Late Breaking Results: Automatic Anomaly Detection Method in Physical Unclonable Functions using Data Mining Techniques

Mohammad Reza Heidari Iman[1], Sergio Vinagrero Gutierrez[2], Elena-Ioana Vatajelu[1], and Giorgio Di Natale[1]
[1]Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, 38000 Grenoble, France
[2]Institut des Nanotechnologies de Lyon (INL)
{mohammadreza.heidari-iman, ioana.vatajelu, giorgio.di-natale}@univ-grenoble-alpes.fr
sergio.vinagrero@ec-lyon.fr

***Abstract*—Physical Unclonable Functions (PUFs) present a promising alternative to traditional cryptographic techniques for securing sensitive information in modern circuits. By exploiting inherent process variability, PUFs generate unique secrets dynamically, thus eliminating the need for data storage. However, a major challenge in PUF-based security is distinguishing valid PUFs from those that may have been tampered with or are invalid (*i.e.*, not belonging to the original design). This paper proposes a data mining-based approach for detecting anomalies and identifying tampered or invalid PUFs. The proposed method mines a set of rules that describe the expected behavior of the PUF, with deviations from these rules signaling potential security issues and vulnerabilities. Experimental results demonstrate that the method effectively identifies invalid or tampered PUFs, showcasing its potential for enhancing PUF-based security systems.**

***Index Terms*—physical unclonable functions, pattern detection in PUF, anomaly detection, data mining in PUF**

I. Introduction

The rise of the Internet of Things (IoT) is driving the deployment of hundreds of interconnected devices requiring secure identification, typically managed through centralized servers. Physical unclonable functions offer a promising alternative to traditional identification systems based on Non-Volatile Memory (NVM), leveraging intrinsic manufacturing variations to produce unique and unclonable responses. Traditional protocols using PUFs often rely on centralized databases storing Challenge-Response Pairs (CRPs) for device identification, resulting in massive databases. To address this scalability issue, some approaches propose compact PUF models as a substitute for CRP storage [1].

This paper introduces a novel method to leverage data mining techniques for PUF traceability and lifecycle management. By maintaining patterns across multiple PUF designs, architectures, and foundries, we propose methods to verify device legitimacy and detect tampering by comparing real-time responses with expected patterns. These approaches work because there is no fully ideal PUF in reality. Some CRPs are slightly biased and these approaches are able to identify the unique properties generated by the bias. Furthermore, periodic monitoring can be employed to flag devices nearing the end of their operational life or showing anomalous behavior, enhancing both security and lifecycle management in IoT networks relying on PUFs.

II. Proposed Methodology

The proposed method, summarized in Figure 1, consists of two main phases: *Pattern Mining* and *Anomaly Detection*. Initially, the behavior of the PUF is studied through the golden CRP dataset extracted during enrollment. Ultimately, the mined rules will be used to flag tampered or invalid instances. In the *Pattern Mining* phase, the proposed pattern miner extracts the set of rules from the CRP dataset. In the *Anomaly Detection* phase, new instances are tested against the mined rules from the previous phase, and differences in the number of matching rules are flagged as potentially tampered or invalid instances, along with a detailed report of the process. In the following subsections, each phase of the method is discussed in more detail.

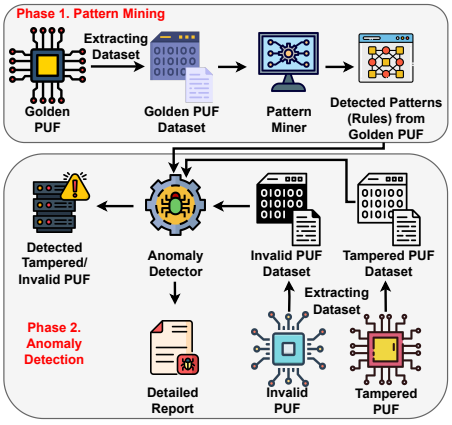

Figure 1: General Flow of the Proposed Method

A. Pattern Mining

During this phase our proposed algorithm leverages techniques from data mining and association rule mining to extract patterns from the CRP dataset at enrollment. Our algorithm operates in three main steps: Initialization and Preprocessing, Rule Mining, and Time Labeling. After preprocessing, during the rule mining step, a set of association rules in the general form of $antecedent \rightarrow consequent$ is mined using the apriori algorithm [2]. In the Time Labeling step, our proposed algorithm is employed to mine a set of temporal association rules in the forms of $next[\mathcal{N}] = antecedent \rightarrow next[\mathcal{N}]consequent$, $before[\mathcal{N}] = antecedent \rightarrow before[\mathcal{N}]consequent$ and $always = antecedent \rightarrow (always)consequent$. The combined use of $next[\mathcal{N}]$ and $before[\mathcal{N}]$ rules reveals correlations between responses, while $always$ rules identify consistent behaviors analogous to autocorrelation. Together, these rules analyze response reliability and inter-correlation patterns, enabling the detection of tampered or invalid PUFs by flagging deviations from

* Institut National Polytechnique Grenoble Alpes

978-3-9826741-0-0/DATE25/© 2025 EDAA

Algorithm 1: Proposed Pattern Miner

```
1  Input: 𝒩, 𝒟𝒮
2  Output: next[𝒩] = antecedent → next[𝒩]consequent,
     before[𝒩] = antecedent → before[𝒩]consequent
     /* Initialization and Preprocessing */
3  ℛ = antecedent → consequent
4  forall f ∈ 𝒟𝒮 do
5      𝒟𝒮′ = MoveUp(f(𝒩))
     /* Rule Mining */
6  ℛ ← apriori(𝒟𝒮′)
     /* Time Labeling */
7  if (𝒩 == 0) and (ℛ.antecedent == (t ∈ 𝒟𝒮′)) and (ℛ.consequent == (f ∈ 𝒟𝒮′))
     then
8      always ← label(ℛ)
9  else if (𝒩 > 0) and (ℛ.antecedent == (t ∈ 𝒟𝒮′)) and
     (ℛ.consequent == (f ∈ 𝒟𝒮′)) then
10     next[𝒩] ← label(ℛ)
11 else if (𝒩 > 0) and (ℛ.antecedent == (f ∈ 𝒟𝒮′)) and (ℛ.consequent ==
     (t ∈ 𝒟𝒮′)) then
12     before[𝒩] ← label(ℛ)
```

the expected behavior as potential security issues. Once extracted, the mined rules are stored in the server and the original dataset can be discarded, reducing the overall storage needs, at the trade-off of reducing the certainty of the validity of the devices.

Algorithm 1 outlines the steps of the pattern miner for extracting rules from the CRP dataset. Here $\mathcal{DS}$ represents the input dataset, $\mathcal{DS}'$ denotes the preprocessed dataset, a dataset row represents the CRPs of an instance and a step interval refers to the distance between rows. The algorithm generates three types of rules in the forms $always$, $next[\mathcal{N}]$ and $before[\mathcal{N}]$, where $\mathcal{N}$ specifies the step interval used to determine correlations between a given row in the dataset and the row located $\mathcal{N}$ steps after or before it. The dataset columns are divided into two groups, target values, represented by t, and feature values, represented by f, and the pattern miner extracts correlations between these t and f values. In line 5 of the algorithm, the dataset is prepared for mining the $next[\mathcal{N}]$ and $before[\mathcal{N}]$ patterns by moving all feature values $\mathcal{N}$ records up from their original positions (MoveUp($f(\mathcal{N})$)), while the position of target values remains unchanged. The apriori algorithm is then applied to this preprocessed dataset, mining a set of association rules. These rules are subsequently forwarded to the Time Labeling step, producing temporal association rules of $always$, $next[\mathcal{N}]$ and $before[\mathcal{N}]$ that are crucial for the *Anomaly Detection* phase.

B. Anomaly Detection

During this phase, the *Anomaly Detector* uses the set of rules mined from the golden CRP dataset and a new set of CRPs that may be either invalid or tampered. The new CRPs are compared with either $antecedent$ or $consequent$ of the rules that have been stored on the server. Any irregularity between the expected pattern of bit-flips (*i.e.*, reliability) defined by the number of matched rules given the new CRPs is flagged as a potential security issue.

III. Experimental Results

We validated our method with a CRP dataset obtained from 84 STM32L152RE microcontrollers from ST Microelectronics [3]. Each device contains 80 kB of memory resulting in 655360 bits. For this study, a total of 16000 rules were mined in approximately 15 minutes using a low-end computer, demonstrating that the proposed method is efficient and not resource-intensive. To simulate the behavior of tampered PUFs, new instances

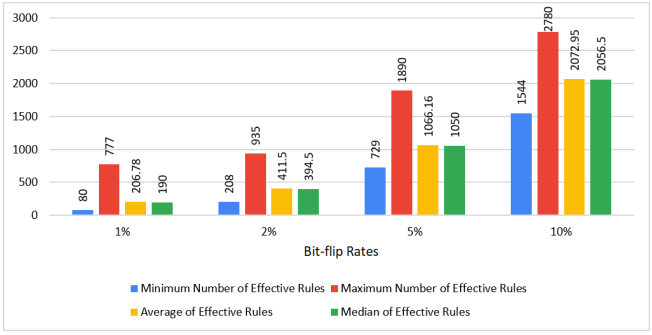

Figure 2: Fault Detection Results for Different Bit-flip Rates

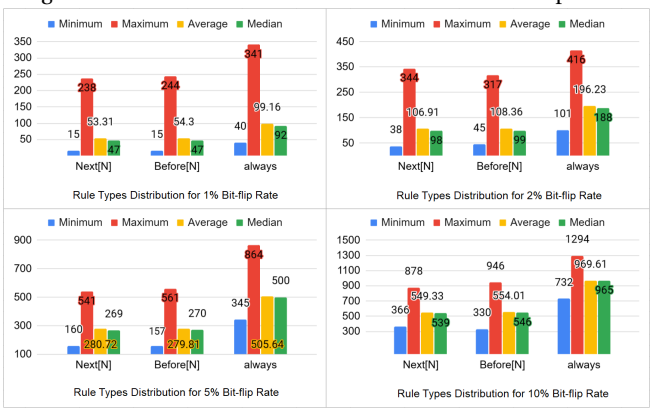

Figure 3: Rule Types Distribution for each Bit-flip Rate

were generated by uniformly flipping responses in the CRP dataset at rates of 1%, 2%, 5%, and 10% over 1000 iterations.

Figure 2 presents the minimum and maximum number of effective rules capable of detecting faults for each flipping rate, along with the calculated average and median values of these rules over 1000 iterations of bit-flipping. The calculated averages and medians for each bit-flip rate are closely aligned, indicating the proposed detection mechanism is consistent, robust, and scalable across all tested bit-flip rates, without being significantly impacted by outliers.

Figure 3 shows the distribution of mined rule types for each bit-flip rate, along with their minimum, maximum, median, and average number of rules. The average and median values for the $next[\mathcal{N}]$ and $before[\mathcal{N}]$ rules are very close across all rates, as the combination of these rules accurately measures the response interdependence, serving as the reference to discriminate new instances.

IV. Conclusion

In this paper, we proposed a data mining-based method valuable for detecting tampered or invalid instances. Experimental results demonstrated the method's effectiveness.

Acknowledgement

This work is partially funded by the "Resilient Trust" and "Neuropuls" projects of the EU's Horizon Europe research and innovation programme under grant agreements No. 101112282 and No. 101070238.

References

[1] A. Ali-Pour *et al.*, "Strong puf enrollment with machine learning: A methodical approach," *Electronics*, vol. 11, no. 4, p. 653, 2022.

[2] M. R. Heidari Iman *et al.*, "ARTmine: Automatic association rule mining with temporal behavior for hardware verification," in *DATE*, 2024, pp. 1–6.

[3] S. Vinagrero *et al.*, "SRAM-Based PUF Readouts," vol. 10, no. 1, p. 333.