APUF Production Line Faults: Uniqueness and Testing

Yeqi Wei, Wenjing Rao, Natasha Devroye
Department of Electrical and Computer Engineering
University of Illinois Chicago, Chicago, IL 60607, USA
Email: {ywei30, wenjing, devroye}@uic.edu

***Abstract*—Arbiter Physically Unclonable Functions (APUFs) are low-cost hardware security primitives that may serve as unique digital fingerprints for ICs. To fulfill this role, it is critical for manufacturers to ensure that a batch of PUFs coming off the same design and production line have different truth tables, and uniqueness / inter-PUF-distance metrics have been defined to measure this. This paper points out that a widely-used uniqueness metric fails to capture some special cases, which we remedy by proposing a modified uniqueness metric. We then look at two fundamental APUF-native production line fault models that severely affect uniqueness: the μ (abnormal mean of a delay difference element) and σ (abnormal variance of a delay difference element) faults. We propose test and diagnosis methods aimed at these two APUF production line faults, and show that these low-cost techniques can efficiently and effectively detect such faults, and pinpoint the element of abnormality, without the (costly) need to directly measure the uniqueness metric of a PUF batch.**

***Index Terms*—arbiter PUF, arbiter PUF faults, testing, diagnosis**

I. Introduction

PUFs (Physically Unclonable Functions) are promising low-cost hardware security primitives that exploit manufacturing randomness to generate unique digital fingerprints for device authentication [1]–[3]. In an APUF (Arbiter based PUF), a series of track pairs are designed to be of equal delay, but due to manufacturing randomness, each pair of tracks differ slightly in their delay values. A binary input, or "challenge" $\mathbf{c} \in \{0,1\}^n$, decides how two racing paths (consisting of consecutive, disjoint delay tracks) are formed, and fed into an arbiter. The output is a binary "response" $R(\mathbf{c}) \in \{\pm 1\}$ that depends on which racing path arrives first. For each manufactured APUF instance, it results in a hopefully unique truth table that consists of 2^n *challenge-response pairs* (CRPs), $\big(\mathbf{c}, R(\mathbf{c})\big)$. APUF is a "strong" PUF that offers CRPs exponential in the number of delay elements, served as a basic building block for more complex strong PUFs.

Contributions "Uniqueness" is a widely used metric for a batch of PUFs, defined as in e.g. [4, Eq (4)] (see Definition 7 later) to evaluate how similar their truth-tables are [5]–[9]. However, as we will see, it fails to capture some intuitive notions of uniqueness. This motivates us to: 1) define a variation of the uniqueness metric which rectifies this. We then 2) evaluate the impact of two APUF-specific manufacturing faults, the μ-fault, and the σ-fault, introduced in [9], on this uniqueness. In [9] the μ-fault was shown to have a great impact on the (old) uniqueness, while the σ-fault, surprisingly, did not. In this paper, we point out that a σ-fault *does* have an impact on the newly defined uniqueness. 3) Finally, we propose a test flow for testing and diagnosing batches of PUFs for these two production line faults that affect the (new) uniqueness. The tests are low-cost and efficient: only a few challenge bits are required for high true positive rates.

* This work was partially supported by NSF under awards 1909547 and 2244479. The contents of this article are solely the responsibility of the authors and do not necessarily represent the official views of the NSF.

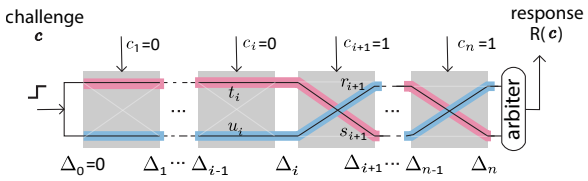

Fig. 1: [9] APUF with challenge bits selecting the parallel (t_i, u_i) or crossed (r_i, s_i) tracks to form two racing paths. The response is the sign of the accumulated delay difference $\Delta_n(\mathbf{c})$.

The proposed tests differ markedly from the standard and straightforward approach which directly measures the uniqueness metric of a PUF batch. Such measurement requires significantly more computation and does not provide insights into fault types and locations. Our test flow starts with a desired uniqueness level, combined with offline simulations will allow us to carefully select thresholds for the proposed algorithms. The test algorithms will decide efficiently whether an APUF batch is good, with a μ- or σ-fault. Then, our diagnosis algorithms will pinpoint the fault location accurately, for either μ- or σ- faults.

II. Preliminaries

A. Arbiter PUF

The architecture of the APUF is illustrated in Fig. 1. The input of an APUF is a binary vector $\mathbf{c} \in \{0,1\}^n$, called a challenge. Two racing signals (red and blue) traverse an n-stage path. A race resolution arbiter compares which signal arrives first and produces the output, a response $R(\mathbf{c}) \in \{\pm 1\}$. The challenge $\mathbf{c}$ determines which of 2^n paths is picked: the signals traverse via the "parallel" tracks of delays (t_i, u_i) if $c_i = 0$, or the "crossed" tracks (r_i, s_i) if $c_i = 1$. The response is $+1$ (-1) if the upper (lower) entrance to the arbiter arrives first.

Since the response relies only on which signal arrives first, it depends on the *delay difference* at each stage i, denoted as *delta elements*, $\delta_i^{(0)} := t_i - u_i$ (selected if $c_i = 0$) and $\delta_i^{(1)} := r_i - s_i$

978-3-9819263-7-8/DATE23/© 2023 EDAA

c_1	c_2	c_3	c_4	$\Delta_n(\mathbf{c})$	
0	0	0	1	$\delta_1^{(0)} - \delta_2^{(0)} - \delta_3^{(0)} + \delta_4^{(1)}$	$\rightarrow \mathcal{C}_{2,-}^{(0)}$
0	0	1	0	$\delta_1^{(0)} - \delta_2^{(0)} + \delta_3^{(1)} + \delta_4^{(0)}$	
0	1	1	1	$\delta_1^{(0)} + \delta_2^{(1)} - \delta_3^{(1)} + \delta_4^{(1)}$	$\rightarrow \mathcal{C}_{2,+}^{(1)}$

Fig. 2: Target set example for a 4-stage APUF.

(selected if $c_i = 1$), or $\delta_i^{(c_i)}$ for short. The response $R(\mathbf{c})$ then can be represented as the sign of the accumulated delay difference at the final stage, $\Delta_n(\mathbf{c})$, as

$$R(\mathbf{c}) = \text{sign}\big(\Delta_n(\mathbf{c})\big) \in \{\pm 1\}, \tag{1}$$

where $\Delta_n(\mathbf{c})$ is computed recursively for $i \in [1, n]$, $\Delta_0 = 0$:

$$\Delta_i(\mathbf{c}) = \begin{cases} +\Delta_{i-1}(\mathbf{c}) + \delta_i^{(0)}, & \text{when } c_i = 0 \\ -\Delta_{i-1}(\mathbf{c}) + \delta_i^{(1)}, & \text{when } c_i = 1 \end{cases} \tag{2}$$

and $\Delta_i(\mathbf{c})$ is the accumulated delay difference after stage i. By expanding (2), $\Delta_n(\mathbf{c})$ can also be represented as the summation of δs with different signs: $\Delta_n(\mathbf{c}) = \sum_{i=1}^{n-1} a_i(\mathbf{c})\delta_i^{(c_i)} + \delta_n^{(c_n)}$, where $a_i(\mathbf{c}) = (-1)^{\sum_{i+1}^{n} c_i} \in \{\pm 1\}$ is the sign of $\delta_i^{(c_i)}$ in $\Delta_n(\mathbf{c})$.

B. Target set

We adopt the definition of target set from [9] as follows:

***Definition 1 (*target set):** A target set $\mathcal{C}_{i,+}^{(x)}$ (or $\mathcal{C}_{i,-}^{(x)}$) with $x \in \{0,1\}, i \in [1, n]$ contains all n-bit challenges preserving (or reversing) the sign of $\delta_i^{(x)}$, which may be derived from (2):

$$\begin{aligned} \mathcal{C}_{i,+}^{(x)} &:= \{\text{challenges with } +\delta_i^{(x)} \text{selected in } \Delta_n\} \\ &= \big\{\mathbf{c} \in \{0,1\}^n : c_i = x, c_{i+1} + \cdots + c_n \text{ is even}\big\}, \\ \mathcal{C}_{i,-}^{(x)} &:= \{\text{challenges with } -\delta_i^{(x)} \text{selected in } \Delta_n\} \\ &= \big\{\mathbf{c} \in \{0,1\}^n : c_i = x, c_{i+1} + \cdots + c_n \text{ is odd}\big\}. \end{aligned}$$

Fig. 2 shows an example: concerning $\delta_2^{(0)}$ and $\delta_2^{(1)}$, the first two challenges belong to the target set $\mathcal{C}_{2,-}^{(0)}$, since $\delta_2^{(0)}$ has a negative sign in $\Delta_n(\mathbf{c})$. The last challenge is in $\mathcal{C}_{2,+}^{(1)}$, since $\delta_2^{(1)}$ has positive sign in $\Delta_n(\mathbf{c})$.

C. Response matrix

Next, as a tool for visualizing faults in a batch of APUFs, we propose the response matrix, defined as:

***Definition 2 (*response matrix):** The response matrix for a set of APUF instances, $\mathcal{A}$ of size M, and a set of challenges, $\mathcal{C}$ of size N, is defined as the $M \times N$ matrix:

$$\mathbf{R}(\mathcal{A}, \mathcal{C}) := \begin{bmatrix} R_1(\mathbf{c}_1) & R_1(\mathbf{c}_2) & \dots & R_1(\mathbf{c}_N) \\ R_2(\mathbf{c}_1) & R_2(\mathbf{c}_2) & \dots & R_2(\mathbf{c}_N) \\ \vdots & \vdots & \ddots & \vdots \\ R_M(\mathbf{c}_1) & R_M(\mathbf{c}_2) & \dots & R_M(\mathbf{c}_N) \end{bmatrix}_{M \times N}$$
$$= \Big[\mathbf{R}\Big(\mathcal{A}, \{\mathbf{c}_1\}\Big)\ \mathbf{R}\Big(\mathcal{A}, \{\mathbf{c}_2\}\Big)\ \dots\ \mathbf{R}\Big(\mathcal{A}, \{\mathbf{c}_N\}\Big)\Big],$$

where $R_i(\mathbf{c}_j)$ is the response of APUF A_i to challenge $\mathbf{c}_j$.

D. Production line faults

We first present two definitions of faulty production lines from [9], as the most relevant APUF-native faults that will affect APUF qualities.

In a good production line, all δ elements are distributed as $\mathcal{N}(0, \sigma^2)$; a μ-fault production line fault corresponds to a fault in the mean of a δ element (non-zero mean), while a σ-fault corresponds to a fault on a manufactured δ element's variance (variance larger than the targeted σ^2). These faults result from unbalanced design from EDA tools, or problematic process variation in the manufacturing process, leading to abnormal individual δ-elements which are key components of the APUF and the random elements that yield the unique PUF properties.

We consider a single δ element fault as the basic scenario to establish the analysis and test / diagnosis methodologies.

Definition 3: (**ideal assumption**) An APUF batch is defined as good if every δ element is generated with standard normal distribution, i.e.

$$\delta_i^{(x)} \sim \mathcal{N}(0, \sigma^2), \quad \forall i \in \{1, \cdots, n\},\ x \in \{0, 1\}.$$

Definition 4: (μ**-fault**) An APUF batch suffers from a μ-fault if there exists a δ element with non-zero mean distribution, i.e. for some $K \neq 0$,

$$\exists i \in \{1, \cdots, n\}, x \in \{0, 1\} :\ \delta_i^{(x)} \sim \mathcal{N}(K\sigma, \sigma^2).$$

Definition 5: (σ**-fault**) An APUF batch suffers from a σ-fault if there exists a δ element with a large variance, i.e. for some $L > 1$,

$$\exists i \in \{1, \cdots, n\}, x \in \{0, 1\} :\ \delta_i^{(x)} \sim \mathcal{N}\left(0, (L\sigma)^2\right).$$

Definition 6: (**fault intensities**) In Definition 4 and 5, the real numbers K and L, are called *fault intensities,* with larger values denoting larger deviation away from the ideal assumption.

III. The Problematic Uniqueness Metric

We now consider one of the most important PUF metrics, uniqueness, which is essential for PUFs as hardware primitives for authentication purposes. Intuitively, a PUF batch is ideal in uniqueness if, when we pick any 2 PUFs, their truth tables differ in half. Consider the example shown on the left of Table I: the uniqueness of batch A (PUFs 1,2,3,4) is ideal, i.e. the responses of every two PUFs differ in half.

Researchers have tried to quantify this intuitive notion in different ways [4], [10]–[12]. One of these is as a re-scaling / normalization of the "inter-PUF distance" or expected Hamming distance (HD) between responses (ideally 0.5), defined as P_{inter} in [4, Equation (3)]. Note that the true uniqueness profile of a PUF should be captured by the distribution of the HD over all PUF pairs (the randomness) over all challenges. How to fully characterize it for subsets of challenges and under non-ideal conditions is challenging. Hence, the simpler single-number metrics or statistics such as the most widely-used uniqueness metric of [4, Equation (4)] is adopted, defined in 7 below.

Definition 7: (**original uniqueness**) The original uniqueness, $U_o(\mathcal{A}, \mathcal{C})$ of a given PUF set $\mathcal{A}$ over a challenge set $\mathcal{C}$, is defined

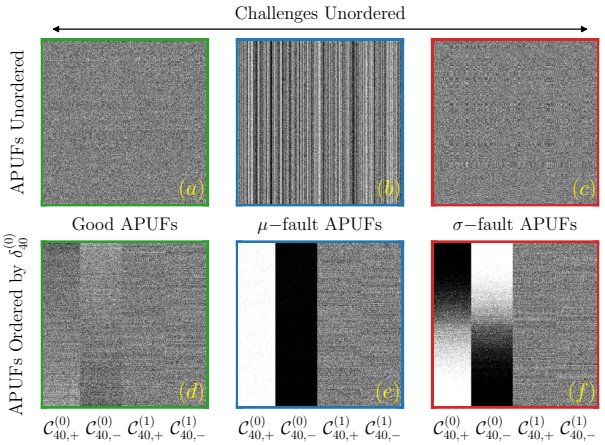

Fig. 3: Response matrices (each is 1000 64-stage APUFs $\times$1000 challenges) of three batches: $\delta_i^{(x)} \sim \mathcal{N}(0,1)$ (left), μ-fault with $\delta_{40}^{(0)} \sim \mathcal{N}(20,1)$ (middle), and σ-fault with $\delta_{40}^{(0)} \sim \mathcal{N}(0,400)$ (right). Bottom plots reordered with APUF rows according to the values of the abnormal $\delta_{40}^{(0)}$, challenge columns according to the target sets of the abnormal $\delta_{40}^{(0)}$.

as the expectation of the *single pair Hamming distance (HD)* between two PUF instances in $\mathcal{A}$ as:

$$U_o(\mathcal{A},\mathcal{C}) := 1 - |2P_{inter} - 1|, \tag{3}$$

where the "inter-PUF distance" is the expected normalized HD of all the PUF pairs:

$$P_{inter} := E\left(\frac{2}{(M-1)M}\sum_{i=1}^{M-1}\sum_{j=i+1}^{M}\frac{HD_{ij}(\mathcal{C})}{N}\right),$$

M is the number of PUFs in the PUF set $\mathcal{A}$, N is the number of challenges in the challenge set $\mathcal{C}$, $HD_{ij}(\mathcal{C})$ denotes the HD between the responses of PUF i and j to all challenges in $\mathcal{C}$, and E is the expectation.

This definition has two problems. First, consider the example of batch B shown in Table I: PUF 6, 7, 8 have identical responses, while PUF 5 has complementary ones, which intuitively is far from ideal in uniqueness. However, according to the above definition U_o, batch B achieves ideal uniqueness, as the complementary pairs "cancel" out the identical pairs to yield a final average of perfect HD (0.5). Secondly, according to this definition, [9] showed that μ-fault production lines affect P_{inter} (and hence its re-scaling, the uniqueness), while σ-fault production lines, surprisingly, do not affect it (same for the uniqueness).

batch A	$R(\mathbf{c}_1)$	$R(\mathbf{c}_2)$	$R(\mathbf{c}_3)$	$R(\mathbf{c}_4)$	batch B	$R(\mathbf{c}_1)$	$R(\mathbf{c}_2)$	$R(\mathbf{c}_3)$	$R(\mathbf{c}_4)$
PUF 1	+1	+1	+1	+1	PUF 5	+1	-1	-1	+1
PUF 2	+1	-1	+1	-1	PUF 6	-1	+1	+1	-1
PUF 3	+1	+1	-1	-1	PUF 7	-1	+1	+1	-1
PUF 4	+1	-1	-1	+1	PUF 8	-1	+1	+1	-1

TABLE I: Example of 2 PUF batches: batch A (1,2,3,4) presents ideal uniqueness: i.e., any pair's HD differ in half; batch B (5,6,7,8) obviously is not ideal in uniqueness, as three (6, 7, 8) are identical. The widely-used metric U_o in Def. 7 fails to discern this and treats both A and B as ideal in uniqueness.

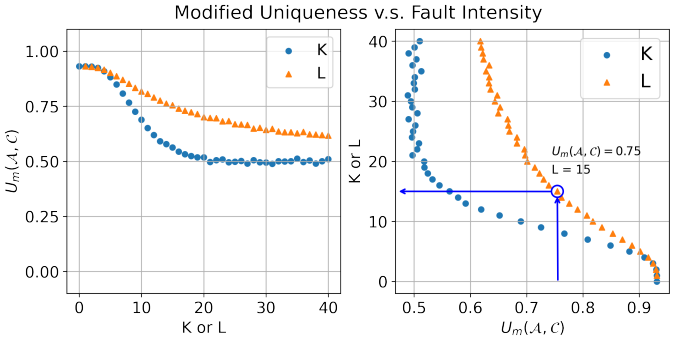

Fig. 4: Impact of μ- and σ- faults on uniqueness via U_m.

However, it is clear that both μ- and σ- faults affect the uniqueness of a batch of APUFs. To see this, we now plot the response matrices of μ- and σ- faults in Fig. 3. At first, the σ-fault's response matrix on the top right looks similar to the good batch on the top left. However, after reordering the matrices according to the target sets that pick the abnormal $\delta_{40}^{(0)}$ element, and then further sorting them according to the values of the abnormal $\delta_{40}^{(0)}$, we observe that many of the APUFs generated from the σ-fault production line have very similar responses to the challenges that pick the abnormal $\delta_{40}^{(0)}$. These APUFs are not unique, and this shows that U_o, defined by P_{inter} and equation (3) fails to capture the true impact of a σ-fault on uniqueness.

The other example provided in batch B of Table I may easily be shown to also yield a $P_{inter} = 0.5$, and hence $U_o(\mathcal{A},C) = 1$, both of which are ideal. Again, in batch B of Table I, three of the PUFs have identical truth tables, and the fourth is simply the opposite. Clearly, the uniqueness metric is not properly defined.

IV. A Modified Uniqueness Metric

In this section, we propose a "modified" uniqueness that captures the impact of both μ and σ production faults and resolves the issue seen in batch B of Table I, where the normalized HD of every PUF pair "cancel" out when averaged: half yield a 0 and half a 1. To avoid this, we propose to use the absolute value of the deviation of the normalized HD from the ideal 0.5, as:

***Definition** 8:* (**modified uniqueness**) The modified uniqueness, $U_m(\mathcal{A},\mathcal{C})$ of given PUF set $\mathcal{A}$ consists of M PUFs and challenge set $\mathcal{C}$ consists of N challenges is defined as

$$U_m(\mathcal{A},\mathcal{C}) := 1 - 2Dev,$$

where *the normalized deviation* from the ideal inter-PUF Hamming distance (0.5) is defined as

$$Dev := \frac{2}{M(M-1)}\sum_{i=1}^{M-1}\sum_{j=i+1}^{M}\left|\frac{HD_{ij}(\mathcal{C})}{N} - 0.5\right|.$$

Remark: Under this definition, the uniqueness lies in the range $[0,1]$, with 0 being the worst and 1 the ideal value. This new uniqueness metric solves the issue seen in the old definition's uniqueness of Table I.

Fig. 4 shows (on the left sub-plot) how the modified uniqueness $U_m(\mathcal{A},\mathcal{C})$ from Definition 8 captures the impact of both

μ- and σ-fault production lines on uniqueness: indeed, both μ- and σ- faults affect uniqueness (contrary to the previous understanding according to the old uniqueness metric $U_o(\mathcal{A}, \mathcal{C})$, where σ-fault does not affect uniqueness). Quantitatively, as the fault intensity (K, L) increases, the uniqueness worsens, with the μ-fault having a more severe impact than the σ-fault. We also see that, under a random challenge set, when the fault intensity (measured by K, L values) is greater than around 5, both μ-faults and σ-faults start to have a marked impact on $U_m(\mathcal{A}, \mathcal{C})$, and the worst uniqueness tends to 0.5.

The sub-plot on the right of Fig. 4 shows how each manufacturer can identify the acceptable fault intensities (K, L) according to their desired acceptable level of uniqueness. As such, we can define a good PUF batch as one where all PUFs that either 1) satisfy Definition 3, *or* 2) it suffers from a μ-fault or σ-fault with small enough fault intensities to guarantee a uniqueness above a desired value, say larger than U_d, obtained from Fig. 4. It is important to "catch" production line faults whose K, L give undesirable uniqueness values, discussed next.

V. Test and Diagnosis Methodologies

As seen in the previous sections, both μ- and σ- faults affect the uniqueness of an APUF batch. To overcome the challenge that such faults are analogue and statistical in nature, we propose a set of efficient methodologies that can precisely test and diagnose them.

Note that one could measure uniqueness directly (using $U_m(\mathcal{A}, \mathcal{C})$ for example), and judge whether a PUF batch meets a desired level or not. However, this is a costly approach – requiring finding the HD over all challenges (size N) of all pairs of PUFs in a batch $O(M^2 \times N)$, and does not identify the root cause (μ- vs. σ- fault), nor diagnose the fault location(s). Our proposed test methods use at most 4 challenges over the M PUFs in a batch ($O(M)$) to identify the fault type, and the diagnosis methods ($O(n \times M \times N)$ where n is the number of stages for the APUFs) can pinpoint which δ has the specific fault.

The test flow shown in Fig. 5 consists of two phases: pre-processing and test / diagnosis. Given the response matrix of a set of APUFs from a production line, a desired uniqueness lower bound, and tradeoff preference for True Positive Ratio vs. False Positive Ratio (TPR, FPR), the pre-processing stage uses a database of simulation results such as the plot in Fig. 4 to obtain the corresponding fault intensities (K_d and L_d), and uses them to find out various thresholds as described in subsection V-C. Then, 4 algorithms are used for test and diagnosis for μ- and σ-fault accordingly.

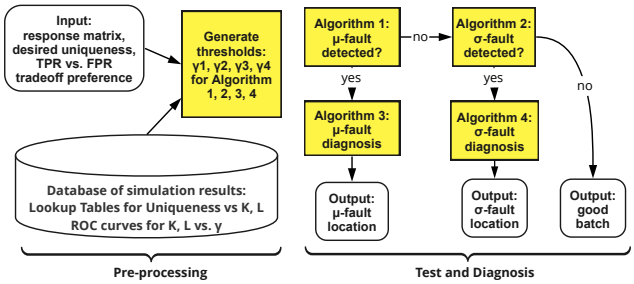

Fig. 5: Overall flow: test, diagnosis of faulty APUF batches.

A. Test methodologies: testing μ- or σ- faults

We propose two low-cost tests, motivated by the response matrices in Fig. 3, with two and four challenges respectively taken over an APUF batch.

1) Algorithm 1 (μ-fault test): It is easy to see that, given a single challenge $\mathbf{c}$, a good APUF batch of size M tends to have half of the APUFs with positive responses and another half with negative responses, which implies $\sum_{i=1}^{M} R_i(\mathbf{c}) = 0$.

With a μ-fault, if the challenge $\mathbf{c}$ picks the abnormal δ, then the number of APUFs with positive response deviates from half, which means $\sum_{i=1}^{M} R_i(\mathbf{c}) \to \pm M$. Thus, the ratio of APUFs with positive response can be used to indicate μ-fault with some given threshold $\gamma_1 \in [0, 1]$:

$$\left| \sum_{i=1}^{M} R_i(\mathbf{c})/M \right| > \gamma_1 \text{: decide } \mu\text{-fault.}$$

To design a μ-fault test without knowing which δ is at fault, as is shown in Algorithm 1, at least two challenges $\mathbf{c}, \bar{\mathbf{c}}$ are needed, where $\bar{\mathbf{c}}$ is the complement of $\mathbf{c}$. This way, any abnormal δ will be picked by either $\mathbf{c}$ or $\bar{\mathbf{c}}$, for some γ_1.

2) Algorithm 2 (σ-fault test): Given the response matrix of a set of M APUFs ($\mathcal{A}$) and two arbitrarily chosen challenges $(\mathbf{c}, \mathbf{c}')$, a good APUF batch tends to have half the APUFs producing the same response for $\mathbf{c}$ and $\mathbf{c}'$. This implies the inner product of $\mathbf{R}(\mathcal{A}, \{\mathbf{c}\})$ and $\mathbf{R}(\mathcal{A}, \{\mathbf{c}'\})$ tends to be 0.

Algorithm 1: μ-fault test with 2 challenges

Input: $\mathbf{R}(\mathcal{A}, \{\mathbf{c}, \bar{\mathbf{c}}\})$ of size $M \times 2$, $\gamma_1 \in [0, 1]$
Output: μ-fault existence
// good batch has both $s_1, s_2 \to 0$
$s_1 \leftarrow \sum_{i=1}^{M} R_i(\mathbf{c})/M$
$s_2 \leftarrow \sum_{i=1}^{M} R_i(\bar{\mathbf{c}})/M$
// either $\mathbf{c}$ or $\bar{\mathbf{c}}$ must select the abnormal δ affected by μ-fault, if exists
$s = \max(|s_1|, |s_2|)$ /* bad batch has $s \to 1$ */
if $s > \gamma_1$ **then**
  report μ-fault exists
end

Algorithm 2: σ-fault test with 4 challenges

Input: $\mathbf{R}(\mathcal{A}, \{\mathbf{c}, \bar{\mathbf{c}}, \mathbf{e}, \bar{\mathbf{e}}\})$ of size $M \times 4$, $\gamma_2 \in [0, 1]$
Output: σ-fault existence
// 1 out of these 4 pairs must both pick the abnormal δ affected by the σ-fault, if exists
$s_1 \leftarrow \mathbf{R}(\mathcal{A}, \{\mathbf{c}\})^T \mathbf{R}(\mathcal{A}, \{\mathbf{e}\})/M$
$s_2 \leftarrow \mathbf{R}(\mathcal{A}, \{\mathbf{c}\})^T \mathbf{R}(\mathcal{A}, \{\bar{\mathbf{e}}\})/M$
$s_3 \leftarrow \mathbf{R}(\mathcal{A}, \{\bar{\mathbf{c}}\})^T \mathbf{R}(\mathcal{A}, \{\mathbf{e}\})/M$
$s_4 \leftarrow \mathbf{R}(\mathcal{A}, \{\bar{\mathbf{c}}\})^T \mathbf{R}(\mathcal{A}, \{\bar{\mathbf{e}}\})/M$
// good batch has all 4 inner products $\to 0$
$s = \max(|s_1|, |s_2|, |s_3|, |s_4|)$ /* bad batch has $s \to 1$ */
if $s > \gamma_2$ **then**
  report σ-fault exists
end

Algorithm 3: μ-fault diagnosis

Input: $\mathbf{R}(\mathcal{A}, \mathcal{C})$ of size $M \times N$, $\gamma_3 \in [0, 1]$
Output: candidate set of δs for μ-fault
$D \leftarrow \emptyset$
for *each* $\delta_i^{(x)}$ **do**
  // a good $\delta_i^{(x)}$ has both s_1 and $s_2 \to 0.5$
  $s_1 \leftarrow$ ratio of $+1$ in $\mathbf{R}(\mathcal{A}, \mathcal{C}_{i,+}^{(x)})$
  $s_2 \leftarrow$ ratio of -1 in $\mathbf{R}(\mathcal{A}, \mathcal{C}_{i,-}^{(x)})$
  $s \leftarrow |s_1 - s_2|$ /* a bad $\delta_i^{(x)}$ has $s \to 1$ */
  if $s > \gamma_3$ **then**
    $D \leftarrow D \cup \{\delta_i^{(x)}\}$
  end
end
report D /* candidate set δs for μ-fault */

Algorithm 4: σ-fault diagnosis

Input: $\mathbf{R}(\mathcal{A}, \mathcal{C})$ of size $M \times N$, $\gamma_4 \in [0, 1]$
Output: candidate set δs for σ-fault
$D \leftarrow \emptyset$
for *each* $\delta_i^{(x)}$ **do**
  $s_1 \leftarrow 0$, $s_2 \leftarrow 0$
  for *each row* $j \in \mathcal{A}$ **do**
    $s_1 += \left|\sum_{\mathbf{c} \in \mathcal{C}_{i,+}^{(x)}} R_j(\mathbf{c})\right|$
    $s_2 += \left|\sum_{\mathbf{c} \in \mathcal{C}_{i,-}^{(x)}} R_j(\mathbf{c})\right|$
  end
  // a good $\delta_i^{(x)}$ has both s_1 and $s_2 \to 0$
  $s = (s_1 + s_2)/(M \times N)$ /* a bad $\delta_i^{(x)}$ has $s \to 1$ */
  if $s > \gamma_4$ **then**
    $D \leftarrow D \cup \{\delta_i^{(x)}\}$
  end
end
report D /* candidate δs for σ-fault */

With a σ-fault, when both $\mathbf{c}$ and $\mathbf{c}'$ pick the abnormal δ, then the number of APUFs producing the same response for the two challenges will deviate from half of M. Specifically, the inner product of $\mathbf{R}\big(\mathcal{A}, \{\mathbf{c}\}\big)$ and $\mathbf{R}\big(\mathcal{A}, \{\mathbf{c}'\}\big)$ tends to $+M$ (or $-M$) if $\mathbf{c}, \mathbf{c}'$ pick the abnormal δ with the same (or opposite) sign in Δ_n. This presents a way to indicate the existence of σ-fault with some given threshold $\gamma_2 \in [0, 1]$:

$$|\mathbf{R}\big(\mathcal{A}, \{\mathbf{c}\}\big)^T \mathbf{R}\big(\mathcal{A}, \{\mathbf{c}'\}\big)/M| > \gamma_2\text{: decide } \sigma\text{-fault.}$$

To design the σ-fault test without knowing which δ is at fault, we must somehow obtain 2 challenges that both pick the abnormal δ. This is achieved by carefully selecting 4 challenges in Algorithm 2. Here, we first randomly choose two different challenges $\mathbf{c}, \mathbf{e}$, and then find their complements: $\bar{\mathbf{c}}$ and $\bar{\mathbf{e}}$. For an APUF batch with a σ-fault at any $\delta_i^{(x)}$, it can be shown that at least 2 of these 4 challenges (selected as complementary pairs, which is crucial) will guarantee to both pick the abnormal $\delta_i^{(x)}$. The responses of all APUFs to these two challenges tend to be identical or completely different and hence the inner product of the responses of this challenge pair tends to $\pm M$.

B. Diagnosis methodologies: pinpointing the faulty δ

The proposed diagnosis algorithms go through each of the $\delta_i^{(x)}$ to test whether it is a fault candidate. Fig. 3 demonstrates clear patterns when the columns of the response matrix are organized by target sets: 1) the responses of a good APUF batch tend to be unbiased (half positive and half negative) over any target set; 2) with a μ-fault, within the target sets of the abnormal δ, the responses tend to be the same overall APUFs (thus showing a "striped" pattern); 3) with a σ-fault, within the target sets of the abnormal δ element, the responses follow a "checkered" pattern. This observation motivates the diagnosis algorithms.

1) Algorithm 3 (μ-fault diagnosis): With a μ-fault, under the target set columns of the abnormal δ, all the APUFs tend to have similar responses. Thus Algorithm 3 checks how much the ratio of $+1$ differs from that of -1 in those responses to identify such a syndrome, iteratively for each δ as a candidate.

2) Algorithm 4 (σ-fault diagnosis): With a σ-fault, the affected δ element has a larger variance, so in many APUFs it will be large and positive and in equally many other APUFs it will be large and negative. As such, for the pair of target sets of this δ elements, half the APUFs will produce positive responses under one target set (say $\mathcal{C}_{i,+}^{(x)}$) and a negative under the other one (say $\mathcal{C}_{i,-}^{(x)}$), and half the APUFs will have the opposite pattern, leading to the "black-white, white-black" checkered pattern. Such a checkered pattern in the target sets can be captured by $\left(\sum_{j=1}^{M} \left|\sum_{\mathbf{c} \in \mathcal{C}_{i,+}^{(x)}} R_i(\mathbf{c})\right| + \sum_{j=1}^{M} \left|\sum_{\mathbf{c} \in \mathcal{C}_{i,-}^{(x)}} R_i(\mathbf{c})\right|\right)/(MN)$. If this score is larger than a threshold γ_4, then the corresponding $\delta_i^{(x)}$ suffers from σ-fault. This is shown in Algorithm 4.

C. Threshold selection according to desired fault intensities

For all 4 algorithms, the proper thresholds $(\gamma_1, \gamma_2, \gamma_3, \gamma_4)$ can be selected from offline simulation results. Starting from a desired lower bound of uniqueness U_d, one can find the corresponding desired upper bound of fault intensities K_d and L_d from Fig. 4. To further obtain thresholds we need Receiver-Operating-Characteristics (ROC) curves to consider tradeoffs of True Positive Rates (TPR) versus False Positive Rates (FPR) of a particular test algorithm. Here, TPR is defined as TP / (TP + FN), and FPR as FP / (FP + TN), for TP = # True Positives, TN = # True Negatives, FP = # False Positives, and FN = # False Negatives. Ideally, TPR = 1 and FPR = 0.

An example of TP would be a μ-fault APUF batch with intensity $K_{real} = 10$, detected as faulty with a desired fault intensity $K_d = 5$; an example of FP would be a μ-fault APUF batch with intensity $K_{real} = 10$, detected as faulty with a desired fault intensity $K_d = 20$.

Fig. 6 show two examples of ROC curves for μ- and σ- fault test and diagnosis, under different fault intensities: the curves are drawn out by varying the threshold γs. The thresholds can be selected by looking at these ROC curves of the associated tests, depending on the desired (TPR, FPR) tradeoff; a general rule of thumb would be to pick one nearest the upper left-hand corner, where TPR = 1 and FPR = 0.

VI. Simulation Results

We now present Monte-Carlo simulation results of the proposed test and diagnosis algorithms in Table II. Results are

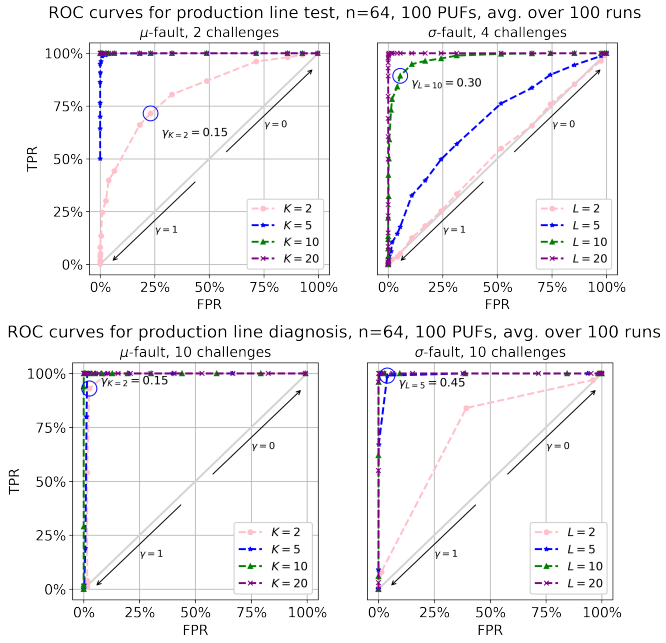

Fig. 6: Threshold γ selection can be done via ROC curves for (top) testing and (bottom) diagnosing based on the tradeoff between TPR and FPR.

evaluated based on TPR (shown in bold fonts in black) and FPR (shown in red color), under different desired fault intensities (K_d, L_d) and real fault intensities (K_{real}, L_{real}). The results are obtained as averages over 1000 or 100 runs (number of batches) respectively. Note that the thresholds used in this section are different from which shown in Fig. 6.

A. Test results for Algorithm 1 and 2

Consider a batch of 100 64-stage APUFs from an unknown production line. For a given desired uniqueness U_d, Algorithm 1 uses 2 challenges to detect whether this batch suffers from μ-fault (with $U_m < U_d$) or not (good or suffers from a σ-fault). Algorithm 2, on the other hand, uses 4 challenges to determine whether a σ-fault exists (i.e. $L_{real} > L_d$ generated based on U_d or not).

Overall, Table II shows the effectiveness in Algorithms 1 and 2: with only a couple of challenges, we can precisely detect faulty production lines when the fault density is high ($K_{real} \geq 5$, $L_{real} \geq 10$). As expected, both TPR and FPR increase as the real fault intensity K_{real} and L_{real} become larger; both TPR and FPR decrease as the desired fault intensity K_d and L_d increase. This illustrates the tradeoff between TPR and FPR that needs to be considered in the test process. In general, the effectiveness for σ-faults is lower than that of μ-faults. This is because for σ-faults, the mean is still zero, but the variance is larger, which intuitively is harder to notice, at least with a batch of 100 APUF samples.

B. Diagnosis results for Algorithm 3 and 4

For diagnosis, we assume the given production line of 64-stage APUFs suffers from μ- (σ-) fault with a fault intensity no smaller than the desired fault intensity, i.e. $K_{real} \geq K_d$ ($L_{real} \geq L_d$), with $M = 100, N = 100$ for Algorithm 3 and 4, respectively.

TABLE II: TPR (bold) and FPR (red) for μ- and σ-fault test and diagnosis results.

K_d or L_d	test								diagnosis			
	μ-fault with K_{real}				σ-fault with L_{real}				good with N challenges			
									μ		σ	
	2	5	10	20	2	5	10	20	100	50	100	200
2	**0.72**	**0.99**	**1**	**1**	**0.65**	**0.85**	**0.99**	**1**	0.02	1	1	1
5	0.23	**0.98**	**1**	**1**	0.55	**0.75**	**0.99**	**1**	0	0.98	0.44	0.05
10	0	0.20	**1**	**1**	0.19	0.41	**0.95**	**1**	0	0.02	0	0
20	0	0	0.75	**1**	0.07	0.21	0.88	**1**	0	0.06	0	0

According to the diagnosis results in Table II, TPR (correct location detected) remains 1, and FPR (non-faulty location detected as faulty) varies based on fault intensity. In general, diagnosis resolutions are excellent for μ-faults, even with K as low as 2. On the other hand, σ-faults are harder to diagnose with higher FPRs. This is as expected, since when the fault intensity is small ($L = 2$), the faulty δ tends to be similar to the good δs and it becomes hard to pinpoint the faulty δs. As the fault intensity increases or the number of challenges used in the test increases, FPR reduces.

VII. Conclusion

Uniqueness is an important quality for APUF as a promising security primitive. We observed that a widely-used uniqueness metric is flawed and defined a modified uniqueness metric. We showed that two types of production line faults, μ-faults and σ-faults, have great impact uniqueness, as is captured by the modified uniqueness metric. We proposed a low-cost and efficient test flow for testing and diagnosing these production line faults for APUF batches.

References

[1] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[2] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency - Practice and Experience*, vol. 16, pp. 1077–1098, 09 2004.

[3] W. Che, F. Saqib, and J. Plusquellic, "Puf-based authentication," in *IEEE ICCAD*, 2015, pp. 337–344.

[4] Y. Lao and K. K. Parhi, "Statistical analysis of MUX-based physical unclonable functions," *IEEE TCAD*, vol. 33, no. 5, pp. 649–662, 2014.

[5] Y. Cui, C. Wang, W. Liu, Y. Yu, M. O'Neill, and F. Lombardi, "Low-cost configurable ring oscillator puf with improved uniqueness," in *IEEE ISCAS*, 2016, pp. 558–561.

[6] S. Khan, A. P. Shah, S. S. Chouhan, N. Gupta, J. G. Pandey, and S. K. Vishvakarma, "A symmetric d flip-flop based puf with improved uniqueness," *Microelectronics Reliability*, vol. 106, p. 113595, 2020.

[7] C. Gu, N. Hanley, and M. O'Neill, "FPGA-based strong PUF with increased uniqueness and entropy properties," in *IEEE ISCAS*, 2017.

[8] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *IEEE HOST*, 2010, pp. 94–99.

[9] Y. Wei, T. Fox, V. Dumoulin, W. Rao, and N. Devroye, "APUF faults: Impact, testing, and diagnosis," in *DATE*, 2022, pp. 442–447.

[10] L. Feiten, M. Sauer, and B. Becker, "On metrics to quantify the inter-device uniqueness of PUFs," Cryptology ePrint Archive, Paper 2016/320, 2016. [Online]. Available: https://eprint.iacr.org/2016/320

[11] Z. Jouini, J.-L. Danger, and L. Bossuet, "Performance evaluation of silicon physically unclonable function by studying physical values," 2011.

[12] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on fpgas," *2010 International Conference on Reconfigurable Computing and FPGAs*, pp. 298–303, 2010.