Strong PUF Security Metrics: Sensitivity of Responses to Single Challenge Bit Flips

Wolfgang Stefani [1], Fynn Kappelhoff [1], Martin Gruber [2], Yu-Neng Wang [3], Sara Achour [3], Debdeep Mukhopadhyay [4], and Ulrich Rührmair [5]

[1] TU Berlin, Berlin, Germany. w.stefani@iosat.de and fynn.kappelhoff@gmail.com
[2] BMW Group, Munich, Germany. martin.gr.gruber@bmw.de
[3] Stanford University, Stanford, USA. wynwyn@stanford.edu and sachour@stanford.edu
[4] IIT Kharagpur, Kharagpur, India. debdeep@cse.iitkgp.ac
[5] TU Berlin, Berlin, Germany, and University of Connecticut, Storrs, USA. ruehrmair@ilo.de

***Abstract*—This paper belongs to a sequence of manuscripts that discuss generic and easy-to-apply security metrics for Strong Physical Unclonable Functions (PUFs). These metrics cannot and shall not fully replace in-depth machine learning (ML) studies in the security assessment of Strong PUF candidates. But they can complement the latter, serve in initial complexity analyses, and allow simple iterative design optimization. Moreover, they are computationally more efficient and far easier to standardize than typical ML-studies. This manuscript treats one very natural, but also very impactful metric, and investigates the effects that the alteration of *single* challenge bits has on the associated PUF-responses. We define several concrete metric scores based on this idea, and demonstrate their predictive power by applying them to various popular Strong PUF design families as test cases. This includes XOR Arbiter PUFs, XOR Bistable Ring PUFs, and Feed-Forward Arbiter PUFs, whose practical security is particularly well known after two decades of intense research. In passing, our manuscript also suggests techniques for representing our metric scores graphically, and for interpreting them in a meaningful manner. Our work demonstrates that if comparable methods had existed earlier, various Strong PUF candidates deemed secure and broken later could have been recognized and winnowed early on.**

***Index Terms*—Strong PUFs, Security Metrics, Modeling Attacks on PUFs, Machine Learning Attacks on PUFs**

I. Introduction

A. Motivation and Overview

In the last twenty years since their introduction [1], [2], Physical Unclonable Functions (PUFs) have exerted a strong influence on both cryptography and computer security. Within the various sub-forms of PUFs treated in the literature — including Weak PUFs [3], Controlled PUFs [4], Reconfigurable PUFs [5], SIMPLs/PPUFs [6] [7], or UNOs [8], amongst many others — so-called Strong PUFs arguably take a role of particular importance for various reasons [9], [10].

According to [9], they are characterized by the following definitional features: *(i)* Strong PUFs must possess a publicly accessible challenge-response mechanism. Everyone holding the Strong PUF or the Strong PUF embedding hardware should be able to apply arbitrary challenges to the Strong PUF, and to read out the resulting responses, without any access restrictions. *(ii)* Strong PUFs must possess a very large number of possible challenges – so many that despite the publicly accessible challenge-response interface, not all challenge-response pairs (CRPs) of the Strong PUF can be read out by adversaries in practice. *(iii)* The challenge-response relation of a Strong PUF should be very complex, so complex that unknown responses cannot be predicted numerically/computationally by adversaries, even if these adversaries know other CRPs of the Strong PUF.

Once a PUF meets these demanding properties, a vast spectrum of different applications opens up: Just naming three examples, Strong PUFs can be used for the remote identification of hardware that has no non-volatile memory on board, and that does not even carry dedicated cryptographic circuitry (except the Strong PUF itself) [1]. This is not possible by several other PUF sub-forms, such as Weak PUFs or Unique Objects [8], [9]. Secondly, when some of their challenges are fixed, Strong PUFs can serve as secret key storage elements, meaning that Strong PUFs could in principle also be used in any Weak PUF applications [11]. Finally, under the assumption that a Strong PUF is exchanged physically between communication partners, token-based cryptographic protocols for advanced schemes become possible. This includes cryptographic key exchange, bit commitment, oblivious transfer, or secure multi-party computation [12] [13].

While the broad applicability and potential impact of Strong PUFs are clear and well-understood, their secure *and* efficient realization in practice is not. Most existing silicon implementations suffer from one joint problem: They are not complex enough to prevent the most effective attack form on Strong PUFs, so-called machine-learning based modeling attacks [14]. In these attacks, an adversary with temporary PUF-access collects some fraction of CRPs, and uses this fraction to train a machine learning (ML) algorithm. If successful, the algorithm can later predict *all* PUF-responses and thus digitally emulate the Strong PUF. Contrary to the PUF, it can be cloned and distributed arbitrarily. This breaks essentially all protocols and applications built on the emulated Strong PUF.

In order to *"prove"* that a given Strong PUF candidate is secure against ML, one popular strategy consists of choosing some ML-algorithms that have broken other PUFs in the past, or that appear suitable for other reasons, and to verify empirically that they fail in breaking the PUF under test. This strategy possesses some inherent shortcomings, however, as already alluded to in the abstract: First of all, the fact that k ML-algorithms have failed on the PUF does not prove that an already existing, but yet untested $(k+1)$-th algorithm will fail, too, let alone that all future algorithms will. Secondly, the question whether a certain ML-method succeeds on a given PUF depends very much on aspects like the invested computation times, numbers of CRPs used in the training phase, or computational resources like size of RAM and processor power. These are hard to standardize, and it is unclear after which time (or after how many employed CRPs, etc.) an algorithm should be considered as failing. Thirdly, also the implementational details of the ML-algorithm play a very important role. This includes, among many other things, the question whether certain tailor-made transformations have been applied to the PUF-challenges prior to the ML training phase. It is known that such challenge transformations, which may incorporate the special properties of the PUF under attack, can dramatically improve ML-performance, and often are key to successful training phases. One prime example illustrating this effect are (XOR) Arbiter PUFs, where often only the use of a certain, meanwhile *"classical"* challenge transformation [15] [14] enables effective ML.

These circumstances make ML-based security testing necessary in practice, but also cumbersome and hard to standardize, sometimes even inconclusive or contradictive in its findings. The recent history of silicon Strong PUF candidates indeed offers various examples where initial ML-experiments had demonstrated a PUF's security, but where just slightly more sophisticated ML-attempts shortly thereafter broke the candidate [16] [14] [17] [18] [19] [20]. This calls for computationally efficient, easy-to-apply, and simple-to-standardize metrics that can complement ML-based security assessments, providing a different angle for Strong PUF evaluations than pure and exclusive ML-studies.

B. Related Work

The literature on systematic Strong PUF security metrics is surprisingly sparse up to this point. Some interesting initial insights can be found in [21] [22], where some first forms of single challenge bit flips and their effects are examined. In another related strand, sound mathematical representations of PUFs have been developed to assess their machine learnability in the Probably Approximately Correct (PAC) framework [23]. Subsequently, a CAD tool framework was proposed [24], which describes the PUF-composition in a formal language, called PUF-G, and automates the process of PAC learnability analysis on new PUF architectures. Finally, PUFmeter [25] was suggested to automate the analysis of PUF designs for evaluating ML resistance. It utilizes a PUF's average sensitivity, noise sensitivity, and k-junta testing to characterize its security. Finally, this paper is a very strongly extended, in-depth journal version of an earlier work by an overlapping set of authors from 2022 [26].

C. Our Contributions

This paper makes the following contributions:

- We present computationally efficient, easy-to-apply, and simple-to-standardize security metrics for Strong PUFs. They work in a black-box manner, and merely require measured/simulated CRPs of the Strong PUF under test, but no internal mathematical model of the PUF, nor any general knowledge on ML-techniques. We stress again that the metrics must not be mixed up with formal security guarantees, let alone security proofs, and that they are intended to complement, *not* to completely replace ML-based security evaluations.
- Concretely, our paper systematically investigates the effects that flipping single bits in PUF-challenges has on the resulting PUF-responses. We mathematically define various figures of merit (or *"scores"*) that measure and quantify these effects. These include the response flip probability caused by inverting the j-th challenge bit of a single Strong PUF instance ($= S_j$); the average of the S_j scores over multiple PUF-instances ($= \overline{S}_j$); the *"distance"* of a single PUF-instance to a hypothetical ideal Strong PUF, for which all S_j possess the value 0.5 ($= \mathsf{I2O}_1$); and the average of the $\mathsf{I2O}_1$ scores over multiple PUF-instances ($= \overline{\mathsf{I2O}}_1$).
- We propose that our metrics can be used for first security assessments of Strong PUF candidates (serving as *"PUF canaries"*), and also for the computationally efficient iterative design optimization of a given Strong PUF.
- We extensively apply our metrics to several popular Strong PUF families as test cases, namely k-XOR Arbiter PUFs and k-XOR Bistable Ring PUFs for growing k, as well as Feed-Forward Arbiter PUFs for growing number of loops l, using noise-free, numerically simulated CRPs from pypuf [27]. We then compare the obtained metric scores on our test cases to the well-studied real-world security of these designs, finding that they match *extremely* well. In passing, we show how our metric scores can be interpreted and graphically represented in the best fashion.
- Our findings demonstrate that our metrics could have unearthed the security vulnerabilities of many previous Strong PUF designs early on, avoiding periods of false security beliefs, if they had already existed at the time of their introduction.
- In the Supplementary Material, we prove two central mathematical results on our metrics: First of all, the exact relation between our above score $\overline{\mathsf{I2O}}$ and another natural figure of merit termed $\mathsf{A2O}_1$, making the latter obsolete in Strong PUF assessments. This greatly simplifies our analyses. Secondly, we prove an unexpected relation between our bit flip probabilities S_j and the challenge-bitwise nonlinearity features of a tested Strong PUF.
- Finally, we make all code for computing and graphically representing our metric scores publicly available [28], fostering their simple application and follow-up works.

II. Background: Examined Strong PUF Families and Numeric CRP-Generation

A. k-XOR Arbiter PUFs

The plain Arbiter PUF [11] consists of a chain of n consecutive *"stages"* (each built of two multiplexers), which are traversed by two parallel electrical signals from left to right. The individual paths of the two signals are determined by n external bits applied at the stages: If the external bit at a stage is 0, the two signals traverse this stage in parallel, maintaining their upper/lower positions; if the external bit is 1, then the two signal paths cross each other, with the upper signal moving down, the lower signal moving up. These n external bits together constitute the applied PUF-challenge C_i. The PUF-response R_i consists of just a single bit: This bit indicates whether the upper signal or the lower signal arrived first at a so-called *"arbiter element"* or latch at the right end of the structure, i.e., after the last stage.

It has been observed early [15] that plain Arbiter PUFs are highly susceptible to machine learning (ML) based modeling attacks (see Section I-A). As a countermeasure, so-called k-XOR Arbiter PUFs have been suggested [29]: k Arbiter PUFs are employed simultaneously and "in parallel". Exactly the same n-bit challenge C_i is applied to all of them, and their k individual responses are XORed to produce the final, single-bit response R_i. (Please bear in mind that formally speaking in this notation, a k-XOR Arbiter PUF for $k = 1$ and a "plain", normal Arbiter PUF are simply the same.)

According to all we currently know, this design trick for sufficiently large k protects k-XOR Arbiter PUFs against all those ML-attacks that merely use plain, single CRPs (and no other information) in the ML-training phase: Increasing k linearly seems to exponentially increase the computational effort of such attacks [14]. Furthermore, increasing the challenge length n seems to raise the ML-efforts only polynomially in this attack type [14]. There are two important caveats, though: Firstly, incrementing k also raises the instability of a k-XOR Arbiter PUF exponentially in k. This obviously puts limits on the practically usable values of k. Secondly, additional side-channel information can exponentially boost ML performance, re-enabling polynomial-time ML-attacks on k-XOR Arbiter PUFs. For example, power side channels that reveal the cumulative number of 0's and 1's in the input to the final XOR gate [30] re-enable polynomial attacks. Also utilizing the stability levels of each single CRP in repeated measurements allows polynomial attack performance [31].

We stress that we will *not* use ML-performance under side-channel information as comparative benchmark in this paper, though. One straightforward reason is that our generic, black-box type metrics cannot take such additional information into account (yet), at least not in a general manner and for arbitrary side channels. Secondly, we felt that a Strong PUF design itself should not be penalized for extra *physical* and *implementation-dependent* attack vectors. Also in the analysis of AES or other symmetric primitives, such physical vectors would be analyzed separately from the primitives' digital complexity and security, as we do in this paper.

B. Feed-Forward Arbiter PUFs with l Loops

The Feed-Forward Arbiter PUF (FF Arb PUF) was introduced immediately after the susceptibility of the plain Arbiter PUF against simple ML-based modeling attacks, in particular against support vector machines (SVMs), had been discovered [16]. One of its goals was to prevent these attacks by additional design features. To this end, a normal Arbiter PUF is amended with some additional, so-called *"feed-forward loops"*. In these loops, an "intermediate time" or "split time" between the two competing electrical signals is taken at some point in the sequence of stages of the Arbiter PUF chain. For this purpose, the two signals are split or branched between two stages: One *"copy"* of each signal continues to race on from left to right in the sequence of stages. The other copy of the signals is fed into one *additional* arbiter element, which determines which of the two signals was faster at that point, and correspondingly outputs a 0 or 1. This output bit is then used as external input bit to a dedicated additional stage further down to the right in the sequence of stages. Please note that this additional stage has no external challenge bit applied, but is only switched by the output bit of the feed-forward loop. This means that a FF Arb PUF with l loops and n-bit challenges has $n + l$ stages; at n of these, external challenge bits are applied, and at l of these, outputs of the arbiter elements in the loops are applied.

While a plethora of loop numbers and loop architectures (adjacent, overlapping, nested, mixtures of all three, etc.) are conceivable in FF Arb PUFs, we employ for comparability the same architecture as in earlier studies [14], [32]: The l loops all have equal sizes, and are distributed equidistantly over the entire length of the Arbiter PUF. At that, the starting and ending points of consecutive or neighbouring loops just overlap. It is known from earlier ML-experiments that when using the *"right"* ML-algorithms, attack performance on these architectures is polynomial both in the challenge length n and in the number of loops k [14]. Further details on the specific loop structures investigated by us are provided right during our analyses in Section IV-B on page 9.

C. k-XOR Bistable Ring PUFs

Our third example design, which leaves the realm of the Arbiter PUF family, is the so-called Bistable Ring PUF (BR PUF) [33]. It consists of a double ring of inverters segmented into n stages, with two inverters per stage. By applying external challenge bits at each stage, one of the two inverters is selected. All selected inverters are then linked or connected in order to form a ring. For an even number of stages/connected inverters, this ring is a bistable system, and can converge exactly to two possible states after being powered up. Which of the two states it finally takes depends on the selected inverters in each stage and on their manufacturing variations, and defines the response of the BR PUF. This implies that a BR PUF with n stages (and $2n$ inverters) has n-bit challenges and 1-bit responses.

Test 1 (Single Challenge Bit Flips on a Single PUF-Instance).

Under Test:

- A single PUF-instance P with n-bit challenges $C_i = b_i^1 \cdots b_i^n$ and single-bit responses R_i.

Test Scores:

- n numbers $S_1(\mathsf{P}), \ldots, S_n(\mathsf{P}) \in [0, 1]$. Each $S_j(\mathsf{P})$ gives the probability that the response R_i of the PUF-instance P to a uniformly randomly chosen challenge C_i will *"flip"* (i.e., will turn from R_i into $R_i \oplus 1$) when the j-th challenge bit b_i^j of C_i is *"flipped"* (i.e., is turned from b_i^j into $b_i^j \oplus 1$). [a]
- One number $\mathsf{I2O}_1(\mathsf{P}) \in [0, 0.5]$, where the acronym stands for *"single-flip instance-to-optimum score"*. It calculates the mean absolute distance of all test scores $S_j(\mathsf{P})$ to the ideal value 0.5:

$$\mathsf{I2O}_1(\mathsf{P}) := \frac{|S_1(\mathsf{P}) - 0.5| + \ldots + |S_n(\mathsf{P}) - 0.5|}{n} \quad (1)$$

- *Notation:* Whenever P is clear from the context we may drop it, simply writing S_j and $\mathsf{I2O}_1$ for brevity.

Estimating $S_1, \ldots, S_n$ and $\mathsf{I2O}_1$ in Practice:

- Any S_j can be estimated by the following method:
  - s challenges $C_1, \ldots, C_s$ are chosen uniformly at random from P's challenge space.
  - The challenges $C_1, \ldots, C_s$ are applied to P. Then in all these challenges $C_1, \ldots, C_s$, the j-th bit is flipped, and the so-obtained challenges $C_1^*, \ldots, C_s^*$ are applied to P. Finally, S_j is calculated as the fraction of responses that flipped when C_i^* was applied instead of C_i.
- When estimating different S_j, usually the same challenge set $C_1, \ldots, C_s$ is employed for simplicity and yet better comparability.
- An estimated value for $\mathsf{I2O}_1$ can then be calculated from the estimated S_j, following Eqn. (1).
- Based on our experience with the PUF-designs of this paper, statistically recommendable values are $s = 1{,}000$ or larger, if possible. Extreme values of $s = 100$ should only be used in very first *qualitative* analyses, and if strictly necessary.

Ideal Test Scores and Simple Interpretation:

- The ideal score for all S_j is 0.5. Deviations (both larger or smaller values within $[0, 1]$) can signal potential weaknesses of the considered PUF-instance. They may also indicate that the effective challenge space of the PUF-instance is reduced, in particular for any S_j very close to 0 or 1.
- The ideal score for $\mathsf{I2O}_1$ is 0. Deviations (i.e., larger values within $[0, 0.5]$) can again signal potential weaknesses of the examined PUF-instance.

[a]We emphasize explicitly that said probability is taken over the uniformly random choice of C_i from P's challenge space, but not over any manufacturing variations, as only one fixed PUF instance is considered here.

As before, a k-XOR BR PUF can be formed by using k BR PUFs in parallel, applying the same external challenge to all of them, and by computing the final response by XORing the individual responses of all k BR PUFs. Even though the BR PUF is one of the most-cited Strong PUF designs [33], the ML-complexity of k-XOR BR PUFs has not been studied with the same intensity as for k-XOR Arbiter PUFs. Existing ML-experiments seem to indicate a similar trend, though: According to all we know, the best ML-performance for k-XOR BR PUFs is exponential in k and polynomial in the challenge length n [34], [35]. This holds at least for ML-attacks that merely use *"plain"* CRPs without any side channel information, a considered by us in this paper (compare end of Section II-A).

D. Simulation of CRP-Data

The PUF CRP data for all figures and analyses in this paper was generated with the python package pypuf (version 3.2.1) [27]. We use non-noisy simulations, i.e., our numerically generated CRPs are perfectly stable. All internal parameters/manufacturing variations of the PUF are chosen independently via a standard normal distribution.

III. Definition of Our Security Metrics

The sensitivity of PUF-responses to flipping single bits in the PUF-challenges could be seen as one of the most suggestive security and quality metrics for Strong PUFs. We formally define an associated Test 1 and two related numeric scores in the grey box above, namely S_j and $\mathsf{I2O}_1$. Please note that Test 1 is dedicated to examining *single* PUF-instances. To deal with r instances (from the same PUF-design and with the same challenge length), Test 2 is introduced on page 5. It adds the averaged numeric scores $\overline{S}_j$, $\overline{\mathsf{I2O}}_1$, and $\mathsf{A2O}_1$ to our toolbox.

Let us elaborate a bit further on these two tests and their interpretation. For a hypothetical *"ideal"* Strong PUF instance P with n-bit challenges, the bit flip probabilities $S_j(\mathsf{P})$ for $j = 1, \ldots, n$ should all be equal to 0.5, implying that $\mathsf{I2O}_1(\mathsf{P})$ will be equal to 0. A noticeable deviation from these ideal values *can* signal potential security weaknesses of the tested PUF-instance against ML-based modeling attacks. Note that deviations in the scores of single PUF-instances may result from a non-optimal general PUF design under test, from the effects that individual manufacturing variations have on the tested instance, or from a combination of both.

Test 2 (SINGLE CHALLENGE BIT FLIPS ON r PUF-INSTANCES).

Under Test:

- r PUF-Instances $\mathsf{P}_1, \ldots, \mathsf{P}_r$ from the same PUF-design, all with n-bit challenges $C_i = b_i^1 \cdots b_i^n$ and single-bit responses R_i.

Test Scores:

- n numbers $\overline{S}_1, \ldots, \overline{S}_n \in [0,1]$. Each $\overline{S}_j$ denotes the arithmetic mean of the r-element set $\{S_j(\mathsf{P}_1), \ldots, S_j(\mathsf{P}_r)\}$:
$$\overline{S}_j := \frac{S_j(\mathsf{P}_1) + \ldots + S_j(\mathsf{P}_r)}{r} \qquad (2)$$
- One number $\overline{\mathsf{I2O}_1} \in [0, 0.5]$. It denotes the arithmetic mean of the r-element set $\{\mathsf{I2O}_1(\mathsf{P}_1), \ldots, \mathsf{I2O}_1(\mathsf{P}_r)\}$:
$$\overline{\mathsf{I2O}_1} := \frac{\mathsf{I2O}_1(\mathsf{P}_1) + \ldots + \mathsf{I2O}_1(\mathsf{P}_r)}{r} \qquad (3)$$
- One number $\mathsf{A2O}_1 \in [0, 0.5]$, where the acronym stands for *"single-flip average-to-optimum score"*. It gives the mean absolute deviation between the arithmetic mean $\overline{S}_j$ and the optimal value 0.5:
$$\mathsf{A2O}_1 := \frac{|\overline{S}_1 - 0.5| + \ldots + |\overline{S}_n - 0.5|}{n} \qquad (4)$$

Estimating $\overline{S}_j$, $\overline{\mathsf{I2O}_1}$ and $\mathsf{A2O}_1$ in Practice:

- Any $\overline{S}_j$ can be estimated as follows:
  - The values $S_j(\mathsf{P}_1)$, ..., $S_j(\mathsf{P}_r)$ are estimated by the method described in Test 1.
  - $\overline{S}_j$ is then estimated by Eqn. (2).
- The value $\overline{\mathsf{I2O}_1}$ can be estimated as follows:
  - The values $\mathsf{I2O}_1(\mathsf{P}_1), \ldots, \mathsf{I2O}_1(\mathsf{P}_r)$ are estimated by the method described in Test 1.
  - $\overline{\mathsf{I2O}_1}$ is then estimated by Eqn. (3).
- The value $\mathsf{A2O}_1$ can be calculated on the basis of the estimated $\overline{S}_j$, following Eqn. (4).

Ideal Test Scores and Simple Interpretation:

- The ideal score for $\overline{S}_j$ is 0.5. For $\overline{\mathsf{I2O}_1}$ and $\mathsf{A2O}_1$ it is 0.
- Deviations from these ideal values (within the possible ranges $[0,1]$ or $[0,0.5]$) can again signal potential weaknesses of the examined PUF-instances and/or the underlying PUF-design.
- For increasing values of r, the above scores gradually become design-specific figures rather than instance-specific ones. Based on our experience with the PUF-designs of this paper, statistically recommendable values are $r = 100$ or larger, if possible.

Similar statements hold for the analysis of r PUF-instances in Test 2: If all were hypothetical *"ideal"* instances, the score $\overline{S}_j$ would be equal to 0.5 for all $j = 1, \ldots, n$, whilst $\overline{\mathsf{I2O}_1} = \mathsf{A2O}_1 = 0$. Once more, noticeable deviations from the ideal values *can* signal potential security weaknesses the tested design. For larger r, such deviation will predominantly result from the non-optimalites of the general design under test; the findings become increasingly design-specific, less instance-specific. Interestingly, a mean score $\overline{S}_j = 0.5$ does not necessarily imply that for all r considered single instances $S_j = 0.5$. A meaningful analysis therefore should not only take the mean values $\overline{S}_j$ into account, but also the statistical spread of the single S_j values across the r instances. We will follow this paradigm in almost all of our later analyses: For example in the statistical box plots of Figures 3, 4, 7, 10, 13, 14, which are based on many PUF-instances, or in Figures 1, 2, 5, 8, 11, 12, which show curves from several PUF-instances side by side for better comparison.

Besides initial assessments of PUF-complexity, the S_j (or $\overline{S}_j$) values also allow statements on the effective challenge space size of an examined PUF-instance (or examined PUF-design, respectively). If many S_j (or $\overline{S}_j$) are close to 0, then the corresponding bit positions in the challenge have no notable effect on the responses; they more or less do not matter and can be excluded. Something similar holds for challenge bit positions whose corresponding values S_j (or $\overline{S}_j$) are close to 1: Their effect on the responses can be approximated by merely taking their parity. Both can simplify attacks.

Finally, we vividly stress again that even ideal test scores (e.g., all $S_j = \overline{S}_j = 0.5$ and $\mathsf{I2O}_1 = \overline{\mathsf{I2O}_1} = \mathsf{A2O}_1 = 0$) do not unconditionally prove the security of a PUF-design or PUF-instance. But relatively *"bad"* scores (apart from pathological and constructed cases) usually can be seen as a negative criterion, which may signal Strong PUF vulnerabilities early on. Such PUFs can then either be improved in an iterative design process, possibly using our metrics as evaluation scores in each design loop. Or they can be winnowed and sorted out in due time, if such improvement turns out impossible.

We will now apply Tests 1 and 2 to various popular Strong PUF families to prove the usefulness of our metrics. The results are presented and discussed over the next subsections.

IV. Application of Our Security Metrics to Popular Strong PUF Families

A. Single Bit Flip Test on k-XOR Arbiter PUFs

We first applied Tests 1 and 1 to six individual Arbiter PUF instances of challenge length 64. Figure 1 illustrates the resulting response flip probabilities S_j plus $\mathsf{I2O}_1$ and $\overline{\mathsf{I2O}_1}$ scores. The S_j increase monotonically with the position j of the flipped bit within the challenge. Their values range

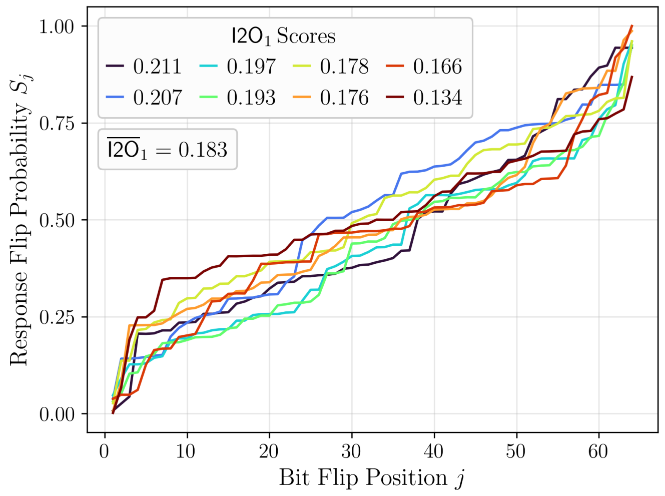

Fig. 1: Response flip probability S_j vs. bit flip position j for eight Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

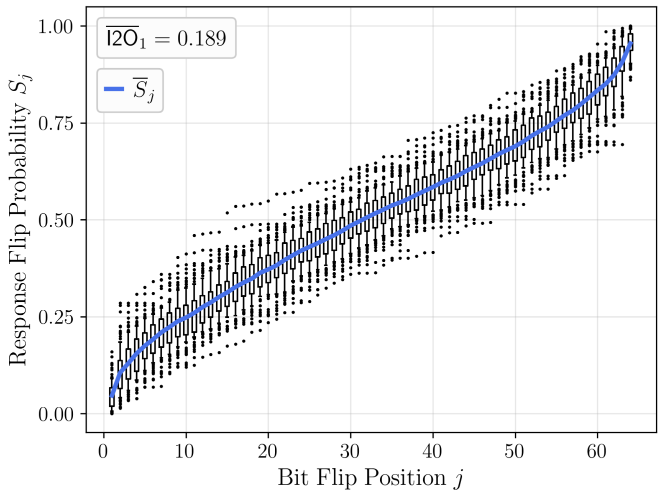

Fig. 3: Response flip probability S_j vs. bit flip position j for hundred Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$. The box plot's whiskers range between the 10th and 90th percentile.

from close to zero (at the very left) to close to one (at the very right). In large parts, but not at its very left and right edges, this leads to a quasi-linear curve. This finding is perhaps somewhat unexpected, but in agreement with earlier literature [36]: The challenge bits in an Arbiter PUF obviously do not have equal influence on its responses. Furthermore, only very few challenge bits are close to the ideal value 0.5, signalling potential vulnerabilities. The IO2$_1$ scores vary somewhat statistically around their (substantially large) mean value $\overline{\text{I2O}_1} = 0.183$.

Next, we applied Test 1 and 2 to 100 Arbiter PUF instances. Figure 3 subsumes the statistics, showing boxplots on the S_j scores, together with the average scores $\overline{S}_j$ and $\overline{\text{I2O}_1}$. While there are (for every position j) notable outliers of S_j-values beyond the boxplot's two whiskers (at the 10th and 90th percentile), an *almost* linear increase in the mean scores $\overline{S}_j$ can be identified with growing j. As before, the curve becomes notably steeper towards its left and right edges. Note that both Figures 1 and 3 exhibit very similar $\overline{\text{I2O}_1}$ scores of 0.183 and 0.189; the small difference is caused by expectable statistical variations.

Subsequently, we ran Tests 1 and 2 on eight k-XOR Arbiter PUFs of challenge length 64 for $k = 2$ and $k = 3$, showing our results in Figures 2 and 5. Complementary large-scale statistics

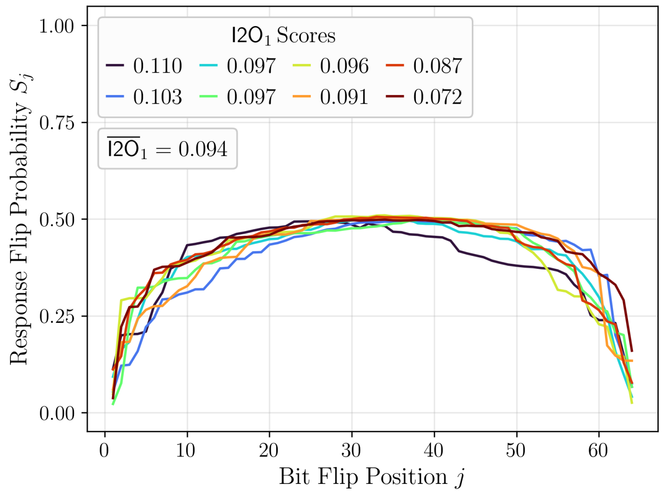

Fig. 2: Response flip probability S_j vs. bit flip position j for eight 2-XOR Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

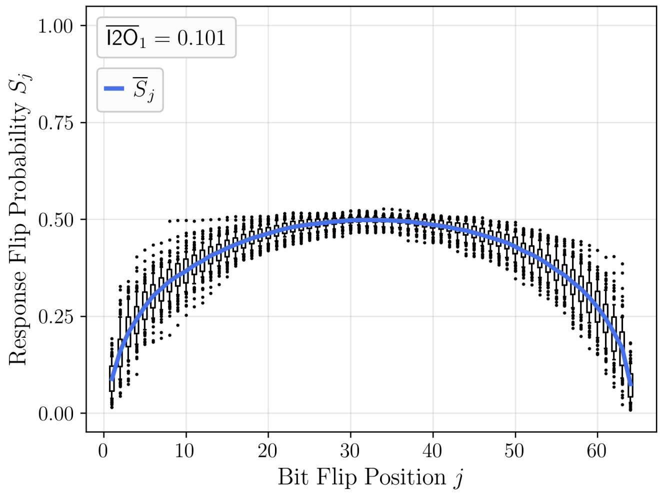

Fig. 4: Response flip probability S_j vs. bit flip position j for hundred 2-XOR Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$. The box plot's whiskers range between the 10th and 90th percentile.

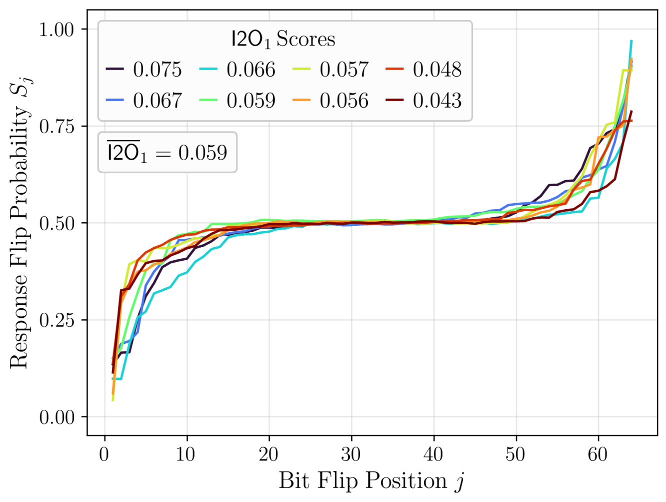

Fig. 5: Response flip probability S_j vs. bit flip position j for eight 3-XOR Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

over one hundred instances are subsumed in the boxplots of Figures 4 and 7. These four figures jointly can be seen as first indication of a more general trend: Increasing k successively yields test scores that are closer and closer to the optimal values. For larger k, the resulting S_j and $\overline{S}_j$ get closer to 0.5 over the entire challenge length, i.e., for all possible j. At the same time, the $\mathsf{I2O}_1$ and $\overline{\mathsf{I2O}}_1$ slowly, but steadily approach 0. This phenomenon is yet more systematically examined in Figure 6 and Table I, where we study the $\overline{\mathsf{I2O}}_1$-values of k-XOR Arbiter PUFs for up to $k = 20$. They show that our metrics can discriminate exceptionally well between k-XOR Arbiter PUF and $(k+1)$-XOR Arbiter PUF, even for large k. This extreme discriminativity could be seen as positive indication for the quality and precision of our metrics.

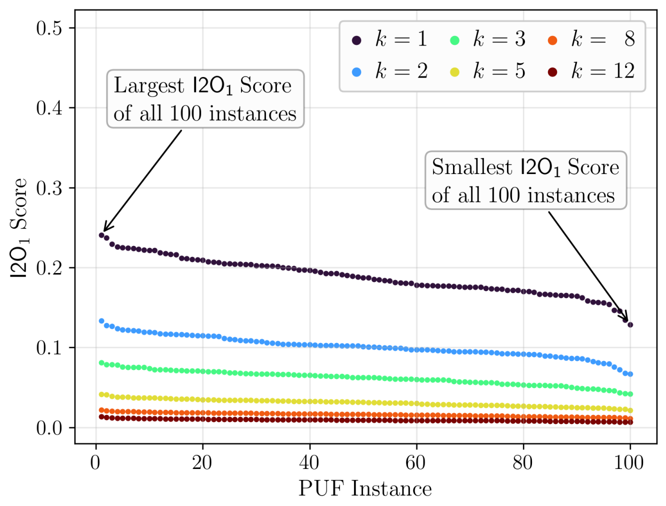

Fig. 6: Descending $\mathsf{I2O}_1$ Scores for 100 k-XOR Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

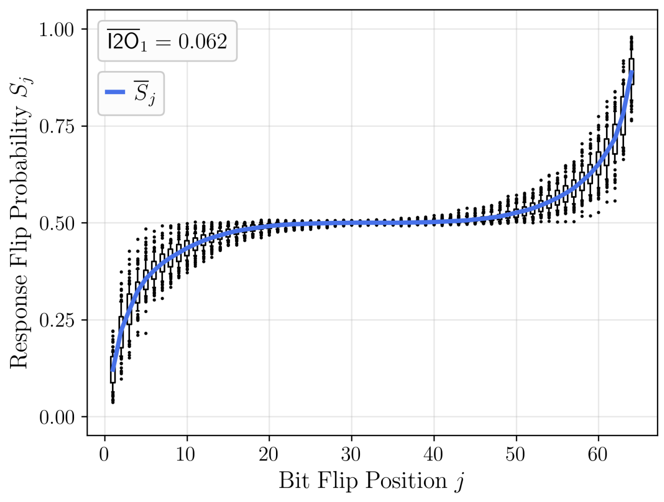

Fig. 7: Response flip probability S_j vs. bit flip position j for hundred 3-XOR Arbiter PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$. The box plot's whiskers range between the 10th and 90th percentile.

Please note that the special way of ordering the randomly generated PUF-instances along the x-axis in Figure 6 (namely

k-XOR Arbiter PUFs		$\overline{\mathsf{I2O}}_1$		
		Number of PUF-Instances		
		10	100	1000
	1	0.18171	0.18866	0.19015
	2	0.09490	0.10127	0.10031
	3	0.05868	0.06190	0.06223
	4	0.04043	0.04202	0.04278
	5	0.02978	0.03096	0.03157
	6	0.02294	0.02378	0.02454
	7	0.01851	0.01927	0.01990
	8	0.01564	0.01613	0.01644
	9	0.01419	0.01366	0.01401
k	10	0.01204	0.01212	0.01206
	11	0.01077	0.01058	0.01051
	12	0.00974	0.00925	0.00927
	13	0.00911	0.00845	0.00832
	14	0.00771	0.00758	0.00745
	15	0.00630	0.00675	0.00675
	16	0.00611	0.00618	0.00617
	17	0.00571	0.00560	0.00563
	18	0.00512	0.00515	0.00519
	19	0.00501	0.00485	0.00482
	20	0.00464	0.00441	0.00448

TABLE I: $\overline{\mathsf{I2O}}_1$ scores for various k-XOR Arbiter PUFs, averaged over different numbers of PUF-instances. The same, fixed challenge set of size $s = 10^5$ was used for computing each single $\mathsf{I2O}_1$-value.

by descending $\mathsf{I2O}_1$-scores) allows grasping the distribution of metric scores over all instances at one glance, even for very large numbers of examined instances. Albeit this simple trick of re-ordering the x-axis appears straightforward, we must admit that we have never encountered it before in the literature. We will also apply it later in Figures 9 and 15.

Another interesting observation is that the statistical spread of our metric scores — for example of the $\mathsf{I2O}$-scores in Figure 6, or of the S_j-scores and boxplots in Figures 1 to 5 and 7 — becomes *"flatter"* for increasing k in k-XOR Arbiter PUFs. This means that the chance of obtaining a low-quality, vulnerable outlier in PUF-production becomes successively smaller. I.e., with increasing k, we observe less variations in the quality of fabricated k-XOR Arbiter PUF instances.

Let us also have a word on Table I. It systematically investigates the behavior of the $\overline{\mathsf{I2O}_1}$ score of k-XOR Arbiter PUFs for growing k, and for increasing number of examined PUF-instances. The same challenge set of size $s = 10^5$ is employed for each PUF-instance and for computing each single $\mathsf{I2O}_1$-score to ensure comparability. The table illustrates the extreme numeric stability of the growing $\overline{\mathsf{I2O}_1}$-values for ever larger k. Even for only ten examined PUF-instances, these values can easily discriminate k-XOR Arbiter PUF designs for up to $k = 20$, indicating the quality of our metrics.

Readers will probably have already observed that the shapes of the shown curves in Figures 1, 5, 10, 7 (for k-XOR Arbiter PUFs with *odd* k) and Figures 2, 4 (for k-XOR Arbiter PUFs with *even* k) systematically differ. *Odd* k lead to curves that are point-symmetric around the center point $(32.5, 0.5)$ in (x, y)-coordinates. For *even* k, the *right* halves of the S_j- and $\overline{S}_j$-curves seem to be *"mirrored downwards"* at the $y = 0.5$ line. This creates axisymmetric curves overall, with a vertical symmetry axis located at $x = 32.5$. Let us focus on the case $k = 2$ for a closer explanation of this interesting phenomenon. It can be attributed to the general functionality of the XOR operation: Recall here that the 2-XOR Arbiter PUF consists of two independent, parallel Arbiter PUFs. If the last challenge bit is flipped, it is highly likely that the response of *both* of these two independent Arbiter PUFs will flip, too. These two flips will then cancel out in the final XOR gate, however; therefore with high probability, the same, unchanged PUF-response results. A similar logic can be extended to all XOR Arbiter PUFs with even k. For uneven k, on the other hand, this effect is exactly inverse: If an uneven amount of parallel Arbiter PUFs all flip, then the final response after the XOR will necessarily flip, too. As expected, both effects become weaker for larger k, and the $\overline{S}_j$ overall move closer to the ideal value 0.5. Again, this is empirically confirmed for up to $k = 20$ in Table I.

All these findings, which are delivered almost automatically and in a simple-to-standardize fashion by our metrics, are in strong agreement with the longly and laboriously studied practical security of k-XOR Arbiter PUFs against ML: For example with the known insecurity of plain Arbiter PUFs and k-XOR Arbiter PUFs for small k, or with the increasing ML-resilience of k-XOR Arbiter PUFs for growing k [14], [32]. Please also compare Section II-A in this context.

B. Single Bit Flip Test on Feed-Forward Arbiter PUFs

Let us now apply Tests 1 and 2 to another well-studied Strong PUF candidate from the Arbiter PUF family, namely FF Arb PUFs. As already stated in Section II-B, the FF Arb PUF historically was proposed in 2004 to thwart the simple ML-attacks on the original Arbiter PUF, such as the application of SVMs [16]. Since it succeeded at that, it was assumed secure for several years, until it was broken around six years after its introduction by another ML-technique known as Evolution Strategies (ES) [14].

Please recall here that the number and positions of the loops in a FF Arb PUF are variable design parameters (see again Section II-B). We will examine an exemplary FF Arb PUF with eight loops and challenge length 64 bits in the following. It is identical to the *"classical"* FF Arb PUF designs already studied in previous works [14], [32]: All eight loops have equal and maximal length, the ending point of the previous loop and the starting point of the new loop just about overlap (meaning that the next loop starts between stage n and $n + 1$ if the previous loop ends in stage $n + 1$), and the very last loop ends in the last stage. More precisely, the starting points and ending points of each of our eight loops can be written as (**6**, 15), (**14**, 22), (**21**, 29), (**28**, 36), (**35**, 43), (**42**, 50), (**49**, 57), (**56**, 64). In this notation, the first entry in each tuple gives the *challenge bit* (not the stage!) after which the two signals are branched and the respective loop starts. The second entry gives the *challenge bit* (not the stage!) after which the output of the respective feed-forward loop is fed into a new, dedicated stage, which employs the loop's output as its sole configuring input bit. Slightly jumping ahead, this notation will be most useful for understanding the dotted lines of Figure 8: They precisely coincide with the starting/ending points of the loops in our above notation.

We then executed Test 1 and 2 on six randomly chosen FF Arb PUFs instances of exactly this loop structure. The results are shown in Figures 8 and 10. They already signal first weaknesses in the FF Arb PUF's design at one glance, since many S_j substantially deviate from the ideal value of 0.5. Those areas of maximal deviation are sharply demarcated by the starting and ending points of some of the individual loops of the examined design. The resulting $\mathsf{I2O}_1$ scores are weak, even comparable to the Arbiter PUF, clearly signalling potential vulnerabilities of the FF Arb PUF. This illustrates the predictive power of our metrics: Had they been known at the time of introduction of the FF Arb PUF in 2004 [16], they would have signalled potential vulnerabilities early on. One could have reasonably concluded that sooner or later, efficient ML-attacks would loom [14], even if the prevailing method of SVMs at the time could not break FF Arb PUFs. Assessments like these are one of the main purposes of our metrics.

Please recall that in Section IV-A, we examined the metric scores of k-XOR Arbiter PUFs for growing k. This suggests a similar analysis for growing numbers of loops l in the FF Arb PUF. Following these traces, we applied Tests 1 and 2 to

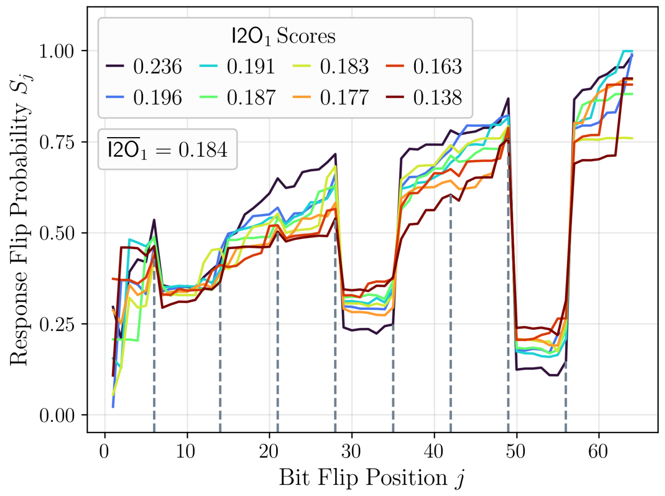

Fig. 8: Response flip probability S_j vs. bit flip position j for six FF Arb PUF instances, each with eight feed-forward (FF) loops and challenge length 64, using a fixed challenge set of size $s = 10^5$. The starting points of the FF loops are marked by dotted vertical lines.

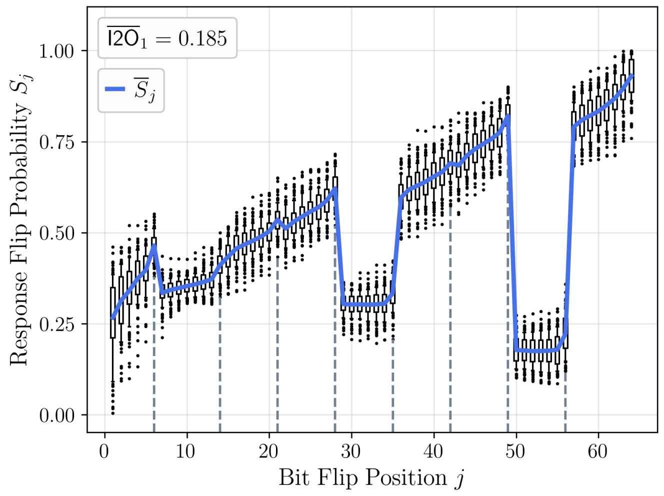

Fig. 10: Response flip probability S_j vs. bit flip position j for 100 FF Arb PUF instances with eight FF loops and challenge length 64, using a fixed challenge set of size $s = 10^5$. The box plot whiskers range between the 10th and 90th percentile. The starting points of FF loops are marked by dotted lines.

FF Arb PUFs with growing numbers of loops $l = 1, \ldots, 10$. The loops are placed via the same rationale as above: They all just overlap (meaning that the next loop starts right before the previous loop ends), are distributed as uniformly as possible over the entire chain of stages, and have equal length. In this construction principle, the entire loop structure is already completely specified by describing the last loop. For full exactness of our exposition, let us do so for all examined FF Arb PUF architectures, where $l = 1, \ldots, 10$:

$$\begin{array}{lll} l = 1 : (2, 64) & l = 5 : (54, 69) & l = \ \ 9 : (64, 73) \\ l = 2 : (33, 66) & l = 6 : (57, 70) & l = 10 : (65, 73) \\ l = 3 : (44, 67) & l = 7 : (60, 71) & \\ l = 4 : (50, 68) & l = 8 : (62, 72) & \end{array}$$

Figures 9 and Table II display our results. They illustrate that *neither* the statistical spread of scores over a population of 100 FF Arb PUFs, *nor* the distribution of $\overline{\mathsf{I2O}_1}$ scores over many instances, change notably with growing numbers of loops l. The parameters l in a FF Arb PUF and k in a k-XOR Arbiter PUF thus have much differing effects: Increasing l hardly affects ML-security and PUF-quality at all.

Again, all the above findings are in strong agreement with

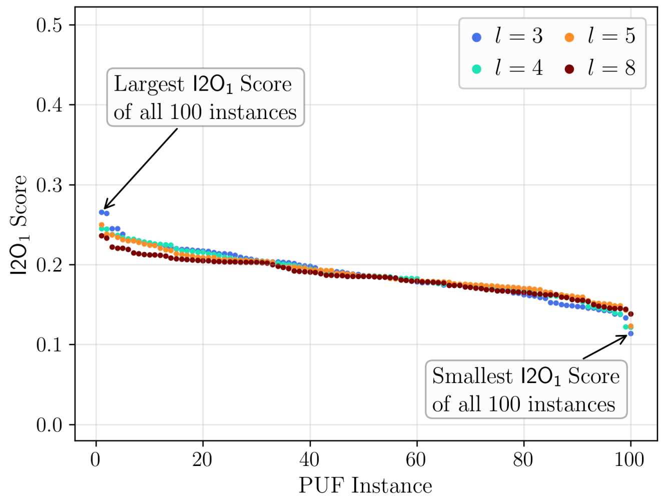

Fig. 9: Descending $\mathsf{I2O}_1$ Scores for 100 FF Arbiter PUF instances with l feed-forward loops and challenge length 64, using a fixed challenge set of size $s = 10^5$.

FF Arb PUF (with l loops)		$\overline{\mathsf{I2O}_1}$		
		Number of PUF-Instances		
		10	100	1000
l	1	0.17905	0.18725	0.18353
	2	0.17777	0.17981	0.17761
	3	0.18398	0.18835	0.18330
	4	0.18077	0.18853	0.18355
	5	0.17996	0.18882	0.18245
	6	0.18110	0.18613	0.18284
	7	0.18009	0.18320	0.18094
	8	0.17981	0.18486	0.18174
	9	0.18382	0.18454	0.18189
	10	0.18115	0.18322	0.18031

TABLE II: $\overline{\mathsf{I2O}_1}$ scores for FF Arb PUFs with loops $l = 1, \ldots, 10$, averaged over different numbers of PUF-instances. The same, fixed challenge set of size $s = 10^5$ was used for computing each single $\mathsf{I2O}_1$-value.

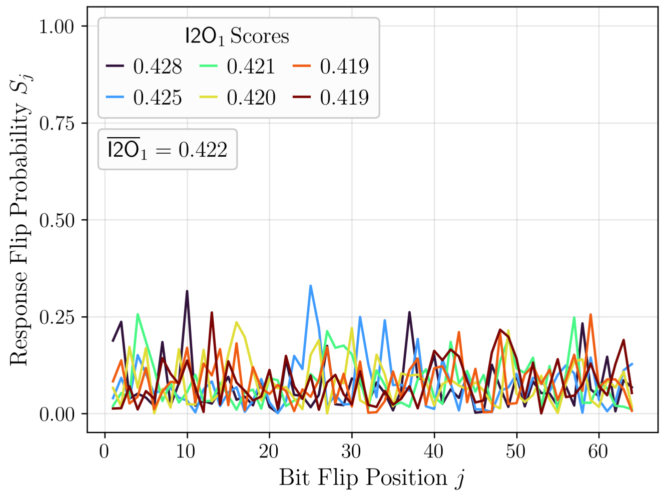

Fig. 11: Response flip probability S_j vs. bit flip position j for six Bistable Ring PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

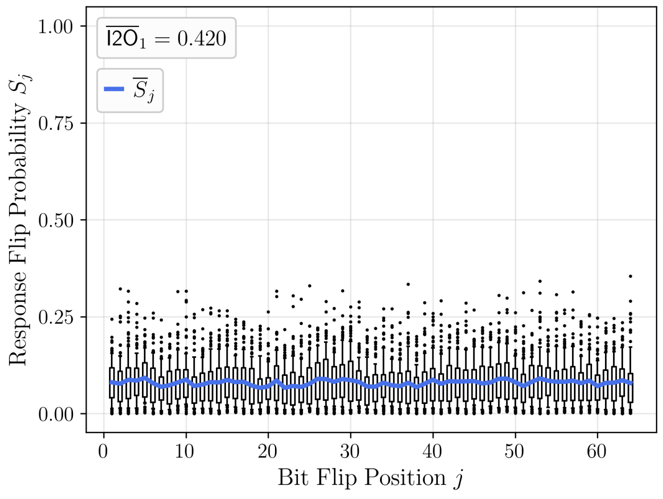

Fig. 13: Response flip probability S_j vs. bit flip position j for 100 Bistable Ring PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$. The box plot's whiskers range between the 10th and 90th percentile.

known ML-results that have evolved over a significant period of time in PUF-research (see, for example, [14], [15], [32]). Once more, this seems to illustrate the predictive quality and usefulness of our metrics.

In passing, let us state that many features of Figure 8 (and of most other presented data) invite further interpretation. For example, some of the feed-forward loops apparently lead to *"valleys"* in the S_j-curves of Figure 8, but others do not. While considering these aspects highly interesting, we must leave a detailed mathematical resolution to follow-up publications, simply for reasons of space. Asides, readers are encouraged to follow these promising routes in their own future works, too.

C. *Single Bit Flip Test on k-XOR Bistable Ring PUFs*

The Bistable Ring PUF [33] from Section II-C rests on a fundamentally different design paradigm than the Arbiter PUF family. Applying our metrics to this varying candidate as test case shall therefore complete the exemplary analyses led in this paper.

When considering the k-XOR BR PUFs architecture, the influence of every single challenge bit on the response intuitively should be smaller than for the k-XOR Arbiter PUF: Recall that merely a single inverter element in the entire bistable ring is altered when flipping a single challenge bit.

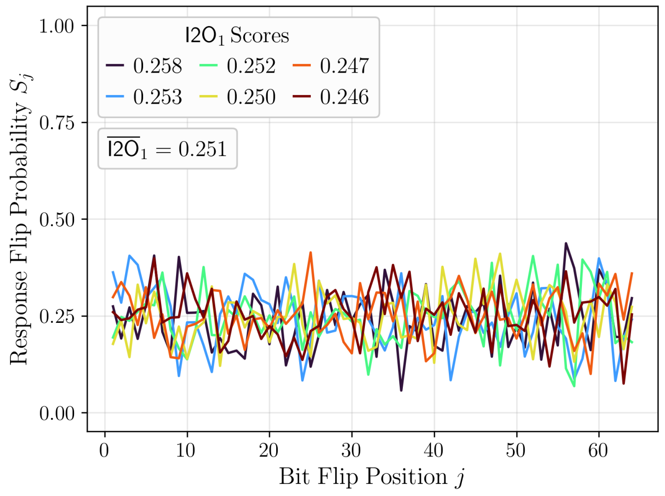

Fig. 12: Response flip probability S_j vs. bit flip position j for six 4-XOR Bistable Ring PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

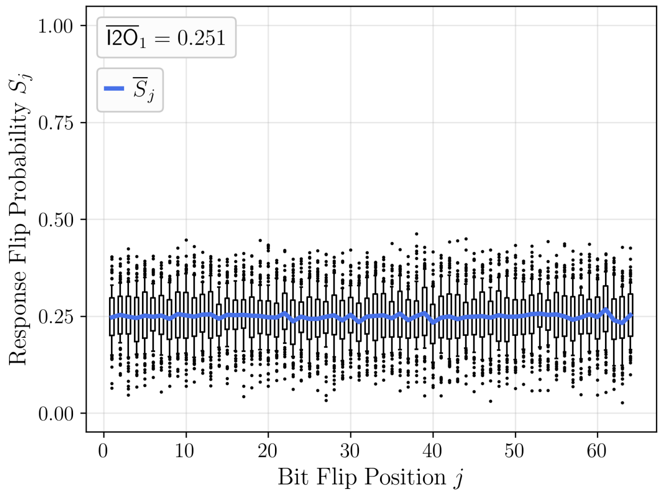

Fig. 14: Response flip probability S_j vs. bit flip position j for 100 4-XOR Bistable Ring PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$. The box plot's whiskers range between the 10th and 90th percentile.

No entire signal trains (and the possibly large delays they have accumulated) are rerouted by a single bit flip, as in a k-XOR Arbiter PUF. Furthermore, due to the rotational symmetry of the BR PUF, no systematic differences in the $\overline{S}_j$-values for different challenge bit positions j should be observable.

These first intuitions are confirmed by the formal analyses of this section. To start with, Figure 11 illustrates that no single bit flip probability for any challenge positions or any PUF-instances lies above 0.3 when considering six instances of plain BR PUFs; most S_j are, in fact, much smaller than that. For each PUF-instance, several S_j-scores are even equal to zero (but the associated challenge bit positions j vary from instance to instance). This means that the effective challenge space for each single instance (but not for the overall BR PUF) is systematically reduced: The instance-specific challenge bits with $S_j = 0$ have no influence on the responses. The above observations are backed up by the statistically significant analysis of 100 BR PUF instances provided in Figure 13.

When progressing to 4-XOR BR PUFs in Figures 12 and 14, we first observe that their S_j-scores have notably improved. No S_j is equal to zero anymore, neither for any of the six instances examined in Figure 12, nor for the 100 instances of Figure 14. Both figures show that the S_j start converging towards the ideal value of 0.5, while still remaining at a notable distance from it, though. Again, this corroborates the usefulness of the XOR gate as some universal *"healer"* for Strong PUF insufficiencies.

The foure Figures 11 to 14 also confirm our earlier hypothesis that there will be no *systematic* difference of the S_j- and $\overline{S}_j$-scores between varying challenge bit positions j in k-XOR BR PUFs. As already mentioned, this is due to the rotational symmetry of the ring-shaped architecture of the BR PUF, which designs from the Arbiter PUF family do not share. Overall, any of the three PUF design families examined in Sections IV-A to IV-C possess their own individual characteristics and well-recognizable metric scores. Again, this seems to point at the strength and versatility of our metrics.

Finally, a systematic analysis of $\overline{\mathsf{I2O}}_1$-scores of k-XOR BR PUFs for growing k and different numbers of PUF-instances is reported in Table III. The table again shows the effect of the XOR-gate as *"healer"* for the insufficiencies of the underlying plain BR PUF design. As in the similar analysis of Section IV-A, the numeric stability of the scores even for small numbers of considered instances, plus their discriminativity for large k, are worth noting.

V. Two Important Mathematical Observations

A. *Exact Relation between* $\overline{\mathsf{I2O}}_1$ *and* $\mathsf{A2O}_1$

Test 2 defined two average-related scores, namely $\overline{\mathsf{I2O}}_1$ and $\mathsf{A2O}_1$. Both were suggestive figures of merit that appeared natural to introduce. Readers will have noticed, though, that in all tests throughout this paper, we solely reported $\overline{\mathsf{I2O}}_1$-scores, never $\mathsf{A2O}_1$-scores. The underlying mathematical reasons are elaborated in the following analysis.

The analysis starts with an important, but relatively technical lemma, which precisely spells out the difference between $\overline{\mathsf{I2O}}_1$ and $\mathsf{A2O}_1$. The work invested here will later help proving our central theorem almost immediately.

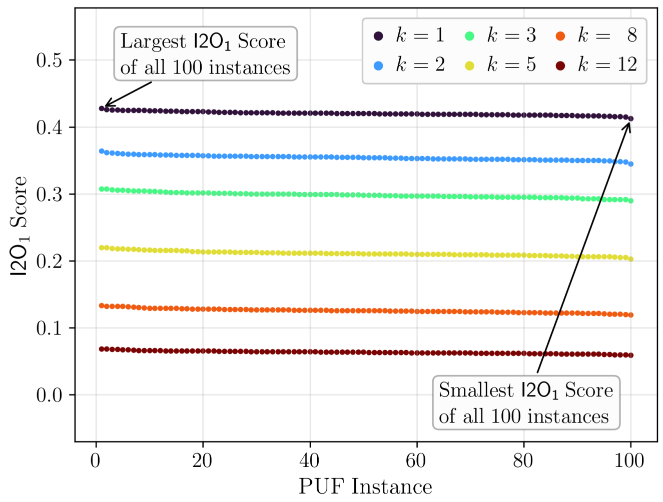

Fig. 15: Descending $\mathsf{I2O}_1$ Scores for 100 k-XOR Bistable Ring PUF instances with challenge length 64, using a fixed challenge set of size $s = 10^5$.

k-XOR Bistable Ring PUFs		$\overline{\mathsf{I2O}}_1$		
		Number of PUF-Instances		
		10	100	1000
k	1	0.42163	0.42025	0.42094
	2	0.35224	0.35423	0.35434
	3	0.29810	0.29824	0.29817
	4	0.25166	0.25122	0.25094
	5	0.21025	0.21100	0.21101
	6	0.17824	0.17809	0.17785
	7	0.15071	0.14968	0.14958
	8	0.12566	0.12551	0.12585
	9	0.10604	0.10589	0.10595
	10	0.08908	0.08944	0.08917
	11	0.07380	0.07495	0.07513
	12	0.06348	0.06333	0.06318
	13	0.05322	0.05349	0.05320
	14	0.04431	0.04461	0.04478
	15	0.03814	0.03757	0.03772
	16	0.03222	0.03181	0.03176
	17	0.02705	0.02668	0.02666
	18	0.02289	0.02248	0.02245
	19	0.01914	0.01880	0.01889
	20	0.01643	0.01597	0.01593

TABLE III: $\overline{\mathsf{I2O}}_1$ Scores for k-XOR Bistable Ring PUFs, averaged over different numbers of PUF-instances. The same, fixed challenge set of size $s = 10^5$ was used for computing each single $\mathsf{I2O}_1$-value.

Lemma 1. *Let* $\mathsf{P}_1, \ldots, \mathsf{P}_r$ *be* r *Strong PUF instances with challenge length* n *and single-bit responses. Now, for all* $j = 1, \ldots, n$*, we define:*

$$M_j^+ := \{i \in \{1, \ldots, r\} \mid S_j(\mathsf{P}_i) > 0.5\}$$

and

$$M_j^- := \{i \in \{1, \ldots, r\} \mid S_j(\mathsf{P}_i) < 0.5\}.$$

Furthermore, let $M_j^{\mathsf{Small}} := M_j^+$ *if*

$$\sum_{i \in M_j^+} (S_j(\mathsf{P}_i) - 0.5) \;\leq\; \left| \sum_{i \in M_j^-} (S_j(\mathsf{P}_i) - 0.5) \right|,$$

and $M_j^{\mathsf{Small}} := M_j^-$ *elsewise. Then it holds that:*

$$\overline{\mathsf{I2O}}_1 - \mathsf{A2O}_1 \;=\; \frac{2}{rn} \sum_{j=1}^{n} \sum_{i \in M_j^{\mathsf{Small}}} |S_j(\mathsf{P}_i) - 0.5|. \qquad (5)$$

Proof. Let us define the index set M_j^{Big} via

$$M_j^{\mathsf{Big}} \in \{M_j^+, M_j^-\} \setminus M_j^{\mathsf{Small}},$$

i.e., M_j^{Big} is the *"other"* element from $\{M_j^+, M_j^-\}$ that is *not* identical to M_j^{Small}. Then the following calculation proves the lemma:

$$\overline{\mathsf{I2O}}_1 - \mathsf{A2O}_1 =$$

$$\overset{(a)}{=} \frac{1}{r}\sum_{i=1}^{r}\left(\frac{1}{n}\sum_{j=1}^{n} |\, S_j(\mathsf{P}_i) - 0.5|\right) - \frac{1}{n}\sum_{j=1}^{n} |\, \overline{S}_j - 0.5\,|$$

$$\overset{(b)}{=} \frac{1}{r}\sum_{i=1}^{r}\left(\frac{1}{n}\sum_{i=1}^{n} |\, S_j(\mathsf{P}_i) - 0.5|\right) - \frac{1}{n}\sum_{j=1}^{n} \left| \left(\frac{1}{r}\sum_{i=1}^{r} S_j(P_i)\right) - 0.5 \right|$$

$$\overset{(c)}{=} \frac{1}{rn}\sum_{j=1}^{n}\left(\sum_{i=1}^{r} |S_j(\mathsf{P}_i) - 0.5| \;-\; \left|\sum_{i=1}^{r} S_j(\mathsf{P}_i) - 0.5\right|\right)$$

$$\overset{(d)}{=} \frac{1}{rn}\sum_{j=1}^{n}\left(\sum_{i \in M_j^{\mathsf{Big}}} |S_j(\mathsf{P}_k) - 0.5| \;+ \sum_{i \in M_j^{\mathsf{Small}}} |S_j(\mathsf{P}_k) - 0.5| - \sum_{i \in M_j^{\mathsf{Big}}} |S_j(\mathsf{P}_k) - 0.5| \;+ \sum_{i \in M_j^{\mathsf{Small}}} |S_j(\mathsf{P}_k) - 0.5|\right)$$

$$\overset{(e)}{=} \frac{2}{rn}\sum_{j=1}^{n} \sum_{i \in M_j^{\mathsf{Small}}} |S_j(\mathsf{P}_i) - 0.5.|$$

The single steps of the calculation are justified as follows: Step (a) follows from the definitions of $\overline{\mathsf{I2O}}_1$ and $\mathsf{A2O}_1$ in Test 2, and Step (b) is implied by the definition of $\overline{S}_j$ in the same test. Step (c) results from rearranging terms, in particular from putting 0.5 into the absolute value. Step (d) is the perhaps most critical step, eventually following from the definition of M_j^{Big} and M_j^{Small}. Note here that the union of M_j^{Big} and M_j^{Small} contains all indices i that lead to non-zero values for the expression $S_j(\mathsf{P}_i) - 0.5$. This explains the first, left part of the term rearrangement in this step. For the second, right part of the term transformation, please note that for all j,

$$\left|\sum_{i=1}^{r} S_j(\mathsf{P}_i) - 0.5\right| = \sum_{i \in M_j^{\mathsf{Big}}} |S_j(\mathsf{P}_k) - 0.5| \;-\; \sum_{i \in M_j^{\mathsf{Small}}} |S_j(\mathsf{P}_k) - 0.5|.$$

Step (e) is then again simple arithmetic. □

The above technical efforts will now pay off, and will strongly simplify the proof of the following theorem.

Theorem 1. *Let* $\mathsf{P}_1, \ldots, \mathsf{P}_r$ *be* r *Strong PUF instances with challenges of length* n *and single-bit responses. Then the following holds:*

T1 $\overline{\mathsf{I2O}}_1 \geq \mathsf{A2O}_1$.

T2 $\overline{\mathsf{I2O}}_1 \neq \mathsf{A2O}_1$ *if and only if there exists (at least) a challenge position* $j \in \{1, \ldots, n\}$*, and (at least) two PUF-instances* P_a *and* P_b *among the* $\mathsf{P}_1, \ldots, \mathsf{P}_r$*, such that* $S_j(\mathsf{P}_a) > 0.5$ *and* $S_j(\mathsf{P}_b) < 0.5$.

Proof. **T1** immediately follows from the fact that the right-hand side in Equation 5 is non-negative.

For **T2**, let us start with the forward direction: We assume that $\overline{\mathsf{I2O}}_1 \neq \mathsf{A2O}_1$. By Equation 5, this means that M_j^{Small} is a non-empty set. This, in turn, implies by the definition of M_j^{Small} that also M_j^{Big} is non-empty. Therefore also both M_j^+ and M_j^- are non-empty. Via the definition of M_j^+ and M_j^-, this means that there are PUF-instances P_a and P_b, such that $S_j(\mathsf{P}_a) > 0.5$ and $S_j(\mathsf{P}_b) < 0.5$, as desired.

For the reverse direction of **T2**, note that if there is such a challenge position j and instances P_a and P_b, then M_j^+ and M_j^- must both be non-empty sets. This means that also M_j^{Small}, which is either equal to M_j^+ or M_j^-, is non-empty. This implies that the right-hand side of Equation 5 is non-zero, and that $\overline{\mathsf{I2O}}_1 \neq \mathsf{A2O}_1$. □

Lemma 1 and Theorem 1 jointly enable us to address several important practical questions on the $\overline{\mathsf{I2O}}_1$- and $\mathsf{A2O}_1$-scores. To begin with, we can observe that for many examined PUF-sets of this paper, $\overline{\mathsf{I2O}}_1 = \mathsf{A2O}_1$. This holds, for example, for the six (or 100) BR PUF instances of Figures 11 (or 13), and the and six (or 100) 4-XOR BR PUF instances of Figures 12 (or 14). The reason is that none of the PUF-instances studied there possesses any S_j-scores larger than 0.5. By statement **T2** of Theorem 1, this implies that $\overline{\mathsf{I2O}}_1 = \mathsf{A2O}_1$. In all those cases of equality, using just one score (namely $\overline{\mathsf{I2O}}_1$) instead of two different scores is trivially justified, of course.

For any examined designs from the Arbiter PUF family, however, **T2** tells us that $\overline{\mathsf{I2O}}_1$ and $\mathsf{A2O}_1$ will not be equal: The reason is that for any of those designs, there exist $j \in \{1, \ldots, n\}$ such that the S_j-values of different PUF-instances lie on both sides of the 0.5-line. Illustrating examples are given in Figures 1 to 5, 7, 8 and 10.

In those cases where $\overline{\mathsf{I2O}}_1 \neq \mathsf{A2O}_1$, this leads to the pressing question which of the two scores will better capture the quality of a given set of PUFs. This may be answered best by an extreme, but illustrating example. Consider, for the sake of our argument, two hypothetical PUFs P_1 and P_2, where it holds for all challenge bit positions $j \in \{1, \ldots, n\}$ that

$$S_j(\mathsf{P}_1) := j \mod 2$$

and

$$S_j(\mathsf{P}_2) := j + 1 \mod 2.$$

For the two PUF-instances $\mathsf{P}_1, \mathsf{P}_2$, it then applies that $\mathsf{A2O}_1 = 0$ and $\overline{\mathsf{I2O}}_1 = 0.5$, as readers may confirm by a quick calculation following Test 2. I.e., the score $\mathsf{A2O}_1$ takes the minimal possible value, signalling good PUF-quality, while $\overline{\mathsf{I2O}}_1$ takes the maximal possible value, indicating strong PUF-vulnerabilities. Only one of these statements can be correct, of course. Indeed, it is not difficult to see that our hypothetical $\mathsf{P}_1, \mathsf{P}_2$ are trivially breakable: Their challenge bits either have no influence (whenever $S_j = 0$), or *always* flip the response (whenever $S_j = 1$). Therefore simple computation of the parity of all the influential challenge bits, plus knowledge of a single other CRP, will eventually allow prediction of all responses. The good $\mathsf{A2O}_1$-score hence is unjustified and misleading.

Analogous considerations also apply to less extreme examples: Whenever $\overline{\mathsf{I2O}}_1 \neq \mathsf{A2O}_1$, and whenever some S_j-scores of certain considered PUF-instances lie on both sides of the 0.5-line, some unwanted *"nullification"* will take place when computing $\mathsf{A2O}_1$: The deviations and non-idealities on *opposite* sides of the 0.5-line will partly cancel out instead of adding up. This puts the $\mathsf{A2O}_1$-score closer to the ideal value of 0 than it should be, unjustly signalling good PUF-quality. The computation of $\overline{\mathsf{I2O}}_1$ differs notably: Here, solely absolute terms of the form $|S_j - 0.5| \geq 0$ are summed up, preventing unwanted cancellation. This renders $\overline{\mathsf{I2O}}_1$ the preciser score whenever $\overline{\mathsf{I2O}}_1 \neq \mathsf{A2O}_1$.

Overall, this suggests using $\overline{\mathsf{I2O}}_1$ in our analyses, and completely dropping $\mathsf{A2O}_1$ from them, as we did in this paper.

B. Non-Linearities and Single Challenge Bit Flips

The second important observation of this section reveals an unexpected relation between the (non-)linearities of a given PUF and its S_j-scores. We start by repeating three (probably already known) definitions:

Definition 1. Let n be a positive integer. For all $j \in \{1, \ldots, n\}$, the *j-th unit vector* $e_j^n \in \{0,1\}^n$ is defined as

$$e_j^n := 0^{j-1} \,||\, 1 \,||\, 0^{n-j}.$$

If n is clear from the context, we may write e_j instead of e_j^n.

Definition 2. A Boolean function is a function that maps n-bit strings to single bits, for some positive integer n. For any fixed n, the set of all Boolean functions with n-bit inputs is denoted by $\mathbf{B}_1^n$.

Definition 3. A Boolean function $F \in \mathbf{B}_1^n$ is called *linear* if $F(x \oplus y) = F(x) \oplus F(y)$ for all $x, y \in \{0,1\}^n$. Furthermore, $F \in \mathbf{B}_1^n$ is called *affine* if $F(x \oplus y) = F(x) \oplus F(y) \oplus a$ for all $x, y \in \{0,1\}^n$ and for some (fixed) $a \in \{0,1\}$.

Linear and affine functions are strongly related: An affine function could be seen as a linear function plus some fixed *"transposition"*, which is expressed by the additive constant a. Due to this similarity, the terms linear/affine (and nonlinear/non-affine) are sometimes used laxly and interchangeably in the literature. Concerning cryptographic security, both affine and linear functions must be considered as equally insecure.

Definition 4. Let P be a PUF with challenge length n and single-bit responses. Then $F_{\mathsf{P}} : \{0,1\}^n \to \{0,1\}$, $F_{\mathsf{P}}(C_i) := R_i$ is called the *function associated with* P.

Having recapitulated the above definitions now allows us to formulate and prove our first main theorem.

Theorem 2. *Let P be a PUF with challenge length n and single-bit responses, and let F_{P} be its associated function. Then F_{P} is affine if and only if $S_j \in \{0,1\}$ for all $j \in \{1, \ldots, n\}$.*

Proof. Let us start with the *forward direction*: If F_{P} is affine, then for all PUF-challenges C_i and challenge positions $j \in \{1, \ldots, n\}$,

$$F_{\mathsf{P}}(C_i \oplus e_j) \;=\; F_{\mathsf{P}}(C_i) \oplus F_{\mathsf{P}}(e_j) \oplus a.$$

Now, we know that the term $F_{\mathsf{P}}(e_j) \oplus a$ obviously is independent of C_i, and that it is either equal to 0 or 1. If $F_{\mathsf{P}}(e_j) \oplus a = 0$, then flipping the challenge bit in position j (regardless of the challenge C_i) *never* has an impact on the response, and thus $S_j = 0$. If, on the other hand, $F_{\mathsf{P}}(e_j) \oplus a = 1$, then flipping the challenge bit in position j (regardless of C_i) *always* flips the response, whence $S_j = 1$. This proves the forward direction.

Let us next show the *inverse direction*, assuming that for all j, $S_j \in \{0,1\}$ for the considered PUF P. By the definition of S_j in Test 1, the fact $S_j \in \{0,1\}$ implies that for any challenges C_i and all challenge positions j:

$$F(C_i \oplus e_j) \;=\; F(C_i) \oplus S_j. \tag{6}$$

Furthermore, we know from basic algebra (or from obvious evidence) that any $y \in \{0,1\}^n$ can be written as

$$y = \bigoplus_{j \in I_y} e_j, \tag{7}$$

where $I_y \subseteq \{1, \ldots, n\}$ is a suitably chosen index set. This implies that for all $x, y \in \{0,1\}^n$:

$$
\begin{aligned}
F_{\mathsf{P}}(x \oplus y) &\overset{(a)}{=} F_{\mathsf{P}}\Big(x \bigoplus_{j\in I_y} e_j\Big) \\
&\overset{(b)}{=} F_{\mathsf{P}}(x) \bigoplus_{j\in I_y} S_j \\
&\overset{(c)}{=} F_{\mathsf{P}}(x) \bigoplus_{j\in I_y} S_j \;\oplus\; F_{\mathsf{P}}(0^n) \oplus F_{\mathsf{P}}(0^n) \\
&\overset{(d)}{=} F_{\mathsf{P}}(x) \oplus F_{\mathsf{P}}\Big(\bigoplus_{j\in I_y} e_j \oplus 0^n\Big) \oplus F_{\mathsf{P}}(0^n) \\
&\overset{(e)}{=} F_{\mathsf{P}}(x) \oplus F_{\mathsf{P}}(y) \oplus a.
\end{aligned}
$$

This proves that F_{P} is affine and completes the inverse direction of our proof. The single steps in the computation are justified as follows: Step (a) follows from Equation 7. Step (b) from repeated application of Equation 6. Step (c) is allowed since $F_{\mathsf{P}}(0^n) \oplus F_{\mathsf{P}}(0^n) = 0$. Step (d) is obtained by repeated *"backwards"* application of Equation 6. Step (e) follows from Equation 7 and from setting $a := F_{\mathsf{P}}(0^n)$. $\square$

Theorem 2 gently suggests the hypothesis that the closer the S_j are to the ideal value of 0.5, the more nonlinear/non-affine a PUF behaves in its j-th challenge bit position. If true, this would be a very important insight: It would directly relate the PUF's nonlinearities, which are essential for its ML-resilience [14], to its S_j-scores. The upcoming definitions, propositions, and theorems follow this trace. They fully spell out and prove the above hypothesis (and a bit more).

Definition 5. Let P be a PUF with challenge length n and single-bit responses, and let F_{P} be its associated function. Then for all $j \in \{1,\ldots,n\}$, the *"fraction of j-affine challenges of F_{P}"*, or $\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}})$ for short, is defined as

$$
\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) = 2^{-n} \cdot \max_{a_j\in\{0,1\}} \Big| \big\{ C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) = F_{\mathsf{P}}(C_i) \oplus F_{\mathsf{P}}(e_j) \oplus a_j \big\} \Big|. \tag{8}
$$

Likewise, the *"fraction of j-nonaffine challenges of F_{P}"*, or $\delta_j^{\mathsf{NA}}(F_{\mathsf{P}})$, is defined as

$$
\delta_j^{\mathsf{NA}}(F_{\mathsf{P}}) = 2^{-n} \cdot \min_{a_j\in\{0,1\}} \Big| \big\{ C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) \neq F_{\mathsf{P}}(C_i) \oplus F_{\mathsf{P}}(e_j) \oplus a_j \big\} \Big|. \tag{9}
$$

Put in everyday language, the value $\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}})$ tells us how *close* the behavior of F_{P} is to that of an affine function in its j-th challenge bit, i.e., with respect to the addition of e_j to its input challenge. It straightforwardly *"counts"* the fraction of challenges for which such an affine behavior occurs. At that, the additive constant a_j in the affine function is chosen from $\{0,1\}$ such that said fraction is maximized. Likewise, $\delta_j^{\mathsf{NA}}(F_{\mathsf{P}})$ tells us how *far apart* F_{P} is from such an affine behavior. The following proposition collects a first interesting fact:

Proposition 1. *Let P be a PUF with challenge length n and single-bit responses, F_{P} be its associated function, and $j \in \{1,\ldots,n\}$. Then it holds that*

$$
\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) + \delta_j^{\mathsf{NA}}(F_{\mathsf{P}}) = 1.
$$

Proof. The statement follows from the insight that the value $a_j \in \{0,1\}$ that *maximizes* the cardinality of the set on the right-hand side of Equation 8 is the very same value that *minimizes* the cardinality of the set on the right-hand side of Equation 9. This eventually implies that the union of the said two sets is equal to $\{0,1\}^n$. So, $\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) + \delta_j^{\mathsf{NA}}(F_{\mathsf{P}}) = 1$. $\square$

Now, we can formulate and prove our second main theorem:

Theorem 3. *Let P be a PUF with challenge length n and single-bit responses, and let F_{P} be its associated function. Then it holds for all $j \in \{1,\ldots,n\}$ that*

T3 $\quad \delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) = 0.5 + |S_j - 0.5|.$

T4 $\quad \delta_j^{\mathsf{NA}}(F_{\mathsf{P}}) = 0.5 - |S_j - 0.5|.$

Proof. Please note we need to prove only one of the two statements **T3** and **T4**; the other one follows via Proposition 1. We choose to prove **T3**, and start by showing that

$$
0.5 + |S_j - 0.5| = \max\{S_j, 1 - S_j\}. \tag{10}
$$

This can be accomplished by a standard case analysis: Assume (as case 1) that $S_j \geq 0.5$. Then $0.5 + |S_j - 0.5| = 0.5 + S_j - 0.5 = S_j = \max\{S_j, 1 - S_j\}$. An analogous analysis holds for the second case $S_j < 0.5$, proving Equation 10.

It remains to show that $\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) = \max\{S_j, 1 - S_j\}$. This can be seen as follows:

$$
\begin{aligned}
&\delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) = \\
&\overset{(a)}{=} 2^{-n} \cdot \max_{a_j\in\{0,1\}} \Big| \big\{ C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) = F_{\mathsf{P}}(C_i) \oplus F_{\mathsf{P}}(e_j) \oplus a_j \big\} \Big|. \\
&\overset{(b)}{=} 2^{-n} \cdot \max\Big\{ \Big| \big\{ C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) = F_{\mathsf{P}}(C_i) \oplus F_{\mathsf{P}}(e_j) \oplus 0 \big\} \Big|, \\
&\qquad \Big| \big\{ C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) = F_{\mathsf{P}}(C_i) \oplus F_{\mathsf{P}}(e_j) \oplus 1 \big\} \Big| \Big\} \\
&\overset{(c)}{=} \max\{S_j, 1 - S_j\}.
\end{aligned}
$$

The steps in the derivation can be justified as follows: Step (a) is due to the definition of δ_j^{Aff} (see Definition 5). Step (b) spells out the two possibilities $a_j = 0$ and $a_j = 1$ inside the max-operation. Step (c) holds as one of the two values $F_{\mathsf{P}}(e_j) \oplus 0$ and $F_{\mathsf{P}}(e_j) \oplus 1$ must be equal to one (for all challenges C_i, i.e., independently of C_i), while the other value is necessarily equal to zero. For the value that is equal to one, the resulting set $\{C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) = F_{\mathsf{P}}(C_i) \oplus 1\}$ has cardinality $2^n \cdot S_j$. For the value equal to zero, the resulting set $\{C_i \in \{0,1\}^n : F_{\mathsf{P}}(C_i \oplus e_j) = F_{\mathsf{P}}(C_i) \oplus 0\}$ has cardinality $2^n \cdot (1 - S_j)$. Multiplication with the value 2^{-n} from outside the max-operation yields the desired result. $\square$

The theorem has a simple corollary:

Corollary 1. *Let* P *be a PUF with challenge length* n *and single-bit responses, and let* F_{P} *be its associated function. Then it holds for all* $j \in \{1, \ldots, n\}$ *that*

$$0 \leq \delta_j^{\mathsf{NA}}(F_{\mathsf{P}}) \leq 0.5 \quad \textit{and} \quad 0.5 \leq \delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) \leq 1.$$

Proof. By Proposition 1, it suffices to show one of the two statements. The second statement that $0.5 \leq \delta_j^{\mathsf{Aff}}(F_{\mathsf{P}}) \leq 1$ follows from **T3** of Theorem 3 and $S_j \in [0,1]$. $\square$

Theorem 3 can be subsumed as follows: Among various other things, our bit flip scores S_j *also* implicitly assess the linearities and nonlinearities (or the affine and non-affine behavior) of F_{P} in the j-th position. This theoretically pinpoints one of several reasons for the surprisingly high predictive power of the S_j-scores for a PUF's ML-resilience, which we empirically demonstrated over previous sections. Please note that the *exact* computation of the nonlinearities of a PUF-under-test with medium or large challenge lengths is practically infeasible [37]. This necessitates some form of nonlinearity estimation or approximation. Our bit flip scores in passing provide such an estimation, with relatively little computational efforts and CRP-consumption.

VI. Summary and Conclusions

This paper belongs to a sequence of works that suggest computationally efficient, easy-to-apply, and simple-to-standardize security metrics for Strong PUF candidates. Their underlying motivation is that purely ML-based security studies on Strong PUFs *alone* and *in isolation* often have led to inconclusive or short-lived results in the past. All our metrics deal with Strong PUFs in a black-box fashion, i.e., they only require plain PUF-CRPs, and neither assume knowledge of an internal mathematical model of the PUF, nor any expertise in machine learning methods. Further, they require relatively little computational efforts and small numbers of PUF-CRPs in their application. All this makes them rather simple and effective to apply.

Our metrics can easily be used in initial Strong PUF security assessments or in the iterative design optimization of Strong PUF candidates. They can also serve as *"PUF canaries"* to recognize and winnow vulnerable Strong PUF architectures early on, avoiding periods of false security beliefs wherever possible. In either case, their target is *not* to completely replace, but to *complement* purely ML-based security studies on Strong PUFs. At that, each metric brings along its own view and perspective on a given design. The final assessment of an examined Strong PUF candidate should then take all available information into account, including *both* ML-studies *and* all available metric scores.

This manuscript investigated a suggestive, but very impactful metric: Namely the effect that flipping single bits in PUF-challenges has on the corresponding PUF-responses. We defined various concrete metric scores associated with this idea. They included our S_j- and $\mathsf{I2O}_1$-values, which can be applied to single PUF-instances, and the $\overline{S}_j$-, $\overline{\mathsf{I2O}}_1$-, and $\mathsf{A2O}_1$-scores, which constitute average-based metrics for a population of more than one PUF-instance. We mathematically proved that $\overline{\mathsf{I2O}}_1$ is a similar, but preferable metric over $\mathsf{A2O}_1$. For this reason, all analyses of this paper were eventually carried out using $\overline{\mathsf{I2O}}_1$, not $\mathsf{A2O}_1$. This key result is presented in V-A of the Supplementary Material on page 11.

In various exemplary studies, we then applied our new metrics to three popular and well-investigated Strong PUF families as test cases, namely k-XOR Arbiter PUFs, Feed-Forward Arbiter PUFs with l loops, and k-XOR Bistable Ring PUFs. For all these architectures, we obtained strongly differing, individual, and characteristic scores, indicating our metrics' versatility and expressiveness. Our analyses demonstrated that the metrics can systematically unearth non-optimal behaviour and potential security weaknesses: The obtained scores were in essentially *perfect agreement* with the long-known real-world security of the examined PUFs against ML-attacks. One *of many* reasons for this close agreement is that the S_j scores are directly related to the nonlinearities of a given Strong PUF, adding to their security relevance. A in-depth mathematical proof of this central fact is given in Section V-B of the Supplementary Material on page 13.

We stress again that our tests do *not* constitute a trivial automatism in the sense that certain metric scores would either unconditionally prove or disprove security. Instead, the results need to be interpreted carefully on an individual basis, and simplistic conclusions must be avoided. Among others, an *inconsiderate* comparison of scores across different challenge lengths, or among different design families, is not recommended. It is also not too difficult to come up with (artificial) mathematical functions that lead to high test scores, but which would be insecure in practice as PUFs, and vice versa. To name one example, Strong PUF candidates with additional input functions [38] may lead to unreasonably high metric scores. In such cases, both the PUF-scores *with* and *without* the input function should be computed and assessed: From a security viewpoint, they represent the realistic practical scenarios in which an attacker can (or cannot) physically circumvent said input function. Similar considerations apply to various other, possibly constructed counterexamples. Our metrics need to be handled and interpreted with care.

Interestingly, the findings of this paper also illustrate that if our metrics had been known at the time of the introduction of certain Strong PUF candidates, such as the 2-XOR Arbiter PUF [29], Feed-Forward Arbiter PUF [16], or Bistable Ring PUF [33], they could have signalled the vulnerabilities of these designs early on. As a consequence, these architecture probably would *not* have been assumed secure for several years before being eventually broken in 2010 [14] or 2015 [34]. This again corroborates the functionality of our metrics as early *"PUF canaries"*. It also once more illustrates the potential shortcomings of purely ML-based security analyses, which merely apply the ML-algorithms most popular at a time to a given Strong PUF candidate in the attempt to conclusively assess its long-term vulnerabilities.

Finally, we have made all code for the computation and graphical representation of our metrics publicly available to foster simple and widespread application of our methods [28].

Future Work: Various research tasks arise from the presented content. Firstly, it appears interesting to investigate the effect of flipping z bits in a PUF-challenge (for $z \geq 2$). This can be expected to reveal yet other, *"higher-dimensional"* weaknesses and imperfections in a PUF-under-test. It will also require new techniques for the graphical representation of the results. Another promising path is the application of our metrics to other long-standing Strong PUF candidates, verifying a natural relation between their scores and their well-known real-world security. Thirdly, PUF-designers could use our metrics to improve the security of their new Strong PUF architectures in iterative design optimization. This might become standard in future design cycles, and could prove one of the most fruitful and important applications of our techniques. Finally, optimization of the scores and analyses of this paper, and introduction of entirely new metrics beyond challenge bit flips, seems a natural expansion of our activities.

Acknowledgements

Y.W. acknowledges funding by the Stanford Graduate Fellowship. D.M. thanks DST (India), C3i IIT Kanpur, and C-HERD, Meity (India) for partially funding this work. U.R. acknowledges support by the EU-project NEUROPULS.

References

[1] R. Pappu *et al.*, "Physical one-way functions," *Science*, vol. 297, no. 5589, 2002.

[2] B. Gassend *et al.*, "Silicon physical random functions," in *CCS*, 2002.

[3] J. Guajardo *et al.*, "Fpga intrinsic pufs and their use for ip protection," in *CHES*, 2007.

[4] B. Gassend *et al.*, "Controlled physical random functions," in *CSAC*, 2002.

[5] K. Kursawe *et al.*, "Reconfigurable physical unclonable functions-enabling technology for tamper-resistant storage," in *HOST*, 2009.

[6] U. Rührmair, "Simpl systems: On a public key variant of physical unclonable functions," *Cryptology ePrint Archive*, 2009.

[7] N. Beckmann *et al.*, "Hardware-based public-key cryptography with public physically unclonable functions," in *IH*, 2009.

[8] U. Rührmair, "Secret-free security: A survey and tutorial," in *Journal of Cryptographic Engineering*, 2022.

[9] U. Rührmair *et al.*, "Pufs at a glance," in *DATE*, 2014.

[10] C. Herder *et al.*, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, 2014.

[11] B. L. P. Gassend, "Physical random functions," *MSc Thesis, Massachusetts Institute of Technology (MIT)*, 2003.

[12] U. Rührmair, "Oblivious transfer based on physical unclonable functions," in *TRUST*, 2010.

[13] C. Brzuska *et al.*, "Physically uncloneable functions in the universal composition framework," in *CRYPTO*, 2011.

[14] U. Rührmair *et al.*, "Modeling attacks on physical unclonable functions," in *CCS*, 2010.

[15] D. Lim, "Extracting secret keys from integrated circuits," Master Thesis, Massachusetts Institute of Technology, 2004. [Online]. Available: https://dspace.mit.edu/handle/1721.1/18059

[16] B. Gassend *et al.*, "Identification and authentication of integrated circuits," *CCPE*, vol. 16, no. 11, 2004.

[17] R. Kumar *et al.*, "On design of a highly secure puf based on non-linear current mirrors," in *HOST*, 2014.

[18] A. Vijayakumar *et al.*, "A novel modeling attack resistant puf design based on non-linear voltage transfer characteristics," in *DATE*, 2015.

[19] Q. Guo *et al.*, "Efficient attack on non-linear current mirror puf with genetic algorithm," in *ATS*, 2016.

[20] J. Ye *et al.*, "Poster: Attack on non-linear physical unclonable function," in *CCS*, 2016.

[21] M. Majzoobi *et al.*, "Testing techniques for hardware security," in *ITC*, 2008.

[22] P. H. Nguyen *et al.*, "Security analysis of arbiter PUF and its lightweight compositions under predictability test," *TODAES*, vol. 22, no. 2, 2017.

[23] F. Ganji, "On the learnability of physically unclonable functions," Ph.D. dissertation, Technical University of Berlin, Germany, 2017.

[24] D. Chatterjee *et al.*, "PUF-G: A CAD framework for automated assessment of provable learnability from formal PUF representations," in *ICCAD*, 2020.

[25] F. Ganji *et al.*, "Pufmeter a property testing tool for assessing the robustness of physically unclonable functions to machine learning attacks," *IEEE Access*, vol. 7, 2019.

[26] F. Kappelhoff *et al.*, "Strong puf security metrics: Response sensitivity to small challenge perturbations," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022.

[27] N. Wisiol *et al.*, "pypuf: Cryptanalysis of physically unclonable functions," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.3901410

[28] W. Stefani *et al.*, "Puf-security-metrics-single-bit-flips: Initial code release (v1.0.0)," 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8364104

[29] G. E. Suh *et al.*, "Physical unclonable functions for device authentication and secret key generation," in *DAC*, 2007.

[30] U. Rührmair *et al.*, "Power and timing side channels for pufs and their efficient exploitation," *Cryptology ePrint Archive*, 2013.

[31] G. T. Becker, "The gap between promise and reality: On the insecurity of xor arbiter pufs," in *CHES*, 2015.

[32] U. Rührmair *et al.*, "Puf modeling attacks on simulated and silicon data," *TIFS*, vol. 8, no. 11, 2013.

[33] Q. Chen *et al.*, "The bistable ring puf: A new architecture for strong physical unclonable functions," in *HOST*, 2011.

[34] X. Xu *et al.*, "Security evaluation and enhancement of bistable ring pufs," in *RFIDSec*, 2015.

[35] F. Ganji *et al.*, "Strong machine learning attack against pufs with no mathematical model," in *CHES*, 2016.

[36] M. Majzoobi *et al.*, "Lightweight secure pufs," in *IEEE/ACM ICCAD*, 2008.

[37] D. Chatterjee *et al.*, "Systematically quantifying cryptanalytic non-linearities in strong pufs," *Cryptology ePrint Archive*, 2022.

[38] M. Majzoobi *et al.*, "Lightweight secure pufs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, 2008.