "A PUF-based hardware mutual authentication protocol," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 107–120, Sep. 2018.

[4] M. Barbareschi, V. Casola, A. Emmanuele, and D. Lombardi, "A Lightweight PUF-Based Protocol for Dynamic and Secure Group Key Management in IoT," *IEEE Internet of Things Journal*, vol. 11, no. 20, pp. 32 969–32 984, Oct. 2024.

[5] R. Román, R. Arjona, and I. Baturone, "A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices," *Future Generation Computer Systems*, vol. 148, pp. 425–435, Nov. 2023.

[6] K. Lata and L. R. Cenkeramaddi, "FPGA-Based PUF Designs: A Comprehensive Review and Comparative Analysis," *Cryptography*, vol. 7, no. 4, p. 55, Dec. 2023.

[7] S. Gehrer and G. Sigl, "Using the reconfigurability of modern FPGAs for highly efficient PUF-based key generation," in *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, Jun. 2015, pp. 1–6.

[8] 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, Mar. 2018.

[9] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2010, pp. 94–99.

[10] 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*. New York, NY: Springer, 2013, pp. 245–267.

[11] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs," in *2010 International Conference on Reconfigurable Computing and FPGAs*, Dec. 2010, pp. 298–303, iSSN: 2325-6532.

[12] A. Wild, G. T. Becker, and T. Güneysu, "A fair and comprehensive large-scale analysis of oscillation-based PUFs for FPGAs," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017, pp. 1–7, iSSN: 1946-1488.

[13] V. P. Yanambaka, S. P. Mohanty, and E. Kougianos, "Making Use of Manufacturing Process Variations: A Dopingless Transistor Based-PUF for Hardware-Assisted Security," *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 2, pp. 285–294, May 2018.

[14] "ISO/IEC 20897-1:2020 - Information security, cybersecurity and privacy protection - Physically unclonable functions - Part 1: Security requirements."

[15] F. Zerrouki, S. Ouchani, and H. Bouarfa, "A survey on silicon PUFs," *Journal of Systems Architecture*, vol. 127, p. 102514, Jun. 2022.

[16] 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.

[17] N. N. Anandakumar, M. S. Hashmi, and M. A. Chaudhary, "Implementation of Efficient XOR Arbiter PUF on FPGA With Enhanced Uniqueness and Security," *IEEE Access*, vol. 10, pp. 129 832–129 842, 2022.

[18] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, "FPGA implementation of modeling attack resistant arbiter PUF with enhanced reliability," in *2017 18th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2017, pp. 313–318, iSSN: 1948-3287.

[19] W. Ge, S. Hu, J. Huang, B. Liu, and M. Zhu, "FPGA implementation of a challenge pre-processing structure arbiter PUF designed for machine learning attack resistance," *IEICE Electronics Express*, vol. 17, no. 2, pp. 20 190 670–20 190 670, 2020.

[20] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 9–14.

[21] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags," in *Proceedings of the Conference on RFID Security*, 2007.

[22] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, Jun. 2008, pp. 67–70.

[23] M. Barbareschi, G. Di Natale, F. Bruguier, P. Benoit, and L. Torres, "Ring oscillators analysis for security purposes in Spartan-6 FPGAs," *Microprocessors and Microsystems*, vol. 47, pp. 3–10, Nov. 2016.

[24] C. Gu, W. Liu, Y. Cui, N. Hanley, M. O'Neill, and F. Lombardi, "A Flip-Flop Based Arbiter Physical Unclonable Function (APUF) Design with High Entropy and Uniqueness for FPGA Implementation," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1853–1866, Oct. 2021.

[25] Z. He, W. Chen, L. Zhang, G. Chi, Q. Gao, and L. Harn, "A Highly Reliable Arbiter PUF With Improved Uniqueness in FPGA Implementation Using Bit-Self-Test," *IEEE Access*, vol. 8, pp. 181 751–181 762, 2020.

[26] K. Sahithi and N. Murty, "Delay based Physical Unclonable Function for Hardware Security and Trust," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2018, pp. 797–803.

[27] C. Aknesil and E. Dubrova, "An FPGA Implementation of 4×4 Arbiter PUF," in *2021 IEEE 51st International Symposium on Multiple-Valued Logic (ISMVL)*, May 2021, pp. 160–165, iSSN: 2378-2226.

[28] J. Yang, X. Yu, and R. Wei, "A low resource consumption Arbiter PUF improved switch component design for FPGA," *Journal of Physics: Conference Series*, vol. 2221, no. 1, p. 012011, May 2022, publisher: IOP Publishing.

[29] Y. Wang, C. Wang, C. Gu, Y. Cui, M. O'Neill, and W. Liu, "A Dynamically Configurable PUF and Dynamic Matching Authentication Protocol," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1091–1104, Apr. 2022.

[30] C. Gu, Y. Cui, N. Hanley, and M. O'Neill, "Novel lightweight FF-APUF design for FPGA," in *2016 29th IEEE International System-on-Chip Conference (SOCC)*, Sep. 2016, pp. 75–80, iSSN: 2164-1706.

[31] K. T. Mursi and Y. Zhuang, "Experimental Examination of Component-Differentially-Challenged XOR PUF Circuits," *Journal of Physics: Conference Series*, vol. 1729, no. 1, p. 012006, Jan. 2021, publisher: IOP Publishing.

[32] H. Idriss, T. Idriss, and M. Bayoumi, "A highly reliable dual-arbiter PUF for lightweight authentication protocols," in *2017 IEEE International Conference on RFID Technology & Application (RFID-TA)*. Warsaw: IEEE, Sep. 2017, pp. 248–253.

[33] J. Zhang and C. Shen, "Set-Based Obfuscation for Strong PUFs Against Machine Learning Attacks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 288–300, Jan. 2021.

[34] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, "Low-cost fortification of arbiter PUF against modeling attack," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4, iSSN: 2379-447X.

[35] A. Aguirre, M. Hall, T. Lim, J. Trinh, W. Yan, and F. Tehranipoor, "A Systematic Approach for Internal Entropy Boosting in Delay-based RO PUF on an FPGA," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2020, pp. 623–626, iSSN: 1558-3899.

[36] D. Deng, S. Hou, Z. Wang, and Y. Guo, "Configurable Ring Oscillator PUF Using Hybrid Logic Gates," *IEEE Access*, vol. 8, pp. 161 427–161 437, 2020.

[37] M. Garcia-Bosque, G. Diez-Senorans, C. Sanchez-Azqueta, and S. Celma, "Proposal and Analysis of a Novel Class of PUFs Based on Galois Ring Oscillators," *IEEE Access*, vol. 8, pp. 157 830–157 839, 2020.

[38] W. Yan, C. Jin, F. Tehranipoor, and J. A. Chandy, "Phase calibrated ring oscillator PUF design and implementation on FPGAs," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. Ghent, Belgium: IEEE, Sep. 2017, pp. 1–8.

[39] M. Choudhury, N. Pundir, M. Niamat, and M. Mustapa, "Analysis of a novel stage configurable ROPUF design," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2017, pp. 942–945, iSSN: 1558-3899.

[40] H. Rabiei, M. Kaveh, M. Reza Mosavi, and D. Mart, "MCRO-PUF: A Novel Modified Crossover RO-PUF with an Ultra-Expanded CRP Space," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 4831–4845, 2023.

[41] M. Karmakar, S. F. Naz, and A. P. Shah, "Fault-tolerant reversible logic gate-based RO-PUF design," *Memories - Materials, Devices, Circuits and Systems*, vol. 4, p. 100055, Jul. 2023.

[42] J. H. Anderson, "A PUF design for secure FPGA-based embedded systems," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. Taipei, Taiwan: IEEE, Jan. 2010, pp. 1–6.

[43] A. Ardakani and S. Baradaran Shokouhi, "A secure and area-efficient FPGA-based SR-Latch PUF," in *2016 8th International Symposium on Telecommunications (IST)*. Tehran, Iran: IEEE, Sep. 2016, pp. 94–99.

[44] A. Lotfy, M. Kaveh, D. Martín, and M. R. Mosavi, "An Efficient Design of Anderson PUF by Utilization of the Xilinx Primitives in the SLICEM," *IEEE Access*, vol. 9, pp. 23 025–23 034, 2021.

[45] I. Cicek and A. Al Khas, "A new read–write collision-based SRAM PUF implemented on Xilinx FPGAs," *Journal of Cryptographic Engineering*, vol. 13, no. 1, pp. 19–36, Apr. 2023.

[46] C. Gu, W. Liu, N. Hanley, R. Hesselbarth, and M. O'Neill, "A Theoretical Model to Link Uniqueness and Min-Entropy for PUF Evaluations," *IEEE Transactions on Computers*, vol. 68, no. 2, pp. 287–293, Feb. 2019.

[47] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 97–109, Jan. 2018.

[48] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley, "Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Apr. 2018, pp. 126–133.

[49] S. Vinagrero, H. Martin, A. de Bignicourt, E.-I. Vatajelu, and G. Di Natale, "SRAM-Based PUF Readouts," *Scientific Data*, vol. 10, no. 1, p. 333, May 2023, publisher: Nature Publishing Group.

[50] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2015, pp. 653–658, iSSN: 1558-1101.

[51] "IEEE Standards Association - 802.11n."

[52] E. Dubrova, "A Reconfigurable Arbiter PUF with 4 x 4 Switch Blocks," in *2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL)*, May 2018, pp. 31–37, iSSN: 2378-2226.

[53] stnolting, "FPGA-PUF," Nov. 2024, original-date: 2021-11-16T20:15:24Z. [Online]. Available: https://github.com/stnolting/fpga_puf

[54] Q. R. II, "PUF-Authentication-Protocol," Aug. 2024, original-date: 2023-12-06T16:21:27Z. [Online]. Available: https://github.com/qkramos2/PUF-Authentication-Protocol

Daniele Lombardi is a PhD student in Information Technology and Electrical Engineering at the University of Naples Federico II. At the same University, in 2021 he obtained his MSc degree cum laude in Computer Engineering. His research areas include hardware security, security protocols in IoT, safety critical systems.

Mario Barbareschi (Associate Member, IEEE) is a Tenured Assistant Professor of Computer Systems at the Department of Electrical Engineering and Information Technologies of the University of Naples Federico II. He received the Ph.D. in Computer and Automation Engineering in 2015 from the University of Naples Federico II. His research interests include Hardware Security and Trust, Approximate Computing, emerging technologies, and embedded systems. He participates in international projects, collaborating with academic institutions and several industrial partners. He has authored more than 80 peer-reviewed papers published in leading journals and international conferences.

Valentina Casola is an Associate Professor at the University of Naples Federico II, Italy. She got a Ph.D. in Computer Engineering from the Second University of Naples in 2004. Her research activities are both theoretical and experimental and focus on security methodologies to design and evaluate distributed systems, including cyber physical infrastructures, cloud systems and web services. These activities are led in cooperation with academic institutions and industrial partners within international projects. She has published more than 100 papers in journals, conference proceedings and books.

Elena Ioana Vatajelu (Senior Member, IEEE) received the Ph.D. degree from the Polytechnic University of Catalunya, Spain, in 2011. She is currently a full-time Researcher with CNRS on the design, test and reliability of integrated circuits with TIMA Laboratory, Grenoble, France. Her expertise is on the reliability and the robustness assessment, design-for-reliability, test strategies and security primitives for CMOS and beyond CMOS RAMs in traditional and non-von neumann computing paradigms.

Giorgio Di Natale received the PhD in Computer Engineering from the Politecnico di Torino in 2003. He works as Director of Research for the French National Research Center (CNRS), and he is the director of the TIMA laboratory in Grenoble since January 2021. His research interests include: hardware security and trust, reliability, fault tolerance, test. He was the chair of the TTTC from 2020 to 2021, Golden Core member of the Computer Society and Senior member of the IEEE.