FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

NON-INVASIVE TECHNIQUES TOWARDS RECOVERING HIGHLY SECURE UNCLONABLE CRYPTOGRAPHIC KEYS AND DETECTING COUNTERFEIT MEMORY CHIPS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Bashir Mohammad Sabquat Bahar Talukder

2021

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Bashir Mohammad Sabquat Bahar Talukder, and entitled Non-invasive Techniques Towards Recovering Highly Secure Unclonable Cryptographic Keys and Detecting Counterfeit Memory Chips, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Ou Bai

A. Selcuk Uluagac

Ananda Mohan Mondal

Md Tauhidur Rahman, Major Professor

Date of Defense: November 09, 2021

The dissertation of Bashir Mohammad Sabquat Bahar Talukder is approved.

Dean John Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2021

© Copyright 2021 by Bashir Mohammad Sabquat Bahar Talukder

All rights reserved.

DEDICATION

To my parents, Mohammad Faizur Rahman Talukder and Anjuman Ara Begum,

my sister, Taieba Fatima Faizun,

and my high school teacher, Ms. Fahmida Sultana, who continuously inspired me to pursue engineering at my early age.

ACKNOWLEDGMENTS

Firstly, I want to thank my advisor Dr. Md Tauhidur Rahman, for his precious guidance, mentorship, and continuous motivation throughout my doctoral research. Without his advice and endless support, this dissertation would never have been pulled off. I am extremely grateful to him for consciously supporting me for the last four and half years and understanding me in my difficult times. I would also like to thank Dr. Ou Bai, Dr. A. Selcuk Uluagac, and Dr. Ananda Mohan Mondal for their insightful comments, guidance, encouragement, and important technical feedback to take this research to the next level. I want to thank my wife and colleague Farah Ferdaus for her emotional support and technical advice, which easily saved many of my days. I would also like to thank Prof. Dr. Aleksandar Milenković, Dr. Biswajit Ray, Dr. Emil Jovanov, Dr. Huaming Zhang, and Dr. Vineetha Menon from the University of Alabama in Huntsville. Their continuous inspiration and technical advice are highly motivating and helped me learning many aspects of Computer Engineering. Lastly, I would like to show my gratitude towards two of my closest friends, Mr. Adnan Jahangir and Mr. Rakibul Kabir Rifat, for their constant support and encouragement. I would also like to express my gratitude towards FIU University Graduate School and the Department of Electrical and Computer Engineering at Florida International University for their continuous administrative support. I would also like to acknowledge the support provided by National Science Foundation; this dissertation is mostly based upon the work supported by National Science Foundation under the grant number CNS-1850241.

ABSTRACT OF THE DISSERTATION

NON-INVASIVE TECHNIQUES TOWARDS RECOVERING HIGHLY SECURE UNCLONABLE CRYPTOGRAPHIC KEYS AND DETECTING COUNTERFEIT MEMORY CHIPS

by

Bashir Mohammad Sabquat Bahar Talukder

Florida International University, 2021

Miami, Florida

Professor Md Tauhidur Rahman, Major Professor

Due to the ubiquitous presence of memory components in all electronic computing systems, memory-based signatures are considered low-cost alternatives to generate unique device identifiers (IDs) and cryptographic keys. On the one hand, this unique device ID can potentially be used to identify major types of device counterfeitings such as remarked, overproduced, and cloned. On the other hand, memory-based cryptographic keys are commercially used in many cryptographic applications such as securing software IP, encrypting key vault, anchoring device root of trust, and device authentication for could services. As memory components generate this signature in runtime rather than storing them in memory, an attacker cannot clone/copy the signature and reuse them in malicious activity. However, to ensure the desired level of security, signatures generated from two different memory chips should be completely random and uncorrelated from each other. Traditionally, memory-based signatures are considered unique and uncorrelated due to the random variation in the manufacturing process. Unfortunately, in previous studies, many deterministic components of the manufacturing process, such as memory architecture, layout, systematic process variation, device package, are ignored. This dissertation shows that these deterministic factors can significantly correlate two memory sig-

natures if those two memory chips share the same manufacturing resources (i.e., manufacturing facility, specification set, design file, etc.). We demonstrate that this signature correlation can be used to detect major counterfeit types in a non-invasive and low-cost manner. Furthermore, we use this signature correlation as side-channel information to attack memory-based cryptographic keys. We validate our contribution by collecting data from several commercially available off-the-shelf (COTS) memory chips/modules and considering different usage-case scenarios.

TABLE OF CONTENTS

CHAPTER PAGE

1. INTRODUCTION . . . 1
1.1 Research Objectives . . . 2
1.1.1 Research Objective 1: Anti-counterfeiting Solution . . . 2
1.1.2 Research Objective 2: Attacking on mPUF keys . . . 4
1.2 Significance of Study . . . 6
1.3 Organization of the Dissertation . . . 7

2. BACKGROUND AND LITERATURE REVIEW . . . 9
2.1 Overview of PUFs . . . 9
2.2 PUF Taxonomy . . . 11
2.3 Memory-based PUFs (mPUFs) . . . 13
2.3.1 Memory Organization and Memory Technologies . . . 14
2.3.2 Basic Principle of Memory-based PUFs (mPUFs) . . . 19
2.4 mPUF Applications . . . 23
2.4.1 Anti-counterfeiting Solution . . . 23
2.4.2 Cryptrographic Key Extraction from PUF Signature . . . 27

3. IMPACTS OF DETERMINISTIC FACTORS ON mPUF SIGNATURES 34
3.1 Introduction . . . 34
3.2 Architectural Variation . . . 35
3.3 Layout Variation . . . 36
3.4 Process Variation . . . 37
3.5 IC packaging . . . 38
3.6 Device Aging . . . 38

4. IDENTIFYING DRAM ORIGIN . . . 42
4.1 Introduction . . . 42
4.2 Objectives and Assumptions . . . 43
4.3 Proposed Method . . . 47
4.4 Performance Evaluation of Proposed Framework . . . 54
4.5 Limitations . . . 60
4.6 Conclusion . . . 61

5. IDENTIFYING SRAM ORIGIN . . . 62
5.1 Introduction . . . 62
5.2 Differences from Prior Works . . . 63
5.3 Proposed Method . . . 64
5.3.1 Feature Selection . . . 64
5.3.2 Identifying Authentic Memory Chips . . . 69
5.3.3 Proposed Framework . . . 71

5.3.4 Identifying Recycled Memory Chips . . . 72
5.4 Performance Evaluation of Proposed Framework . . . 72
5.4.1 Visualizing Features . . . 75
5.4.2 Labeling Test Memory Chips . . . 79
5.4.3 Identifying Recycled Memory Chips . . . 86
5.4.4 Evaluation Time . . . 89
5.5 Limitations . . . 89
5.6 Conclusion . . . 90

6. NON-INVASIVE HEURISTIC ATTACK ON mPUF . . . 91
6.1 Vulnerabilities of mPUFs and weakness of PUF Metrics . . . 91
6.1.1 Deterministic Components of mPUF keys . . . 91
6.1.2 Acceptance of Errors in PUF Protocol . . . 92
6.1.3 Weakness of PUF Metrics . . . 93
6.2 Threat Model . . . 95
6.3 Proposed Method . . . 96
6.3.1 Analyzing PUF Vulnerabilities . . . 96
6.3.2 Attacking an SRAM PUF . . . 97
6.4 Performance Evaluation of Proposed Attack . . . 103
6.4.1 Error Characterization of SRAM Signature . . . 104
6.4.2 Attacking an SRAM PUF . . . 105
6.4.3 Attacking in the Presence of a Fuzzy Extractor . . . 112
6.5 Mitigation . . . 113
6.5.1 Avoiding Error Correction Scheme . . . 113
6.5.2 Using Properly Designed Fuzzy extractor . . . 114
6.5.3 Pattern Distribution-aware CRPs . . . 114
6.5.4 Locality-aware CRPs . . . 115
6.6 Conclusion . . . 115

7. CONCLUSION AND FUTURE WORK . . . 117

BIBLIOGRAPHY . . . 121

APPENDICES . . . 142

VITA . . . 144

LIST OF TABLES

TABLE PAGE

2.1 Different types of IC counterfeitings. 24

2.2 Summary of counterfeit chip detection/identification methods. 28

4.1 Memory modules used in the data set. 55

4.2 Results from the one-class classifier. 58

5.1 List of SRAM chips in experiment. 74

5.2 Identifying SRAM manufacturer. 82

5.3 Identifying SRAM part number. 83

5.4 Mean accuracy, precision, recall, and F_1 score at high temperature . . . 85

6.1 List of SRAM chips used as *train devices*. 103

6.2 16-bit pattern characteristics and vulnerability to heuristic attack. . . . 106

6.3 Fuzzy min-entropy for SRAM chips. 113

LIST OF FIGURES

FIGURE PAGE

2.1 single bit arbiter PUF with m-bit challenge. 11
2.2 Memory organization in modern computer system. 14
2.3 A $4 \times 4 \times 4$ memory array. 15
2.4 Basic memory cell structure. 17
2.5 DRAM timing at read cycle. 17
2.6 IC design flow vs possible covert channels to introduce counterfeit ICs. . . 24
2.7 PUF applications. 27
2.8 `Rep` and `Gen` block of fuzzy extractor. 29
2.9 Simplified mPUF-based IOT device authentication protocol. 31
3.1 Different types of process variation. 38
4.1 Proposed framework for identifying the origin of the DRAM. 52
4.2 Selection of λ with different distribution of PPR_{neg} and PPR_{pos}. 53
4.3 Spatial locality of failed bits from randomly chosen DRAM pages. 56
4.4 Visualizing data in feature-space. 57
4.5 Visual representation of DRAM modules from Class 60
5.1 Illustration of Φ_1, Φ_2, and Φ_3. 65
5.2 Proposed protocol to identify counterfeit SRAM. 72
5.3 Visualizing feature distribution by manufacturer. 76
5.4 Visualizing feature distribution by part number. 78
5.5 Representation of SRAMs in feature-space, clustered by manufacturer. . 79
5.6 Representation of SRAMs in feature-space, clustered by part number. . 80
5.7 Visualizing feature distribution: fresh vs. aged. 87
6.1 Binomial vs. Gaussian distribution. 93
6.2 Constructing n-bit key from n'-bit word. 102

6.3 SRAM chip error characterization. 105

6.4 Visualizing 16-bit pattern distribution. 107

6.5 Spatial distribution of 0's and 1's. 107

6.6 Number of attempts vs. recovered CRPs. 109

6.7 Dependency of pattern heuristics with respect to logical address. 111

CHAPTER 1

INTRODUCTION

The field of cybersecurity has predominantly focused on the security of the software and the communication network due to the assumption that the underlying hardware is trustworthy and reliable. However, emerging attack vectors on hardware originating from the untrusted global supply chain, along with unintentional design and integration-caused hardware vulnerabilities, questions the well-regarded notion of hardware "root of trust". In the era of secure computing, many hardware-based security primitives use memory-based signatures to ensure device security, authenticity, and users privacy. These memory signatures are comparable to silicon fingerprints due to their reproducible nature. Currently, memory-based signatures are commercialized to generate device identifiers (IDs) for chip authentication or counterfeit chip detection and generating keys for various cryptographic applications.

From the application perspective, memory-based signatures are considered as the physical unclonable functions (PUF) [SRR16, SMRR20, KPHM18, G$^+$07]. By definition, PUF is a circuit that can generate a unique signature ("response") depending on its input pattern ("challenge") [RSS$^+$10]. However, unlike traditional PUFs (such as Arbiter PUF and Ring Oscillator PUF), memory-based PUFs (mPUFs) are only capable of generating a limited number of challenge-response pairs (CRPs) [RH14]. Therefore, traditional PUFs are also known as strong PUF, whereas mPUFs are known as weak PUF [RH14]. Although strong PUFs are the preferred choice for many cryptographic applications, the mPUFs provide a low-cost alternative solution as they do not require any additional hardware. Moreover, mPUF can replace strong PUFs in all possible applications with necessary adjustments in the PUF protocol [HYKD14]. Nevertheless, to ensure a similar level of security as strong PUFs,

mPUFs must have the following three properties- **(i)** reproducibility (reliability), (**ii**) uniqueness, and **(iii)** randomness (uniformity) [MGS13]. The reproducible property of mPUFs comes from the fact that electronics properties of mPUFs are immutable once it is fabricated. On the other hand, the uniqueness and the randomness of the mPUFs signature are believed to be a direct consequence of the random variation of the memory manufacturing process. However, we observe that mPUF signatures may have a significant correlation if they share the same manufacturing resources (fabrication facility and design specifications) due to similarities in their architecture, layout, and die packaging. unfortunately, these observations have been ignored in all previous research, which leads us to two distinct research opportunities- **(i)** exploiting mPUF correlation to simplify the mPUF-based counterfeit IC (Integrated circuit) detection technique, and **(ii)** using this same signature correlation as side-channel information to attack the mPUF key.

1.1 Research Objectives

1.1.1 Research Objective 1: Anti-counterfeiting Solution

Electronic counterfeiting is a longstanding problem with adverse long-term effects for many sectors, remaining on the rise. Globalization of the semiconductor supply chain has allowed worldwide fabrication of both authentic and counterfeit chips. In an established global semiconductor supply chain, several untrusted parties (foundry, assembly, third-party IPs, market distributor, etc.) are involved, any of whom can clone IP (intellectual property), insert hardware trojan, and/or include recycled, remarked, overproduced, out-of-spec/defective, and forge-documented chips [BB15, GHD+14, FC18]. Without question, the presence of counterfeit or inferior compo-

nents in commercial or military cyber-infrastructure could have catastrophic and far-reaching consequences on security and reliability domains because of their sub-standard quality, poor performance, and shorter life span [BB15, GHD+14, FC18]. Current techniques to detect and prevent major counterfeit types (e.g., recycled, remarked, etc.) are either invasive or expensive [GHD+14, FC18]. Furthermore, existing countermeasures target only a single counterfeit type.

Recently, mPUF-based anti-counterfeiting solutions gained a lot of attention as they can identify the major types of counterfeit chips (e.g., recycled, remarked, cloned, etc.) by tagging each authentic chip with a unique identifier [GHD+14, FC18]. As memory components are the essential part of most electronic chips, the mPUF-based solution does not incur additional hardware costs. Unfortunately, this solution is still not suitable for low-cost hardware as it requires an exhaustive registration process and maintains a centralized database of authentic chips' identifiers. In this dissertation, we will try to answer following three research questions (RQs) in order to simplify mPUF-based anti-counterfeiting solutions-

- **RQ1:** Can we avoid the exhaustive device registration phase? Can we use a statistical model to identify a group of memory chips instead of using a unique identifier for each chip?
- **RQ2:** Can we avoid the interactive communication protocol of mUF-based device authentication? Can we authenticate the device in offline mode?
- **RQ3:** What type of device counterfeitings can we identify?

To answer the above questions, this dissertation will explore the following three goals-

1. **Extraction of salient features:** We need a set of features that reliably classify memory manufacturer and memory specification, which is a non-trivial task and one of the novelties of our research. The objective is to extract a set of features

from memory-based signatures, which can uniquely correlate to a group of memory chips if they share the same architecture, layout, systematic process variation, and die packaging (i.e., sharing the same manufacturer and part number).

2. **Development of machine-learning-based framework:** An efficient machine-learning-based framework is required to learn manufacturer and part number-specific information from the extracted features. We use this statistical model (offline or online) to identify the manufacturer and part number for an unknown chip. It is worth mentioning that identifying the manufacturer and part number can provide almost the same level of protection against counterfeit memory chips as mPUF-based identifiers.

3. **Detection of other counterfeit types:** As the recycled chips share the same manufacturing resources and design specifications as the fresh chips, identifying the manufacturer and part number might not be sufficient to detect recycled chips. However, some electrical properties of recycled chips deviate due to the device age [KBTS+18], which might be observed from the feature distribution of the recycled chips. Therefore, distinctive distribution of one or multiple features can potentially identify recycled memory chips.

1.1.2 Research Objective 2: Attacking on mPUF keys

In order to be useful in practical security applications, the fingerprint or key generated by the PUF should be reliable and attack-resistant. Due to the open-access nature of the strong PUFs, numerous successful attacks have been proposed against those PUFs where the attacker does not require any physical access to the PUF [RSS+10, RvD13, RJA12]. On the contrary, mPUFs (more generally, weak PUFs) are resistant to such attacks as their protocol does not support accessing them di-

rectly from the outside world [RH14]. While most research focuses on low-cost techniques that generate reliable signatures from mPUFs, this dissertation will focus on security aspects of mPUFs that have been widely overlooked and answer the following four research questions (RQs).

- **RQ4:** What are the major challenges of mPUFs from using to generate low-cost keys or unique identifiers? Can we avoid expensive key generation schemes (e.g., fuzzy extractor [FRS20, FRS16]) for generating high-quality identifiers or keys for low-cost applications?
- **RQ5:** Do mPUFs have any weaknesses that can lead to non-invasive attacks? For example, can we exploit the correlation between to mPUF signature as side-channel information?
- **RQ6:** There have been several PUF metrics that are reliable and used to quantify the security of PUFs. Do any of these metrics have any weaknesses?
- **RQ7:** Can we develop invasive attack methodologies that do not need physical access to the target system?

Below is the four specific aims of this dissertation that will answer the above research questions.

1. **Discovering the weaknesses of memory-based signatures:** We will discover the vulnerabilities of an mPUF that can be exploited to recover or model the PUF responses in non-invasive ways without any physical access to the target device.
2. **Discovering the weaknesses of existing PUF quality metrics:** With the support of current PUF metrics, the mPUF is considered secure. This dissertation will aim to discover the weakness of existing PUF metrics and show that they do not always guarantee the security of mPUF signatures.
3. **Recovering/predicting memory-based PUF outputs:** The mPUFs are

considered robust against non-invasive attacks because of their limited CRPs. This dissertation will develop techniques to predict or model PUF outputs in non-invasive ways by exploiting the mPUF vulnerabilities.

4. **Developing attack-resistant mPUFs at a low cost:** An expensive fuzzy extractor generates high-quality keys/identifiers, which is not viable for low-cost systems [FRS20, FRS16]. Furthermore, other low-cost alternatives to fuzzy extractors can also be vulnerable to our new non-invasive attack. We will develop low-cost techniques for generating attack-resistant mPUFs from memory chips.

1.2 Significance of Study

Recent studies show that the global market share of counterfeit integrated circuits (ICs) is worth $169 billion [Kar20]. Among different counterfeit ICs, memory chips share the most considerable portion of the counterfeit IC market (∼17%) [FC18]. Moreover, another 28% of the fake chips are contributed by memory-integrated microprocessors, microcontrollers, and FPGAs [FC18], which are the most expensive components of any system. Hence, identifying counterfeit memory elements might be able to element those counterfeit components as well. Recent statistics showed that counterfeit chips incidents are growing at a rate of 25% each year [FC18]. The Senate Armed Services Committee confirmed more than 1,800 counterfeit electronic components incidents in U.S. military hardware in the early 2010's decade, and most of them are associated with counterfeit memory [Dep11]. The inclusion of counterfeit memory in an electronic system can endanger personal and national privacy, sabotage critical infrastructure, and damage the viability of entire business sectors because of their sub-standard quality, poor performance, and shorter lifespan. Section 818 of the U.S. National Defense Authorization Act (NDAA) requires defense

contractors to tighten supply chain traceability and parts procurement to minimize counterfeit risk [U.S18]. Although industry and academic researchers developed several methods to combat counterfeit memories, none of them are suitable for low-cost memory chips or memory integrated embedded devices. Hence, this research can benefit- individuals and the semiconductor industry from potential financial loss and the government from compromising national security and privacy. Additionally, Identifying the manufacturer and the specification is also essential to- **(i)** enforce the license agreement, **(ii)** tracking the chip quality, and **(iii)** ranking the memory products and worthiness [BTMR+20].

In addition to identifying counterfeit memory chips, it is also essential to determine the vulnerability of mPUF-based cryptographic applications. The mPUF-based cryptographic key has become very popular as they do not require any additional cost, and consequently, they have been deployed in many commercial embedded devices. Hence, analyzing the vulnerability to mPUFs is essential to ensure the risk-free application in privacy-, safety- and security-critical operations.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 provides necessary backgrounds on different memory technologies, the detailed construction of mPUFs using different memory technologies, and PUF applications on counterfeit device identification and cryptographic key generation. We also compare other existing anti-counterfeiting solutions with PUFs and known vulnerabilities of the mPUF key. In Chapter 3, we discuss different non-deterministic factors that can impact mPUFs' signature during the manufacturing process. In Chapters 4 and 5, we proposed a new anti-counterfeiting solution by extracting a feature-set from mPUF-based signatures.

Our proposed technique is much simpler than the PUF-based solution, which is demonstrated by two different types of memory technologies (DRAM and SRAM). Chapter 6 assesses new possible vulnerabilities of the mPUF key and demonstrates our idea on SRAM-based silicon data. Finally, we conclude our dissertation and propose the future direction of our research in Chapter 7.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This dissertation starts where other research works on memory-based PUF (mPUFs) have been called off. This chapter will provide an overview of PUFs, the fundamental difference between mPUFs and other PUFs, the construction of mPUFs, and PUF-based anti-counterfeiting solution and cryptographic key extraction.

2.1 Overview of PUFs

A PUF is a hardware security primitive that can translate a input ("challenge" or "stimuli") to a unique sequence ("signature" or "response") which is specific to the electrical characteristics of that system [RSS+10, MGS13, MBW+19, HYKD14]. The key idea behind a PUF is, although one can use the same architecture and manufacturing process to fabricate multiple PUF circuitry, the random variability in the manufacturing process impacts the electrical characteristics (e.g., timing path delay) randomly [HYKD14]. PUF utilizes this random variation on the electrical property to create a unique sequence. In general, an ideal PUF should have following three properties for reliable operations [RRFT15, MGS13, BTRFR19]-

1. **Uniqueness:** The PUF response should be unique and should not collide if multiple PUF devices are stimulated with the same challenge. In other words, if the responses generated from two PUFs (with the same challenge) are properly random, the *Hamming distance*[1] between two PUF responses should be $\sim 50\%$. However, if the number of test devices is large, then one-to-one *Hamming distance* becomes a very expensive metric to test (due to a large number of possible combinations). Furthermore, a single PUF is usually capable of producing multi-

[1] *Hamming distance* (or simply "distance") is defined as the number of bit difference between two binary strings.

ple challenge-response pairs. Consequently, assessing *Hamming distance* for each CRP makes it computationally expensive. Fortunately, *bit-aliasing* is a cheaper alternative to test uniqueness. For this metric, we create a binary string using all memory-bits from a specific bit-location of all PUF devices and measure the *Hamming weight* [MGS13, RHG+17] (Equation 2.1).

$$\beta_{l,C} = \frac{1}{k} \sum_{i=1}^{k} b^{i}_{l,C} \tag{2.1}$$

Here, $\beta_{l,C}$ is the bit-aliasing computed at the lth bit using k PUF devices with a unique challenge C, and $b^{i}_{l,C}$ is the lth bit response generated from the ith PUF device. Hence, for the n-bit responses generated from all k PUF devices, we have a set of βs. To make it simpler, researchers only focus on the average *bit-aliasing* defined by the Equation 2.2.

$$\beta_{C} = \frac{1}{n} \sum_{l=1}^{n} \beta_{C,l} \tag{2.2}$$

For a set of ideal PUFs, β_C (with unique challenge C) should be close to 0.5. This condition suggests that the probability of getting "0" (or "1") from a specific bit location over multiple PUFs is exactly 50%.

2. **Uniformity:** The PUF response should have a uniform distribution of "0" and "1". To satisfy this condition, the average *Hamming weight* of the PUF responses should be close to 50% [MGS13, RHG+17]. Mathematically, the uniformity can be expressed with Equation 2.3.

$$U_{C} = \frac{1}{n} \sum_{l=1}^{n} b_{l,C} \tag{2.3}$$

Here, $b_{l,C}$ is the lth bit of the response, which corresponds to challenge C. For a uniform response, the uniformity, U_C should be close to 50%.

3. **Reproducibility:** An ideal PUF device should produce the exact same response while assessed with the same challenge [MGS13]. Alternatively, two responses

produced by the same PUF with the same challenge should have a *Hamming distance* of 0. Ideally, A robust PUF should retain this property even with extreme variations in operating conditions.

2.2 PUF Taxonomy

The strength of PUF mainly depends on the number of challenge-response pairs (CRPs) that it can produce. Based on the number of CRPs, PUFs are categorized into two major types [MBW+19, HYKD14, RH14]-

1. **Strong PUF:** Strong PUFs typically support a large number of CRPs; more precisely, the number of CRPs grows exponentially with respect to the growth of strong PUF resources [MBW+19]. One of the popular architectures of strong PUF is MUX-controlled Arbiter PUF and Ring Oscillator PUF [SD07]. A single-

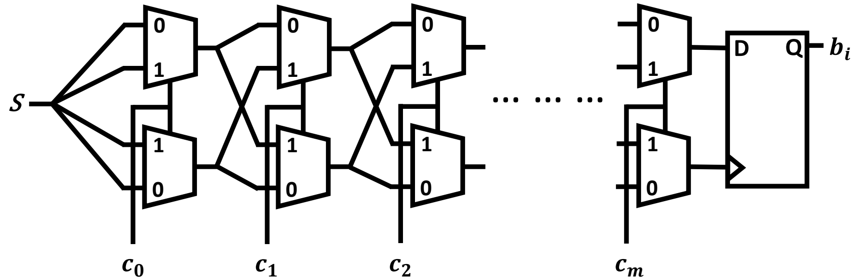


Figure 2.1: single bit arbiter PUF with m-bit challenge.

bit m-stage arbiter PUF is shown in Figure 2.1. In arbiter PUF, each bit of the challenge sequence (c_j) controls a pair of MUXes, and each pair of MUXes uses the output of the previous pair of MUXes. For any MUX, if the selector, c_j is 0, the top and bottom MUX conduct the same signal from the previous stage's top and bottom MUX, respectively. However, if c_j is 1, the top and bottom signals switch the side. If the source S generates a rising signal, the D flip-flop latches "0" if the delay at the D pin is higher than the delay at the clock pin;

otherwise, it latches “1” at the output. Note that a multi-bit arbiter PUF is constructed by cascading multiple single-bit arbiter PUFs. Due to the random process variation, each MUX consists of slightly different delay characteristics. Hence, the latched value at the D flip-flop is completely random and depends on the signals applied on each selector pin of each stage. Note that if we increase the Arbiter PUF structure by one pair MUXes (i.e., adding one more stage), it will support $(m + 1)$-bit challenge, and the number of CRPs will be increased two-fold. As a strong can produce a large number of CRPs, only a randomly chosen subset of CRPs are used in real applications.

2. **Weak PUF:** Weeks PUFs can only produce a limited number of CRPs (in the extreme case, only one CRP). Unlike the strong PUF, the number of CRPs grows linearly with respect to the weak PUF resources [MBW+19]. Note that most of the mPUFs are considered weak PUF, which are discussed in Section 2.3. In such mPUFs, the address of the memory is used as the challenge while the content of that address location (with some special arrangement) is fetched as the response. Due to the array-like structure of the memory chips, the number of CRPs linearly grows with the growth of the memory size (i.e., the number of CRPs is doubled if the number of the memory cell is also doubled).

Aside from the number of CRPs, there are other differences between the strong PUF and the Weak PUF. For example, the strong PUF uses an open-access protocol as opposed to the weak PUF [MBW+19, HYKD14, RH14]. In an open-access protocol, anyone can challenge the strong PUF and collect corresponding responses. However, an attacker cannot feasibly collect the whole set of CRPs from a strong because of its enormous size. Furthermore, for a strong PUF, a randomly sampled subset of CRPs ($S_{used,CRP}$) is used in real applications. Hence, even if an attacker succeeds to collect a subset of the CRPs ($S_{attacker,CRP}$), the probability of $S_{used,CRP} \cap S_{attacker,CRP}$

to be a non-empty set is very negligible. Moreover, as the number of CRP in strong PUF is very large, strong PUF uses a particular CRP only once to prevent reply attacks [MS09] and simplify the PUF protocol. On the other hand, memory-based weak PUFs are a cheaper solution compared to strong PUFs as a device can utilize its existing memory components without incurring any additional hardware cost. However, Weak PUF might require complex protocol to support some special application (e.g., IOT device authentication for cloud services) [HYKD14].

This dissertation mainly focuses on memory-based weak PUFs (or simply mPUFs) as they are more common than strong PUFs and are available in many commercial devices. In the next couple of sections, we will discuss different types of mPUFs and their typical applications.

2.3 Memory-based PUFs (mPUFs)

Memory chips are the ubiquitous component of modern computer systems. Typically, a modern computer system consists of multiple memory components. Interestingly, most types of memory chips are capable of generating PUF-like signatures without making any design modifications. However, generally, they require some non-standard memory operations (e.g., reducing read/write latency, latching start-up state, etc.) to function like a PUF [G+07, HBF07, BTRFR19, SRR16, KPHM18, TKYC16, SMRR20, JXW+15, MRFA15]. Fortunately, those operations can be performed by only OS/firmware level manipulation without making any hardware modification. Hence, except for some software overhead, the cost associated with mPUFs is very negligible. To understand the operation of mPUFs in more detail, here, we present the modern memory organization briefly and the basic principle of different types of memory components.

2.3.1 Memory Organization and Memory Technologies

Most of the modern computer system maintains a hierarchical memory organization to read and write data efficiently (see fig. 2.2). As technology does not allow a memory to be fast and cost-effective at the same time [Eve], different types of memories are used at each level of memory hierarchy to balance between cost and performance. In most of computer system, memory organization maintains the following hierarchy [PH16, BO16]- CPU → registers → private cache → shared cache → main memory → storage memory. Note that, in many smaller embedded systems, one or more components from this hierarchy might be fused together; for example, smaller micro-controllers may use a single SRAM (static random access memory) chip as both cache and main memory. Nevertheless, starting from the beginning down to the end of this hierarchy, each memory component is slower but bigger and cheaper than the previous one.

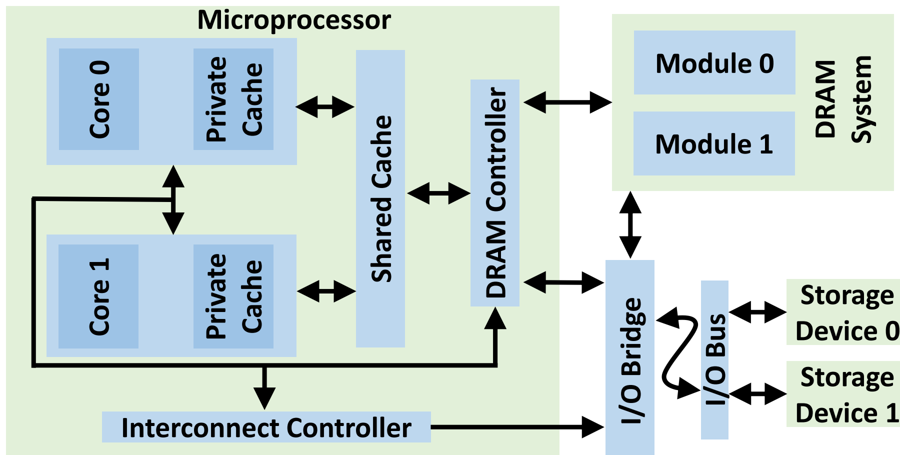


Figure 2.2: Memory organization in modern computer system.

In general, most of the memory components are analogous to a 2-D array structure [vdGS02]. Selecting the appropriate row and column provides one-bit data. Usually, memory chips contain multiple memory arrays to produce a multi-bit data

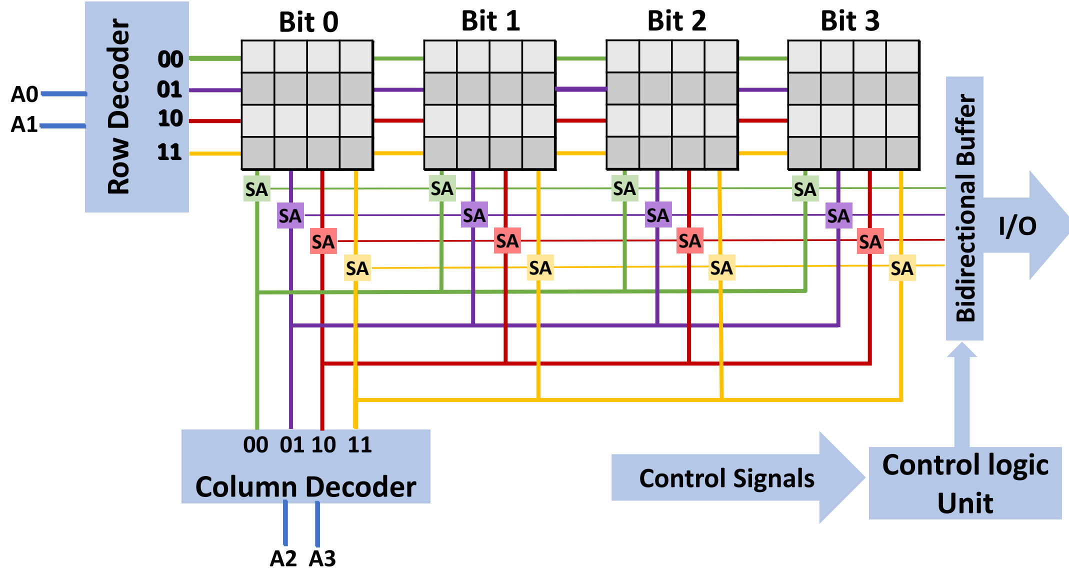


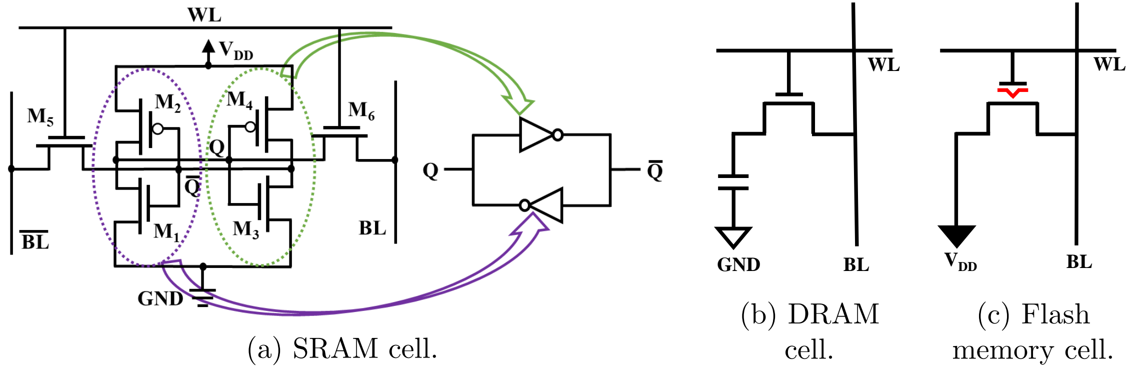
Figure 2.3: A $4 \times 4 \times 4$ memory array.

word. In Figure 2.3, we present a $4 \times 4 \times 4$ memory[2] chip with a 4-bit address (two bits each for row addressing and column addressing) [vdGS02, JWN10, GWHS19]. The appropriate row and column addresses are selected via a row address decoder and a column address decoder. Note that rows and columns are also called wordline and bitline, respectively. Sometimes, memory cells connected in a single row are called a page. Each bitline on a memory array is connected with a series of sense amplifiers (SA). A series of sense amplifiers is also known as the row-buffer. During a read operation, the bitlines are precharged to a certain voltage level. Once the appropriate row is selected, the bitline voltage is interrupted. The sense amplifier senses this voltage perturbation and latches the corresponding data. Then the appropriate selection of the columns drives the sense amplifier data to the output. On the other hand, the sense amplifier acts as a buffer to hold the write data during the write operation. Once the target memory cell is activated via appropriate address selection, the memory cell latches the data from the row-buffer. Depending

[2]memory size is expressed as $r \times c \times w$, where, r = number of rows, r = number of columns, and w = word size in bits.

on memory type, each memory cheap may have different memory cell structures. In following subsequent paragraph we discuss necessary details of different memory cell structure associated with the memory hierarchy-

1. **Registers:** In multi-core microprocessors, each core has its own set of private registers (not shown in fig. 2.2) and can be directly accessed. Registers are very fast, but only a small amount of data can be stored. Usually, modern CPUs' registers are implemented with specially designed high speed SRAM cells.
2. **Cache memory:** Cache memory is made of SRAM and usually divided into multiple levels; some of the levels are private to the individual CPU core, and some can be shared among the cores [JWN10]. The size of the cache memory can be ranged between a few hundred bytes to a few Mbytes. Although many different structures of SRAM cell exists [RSS17, AKJ+13, CMK+07, CMN+08], the 6-transistor (6T) SRAM cell is the most popular form among them (shown in Figure 2.4a). A 6T SRAM cell is made of two coupled inverters along with two access transistors. The access transistors are controlled by wordline (WL) and connect the memory cell to the bitleines (BL and $\overline{\text{BL}}$). The BL and $\overline{\text{BL}}$ provide a complementary logic pair during the read (or write) operation. After the write operation, the memory cell maintains the desired logic by creating a logic-loop between the coupled inverter-pair.
3. **Main memory:** In most of the computer system, DRAM (dynamic random-access memory) serves as the main memory. DRAMs are cheaper and larger (a couple of hundred Mbytes to a couple of TBytes) than SRAMs. Although the smaller DRAM cell structure reduces silicon die cost, the DRAM operations are more complex, slower, and energy-consuming than the SRAM. A DRAM cell consists of two components- an access transistor and a capacitor to hold the charge (Figure 2.4b). The access transistor connects the capacitor with a bitline and



(a) SRAM cell.

(b) DRAM cell.

(c) Flash memory cell.

Figure 2.4: Basic memory cell structure.

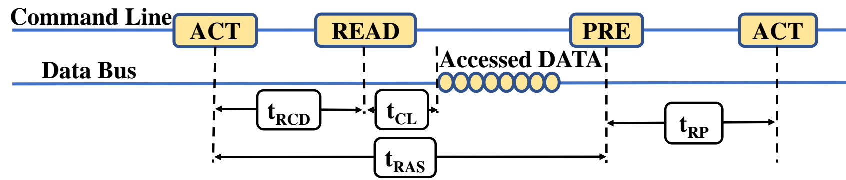


Figure 2.5: DRAM timing at read cycle.

is controlled by the wordline. The state of charge in the capacitor determines the memory content (i.e., "1" or "0"). Depending on the DRAM design, DRAM memory cells can be categorized into *true-cell* and *anti-cell* [LJK+13]. A *true-cell* stores logic '1' with a fully charged capacitor and "0" with an empty capacitor. On the other hand, an *anti-cell* stores "0" with a fully charged capacitor and "1" with an empty state. Nonetheless, the stored charge in the capacitor leaks away, which leads to an incorrect reading after a certain amount of time. So, to ensure the integrity, the content of a DRAM cell needs to be refreshed periodically before the memory contents flip. This time interval is known as the *Retention* time, which is 32 or 64 milliseconds (ms) [JED12]. A failure to refresh before the retention time can alter the memory content. To write/access the cell content, the column (or bitline) voltage need to be precharged at a specific reference voltage ($V_{ref} = \frac{V_{DD}}{2}$). Figure 2.5 presents a simplified version of DRAM read operation [CKH+16]. A read operation starts with an ACTIVATE (ACT) command ex-

ecuted by the memory controller. t_{RCD} is called the *Activation* latency which is required to activate (turn *ON*) the access transistor properly. The activated access transistor creates a conducting path between the storage capacitor and the bitline. The charge stored in the capacitor perturbs the bitline voltage. The sense-amplifier senses the perturbed voltage and amplifies it to an appropriate binary value. At this moment, the memory controller applies the READ command to read the data and fetch it to the data bus. The minimum time latency between the READ command and the first data bit to appear in data-bus is called *Column Access Strobe* latency or *CAS* latency (t_{CL}). DRAM's read operation is a destructive process. Therefore, the charge on the DRAM storage capacitor needs to be restored after each successful reading. The time required to activate an access transistor and to restore the charge on the corresponding storage capacitor is known as the *Row Active* latency or *Restoration* latency (t_{RAS}). At the end of the restoration process, the memory controller again applies the *PRE* command to re-initiate all the bitlines for the next read/write operation. The *PRE* command precharges all bitlines to V_{ref}. The time required to precharge all bitlines properly is called *Precharge* time (t_{RP}). The *PRE* command also deactivates all previously activated access transistor. The $t_{RAS} + t_{RP}$ is the total time required to read a DRAM row properly; this total time is called *Row Cycle* Time (t_{RC}).

4. **Storage memory:** Storage memory is a non-volatile memory (e.g., flash memory, magnetic hard drive) which is used to store and recall information that is previously used. The storage memories are very cheap and large in volume. However, they are extremely slow, energy consuming, and their endurance (i.e., number of write cycles) is very limited. Usually, storage memories are integrated with a controller that evenly distributes the writing operation on all memory

blocks (wear-leveling [Cha07]) to maximize the service life of the storage memory. Although many forms of storage memory devices exist, flash memories have become very popular in past decade as they are faster and more reliable than the traditional hard disk drive. A basic structure of a flash memory cell is shown in Figure 2.4c [KBTS+18]. The flash memory cells consist of only one specially designed access transistor. Unlike a regular transistor, this transistor has a floating gate (red marked in Figure 2.4c) underneath the main control gate. When a high voltage is applied to the control gate, electrons tunnels through the oxide layer and accumulate on the floating gate. This process results in the transistor being non-conductive while the regular voltage is applied to the control gate. A charged floating gate represents logic "0" (program state), whereas the non-charge state represents logic "1" (erase state) in the Flash memory cell. A large negative voltage is applied to the control gate to erase the data (i.e., converting "0" to "1") from the flash memory cell. The negative voltage on the control gate removes the electrons from the floating gate. During the read operation, the active wordline causes the bitline to be shorted with V_{DD} if logic "1" is stored (consequently, sense amplifier senses the logic "1"). Otherwise, the bitline retains its initial voltage, and the sense amplifier latches logic "0".

2.3.2 Basic Principle of Memory-based PUFs (mPUFs)

Almost all types of memory chips can be performed as a standalone PUF. In this section we discuss the basic principle of few popular mPUFs using different types of memory technologies.

SRAM PUFs: Among different mPUFs, SRAM PUF is the most popular type and commercially available in many commodity devices. The SRAM PUF takes

advantage of the slight asymmetricity of the coupled inverters, which is arisen from the random variation of the manufacturing process. Ideally, the cross-coupled inverters are symmetrically laid out to maximize the static noise margin (SNM) [G$^+$18, CDHS12]. SNM is defined as the maximum allowable noise that can tolerate an SRAM cell without flipping its value [MMR10]. However, the inevitable random dopant fluctuation (RDF) effect leads to threshold voltage variation and introduces asymmetricity between SRAM inverters [KCL$^+$12]. Therefore, during power-up, these two inverters race each other and settle to "1" or "0" [G$^+$18, CDHS12]. In SRAM PUF, this random power-up (or start-up) is used as the PUF signature. Although SRAM PUFs are very cheap and robust compared to other mPUFs, the disadvantage of SRAM PUF is, it requires an entirely new power cycle for evaluation.

DRAM PUFs (DPUFs): Recently, DPUFs have also gained a lot of attention as modern computers are equipped with large DRAM systems. Due to the large capacity, DRAM PUFs can produce a very large set of CRPs and almost provide a similar capability of strong PUFs. Researchers have been proposed multiple approaches to produce DRAM-based signatures-

1. **Retention-based DPUFs:** Signatures are generated by disabling the refresh interval for a certain and sufficient amount of time [SRR16, KPHM18]. The DRAM cells are leaky, and therefore, the DRAM contents need to be refreshed periodically, usually 64ms or 32ms according to the JEDEC specification [JED12], to ensure the data integrity [SRR16]. Failing to refresh periodically within this time interval introduces errors due to the leaky property of DRAM cells. The error pattern generated from the retention failure is unique from chip to chip and is used to generate device signatures [SRR16, KPHM18]. Although, Retention-based DRAM PUFs are robust, they are significantly slow compared to other DPUFs and take in order of a minute to evaluate.

2. **Latency-based DPUFs:** The reduction in t_{RCD} introduces erroneous read/write operation (see Section 2.3.1), which can be used to generate device signatures [KPHM18]. This latency-based PUF generates signature at a much faster rate [KPHM18]. The reported result shows that the mean evaluation time is ~88.2ms (outperforms all previously proposed retention-based DPUF [SRR16, XSA+16, KGKF14]). However, it still requires multiple row cycles to evaluate the PUF response. This latency-based DPUF also needs a filtering mechanism in each access that adds both hardware and computational overheads. To overcome this problem, we have proposed a novel technique to evaluate DPUF by reducing the precharge time (t_{RP}) [BTRFR19]. Reducing t_{RP} also causes erroneous output. Our experimental result showed that a subset of DRAM cells (which can be selected in the early characterization phase) generates a reproducible signature with reduced t_{RP}. Our proposed technique is much straight-forward and faster than previous latency-based DPUF (i.e., by reducing t_{RCD}).
3. **Start-up-based DPUFs:** In start-up based DPUF [TKYC16], the device signature is generated from the start-up states of DRAM cells. Initially, the bitlines are charged to $\frac{V_{DD}}{2}$. But the process variations on the storage capacitor and bitline capacitance slightly deviate the *bitline* voltage to $\frac{V_{DD}}{2}+\delta$ or $\frac{V_{DD}}{2}-\delta$, where δ represents a small voltage. The *sense-amplifier* senses the voltage difference and latches "1" or "0". Consequently, upon power-up, the DRAM cells generate "1"s and "0"s randomly. Similar to SRAM PUF, start-up-based DPUFs also require a new power cycle. Unfortunately, adding a new DRAM power cycle is expensive, as it requires storing the whole DRAM data to a secure memory location at each cycle. Hence, in practice, start-up-based DPUFs have very limited use.
4. **Rowhammer DPUFs:** The errors caused by the rowhammer disturbance are used to generate device signatures [SXA+17, AAF+18]. Rowhammer is a pro-

cess where a DRAM cell is repeatedly accessed. Such repeated operation causes partial activation to its adjacent rows (due to the coupling capacitance) and may cause data corruption at adjacent rows. The average evaluation time of a rowhammer PUF is in order of minutes and has an almost similar performance of Retention-based DPUFs. However, all DRAMs are not vulnerable to rowhammer [SXA+17]; hence, it is not a universal method to generate DRAM signature.

Flash PUFs: Recently, many researchers have proposed efficient implementation using flash memory chips. The most popular form of flash memory PUFs relies on three different techniques- **(i)** partial programming, **(ii)** partial erasing, and **(iii)** program disturbance [SMRR20, JXW+15]. Due to the random process variation, the program time and erase time vary from cell to cell. In partial programming and partial erasing methods, the program or erase voltage is applied on the control gate (Figure 2.4c) for a shorter time. However, the time period is selected in such a way so that it causes random bit failure in the flash chip. This erroneous output of the flash chip can be used as the device signature. In the program disturbance method, random bit failures are induced by repeatedly programming adjacent cells. It has been observed that if a memory cell is repeatedly programmed with high voltage, the adjacent cell also might get programmed (i.e., "1" to "0") due to the parasitic coupling (or parasitic capacitance) between these two cells. As the random process variation affects the parasitic capacitance, the bit flipping due to the program disturbance also creates a random pattern, which can be essentially used as the PUF signature. Although the flash chips are capable of producing a very large set of CRPs, the flash chips are much slower. Furthermore, flash memories have much smaller endurance (wears out with a small number of program-erase cycles); hence, using flash memory chips as a PUF causes the device's early failure.

2.4 mPUF Applications

mPUFs can potentially be used in two different applications- (**i**) counterfeit IC detection on semiconductor supply chain and (**ii**) as a secure key generator in different cryptographic applications [MBW+19, HYKD14].

2.4.1 Anti-counterfeiting Solution

Semiconductor Supply Chain Vulnerability: In recent times, IC counterfeiting has become a global problem due to the globalization of the semiconductor supply chain. In this model, a chip is designed in one place while fabricated in a different place and involves several parties to reduce the fabrication cost and time to market. Because of traveling IPs in different formats and involvement of untrusted parties, the modern semiconductor supply chain suffers from counterfeiting (such as hardware trojan or malicious change in third-party IP or chip layout, cloning IPs/ICs, remarking, etc.) [FC18, GHD+14, BTMR+20, BB15, RFS+14, CRT13]. Figure 2.6 shows the IC design flow for authentic-chip and pirated-chip production cycle. The untrusted party (the third-party IP developer, the foundry, the assembly, the distributor, etc.) can perform counterfeiting at different phases of the manufacturing process. An untrusted party can send out overproduced, and out-of-spec/defective ICs to the market. The untrusted party also can clone the original chip by stealing IPs or by reverse-engineering (from a post-fabricated product) to avoid research and development (R&D) costs. Table 2.1 defines each type of counterfeiting more elaborately.

To avoid/detect counterfeit chips, researchers proposed many countermeasures, such as SST, hardware metering, blockchain-based traceability, split manufacturing, IC camouflaging, Electronic Chip ID (ECID), On-chip sensor, DNA marking, etc.,

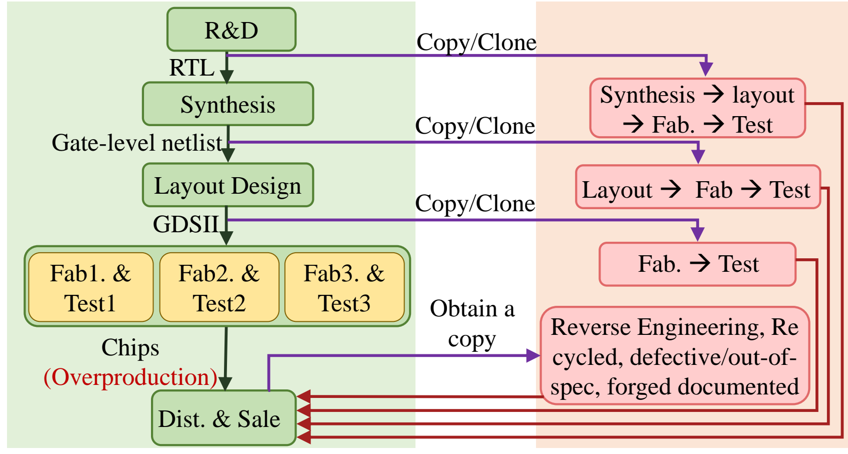


Figure 2.6: IC design flow vs possible covert channels to introduce counterfeit ICs.

Counterfeit Type	Definition [FC18, GHD+14]
Recycled	Recycling chips from old PCB and selling them as new. In a more sophisticated recycling process, the plastic/ceramic encapsulation of the chip die is removed and repackaged to make its appearance new. Recycled chips shares >80% of the counterfeit market.
Remarked	Inferior quality chips are marked as the superior one.
Forged documented	Faking the chip documentation (e.g., faking safety and security certification).
Reverse Engineered	Recover the functional netlist from the chip by an electro-chemical process. Counterfeiter may use this netlist to avoid the R&D cost.
Cloned	A untrusted party (e.g., physical design house) can copy the chip design (netlist, GDSII); later, they can produce unauthorized chips.
Overproduced	Untrusted fabrication facility can produce and market chips outside of the contract, i.e., without authorization of the IP (intellectual properties) owner.
Tampered	Tampering the original chip design. For example, untrusted physical design house can insert hardware trojan in the netlist and create a security backdoor.
Out-of-spec/defective	Selling chips that are failed in the functionality test.

Table 2.1: Different types of IC counterfeitings.

might be used to prevent counterfeiters [GHD+14, FC18, RFS+14, RSK13, KQP01, AEM18, GFT16, ZT14, HHT15, HCM13, AK07, ZTT12, HM11, EBM12, IPK18, RSSK13, WGR+17, ZWRT18]; however, these techniques suffer from different drawbacks. For example, SST and hardware metering techniques provide control over post-fabrication, but it requires a change in traditional fabrication flow. Furthermore, this technique requires exhaustive communication between the foundry and the manufacturer. On the other hand, ECID tags each chip with a unique ID by adding a one-time programmable (OTP) memory. Nevertheless, this method is not suitable for all kinds of chips. For example, the overhead of adding an extra OTP memory component will be very high for an small microcontroller. With an on-chip sensor, each chip is equipped with an additional hardware component, which modifies its properties due to aging. These properties can be used to detect recycled chips. However, on-chip sensor-based countermeasures need additional hardware overhead and are not feasible for inexpensive systems. In DNA marking, each memory component is marked with a unique DNA sequence. DNA marking suffers from impracticality as it requires a complex authentication scheme. Other techniques, such as blockchain-based traceability, split manufacturing, IC camouflaging, etc., require modified fabrication flow or design techniques that are not suitable for low-cost memory chips.

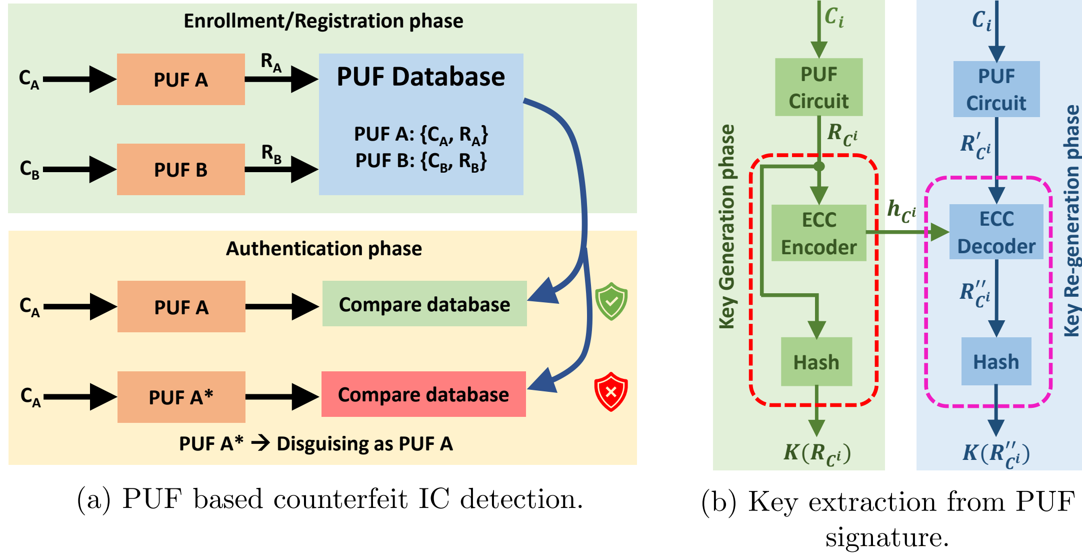
Physical inspection-based schemes [GHD+14, WKP14, ABN+16, H+16], such as X-Ray imaging and scanning electron, can detect counterfeit/recycled chips. However, these techniques require expensive equipment and not viable for low-cost chips. Moreover, expensive equipment and complex authentication schemes are also not suitable for general users who want to verify their purchased products' authenticity.

PUF-based Anti-counterfeiting Solutions: PUF based anti-counterfeiting solution is quite straightforward; the simplified PUF-based anti-counterfeit solution

is shown in Figure 2.7a [MBW+19]. This authentication technique is a two-step process. The first step is known as the enrollment or registration process, where the manufacturer creates a database by recording a pair of CRP from each device. This step must be completed before releasing devices into the market, and the databases must be maintained by the manufacturer or by a trusted third party. In the authentication phase, the device sends an authentication request to the manufacturer; in reply, the manufacturer sends the same challenge (as of the enrollment phase) to the manufacturer. Now, the device generates the PUF signature corresponds to the challenge and sends it to the manufacturer. The manufacturer verifies the response by comparing the response with the database. For an authentic device the response should be matched with the database within a small threshold.

PUFs can directly identify overproduced, remarked, cloned, reverse engineered IC [GHD+14, GFT13]. Additionally, with a typical usage pattern, bit error in mPUF signature usually increases with device age (see Section 3.6 for more details). Hence, recycled IC can be easily identified by analyzing the degree of signature mismatch with the manufacturer's database [GMAS+17]. Moreover, the PUF-based key can also identify the device's integrity as a root-of-trust (discussed in Section 2.4.2) and potentially identify tampered devices [vH18, ZZH+14]. Although both traditional strong PUFs and mPUFs can be used to identify counterfeit chips, the advantage of mPUF is that it can be readily used for any device which contains a memory chip. Although PUF based technique is simple, it requires maintaining a large database which is expensive. Besides, it adds an extra step in the supply chain and increases market lead time. Furthermore, as described in Figure 2.7a, many small devices are not configured to maintain a network protocol; hence, they cannot be authenticated by this protocol.

Table 2.2 summarizes different types of available anti-counterfeit solutions, their applicability, relative complexity and cost.



(a) PUF based counterfeit IC detection.

(b) Key extraction from PUF signature.

Figure 2.7: PUF applications.

2.4.2 Cryptrographic Key Extraction from PUF Signature

mPUF are capable of generating cryptrographic key on fly. Almost all PUFs follow a similar key extraction scheme as shown in Figure 2.7b [CSP15, SD07, CZZ17]. PUF-based key extraction consists of two phases. In the key enrollment/generation phase, the PUF device generates a response R_{C^i} depending on a given challenge C_i. Then, an ECC (Error Correction Code) encoder computes the corresponding error correction code/helper data h_{C^i} and generates the key, $K(R_{C^i})$ (e.g., using hash). Although the final key is not stored, the C_i and corresponding helper data h_{C^i} is stored in a storage device. In the key re-generation phase, the PUF circuits again receive the same challenge C_i, and generates the corresponding response R'_{C^i}. Due to the different internal and external noises (variations in operating condition, device aging, etc.), some errors are always expected in the generated response R'_{C^i}

Category	Methods		Recycled	Remarked	Forged Documented	Reverse Engineered	Cloned	Overproduced	Tampered	Out-of-spec/ Defective	Require design modification?	Complexity$^{\intercal}$	Cost$^{\nabla}$	Memory chips	Non-memory chips
			Types of counterfeiting											Applicability	
Counterfeit-resilient chip manufacturing	SST [RFS+14]		No	No	No	No	Yes	Yes	No	Yes	Yes	High	High	Yes	Yes
	On-chip sensor [HHT15, ZT14]		Yes	Yes	No	No	No	No	No	No	Yes	Low	High	Yes	Yes
	Hardware metering [KQP01, AK07]		No	No	No	No	Yes	Yes	No	No	Yes	Medium	High	Yes	Yes
	Chip tagging [EBM12]		No	Yes	No	Yes	Yes	Yes	No	No	Yes*	Low-High	Low-High	Yes	Yes
	Others [IPK18] [RSK13] [RSSK13]	Blockchain	No	No	Yes	Yes	Maybe	No	No	No	Yes	High	Low	Yes	Yes
		Split-manufacturing	No	No	No	No	Yes	Yes	No	No	No^2	High‡	Medium	Yes	Yes
		IC camouflaging	No	No	No	Yes	No	No	No	No	Yes	Medium	High	Maybe	Yes
Post-fabrication chip characterization	Physical Inspection-based test [GHD+14]		Maybe	Maybe	No	No	No	No	No	No	No	Medium	High	Yes	Yes
	Electrical test [GHD+14]		Maybe	Maybe	No	No	No	No	Maybe	Maybe	No	Medium	High	Yes	Yes
	burn-in test [GHD+14]		Maybe	Maybe	No	No	No	No	No	Maybe	No	Medium	High	Yes	Yes
	PUF-based authentication [GFT13]		Yes	Yes	No	Yes	Yes	Yes	Maybe	Maybe	No	Medium	High	Yes	Yes§

Table 2.2: Summary of counterfeit chip detection/identification methods.

$^{\intercal}$Combined complexity of supply chain, communication protocol, and IC fabrication flow.
$^{\nabla}$Combined cost of extra computation, hardware, and infrastructure (e.g., server).
*Might Require special packaging. ‡Requires complex supply management. §PUF circuits are required to be integrated.

(compared to R_{C^i}). However, corrected response R''_{C^i} is computed using the helper data h_{C^i}. Subsequently, the key, $K(R''_{C^i})$, is generated from the R''_{C^i}. If the PUF circuit is not altered in between these two phases, the $K(R_{C^i})$ and $K(R''_{C^i})$ should be the same.

Fuzzy Extractor: The Fuzzy extractor is used to enhance the security of the PUF-based key [FRS20, FRS16, CFP+16, Boy04, KHK+14]. The security enhancement is performed by (**i**) removing noise without leaking significant information and (**ii**) extracting the entropy (i.e., increase the entropy density on fuzzy extractor's output) [FRS20]. Theoretically, the fuzzy extractor combines an ECC encoder/decoder (secure sketch) and a hash function [DGV+16]. However, instead of using a fixed-sized hash output, the length of the output depends on the entropy of the fuzzy extractor's input. If the input has smaller entropy, it reduces the output size and increases the entropy density of the key.

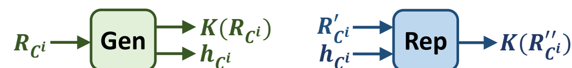


Figure 2.8: `Rep` and `Gen` block of fuzzy extractor.

The fuzzy extractor has two major blocks: (i) a generate algorithm (`Gen`) in the key generation phase and (ii) a reproduce algorithm (`Rep`) in the re-generation phase (Figure 2.8). The `Gen` and `Rep` replace the components from the red and magenta boxes of Figure 2.7b, respectively. The `Gen` takes the raw PUF response R_{C^i} and produces the output key $K(R_{C^i})$ and h_{C^i}. In the re-generation phase, the `Rep` takes the noisy input R'_{C^i} and h_{C^i} and produces the corrected key $K(R''_{C^i})$. The noisy behavior of PUFs requires to allow some errors, which reduce the entropy of the input PUF key (R). A well-designed fuzzy extractor can avoid this problem by adding a penalty on output key length. Mathematically, it can be shown that the output key is secured as long as the length of the output key does not exceed

the fuzzy min-entropy [FRS20, FRS16]. The fuzzy min-entropy (H_∞^{fuzzy}) of a fuzzy extractor is defined as follows-

$$H_\infty^{fuzzy}(\mathcal{R}) = -\log \max_{R'}\{Pr[\mathcal{R} \in B_q(R')]\} \tag{2.4}$$

Here, $\mathcal{R}$ is the distribution of the original response R and q is the number of allowed erroneous bits. To maximize the chance; the attacker would select R' that maximize the total probability mass of all responses (drawn from distribution $\mathcal{R}$) within the ball $B_q(R')$ of radius q around R' [FRS20, FRS16]. The main difficulty of designing the fuzzy extractor is to estimate $Pr[\mathcal{R} \in B_q(R')]$. For a reasonable estimation of the probability distribution, one might require a large number of sample sources. Moreover, an attacker can always make a better probabilistic estimation with a larger set of samples [FRS20, FRS16].

Application of mPUF Key: There are two possible scenarios of mPUF applications. In the first type of application, mPUF is used inside the device and is not accessible from outside. Example of such application is software IP protection, securing key vault, and establishing the root of trust [MBW+19, HYKD14, ZZH+14]. For software IP protection and securing key vault, the PUF key is used to encrypt/decrypt the storage device while the storage device is being used to store the software IP or the users' key. As the PUFs generate the key on the fly, an attacker cannot clone the encrypted storage memory and use the content in another device. Similarly, the mPUF key can also be used to check the software and hardware integrity of the device at the boot-up stage (i.e., establishing the root of trust or a trusted execution environment) [ZZH+14, vH18]. In the most simplest technique, the security module of the computing system (e.g., trusted platform module) hashes the software and hardware information by salting with the PUF key. Then, the security module compares this hash value with the hash value of the previous boot

and identifies if there is any unauthorized hardware or software tampering between these two boots.

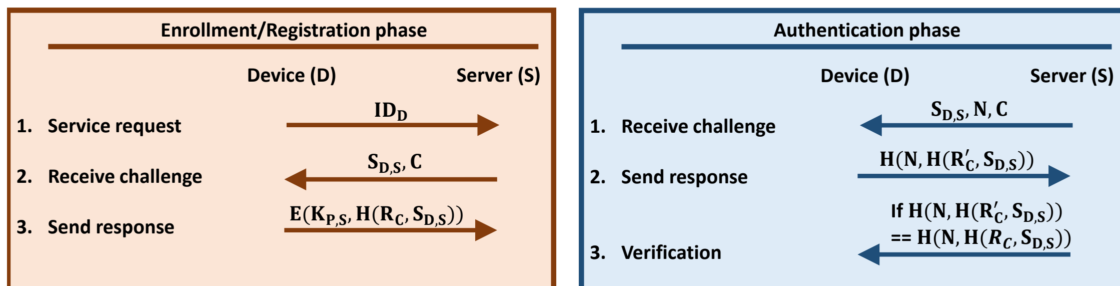


Figure 2.9: Simplified mPUF-based IOT device authentication protocol.

In the second type of mPUF application, mPUF key is used for device authentication to access cloud services. Traditionally, strong PUF is preferred for this application (i.e., using one CRP only once and discard it after usages) [HYKD14]. However, with a little modification in communication protocol (Figure 2.9), the mPUF-based key can also be used for the same purpose [YGH18, JCJ+21]. During the service enrollment/registration phase, the IOT device sends a service request to the server by sending its device ID (ID_D). Next, the server generates a unique value $S_{D,S}$ and a challenge C. In reply, the device generates a hash value ($H(R_C, S_{D,S})$) using the PUF response R_C and $S_{D,S}$. Then, the device encrypts the $H(R_C, S_{D,S})$ using the public key of the server and sends the encrypted data to the server. The server stores $\{ID_D, S_{D,S}, C, H(R_C, S_{D,S})\}$ to its database. In the authentication phase, the server sends $S_{D,S}$, C, and a nonce N to the device. In reply, the device generates the PUF response R'_C and creates a hash value $H(N, H(R'_C, S_{D,S}))$. If the R'_C and the PUF response during the enrollment (R_C) matches, the $H(N, H(R'_C, S_{D,S}))$ and the computed hash value using the server database ($H(N, H(R_C, S_{D,S}))$) should be same, and consequently, the server can identify the unauthorized service request.

Known Attacks on mPUF Keys: There have been several techniques to predict/model PUF responses, although PUFs are widely accepted as highly secure

because of the unclonable and immutable nature of manufacturer process variations [vDR14, RvD13, RJA12, KS10, MSSS11]. These attacks can be categorized into (**i**) *protocol attack* and (**ii**) *silicon attack*. In the *protocol attack*, the attackers usually exploit existing design/protocol flaws in the PUF network protocol such as gaining temporary access on a PUF, re-using the PUF responses from previous sessions, creating a covert channel to perform malicious activity on PUF, leaking information via error correction scheme, etc. [vDR14, RvD13]. To defend against such attacks, we need to fix the network protocol weaknesses and vulnerabilities as soon as they are exposed.

On the other hand, in the *silicon attack*, an attacker tries to (**i**) imitate the characteristics of the PUF device and generate the correct responses, (**ii**) forge/clone all possible CRPs, and (**iii**) modify the PUF responses. For the first case, the attacker needs to derive some deterministic relationship (e.g., training a machine learning model) between the challenges and responses. Once the relationship is established, the attacker can produce the correct response for a given challenge. However, this attack's success depends on the availability of a large number of CRPs to an attacker [MGH+18], which is possible only for strong PUFs due to their open-access interface [MBW+19, RSS+10]. Thus, an attacker usually uses second and third attack strategies to attack a weak PUFs. Usually, cloning or modifying PUF output requires physical access to the device, which makes those attacks impractical to mount in a protected environment [AAR+18, OSW13, NSHB13, HBNS13].

Anagnostopoulos *et al.* [AAR+18] successfully forged the SRAM-based PUF fingerprint by leveraging SRAM memory cells' data remanence property at low temperatures. In this attack, the attacker writes a known bit-stream 'X' on the SRAM device and change the operating temperature (-110°C to -40°C) before shutting it down. At low temperatures, SRAM cells are capable of retaining the data for more

than 10ms after turning off the SRAM power (due to the data remanence effect). In such circumstances, when the device is booted, SRAM-PUF creates a fingerprint that is very similar to the known bit-stream 'X' (up to 93% similarity). The *remanence attack* has been further improved to clone PUF CRPs by injecting some faults on the PUF [OSW13]. In this method, the attacker uses the *remanence decay* effect to introduce a sequence of random faults on the PUF-based cryptographic key and observes corresponding outcomes of a particular operation (e.g., ciphering a random text). Finally, the attacker analyzes these outcomes and derive the PUF outputs. However, this attack is still impractical as it needs precise control over the operating condition (i.e., operating temperature and voltage).

In the third approach (i.e., modification of PUF output), an attacker alters the PUF response by exposing the PUF to high-voltage and high-temperature [RS16]. In such adverse operating conditions, the SRAM-PUF output is altered due to the bias temperature instability (BTI) [RS16]. This attack is mainly used to perform a denial-of-service attack (e.g., prevention of authentication).

Helfmeier *et al.* [HBNS13] demonstrated that an SRAM-PUF could be cloned to a new SRAM chip by editing the microstructure (e.g., transistor size) with a focused ion beam (FIB). In this technique, the attacker first learns the SRAM-PUF response through physical access and then modify the microstructure of a new SRAM chip to get the similar response of the victim PUF.

Notably, all of these *silicon attacks* against mPUFs assume that the attacker can gain physical access to the victim's (i.e., target mPUF) device at least once or has precise control over the operating condition, which is not practical for many PUF-based security applications. Furthermore, the physical access in some attacks might leave traces on the victim's device, and it is easier to identify if there is any tampering on the device.

CHAPTER 3

IMPACTS OF DETERMINISTIC FACTORS ON mPUF SIGNATURES

3.1 Introduction

In this chapter, we briefly discuss different factors that can create a deterministic impact on mPUFs. Although in earlier studies, the random variation of the manufacturing process was researched exhaustively, other non-random factors such as architectural variation, layout variation, process variation, IC packaging, and aging effect remain unexplored. Nevertheless, these factors are important as they potentially produce a unique correlation among the mPUF signatures if they share same manufacturing resources (i.e., same manufacturer, specification-set, fabrication facility, etc.) [BTMR+20, TFR21]. In the next couple of chapters, we show that such correlations can be anticipated using a unique set of visual descriptors and consequently can identify the memory manufacturer and part number. Furthermore, the correlation information can also be used as side-channel information and can be used to device a heuristic attack (detailed discussed in Chapter 6). These non-random variations are ubiquitous due to the universal array-like structure of memory components (as shown in Figure 2.3). However, we discuss those variations mainly from the SRAM perspective as SRAM chips are more common memory components than any other memory type and are dominantly used as a PUF. Furthermore, due to the rapid growth DRAM market [KBV], they become vulnerable to counterfeiting; hence, we also discuss those variations briefly from the DRAM perspective.

3.2 Architectural Variation

The similarity in architecture and layout of memory chips are among the main reasons for correlated signature properties between two mPUFs with the same model number (i.e., part number). All semiconductor memory chips share a very similar row-column architecture (See Section 2.3.1). However, the address decoder circuitry for logical to physical address mappings, control circuitry to determine the I/O state (read/write), optimal array structure to the trade-off between cost and performance, power-saving techniques, the structure of redundant blocks (rows or column) to replace corrupted memory arrays entirely or partially, etc. vary from one manufacturer to another or from one model to another. Moreover, manufacturers might also shrink the die size to reduce the cost per cell, causing the memory chips to be more susceptible to noise and less robust. For example, the manufacturer may follow different SRAM cell structures (e.g., 4T, 5T, 6T, 7T, 8T, 9T) for further optimization [PSK15, RSS17]. Some of those architectures are asymmetric and are more likely to produce biased start-up signature (see Section 2.3.2). Although symmetric SRAM architectures are preferred for PUF applications [BH14], a symmetric SRAM cell may still produce deterministic biased outputs because of asymmetric fabrication technology (e.g., asymmetric silicon doping [MGP+11]), asymmetric driving strength of bitline pair [KYT+08], etc. Although such small asymmetricity might not impact the regular operation of SRAMs, that might still be visible from the SRAM start-up data statistics (by slightly favoring logic “0” or “1”).

Similarly, for DRAM chips, manufacturers may follow different array structures to trade-off between cost and performance [LKS+17]. For example, DRAM cell structure may vary from one architecture to another (trench capacitor vs. stacked capacitor) and may impact the overall speed and performance (noise immunity vs.

cost) [JWN10]. As DRAM PUFs mostly rely on manipulating retention/latency parameters (See Section 2.3.2), variations on speed and noise immunity leave traces on DRAM signature variation. Similar to SRAM and DRAM, manufacturers may also follow different architectures for flash memory chips [KCHC08]. For example, flash memory may have a NAND-like structure for better density vs. a NOR-like structure for random access capability, better reliability, and higher read speed. Also, flash memory cells can be configured as SLC (single level cell) or MLC (multi-level cells). An SLC cell only stores one bit per cell and gives better reliability and endurance, where the MLC cell can store multiple bits in a single cell to increase density and reduce cost. Nevertheless, such differences in flash memory may impact flash-based PUF.

3.3 Layout Variation

Chip layout variation from one manufacturer to another may originate from several sources such as chip area, floorplanning, placement, and routing, etc. This layout variation may affect different electrical characteristics such as RC path delay, power utilization (I^2R loss), noise margin, etc., which can be reflected in the memory latency parameters. For example, Apostolidis *et al.* [ABK16] reported six different layout designs for symmetric 6T SRAM structure, and each of them has different pros and cons. For example, they have different power utilization, delay, and noise characteristics. In addition to this, some implementing and resource constrain may introduce some asymmetric nature in memory cells, leading to slight bias to a specific logic at device start-up. For example, using multiple metal layers may introduce unmatched wiring between the inverter pair. Moreover, the difference in CAD tools' configurations may also introduce variations in memory layout.

Similar to SRAM, DRAM layout variation can also impact its PUF performance. For example, while the open bitline layout are more scalable and cost-efficient, the folded bitline layout are more noise immune and provide better speed [JWN10].

3.4 Process Variation

Process variation (PV) is an intrinsic phenomenon during the fabrication process that introduces the variability on transistors' attributes, interconnecting metal, and dielectric layers. The PV can be divided into two categories- global PV and local PV [Liu07]. However, the PV can be broken down into finer grains, as shown in Fig. 3.1 [Liu07, ZG14, HKX+14, HKCM13, CGS+11]. The systematic variations are deterministic and correlated with the locality of the chip layout, characteristics of the silicon wafer/fabrication facility, node technology, etc. [BPS+17]. In contrast, the random PV is entirely indeterministic. Furthermore, the systematic PV can vary from wafer to wafer (i.e., inter-wafer) and within the wafer (intra-wafer). The inter-wafer PV impacts each silicon wafer differently and may depend on the wafer characteristics. Moreover, the intra-wafer variation describes the variation among dies fabricated in a single wafer. The intra-wafer process variation may be further divided into two groups- inter-die variation and intra-die variation. The inter-die variation affects each dies differently (i.e., multiple dies fabricated in the same wafer). On the other hand, the intra-die variation defines the spatially correlated variation within a single die.

The random process variation is extremely difficult to model, especially to determine the exact mismatch between coupled inverters of SRAMs or exact capacitance value of DRAM cell capacitor. The inter-wafer process variation might also be challenging to model as it depends on the wafer manufacturing process and corre-

sponding intrinsic parameters. However, the intra-wafer variation might have deterministic patterns and might be learned by statistical analysis [Liu07, MJ06, AAT14].

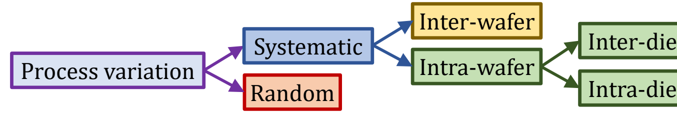


Figure 3.1: Different types of process variation.

Additionally, in some process technology, the fabrication plant may have a specific set of design rules depending on the layout pattern (for example, different design rules for a mirrored layout pattern). The manufacturer might need to break the symmetricity of the layout to facilitate such design rules. Hence, a perfectly symmetric 6T-SRAM cell layout can become asymmetric after fabrication. Recent studies show that systematic process variation can significantly impact the SRAM start-up data (i.e., might get biased to '0' or '1') [GK14].

3.5 IC packaging

Chip die is encapsulated inside a protected "package" to prevent corrosion and physical damage. The difference in IC packaging may also alter some device characteristics. Usually, manufacturers introduce different kinds of packaging to trade-off among cost, noise immunity, and supporting different operating conditions [KMHH15, MKD04].

3.6 Device Aging

Usually, device age alters PUF signature. In this section, we discuss this effect from SRAM, DRAM, and flash memory perspectives.

The SRAM signature (PUF) can be characterized by $PSNM_{noise}$ (PUF SNM noise) [CDHS12, MSM+20]. The $PSNM_{noise}$ measures how easily an SRAM cell can be initiated to logic "0" or "1". A larger value of $PSNM_{noise}$ ensures more robust SRAM signatures. However, the $PSNM_{noise}$ heavily depends on SRAM transistors' threshold voltage [CDHS12]. Two well-known phenomena, negative and positive bias temperature instability (NBTI, PBTI), can cause transistor threshold voltage to shift [GWHS19] by trapping some charge on the gate oxide [DA+18]. NBTI and PBTI are the direct consequence of transistor aging and affects pMOS and nMOS respectively [DA+18]. Hence, the SRAM $PSNM_{noise}$ can be changed over its usages due to the change in its transistors' threshold voltage (see Section 2.3.1).

Depending on SRAM usage data pattern, the change in $PSNM_{noise}$ can affect the SRAM start-up signature: (**i**) a noisy signature bit[1] might get biased to "0" or "1", (**ii**) a weak "0" or "1" might become strong "0" or "1", (**iii**) a stable signature bit can be flipped (stable "0" to stable "1" or stable "1" to stable "0"), and (**iv**) a stable signature bit can become a noisy one. Hence, the change in $PSNM_{noise}$ will affect the overall distribution of logic "0" and "1" on SRAM signature. The first three factors will increase the total number of stable signature bits; whereas, the fourth factor will produce more noisy signature bits. However, the cumulative impact of the first three factors dominates the fourth factor. Hence, the total number of noisy signature bits will reduce with device usage (which does not indicate the PUF will be more robust with aging [RHG+17]). Minimizing the mismatch between two inverters can strengthen the impact of the fourth factor, which is difficult to achieve. The equalization of transistors' threshold voltage requires a calculative usage data pattern during the entire chip lifetime [PKR09].

[1]Noisy signature bit shows inconsistent behaviour; sometimes it produces "0" with new power cycle, other time it produces "1"

In an ideal case, the percentage of 0's or 1's should be identical in a new symmetric SRAM chip. With a typical usage pattern, an SRAM cell experiences more logic "0" bits than the logic "1" bits [WDZ+16]. Such usages pattern creates more stress on "M4" pMOS (Fig. 2.4a). Hence, over time, the threshold voltage difference of "M4" and "M2" PMOS increases due to the NBTI effect and causes the SRAM cell to be biased with "1" at power-up state.

Similar to the SRAMs, DRAM chips also experience the aging effect. Due to BTI, the threshold voltage of the access transistor increases over usage [NRT19] (Figure 2.4b). As the threshold voltage of the DRAM access transistor changes over usage, the leakage current of the transistor decreases. Hence, charge leakage of the storage capacitor attached to the access transistor reduces over time and improves the data retention capability. Consequently, the signature generated by retention-based DRAM PUF changes over time (see Section 2.3.2). Aging on DRAM access transistor also might impact the *Activation* latency PUF as the increment in access transistor's threshold voltage also increases the effective *Activation* latency. Apart from the BTI effect on transistors, several other electrical phenomena [Fie17] such as temperature-dependent dielectric breakdown, hot carrier injection, trap charge also might impact other DRAM attributes (e.g., DRAM startup data [TKYC17], rowhammar characteristics [YL19], etc.), and subsequently, alter other DRAM types of PUF signature over time.

Although DRAM chips usually have a little less endurance than SRAM, the DRAM's aging process is much slower than SRAM's [Eve]. The main reason behind that is that the coupled inverters of SRAM cells always remain stressed as long as the device is turned on; on the other hand, DRAM access transistor only stresses whenever accessed. Furthermore, the access frequency of DRAM is much lower than the SRAM chips as the DRAM modules are located underneath the cache (SRAM)

in the memory hierarchy (see Section 2.3.1), and DRAM is only accessed on the target data is missing from the SRAM cache.

The aging effect has a much higher effect on flash memory [Eve]. Flash memories have much smaller endurance since flash cells are exposed at very high voltage during the program cycle (see Section 2.3.1), and their electrical property breaks down very quickly. In several previous studies showed that the electrical characteristics (e.g., program time, erase time, read latency, etc.) of flash memory changes over usage [KBTS+18, SKBT+18, GCC+09, RNP+16, FSN+09], which inevitably affects the flash PUF accuracy.

CHAPTER 4

IDENTIFYING DRAM ORIGIN

4.1 Introduction

This chapter introduces a novel technique of identifying DRAM manufacturer and part number by extracting a common set of features from the DRAM signature. Due to the recent trend of big data applications, modern computer systems require larger main memory and consequently created huge pressure on the DRAM supply chain [DIG]. However, this situation is favorable for counterfeiting, and hence DRAM chips/modules become one of the major targets of the counterfeiters [Kan]. However, detecting counterfeit DRAMs is very challenging because of their nature and ability to pass the initial testing [BTMR+20, CYG+17].

Although, DRAM manufacturers provide the information of all DRAM timing parameters in a small read-only memory (ROM) which is integrated with the DRAM module [JEDb], it has been demonstrated that a counterfeiter can modify this information (aka serial presence detect or SPD information) or replace the ROM to make a DRAM authentic or superior [Kem]. Hence, extracting SPD information for identifying DRAM manufacturer and the part number is not a reliable process. In this chapter, we propose a technique to identify the DRAM origin (i.e., the origin of the manufacturer and the specification of individual DRAM) to detect and prevent counterfeit DRAM modules [BTMR+20]. The main contributions of this chapter are noted as follows-

- We propose a framework to identify the origin of the DRAM manufacturer by exploiting the facts that the architectural, layout, and manufacturing process

variations are reflected in latency variations. The framework is also capable of verifying specification of individual DRAM module.

- We extract the most appropriate features from the latency-based erroneous patterns in DRAM modules to amplify the variations among manufacturers and specifications.
- We propose a machine learning approach to determine the origin of the DRAM manufacturer based on the extracted features. The same method also separates DRAM modules of different specifications that are from the same manufacturer.
- We validate our proposed framework with commercial off-the-shelf (COTS) DRAM modules from three major manufacturers- Micron, Samsung, and SK Hynix [Sta].
- We validate the robustness of our proposed technique against temperature and voltage variations.

4.2 Objectives and Assumptions

The objective of this work is to design a DRAM anti-counterfeiting solution by identifying the DRAM origin (i.e., the origin of the manufacturer and the specification of individual DRAM) reliably by capturing all variabilities (as discussed in Chapter 3). Our main goal is to propose a anti-counterfeiting framework with following properties-

- **Avoiding upfront cost:** Our work aims to store minimal information (a few statistical parameters) to attest and detect a large group of memory modules/chips. On the other hand, for example, in the PUF-based technique, we need to create a golden/reference dataset by accumulating all CRPs for an individual memory module/chip, which adds a large upfront cost to the PUF-based framework.

- **Uninterrupted traditional supply chain:** Several existing anti-counterfeiting techniques (such as PUF, SST, hardware metering, etc.) require registering each device before releasing them on the market. This registration process is exhaustive and adds an extra step to the supply chain. Consequently, those techniques increase market lead time and manufacturing costs. In this work, we aim to avoid such exhaustive registration processes. Later on in this work, we show that analyzing a small number of samples from a large group of modules is sufficient to reveal that group's detailed statistical information.
- **Avoiding hardware modification:** We avoid any design modification or adding new hardware to make our solution work. Hence, our solution is more practical than many existing techniques such as hardware metering, SST, on-chip sensor-based authentication, ECID, etc. [GHD+14].
- **Non-invasive and low-cost solution:** Our goal is to identify counterfeit DRAMs by means of a non-invasive and non-destructive process without requiring any high-end lab facility. This attribute of an anti-counterfeiting solution is expected so that anyone can verify their own DRAM chips without incurring any additional cost.

The existing techniques on attesting of the foundry rely on simulation or test data from fabrication and packaging facilities [WKP14, ABN+16]. Unfortunately, in most cases, the testing data are not made publicly available and, therefore, a party that does not have access to those test data cannot make the classifier and (or) cannot verify the ratified foundry. In contrast, in our proposed technique, the DRAM chips can be authenticated based on trained classifier provided by the manufacturer or a trusted third party. In the proposed technique, the verifier does not require any prior knowledge of the manufacturing process. The classifier input, a set of features, can be easily evaluated in any low-cost embedded or FPGA-based

system. Our proposed technique of identifying the memory manufacturer is based on the following assumptions-

- **Memory class**: A manufacturer ships memory module with a part number on it, which contains the manufacturer information and chip specification, such as density, speed grade, package, temperature range, bus width, die generation, etc. [SK]. In addition to part number, a manufacturer also provides additional module specification on the module label, such as JEDEC PCB layout version [JEDa], SPD version [JEDb], manufacturing country, manufacturing lot number, etc. Timing parameters of the DRAM module are specified into SPD data [JEDb]. In this work, two DRAM modules are considered as two different classes, if one of the following information is mismatched: i) manufacturer, ii) part number, and iii) PCB layout version/SPD data. A change in one specification can lead to different GDSIIs (related to the die generation, and specification), packaging, PCB layouts, or SPD data. Note that a manufacturer may send a single GDSII (Graphic Design System II) file to different fabrication plants. For such a case, it is assumed that fabrication plants with the same GDSII follow similar design rules to minimize the effect of systematic process variations. We also assume that a manufacturer may produce memories with a different specification but with the same set of fabrication plants or design memories with a slight change in specifications (for example, only change in the die package or PCB version). In such a case, these memories may have two sets of features with subtle variation, which leads to a complex classification problem to identify the memory correctly.
- **Feature definition and extraction**: Manufacturers/trusted third parties are responsible for defining a set of features that define their product best and captures the architectural, layout, and process variation. Prior knowledge of memory architecture might enable them to define a better set of features. The feature

extraction process should be independent and straightforward enough to be extracted on the user's system; hence, it relaxes the requirement of any special tool or environment requirement. In our proposed scheme, the manufacturer or a trusted party is responsible for training the statistical model using this feature-set. While training the statistical model for a particular memory class, we also assume that manufacturers do not have any statistical information from other memory classes (i.e., from the negative class). However, in practice, the manufacturer may collect a few random samples from other classes. Training statistical models with some negative examples might be beneficial, but it is almost impossible to collect all samples from all negative classes because of the diversity of memory chips/modules and the presence of several manufacturers. We also assume that the chips/modules used for learning statistical model are pre-verified and authentic.

- **Classification:** Classifying memory (authentic vs. counterfeit) can be done either by manufacturer or consumer, depending on the application. For example, if the manufacturer is reluctant to release the statistical model publicly, it might ask for the features from memory under test (MUT) to verify the authenticity. On the contrary, to reduce the communication overhead and complexly, the manufacturer may release the statistical model publicly, and the MUT can be verified on the user's system. Regular consumers should be able to verify their purchased DRAM class by only using the statistical model. We also assume that consumers do not have any knowledge of memory architecture and manufacturing processes.

Note that, in this dissertation, "sample", "positive sample", and "negative sample" mean memory module/chip under test, memory module/chip that originally belong to the target class, and memory module/chip that originally do not belong to the classifier target class (i.e., belong to the outlier region), respectively.

4.3 Proposed Method

To extract all possible variabilities, we reduce the DRAM timing latency and obtain the signatures (i.e., the error pattern or fail bit count) that reflect the architectural, layout, and process variations. Below, we present a framework to identify the DRAM class that involves several steps.

Step 1: Data acquisition. The experimental results show that the latency-induced error pattern depends on the data written into the memory, the amount of latency reduction, and the DRAM module [KLM16]. To capture the maximum variations among the memory classes, we write four different sets of data to the memory module: (i) Data set 1: solid data pattern (all 1's), (ii) Data set 2: inverse solid data pattern (all 0's), (iii) Data set 3: column stripe data pattern (101010$\cdots$), (iv) Data set 4: inverse column stripe data pattern (010101$\cdots$). Then, each data set is read back from the DRAM module at the reduced *Activation* time (t_{RCD}) to capture module dependent erroneous outputs (See Section 2.3.1 and Figure 2.5). Note that one may choose a different timing parameter (e.g., reducing *Precharge* time) to obtain the DRAM signature [BTRFR19].

Step 2: Feature selection. It is crucial to select the optimum number of features since the performance of classifiers is sensitive to the choice of the features and features' attributes such as correlation, noise, and other factors. In this step, we select the key features that can effectively capture architectural, layout, and process variations observed in DRAM since they directly impact the accuracy, computation time, and storage (of golden data) of our proposed technology. The classification models are created based on a total of 26 features collected from the four sets of data. Features are extracted from the whole data that is read out from one page. A single bank from 1GB memory module contains 16k+ pages per bank (eight banks

per module), and for the case of a 2GB memory module, each bank includes 32k+ pages. Each memory page contains 1,024 words and each word contains 64-bits of data. The data collected (at reduced t_{RCD}) from each memory pages are then rearranged into a 1,024×64 (denoted as d_R of size $w \times b$, where, w = number of words in a page, b = number of bits in each word) binary array. Moreover, for each page, we create another array, d_F (same size of d_R) which tracks the location of flipped bits. Note that, the $d_F(i,j) = 1$, if $d_R(i,j)$ is flipped with respect to the actual data that is written to the DRAM otherwise, it will be 0. The following features have been chosen from each page to identify the DRAM origin (i.e., the origin of the manufacturer and the specification of individual DRAM).

Feature 1 (Ψ_1): The total number of flips, also known as failed bit count (FBC), is used to capture the data dependency, process variation, and layout variation of the DRAM chips. The silicon results show that the FBC counts change from one DRAM module to another module.

Feature 2 (Ψ_2): The subset of FBC bits that are flipped to logic 1.

Feature 3 (Ψ_3): The compression ratio (r) depends on the distribution of ones and zeros in the data, d_R. The compression ratio effectively measures the randomness in the data. Random data have a higher compression ratio than non-random data. compression ratio is defined as Equation 4.1.

$$r = \frac{S_u}{S_c} \tag{4.1}$$

Where S_u and S_c are the sizes of the uncompressed and compressed data respectively. Our preliminary experimental results show that the compression ratio of d_R varies from one manufacturer to another. We compress data using standard the ZLIB library [DG], which is well-known and well optimized for the minimal computational overhead for data compression applications.

Feature 4 **(**Ψ_4**):** The whole block of d_F is divided into a set of smaller blocks (each block is 64×1 of size and denoted by B_w). The standard deviation on the FBCs in these B_ws are considered as a feature. This feature captures the spatial locality of FBC along the dimension w. The higher value of standard deviation represents a greater spatial locality.

Feature 5 **(**Ψ_5**):** The block d_F is divided into a smaller block, B_b (of size 1×8) and then FBC is counted on each of the smaller blocks. Then we choose the standard deviation of those FBCs as the feature Ψ_5. The spatial locality along the dimension b is captured with this feature Ψ_5.

Feature 6 **(**Ψ_6**):** Like Ψ_4, we calculate the standard deviation of FBCs on 64 blocks (of size 1024×1). This feature captures the fact that some cell locations of each 64-bit words are more error-prone than others.

Feature 7 **(**Ψ_7**):** Like Ψ_5, we calculate the standard deviation of FBCs on 1024 blocks (of size 1×64). This feature explores the fact that some bitlines are more error-prone than others.

Our experimental results show that the above set of features provides an excellent texture description of DRAM signature data [WMNS00]. All features except the Ψ_2 is extracted from all four data sets. The feature Ψ_2 is only extracted from the dataset 3 and dataset 4 (see Step 1). Choosing block-size for Ψ_4 through Ψ_7 is correlated to the DRAM organization. For example, we choose a block size of 1×8 (B_b) for Ψ_5 to capture the variations among the chips in a DRAM module. Our tested DRAM modules consist of four or eight chips, and each chip shares $1,024 \times 16$ or $1,024 \times 8$ blocks. We use block (B_b) height of 8 to extract average variations among the chips and to ensure consistency among classes.

Step 3: Machine-learning algorithms for detecting the DRAM origin. After extracting the most suitable feature, we develop a machine-learning

based technique to identify counterfeit DRAM modules. In our proposed technique, we use one-class classifier. Although one-class classifier is a more complex statistical problem, recent works demonstrated that it is more advantageous compared to binary-class or multi-class classifier while detecting counterfeit ICs [HLK+15, SKR+13, ABN+16, HCM12]. In the traditional binary-class classifier, if the statistical diversity is enormous in the negative samples, the classifier might provide poor decision boundary due to the small negative train data [CLL13, TD04]. In such a scenario, it will be very expensive or even impossible to collect data from the negative class covering the wide statistical diversity. This situation is particularly true for counterfeit IC detection as counterfeit ICs can be introduced from a wide variety of sources (see Chapter 1). On the contrary, the one-class classifier [CLL13, TD04, SSWB00, CPBBFR15, KM14] is trained by only positive class samples. In our proposed method, we use Support Vector Data Description (SVDD) [CLL13, TD04] to detect the outliers of a specific class. SVDD creates a spherical decision boundary in feature-space around the train dataset of a given class. For a given training data $x_i \in \mathcal{R}^n, i = 1, 2, 3, ..., l$, Tax *et al.* [TD04] solved the following optimization problem given by Equation 4.2.

$$
\begin{aligned}
&\min_{R,\boldsymbol{a},\xi} R^2 + \mathcal{C} \sum_{i=1}^{l} \xi_i \\
&\text{subject to, } \|\varphi(\boldsymbol{x}) - \boldsymbol{a}\|^2 \leq R^2 + \xi_i, i = 1, 2, 3, ..., l \\
&\qquad\qquad \xi_i \geq 0, i = 1, 2, 3, ..., l
\end{aligned}
\tag{4.2}
$$

Here, ξ_i is a slack variable and $\varphi(\boldsymbol{x})$ is the mapping function from the lower dimension to a higher dimension. R and $\boldsymbol{a}$ are the radius and center of the encircling boundary, and $\mathcal{C}$ is the regularization parameter. A smaller value of $\mathcal{C}$ causes more training samples to be treated as an outlier. A sample will be considered as an

outlier if $\|\varphi(\boldsymbol{x}) - \boldsymbol{a}\|^2 > R^2$. However, Equation 4.2 can be efficiently solved by the Equation 4.3.

$$\begin{aligned} &\max_{\alpha} \sum_{i} \alpha_i \mathcal{K}_{i,i} - \sum_{i,j} \alpha_i \alpha_j \mathcal{K}_{i,j} \\ &\text{where, } \sum_{i=1}^{l} \alpha_i = 1 \end{aligned} \tag{4.3}$$

Here, $\mathcal{K}$ is the kernel function (i.e., $\mathcal{K}_{i,j} = \langle \varphi(x_i)^T \cdot \varphi(x_j) \rangle$). In our case, we have chosen the radial basis function (RBF) as the kernel function [SSB+97]. The radial basis function is useful when the data are not linearly separable. The RBF function can be expressed with Equation 4.4.

$$\mathcal{K}_{i,j} = exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \tag{4.4}$$

In Equation 4.4 γ is a free parameter. A larger value of γ enables the classifier to capture more complex attributes of the training data. However, the trained model might suffer from overfitting problem if the value of γ is too large. However, the $\mathcal{C}$ and γ can be optimized more efficiently by introducing artificial outliers and applying k-fold cross-validation [TD01]. Moreover, the classifier accuracy can be increased by introducing some real negative examples during training [TD04].

Step 4: Constructing a framework to detect DRAM manufacturer. Figure 4.1 presents the proposed framework to identify the DRAM origin (i.e., the origin of the manufacturer and the specification of individual DRAM). In the proposed framework, we assume that the manufacturer or a trusted party provides the classifier model to the consumer and also defines a threshold for *Positive Page Rate* (PPR). The *positive page rate* (PPR) is defined as follows-

$$PPR = \frac{\textit{No. of pages that are classified as "positive"}}{\textit{No. of test pages from the memory module}} \tag{4.5}$$

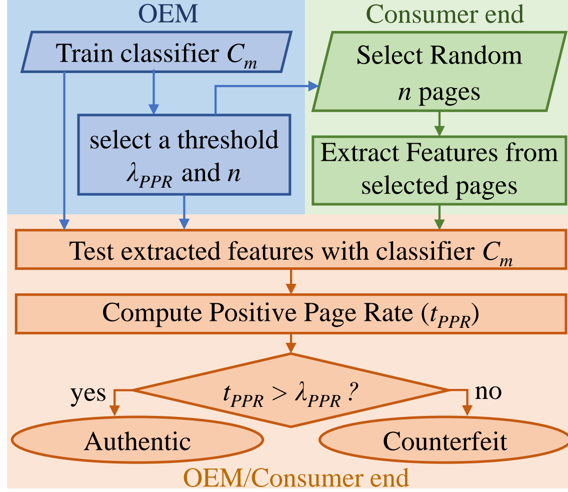
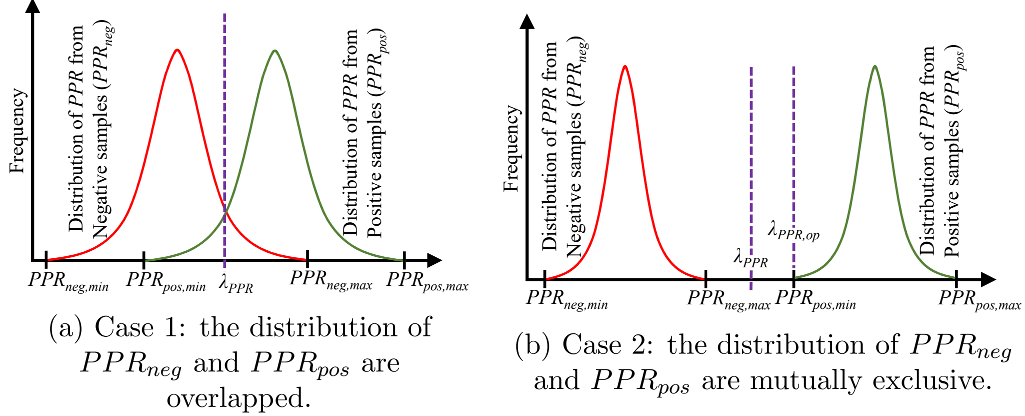


Figure 4.1: Proposed framework for identifying the origin of the DRAM.

In Figure 4.1, the steps, shown in the blue region, are performed by the OEM (Original Equipment Manufacturer) or a trusted third party assigned by OEM, and the steps shown in the green region are performed at the consumer end. All other steps (covered by the orange region) can be processed in either consumers' system or manufacturers' system. Initially, the OEM or a trusted party trains a classifier based on all page data that are captured from one or multiple DRAM samples of the target class. The OEM or trusted party should also specify the number of memory pages that need to be tested from a memory module to prove its authenticity. Then, based on the sample statistics, the OEM should choose a threshold (λ_{PPR}) to decide whether a DRAM is manufactured by them or not. If the PPR from the test module is higher than the threshold, then the memory module should be considered as authentic. The selection of the threshold value λ_{PPR} and the number of test pages (n) depend on the quality of the classifier and the manufacturing process. Higher process variations might cause a large statistical diversity on the manufactured memory modules and may increase the chance of miss-classification. Besides, a larger process variation may lead to a higher statistical variation among the memory pages from the same DRAM module. In such a case, we might need more randomly

sampled memory pages (i.e., a larger value of n) to capture all architectural and manufacturing process variations of a DRAM module. The choice of λ_{PPR} mostly depends on the quality of classifier. Figure 4.2a shows that the distribution of PPR



(a) Case 1: the distribution of PPR_{neg} and PPR_{pos} are overlapped.

(b) Case 2: the distribution of PPR_{neg} and PPR_{pos} are mutually exclusive.

Figure 4.2: Selection of λ with different distribution of PPR_{neg} and PPR_{pos}.

from positive samples and negative samples have an overlapping region. In such case, it is not possible to select a λ_{PPR} that creates a clear boundary between the positive samples and the negative samples. On the other hand, if the distribution of the PPR is mutually exclusive (Figure 4.2b), selecting a λ_{PPR} within the interval $[PPR_{neg,max}$, $PPR_{pos,min}]$ will separate positive and negative samples. Therefore, the ideal goal should be, maximizing the separation between $PPR_{neg,max}$ and $PPR_{pos,min}$ for a suitable value of λ_{PPR} during classification. As it is difficult/impossible to collect the negative class data that covers the whole distribution (discussed in Step 3), the λ_{PPR} should be defined with the highest possible value (i.e., $\lambda_{PPR,op} = PPR_{pos,min}$). In our proposed scheme, the OEM should train a classifier C_m, corresponding to a specific memory class and make the classifier parameter public. Then, the user should choose random n test pages from the memory module that is under test. The general information given with the classifier C_m should enable the user to extract features form those selected pages. Then, for each of those n test pages, the OEM/user

should test the extracted features using the classifier C_m. If the PPR (calculated from Equation 4.5) is higher than the threshold λ_{PPR}, the memory module should be marked as authentic. Otherwise, it should be identified as a counterfeit one.

4.4 Performance Evaluation of Proposed Framework

We evaluated our proposed technique by collecting data from 25 commercial off-the-shelf single rank DDR3 SODIMMs (small outline dual in-line memory module) from 3 major DRAM manufacturers (see Table 1) [Sta]. We tagged the memory class based on the part number, the Garber version (reference PCB layout version [JEDa]), and the SPD [JEDb] data (see Table 1). Among the first six classes, we found at least one mismatch in their SPD data (detailed of SPD data is not presented in this article). On the other hand, the last two classes only differed by their PCB layout version and SPD version. From each memory module, we collected data from all memory banks (each DRAM module contains eight banks). The testing platform has been implemented using a Xilinx Virtex ML605 evaluation board with SoftMC [HVK+17]. SoftMC is a reconfigurable DRAM controller framework that is designed for the Xilinx Virtex ML605 evaluation board. Data have been written and fetched from the DRAM memory module with two 32-byte data bursts. For all memory modules, we observed error patterns below 7.5ns of *Activation* time (the recommended *Activation* time is in between 10ns to 15ns [JED12]). Although shorter *Activation* time maximizes the number of corrupted data bits and our platform can reduce activation time to 2.5ns, in this work, we conducted our experiment with a 5ns of *Activation* time to make our experimental result reproducible by most of the system. Note that, in most DRAM-based systems, the memory controllers interface the DRAM module with a >200MHz clock (with a clock period <5ns).

Manufacturer	Part Number †	Country Origin	Quantity	SPD-Garber Version	Class tag
Micron	M1	China	2	10-C1	1
	M2	Singapore	3	10-B1	2
		China	4		
Samsung	S1	China	1	10-B1	3
	S2	China	1	11-B2	4
	S3	China	4	11-B2	5
		Philippines	3		
SK Hynix	H1	Korea	5	10-B1	6
	H2	China	1	11-B2	7
		Korea	1		

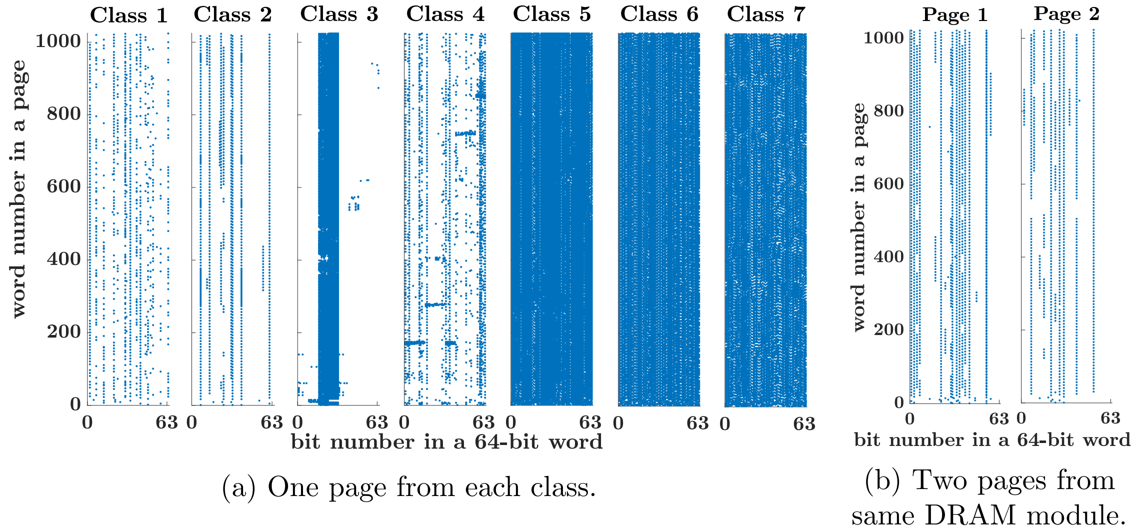
Table 4.1: Memory modules used in the data set.

In order to quantify the robustness of our proposed technique, we evaluated our proposed method in four different operating condition- i) nominal voltage (1.5v) and room temperature (25°C) (NVRT), ii) high voltage and room temperature (HVRT), iii) low voltage and room temperature (LVRT), and iv) nominal voltage and high temperature (NVHT). For HVRT and LVRT, we changed the input voltage (V_{DD}) by ±1% as most of the memory controllers limit the voltage ripple within ±1% [Tex]. For NVHT, we changed the operating temperature by +15°C from the room temperature.

Figure 4.3 presents the spatial locality of failed bits in a randomly chosen page form each memory class at NVRT operating condition. From the scatter plot, we observe that the error pattern is different for different classes. Note that, the PCB layout version only differs class 6 and class 7 and the subtle difference is difficult to understand from the figure (Figure 4.3a). In Figure 4.3b, we present the spatial locality of failed bits on two random pages from the same memory module of the

† **M1:** MT4JSF12864HZ-1G4D1, **M2:** MT8JSF12864HZ-1G4F1,
S1: M471B2873EH1-CF8, **S2:** M471B2873GB0-CH9, **S3:** M471B5773DH0-CH9,
H1, H2: HMT325S6BFR8C-H9;
M1, M2, S2: 1GB 1333MT/s, **S1:** 1GB 1066MT/s, **S3, H1, H2:** 2GB 1333MT/s

same class. Although there is some similarity in their texture, the pattern is not consistent. However, the features extracted from these samples (as discussed in Section 4.3) are still capable of separating these classes.



(a) One page from each class.

(b) Two pages from same DRAM module.

Figure 4.3: Spatial locality of failed bits from randomly chosen DRAM pages.

From each memory page, we extracted a total of 26 features as described in Section 4.3. We applied these 26 features directly to train and test the classifier. Then, we used the Linear Discriminant Analysis (LDA), a linear transformation [CG15], to provide the best visualization of the class separability. The LDA projects the data into a lower dimension feature-space by keeping the maximum separability among the classes. In the LDA, the lower dimension and the higher dimension features are linearly dependent. The data distribution in lower dimension ($\boldsymbol{\Psi}$-space to $\boldsymbol{\psi}$-space) feature-space is presented in Figure 4.4. In this figure, we only considered the most significant 5 dimensions ($\psi_1, \psi_2, \psi_3, \psi_4$, and ψ_5) in the new feature-space that provides the maximum separability (explained variance). From the figure, we observe that each of the class forms a cluster in the feature space, which enables the separation of manufacturers.

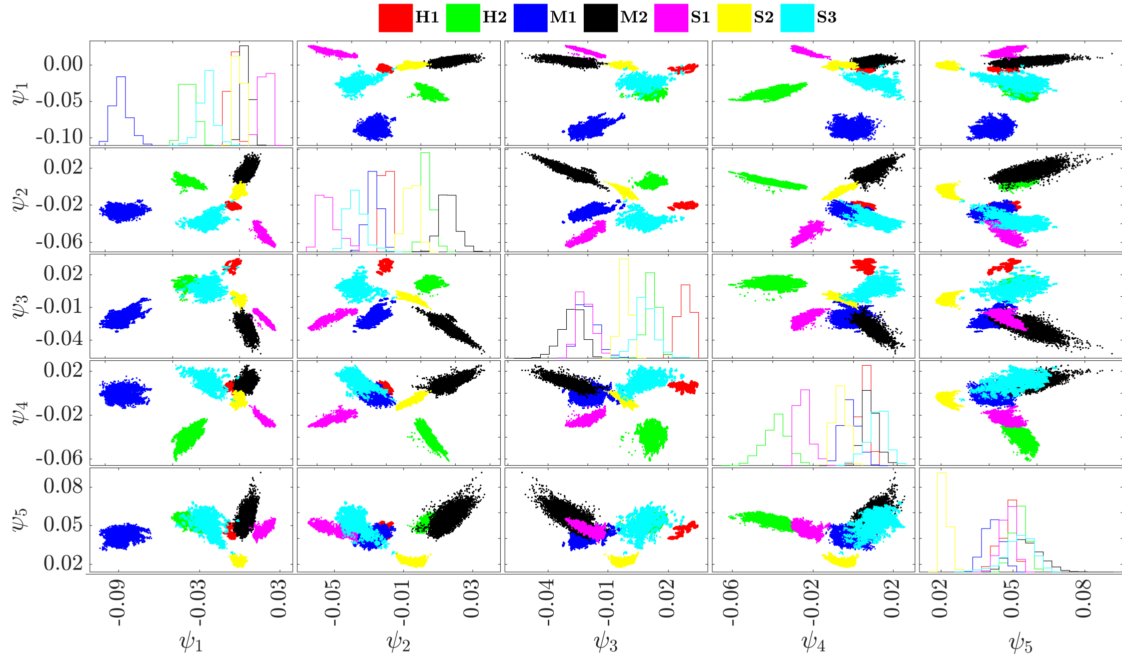


Figure 4.4: Visualizing data in feature-space.

To demonstrate our proposed method, we trained one-class SVDD classifier (as discussed in Section 4.3) for each class where we assume that the manufacturer does not have any prior knowledge of the memory modules from other class (See Section 4.2). The classifier was only trained at NVRT operating condition, and the same classifier was used to test the data from other operating conditions. Training one class classifier is a more complex statistical problem compared to the multi-class problem. We used LIBSVM library to implement the one class classifier [CL11]. For each class, we selected only one module to train the classifier (using 26 features from all pages collected at NVRT condition) and then tested all the pages from the rest of the 24 modules with the trained classifier with all operating conditions.

To validate our algorithm, we chose all possible combinations of training and testing data sets. In Table 4.2, we presented the result from the one-class classifier.

*For class 3 and class 4, we have only one sample which is used to train the statistical model. There is no positive test sample left for these two classes.

Class Tag	Operating Condition	$PPR_{pos,\mu}$	$PPR_{pos,\sigma}$	$PPR_{pos,min}$	$PPR_{neg,\mu}$	$PPR_{neg,\sigma}$	$PPR_{neg,max}$
1	NVRT	0.00	0.00	0.00	0.04	0.26	1.77
	HVRT	0.00	0.00	0.00	0.03	0.20	1.37
	LVRT	0.00	0.00	0.00	0.03	0.22	1.51
	NVHT	0.00	0.00	0.00	0.05	0.31	2.08
2	NVRT	99.07	1.16	95.40	0.00	0.02	0.17
	HVRT	98.04	3.54	87.60	0.00	0.01	0.10
	LVRT	99.09	0.98	96.49	0.00	0.02	0.19
	NVHT	99.33	0.58	97.53	0.00	0.01	0.07
$3^\star$	NVRT	–	–	–	0.00	0.01	0.06
	HVRT	–	–	–	0.00	0.01	0.06
	LVRT	–	–	–	0.00	0.01	0.05
	NVHT	–	–	–	0.00	0.01	0.07
$4^\star$	NVRT	–	–	–	0.00	0.00	0.00
	HVRT	–	–	–	0.00	0.00	0.00
	LVRT	–	–	–	0.00	0.00	0.00
	NVHT	–	–	–	0.00	0.00	0.00
5	NVRT	84.19	6.92	59.79	0.38	1.02	6.8
	HVRT	84.74	7.16	59.31	0.41	1.10	7.19
	LVRT	84.19	6.06	60.92	0.40	1.02	6.7
	NVHT	82.56	7.63	57.00	0.38	1.03	6.96
6	NVRT	90.29	9.31	69.58	1.58	4.16	25.91
	HVRT	90.35	9.39	68.98	1.54	4.14	25.78
	LVRT	81.98	9.49	69.74	1.69	4.52	28.65
	NVHT	90.32	9.31	69.17	1.52	3.80	23.64
7	NVRT	73.81	10.46	66.41	1.56	4.73	21.72
	HVRT	74.22	9.36	68.11	1.61	4.91	22.98
	LVRT	71.55	14.67	61.18	1.56	4.75	21.19
	NVHT	76.01	6.53	71.39	1.53	4.72	22.15

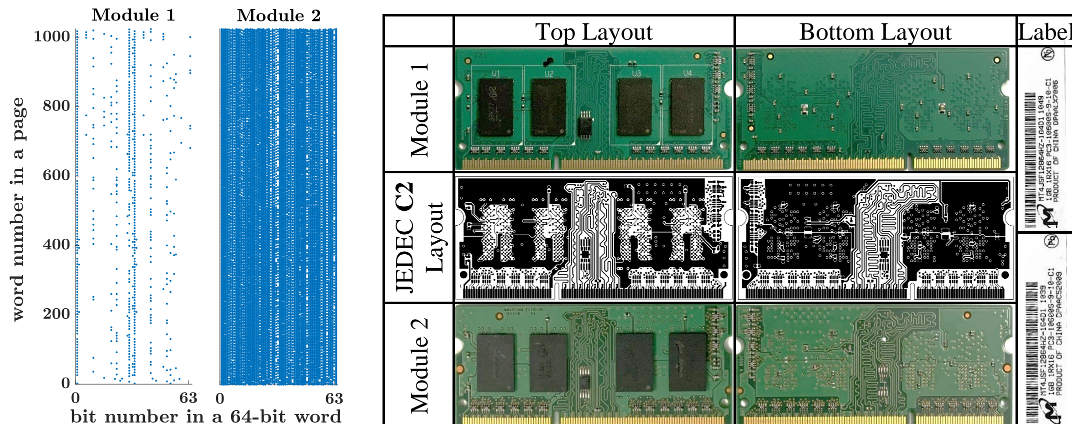
Table 4.2: Results from the one-class classifier.

The third, fourth, and fifth columns of the table represent the mean, the standard deviation, and the minimum PPR from the positive samples for each classifier (PPR is calculated from each test module). The sixth, seventh, and eighth columns of the table represent the mean, the standard deviation, and the maximum PPR from the negative samples. For the ideal case, the PPR_{pos} and PPR_{neg} should be 100% and 0% respectively. The standard deviation for both cases should be 0%. A larger gap

between the PPR_{pos} and PPR_{neg} provides us the flexibility of choosing appropriate values of λ_{PPR} and n for identifying the origin of the manufacturers along with specification with high confidence (Section 4.3). Our silicon results provide a satisfactory difference between the PPR_{pos} and PPR_{neg}, which can be further improved by learning statistical model with more positive samples and/or introducing negative samples. Table 4.2 also presents that a small change in voltage and temperature has a very insignificant effect on classifier performance. This is expected because a small change in voltage or temperature has a negligible effect on *Activation* time. [KPHM18, CKH+16, CYG+17, CGW+14].

A suspicious DRAM module: From the results shown in Table 3, for class 1, we observe that the PPR_{pos} is 0% (the ideal PPR_{pos} is 100%). Note that we have two samples available for this class: one is used for training, and another one is for testing. Therefore, we suspect that one of them is counterfeit. Figure 4.5a presents the spatial locality of failed bits from 2 random pages from those two samples. The results show that they have distinct FBC properties. Furthermore, the dissimilarities found in visual inspection (Figure 4.5b) suggests that one of them might be counterfeit (i.e., from a fake manufacturer). The layout difference between these two modules suggests that the reference layout version should be different for these two modules. However, the reference layout version is described as "C1" on both modules' label. From the SPD data, we found that the reference raw card (i.e., layout) version is specified as "C" (which represents- "C0", "C1", "C2" etc.) for both modules. For further investigation, we checked the layout provided by the JEDEC [JEDa] and found that the second module layout version is "C2" instead of "C1" (as shown in Figure 4.5b). Additionally, we also found that the PCB quality of the second module is much inferior compared to the first one. Therefore, we conclude that the second module is either from the fake manufacturer or mislabeled

(with layout version “C1”).



(a) Spatial locality of erroneous bits of 2 random pages of each samples.

(b) Visual appearance of each sample (Module 2 is suspicious).

Figure 4.5: Visual representation of DRAM modules from Class

.

4.5 Limitations

Our proposed method can be used to detect various counterfeit types. However, the proposed work cannot identify overproduction from the same foundry [RFS+14]. An untrusted foundry can reuse the GDSII of and may sneakily produce chips out of contract. As those overproduced chips experience the same architectural, layout, systematic process variation, our extracted feature would not be able to draw any difference between authentic and counterfeit memory chips. In addition to that, an adversary might use the proposed method to select recycled memory chips that match an original OEM DRAM module. To prevent such a case, we will also have to find a different type of DRAM signature or a different set of features that identify recycled DRAM as well. For example, selecting a retention-based DRAM signature

might provide a better set of features to identify recycled DRAM as DRAM retention behavior changes over its usage [NRT19].

4.6 Conclusion

In this chapter, we proposed a simple non-invasive and low-cost scheme for identifying the origin of a DRAM manufacturer and verifying individual DRAM's specification. The proposed method exploits the DRAM latency variations to capture the architectural, layout, and process variations. At first, we chose the most appropriate features from the DRAM signature, and then we used a one-class classifier to verify the memory class without knowing the information from other classes (i.e., other manufacturers).

CHAPTER 5

IDENTIFYING SRAM ORIGIN

5.1 Introduction

This chapter presents a detailed description of identifying SRAM manufacturer and part number by extracting a set of features from the SRAM startup signature [TFR21]. Although SRAM is a relatively small component of a system, many expensive ICs of the system (e.g., microprocessor's cache, microcontroller's main memory, FPGA's BRAM, etc.) are integrated with the SRAM-based memory components (i.e., on-chip component). Hence, identifying counterfeit SRAM can potentially identify a large family of counterfeit ICs. The major contributions of this chapter include-

- We have extracted a set of features from the start-up state of SRAM chips to capture the architectural, layout, and process variations. We found that our proposed set of features can be used to identify the memory manufacturer and part-number.
- We have tested the robustness of our proposed method by varying operating temperature and testing platforms.
- We have also compared the extracted features between the fresh and aged (recycled) chips. The practical aging state of SRAM memory has been emulated by stressing the memory chip under high-temperature and supply-voltage conditions.
- We have validated our proposed technique with the data collected from 345 commodity SRAM chips (manufactured by five major vendors).
- We have provided a practical guideline to improve the accuracy of our proposed method with a realistic demonstration.

5.2 Differences from Prior Works

Although the feature extraction technique used for DRAM and SRAM identification has some similarities, extracting features from SRAMs is more challenging than the DRAMs. For instance, While we have identified DRAM modules (a combination of few DRAM chips), we identify individual SRAM chips in this work. Additionally, SRAM chips are much smaller than DRAM chips (few Kbytes to few Mbytes vs. couple of hundreds of Mbytes to fey Gbytes). Consequently, a small number of DRAM modules (containing multiple DRAM chips) can capture large systematic process variation and produce a more accurate statistical model. However, to capture the same degree of systematic process variation, one might require more SRAM samples to train the statistical model. Additionally, the SRAM chip architecture is less standardized than the DRAM chips. For example, it is up to the manufacturers' choice to divide the address bits into row-address bits or column-address bits for SRAM chips. On the contrary, the DRAM addressing is standard (usually, the least significant bits are used for the column addressing). Such ambiguity makes it hard to extract 2D spatial information from SRAM chips. In addition to these challenge, this work has the following additional differences with the prior work-

- In our previous work, we have used latency variation to extract the memory signature; however, in this work, we used the start-up data signature from SRAM chips. This work generalized memory signature requirement, i.e., any memory signature might be used to identify memory manufacturer/part-number.
- We have modified the previously proposed authentication framework. In our proposed work, we used only a single one-class classifier to identify both manufacturer and pat-number. However, later we realized that the part-number identification is more challenging using a one-class classifier and more practical to use a

binary/multi-class classifier. Hence, we proposed a one-class classifier to identify the manufacturer and a binary/multi-class classifier to identify the part-number in this work.

- We also demonstrated our proposed work's viability to identify the recycled memory chips, which was not presented in our previous work.

5.3 Proposed Method

In this work, we make a similar set of assumptions as we made to identify the DRAM origin (Section 4.2). However, the feature definition, extraction, and classification are much different than we presented in Section 4.3. Below, we summarize our proposed method for identifying SRAM origin.

5.3.1 Feature Selection

In our proposed method, we collected 20 sets of start-up data ($\{D_1, D_2, \cdots, D_{20}\}$) from each SRAM chip. We constructed a unified data, D, based on majority voting[1] cast by $\{D_1, D_2, \cdots, D_{20}\}$. SRAM memory cells are generally arranged in a 2-D array of size $r \times c$ ($r = number\ of\ rows$ and $c = number\ of\ columns$). If each word of a SRAM chip consists of w_l bit data, then, for simplicity, we can assume that there is a total of w_l 2-D array of single bit contributing 1-bit data to each data word. So, the data D should be 3-D data of size $r \times c \times w_l$. However, to reduce the complexity, we rearrange the whole data in a 2-D array of size $n_w \times w_l$ ($= dim(D)$), where $n_w = r \times c$ is the number of words in the memory. Now we extract the following seven features from the start-up data D [BTMR+20]:

[1] In the majority voting technique, each PUF bit is sampled multiple times, and the value of that PUF bit is assigned as the majority of the samples [SMK+18].

- ***Feature 1 (Φ_1):*** This feature quantifies the "cell biasness" by counting the number of logic "1" bits in the start-up data. The evaluation of Φ_1 is illustrated in Figure 5.1. In this example, we presented start-up data from an 8×4 ($n_w \times w_l$) SRAM chip containing four 8-bit words. In this figure, 16 bits contain logic "1" out of 32 bits. Hence, according to our definition, $\Phi_1 = 16/32 = 0.5$. Cell bias qualitatively measures the asymmetricity of the cross-coupled inverters (see Section 2.3.1). For an ideally symmetric SRAM cell structure, this value should be 0.5 (i.e., no "cell bais"). However, in practice, this value is usually deviated from 0.5 because of the different variations discussed previously (see Section 3.1).

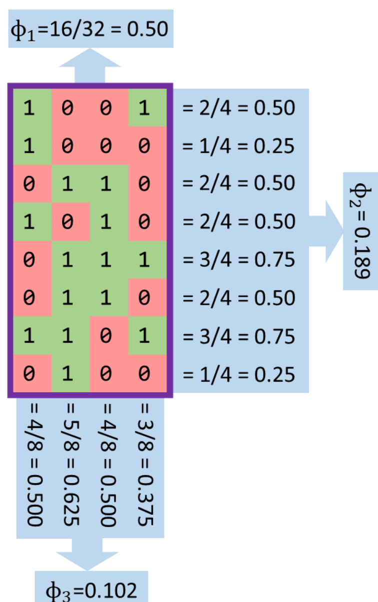


Figure 5.1: Illustration of Φ_1, Φ_2, and Φ_3 (8×4 SRAM).

- ***Feature 2 (Φ_2):*** The fraction of logic bit "1" is counted in each word of data D; then, the standard deviation of those values was taken as the feature Φ_2. Φ_2 is also illustrated using Figure 5.1. In this figure, we first calculated the fraction of logic "1" from each word (along the row), and then Φ_2 is estimated by computing the standard deviation of those values. In an ideal case, the distribution of logic

bit "1" from each data word should be normally distributed with a mean of 50%. Our experimental results demonstrate that the mean is close to Φ_1. However, the standard deviation of distribution may vary from chip to chip depending on memory specification (i.e., for some memory chips, the distribution can be flatter than other chips of different specifications). Φ_2 quantifies the symmetricity of the SRAM cell array. For example, each SRAM cell might experience different systematic process variations due to the local layout patterns[2]; hence, data words from different address locations might experience different logic distribution at start-up. A larger variation on local logic distribution will result in a larger value of Φ_2.

- ***Feature 3 (Φ_3):*** An SRAM chip of word size w_l can be assumed as a series of w_l 2-D SRAM arrays. We counted a fraction of "1" from each 2-D array for this feature and took the standard deviation as the feature Φ_3. If each of the 2-D arrays follows similar data distribution, and the Φ_3 should be close to 0. In Figure 5.1, each 2-D array is rearranged in a single-dimensional vector for visualization purposes and presented along each column. Now, to evaluate Φ_3, we computed the fraction of logic "1" along each column, and then standard deviation is calculated using those values. Φ_3 can capture different physical properties of the SRAM chips. For example, if the area constraint is too tight, all 2-D memory arrays can be located in close proximity or may be fused together. In that case, they may have a smaller difference in logic distribution due to smaller process variations.
- ***Feature 4 (Φ_4):*** The compression ratio (r, where, $r \geq 1$) of the start-up data is selected as one of the features. A start-up data with regular patterns have larger data redundancy and can be significantly compressed without any information

[2]Local layout patterns might be different from one cell to another, e.g., memory cells near the sense amplifier vs. memory cells at the middle of the SRAM array.

loss. However, start-up data with randomly distributed zeros and ones can be squeezed very little and causes a smaller value of compression ratio (closer to 1). Φ_4 can capture the impact of the random process variation on SRAM chips. The compression ratio is defined as Equation 4.1.

- ***Feature 5 (**Φ_5**):*** All data words from each SRAM chip are split into multiple blocks to extract this feature, where each block consists of 512 consecutive data words. Then we compute the fractional value (P_1) of each block of data that exhibits logic "1". We, then, calculate the standard deviation of P_1 calculated from each block. We select this standard deviation as feature Φ_5. This feature captures the spatial locality of logic "0" and logic "1" of start-up data. A higher value of Φ_5 signifies a larger spatial locality. Although we select the block size of 512, the manufacture may wish to select a different size that describes the best structural granularity in memory space. A smaller value of the block size might capture more spatial details; however, the Φ_5 will also be largely influenced by the local noise if the block size is too small. We experimented with different block sizes and found that 512 provided the best result for memory classification. It is worth mentioning that Φ_2 is similar to Φ_5, where the block size of Φ_2 is only one word. Hence, Φ_2 captures finer grain spatial information more effectively. However, Φ_2 may also capture the local noise information.
- ***Feature 6 (**Φ_6**):*** For each memory cell, we have collected SRAM data a total of 20 times and mark those memory cells as noisy if logic "1" is observed 8 to 12 times. We marked those cells as noisy signature bits. For this feature, we counted the percentage of noisy signature bits. In a well-designed SRAM memory cell, the coupled inverters are highly matched, and corresponding signature bits are largely affected by the external/internal noises (e.g., voltage fluctuation, thermal noise, etc.). Furthermore, we believe that this feature can contribute highly to

detect recycled memory chips. Over the usage, there will be more cells with large threshold voltage mismatch in recycled memory chips [GWHS19] and will produce large $PSNM_{noise}$ (see Section 3.6). Hence, a recycled SRAM chip should produce less noisy signature bits and reduce the value of Φ_6 over time.

- ***Feature 7 (Φ_7):*** This feature is similar to the Φ_2. In this feature, instead of accounting for the theoretical normal data distribution, we made a $(w_l + 1)$-bin histogram. If a data-word ($\in D$) occupies a total t bit of logic "1", and then it is placed in tth histogram bin. The standard deviation of the bin size quantifies as the feature Φ_7. If the distribution is normal, then Φ_2 and Φ_7 should be approximately the same (also well-known as *the normal approximation for probability histogram*). Hence, the Φ_7 measures the skewness on word ($\in D$) distribution from the normal distribution.

We extract all these seven features from both fresh (i.e., new) and aged (i.e., recycled) SRAM chips. Then we show that these features form visually separated clusters in feature space depending on the SRAM module type (manufacturer "A" vs. Manufacturer "B", Part number "X" vs. Part number "Y", fresh vs. aged/recycled).

In addition to above features, manufacturers may choose a different feature-set that describe their chips more concisely. However, when the manufacturer itself does not define the features and assign the responsibility to a third-party, one or few features might not obtain the exact electrical characteristics as intended due to the special modification at the architectural or layout level (which might not be known to the third-party). For example, bit-level scrambling in the data word may limit the usefulness of feature Φ_3 [EP17]. Nevertheless, as we are using multiple features, a well-trained statistical model (described in Section 5.3.2) might still learn the difference between two groups of SRAM chips by utilizing other features available from the feature pool.

It is worth mentioning that the features described above only provide qualitative information of different physical properties of the SRAM chips; however, they do not provide any quantitative information. Furthermore, each feature described above might be impacted by combined information from multiple physical properties. For example, although Φ_5 primarily varies from one memory class to another due to spatial variation, Φ_5 might also be impacted by the address scrambling caused by the architectural difference in the address decoder [vdGS02].

5.3.2 Identifying Authentic Memory Chips

Usually, memory chips with the same manufacturer and specification are labeled with a unique part number; hence, to identify a memory authenticity, we need to identify the memory part number. We propose a machine learning-based approach to classify the memory part number after extracting features from the start-up data. However, the classification can be done with two different approaches- a) learning a binary classifier (positive vs. negative) for each class, and b) learning a one-class classifier for each class. In the first approach, we learn a binary classifier for each class to differentiate between positive samples and negative samples (i.e., authentic vs. counterfeit). This approach is only applicable when both positive and negative sample is available while training the classifier. Nonetheless, it is not a practical approach due to the enormous diversity in negative samples. Collecting negative samples from whole statistical distribution is not cost-effective and time-efficient. In the second approach, we do not need any samples from the negative class, and only positive samples are sufficient to learn the classifier. In our previous study (see Chapter 4), we showed that a one-class classifier is preferable for counterfeit IC detection as the statistical diversity of the counterfeit chips (negative class) is too

large, and they can be introduced from a large number of sources (see Chapter 1). Unfortunately, one-class classification is a complex statistical problem and might reduce the accuracy. Hence, we propose a two-step approach to solve this issue:

1. **Identifying manufacturer:** Different vendors use different memory cell designs, design flow, and possibly fabrication facilities. Furthermore, they may integrate different peripheral inside the memory; for example, altering row-decoder may alter apparent start-up data locality seen from outside of the memory. Hence, multiple sources may contribute to start-up data variation among SRAMs manufactured by different vendors. In other words, SRAMs for different manufacturers appeared to have a larger difference in their features (large inter-manufacturer feature distance), which ease identifying the SRAM manufacturer (e.g., manufactured by vendor "A" or not). However, while training a binary-classifier, it is impossible to learn all the negative samples that the target vendor does not manufacture. Therefore, we propose a one-class learner (e.g., one-class Neural Network, one-class SVM, SVDD, etc. [BTMR+20, ABN+16, RBT21, SKR+13, HCM12]) only to identify the manufacturer information. However, one may choose to train a binary-class classifier with all available negative samples along with a one-class classifier to improve the accuracy. Note that, in this work, we only used a simple binary classifier for identifying memory manufacturers to reduce the complexity of our experimental evaluation.
2. **Identifying part number:** A manufacturer usually produces different memory chips with different specifications with different part numbers. However, they may use the same design facility and similar peripherals for all of them, leading to a more subtle feature difference among memories. Fortunately, we can assume that a manufacturer can easily access all memories that they manufacture. Therefore, once the manufacturer is identified, the target manufacturer

can easily provide a binary (target class vs. others) or a multi-class classifier to identify each memory part number produced by them. As we mentioned earlier, the one-class classifier is a complex learning task; hence we should avoid it when we have access to the negative samples from the whole statistical distribution. In this particular scenario, one-class learning is more difficult as we have a smaller feature distance among part numbers produced by the same manufacturer.

5.3.3 Proposed Framework

We propose a machine learning-based algorithm that uses the device signature to verify the manufacturer and the part number. Figure 5.2 represents the detailed framework of our proposed technique. Using a golden set of sample memory chips, the manufacturer needs to extract a set of features as explained in Section 5.3.1 and train classifiers to identify counterfeit chips. The manufacturer can train the classifier in two steps: (**i**) learning manufacturer-specific property (C_m) and (**ii**) learning part number-specific property (C_p). Manufacturing-specific property can be learned by a one-class classifier (i.e., only learning the target manufacturer) and might be assisted by a binary classifier (i.e., target manufacturer vs. others). For the second step, the manufacturer can train either a multi-class classifier for all part numbers or a multiple binary (one vs. all) classifier for each part number. By using publicly available information provided by the manufacturer, a user should be able to collect the signature from his samples and extract the feature-set. If the classifier information is available, the user can verify the chip authenticity by himself. Otherwise, the user can send the extracted feature-set to the manufacture, and the manufacturer can verify the authenticity of the test memory chip.

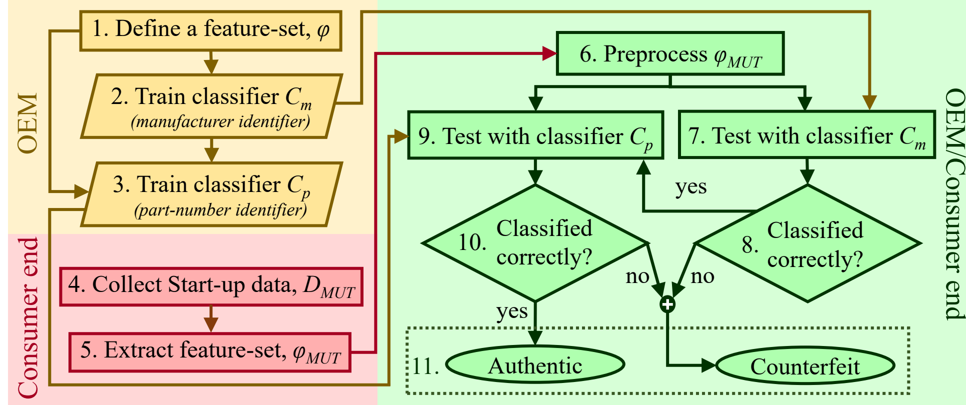


Figure 5.2: Proposed protocol to identify counterfeit SRAM.

5.3.4 Identifying Recycled Memory Chips

Although identifying memory manufacturer and part number can prevent many types of counterfeitings [BTMR+20], identifying memory manufacturer and part number does not capture the recycled memory chips. Fortunately, the features we described in Section 5.3.1 can also be used for identifying recycled memory chips. For example, the distribution of the 0's and 1's can be skewed over time due to the skewed distribution of 0's and 1's in functional memory usage, which can be easily captured by Feature 1 [GWHS19]. Additionally, we observe that the distribution of other features may help to identify recycled SRAM chips in extreme cases, i.e., when only the symmetric data patterns are used over functional memory usage (see Section 5.4.3).

5.4 Performance Evaluation of Proposed Framework

We evaluated our proposed framework by using silicon data. In our experiment, we collected SRAM start-up signatures to demonstrate our proposed technique. Typically, the success of any machine learning (ML) model relies on the sample quality and sample size. However, it is difficult to collect data from a large set of

sample chips in a lab environment and imitate all possible operating conditions. Therefore, we divide the data collection process into the following tasks:

1. We used Arduino Due board [Ard] for collecting start-up data from SRAM chips. We used 345 4-Mbit (256K×16) SRAM chips from 5 major manufactures and 23 different part numbers (i.e., 23 memory classes). All of these 23 part numbers are tabulated in Table 5.1. From now on, we will use the "tag" (specified in Table 5.1) to recall a specific memory part number/class. We used 230 SRAM chips to train ML models (10 chips from each class) and 115 chips to test the model (5 chips from each class).
2. We collected data from both test chips and train chips at a nominal voltage (3.3V) and room temperature (25°C). We used two different Arduino boards to emulate the platform variation among different embedded systems and utilized them to collect start-up signatures from test samples. We found that the operating voltage of these two boards is within ±35mV of the nominal voltage.
3. We used a one-vs-all binary classifier (positive vs. negative) for both manufacturer identification and part number identification. As we explained in Section 5.3.2, the one-classifier would be the best for the manufacturer identification. However, the one-class classification task is a complex statistical problem and might require a large number of samples to train the model.
4. Data noise can impact the classification models severely. To reduce noise, we collected start-up data from the same SRAM chips 20 times. We maintained a constant sampling interval of 2 minutes. We shorted the power pin (V_{CC}) and other control pins of the SRAM chip with the ground within this time interval. We maintained such settings using relay circuits (also controlled by the same Arduino Due board). This experimental setup should be sufficient to avoid the potential discharge inversion effect on the SRAM start-up state [LAWG17]. We

combined those 20 sets of data in a single set using the majority voting technique [SMK+18].

Vendor[ϒ]	CY					IDT						ISSI					AMI			REA			
Part Number	CY7C1041G30-10ZSXI	CY7C1041CV33-20ZSXA	CY7C1041G18-15ZSXI	CY62147G30-55ZSXE	CY62146EV30LL-45ZSXIT	IDT71V416S10PHG8	IDT71V416S12PHG8	IDT71V416L15PHG8	IDT71V416S10PHGI	IDT71V416S12PHG	IDT71V416L15PHG	IS61LV25616AL-10TL	IS61WV25616BLL-10TL	IS61WV25616BLL-10TLI-TR	IS61LV25616AL-10TLI	IS61C25616AS-25TLI	AS7C34098A-10TCN	AS7C34098A-10TIN	AS6C4016-55ZIN	RMLV0414EGSB-4S2#AA1	RMLV0414EGSB-4S2#HA1	RMLV0416EGSB-4S2#AA1	RMLV0416EGSB-4S2#HA1
Tag	CY1	CY2	CY3	CY4	CY5	IDT1	IDT2	IDT3	IDT4	IDT5	IDT6	ISSI1	ISSI2	ISSI3	ISSI4	ISSI5	AMI1	AMI2	AMI3	REA1	REA2	REA3	REA4

Table 5.1: List of SRAM chips in experiment.

5. The variance error is expected when the sample size is too small [PS04]. A model with high variance provides too much attention to the data that are trained with and prone to overfitting. Hence, to reduce the variance error in the trained model, we segmented the SRAM signature data in 16 chunks and virtually increased the sample count by treating each segment as an individual memory chip (i.e., extracting an individual set of features from each segment). However, in the inference phase, the class of a test sample is determined by the majority voting method using all 16 segments. If the same number of votes supports multiple class labels, the tie is broken by comparing the cumulative posterior probabilities[3] of all 16 segments.

[ϒ] CY: Cypress Semiconductor; IDT: Integrated Device Technology; ISSI: Integrated Silicon Solution, Inc.; AMI: Alliance Memory, Inc.; REA: Renesas Electronics.

[3] The posterior probability quantifies the confidence level of inferencing a sample to a particular class [HTF09].

6. To examine the temperature sensitivity of our proposed technique, we collected data from test samples at high temperatures ($\sim 45°C$) and validated the same trained model learned in task 3.

5.4.1 Visualizing Features

The accuracy and efficiency of an ML algorithm largely depend on the quality of the features. Hence, to demonstrate the feature-merit (explained in Section 5.3.1), we present the feature distribution of train chip across different manufacturers and different part numbers in Figure 5.3 and 5.4, respectively. These figures show that most features are normally distributed (median is centered), and in many cases, at least one feature distribution of a particular class produces a clear visible separation with other classes (i.e., manufacturer "A" vs. all and part number "X" vs. all). For example, in Figure 5.3, the SRAM chips manufactured by Renesas Electronics are readily separable by the distribution of feature Φ_5. Similarly, Figure 5.4a demonstrates that SRAM chips from CY4 are easily distinguishable from the distribution of feature Φ_2. Unfortunately, in our case, many of the classes can not be separated from other classes based on their feature distribution due to the inter-dependency among those features. For example, feature Φ_1 (number of 1's) and Φ_4 (compression ratio) might have a close relation; for instance, if the signature data is highly random, the Φ_1 should be close to 0.5, and Φ_4 should be close to 1.

For such cases, the class separability can still be visualized if the current feature-space (Φ-space) is transferred to a new feature-space (φ-space), where the $\varphi_i = f(\Phi_1, \Phi_2, \cdots, \Phi_n)$. If the $\Phi_1, \Phi_2, \cdots, \Phi_n$ are nonlinearly correlated, then the $f(\cdot)$ is a non-linear function. In our experiment, we used generalized discriminant analysis (GDA) [BA00] to transform the Φ-space to φ-space, where data points are linearly

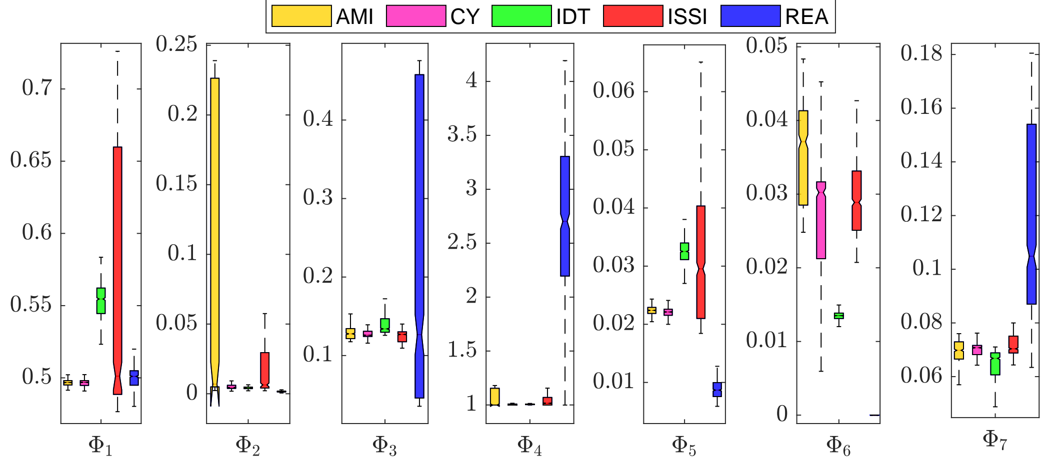
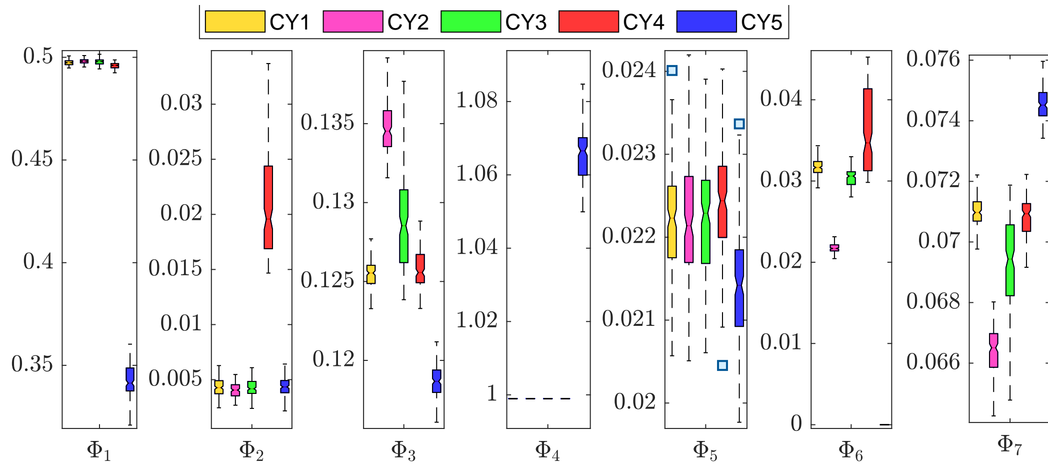


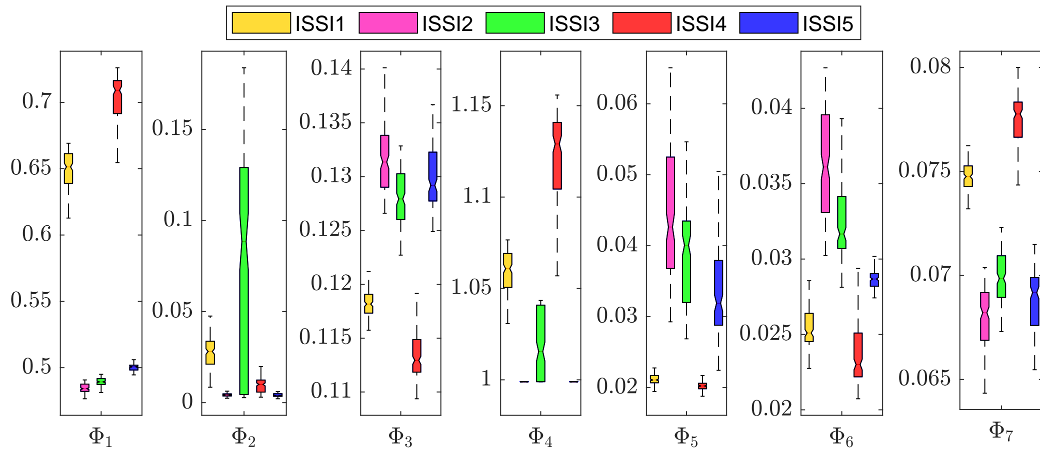
Figure 5.3: Visualizing feature distribution by manufacturer.

separable at φ-space. GDA[4] is a supervised machine learning technique to find a reduced set of features that preserves the maximum separability among the classes. This reduced set of features is related to the old feature space by a non-linear kernel function. In our experiment, we used an *RBF* kernel function (see Equation 4.4). The *RBF* functions' parameter (γ) is determined by the 10-fold cross-validation method and ensured minimum distance between samples and corresponding centroids. Figure 5.5 and 5.6 represent the test memory chips in φ-space (in 2D projection) and demonstrates the manufacturer and part number separability. Each dot in Figure 5.5 and 5.6 represent each memory segment as explained in task-5. Those two figures demonstrate that memory classes (manufacturer "A" vs. "B" and part number "X" vs. "Y") are fairly distinguishable in at least one 2D projection of the φ-space. While transforming the feature-space of a K-class problem, it is worth mentioning that at most $K - 1$ dimensions are required in the new feature-space without losing any information of class separability [HTF09]. However, for IDT, adding more than three dimensions (Figure 5.6c) only adds very small details on class separability (which is not recognizable from visual appearance). However, we

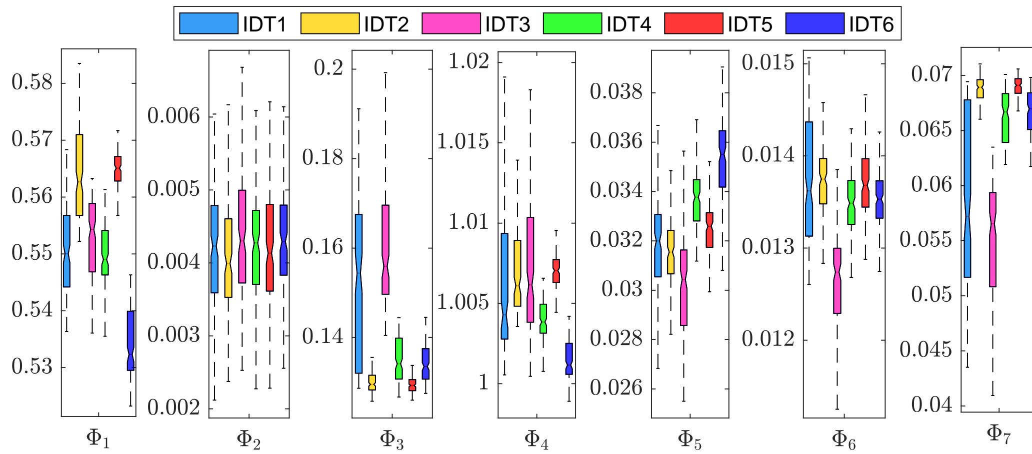
[4]Reference implementation of GDA: `https://github.com/mhaghighat/gda`



(a) Cypress Semiconductor

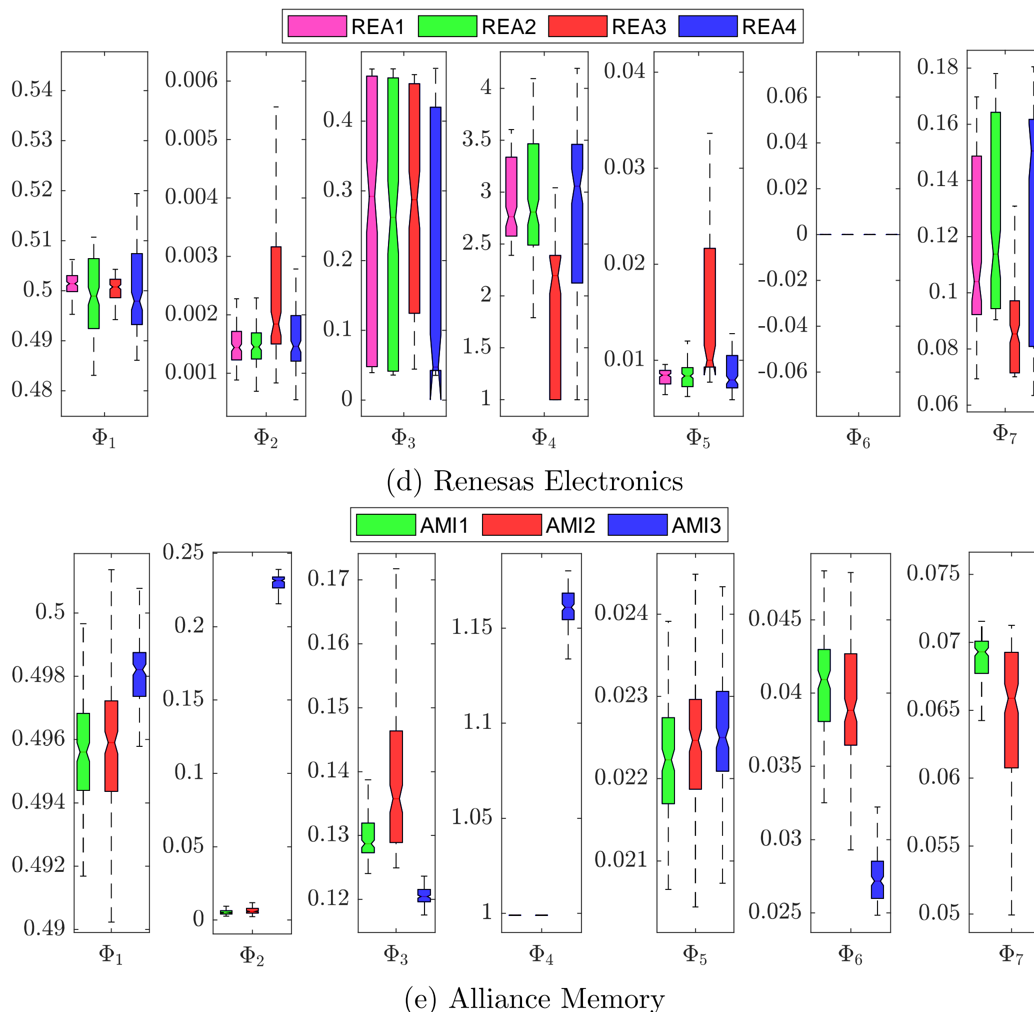


(b) Integrated Silicon Solution, Inc.



(c) Integrated Device Technology

Figure 5.4: Visualizing feature distribution by part number.



(d) Renesas Electronics

(e) Alliance Memory

Figure 5.4: Visualizing feature distribution by part number (Cont.).

used $K - 1$ dimensional new space for a K-class problem for other cases in Figure 5.5 and 5.6.

Note that some overlapping between multiple classes is still visible in the φ-space due to the random process variation. However, such overlapping can be reduced by further optimizing the RBF parameters, given that more train samples are available (we have only ten samples from each part number). While classifying the test memory chips, the impact of such overlap is minimized by assigning equal weight on all 16 segments of the chip and casting a "vote" from each segment.

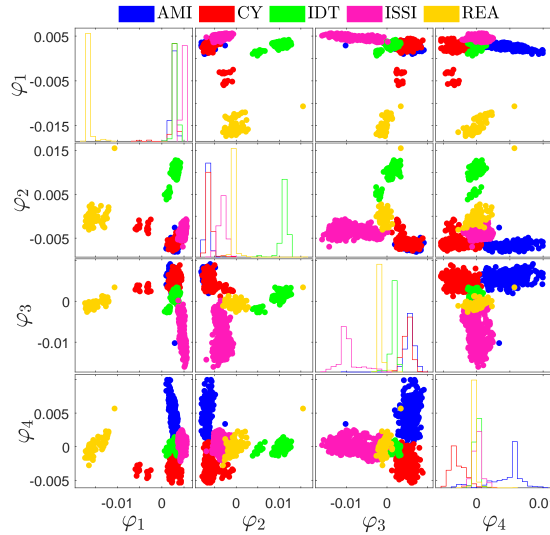
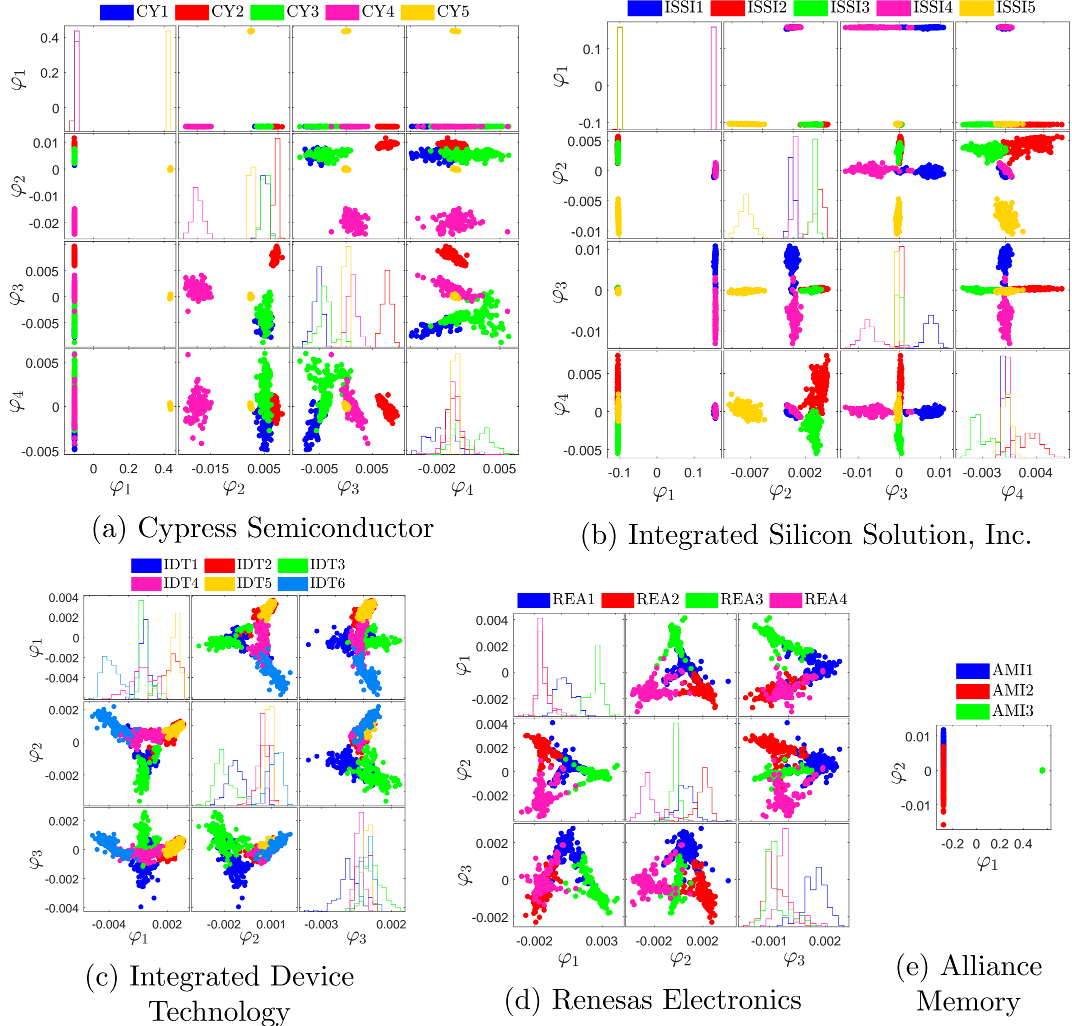


Figure 5.5: Representation of SRAMs in feature-space, clustered by manufacturer.

5.4.2 Labeling Test Memory Chips

Although the GDA can be used for both visualization and classification tasks, GDA is not ideal for a small sample size. Fortunately, the ensemble learning technique can still perform reasonably better even with a small set of samples [Die02]. In the ensemble technique, multiple base models are learned with different configurations, and then the output label of the test sample is determined based on the vote cast by each model. Although several ensemble algorithms are available, we used the bagging (**b**ootstrap **agg**regat**ing**) method in our experiment. The bagging method is similar to other ensemble methods, except the base model is trained with a different set of train data (sampled with replacement). The bagging method has the inherent ability to reduce the variance error of the trained model and can out-perform other ML algorithms when the train sample size is small [YIKA16].

In our experiment, we trained multiple ensemble models using different base classifiers (e.g., SVM, Decision Tree, Naive Bayes, Discriminant Analysis, Kernel, etc.), and the best model is chosen based on the 10-fold cross-validation score. Then



(a) Cypress Semiconductor

(b) Integrated Silicon Solution, Inc.

(c) Integrated Device Technology

(d) Renesas Electronics

(e) Alliance Memory

Figure 5.6: Representation of SRAMs in feature-space, clustered by part number.

we generated the test score based on our test samples. We represented the test score in Table 5.2 and 5.3. These tables present four types of test scores: *Precision* (P), *Recall* (R), F_1 score, and accuracy, which are defined by Equation 5.1, 5.2, 5.3, and 5.4, respectively. P quantifies the trained model's accuracy out of all predicted positives, and the R computes the fraction of positives that the model captures correctly. On the other hand, F_1 score is the harmonic mean of the P and R. For an ideal case, all of these test scores should be close to 1. Note that, the accuracy

is not a very useful metric when the test samples from the positive and negative classes are not equal (unbalanced data). In our experimental setup, the number of test samples for binary (one vs. all) classifiers is unbalanced; hence, we emphasize the P, R, and F_1 score in our discussion.

$$Precision\ (P) = \frac{tp}{tp + fp} \tag{5.1}$$

$$Recall\ (R) = \frac{tp}{tp + fn} \tag{5.2}$$

$$F_1 = \frac{2}{(P)^{-1} + (R)^{-1}} \tag{5.3}$$

$$A = \frac{tp + tn}{tp + tn + fp + fn} \tag{5.4}$$

Where,

tp = True positive

tn = True negative

fp = False positive

fn = False negative

We trained our binary model by utilizing the samples from the target class and the samples from the outlier class (i.e., not belong to the target class). Target class implies manufacturer (or part number), which is targeted to separate from other manufacturers (or part numbers). Note that we can either consider the target class as the positive class or the outlier class as the positive class in Equation 5.1, 5.2, 5.3, and 5.4; depending on the definition of positive class, the P, R, and F_1 score can be different for unbalanced test samples. We focus on the test scores produced by considering the target class as the positive class as it delivers the worse set of test scores.

Table 5.2 and 5.3 present a single accuracy score and two sets of P, R, and F_1 score considering both objectives as discussed above. In Table 5.2, the 1st row represents the target manufacturer, and the 2nd row represents the corresponding accuracy score. Row 3, 4, and 5 represent the P, R, and F_1 score considering the target class as the positive class. Similarly, row 6, 7, and 8 represent the P, R, and F_1 score considering the outlier class as the positive class. Column 2–6 represents the classifier score for each manufacturer, and column 7 (μ^V) represents the average classification score considering all manufacturers. The table shows that the average test scores are $\geq 0.92\%$ (positive class = target class), which is promising considering such a small number of samples. However, the classification score is a little lower for CY and AMI than the other manufacturers, resulting from the fact that CY and AMI slightly overlap in feature space (blue and red dots in 5.5). However, the classification scores can be improved by adding more samples and further optimization of the classifiers.

Vendor		CY	IDT	ISSI	AMI	REA	μ^V
A		0.93	1.00	0.99	0.97	0.99	0.98
Target Class	P	0.87	1.00	1.00	0.87	1.00	0.95
	R	0.80	1.00	0.96	0.87	0.95	0.92
	F_1	0.83	1.00	0.98	0.87	0.97	0.93
Outlier	P	0.98	1.00	0.99	0.98	0.99	0.98
	R	0.98	1.00	1.00	0.98	1.00	0.99
	F_1	0.98	1.00	0.99	0.98	0.99	0.99

Table 5.2: Identifying SRAM manufacturer.

In Table 5.3, we presented the classification score for the part number identification, where the 2nd row represents the target part number. Note that, μ^M represents the average classification score over the corresponding manufacturer, and the μ^V columns represents the average classification score over all manufacturers. Similar to Table 5.2, rows 4–9 of table 5.3 represent two sets of P, R, and F_1 scores.

Vendor		CY						IDT							ISSI						AMI				REA					μ^V
Tag		CY1	CY2	CY3	CY4	CY5	μ^M	IDT1	IDT2	IDT3	IDT4	IDT5	IDT6	μ^M	ISSI1	ISSI2	ISSI3	ISSI4	ISSI5	μ^M	AMI1	AMI2	AMI3	μ^M	REA1	REA2	REA3	REA4	μ^M	–
A		0.88	1.00	0.88	1.00	1.00	0.95	0.87	0.87	1.00	0.77	0.90	0.77	0.86	1.00	0.92	0.92	1.00	1.00	0.97	0.73	0.73	1.00	0.82	0.65	0.65	0.85	0.90	0.76	0.88
Target Class	P	0.75	1.00	0.67	1.00	1.00	0.88	1.00	1.00	1.00	0.33	1.00	**0.42**	0.79	1.00	0.80	0.80	1.00	1.00	0.92	**0.56**	1.00	1.00	0.85	**0.33**	**0.33**	1.00	0.71	0.60	0.81
	R	0.60	1.00	0.80	1.00	1.00	0.88	**0.20**	**0.20**	1.00	**0.40**	**0.40**	1.00	0.53	1.00	0.80	0.80	1.00	1.00	0.92	1.00	**0.20**	1.00	0.73	**0.40**	**0.40**	**0.40**	1.00	0.55	0.72
	F_1	0.67	1.00	0.73	1.00	1.00	0.88	0.33	0.33	1.00	0.36	0.57	0.59	**0.53**	1.00	0.80	0.80	1.00	1.00	0.92	0.71	0.33	1.00	0.68	0.36	0.36	0.57	0.83	0.53	0.71
Outlier	P	0.90	1.00	0.95	1.00	1.00	0.97	0.86	0.86	1.00	0.88	0.89	1.00	0.92	1.00	0.95	0.95	1.00	1.00	0.98	1.00	0.71	1.00	0.90	0.79	0.79	0.83	1.00	0.85	0.93
	R	0.95	1.00	0.90	1.00	1.00	0.97	1.00	1.00	1.00	0.84	1.00	0.72	0.93	1.00	0.95	0.95	1.00	1.00	0.98	0.60	1.00	1.00	0.87	0.73	0.73	1.00	0.87	0.83	0.92
	F_1	0.95	1.00	0.92	1.00	1.00	0.97	0.93	0.93	1.00	0.86	0.94	0.84	0.91	1.00	0.95	0.95	1.00	1.00	0.98	0.75	0.83	1.00	0.86	0.76	0.76	0.91	0.93	0.84	0.92

Table 5.3: Identifying SRAM part number.

Unlike manufacturer identification, the part number classification score for some manufactures is not up to the mark; especially, the P or the R (and corresponding F_1 score) scores to identify a few part numbers of IDT, REA, and AMI are unacceptably low (shown in red). Nevertheless, such low test scores can be explained from multiple perspectives. For example, the model used to classify manufacturers trained based on 40–60 samples per class; however, due to the extremely limited number of samples from each part number (10 from each), it is harder to learn part number classifiers. Besides, the differences among a few memory part numbers, especially from IDT, REA, and AMI, are not well-understood from their electrical characteristics mentioned in the datasheets. For example, the only noticeable difference between IDT2 and IDT5 is how they are packed during shipping (tube/tray vs. tape/reel). Hence, these two part numbers might be equivalent based on their electrical characteristics. Similarly, the following pair of the part numbers- (IDT3, IDT6), (REA1, REA2), and (REA3, REA4) do not have any recognisable difference other than their packing method. Hence, to extract the perfect set of features to

differentiate those chips (IDT2 vs. IDT5, IDT3 vs. IDT6, REA1 vs. REA2, and REA3 vs. REA4), we might require more detailed information about the chip characteristics. On the other hand, the IDT1 and IDT4 memory chips are only differed by the temperature grade, and possibly have only difference in their die packaging along with some minor fabrication imperfections [MKD04]. Hence IDT1 and IDT4 may have very subtle differences due to the possible similarity in die architectural, layout, and systematic process variation. We found the similar problem for AMI1 and AMI2, which are also only differed by the temperature grade. Note that, the difference between IDT1 and IDT4 (or, between AMI1 and AMI2) might be still captured by using more train samples.

In Table 5.4, we also presented the summary result (only average test score) by changing the operating temperature of the test samples to $\sim 45°$C. The 2nd and last row of Table 5.4 represents the average score for the manufacturer and part number detection (respectively) from all manufacturers. On the other hand, rows 3–7 represent the average score for part number detection from the corresponding manufacturer and the row 8 represents the average score for part number detection over all manufacturers. From Table 5.2, 5.3, and 5.4, it is apparent that our proposed technique is not very sensitive to temperature. The temperature insensitivity of our selected features is reasonable; previous work shows that varying $+60°$C only changes the SRAM start-up data by $\sim 12\%$ [PSK15].

With the temperature increase, the average test score for manufacturer identification almost retain the same score as of the nominal temperature. However, the average part number identification across all manufacturers is slightly degraded (presented in red in Table 5.4); for example, the F_1 score to identify the target class reduced from 0.71 to 0.67 (presented in cyan in Table 5.4). Especially, the SRAM chips from IDT and REA are affected most while we increased the temperature. For

Classification goal		A	Target Class			Outlier		
			P	R	F_1	P	R	F_1
Manufacturer (μ^V)		0.97	0.93	0.94	0.93	0.99	0.98	0.98
Part number	CY (μ^M)	0.97	0.93	0.96	0.93	0.99	0.97	0.98
	IDT (μ^M)	0.81	**0.54**	**0.47**	**0.43**	0.90	0.88	0.88
	ISSI (μ^M)	0.95	0.93	0.88	0.87	0.97	0.97	0.97
	AMI (μ^M)	0.82	0.76	0.73	0.72	0.89	0.87	0.86
	REA (μ^M)	0.73	**0.58**	**0.55**	**0.48**	0.85	0.78	0.81
	μ^V	**0.86**	**0.74**	0.71	**0.68**	0.92	0.90	0.91

Table 5.4: Mean accuracy, precision, recall, and F_1 score at high temperature

IDT, the average F_1 score for part number identification is reduced by 19% (0.53 to 0.43), and for REA, the F_1 score is degraded by 9%. For IDT and REA, we expected such results as the features associated with those part numbers are very closely distributed (as explained in the previously). Hence, a slight thermal noise on start-up data impacted the corresponding classifiers heavily. Interestingly, the classification score improved by a little margin for AMI, although chips from AMI1 and AMI2 are closely located in feature-space (Figure 5.6e). With closer observation, we found that the features from AMI1 impacted heavily at higher temperatures and shifted away from the AMI2, which provided a relatively better separation between AMI1 and AMI2. The temperature sensitivity of AMI1 is not surprising as AMI1 possesses a lower temperature grade than AMI2.

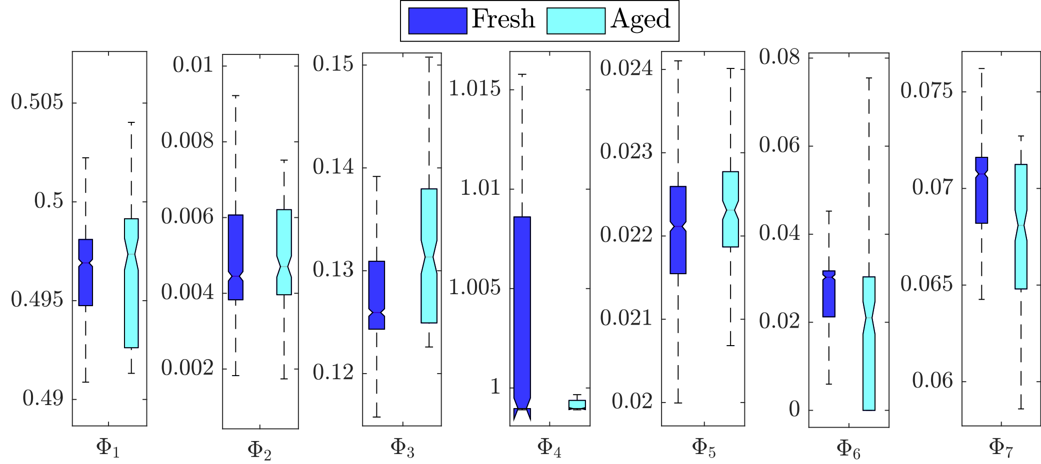
In Table 5.2, 5.3, and 5.4, we trained the classifier using only one entropy source (i.e., all features are extracted from start-up data at nominal voltage). Our proposed technique can be further improved if more features can be extracted from different entropy sources. For example, we collected three sets of start-up data at low voltage (3.0V), nominal voltage (3.3V), and high voltage (3.6V) from all IDT chips. Then, we only extracted feature Φ_1, Φ_4, Φ_6, and Φ_7 from all of those three datasets and concatenated them in a single feature set (total 12 features). We trained ML models from train samples as we did earlier and used the model to identify part numbers

from IDT. The outcome of the experiment was aligned with our expectation; The average F_1 score of part number identification is improved to 0.6 from 0.53 (presented in cyan in Table 5.4). Note that, for most of electronic ICs, the external voltage is down-regulated using a voltage regulator. However, our experiment shows that varying the operating voltage to 3V or 3.6V slightly alters the SRAM startup data ($\sim$5%) (result not included in this manuscript). We also found a similar mismatch ($\sim$3%) on openly accessible SRAM data published by Forte *et al.* [FTGR]. Such slight differences in startup data might be introduced from a subtle shift in core voltage due to the voltage regulator's input-output voltage characteristics.

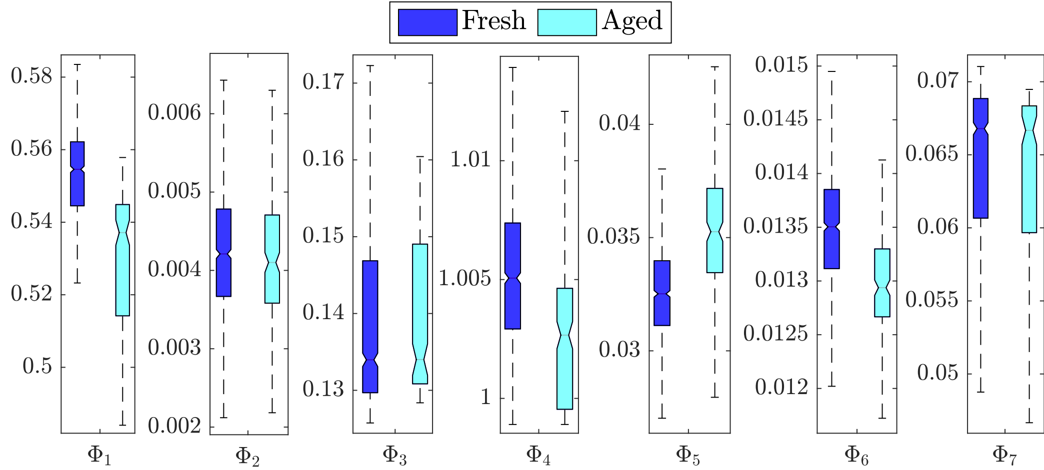
5.4.3 Identifying Recycled Memory Chips

As we explained in Section 3.6, the recycled (aged) and fresh (aged) SRAM chip can be distinguished by only observing the number of 1's in start-up data [GWHS19]. As our method also uses the number of 1's as a feature (Φ_1), our method is more generalized. Moreover, identifying the recycled chips by observing the number of 1's is only possible if the SRAM chips experience more logic "0" than the logic "1" (skewed data distribution). Although such a scenario is practical over the natural usage of the SRAM chips, we conducted an experiment without making the assumption of skewed data distribution.

Our experiment has used the "accelerated aging" [G^+18] method by continuously writing random bits on SRAM chips. In accelerated aging, we exposed the memory chips in high voltage (3.6V) and high temperature (80°C) for 1 hour and continuously wrote different random numbers (with the normal distribution of 0's and 1's). The temperature of the chip was controlled by a thermostream system [The]. We collected start-up data before and after the aging process and extracted



(a) Cypress Semiconductor (CY)



(b) Integrated Device Technology (IDT)

Figure 5.7: Visualizing feature distribution: fresh vs. aged.

features from them. The aging process is time-consuming, and we had limited access to the thermostream system; hence, we were only able to experiment with a limited number of chips. Our experiment used used 2 SRAM chips from each part number of CY and IDT (10 CY chips and 12 IDT chips). Although this small number of chips is not sufficient for the ML algorithm, our experiment demonstrates the impact of the aging process on features that are selected in Section 5.3.1.

We presented the distribution of the features from fresh chips and aged chips in Figure 5.7. Figure 5.7a and 5.7b represent feature distribution for CY and IDT,

respectively. Because of using random numbers (uniform distribution of 0's and 1's) to age the device, we have an unpredictable shift on the Φ_1 (number of 1's) distribution, which is used in previously proposed method to identify recycled SRAM chips [GWHS19]. However, we observed some other features might be extremely useful even with the presence of the uniform data pattern. For example, the distribution of Φ_6 (number of noisy signature bits) always tends to shift towards 0. With sufficient aging, the distribution of Φ_6 from the fresh and aged chips will be completely separable. During the aging process with the random data pattern, the number of 0's or 1's experienced by each memory cell will be a normal distribution. Hence, some of the noisy signature bits (located at distribution tail) will experience more 0's or 1's than others. With the same argument presented in [GWHS19], we can argue that this will bias those noisy signature bits either toward "1" or "0" and reduce the total number of noisy signature bits (see Section 3.6 for details). Note that, even with the biased data pattern (dominate by "0" or "1"), the number of the noisy signature bits will also be reduced (noisy signature bits will achieve either stable "1" or "0").

We also observe a shift in the distribution of other features. For example, now the compression ratio is closer to 1 (distribution of Φ_4). This is also understandable as the random distribution on the data pattern biased the SRAM cells randomly and randomizes the start-up data. However, this distribution might shift upward if the usage data pattern is biased towards either "0" or "1" (i.e., start-up data will have more "1" or "0" after usages). Hence, imposing a boundary condition on Φ_4 distribution might also be helpful to identify recycled SRAMs.

5.4.4 Evaluation Time

Our proposed method is aimed to identify counterfeit memory chips from the consumer end (or at least start-up signature should be collected at consumers' end (See Figure 5.2)). Nevertheless, our proposed method can also be scaled up for bulk testing. A single FPGA or high-speed embedded system can be used to collect and analyze data for bulk testing purposes. The average access time for a Commercial off-the-shelf (COTS) SRAM is $<$15ns/word. Hence the total access time for a 4Mb (256K$\times$16) SRAM is $<$4ms ($\approx$15ns$\times$256K). In our experiment, we collected start-up data 20 times. Additionally, to avoid the discharge inversion effect, the sampling interval of 10s should be more than sufficient [LAWG17]. The inference time of the machine learning model is very negligible compared to the data collection process (order of μs). Hence, the total time required to test an SRAM chips' authenticity is $\sim$3min ($\approx$19$\times$10s+20$\times$4ms), which is the time required for collecting the SRAM start-up data.

5.5 Limitations

Identifying memory manufacturer and part number are useful for identifying many counterfeitings, which might be introduced at a different supply chain stage. However, our feature-based manufacturer and part number detection technique will not be effective for overproduced memory chips introduced by a malicious foundry (i.e., the chip produced beyond the IP owner's consent with original GDSII and package in the same foundry facility). Additionally, our proposed framework will not identify a defective/tampered IC integrated with a fully functional SRAM chip.

5.6 Conclusion

This chips presents a non-invasive and low-cost technique to (i) identify the memory manufacturer and part number and (ii) recycled SRAM chips without requiring any additional hardware. This proposed framework has potential to use for other volatile and nonvolatile memory chips and help stop spreading them in the supply chain. Finally, to train a more practical and accurate ML model, we need more train samples which might require an industry scale setup and crowd-sourcing.

CHAPTER 6

NON-INVASIVE HEURISTIC ATTACK ON mPUF

In Chapter 3, we discussed several deterministic factors that can create a common trait among memory-based signatures if they share the same design specification and manufacturing resources. In Chapter 4 and 6, we extracted a set of features that can provide an excellent visual description. Although those features do not provide any quantitative measurement of those mPUFs' signature, they can be utilized to identify the memory origin. In this chapter, we make a probabilistic estimation of the mPUF outcome and design a heuristic attack on the mPUF key [BTFR21]. The major contributions of this chapter include-

- We briefly discuss the weakness in PUF metrics, which are widely used to quantify the "goodness" of many mPUFs. Without proper attention, an attacker may use such weaknesses to attack mPUFs.
- We demonstrate that if an attacker has access to similar devices to the victim's mPUF device, the attacker can learn the characteristics of the target mPUF and perform a heuristic attack to recover mPUF responses.
- We propose a set of recommendations to secure mPUF.

6.1 Vulnerabilities of mPUFs and weakness of PUF Metrics

In this section, we revisit several aspects of mPUFs overlooked by researchers. From now on, we focus on SRAM PUFs (the most common type of mPUF).

6.1.1 Deterministic Components of mPUF keys

In Chapter 3, we discussed the deterministic impacts of architectural, layout, and process variations on mPUF keys' entropy. Although, a well-designed fuzzy ex-

tractor (see Section 2.4.2) can improve the entropy density, a low-cost error correction scheme may replace it in many low-cost PUF-based devices [KHK+14, HKS20, YD10, YMSD11, SRR16, SD07, CZZ17, YGH18], where the error correction scheme might not maintain the constraint presented by Equation 2.4. In a most simple design, the error corrector does not extract the entropy or increase the entropy density at the output; instead, it only corrects the error of the inputs. Many researchers proposed such error correctors to replace the fuzzy extractor in low-cost applications [KHK+14, SRR16, SD07, CZZ17, YGH18]. In this paper, we assume a simple error correction scheme (i.e., the entropy of the key remains unaffected) instead of a complex fuzzy extractor and explore the impact of both the error correction scheme and the non-uniform distribution of the PUF response. However, in Section 6.4.3, we also discuss the scope and limitation of our proposed attack by considering a fuzzy extractor.

6.1.2 Acceptance of Errors in PUF Protocol

If a q-bit error is allowed in an n-bit response (R), then all n-bit binary strings at a q-bit distance (from R) will be accepted in the key re-generation phase (Figure 2.7b). Now, using the *covering code* theory [Ost91, Zha91], an attacker can reduce the number of possible patterns that cover all possible n-bit strings (i.e., smaller search space to find the valid key). For example, if $n = 16$ and $q = 2$, then only < 960 patterns are sufficient to represent all 2^{16} patterns [Zha91]. Hence, only allowing 2-bit error will reduce the entropy of response (R) from 16 bits to $< \log_2(960) < 10$ bits [B+17]. However, a properly designed fuzzy extractor can eliminate the PUF weakness introduced by the error correction scheme (see Section 2.4.2).

6.1.3 Weakness of PUF Metrics

In general, to get a uniform response from an n-bit PUF, the probability of getting "0" or "1" at any bit location should be $\frac{1}{2}$ (denoted with $p = \frac{1}{2}$). Alternatively, all possible patterns encoded with n-bit should have an equal probability of being a PUF response. If a PUF satisfies this condition, the *Hamming weight* (HW) of all the possible responses should follow a Binomial distribution [KLB95] with a mean ($\mu = np$) and standard deviation ($\sigma = \sqrt{np(1-p)}$) (see Appendix A). If the μ and σ are large enough, such a Binomial distribution can be approximated with a Gaussian distribution of the same mean and standard deviation (i.e., $\mu = np$ and $\sigma = \sqrt{np(1-p)}$) [Boa05]. However, this condition is not checked adequately in many PUF related works [SRR16, HBF07, G+07, BTRFR19, SvdL12], and sometimes, a Gaussian curve is fitted around the distribution of *Hamming weight* with an arbitrarily chosen σ (i.e., only satisfying the condition, $\mu = np$). If the condition of μ or σ is not appropriately maintained, the PUF might produce biased outputs.

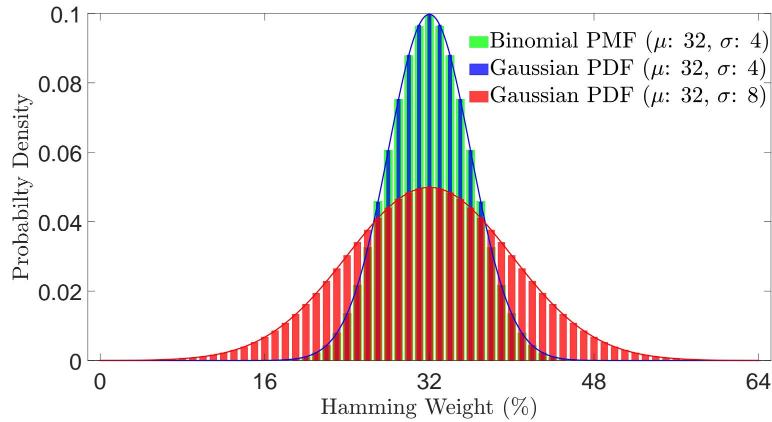


Figure 6.1: Binomial vs. Gaussian distribution ($n = 64$).

In Figure 6.1, we present a comparison between the Binomial distribution and the Gaussian Distribution with $n = 64$. For the Binomial distribution, we assume that all patterns encoded with 64-bit Binomial string have an equal probability of

occurrence (i.e., $p = 0.5$). This Binomial distribution returns a mean, $\mu = np = 32$ and standard deviation, $\sigma = \sqrt{np(1-p)} = 4$ (plotted with green bar). This Binomial distribution can be approximated with a Gaussian curve with the same mean and standard deviation (plotted with blue color). However, if $\sigma \neq \sqrt{np(1-p)}$ and fitted with an arbitrarily chosen σ (plotted in red color), a set of patterns with some specific *Hamming weights* have a higher chance to be the PUF outcome (for this case, pattern from the tail section of the red curve). In such cases, it is much easier to reveal the key by the heuristic analysis of pattern frequency in such instances. Note that the perfect Binomial distribution of *Hamming weight* does not solely guarantee the perfect randomness of CRPs. For example, two different patterns, X_1 and X_2, may share the same *Hamming weight* of h_X; however, a perfect probabilistic value of h_X (drawn from the binomial distribution) does not imply that X_1 and X_2 have an equal probability of being PUF responses. Therefore, we recommend checking the uniformity of pattern distribution on mPUF signature data (see Section 6.4). In summary, the impacts of the *Hamming weight* distribution are listed below:

- For a given unbiased n-bit PUF, all possible n-bit patterns should have a uniform probability to be the PUF outcome. Therefore, the corresponding *Hamming weight* distribution of PUF responses (for all possible challenges) should follow a Binomial distribution (with $\mu = \frac{n}{2}$, and $\sigma = \frac{\sqrt{n}}{2}$). This distribution can be approximated with a Gaussian distribution with the same mean and standard deviation.
- If the *Hamming weight* does not obey the above distribution, the PUF will be more susceptible to a heuristic attack.
- The Gaussian distribution (with $\mu = \frac{n}{2}$, and $\sigma = \frac{\sqrt{n}}{2}$) in *Hamming weight* does not guarantee the uniform probability of all patterns. Hence, we also recommend checking uniformity on pattern distribution.

6.2 Threat Model

In most attacks on weak mPUFs, attackers require physical access to the target PUF device at least once (see Section 2.4.2). However, gaining physical access to most devices is practically difficult (especially, for IoT device authentication). This section discusses a more practical scenario of the attack surface and the attackers' capability.

1. The attacker needs to know the memory model (manufacturer and part number) of the target mPUF, which can be revealed by analyzing the victim's device information, such as the device manufacturer, model/variant, version, etc. They can mostly obtain that information by using existing *detection* techniques [WK12, Eck10] or/and from the victim's browser fingerprints.
2. The attacker can own some mPUFs with the same vendor and part number as the victim's device. We also assume that the attacker can apply any invasive/semi-invasive/non-invasive technique to characterize his acquired mPUFs and learn signature heuristics, which can be used to attack the victim's mPUF.
3. The attacker also knows about the error margin in the key re-generation phase (Figure 2.7b). The previous assumption can relax this assumption. If the attacker has access to similar PUF models, he or she can estimate this error margin. If the ECC encoder/decoder is part of the mPUF devices (Figure 2.7b), the attacker can easily determine the error limit by examining the ECC encoder/decoder circuit. Otherwise, the attacker can estimate the error statistically. For example, if a bunch of SRAM PUF of model "X" produces an average q-bit error between two valid operating conditions, the attacker can assume that the victim's device (of model "X") will also produce on an average q-bit error in those two operating conditions.

4. In an unsuccessful attempt made by the attacker, the degree of mismatch between $K(R_{C^i})$ and $K(R''_{C^i})$ (or between R_{C^i} and R''_{C^i}) is not released in the key regeneration phase (see Figure 2.7b).

The above assumptions are more practical than all previous attacking scenarios. Our proposed attack is non-invasive as we do not require any physical access to the victim's PUF. In Section 6.5, we will discuss a set of countermeasures that can narrow down the attack surface.

6.3 Proposed Method

To recover the signature/key from mPUFs, at first, we discover possible vulnerabilities of mPUF. Second, we propose an attacking scheme on mPUF by exploiting these mPUF vulnerabilities. Finally, we propose a set of techniques to generate secure Keys/signatures. We explain the first and second tasks in the following two subsections and the third task in Section 6.5.

6.3.1 Analyzing PUF Vulnerabilities

Aside from the deterministic impact of architectural, layout, and systematic process variations, we anaylyze following two aspects of mPUFs to assess their security.

Analyzing Pattern Frequency: PUF outputs must not be biased toward a specific pattern. To capture the weakness of mPUF, we analyze the pattern (observed as the PUF outcome) frequency. Primarily, most of the applications use at least a 128-bit or 256-bit key for authentication or cryptographic operation. However, a 64-bit key may also be used in resource-constrained applications. A 128-bit key can form 2^{128} unique patterns (i.e., possible keys). Analyzing such a massive number of patterns is not always feasible due to the limitations of computational resources and

time constraints. Furthermore, analyzing pattern distribution of 2^{128} CRPs requires a large source (memory chip) size ($> 2^{128} \times 128$ bits $= 2^{92}$ Tbyte). which is not viable for commercial off-the-shelf (COTS) memory chips due to the limited address space. Therefore, we consider each PUF response as multi-word data and perceive PUF responses using the 16-bit word characteristics. We computed the relative frequency of the patterns formed by each of the 16-bit words of the memory and quantified the deviance of *Hamming weight* distribution as discussed in Section 6.1.3. For an ideal SRAM-PUF, all patterns should have a uniform probability of occurrence (see Section 6.1.3). However, from frequency analysis, we observe that some patterns have more probability of occurring than others. Note that in our experiment, we use 4-Mbit SRAM chips, where each memory chip provides 2^{18} 16-bit words.

Error Estimation of SRAM Signatures: According to our third assumption (see Section 6.2), the adversary must know the amount of error victim PUF can tolerate (i.e., the capability of the error correction scheme adopted for the victim system). However, the victim system does not reveal such information. To understand the error profile of the victim mPUF, we characterize errors of mPUFs that possess the same part number as the victim one. Error characterization is performed by collecting two set start-up data at two different operating conditions and computing *Hamming distance* between them. Once the error rate is estimated, the number of patterns (i.e., key search-space) can be reduced, as discussed in Section 6.1.2 (for algorithm, see Section 6.3.2).

6.3.2 Attacking an SRAM PUF

In a traditional brute-force attack, an attacker guesses a possible set of keys and continuously tries them, hoping that one combination will match the original key.

In such attacks, attackers do not have any prior knowledge about the key. However, some prior knowledge in the victim's device might provide some insights on pattern heuristics, enabling the attacker to order all possible patterns according to their probability of being matched. In this section, we mainly focus on characterizing frequent patterns and ordering them. From now on, we will frequently use two terms: *target device* and *train devices*. The *target device* is owned by the victim and targeted by the attacker. On the other hand, the *train devices* are possessed by the attacker (assumption 2 in Section 6.2), and they are similar (i.e., same manufacturer and part number) to the *target device*. Our proposed attacking scheme uses *train devices* to learn and characterize the *target device*.

We assume that the PUF response length is 64 ($n = 64$) in the rest of the chapter for simplification. However, a 64-bit binary number can form 2^{64} possible unique patterns, which require, on average, 2^{63} attempts to discover the key using a simple brute-force attack [MGR13]. In our proposed attacking scheme, we aim to reduce the number of attempts (i.e., reducing the search space) by analyzing *train devices* and ordering all possible 64-bit patterns by assigning a relative probability to each pattern.

Most of the SRAM chips use either 16-bit or 8-bit parallel interface. Hence, we need to concatenate 4 (for 16-bit memory) or 8 (for 8-bit memory) addresses to produce a 64-bit key. We use a 4-Mbit 16-bit SRAM chips for our experiment and assume that a 64-bit key is evaluated from 4 consecutive memory addresses. Therefore, we can generate a total of $65,536$ CRPs ($\ll 2^{64}$) from a 4-Mbit mPUF.

Ordering 2^{64} patterns from a 64-bit number is a difficult task. Therefore, we have fragmented the 64-bit key into four 16-bit words ($n' = 16$). In our proposed attacking scheme, we first estimate the error rate in each PUF response from the *train devices* accepted in the key re-generation phase. Then we observe the frequencies of all

16-bit patterns in *train devices*. In the next step, using the estimated error rate and relative frequency information of the 16-bit patterns, we have reduced the number of 16-bit patterns using a custom *covering code* algorithm and ordered them.

We use Algorithm 1 to estimate errors in PUF responses of the *target device*. To estimate the error, we first record all responses (corresponding to all possible challenges) of all *train devices* at two different valid operating conditions (e.g., at different voltage and/or temperature). Next, we compute errors using the *Hamming distance* between these two sets of responses. As the *train devices* are similar to the *target device*, the *train devices'* maximum error is the hard error limit of the *target device* (i.e., an attacker needs to produce a response within that error limit to make a successful attack). However, due to uncertainty associated with the statistical measurement, this assumption might seem too optimistic from the attacker's side (i.e., the same maximum error limit of *train devices* and *target devices*). Hence, to relax the condition, an attacker may choose the ith percentile of the error distribution (distribution of $\mathcal{D}$ from Algorithm 1) and set the value as the estimated maximum error limit of the *target device* (i.e., the value of q).

Algorithm 1: Quantifying error limit during key regeneration/authentication phase.

Data: Two sets of n-bit response data, $\mathcal{R}_{oc1}$ and $\mathcal{R}_{oc2}$ from *train devices* at two valid operating conditions, *op1* and *op2*.
$\mathcal{R}_{oc1} \leftarrow \{R^0_{oc1},\ R^1_{oc1},\ R^2_{oc1},\ \ldots\}$;
$\mathcal{R}_{oc2} \leftarrow \{R^0_{oc2},\ R^1_{oc2},\ R^2_{oc2},\ \ldots\}$;

Result: q, The number of errors accepted out of n-bit response.

```
1 begin
      /* Compute element-wise Hamming distance (d(·)) between R_oc1 and R_oc2 */
2     D ← {D_i | D_i = d(R^i_oc1, R^i_oc2)};
3     q ← max(D);
4 end
```

Algorithm 2: Reducing and ordering n-bit patterns.

```
Data: n: PUF response length; n′: Word length in SRAM memory
      q: Input from Algorithm 1; t: Number of attempts (n-bit pattern)
Result: T: Serialized n-bit patterns; len(T) ≪ 2^n
1  k ← n/n′; // The number of words to represent n-bit response
2  q′ ← q/k; // Allowed bit-error in n′-bit word.
3  X ← {0, 1, ..., 2^{n′} − 1}; // Set of n′-bit words
4  p_{n′} ← [ ]; // List of reduced n′-bit patterns
5  w_p ← [ ]; // Weights of p_{n′}
6  t′ ← ceil(t^{1/k}); // Required number of n′-bit patterns
   /* Compute similarity matrix of all n′-bit patterns */
7  M ← ( d(0,0)              d(0,1)              ···  d(0, b_{2^{n′}−1})
         d(1,0)              d(1,1)              ···  d(1, b_{2^{n′}−1})
         ⋮                   ⋮                   ⋱    ⋮
         d(b_{2^{n′}−1}, 0)  d(b_{2^{n′}−1}, 1)  ···  d(b_{2^{n′}−1}, b_{2^{n′}−1}) );
8  begin
9     B_X ← {b_0, b_1, ..., b_{2^{n′}−1}}; // Histogram of X
10    idx ← argsort(B_X); // Indices that sorts B_X
11    X^s ← X[idx]; // Sorted X
12    B^s_X ← B_X[idx]; // Sorted B_X
13    M^s ← M[idx, :]; // Sorted the row of M
14    M^s ← M^s[:, idx]; // Sorted the column of M
15    M^s_b ← (M^s ≤ q′); // 2^{n′} × 2^{n′} Boolean matrix
16    for ( i ← 0; (len(X^s) > 0) && (i < t′); i ← i + 1 ) {
17        p_{n′}.append(X^s[i]);
18        w ← M^s_b[i, :] • B_X; // Weight of p_{n′}[i]
19        w_p.append(w);
20        j ← args(M^s_b[i, :] == 1); // Pattern indices covered by X^s[i]
21        delete X^s[j], M^s_b[j, :], M^s_b[:, j]; // Remove covered elements
22    }
      /* Permute (w/repetition) indices of p_{n′} vector by taking k elements */
23    idx^p ← perm({0, 1, ..., len(p_{n′}) − 1}, k);
24    p^k_{n′} ← p_{n′}[idx^p]; // Permute p_{n′}
25    w_k ← w_p[idx^p]; // Permute w_p
26    w^p_k ← sum(w_k, row); // Each permutation's weight
27    idx_{p,f} ← argsort(w^p_k); // Indices that sorts w^p_k
28    T ← horstack(p^k_{n′}[idx_{p,f}, :]);
29 end
```

Next, we propose Algorithm 2 to order all n-bit patterns. In our proposed attacking scheme, an attacker uses pre-ordered n-bit patterns to perform a heuristic attack. In this heuristic attack, patterns from the top of the order have a higher probability of succeeding than the bottom of the order. Algorithm 2 generates the most significant unique t attempts of n bits. As we explained earlier (Section 6.3.1), ordering an n-bit pattern is a difficult task; we first order all possible $2^{n'}$ patterns expressed by each n'-bit word. Note that the number of allowed erroneous bits in an n'-bit pattern is denoted by q', and the number of unique n'-bit patterns is denoted by t'. Later, these n'-bit patterns are joined to generate total t unique n-bit patterns.

First, all possible n' bit patterns are listed in a variable X (of length $2^{n'}$). Next, we compute a similarity matrix M of size $2^{n'} \times 2^{n'}$, where $M[i, j]$ is the *Hamming distance* between $X[i]$ and $X[j]$. We compute the histogram corresponding to each pattern in X by observing their frequency in all *train devices* and denoted with B_X (see line 9 of Algorithm 2). Next, the indices that would sort (in descending order) the B_X are computed and saved in idx (line 10). We, then, reorder X, B_X, and M (both row-wise and column-wise) according to idx and store them in X^s, B_X^s, and M^s, respectively. In line 15, we computed a Boolean matrix M_b^s, in which, $M_b^s[i, j] = 1$, if $M^s[i, j] \leq q'$ and 0 otherwise. If $M_b^s[i, j] = 1$, then $X^s[j]$ is covered by $X^s[i]$ within q' bit error. From lines 16 to 22, we computed the reduced set of n'-bit pattern ($p_{n'}$) and corresponding weight (w). In ith iteration of the loop, we appended the most frequent n'-bit pattern ($X^s[i]$) in $p_{n'}$. We compute the weight of $p_{n'}[i]$ by accumulating only those bins in B_X^s that are codependent on the patterns covered by $p_{n'}[i]$ (or $X^s[i]$). In line 20, we compute all pattern indices that are covered by $X^s[i]$, and then we remove corresponding entries from X^s, and rows/columns from M_b^s. At line 23, we compute the special indices idx^p that generate all possible

permutations of the input vector (in this case, indices of $p_{n'}$) by taking k elements (with repetition). A simple Cartesian product is used to do such kind of task. For example, if we consider a set,

$A = \{a_0, a_1, a_2\}$; and corresponding indices,

$idx_A = \{0, 1, 2\}$; then,

$$idx_A^p = perm(idx_A, 2) = idx_A \times idx_A$$
$$= \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2)\}$$
$$A[idx_A^p] = \{(a_0, a_0), (a_0, a_1), (a_0, a_2),$$
$$(a_1, a_1), (a_1, a_2), (a_2, a_2)\}$$

Now, using this idx^p, we compute the corresponding permutation of $p_{n'}^k$ and w_k. We, then, calculate w_k^p by summing the weights from each row of w_k. Next, we compute the indices $idx_{p,f}$ that sorts the w_k^p in descending order. Finally, we sort the row of $p_{n'}^k$ based on the indices $idx_{p,f}$, and horizontally stacked all binary patterns of the row in a single string to form an n-bit pattern (stored in T). The attacker can use this pre-ordered n-bit string in T to perform the attack.

Figure 6.2: Constructing n-bit key from n'-bit word.

Algorithm 2 can be explained using Figure 6.2. We concatenate k n'-bit words from the memory chip to form an n-bit key ($n = k \times n'$). Here we assume that each of the n'-bit words are independent of each other. So, to attempt with t n-bit pattern, we only need the most frequent $t'(= \sqrt[k]{t})$ n'-bit words. Later, these n'-bit words are combined with different permutations to form t n-bit pattern. Then these n-bit patterns are sorted based on their members' (k n'-bit words) accumulated frequency.

6.4 Performance Evaluation of Proposed Attack

We have evaluated the performance of our proposed attack experimentally and compared our result with the traditional brute-force attack. For our evaluation, we collected SRAM start-up data from 139 4-Mbit commercial SRAM memory chips from four major manufacturers. We used these chips as the *train devices* (see Section

Manufacturer¶	Part number	Chip tag	#Chips
ISSI	IS61LV25616AL-10TL	ISSI-A	14
	IS61WV25616BLL-10TLI-TR	ISSI-B	6
	IS61LV25616AL-10TLI	ISSI-C	14
IDT	IDT71V416S10PHG8	IDT-A	7
	IDT71V416S12PHG8	IDT-B	14
	IDT71V416L15PHG8	IDT-C	14
CY	CY7C1041G18-15ZSXI	CY-A	14
	CY62147G30-55ZSXE	CY-B	10
	CY62146EV30LL-45ZSXIT	CY-C	8
AMI	AS7C34098A-10TCN	AMI-A	14
	AS7C34098A-10TIN	AMI-B	14
	AS6C4016-55ZIN	AMI-C	10

Table 6.1: List of SRAM chips used as *train devices*.

6.3.2). We list all of these memory chips in Table 6.1 with the corresponding manufacturer and part number. These chips are also tagged with unique identifier, *Chip tag*, and will be used in the rest of the chapter. To emulate the proposed attack, we collected data from another new chip from each group (not included in Table 6.1) and used them as the *target devices* (see Section 6.3.2).

The data collection process is almost similar that we used in Section 5.4. We collected SRAM start-up data using an Arduino platform [Ard] at a nominal voltage (3.3V) and room temperature (25°C). The effect of noise was minimized by collecting

¶ISSI: Integrated Silicon Solution, Inc.; IDT: Integrated Device Technology; CY: Cypress Semiconductor; AMI: Alliance Memory, Inc. The listed SRAM samples of this table are the subset of the samples described in Section 5.4.

the start-up data 19 times from each chip and combining them using the majority voting technique [BCM12].

6.4.1 Error Characterization of SRAM Signature

According to our threat model (Section 6.2), the adversary needs to know the degree of errors accepted by the PUF protocol (Figure 2.7b) obtained from the *train devices*. The error is estimated using Algorithm 1, where the attacker needs to collect the PUF responses from the *train devices* at a minimum of two different operating conditions. In our experiment, we chose a random subset of the *train devices* at two operating conditions (at room temperature vs. at 45°C). Then we used those CRPs in Algorithm 1 to estimate the errors. Figure 6.3 represents the error characteristic between these two sets of CRPs. The results show that SRAM chips from different manufacturers and part numbers have different levels of error tolerance. For example, the memory signatures from IDT-C are highly error-tolerant (see Figure 6.3); almost 99.9% CRPs can be produced within $\sim$ 11% of errors. On the other hand, CY-A is highly susceptible to noise; only 50% of the CRPs can be reproduced within $\sim$ 11% error limit. To defend the system from various attacks (e.g., replay attack [MS09]), multiple CRPs are used over the device's lifetime. However, increasing the number of usable CRPs also requires increasing the number of acceptable erroneous bits. For example, for CY-A, CY-B, and AMI-C, using 99.9% of the CRPs requires to allow a large number of error bits. On the other hand, for IDT-C, 99.9% of the CRPs can be used by just allowing $<$ 11% errors. For better error estimation, the adversary can always collect start-up data at extreme operating conditions from the *train devices* and observe the noise performance. However, to keep our experiment simple, for Algorithm 2, we assumed 12.5% erroneous bits (1 out of 8 bit) is the

maximum acceptable limit by the PUF protocol, which is reasonable for most of the cases [SvdL12].

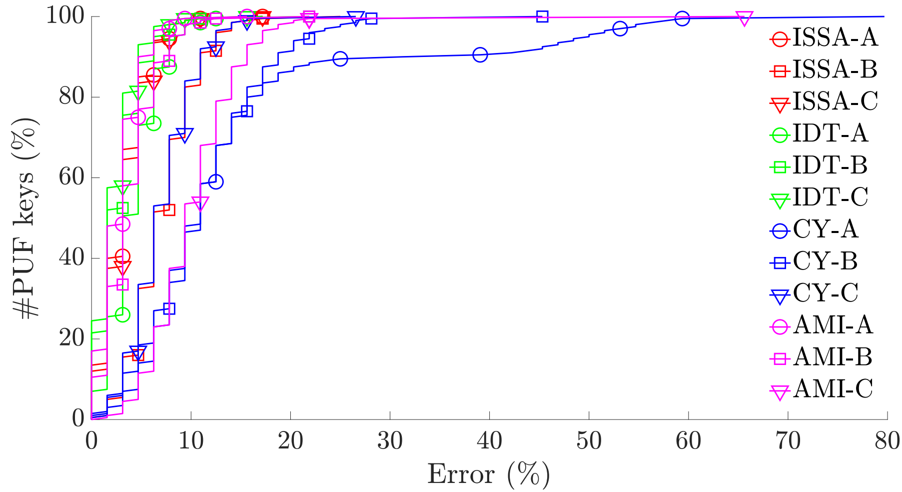


Figure 6.3: SRAM chip error characterization.

6.4.2 Attacking an SRAM PUF

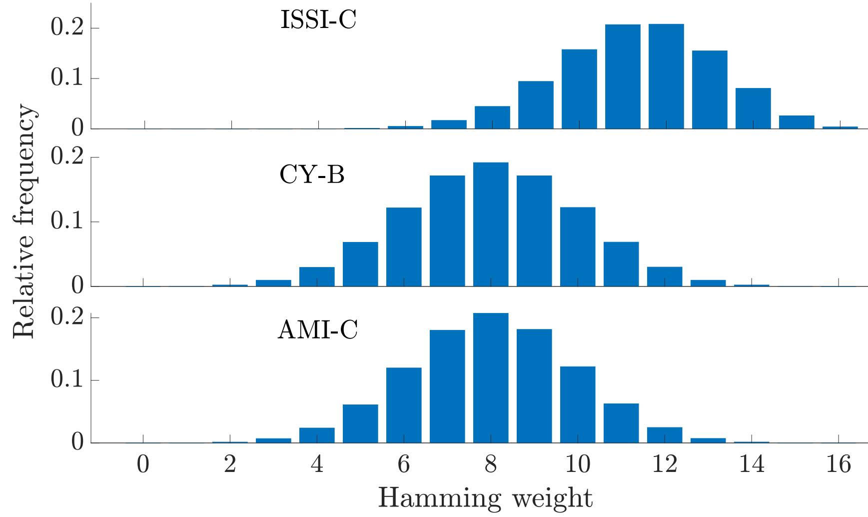
Table 6.2 represents the silicon results of each memory group. In the second and third columns of Table 6.2, the *Hamming weight* distribution of all 16-bit words from all *training devices* is presented. For the uniform possibility of all patterns, the mean and standard deviation of *Hamming weight* should be close to $\frac{16}{2} = 8$ and $\frac{\sqrt{16}}{2} = 2$ (see Section 6.1.3). However, from Table 6.2, we notice that not all SRAM signatures follow these criteria. Moreover, the distribution of *Hamming weight* does not solely guarantee the uniform probability of all patterns (see Section 6.1.3), which is demonstrated in Figure 6.4. Figure 6.4a represents the distribution of the *Hamming weight* for different cases, and in Figure 6.4b, we present the corresponding relative frequency of each 16-bit pattern (pattern 0 to 65535). For ISSI-C, the mean of the *Hamming weight* largely deviates from 8 (Figure 6.4a), and the corresponding relative frequency plot (Figure 6.4b) shows that a few 16-bit patterns have a higher

Chip tag	HW distribution (16-bit patterns)		β_C		#Reduced patterns (16-bit)	CRP_{rec} (%)
	μ	σ	μ	σ		
ISSI-A	10.49	1.90	0.66	0.02	1953	13.06
ISSI-B	7.92	2.03	0.50	0.04	1830	14.99
ISSI-C	11.30	1.83	0.71	0.02	1922	32.54
IDT-A	8.84	2.60	0.55	0.11	1747	8.70
IDT-B	8.96	2.10	0.56	0.03	1877	1.65
IDT-C	8.87	2.57	0.55	0.10	1799	13.77
CY-A	8.01	2.11	0.50	0.02	1694	0.11
CY-B	8.00	2.05	0.50	0.02	1741	0.23
CY-C	5.60	1.93	0.35	0.02	1923	11.51
AMI-A	7.99	2.08	0.50	0.03	1688	0.16
AMI-B	7.99	2.25	0.50	0.05	1740	0.13
AMI-C	8.01	1.92	0.50	0.02	1934	44.33

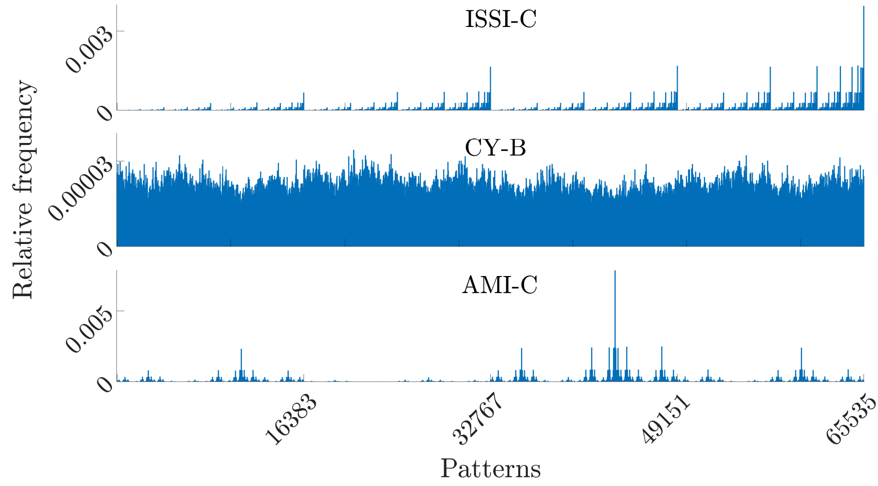
Table 6.2: 16-bit pattern characteristics and vulnerability to heuristic attack.

probability of appearing than others. On the other hand, for CY-B and AMI-C, the mean and standard deviation of the *Hamming weight* are close to 8 and 2 (Figure 6.4a), respectively. This also indicates that the chips from CY-B and AMI-C are not biased toward “0” or “1”. Such unbiased property is also clearly visible from Figure 6.5, which presents the spatial distribution of 0’s and 1’s from CY-B and AMI-C using consecutive 1KByte memory space. Despite both CY-B and AMI-C exhibit almost an ideal *Hamming weight* distribution, some patterns from AMI-C have more probability of occurring than others (Figure 6.4b). Hence, to ensure a uniform probability of each pattern, we recommend checking actual pattern distribution rather than focusing on *Hamming weight* distribution (see Section 6.1.3).

Columns 4 and 5 of Table 6.2 represent the mean and standard deviation of β_C (Equation 2.1 and 2.2) computed from all possible challenges, C. For most memory groups, the mean value is ~ 0.5, with a very small standard deviation. However, for ISSI-A, ISSI-C, and CY-C, the mean value of β_C largely deviates from 0.5, which can be explained by the *Hamming weight* distribution. For example, the

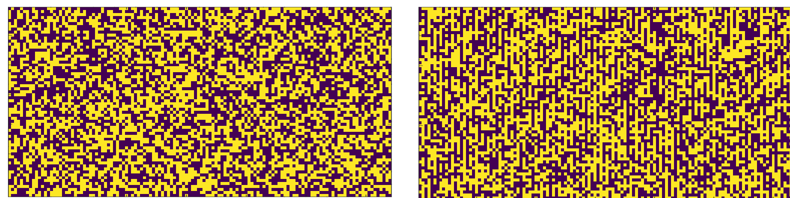


(a) *Hamming weights* distribution of 16-bit patterns



(b) Frequency histogram of corresponding 16-bit patterns

Figure 6.4: Visualizing 16-bit pattern distribution.



(a) CY-B (b) AMI-C

Figure 6.5: Spatial distribution of 0's and 1's.

mean *Hamming weight* of 16-bit patterns in ISSI-A is 10.49, i.e., the probability of getting "1" in such device is $\frac{10.49}{16} \approx 0.66$, which is exactly the mean value of β_C. This unique correlation of *Hamming weight* distribution and *bit-aliasing* is true for all SRAM groups presented in Table 6.2. Such a correlation is expected as the *Hamming weight* measures bias towards a specific logic ("0" or "1") and *bit-aliasing* measure the probability of getting "1" (or "0") from a given bit location. If the distribution of "1" (or "0") is uniform across the whole memory chip and follows the same distribution over all chips, then the probability value calculated from the *Hamming weight* and the *bit-aliasing* should be the same.

Column 6 of Table 6.2 presents the reduced number of 16-bit patterns covering all 16-bit patterns within a 12.5% error limit. Note that the number of patterns is higher than the range specified in [Zha91] (< 960). Because, in our Algorithm 2, we emphasize maximizing the probability of success for each attempt instead of ensuring the smallest covering set in the proposed heuristic attack. However, as only < 2000 patterns cover all of the 16-bit patterns, the entropy of each 16-bit word reduces to $< \log_2(2000) < 11$. Hence, the allowance of 12.5% of errors (i.e., 2 out of 16-bit) reduces the entropy by > 5 bits.

Column 7 of Table 6.2 presents the percentage of recovered CRPs using Algorithm 2 with 10 million (10×10^6) attempts. To generate these 10 million 64-bit unique patterns, we only use the most frequent 57 ($\approx \sqrt[k]{t} = \sqrt[4]{10 \times 10^6}$) 16-bit words. In each attempt, we use a different 64-bit pattern. For each group of memory, the number of attempts vs. the percentage of recovered CRPs is presented in Figure 6.6. From Table 6.2 (column 7) and Figure 6.6, we can divide our results into the following three categories.

1. **Category–1:** Chips from this category (IDT-B, CY-A, CY-B, AMI-A, AMI-B) are less vulnerable to our heuristic attack (success rate is $< 2\%$). Chips for this

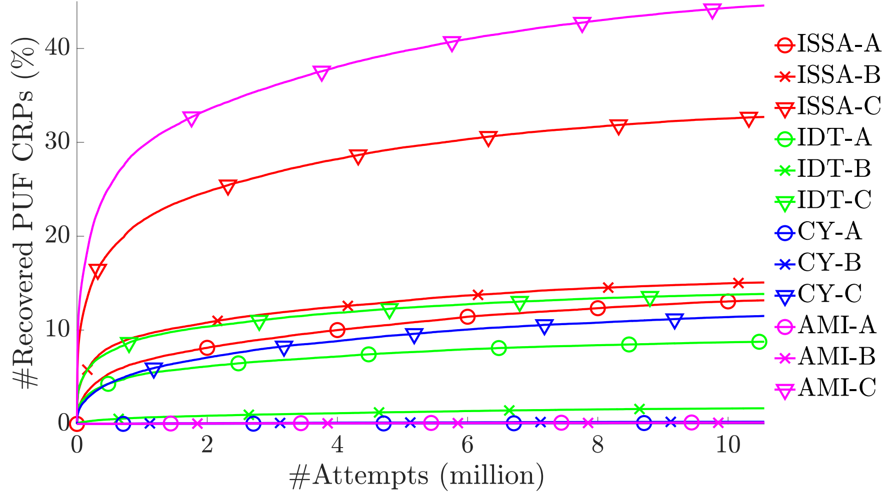


Figure 6.6: Number of attempts vs. recovered CRPs.

category possess almost a perfect distribution of Hamming weight (see columns 2 and 3 of Table 6.2; $\mu \approx \frac{16}{2} = 8$ and $\sigma \approx \frac{\sqrt{16}}{2} = 2$) and maintain a nearly uniform distribution on their 16-bit pattern (e.g., CY-B in Figure 6.4b).

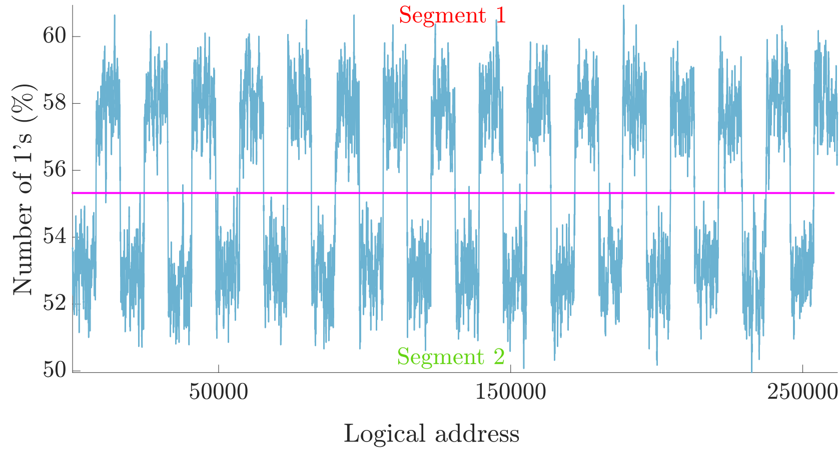
2. **Category–2:** Using our proposed attacking algorithm, we revealed 8–15% CRPs from this category (ISSI-A, ISSI-B, IDT-A, IDT-C, CY-C). For chips from ISSI-A and CY-C, the consequence is directly understandable from the skewed *Hamming weight* distribution of the 16-bit pattern (μ deviated from the ideal value of 8; see columns 2 of Table 6.2). However, although the chips from ISSI-B, IDT-A, IDT-C follows a nearly perfect Hamming weight distribution, 16-bit patterns of those chips deviate from the uniform distribution (not shown in Figure 6.4b).

3. **Category–3:** Chips from this category are highly vulnerable to our proposed attack (ISSI-C and AMI-C). For those chips, we recovered 32.54% and 44.33% CRPs, respectively. ISSI-C is highly skewed from the *Hamming weight* distribution of the 16-bit pattern ($\mu = 11.30$, see columns 2 of Table 6.2). Although, the vulnerability of AMI-C is not visible from the Hamming weight distribution ($\mu \approx 8$ and $\sigma \approx 2$), the actual pattern distribution (Figure 6.4b) can capture

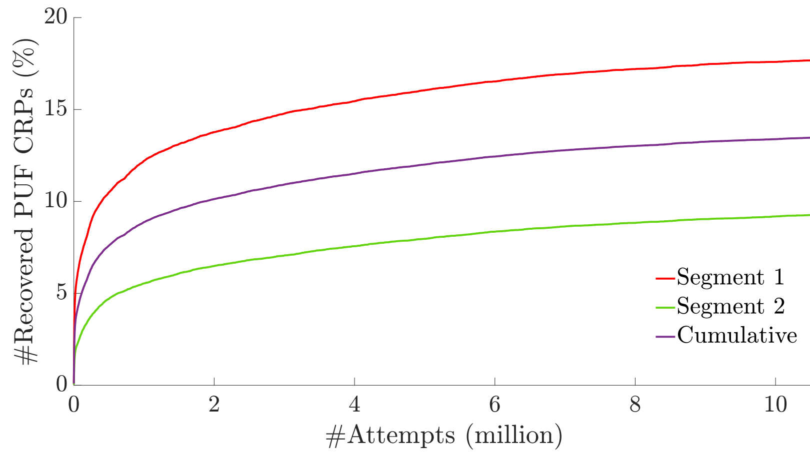
it. For AMI-C, some 16-bits patterns are highly probable than others, making it easier to guess the possible value of the PUF outcome (see Figure 6.4b).

Note that using our algorithm, we are able to recover, on average, $\sim 12\%$ of the CRPs by using only 10 million attempts (i.e., we only used $< 10^{-10}\%$ of all possible 64-bit patterns). As these 64-bit patterns are pre-computed and pre-ordered, the computational overhead of our proposed attack is negligible. However, in an ideal scenario (i.e., with equal probability of all 64-bit patterns and 0-bit error tolerance), finding a single CRP with a traditional brute-force attack is very low ($\approx 5.72 \times 10^{-11}\%$, see Appendix B).

Our attacking algorithm can be further improved by extracting some spatial locality information from the *train devices*. In Section 5.4.1, we presented that the SRAM start-up data characteristics are largely correlated with the features extracted from spatial locality information. However, those features do not provide any quantitative measurement of signature correlation, and it is quite challenging to uncover detailed spatial locality information from the start-up data due to the internal data and address scrambling [vdGS02, EP17]. Nonetheless, we observe some periodic relationship between the start-up data and the logical address of the memory (especially, SRAM chips manufactured by the IDT). For example, Figure 6.7a shows a particular case for IDT-C SRAM memory. Along the horizontal axis, we present the logical address of the 4-Mbit memory, and along the vertical axis, the percentage of logic “1” from the next consecutive 256 words. This plot shows that the number of 1’s in this memory has a periodic nature (with a period of 16,384), which might come from several sources. Instead of complete randomization of address and data, the manufacturer may choose to quantize the whole memory in small segments and scramble those segments only can be one of the reasons. To improve our attacking scheme, we divide memory chips into two segments by drawing



(a) Start-up data dependency on logical address.



(b) Number of attempts vs. recovered CRPs from different segments of the memory.

Figure 6.7: Dependency of pattern heuristics with respect to logical address.

a magenta straight line (Figure 6.7a). The logical addresses above the straight-line are grouped in segment 1, and under the straight-line are grouped in segment 2. Then, we applied the same Algorithm 2 to attack CRPs from those two segments independently (on the *target device*), and the corresponding result is shown in the 6.7b. Figure 6.7b shows that a higher percentage of CRPs from segment 1 than segment 2 is recovered with the same number of attempts (17.59% of CRPs from segment 1 (red curve), 9.18% from segment 2 (green curve), and 13.39% on average

(violet curve)). This result is expected, as the mean *Hamming weight* of patterns from segment 1 largely deviates from the ideal, i.e., 50%.

6.4.3 Attacking in the Presence of a Fuzzy Extractor

As we discussed in Section 2.4.2, in many PUF applications, a fuzzy extractor is frequently used to correct PUF output and enhance security. To guarantee security, the fuzzy extractor must satisfy the condition provided by Equation 2.4. The fuzzy min-entropy from each group of SRAM chips is shown in Table 6.3 (calculated using a *training device*). Although fuzzy extractors constructed with fuzzy min-entropy can enhance the PUF security, the security of the fuzzy extractor still might not be satisfactory. We mentioned in Section 2.4.2 that it is hard to quantify the probability associated with each input pattern of the fuzzy extractor from the experimental data. Such imperfection in statistical estimation arises from the uncertainty associated with the experiment. For example, for CY-B, a 16-bit input of a fuzzy extractor should produce ∼8-bit long outputs. To produce a 64-bit key, we require at least eight 16-bit words from the CY-B SRAM chip (i.e., total 128-bit input). Now, to achieve the same level of success as presented in Table 6.2 (0.23%), we require all possible combinations of the same $57 (= t')$ 16-bit words (as explained in Section 6.4.2) to produce 128-bit input, where $k = \frac{128}{16} = 8$ (see Algorithm 2). So, even after using a properly designed fuzzy extractor for CY-B, a total of $t = (t')^k = 57^8 \approx 111$ trillion 128-bit raw input patterns would be sufficient to recover 0.23% of 64-bit CRPs. However, Equation B.3 (Appendix B) shows that 111 trillion trials would recover 0.0009% of 64-bit CRPs, while the PUF outcome is completely random. That is, even by following the recommended boundary condition of the fuzzy extractor; it might not be possible to achieve the target level of security.

Chip tag	ISSI-A	ISSI-B	ISSI-C	IDT-A	IDT-B	IDT-C	CY-A	CY-B	CY-C	AMI-A	AMI-B	AMI-C
$H_{\infty}^{fuzzy}(R)$	4.36	4.59	3.18	5.09	6.46	5.18	8.24	8.19	4.42	8.41	7.48	2.78

Table 6.3: Fuzzy min-entropy for SRAM chips;
given that, $length(R) = 16$ and $q = 2$.

6.5 Mitigation

We propose several countermeasures against the proposed heuristic attack in order to generate secure and robust signatures from SRAM chips.

6.5.1 Avoiding Error Correction Scheme

We have demonstrated that using the error correction scheme reduces entropy. Avoiding error correction schemes in PUF protocol narrows down the attack surface in two ways: i) attackers require to guess the CRPs with an exact match, ii) it prevents the attacker from reducing the number of patterns as we proposed in Algorithm 2. Researchers have proposed several robust SRAM cell design techniques (e.g., [JG15]) and selection techniques (e.g., [XRF$^+$14]) to avoid error correction schemes. Unfortunately, the robust SRAM cell design methods usually require a modified mask layout and might require satisfying additional design rules; whereas, the robust SRAM cell selection technique might not be suitable in the presence of memory address/data scrambling [vdGS02, EP17]. Moreover, they may still require error correction code; however, applying such methods requires correcting a smaller number of erroneous bits, significantly reducing the attack surface.

6.5.2 Using Properly Designed Fuzzy extractor

Although fuzzy extractors are popularly used for correcting errors and improving security, many researchers proposed simpler versions of fuzzy extractors that use a fixed-length input to produce the fixed-length output [KHK+14, HKS20, YD10, YMSD11], regardless of the fuzzy min-entropy (Equation 2.4). However, based on our findings, we recommend using a fuzzy extractor that strictly follows the fuzzy min-entropy boundary condition. Additionally, we also recommend using a conservative probabilistic estimation of each input pattern of the fuzzy extractor. For example, one can use the upper bound of 99% confidence interval of estimated probability to avoid the uncertainty associated with the estimation. However, such pessimistic estimation might reduce the number of CRP drastically; in the worst case, it might not be possible to generate any CRP from a given type of chip. Moreover, our research suggests that memory chips manufactured with different specification-set might produce different signature statistics. Therefore, the estimated probability of input patterns will be altered if the memory component is substituted. Hence, a fuzzy extractor designed for one device might not be reused on another device (non-reusable hardware IP). Such one-time used hardware IPs are extremely cost-inefficient and not suitable for low-cost devices. In addition to the above limitations, fuzzy-extractors are vulnerable to side-channel attacks [MSSS11].

6.5.3 Pattern Distribution-aware CRPs

In Section 6.1.3 and Section 6.4.2, we emphasize uniform distribution of PUF outcome (response) for all possible challenges (address) to maximize the entropy of the PUF output. The uniform distribution of patterns prevents the attacker from ordering patterns, as we proposed in Algorithm 2. For example, in Figure 6.4, the

16-bit pattern distribution for CY-B is more uniform than ISSI-C and AMI-C. As a result, Table 6.2 shows that CY-B is significantly less vulnerable to our proposed attack than the ISSI-C or AMI-C. Hence, we recommend choosing a subset of SRAM addresses (challenges) at the key enrollment/generation phase to make a uniform distribution of all possible responses (i.e., addresses with highly repeated patterns should be discarded).

6.5.4 Locality-aware CRPs

In Section 6.4.2, we noticed that PUF responses generated from some specific logical address segments might be more vulnerable to other segments (Figure 6.7). So, choosing a less vulnerable logical address segment to generate CRPs might help avoid the proposed heuristic attack. However, if the attacker can reveal the SRAM layout by reverse engineering [BFL+93, M+08, QCF+16, Kum00], this task might become more challenging. Such knowledge of chip layout can enable the attacker to redesign Algorithm 2 more efficiently. Hence, for critical applications, the SRAM vendor is responsible for providing a guideline on PUF usage as they have the precise knowledge of SRAM physical layout.

6.6 Conclusion

In this paper, we used the vulnerabilities of SRAM PUF and the weakness of existing metrics to mount a non-invasive attack. Traditionally, the mPUF response is believed to be a direct consequence of random process variation. However, we demonstrated that memory chips sharing the same specification and manufacturing resources might have a common train on their pattern distribution. Furthermore, we also demonstrated that the error correction scheme might severely weaken the

PUF security, and the traditional *Hamming weight*-based metric of PUF "goodness" does not ensure the PUF quality. If we do not consider these factors carefully, the unclonable PUF will not be secure anymore.

To avoid such an attack, we emphasize reducing error correction code usage, checking the pattern uniformity on memory-based signatures, and locality aware challenge-response pairs (CRPs). Although our proposed countermeasures might reduce the total number of CRPs, we believe that such techniques will significantly improve the PUF security.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this Dissertation, we have demonstrated that, besides random variation in the manufacturer process, few other factors can produce a deterministic impact on mPUFs' signature. Subsequently, we analyzed common traits on mPUFs' signature and proposed a novel method to identify memory manufacturer and specification, which can effectively prevent memory counterfeiting. Furthermore, we showed that those common traits on mPUFs' signature could be used as side-channel information, which an attacker can use to design a heuristic attack on mPUF key.

Our proposed anti-counterfeiting solution has almost similar effectiveness as the mPUF-based solution (except identifying overproduced chips). The additional benefits of our proposed technique are- it does not require any golden database (server), free from complex authentication protocol, and does not require any exhaustive registration process for each chip. To demonstrate our method we used commercial off-the-shelf DRAM modules and SRAM chips. However, the challenges associated with each memory component are unique.

For identifying DRAM origin (i.e., manufacturer and part number), we have 26 different features by collecting latency-based signature data. We have demonstrated that our feature is capable of separating memory modules with high confidence even with a very small set of training samples. Our extracted features are very simple to compute and do not require any special knowledge (e.g., manufacturing process, architecture, chip packaging, etc.). Furthermore, we proposed a one-class classifier-based authentication protocol, which can be efficiently used to identify counterfeit DRAM chips from both the user and manufacturer end.

In our work on SRAM origin detection, we proposed an improved authentication protocol than our previous work. In this work, we used a two-step approach

for identifying the SRAM manufacturer and specification. We observed that memory chips produced by the different manufacturers have larger differences in their features than the memory chips from the same manufacturer (with different part numbers). However, due to the wide diversity of negative samples (i.e., not belong to the target manufacturer), it is not realistic to train a solo binary classifier to identify the manufacturer. Hence, we proposed a one-class classifier for manufacturer identification, which might be further assisted by a binary classifier. On the other hand, we used a multi-class or binary classifier for part number identification. We can safely assume that the manufacturer has access to the samples from all of its part numbers. In contrast to DRAM chips, a typical SRAM chip contains a much smaller memory array; hence, it is difficult to capture all spatial information and produce accurate statistics. However, using a small set of SRAM samples, we were still able to identify SRAM origin with a reasonable test score. The test score can be further improved by adding more samples to the training data.

In our last work on analyzing the vulnerability of mPUF keys, we showed that an attacker could learn the mPUFs signature characteristics if he has access mPUFs samples similar to the victim's one. Then the attacker used this information to sort the possible values of the mPUF key and attack the victim's mPUF. However, we proposed a set of recommendations to reduce the attack surface- (**i**) reducing the number of correctable error bits, (**ii**) designing a fuzzy extractor with a conservative probabilistic estimation of fuzzy extractor inputs, (**iii**) using pattern distribution aware CRPs, and (**iv**) using a subset of address-space with the less spatial locality.

Hereafter, we present the future direction of our research work-

- Our proposed technique of identifying memory origin is more suitable for testing consumer electronics from the user end. However, this technique can be scaled up and fine-tuned for industry applications by selecting a more appropriate set of

features. For example, prior knowledge of architecture may help extract a better set of features and improve memory identification accuracy. However, gathering such knowledge is challenging, as it requires reverse engineering or industry collaboration. Additionally, for some cases, the evaluation time is a little longer than expected (e.g., 3 minutes for identifying SRAM origin); however, the evaluation can be further reduced by introducing different features that do not require multiple measurements.

- Our technique of identifying memory origin can possibly be extended for many different types of memory chips (for example, Flash chips, Resistive RAM, Magnetoresistive RAM, Phase-change memory, etc.). For example, recently, Sakib *et al.* also proposed a similar technique to identify flash memory origin by manipulating erase time [SMR21]. Hence, we believe that our technique can also be applicable for other memory types by choosing a suitable set of signature data, which can be another direction of our research.
- Our proposed attacking scheme on mPUF key can be further sophisticated by revealing the chip layout of the victim's mPUF. Invasive or semi-invasive methods can be used to reverse engineer the chip layout [BFL+93, M+08, QCF+16, Kum00]. Reverse engineering of chips allows the attacker to extract exact architectural and layout design and enable the attacker to compute spatial distribution resulting from the architectural layout. Furthermore, the deterministic components of the process variations can also be reconstructed to some extent [MJ06, AAT14]; however, learning the exact spatial distribution of process variation for a memory chip is difficult due to internal address randomization [vdGS02] and data scrambling [EP17]. Nonetheless, using these spatial distributions, the attacker can form multiple groups of CRPs based on spatial location and generate independent sets of ordered patterns (as described in Algorithm 2) for each group. We expect

that such techniques will improve our proposed attacking scheme, and we will investigate that experimentally in our future work.

BIBLIOGRAPHY

[AAF+18] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Yufan Fan, Christian Hatzfeld, André Schaller, Wenjie Xiong, Manishkumar Jain, Muhammad Umair Saleem, Jan Lotichius, Sebastian Gabmeyer, et al. Intrinsic run-time row hammer PUFs: Leveraging the row hammer effect for run-time cryptography and improved security. *Cryptography*, 2(3):13, 2018.

[AAR+18] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Markus Rosenstihl, André Schaller, Sebastian Gabmeyer, and Stefan Katzenbeisser. Low-temperature data remanence attacks against intrinsic SRAM PUFs. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 581–585. IEEE, 2018.

[AAT14] Aditya Agrawal, Amin Ansari, and Josep Torrellas. Mosaic: Exploiting the spatial locality of process variation to reduce refresh energy in on-chip eDRAM modules. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 84–95. IEEE, 2014.

[ABK16] Giorgos Apostolidis, Dimitrios Balobas, and Nikos Konofaos. Design and simulation of 6T SRAM cell architectures in 32nm technology. *Journal of Engineering Science and Technology Review*, 9(5):145–149, 2016.

[ABN+16] Ali Ahmadi, Mohammad-Mahdi Bidmeshki, Amit Nahar, Bob Orr, Michael Pas, and Yiorgos Makris. A machine learning approach to fab-of-origin attestation. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6, 2016.

[AEM18] Nail Etkin Can Akkaya, Burak Erbagci, and Ken Mai. Secure chip odometers using intentional controlled aging. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 111–117, 2018.

[AK07] Yousra M. Alkabani and Farinaz Koushanfar. Active hardware metering for intellectual property protection and security. In *16th USENIX Security Symposium (USENIX Security 07)*, Boston, MA, August 2007. USENIX Association.

[AKJ+13] Vivek Asthana, Malathi Kar, Jean Jimenez, Jean-Philippe Noel, Sebastien Haendler, and Philippe Galy. Circuit optimization of 4T, 6T, 8T, 10T SRAM bitcells in 28nm UTBB FD-SOI technology using back-gate bias control. In *2013 Proceedings of the ESSCIRC (ESSCIRC)*, pages 415–418, 2013.

[Ard] Arduino. Arduino Due. Available: `https://store-usa.arduino.cc/products/arduino-due`. Accessed: 16 Sep 2021.

[B+17] William Burr et al. Electronic authentication guideline (sp 800-63-2). *National Institute of Standards and Technology (NIST)*, June 2017.

[BA00] Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10):2385–2404, 2000.

[BB15] Abhishek Basak and Swarup Bhunia. P-Val: antifuse-based package-level defense against counterfeit ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(7):1067–1078, 2015.

[BCM12] Mudit Bhargava, Cagla Cakir, and Ken Mai. Reliability enhancement of bi-stable PUFs in 65nm bulk CMOS. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 25–30. IEEE, 2012.

[BFL+93] Simon Blythe, Beatrice Fraboni, Sanjay Lall, Haroon Ahmed, and Ugo de Riu. Layout reconstruction of complex silicon chips. *IEEE journal of solid-state circuits*, 28(2):138–145, 1993.

[BH14] Todd Bauer and Jason Hamlet. Physical unclonable functions: A primer. *IEEE Security Privacy*, 12(6):97–101, 2014.

[BO16] Randal E Bryant and David Richard O'Hallaron. *Computer systems: a programmer's perspective*, volume 2. Boston: Pearson Education, 2016.

[Boa05] Mary L. Boas. *Mathematical Methods in the Physical Sciences*, page 762. John Wiley & Sons, Inc., third edition, jul 2005.

[Boy04] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the ACM conference on Comp. and commu. security*, page 82–91, 2004.

[BPS+17] Nicholas Boynton, Andrew Pomerene, Andrew Starbuck, Anthony Lentine, and Christopher T DeRose. Characterization of systematic process variation in a silicon photonic platform. In *2017 IEEE Optical Interconnects Conference (OI)*, pages 11–12. IEEE, 2017.

[BTFR21] B. M. S. Bahar Talukder, Farah Ferdaus, and Md Tauhidur Rahman. Memory-based PUFs are vulnerable as well: A non-invasive attack against SRAM PUFs. *IEEE Transactions on Information Forensics and Security*, 16:4035–4049, 2021.

[BTMR+20] B. M. S. Bahar Talukder, Vineetha Menon, Biswajit Ray, Tempestt Neal, and Md Tauhidur Rahman. Towards the avoidance of counterfeit memory: Identifying the DRAM origin. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 111–121, 2020.

[BTRFR19] B. M. S. Bahar Talukder, Biswajit Ray, Domenic Forte, and Md Tauhidur Rahman. PreLatPUF: Exploiting DRAM latency variations for generating robust device signatures. *IEEE Access*, 7:81106–81120, 2019.

[CDHS12] Mafalda Cortez, Apurva Dargar, Said Hamdioui, and Geert-Jan Schrijen. Modeling SRAM start-up behavior for physical unclonable functions. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, 2012.

[CFP+16] Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam Smith. Reusable fuzzy extractors for low-entropy distributions. In *Advances in Cryptology – EUROCRYPT*, pages 117–146, 2016.

[CG15] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[CGS+11] Lerong Cheng, Puneet Gupta, Costas J Spanos, Kun Qian, and Lei He. Physically justifiable die-level modeling of spatial variation in view of systematic across wafer variability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(3):388–401, 2011.

[CGW+14] Karthik Chandrasekar, Sven Goossens, Christian Weis, Martijn Koedam, Benny Akesson, Norbert Wehn, and Kees Goossens. Exploiting expendable process-margins in DRAMs for run-time performance optimization. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2014.

[Cha07] Li-Pin Chang. On efficient wear leveling for large-scale flash-memory storage systems. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 1126–1130, 2007.

[CKH+16] Kevin K Chang, Abhijith Kashyap, Hasan Hassan, Saugata Ghose, Kevin Hsieh, Donghyuk Lee, Tianshi Li, Gennady Pekhimenko, Samira Khan, and Onur Mutlu. Understanding latency variation in modern DRAM chips: Experimental characterization, analysis, and optimization. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pages 323–336, 2016.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[CLL13] Wei-Cheng Chang, Ching-Pei Lee, and Chih-Jen Lin. A revisit to support vector data description. *Dept. Comput. Sci., Nat. Taiwan Univ., Taipei, Taiwan, Tech. Rep*, 2013.

[CMK+07] Ching-Te Chuang, Saibal Mukhopadhyay, Jae-Joon Kim, Keunwoo Kim, and Rahul Rao. High-performance SRAM in nanoscale CMOS: Design challenges and techniques. In *2007 IEEE International Workshop on Memory Technology, Design and Testing*, pages 4–12, 2007.

[CMN+08] Leland Chang, Robert K. Montoye, Yutaka Nakamura, Kevin A. Batson, Richard J. Eickemeyer, Robert H. Dennard, Wilfried Haensch, and Damir Jamsek. An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches. *IEEE Journal of Solid-State Circuits*, 43(4):956–963, 2008.

[CPBBFR15] Enrique Castillo, Diego Peteiro-Barral, Bertha Guijarro Berdiñas, and Oscar Fontenla-Romero. Distributed one-class support vector machine. *International journal of neural systems*, 25(07):1550029, 2015.

[CRT13] Gustavo K Contreras, Md Tauhidur Rahman, and Mohammad Tehranipoor. Secure split-test for preventing ic piracy by untrusted foundry and assembly. In *2013 IEEE International symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFTS)*, pages 196–203. IEEE, 2013.

[CSP15] Wenjie Che, Fareena Saqib, and Jim Plusquellic. PUF-based authentication. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 337–344, 2015.

[CYG+17] Kevin K Chang, A Giray Yağlıkçı, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu. Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–42, 2017.

[CZZ17] Chip-Hong Chang, Yue Zheng, and Le Zhang. A retrospective and a look forward: Fifteen years of physical unclonable function advancement. *IEEE Circuits and Systems Magazine*, 17(3):32–62, 2017.

[DA+18] Seelam V Sai Viswanada Prabhu Deva, Shyam Akashe, et al. Reliability analysis of comparator: Nbti, pbti, hci, ageing. In *International Conference on Communication, Networks and Computing*, pages 606–619. Springer, 2018.

[Dep11] Department of Defense Supply chain. The committee's investigation into counterfeit electronic parts in the department of defense supply chain. Available: `https://www.govinfo.gov/content/pkg/CHRG-112shrg72702/html/CHRG-112shrg72702.htm`, November 2011. Accessed: 16 Sep 2021.

[DG] Peter Deutsch and Jean-Loup Gailly. ZLIB compressed data format specification version 3.3. Available: `https://www.ietf.org/rfc/rfc1950.txt`. Accessed: 16 Sep 2021.

[DGV+16] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day Mandel Yu. Efficient fuzzy extraction of PUF-induced secrets: Theory and applications. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 412–431. Springer, 2016.

[Die02] Thomas G. Dietterichl. Ensemble learning. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 405–408. MIT Press, 2002.

[DIG] DIGITIMES. Digitimes Research: Global 4Q13 DRAM capacity to reduce 5.7% on SK Hynix China plant fire. Available: `http://digitimes.com:8080/news/a20131016PD209.html`. Accessed: 16 Sep 2021.

[EBM12] Kaoutar Elkhiyaoui, Erik-Oliver Blass, and Refik Molva. Checker: On-site checking in RFID-based supply chains. In *Proceedings of the fifth ACM conference on security and privacy in wireless and mobile networks*, pages 173–184, 2012.

[Eck10] Peter Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies*, pages 1–18, 2010.

[EP17] Peter Ehlig and Salvatore Pezzino. Error detection in SRAM. Available: `https://www.ti.com/lit/an/spracc0a/spracc0a.pdf`, Nov 2017. Accessed: 16 Sep 2021.

[Eve] Everspin Technologies. Spin-transfer torque MRAM technology. Available: `https://www.everspin.com/spin-transfer-torque-mram-technology`. Accessed: 16 Sep 2021.

[FC18] Domenic Forte and Rajat Subhra Chakraborty. Counterfeit integrated circuits: Threats, detection, and avoidance. In *Conference on Cryptographic Hardware and Embedded Systems*, 2018.

[Fie17] Moritz Fieback. DRAM reliability: Aging analysis and reliability prediction model. *Delft University of Technology*, 2017.

[FRS16] Benjamin Fuller, Leonid Reyzin, and Adam Smith. When are fuzzy extractors possible? In *Advances in Cryptology – ASIACRYPT*, pages 277–306, 2016.

[FRS20] Benjamin Fuller, Leonid Reyzin, and Adam Smith. When are fuzzy extractors possible? *IEEE Tran. on Information Theory*, 66(8):5282–5298, 2020.

[FSN+09] Albert Fayrushin, KwangSoo Seol, JongHoon Na, SungHoi Hur, Jung-Dal Choi, and Kinam Kim. The new program/erase cycling degra-

dation mechanism of NAND flash memory devices. In *2009 IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, 2009.

[FTGR] Domenic Forte, Mark Tehranipoor, Zimu Guo, and Md Tauhidur Rahman. Flash and SRAM memory chips aging and recycling detection. Available: `https://trust-hub.org/#/data/memory-aging-recycling`. Accessed: 16 Sep 2021.

[G+07] Jorge Guajardo et al. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems*, pages 63–80, 2007.

[G+18] Zimu Guo et al. Scare: An SRAM-based countermeasure against IC recycling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(4):744–755, 2018.

[GCC+09] Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. Characterizing flash memory: Anomalies, observations, and applications. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 24–33, 2009.

[GFT13] Ujjwal Guin, Domenic Forte, and Mohammad Tehranipoor. Anti-counterfeit techniques: From design to resign. In *2013 14th International workshop on microprocessor test and verification*, pages 89–94. IEEE, 2013.

[GFT16] Ujjwal Guin, Domenic Forte, and Mark Tehranipoor. Design of accurate low-cost on-chip structures for protecting integrated circuits against recycling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4):1233–1246, 2016.

[GHD+14] Ujjwal Guin, Ke Huang, Daniel DiMase, John M Carulli, Mohammad Tehranipoor, and Yiorgos Makris. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. *Proceedings of the IEEE*, 102(8):1207–1228, 2014.

[GK14] Achiranshu Garg and Tony T Kim. Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1941–1944. IEEE, 2014.

[GMAS+17] Yansong Gao, Hua Ma, Said F. Al-Sarawi, Derek Abbott, and Damith C. Ranasinghe. Detecting recycled commodity SoCs: Exploiting aging-induced SRAM PUF unreliability, 2017.

[GWHS19] Ujjwal Guin, Wendong Wang, Charles Harper, and Adit D. Singh. Detecting recycled socs by exploiting aging induced biases in memory cells. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 72–80, 2019.

[H+16] Ryan L. Helinski et al. Electronic forensic techniques for manufacturer attribution. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 139–144, 2016.

[HBF07] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *In Proceedings of the Conference on RFID Security*, 2007.

[HBNS13] Clemens Helfmeier, Christian Boit, Dmitry Nedospasov, and Jean-Pierre Seifert. Cloning physically unclonable functions. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 1–6. IEEE, 2013.

[HCM12] Ke Huang, John M Carulli, and Yiorgos Makris. Parametric counterfeit ic detection via support vector machines. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 7–12. IEEE, 2012.

[HCM13] Ke Huang, John M. Carulli, and Yiorgos Makris. Counterfeit electronics: A rising threat in the semiconductor manufacturing industry. In *2013 IEEE International Test Conference (ITC)*, pages 1–4, 2013.

[HHT15] Kai He, Xin Huang, and Sheldon X.-D. Tan. Em-based on-chip aging sensor for detection and prevention of counterfeit and recycled ICs. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 146–151, 2015.

[HKCM13] Ke Huang, Nathan Kupp, John M Carulli, and Yiorgos Makris. Process monitoring through wafer-level spatial variation decomposition. In *2013 IEEE International Test Conference (ITC)*, pages 1–10. IEEE, 2013.

[HKS20] Matthias Hiller, Ludwig Kürzinger, and Georg Sigl. Review of error correction for PUFs and evaluation on state-of-the-art FPGAs. *Journal of Cryptographic Engineering*, 10(3):229–247, 2020.

[HKX+14] Ke Huang, Nathan Kupp, Constantinos Xanthopoulos, John M Carulli, and Yiorgos Makris. Low-cost analog/RF IC testing through combined intra-and inter-die correlation models. *IEEE Design & Test*, 32(1):53–60, 2014.

[HLK+15] Ke Huang, Yu Liu, Nenad Korolija, John M Carulli, and Yiorgos Makris. Recycled IC detection based on statistical methods. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(6):947–960, 2015.

[HM11] James A Hayward and Janice Meraglia. Dna marking and authentication: A unique, secure anti-counterfeiting program for the electronics industry. In *International Symposium on Microelectronics*. International Microelectronics Assembly and Packaging Society, 2011.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[HVK+17] Hasan Hassan, Nandita Vijaykumar, Samira Khan, Saugata Ghose, Kevin Chang, Gennady Pekhimenko, Donghyuk Lee, Oguz Ergin, and Onur Mutlu. SoftMC: A flexible and practical open-source infrastructure for enabling experimental DRAM studies. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 241–252. IEEE, 2017.

[HYKD14] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[IPK18] Md Nazmul Islam, Vinay C Patii, and Sandip Kundu. On ic traceability via blockchain. In *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–4, 2018.

[JCJ+21] Changhun Jung, Jinchun Choi, Rhongho Jang, David Mohaisen, and DaeHun Nyang. A network-independent tool-based usable authentication system for internet of things devices. *Computers & Security*, page 102338, 2021.

[JEDa] JEDEC. Design files for DDR3. Available: https://www.jedec.org/standards-documents/focus/memory-module-designs-dimms/ddr3/all. Accessed: 16 Sep 2021.

[JEDb] JEDEC. Serial presence detect (SPD), general standard. Available: https://www.jedec.org/sites/default/files/docs/4_01_02R19.pdf. Accessed: 16 Sep 2021.

[JED12] JEDEEC. DDR3 sdram standard. Available: https://www.jedec.org/sites/default/files/docs/JESD79-3F.pdf, July 2012. Accessed: 16 Sep 2021.

[JG15] Jae-Won Jang and Swaroop Ghosh. Design and analysis of novel SRAM PUFs with embedded latch for robustness. In *Sixteenth International Symposium on Quality Electronic Design*, pages 298–302. IEEE, 2015.

[JWN10] Bruce Jacob, David Wang, and Spencer Ng. *Memory systems: cache, DRAM, disk*. Morgan Kaufmann, 2010.

[JXW+15] Shijie Jia, Luning Xia, Zhan Wang, Jingqiang Lin, Guozhu Zhang, and Yafei Ji. Extracting robust keys from NAND flash physical unclonable functions. In *International Conference on Information Security*, pages 437–454. Springer, 2015.

[Kan] Michael Kanellos. Counterfeit DRAM chip ring found. Available: https://www.cnet.com/tech/services-and-software/counterfeit-dram-chip-ring-found/. Accessed: 16 Sep 2021.

[Kar20] Paul Karazuba. Combating counterfeit chips. Available: https://semiengineering.com/combating-counterfeit-chips/, August 2020. Accessed: 16 Sep 2021.

[KBTS+18] Preeti Kumari, B. M. S. Bahar Talukder, Sadman Sakib, Biswajit Ray, and Md Tauhidur Rahman. Independent detection of recycled flash memory: Challenges and solutions. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 89–95. IEEE, 2018.

[KBV] KBV Research. Semiconductor Memory Market Size, Growth & Forecast 2026. Available: https://www.kbvresearch.com/semiconductor-memory-market/. Accessed: 16 Sep 2021.

[KCHC08] Tei-Wei Kuo, Yuan-Hao Chang, Po-Chun Huang, and Che-Wei Chang. Special issues in flash. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 821–826. IEEE, 2008.

[KCL+12] Jinmo Kwon, Ik Joon Chang, Insoo Lee, Heemin Park, and Jongsun Park. Heterogeneous SRAM cell sizing for low-power H.264 applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(10):2275–2284, 2012.

[Kem] Zak Kemble. Modifying ram SPD data. Available: https://blog.zakkemble.net/modifying-ram-spd-data/. Accessed: 16 Sep 2021.

[KGKF14] Christoph Keller, Frank Gürkaynak, Hubert Kaeslin, and Norbert Felber. Dynamic memory-based physically unclonable function for the generation of unique identifiers and true random numbers. In *2014 IEEE international symposium on circuits and systems (ISCAS)*, pages 2740–2743. IEEE, 2014.

[KHK+14] Hyunho Kang, Yohei Hori, Toshihiro Katashita, Manabu Hagiwara, and Keiichi Iwamura. Cryptographie key generation from PUF data using efficient fuzzy extractors. In *16th International Conference on Advanced Communication Technology*, pages 23–26, 2014.

[KLB95] Hakam Kalouti, Dejan E. Lazic, and Thomas Beth. On the relation between distance distributions of binary block codes and the binomial distribution. In *Annales Des Télécommunications*, volume 50(9), page 762–778, Sep 1995.

[KLM16] Samira Khan, Donghyuk Lee, and Onur Mutlu. PARBOR: An efficient system-level technique to detect data-dependent failures in DRAM. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 239–250. IEEE, 2016.

[KM14] Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[KMHH15] R. Khazaka, L. Mendizabal, D. Henry, and R. Hanna. Survey of high-temperature reliability of power electronics packaging components. *IEEE Transactions on Power Electronics*, 30(5):2456–2464, 2015.

[KPHM18] Jeremie S Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu. The DRAM latency PUF: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity DRAM devices. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 194–207. IEEE, 2018.

[KQP01] Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Intellectual property metering. In *International Workshop on Information Hiding*, pages 81–95. Springer, 2001.

[KS10] Deniz Karakoyunlu and Berk Sunar. Differential template attacks on PUF enabled cryptographic devices. In *2010 IEEE International Workshop on Information Forensics and Security*, pages 1–6. IEEE, 2010.

[Kum00] Jean Kumagai. Chip detectives [reverse engineering]. *IEEE Spectrum*, 37(11):43–48, 2000.

[KYT+08] Atsushi Kawasumi, Tomoaki Yabe, Yasuhisa Takeyama, Osamu Hirabayashi, Keiichi Kushida, Akihito Tohata, Takahiko Sasaki, Akira Katayama, Gou Fukano, Yuki Fujimura, et al. A single-power-supply 0.7 V 1GHz 45nm SRAM with an asymmetrical unit-ß-ratio memory cell. In *2008 IEEE International Solid-State Circuits Conference-Digest of Technical Papers*, pages 382–622. IEEE, 2008.

[LAWG17] Zhonghao Liao, George T. Amariucai, Raymond K. W. Wong, and Yong Guan. The impact of discharge inversion effect on learning SRAM power-up statistics. In *2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 31–36, 2017.

[Liu07] Frank Liu. A general framework for spatial correlation modeling in VLSI design. In *2007 44th ACM/IEEE Design Automation Conference*, pages 817–822. IEEE, 2007.

[LJK+13] Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu. An experimental study of data retention behavior in modern

DRAM devices: Implications for retention time profiling mechanisms. *ACM SIGARCH Computer Architecture News*, 41(3):60–71, 2013.

[LKS+17] Donghyuk Lee, Samira Khan, Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Gennady Pekhimenko, Vivek Seshadri, and Onur Mutlu. Design-induced latency variation in modern DRAM chips: Characterization, analysis, and latency reduction mechanisms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–36, 2017.

[M+08] G Masalskis et al. Reverse engineering of cmos integrated circuits. *Elektronika ir elektrotechnika*, 88(8):25–28, 2008.

[MBW+19] Thomas McGrath, Ibrahim E. Bagci, Zhiming M. Wang, Utz Roedig, and Robert J. Young. A PUF taxonomy. *Applied Physics Reviews*, 6(1):011303, 2019.

[MGH+18] Qingqing Ma, Chongyan Gu, Neil Hanley, Chenghua Wang, Weiqiang Liu, and Maire O'Neill. A machine learning attack resistant multi-PUF design on FPGA. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 97–104, 2018.

[MGP+11] Farshad Moradi, Sumeet Kumar Gupta, Georgios Panagopoulos, Dag T Wisland, Hamid Mahmoodi, and Kaushik Roy. Asymmetrically doped finfets for low-power robust SRAMs. *IEEE transactions on electron devices*, 58(12):4241–4249, 2011.

[MGR13] Hector Marco-Gisbert and Ismael Ripoll. Preventing brute force attacks against stack canary protection on networking servers. In *2013 IEEE 12th International Symposium on Network Computing and Applications*, pages 243–250. IEEE, 2013.

[MGS13] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*, pages 245–267. Springer New York, New York, NY, 2013.

[MJ06] Ke Meng and Russ Joseph. Process variation aware cache leakage management. In *Proceedings of the 2006 international symposium on Low power electronics and design*, pages 262–267, 2006.

[MKD04] R Mishra, M Keimasi, and D Das. The temperature ratings of electronic parts. *Electronics Cooling*, 10(1):20, 2004.

[MMR10] Debasis Mukherjee, Hemanta Kr Mondal, and BVR Reddy. Static noise margin analysis of SRAM cell for high speed application. *International Journal of Computer Science Issues (IJCSI)*, 7(5):175, 2010.

[MRFA15] Anas Mazady, Md Tauhidur Rahman, Domenic Forte, and Mehdi Anwar. Memristor PUF–a security primitive: Theory and experiment. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):222–229, 2015.

[MS09] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *2009 47th annual Allerton conference on communication, control, and computing (Allerton)*, pages 911–918. IEEE, 2009.

[MSM+20] Shayesteh Masoumian, Georgios Selimis, Roel Maes, Geert-Jan Schrijen, Said Hamdioui, and Mottaqiallah Taouil. Modeling static noise margin for FinFET based SRAM PUFs. In *2020 IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2020.

[MSSS11] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-channel analysis of PUFs and fuzzy extractors. In *Trust and Trustworthy Computing*, pages 33–47, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[NRT19] Abdessamad Najdi, Daniele Rossi, and Vasileios Tenentes. Analysis on retention time and adaptive refresh in embedded DRAMs with aging benefits. In *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 281–286, 2019.

[NSHB13] Dmitry Nedospasov, Jean-Pierre Seifert, Clemens Helfmeier, and Christian Boit. Invasive PUF analysis. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 30–38. IEEE, 2013.

[Ost91] P. R. J. Ostergard. Upper bounds for q-ary covering codes. *IEEE Transactions on Information Theory*, 37(3):660–664, 1991.

[OSW13] Yossef Oren, Ahmad-Reza Sadeghi, and Christian Wachsmann. On the effectiveness of the remanence decay side-channel to clone memory-based PUFs. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 107–125. Springer, 2013.

[PH16] David A Patterson and John L Hennessy. *Computer organization and design ARM edition: the hardware software interface*. Morgan kaufmann, 2016.

[PKR09] Sang Phill Park, Kunhyuk Kang, and Kaushik Roy. Reliability implications of bias-temperature instability in digital ics. *IEEE Design Test of Computers*, 26(6):8–17, 2009.

[PS04] Peter Van Der Putten and Maarten Van Someren. A bias-variance analysis of a real world learning problem: The coil challenge 2000. *Machine learning*, 57(1):177–195, 2004.

[PSK15] C Premalatha, K Sarika, and P Mahesh Kannan. A comparative analysis of 6T, 7T, 8T and 9T SRAM cells in 90nm technology. In *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–5. IEEE, 2015.

[QCF+16] Shahed E Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. A survey on chip to system reverse engineering. *ACM journal on emerging technologies in computing systems (JETC)*, 13(1):1–34, 2016.

[RBT21] Md Tauhidur Rahman and B. M. S. Bahar Talukder. Systems and methods for identifying counterfeit memory, October 5 2021. US Patent 11,139,043.

[RFS+14] Md Tauhidur Rahman, Domenic Forte, Quihang Shi, Gustavo K Contreras, and Mohammad Tehranipoor. CSST: Preventing distribution of unlicensed and rejected ics by untrusted foundry and assembly. In *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 46–51, 2014.

[RH14] U. Rührmair and D. E. Holcomb. Pufs at a glance. In *Design, Automation Test in Europe Conference Exhibition*, pages 1–6, 2014.

[RHG+17] Md Tauhidur Rahman, Alison Hosey, Zimu Guo, Jackson Carroll, Domenic Forte, and Mark Tehranipoor. Systematic correlation and cell neighborhood analysis of SRAM PUF for robust and unique key generation. *Journal of Hardware and Systems Security*, 1(2):137–155, 2017.

[RJA12] Ulrich Rührmair, Christian Jaeger, and Michael Algasinger. An attack on PUF-based session key exchange and a hardware-based countermeasure: Erasable PUFs. In George Danezis, editor, *Financial Cryptography and Data Security*, pages 190–204, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[RNP+16] Davide Resnati, Gianluca Nicosia, Giovanni M. Paolucci, Angelo Visconti, and Christian Monzio Compagnoni. Cycling-induced charge trapping/detrapping in flash memories–part I: Experimental evidence. *IEEE Transactions on Electron Devices*, 63(12):4753–4760, 2016.

[RRFT15] Md Tauhidur Rahman, Fahim Rahman, Domenic Forte, and Mark Tehranipoor. An aging-resistant RO-PUF for reliable key generation. *IEEE Transactions on Emerging Topics in Computing*, 4(3):335–348, 2015.

[RS16] Alec Roelke and Mircea R Stan. Attacking an SRAM-based PUF through wearout. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 206–211, 2016.

[RSK13] Jeyavijayan Rajendran, Ozgur Sinanoglu, and Ramesh Karri. Is split manufacturing secure? In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1259–1264, 2013.

[RSS+10] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249, 2010.

[RSS17] R. Rollini, Jenyfal Sampson, and P. Sivakumar. Comparison on 6T, 5T and 4T SRAM cell using 22nm technology. In *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, pages 1–4, 2017.

[RSSK13] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 709–720, 2013.

[RvD13] Ulrich Rührmair and Marten van Dijk. PUFs in security protocols: Attack models and security evaluations. In *2013 IEEE symposium on security and privacy*, pages 286–300. IEEE, 2013.

[SD07] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14, 2007.

[SK] SK hynix INC. Part no. decoder. Available: `https://product.skhynix.com/support/downloads/techs.go`. Accessed: 16 Sep 2021.

[SKBT+18] Sadman Sakib, Preeti Kumari, B. M. S. Bahar Talukder, Md Tauhidur Rahman, and Biswajit Ray. Non-invasive detection method for recycled flash memory using timing characteristics. *Cryptography*, 2(3):17, 2018.

[SKR+13] Ozgur Sinanoglu, Naghmeh Karimi, Jeyavijayan Rajendran, Ramesh Karri, Yier Jin, Ke Huang, and Yiorgos Makris. Reconciling the ic test and security dichotomy. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2013.

[SMK+18] Yizhak Shifman, Avi Miller, Osnat Keren, Yoav Weizmann, and Joseph Shor. A method to improve reliability in a 65-nm SRAM PUF array. *IEEE Solid-State Circuits Letters*, 1(6):138–141, 2018.

[SMR21] Sadman Sakib, Aleksandar Milenković, and Biswajit Ray. Flash-DNA: Identifying NAND flash memory origins using intrinsic array properties. *IEEE Transactions on Electron Devices*, 68(8):3794–3800, 2021.

[SMRR20] Sadman Sakib, Aleksandar Milenković, Md Tauhidur Rahman, and Biswajit Ray. An aging-resistant NAND flash memory physical unclonable function. *IEEE Transactions on Electron Devices*, 67(3):937–943, 2020.

[SRR16] Soubhagya Sutar, Arnab Raha, and Vijay Raghunathan. D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems. In *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2016.

[SSB+97] Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[SSWB00] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[Sta] Statista. DRAM manufacturer market share by quarter 2021. Available: `https://www.statista.com/statistics/271726/global-market-share-held-by-dram-chip-vendors-since-2010/`. Accessed: 16 Sep 2021.

[SvdL12] Geert-Jan Schrijen and Vincent van der Leest. Comparative analysis of SRAM memories used as PUF primitives. In *Proceedings of the Conference on Design, Automation and Test in Europe*, page 1319–1324, 2012.

[SXA+17] André Schaller, Wenjie Xiong, Nikolaos Athanasios Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. Intrinsic rowhammer PUFs: Leveraging the rowhammer effect for improved security. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 1–7. IEEE, 2017.

[TD01] David MJ Tax and Robert PW Duin. Uniform object generation for optimizing one-class classifiers. *Journal of machine learning research*, 2(Dec):155–173, 2001.

[TD04] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[Tex] Texas Instruments. Complete DDR, DDR2 and DDR3 memory power solution synchronous buck controller, 3–A LDO, buffered reference for embedded computing systems. Available: `https://www.ti.com/lit/ds/symlink/tps59116.pdf`. Accessed: 16 Sep 2021.

[TFR21] B. M. S. Bahar Talukder, Farah Ferdaus, and Md Tauhidur Rahman. A non-invasive technique to detect authentic/counterfeit SRAM chips, 2021. Available: `https://arxiv.org/abs/2107.09199`.

[The] Temptronic ThermoStream. ATS-605 thermostream. Available: https://www.intestthermal.com/temptronic/thermostream. Accessed: 16 Sep 2021.

[TKYC16] Fatemeh Tehranipoor, Nima Karimian, Wei Yan, and John A Chandy. Dram-based intrinsic physically unclonable functions for system-level security and authentication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(3):1085–1097, 2016.

[TKYC17] Fatemeh Tehranipoor, Nima Karimian, Wei Yan, and John A. Chandy. Investigation of DRAM PUFs reliability under device accelerated aging effects. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017.

[U.S18] U.S. Government Publishing Office. Hr 5515–john s. mccain national defense authorization act for fiscal year 2019. In *115th Congress Public Law 232*, volume 13, August 2018.

[vdGS02] A.J. van de Goor and I. Schanstra. Address and data scrambling: causes and impact on memory tests. In *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications '2002*, pages 128–136, 2002.

[vDR14] Marten van Dijk and Ulrich Rührmair. Protocol attacks on advanced PUF protocols and countermeasures. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2014.

[vH18] Marten van Hulst. Physically unclonable functions as a solid foundation of platform security architecture. Available: https://community.arm.com/iot/b/internet-of-things/posts/physically-unclonable-functions-as-a-solid-foundation-of-platform-security-architecture, November 2018. Accessed: 16 Sep 2021.

[WDZ+16] Debao Wei, Libao Deng, Peng Zhang, Liyan Qiao, and Xiyuan Peng. Nrc: A nibble remapping coding strategy for NAND flash reliability extension. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(11):1942–1946, 2016.

[WGR+17] Xiaoxiao Wang, Yueyu Guo, Tauhid Ramhan, Dongrong Zhang, and Mark Tehranipoor. Dost: Dynamically obfuscated wrapper for split

test against ic piracy. In *2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6. IEEE, 2017.

[WK12] Jiyoung Woo and Huy Kang Kim. Survey and research direction on online game security. In *Workshop at SIGGRAPH Asia*, page 19–25, 2012.

[WKP14] James B. Wendt, Farinaz Koushanfar, and Miodrag Potkonjak. Techniques for foundry identification. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014.

[WMNS00] Peng Wu, Bangalore S Manjunath, Shawn Newsam, and H. D. Shin. A texture descriptor for browsing and similarity retrieval. *Signal processing: Image communication*, 16(1-2):33–43, 2000.

[XRF+14] Kan Xiao, Md Tauhidur Rahman, Domenic Forte, Yu Huang, Mei Su, and Mohammad Tehranipoor. Bit selection algorithm suitable for high-volume production of SRAM-PUF. In *2014 IEEE international symposium on hardware-oriented security and trust (HOST)*, pages 101–106. IEEE, 2014.

[XSA+16] Wenjie Xiong, André Schaller, Nikolaos A Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. Run-time accessible DRAM PUFs in commodity devices. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 432–453. Springer, 2016.

[YD10] Meng-Day Yu and Srinivas Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design Test of Computers*, 27(1):48–65, 2010.

[YGH18] Yildiran Yilmaz, Steve R. Gunn, and Basel Halak. Lightweight PUF-based authentication protocol for iot devices. In *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, pages 38–43, 2018.

[YIKA16] Selen Yılmaz Isıkhan, Erdem Karabulut, and Celal Reha Alpar. Determining cutoff point of ensemble trees based on sample size in predicting clinical dose with dna microarray data. *Computational and mathematical methods in medicine*, 2016.

[YL19] Thomas Yang and Xi-Wei Lin. Trap-assisted DRAM row hammer effect. *IEEE Electron Device Letters*, 40(3):391–394, 2019.

[YMSD11] Meng-Day Mandel Yu, David M'Raihi, Richard Sowell, and Srinivas Devadas. Lightweight and secure PUF key storage using limits of machine learning. *Cryptographic Hardware and Embedded Systems - CHES 2011 Lecture Notes in Computer Science*, page 358–373, 2011.

[ZG14] Jizhe Zhang and Sandeep Gupta. Sram array yield estimation under spatially-correlated process variation. In *2014 IEEE 23rd Asian Test Symposium*, pages 149–155. IEEE, 2014.

[Zha91] Zhen Zhang. Linear inequalities for covering codes: Part I– pair covering inequalities. *IEEE transactions on information theory*, 37(3):573–582, 1991.

[ZT14] Xuehui Zhang and Mohammad Tehranipoor. Design of on-chip lightweight sensors for effective detection of recycled ICs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(5):1016–1029, 2014.

[ZTT12] Xuehui Zhang, Nicholas Tuzzio, and Mohammad Tehranipoor. Identification of recovered ics using fingerprints from a light-weight on-chip sensor. In *DAC Design Automation Conference 2012*, pages 703–708, 2012.

[ZWRT18] Dongrong Zhang, Xiaoxiao Wang, Md Tauhidur Rahman, and Mark Tehranipoor. An on-chip dynamically obfuscated wrapper for protecting supply chain against IP and IC piracies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(11):2456–2469, 2018.

[ZZH+14] Shijun Zhao, Qianying Zhang, Guangyao Hu, Yu Qin, and Dengguo Feng. Providing root of trust for ARM TrustZone using on-chip SRAM. In *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices*, TrustED '14, page 25–36, New York, NY, USA, 2014. Association for Computing Machinery.

APPENDIX A

HAMMING WEIGHT DISTRIBUTION

An n-bit binary string can form a total of 2^n outcomes. For proper randomness, each of the outcomes should have an equal probability of $\frac{1}{2^n}$. Alternatively, the probability of getting a specific symbol ("0" or "1") at t^{th} bit location of the string is $\frac{1}{2}$ (assuming the outcome is not biased to any symbol).

Next, consider a random n-bit string (S_i) of *Hamming weight* k (i.e., out of n bits, k bits of the string is "1"). So, the probability of getting such string as the PUF outcome,

$$p(HW = k) = \sum_i p(S_{i,HW=k}) = \binom{n}{k} \times (\frac{1}{2^n})$$

$$\Rightarrow p(HW = k) = \binom{n}{k} \times (\frac{1}{2})^k \times (\frac{1}{2})^{n-k} \tag{A.1}$$

Eqn. A.1 is the probability mass function (PMF) of the Binomial distribution, where the second and third term in the right-hand side comes from the fact that the probability of getting "0" or "1" in any specific position of the bit string is $\frac{1}{2}$.

APPENDIX B

mPUF KEY RECOVERING USING BRUTE-FORCE ATTACK

If a random source (i.e., SRAM chip) can produce m n-bit keys (denoted as S_m), the probability of m keys being unique is defined as:

$$p_{unique} = \prod_{j=0}^{m-1} \frac{2^n - j}{2^n} \tag{B.1}$$

Hence, the probability of having a duplicate key, $p_{duplicate} = 1 - p_{unique}$. In this paper, we have used $n = 64$ and $m = 65,536$, which gives $p_{duplicate} \approx 1.164 \times 10^{-10}$. For simplicity, we can eliminate such cases of having duplicate patterns in S_m.

Now, let's assume we have randomly selected t unique n-bit patterns (denoted as S_t), where $t \ll 2^n$. Hence, the probability of matching l pattern from S_t with S_m:

$$p(S_t \cap S_m = l) = \binom{m}{l} \times \left[\prod_{j=1}^{l} \frac{t - (j-1)}{2^n - (j-1)} \right] \tag{B.2}$$

Therefore, the expected number of recovered keys from t patterns can be expressed with the following equation:

$$E(\#key) = \sum_{j=1}^{m} \left[j \times p(S_t \cap S_m = j) \right] \tag{B.3}$$

With $l = 10$ million, $E(\#key) \approx 5.72 \times 10^{-11}\%$.

VITA

BASHIR MOHAMMAD SABQUAT BAHAR TALUKDER

2015	B.Sc., Electrical and Electronic Engineering Bangladesh University of Engineering and Technology Dhaka, Bangladesh
2015–2017	Physical Design Engineer PrimeSilicon Technology (BD) Ltd. Dhaka, Bangladesh
2021	Ph.D., Electrical and Computer Engineering Florida International University Miami, Florida, USA

SELECTED PUBLICATIONS AND PATENTS

Md Tauhidur Rahman, and B. M. S. Bahar Talukder. Systems and methods for identifying counterfeit memory, October 05, 2021. US Patent 11,139,043.

B. M. S. Bahar Talukder, Farah Ferdaus, and Md Tauhidur Rahman. Memory-based PUFs are Vulnerable as Well: A Non-invasive Attack against SRAM PUFs. *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4035–4049, 2021.

Md Imtiaz Rashid, Farah Ferdaus, BMS Bahar Talukder, Paul Henny, Aubrey N. Beal, and Md Tauhidur Rahman. True Random Number Generation Using Latency Variations of FRAM. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 14–23, Jan. 2021.

B. M. S. Bahar Talukder, Biswajit Ray, Domenic Forte, and Md Tauhidur Rahman. PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures. *IEEE Access*, vol. 7, pp. 81106–81120, 2019.

Sadman Sakib, Preeti Kumari, B. M. S. Bahar Talukder, Md Tauhidur Rahman, and Biswajit Ray. Non-Invasive Detection Method for Recycled Flash Memory Using Timing Characteristics. *Cryptography*, 2018.

B. M. S. Bahar Talukder, Farah Ferdaus, and Md Tauhidur Rahman. A Non-invasive Technique to Detect Authentic/Counterfeit SRAM Chips. *arXiv e-prints*, 2021. Available: `https://arxiv.org/abs/2107.09199`.

Farah Ferdaus, B. M. S. Bahar Talukder, and Md Tauhidur Rahman. Approximate MRAM: High-performance and Power-efficient Computing with MRAM Chips for

Error-tolerant Applications. *arXiv e-prints*, 2021. Available: `https://arxiv.org/abs/2105.14151`.

Emil Jovanov, B. M. S. Bahar Talukder, David C. Schwebel, and W. Douglas Evans. Design and Feasibility of a Safe Pill Bottle. *Applied System Innovation*, 2018.

Farah Ferdaus, B. M. S. Bahar Talukder, Mehdi Sadi, Md Tauhidur Rahman. True Random Number Generation by Write Latency Variation Using Commercial MRAM chips. *International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 510–515.

B. M. S. Bahar Talukder, Vineetha Menon, Biswajit Ray, Tempestt Neal, Md Tauhidur Rahman. Towards the Avoidance of Counterfeit Memory: Identifying the DRAM Origin. *IEEE Hardware-Oriented Security and Trust Symposium (HOST)*, 2020, pp. 111–121.

B. M. S. Bahar Talukder, Joseph Kerns, Biswajit Ray, Thomas Morris, and Md Tauhidur Rahman. Exploiting DRAM Latency Variations for Generating True Random Numbers. *IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1–6.

B. M. S. Bahar Talukder, Emil Jovanov, David C. Schwebel, and W. Douglas Evans. A New Method to Prevent Unintentional Child Poisoning. *IEEE Engineering in Medicine and Biology Conference (EMBC)*, 2018, pp. 5142–5145.

Preeti Kumari, B. M. S. Bahar Talukder, Sadman Sakib, Biswajit Ray, and Md Tauhidur Rahman. Independent Detection of Recycled Flash Memory: Challenges and Solutions. *IEEE Hardware-Oriented Security and Trust Symposium (HOST)*, 2018, pp. 89–95.