UNIVERSITY OF CALIFORNIA

Los Angeles

Hardware-Enabled Design for Security (DFS) Solutions

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Wei-Che Wang

2018

ABSTRACT OF THE DISSERTATION

Hardware-Enabled Design for Security (DFS) Solutions

by

Wei-Che Wang
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2018
Professor Puneet Gupta, Chair

The Integrated Circuit (IC) supply chains of modern companies often involve multiple business entities on a global scale, including offshore manufacturing, system integration and distribution of VLSI chips and systems. While the industry is trying to lower the risks imposed by the global supply chain production model, most existing techniques, such as Physical Uncolonable Function (PUF), logic obfuscation, and hardware metering often suffer from their unreliability characteristics for their parametric nature or high implementation cost of the whole security system. Therefore, IC/IP Design for Security (DFS) solutions that are efficient and practical for the industry are still yet to be discovered.

In this dissertation we study the behavior of PUFs and propose several sources of randomness to construct stability-guaranteed PUFs through Locally Enhanced Defectivity (LED) mechanisms, such as Directed Self Assembly (DSA) and transistor gate oxide breakdown. These PUFs are fabricated and demonstrated to be stable and random, which can be used as reliable sources of hardware root-of-trust for DFS techniques.

To study the security of PUFs and to show the benefits of our proposed stability-guaranteed PUFs, we present a new unified framework for evaluating PUF security through guesswork analysis. This framework enables us to evaluate and quantify the effect of noise, bias and model attacks on security. We also relate guesswork to other security measures such as min-entropy, and mutual information. The model quantitatively measures the security of various PUFs under different scenarios, and by doing so enables us to compare the security

level of different sorts of PUFs.

To further utilize the stable PUFs, a secure lightweight entity authentication hardware primitive (SLATE) is proposed and shown to be much smaller than existing strong PUFs and lightweight ciphers. The proposed SLATE is a practical DFS solution for its extremely lightweight implementation and is proven to be secure from both empirical and theoretical perspectives.

Finally, the dissertation proposes an effective attack to reconstruct missing connections in 2.5D split manufacturing, which is a technique used to prevent reverse engineering from malicious foundry. A Satisfiability Modulo Theories (SMT) based grouping algorithm depending purely on the circuit functionality but not physical implementation is proposed to significantly reduce the runtime of Boolean Satisfiability (SAT) solver, which is used to recover configuration keys of the connection network. Defence strategies of our attacks are also studied.

The dissertation of Wei-Che Wang is approved.

Jane Pei-Chen Chang

Chih-Kong Ken Yang

Suhas N. Diggavi

Puneet Gupta, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to express my deepest gratitude to my advisor, Professor Puneet Gupta. I am fortunate enough to have Prof. Gupta as my advisor and as a mentor who supports and guides me not only to obtaining my Ph.D but also in many aspects beyond research. His high standards and rigorous research attitude have greatly improved the quality and presentation of my work, and his valuable life and career advice have also made me a more successful person.

I would like to thank Professor Suhas Diggavi and Dr. Yair Yona for the guidance and insightful discussions on guesswork analysis and the security theory for PUFs. The cooperation with Prof. Suhas Diggavi and Dr. Yair Yona was really enjoyable and their creative ideas inspired me.

I would like to thank Michael Chen, who was the manager and my mentor at Mentor Graphics for my summer internships. Michael and his team members (Joseph, Mario, Gilduin, Dan, Marc, Donovan, Steven) brought me into the industrial world and inspired me on the layout agnostic PUF research. I thank Dr. Andres Torres at Mentor Graphics for the support of the DSA simulator for the LEDPUF research. I would also like to thank Dr. Ron Cocchi and Lap from Inside Secure for the opportunity to work on industrial hardware security projects and their recognition and appreciation of my knowledge.

The NanoCad Lab is the best lab in the world! I appreciate the friendships we made in my years here: Lerong, Tuck, Rani, Abde, Liangzhen, Shaodi, Mark, Yasmine, Saptadeep, Irina, Wojciech, Tianmu, Yizhang, YooJin, and of course my roommate Albert. It was so amazing to have you around; I love those moments of discussion, collaboration (especially in group meetings), celebration dinners, and sharing life with difficulties, happiness, frustration, and surprises. I have learned a lot from you and thank you for making my time at UCLA one of the best times in my life.

Lastly, I would like to thank my family for everything they have done in my years at UCLA. I thank my parents for their unselfish help during the difficult and struggling times in the long pursuit of a Ph.D. I thank my wife Yi-Chuang Chang from the bottom of my

heart for all the support she gave me during my difficult times. I could not have finished this journey without the love and support from her over the years.

VITA

2007	B.S. in Computer Science and Information Engineering National Taiwan University, Taipei, Taiwan.
2009	M.S. in Electronics Engineering National Taiwan University, Taipei, Taiwan.
2009–2013	Standard Cell Engineer Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan.
2013–2018	Graduate Student Researcher Department of Electrical and Computer Engineering University of California, Los Angeles.

CHAPTER 1

Introduction

In order to optimize resource and cost, the Integrated Circuit (IC) supply chains of modern companies often involve multiple business entities on a global scale, including offshore manufacturing, system integration and distribution of VLSI chips and systems. The diverse and complex fabrication environment have made various attacks to the IC design easier, and huge revenue loss has been reported for the past few years.

For an authentic Intellectual Property (IP) owner, the global multi-stage environment makes it difficult to monitor the whole production process, and a rouge element, such as a tester or even a foundry, can easily steal or misuse confidential secrets of a design. For example, a malicious foundry may reverse engineer or overproduce the IC to cause unexpected loss to the IC design company. In 2013, the United States made over 6,700 seizures of consumer electronic counterfeit shipments, and legitimate electronics companies miss out on about $100 billion of global revenue every year because of counterfeiting. The IC supply chain security has become a serious concern for the industry.

While the industry is trying to lower the risks imposed by the global supply chain production model, most existing techniques, such as Physical Uncolonable Function (PUF), logic obfuscation, and hardware metering, often suffer from their unreliability characteristics for their parametric nature or high implementation cost of the whole security system. For example, for a SRAM PUF to generate a 128-bit secret key, more than 4k SRAM cells are needed under a condition with 15% bit error probability, which translates into a roughly 30x overhead not including the hardware implementation of the Error Correction Code (ECC). Therefore, practical IC/IP Design for Security (DFS) solutions that are efficient and practical for the industry are still yet to be discovered.

For the rest of this chapter, a brief introduction is given regarding existing supply chain security techniques, followed by an outline of my dissertation, including the improved reliable DFS solutions that are adoptable to the industry.

1.1 Physical Unclonable Function (PUF)

A Physical Unclonable Function (PUF) is a small piece of circuitry such that its behavior, or Challenge Response Pair (CRP), is uniquely defined and it is hard to be predicted and replicated because of the intrinsic random physical nature and the uncontrollability of process variations.

As a security primitive, PUF can enable low overhead hardware identification, tracing, and authentication during the global manufacturing chain. The first PUF was introduced more than a decade ago. Since then, many silicon PUF implementations have been proposed, such as Arbiter PUF, Ring Oscillator (RO) PUF, SRAM PUF, and many other variations.

Figure 1.1 shows a conventional Ring Oscillator (RO) PUF. The PUF is composed of many ROs with same design, and a challenge selects a pair of ROs to compare their frequencies. If the value of the difference register is greater than zero, then the response is one; otherwise, the response is zero. Even though the ROs are designed to be identical, the frequencies of the ROs will be different due to the uncontrollable process variations, which are considered to be random and unique for each chip. Therefore, a ROPUF can be used to generate a secret key without storing any secret information in a memory device.

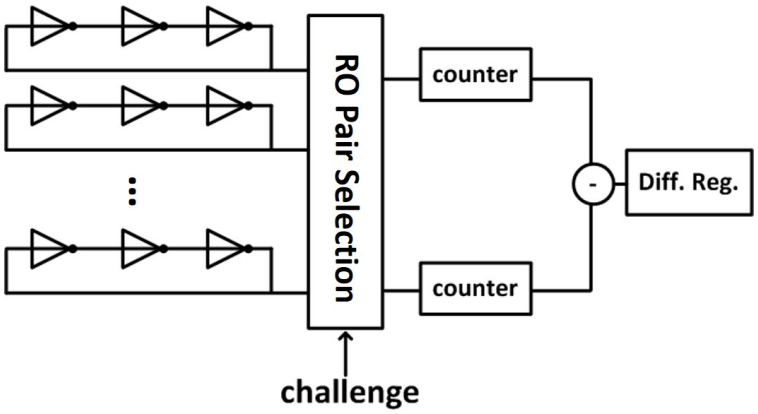


Figure 1.1: Ring Ocsillator (RO) PUF. The one-bit response is obtained from the value of the different register.

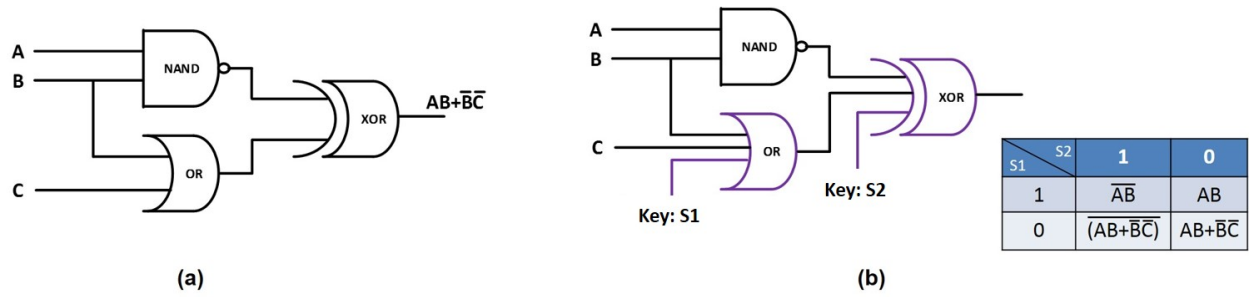
Since the key commonality between all current silicon PUF implementations is their use of parametric manufacturing variations as the source of randomness, there exist several limitations including noise and environmental fluctuations that can cost expensive implementation overhead.

1.2 Function Obfuscation

The goal of logic obfuscation is to prevent reverse engineering by hiding the function of the design. The obfuscation is achievable based on variety of techniques, such as embedded reconfigurable logic insertion or IC camouflaging.

One of the most common function obfuscation implementations is to insert additional XOR gates or wire swapping units to change the original circuit function. Figure 1.2 shows a simple example of function obfuscation with two keys $S1$ and $S2$, which are only known to the circuit designer. Before the insertion of keys, the circuit function is $AB + \bar{B}\bar{C}$ as shown in Figure 1.2 (a). After the key insertion, the function of the circuit is obfuscated. The circuit will perform one of the four possible functions as shown in Figure 1.2 (b) depending on the values of the keys $S1$ and $S2$, where the correct values ($S1 = 0$ and $S2 = 0$ in this case) are only known to the circuit designer. Since the number of possible functions grow

exponentially wit the number of keys inserted, it is difficult for one to find out the correct function of the circuit without knowing the secret key.



S1 \ S2	1	0
1	$\overline{AB}$	AB
0	$\overline{(AB+\bar{B}\bar{C})}$	$AB+\bar{B}\bar{C}$

Figure 1.2: Function obfuscation with secret keys. (a) Original circuit with function $AB + \bar{B}\bar{C}$. (b) Four possible functions when two keys (secret bits only known to the designer) are inserted. The circuit will only function correctly when correct values of $S1$ and $S2$ are given.

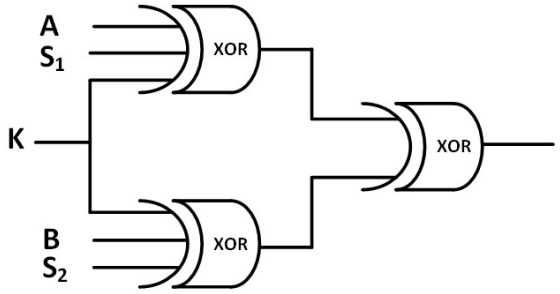
For the gate insertion techniques, since the keys of the inserted gates are the same for all fabricated chips, it is still possible for the attacker to obtain a working chip and correct keys from the open market and reverse engineer the circuit function.

1.3 Hardware Metering

The goal of hardware metering is to prevent IC overproduction from malicious foundries. Once a design house delivers a design to a foundry, the design house will have no control of the amount of copies that the foundry can produce. Hardware metering techniques try to lock each chip with a unique key so the IC/IP owner can have control over the number of fabrications. Please note that the major difference between hardware metering and function obfuscation is that a unique key of each chip is required for hardware metering, while for function obfuscation, all chips would have a same key. Therefore, hardware metering often provides higher security than function obfuscation.

Similar to function obfuscation, common hardware metering approaches use augmented gates to lock and shuffle the I/O ports of each chip so no information about the function is leaked to foundries or testers. Figure 1.3 shows a metering example with a unique key

K and additional configurations $S1$ and $S2$. Assume that the correct function is $AB + \bar{A}\bar{B}$ and the unique key of each chip is only known to the IC/IP owner. If $K = 0$ for a specific chip, the configurations to unlock the chip would be $(S1, S2) = (0, 1)$; if $K = 1$ for another chip, the configurations to unlock the chip would be $(S1, S2) = (1, 0)$. The IC/IP owner can decide the number of configurations released to control the number of working chips in the open market.



(S_1 xor K , S_2 xor K)	Function
(0 , 0)	$A\bar{B} + \bar{A}B$
(1 , 1)	$\bar{A}B + A\bar{B}$
(0 , 1)	$AB + \bar{A}\bar{B}$
(1 , 0)	$\bar{A}\bar{B} + AB$

Figure 1.3: Hardware metering. If $K = 0$ for a specific chip, the configurations to unlock the chip would be $(S1, S2) = (0, 1)$. Since K is unique for each chip, the IC/IP owner would have control of the number of chip to unlock.

To generate unique keys, many existing hardware metering techniques exploit PUFs as sources of the secret randomness, therefore the PUF responses should be stable. Also, the insertion location should be chosen carefully to prevent advanced key-recovery attacks such as fault analysis attack and Boolean satisfiability (SAT) attack.

1.4 Entity Authentication

PUF-based entity authentication protocols have been proposed since the idea of PUF was introduced. The simple authentication scheme proposed in [2] employs the CRP mechanism of strong PUFs, but the protocol is not practical because the PUF itself is not secure and suffers from modeling attacks. Using an Optical PUF as stated in [3] is believed to be more secure against modeling attacks, however it contradicts the low-cost design principle of a PUF application. PUF-based mutual authentication protocols often require synchronized

time stamp [4] or heavy computation costs, such as the reverse fuzzy extractor proposed in [5].

Most of the existing PUF-based entity authentication protocols aim to compensate the limitations of instability and vulnerability to modeling attacks, but unfortunately, these approaches often undermine the benefits provided by the PUF technology, such as the lightweight implementation or the replacement of costly secure data storage. Also, the use of fuzzy extractor and helper data can often introduce unexpected security vulnerabilities [6], making most existing PUF-based protocols impractical. Therefore, a lightweight and secure authentication primitive is still yet to be discovered.

1.5 Split Manufacturing

In order to protect IC/IP owners from the malicious foundries or attackers, Layer-based Split Manufacturing (LSM) was recently proposed as a protecting mechanism to minimize the aforementioned risks. LSM divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) parts, and different parts are fabricated at different foundries. The FEOL (higher complexity and cost) part is fabricated at an untrusted foundry. Since the complete connections of the circuit are unknown to the untrusted foundry, the design cannot be fully reverse engineered. After the FEOL fabrication, the wafer is shipped back to an onshore trusted foundry for the BEOL fabrication and integration. While LSM may fit well with the advanced 3D IC fabrication model, however, the yield loss due to wafer transportation, integration, and the requirement of design rule compatibility of two foundries are still remaining as the major challenges.

Another split manufacturing strategy is the Module-based Split Manufacturing (MSM), where the design is split into different modules, and all layers including FEOL and BEOL of a module are fabricated at a same foundry. The modules are then sent back to an integrator for final integration. Compared with LSM, the advantages of MSM include: 1) Better yield because less transportation and alignment risks, and each module is packaged and tested as normal chip before being sent to the integrator. 2) No design rule compatibility requirements

since the connection is usaully done through chip-to-chip interposers.

1.6 Dissertation Outline

This dissertation first investigates sources of instabilities of existing delay based PUFs and explores a new side channel attack that exploit systematic wafer fabrication correlations. Then, the dissertation presents a physical implementation agnostic PUF that its biased responses are efficiently eliminated during Register Transfer Level (RTL) design stage. Next, two stability-guaranteed PUFs are presented with security evaluation frameworks. Following the stable PUFs, a secure lightweight entity authentication hardware primitive is presented and shown to be secure against many existing attacks. Finally, a MSM attacking model based on SAT and Satisfiability Modulo Theories (SMT) solvers is proposed to show the effectiveness to recover the missing connections and strategies to prevent such attack. The rest of the dissertation is organized as follows:

Chapter 2 - Assessing Viability of Delay-Based PUFs: In this chapter, we explore a new silicon side channel attack for delay-based PUFs that exploits wafer fabrication correlations and reemphasize that the local variation is the only desired entropy source of a PUF. Furthermore, we investigate sources of noise in measured PUF delay signatures such as measurement errors from metastability and temporal jitter.

Chapter 3 - UNBIAS PUF: A Physical Implementation Bias Agnostic PUF: In this chapter, we propose novel weak and strong UNBIAS PUFs that can be implemented purely by Register Transfer Language (RTL), such as verilog, without imposing any physical design constraints or delay characterization effort to solve the biased responses caused by asymmetric routing or systematic variations.

Chapter 4 - LEDPUF: Stability-Guaranteed Physical Unclonable Functions through Locally Enhanced Defectivity: In this chapter, we propose several weak PUFs and strong PUFs that are completely stable with 0% intra-distance. These PUFs are called Locally Enhanced Defectivity Physical Unclonable Function (LEDPUF). A LEDPUF is a pure functional PUF which eliminates the instability of conventional parametric PUFs, there-

fore no helper data, fuzzy comparator, or any kinds of correction schemes are required. Two sources of stable randomness are presented in this chapter: Directly Self Assembly (DSA) LEDPUF and gate oxide breakdown LEDPUF using plasma induced damage during semiconductor manufacturing and voltage stressed damage post manufacturing

Chapter 5 - PUF Security Evaluation through Guesswork Analysis: In this chapter we develop a new unified framework for evaluating the security of PUFs, based on password security, by using information theoretic tools of guesswork. The guesswork model allows us to quantitatively compare, with a single unified metric, PUFs with varying levels of stability, bias and available side information. In addition, it generalizes other measures to evaluate the security level such as min-entropy and mutual information. We evaluate guesswork based security of some measured Static Random Access Memory (SRAM) and Ring Oscillator PUFs as an example and compare them with LEDPUFs to show that stability has a more severe impact on the PUF security than biased responses. Furthermore, we find the guesswork of two new problems: Guesswork under probability of attack failure, and the guesswork of strong PUFs that are used for authentication.

Chapter 6 - SLATE: A Secure Lightweight Entity Authentication Hardware Primitive: In this chapter, we propose a novel Secure Lightweight Entity Authentication hardware primitive called SLATE, where its secret key can be stored in a form of a weak PUF or any secure key storage. SLATE is resistant to known attacks to strong PUFs or logic obfuscations, such as model building attacks and Boolean Satisfiability (SAT) attacks, and we show that the implementation cost of SLATE with a 176-bit key and 2^{44} CRPs is only 663 Gate Equivalents (GE). Compared to lightweight ciphers and existing secure strong PUFs, which is 44% to 7.1× larger than SLATE, we show that SLATE is a practical security primitive for resource constrained systems.

Chapter 7 - Reverse Engineering of 2.5D Split Manufactured ICs: In this chapter we propose a SAT-based attack to reconstruct the wire connections of the 2.5D split manufacturing netlists. Unlike previous attacks to split manufacturing that do not guarantee 100% accuracy of the connections, our SAT-based attack can fully reconstruct the missing wires and therefore the functionality of the chip can be completely reverse engineered.

In addition, we show that the runtime of SAT attack is significantly reduced by applying grouping hints obtained from SMT-based grouping algorithm, which is purely depending on the circuit functionality, so no physical defensive mechanisms can prevent such attack. We show that our grouping algorithm can speed up the SAT attack runtime by more than 1,000X and can successfully reverse engineer reasonable size benchmarks even when the split nets contains more than one fanouts and the total cut size is relatively large.

Chapter 8 - Conclusion: This chapter concludes the dissertation by providing a brief review of the motivation and works accomplished, and finally summarizes the contributions of the dissertation.

CHAPTER 2

Assessing Viability of Delay-Based PUFs

Physical Unclonable Function (PUF) is a promising hardware security primitive because of its low area and power consumption. However, the stability of a PUF and how resilient it is to attacks have been two major concerns for practical use of it. In this chapter we explore a new silicon side channel attack for delay-based PUFs that exploits wafer fabrication correlations and reemphasize that the local variation is the only desired entropy source of a PUF. Furthermore, we investigate sources of noise in measured PUF delay signatures such as measurement errors from metastability and temporal jitter. In each of these studies we use a mix of actual silicon results and models to illustrate the challenges in delay-based PUF adoption.

2.1 Introduction

Hardware security has become an important aspect in modern Integrated Circuit (IC) design industry because of the global supply chain business model. Identifying and authenticating each fabricated components of a chip could be a challenging task [7]. A Physical Unclonable Function (PUF) is a small piece of circuitry such that its behavior, or Challenge Response Pair (CRP) [3], is uniquely defined and is hard to be predicted and replicated because of the intrinsic random physical nature and the uncontrollability of process variations. As a security primitive, PUF can enable low overhead hardware identification, tracing, and authentication during the global manufacturing chain. Also, the compact area and low energy requirements make PUF an ideal solution for the security requirements of Internet of Things (IoT), which is a rapidly emerging paradigm that the security between untrusted subsystems is known to

be of essential importance [8].

The first PUF was introduced more than a decade ago [9]. Since then, many silicon PUF implementations are proposed, including Arbiter PUF [10], Ring Oscillator (RO) PUF [11], SRAM PUF [12], and many other variations. The fundamental application of PUFs lies in its identification purposes, thus the calculation of inter-distance and intra-distance [3] inherently became important measuring metrics on uniqueness and stability of a PUF respectively. Inter-distance is defined as the distance of two responses from two PUFs given a same challenge, and intra-distance is defined as the distance of two responses from a PUF given a same challenge twice. The distances are often represented in Hamming Distance (HD), which is defined as the number of bits that differ between two bit strings, and Fractional Hamming Distance (FHD), which is defined as the HD divided by the length of the compared strings. For an ideal PUF, the inter-distance should be 50%, and the intra-distance should be 0%, meaning that the responses are unique and can be reproduced.

Studies of attacking PUF and PUF protocols have been considered the driving force of the evolution of PUF. Machine-learning based attacks [1] extrapolate the PUF behavior on whole CRP database by building a model from observing a limited fraction of CRPs, and side-channel attacks [13] explore the weakness from the practical implementation of a system. Invasive attacks and physical cloning tailored specifically on SRAM PUFs have also been carried out [14].

While many studies of side channel attacks on timing analysis and power consumption have been proposed [13], the systematic fabrication variation has not been widely discussed and can also be analyzed for silicon side channel attacks. In our work, we model the fabrication variations obtained from real silicon data, and demonstrate the effectiveness of the silicon side channel attacks based on the fabrication variation model. Our results show that local randomness should be the only desired entropy source of the delay based PUF, which has been observed previously as well in [15]; otherwise the silicon fabrication information can be exploited as side-channel attack. Even though this is known but unfortunately the importance of it has been ignored in studies of PUFs (for instance by assuming all variation to be local in Monte Carlo simulations). For completeness, we evaluate memory PUFs as

well for their susceptibility to such process side channel attacks.

2.1.1 Our Contributions

Key contributions of this work are summarized as follows:

- A silicon wafer variation model is utilized to demonstrate novel effective side channel attack on delay-based silicon PUFs.
- Models of metastability, jitter accumulation, and corresponding error probabilities are derived, and their impact on PUF statistics is discussed for the first time.

2.2 PUF Implementations

2.2.1 Experiment Platform

In our experiments, we use fifteen commercial 45nm SOI test chips with 176kB data memory each to evaluate SRAM PUF, and 6 RO PUF chips with 13 RO pairs each for RO PUF evaluation. The frequencies of these RO pairs are measured and compared, so the response is 13 bits long for each PUF.

For arbiter PUF simulation, we gather 14 delay paths to form an arbiter PUF that are measured one time from a commercial 65nm technology process. There are 36 thousand arbiter PUFs from 23 wafer lots which contains 16 to 25 wafers each, and 90 arbiter PUFs are fabricated on each wafer. The exact location of each PUF is known for silicon variation analysis. The structure of the arbiter PUF used in our experiments is given in Figure 2.1. 2 of 14 delay paths are chosen to compare the timing difference, so each PUF can generate a 91-bit response. Since the focus of this work is to assess the impact of silicon correlations on PUFs but not the model building attacks, the arbiter PUF structure with each path specifically implemented is used.

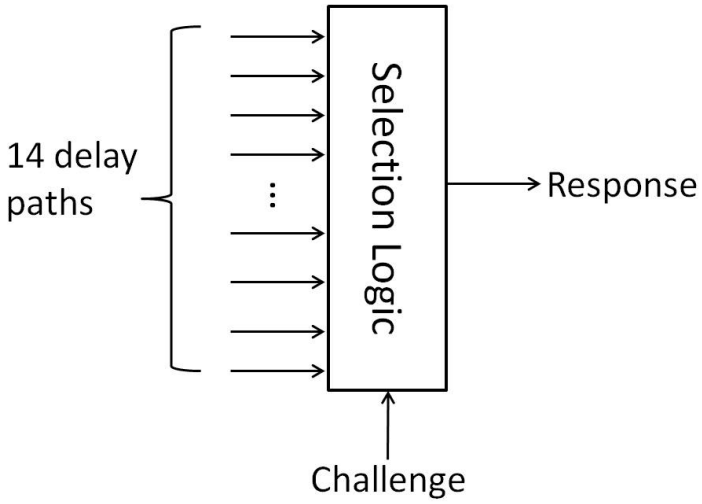


Figure 2.1: The arbiter PUF structure for silicon variation evaluation.

2.2.2 Memory based PUF

A memory based PUF exploits the inherent threshold variation of the cross-coupled devices to generate random responses, which can be used as a unique fingerprint of the IC or random key generation. There are many implementations of memory based PUF on different storage techniques including SRAM PUF [12], Flash [16], DRAM [17], and Memristors [18]. Among these variations of memory based PUFs, SRAM PUF is considered to be one of the most popular types of PUFs because they are easy to manufacture. Therefore, the focus of our work on memory based PUF will be on SRAM PUF.

SRAM PUF is a primitive well suited for secure key generation because the key is not stored on a non-volatile memory and is derived only when needed, meaning that it is presented only for a very short period of time. In a cryptographic secret key generation procedure, it is important that the key is generated with high randomness, and SRAM PUF shows a highly random response that has little fabrication correlation because the cross-coupled transistors are placed very closely to each other, which intrinsically cancel out the spatial correlations and systematic variations.

We evaluate the inter-distance distribution of SRAM PUF using fifteen commercial 45nm SOI test chips, where each consists 176kB data memory. The results show that the intra-

distance distribution is closely matched to a normal distribution with mean of 48.33%. This is expected since the response of SRAM PUF is highly random. For intra-distance measurement, the power-up state is measured 10 times during an 8-hour period, and the mean of intra-distance distribution is only 2.57%.

Because of the independence of each SRAM cell, it is not possible to predict SRAM PUFs using modeling or machine learning techniques. However, the states of SRAM cells are known to be observable by physical read-out utilizing laser stimulation even if care is taken to prevent the values from being revealed over standard channels. Therefore, the design of SRAM PUF must be tamper evident, meaning that the invasive attacks must alter the memory cell characteristics in such a way that the key derived from the PUF response becomes unrecoverable [19]. For rest of this chapter we focus on delay-based PUFs where the prominence of local variation is not guaranteed.

2.2.3 Delay based PUF

2.2.3.1 Arbiter PUF

The idea of an arbiter PUF [20] is to introduce a race condition on two paths and an arbiter circuit is used to decide which one of the two paths reached first. The two paths should be designed identically, then the outcome of the race will be unpredictable due to the inevitable manufacturing variations.

An arbiter PUF is often used as an authentication primitive because it is fast and has large CRP space. It is known to be vulnerable to model building attacks [21], but more complex configurations and restricted access in real application can reduce the risk greatly [22].

Please note that the comparison of the two path delays are done in a post processing procedure, so the arbiter circuit is assumed to be ideal. For real arbiter circuit such as a D flip-flop or a SR latch, a measurement noise coming from metastable behaviors will be discussed in section 2.4.

Figure 2.2 shows the distribution of inter-distance gathered from 36 thousand arbiter

PUFs fabricated by a commercial 65nm technology process. The distribution is closely fit to a normal distribution with a FHD mean of 0.4998 and variance of 0.0297.

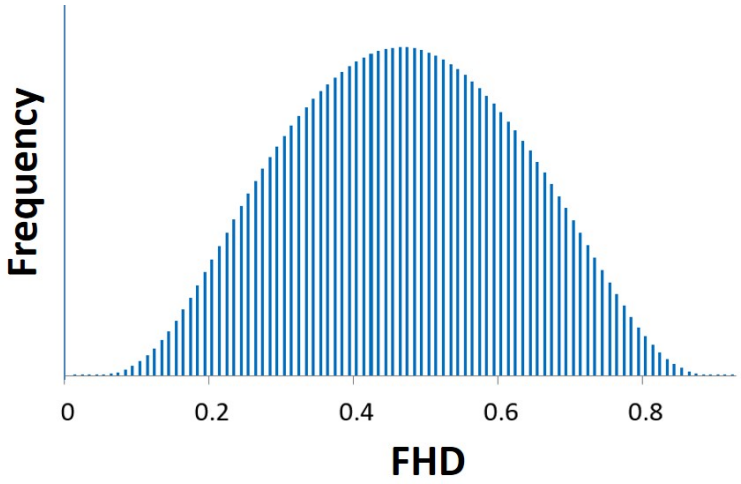


Figure 2.2: The FHD inter-distance of 36 thousand arbiter PUFs with mean=0.4998 and variance=0.0297.

2.2.3.2 Ring Oscillator (RO) PUF

A RO PUF consists ROs with the same intended frequency implemented in parallel. Similar to arbiter PUF, the two frequencies will be slightly different due to process variations, and a one-bit response is produced by comparing the frequencies [9]. RO PUF is often used as static cryptographic key generation because of its high cost of timing and power for each response generation.

We measure the inter-distance of 6 RO PUFs implemented as described in Section 2.1. Results shows that the mean of inter-distance is 54.36%.

For intra-distance measurement, each pair of ROs is measured 300 times over a 15-hour period. The results given in Table 2.1 which show that intra-distances varies from 0.3% to 18%. This could be due to many reasons including temperature or voltage variations, measurement uncertainties, or random noises. Detailed discussion on the sources of intra-distance noises will be presented in section 5.

Table 2.1: RO PUF Measurement

	Biasing Level	Intra-Distance (mean)
PUF 1	84.6%	18.0%
PUF 2	100.0%	6.6%
PUF 3	76.9%	4.1%
PUF 4	61.5%	6.6%
PUF 5	76.9%	12.2%
PUF 6	100.0%	0.3%

In addition to intra-distance, Table 2.1 also includes the biasing measurement, which is defined as the percentage of the majority bit value in a response. For example, PUF6 has a 100% biasing measurement, meaning that all the bits in the response are either ones or zeros. The results show that low intra-distance is an indication of high global correlation. This is an intuitive and interesting phenomenon. Small intra-distance means that the response is stable and resilient to noises, but it also means that the intrinsic difference is due to strong systematic correlations. In our RO PUF layout implementation, 26 ROs are evenly placed in two blocks, where a RO pair is formed by selecting two ROs from each of the block. It is possible that ROs from a block is always faster than ROs from the other block due to systematic wafer variation. Of course this biasing behavior can be eliminated or hidden by a more complex layout consideration, however, the physical silicon systematic variation will still occur and can be exploited as a side channel attack.

2.3 Process Side Channel Attacks

2.3.1 Variation models

To study the impact of silicon variation on delay-based PUFs, we adopt the models presented in [23] to represent the variations of inter-wafer, inter-die, intra-wafer, and intra-die of a commercial 65nm technology process. Each variation is represented by the variance of the

corresponding data. Among all the sources of variations, intra-die variation, or also known as local variation, is the only desired entropy source of a PUF. The variation calculations are given below:

- Total variation σ^2_{total}: Total variation is the sum of all the variations. The mean $m(total)$ is mean of all data, and the variance σ^2_{total} of all the data is the total variation.

- Intra-die variance σ^2_{local}: Each die is located on a (X,Y) location of a wafer. We first calculate the mean $m(w, x, y)$ of each die on each (X,Y) location to eliminate the effect of intra-die local variations. Then we obtain the variance of these means to get σ^2_{global}. The intra-die variation is then calculated as below:
  $\sigma^2_{local} = \sigma^2_{total}$ - σ^2_{global}.

- Inter-wafer variance σ^2_{wafer}: Calculate the mean of each wafer $m(w)$, and inter-wafer variation σ^2_{wafer} is the variance of these means.

- Intra-wafer variance σ^2_{XY}: Intra-wafer variation is a systematic variation that can be modeled by the (X,Y) location on a wafer. For each (X,Y) location, we calculate the mean $m(X, Y)$ for all wafers, and the variance σ^2_{XY} is the intra-wafer variation.

- Inter-die variance σ^2_{die}: To obtain inter-die variation, we need to subtract inter-wafer and intra-wafer variations from σ^2_{global}. To model systematic correlations on (X,Y), we use
$$fitting(X, Y) = aX^2 + bY^2 + cX + dY + eXY + f \tag{2.1}$$
adopted from [23]. Then the inter-die variation σ^2_{die} is defined as the variance of $m(w, X, Y) - fitting(X, Y) - m(w)$.

In summary, the total variation is the summation of the four independent variations:

$$\sigma^2_{total} = \sigma^2_{wafer} + \sigma^2_{die} + \sigma^2_{XY} + \sigma^2_{local}$$

The values of standard deviations are given in Table 2.2. Table 2.3 compares the mean of inter-distance of the PUFs that are randomly selected, fabricated on the same wafer, and

Table 2.2: Process Variations

	Description	Variance
σ^2_{wafer}	Inter-Wafer	0.0018730
σ^2_{die}	Inter-Die	0.0015720
σ^2_{XY}	Intra-Wafer	0.0007158
σ^2_{local}	Intra-Die	0.0006204

Table 2.3: Intra-distances Comparison

	FHD mean
Randomly Selected PUFs	0.4998
PUFs on the same wafer	0.4522
PUFs on the same (X,Y)	0.2886

fabricated on the same (X,Y) location on different wafers. Figure 2.3(a) shows the inter-distance of PUFs selected from a same (X,Y) location on different wafers and Figure 2.3(b) shows the inter-distance of PUFs selected from same wafer. The mean of the randomly selected PUFs is close to 0.5 as labeled in the figure. It obvious that the mean of inter-distance on the same (X,Y) is smaller than the mean of inter-distance on a same wafer. Therefore, for silicon fabrication side channel attacks, an adversary with possession of a reference PUF which is fabricated at the same (X,Y) location as the target PUF would have a higher probability to guess the correct answer than random guessing or with possession of a reference PUF that is fabricated on the same wafer as the target PUF.

When measuring inter-distance of two PUFs on a same wafer, the variations are composed of σ^2_{local}, σ^2_{die}, and σ^2_{XY}. On the other hand, when measuring inter-distance of two PUFs on a same (X,Y) location, the variances are σ^2_{die} and σ^2_{local}. Less variation sources explains why PUFs fabricated on a same (X,Y) location are more similar to each other than PUFs fabrication on a same wafer. In our experiment, the intra-die variation is the smallest variation of all. The silicon side channel attacks would be less effective than we demonstrated

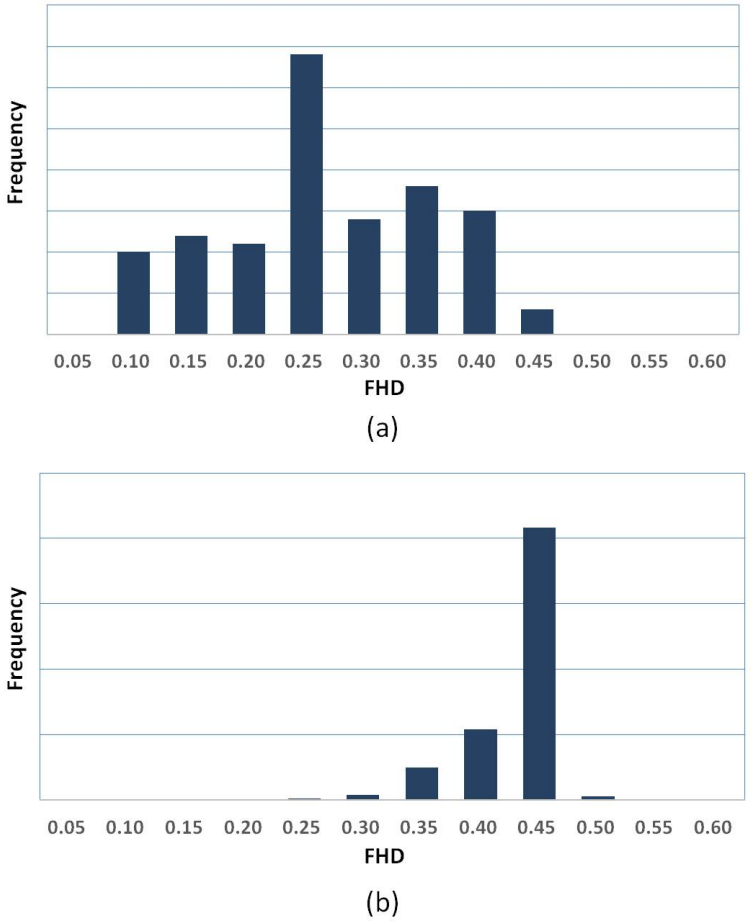


Figure 2.3: The FHD inter-distance of (a) PUFs from same (X,Y) location on different wafers (b) PUFs from a same wafer

if the intra-die variation, which is the true random local variation, is the dominating variation of the total variation.

2.3.2 Process Side-Channel Attacks

From the silicon variation model and our experiments, we know that it is possible for an adversary to exploit silicon fabrication side-channel information to attack delay-based PUFs. One way is to use reference PUFs that are fabricated on the same wafer as the target PUF to predict the behavior of the target PUF. In this kind of attack, the adversary should keep at least one reference PUF from each wafer, and the probability of predicting each bit response correctly is slightly better than not using silicon side-channel attack at all. A

better way of using silicon side-channel attack is to have access of a reference PUF that is fabricated on another wafer but on the same (X,Y) location on a wafer as the target PUF. The probability of guessing the correct bit response is much higher than the first attack as shown in Table 2.3. In addition, since the intra-wafer variation is a parabolic function, the adversary can just keep several reference PUFs with different distances to the center of the wafer instead of keeping all PUFs on a sacrificial wafer. This would be helpful to the adversary because the number of reference PUFs needed to predict the target PUF is reduced, and the exact fabrication information of the (X,Y) location of the target PUF is not needed, which otherwise would be difficult to obtain. The adversary then compares which one of the reference PUFs has the smallest distance to the target PUF. The reference PUF or the interpolation of the two reference PUFs can be used to predict the behavior of the target PUF if the model $fitting(X, Y)$ is also known to the adversary. If more overhead is acceptable, then adversary could keep few reference PUFs on every wafer. An example is shown in Figure 2.4. Reference PUFs are selected at a fixed distance along a straight line across the wafer center. Because of the radial nature of the variation, the adversary can identify one of the reference PUFs that best matches the behavior of the target PUF.

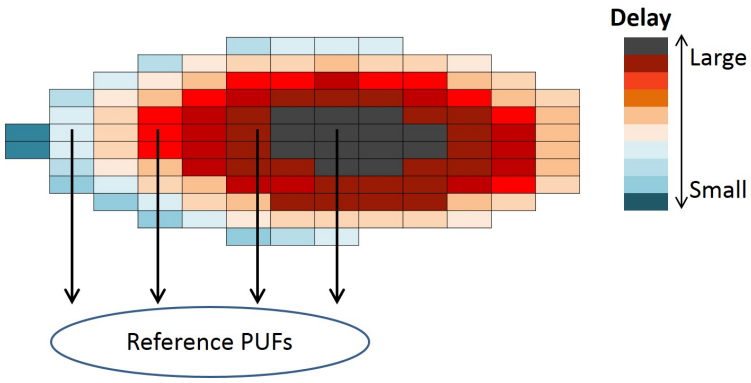


Figure 2.4: (X,Y) variation shows a parabolic shape with the peak at the center of the wafer.

2.4 Stability model

One of the most important properties of a PUF is the reliability. When A PUF is given a same challenge multiple times, the responses should be as close to each other as possible. However, PUF responses are much more instable than classical digital systems because PUFs usually operate more closely at their stability limits. There are many sources that can cause a PUF to be unreliable. In our work, we focus on two sources of instabilities that have provisionally been ignored.

2.4.1 Metastability of the arbiter circuit

For an arbiter PUF, the arbiter circuit is usually a D flip-flop or a SR latch. If two signals arrive at an arbiter within a short time, the arbiter circuit may enter a metastable state due to setup time violation [24]. Once the arbiter circuit is in metastable state, the response becomes unstable. The comparison done in [25] illustrates that SR latch is a better arbiter circuit than D flip-flop because its smaller setup time window, and the arbiter circuit should use up-sized transistors to minimize the metastable window. The metastable window for an arbiter circuit is calculated as:

$$\delta = T_0 e^{\frac{-T_r}{\tau}}$$

where T_r is the required resolving time, T_0 and τ are circuit dependent parameters [24].

One common technique to achieve better stability of a arbiter PUF is to select the paths that intrinsically have large delay difference. Assume that metastability is the only source of intra-distance. To eliminate the inconsistency caused by metastability of the arbiter circuit, one way is to choose the paths that have a delay difference larger than the metastable window δ. As shown in Figure 2.5, two delay paths are assumed to have a same normal distribution $N \sim (m, \sigma^2)$ with mean m and process variance σ^2. The difference of the two paths will then also be a normal distribution with $N \sim (0, 2\sigma^2)$. Let δ be the difference threshold such that a pair of paths will only be selected as an input challenge when the difference is larger

than δ. The probability that a CRP be discarded as illustrated in Figure 2.5 is given below:

$$P[discarded] = 2 \times \Phi(\frac{\delta}{\sqrt{2}\sigma}) - 1$$

where the cumulative function $\Phi(x)$ gives the probability of the interval $[-\infty, x]$ of the normal probability density function.

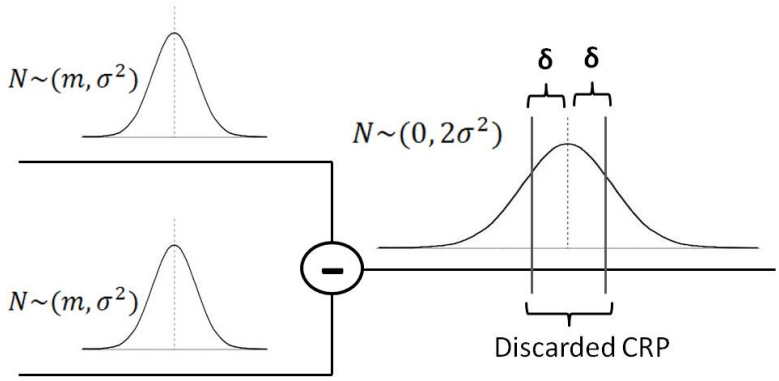


Figure 2.5: The difference of two normal distributions $N \sim (m, \sigma^2)$ is another normal distribution $N \sim (0, 2\sigma^2)$.

Consider a simple example where for a D flipflop the metastability window is 10ps, delay chain nominal delay of 1ns and local variation standard deviation of 2.5%, the fraction CRPs that would be discarded due to metastability would be about 11%. One easy way to improve this is to design longer delay chains such that their absolute delay variance is larger. Of course, one still has to make sure that the layout remains compact, otherwise, the arbiter PUF becomes susceptible to process side channel attack as discussed earlier.

2.4.2 Jitter accumulation

In this section, we analyze the level of impact of jitter as a noise source of RO PUF intra-distance. Jitter is known as frequency/phase instability due to the delay variance of logic gates connected into a ring, thus is considered to be a possible source of the RO PUF intra-distance. One of the most important aspects of the jitter is its accumulation in time. For free-running ROs in a RO PUF, the variance of accumulating jitter is proportional to the

square root of the measuring time as given below:

$$\sigma_{ac} = \sqrt{2kl}\sigma_g$$

where k is the number of stages in the oscillator, l is the measuring time in average oscillating periods, σ_g is the random jitter deviation of each gate, and σ_{ac} is the accumulated random jitter deviation [26].

Let RO_1 and RO_2 be the two ROs to be compared to generate a bit response, and let T_1 and T_2 be the oscillating period of RO_1 and RO_2, respectively. As an example of RO_1 shown in Figure 2.6, the counter is enabled for a length of T_m time, so the number of counts of RO_1 and RO_2 is $\lfloor \frac{T_m}{T_1} \rfloor$ and $\lfloor \frac{T_m}{T_2} \rfloor$, respectively.

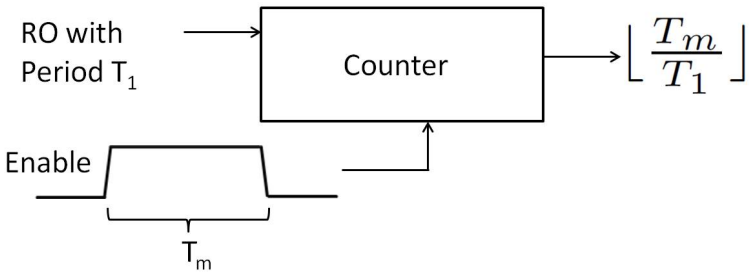


Figure 2.6: The counter is enabled for a length of T_m time and counts the number of cycles of RO_1.

Assume that T_m is large enough that the accumulated jitter of RO_1 and RO_2 can be approximated as:

$$\sigma_{acr1} = \sqrt{2k\frac{T_m}{T_1}}\sigma_g$$
$$\sigma_{acr2} = \sqrt{2k\frac{T_m}{T_2}}\sigma_g$$

For RO_1, this is equvalent to that T_{m1} becomes a random variable with $N \sim (T_m, \sigma^2_{acr1})$, thus $\frac{T_m}{T_1}$ becomes a normal distribution with $N \sim (\frac{T_m}{T_1}, \frac{\sigma^2_{acr1}}{T_1^2})$. Same for RO_2. Therefore, to find the probability of the arbiter value $sign(\lfloor \frac{T_m}{T_1} \rfloor - \lfloor \frac{T_m}{T_2} \rfloor)$ changes due to jitter, a new distribution formed subtracting random variable $\frac{T_m}{T_2}$ from $\frac{T_m}{T_1}$ is given as:

$$N \sim (T_m(\frac{T_1 - T_2}{T_1 T_2}), 2kT_m\sigma_g^2(\frac{1}{T_1^3} + \frac{1}{T_2^3}))$$

The probability that the arbiter value is different from the majority value is thus derived as:

$$P[jitter\_error] = \Phi(-\frac{|T_1 - T_2|}{\sigma_g}\sqrt{\frac{T_m T_1 T_2}{2k(T_1^3 + T_2^3)}})$$

As shown in Figure 7, the shaded regions represents the probability $P[jitter\_error]$ when T_1 is larger than T_2. As T_m gets larger, the mean $T_m(\frac{T_1-T_2}{T_1T_2})$ becomes farer from the origin, which makes $P[jitter\_error]$ smaller. For a fixed T_1 and T_2. We can see that $P[jitter\_error]$ becomes 0.5 in an extreme case when $T_1 = T_2$

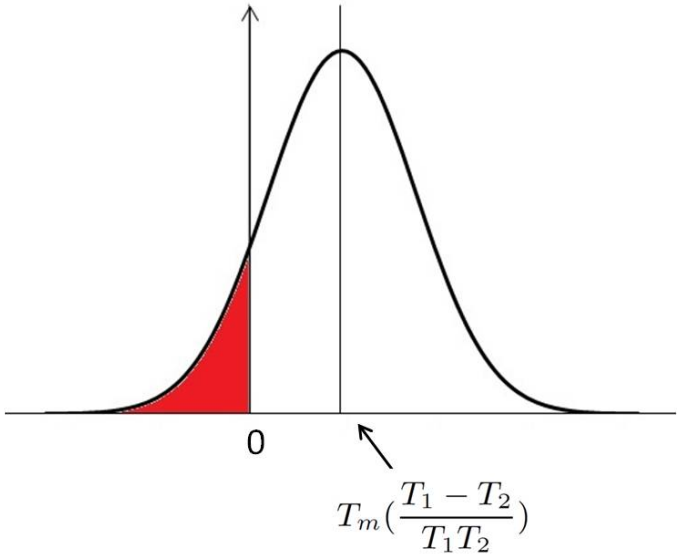


Figure 2.7: The shaded region gives the P[jitter_error] when T_1 is larger than T_2.

Consider a simple example with σ_g=35ps, k=7, and T=10ns from [26], assuming that the difference between T_1 and T_2 is 0.5%. For T_m=100ns, which is roughly 10 cycles, we obtain $P[jitter\_error]$=0.2033, and for T_m=1000ns, we get $P[jitter\_error]$=0. This shows that as the measurement time gets larger, $P[jitter\_error]$ becomes smaller, however the timing and power cost for response generation also gets larger.

2.5 Conclusions

From the experiments, only 13% of the total silicon process variation is the local variation that can really be the entropy source of a delay-based PUF as given in Table 2.2. This leads

to possible silicon process side-channel attacks and we show that two PUFs on the same (X,Y) location on different wafers are highly correlated. Also, since the portion of local variation is relatively low compared with other sources of variation, our results in Table 2.1 shows that the low intra-distance could be coming from strong systematic correlations, which will make the responses biased and with low randomness. Techniques such as entropy distillation [15] can be applied to minimize the global and systematic wafer variation to further increase the PUF security level. Unfortunately, the local variation has to be much larger than sources of noise combined including temperature/voltage variations, wear-out, jitters, and metastability. Given the relatively modest amount of local variations visible in delay, further careful analysis and modeling to combine effects of causes of reduced stability of delay-based PUFs is necessary and part of our ongoing work.

To conclude this chapter, we measured uniqueness and stability of SRAM PUFs, arbiter PUFs, and RO PUFs based on silicon results. We also show that a delay-based PUF are susceptible to silicon side-channel attacks because of systematic fabrication variations. Finally, we model two sources of instabilities of delay-based PUFs: metastability of the arbiter circuit, and jitter accumulation of RO. We calculate the probability of a unstable CRP being discarded due to metastability and suggest that longer (but compact in layout extent) delay chains is one possible way to reduce impact of metastable comparisons. For the impact of random jitter, we derived the probability of a jitter caused measurement error which shows that increased measurement time is a viable way to decrease such error probability. In Chapter 3 we will discuss a physical implementation agnostic PUF that is immune to biased wire delay and systematic variations but is purely based on local process variations. In Chapter 4 we propose two stable sources of randomness to construct stability-guaranteed PUFs that do not suffer from unstable CRPs.

CHAPTER 3

UNBIAS PUF: A Physical Implementation Bias Agnostic PUF

[1] The Physical Unclonable Function (PUF) is a promising hardware security primitive because of its inherent uniqueness and low cost. To extract the device-specific variation from delay-based PUFs, complex routing constraints are imposed to achieve symmetric path delays; and systematic variations can severely compromise the uniqueness of the PUF. In addition, the metastability of the arbiter circuit of an Arbiter PUF can also degrade the quality of the PUF due to the induced instability. In this chapter we propose novel weak and strong UNBIAS PUFs that can be implemented purely by Register Transfer Language (RTL), such as verilog, without imposing any physical design constraints or delay characterization effort to solve the aforementioned issues. Efficient inspection bit prediction models for unbiased response extraction are proposed and validated. Our experimental results of the strong UNBIAS PUF show 5% intra-Fractional Hamming Distance (FHD) and 45% inter-FHD on 11 Field Programmable Gate Array (FPGA) boards without applying any physical layout constraints or additional XOR gates. The UNBIAS PUFs are also scalable because no characterization cost is required for each challenge to compensate the implementation bias. The intra-FHD measured at a high temperature condition is 8%, which is still well below the margin of conventional Error Correction Code (ECC) with error reduction techniques for PUFs.

[1]This work is done in collaboration with Michael Chen and his team at Mentor Graphics. Their contribution is much appreciated.

3.1 Introduction

Physical Unclonable Functions (PUFs) [9] have been considered as promising security primitives for the Internet of Things (IoT) where the security between untrusted subsystems is known to be of essential importance [8]. PUFs can also be exploited in a variety of applications, such as identification [27] or secret key generation [28]. Silicon delay based strong PUFs have been studied intensively since its first appearance in [9] because of its low implementation cost and large CRP space compared with a weak PUF [29]. Over the years, a lot of effort has been devoted to achieve better security, stability, and efficient entropy extraction of the PUFs [30, 31]. However, there are still design challenges that restrain a strong PUF from being put in a widespread practical use.

One of the major design challenges for a silicon delay based PUF is the strict symmetric delay path layout requirement. The wire delays of the competing paths should be designed and matched carefully to avoid biased responses both for Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC) designs, otherwise low inter-chip uniques would make the PUF unusable [32, 33]. In addition to asymmetric routing, another source of the biased responses for silicon based PUF is the systematic process variation, which can also degrade the quality of a PUF, such as uniqueness or unpredictability. Finally, the metastability issue of the arbiter circuit for an Arbiter PUF can cause unstable PUF responses, making a portion of the CRP unusable due to their instabilities [34].

3.1.1 Asymmetric Path Delay Routing

For a delay based PUF, the randomness should be contributed only by the subtle variations between devices, so having biased delay differences due to asymmetric routing is detrimental to delay based PUFs, and such impact should be eliminated. However, a precise control of the routing can be a difficult and time consuming task.

To validate how severe the biased routing delay could be, we implemented a simple Ring Oscillator (RO) PUF [9] on two Xilinx Artix-7 boards without imposing any routing constraints. The RO PUF consists 72 pairs of ROs, where each RO is composed of 3 inverters

as shown in Figure 3.1. The wire delays in nanosecond reported from the board of a RO pair implementation are also included in the figure. From our experiments, the inter-Fractional Hamming Distance (FHD) [3] of the 72-bit responses from the two boards is only 6%, which means that this design is essentially only a constant number generator but not a PUF. The reason is that the frequency is dominated by the wire delay, which ranges from 0.199ns to 0.710ns as Figure 3.1 shows. Since the wire delay is already comparable to the inverter delay [35], the logic gate variations can be easily masked. This experiment shows that biased delay can be harmful to delay based PUFs.

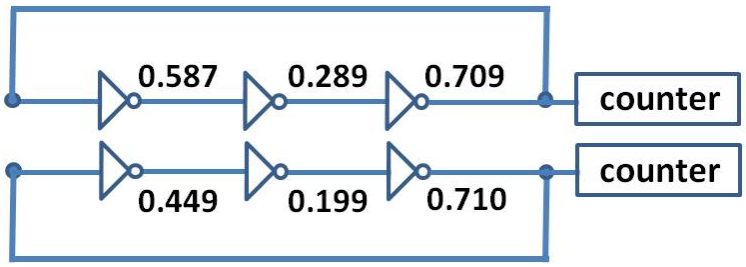


Figure 3.1: RO PUF implemented on two FPGAs. The numbers indicate the wire delay in nanoseconds, where the bias is so severe that the local process variation cannot be observed.

An implementation of an Arbiter PUF on FPGA is considered more difficult than a RO PUF because the connections to the arbiter circuit must also be symmetric [36], and performing completely symmetric routing is physically infeasible in most cases [33], resulting small inter-FHD for an Arbiter PUF [37]. One of the most common solutions to the asymmetric routing is to use hard-macros in FPGA designs [31, 38], but it could be complicated, and some less commonly-used features of the FPGA would be required especially for Arbiter PUFs [39]. Other approaches try to extract randomness by XORing the outputs of multiple Arbiter PUFs at the cost of large hardware overhead and less stability [40]. In [33, 41], the authors try to insert configurable delay modules or programmable delay lines to balance out the biased delay difference. However, a certain level of control of the routing is still required because each delay unit can only provide limited adjustment, and each challenge should have different delay module configurations, which limits the scalability for strong PUF designs. In [42], the authors proposed using 'middle' bits instead of the Most Significant Bit (MSB) as

the RO PUF response measurement. The measurement can effectively eliminate the biased responses, but an efficient way of predicting the inspection bit is not described, and the measurement can only work on a weak PUF but not a strong PUF, which provides much more CRPs and applications [29]. A RTL-based PUF bit generation unit was proposed in [43], but to the best of our knowledge, a strong PUF that can be implemented efficiently without any layout constraints has not yet been proposed.

3.1.2 Systematic Process Variation

The existence of systematic process variation can degrade the quality of silicon based PUFs because the local randomness should be the only desired entropy source of the delay based PUF [15]. For example, different ROs fabricated on different regions on a die may have different intra-wafer systematic variation, which can make ROs on a certain region of a die always oscillate faster than ROs on another region [31]. The effect of systematic process variation is similar to having biased wire delay between two ROs, which can also damage the uniqueness of the PUF. Another possible vulnerability caused by systematic variation is the induced process side channel attack as described in [44]. Due to intra-wafer systematic variation [23], PUFs fabricated at the same region on different wafers can have similar systematic behavior, which can be exploited as a process side channel attack.

To account for systematic variations, a compensation technique is proposed in [31], which requires careful design decisions to compare RO pairs that are physically placed close to each other. Many other techniques to extract true random local variation have also been proposed. In [15], the systematic variation is modeled and subtracted from the PUF response to distill true randomness with the cost of model calculation. Similarly, in [30], the averaged RO frequency is subtracted from the original frequency, where the multiple measurements of each RO can lead to large latency overhead. In [45], a method is proposed to extract local random process variation from total variation, however, a second order difference calculation is needed, and hard-macro technique must be applied to construct symmetric delay paths.

3.1.3 Metastability of the Arbiter Circuit

The idea of an Arbiter PUF is to introduce a race condition on two paths and an arbiter circuit is used to decide which one of the two paths reached first. The two paths should be designed identically, then the outcome of the race will be unpredictable due to the inevitable manufacturing variations. The arbiter circuit is usually a D flip-flop or a SR latch. If two signals arrive at an arbiter within a short time, the arbiter circuit may enter a metastable state due to setup time violation [24]. The metastable window δ for an arbiter circuit can be calculated as:

$$\delta = T_0 e^{\frac{-T_r}{\tau}}$$

where T_r is the required resolving time, T_0 and τ are circuit dependent parameters [24]. Once the arbiter circuit is in metastable state, the response becomes unstable.

To eliminate the inconsistency caused by metastability of the arbiter circuit, one way is to choose the paths that have a delay difference larger than the metastable window δ at the cost of CRP characterization and discarding the unstable CRPs [34]. To estimate the probability of unstable CRPs due to metastability, two delay paths are assumed to have a same normal distribution $N \sim (m, \sigma^2)$ with mean m and process variance σ^2 as shown in Figure 3.2. The difference of the two paths will then also be a normal distribution with $N \sim (0, 2\sigma^2)$. Let δ be the difference threshold such that a pair of paths will only be selected as an input challenge when the difference is larger than δ. The probability that a CRP be discarded as illustrated in Figure 3.2 is given below:

$$P[discarded] = 2 \times \Phi(\frac{\delta}{\sqrt{2}\sigma}) - 1$$

where the cumulative function $\Phi(x)$ gives the probability of the interval $[-\infty, x]$ of the normal probability density function. Consider a simple example where for a D flipflop the metastability window is 10ps, delay chain nominal delay of 1ns and local variation standard deviation of 2.5%, the fraction CRPs that would be discarded due to metastability would be about 11%.

In this chapter, we propose the physical implementation bias agnostic (UNBIAS) PUF

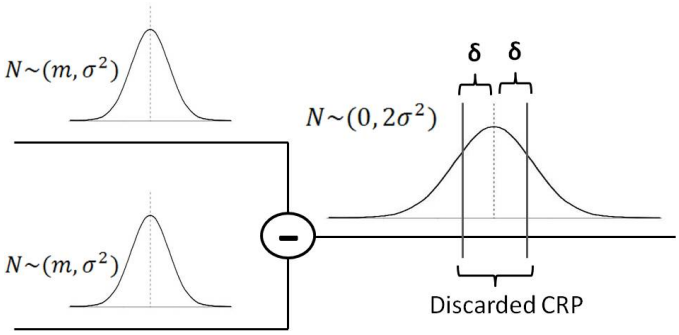


Figure 3.2: The difference of two normal distributions $N \sim (m, \sigma^2)$ is another normal distribution $N \sim (0, 2\sigma^2)$.

that is immune to physical implementation bias and metastability issues. The contributions of this chapter include:

- The proposed weak and strong UNBIAS PUFs can be implemented purely by RTL without imposing any physical routing constraints.
- Efficient inspection bit selection strategy based on intra-/inter-FHD prediction models are proposed and verified on both UNBIAS PUFs.

The rest of the chapter is organized as follows: In Section 3.2, we introduce the proposed weak UNBIAS PUF and strong UNBIAS PUF. In Section 3.3, the concept of the inspection bit is described with unbiased and biased delay examples. In Section 3.4, we propose intra-FHD and inter-FHD prediction models for inspection bit identification. Finally, the experimental results are presented in Section 3.5, and we conclude the chapter in Section 3.6.

3.2 Proposed UNBIAS PUF Structures

3.2.1 Weak UNBIAS PUF

The weak UNBIAS PUF is essentially a basic RO PUF without any layout constraints, and the final one-bit response will be selected from the difference register, which is a multi-bit

long register. As shown in Figure 3.3, the selection logic selects the RO pair for comparison according to the challenge, and the counter counts the RO frequency for a fixed period of time. The difference value is then stored in the difference register to generate the final response.

Please note that the weak UNBIAS PUF is similar to previous work presented in [42]. The main purpose of the weak UNBIAS PUF is to validate the proposed prediction models presented in Section 3.4.

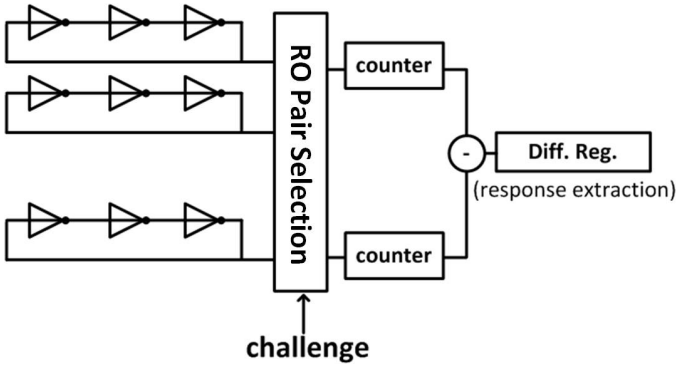


Figure 3.3: The proposed Weak UNBIAS PUF structure without any symmetric routing constraints.

3.2.2 Strong UNBIAS PUF

The proposed strong UNBIAS PUF compares two delay paths to generate PUF responses. Similar to Arbiter PUF, each bit of the challenge of the UNBIAS PUF specifies the path configuration of the delay path. As shown in Figure 3.4, the challenge c1 and c2 specify the path configurations, and an one-bit response is extracted from the difference register, which can be of several bits long. Once a challenge is given, a signal is applied at Trigger. The two Clock counters begin to count the number of clock cycles of the system clock (CLK) whenever the the signal from Trigger is propagated to the wire START, and stop counting whenever the the signal from Trigger is propagated to the wire STOP. For each challenge, the difference value of the two Clock counters is then stored in the difference register for further response extraction, which is described in details in Section 3.3.

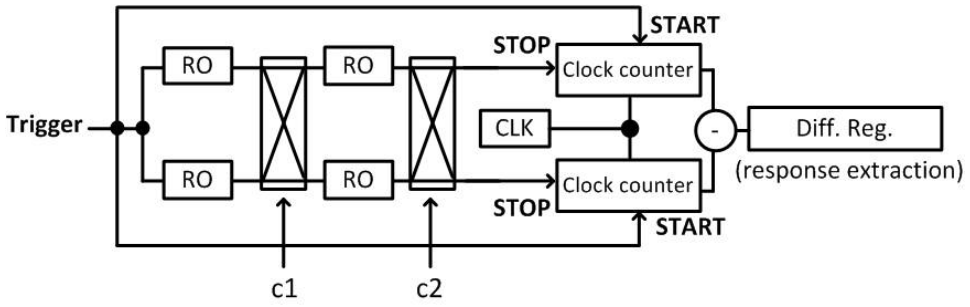


Figure 3.4: The proposed strong UNBIAS PUF. The Clock counter starts counting clock cycles of the system clock (CLK) when START arrives and stops when STOP arrives. The difference of two Clock counters are stored in the difference register for further response extraction.

The purpose of the ROs inserted between path configurations is to increase the path delay so that it will take multiple clock cycles for the signal to propagate to stop the clock counter. Without the ROs, the delay would be too short and the clock counters would become unstable because STOP would arrive quickly after START. As shown in Figure 3.5, each RO is associated with a RO counter that counts the number of oscillations of the RO. The RO counter starts counting when the signal from its previous path configuration is arrived, and propagates the signal to the next path configuration only when the count reaches a certain threshold. All the ROs are composed of same number of inverters and neither configurations nor any layout constraints are needed.

Unlike the conventional Arbiter PUF, the strong UNBIAS PUF has no metastability issues caused by a D flip-flop or a latch. The delay difference of the two paths is transformed into counter values of the system clock. By judiciously extracting the response from the difference register, the physical implementation bias can be effectively mitigated, therefore the UNBIAS PUF can be implemented purely by RTL without any routing or layout constraints. Details of the response extraction are described in Section 3.3.

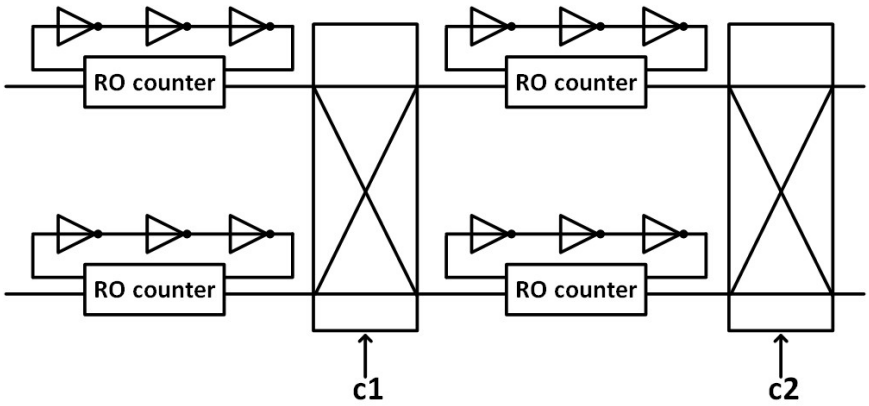


Figure 3.5: ROs are inserted between path configurations to increase the path delay for better stability. The signal from previous path configuration is propagated only when the count of the RO counter reaches a certain threshold.

3.3 Bias-Immune Response Extraction

3.3.1 Modulo-like Operation

For the difference register described in Section 3.2, using a *middle* bit as the inspection bit to generate the final one-bit response [42] can effectively bypass biased responses because it essentially performs a modulo-like operation to the value stored in the register. Define bit_i as the i^{th} bit of the register and N as the decimal value of the register. For example, the Least Significant Bit (LSB) is bit_0. Assuming $N > 0$, the value of bit_i is given below:

$$bit_i = \begin{cases} 0, & \text{if } 0 \leq (N \mod 2^{i+1}) < 2^i \\ 1, & \text{otherwise} \end{cases} \tag{3.1}$$

Figure 3.6 shows an example of why performing modulo operation with a proper divisor can help to compensate the biased delay and expose the desired gate delay variation. If the delay values are directly compared in Figure 3.6, the upper path delay is always larger than the lower path delay due to the significant bias. However, after performing modulo with a proper divisor (4 in the example), the subtle delay difference between the gates are revealed, so the two PUFs can be distinguished.

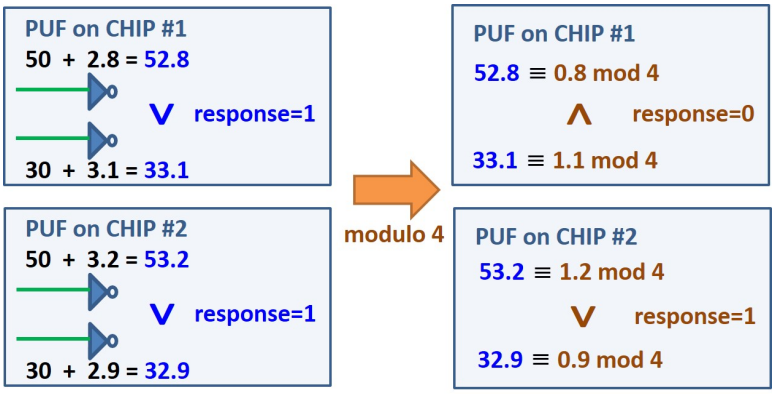


Figure 3.6: Gate delays and biased wire delays are shown. Before the modulo operation the two PUFs have the same response. After the modulo operation, the effect of the biased wire delay is mitigated the responses of the two PUFs are different.

3.3.2 Inspection Bit on Unbiased/Biased Paths

In this section we describe how different selections of the inspection bit can change the intra- and inter-FHD. Figure 3.7 shows an example of a distribution of values from difference registers of symmetrically routed UNBIAS PUFs. The length of the difference register is 22-bit, so the range of the register value is between -2^{21} and $2^{21} - 1$ as represented in 2's-complement. The large inter-chip measurement curve gives the distribution of the values across all PUFs. Since the PUF is unbiased, roughly half of the difference values would be greater than zero due to random local process variation, therefore the inter-FHD of the UNBIAS PUFs would be close to 50%. In this case, the inspection bit is simply the MSB, which divides the range of 22-bit difference value into two groups *bin_1* and *bin_0*. All measurements fall into *bin_1* on the left output a 1; others output a 0. The small intra-chip measurement curve gives the distribution of multiple measurements of the PUF on a same chip. Due to noise, the difference values could be different, so the intra-FHD of the difference register may not be a perfect 0%.

Even though symmetric UNBIAS PUF layout is much preferred, it is difficult and takes much effort and overhead to achieve such requirement as described in Section 3.1.1. In practice, if no layout constraints are imposed, the measurement distribution of the difference

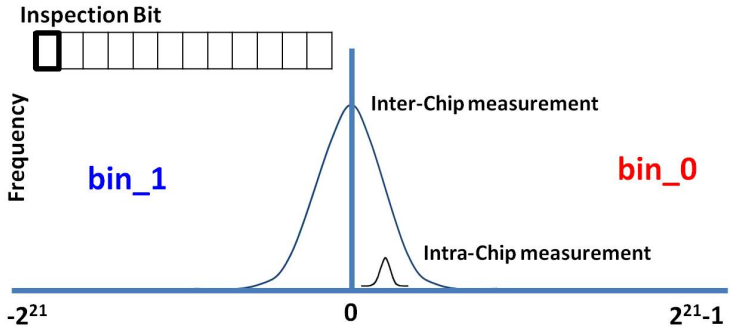


Figure 3.7: For a symmetrically routed PUF, the inter-FHD would be close to 50%. The intra-FHD may not be zero due to measurement noise.

register can be as shown in Figure 3.8, where most of the difference values across chips are greater than zero. In this case, using the MSB as the inspection bit would cause low inter-FHD of the PUFs because most MSBs are 0's.

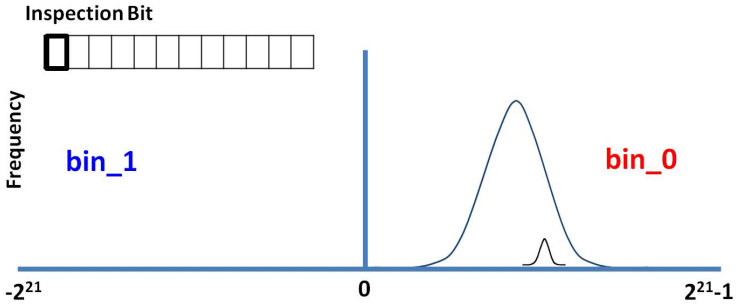


Figure 3.8: For a biased PUF, most of the difference values across all chips could be greater than zero, causing a low inter-FHD if the MSB is the inspection bit.

For the same biased distribution shown in Figure 3.8, if the i^{th} bit is used as the inspection bit of the difference register as Figure 3.9 shows, the range of the 22-bit difference value is divided into multiple bins with width 2^i, where the output of the measurement is decided by the bin in which it resides. Note that in this case the response is not an indicator of which delay is longer in the comparison. The smaller the width of the bin is, the closer the inter-FHD is to 50% because roughly half of the outputs would reside in bin_1 even with biased delay. On the other hand, the width of the bin should be large enough so that multiple measurements of a same PUF should always fall into the same bin. In other words, the width

of the bin should be larger than the variation of the intra-chip measurement distribution. Therefore, the choice of inspection bit is a tradeoff between inter-FHD and intra-FHD for a PUF with asymmetric routing.

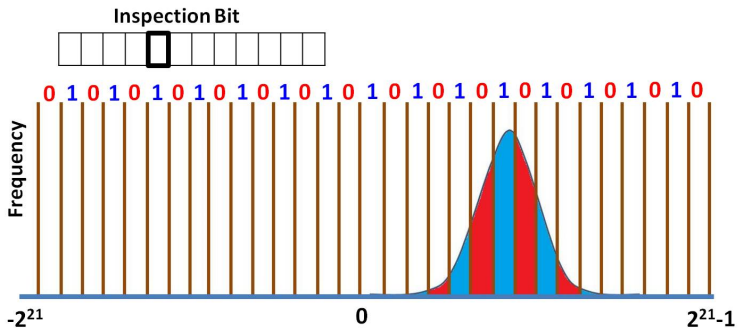


Figure 3.9: For an asymmetrically routed PUF with proper inspection bit, roughly half of the difference values across all chips would fall in bin_1, therefore the inter-FHD would be close to 50%.

3.4 Inspection Bit Identification

3.4.1 Intra-FHD Prediction Model

The intra-FHD depends on the width of the bins $w = 2^i$ when the inspection bit is bit_i. A straightforward way to determine the associated intra-FHD for each inspection location is to gather multiple measurements of the same challenge on a same PUF, and simply calculate the intra-FHD for each bit_i. A more efficient approach is to predict the intra-FHD without calculating it for each bit_i.

To predict intra-FHD$_k$ of a challenge C_k of an inspection bit, we first obtain t measured difference registers of the challenge C_k of a same PUF. Since the bin width and the range of the difference register is known, the t difference values can be divided into two groups (responses) according to the bins they reside in. Let the number of difference values fall in bin_1 be n_{one}, and number of difference values fall in bin_0 be n_{zero}. n_{one} and n_{zero} represent the number of responses of the challenge C_k to be one and zero during the t measurements, respectively. Since the intra-FHD is essentially calculated from the response

difference between any two measurements, the predicted intra-FHD$_k$ is calculated as:

$$intra\text{-}FHD_k = \frac{n_{one} \times n_{zero}}{\binom{t}{2}} \times 100\%, \tag{3.2}$$

where the final predicted intra-FHD would be the averaged intra-FHD$_k$ of all challenges.

As shown in Figure 3.10, the expected intra-FHD$_1$ is 0% because all measurements fall in the same bin and $n_{one} \times n_{zero} = 0$. The expected intra-FHD$_2$ depends on the portion of measured values that fall in bin_1. With larger bin width w, it is more likely that all responses would fall into the same bin

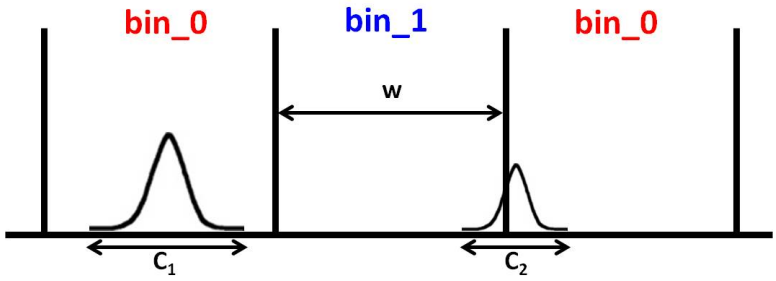


Figure 3.10: Magnified view of Figure 3.9 with three bins. w is the bin width and the measurement ranges for challenges C_1 and C_2 are specified. The expected intra-FHD$_1$ is 0% and the expected intra-FHD$_2$ depends on the portion of measured values that fall in bin_1.

3.4.2 Inter-FHD Lower Bound Prediction Model

The inter-FHD depends on the bin width w with a given inspection bit bit_i. Assume the distribution of inter-chip difference value is a Normal distribution $N \sim (\mu, \sigma^2)$. Define ϵ to be the distance between the mean μ and the closest bin boundary on the left as Figure 3.11 shows. We first prove that the worst-case inter-FHD happens when $\epsilon = 0.5w$, followed by the prediction model of the inter-FHD for the worst-case scenario.

3.4.2.1 Worst-Case Inter-FHD Identification

Given a fixed w, define $A_1(\epsilon)$ and $A_0(\epsilon)$ to be the total underlying area in bin_1 and bin_0 as functions of ϵ, respectively. For any Normal distribution, $A_1(\epsilon)$ and $A_0(\epsilon)$ are calculated as:

$$A_1(\epsilon) = \sum_{n=-\infty}^{\infty} F(-\epsilon + 2nw + w) - F(-\epsilon + 2nw) \tag{3.3}$$

$$A_0(\epsilon) = 1 - A_1(\epsilon, w) \tag{3.4}$$

where $F(\cdot)$ is the Cumulative Distribution Function (CDF) of the Normal distribution, and n is the index for bin area summation.

The ratio $Ratio(\epsilon)$ is defined as:

$$Ratio(\epsilon) = \frac{A_1(\epsilon)}{A_0(\epsilon)}, \quad 0 < \epsilon < w \tag{3.5}$$

where the range of ϵ is from 0 to w because of its periodic structure.

The closer the $Ratio(\epsilon)$ is to one, the closer the inter-FHD would be to 50% because the two areas are closer to each other. We want to show that the largest (most unbalanced) ratio happens at $\epsilon = 0.5w$ as Figure 3.11 shows.

To find the extreme value of $Ratio(\epsilon)$ given a fixed w, we take derivative with respective to ϵ of Equation 3.5 and replace $A_0(\epsilon)$ by $1 - A_1(\epsilon)$ from Equation 3.4:

$$\frac{d}{d\epsilon} Ratio(\epsilon) = \frac{A_1'(\epsilon)}{(1 - A_1(\epsilon))^2} \tag{3.6}$$

From Equation 3.6 we see that to find the extreme value of $Ratio(\epsilon)$, it is equivalent to find the solution of $A_1'(\epsilon)$, which is given below:

$$\frac{d}{d\epsilon} A_1(\epsilon) = \sum_{n=-\infty}^{\infty} f(-\epsilon + 2nw + w) - f(-\epsilon + 2nw) \tag{3.7}$$

where $f(\cdot)$ is the Probability Density Function (PDF) of the Normal distribution. Equation 3.7 shows that $A_1'(\epsilon)$ is the summation of differences between two PDF terms where one is a shifted version by w of another. Therefore, applying $\epsilon = 0.5w$ to Equation 3.7, we get a zero. Figure 3.11 shows that when $\epsilon = 0.5w$, each difference term in Equation 3.7 has its counter part at the mirrored location to the center, so that the summation becomes zero.

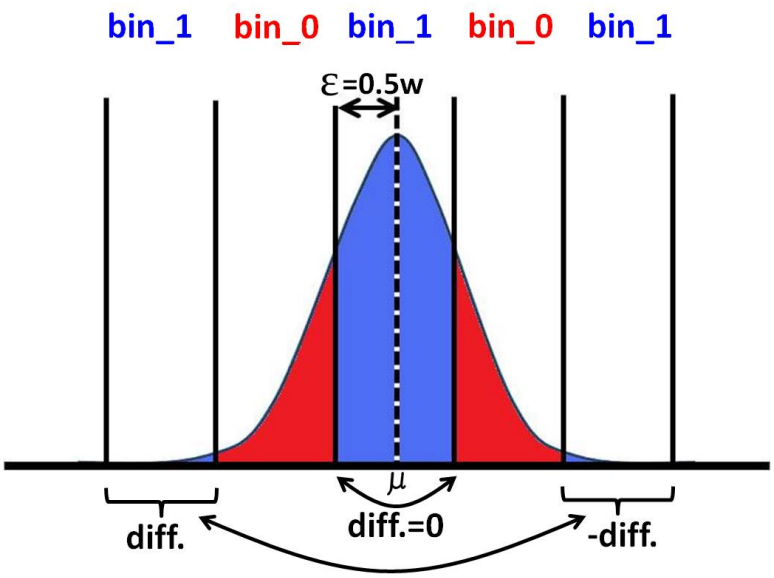


Figure 3.11: Worst Inter-FHD happens when the mean is at the middle of a bin.

To conclude our derivation, given a w of an inspection bit, the extreme value of $Ratio(\epsilon)$ happens when $\epsilon = 0.5w$, and the inter-chip stander deviation σ is needed for the $Ratio(\epsilon)$ calculation.

3.4.2.2 Inter-FHD Lower Bound Prediction

To predict inter-FHD, we calculate the probability of which any pair of chips produce different responses. The inter-FHD prediction given the width w of the inspection bit is:

$$inter\text{-}FHD = \frac{2Ratio(\epsilon)}{(1 + Ratio(\epsilon))^2} \tag{3.8}$$

With $Ratio(\epsilon) = 1$, the two areas are the same, resulting a predicted 50% inter-FHD. Given a selected bit_i, plugging in $\epsilon = 0.5w$ to Equation 3.8 would give the predicted inter-FHD lower bound.

Please note that to predict inter-FHD, the inter-chip standard deviation σ is needed because the calculation involves the CDF. However, the mean μ does not affect the prediction because the extreme value is obtained by finding the worst-case ϵ. Also, since changing the inspection bit results at least a 2x change of w, the inter-chip σ does not have to be calculated with high accuracy. It can be obtained by pre-layout simulation or measuring a small number

of chips.

3.4.3 Inspection Bit Selection

Given the Error Correction Code (ECC) specification corresponding to the PUF design, the intra-FHD threshold can be defined. From the intra-FHD prediction model, choose a set of candidate bits that would satisfy the intra-FHD threshold requirement. From the candidate bits, a best inspection bit can be determined by applying the inter-FHD prediction model given the standard deviation σ of the inter-chip delay distribution.

Please note that only one chip is needed for the inspection bit selection assuming that the measurement noise is similar for all chips and the σ is obtained from pre-layout simulation. The location of the final inspection bit, which is a public information, is passed to all PUFs for the secret response generation.

3.5 Experimental Results

3.5.1 UNBIAS PUF Implementation

The UNBIAS PUF structures are implemented on 11 Altera DE2-115 FPGA boards. In our implementation, no physical constraints, additional XORs, tunable delay units, or any systematic variation compensation techniques are used. The design is purely a RTL design.

3.5.1.1 Weak UNBIAS PUF Implementation

The weak UNBIAS PUF contains 91 ROs with 90 RO pairs for less correlated responses [31], therefore the number of CRPs is 90 and each PUF produces 90 bits. 19 inverters are used in each RO to make sure that there is no timing violation of the adder and the counter register. The response is stored in a 22-bit difference register, which is long enough to prevent the counter from overflow.

3.5.1.2 Strong UNBIAS PUF Implementation

The ROs inserted between path configurations are composed of 19 inverters, and the signal will be propagated to the next path configuration when the RO counter associated to the RO reaches a count of 50 thousand. The UNBIAS PUF has 10 path configurations, therefore the length of the challenge is 10-bit long. The length of the difference register is 19-bit, and the length of the final response for each challenge is one bit. For our experiment, 120 challenges are applied, therefore 120 bits are obtained for each PUF. Please note that the RO structure and the count of the RO counter are selected given the 50 MHz system clock of the FPGA. The results are similar as long as no overflow occurs at the 19-bit difference register.

3.5.2 Prediction Model Validation

The inter-FHD is obtained from 11 FPGAs, and the intra-FHD is calculated by measuring each PUF 10 times. The model validation is done on both weak and strong UNBIAS PUFs. To show inter-chip variation and measurement noise of our experimental setup, we measure the frequency of a single RO across the chips 10 times, and the inter-chip variation is 6.1% with 0.2% measurement noise.

3.5.2.1 Weak UNBIAS PUF Model Validation

To validate the intra-FHD prediction model, we follow the procedure described in Section 3.4.1 with $t = 10$ measurements. The results of the intra-FHD prediction of bit_{14} and bit_9 are shown in Figure 3.12. The intra-FHD of bit_9 is much higher than bit_{14} because its bin width is much smaller.

To validate the inter-FHD prediction model, for each RO pairs, we obtain an inter-chip standard deviation σ from 11 FPGAs, and the final σ used in the prediction model is the median of the σ from 90 RO pairs, which gives $\sigma = 9721$. The result of the inter-FHD lower bound prediction is shown in Figure 3.14. The prediction gap is relatively large when w is much larger than σ. However, as w becomes comparable to σ, where potential inspection

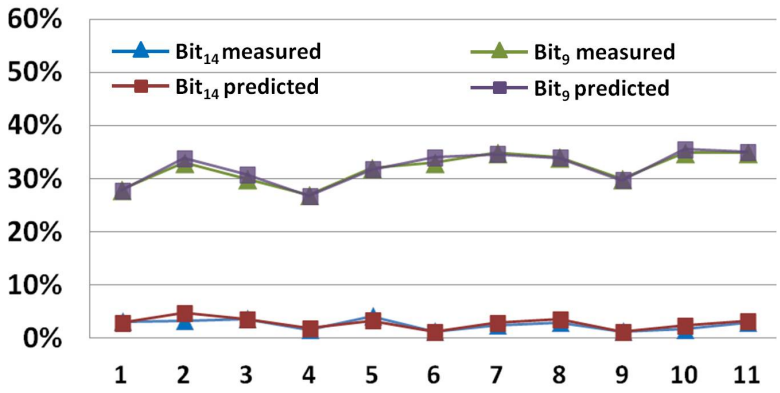


Figure 3.12: Weak UNBIAS PUF intra-FHD predictions of bit_9 and bit_{14} of 11 FPGAs. bit_9 has much larger intra-FHD because its bin width is smaller.

bits begin to occur, the prediction curve rises up quickly and matches the measured data well. To demonstrate that the inter-FHD prediction model does not require an accurate inter-chip σ estimation, Figure 3.14 also shows the prediction range with $\sigma \pm 15\%$ variation. We can see that the differences of the predictions are limited, which indicates that the σ can either be obtained from pre-layout simulation or measurements of a small number of chips. Figure 3.14 shows that and bit_{14} should be a proper inspection bit because the intra-FHD is low and the inter-FHD is close to 50%.

3.5.2.2 Strong UNBIAS PUF Model Validation

Similar to the weak UNBIAS PUF, for the intra-FHD prediction model validation, same procedure described in Section 3.4.1 is applied with $t = 10$ measurements. Figure 3.13 shows the results of the intra-FHD prediction of bit_5 and bit_{10}. The intra-FHD of bit_5 is much higher than bit_{10} because its bin width is much smaller.

Similar to the weak UNBIAS PUF, for each challenge, we obtain an inter-chip standard deviation σ from 11 FPGAs, and the final σ used in the prediction model is the median of the σ from 120 challenges, which gives $\sigma = 521$. The results shown in Figure 3.15 indicate that the inter-FHD lower bound prediction is well matched with the measured data, and the predictions with a $\pm 15\%$ variation of the σ are also presented. Figure 3.15 also shows that

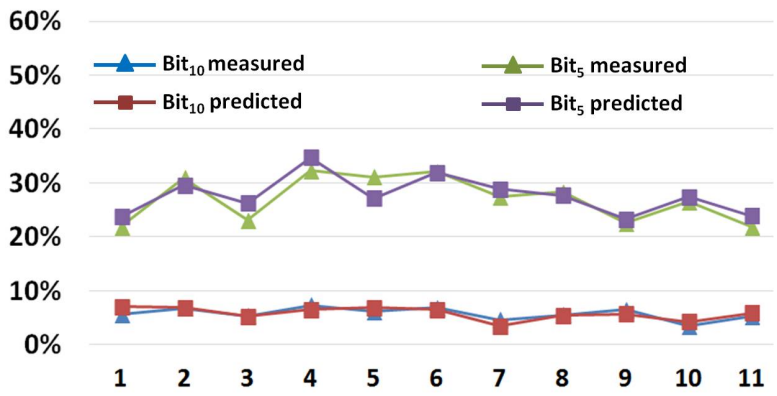


Figure 3.13: Strong UNBIAS PUF intra-FHD predictions of bit_5 and bit_{10} of 11 FPGAs. bit_5 has much larger intra-FHD because its bin width is smaller.

bit_{10} should be a proper inspection bit because the intra-FHD is low and the inter-FHD is close to 45%

3.5.3 Uniqueness and Reliability Evaluation

For the weak UNBIAS PUF, the results of inter-FHD and intra-FHD with different inspection bit selections are shown in Figure 3.14. As we can see from the figure, using bits closer to the MSB gives low intra-FHD but also low inter-FHD. This verifies the fact that the delay paths are biased if no physical implementation constraints are imposed. On the other hand, using bits closer to the LSB gives 50% on both intra-FHD and inter-FHD because of the measurement noise. As predicted, the best inspection location appears at bit_{14} with 47.1% inter-FHD and 2.4% intra-FHD. The results also indicate that the systematic variation is mitigated because no constraints are imposed on the locations of the ROs.

For the strong UNBIAS PUF, similar trends are shown in Figure 3.15. As predicted, the best inspection location appears at bit_{10} with 45.7% inter-FHD and 5.1% intra-FHD.

Table 3.1 shows comparison results with previous work. With conventional Arbiter PUF (APUF) shown in the second column, the results from [37] show that the circuit is essentially a constant number generator with very little inter-FHD. The third column shows the 3-1 double Arbiter PUF with XORs [40], where symmetric layout and high hardware cost from

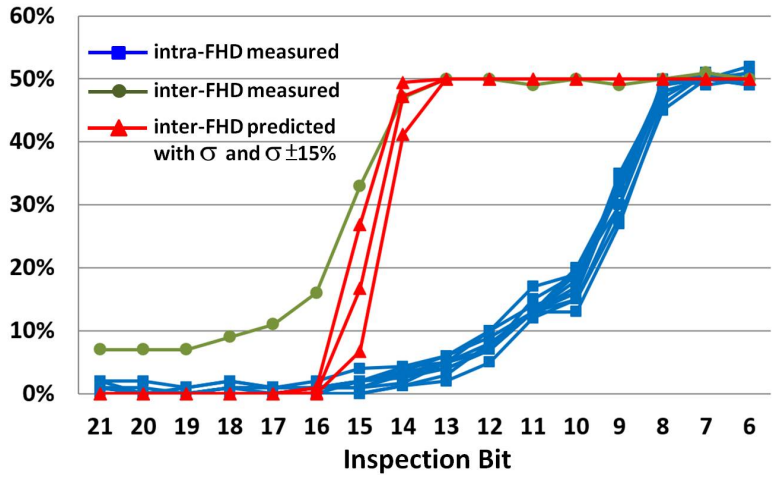


Figure 3.14: Inter-, intra-FHD, and inter-FHD prediction using $\sigma = 9721$ with different inspection bit selections of the weak UNBIAS PUF.

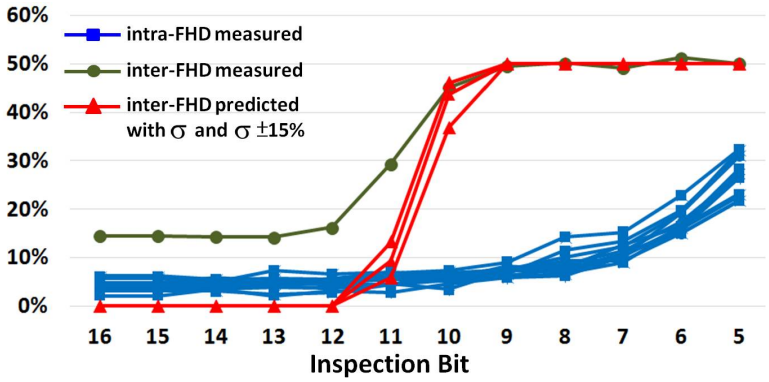


Figure 3.15: Inter-, intra-FHD, and inter-FHD prediction using $\sigma = 521$ with different inspection bit selections of the strong UNBIAS PUF.

the duplicated circuits are required. The inter-FHD is close to 50% but the intra-FHD is high due to the XORs. The fourth column shows the results from Path Delay Line (PDL) PUF [33]. Symmetric PDL and delay characterization for each CRP are required, which can cause scalability issues. The last column shows the proposed strong UNBIAS PUF. Its behavior is unique and stable, and no symmetric layout or high characterization for each CRP are required. As the number of path configurations increases or with faster system clock, the ROs inserted could be reduced depending on the design environment.

Table 3.1: Comparison between previous Arbiter PUFs and strong UNBIAS PUF

	APUF [37]	XOR [40]	PDL [33]	UNBIAS PUF
inter-FHD	7.2%	50.6%	45.25%	45.7%
intra-FHD	0.24%	11.8%	4.1%	5.1%
Symm. Layout	No	Yes	Yes	No
Characterization	No	No	Yes	No
Hardware Overhead	No	>200%	PDL	RO

3.5.4 Temperature Variation

For temperature variation, we compare the intra-FHD at 20°C and between 20°C and 75°C, which is the reliability of the PUF if it is enrolled at 20°C but verified at 75°C.

Figure 3.16 shows the intra-FHD weak UNBIAS PUF using bit_{14} as the inspection bit. The figure shows that for most PUFs, the intra-FHD at the extreme temperature is less than 15% except for two PUFs with about 18%. The instability of the weak UNBIAS PUF is relatively large and similar to the results presented in [42] for the standard RO PUF.

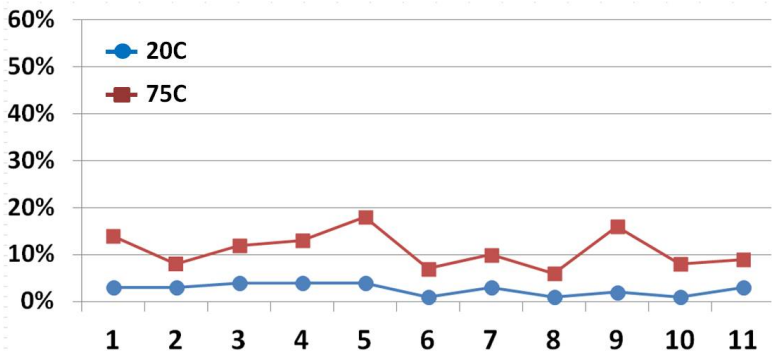


Figure 3.16: Weak UNBIAS PUF intra-FHD.

For the strong UNBIAS PUF, we use bit_{10} as the inspection bit for the measurement. The results are presented in Figure 3.17. The averaged intra-FHD at high temperature is about 8% and the worst case is still less than 10%, which is within conventional ECC margin with error reduction techniques for PUFs [46,47]. One possible explanation of smaller intra-

FHD for the strong UNBIAS PUF is that with multiple RO delay units, the overall delay variation is canceled out, where for the weak UNBIAS PUF, the variation of each RO is directly compared.

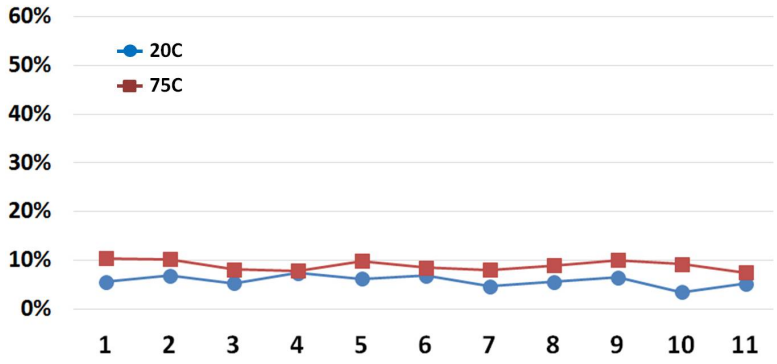


Figure 3.17: Strong UNBIAS PUF intra-FHD.

3.6 Conclusions

The proposed UNBIAS PUF effectively reduces PUF implementation efforts by mitigating the impact of biased delay paths and metastability issues. Without complex post-layout analysis or hand-crafted physical design effort, the proposed measurement can still extract local device randomness. The inspection bit can be determined efficiently from the intra-FHD and inter-FHD prediction models.

Two UNBIAS PUFs, a weak and a strong PUF, are implemented on 11 FPGAs without imposing any physical layout constraints. Experimental results show that the intra-FHD of the strong UNBIAS PUF is 5.1% and the inter-FHD is 45.7%, and the prediction models are closely fitted to the measured data for both UNBIAS PUFs. The intra-FHD of the strong UNBIAS PUF at high temperature is about 8%, which is still within the margin of conventional ECC techniques. The fact that the proposed scheme is immune to physical implementation bias would allow the UNBIAS PUF to be integrated in a high-level description of the design, such as during RTL design.

CHAPTER 4

LEDPUF: Stability-Guaranteed Physical Unclonable Functions through Locally Enhanced Defectivity

Instability has been an Achilles heel for physical unclonable functions (PUF) requiring complex error correction or other stability enhancement approaches. This instability originates from parametric nature of variations leveraged as a source of randomness, which constraints PUF from being put in widespread practical use. In this chapter, we propose several weak PUFs and strong PUFs that are completely stable with 0% intra-distance. These PUFs are called Locally Enhanced Defectivity Physical Unclonable Function (LEDPUF). A LEDPUF is a pure functional PUF which eliminates the instability of conventional parametric PUFs, therefore no helper data, fuzzy comparator, or any kinds of correction schemes are required. We propose two sources of randomness for LEDPUFs. The first is to use the Directed Self Assembly (DSA) process to form random connections that are permanently closed or opened. The weak DSA LEDPUF is constructed by forming arrays of DSA random connections, and the strong DSA LEDPUF is implemented by using the weak LEDPUF as the key of a keyed-hash message authentication code (HMAC). Our simulation and statistical results show that the entropy of the weak LEDPUF bits is close to ideal, and the inter-distances of both weak and strong LEDPUFs are about 50%, which means that these LEDPUFs are not only stable but also unique. The second source of randomness is extracted using two random gate oxide breakdown mechanisms: plasma induced damage during semiconductor manufacturing and voltage stressed damage post manufacturing. These gate oxide breakdown LEDPUFs can be easily implemented in commercial silicon processes without extra cost on PUF manufacturing and design, and they are stable and resistant to physical attacks. We fabricated bit generation units for the stable PUFs on 99 testchips with 65nm

CMOS bulk technology. Measurement results show that the plasma induced breakdown can generate completely stable responses for all 2871 bits (29 bits from each of the testchip) and significant area reduction compared with SRAM PUF can be achieved by eliminating the error correction code (ECC) hardware implementation. For the voltage stressed breakdown, the area cost is further reduced, and its 0.12% bit error rate at a worst case corner can be effectively accommodated by taking the majority vote from multiple measurements without ECC. We show that the responses of gate oxide breakdown PUFs are unique. In addition, we analyze the data of our testchips and show through various statistical distance measures that the bits of our fabricated PUFs are independent.

4.1 Introduction

A Physical Unclonable Function (PUF) is a small piece of circuitry such that its behavior, or Challenge Response Pair (CRP) [3], is uniquely defined and it is hard to be predicted and replicated because of the intrinsic random physical nature and the uncontrollability of process variations. As a security primitive, PUF can enable low overhead hardware identification, tracing, and authentication during the global manufacturing chain. The first PUF was introduced more than a decade ago [9]. Since then, many silicon PUF implementations have been proposed, such as Arbiter PUF [10], Ring Oscillator (RO) PUF [11], SRAM PUF [12], and many other variations. However, since the key commonality between all current silicon PUF implementations is their use of *parametric* manufacturing variations as the source of randomness, there exist several limitations that can cost expensive implementation overhead.

4.1.1 Limitations of Parametric PUFs

4.1.1.1 Random Local Variation Extraction

One of the major concerns of parametric PUFs is that local variation should be the *only* entropy source for these PUFs [31]. However, from our experiments on a large silicon data set [23], only 13% of total variation is random local variation, which means that most variation is

coming from global or spatial variation. Any attempt to use global or spatial variation as the source of randomness can make them vulnerable to a class of *process side channel attacks*. For instance, two PUFs on the same (X,Y) location on different wafers are highly correlated (due to large wafer-level systematics present in most modern fabrication processes). As a result, a few sacrificial wafers can aid in developing a relatively straightforward side channel attack. We tested this side channel attack on silicon RO PUF measurements in 65nm technology across 300 wafers. Figure 4.1 shows that the inter-distance [31] on the same (X,Y) is much smaller than the inter-distance across all PUFs. Therefore, an adversary with possession of a reference PUF, which is fabricated at the same (X,Y) location as the target PUF, would have a higher probability of guessing the correct answer than random guessing. The radial nature of systematic across wafer variation [23] means that just a few reference PUFs drawn carefully may be sufficient for attackers instead of keeping full sacrificial wafers.

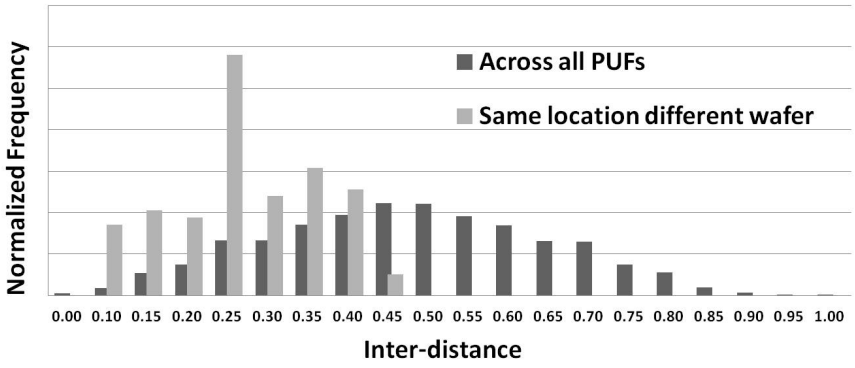


Figure 4.1: The inter-distance of PUFs from same (X,Y) location on different wafers is much smaller than that of across all PUFs, which demonstrates a possible side channel attack.

4.1.1.2 Measurement Noise

Measurement noise could be another big issue for parametric PUFs and it needs to be accounted for carefully. For instance, metastability of the arbiter circuit for Arbiter PUFs and accumulated jitter in RO PUFs can be sources of measurement noises. For weak PUF measurement, we evaluate the intra-distance [31] of SRAM PUFs using fifteen commercial 45nm SOI test chips, where each consists 176kB data memory. The power-up state is measured 10

times during an 8-hour period, and the mean of intra-distance distribution is 2.57%. Since the experiment is done in room temperature with exactly same settings, the difference is essentially contributed by the measurement noise.

4.1.1.3 Environmental fluctuations and wearout

Existing silicon PUFs are in nature susceptible to environmental fluctuations [48] and wearout [49]. To account for the instability issue, techniques such as error correction code (ECC), helper data or fuzzy comparator must be applied. A possible worst case scenario is that when the environmental factors change significantly but yet remain constant. For instance, the PUF is enrolled at 20C and is verified at 80C. In this case, a fuzzy extraction process may not be able to recover the initial PUF response, even for multiple samples of the PUF.

4.1.2 Techniques to Improve Parametric PUF Quality

A variety of techniques have been intensively studied over the years to extract random local variations or to make a PUF more stable and reliable. A Non-Volatile Memory (NVM) based PUF without helper data is presented in [50]. However, the PUF comes with hardware and calibration overhead, and the results of uniqueness and entropy analysis are missing. In [15], the local randomness is distilled by modeling and subtracting the systematic variation. A similar technique is to subtract the averaged frequency from multiple measurements to reveal the true local random variation [51]. However, the calculation and information storage requirement comes with the cost of addition latency and hardware. Taking the majority vote [52] or finding stable responses [34] are possible techniques to eliminate the measurement noise, however, at the cost of large latency or reduced number of challenges. Other complex implementations have been proposed to mitigate stability issues, and they often induce lower hardware efficiency [31], additional circuit complexity [28], or making the PUF more susceptible to attacks [53]. Also, to protect PUFs from the worst case scenario as described creates a huge overhead as it requires to employ very strong ECC [54].

4.1.3 The LEDPUF

The issues of parametric PUFs, such as the described instability, wearout, measurement noise, limited local variation, and limited side channel attack resiliency, clearly motivate the need to design PUFs that do not rely on parametric performance variations as the entropy source. In this chapter we propose a weak and a strong LEDPUF.

Rather than comparing parameter deviations, the response of an LEDPUF is stability-guaranteed because it depends on *permanent randomness* generated in (1) Directed Self Assembly (DSA) process, which is highly compatible with existing CMOS technology and is expected to be used in manufacturing in the near future [55], or (2) plasma induced oxide breakdown and the voltage stressed oxide breakdown.

Compared to similar parametric PUFs such as hardware obfuscation [56] or digital PUFs [57], LEDPUF is completely stable and less susceptible to side channel attacks or model building attacks. The proposed LEDPUFs are also a functional PUFs where the logic function itself is the signature and the strong LEDPUF can generate a variety of challenge-response pairs as needed. The Boolean nature of the response without any parametric dependence means that LEDPUF is not only immune to measurement noise and wearout, but also offers a greater level of reliability compared to existing PUFs as the output is resistant to changes in the environmental factors.

The contributions of this chapter are:

- The first stability-guaranteed silicon PUFs through locally enhanced defectivity are proposed.
- Detailed constructions of the weak LEDPUF using random DSA connections are presented. It is the first weak PUF with 0% intra-distance without using any stability enhancement techniques.
- The weak DSA LEDPUF with 0% intra-distance enables the construction of the strong LEDPUF based on cryptographic hash functions.

- The simulation statistics and entropy calculation are presented. The results show that the proposed LEDPUFs can generate unique responses and their behaviors are hard to predict.

- The plasma induced oxide breakdown and the voltage stressed oxide breakdown are proposed to construct stable LEDPUFs. The oxide breakdown PUFs are resistant to invasive attacks such as imaging attacks.

- Test structures violating antenna rules are fabricated with 65nm CMOS bulk technology. Measured results from 99 testchips show that the responses are highly stable across combinations of voltage (0.8V, 1.0V, 1.2V) and temperature variations (25°C, 100°C). Compared to a practical SRAM PUF, significant area reduction can be achieved by eliminating ECC implementation for the highly stable responses.

- We analyze the data from these testchips and show based on various statistical distance measures that pairs of bits with the same antenna ratio as well as bits that are located next to each other are effectively statistically independent.

4.2 LEDPUF construction through DSA

4.2.1 DSA Randomness Extraction

[1] Self-assembly is a mechanism that describes block copolymers (BCP) composed of immiscible blocks phase-separate into certain structures [58]. The guiding templates, which are used to *guide* the self-assembly process [59], can be lithographically-printed trenches (Graphoepitaxy) or chemically-treated surfaces (Chemoepitaxy). During the graphoepitaxy process for contact or via holes, the guiding templates are first lithographically printed, then the surface is spin-coated with the BCP solution. The phase separation occurs during the thermal annealing, and with a particular BCP and surface treatment of substrate [60], cylinders are formed of one block in a matrix of the other block [61].

[1] I would like to express my appreciation to Dr. Andres Torres at Mentor Graphics for his support on the DSA simulator.

In case of a diblock copolymer made of two blocks, say A and B: at equilibrium, the microphase separation is established by an energy balance between the stretching energy for the polymer chains and the energy of interactions at the interface between A and B microdomains [62]. Thermal equilibrium is achieved when the free energy is minimized, and the minimum energy state strongly depends on the level of confinement achieved by the layout of guiding templates. In other words, the size, shape, and critical dimension (CD) of the guiding template can greatly affect the DSA defect density [63–65].

For bigger-sized templates, it becomes energetically less expensive to induce a defect than to achieve a defect-free energy minimization [64, 66, 67]. Also, with less confinement forces from the guiding template due to its large size, random interactions from thermal fluctuation [68] or initial kinetics of collective density and state fluctuations [69] begin to dominate the assembly process. Therefore, final assembly results can be random by designing guiding templates that are large enough to cause random assembly errors even if there are no lithographic variations. Figure 4.2 shows three simulation results of the same large guiding template with an existing DSA simulator [70], where the model of the PS-b-PMMA copolymer has been validated in [71]. The three layers inside the polygonal guiding template are the top, middle, and bottom layers of a via. If a cylindrical via hole is formed correctly, the three layers should be three overlapped concentric circles. However, for the large guiding template, random arrangement with different orientations begin to occur. In other words, the randomness of DSA is confined within predetermined local areas only by deliberately designing "bad" guiding templates.

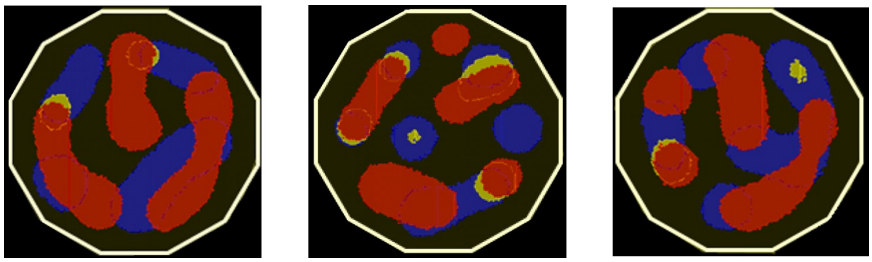


Figure 4.2: Random via formations with a same large guiding template.

4.2.1.1 Hard Defective Connection Formation

We leverage the randomness extracted from DSA to form randomly assembled connections, and these connections are then used to fabricate LEDPUF. Though in conventional DSA, the goal of the guiding template design is to achieve high confinement and avoid regions of random phase transitions, we use the same theory but to enhance randomness in assembly. To construct a DSA random hard defective connection, we configure the size of the guiding template so that two vias are formed with a certain probability that they are connected permanently. A DSA hard defective connection is composed of the two vias along with the connection.

In our experiment, each simulation contains three guiding templates with a same shape, and two vias are formed in each of the guiding templates. If the via pair in a same guiding template is merged, the DSA hard defective connection is in closed state; otherwise, the connection is in opened state. The states of the three connections in a simulation are independent of each other as expected in real DSA process [71]. In our statistical analysis, an open state is represented by a logic "1", and a closed state is represented by a logic "0". 500 simulation were performed in our experiments, so 1500 bits of raw data is obtained from the simulation. Based on our analysis, the empirical entropy of triples of bits is only 0.0063 bits smaller than the entropy of independent triple of bits. Examples of a simulation result in 2D and 3D views are shown in Figure 4.3 (a) and (b), respectively. In the 2D view, the rectangular shapes with rounding corners are the guiding templates, and the shapes inside of the guiding templates are the vias. If the via pair in a same guiding template is merged, the DSA hard defective connection is formed as shown in Figure 4.3 (c), and it is in permanent closed state; otherwise, the DSA hard defective connection is in permanent opened state as shown in Figure 4.3 (d). In Figure 4.3 (a) and (b), two hard defective connections are in opened state, and one connection is in closed state.

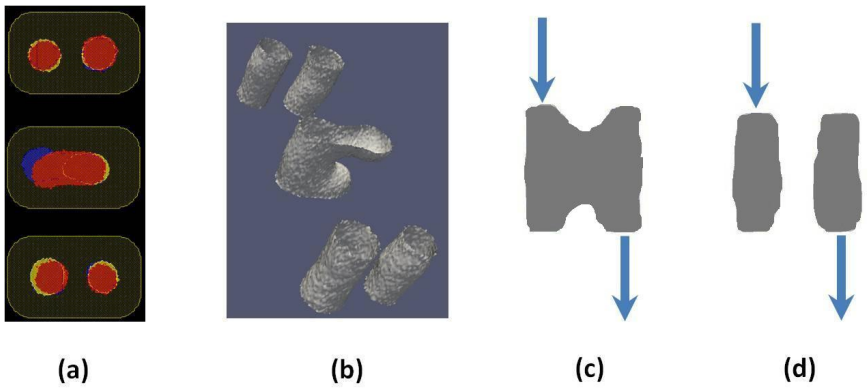


Figure 4.3: (a) 2D view of 3 DSA hard defective connections on three pairs of vias. (b) 3D view of 3 DSA hard defective connections on three pairs of vias. The connections on the top and bottom are in permanent opened state; the middle one is in permanent closed state. (c) Vias are partialy merged, so the DSA hard defective connection is in closed state. (d) Vias are not merged, so the DSA hard defective connection is in opened state.

4.2.2 DSA LEDPUF Construction

4.2.2.1 Weak LEDPUF Construction

The proposed weak LEDPUF is composed of arrays of SSUs. Each SSU is constructed from a DSA defective connection, which can be considered as random switches with permanent states that determine the unique and stable function of the circuit. Figure 4.4 (a) shows the implementation of a SSU. Two ends of the DSA connection are connected to VDD and GND through opposite switches. Figure 4.4 (b) shows the abstraction of a SSU. In standby mode or before the evaluation, the evaluation signal EVA is low and the output is zero. During evaluation mode, EVA becomes high, and the output is either one or zero depending on the permanent state of the DSA connection. If the DSA connection is closed, the output is one; otherwise, the output is zero.

The proposed weak LEDPUF is constructed by arranging the SSUs in forms of arrays. Figure 4.5 illustrates a weak LEDPUF with n rows and m columns, where the number of SSUs is nm, and the number of CRPs is n. Since only one of the rows is being evaluated at a time, a one-hot decoder is used so that only one bit of the EVA vector is logic 1. The

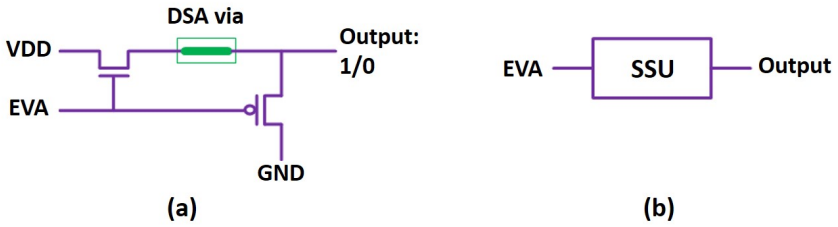


Figure 4.4: (a) Stable signal unit implementation. When EVA is high, the output is either one or zero permanently depending on the state of the DSA via. (b) Abstraction of a SSU.

challenge fed into the decoder is a $log(n)$-bit input, and the response is a m-bit output.

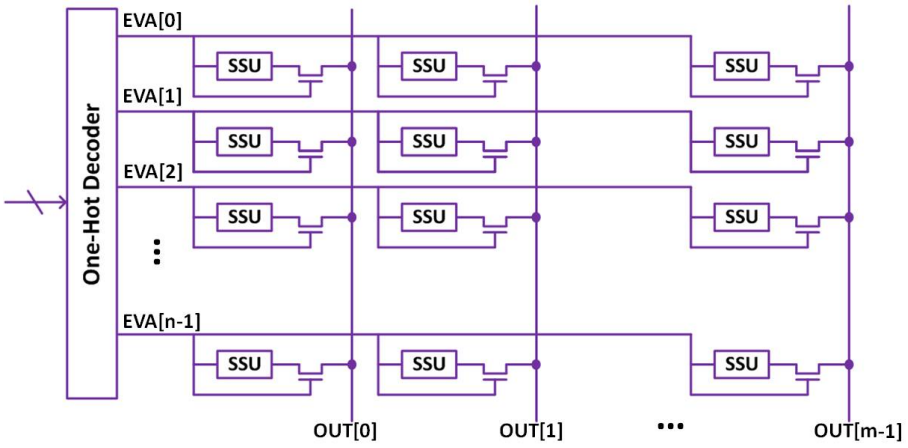


Figure 4.5: A weak LEDPUF with n challenges and m-bit response. Only one bit of the EVA vector is logic 1 at a time.

Compared with existing weak SRAM PUFs, the weak LEDPUF has several evident advantages:

- It is completely stable, so it has no area or latency overhead. To generate a bit response, the weak LEDPUF requires only one SSU and a transistor, or 3 transistors equivalently, as for a standard SRAM cell, 6 transistors are required. Once the state of the DSA via is determined, the output is fixed permanently, so no additional ECC, fuzzy extraction, or helper data is needed. As stated in [46], for a SRAM PUF to generate a 128-bit response, more than 4k SRAM cells are needed under a condition with 15% bit error probability. Therefore, the total number of transistors needed for

SRAM PUF to generate a 128-bit response would be 24k, where for the weak LEDPUF, only 384 transistors are needed, which is more than 600x less than a SRAM PUF, thus the area is also much smaller even assuming that the hardware cost of the peripheral circuits are similar.

- In addition to model building attacks [29], the weak LEDPUF is also more resistant to existing attacks to SRAM PUFs, such as laser stimulation [72] or Photonic Emission Analysis (PEA) [73]. The laser stimulation attack focuses on retrieving the on/off state of transistors, but for weak LEDPUF, the states of the transistors, which depend on the EVA signal, do not reveal secret information. The PEA attack does not work effectively because for each SSU, the source voltage (VDD) of the NMOS is always higher than the drain voltage, and the PMOS at the output will not stay in saturation region since the output will be pulled down even if the DSA connection is formed.

When using a weak PUF for a CRP authentication scheme, it is meaningful to consider the chance of guessing the response. The min-entropy [74]

$$H_{\min}(X) = -\log_2\left(\max_i p_i\right) \tag{4.1}$$

is a means of quantifying the chance of guessing the response in a single round. It corresponds to the exponent of the probability of the most probable response, assuming that each element is identically and independently distributed. Based on our own experimental results for the formation of connections we evaluate the probability mass function of a bit generated by a LEDPUF

$$p_X(1) = 0.4626 \quad p_X(0) = 0.5374. \tag{4.2}$$

The min-entropy of the empirical probability mass function is

$$H_{\min}(X) = 0.8962 \tag{4.3}$$

whereas the maximal min-entropy, which is achieved by a fair coin toss, equals to 1. Hence, when the response consists of m bits, the probability of guessing the response of the LEDPUF is equal to $2^{-0.8962m}$. Essentially, it means that when the response of a LEDPUF is $1.11m$

bits long, the probability of guessing the response is equal to 2^{-m} which is the probability of guessing the result of m independent fair coin tosses.

Another possible attack is a dictionary attack in which the attacker guesses the most probable responses in an ascending order. The number of attempts it takes to find the response is coined *guesswork* [75] which we denote by G. For a stream of m bits which are drawn i.i.d the expected guesswork is lower bounded by [76]

$$E\{G\} \geq \frac{1}{4}2^{mH_{Sh}(x)} \tag{4.4}$$

where $H_{Sh}(x) = \sum_i -P_i \log_2(p_i)$ is the Shannon entropy. Assigning (5.23) to the Shannon entropy we get that

$$E\{G\} \geq \frac{1}{4}2^{0.996m}. \tag{4.5}$$

Further, the exponential growth rate of the expected guesswork (as a function of m) scales according to the Renyi entropy [75] with parameter $\alpha = \frac{1}{2}$

$$\lim_{m\to\infty} \frac{1}{m} \log_2 E\{G\} = H_{1/2}(X) = 2 \cdot \log_2 \left(\sum_i p_i^{1/2} \right). \tag{4.6}$$

Assigning the empirical probability (5.23) to (4.6), gives a growth rate that equals to 0.998, whereas the maximal growth rate which is achieved by a fair coin toss, is again equal to 1. Therefore, when the response is $1.002m$ bits long, the guesswork scaling behavior is equal to the scaling behavior of m independent fair coin tosses.

Interestingly, the loss induced by the fact that the bits are not drawn uniformly, is much higher when considering only a single guess. On the other hand, when considering multiple guesses, the loss decreases significantly. It is also worth mentioning that even though the loss for a single guess is not negligible, for large m the chance of guessing the response is still very low.

Based on the empirical results (5.23) we deduce that the bits are drawn according to a biased probability function. However, the probability mass function can always be adjusted by changing slightly the size of the guiding template. Another possibility to balance out the probability, is by using a randomness extractor [77], which outputs a shorter stream of bits that correspond to independent fair coin tosses.

4.2.2.2 Strong LEDPUF Construction

One of the shortcomings of using memory-based PUFs for CRPs, is the scaling of the hardware size as a function of the number of CRPs [78]. In general, each channel response pair requires a different set of circuits, and as a result the hardware size is proportional to the number of CRPs. On the other hand for strong PUFs the hardware size scales logarithmically as a function of the number of CRPs.

In order to create a strong LEDPUF we consider a keyed hash function along with a weak LEDPUF. The weak LEDPUF response is used as a key for the keyed hash function. The challenges serve as the input to the hash function, whereas the response is the output of the keyed hash function. Figure 4.6 presents a strong LEFPUF based on a keyed hash function and on a weak LEDPUF.

It is important that the keyed hash function uses the key in such a way that does not enable the attacker to predict responses to unobserved challenges based on the observed ones. Therefore, concatenating the key directly to the challenge, which is vulnerable to extension or collision attacks, is not a good realization of the strong LEDPUF.

We create strong LEDPUF by using a weak LEDPUF as a key for a keyed-hash message authentication code (HMAC) [79]. Any cryptographic hash function, such as SHA-1 or SHA-2 can be used for HMAC. It is worth mentioning that in [80] the authors also propose the use of PUFs with an HMAC in a somewhat similar manner; however, they do not take into consideration the overhead incurred by the instability of parametric PUFs.

To give a rough estimation of the hardware implementation cost of the strong LEDPUF, for a HMAC-SHA1, the implementation requires about 30k gates [81], and just the ECC part, BCH for example, of a parametric PUF would require same order of gates [54].

The level of security of a strong LEDPUF depends on the underlying hash function and the quality of the weak LEDPUF that serves as a key, whereas weak LEDPUFs rely solely on the randomness in the manufacturing process.

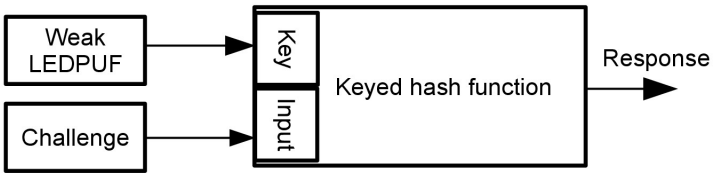


Figure 4.6: A strong LEDPUF based on a keyed hash function (HMAC or NMAC) and a weak LEDPUF.

Table 4.1: Fractional Inter-distance of the LEDPUF

	Response Bits	Mean	Standard Deviation
Weak LEDPUF	512	0.503	0.02
Strong LEDPUF	256	0.500	0.03

4.2.3 Experimental Results

4.2.3.1 Uniqueness Evaluation

For the weak LEDPUF, the uniqueness is evaluated by calculating the fractional inter-distance [31] of 1000 weak LEDPUFs, each producing 512 bits of response. The distribution is with mean=0.503 and standard deviation=0.02 as shown in the second row of Table 4.1. Since the variance value is proportional to the inverse of the length of the response, as the length of the response increases the variance value goes to zero while the mean value goes to 0.505.

For the strong LEDPUF, the structure used in our experiment is based on the NMAC structure, and the results are obtained from simulations. Each strong LEDPUF consists of a weak LEDPUF that provides 2x256 bits for the two initial vectors (IV) of the nested hash, and each response is a 256-bit stream because SHA-256 is used in the construction. The same challenge is given to 1000 strong LEDPUFs, and the inter-distance of the responses is a distribution with mean=0.500 and standard deviation=0.03 as shown in the third row of Table 4.1.

4.2.3.2 Stability-Guaranteed Weak LEDPUF Requirement

To construct a strong LEDPUF, only the weak LEDPUF can be used because of its 0% intra-distance. If other existing weak PUFs with even small intra-distance are used, the intra-distance of the strong LEDPUF would be increased dramatically due to the avalanche effect. In other words, even a single bit flip of the weak PUF can completely change the response of the strong LEDPUF. Figure 4.7 (a) shows that the intra-distance of the strong LEDPUF jumps from 0% to 50% as the number of bit flips increases from zero to one.

Figure 4.7 (b) shows how the intra-distance of the strong LEDPUF rises as the intra-distance of the weak PUF increases in logarithmic scale. Since 2x256 bits of the IVs are from the weak PUF, for a weak PUF with 0.1% intra-distance, the probability that it generates a same 512-bit response twice is about 60%, which translates to a roughly 20% intra-distance of the strong LEDPUF. Therefore, only the weak LEDPUF with a guaranteed 0% intra-distance can be used for the IV generation.

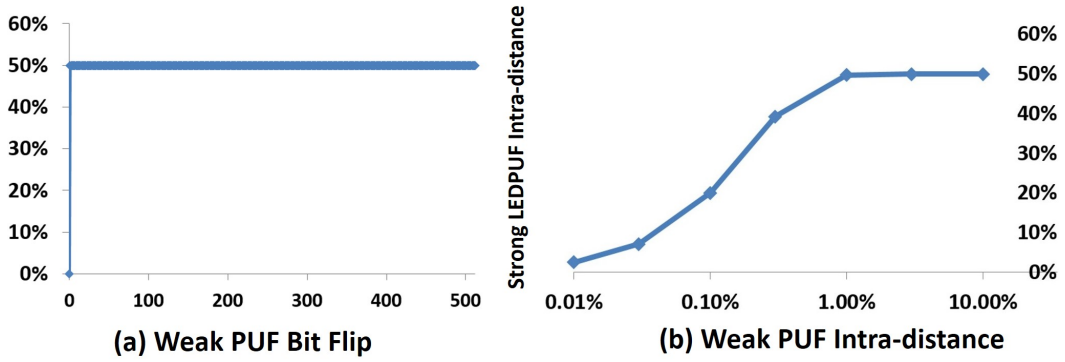


Figure 4.7: (a) A single bit flip from the weak PUF can induce a completely different response of the strong LEDPUF due to the avalanche effect of the hash function. (b) Intra-distance of the strong LEDPUF rises dramatically if other weak PUFs with small intra-distances are used in the strong LEDPUF construction.

4.3 LEDPUF construction through gate oxide breakdown

In this section, we first introduce the gate oxide breakdown and describe two approaches exploiting the gate oxide breakdown as randomness sources of stable PUFs, followed by PUF bit generation and attack resilience analysis.

4.3.1 Gate Oxide Breakdown

The gate oxide breakdown is detrimental to metal-oxide-semiconductor (MOS) devices because it can cause significant drifts of transistor parameters. The breakdown can be categorized into two types: soft breakdown and hard breakdown, where both mechanisms introduce significant sudden increase of the leakage current [82]. For soft breakdown, the conducting path from gate to the substrate is formed by the charged traps in the gate oxide. Once there is conduction, new traps begin to accumulate due to thermal damage, which in turn increases the conductance. The positive feedback eventually leads to thermal runway and oxide is physically melted in the breakdown spot. This type of breakdown is called hard breakdown. The gate leakage current of an oxide with breakdown can be 100X larger than the leakage current of an oxide without breakdown [83].

4.3.1.1 Plasma Induced Gate Oxide Breakdown

During silicon wafer fabrication, radio frequency (RF) plasma processes are widely used for etching, photoresist stripping, or ion implantation [84]. In the plasma ambient, metal segments, VIAs, or polysilicon electrodes, which are the antenna segments, can be electrically charged by ions or electrons when the currents produced from the ion and electron do not cancel out with each other through each RF cycle [85], and therefore produce the antenna voltage. For the antenna segments connected to the gate inputs, the resulting electrical stress from the antennas can potentially damage the underlying gate oxide and create a conducting path from the gate to the substrate. The phenomenon is called plasma induced gate oxide breakdown, or the antenna effect.

Though the maximum voltage rise over half of the RF period can be modeled [84], the actual voltage still cannot be predicted because the exact motion and amounts of ions and electrons collected by the antenna segment are random and unpredictable. The higher the gate voltage is, the higher the probability for the gate oxide breakdown to occur, thus causing a device to fail. Also, systematic plasma variation across wafer does not have much impact on the local randomness because the variation is negligible to a die [84].

To avoid the antenna effect, design rules of the antenna ratio (AR) [86] as shown in equation (4.7) must be strictly followed during fabrication [87]. Practical design rules of AR range from 100 to 5000 depending on the process details [86].

$$AR = \frac{\text{exposed antenna area}}{\text{gate oxide area}} \tag{4.7}$$

Since both soft breakdown and hard breakdown can induce about 100X or more leakage current than a good oxide, they are both considered as breakdown in our proposed stable PUF construction. In [88], a device is considered as a failure if the gate leakage current is larger than 1nA, and based on the criterion the author proposed a failure probability prediction formula. However, the process parameters of our testchip fabrication are unknown prior manufacturing therefore we implemented a variety of antenna ratios to measure breakdown probabilities, which are presented in Section 4.3.2.2.

Many techniques have been proposed to solve antenna effect, such as jumper insertion [89] or antenna-aware routing [90]. However, while foundries try to avoid antenna effect during manufacturing, we exploit the uncontrollable physical phenomena as another randomness source of a stable PUF.

4.3.1.2 Voltage Stressed Gate Oxide Breakdown

The purpose of antenna rules is to protect all transistors from having deviated parameters, for example 20% gate leakage increase at 1.4xVDD [91], which could be harmful for a normal fabrication but still far from causing a real breakdown. Therefore, to introduce a noticeable plasma induced breakdown (100X increase of leakage current) with 50% probability of a

transistor, an AR larger than 1000X antenna rule may be required, which can result in large area overhead.

To avoid using large antenna segments, we propose to induce gate oxide breakdown post fabrication by applying high voltage stress to the gate of a transistor that essentially mimics the charge accumulation during the plasma process. By voltage stressing the gate terminal of a transistor, oxide breakdown can be introduced with small AR or even without violating the antenna rules. The advantage of voltage stressed induced breakdown is that large antenna segments are not required, while the uncontrollable process variation of gate oxide thickness is magnified to achieve a breakdown probability close to 50%, which is desirable as a source of randomness for PUFs. On the other hand, such a PUF construction requires an additional one-time stress step post manufacturing (or during PUF enrollment). Please note that our proposed voltage stressed gate oxide breakdown mechanism is different from the Erasable PUF proposed in [92], where oxide breakdown is introduced to erase targeted bit cells instead of being used as a stable source of randomness.

4.3.1.3 Stable Signal Unit Construction

The permanent gate oxide breakdown mechanism, which can be caused by plasma damage or voltage stressed damage, is used to construct a Stable Signal Unit (SSU) as a source of permanent defectivity. A SSU is a p-MOS transistor designed to violate antenna rules, and its drain, source, and bulk terminals are connected to capture the effect of the gate oxide breakdown at all possible locations. Similar to a gate oxide breakdown model given in [93], the SSU is attached in series to a precision resistor as given in Fig. 4.8, where Fig. 4.8 (a) shows a SSU without oxide breakdown and Fig. 4.8 (b) shows a SSU with oxide breakdown. If no breakdown occurs as depicted in Fig. 4.8 (a), the device is essentially a capacitor or a resistor much larger than the precision resistor, thus the output voltage would be lower than 50% VDD when the evaluation signal EVA is VDD; if a breakdown happens, as shown in Fig. 4.8 (b), the device can be seen as resistors much smaller than the precision resistor, thus the output voltage would be higher than 50% VDD when EVA is VDD. The resistance

of the precision resistor (10MΩ) is determined by actual measurements from 99 testchips as described in Section 4.3.2.2. Different from the bit generation units in [94], our SSU does not suffer from potential response time latency due to the limited leakage current when no breakdown occurs.

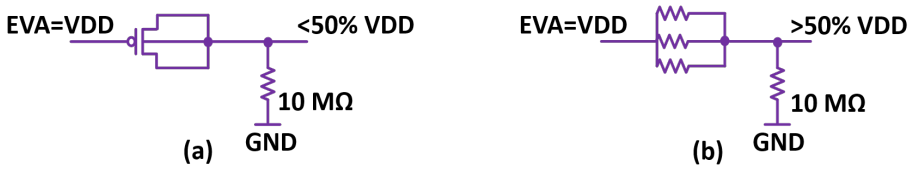


Figure 4.8: Schematic of antenna SSU attached to a precision resistor.

4.3.1.4 Attack Resilience

It is worth mentioning that the SSU is more secure than an antifuse cell because an antifuse cell is programmed with hard breakdown only, while the output of the SSU is decided by both soft breakdown and hard breakdown, and a soft breakdown is much harder to detect than a hard breakdown (albeit possible for a very resourceful attacker). For probing attack, the efficiency is limited by the mechanical constraints. For imaging attacks, such as Scanning Electron Microscopy (SEM), Transmission Electron Microscopy (TEM), or Electron Beam Induced Currents (EBIC), it is difficult to efficiently identify a soft breakdown for several reasons:

1. It is difficult to detect a soft breakdown because its physical appearance is very similar to a fresh gate oxide without any visible holes. Furthermore, SEM has limited ability to observe traps inside the oxide, therefore it is difficult to see if a conducting path formed by traps, or a soft breakdown, exists. It is also challenging for EBIC to identify a soft breakdown because the limited current of a soft breakdown can induce measurement noises [95], and the throughput of the electron beam is low.

2. It is difficult to observe a soft breakdown from a top-down or cross-section TEM because the image does not effectively tell the depth of the traps [96]. In addition, to obtain a

cross-section TEM, the chip has to be vertically cut into thin films, which will destroy the neighboring SSUs. Therefore, even if a hard breakdown information might be retrieved from a cross-section view, the attacker cannot obtain the secrets of all SSUs of a same PUF because of the destructive observation.

4.3.2 Testchip Fabrication and Measurement Results

4.3.2.1 SSU Implementations

The proposed SSUs are implemented and fabricated on 99 testchips with commercial 65nm GP 1P9M_6X1Z1U CMOS bulk technology with 1V nominal voltage. The smallest gate size ($0.0072\mu m^2$) of the technology is used for all the SSUs. In our testchips the fabricated SSUs intentionally violate antenna rules by a few hundred times to a few thousand times on different layers.

On each chip, 29 SSUs are implemented with 17 different ARs, therefore the total number of SSU implementations is 2871 from 99 chips. For each of the SSUs, the cell area and detailed antenna violation report are given in Table 4.2, where a zero indicates that there is no antenna rule violation on such layer. The antenna rule violation reports are provided to the foundry to skip such design rule checks without extra cost for the foundry. The M_T, V_T, and P_T structures test the effects of metal, VIA, and polysilicon layers from small AR to large AR, respectively. For each of the M_T, V_T, and P_T, two SSUs with same AR are implemented, therefore 24 bits of responses are obtained from these SSUs on a chip. The remaining five test structures are of various combinations of the violating layers, and one SSU is implemented for each of the five test structures. In summary, on each chip, 29 bits are measured, and 24 bits of them are obtained from the duplicated 12 structures of M_T, V_T, and P_T.

Table 4.2: Cell area, accumulated areas of VIA, metal, polysilicon, and polysilicon perimeter of SSUs fabricated. The numbers are in μm^2. A zero indicates no antenna rule violation on such layer.

	Cell	VIA	Metal	Poly	Poly Perim. (μm)
M_T1	36	0.87	1144.57	0.00	0.00
M_T2	360	1.17	1468.57	0.00	0.00
M_T3	1200	0.00	4398.88	0.00	0.00
M_T4	4800	0.16	36781.89	0.00	0.00
V_T1	2.4	0.87	1108.57	0.00	0.00
V_T2	8	2.31	1108.57	0.00	0.00
V_T3	90	15.27	1185.66	0.00	0.00
V_T4	804	144.91	1895.05	0.00	0.00
P_T1	4.8	1.26	1917.53	0.00	0.00
P_T2	27	1.26	1917.53	18.17	55.59
P_T3	203	1.26	1917.53	180.07	128.43
P_T4	1800	1.26	1917.53	1800.07	222.46
Test1	804	1071.86	5631.11	0.00	0.00
Test2	4.7	1.86	0.00	0.00	0.00
Test3	80	0.26	299.20	0.00	0.00
Test4	60	20.84	318.78	28.07	83.81
Test5	118	54.40	617.25	56.39	164.72

4.3.2.2 Breakdown Probability Evaluation

To determine the gate oxide breakdown of a SSU, we use Agilent 34411A Digital Multimeter to measure the equivalent resistance R_{eq} of each SSU, and from the distribution of R_{eq} we choose a proper precision resistor as shown in Fig. 4.8 to determine whether or not an oxide breakdown has occurred. Fig. 4.9 shows R_{eq} distribution of a SSU implementation (V_T1) with plasma induced and voltage stressed breakdown on 99 chips in an increasing order at 25°C, 1V. For both distributions, the R_{eq} of a SSU implementation without oxide breakdown is at least 100X larger than a SSU with oxide breakdown. After voltage stress, the R_{eq} are in general smaller and much more oxide breakdowns are introduced. The results are similar for all SSUs. The large gap in the figure can be effectively exploited to generate stable digital signals from SSUs. Therefore, we choose, according to the R_{eq} measurements, a 10MΩ precision resistor to measure the gate oxide breakdown of each SSU.

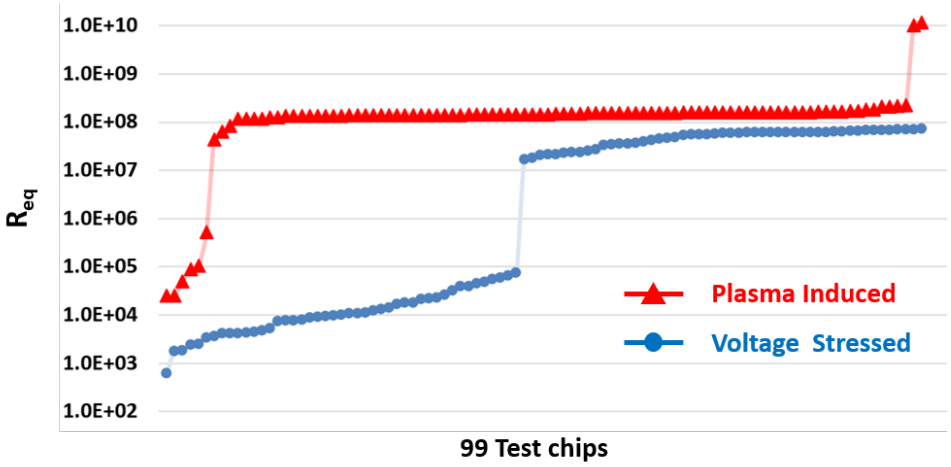


Figure 4.9: The R_{eq} distribution of a SSU implementation (V_T1) with plasma induced and voltage stressed oxide damage on 99 chips at 25°C, 1V.

4.3.2.3 Plasma Induced Breakdown

For the plasma induced breakdown, the results of breakdown probabilities of SSU implementations on 99 chips are shown in Table 4.3. From the table we see that the breakdown probability of each SSU after plasma induced oxide damage is well below 50%. The Test1 SSU

implementation has the highest breakdown probability of 16%, which means the responses of SSUs are highly biased. This is undesirable for its low randomness in each response bit. Using larger AR to further increase the breakdown probability may not be a proper approach due to large area overhead. Also, as seen from Table 4.2 and Table 4.3, the breakdown probability does not increase dramatically as the AR increases. Our results show that even when the AR is more than 1000X larger than the antenna rule, the breakdown probability is still much lower than 50%.

4.3.2.4 Voltage Stressed Breakdown

For the voltage stressed breakdown, we stress 24 SSUs (M_T, V_T, and P_T groups) on each testchip by applying 5.5V to the EVA for 10 seconds. The results of the stress are shown in Table 4.3. From the table we can see that breakdown probabilities, which are only slightly correlated with the ARs, are elevated to at least 50% even for the SSUs with the smallest ARs. Different stress voltages have been tried in our experiments, but only when the voltage is 5.5V will the breakdown probability be elevated to 50%. These results show that more unbiased responses compared to plasma induced breakdown can be achieved by using small SSUs such as V_T1. Therefore, a SSU can be implemented with much smaller area, possibly even without violating the antenna rule, than the plasma induced breakdown approach.

4.3.2.5 Stability Evaluation

To evaluate the stability of the SSUs, we measure all SSU responses from 99 chips at 6 corners: temperatures at 25°C and 100°C with ±20% voltage variation at 0.8V, 1V, and 1.2V.

4.3.2.6 Plasma Induced Breakdown

For the plasma induced breakdown, all SSUs from 99 chips (total 2871 bits generated) are completely stable at all corners during multiple measurements. This can be explained by the fact that the change of R_{eq} at different corners are limited. Fig. 4.10 shows the change of

Table 4.3: Breakdown probability of 17 AR implementations on 99 testchips.

	Plasma Induced	Voltage Stressed
M_T1	0.5%	57.6%
M_T2	0.5%	51.5%
M_T3	2.5%	57.1%
M_T4	2.0%	51.0%
V_T1	0.5%	50.0%
V_T2	6.1%	54.0%
V_T3	0.0%	64.7%
V_T4	0.0%	58.6%
P_T1	1.0%	50.5%
P_T2	2.5%	51.5%
P_T3	1.0%	58.6%
P_T4	1.0%	60.0%
Test1	16.2%	N/A
Test2	2.0%	N/A
Test3	5.1%	N/A
Test4	1.0%	N/A
Test5	3.0%	N/A

R_{eq} of a SSU (Test1) under voltage and temperature variations. In Fig. 4.10 (a), the R_{eq} of the SSU with breakdown is only a few KΩ and the changes under extreme temperature and voltage variations are limited. On the other hand, Fig. 4.10 (b) shows a SSU without oxide breakdown, where the R_{eq} remains at less than 45MΩ, which is still orders of magnitude larger than the SSU with oxide breakdown.

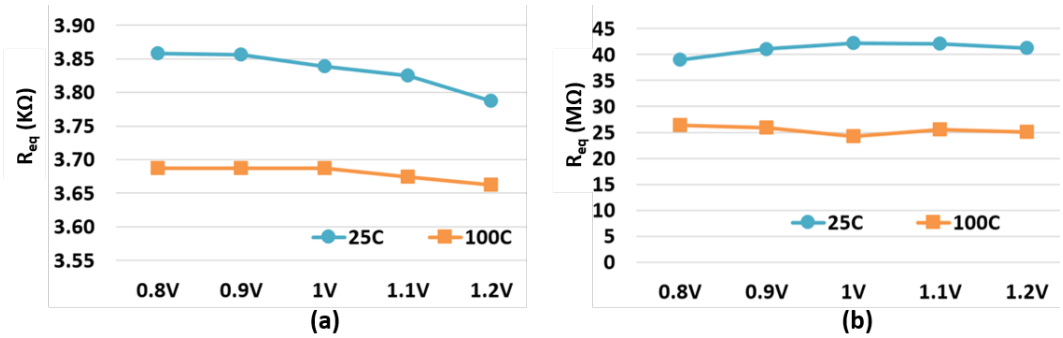


Figure 4.10: Equivalent resistance under extreme voltage and temperature variations. (a) SSU with oxide breakdown. (b) SSU without oxide breakdown.

4.3.2.7 Voltage Stressed Breakdown

Unlike the plasma induced breakdown, for the voltage stressed breakdown, an extremely small portion of the SSUs are not completely stable. To quantize the results of stability evaluation for the voltage stressed breakdown, each SSU is measured 10 times at each corner and we define the responses measured at 25°C with 1V, where all responses are consistent, as the reference responses. A SSU is unstable at a corner if at least one of its values from the 10 measurements is different from the reference response. We define bit error rate (BER) the number of unstable bits divided by 2376, which is the total number of SSUs stressed (24 SSUs on each of the 99 chips). Table 4.4 shows the BER at each corner. We found that at several corners, 1 to 3 SSUs out of 2376 SSUs implemented are unstable for the voltage stressed breakdown. Since most responses of unstable SSUs are still consistent with the reference responses, instead of performing a "afterburn" phase to all broken oxides, where additional hardware and calibration are required [97], we take majority vote of multiple measurements

Table 4.4: Bit Error Rates of 2376 SSUs of the voltage stressed breakdown at 6 corners.

Corners	0.8V	1V	1.2V
25°C	0.04%	0.00%	0.12%
100°C	0.08%	0.08%	0.08%

to effectively eliminate the erroneous responses.

4.3.2.8 Uniqueness Evaluation

The inter-Fractional Hamming Distance (FHD) [37] is calculated as the uniqueness evaluation of SSUs. Consider the 24 voltage stressed SSUs on each chip as a 24-bit weak PUF [98], the distribution of inter-FHD of 99 chips are presented in Fig. 4.11. The average of inter-FHD is 51.7% and the standard deviation is 11.4%, where for an ideal Binomial distribution with success probability P=0.5, the mean is 50% and the standard deviation is 10.2%. Please note that the results of uniqueness evaluation are focused on the voltage stressed breakdown SSUs because for the plasma induced breakdown SSUs, the responses are highly biased and post processing would be required to extract randomness, for example using OR gates at the outputs of multiple SSUs to generate an unbiased bit as explained in Section 4.3.3.1.

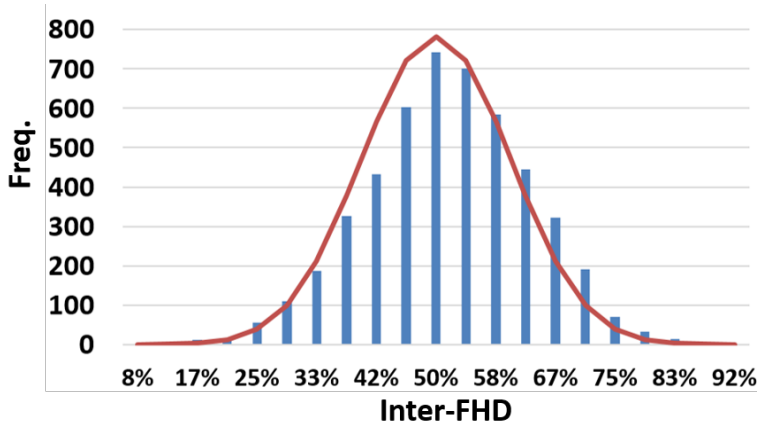


Figure 4.11: Inter-FHD distribution of voltage stressed SSUs on 99 chips overlaid with an ideal Binomial distribution curve with success probability P=0.5.

4.3.2.9 Statistical Analysis of the PUF Responses

In this section we provide a statistical analysis for the data of the fabricated SSUs after voltage stressed oxide breakdown as presented in Section 4.3.2. We evaluate the statistical dependence between pairs of bits using various statistical distance measures. We consider pairs as we have only 99 bits per location, and so going beyond the pairwise probability mass function can lead to more noisy and less reliable evaluation. We are interested in the level of independence because the more independent the bits are, the more secure the PUF is.

Essentially, we use that data to evaluate the pairwise probability mass functions of bits under the following two restrictions: The pairwise probability mass function of bits that have the same antenna ratio; the pairwise probability mass function of bits that are located next to each other. This in turn enables us to evaluate the statistical dependence of element that are more likely to be statistically dependent, that is, statistical dependence due to similar design rules as well statistical dependence between PUFs that are close together.

We calculate the distance between the evaluated probability mass function (i.e., $P_{X,Y}(x,y)$) and an independent one with the same marginal probability mass functions (i.e., $P_X(x) \cdot P_Y(y)$) by assigning them to various statistical distance measures. This enables us to demonstrate the level of independence between pairs of bits. The results are presented in Table 4.5 for the following statistical distance measures: The Kullback-Leibler (KL) divergence [99]; total variation distance (TVD) [100]; and guesswork (GW) [44].

Table 4.5 shows that the average statistical distance between $P_{X,Y}(x,y)$ and $P_X(x) \cdot P_Y(y)$ is very small across measures, which indicates that this PUF response is very close to being statistically independent.

4.3.3 Gate Oxide Breakdown PUF Implementations

4.3.3.1 Plasma Induced Breakdown PUF

Our measurement results show that the probability of plasma induced breakdown due to antenna rule violation is much lower than the ideal 50%, which means that most responses

Table 4.5: Statistical distances based on the collected data. In each entry the left side represents the statistical distance of bits that are located next to each other, whereas the right side represents the distance of bits that have the same antenna ratio.

Statistical Distance	Max	Min	Mean
KL	0.11/0.057	0.0002/0.0001	0.022/0.015
TVD	0.19/0.13	0.009/0.007	0.07/0.05
GW	0.06/0.029	0.0001/0.00009	0.011/0.008

are zeroes. To reduce the bias, we propose to use OR gates at the output of SSUs as a more area-efficient approach than using even larger antenna segments, which shows limited impact on increasing the breakdown probability. Fig. 4.12 (a) shows an exemplary implementation of plasma induced breakdown PUF. The 10MΩ precision resistor is shared between two SSUs, where only one of EVA_1 and EVA_2 will be asserted. Please note that a precision resistor can be shared by more than two SSUs but only one of the SSUs is asserted at a time. The outputs of buffer gates are determined by the breakdown of the SSU.

Take Test3 as an example. When 11 Test3 SSUs are ORed together, the probability of generating a zero is $(1 - 5.1\%)^{11} = 56\%$, and the area is $880\mu m^2$, which is still more area-efficient than a practical SRAM PUF implementation where (511,19,119)-BCH is suggested to correct 15% error probability at different corners [46]. For such SRAM PUF to generate 19 information bits, the estimated BCH implementation is 12000 XOR gates [54] or an area of $54000\mu m^2$ for the 65nm technology we used. To generate the same number of 19 bits of response with Test3, the estimated area is about $16720\mu m^2$. The comparison shows that the *SRAM PUF is more than 3X of size of the plasma induced breakdown PUF*. In addition, the ECC execution latency is eliminated for the plasma induced breakdown PUF.

4.3.3.2 Voltage Stressed Breakdown PUF

The probability of voltage stressed breakdown is much higher than the plasma induced breakdown, therefore no OR gates are needed to reduce the response bias, but a stress path for

each SSU is required. Fig. 4.12 (b) shows an exemplary implementation of voltage stressed breakdown PUF. A precision resistor is shared by 3 SSUs. Before response generation, the PUF is stressed through the stress path and outputs of SSUs are connected to GND with all EVA signals set to zero. Once SSUs are stressed, a normal voltage is applied to the stress path and one of the EVAs is asserted at a time for evaluation. To generate a bit, approximately 1 inverter and 4 transistors are needed, which translates to an area of only $4\mu m^2$ for 65nm technology. The PUF can be stressed on chip, for example with a charge pump with an area overhead of $12200\mu m^2$ [101]. Therefore, to generate 19 bits of response, the total area is approximately $12276\mu m^2$, which is about 30% smaller than the plasma induced breakdown PUF. As the number of bits increases, the area reduction becomes more evident since the charge pump is shared among multiple bits. The PUF can also be stressed from outside of the chip to save even more area, but an antifuse cell may be needed at the stress path. To stress the PUF, the antifuse cell has to be permanently programmed to closed state. Therefore, if the antifuse cell is already in closed state before stress, it means that the PUF has been contaminated and should be discarded. Please note that if the PUF is stressed from outside of the chip, an attacker may destroy the PUF or introduce more breakdowns by further stressing the PUF, but the PUF is not programmable or clonable because the breakdown of each transistor cannot be controlled.

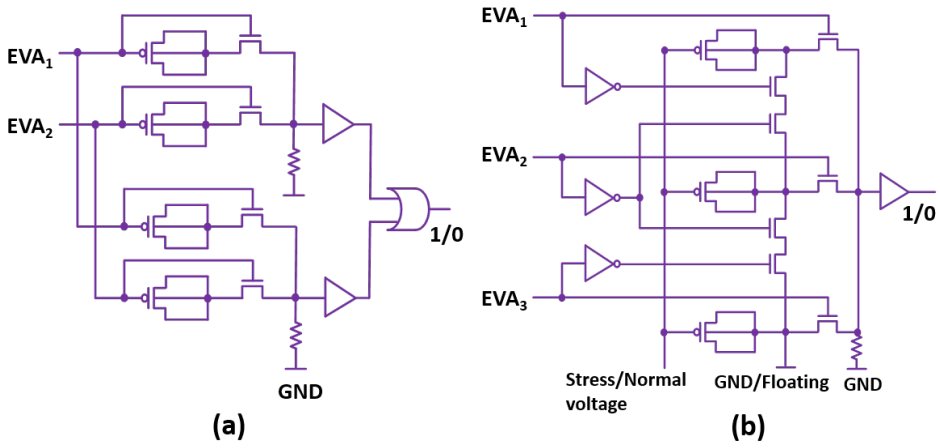


Figure 4.12: (a) Plasma induced breakdown PUF implementation. (b) Voltage stressed breakdown PUF implementation.

4.4 Conclusion

In this chapter we propose the first stability-guaranteed PUF that requires no stability enhancement techniques, where the source of randomness is extracted from (1) locally enhanced DSA process and (2) gate oxide breakdown. Detailed constructions of two DSA LEDPUFs: the weak DSA LEDPUF and the strong DSA LEDPUF, are presented. Inter-distance measurements on the LEDPUFs show that both weak and strong LEDPUFs are ideally unique. The area and latency of the weak LEDPUF is much smaller than existing weak PUFs because no error correcting schemes are needed. The strong LEDPUF provides large CRP space because of its cryptographic hash based structure. The weak LEDPUF used in the strong LEDPUF construction cannot be replaced by existing weak PUFs because an absolute 0% intra-distance is required for the weak PUF to avoid the avalanche effect of the strong LEDPUF. Furthermore, we quantify the level of security provided by weak LEDPUF by calculating the expected guesswork resulting from weak LEDPUFs empirical probability function; the loss compared to a fair coin toss is negligible.

We also implement highly stable PUFs exploiting uncontrollable plasma induced and voltage stressed gate oxide damage. The proposed SSUs are fabricated and measured from 99 testchips. Measurement results show that the SSUs are highly stable, therefore significant area reduction can be achieved by eliminating ECC implementation. We show that the responses are unbiased and unique, and we analyze the data of our testchips using various statistical distance measures to show that these bits are independent.

CHAPTER 5

PUF Security Evaluation through Guesswork Analysis

[1] In this chapter we develop a new unified framework for evaluating the security of PUFs, based on password security, by using information theoretic tools of guesswork. The guesswork model allows us to quantitatively compare, with a single unified metric, PUFs with varying levels of stability, bias and available side information. In addition, it generalizes other measures to evaluate the security level, such as min-entropy and mutual information. We evaluate guesswork based security of some measured Static Random Access Memory (SRAM) and Ring Oscillator PUFs as an example and compare them with LEDPUFs to show that stability has a more severe impact on the PUF security than biased responses. Furthermore, we find the guesswork of two new problems: guesswork under probability of attack failure, and the guesswork of strong PUFs that are used for authentication.

5.1 Introduction

In order to impersonate the hardware, the PUF attacker needs to respond to a challenge with a correct response (*i.e.,* guess a secret). [2] Comprehensive security models for PUFs are described in [102], including a precise identification of required PUF properties, such as indistinguishability and tamper-resilience. Though this specifies the security requirements, as a "checklist", we believe that a more quantitative assessment of PUF security can be valuable for both PUF designers and PUF users. In this chapter we do not study "machine learning"

[1] This work is done in collaboration with Prof. Suhas Diggavi and Dr. Yair Yona and their contribution is much appreciated.

[2] In this attacker model the adversary does not have access to the PUF, but rather is trying to impersonate it.

attacks, in which the attacker has access to the PUF and is challenging the PUF to learn the underlying randomness. Inter- and intra- fractional Hamming Distance (FHD) [31], and other statistical tests for randomness [103], have been used for quantifying PUF security. Though it is reasonable that having larger inter-FHD is more secure, it does not tell the PUF designer how much more secure it is. For example, is it worth raising the inter-FHD from 40% to 49% at a cost of extra hardware? In this work we present a more principled way to analyze PUF security by connecting it to how one could evaluate password security, through a guesswork framework. We derive a theoretical framework for PUF security evaluation that brings together two important properties of existing PUFs: predictability and reproduceability [3]. This framework enables a unified security quantification of several effects including bias, noise, and side-channels on PUFs, as well as the security over multiple challenge-response pairs, providing design guidance by quantifying the security level of a PUF.

In the context of the question raised in the previous chapter, we show that in terms of guesswork the effect of noise is far worse than the effect of bias. Therefore, the tools presented in this chapter to evaluate PUFs security can be used by a PUF designer to determine how to maximize the security level at minimum cost in terms of resources. The actual answer to this question depends on the trade-off between the security level in terms of guesswork, and the actual cost in resources on the designer end, required to achieve this.

One can think of PUF signatures like passwords, and their breakability should be evaluated by how strong they are, for example, how many attempts (on average) does it take to compromise them. This *guessing* framework has been studied in the information theory literature and has recently been adopted by NIST as a measure of password security [104]. We bring this framework to evaluating the security of PUFs.

5.2 Guesswork as A Unified Framework for Evaluating The security level

5.2.1 Why Consider Guesswork?

Consider the following game: Bob draws a sample x from a random variable X, and an attacker Alice who does not know x but knows the probability mass function $P_X(\cdot)$, tries to guess it. An oracle tells Alice whether her guess is right or wrong. This is the situation where an attacker tries different passwords to access an account.

If Alice has only one guess, then the optimal strategy that maximizes her chance of guessing x successfully is choosing the most probable x. In this case the chance of guessing x is $\max_{x' \in X} P_X\left(x'\right)$ and the *predictability* of X is given by its min-entropy [74]:

$$H_\infty(P_X) = -\log_2\left(\max_{x' \in X} P_X\left(x'\right)\right). \tag{5.1}$$

If Alice is allowed to make as many guesses as required until she finds x, then the optimal strategy is guessing elements in X based on their probabilities in ascending order [75]. It has been shown that the average number of guesses it takes Alice to find x (denoted by $G(X)$ and termed guesswork) is not given by the traditional Shannon entropy [75]. For example, when drawing a random vector of length m, $\underline{X}$, which is independent and identically distributed (i.i.d.) with distribution $P = [p_1, \ldots, p_L]$, the exponential growth rate of the guesswork scales according to the Renyi entropy $H_\alpha(X)$ with parameter $\alpha = 1/2$ [75]:

$$\lim_{m\to\infty} \frac{1}{m} \log_2\left(E\left(G\left(X\right)\right)\right) = H_{1/2}(P) = 2 \cdot \log_2\left(\sum_i p_i^{1/2}\right) \tag{5.2}$$

where $H_{\frac{1}{2}}(P) \geq H(P)$ with equality only for the uniform probability mass function (in the context of Figure 4.5 m is the length of the PUF response and $L = 2^m$).

The security of a PUF is predicated by the inherent random signature in the hardware. An attacker wants to either impersonate a hardware by guessing its random signature, or to learn it by eavesdropping.[3] In order to impersonate the hardware, the attacker needs to

[3] In this attacker model the adversary does not have access to the PUF, but rather is trying to impersonate it.

respond to a challenge with a correct response. In order to evaluate the security of a PUF we connect to the framework for password security [104]. For a dictionary attack, a guessing framework quantifies security through the number of guesses the impersonator has to make in order to identify the password (or inherent randomness) and therefore respond correctly to all possible challenges. Therefore, we quantify the security level of a PUF through the number of guesses required to break it.

In subsection 5.2.3 we show that guesswork can serve as a unified framework for evaluating and quantifying the security of PUFs. Essentially, other measures of evaluating the security level such as min-entropy and mutual information are special cases of guesswork (as shown in Subsection 5.2.3 min-entropy is the probability of correctly guessing the PUF response in a single guess, that is, the probability that the guesswork is equal to 1, and so it does not capture the entire probability mass function of the number of guesses; in terms of guessing, guesswork is a more general security criterion than min-entropy). Furthermore, guesswork allows to quantify the security level under more elaborate scenarios such as the security level when key stretching mechanism is used [105] as well as when allowing an attack failure probability (this problem is presented in Subsection 5.2.3).

Characterizing the security of a PUF through guesswork reveals an interesting interplay between the bias of a PUF response, and the noise (due to instability) which is incorporated in each sample. Guesswork is very sensitive to the presence of instability, but yet is not very susceptible to bias. These properties are discussed in subsection 5.2.5. Therefore, guesswork highlights the advantage of stable PUFs over unstable PUFs, when evaluated through the number of guesses required to break a PUF.

Moreover, we present a formal evaluation methodology for PUFs security, while identifying the impact of bias, noise and side-channels.

Note that the interplay between noise and bias, as well as the advantage of stable PUFs have been reported in the literature. For example, [106] discusses the advantage of stable PUFs, and [107] considers the interplay between noise and the PUF response in terms of efficient post-processing methods. However, our results provide methods to analytically

evaluate the security level of PUFs, based on their fundamental properties (i.e., noise level and bias).

5.2.2 Background

The guesswork $G(X)$ is a random variable that represents the number of guesses required to guess a random variable x. Therefore, the probability of having $G(x)$ guesses is $P_X(x)$. The ρth moment of guesswork is

$$E\left(G\left(X\right)^{\rho}\right)=\sum_{x}G\left(x\right)^{\rho}\cdot p_X\left(x\right). \tag{5.3}$$

The definition of guesswork can be extended to the case where the attacker has a side information Y available. In this case the average guesswork for $Y=y$ is defined as $G(X|Y=y)$, and the ρth moment of $G(X|Y)$ is

$$E\left(G\left(X|Y\right)^{\rho}\right)=\sum_{y}E\left(G\left(X|Y=y\right)^{\rho}\right)\cdot p_Y\left(y\right). \tag{5.4}$$

Massey [76] noted that a dictionary attack minimizes the expected number of guesses (i.e., guessing the values in the decreasing order of $P_X(x)$). Arikan [75] has bounded the ρth moment of the optimal guesswork, $G^*(X|Y)$, by

$$\begin{aligned}\left(1+\ln\left(M\right)\right)^{-\rho}\sum_{y}\left(\sum_{x}P_{X,Y}\left(x,y\right)^{\frac{1}{1+\rho}}\right)^{1+\rho}\leq\\ E\left(G^*\left(X|Y\right)^{\rho}\right)\leq\sum_{y}\left(\sum_{x}P_{X,Y}\left(x,y\right)^{\frac{1}{1+\rho}}\right)^{1+\rho}\end{aligned} \tag{5.5}$$

where $M=|X|$ is the cardinality of X. Furthermore, in [75] it has been shown that when X and Y are strings of length m, where the pairs (X_i,Y_i) are drawn i.i.d.and $1\leq i\leq m$, the exponential growth rate of the optimal guesswork is

$$\lim_{m\to\infty}\frac{1}{m}\log_2\left(E\left(G^*\left(X|Y\right)^{\rho}\right)\right)=\rho\cdot H_{\frac{1}{1+\rho}}\left(P_{X,Y}\left(x,y\right)\right) \tag{5.6}$$

where m is the size of X and Y, and

$$H_{\frac{1}{1+\rho}}\left(P_{X,Y}\left(x,y\right)\right)=\frac{1}{\rho}\log_2\left(\sum_{y}\left(\sum_{x}P\left(x,y\right)^{\frac{1}{1+\rho}}\right)^{1+\rho}\right) \tag{5.7}$$

Table 5.1: The average guesswork as a function of m when $p = 1/2$.

$\rho = 1$, $p = 1/2$	m=64	m=128	m=256	m=1024
Lower bound	$1.4 \cdot 2^{58}$	$1.4 \cdot 2^{121}$	$1.4 \cdot 2^{248}$	$1.4 \cdot 2^{1014}$
Upper bound	2^{64}	2^{128}	2^{256}	2^{1024}

is Renyi's conditional entropy of order $\frac{1}{1+\rho}$ [75].

Two remarks are in order regarding why considering the growth rate is meaningful.

Remark 1 (The non-asymptotic behavior is also dictated by growth rate)**.** *Note that although* (5.6) *is an asymptotic result, it converges very quickly. This is because in [75] it was shown that the guesswork of any moment is lower bounded by* $(1 + m \cdot \ln(2))^{-\rho} \cdot 2^{m \cdot \rho \cdot H_{1/(1+\rho)}(p)}$ *and upper bounded by* $2^{m \cdot \rho \cdot H_{1/(1+\rho)}(p)}$*, when* X *is of size* m *and is drawn i.i.d. Bernoulli*(p)*. Table 5.1 presents the lower and upper bounds for various values of* m*.*

Remark 2 (the operational meaning of growth rate)**.** *Based on the bounds presented in Table 5.1 it can be shown that even for finite values of* m*, a decrease in growth rate can have a tremendous impact on the security level. For example, when* $p = 0.0015$*, which leads to* $H_{1/2}(0.0015) = 0.1$*, the average guesswork for* $m = 256$ *is lower bounded by* $1.4 \times 2^{17.6}$ *and upper bounded by* $2^{25.6}$ *which is far smaller than* 2^{256} *(or a lower bound of* 1.4×2^{248} *as presented in Table 5.1) in the case when* $p = 1/2$*.*

Furthermore, growth and decrease rates are commonly used when evaluating the security level. For instance the min-entropy is a measure of the decrease-rate of the probability of guessing a password.

Another extension of guesswork [108] considers a game in which it is sufficient to guess x up to a certain level of distortion D, according to some distance metric $d(x, \hat{x}(i)) = \sum_{i=1}^{m} d(x_i, \hat{x}(i))$. Essentially, when $G(x) = i$, the word $\hat{x}(i)$ is guessed such that $d(x, \hat{x}(i)) \leq m \cdot D$, that is, when the attacker guesses a word which is within a Hamming distance $m \cdot D$ of x the game is over. The authors in [108] have solved this problem for the general case; more

specifically, for a binary source which is drawn i.i.d. Bernoulli(p) and Hamming distortion

$$d\left(x_j, \hat{x}_j\right) = \begin{cases} 1 & x_j = \hat{x}_j \\ 0 & x_j \neq \hat{x}_j \end{cases} \tag{5.8}$$

where $1 \leq j \leq m$, they have shown that the exponential growth rate of the guesswork equals

$$\lim_{m\to\infty} \frac{1}{m} \log_2 \left(E\left(G^*\left(X, D\right)^\rho\right)\right) = \rho \cdot E\left(D, p\right) = \max\left(\rho H_{\frac{1}{1+\rho}}\left(p\right) - \rho \cdot H\left(D\right), 0\right) \tag{5.9}$$

where $H\left(D\right) = -D\log_2\left(D\right) - \left(1-D\right)\log_2\left(1-D\right)$ is the binary Shannon entropy [99].

Guesswork has been analyzed in many other scenarios such as guessing under source uncertainty with and without side information [109], [110], using guesswork to lower bound the complexity of sequential decoding [75], guesswork for Markov chains [111], and guesswork for multi-user systems [112].

5.2.3 Extending Guesswork

In this subsection we extend the definition of guesswork and show that it can serve as a unified framework for evaluating the security level of PUFs by incorporating noise and bias. In addition, we relate guesswork to other measures such as mutual information and min-entropy.

We begin by finding the moments of the guesswork of a noisy weak PUF.

Theorem 1. *When the response of a weak PUF is noisy such that the noise is additive and drawn i.i.d. Bernoulli(D), and the original response is i.i.d. Bernoulli(p) , the ρth moment of the guesswork increases at rate $\rho \cdot E\left(D, p\right)$ as defined in* (5.9).

Proof. The idea behind the proof is that guessing within Hamming distance $m \cdot D$ of the original response, enables the attacker to find the original response by using the helper-data.

Essentially there are two options. The first possibility is to construct a code to guess the original response up to Hamming distance $m \cdot D$ as is done in [108], and then use the helper-data in order to find the original response, in which case the rate of the ρth moment

is $\rho \cdot E(D,p)$. The second possibility is to use the helper data (e.g., the coset of an ECC) to guess over a subset. In this case, since the helper-data breaks up the space into subspaces of size $2^{(1-H(D))\cdot m}$ [99], guessing through the subspace can only bring the rate of the ρth moment down to $\rho \cdot E(D,p)$. Therefore, the minimal rate is $\rho \cdot E(D,\rho)$. □

Remark 3. *$\rho \cdot E(D,p)$ is the maximum rate at which the ρth moment of the guesswork of a noisy PUF can increase. This can be achieved by employing an ECC that operates very close to the channel capacity under the statistical profile of the noise [99]. However, the hardware size required to implement this ECC may be large, and so PUF designers may resort to other ECCs that can be implemented more efficiently in terms of their hardware size but yet cannot achieve the channel capacity. In this case, the decrease in rate is expressed in the amount of information revealed by the helper data, W, whose entropy is larger than the noise entropy, that is, $H(W) \geq H(D)$. This in turn brings the guesswork down to $\max\left(\rho H_{\frac{1}{1+\rho}}(p) - \rho \cdot H(W), 0\right) \geq \rho \cdot E(D,p)$.*

Remark 4. *A meaningful way to compare the efficiency of PUFs is by fixing their security level as well as the probability of error of the ECC to certain values and then compare the hardware size required to achieve these by different PUFs, as presented in [106]. In [106] the security level is evaluated through the Shannon entropy. However, Guesswork can also be used in the method presented in [106] to evaluate the security level. This in turn provides a measure that has a wider operational meaning in terms of security than the Shannon entropy. For example, the average guesswork is directly related to the average number of guesses required to guess the PUF response (this is highlighted in Theorem 2 and Remark 8).*

Remark 5. *In [107] and [113] the noise of a PUF is not distributed Bernoulli(D), but rather is asymmetric and affected by the bias level such that the conditional transition probability is different when the PUF response is equal to* 1 *and* 0*, and the combined transition probability is D (see subsection 5.2.5 for more details). In this asymmetric case the rate at which the ρth moment of the maximum guesswork increases is lower bounded by $\rho \cdot E(D,p)$ due to the convexity of of $E(D,p)$ in the noise distribution.*

Before we present a new game that extends the traditional definition of guesswork, let

us define the type of a vector.

Definition 1. *Consider a binary vector x of size m and assume that $N(x|1)$ is the number of elements of this vector that are equal to 1. In this case when $N(x|1)/m = q$ the vector x is of type q.*

We now define a new guessing game that captures different measures for evaluating the security level.

Definition 2 (Guesswork under attack failure constraint)**.** *Consider the following game: Bob draws a vector x of size m i.i.d. Bernoulli(p). The attacker Alice has to guess x up to Hamming distance $m \cdot D$ as defined in subsection 5.2.2, under the constraint that the probability of attack failure is smaller than or equal to $2^{-\alpha \cdot m}$ where $\alpha \geq 0$, that is, Alice may decrease the number of guesses by guessing only a subset of all possible words, which leads in turn to a certain probability of attack failure. We define the optimal guesswork for this game as $G^*(X; D, \alpha)$. Furthermore, we define the guesswork in the case where the probability of attack failure is zero as $G^*(X; D, \infty) = G^*(X; D)$.*

Remark 6. *The relation between $G^*(X; D, \alpha)$ and previous works is as follows:*

- *$Pr(G^*(X; 0, \infty) = 1) = 2^{-m \cdot H_\infty(p)}$, that is, the min-entropy.*
- *$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G^*(X; D, \infty)^\rho\right)\right) = \rho \cdot E(D, p)$ as defined in (5.9).*

The following theorem characterizes a lower bound for $G^*(X; D, \alpha)$ for any moment $\rho > 0$ in the case where the attacker is allowed not to guess certain types.

Theorem 2.

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G^*(X; D, \alpha = D(s||p))^\rho\right)\right) \leq \begin{cases} \rho \cdot \left(H_{\frac{1}{1+\rho}}(p) - H(D)\right) & s^* \leq s \leq 1 \\ \rho \cdot (H(s) - H(D)) - D(s||p) & p < s \leq s^* \end{cases}$$

$0 \leq D \leq p \leq 1/2$, *and*

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G^*(X; D, \infty)^\rho\right)\right) = \rho \cdot \left(H_{\frac{1}{1+\rho}}(p) - H(D)\right)$$

where $s^* = \frac{p^{(1+\rho)^{-1}}}{p^{(1+\rho)^{-1}}+(1-p)^{(1+\rho)^{-1}}}$*, the probability of attack failure decreases like* $2^{-m\cdot\alpha}$*,*

$$D\left(s||p\right) = s \cdot \log_2\left(s/p\right) + \left(1-s\right) \cdot \log_2\left(\left(1-s\right)/\left(1-p\right)\right)$$

is the Kullback-Leibler divergence [99], and Alice chooses a set $A = \{q_1, \ldots, q_L\}$ *of types whose vectors are not guessed, such that the probability that* $N\left(x|1\right)/m \in A$ *is smaller than or equal to* $2^{-\alpha\cdot m}$*, that is, Alice guesses words in* A^C*.*

Note that $\rho \cdot H\left(s^*\right) - D\left(s^*||p\right) = \rho \cdot H_{\frac{1}{1+\rho}}\left(p\right)$

The next three remarks point out a few properties of $G^*\left(X; D, \alpha\right)$.

Remark 7. *When an attacker attempts to break a very large number of independent PUF responses (or passwords), where the probability of attack failure is* $2^{-m\cdot\alpha}$*, he is very likely to break a fraction of* $1 - 2^{-m\cdot\alpha}$ *of the PUF responses (passwords), and this in turn leads to* ρ*th moment of guesswork across PUF responses (passwords) that increases at a rate* $\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G^*\left(X; D, \alpha\right)^\rho\right)\right) \leq \rho \cdot \left(H_{\frac{1}{1+\rho}}\left(p\right) - H\left(D\right)\right) = \rho \cdot E\left(D, p\right)$*, when* $D \leq p$*, that is, the moments of guesswork decrease as the probability of attack failure increases.*

Remark 8. *When* $s = p + \epsilon$ *the average guesswork is approximately* $H\left(p\right) - H\left(D\right)$*, which is the rate distortion function of Hamming distortion [99].*

Remark 9. *Note that when* $p = 1/2$ *also* $s^* = 1/2$ *and the upper bound of Theorem 2 is equal to* $\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G^*\left(X; D, \infty\right)^\rho\right)\right)$*, that is, when guessing according to the method presented in Theorem 2 Alice does not gain anything from having a failure probability larger than zero.*

We now derive an expression for the min-entropy when the attacker has to guess a word that is within Hamming distance $m \cdot D$ of the password. In this case the min-entropy of a binary i.i.d. source subject to Hamming distortion D is equivalent to choosing a word which is in a ball of radius $m \cdot D$ around the most likely word (i.e., the probability of guessing a word which is in the most likely ball). The asymptotic value of the min-entropy subject to distortion D is given by the following lemma.

Lemma 1. *Consider a binary word of length m for which each element is drawn i.i.d. from Bernoulli(p). The min-entropy subject to Hamming distortion D converges to*

$$-\lim_{m\to\infty}\frac{1}{m}\log_2\left(P_X^{ball}\right)=\begin{cases} D\left(D||p\right) & 0\le D\le p \\ 0 & p<D\le 1 \end{cases} \tag{5.10}$$

where $p\le 1/2$, $P_X^{ball}=\sum_{i=0}^{m\cdot D}\binom{m}{i}p^i\left(1-p\right)^{m-i}$.

Remark 10. *Note that the result of Lemma 1 can also be interpreted as $Pr\left(G^*\left(X;0,\infty\right)\le 2^{m\cdot H(D)}\right)$ when a password of length m is drawn i.i.d. Bernoulli(p), where $0\le D\le 1/2$.*

5.2.4 Examples for Quantifying the Security of PUFs

In this subsection we present a few examples that illustrate how to use guesswork in order to quantify the security level of PUFs. We address evaluations for unstable PUFs as well as for stable PUFs. We incorporate into the expressions noise, bias, and side information coming from other PUFs or from side channel/model attacks.

The first step in calculating the guesswork of a PUF is evaluating the probability function according to which it is drawn, as well as the noise level. In this subsection we assume that the bits are i.i.d. for which case the first step is evaluating the bias of the stable bits and then estimating the noise level of the unstable bits; evaluating the bias of the stable bits enables us to state that the PUF response is drawn i.i.d. from the probability function

$$P_0=p \quad P_1=1-p \tag{5.11}$$

whereas the probability of transition of a bit when re-sampling a PUF is q, such that

$$x^{(2)}=x^{(1)}\oplus e \tag{5.12}$$

where $x^{(1)}$, $x^{(2)}$ are the first and second samples of the unstable (noisy) PUF, and $Pr\left(e_j=1\right)=q$, $1\le j\le m$.

For stable PUFs such as the LEDPUF, it is sufficient to calculate the bias and assign the probability function to $\rho\cdot H_{\frac{1}{1+\rho}}\left(P\right)$ in equation (5.6), in order to get the ρth moment of

guesswork. For example, when the stable PUF is drawn i.i.d. according to Bernoulli(0.47) the average guesswork of a PUF of large enough size ($m = 256$, say) is proportional to

$$2^{H_{1/2}(0.47)\cdot m} = 2^{0.9987\cdot m} \tag{5.13}$$

whereas the largest guesswork that we can expect for is achieved by an unbiased PUF for which each bit is drawn i.i.d. Bernoulli(0.5), and is proportional to

$$2^{H_{1/2}(0.5)\cdot m} = 2^{m}. \tag{5.14}$$

For unstable PUFs, re-sampling the PUF yields a noisy version of the original response as presented in equation (5.12). When the probability of transition is q, Theorem 1 shows us that it is sufficient to guess the original response $x^{(1)}$ up to Hamming distance $m \cdot q$. The intra distance can be used to evaluate the noise level. For example, when considering an unbiased unstable PUF with a transition probability $q = 0.1$, we get that the guesswork is proportional to

$$2^{m\cdot(1-H(0.1))} = 2^{0.531\cdot m} \tag{5.15}$$

which means that noise decreases the average number of guesses significantly.

The conditional guesswork (5.4) enables us to quantify the effect of side information on the security level of both stable and unstable PUFs. In order to evaluate the conditional guesswork we first need to characterize the conditional probability. The conditional probability depends on the type of attack which is being carried; in some cases characterizing its effect on the randomness of the response requires some effort. A simple example for a side information attack is one in which an attacker has another PUF which is correlated with the original one. For example, consider an unbiased stable PUF x for which each element is drawn i.i.d. Bernoulli(0.5), and assume that an attacker has another unbiased stable PUF, y, which is correlated with x such that

$$P\left(y|x\right) = P\left(e\right) \tag{5.16}$$

where e is drawn i.i.d. Bernoulli(0.2). In this case the unconditional guesswork $G\left(X\right)$ is proportional to

$$2^{H_{1/2}(1/2)\cdot m} = 2^{m} \tag{5.17}$$

whereas the conditional guesswork $G(X|Y)$ is proportional to

$$2^{H_{1/2}(0.2)m} = 2^{0.848 \cdot m} \tag{5.18}$$

because of the fact that in this case x given y is also drawn i.i.d. Bernoulli(0.2). In general, the correlation between PUFs can be evaluated through the inter distance.

Conditional guesswork subject to distortion enables to evaluate the guesswork of an unstable PUF when side information is available. The method of evaluating the guesswork is similar to the previously mentioned methods for evaluating conditional guesswork and guesswork subject to distortion.

5.2.5 The Effect of Noise Vs. The Effect of a Bias

In this subsection we analyze the expressions for guesswork as well as min-entropy subject to distortion, and quantify the impact that noise and bias have on PUFs. Furthermore, we show that the effect of noise is far worse than the effect of bias in terms of average guesswork.

First, let us focus on the effect of noise and bias on the expected value of the guesswork (i.e., the case where $\rho = 1$), when the noise is Bernoulli(D). From Theorem 1 we get that when the transition probability is D, the asymptotic growth rate of the expected value of the guesswork is

$$H_{1/2}(P_X) - H(D). \tag{5.19}$$

On the other hand the asymptotic growth rate of the expected value of the guesswork of a stable PUF whose bits are drawn i.i.d. from Bernoulli(p) is

$$H_{1/2}(p) = 2 \cdot \log_2\left(\sqrt{p} + \sqrt{1-p}\right). \tag{5.20}$$

The first derivative of equation (5.19) equals

$$\log_2(D) - \log_2(1-D) \tag{5.21}$$

which diverges as D approaches 0. Therefore, even when the noise level (and D) is very small, it decreases the expected value significantly. On the other hand the first derivative of

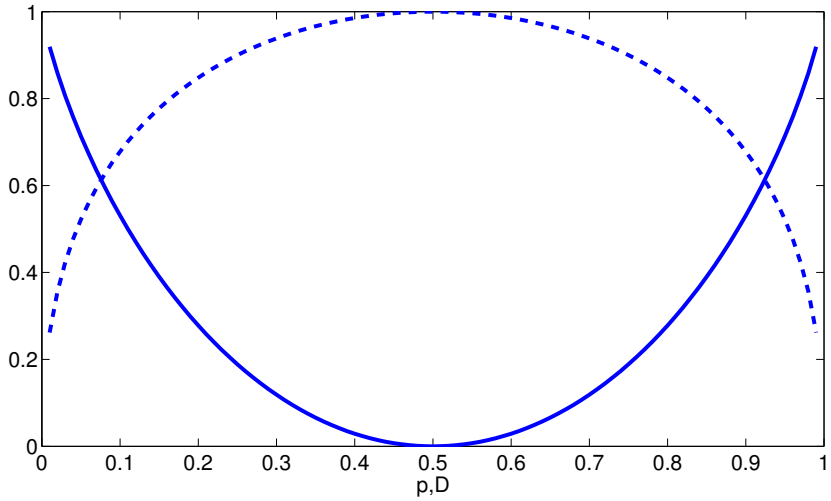


Figure 5.1: The solid line presents the average guesswork of an unstable unbiased PUF $1-H(D)$, whereas the dotted line is the average guesswork of a stable biased PUF $H_{1/2}(p)$.

(5.20) is equal to zero at $p = 1/2$ (i.e., when there is no bias). The first derivative around $p = 1/2$ is very small and therefore bias does not affect the guesswork as much as noise. Figure 5.1 presents the guesswork of an unstable unbiased PUF and the guesswork of a stable biased PUF.

For example, the asymptotic exponential growth rate of the guesswork of an unbiased ($p = 1/2$) unstable PUF with transition probability $D = 0.1$ (i.e., a 10% noise) is equal to 0.53 which is the guesswork of a stable biased PUF with $p = 0.05$ (i.e., a 95% bias).

In terms of min-entropy as presented in Lemma 1, the divergence $D(D||p) = -H(D) - D\log_2(p) - (1-D)\log_2(1-p)$ and therefore its first derivative also diverge as D goes to zero. Therefore, min-entropy is also very sensitive to the presence of noise. On the other hand, the min-entropy of a stable PUF is equal to

$$-m\log_2(1-p). \tag{5.22}$$

The first derivative of (5.22) equals $\frac{m}{1-p}$ when $0 \leq p \leq 1/2$ and therefore it does not diverge. Hence, the effect of bias on the min-entropy is also less significant than the effect of noise.

For example, the asymptotic min-entropy of an unbiased ($p = 1/2$) unstable PUF with

Table 5.2: The average guesswork when the transition probability of the noise has states versus the case when the noise is Bernouli(D) (in parentheses)

p/D	$D = 0.1$	$D = 0.2$
$p = 0.49$	0.53 (0.53)	0.2781 (0.2779)
$p = 0.4$	0.5197 (0.516)	0.2707 (0.2634)

transition probability $D = 0.1$ is equal to $1 - H(0.1) = 0.53$ which is the min-entropy of a stable biased PUF with $p = 0.31$ (i.e., a bias level of 69%).

Note that in general the first derivative of the min-entropy does not equal to zero at $p = 1/2$, and therefore bias has a stronger effect on min-entropy than on average guesswork.

Figure 5.2 presents the behavior of the min-entropy as a function of p. It shows that it is more sensitive to bias than $H_{1/2}(p)$.

So far we have discussed the case where the noise is Bernoulli(D). However in [107] and [113] it was shown that the conditional probability of the noise can be as follows

$$P(e = 1|X = 0) = \frac{D}{2 \cdot (1 - p)}$$

and

$$P(e = 1|X = 1) = \frac{D}{2p}$$

which satisfies both $P(e = 1) = D$ and $P(X = 1) = p$. In this case as stated in Remark 5 $\rho \cdot E(D, p)$ is a lower bound on the rate at which the ρth moment of the guesswork increases, due to the concavity of the Shannon entropy. However, the actual rate at which the ρth moment increases can be related to the amount of information revealed by the helper data, that is, $H(D|X)$ such that the rate at which the average guesswork increases is equal to $H_{1/2}(P) - H(D|X) \geq H_{1/2}(P) - H(D)$. Table 5.2 presents $H_{1/2}(P) - H(D|X)$ for various values of p and D versus $H_{1/2}(P) - H(D)$; these results show that the behavior in both cases is very similar and that the lower bound is very tight.

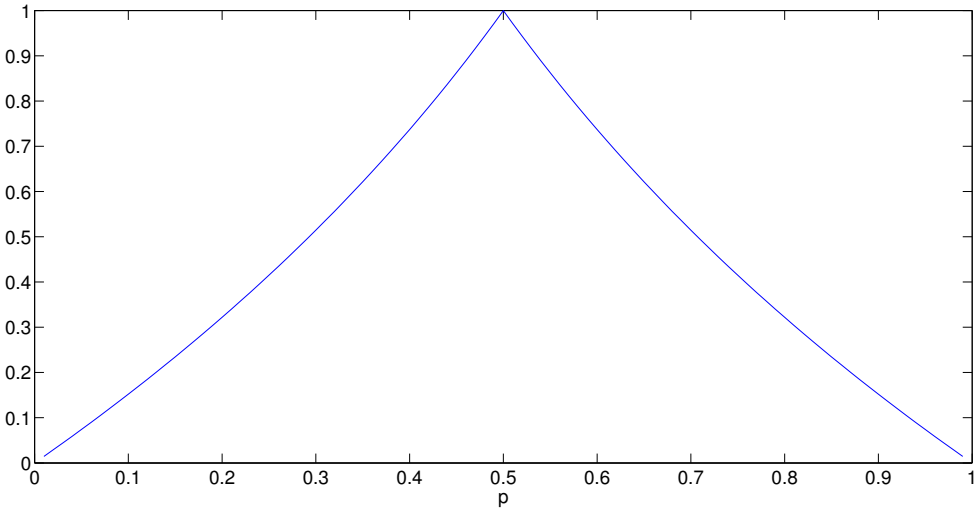


Figure 5.2: The min-entropy as a function of p. Note that the first derivative is always larger than zero.

5.3 Evaluating the Security Level of Weak PUFs Through Guesswork

5.3.1 Evaluation of Weak LEDPUF

We evaluate the probability mass function of a bit generated by a weak LEDPUF based on simulation results for the formation of connections

$$p_X(1) = 0.4626 \quad p_X(0) = 0.5374. \tag{5.23}$$

The uniqueness is evaluated by calculating the fractional inter-distance [31] of 1000 weak LEDPUFs, each producing 512 bits of response. The distribution is with mean=0.503 and standard deviation=0.02. Since the variance value is proportional to the inverse of the length of the response, as the length of the response increases the variance value goes to zero while the mean value goes to 0.505.

Table 5.3: Noisy PUF measurements. All numbers are percentages.

	intra-FHD	inter-FHD	Stability	Bias Level	
				One	Zero
SRAM PUF	2.26	48.33	93.42	49.13	50.87
RO PUF	2.48	47.13	91.19	51.38	48.62

5.3.2 Measurements of Noisy Weak PUFs

Two noisy silicon PUFs: SRAM PUF and RO PUF, are measured at 20°C in our experiments. For the SRAM PUF, responses from 10 commercial 45nm SOI test chips with 176k byte of SRAM cells each are obtained. Every SRAM PUF is measured 10 times. The RO PUF is implemented on 15 Altera DE2-115 FPGA boards. To avoid correlated CRPs, 90 CRPs are generated from the 91 ROs in each RO PUF. Each RO PUF is measured 10 times. No error correcting techniques are applied on these PUFs.

The measurement results of noisy PUFs are summarized in Table 5.3. The intra-FHD and inter-FHD are given in the second and third columns, respectively. Both PUFs show good results of small intra-FHD and close to 50% inter-FHD. The stability shown in the forth column gives the percentage of stable bits through all 10 measurements, where a stable bit is a bit that remains the same during all measurements. A 93% stability for the SRAM PUF, for example, means that 7% of the bits flip at least once during the 10 measurements. For LEDPUF, the intra-FHD is 0% and the stability is 100%. The bias level (percentages of ones and zeros) are given in the last two columns.

For the RO PUF, in addition to the intra-FHD at 20°C, we also compare the intra-FHD between 20°C and 60°C, which is the reliability of the PUF if it is enrolled at 20°C but verified at 60°C. The results are presented in Figure 5.3. We can see that for most PUFs, the averaged intra-FHD at the extreme temperature is about 12%, which implies that conventional ECC margin with error reduction techniques for the PUF would be required.

For noisy SRAM PUF and RO PUF, the expected growth rate is calculated by plugging the intra-FHD to equation (5.19). The expected growth rate of weak LEDPUF is obtained by

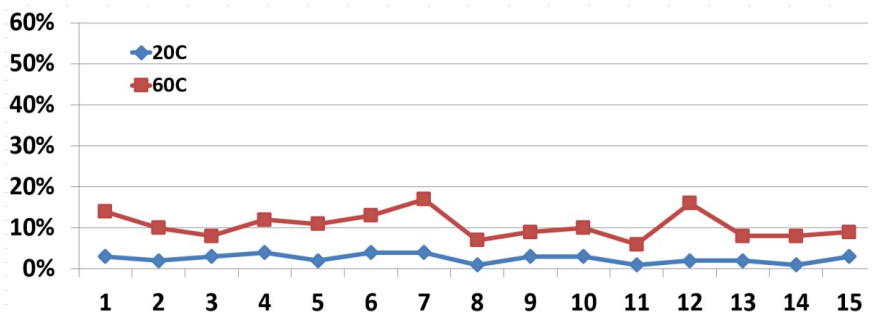


Figure 5.3: Intra-FHD at extreme temperature variation for 15 RO PUFs.

Table 5.4: Growth rate of the expected value of the guesswork. When the key size of the PUF is 32, the average guesswork of the SRAM PUF is proportional to $2^{32\times 0.8442}$, and the average guesswork of the LEDPUF is proportional to $2^{32\times 0.9980}$.

PUF Type	SRAM	RO at 20°C	RO at 60°C	LEDPUF
Growth rate	0.8442	0.8323	0.4706	0.9980

applying the bit probabilities given in (5.23) to equation (5.20). The results are summarized in Table 5.4. We can see that even though the weak LEDPUF is more biased than the noisy PUFs, its guesswork growth rate is still higher than noisy PUFs. For RO PUF at 60°C, the guesswork growth rate becomes much worse compared with RO PUF at 20°C, which implies quantitatively how insecure a PUF can become under environmental variations.

To give an estimated area comparison if error correcting techniques are applied, for a SRAM PUF to generate a secure 128-bit response with Equal Error Rate (EER) $< 10^{-9}$, the area required is about $1630\mu m^2$ using 65nm technology [106]. Using the same technology, for the weak LEDPUF to generate a bit, the area of a SSU as shown in Figure 4.4 is $3.24\mu m^2$. From Table 5.4 we know that for weak LEDPUF to generate a secure 128-bit response, we need $\frac{128}{0.998}$ bits (SSUs), which roughly translates to an area of $415\mu m^2$. This shows that *the area of the SRAM PUF with EER* $< 10^{-9}$ *is more than 3X larger than the area of a weak LEDPUF.* For the SRAM PUF with ERR $< 10^{-6}$, the area to generate 128 bits is about $604\mu m^2$, which is still about 1.4X larger than the area of a weak LEDPUF.

5.4 Evaluating the Security Level of Strong PUFs Through Guesswork

5.4.1 The Guesswork of any Strong PUF

In this subsection we quantify the security of strong PUFs in terms of the number of guesses required to break them. Our results quantify the number of secure authentications for which any strong PUF is good for. Furthermore, we compare the guesswork of our proposed strong LEDPUF to the guesswork of other strong PUFs that have been introduced in the literature. Finally, to demonstrate the importance of stability of a strong PUF, we show that the guesswork of a stable XOR arbiter PUF is larger than the guesswork of noisy ones, for the same number of observed CRPs.

We begin by defining the following game.

Definition 3. *Consider a strong PUF, which is used by an authentication scheme to authenticate n unique challenges through observing their responses. The authentication problem is defined as follows:*

1. *For each challenge the attacker has to guess with a single response.*
2. *When the attacker does not guess correctly, it can mask itself to receive a new challenge.*
3. *Once the attacker makes a correct guess it is authenticated.*

Remark 11. *Note that when authenticating a strong PUF through CRPs each challenge can be used only once. Furthermore, the problem defined above captures a strict security requirement that the system is compromised once the attacker manages to deceive the verifier.*

Remark 12. *When the attacker fails to guess any response correctly the attack fails. When the number of challenges is large we show that this event decays exponentially fast.*

We now find the average guesswork of the game presented in Definition 3 as well the probability that the number of guesses is smaller than or equal to a certain number.

Theorem 3. *The average guesswork of the authentication problem presented in Definition 3 is*

$$E(G) = 2^{-H_\infty(1)} + \sum_{i=2}^{n} i \cdot 2^{-H_\infty(i)} \cdot \prod_{k=1}^{i-1} \left(1 - 2^{-H_\infty(k)}\right) \tag{5.24}$$

where $2^{-H_\infty(k)}$ is the most probable response to the kth challenge either given that the guesses for challenges $1, \ldots, k-1$ were incorrect or given the previous $k-1$ CRPs (these two scenarios can lead to different min-entropy). Furthermore, the probability that the number of guesses is smaller than or equal to l is

$$Pr(G \in \{1, \ldots, l\}) = 2^{-H_\infty(1)} + \sum_{i=2}^{l} 2^{-H_\infty(i)} \prod_{j=1}^{i-1} \left(1 - 2^{-H_\infty(j)}\right)$$

Finally, the probability of attack failure is $\prod_{i=1}^{n} \left(1 - 2^{-H_\infty(i)}\right)$.

Proof. Each response has a certain statistical profile based on the previous CRPs. When the attacker knows this profile the optimal strategy to minimize the number of guesses is to guess the most probable one. This in turn leads to $2^{-H_\infty(i)}$ where $H_\infty(i)$ is the min-entropy given that either the previous $i-1$ guesses were not correct or that the previous $i-1$ CRPs were revealed to the attacker. The results of the theorem follow directly from this argument. □

Corollary 1. *When the statistical profile does not change across challenges we get that*

$$E(G) = 2^{H_\infty} - \left(1 - 2^{-H_\infty}\right)^n \cdot \left(n + 2^{H_\infty}\right) \tag{5.25}$$

and

$$Pr(G \in \{1, \ldots, l\}) = 1 - \left(1 - 2^{-H_\infty}\right)^l. \tag{5.26}$$

Remark 13. *Note that the average guesswork (5.25) is equal to*

$$E(G) = 2^{H_\infty} - \epsilon \tag{5.27}$$

when $n \gg 1$, where $\epsilon \ll 1$ decays exponentially fast.

Remark 14. *When the attacker does not know the statistical profile of the response (for instance when the structure is too complex for him to infer it), all he can do is to guess a response uniformly, which leads in turn to $H_\infty = 1$.*

Furthermore, in some cases the attacker can infer the statistical profile based on the structure of a strong PUF and a set of CRPs that have been revealed to him (see [1, 114] for attacks on arbiter PUFs, XOR arbiter PUFs, etc.).

5.4.2 Quantifying the Security of Specific Strong PUFs

In this subsection we quantify the security level of various strong PUFs in terms of their guesswork.

The result in (5.27) also applies to the case when an attacker can observe multiple CRPs. For example model attacks over strong PUFs [1, 114] enable attackers to accurately guess responses based on previously observed CRPs. In terms of guesswork it means that once an attacker observes a certain set of CRPs, conditioned on the CRPs that have been revealed so far, the most likely conditional probability can be very high.

The average guesswork in (5.27) allows us to quantify how secure strong LEDPUF is compared to other strong PUFs from the literature that are susceptible to model building attacks.

Strong LEDPUFs are based on HMAC, and so idealy an attacker can not infer anything from observing CRPs in this case. Therefore, when the number of bits at the output of a strong LEDPUF is m, the average guesswork converges to 2^m after observing any amount of CRPs. Even when the key is biased such that each bit is drawn Bernoulli(0.53) [44], the average guesswork when HMAC is a strongly universal set of hash functions [115], is $2^{-\log_2(0.53)\cdot m} = 2^{0.91\cdot m}$ based on (5.27).

On the other hand, in [1] it has been shown for various noise-free PUF simulations such as arbiter PUFs, XOR arbiter PUFs and Feed-Forward Arbiter PUFs that the prediction rate varies between 97% and 99%, after observing a few hundreds of thousands of CRPs and implementing a model building attack. Essentially, this type of attacks achieve a prediction rate, which is an estimation of the probability mass function of the next response, conditioned on a certain set of CRPs. In [1] it leads to conditional probability of at least $2^{\log_2(0.97)\cdot m} = 2^{-0.04\cdot m}$ when the prediction rate is 97%, and so the average guesswork under this model

building attack is achieved by assigning this probability to the average guesswork in (5.27) in which case we get $2^{0.04 \cdot m}$. Note that when the prediction rate is 99% the average guesswork is $2^{-\log_2(0.99) \cdot m} = 2^{0.014 \cdot m}$; therefore, when $m = 256$ we get that at 97% the PUF is $2^{0.026 \times 256} =$ 100 times more secure than at 99%.

Therefore, guesswork provides a unified framework for comparing the security level of different PUFs under model building attacks. It can also be used as a means of understanding what is the desired prediction rate for a model based attack, and as a result how many challenge response pairs should be observed.

Next, we compare the security of stable and noisy XOR arbiter PUFs [1] under model based attacks in terms of the number of guesses for which the probability of guessing the correct response is 99%. We use equation (5.26) to derive the results of this subsection.

The expression in (5.26) depends on the min-entropy, and so for noisy PUFs we need to incorporate the effect of the noise into the min-entropy. For this we use Lemma 1 in which the min-entropy is extended to the noisy case. In Table 5.5 we use guesswork to compare the security of stable XOR arbiter PUFs to the one of noisy XOR arbiter PUFs with the same number of XORs, under model based attacks. We assign the prediction rates and noise levels reported in [1] to (5.26), and find the number of guesses under model based attacks in which the verifier has to take into account the noise as the PUF owner observes noisy responses. The table shows how much more secure stable XOR arbiter PUFs are when compared to the noisy versions under model based attacks. Essentially, it shows how susceptible such arbiter-based strong PUF is after observing a certain number of CRPs. In fact, when the noise level is over 5%, the probability of guessing the correct response up to the noise level is very close to one (about $1 - 10^{-10}$), which means that this PUF is completely broken. This is because the prediction rate of each bit as reported in [1] is 97.34%, whereas the noise level is 5% and so the chance that the guessed response is not within the noise level of the original response is extremely small for reasonable values of m. Therefore, guesswork enables us to incorporate the effect of noise and model based attacks into one framework that allows comparison between different PUFs and various scenarios.

Table 5.5: The number of guesses for which the probability of guessing the correct response is lager than 99% when $m = 1024$, for a stable XOR arbiter PUF ($D = 0\%$), and noisy ones, under model based attacks for which 200 thousand and 500 thousand CRPs are observed. The values are based on the noise levels and prediction rates reported in [1]. The second row presents 4-XORs, whereas the third row presents 5-XORs.

CRPs ($\times 10^3$)	D=0%	D=2%	D=5%	D=10%
200	41	10	1	1
500	22	10	1	1

5.5 Conclusion

In this chapter we develop a unified guesswork-based analyses for PUFs. We show through guesswork analysis that stability has a more severe impact on the PUF security than biased responses. In addition, we analyze guesswork for two new problems: Guesswork under probability of attack failure, and the guesswork of strong PUFs.

CHAPTER 6

SLATE: A Secure Lightweight Entity Authentication Hardware Primitive

Several stable weak PUFs have been proposed as key storage primitives in recent years. In this chapter, we exploit these stable weak PUFs and propose a novel Secure Lightweight Entity Authentication hardware primitive called SLATE, where its secret key can be stored in a form of a weak PUF or any secure key storage. Even though the authentication of SLATE is done through Challenge Response Pair (CRP) verification similar to strong PUFs, SLATE itself is a pure digital structure without exploiting any process variation, therefore SLATE has no reliability concerns. Another main advantage of SLATE over most existing strong PUFs being an entity authentication primitive is that SLATE is resistant to known attacks to strong PUFs or logic obfuscations, such as model building attacks and Boolean Satisfiability (SAT) attacks. Furthermore, we show that the implementation cost of SLATE with a 176-bit key and 2^{44} CRPs is only 663 Gate Equivalents (GE). Compared to lightweight ciphers and existing secure strong PUFs, which is 44% to 7.1× larger than SLATE, we show that SLATE is a practical security primitive for resource constrained systems. In addition, it is shown that SLATE is information theoretically secure when valid CRPs are communicated through insecure channels. Finally, to ensure the unpredictability and the unclonability of SLATE when used with a weak PUF, we propose a novel tamper evident one-time-read method that guarantees the confidentially and the integrity of the extracted secrets from a stable weak PUF.

6.1 Introduction

Physical Unclonable Functions (PUFs) have been considered as promising security primitives that enables lightweight hardware implementations of identification [27], authentication [116], or secret key generation and storage [29]. The randomness of a PUF is extracted from random uncontrollable process variations, and its behavior, or Challenge Response Pair (CRP), is uniquely defined and is hard to predict or replicate. Recently, many PUFs designs focusing on the enhancement of stability have been proposed, including both weak PUFs and strong PUFs [98]. These stability-guaranteed techniques allow PUFs to be exploited in more practical applications. However, the advantages of stability-guaranteed PUFs have not yet been fully explored. A stable PUF can provide much more than just the hardware area/cost savings from Error Correction Code [54] or other stability enhancement techniques. In particular, in this chapter we demonstrate another use of stable PUFs by designing a Secure Lightweight Entity Authentication (SLATE) primitive, which is a pure digital structure without exploiting any process variation, therefore SLATE itself has no reliability concerns. SLATE can be constructed from a stable weak PUF or *any other secure key storage* based on the concept of a model-based PUF [29], where the behavior of a strong PUF can be calculated from a compact model, therefore no CRP storage is needed. Most importantly, SLATE is more secure and hardware efficient (3.1× to 7.1× smaller) than existing strong PUFs from both empirical and theoretical perspectives. SLATE is also much more hardware efficient (more than 40% smaller) than existing lightweight ciphers, which can also be used as entity authentication primitives.

6.1.1 PUF-Based Authentication Protocols

Many PUF-based authentication protocols have been proposed since the PUF was introduced [9]. The simple authentication scheme proposed in [2] employs the CRP mechanism of strong PUFs, but the protocol is not practical because the PUF itself suffers from modeling attacks [1]. Recently, several secure strong PUFs are proposed and shown to be resistant to model building attacks [117,118]. It has been shown that the use of XOR gates can effectively

increase the difficulty of model building attacks. However, due to the number of XOR gates or parallel structures required, the hardware implementation costs of these secure strong PUFs are even higher than existing lightweight block ciphers, which can also be used as entity authentication primitives. Using an Optical PUF as stated in [3] is believed to be resistant to modeling attacks, however it may not adhere to the low-cost design principle of a PUF application. Most of the existing PUF-based authentication protocols aim to compensate the vulnerability to modeling attacks. Unfortunately, these approaches often undermine the benefits provided by the PUF technology, such as the lightweight implementation or the replacement of costly secure data storage, making most existing PUF-based protocols impractical [119].

6.1.2 Stability-Guaranteed PUF

One of the major research directions of PUF is the enhancement of its stability, and extensive effort has been devoted to the area (see [120–122] and references therein). It has been shown that stable PUFs can provide practical solutions including logic obfuscation [56, 123] and hardware metering [124]. Despite of the attractive benefits of a stable PUF, however, many applications require a guaranteed one-time secret key extraction, but to the best of our knowledge, the detailed physical mechanism of such key extraction has not been explained. A common one-time-read approach is to read the secret through eFuse and burn out the connections after reading out the values [56]. The approach assumes that the PUF is in a secure environment before reaching to the verifier, but this assumption may not be true. For example. the attacker can *read out the secret without destroying the eFuse access to the secret at any point after the fabrication*, and no one would know that the PUF has already been compromised. Another approach is to use asymmetric cipher framework, which comes with high design and implementation cost. Therefore there is a need to develop a secure and efficient mechanism to extract the key for the stable weak PUF.

6.1.3 Model-based PUF

The concept of model-based PUF has been discussed in [29]. The advantages of a model-based PUF is that the verifier in an authentication protocol only stores a model of the PUF instead of storing a large CRP database. In [125], the delay signature of delay cells are extracted, so the verifier can emulate the response given the challenge. However, in addition to the complex delay characterization circuitry, the details of the one-time secret extraction mechanism is missing. In [126] the authors suggest that a strong PUF can be constructed from a weak PUF, however, the hardware cost of a digital Random Number Generator (RNG) or a stream cipher is commensurate to a cryptographic hash function [127, 128]. Similarly in [44], a strong Locally Enhanced Defectivity (LED)PUF is proposed by using a stable weak PUF as a key to a Hash-based Message Authentication Code (HMAC). However, the implementation cost of the strong LEDPUF is even higher than a block cipher due to the cryptographic hash function implementation. Therefore, a lightweight and secure authentication primitive is still yet to be discovered even when a stable weak PUF can be used as the source of the secret key.

6.1.4 Main Contributions

The contributions of this chapter include:

- A pure digital secure lightweight entity authentication primitive (SLATE) is proposed. We show that SLATE is 44% to 7.1× more hardware efficient than existing lightweight block/stream ciphers and secure strong PUFs.
- We show that SLATE is resistant to attacks that are effective to existing PUFs and digital logic obfuscations. Results of model building attacks show that the prediction rate is similar to random guessing, and the run time of Boolean Satisfiability (SAT) attack grows exponentially with the hardware size.
- From information theoretical perspective we show that SLATE is secure when the CRPs are communicated through an insecure channel.

- A physical tamper evident one-time-read secret extraction method is presented.

6.2 The Proposed Cascaded Architecture

In this section we first present the proposed cascaded architecture as the core design of the SLATE, and then we show that the cascaded architecture itself is resistant to model building attacks that are effective to existing strong PUFs. Following the model building attack results, we show that the cascaded structure is nevertheless not resistant to linear equation solving attack and single unit querying attack. To overcome these vulnerabilities, we present the complete secure SLATE structure in Section 6.3.1.

6.2.1 Cascaded Unit Structure

The schematic of the Unit in the cascaded structure is given in Figure 6.1. All inputs and outputs of the Unit are 2-bit values. If k is equal to b_1 or b_2, then k is a *match*. The output of the Compare module in the Unit is defined as:

$$\text{Compare output} = \begin{cases} k, & \text{if } k \text{ is a match} \\ r, & \text{otherwise} \end{cases} \tag{6.1}$$

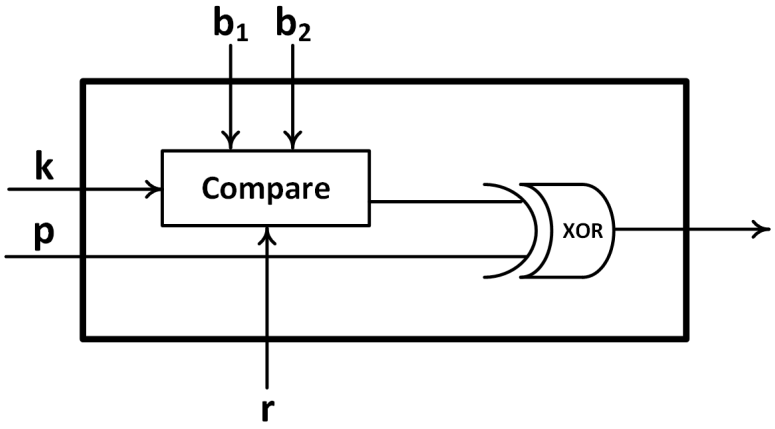


Figure 6.1: Schematic of a Unit in the cascaded structure.

The proposed model building attack resistant structure is composed of cascaded Units

as shown in Figure 6.2. b_{i1}, b_{i2} and r_i of the i^{th} Unit are secret values provided by a stable weak PUF or any secret key, and k_i is the input challenge of the i^{th} Unit. For the first Unit, the output is simply the output of the Compare module; for all other Units, the output is $k \oplus p$, where p is from the previous Unit.

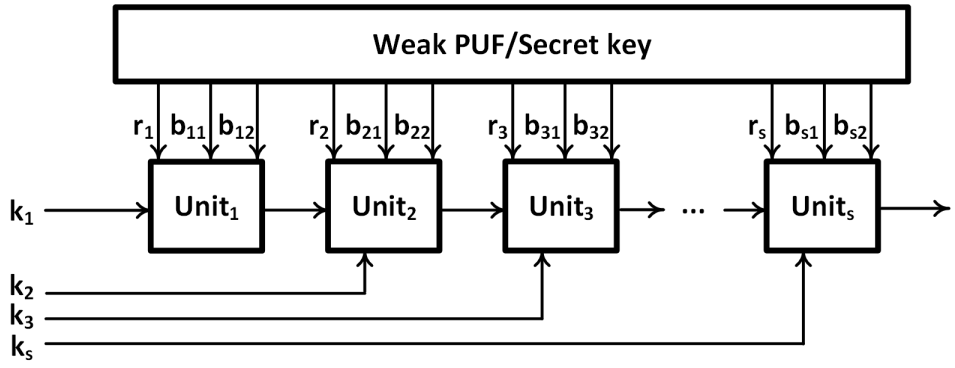


Figure 6.2: Schematic of a secure cascaded structure.

For the cascaded structure, the entity authentication is done through CRP validation similar to a strong PUF where the weak PUF values are obtained by the verifier during enrollment. The difference is that for the cascaded structure, a challenge is *valid* only when k_i is a match for all Units. During authentication, only valid CRPs that have valid challenges are deployed. If any one of the Unit is not a match, the output of the Unit becomes $r \oplus p$, which is a value that will not be used in any valid response. The only way to create a valid response for the attacker is to match every single Unit. For a cascaded structure with s Units and the length of b_1 and b_2 being 2-bit long, the probability of hitting a meaningful CRP is approximately $\frac{1}{2^s}$. Therefore, most CRPs collected by the attacker from the cascaded structure are not valid, making model building attacks ineffective. Different from [129], which is unstable and can be predicted with 80% accuracy, our proposed cascaded structure is pure digital without any parametric variations.

6.2.2 Machine Learning Attack

Many strong PUFs with cascaded structure, such as Arbiter PUF [10], Feed Forward PUF [20], Lightweight secure PUF [130], or XOR-Arbiter PUF [10, 29], have been proven to be

Table 6.1: Prediction rate of each of the output bit of a 22-stage cascaded Unit structure using LR and SVM attacks. A value close to 50% indicates that the prediction rate is similar to random guessing. Increasing the number of training CRPs does not improve the rate. These values show that these attacks are as effective as random guessing only.

Training CRPs	200	500	2,000	10,000	100,000
(LR) First Bit	50.0%	48.4%	49.1%	47.6%	49.4%
(LR) Second Bit	50.7%	50.0%	48.0%	49.9%	49.2%
(SVM) First Bit	48.3%	50.9%	50.3%	49.6%	51.1%
(SVM) Second Bit	48.2%	49.2%	46.8%	49.5%	50.5%

vulnerable to machine learning attacks [1, 131]. In [114], the authors even successfully break a commercial XOR PUF. In our model building attack model, we assume that the attacker has physical access to the cascaded structure and can collect as many CRPs as possible for training. We first try using Logistic Regression (LR) with sigmoid function [1] and Support Vector Machine (SVM) with Radial Basis kernel function to predict 1,000 unseen responses of a simulated 22-stage cascaded structure, assuming that the key bits from the weak PUF are random. The input features of the attacks are k_i for all 22 stages (44 bits) and the training CRPs are generated randomly. Table 6.1 shows results of the prediction rate, which is defined as the probability of a correct prediction. We can see that the values are close to 50% for both attacks, which means that the prediction is ineffective and is similar to random guessing only, and there is no correlation between the prediction rate and the number of training CRPs.

Next, we try to predict 1,000 unseen responses of the cascaded structure with a Multilayer Perceptron Neural Network (NN) using TensorFlow [132], which is another effective attack to existing PUFs [131]. Various numbers of hidden layers and units are tested, and the results are similar as shown in Figure 6.3, which is with 3 hidden layers, each has 128 units. For all training sizes (1,000 CRPs, 5,000 CRPs, and 100,000 CRPs) shown in the figure, the prediction rates of each of the response bit fall between 51% and 49% for all 10 attack trials

(attempts), which means that the attack is not better than random guessing. Also, using a larger training set does not improve the prediction rate.

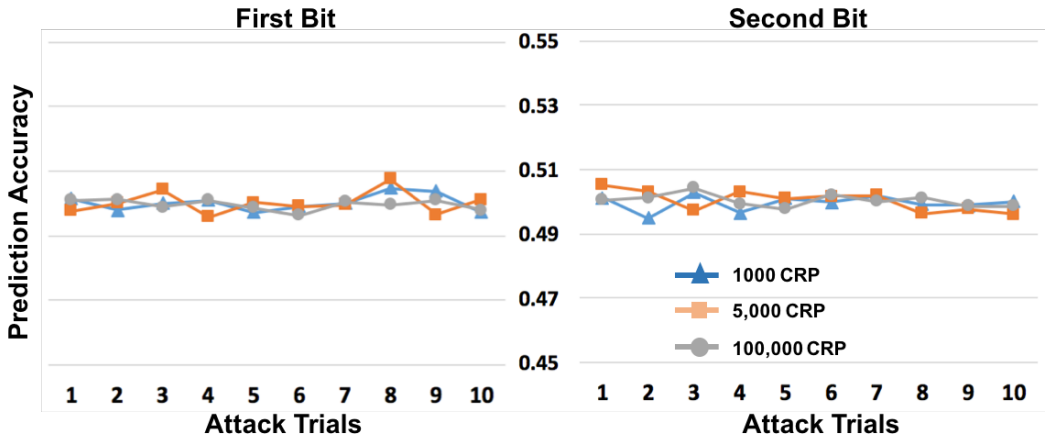


Figure 6.3: NN attack results of 10 trials with different numbers of training CRPs. Prediction rates of both output bits are close to 50%, indicating that the attack is ineffective.

These results show that the proposed cascaded structure is resistant to machine learning attacks because it is difficult for the attacker to obtain valid or *meaningful* CRPs for the learning procedure. A valid challenge must match either b_1 or b_2 for all Units in order to generate a valid response; otherwise, the response is not a valid response and therefore does not provide useful information to predict the response of an unseen valid challenge.

6.2.3 Linear Equation Solving Attack

In this section we show that the cascaded structure is vulnerable to linear equation solving attack if the attacker can observe many valid CRPs and has no physical access to the cascaded structure. Let s be the number of stages in the cascaded structure. Define l be the number of valid CRPs observed by the attacker. By comparing two valid challenges, the attacker knows whether the match of each Unit is changed or not. For example, when comparing 2 valid challenges 110101 and 110001 of a 3-stage cascaded structure, the attacker knows that the match of the second Unit is changed from 01 to 00, say from b_{21} to b_{22}. Let the match of the first challenge of the i^{th} Unit be b_{i1} for all i, and the responses of the first

and second challenges be y_1 and y_2, respectively. The attacker can now construct a matrix multiplication formula $K_{valid} \cdot X = Y_{valid}$ from the two CRPs observed as:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \\ b_{31} \\ b_{32} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

, where K_{valid} is a $l \times 2s$ matrix representing the selection of matches, X is a $2s \times 2$ matrix representing b_{i1} and b_{i2} of all Units, and Y_{valid} is a $l \times 2$ matrix representing the 2-bit responses. The addition operation in the matrix multiplication is a GF(2) operation, which is essentially the XOR operation denoted as $\oplus$.

Whenever the attacker observes a new valid CRP, the number of rows in K_{valid} and Y_{valid} can be incremented by 1 following the same strategy. If adding the new row to K_{valid} does not increase the rank of K_{valid}, it means that the new CRP is a linear combination of observed CRPs, which indicates that the new response can be predicted. If adding the new row to K_{valid} increases its rank, the new CRP cannot be predicted. However, since K_{valid} has only $2s$ columns, the rank of it cannot exceed $2s$, therefore the number of CRPs that cannot be predicted is at most $2s$. In other words, during the life time of the cascaded structure, the number of secure valid CRP is at most $2s$, which is only linear to the number of stages.

6.2.4 Single Unit Querying Attack

In this section we show that the cascaded structure is vulnerable to single unit querying attack if (1) the attacker can only observe limited valid CRPs and (2) has physical access to the cascaded structure. Assume that the attacker observes a valid challenge $K = (k_1, k_2, \cdots, k_i, \cdots, k_s)$ and the response Y. For any i^{th} Unit, it is now known to the attacker that k_i is equal to either b_{i1} or b_{i2}. The attacker can then do the following steps:

1. Assume that k_i is equal to b_{i1} for all Units, the response can be represented as $Y = b_{11} \oplus b_{21} \oplus \cdots \oplus b_{i1} \oplus \cdots \oplus b_{s1}$.

2. For the i^{th} Unit, since k_i is a 2-bit value, the attacker can query all other 3 possible input values k_{i1}, k_{i2}, and k_{i3} to observe the 3 responses Y_1, Y_2, and Y_3.

3. At least 2 of the responses Y_1, Y_2, and Y_3 will be the same because 2 of the k_{i1}, k_{i2}, and k_{i3} correspond to non-match inputs, where the output of the module is r_i.

4. Let Y_1 and Y_2 be the same responses, a non-valid response $Y_1 = Y_2 = b_{11} \oplus b_{21} \oplus \cdots \oplus r_i \oplus \cdots \oplus b_{s1}$ is obtained, and an unseen valid response $Y_3 = b_{11} \oplus b_{21} \oplus \cdots \oplus b_{i2} \oplus \cdots \oplus b_{s1}$ with challenge $(k_1, k_2, \cdots, k_{i3}, \cdots, k_s)$ corresponds to the matched b_{i2} is obtained.

In fact, the attacker can further obtain $b_{i1} \oplus b_{i2}$ for the i^{th} Unit by performing $Y \oplus Y_3$. The information can be further exploited to calculate more unseen valid CRPs.

To prevent the linear equation solving attack, the matrix K_{valid} must be hidden to the attacker. In other words, when any 2 valid challenges are observed by the attacker, the selection of the matched b_1 or b_2 of any Unit should not be revealed, therefore the attacker cannot construct the matrix K_{valid}. To prevent the single unit querying attack, the cascaded structure needs to cause confusion to the attacker when different k_{i1}, k_{i2}, and k_{i3} are applied to the i^{th} Unit. The attacker should not be able to figure out which one is a real match from the responses. We propose SLATE architecture in the following section as a cascaded structure that is resistant to all aforementioned attacks.

6.3 The Proposed SLATE Architecture

6.3.1 Secure SLATE Structure

The secure SLATE is composed of cascaded S-Units as shown in Figure 6.4. The Compare module is the same as defined in Section 6.2.1. However for a S-Unit, $g = k \oplus a$ is the input to the Compare module.

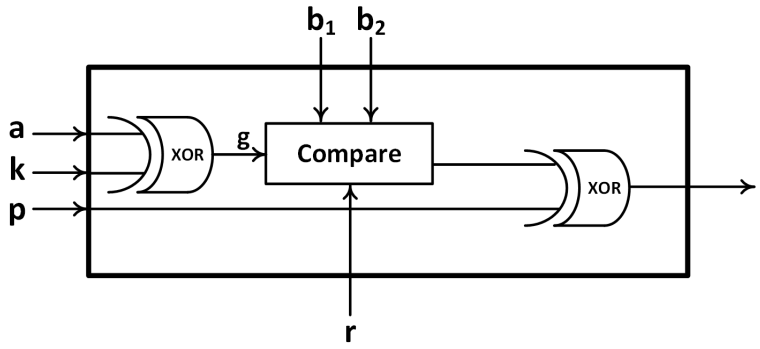


Figure 6.4: Schematic of the S-Unit for SLATE.

The complete SLATE structure with s S-Units is given in Figure 6.5. Define $A = (a_1, a_2, \cdots, a_s)$ and $R = (r_1, r_2, \cdots, r_s)$ the outputs of the two Linear Feedback Shift Registers (LFSRs). $G = (g_1, g_2, \cdots, g_s)$ is calculated from $K \oplus A$. The initial state of the first LFSR is determined by (N,W_1), and the initial state of the second LFSR is determined by (K,W_2), where W_1 and W_2 are secret bits of length N obtained directly from the secrete bits. After the initial bits are loaded, the LFSRs are "warmed-up" by running a fixed F cycles to randomize the outputs of LFSRs sufficiently and to make sure that the outputs are depending on both C and the weak PUF. A and R, which depend on (N,W_1) and (K,W_2), respectively, are the inputs to the cascaded S-Units. For each of the i^{th} S-Unit, k_i is XORed with a_i to get g_i. If g_i is a match, then $g_i \oplus p_i$ is propagated to the next S-Unit; otherwise, $r_i \oplus p_i$, which will not exists in any valid challenge, is propagated to the next S-Unit. The input, or challenge, to the SLATE is $C = (K, N)$, and the response of the SLATE is the output of the last S-Unit.

6.3.2 Authentication Protocol

The operation of the SLATE is similar to a model-based strong PUF except for that the model of the SLATE is extracted from the associated weak PUF or the secret key. The enrollment and authentication phases are described as follows:

Enrollment phase: The verifier obtains the secret key or extracts the key from the weak PUF using the one-time-read method presented in Section 6.5. No CRP collection is

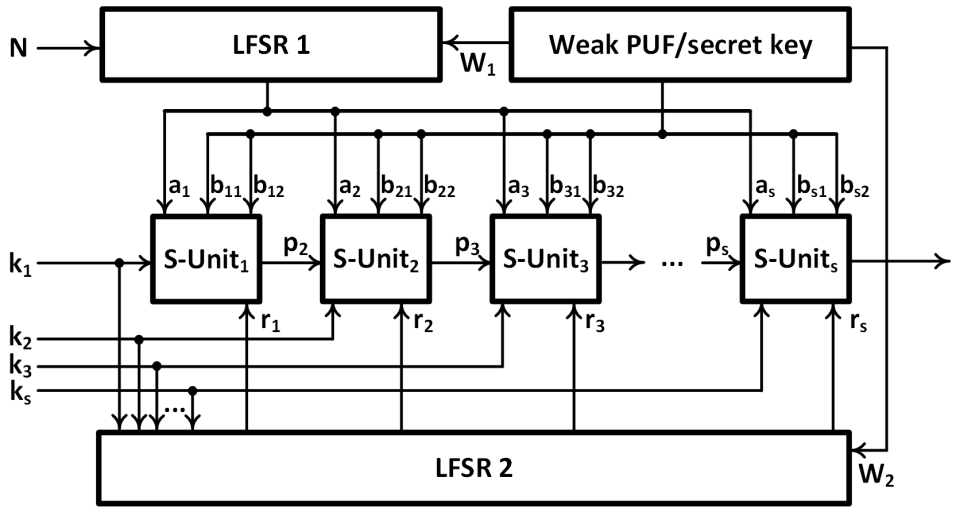


Figure 6.5: Structure of the proposed cascaded SLATE.

required.

Authentication phase:

1. The verifier generates a *valid* challenge $C = (K, N)$, which is a challenge that matches either b_1 or b_2 for all S-Units of the SLATE module.

2. The first LFSR and second LFSR take (N,W_1) and (K,W_2) as initial seeds, respectively. Both LFSRs run a "warm-up" phase for a fixed number F of clock cycles to generate inputs of the S-Unit.

3. The final response generated from the cascaded S-Units is sent to the verifier.

4. The verifier calculates the corresponding response of C and examines the response from the SLATE. If the response is correct, the entity is authenticated; otherwise, the entity is not authenticated.

A valid challenge can be calculated easily by the verifier because the verifier knows W_1, W_2, b_1 and b_2 of all S-Units. Once a N is decided, the verifier calculates the states of the LFSRs after F cycles and find K that matches all S-Units. Since every N is used only once to prevent the linear equation solving attack, the number of valid CRPs is 2^N.

The SLATE structure used in the proposed authentication protocol is resistant to model building attack because its underlying cascaded structure is secure. Also, it is resistant to linear equation solving attack because when comparing 2 valid challenges, the attacker cannot figure out which one of the b_1 and b_2 is the match as the output of the first LFSR changes for every valid challenge. Therefore the attacker cannot construct K_{valid} given that a new N is used for every new valid challenge. It is resistant to single S-Unit querying attack because even if the attacker can fix N and try different K, the r of each S-Unit also changes because of the second LFSR, therefore the attacker will not be able to distinguish the r from a match.

6.3.3 Boolean Satisfiability (SAT) Attack

In this section we show that the SLATE is resistant to SAT attack proposed in [133], which has been shown to be an effective attack to retrieve the correct key of many logic obfuscation schemes. The SAT attack algorithm allows an attacker to decipher an obfuscated circuit using a small number of carefully selected input patterns and their corresponding outputs (distinguishing input/output pairs or DIPs) which can be observed from an activated functional chip. The algorithm finds such DIPs ($\boldsymbol{X}^d/\boldsymbol{Y}^d$) iteratively and formalizes them as a sequence of SAT formulas which can be solved by SAT solver. Each DIP rules out a subset of wrong key combinations and the algorithm terminates when all wrong keys have been removed. The SAT attack algorithm guarantees to find the equivalent class of the correct key upon the termination [134].

The objective of an attacker is to obtain the correct key of the obfuscated circuit. In the case of SLATE, the attacker wishes to find the value of $\boldsymbol{B} = (b_{11}, b_{12}, b_{21}, b_{22}, ..., b_{s1}, b_{s2})$, and $\boldsymbol{W} = (W_1, W_2)$. We hereby note these key values as $\boldsymbol{WB}$. The attack model assumes that the attacker has access to the following two components:

1. The gate-level netlist of the SLATE, which can be represented as $\boldsymbol{Y} = f(\boldsymbol{X}, \boldsymbol{WB})$, where $\boldsymbol{X} = (N, K)$ is the primary input and $\boldsymbol{Y}$ is the primary output of the circuit. The SAT formula of the netlist in conjunction normal form (CNF) is represented as

$C(\boldsymbol{X},\ \boldsymbol{WB},\ \boldsymbol{Y})$.

2. The physical victim SLATE module, which is used to observe the correct output given an input $\boldsymbol{Y} = eval(\boldsymbol{X})$.

Algorithm 1 SAT Attack Algorithm [133]

Input: CNF and *eval*

Output: $\boldsymbol{WB}_C$

1: $i = 1$;
2: $G_i = True$;
3: $F_i = C(\boldsymbol{X},\ \boldsymbol{WB}_1,\ \boldsymbol{Y}_1) \wedge C(\boldsymbol{X},\ \boldsymbol{WB}_2,\ \boldsymbol{Y}_2) \wedge (\boldsymbol{Y}_1 \neq \boldsymbol{Y}_2)$;
4: **while** $sat[F_i]$ **do**
5: $\quad \boldsymbol{X}_i^d = sat\_assignment_X[F_i]$;
6: $\quad \boldsymbol{Y}_i^d = eval(\boldsymbol{X}_i^d)$;
7: $\quad G_{i+1} = G_i \wedge C(\boldsymbol{X}_i^d,\ \boldsymbol{WB},\ \boldsymbol{Y}_i^d)$;
8: $\quad F_{i+1} = F_i \wedge C(\boldsymbol{X}_i^d,\ \boldsymbol{WB}_1,\ \boldsymbol{Y}_i^d) \wedge C(\boldsymbol{X}_i^d,\ \boldsymbol{WB}_2,\ \boldsymbol{Y}_i^d)$;
9: $\quad i = i + 1$;
10: **end while**
11: $\boldsymbol{WB}_C = sat\_assignment_{\mathbf{WB}}(G_i)$;

When modeling SLATE into CNF form, we first expand both LFSRs in Figure 6.5 into XOR networks. The goal of the attacker is to infer the fixed secret key of $\boldsymbol{B} = (b_{11}, b_{12}, b_{21}, b_{22}, ..., b_{s1}, b_{s2})$ and $\boldsymbol{W} = (W_1, W_2)$ using the SAT attack algorithm shown in Algorithm 1 to attack SLATE.

We apply SAT attack on SLATE with different number of stages with scaled key size to evaluate the effectiveness of the SAT attack. The execution time required to solve the correct key is presented in Table 6.2. We use the SAT Solver developed in [135] and implement the attack in C++. The attack is run on a Quad Intel Xeon 2.8GHz CPU server, and the run time limit for the attack is set to 10 hours. The results show that the execution time of SAT attack algorithm grows exponentially with the number of SLATE stages. For SLATE with

Table 6.2: SAT attack on SLATE with different number of S-Units. The required time grows exponentially with the number of S-Units.

S-Units	6	8	10	12	14	22
Time (s)	36	228	1,125	27,237	N/A	N/A

Table 6.3: Bit specifications of a variety of SLATE implementations. Last column shows the number of valid CRPs.

	$N(K)$	$W_1(W_2)$	LFSR	Key	CRP
176-bit key SLATE	44	44	44	176	2^{44}
128-bit key SLATE	32	32	32	128	2^{32}
104-bit key SLATE	26	26	26	104	2^{26}

14 or more S-Units, the SAT attack fails to find the correct key within the 10-hour time limit. These results indicate that SLATE is resistant to the SAT attack.

6.3.4 Hardware Implementation

The hardware implementation of SLATE shown in Fig. 6.5 is not unique but rather depending on the choice of trade-off between area and the number of CRPs. The main deciding factor is the selection of LFSRs. A maximum-length N-bit LFSR along with W_1 provides 2^N different A's to mask input K, therefore the number of valid CRPs is 2^N. Since the length of b_1 and b_2 are 2-bit each, a N-bit LFSR after each warm-up can support $\frac{N}{2}$ units, which determines the probability of a CRP being valid. Since each units requires 2×2-bit keys and the length of W_1 and W_2 are both N, the total length of the key is $4N$. Table 6.3 shows three version of possible SLATE implementations.

In this section we present the detailed hardware implementation cost of the 176-bit key SLATE with 44-bit LFSR and 22-stages, where the probability of a CRP being valid is $\frac{1}{2^{22}}$ for the number of stages. The Finite State Machine (FSM) controller module takes N, K, W_1, and W_2 to initialize the LFSRs. Since the values of the weak PUF or secret key are not

directly accessed but one-time padded before the real matching, the attacker learns nothing about the secret as proven in Section 6.4. In addition, SLATE itself is completely stable.

The SLATE design is implemented in Verilog and synthesized using a commercial 65nm CMOS technology using Cadence Genus. We use Gate Equivalents (GE) for area evaluation, where one GE is equivalent to the area of a NAND2 gate with the lowest driving strength of the corresponding technology. The SLATE implementation with minimized area is presented in Figure 6.6. Instead of implementing all 22 S-Units, a 2-bit register is added to the S-Unit and the cascaded structure is implemented as a sequential loop. The FSM controller reads K, N, and sends the address to the key storage to request the key one bit per clock cycle. A counter in the FSM controller is used to start the initialization (warm-up) of LFSRs. After the initialization, b_1 and b_2 are requested by the FSM controller and the evaluation of the S-Unit starts.

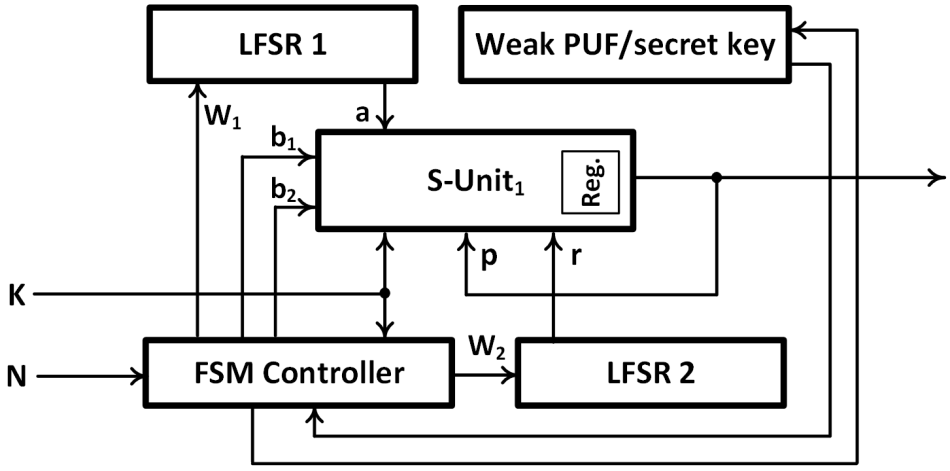


Figure 6.6: Compact SLATE implementation.

The LFSR implemented is a 44-bit maximum-length internal LFSR with primitive polynomial $x^{44} + x^6 + x^5 + x^2 + 1$ [136] as shown in Figure 6.7. The initial bits of the LFSR is loaded one bit per clock cycle from the feedback XOR loop. Once all bits are loaded, the LFSR runs $4 \times 44 = 176$ cycles in the warm-up phase before evaluating the S-Unit.

The implementation cost of the S-Unit including the 2-bit register is 33.75 GE. It takes 205.5 GE to implement the 44-bit LFSR using pules-latch structure [137]. The FSM controller would require 218.25 GE. Therefore the total GE required to implement the 176-bit

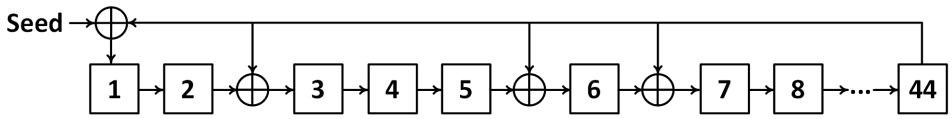


Figure 6.7: 44-bit LFSR with polynomial $x^{44} + x^6 + x^5 + x^2 + 1$

Table 6.4: GE comparisons between 176-bit SLATE and secure strong PUFs with 2^{44} CRPs.

PUF	GE	Ratio to SLATE
60-Stage Secure XOR PUF [117]	3,369	3.10
Strong LEDPUF [44]	3,517	3.23
44 x 44 LRR-DPUF [118]	7,744	7.11
SLATE + OTR weak LEDPUF	1,088	1.00

key SLATE is $33.75 + 205.5 \times 2 + 218.25 = 663$ GE, which is about 955 μm^2 for 65nm technology. If a 176-bit weak LEDPUF [44] is used as the key storage for SLATE, the cost of the PUF and the one-time-read (OTR) implementation in Section 6.5 is about 425 GE, including an AntiFuse (AF), a current comparator, an address decoder at the weak PUF output and an eFuse cell connected to the output of the decoder.

Table 6.4 shows the comparisons between stable and secure strong PUFs and SLATE with a OTR 176-bit weak LEDPUF. In [117] the authors suggest that a XOR PUF is secure when no less than 10 delay-based strong PUFs are XORed in parallel. However, only 0.0028% of the CRPs are stable after more than 10 XORs, therefore, to provide 2^{44} CRPs, 60-stage delay-based PUFs are required, and its area is about 3.1× of the SLATE. In [44] the authors proposed to use a cryptographic hash function with a weak PUF to construct a strong PUF. However, the combined area of a lightweight SHA-3 [138] and the weak PUF decoder is more than 3.2× of the SLATE. Another secure strong LRR-DPUF uses XOR network to generate secure CRPs [118]. To provide 2^{44} CRPs, a 44 x 44 LRR-DPUF is required, and the area is 7.1× of the SLATE. For a LRR-DPUF with similar GE as SLATE, the number of CRPs is approximately 2^{17}, which is too small and can be broken by simply collecting all possible CRPs.

Table 6.5: Area (GE) comparisons between SLATE and lightweight ciphers with 128-bit key. The areas shown do not include the key storage.

Cipher	GE	GE Ratio to 176 / 128 / 108 SLATE
* PRESENT [139]	1,570	2.37 / 2.85 / 3.18
SIMON [140]	958	1.44 / 1.74 / 1.94
SPECK [140]	996	1.50 / 1.81 / 2.02
PICOLLO [141]	1,362	2.05 / 2.48 / 2.76
Grain [128]	1,857	2.80 / 3.38 / 3.76
* Trivium [128]	2,599	3.92 / 4.73 / 5.26
176-bit key SLATE	663	1.00 / 1.21 / 1.34
128-bit key SLATE	550	0.83 / 1.00 / 1.11
104-bit key SLATE	494	0.75 / 0.90 / 1.00

* 80-bit key cipher

Since entity authentication can also be accomplished by ciphers, in Table 6.5 we compare the implementation costs of three versions of SLATE to existing lightweight block ciphers and stream ciphers excluding the key storage. From the table we can see that the hardware cost in GE of existing ciphers are 44% to 5.26× larger than SLATE. These results show that for the purpose of entity authentication, SLATE is a much more compact primitive than existing solutions.

6.4 Theoretical Guarantees

In this section we show that under idealized assumption SLATE is secure from information theoretical perspective [115] when the attacker can observe valid CRPs. We begin by describing a random matrix multiplication problem, we then show that it is information theoretically secure, and then connect it to the scheme of SLATE.

First let us define the following problem of random matrix multiplication.

Definition 4. *Consider the following set of equations*

$$Y = G \cdot X \tag{6.2}$$

where Y consists of l rows and two columns, G is a $l \times s$ matrix, and X has s rows and two columns. Furthermore, each entry of G as well as the entries of X are independent and identically distributed (i.i.d.) Bernoulli$(1/2)$*. Finally, $\mathcal{B}$ represents the set of invertible matrices with s rows and two columns.*

The following lemma shows that the mutual information [99] between X and Y is equal to zero as long as X is invertible, and also shows that the probability that X is not full rank decreases exponentially with s.

Lemma 2. *When $X \in \mathcal{B}$ the mutual information between X and Y is equal to zero, that is,*

$$I\left(X; Y | X \in \mathcal{B}\right) = 0. \tag{6.3}$$

Furthermore, $P\left(X \notin B\right) \leq 3 \cdot 2^{-s}$.

Proof. All entries of G are i.i.d. Bernoulli$(1/2)$. When X is invertible it means that its columns are different and both are not equal to zero, and so the entries of Y are also i.i.d. Bernoulli$(1/2)$. Therefore, the entropy of Y and $X|Y$ is the same and so the mutual information in this case is zero. In the case when X is not full rank, the mutual information between X and Y is no longer zero, and this happens when either the columns are equal or one of them is equal to zero. The probability of this event is upper bounded by $3 \cdot 2^{-s}$, that is, $P\left(X \notin \mathcal{B}\right) \leq 3 \cdot 2^{-s}$. □

Theorem 4. *Assume that G is one-time-padded such that $K = G \oplus A$ is observed, where K corresponds to the partial challenge and A corresponds to the first LFSR output of the SLATE (i.e., l responses as those are matrices with l rows). The matrix X represents the secret values of all S-Units, that is, b_1 and b_2, and Y represents l observed responses of SLATE. When both K and A are drawn i.i.d. Bernoulli*$(1/2)$*, the amount of information which is exposed when observing K along with l responses (Y), is zero with probability which is upper bounded by $3 \cdot 2^{-s}$, where the number of S-Units is equal to s. Therefore, as s increases this scheme becomes information theoretically secure.*

Proof. For simplicity let us assume that the possible values in the S-Units are either zero or two random bits that are drawn i.i.d. Bernoulli(1/2). This simplified assumption also holds when the challenge chooses between two pairs of random bits. As long as G is XORed with A, it can be assumed that no information on the matrix G is exposed when observing K (i.e., the mutual information between G and the K is equal to zero) and so the attacker does not know the matrix G in equation (6.2). In this case, based on Lemma 2 we get that the amount of information, which is revealed when observing valid values of K along with l responses (Y), is equal to zero as long as X is invertible, and that the probability that this is not the case is upper bounded by $3 \cdot 2^{-s}$. This proves the theorem. □

Remark 15. *When the number of S-Units in the scheme $s = 22$ we get that the probability that the mutual information is not equal to zero is upper bounded by $3 \cdot 2^{-22}$.*

Remark 16. *In practice a challenge is XORed with the output of the first LFSR of the scheme presented in Section 6.3.1 and not with a "pure" source of randomness.*

Remark 17. *It is assumed that the attacker observes both Y and $K = G \oplus A$, where A is i.i.d. Bernoulli(1/2). Therefore, one may be interested in $I(X; G \oplus A, Y)$. However, in our setting $I(X; G \oplus A, Y) = I(X; Y)$. This is because $I(X; G \oplus A|Y) = 0$ as Y, X and G are statistically independent of A, which is drawn i.i.d. Bernoulli(1/2). This is the reason why in theorem 4 we focus on $I(X; Y)$.*

6.5 Tamper Evident One-Time-Read Method

We propose a tamper evident one-time-read method for the practical use of SLATE but not limited to it. The method can be applied to any applications where one-time access to sensitive data is required.

In the method both eFuse and AF cells are used. The implementation of an One-Time-Programmable (OTP) module is given in Figure 6.8 (a). It is composed of an AF cell and a common current comparator as mentioned in the AF memory design in [142]. Since the current difference of AF between default (open) and programmed (close) states are larger

than 100X, when one (VDD) is applied to the AF, the comparator can effectively distinguish the two states to generate different outputs as demonstrated in [142] with AF cells fabricated using 0.18μm technology. When applied with one, the current of a default AF is less than $100pA$, and for a programmed AF, the current is larger than $20\mu A$. Therefore, with $2\mu A$ reference current the comparator can distinguish and generate different outputs accordingly. As shown in Figure 6.8 (b), the output of the OTP is used to activate the weak PUF, such as the enable signal or the power gating signal to any PUF modules. Once the AF cell is permanently programmed, the weak PUF, which provides secret values to the proposed SLATE module, can be evaluated through eFuse connections. The steps of the method is given in the following:

Tamper Evident One-Time-Read Method:

1. Set_1 is set to one (VDD) and the outputs of the eFuse connections are evaluated. If the values of weak PUF can be read, it means that the AF connections have been programmed to the closed states, which implies that an unexpected read is detected and the PUF should be discarded. If the output is floating, continue to step 2.
2. S_1 programs the AF cell to the closed state.
3. OTP activates the weak PUF and the secrets are read through eFuse connections.
4. Set_2 programs the eFuse connections to the open states after the reading.

By examining the states of the AF connections in step 1, the PUF user knows the reading history of the weak PUF and can discard the PUFs that have been read before. Therefore, the proposed method is tamper evident. The method also guarantees one-time-read because the eFuse connections are programmed to open states in step 4, where the read channel is destroyed permanently. The AF cell cannot be replaced by eFuse because otherwise the default states of eFuse is closed and an attacker does not have to program the eFuse to

activate the weak PUF. The eFuse connections cannot be replaced by AF cells either because once the AF cell is programmed, it cannot be reversed. Please note that the SLATE module in Figure 6.8 (b) can be any applications that reads secret values from a weak PUF.

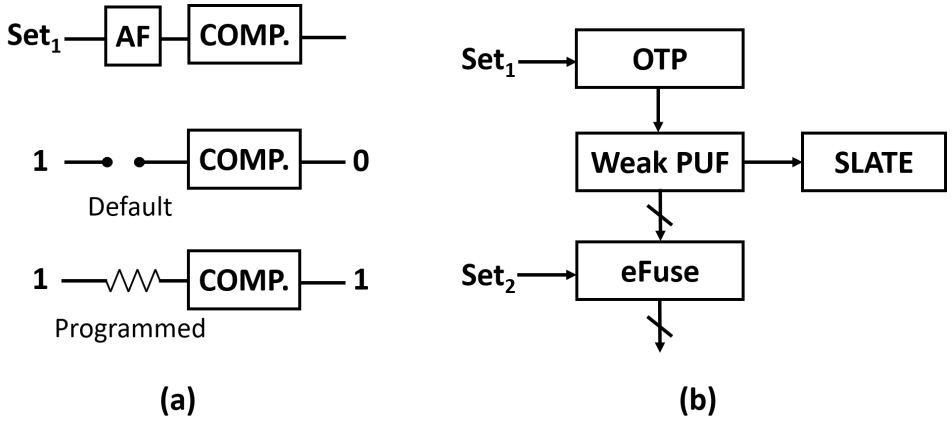


Figure 6.8: (a) OTP module implementation. An AF cell is connected to a current comparator to generate different outputs. (b) One-time-read secret extraction. The OTP activates the weak PUF after the AF cell is irreversibly programmed. Once the weak PUF is read, the eFuse connections is burnt so only the SLATE module has access to the weak PUF.

6.6 Conclusions

In this chapter we propose SLATE as a lightweight and secure entity authentication primitive. We show that SLATE is resistant to model building attacks and SAT attacks that are known to be effective to most existing strong PUFs or logic obfuscation. We compare the hardware implementation area of SLATE to secure strong PUFs and ciphers to show that existing entity authentication solutions are at least 44% to 7.1× larger than SLATE. Also, we show that ideally SLATE is information theoretically secure in terms of the amount of information revealed by observing valid CRPs. Finally, we propose a tamper evident one-time-read method to ensure the unpredictability and the unclonability of SLATE.

CHAPTER 7

Reverse Engineering of 2.5D Split Manufactured ICs

Integrated circuit (IC) split manufacturing has been shown to be one of the most effective protection schemes to prevent reverse engineering from malicious foundries. Among the existing split manufacturing approaches, the 2.5D split manufacturing using silicon interposer has much less fabrication and testing costs compared to layer splitting approaches. In this chapter we propose a Boolean Satisfiability(SAT)-based attack to reconstruct the wire connections of the 2.5D split manufacturing netlists. Unlike previous attacks to split manufacturing that do not guarantee 100% accuracy of the connections, such as the Proximity attack or the Network-flow attack, our SAT-based attack can fully reconstruct the missing wires and therefore the functionality of the chip can be completely reverse engineered. In addition, we show that the runtime of SAT attack is significantly reduced by applying grouping hints obtained from a Satisfiability Modulo Theories (SMT)-based grouping algorithm, which is purely depending on the circuit functionality, so no physical defensive mechanisms can prevent such attack. In our experiments we show that our grouping algorithm can speed up the SAT attack runtime by more than 1,000X and can successfully reverse engineer reasonable size benchmarks even when the split nets contains more than one fanouts and the total cut size is close to 1,000.

7.1 Introduction

The globalization of Integrated circuit (IC) supply chain due to the higher fabrication cost and increasing complexity of modern designs has led to new security threats including Trojan insertion [143], IC overproduction [144], intellectual property (IP) piracy [145] and reverse

engineering [146]. In the cost-effective fabless model, the foundries that the IC/IP owner outsourced the design to might not be trustworthy. Once the foundry obtains the whole GDSII of the design, it can overproduce or perform reverse engineering to obtain all design details, which leads to significant revenue lost. Split manufacturing, as one of the most promising defensive mechanisms to prevent foundry reverse engineering, has been studied intensively in recent years.

7.1.1 Layer-based Split Manufacturing

In order to protect IC owners from the malicious foundries or attackers, Layer-based Split Manufacturing (LSM) has been proposed as a protecting mechanism to minimize the aforementioned risks [147]. LSM divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) parts, and different parts are fabricated at different foundries. The FEOL (higher complexity and cost) part is fabricated at an untrusted foundry. Since the complete connections of the circuit are unknown to the untrusted foundry, the design cannot be fully reverse engineered. After the FEOL fabrication, the wafer is shipped back to an onshore trusted foundry for the BEOL fabrication and integration. While LSM may fit well with the advanced 3D IC fabrication model, however, the yield loss due to wafer transportation, integration, and the requirement of design rule compatibility of two foundries are still remaining as the major challenges [148]. Also, the cost of splitting lower metal layers can induce even higher cost [149], while splitting at higher layers may not provide sufficient security [150].

7.1.2 Module-based Split Manufacturing

Another split manufacturing strategy is the Module-based Split Manufacturing (MSM), where the design is split into different modules, and all layers including FEOL and BEOL of a module are fabricated at a same foundry. The modules are then sent back to a trusted integrator for final integration.

The MSM can be considered as a security-purpose 2.5D integration [151], which has been a promising IC integration technology that is designed to improve system performance

by using silicon interposers [152] offering a high density package system with low cost and performance benefits [153], such as inter-chip bandwidth and power reduction [154]. While large scale 3D ICs are still being developed, 2.5D products are already commercially available [155].

Compared to LSM, the advantages of MSM include:

1. Better yield because less transportation and alignment risks, and each module is packaged and tested as normal chip before being sent to the integrator.

2. No design rule compatibility requirements since the connection is done through chip-to-chip interposers.

In this chapter we focus on reconstructing the missing connections of the MSM strategy from an adversary's perspective.

7.1.3 Attacking Model

The adversary who tries to obtain the complete GDSII of the circuit is assumed to have access to the split manufactured parts of the design and can obtain the complete design from open market to get correct outputs of arbitrary inputs of the IC. Since the splitting parts are known to the adversary, the intermediate input/output at the splitting interface can also be obtained by the adversary. The only thing the adversary does not know is the connection between the splitting parts.

Figure 7.1 shows an example of a circuit split using the MSM strategy. Partition 1 and partition 2 can either be fabricated at the same or different foundries but none of them know the final connections between the two parts as indicated by the circles. The split can have no-fanout as indicated in Figure 7.1 (a) or include the fanout (3x4) as shown in Figure 7.1 (b). The cut nets of both fanout and no-fanout splits can be the same but the cut sizes are different. For both splits it is impractical for the adversary to brute-force all connections to find a correct solution simply because of the size of the solution space.

To address this problem, we propose to use a Boolean Satisfiability (SAT) solver with

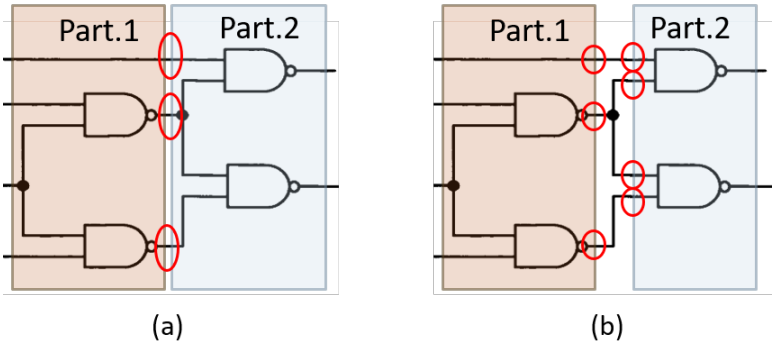


Figure 7.1: MSM example (a) without fanout and (b) with fanout split. Partition 1 and partition 2 can be fabricated at the same or different foundries but the connections between them are hidden. The goal of the adversary is to connect outputs of partition 1 to the inputs of partition 2 correctly.

hints obtained from circuit function analysis. Different from the SAT attack proposed in [156], we introduce grouping hints that can significantly reduce the runtime of the SAT attack. The main contributions of this chapter include:

- Two SAT-based attacks are compared. Results show that the grouping hint is much more effective than the no-fanout hint, which turns out to be even worse than without applying any hint due to the increase of virtual gate counts.
- Hard grouping algorithm and soft grouping strategies are proposed to significantly reduce the runtime of SAT attack. The hard grouping algorithm is independent of physical implementation of the split therefore no physical defensive mechanism, such as place and route perturbation, can be effective. The use of soft grouping strategies can further reduce attack runtime using the hints obtained from physical implementations of the design.

7.2 The SAT-Based Attack

7.2.1 SAT Attack Modeling

In this section we model the reconstruction of missing connections of split manufacturing as SAT-based attack proposed in [133], which has been shown to be an effective attack to retrieve the correct key of many logic obfuscation schemes. The SAT-based attack algorithm allows the adversary to decrypt an obfuscated netlist using a small amount of input patterns and their corresponding outputs (distinguishing input/output pairs or DIPs) from a functional circuit. The algorithm iteratively finds such DIPs and formalize them as a sequence of SAT formulas to be solved by a SAT solver. Each DIP can rule out a subset of wrong keys and the algorithm is guaranteed to find an equivalent class of the correct key.

To formulate the problem the first step is to model the missing connections with virtual multiplexer (mux) or demultiplexer (demux) gates with selection keys. As shown in Figure 7.2, there are two possible ways to model the connections with a cut size of 3 MSM. Figure 7.2 (a) shows a connection network using 3 mux gates where each of the mux is configured by a key k_i. The network models that m_i can be connected to anyone of the n_i wires. Figure 7.2 (b) shows a connection network using 3 demux gates representing a model that each n_i can only connect to one m_i, which is the ORed value of all demuxes. One of the demux outputs will be n_i and others will be zero depending on the key k_i of the demux. This model intrinsically constrains the connection to be no-fanout while maintaining the same size of keys as in Figure 7.2 (a), which is $n * log(n)$ bits for a cut size of n.

Combining the virtual connection network and the split parts we can now apply SAT attack to the design to solve the keys. The objective is to retrieve the correct values of all key bits in order to reconstruct the missing connections. The attacking model assumes we have access to following aspects:

1. The gate-level netlists of both partition 1 and partition 2. Along with the model for missing wires from partition 1 to partition 2, the conjunction normal form (CNF) $C(X, K, Y)$ of whole design can be obtained and the function of the design is rep-

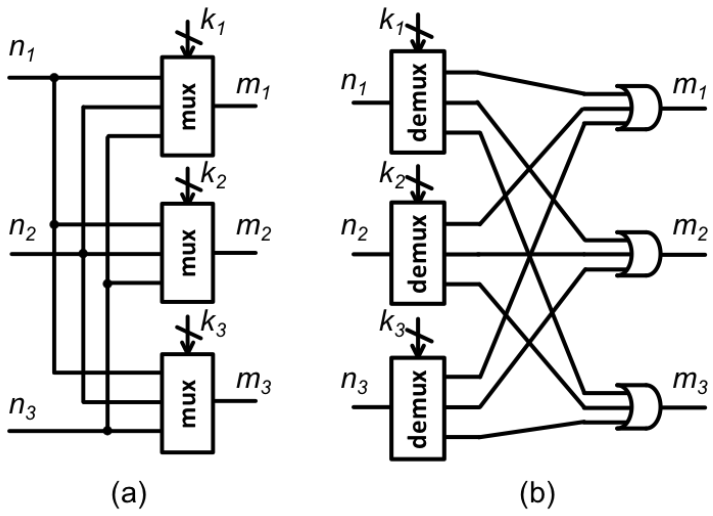


Figure 7.2: Modeling example of no-fanout MSM with cut size of 3. (a) mux network (b) demux network.

resented as $Y = f(X, K)$, where X is the primary input of the circuits and $K = (k_1, k_2, ..., k_i)$ is the selection keys to all mux or demux gates.

2. A fully functional chip obtained from the market, from which an attacker can observe the correct output of the circuit given an input $Y = eval(X)$

7.2.2 Runtime Results

In our experiments we use ISCAS85 and Microelectronics Center of North Carolina (MCNC) benchmarks to evaluate the runtime of two connection networks. In our experiments the attack terminates either when correct keys are found or the runtime is larger than 24 hours (TO). Once a correct key is found, the circuit will behave exactly the same as the original circuit before split. Different sizes of cut nets are tried and the attack tries to find a key that matches all outputs. Table 7.1 shows the runtime of the mux and demux networks. There are multiple ways to cut nets for a design while maintaining the balance of size of partitions but in general the runtime is proportional to the cut size. For some benchmarks if the cut size is larger than 100 then it becomes difficult to find the correct key. The demux network models the connection in such a way that every output of partition 1 can only connect to one

Table 7.1: mux and demux network runtime results in seconds.

circuits	Cut size	mux	demux	Cut size	mux	demux
c3540	52	90	159	115	1,588	TO
c5315	93	690	3,108	120	TO	TO
c7552	50	257	604	108	TO	TO
seq	70	165	682	165	TO	TO
apex4	47	26	25	251	20,666	TO
ex1010	72	2,007	1,060	281	TO	TO
des	85	38	219	346	8,339	TO

input of partition 2, which exploits the information to the adversary that the connections are without fanout for the split shown in Figure 7.1 (a). However, except for some small cut sizes, most runtimes of demux networks are much larger than the mux network even though the size of the keys are the same. One possible reason is that there are n OR gates each with n inputs in the demux network, which increases the number of clauses significantly in CNF and can slow down the SAT solver [157]. In the rest of the chapter we will focus on the mux connection work attacks.

7.2.3 Grouping Hints

From Table 7.1 we know that the runtime of solving the key can be effected significantly by the complexity of the connection network in addition to the cut size. Therefore, one way to reduce the runtime is to use a simpler connection network that translates to fewer CNFs. In other words, if the connections can be represented by smaller mux gates the runtime can be significantly reduced. One approach to reduce the mux network complexity is to apply a grouping hint to each of the mux, which contains the information of the candidates from partition 1 to partition 2.

With grouping hints the adversary can model the connection with a smaller mux network because now the candidate connections of m_i are not all n_i but can be a sub-group of n_i.

Table 7.2: mux network runtime results in seconds with grouping hints.

circuits	Cut size	No Hint	50%	33%	20%
c3540	115	1,588	379	143	79
c5315	120	TO	907	128	39
c7552	108	TO	TO	10,824	378
seq	165	TO	TO	895	244
apex4	251	20,666	4,574	2,791	974
ex1010	281	TO	TO	6,568	3,923
des	346	8,339	2,696	1,386	527

For example, the key length for the no-fanout split is not $n * log(n)$ anymore but becomes $n * log(pn)$ where p is the grouping hint percentage, which means that an input of partition 2 can only be connected to p portion of connections from partition 1. Table 7.2 shows the results when different p's are imposed to the mux connection network. We can see that some testbenches show significant runtime reduction when 50% of p is imposed, and as p keeps getting smaller, all benchmarks can be solved and most of the runtime are almost hundred times smaller compared to no hints.

From Table 7.2 we know that grouping can help reduce the runtime significantly. However a wrong grouping hint can cause the SAT solver a long runtime yet still cannot find the correct solution, therefore the grouping hints should be carefully computed. To address this issue we propose an algorithm to find hard grouping hints that are guaranteed to include correct groupings irrespective to physical constraints and routing heuristics. Details of soft grouping hints, which do not guarantee to include the correct connections, will be discussed in Section 7.4.

7.3 Hard Grouping Hints and Results

7.3.1 Hard Grouping Algorithm

In this section we present the algorithm to find hard grouping hints that are irrespective to physical constraints or routing heuristics. Hard grouping hints guarantee to include correct connections because they are completely independent of how the circuit is physically implemented but purely depending on the functionality of the circuit itself. For a cut size of n, define the n-bit output from partition 1 as ***Z1***$=(z1_1, z1_2, ...z1_n)$ and input to partition 2 as ***Z2***$=(z2_1, z2_2, ...z2_n)$. The goal of the hard grouping algorithm is to find the candidate connections of each $z2_i$ from ***Z1***.

The hard grouping algorithm is implemented in Python as a Satisfiability Modulo Theories (SMT) problem and solved by an existing SMT solver [158], which is a verification engine that understands a satisfiability problem at a higher level of abstraction other than Boolean formulas while still retaining the speed and efficiency of modern SAT engines.

The hard grouping algorithm first assigns ***Z1*** with fixed number of bits being one (or zero), which we note as *hot bits*, to SMT solver and it finds a valid input-output pair (X, Y) of the whole complete design to generate such ***Z1*** for partition 1. Next step is to find all possible patterns of ***Z2*** with the same number of hot bits which can reproduce the same Y of the whole complete design. The grouping information can then be found by mapping hot bits in ***Z1*** to hot bits in each of ***Z2***.

In our attacking model, the adversary has access to (1) gate-level netlists of both partition 1 and partition 2 such that partitions can be represented as a set of SMT formula, $S(X, Z1, Z2, Y)$, and (2) a fully functional chip obtained from the market which can be used to observe the correct output given an input, $Y = eval(X)$. The algorithm for finding hard grouping hints for n-bit ***Z1*** using a specified number of hot bits hb, and hot bit type (1 or 0), hb_type, is shown in Algorithm 2. It returns a map from each net in ***Z1*** to all possible candidates in ***Z2***.

A simple example to illustrate the idea of the algorithm is given in Figure 7.3: a

Z1=(11000) and its corresponding (X, Y) is found. Under such constraints, assume ***Z2***=(00110) is the only solution found to generate the same Y for partition 2. Now we know that $z1_1$ and $z1_2$ can only connect to $z2_3$ and $z2_4$ (all locations with 1's in ***Z2***), and all other connections of ***Z1*** can only connect to $z2_1$, $z2_2$, and $z2_5$ (all locations with 0's in ***Z2***). If there are multiple solutions of ***Z2*** for the same ***Z1*** and (X, Y), the union of the groupings should be the final grouping found for the ***Z1***. For every ***Z1*** with different solutions we take the intersection of the groupings found so far to obtain smaller grouping sizes because a correct connection should always exist no matter what inputs or ***Z1***s are.

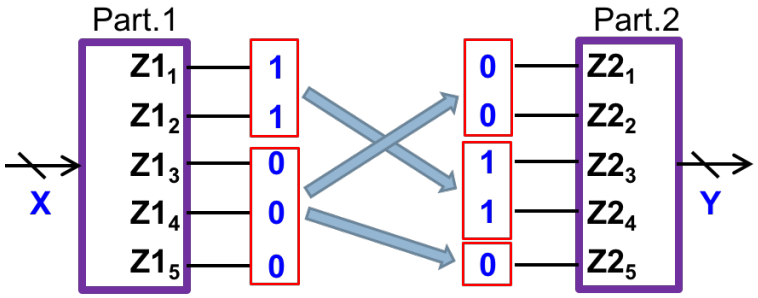


Figure 7.3: Hard grouping example.

To enhance the speed of this algorithm, we introduce the concept of Distinguishing ***Z2***($Z2^d$). For a fixed ***Z1***, it is only necessary to find ***Z2*** which can reveal new grouping information instead of all possible solutions of ***Z2***. For instance, if $Z1 = 01100$ and we have found $Z2 = 00110$ and 01010, from the perspective of bits of value 1 in ***Z1***, we know that possible candidates are $z2_2$, $z2_3$ and $z2_4$. The next ***Z2*** to be found is distinguishing if and only if it can reveal new candidates, which are $z2_1$ and $z2_5$. Thus, $Z2 = 10100$ is a $Z2^d$ as it reveals $Z2_1$ as an additional possible candidates while $Z2 = 01100$ is not distinguishing. Applying the constraints of finding distinguishing ***Z2*** after a new ***Z2*** is found speeds up the algorithm significantly.

7.3.2 Number of Hot Bits

In cases of large cut size, constraints in line 3 of Algorithm 2 are usually unsatifiable if desired number of hot bits hb in ***Z1*** and ***Z2*** is small. For example in DES with no-fanout

Algorithm 2 SMT Find_Grouping Algorithm

1: **function** FIND_GROUP($eval, hb\_type, hb$)
2: $i = 1$;
3: $\boldsymbol{F} = S(X, Z1, Z2, Y) \wedge (Y = eval(X))$
4: $\boldsymbol{F} = \boldsymbol{F} \wedge$ (number of hb_type in Z1 and Z2 $= hb$)
5: **while** $sat[\boldsymbol{F}]$ **do**
6: $Z1_i = smt\_assignment_{Z1}[\boldsymbol{F}]$
7: $X_i = smt\_assignment_X[\boldsymbol{F}]$
8: $\boldsymbol{F}_{new} = \boldsymbol{F} \wedge (Z1 = Z1_i) \wedge (X = X_i)$
9: $j = 1$
10: **while** $sat[\boldsymbol{F}_{new}]$ **do**
11: $Z2_j = smt\_assignment_{Z1}[\boldsymbol{F}_{new}]$
12: **for** 1's in $Z1_i$ **do**
13: $Group_{one} = Group_{one} \cup$ (1's in $Z2_j$)
14: **end for**
15: **for** 0's in $Z1_i$ **do**
16: $Group_{zero} = Group_{zero} \cup$ (0's in $Z2_j$)
17: **end for**
18: $\boldsymbol{F}_{new} = \boldsymbol{F}_{new} \wedge (Z2 \neq Z2_j^d) \wedge (Z2$ is a $Z2^d)$
19: $j = j + 1$
20: **end while**
21: $\boldsymbol{F} = \boldsymbol{F} \wedge (Z1 \neq Z1_i)$
22: $i = i + 1$;
23: **end while**
24: **for** all $z1_k$ in $Z1$ **do**
25: $Group[z1_k] = Group_{one}[z1_k] \cap Group_{zero}[z1_k]$
26: **end forreturn** $Group$
27: **end function**

cut size 346, it is unsatifiable to find an input X to generate only one 1 and 345 0's at the output of partition 1. Therefore starting with one hot bit may be an inefficient approach.

Figure 7.4 shows the decoupled runtime of different ending *hb* of DES with 346 cut size starting from 173 hot bits to the ending hot bits indicated. Note that we start with number of hot bits being half of the cut size because we try both 1's and 0's as hot bit types. We can see that the smaller the ending *hb* is, the faster it is for the SAT attack to find the connection because the size of groups are smaller, however the time spent on the grouping algorithm also become longer simply because the number of iterations the algorithm is executed. For some small ending *hb* the grouping time itself is already longer than the total time. Therefore from empirical observations we propose to start Algorithm 2 with *hb* being half of the cut size and decrease *hb* by one until more than half of group sizes are smaller than 20% or when the overall group sizes are not getting smaller. The complete algorithm for finding hard grouping is shown in Algorithm 3.

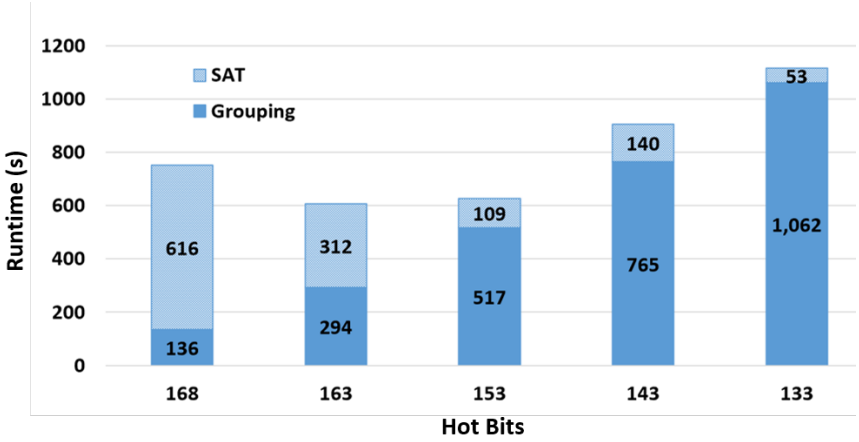


Figure 7.4: DES runtime with 346 cut size and different ending hot bits.

7.3.3 Hard Grouping Results without Fanout

In Table 7.3 we show the runtime of the hard grouping algorithm and SAT attack algorithm after applying hard grouping hints. Compared to original SAT runtime in Table 7.1, the total runtime with hard hints has been improved by 13X to more than 1,440X.

Algorithm 3 Complete SMT Grouping Algorithm

Input: SMT_formula and *eval*

Output: *Final_Group*

1: **for** i =*half of cut size* **do**
2: $Group_{one\_i} = FIND\_GROUP(eval, 1, i)$
3: $Group_{zero\_i} = FIND\_GROUP(eval, 0, i)$
4: $Group = Group \cap Group_{one\_i} \cap Group_{zero\_i}$
5: **if** $Group \leq 20\%$ **then**
6: *break*
7: **end if**
8: $i = i - 1$
9: **end for**

Table 7.3: Runtime (seconds) of hard grouping hints and reduction ratio compared to no hints. The total runtime compared is the sum of grouping and SAT time.

circuits	Cut size	Grouping Time	SAT Time	Reduction Ratio
c3540	115	51	5	28.4
c5315	120	23	37	>1,440.0
c7552	108	15	210	>384.0
seq	165	169	493	>130.5
apex4	251	482	233	28.9
ex1010	281	648	6,173	>12.7
des	346	294	312	13.8

Figure 7.5 shows the group size distribution of DES with 346 cut size after running Algorithm 3. Most group size are smaller than 20% of 346 and about 40% of the connections have group size of one, which means that these connections are already determined during the grouping procedure.

Figure 7.5: Group size distribution of DES with cut size 346.

7.3.4 Hard Grouping with Fanout

In the case of splitting with fanout where number of bits in ***Z2*** is larger than ***Z1*** as shown in Figure Figure 7.1 (b), line 4 in Algorithm 2 can be modified to $\boldsymbol{F} = \boldsymbol{F} \wedge$ (number of *hb_types* in Z1= *hb*) $\wedge$ (number of 1's/0's in Z2<=number of 1's/0's in Z1) to solve for the grouping. The idea is that the number of 1's/0's in ***Z2*** should be greater or equal to the number of 1's/0's in ***Z1***. For example, if $Z1 = (100)$, and there are five bits of ***Z2***, then all possible solutions of $Z2$ containing one to three 1's need to be found to construct the grouping from ***Z1*** to ***Z2***.

Table 7.4 shows the runtime of fanout split with and without hard hints. Compared to Table 7.2, the runtime of split with fanout on the same nets are much longer than split without fanout. For most benchmarks the key cannot be resolved in 24 hours without hints, but with hard grouping hints the runtime can be significantly improved.

Table 7.4: Runtime (second) of fanout split with hard grouping hints and reduction ratio compared to no hints. The total runtime compared is the sum of grouping and SAT time.

circuits	Cut size	No hint	Grouping	SAT	Reduction Ratio
c3540	115x187	4,706	186	27	22.1
c5315	120x269	TO	157	877	>83.6
c7552	108x188	TO	108	299	>212.3
seq	165x239	TO	226	2,839	>28.2
apex4	251x710	TO	36,501	9,727	>1.9
ex1010	281x677	TO	70,240	5,838	>1.1
des	346x455	TO	1,794	245	>42.4

7.4 Soft Grouping Hints and Results

7.4.1 Soft Grouping Strategy

Besides hard grouping hints, another way to reduce the mux network complexity is to apply soft hints from physical implementation constraints. Similar to proximity attacks [159], the adversary knows that a wire of partition 1 is likely to connect to the wires that are physically close to itself in partition 2 due to the interposer delay. Figure 7.6 shows simulated results of interconnect delay and transition slew using a commercial 65nm technology. Each interconnect connecting to the input of an inverter cell is driven by the largest buffer cell available in the standard cell library. Metal 8 with $5um$ width is used to emulate the interposer interconnect wire. We can see that as the wire becomes longer the delay and slew become larger. This information can be exploited by the adversary to narrow down possible connections and thus simplify the connection network. For example, a 2GHz design would require the path delay to be smaller than 500ps. In the 65nm technology we used, a normal setup time for a D flip-flop is about 100ps, which leaves 400ps margin for the total gate delay. As shown in Figure 7.6 if the wire is longer than 20mm, the wire delay is already larger than 400ps, therefore such connection can be pruned out in the connection network

model. Another information is the transition slew. The library defines that the max slew is about 385ps, and if a wire is longer than 10mm the slew becomes larger than 385ps, which tells the adversary that a connection longer than 10mm is not likely to be made.

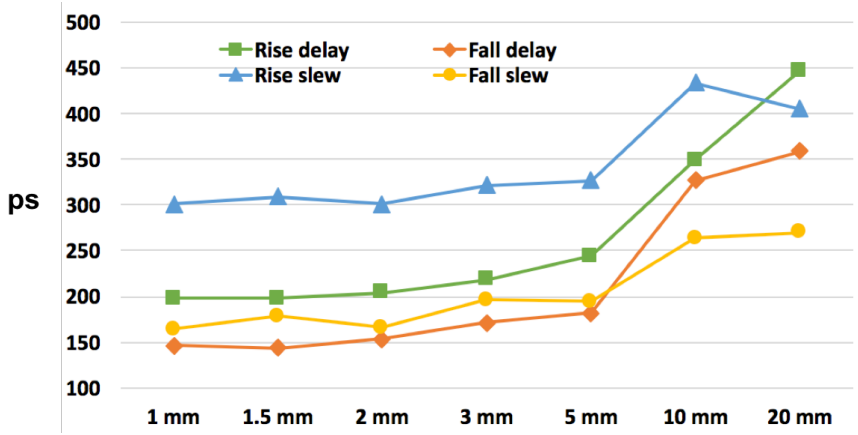


Figure 7.6: Interconnect delay and transition slew.

Given the modern GPU design specifications [155], which contains billion of gates and die size as large as about $600mm^2$, having a cut size of hundreds of thousands between the two partitions is expected from our empirical observations. Assume the design is split into two parts, each with the dimension of $10mm$ by $10mm$. With existing interposer technology of $50um$ pitch [160], the allowable number of interposers on each partition is about 40,000, therefore it is possible that most interposer sites will be used after split. The exemplary analysis of delay and transition constrains present in Figure 7.6 tells the adversary that connections from the left edge of partition 1 to the right edge of the partition 2 is not likely to be made as illustrated in Figure 7.7, which shows an unlikely connection marked as "X" and possible connections marked as "O".

The difference between hard hints and soft hints is that when soft hints are applied, the correct connections are no longer guaranteed to be included after the grouping, and the keys of the connection network may not be found. This is because the IC/IP designer can perform routing or placement perturbation to violate the physical design principles [159].

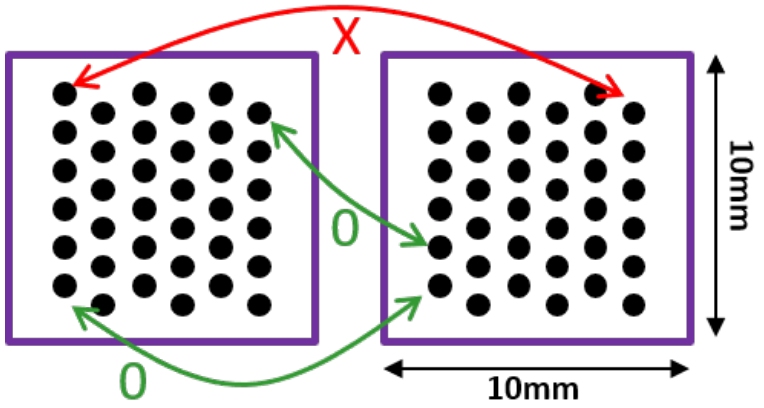


Figure 7.7: Interposer soft grouping example. The connection marked as "X" is not likely to happen due to interconnect delay and transition constraints

7.4.2 Results

To integrate soft grouping with hard grouping, the adversary can first apply hard grouping to obtain connections for partial nets and then apply soft hints to the rest of the nets to further reduce the runtime. Table 7.5 presents the percentages of solved nets (nets with grouping size one) after hard grouping and runtime results when 50% soft grouping is applied after hard grouping. We can see that about 11% to 64% of the connections are already solved without even applying SAT attack, and these connections are guaranteed to be correct because they are found by hard grouping algorithms, which have the same runtime as in Table 7.4. For those unsolved nets, the runtime of STA attack is significantly reduced compare to Table 7.4 because of the soft grouping. However, once soft grouping is applied, there is no guarantee that the corrcet connections can be found. In this experiment we include the correct connections for the purpose of demonstrating the runtime improvements.

Table 7.6 shows the results when 50% soft grouping hints are applied before executing the hard grouping algorithm. The runtime is further reduced compared to hard grouping only as shown in Table 7.4, but the final groupings are not guaranteed to include the correct connections. In practical the adversary can apply hard grouping hints first to find solutions for partial nets that are guaranteed to be correct and then apply soft hints to the rest of the

Table 7.5: Runtime (seconds) of fanout split of hard grouping first followed by 50% soft grouping.

circuits	Cut size	Solved Nets	SAT	Reduction Ratio
c3540	115x187	30%	15	23.4
c5315	120x269	12%	401	>154.8
c7552	108x188	16%	99	>417.4
seq	165x239	11%	290	>167.4
apex4	251x710	50%	2,591	>2.2
ex1010	281x677	64%	4,409	>1.2
des	346x455	56%	60	>46.6

nets, or apply the soft groupings that are highly likely to be true before the hard grouping to reduce the runtime. The grouping sequences can be applied in an arbitrary order depending on the actual implementation of the victim design.

7.5 Defense Strategies

Create floating connections. To defend the hard grouping algorithm, the IC/IP designer can create redundant floating connections at the output of partition 1 to cause confusion or even create unsolvable grouping solutions of Algorithm 2. For example, say ***Z1***=(1001) for a no-fanout split, $z1_4$ is the redundant floating net that does not connect to partition 2 and a corresponding input X of partition 1 is found. The algorithm tries to find ***Z2*** with two 1's to generate $Y = eval(X)$, but since $z1_4$ is floating, such solution for ***Z2*** may not exist, so the grouping hints may not be generated. From our experiments we can see that the ability of SAT attack itself is limited, therefore without the help of hard grouping algorithm the overall performance of the attack is significantly weakened.

Split the design into more partitions. To model the connection network the adversary needs to know the topological order of the partitions. When there are only two

Table 7.6: Runtime (seconds) of fanout split of 50% soft grouping first followed by hard grouping.

circuits	Cut size	No hint	Grouping	SAT	Reduction Ratio
c3540	115x187	4,706	179	10	24.9
c5315	120x269	TO	140	158	>289.9
c7552	108x188	TO	99	39	>626.1
seq	165x239	TO	224	407	>136.9
apex4	251x710	TO	4,277	8,978	>6.4
ex1010	281x677	TO	37,961	4,061	>2.3
des	346x455	TO	1,723	71	>48.2

partitions the order of the partitions can be easily figured out. If there are more than two partitions, finding the topological order becomes a more difficult task and there is no straightforward way to translate our attacking algorithm to solve partitions with unknown orders. The complexity may be too high for the attack to solve the key in practical runtime.

7.6 Conclusion

In this chapter we present SAT attacks to 2.5D split manufacturing based on the hard grouping hints obtained from SMT-based grouping algorithms. We first show that the runtime of SAT attack can be significantly affected by the complexity of the connection network, therefore to reduce the runtime a simplified network should be used. Then we propose hard grouping algorithms to find grouping hints that guarantee to include correct connections to effectively simplify the connection network and reduce the runtime of SAT attack significantly. Our experiments are done on both fanout and no-fanout splits and results show that the runtime is improved by more than hundred times for some testbenches compared to SAT attack without hints. Finally we discuss several defense strategies from designer's perspective to protect the split manufacturing from our attack. Our future work will focus on the attacks on multiple-split interfaces and splitting with floating or sequential elements.

CHAPTER 8

Conclusion and Future Work

This dissertation is devoted to developing new techniques and methodologies to protect IC/IP designer from attacks of malicious parties. We first address several vulnerabilities in the modern IC supply chain environment and limitations of existing DFS approaches, followed by our proposed solutions and experimental results showing the much improved security. In the second part of this chapter we discuss several future research directions. The major contributions are summarized as follows:

1. **Assessing Viability of Delay-Based PUFs.** The uniqueness and stability of SRAM PUFs, arbiter PUFs, and RO PUFs are measured based on silicon results. We show that a delay-based PUF are susceptible to silicon side-channel attacks because of systematic fabrication variations. Furthermore, we model two sources of instabilities of delay-based PUFs: metastability of the arbiter circuit, and jitter accumulation of RO. We show that the probability of a unstable CRP being discarded due to metastability and suggest that longer (but compact in layout extent) delay chains is one possible way to reduce impact of metastable comparisons. For the impact of random jitter, we derived the probability of a jitter caused measurement error which shows that increased measurement time is a viable way to decrease such error probability.

2. **UNBIAS PUF: A Physical Implementation Bias Agnostic PUF.** The proposed UNBIAS PUF effectively reduces PUF implementation efforts by mitigating the impact of biased delay paths and metastability issues. Without complex post-layout analysis or hand-crafted physical design effort, the proposed measurement can still extract local device randomness. The inspection bit can be determined efficiently from the intra-FHD and inter-FHD prediction models. Two UNBIAS PUFs, a weak and a strong

PUF, are implemented on 11 FPGAs without imposing any physical layout constraints. The fact that the proposed scheme is immune to physical implementation bias would allow the UNBIAS PUF to be integrated in a high-level description of the design, such as during RTL design.

3. **LEDPUF: Stability-Guaranteed Physical Unclonable Functions through Locally Enhanced Defectivity.** We propose abd fabricate the first stability-guaranteed PUFs that requires no stability enhancement techniques, where the source of randomness is extracted from (1) locally enhanced DSA process and (2) gate oxide breakdown. Detailed constructions of the LEDPUFs are presented. Inter-distance measurements on the LEDPUFs show that both weak and strong LEDPUFs are ideally unique. The area and latency of the weak LEDPUF is much smaller than existing weak PUFs because no error correcting schemes are needed.

4. **PUF Security Evaluation through Guesswork Analysis.** A unified guesswork-based analyses for PUFs is developed. We show through guesswork analysis that stability has a more severe impact on the PUF security than biased responses. In addition, we analyze guesswork for two new problems: Guesswork under probability of attack failure, and the guesswork of strong PUFs.

5. **SLATE: A Secure Lightweight Entity Authentication Hardware Primitive.** SLATE is proposed as a lightweight and secure entity authentication primitive. We show that SLATE is resistant to model building attacks and SAT attacks that are known to be effective to most existing strong PUFs or logic obfuscation. We compare the hardware implementation area of SLATE to secure strong PUFs and ciphers to show that existing entity authentication solutions are at least 44% to 7.1× larger than SLATE. Also, we show that ideally SLATE is information theoretically secure in terms of the amount of information revealed by observing valid CRPs. Finally, we propose a tamper evident one-time-read method to ensure the unpredictability and the unclonability of SLATE.

6. **Reverse Engineering of 2.5D Split Manufactured ICs.** We present SAT attacks

to 2.5D split manufacturing based on the grouping hints obtained from SMT-based grouping algorithms. We first show that the runtime of SAT attack can be significantly affected by the complexity of the connection network, therefore to reduce the runtime a simplified network should be used. Then we proposed grouping algorithms to find hard grouping hints that guarantee to include correct connections to effectively simplify the connection network and reduce the runtime of SAT attack dramatically. Our experiments are done on both fanout and no-fanout splits and results show that the runtime is improved by more than 1,000 for some testbenches compared to SAT attack without hints. Finally we discuss several defense strategies from designer's perspective to protect the split manufacturing from our attack.

The ideas and methodologies proposed in this dissertation could be further examined in the following interesting research directions:

1. **ATPG and SAT secure obfuscation.** Built upon the proposed stability-guaranteed LEDPUF and the tamper evident one-time-read mechanism, it is worth studying the possibility of Automatic Test Pattern Generation (ATPG) and SAT secure obfuscation techniques since the output of a weak stable PUF can be considered as a stuck-at-one/zero fault. The obfuscation should leverage the concept of sequential ATPG and relate the key-solving problem of the obfuscation to existing sequential ATPG problems. The SAT secure obfuscation is another interesting research direction since existing techniques often suffer from removal attack or approximate attacks.

2. **Remote activation and tracking through key exchanging protocol.** In the IC supply chain, the designer must obtain the PUF value to activate or track each component. However it is a difficult task if no one is trustworthy. Existing solutions often require asymmetric ciphers or assume that a part of the secret is communicated through a secure channel. An interesting research direction could be on how to establish shared secrets between the chip and the designer using the concept of key exchanging, such as the Diffie-Hellman key exchange protocol. For example, instead of transmitting

the PUF values in the open, the values are encrypted with a hidden pattern or a watermarking of the function only known to the designer before being transmitted.

3. **Distributed mutual authentication.** In the IoT environment, such as smart home, automotive components, or sensor grids, it could be that there is no central trusted server or the communication between the server and the nodes can be expensive. In such case, the cost may be significantly reduced if the connected nodes can authenticate themselves with distributed mutual authentication protocols, similar to the blockchain technology or the chained hash functions. The attacker would have to modify more than 50% of the connected nodes or all the parents of the victim node to avoid being detected by the network. The construction of a self-authenticating network could be another interesting topic that deserves future examination.

REFERENCES

[1] U. Ruhrmair, *et al.*, "PUF Modeling Attacks on Simulated and Silicon Data," *IEEE TIFS*, 2013.

[2] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and Implementation of PUF-Based "Unclonable" RFID ICs for Anti-Counterfeiting and Security Applications," in *International Conference on RFID*, April 2008.

[3] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions," in *Towards Hardware-Intrinsic Security*. Springer Berlin Heidelberg, 2010, pp. 3–37.

[4] S. W. Jung and S. Jung, "HRP: A HMAC-based RFID mutual authentication protocol using PUF," in *The International Conference on Information Networking*, Jan 2013.

[5] A. V. Herrewege *et al.*, "Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs," in *International Conference on Financial Cryptography and Data Security*, July 2012.

[6] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A Survey on Lightweight Entity Authentication with Strong PUFs," in *ACM Computing Surveys*, Nov 2015.

[7] S. U. Hussain, S. Yellapantula, M. Majzoobi, and F. Koushanfar, "BIST-PUF: Online, Hardware-based Evaluation of Physically Unclonable Circuit Identifiers," in *Proc. ICCAD*, 2014.

[8] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT Systems: Design Challenges and Opportunities," in *Proc. ICCAD*, 2014.

[9] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. CCSC*, 2002.

[10] G. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Proc. DAC*, 2007.

[11] C.-E. Yin, G. Qu, and Q. Zhou, "Design and implementation of a group-based RO PUF," in *Proc. DATE*, 2013.

[12] D. Holcomb, W. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, 2009.

[13] X. Xu and W. Burleson, "Hybrid side-channel/machine-learning attacks on PUFs: A new threat?" in *Proc. DATE*, 2014.

[14] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning Physically Unclonable Functions," in *IEEE International Symposium on HOST*, 2013.

[15] C.-E. Yin and G. Qu, "Improving PUF security with regression-based distiller," in *Proc. DAC*, 2013.

[16] P. Prabhu *et al.*, "Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations," in *Proc. TRUST*, 2011.

[17] S. Rosenblatt, S. Chellappa, A. Cestero, N. Robson, T. Kirihata, and S. Iyer, "A Self-Authenticating Chip Architecture Using an Intrinsic Fingerprint of Embedded DRAM," *IEEE JSSC*, 2013.

[18] P. Koeberl, U. Kocabas, and A.-R. Sadeghi, "Memristor PUFs: A new generation of memory-based Physically Unclonable Functions," in *Proc. DATE*, 2013.

[19] C. Helfmeier *et al.*, "Physical vulnerabilities of Physically Unclonable Functions," in *Proc. DATE*, 2014.

[20] J. Lee *et al.*, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *IEEE International Symposium on VLSI Circuits*, 2004.

[21] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter pufs exploiting cmos device noise," in *IEEE International Symposium on HOST*, 2013.

[22] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. ACSAC*, 2002.

[23] L. Cheng *et al.*, "Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across Wafer Variability," *IEEE TCAD*, 2011.

[24] L. Portmann and T. H.-Y. Meng, "Metastability in CMOS library elements in reduced supply and technology scaled applications," *IEEE JSSC*, 1995.

[25] L. Lin, S. Srivathsa, D. Krishnappa, P. Shabadi, and W. Burleson, "Design and Validation of Arbiter-Based PUFs for Sub-45-nm Low-Power Security Applications," *IEEE TIFS*, 2012.

[26] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer, "Modeling and observing the jitter in ring oscillators implemented in FPGAs," in *IEEE Workshop on DDECS*, 2008.

[27] A. R. Krishna *et al.*, "MECCA: a robust low-overhead PUF using embedded memory array," in *CHES*, Oct 2011.

[28] M. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, "ARO-PUF: An aging-resistant ring oscillator PUF design," in *Proc. DATE*, March 2014.

[29] C. Herder *et al.*, "Physical Unclonable Functions and Applications: A Tutorial," *Proc. of the IEEE*, pp. 1126–1141, Aug 2014.

[30] L. Feiten *et al.*, "Improving RO-PUF Quality on FPGAs by Incorporating Design-Dependent Frequency Biases," *IEEE ETS*, 2015.

[31] A. Maiti and P. Schaumont, "Improving the Quality of a Physical Unclonable Function using Configurable Ring Oscillators," in *International Conference on FPL*, 2009.

[32] D. Lim *et al.*, "Extracting secret keys from integrated circuits," *IEEE Transactions on VLSI Systems*, 2005.

[33] D. P. Sahoo, R. S. Chakraborty, and D. Mukhopadhyay, "Towards Ideal Arbiter PUF Design on Xilinx FPGA: A Practitioner's Perspective," in *Euromicro Conference on DSD*, 2015.

[34] T. Xu and M. Potkonjak, "Robust and flexible FPGA-based digital PUF," in *International Conference on FPL*, 2014.

[35] "Xilinx Artix-7 FPGAs Data Sheet," 2015.

[36] D. P. Sahoo *et al.*, "Architectural Bias: a Novel Statistical Metric to Evaluate Arbiter PUF Variants," Cryptology ePrint Archive, Report 2016/057, 2016.

[37] V. G. A. Maiti and P. Schaumont, "A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions," in *Embedded Systems Design with FPGAs*, 2013.

[38] C. Gu and M. O'Neill, "Ultra-compact and Robust FPGA-based PUF Identification Generator," in *IEEE ISCAS*, 2015.

[39] S. Morozov *et al.*, "An analysis of delay based PUF implementations on FPGA," in *Reconfigurable Computing: Architectures, Tools and Applications*. Springer Berlin Heidelberg, 2010.

[40] M. I. Takanori Machida, Dai Yamamoto and K. Sakiyama, "A New Arbiter PUF for Enhancing Unpredictability on FPGA," in *The Scientific World Journal*, 2015.

[41] B. Halak, Y. Hu, and M. Mispan, "Area Efficient Configurable Physical Unclonable Functions for FPGAs Identification," in *IEEE ISCAS*, 2015.

[42] F. Kodtek and R. Lrencz, "A Design of Ring Oscillator Based PUF on FPGA," in *IEEE International Symposium on DDECS*, 2015.

[43] J. H. Anderson, "A PUF design for secure FPGA-based embedded systems," in *Proc. ASP-DAC*, 2010.

[44] S. D. Wei-Che Wang, Yair Yona and P. Gupta, "LEDPUF: Stability-Guaranteed Physical Unclonable Functions through Locally Enhanced Defectivity," in *HOST*, 2016.

[45] Q. Zhang *et al.*, "FROPUF: How to Extract More Entropy from Two Ring Oscillators in FPGA-Based PUFs," *IACR*, 2015.

[46] J. Guajardo, G.-J. S. S. S. Kumar, and P. Tuyls, "FPGA Intrinsic PUFs and Their Use for IP Protections," in *CHES*, Sep 2007.

[47] M.-D. M. Yu and S. Devadas, "Secure and Robust Error Correction for Physical Unclonable Functions," in *IEEE Des. Test*, 2010.

[48] L. Lin, S. Srivathsa, D. Krishnappa, P. Shabadi, and W. Burleson, "Design and Validation of Arbiter-Based PUFs for Sub-45-nm Low-Power Security Applications," *IEEE TIFS*, 2012.

[49] D. Ganta and L. Nazhandali, "Study of IC Aging on Ring Oscillator Physical Unclonable Functions," in *IEEE ISQED*, 2014.

[50] W. Che, J. Plusquellic, and S. Bhunia, "A Non-Volatile Memory Based Physically Unclonable Function without Helper Data," in *Proc. ICCAD*, 2014.

[51] L. Feiten, T. Martin, M. Sauer, and B. Becker, "Improving RO-PUF Quality on FPGAs by Incorporating Design-Dependent Frequency Biases," in *IEEE ETS*, 2015.

[52] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *IEEE International Workshop on Information Forensics and Security*, 2010.

[53] J. Delvaux and I. Verbauwhede, "Key-recovery attacks on various RO PUF constructions via helper data manipulation," in *Proc. DATE*, March 2014.

[54] X. Zhang, "VLSI Architectures for Modern Error-Correcting Codes," 2015.

[55] I. T. R. for Semiconductors, "http://public.itrs.net/."

[56] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using PUF-based logic," in *Proc. ICCAD*, 2014.

[57] T. Xu, J. B. Wendt, and M. Potkonjak, "Secure Remote Sensing and Communication Using Digital PUFs," in *Proc. ANCS*, 2014.

[58] B. Xu, R. Piñol, M. Nono-Djamen, S. Pensec, P. Keller, P.-A. Albouy, D. Lévy, and M.-H. Li, "Self-assembly of liquid crystal block copolymer peg-b-smectic polymer in pure state and in dilute aqueous solution," *Faraday discussions*, vol. 143, 2009.

[59] N. Jarnagin, "High x block copolymers for sub 20 nm pitch patterning: Synthesis, solvent annealing, directed self assembly, and selective block removal," Ph.D. dissertation, Geogia Institute of Technology, Dec. 2013.

[60] M. Kim, E. Han, D. P. Sweat, and P. Gopalan, "Interplay of surface chemical composition and film thickness on graphoepitaxial assembly of asymmetric block copolymers," *Soft Matter*, vol. 9, no. 26, pp. 6135–6141, 2013.

[61] Y. Badr, A. J. Torres, and P. Gupta, "Mask assignment and synthesis of DSA-MP hybrid lithography for sub-7nm contacts/vias," in *Proc. DAC*, June 2015.

[62] H.-C. Kim, S.-M. Park, and W. D. Hinsberg, "Block copolymer based nanostructures: materials, processes, and applications to electronics," *Chemical reviews*, vol. 110, no. 1, pp. 146–177, 2009.

[63] H. Pathangi *et al.*, "Defect mitigation and root cause studies in IMEC's 14 nm half-pitch chemo-epitaxy DSA flow," *Proc. SPIE*, 2015.

[64] D. Sundrani, S. Darling, and S. Sibener, "Hierarchical assembly and compliance of aligned nanoscale polymer cylinders in confinement," *Langmuir*, vol. 20, no. 12, pp. 5091–5099, 2004.

[65] G. Fenger, J. A. Torres, Y. Ma, Y. Granik, P. Krasnova, A. Fouquet, J. Belledent, A. Gharbi, and R. Tiron, "Compact model experimental validation for grapho-epitaxy hole processes and its impact in mask making tolerances," *Proc. SPIE*, 2014.

[66] C. T. Black, "Polymer self-assembly as a novel extension to optical lithography," *ACS Nano*, vol. 1, no. 3, pp. 147–150, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1021/nn7002663

[67] H. Kang, G. S. Craig, E. Han, P. Gopalan, and P. F. Nealey, "Degree of perfection and pattern uniformity in the directed assembly of cylinder-forming block copolymer on chemically patterned surfaces," *Macromolecules*, vol. 45, no. 1, pp. 159–164, 2011.

[68] K. Yoshimotoa and T. Taniguchi, "Large-Scale Dynamics of Directed Self-Assembly Defects on Chemically Pre-Patterned Surface," *Proc. SPIE*, 2013.

[69] M. Mller, W. Li, J. C. O. Rey, and U. Welling, "Kinetics of directed self-assembly of block copolymers on chemically patterned substrates," *Journal of Physics: Conference Series*, vol. 640, no. 1, p. 012010, 2015.

[70] B. L. Peters *et al.*, "Graphoepitaxial Assembly of Cylinder-Forming Block Compolymers in Cylindrical Holes," in *Journal of Polymer Science*, Dec 2014.

[71] F. Detcheverry *et al.*, "Monte Carlo simulations of a coarse grain model for block copolymers and nanocomposites," in *Macromolecules*, 2008, pp. 4989–5001.

[72] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF Analysis," Aug 2013, pp. 30–38.

[73] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning Physically Unclonable Functions," in *IEEE International Symposium on HOST*, June 2013, pp. 1–6.

[74] Y. Dodis, A. Reyzin, and A. Smith, "Fuzzy extractor, A brief survey of results from 2004 to 2006," in *Security with Noisy Data*. Springer Berlin Heidelberg, 2007.

[75] E. Arikan, "An inequality on guessing and its application to sequential decoding," *IEEE Tran. on Inf. Th.*, vol. 42, 1996.

[76] J. Massey, "Guessing and entropy," in *ISIT 1994*, 1994.

[77] J. Neumann, "Various techniques used in connection with random digits," *Applied Math Series*, 1951.

[78] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: a Study on the State of the Art and Future Research Directions," *Towards Hardware-Intrinsic Security*, 2010.

[79] M. Bellare, R. Canetti, and H. Krawczyk, "Keyed Hash Functions and Message Authentication," in *Crypto*, 1996.

[80] S. Jung and S. Jung, "HRP: A HMAC-based RFID mutual authentication protocol using PUF," in *ICOIN 2013*.

[81] M.-Y. Wang *et al.*, "An HMAC processor with integrated SHA-1 and MD5 algorithms," in *Proc. ASP-DAC*, 2004.

[82] J. Sune, G. Mura, and E. Miranda, "Are soft breakdown and hard breakdown of ultrathin gate oxides actually different failure mechanisms?" *IEEE Electron Device Letters*, April 2000.

[83] S. U. Kim, "Analysis of Thin Gate Oxide Degradation During Fabrication of Advanced CMOS ULSI Circuits," *IEEE Transactions on Electron Devices*, 1998.

[84] H. C. Shin and C. Hu, "Thin gate oxide damage due to plasma processing," *Semiconductor Science and Technology*, Apr. 1996.

[85] S. Fang and J. P. McVittie, "A model and experiments for thin oxide damage from wafer charging in magnetron plasmas," *IEEE Electron Device Letters*, June 1992.

[86] Z. Wang *et al.*, "Strategies to cope with plasma charging damage in design and layout phases," in *International Conference on ICICDT*, May 2005.

[87] P. Liao *et al.*, "Physical origins of plasma damage and its process/gate area effects on high-k metal gate technology," in *IEEE IRPS*, April 2013.

[88] Z. Wang, "Detection of and Protection against Plasma Charging Damage in Modern IC Technology," in *Ph.D. thesis, Enschede*, Sep. 2004.

[89] J. Wang and H. Zhou, "Optimal Jumper Insertion for Antenna Avoidance Considering Antenna Charge Sharing," *IEEE TCAD*, Aug. 2007.

[90] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with jumper insertion for antenna avoidance," in *Proc. IEEE International SOC Conference*, Sept 2004.

[91] P. J. L. others, "Physical origins of plasma damage and its process/gate area effects on high-k metal gate technology," in *IEEE IRPS*, April 2013.

[92] U. Rührmair, C. Jaeger, and M. Algasinger, "An attack on puf-based session key exchange and a hardware-based countermeasure: Erasable PUFs," in *International Conference on FC*, 2011.

[93] K. K. Kim, "Reliable CMOS VLSI Design Considering Gate Oxide Breakdown."

[94] F. Tang *et al.*, "CMOS On-Chip Stable True-Random ID Generation Using Antenna Effect," *IEEE Electron Device Letters*, Jan 2014.

[95] J. Chen *et al.*, "Electron-Beam-Induced Current Study of Breakdown Behavior of High-K Gate MOSFETs," *Solid State Phenomena*, 2010.

[96] N. Tanaka, "Present status and future prospects of spherical aberration corrected TEM/STEM for study of nanomaterials," *Science and Technology of Advanced Materials*, 2008.

[97] N. Liu, S. Hanson, D. Sylvester, and D. Blaauw, "OxID: On-chip one-time random ID generation using oxide breakdown," in *Symposium on VLSI Circuits*, June 2010.

[98] U. Rührmair and D. E. Holcomb, "PUFs at a glance," in *Proc. DATE*, March 2014.

[99] T. Cover and J. Thomas, *Elements of Information Theory.* John Wiley& Sons, 2006.

[100] J. E. Kennedy and M. P. Quine, "The total variation distance between the binomial and poisson distributions," *Ann. Probab.*, Jan 1989.

[101] X. Li, H. Zhong, Z. Tang, and C. Jia, "Reliable Antifuse One-Time-Programmable Scheme With Charge Pump for Postpackage Repair of DRAM," *Proc. VLSI Design*, Sep 2015.

[102] F. Armknecht *et al.*, "Towards a Unified Security Model for Physically Unclonable Functions," in *Topics in Cryptology.* Cham: Springer International Publishing, 2016.

[103] A. Rukhin *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," 2010.

[104] K. Scarfone and M. Souppaya, "Guide to enterprise password management," *Recommendations of the NIST*, 2009.

[105] J. Kelsey *et al.*, "Secure applications of low entropy keys," *Proc. of ISW*, Sep 1998.

[106] R. Maes, *PUF-Based Entity Identification and Authentication.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 117–141. [Online]. Available: https://doi.org/10.1007/978-3-642-41395-7_5

[107] A. Aysu, Y. Wang, P. Schaumont, and M. Orshansky, "A new maskless debiasing method for lightweight physical unclonable functions," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2017, pp. 134–139.

[108] E. Arikan and N. Merhav, "Guessing subject to distortion," *IEEE Transactions on Information Theory*, May 1998.

[109] R. Sundaresan, "Guessing under source uncertainty," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 269–287, Jan 2007.

[110] ——, "Guessing Under Source Uncertainty With Side Information," in *IEEE ISIT*, July 2006.

[111] D. Malone and W. G. Sullivan, "Guesswork and entropy," *IEEE Transactions on Information Theory*, March 2004.

[112] M. M. Christiansen, K. R. Duffy, F. du Pin Calmon, and M. Medard, "Multi-User Guesswork and Brute Force Security," *IEEE Transactions on Information Theory*, Dec 2015.

[113] R. Maes, V. van der Leest, E. van der Sluis, and F. Willems, *Secure Key Generation from Biased PUFs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 517–534. [Online]. Available: https://doi.org/10.1007/978-3-662-48324-4_26

[114] G. T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs," in *CHES*, Sep. 2015.

[115] J. L. Carter and M. N. Wegman, "Universal Classes of Hash Functions (Extended Abstract)," in *Proc. ACM Symposium on Theory of Computing*, 1977.

[116] W. Che, F. Saqib, and J. Plusquellic, "PUF-based authentication," in *Proc. ICCAD*, Nov 2015.

[117] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and Reliable XOR Arbiter PUF Design: An Experimental Study Based on 1 Trillion Challenge Response Pair Measurements," in *Proc. DAC*, June 2017.

[118] J. Miao, M. Li, S. Roy, and B. Yu, "LRR-DPUF: Learning Resilient and Reliable Digital Physical Unclonable Function," in *Proc. ICCAD*, Nov 2016.

[119] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible?" in *CHES*, Sept 2014.

[120] M. Gao, K. Lai, and G. Qu, "A highly flexible ring oscillator PUF," in *Proc. DAC*, June 2014.

[121] T. Xu and M. Potkonjak, "Stable and secure delay-based physical unclonable functions using device aging," in *IEEE ISCAS*, May 2015.

[122] R. Liu, H. Wu, Y. Pang, H. Qian, and S. Yu, "A highly reliable and tamper-resistant RRAM PUF: Design and experimental validation," in *IEEE HOST*, May 2016, pp. 13–18.

[123] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *IEEE HOST*, May 2016.

[124] M. T. Rahman, D. Forte, Q. Shi, G. K. Contreras, and M. Tehranipoor, "CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly," in *IEEE International Symposium on DFT*, Oct 2014.

[125] M. Majzoobi and F. Koushanfar, "Time-Bounded Authentication of FPGAs," *IEEE TIFS*, Sept 2011.

[126] U. Ruhrmair, J. Solter, and F. Sehnke, "On the foundations of physical unclonable functions," *IACR Cryptology ePrint Archive*, 2009.

[127] V. D. Leest *et al.*, "Efficient Implementation of True Random Number Generator Based on SRAM PUFs," in *Cryptography and Security*, 2012.

[128] T. Good and M. Benaissa, "Hardware results for selected stream cipher candidates," in *SASC, Workshop Record*, 2007.

[129] Q. Ma *et al.*, "A machine learning attack resistant multi-PUF design on FPGA," in *ASP-DAC*, Jan 2018.

[130] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight Secure PUFs," in *Proc. ICCAD*, Nov 2008.

[131] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *IEEE WIFS*, Dec 2012.

[132] "TensorFlow," https://www.tensorflow.org.

[133] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE HOST*, May 2015.

[134] Y. Xie and A. Srivastava, "Mitigating SAT Attack on Logic Locking," in *CHES*, Aug 2016.

[135] T. Balyo and M. Heule, "Chbr_glucose," in *Proceedings of SAT Competition*, 2016.

[136] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. Morgan Kaufmann Publishers Inc., 2006.

[137] K. H. Kishore *et al.*, "Power and Area Efficient LFSR with Pulsed Latches," in *IJPAM*, 2017.

[138] P. Pessl and M. Hutter, "Pushing the Limits of SHA-3 Hardware Implementations to Fit on RFID," in *CHES*, Aug 2013.

[139] A. Bogdanov *et al.*, "PRESENT: An Ultra-Lightweight Block Cipher," in *CHES*, Sept 2007.

[140] Beaulieu *et al.*, "The SIMON and SPECK families of lightweight block ciphers," in *IACR eprint archive*, 2012. [Online]. Available: https://eprint.iacr.org/2013/404.pdf

[141] K. Shibutani *et al.*, "Piccolo: An Ultra-Lightweight Blockcipher," in *CHES*, Sept 2011.

[142] L. Xian, Z. Huicai, J. Cheng, and L. Xin, "A 4-kbit low-cost antifuse one-time programmable memory macrofor embedded applications," in *Journal of Semiconductors*, May 2014.

[143] K. Xiao *et al.*, "Hardware Trojans: Lessons Learned After One Decade of Research," *ACM TODAES*, May 2016.

[144] M. T. Rahman *et al.*, "CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly," in *IEEE International Symposium on DFT*, Oct 2014.

[145] S. Devadas *et al.*, "Design and Implementation of PUF-Based "Unclonable" RFID ICs for Anti-Counterfeiting and Security Applications," *IEEE International Conference on RFID*, April 2008.

[146] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in *CHES*, Sep. 2009.

[147] J. Valamehr *et al.*, "A 3-D Split Manufacturing Approach to Trustworthy System Development," *IEEE TCAD*, April 2013.

[148] M. Jagasivamani *et al.*, "Split-fabrication obfuscation: Metrics and techniques," in *IEEE HOST*, May 2014.

[149] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *IEEE HOST*, May 2015.

[150] K. V. others, "Building trusted ICs using split fabrication," in *IEEE HOST*, May 2014.

[151] Y. Xie, C. Bao, and A. Srivastava, "3D/2.5D IC-Based Obfuscation," in *Hardware Protection through Obfuscation*, Jan 2017.

[152] V. Sundaram *et al.*, "Low cost, high performance, and high reliability 2.5D silicon interposer," in *ECTC*, May 2013.

[153] Y. Sun *et al.*, "Modeling and fabrication of the redistribution layer on the 2.5D Si interposer," in *ICEPT*, Aug 2017.

[154] C. B. Yang Xie and A. Srivastava, "Security-Aware Design Flow for 2.5D IC Technology," *TrustED*, 2015.

[155] N. T. P100, "The Most Advanced Datacenter Accelerator Ever Built," in *NVIDIA White Paper*, 2016.

[156] Y. Xie, C. Bao, and A. Srivastava, "Security-Aware 2.5D Integrated Circuit Design Flow Against Hardware IP Piracy," *Computer*, May 2017.

[157] C. Yu *et al.*, "Incremental SAT-Based Reverse Engineering of Camouflaged Logic Circuits," *IEEE TCAD*, Oct 2017.

[158] L. de Moura and N. Bjrner, "Z3: an efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, April 2008.

[159] Y. Wang, P. Chen, J. Hu, and J. J. V. Rajendran, "The cat and mouse in split manufacturing," in *Proc. DAC*, June 2016.

[160] K. Cho, Y. Kim, H. Lee, H. Kim, S. Choi, J. Song, S. Kim, J. Park, S. Lee, and J. Kim, "Signal Integrity Design and Analysis of Silicon Interposer for GPU-Memory Channels in High-Bandwidth Memory Interface," *IEEE TCPMT*, Jan. 2018.