November 2015

Threat Analysis, Countermeaures and Design Strategies for Secure Computation in Nanometer CMOS Regime

Raghavan Kumar
University of Massachusetts - Amherst

THREAT ANALYSIS, COUNTERMEASURES AND DESIGN STRATEGIES FOR SECURE COMPUTATION IN NANOMETER CMOS REGIME

A Dissertation Presented

by

RAGHAVAN KUMAR

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2015

Electrical and Computer Engineering

THREAT ANALYSIS, COUNTERMEASURES AND DESIGN STRATEGIES FOR SECURE COMPUTATION IN NANOMETER CMOS REGIME

A Dissertation Presented

by

RAGHAVAN KUMAR

Approved as to style and content by:

Wayne P. Burleson, Chair

Sandip Kundu, Member

Christof Paar, Member

Hava T. Siegelmann, Member

Christopher V.Hollot, Department Head
Electrical and Computer Engineering

To my parents

ACKNOWLEDGMENTS

It is really an incredible feeling to think that I have reached this last phase of my graduate life. It is not about writing a chapter for my dissertation or submitting a journal, rather it is about framing the appropriate sentences for the *acknowledgments* section. This section might be *optional*, but it holds the key to how this document and in turn the entire work shaped up during the past few years.

First and foremost, I would like to thank my advisor Prof. Wayne Burleson for his constant guidance and support throughout my doctoral program. I am deeply indebted to him for providing me an opportunity to join his group when I was stuck in a not so noteworthy transition period. I appreciate him for allowing me to explore various research problems of my interest throughout my PhD curriculum. He also inspired me to actively collaborate with other research groups, keeping in mind the *significant* contribution made by individual authors in a particular project. I am immensely grateful to Prof. Sandip Kundu for supporting my profile to Prof. Wayne Burleson while he was looking for a fresh PhD candidate. He has always been a source of inspiration and knowledge throughout my stint at UMass. He has had a major constructive impact on some of the research projects presented in this dissertation. Special thanks go to Prof. Christof Paar and Prof. Hava Siegelmann for agreeing to serve on my dissertation committee and for their constructive insights and comments. I would also like to thank all the members of VLSI Circuits and Systems Group (VCSG) at UMass for creating a positive work atmosphere and also helping my transition as a PhD student during the initial days. My graduate life would have been utterly boring if not for the amazing friends I have made at Amherst. I will forever cherish the memories I had with the "Dunkin Coffee" group. Those long

walks to the various parking lots depending on the commute of the day will be sorely missed.

During my PhD program, I had a unique opportunity to spend five amazing months at University of Passau in the splendid Bavarian region of Germany. I thank Dr. Ilia Polian for accepting to host me as a "Gastwissenschaftler" in his group. A major portion of the work on parametric fault-injection attacks presented in this dissertation was done during my stint at Passau. Special mention goes to Dr. Phillip Jovanovic for being an admirable colleague and contributor/co-author during and after my stay at Passau. *Danke Schön*. Apart from my stint at Passau, I also spent three highly productive months at Intel Circuits Research Lab (CRL) as an intern. I take this moment to thank everyone at CRL for providing me an opportunity to apply my "academic" skill-sets for solving various state-of-the-art industry oriented research problems.

My entire graduate life wouldn't have been possible, if not for the constant love and support of my wonderful parents. They have absorbed all the immense pressure and troubles created by me when things were not particularly going well during my graduate life. Thanking them alone wouldn't do enough justice. This entire dissertation is dedicated to them.

ABSTRACT

THREAT ANALYSIS, COUNTERMEASURES AND DESIGN STRATEGIES FOR SECURE COMPUTATION IN NANOMETER CMOS REGIME

SEPTEMBER 2015

RAGHAVAN KUMAR

B.E., ANNA UNIVERSITY, CHENNAI, INDIA

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wayne P. Burleson

Advancements in CMOS technologies have led to an era of *Internet Of Things (IOT)*, where the devices have the ability to communicate with each other apart from their computational power. As more and more sensitive data is processed by embedded devices, the trend towards lightweight and efficient cryptographic primitives has gained significant momentum. Achieving a perfect security in silicon is extremely difficult, as the traditional cryptographic implementations are vulnerable to various active and passive attacks. There is also a threat in the form of "hardware Trojans" inserted into the supply chain by the untrusted third-party manufacturers for economic incentives. Apart from the threats in various forms, some of the embedded security applications such as random number generators (RNGs) suffer from the impacts of process variations and noise in nanometer CMOS. Despite their dis-

advantages, the random and unique nature of process variations can be exploited for generating unique identifiers and can be of tremendous use in embedded security.

In this dissertation, we explore techniques for precise fault-injection in cryptographic hardware based on voltage/temperature manipulation and hardware Trojan insertion. We demonstrate the effectiveness of these techniques by mounting fault attacks on state-of-the-art ciphers. Physically Unclonable Functions (PUFs) are novel cryptographic primitives for extracting secret keys from complex manufacturing variations in integrated circuits (ICs). We explore the vulnerabilities of some of the popular "strong" PUF architectures to modeling attacks using Machine Learning (ML) algorithms. The attacks use silicon data from a test chip manufactured in IBM 32nm silicon-on-insulator (SOI) technology. Attack results demonstrate that the majority of "strong" PUF architectures can be predicted to very high accuracies using limited training data. We also explore the techniques to exploit unreliable data from "strong" PUF architectures and effectively use them to improve the prediction accuracies of modeling attacks. Motivated by the vulnerabilities of existing PUF architectures, we present a novel modeling attack resistant PUF architecture based on non-linear computing elements. Post-silicon validation results are used to demonstrate the effectiveness of the non-linear PUF architecture against modeling and fault-injection attacks. Apart from the techniques to improve the security of PUF circuits, we also present novel solutions to improve the performance of PUF circuits from the perspectives of IC fabrication and system/protocol design. Finally, we present a statistical benchmark suite to evaluate PUFs in conceptualization phase and also to enable fine-grained security assessments for varying PUF parameters. Data compressibility analyses for validating the statistical benchmark suite are also presented.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Hardware Security and Vulnerabilities

The role of embedded systems in day-to-day life has improved significantly in the last decade and this trend will likely continue in the forthcoming days. Some major embedded systems include smartphones, tablets, payment systems, smart cards and medical devices. The trend of *ubiquitous computing* has therefore seen a great scope of improvement and has led to an era of *Internet of Things (IoT)*, where the devices have communication ability in addition to computation power. However, because of their ubiquitous nature, they also bring out security and privacy issues as they pose an ideal target for attackers. In particular, protection of sensitive data stored on the devices has brought forth an alarming issue and demands cryptographic protection.

Majority of the embedded systems pose tight constraints on area and energy because of their low cost. So, the trend towards lightweight cryptographic primitives is becoming extremely popular in the design market. Most of the primitives based on classical cryptography are based on the concept of a *secret binary key* embedded on the device. However, they pose some serious security vulnerabilities especially against physical attacks (invasive, non-invasive and side-channels) and software attacks. The fact that the key has to be stored in a non-volatile memory further aggravates the problem.

As explained above, cryptographic hardware blocks in safety and security critical systems increasingly constitute a target for attackers. *Fault-based attacks* [6, 10, 12] aim at determining the secret key or other protected data by actively manipulating

the system during operation and thus compromising the system integrity. Such approaches fall into the category of *active side-channel cryptanalysis*, in order to distinguish them from passive techniques that derive the secret information from measured operational parameters such as power consumption or timing [40, 63]. A number of fault-based attacks have been proposed in the recent years and were successful in breaking state-of-the-art ciphers using one or a small number of faults [29, 34, 37, 56]. Consequently, current and emerging cryptographic circuits must be capable of withstanding such attacks. In order to design appropriate countermeasures, it is important to understand the attacks, their limits, and the criteria for their success.

The attacks from the latest generation mentioned above require a very high precision on the fault injection. One part of this requirement is *spatial resolution*: the fault must show up in the desired locations (memory cells, registers or logic gates) while not affecting other locations. The second relevant property is the *temporal resolution*: the fault must be present in a given point of time (e. g., after the end of a specific round of encryption) and absent at other times. For example, the one-fault-injection-attack on AES-128 [56] identifies a set of secret key candidates which is sufficiently small for practical brute-force search under the condition that the fault affects one or multiple bits of one byte of the cipher state after round 8. If multiple bytes of this state or bytes during rounds other than 8 are affected by the fault injection, the mathematical analysis will loose validity and the correct secret key will no longer be found.

The physical techniques to inject faults can be broadly divided into low-cost, low-precision and high-cost, high-precision approaches [6]. Here, the term "cost" refers to the necessary equipment as well as to the qualification of its operator. Low-cost, low-precision fault-injection methods include operating the device under a reduced power supply (underfeeding), tampering with the clock signal (introducing a glitch), overheating the device, irradiating it with X-rays, ion beams, white or ultraviolet

light and may or may not involve de-packaging the circuit in order to expose the active areas of its transistor. These techniques typically do not achieve good spatial and temporal resolutions at the same time. They are suited for attacks such as manipulating the round counter of a cipher in order to reduce the number of rounds applied during encryption, or manipulating the program counter of a microprocessor in order to jump over certain instructions. Methods with high or very high spatial and temporal resolution include laser irradiation, precise application of electromagnetic pulses[1], and the use of focused-ion beam.

In this work, we suggest two techniques for fault injection that do not require elaborate equipment while providing sufficient precision for attacks of the latest generation. The first technique is based on careful selection of the parameters under which the circuit is operated: power supply voltage V_{dd} and temperature T. We search for V_{dd}/T combinations which lead to injection of faults that satisfy the requirements for the cryptanalysis. In contrast, earlier approaches produced faults of low precision and predictability and were not suited for latest-generation attacks. We demonstrate this technique, called V/T fault injection, using a complex two-stage attack on two recent lightweight block ciphers.

The second fault-injection technique employs *hardware Trojans* to facilitate fault injection. Hardware Trojans [87] are malicious modifications of a circuit unintended by its designer; they can be applied by the untrusted manufacturer or a third-party intellectual property block provider. Here, we introduce a new class of hardware Trojans, called *MAnufacturing Process LEvel*, or MAPLE Trojans. These Trojans are applied to individual logic gates of the circuit and are activated non-deterministically, with a probability (called (*triggering factor*) being sufficiently high to conduct fault-

[1]EM techniques are quoted in [6] under the low-cost category. However, recent results suggest that they can be used for high-precision attacks [93].

based cryptanalysis, but sufficiently low to make their detection extremely challenging or even impossible in practice.

We evaluate both fault-injection methods using the same settings and compare them with each other and with previously introduced fault-injection techniques. In particular, we demonstrate that both methods are applicable for successful fault-based cryptanalysis of state-of-the-art ciphers. We discuss in-depth the effectiveness of known countermeasures against the proposed fault-injection methods and outline the key differences to earlier threats. In particular, we show that the post-manufacturing detection methods that traditionally have been considered as a remedy against hardware Trojans are of little use against MAPLE Trojans. While we have no indication of actual circuits having been manipulated in this way, the theoretical existence of the threats outlined by our work suggests the need to re-think protection of safety- and security-relevant hardware components.

The vulnerabilities of existing cryptographic hardware blocks have been one of the major driving forces behind the search of novel, efficient and secure cryptographic primitives. Moreover, the ever-shrinking transistor sizes has resulted in tiny embedded devices with computing and networking power. Some examples include radio-frequency identifiers, smart cards, PDAs, etc. As these devices store and process sensitive data, they demand efficient and lightweight cryptographic protection. The traditional cryptographic implementations are based on a *digital secret key* stored in the device. As some of these embedded devices are extremely resource constrained, storing a *secret key* in a non-volatile memory might be extremely expensive. Motivated by these drawbacks, Physically Unclonable Functions (PUFs) have been proposed in the literature as an efficient way of generating unique and secret identifiers from the complex and unpredictable nature of silicon.

1.2 Exploiting Process Variations in Security

The semiconductor industry has been continuously driven by Moore's law, which states that the number of transistors in an integrated circuit doubles approximately every two years. The trend is shown in Figure 1.1. Technology scaling has been the dominant force behind Moore's law. However, aggressive technology scaling is also impacted by variations from process, voltage, temperature and aging to certain extent, also known as PVTA variations. Significant effort is spent by designers to combat the variations, especially in complex systems consisting of billions of transistors. Apart from these variations, the semiconductor industry is also facing non-certainties in the form of scaling limits. It has been predicted that the technology scaling can continue up to around 7nm, below which the number of transistors in the channel region will not be sufficient to generate enough drain current. Researchers are exploring various post-CMOS devices like nano-wires, graphene, carbon nanotubes, etc.

Managing and mitigating process variations in integrated circuits have been extensively explored, especially in sub-nm design era. Random dopant fluctuation (RDF) is one of the major sources of process variations. RDF results in a variation in the number of dopant atoms in the channel region. This in turn changes the threshold voltage of a transistor. Due to the impact of process variations, logic gates suffer from delay variations. Process variations also impact the performance of memory elements such as Static Random Access Memory (SRAM). As the amount of process variations increase with reducing transistor sizes, significant effort is spent on managing them. However, there are also possibilities to utilize increasing process variations in a constructive fashion. Such techniques are gaining popularity in semiconductor industry, where the process variations are used to generate unique signatures in integrated circuits [66, 80]. These circuits are often referred to as Physically Unclonable Functions (PUFs). The concept of PUFs were first introduced in [86].

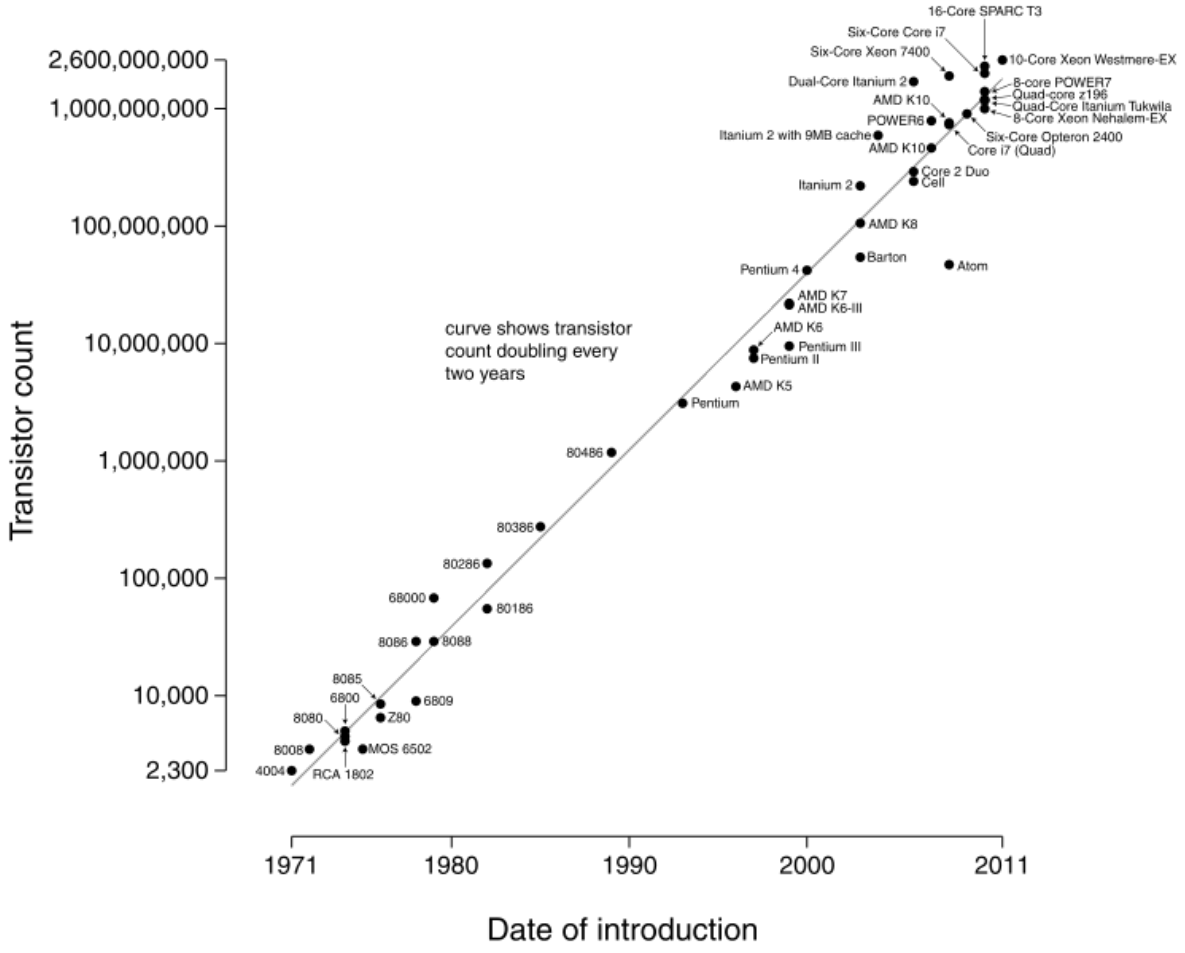


Figure 1.1. Moore's law [68]

A PUF is a partially disordered system that maps a set of external inputs also known as challenges C to a response R. A challenge associated with its response is known as a challenge-response pair (CRP). In silicon PUFs, the mapping function is decided by process variations arising during the manufacturing process. The manufacturing process is extremely complicated especially in sub-45 nm design space and is hard to control even by the manufacturer. This ensures that a manufacturer cannot produce an identical tuple of ICs with the same layout. This behavior is exploited in the design of PUF circuits. PUFs can be employed in several security related applications and the scope is often limited by the number of responses that can be

generated. PUFs can be broadly classified into "strong" and "weak" PUFs based on the number of independent responses that it can generate. Please note that the terms "strong" and "weak" do not have any reference to the security level of a PUF. A strong PUF can produce a large number of responses (R_i) for different inputs C_i and can be used in security protocols, key establishment and device authentication. A classical example for a strong PUF is an arbiter PUF [86]. A weak PUF, on the other hand, has a reduced response space and produce only a single response in the worst case scenario. So, the response(s) must be kept secure from the external world and must never be shared with a third party. The weak PUFs can be used in classical crypto-systems for deriving the secret key. One of the typical examples for a weak PUF is an SRAM PUF [30, 31].

As PUFs find strong potential for deployment in security systems, they must satisfy some properties. Some of the security properties of a strong PUF as described in [78] are: (i) Cloning a strong PUF is highly impossible. (ii) Entire CRP collection by an attacker within a short amount of time is impossible. (iii) A subset of CRPs should not leak any information to predict the response of a challenge outside the subset. These properties have been exploited in the literature for the development of various security protocols based on PUFs. Some examples include device identification [86], key exchange [73], oblivious transfer [76] etc. The commercial applications employing PUF circuits require that any two responses from two different PUF instances of the same type should have a significant difference. This property of PUFs is referred to as uniqueness. To ensure stable authentication, PUFs are expected to produce the same response for a challenge under any operating condition, which is measured in terms of reliability. Finally, a PUF should be unpredictable such that an attacker possessing a subset of CRP pairs should be unable to predict the response for a challenge outside that subset. More details on performance metrics of PUF circuits can be found in chapter 2.

Although PUFs seem to be a promising alternative to classical cryptography, they must overcome some concerns to be trusted fully secure. One such concern is the robustness of PUF circuits. A PUF is expected to produce the same response whenever queried with a particular challenge. However, due to the presence of on-chip substrate noise, temperature and voltage fluctuations, some of the responses become highly unreliable. Several circuit and system level schemes to improve the reliability of PUF circuits have been proposed in the literature. Prominent examples include error-correction schemes [92], feedback-supply control [45, 49, 89], helper data [14], fuzzy extractors [19], etc. Another major concern for PUFs is their vulnerability to modeling attacks. Moreover, unreliable challenge-response pairs can also be used to improve the performance of modeling attacks. The unreliable challenge-response pairs can leak some side-channel information to extract more data-dependent information from PUF circuits. So, unreliability and security vulnerabilities of PUF circuits are closely related to each other and must be addressed in order to consider them fully secure.

In this work, we study the vulnerabilities of some popular "strong" PUF architectures to various attacks using simulated and silicon data. The PUF circuits were implemented in IBM 32nm Silicon-on-Insulator (SOI) technology and validated using post-silicon measurements. In particular, we study the vulnerabilities of PUF circuits to modeling attacks using Machine Learning (ML) algorithms. The ML algorithms construct a model based on the challenge-response training set. The performance of modeling attacks depends on the robustness of the challenge-response pairs. If there are some unreliable challenge-response pairs in the training set, the learning phase of the ML algorithms is severely impacted and limited prediction accuracies are achieved. To that extent, we propose a hybrid attack that exploits the data-dependent information present in unreliable challenge-response pair and uses it constructively to push the prediction accuracies achieved by ML algorithms. Motivated by the vulnera-

bilities of existing PUF circuits, we also present a modeling attack resistant PUF architecture using non-linear current mirrors. The post-silicon validation results of the proposed non-linear PUF architecture are also presented. Finally, a statistical benchmark suite to evaluate and enable fine-grained security assessments of PUF architectures is presented.

1.3 Contributions and Organization

The major contributions of this dissertation include:

1. Techniques to extract secret keys from cryptographic hardware through the use of hardware Trojans and precise voltage and temperature manipulation.
2. Techniques to attack delay-based PUF designs using modeling attacks.
3. Methodologies for exploiting side-channel information to improve the performance of modeling attacks.
4. Design and post-silicon validation of a modeling attack resistant PUF design in 32nm SOI process.
5. Design strategies to improve the performance of PUF circuits.
6. Statistical benchmark suite for analysis of PUF architectures.

The rest of this dissertation is organized as follows. In chapter 2, some background information on various sources of process variations is presented. We also discuss the different performance metrics used to characterize PUF circuits. In chapter 3, we present the different techniques to extract the secret key from cryptographic blocks. Different techniques include voltage and temperature manipulation and hardware Trojan insertion. We also present the performance of fault-injection techniques by using them to extract the secret key from state-of-the-art ciphers. Some of the

countermeasures against fault-injection attacks are also presented. Chapter 4 focuses on modeling and hybrid attacks on popular delay-based PUF designs. Simulation and silicon data from PUF circuits are used to validate the performance of modeling and hybrid attacks. In chapter 5, we present the design and post-silicon measurements of a novel modeling attack resistant PUF design based on non-linear current mirrors. The performance of the proposed PUF is compared against the best-in-class current-based PUF architecture and the results are presented. Chapter 6 discuss the different design strategies to improve the performance metrics of PUF circuits. The different strategies include fabrication-aware design of PUF circuits and a PUF based protocol to improve the authentication capabilities of security systems. In chapter 7, we present the architecture of the 32nm test chip known as *sugarloaf* and the post-silicon validation setup. Finally, chapter 8 discusses the statistical benchmark suite and data compressibility analysis of PUFs.

CHAPTER 2

BACKGROUND

Some background information related to PUFs are presented in this chapter. Along with some background information, the methodology to compute the performance metrics of a PUF circuit are also presented. The methodology presented is the basis for all performance metrics computation described in the document.

2.1 Sources of CMOS Process Variations

The sources of process variations in ICs are summarized in this section. Some of the sources of variations are shown in Figure 2.1. From the perspective of PUF circuits, the sources of variations can be either *desirable* or *undesirable*. The desirable source of variations refers to process manufacturing variations (PMV) as identified in [39]. The environmental variations and aging are undesirable for PUF circuits.

2.1.1 Manufacturing Process and Variations

The IC manufacturing process consists of several steps [57]: patterning, etching, doping, film deposition and planarization. A monochromatic light source is focused via a set of optical lenses on a mask containing the desired pattern to be printed onto the silicon wafer. After the wafer is exposed, some parts are etched out through a chemical process and then the surface is planarized using Chemical Mechanical Polishing (CMP) [57]. This process is repeated several times over for printing the patterns. Moreover, diffusion/ion implantation is used to dope certain regions of the wafer. Each of these steps cannot be repeated faithfully from wafer to wafer

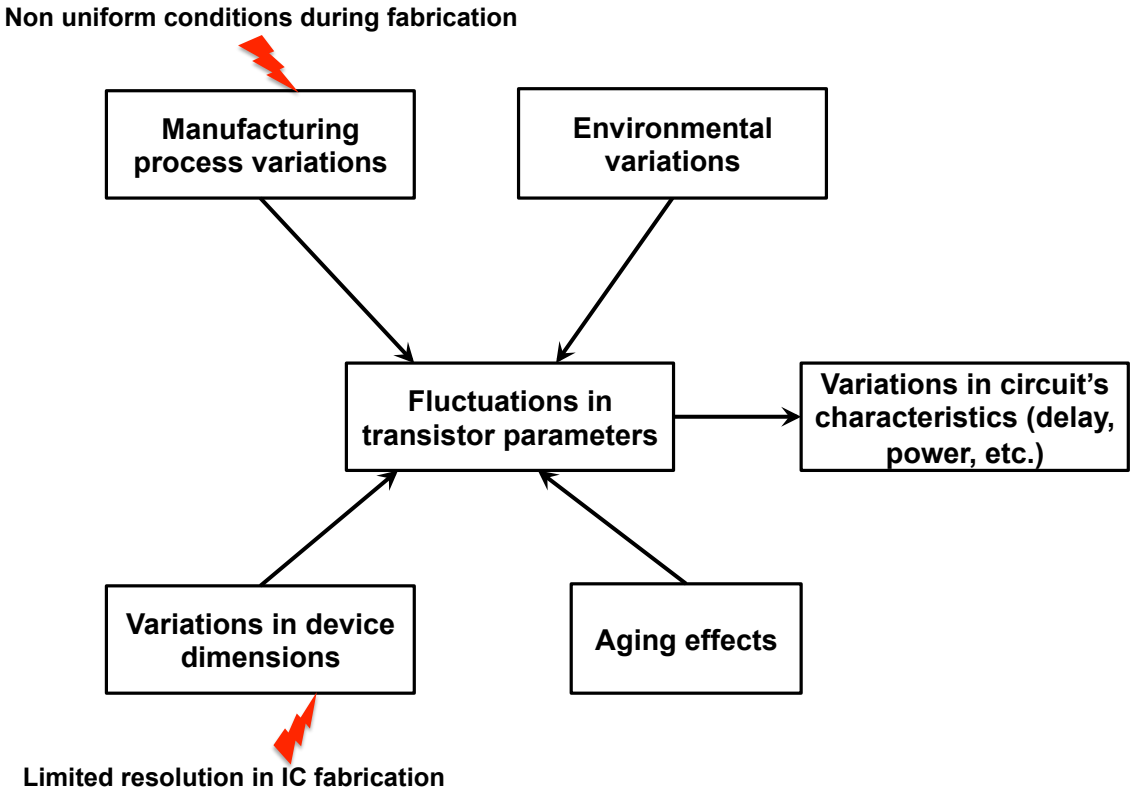


Figure 2.1. Sources of variations in ICs

or even from die to die within the same wafer. Owing to lens material limitations, the wavelength of the light source has not scaled down below 193nm, though the transistor dimensions have scaled down to 22nm. This is also a significant contributor to variations in structures printed on the wafer. Variations occur due to imperfections in:

- Light source - exposure intensity (dose)
- Lens system - aberration
- Mask
- Etching process - Line Edge Roughness (LER)

- Doping process
- CMP process
- Alignment - defocus
- Optical Proximity Effects

These imperfections result in variations in physical parameters that lead to variations in electrical parameters like threshold voltage and current. This in turn affects timing, power consumption, etc. These variations can be random or systematic [57]. Systematic variations should be suppressed as they affect the uniqueness of PUFs. On the other hand, random variations are unpredictable and improve the performance of a PUF.

2.1.1.1 Systematic Variations

Systematic component of process variations includes variations in lithography system, nature of layout and CMP [11]. By performing a detailed analysis of the layout, the systematic sources of variations can be predicted in advance and accounted in design step. If the layout is not available for analysis, the variations can be assigned statistically [11].

2.1.1.2 Random Variations

Random variations refer to non-deterministic sources of variations. Some of the random variations include random dopant fluctuations (RDF), line edge roughness (LER) and oxide thickness variations. The random variations are often modeled using random variables for design and analysis purposes.

2.1.2 Environmental Variations and Aging

Environmental variations are detrimental to PUF circuits. Some of the common environmental sources of variations include power supply noise, temperature fluc-

tuations and external noise. These variations must be minimized to improve the reliability of PUF circuits.

Aging is a slow process and it reduces the frequency of operation of circuits by slowing them down. Circuits are also subjected to increased power consumption and functional errors due to aging [90].

2.2 PUF Terminologies and Performance Metrics

The terminologies and performance metrics used in the field to evaluate PUF devices are briefly summarized in the following sections.

2.2.1 Challenge-Response pairs (CRP)

As the name suggests, PUF circuits can be envisioned as a function mapping a set of inputs to outputs. However, as identified in [58], PUF circuits do not implement a true function as they can produce different outputs for an input under different operating conditions. The inputs to a PUF circuit are known as *challenges* and outputs are referred to as *responses*. A challenge associated with its corresponding response is known as a *Challenge-Response pair (CRP)*. In an application scenario, responses of a PUF circuit are collected and stored in a database. This process is generally known as *enrollment*. Under *verification* or *authentication* process, the PUF circuit is queried with a challenge from the database. The response is then compared against the one stored in the database. If the responses match, the device is authenticated.

2.2.2 Performance Metrics

There are different important metrics used to analyze a PUF circuit, namely uniqueness, reliability, unpredictability, uniformity and bit-aliasing probability [60]. The performance metrics are defined in different measurement dimensions as shown in Figure 2.2. The different measurement dimensions are time, device and space.

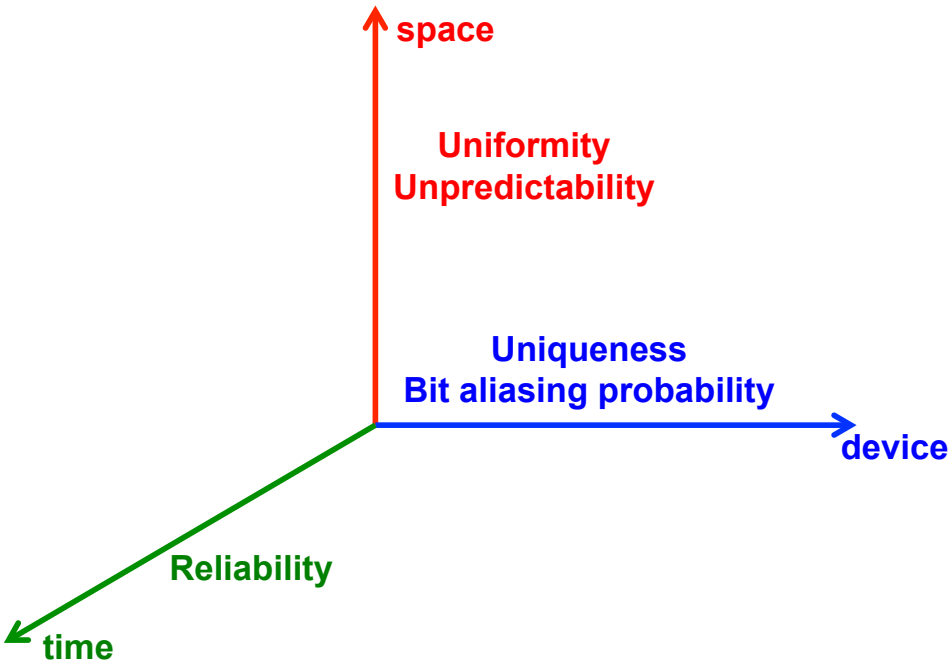


Figure 2.2. PUF Performance metrics and dimensions

Uniqueness and bit-aliasing probability are measured across different devices. Reliability is measured across time and uniformity and unpredictability are measured across space.

2.2.2.1 Uniqueness

PUF devices are primarily used to generate unique signatures for device authentication. In this application, it is desirable to have a large difference between responses from any two PUF instances. Here, the two PUF instances may be from the same wafer or different wafers. A typical measure used to analyze uniqueness is known as *inter-distance* and is given by [39, 60]:

$$d_{inter}(C) = \frac{2}{k(k-1)} \sum_{i=1}^{i=k-1} \sum_{j=i+1}^{j=k} \frac{\mathrm{HD}(R_i, R_j)}{m} \text{ x } 100\%. \tag{2.1}$$

In equation 2.1, $\mathrm{HD}(R_i, R_j)$ is the Hamming distance between two responses R_i and R_j of m bits long for a particular challenge C and k is the number of PUF

instances under consideration. The desired inter-distance is 50%. By carefully looking at equation 2.1, one can correspond the inter-distance $d_{inter}(C)$ to the mean of the Hamming distance distribution obtained over k chips for a challenge C. It is also useful to obtain the standard deviation of Hamming distance distribution given by $\sigma_{inter}(C)$, which measures the extent of deviation in Hamming distance from the desired inter-distance. Lower $\sigma_{inter}(C)$ is preferable for PUF design. As uniqueness is measured across devices, it is denoted in the device axis in Figure 2.2.

While designing a PUF circuit, inter-distance is often measured through circuit simulations. A common practice is to perform Monte Carlo simulations over a large population of PUF instances. Though there is no single concrete number for the number of PUF instances to be considered for simulation purposes, it is safe to assume around 1000 samples to obtain a good estimate of uniqueness. In simulations, care must be taken to efficiently model various sources of manufacturing variations in CMOS circuits, as they directly translate into uniqueness. During the simulations, manufacturing variations are modeled using a gaussian distribution. In such cases, mean and standard deviation of the gaussian distribution under consideration must correspond to either inter-die or inter-wafer variations' statistics.

2.2.2.2 Reliability

A challenge applied to a PUF operating on an integrated circuit will not necessarily produce the same response under different operating conditions as the circuit is subject to environmental variations. The robustness of PUF's responses is measured in terms of reliability. Reliability of a PUF refers to its ability to produce the same response for a particular challenge under varying operating conditions. Reliability can be measured by looking at the average number of flipped bits in responses for the same challenge under different operating conditions. A common measure of reliability is *intra-distance* given by [24, 39, 60]:

$$d_{intra}(C) = \frac{1}{s}\sum_{j=1}^{s}\frac{\mathrm{HD}(R_i, R'_{i,j})}{m} \text{ x } 100\%. \tag{2.2}$$

In equation 2.2, R_i is the response of a PUF to challenge C under nominal conditions, s is the number of samples of response R_i obtained at different operating conditions, $R'_{i,j}$ corresponds to j^{th} sample of response R_i for challenge C and m is the number of bits in the response. Intra-distance is expected to be 0% for ideal PUFs, which corresponds to 100% reliability. The terms intra-distance (d_{intra}) and reliability have been used interchangeably further in this chapter. Given d_{intra}, reliability can always be computed ($100 - d_{intra}(\%)$).

2.2.2.3 Unpredictability

Responses from a PUF circuit must be unpredictable in order to ensure that the signatures/keys are safe from adversaries possessing information about the responses to different challenges from the same device. One of the measures of unpredictability is the amount of randomness in responses from the same PUF device. This can be evaluated using NIST tests [24, 58]. Silicon PUFs produce unique responses based on intrinsic process variations, that are very difficult to clone or duplicate by the manufacturer. However, by measuring responses from a PUF device for a subset of challenges, it is possible to create a model that can mimic the PUF under consideration. Several modeling attacks on PUF circuits have been proposed in the literature [32, 51, 53, 54, 62, 78]. The type of modeling attack depends on the PUF circuit. A successful modeling attack on a PUF implementation may not be effective for other PUF implementations. Modeling attacks can be made harder by employing some control logic surrounding the PUF block, that prevents direct read-out of its responses. One such technique is to use a secure one-way hash over PUF responses. However, if PUF responses are noisy and have significant intra-distance, this technique will require some sort of error-correction on PUF responses prior to hashing [92].

2.2.2.4 Uniformity

Uniformity is a measure of the proportion of '0's and '1's in a PUF's k bit response. In other words, the number of challenges that produce '0's and '1's must be equal in an ideal scenario. Uniformity of a PUF is evaluated using,

$$(\text{Uniformity})_i = \frac{1}{k}\sum_{j=1}^{k} r_{i,j} \text{ x } 100\% \tag{2.3}$$

where $r_{i,j}$ is the j^{th} bit of a k bit response from chip i. As uniformity is measured across the k-bit response, it is denoted in the space axis in Figure 2.2.

2.2.2.5 Bit-aliasing probability

If bit-aliasing happens in a PUF circuit, then different chips may produce nearly identical responses. Bit-aliasing of the i-th bit in PUF response is given by,

$$(\text{Bit-aliasing})_j = \frac{1}{n}\sum_{i=1}^{n} r_{i,j} \text{ x } 100\% \tag{2.4}$$

where $r_{i,j}$ is the j-th bit of a k bit response from chip i. As bit-aliasing is measured across n devices, it is denoted in the device axis in Figure 2.2.

2.3 PUF Design Flow

This section focuses on a generalized design flow adopted for any type of PUF instantiation using standard CMOS gates. PUF design flow includes two broad steps, namely

- Pre-fabrication phase and
- Post-fabrication phase

We describe pre-fabrication phase in section 2.3.1 and post-fabrication phase in section 2.3.2.

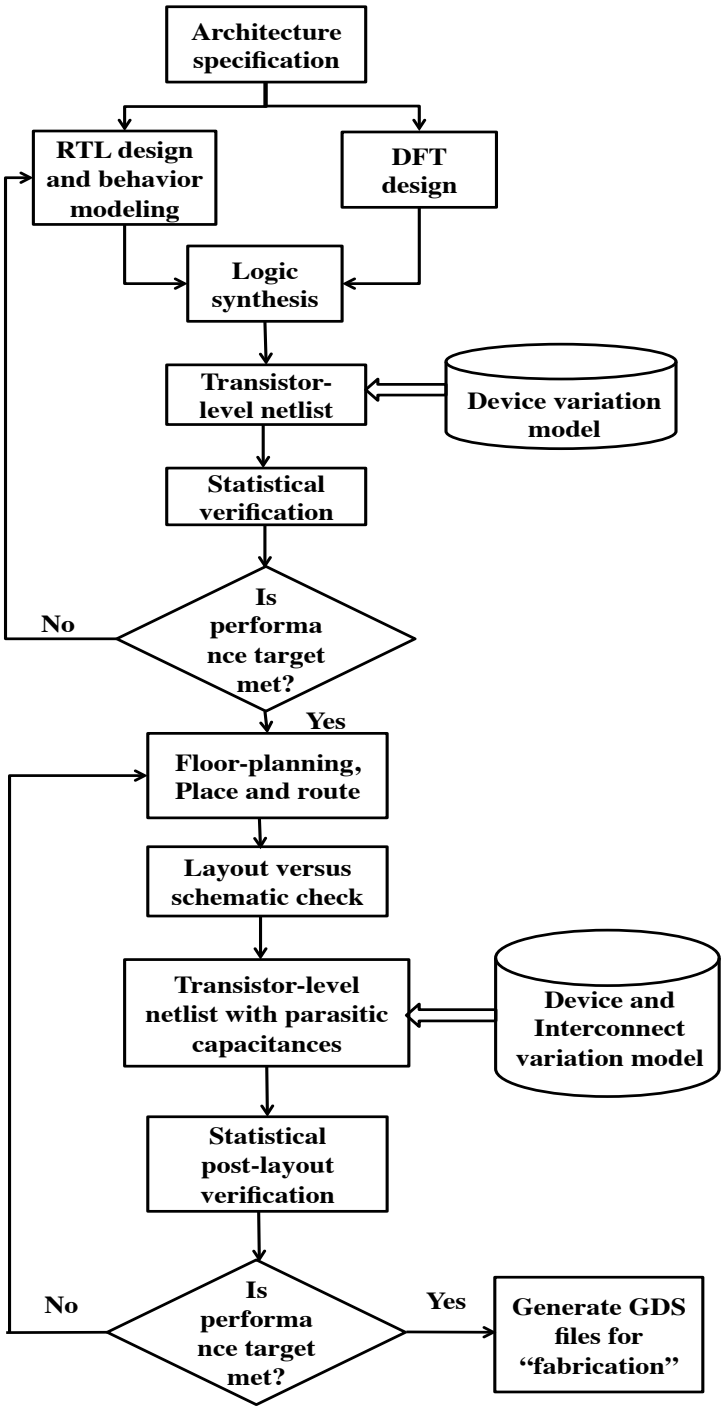


Figure 2.3. PUF design flow

2.3.1 Pre-fabrication phase

Any PUF design flow is built upon traditional ASIC design flow, with some specific steps for variation modeling and statistical verification. A complete PUF design flow is shown in Figure 2.3. As shown in Figure 2.3, PUF design starts from the architecture specification by defining general design goals and performance requirements. This is followed by a register-transfer level (RTL) design for the PUF circuit under consideration using Verilog and/or VHDL. Together with design for testing (DFT) techniques, a logic synthesis procedure is performed upon the developed RTL code to generate a gate-level netlist along with sub-circuit modules to describe the transistor-level netlist. A PUF-specific design process of introducing device variation models is important to enable statistical verification of PUF metrics such as uniqueness and reliability. If this verification fails, i.e., the performance metrics target of the PUF under consideration is not met, then the process to this step should be repeated with modifications to improve the PUF metrics. Upon statistical verification failure, a common place in the design process to inspect is the standard cell library. Often, the standard cell libraries are optimized to tolerate process variations. In such cases, standard cell libraries specific for PUF design process must be built. If the verification process succeeds, the design process continues further to physical design phase including floor-planning, placement and routing and layout-versus-schematic verification. Once the physical model of PUF circuit is available, the more realistic device and variation models with extracted parasitics can be used for post-layout statistical verification. If this final verification fails, either the layout or the original design has to be modified. A successful verification at this phase should result in the delivery of GDSII files for fabrication.

To introduce variations to a specific device parameter during the verification phase, Monte Carlo (MC) based approach can be used to instantiate random values from a normal distribution or from a post-silicon parameter variation profile available

through chip measurements. Each MC iteration produces one PUF instance. Usually, a large number of PUF instances (MC iterations) are required for uniqueness validation process. Supply voltage and temperature fluctuations are assigned over the netlist with extracted parasitic capacitances and simulations are performed for reliability and security/unpredictability analysis. Note that the focus of verification should be the post-layout stage and sufficient number of MC iterations should be performed, while the verification of pre-layout netlist can have reduced number of MC iterations.

2.3.2 Post-fabrication phase

Once the pre-fabrication phase is complete, the generated GDS files are sent for fabrication. The fabricated PUF circuit must be tested for exact real time perfor-

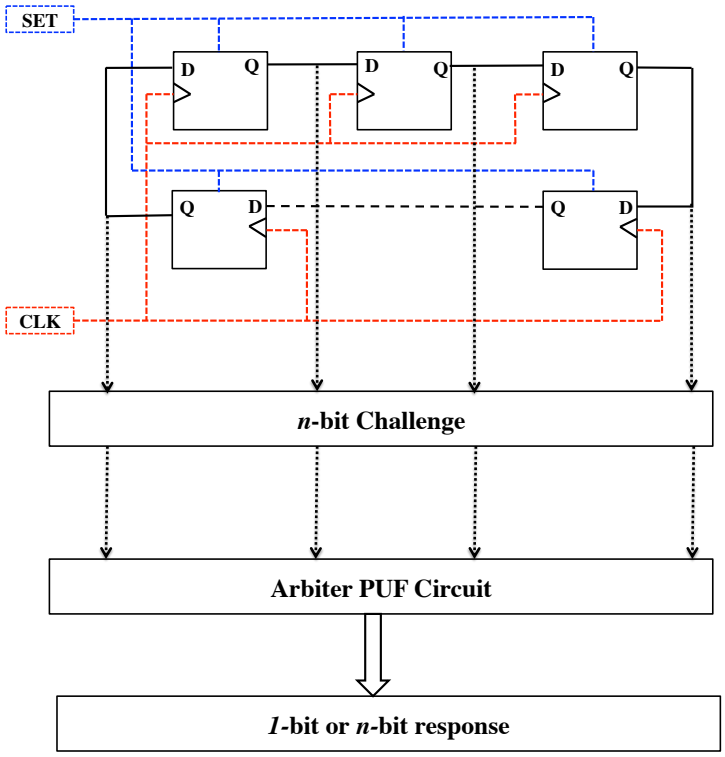


Figure 2.4. Arbiter PUF circuit with on-chip LFSR for challenge generation

mance analysis. The challenges for the PUF circuit under consideration can be generated using an external equipment or generated inside the chip using a pseudo-random number generator like Linear Feedback Shift Register (LFSR), as shown in Figure 2.4. Although an arbiter PUF circuit is shown, any "strong" PUF implementation can be used. LFSR will start generating pseudo-random challenges, when *set* signal is pulled high. Usually post-silicon validation of PUF is done by storing the waveform of responses along with challenges. The CRPs are then extracted from the waveform files. The CRP extraction methods for pre-silicon and post-silicon validation are almost similar in nature. The only minor concern is that pre-silicon waveforms have accurate time reference and post-silicon measurements may have some time uncertainties due to clock jitter and external noise. Hence, automatic data synchronization must be employed to use the same CRP extraction method for pre- and post-silicon validations.

CHAPTER 3

PARAMETRIC FAULT INJECTION ATTACKS ON CRYPTOGRAPHIC HARDWARE

In this chapter, we present the techniques to extract secret keys from cryptographic blocks. The first technique is based on voltage and temperature manipulation such that precise faults are injected. By carefully manipulating the V_{dd} and T values, precise faults up to single bit precision can be injected[1]. This technique is referred to as V/T injection further in the document. The second technique is based on *harware Trojans* insertion to facilitate transient fault injection. The proposed hardware Trojans, also known as MAPLE (MAnufacturing Process LEvel) Trojans, are based on manipulating the electrical characteristics of logic gates, such that non-deterministic fault injection is achieved[2]. The triggering probability of MAPLE Trojans is sufficiently low to mask their detectability. The techniques are validated and compared using the same settings to enable fair comparison. In particular, we demonstrate the effectiveness of the proposed techniques by employing them to break state-of-the-art ciphers.

The remainder of this chapter is organized as follows. The next section provides background on fault-injection techniques and hardware Trojans. The V/T injection and the MAPLE Trojan-based injection are described in detail in Sections 3.2 and 3.3, respectively. Attacks on several cryptographic circuits using both fault-injection techniques are reported in Section 3.4.

---

[1]This work was published in [48]

[2]This work was published in [47]

Table 3.1. Comparison of Fault-injection Techniques

	Method	Properties			Countermeasures				
		Effort	No. of attempts	Foundry-side attacker	Light sensors, shielding	Voltage drop sensors	Fixed V_{dd}	Concurrent error-detection	Frequent key regeneration
Low-precision	V_{dd} manipulation	Low	Small	No	No	Yes	Yes	Yes	Limited
	Clock manipulation	Low	Small	No	No	No	No	Yes	Limited
	Overheating	Low	Medium	No	No	Potentially	No	Yes	Somewhat
	X-rays, ion beam	Medium	Small	No	Limited	Yes	No	Yes	Limited
	White / UV light	Medium	Large	No	Yes	Yes	No	Yes	Yes
High-pr.	Laser	High	Small	No	Yes	Yes	No	Yes	Limited
	Electromagnetic	High	Small	No	Somewhat	Yes	No	Yes	Limited
	Focused ion beam	Very high	Small	No	Yes	Yes	No	If functional	Limited
New	V/T manipulation	Low-medium	Large	No	No	Unlikely	Yes	Yes	Yes
	Trojan-assisted	Low	Medium-large	Yes	No	Unlikely	Yes	Yes	Yes

3.1 Background

This section presents an overview on fault injection techniques[3] in order to relate them to the methods introduced in this chapter. Moreover, since one of the techniques is based on hardware Trojans, we also provide background information on this class of threats to hardware security in Section 3.1.2.

3.1.1 Fault Injection Techniques

Methods to inject faults into digital circuits with the aim of fault-based cryptanalysis are summarized in Table 3.1. Low-cost techniques [4, 6] include manipulation of power-supply voltage V_{dd}, manipulation of the clock signal and overheating of the device. All these techniques do not require elaborate equipment or a cooperating attacker at the foundry which manufactures the circuit. They are usually effective in introducing faults quickly but are not very good in controlling the location and the time of the injected faults.

Since the techniques in this chapter involve V_{dd} manipulation, we discuss earlier related approaches in more detail. Voltage manipulation based attacks can involve either introducing a well-timed glitch into the power supply or under-powering the device. Glitch based fault injection alters the state of the latches or flip-flops, thereby

[3]It is important to distinguish malicious fault injections for cryptanalysis from fault-injection campaigns run on circuit models in order to study their resilience to transient faults [3, 55], which are not focused in this chapter.

affecting the control and data path logic of the circuit. A successful attack on a RSA implementation using this technique was proposed in [81], and a similar attack on AES implemented in an SRAM-based FPGA was demonstrated in [15]. In underpowering attacks, the propagation delays of some combinational gates may increase and lead to timing errors. This can cause a flip-flop to capture the erroneous value. One such attack to break a smart card implementation of AES was shown in [82]. All these attacks do not require high precision.

Two further low-precision techniques from Table 3.1 use X-rays/ion beam or illuminating by strong ultraviolet or white light. These approaches require some equipment and may necessitate de-processing of the circuit. The expected time until first errors show up depends on the energy of the used source, but tends to be rather long for light-based methods.

High-precision methods that can target individual circuit structures are laser light, electromagnetic emissions and focused ion beam. They generally require de-processing of the circuit, expensive or very expensive equipment and skilled operators. Furthermore, identifying the circuit structure for fault injection requires either knowledge about the circuit layout and the correspondence of layout locations to specific variables in the cryptographic algorithm, or the ability to derive such correspondence by reverse engineering.

The techniques introduced in this chapter, are associated with high precision and low effort. V/T manipulation, explained in Section 3.2, requires some effort to determine the voltage and the temperature which are optimal for fault injection, and also some equipment to accurately control the temperature. The Trojan-assisted technique from Section 3.3 of this chapter relies, as the sole technique in Table 3.1, on a cooperating attacker involved in the manufacturing process. However, once the manipulated circuit has been produced, fault injection is easy. Both techniques are

probabilistic: not every run of the circuit will result in a fault, and not all injected faults are exploitable for cryptanalysis.

The rightmost five columns of Table 3.1 summarize relevant countermeasures against malicious fault injections and their effectiveness. One class of countermeasures aims at preventing access to critical circuit structures or identifying attempts to open the package to gain such access. These methods are effective against optical fault injection and may be effective against further methods (X-rays, ion beam, EM) if they require de-packaging, but not against manipulation of voltage, clock frequency or temperature. Voltage droop sensors detect significant deviations of V_{dd} at the sensed node(s) from their nominal values. They will identify direct and substantial manipulation of power supply voltage and should be rather effective against methods that induce parasitic currents, including X-rays, ion beams, white/UV/laser light, EM, and, to some extent, overheating. The proposed injection methods do alter V_{dd}, but the extent of manipulation is limited (within 10–20% of the nominal value). While it is possible to deploy sensors that will detect such small deviations, they will frequently trigger false alarms because these values are regularly observed in normal operation (in absence of any fault injection) due to power-supply noise [74].

If the circuit is equipped with circuitry that fixes V_{dd} at its specified value and prevents its change, all fault-injections involving voltage, including the proposed techniques, are thwarted. It may be possible to circumvent this protection, but this will necessitate opening the package. Concurrent error-detection (by duplication and comparison or by employing error-detecting codes) is generally effective against faults due to arbitrary causes, including malicious fault injections. However, the cost of these schemes is often prohibitive (100% and more in area and power consumption) and not all faults are detected. Using focused ion beam, it is possible to deactivate the error-detecting circuitry.

The final countermeasure in Table 3.1 is on protocol level: the key is frequently exchanged. This approach is effective against fault injections which require a large number of clock cycles, including the proposed techniques. Successful cryptanalysis may require multiple exploitable fault injections with the same secret key, and the key may loose validity before enough exploitable faults have been injected. Note that key regeneration does not detect an attack but only prevents it. In general, frequent regeneration and secure distribution of secret keys is associated with costs and may be difficult to perform when the attacker has physical access to the device, as assumed in fault-based attack scenario.

Note that all countermeasures mentioned in Table 3.1 can, in principle, be circumvented by the attacker with some degree of efforts. It is obvious from the table that V/T manipulation and Trojan-assisted fault injection have a countermeasure profile which is completely different compared to earlier fault-injection methods and may require countermeasures which have typically not been employed in the past.

Table 3.2. Detection of Hardware Trojans

Type	Functional testing	Side-channel analysis	Optical inspection
Small Trojan	Easy	Difficult	Medium
Large Trojan	Difficult	Easy	Easy
Dopant-level	Medium	Very difficult	Very difficult
MAPLE Trojan	Impossible	Very difficult	Very difficult

3.1.2 Hardware Trojans

The term "hardware Trojans" subsumes diverse techniques ranging from manipulation of the circuit by the foundry to threats in intellectual-property blocks from third-party providers or in CAD tools used to design the circuit. Since the MAPLE Trojans used in this work belong to the class of foundry-side manipulations, we restrict the discussion in this section to such threats; more details can be found in [87]. In general, the third-party manufacturer modifies the circuit such that it can develop undesired behavior, including:

- Denial of service (deactivating or producing constant or random outputs).
- Changing the circuit function.
- Establish a hidden side-channel.

The MAPLE Trojans used in this work have been designed for active fault-based cryptanalysis. We are not aware of earlier hardware Trojans created or used for this purpose.

A hardware Trojan consists of two parts: *trigger*, which activates the Trojan, and *payload*, which performs the undesired action. The Trojan may be triggered by an external event (such as the application of a specific input combination to the circuit inputs or setting V_{dd} to a particular value) or by an internal event (such as a counter reaching a pre-defined value or transistor wear-out of some degree [21]). Most hardware Trojans assume a *cooperative attack model*: one attacker (called *foundry-side attacker* in this chapter) is located within the foundry and manipulates the manufacturing process, and the second attacker triggers the Trojan payload in a manufactured circuit that is used in application.

Since hardware Trojans are present in the manufactured circuit, it is generally possible to identify them by testing. Three basic strategies are known for Trojan detection [69]:

- Functional testing [16] checks if the function implemented by the circuit corresponds to the specification by applying input vectors using either an external tester or on-chip built-in self-test blocks.
- Side-channel analysis [87] determines parameters such as timing or power consumption of the potentially affected circuit and compares them with expected values for a Trojan-free circuit.

- Optical inspection [33] is a destructive technique consisting of mapping the individual active and metal layers of the de-packaged circuit and checking them against the original circuit layout.

Table 3.2 summarizes the general detection properties of these methods for conventional Trojans, MAPLE Trojans introduced in this chapter, and related dopant-level Trojans [9].

It turns out that the complexity (and thus the size) of the Trojan is key to choosing the optimal detection method. A large Trojan might have a complex triggering condition which could be difficult to find, and therefore it may not be feasible to activate the Trojan during functional testing and observe its effects. On the other hand, the large circuitry will have impact on the circuit's delay and power consumption and will likely be identified by side-channel analysis. In contrast, the triggering condition of a small Trojan must be simple because it is implemented with only a few logic gates. As a consequence, it is feasible to check all such simple conditions during functional test such as to detect the Trojan. Side-channel analysis will be less effective as the few logic gates have a smaller contribution to the overall delay or power consumption and will probably be dominated by random variations. Finally, the optical analysis will, in theory, find both small and large Trojans, but it is obviously easier to find Trojans that are associated with more added or changed logic gates and/or affect more circuit locations.

A very recent Trojan insertion technique, which inspired our MAPLE Trojans, modifies the logic function of a logic gate by exchanging the dopant polarity of transistors within the gate [9]. This modification corresponds to a stuck-at fault of a multiplicity equal to the number of manipulated logic gates and is detectable by functional testing. Note that the attack on the random number generated reported in [9] was not detected because it was tested by a built-in test self block and the expected signature was manipulated. This circumvention would not be possible in

the usual tester-based scenario. Dopant-level Trojans may create side-channel effects and could be detected by side-channel analysis, but if only few logic gates are affected, the test equipment must have a very high sensitivity. Detecting the Trojans from [9] by optical inspection is not impossible but very difficult because no logic is added and removed and the modifications only involve layers transparent to most analytical methods. Recent work indicates that dopant-level Trojans can be detected using optical reverse-engineering at a very high cost that is almost 16 times higher than the detection of metal layers by optical detection [85].

The MAPLE Trojans inherit poor detectability of dopant-level Trojans by side-channel analysis and optical inspection and avoid detection by functional testing due to their dependency on V_{dd} and non-deterministic nature. This will be explained in detail in Section 3.3.5.

3.2 V/T Fault Injection

Fault injection into cryptographic hardware can have one of the following three consequences. First, the fault may have no visible effect on the circuit outputs, either because the injection was not strong enough or because it did not propagate to the output. We call this situation *fault-free* (FF). Second, the fault may be *exploitable* (EX), that is, correspond to the requirements of the considered attack scenario. Third, it may not correspond to these requirements; we call such a fault *not exploitable* (NE).

For example, the attack on the LED-64 block cipher considered in this work requires fault injection in one of the 16 four-bit *nibbles* of the state at a certain point of time within round 30. A fault that flips one, two, three or all four bits of one such nibble while leaving the other nibbles untouched is exploitable, while faults flipping bits in multiple nibbles or at different time points are not. The other cipher considered here, PRINCE, has two types of exploitable faults: one restricted to a single nibble in round 9 and the other restricted to a single nibble in round 8. Cryptanalysis takes

the ciphertext C calculated by the fault-free circuit from plaintext P using secret key k, and the *fault-affected* ciphertext C' calculated from the same P and k but by the circuit to which the fault injection was applied. If the injected fault was exploitable, cryptanalysis will either derive the key k from C and C' or restrict the number of key candidates. This is done by constructing a system of equations and solving them after k. If the injected fault was not exploitable, the equations will be inconsistent and in most cases have no solution.

The idea of V/T manipulation is to operate a circuit under reduced power-supply voltage V_{dd} and/or elevated temperature T. Running multiple encryptions under such out-of-spec parameters will lead to bit-flips at different circuit locations. These bit-flips are random to some extent, because they are aggravated by local noise. In general, the further V_{dd} and T are from their nominal values, the higher is the probability of fault injection at any location in the circuit. If the parameter are close to nominal, most encryptions will be from class FF. If the parameters are very far from nominal, many locations will tend to flip, and the injected faults will be from class NE. The aim of our technique is to find V_{dd}/T combinations under which EX faults occur with a sufficient rate for practical cryptanalysis.

It is important to stress that the fault injection is *not* based on increasing the delay of paths within the circuit and inducing a delay fault at the circuit outputs or flip-flops. While this mechanism is possible, it is extremely prone to process variations and not well-controllable. We assume that the circuit is run at a sufficiently low clock frequency for all gates to switch and that the faults are due to noise-induced bit-flips at individual gates.

In order to find the V_{dd}/T combination suited for injection of EX faults, we perform an analysis in two steps. The first step is an electrical-level simulation analysis of a circuit model which offers the access to all internal nodes. For several values of T, the power-supply voltage is swept between a small and a large value with small intervals.

For each combination, several encryptions are run and the number of FF, EX and NE faults are determined by leveraging the access to the circuit structures. For example, the attacks considered here require single-nibble fault injection; this is checked by simply looking up the value stored in the registers. Starting the voltage sweep with a value close to nominal V_{dd}, most encryptions will be FF, then slowly reducing the power-supply voltage, the number of FF encryptions will decrease and more faults will be EX (restricted to a single nibble) or NE. At some point, the voltage will be so low that multiple nibbles will be affected in nearly all cases and faults will become NE. The V_{dd}/T combination which lead to the highest proportion of EX faults is then recorded.

While the first step of the procedure assumes a simulation model of the circuit, the actual fault-based cryptanalysis is applied to a specific manufactured instance of the circuit. Because of manufacturing process variability, the V_{dd}/T combination obtained for the simulation model with nominal parameters may not be the best for that instance. Therefore, in the second step, the characterization is continued for the actual manufactured circuit instance. The voltage is swept around the values obtained during the simulation-based first step. A key difference with the first step is the lack of access to internal structures of a physical circuit. For this reason, the classification of a fault is performed as follows. Any obtained ciphertext C' is first compared with the fault-free ciphertext C; if they match, the encryption is categorized as FF. Then, cryptanalysis using C and C' is attempted. If it yields no solution, an NE fault is assumed, otherwise an EX fault is assumed. Strictly speaking, an NE fault might result in an equation system with an (incorrect) solution and therefore be wrongly categorized as EX, but such situations are rare.

Once the V_{dd}/T combination has been determined, the actual attack is performed. Note that some cryptographic functions, including the LED-64 cipher, require only one EX fault injection for a successful attack. In such a situation, several encryp-

tions are performed and cryptanalysis is attempted after each of them; once it is successful, the key is determined and the protection is broken. Other attacks, such as the attack on PRINCE, necessitate multiple fault injections using the same secret key. Again, several encryptions are performed and the fault-affected ciphertexts corresponding to EX faults are stored. Once a sufficient number of them has been collected, cryptanalysis can be invoked and will yield the secret key.

To give more insights on the characterization procedure, the ciphers PRINCE and LED described in Section 3.4 were designed in 45nm CMOS technology and the impact of voltage sweep on the device was observed. Voltage was varied in steps of 2mV, although more fine tuning is possible [5]. To increase the probability of noise-induced fault injection, temperature T was also varied. The percentage of correct and faulty computations under different V_{dd} and T values are shown in Fig 3.1. The probability of correct and faulty computations converge to 50% at some V_{dd} for a given T. Below this point, multiple faults are injected into the circuit and may be non-exploitable. The exact numbers for exploitable faults along with the stimuli values are presented in Section 3.4. As the cryptanalysis framework for LED-64 and PRINCE requires fault injection in a single nibble, V_{dd} and T values over which a fault is injected into a single nibble were evaluated. The distributions of single nibble fault injection as a function of V_{dd} and T are shown in Fig 3.2. Note that the single nibble faults shown in Fig 3.2 include the fault injections happening throughout the cipher circuit. However, only the faults injected in certain rounds are exploitable (details in Section 3.4). So, the percentage of exploitable faults will be lower than the bounds shown in Fig 3.2. The exploitable faults' statistics are presented in Sections 3.4.1.5 and 3.4.2.5.

3.3 MAPLE Trojan-based Fault Injection

This section presents MAnufacturing Process LEvel (MAPLE) techniques for hardware Trojan insertion in an IC to facilitate fault-injection attacks. MAPLE

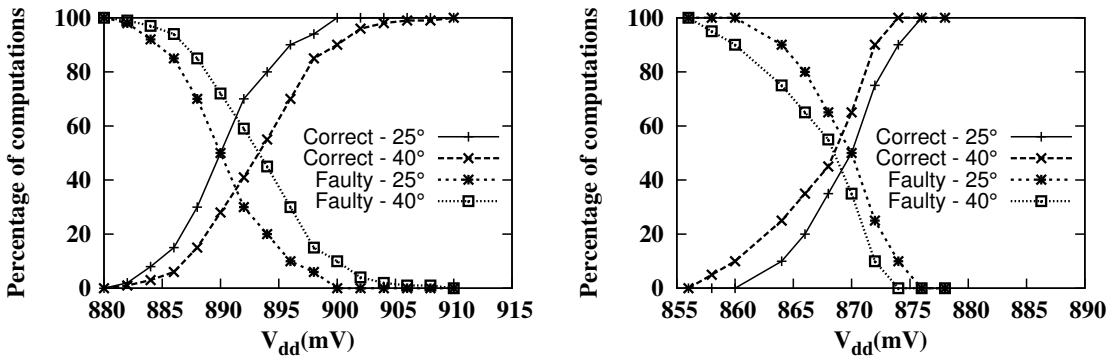


Figure 3.1. Percentage of correct and faulty computations as a function of V_{dd} and temperature in PRINCE (left) and LED-64 (right)

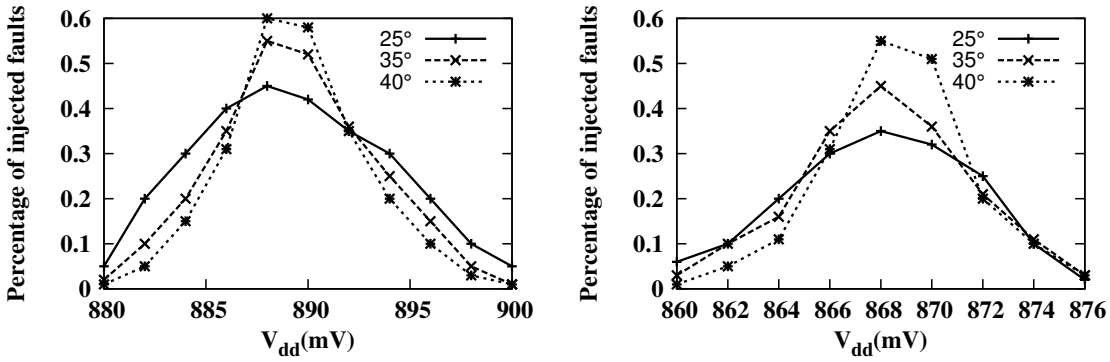


Figure 3.2. Percentage of injected faults in a single nibble as a function of V_{dd} and temperature in PRINCE (left) and LED-64 (right)

Trojans are based on altering a logic gate's electrical characteristics by modifying the doping concentration or dopant area within a transistor of the attacked logic gate. Inspired by dopant-level Trojans [9], MAPLE Trojans do not modify the metal, polysilicon and active areas and therefore are extremely hard to detect by optical inspection [9, 33]. However, the major difference of MAPLE Trojans from dopant-level Trojans is their probabilistic activation: the fault is injected with rather low probability (which makes it hard to detect by functional testing but is still sufficient for cryptanalysis) whereas the Trojans from [9] induced deterministic, stuck-at like behaviour. As MAPLE Trojans are inserted in either manufacturing process or layout

levels and involve changes in electrical characteristics, they fall under the category of *parametric Trojans* [87].

3.3.1 TrojanConc: Doping Concentration Manipulation

This technique focuses on altering the V_{in}/T-V_{out}/T characteristic also known as *Voltage Transfer Characteristic (VTC)* of the target logic gate by reduction of substrate doping concentration in one or more transistors. The doping concentrations play a vital role in determining the threshold voltage of a transistor (V_{th}) as identified by Equation 3.1,

$$V_{\text{th}}^{\text{NMOS}} = 2\phi_b + \frac{\sqrt{2\epsilon_{si} q N_a 2\phi_b}}{C_{ox}} + V_{\text{fb}}, \tag{3.1}$$

where ϕ_b is the bulk potential, ϵ_{si} is the permittivity of silicon, N_a is the doping concentration of the carriers in the substrate, N_i is the intrinsic carrier concentration of undoped substrate and C_{ox} is the gate oxide capacitance. The bulk potential ϕ_b is given by,

$$\phi_b = \frac{kT}{q} \ln \frac{N_a}{N_i}, \tag{3.2}$$

where k is the Boltzman's constant, T is the temperature and q is the electronic charge. At ambient room temperature ($T = 298$ K), the value of $\frac{kT}{q}$ is around 25 mV. MAPLE Trojans exploit the dependence of threshold voltage on carrier concentration N_a, which is clearly observed in Equation 3.1. By reducing N_a, the threshold voltage V_{th} of an nMOS transistor is increased. In a pMOS transistor, V_{th} is reduced by increasing doping concentration. Therefore, the malicious foundry can create a Trojan gate, e.g., a Trojan inverter with a manipulated VTC as shown in Fig 3.3 by changing N_a. The MAPLE Trojan shown in Fig 3.3 highlights the change in doping concentration for only nMOS transistors, even though doping concentrations can be altered in both the transistors.

The VTC of a Trojan inverter along with the VTC of a normal inverter is shown in Fig 3.5, where the doping concentration was varied by a factor of $10^{\pm 3}$. Note that the transistor becomes stronger and induce a shift in the switching voltage V_{m} (the input voltage for which the output voltage of the gate is around 0.5 V) because the threshold voltages V_{th} were reduced by doping concentration manipulation based on Equation 3.1. This type of gate modification is referred to as the MAPLE Trojan *TrojanConc*. Due to the modified VTC of the Trojan inverter, there could exist input voltage points greater than 0.5 V such that the output voltage is still above 0.5 V. This forces the logic gate driven by the Trojan inverter to observe an input '1' rather than '0' and the driven gate's output will flip unless its side-input is controlling.

Insertion effort

The doping concentration manipulation *per se* is a regular part of many manufacturing processes. Controlling N_a is exploited by (trusted and untrusted) foundries to create transistors with different threshold voltage levels, such as low-V_{th}, high-V_{th}, or ultra high-V_{th} etc. For this purpose, different exposure time to the carrier beam is used for different logic gates within the same circuit. This in turn requires a different mask for each concentration level to be used within the circuit. Inserting a *TrojanConc* Trojan is equivalent to introducing an additional concentration level, which however is far outside the regular process specification. As a consequence, the efforts amount to preparing additional masks and introducing a new process step which does not exist for Trojan-free circuits. This effort can be considered substantial and is visible at many levels within the foundry, yet technically the insertion is feasible.

3.3.2 TrojanArea: Dopant Area Manipulation

The transistor's strength can be altered by manipulating the doped area under the active area of a transistor. This concept was exploited in [9] for introducing a hidden side-channel inside an AES implementation. The MAPLE Trojan *TrojanArea* utilizes

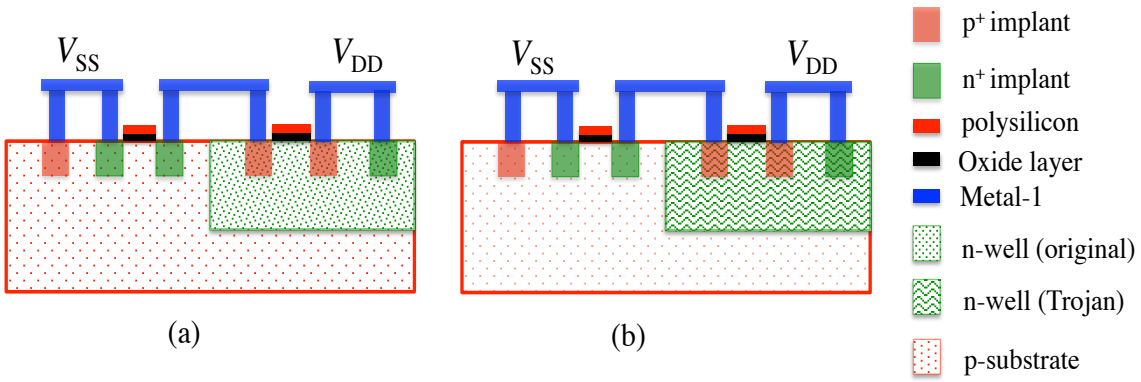


Figure 3.3. Cross-sectional view of (a) original inverter and (b) Trojan inverter using doping concentration manipulation

dopant area manipulation to create spots where transient faults are injected into a circuit. By reducing the dopant area within the active area, a Trojan inverter with a VTC shown in Fig 3.5 is created. The layouts of the normal and Trojan inverters are shown in Fig 3.4. The fundamental principle is the VTC drift towards the weak transistor.

Since process variations are increasingly dominant in sub-45nm design space, the effect of inserting *TrojanArea* must exceed the effect of variability. Based on simulations using 45 nm CMOS inverter from the standard cell library with $W_p = 540\,\text{nm}$ and $W_n = 360\,\text{nm}$, it was estimated that a reduction of 30% in dopant area of the nMOS transistor pushes the switching threshold V_m by around 0.2 V. The results of a comparative analysis of this drift with changes due to process variations are shown in Fig 3.6. For process variation simulations, threshold voltage distribution with $3\sigma = 150\,\text{mV}$ was assumed as in ITRS specifications [1]. Consequently, the vast majority of transient faults will be induced by the Trojan gates in their specific locations selected to benefit fault-based cryptanalysis, and not by random variability which manifests itself everywhere in the circuit.

Insertion effort

The dopant-area manipulation can be done both at the layout level (by a malicious designer in charge of the low-level optimization of the circuit) and by a malicious foundry. If the manipulation is carried out by the designer and the foundry in charge of producing the circuit is trusted, it may validate the layout before manufacturing it. However, the transistors in standard cells are often upsized for lower propagation delays, and a reduction in strength of one or two transistors making up the circuit will likely go unnoticed, as long as the design meets the `DRC` and `LVS` specifications.

If the circuit layout has no Trojans and the adversary is within the foundry, he just manually modifies the specification of the mask used to define the manipulated area. Masks are routinely modified compared with the original layout data for the sake of optimal proximity correction and other yield-enhancing post-processing steps. As a consequence, even if a deviation is discovered, it is improbable to raise concerns. In contrast to *TrojanConc* which requires major modifications of the whole manufacturing process, only small changes in the mask definitions are needed for the insertion of *TrojanArea*. Therefore, their insertion effort is considerably lower.

3.3.3 Trojan Activation

In order to activate the Trojans such that transient faults can be injected, the input voltage has to be close to the switching threshold V_m. This can be done by slightly reducing the supply voltage V_{dd}. If the supply voltage is slightly reduced and is noisy enough, the Trojan inverter will flip its output, when the supply voltage crosses V_m. In order to ensure that only the Trojan inverter flips its output, V_m of the Trojan inverter must be pushed far away from the nominal V_m. As the activation of MAPLE Trojans is based on noisy supply voltage, the switching behaviour is completely probabilistic. We define the term *trigger factor*, which denotes the probability of Trojan activation. To measure the trigger factor, the supply voltage was modeled using a Gaussian

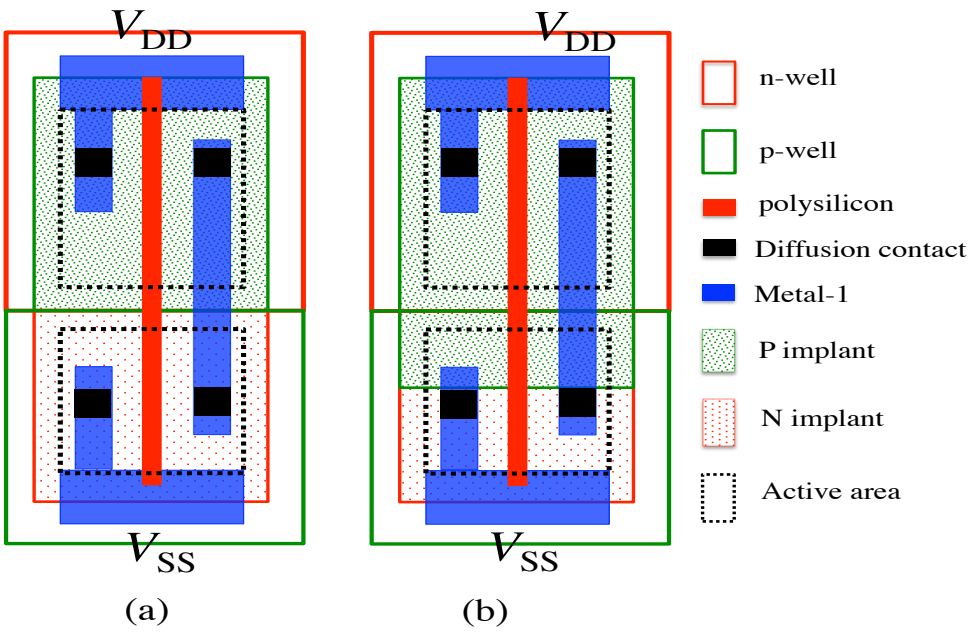


Figure 3.4. Layout of (a) original inverter and (b) Trojan inverter using dopant area manipulation

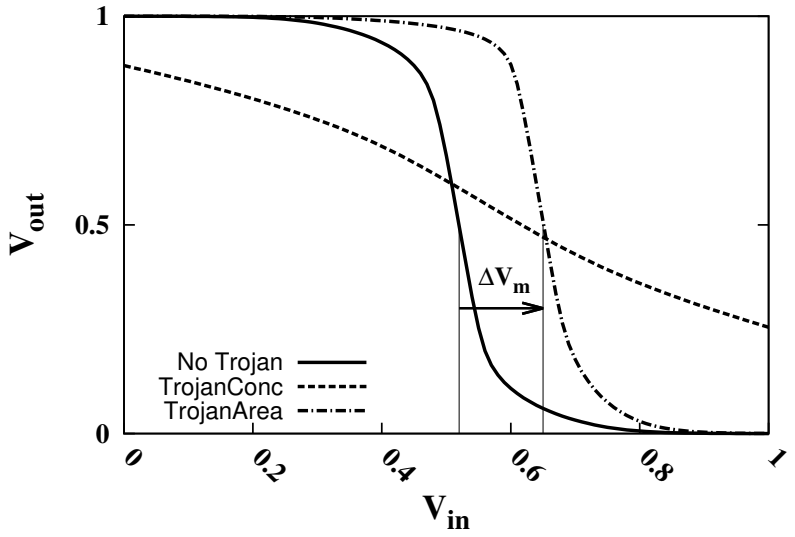


Figure 3.5. Electrical characteristics of the unmodified and Trojan inverters

distribution with 3σ deviation of $\pm 10\%$ of the mean V_{dd}, that is typically observed in power grids. The Trojan inverters (*TrojanConc* and *TrojanArea*) were designed with the VTC's shown in Fig 3.5 and were analysed under noisy V_{dd}. The trigger

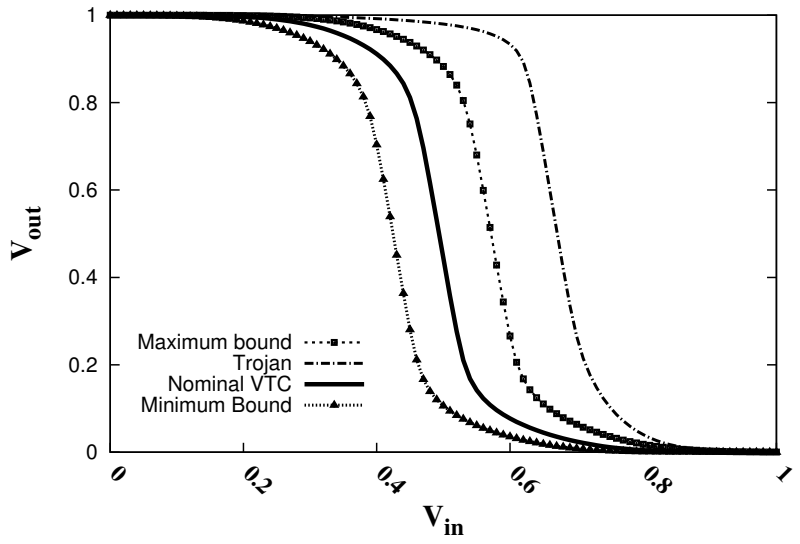


Figure 3.6. Impact of process variations on Nominal VTC and the Trojan VTC

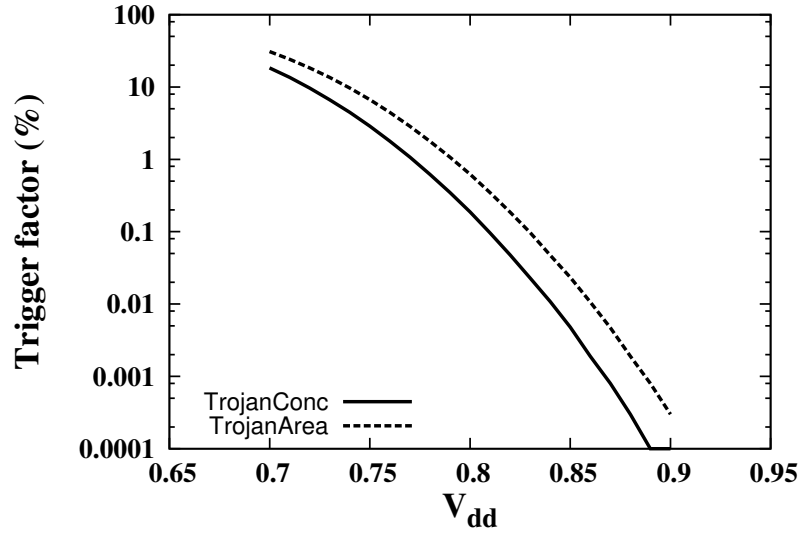


Figure 3.7. Triggering factor of the Trojan inverters

factor of the Trojan inverters are shown in Fig 3.7. The trigger factor is sufficiently low ($< 10^{-6}$) for 10% reduction in $V_{\rm dd}$ and increases with decreasing $V_{\rm dd}$.

The activation of MAPLE Trojans is fundamentally different from Dopant-level Trojans proposed in [9]. The dopant-level manipulation from [9] manifests itself as a "stuck-at" fault and based on the manipulated dopant polarities, they can exhibit

a stuck-at-0 or stuck-at-1. In contrast, MAPLE Trojans are activated stochastically, with very low probability determined by the amount of manipulation (dopant-area or concentration). Such a stochastic nature renders MAPLE Trojans nearly undetectable while still allowing precise fault injections.

3.3.4 Threat Model

Like other hardware Trojans, MAPLE Trojans are exploited by two adversaries: the foundry-side attacker who manipulates the manufacturing process and the operation-time attacker who uses the maliciously manufactured circuits in field. These adversaries will usually not be the same persons. The foundry-side attacker could sell the information about Trojans to a potential attacker, who then can extract the secret key from the cryptographic circuit. MAPLE Trojans can also be used for introducing "back-doors" by criminal organizations or authorities.

The complexity of inserting Trojans *TrojanConc* and *TrojanArea* by the foundry-side attacker has already been discussed in the end of the respective Sections 3.3.1 and 3.3.2. Once the Trojan-affected circuit is fabricated, mounting the attack requires minimal effort. The Trojans are activated by reducing V_{dd} as explained in Section 3.3.3. Exploitable faults are injected with low probability which depends on the supply voltage level. Therefore, the attacker must be able to control V_{dd} of the circuit and to invoke multiple runs of the same encryption. Both capabilities do not require sophisticated equipment.

As is usual in cryptanalysis, the considerations in this article assume Kerckhoff's principle "the enemy knows the system": the foundry-side attacker knows the location of circuit structures where injected faults are exploitable. This implies that the foundry can track locations in the layout and mask data to registers and logic gates of the behavioural and/or gate-level circuit description. If the foundry does not have this information, reverse-engineering is required to obtain it before performing

the manipulation. While the designer may complicate reverse-engineering, he cannot prevent it if the attacker devotes to it sufficient time and resources.

3.3.5 Countermeasures and Detection

Countermeasures against various types of fault injection have already been summarized in Table 3.1. Light sensors and shielding do not identify MAPLE Trojans because the circuit is not depackaged. Voltage sensors are ineffective because, as discussed above, $V_{\rm dd}$ reduction is limited and cannot be reliable distinguished from power-supply noise. However, preventing the attacker from setting the desired $V_{\rm dd}$ would render the fault-injection efforts very high. Concurrent error-detection is, in general, effective against injected faults but it does not guarantee detection with certainty and is rather expensive. Regenerating the secret key with rate that exceeds trigger factor will effectively thwart the attack, even though the attack will not be identified. Note, however, that even if the system designer anticipates the possibility of such attacks and would like to counteract by enforcing frequent key exchange, he does not know the Trojan trigger factor and cannot select an adequate regeneration rate.

Since MAPLE Trojans modify the manufactured circuit, they can be, in principle, detected during post-manufacturing testing. As already has been indicated in Table 3.2, all known methods for such testing are ineffective against MAPLE Trojans, as detailed below:

3.3.5.1 Functional testing

When testing under nominal $V_{\rm dd}$, MAPLE Trojans have extremely low or even negligible triggering factor ($< 10^{-7}$) and hence are not activated and cannot be detected. It is possible to apply the test at reduced voltage [20]. If $V_{\rm dd}$ is reduced slightly, MAPLE Trojans can be invoked, but still with a low trigger factor (e. g., 10^{-6} for voltage reduction of 10%). Recall that this is adequate for the operation-

time attacker who knows about the inserted Trojan and can repeat the encryption 100,000 times or more until the desired fault has been injected. In contrast, the test engineer does not know with certainty whether a Trojan is present at all, where it is located, how it is activated, and what triggering factor it has. Repeating all the tests 100,000 times or more does not appear economically feasible. Moreover, if no deviation is observed, the Trojan may still be inserted but have a higher triggering factor than the test engineer anticipated. Finally, a detected fault is transient and not repeatable and may be easily confused with a failure of the test equipment or an effect of radiation or electrical noise.

One way to significantly increase the triggering factor is to reduce V_{dd} by a large amount (e. g., by 30-40%). However, this will result in timing errors and failures observed at the circuit outputs which cannot be distinguished from the effect of a MAPLE Trojan. Under significantly reduced V_{dd}, the Trojan induced faults are highly masked by the faults due to timing errors, although the trigger factor of Trojans is significantly high at such voltage levels.

3.3.5.2 Side-channel analysis

As the manipulated gates have different electrical characteristics, their power consumption profiles will be different from regular gate versions. This implies that they can be, in principle, detected by power side-channel analysis such as I_{DDQ} testing. If the manipulated gate is considered in isolation, its measured current-consumption profile will clearly show the difference to the Trojan-free gate. However, in a real circuit, the number of manipulated gates is very low. For example, our attack on PRINCE involves manipulating six inverters out of ∼8,000 gates within the PRINCE block, which in turn will be embedded in an even larger circuit. The currents drawn elsewhere on the chip will effectively mask the contribution of the manipulated gates in presence of process variations.

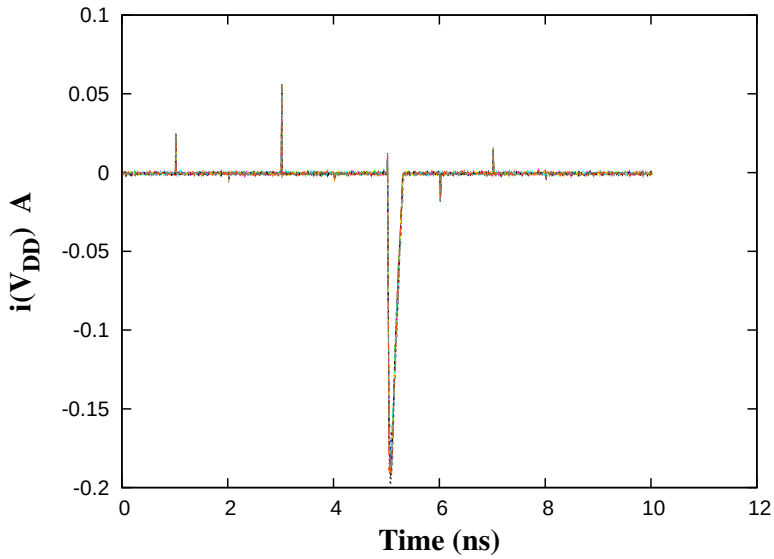


Figure 3.8. Supply drawn current for PRINCE

To illustrate the low detectability of MAPLE Trojans by I_{DDQ} testing, the current drawn from the supply during ten random encryptions by the PRINCE circuit from Section 3.4.2.2 is shown in Fig 3.8. The peak in the current profile shows the latching of key and plaintext into registers. The Trojans inserted into PRINCE are activated between the time duration 8.7 and 8.75 ns. This duration is magnified and shown in Fig 3.9. From ten encryptions, four different patterns were chosen that corresponds to the transitions possible for the normal and manipulated inverters (0→1 and 1→0). The relative difference between the current profiles for the corresponding transitions is minimal and will likely be masked by I_{DDQ} drawn by other circuit structures.

3.3.5.3 Optical Inspection

The manipulations performed during MAPLE Trojan injection keep all metal, polysilicon and active-area structures, which are recognizable during optical inspection, unaltered. The modified structures require high-effort analysis methods which will are difficult to apply in all suspicious locations of the circuit. As a consequence, MAPLE Trojans are nearly immune to optical reverse-engineering [33]. A very recent

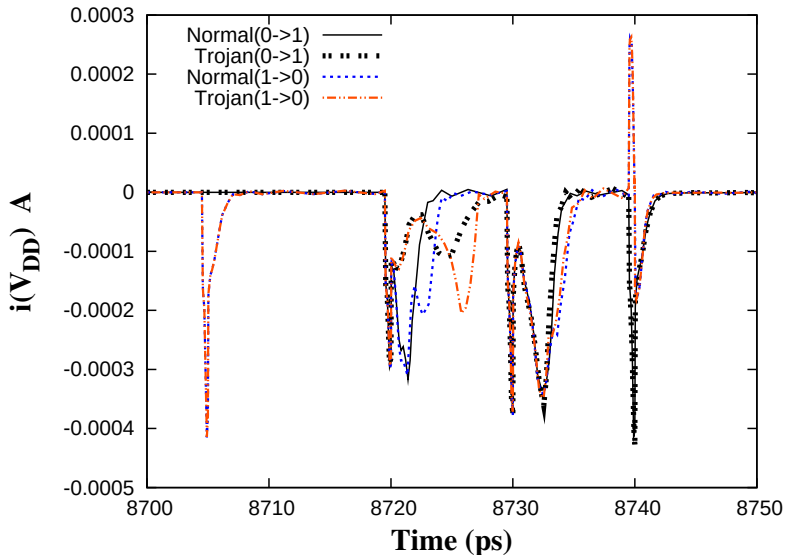


Figure 3.9. Supply drawn currents for the original and Trojan inverters

work has demonstrated optical detection of Dopant-level Trojans [85]. This technique can work for *TrojanArea*, as it is similar to Dopant-level Trojans, albeit at an enormous as demonstrated in [85]. However, $TrojanConc$ will still be immune to optical detection, as all the structures including dopants remain intact.

3.4 Results

Both fault-injection techniques, the V/T manipulation and the Trojan-assisted approach, have been applied to recent attacks on two state-of-the-art lightweight block ciphers: LED-64 (for which one fault-injection is sufficient in most cases) and PRINCE (for which the attack consists of two stages and both stages typically require 2-3 fault injections). In the following, the algorithms, their circuit implementations used for fault-injections and the execution of the attack are presented for both ciphers. The usage of V/T and Trojan-assisted fault injection for executing the attacks and the results of the cryptanalysis are discussed afterwards.

3.4.1 LED-64

3.4.1.1 General Layout

LED-64[4] [26] bears parallels to the well-known AES block cipher, but has been optimized for a resource-efficient hardware implementation and the layout is shown in Fig 3.10. The *state* S of the cipher consists of 64 bits organized in 4-bit *nibbles* $s_0, \ldots, s_{15}$, which can be written as a 4×4 matrix. Each s_i is identified with an element of $\mathbb{F}_{16} \cong \mathbb{F}_2[x]/\langle x^4 + x + 1 \rangle$.

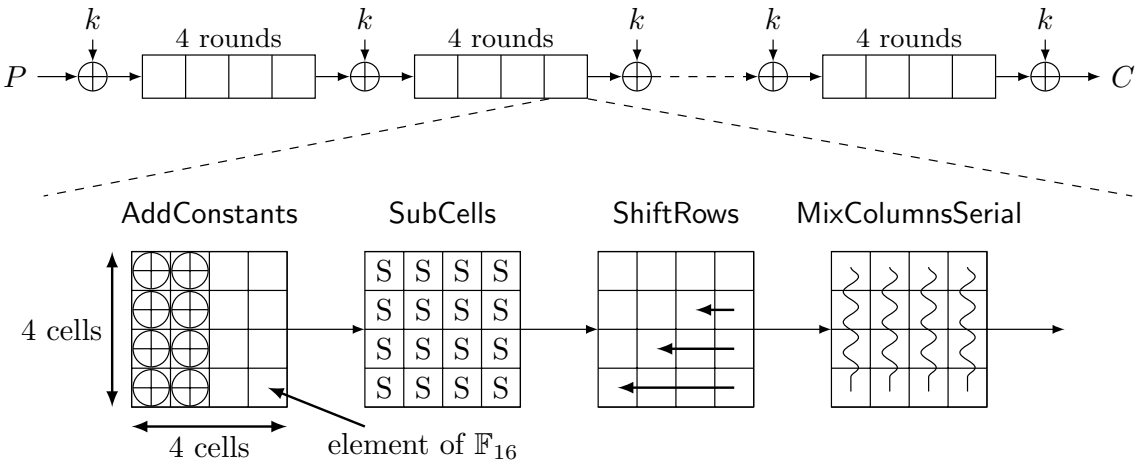


Figure 3.10. Layout of LED-64

The encryption consists in applying 32 rounds to the 64-bit plaintext P. Each round consists of four operations:

- *AddConstants (AC)*: perform an XOR operation of the state with a round-specific constant. All constant entries in the two rightmost columns of the matrix are equal to 0.
- *SubCells (SC)*: apply a non-linear mapping SBox to each nibble of the state: s_i is mapped to $SBox[s_i]$.

[4]The LED-128 version, which uses two independent 64-bit keys instead of one and has 48 instead of 32 rounds, is not considered in this article.

- *ShiftRows (SR)*: leave the first row of the state matrix untouched and shift the second, the third and the fourth row by one, two and three positions circularly to the left.

- *MixColumnsSerial (MCS)*: multiply the state (over $\mathbb{F}_{16}$) by a MDS matrix M.

The constant of the individual rounds, the SBox mapping and the matrix M can be found in the LED specification [26]. The LED-64 secret key k consists of 64 bits, which are also organized in 4-bit nibbles and arranged in a 4×4 matrix. After every four rounds, the key is XORed with the state. The ciphertext is the circuit state after round 32 and the last XOR operation with k.

3.4.1.2 Circuit implementation

For fault injection attacks, the LED-64 cipher was implemented in 45nm CMOS technology using the Nangate open cell library. The cipher produces an output in 32 clock cycles, where each cycle corresponds to one round of encryption. The gate count detail is presented in Table 3.5. The performance numbers are highly competitive with the original circuit reported in [26].

3.4.1.3 Fault-based cryptanalysis

The cryptanalysis of LED-64 assumes that a fault has been injected in the beginning of round 30, such that three full rounds and one XOR operation with the secret key k are performed (Fig 3.11, before the ciphertext is output. The fault must involve an arbitrary number of bits within one four-bit nibble but not across multiple nibbles. Recall that such a fault is called exploitable (EX) while faults not satisfying this condition are not exploitable (NE). For illustration, we will consider faults in the first nibble s_0 but very similar procedures apply for faults in other 15 nibbles $s_1, \ldots, s_{15}$. If the affected nibble is not known, cryptanalysis is repeated up to 16 times assuming different nibbles. The attacker uses the ciphertext C' observed at

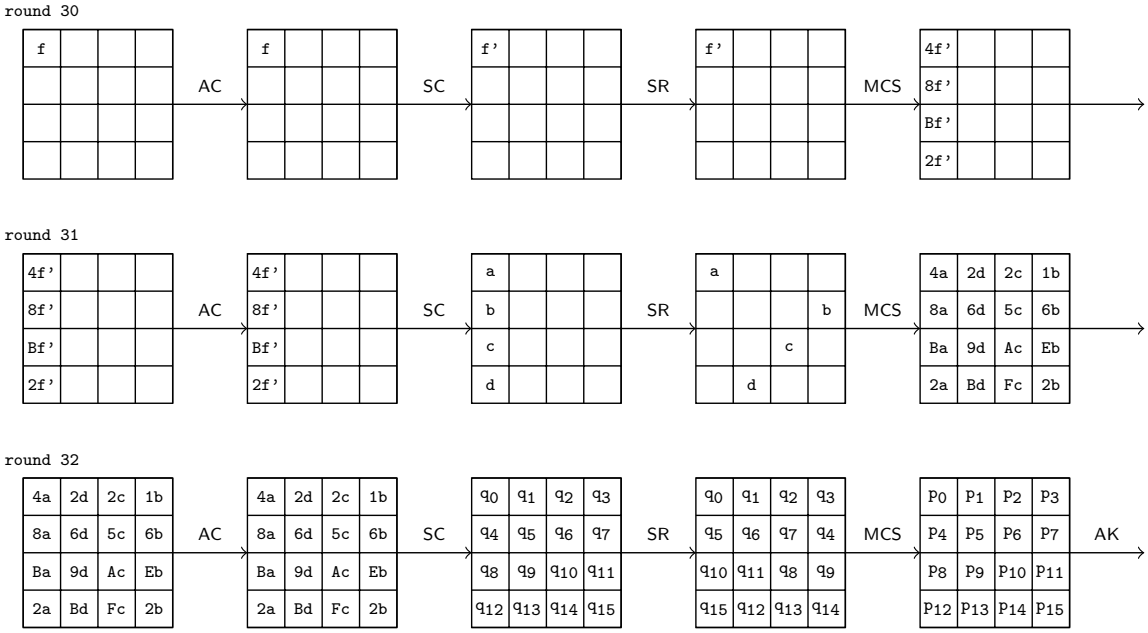


Figure 3.11. Propagation of the injected fault

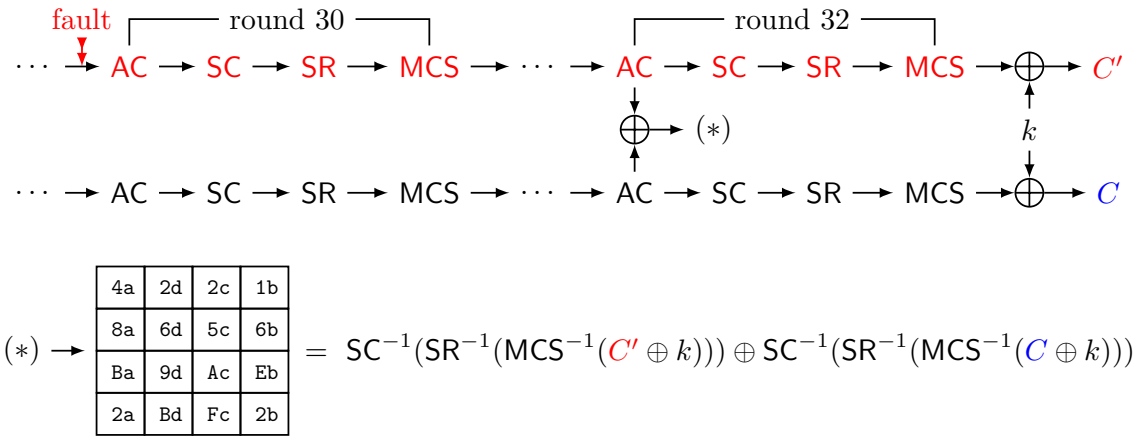


Figure 3.12. Equation construction for the attack on LED-64

the output of the fault-affected cipher and the fault-free ciphertext C for analysis. C must be obtained using the same secret key k as C', otherwise, if the key has been exchanged between the observation of C and C', the analysis will fail. The plaintext P may or may not be known to the attacker but is not used for cryptanalysis.

While the states S of the fault-free and S' of the fault-affected cipher at the end of round 29 (before the fault injection) are not known, they must be identical: $S \oplus S' = 0$. If the fault injected in the beginning of round 30 is exploitable, S and S' must differ in the top-left entry by an (unknown) value f, as shown in Fig 3.11. The subsequent mathematical analysis reasons about the difference $S \oplus S'$ after the fault injection. Adding identical constants to both S and S' (AC) does change $S \oplus S'$. Applying non-linear SBoxes (SC) to unknown state nibbles will change the difference of nibbles s_0 and s'_0 from f to $f' := SBox[s_0] \oplus SBox[s'_0]$, which depends on specific values of s_0 and s'_0, and therefore cannot be predicted. However, all other nibble pairs with $s_i = s'_i$ remain identical after the application of the SBox to each of them. The SR operation does not change the first row of the matrix which contains the difference. The matrix multiplication (MCS) is a linear operation which spreads the difference f' over the leftmost column of the state matrix as shown in Fig 3.11.

Similar reasoning can be applied to round 31: AC does not change the difference; SC replaces the differences in the leftmost column by unknown values a, b, c and d; SR moves the differences horizontally and MCS spreads them over the whole matrix. The AC operation of round 32 does not modify these differences.

The matrix obtained after the AC operation of round 32 is used for construction of fault equations with the secret key and a, b, c and d as unknown variables. To construct these equations, we notice that the state of the fault-affected cipher before the final XOR operation with the secret key k is $C' \oplus k$. Consequently, the state of the fault-affected cipher after AC of round 32 is $\mathrm{SC}^{-1}(\mathrm{SR}^{-1}(\mathrm{MCS}^{-1}(C' \oplus k)))$. Similarly, the state of the fault-free cipher after AC of round 32 is $\mathrm{SC}^{-1}(\mathrm{SR}^{-1}(\mathrm{MCS}^{-1}(C \oplus k)))$.

The XOR of these two expressions must equal the state difference matrix calculated in Fig 3.11. This is illustrated in Fig 3.12. While we are not describing the equations here in detail, they are used to obtain filtering rules which restrict the set of key candidates (keyspace) from 2^{64} down to values for which brute-force search is practical. Details are found in [34].

3.4.1.4 Parametric fault injection

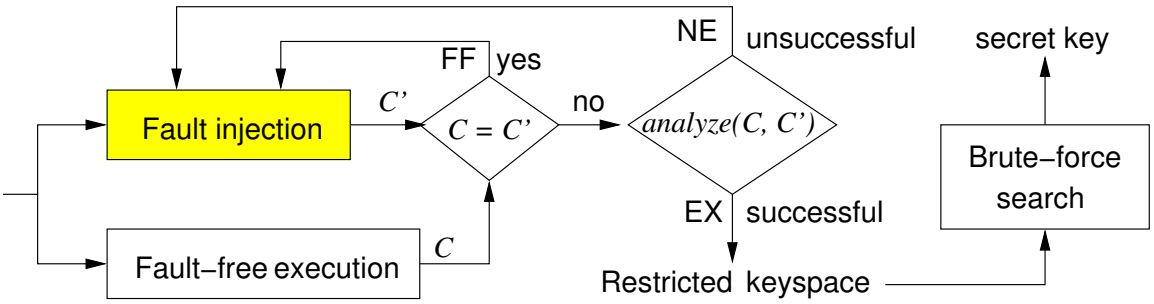


Figure 3.13. Overview of attack on LED-64 using fault injection

The theoretical attack framework from [34] assumed that the injected fault is exploitable. This assumption does not always hold when using parametric fault injections: some fault injection attempts may not result in a faulty output at all (FF), and some may result in NE faults. Fig 3.13 shows how the attack is performed using parametric fault injection. After the fault injection, the obtained C' is compared with the fault-free C to identify the FF case. After that, the cryptanalysis procedure outlined in the last section and called $analyze(C, C')$ is attempted. If it is successful, the restricted keyspace is calculated and brute-force search, i. e., simulation of each key candidate from the keyspace is performed until one candidate leads to ciphertext C. Otherwise, the system of fault equations will be inconsistent and have no solution; the keyspace will be empty. In this case, the fault was NE and the fault injection must be repeated.

If the restricted keyspace is too large for brute-force search, it is possible to use an additional EX fault to obtain further fault equations, which together with the original equations define a much smaller keyspace. In our experiments, we use a second EX fault injection when the restricted keyspace had more than 2^{23} elements after the first EX injection.

3.4.1.5 V/T fault injection results

Based on the V/T fault injection mechanism explained in Section 3.2, the exploitable faults were injected into the circuit after the characterization phase, where V_{dd} and T intervals for single nibble fault injection are determined. Similar to the experiment for PRINCE described in Fig 3.2, the intervals for LED-64 were determined where single-nibble faults were injected. These intervals were 865 – 890 mV for V_{dd} and 25 – 40 °C for T. The probability of single nibble fault injection decreases for temperatures higher than 40 °C, as the faults start spreading to multiple nibbles. Based on simulations over 10,000 random datasets (P and k), the percentage of exploitable faults was estimated to be around 0.4%. The impact of process variations on the supply voltage interval was also analysed over 40 different instances of LED-64 circuit. Monte Carlo simulations were performed to obtain different instances of the circuit. Although offsets in the mean V_{dd} for single-nibble fault injection were observed, the V_{dd} interval was similar for all the instances. The best percentages of exploitable faults for eight different instances along with the mean V_{dd} of the supply voltage interval are shown in Table 3.3. Average statistics are shown in Table 3.5 for comparison with Trojan-based fault injection and with PRINCE.

3.4.1.6 Trojan fault injection results

In order to inject faults into round 30, as required by the cryptanalysis, the round constants of that round, implemented by inverters, were targeted for MAPLE Trojan insertion. The round constant for round 30 is `2E`, and the inverters corresponding

to `E` were replaced with Trojan inverters. The average trigger factor of the Trojan inverters over 40 instances of LED-64 circuit is shown in Fig 3.18. For this analysis, *TrojanArea* with 30% reduction in dopant area was used. Note that the trigger factor is proportional to ΔV_m and *TrojanConc* with similar ΔV_m can be synthesized. Reducing V_{dd} increases the trigger factor but not necessarily the percentage of EX faults. This percentage will decrease beyond a certain V_{dd}, as timing induced faults start to dominate the influence of Trojans, as shown in Fig 3.19. The best-case percentages of EX faults for Trojan induced fault injection are shown in Table 3.5.

Table 3.3. Best-case V/T Fault Injection percentages for eight random instances of LED-64

	Instance							
	1	2	3	4	5	6	7	8
V_{dd}	0.872	0.87	0.874	0.87	0.868	0.874	0.876	0.87
T	40	40	40	40	40	40	40	40
FF	82.44	82.45	82.32	82.6	82.52	82.36	82.73	82.82
EX	0.36	0.35	0.38	0.4	0.38	0.34	0.37	0.38
NE	17.2	17.2	17.3	17	17.1	17.3	16.9	16.8

3.4.1.7 Cryptanalysis results

To validate our methods, we generated a data set of $50,000$ instances, each consisting of 25 tuples (P, C, C'), with P, C and C' denoting plaintext, fault-free ciphertext and fault-affected ciphertext, respectively. All tuples of a particular instance were encrypted with the same key. The faulty ciphertexts C' were obtained through the fault injection methods introduced earlier. We applied our cryptanalysis method from Section 3.4.1.3 on the above tuples. We used the threshold $\tau = 2^{23}$ for the following optimization. If the keyspace restricted after one EX fault injection was smaller than τ, we applied brute-force search directly. Otherwise, we injected a second fault to further restrict the keyspace. Note that increasing τ would reduce the number of required fault injections but increase the run-time of the analysis procedure.

For each of the $50,000$ instances we were able to successfully reconstruct the secret key using an average of 1.62 tuples (out of 25 available). This corresponds to the need for 1.62 EX fault injections on average. The combined time required to run the analysis on the pairs (C, C') of a particular instance, followed by an exhaustive search over the reduced key space was 10.21 seconds on average. The sizes of the keyspaces after one and two fault injections were $2^{23.34}$ and $2^{2.27}$ on average. Fig 3.14 shows the distribution of keyspace sizes after 1 and 2 fault injections, respectively. Note that the largest size was approximately 2^{29}, which is still easy to manage with a bruteforce search on modern hardware, and thus would allow to execute the attack with a single fault injection. However, as already mentioned above, the overall run-time for the analysis of one instance would be a lot higher than 10.21 seconds, unless advanced tricks like parallelization are used, and the analysis of $50,000$ instances would have required a lot more time. This was the main reason why we chose a threshold of $\tau = 2^{23}$, and traded a little increase in the amount of fault injections for a highly reduced run-time.

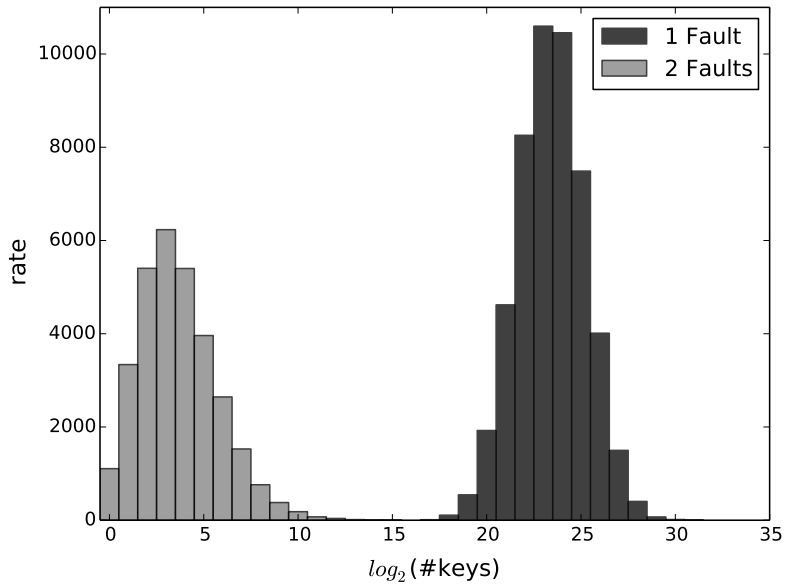


Figure 3.14. LED-64 analysis results

3.4.2 PRINCE

3.4.2.1 General Layout

PRINCE [13] is a 64-bit block cipher with a 128-bit secret key. As in LED, the 64-bit cipher state is organized in 4-bit nibbles. Before an encryption (or decryption) is executed, the secret key $K = k_0 \parallel k_1$ is extended to 192 bit $k_0 \parallel k_1 \parallel k_2$ with $k_2 = (k_0 \ggg 1) \oplus (k_0 \gg 63)$. The subkeys k_0 and k_2 are used for input- and output-whitening. The key k_1 is solely used in the core of PRINCE, which is a 12-round block cipher. Fig 3.15 gives an overview.

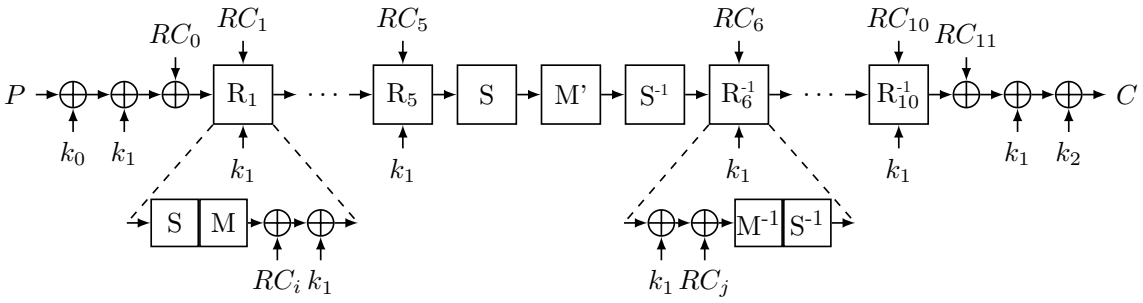


Figure 3.15. Layout of PRINCE

Each round R_i and R_{i+5}^{-1} with $i \in \{1, \ldots, 5\}$ consists of a key addition (XOR with k_1), an S-layer (application of a 4-bit non-linear SBox to each nibble of the state), a linear layer (multiplication of the state represented by a 64-bit row vector by a 64×64 matrix M, and the addition (XOR) of a round constant RC_j, with $j \in \{0, \ldots, 11\}$. Rounds R_i and R_{i+5}^{-1} are separated by a middle layer, which consists of two S-layers, where SBoxes S and S^{-1} are applied, interleaved with the multiplication of a matrix M'. The particular operations can be found in the specification of PRINCE [13]. However, we quote the round constants in Table 3.4 because their values are important for Trojan-assisted fault injection.

Table 3.4. The PRINCE round constants

i	RC_i	i	RC_i
0	0000000000000000	1	13198a2e03707344
2	a4093822299f31d0	3	082efa98ec4e6c89
4	452821e638d01377	5	be5466cf34e90c6c
6	7ef84f78fd955cb1	7	85840851f1ac43aa
8	c882d32f25323c54	9	64a51195e0e3610d
10	d3b5a399ca0c2399	11	c0ac29b7c97c50dd

3.4.2.2 Circuit implementation

Similar to LED-64, the PRINCE cipher was implemented using Nangate open cell library (45 nm). We implemented both combinational and sequential versions of PRINCE, as the fault injection statistics depend highly on the type of implementation. The combinational version produces a ciphertext in one clock cycle, whereas the sequential version produces a ciphertext in 10 clock cycles. However, the clock frequency of sequential version is substantially higher than the combinatorial version. The area numbers for the implementations can be found in Table 3.5.

3.4.2.3 Fault-based cryptanalysis

The attack on PRINCE [35] is done in two stages, where each stage is conceptually similar to the attack on LED from Section 3.4.1.3. We omit the mathematical details of the attack and only briefly sketch the properties that are relevant for fault injection. Again, C denotes the fault-free and C' the fault-affected ciphertext observed by the attacker. In stage 0, the faults are injected into round 9 and the constructed fault equations are used to restrict the number of candidates for the expression $(k_1 \oplus k_2)$. In general, the restriction yielded by one fault injection is insufficient and multiple EX fault injections are required. Similar to LED, a fault injection is EX if it only affects one 4-bit nibble of the cipher state in round 9. We employ an adaptive approach: define a threshold τ_0 and continue injecting faults until the size of the restricted keyspace for $(k_1 \oplus k_2)$ falls below τ_0.

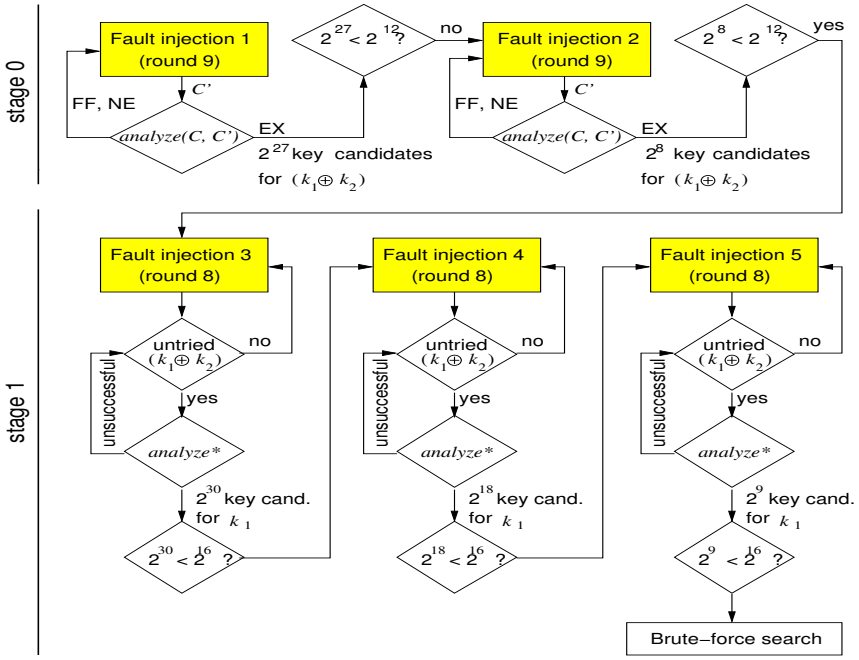


Figure 3.16. Overview of attack on PRINCE using 2 EX fault injections in stage 0 and 3 EX injections in stage 1 with $\tau_0 = 2^{12}$ and $\tau_1 = 2^{16}$.

In stage 1 of the attack, the faults are injected into round 8 and the objective is to restrict the number of candidates for k_1. Assuming that the fault was EX (again, this means only one 4-bit nibble of the state was involved), the following procedure is applied. For each $(k_1 \oplus k_2)$ candidate from the keyspace calculated in stage 0, the states of the fault-affected and the fault-free ciphers after round R_{10}^{-1} are calculated as $C' \oplus (k_1 \oplus k_2) \oplus RC_{11}$ and $C \oplus (k_1 \oplus k_2) \oplus RC_{11}$, respectively. These values are used instead of C' and C for the same cryptanalysis as in stage 0. Since only one $(k_1 \oplus k_2)$ candidate from stage 0 is correct, the cryptanalysis in stage 1 has to be repeated up to τ_0 times before a keyspace restriction is obtained. Like in stage 0, the achieved restriction may not allow brute-force search. For this reason, another threshold τ_1 is defined and fault injection are repeated until the number of candidates for k_1 is less than τ_1. Once the keyspace has been restricted, the complete key is reconstructed from k_1 and $(k_1 \oplus k_2)$ for each candidate and a brute-force search is applied.

3.4.2.4 Parametric fault injection

Like in LED analysis, not all injected faults are exploitable; some (in practice, the majority) of fault injections result in FF and NE outcomes. Fig 3.16 shows how one particular two-stage attack scenario is conducted. In stage 0, faults are injected in round 9 until one of them is EX (the calculation of the fault-free ciphertext C and its comparison with C' are omitted from the figure). Assume that the resulting keyspace has 2^{27} candidates for $(k_1 \oplus k_2)$, which exceeds the stage-0 threshold $\tau_0 = 2^{12}$; consequently, a second fault injection is required. It is repeated until cryptanalysis $analyze(C, C')$ is successful. Assume that the improved restriction is 2^8, which is below τ_0; then, stage 0 is finished. Note that it used two EX faults but possibly many more fault-injection attempts which resulted in FF and NE faults.

Stage 1 starts with fault injection in round 8. In order to check whether the fault was exploitable, the cryptanalysis has to deliver a consistent system of fault equations. This in turn necessitates trying $(k_1 \oplus k_2)$ candidates from stage 0 in order to invoke $analyze(C \oplus (k_1 \oplus k_2) \oplus RC_{11}, C' \oplus (k_1 \oplus k_2) \oplus RC_{11})$. If none of 2^8 candidates for $(k_1 \oplus k_2)$ leads to a consistent cryptanalysis, the fault must have been NE and the fault injection must be repeated. Assume that the first successful cryptanalysis yielded $2^{30} > \tau_2$ candidates for k_1. The same procedure is repeated for the fourth fault injection (second in stage 1) in order to further restrict the keyspace. Assume that 2^{18} candidates are obtained, which is still above τ_1. The fifth EX fault injection is then needed, which results in 2^9 candidates. Since this value is now below the threshold, they can be simulated. Note that $(k_1 \oplus k_2)$ is known at this point.

3.4.2.5 V/T fault injection results

The supply voltage and temperature intervals for single nibble fault injection in PRINCE are shown in Fig 3.2. Like in LED-64, the fault injection statistics were observed over 40 instances of PRINCE circuit. The percentage of exploitable faults

over the voltage and temperature intervals are shown in Table 3.5. The percentage of exploitable faults in sequential version of PRINCE is almost 10× higher than the combinatorial version. This is because of the fact that the sequential version has a time reference for fault injection (8th and 9th clock cycles) as required by the cryptanalysis framework, whereas the combinational version does not have a time reference. As in LED-64, we report the best-case percentages of exploitable faults in Table 3.5.

3.4.2.6 Trojan fault injection results

The MAPLE Trojans were inserted into the round constants in the rounds 8 and 9, which are depicted in Table 3.4. Both of these rounds include a nibble of value `d`, which has been picked for Trojan manipulation because it is implemented using three inverters and thus maximizes the probability of a fault. *TrojanArea* insertion into this nibble is shown in Fig 3.17. Three inverters, shown in black, are manipulated by changing the dopant area, whereas all other inverters remain regular. If any of these three inverters, any pair of these inverters or all three inverters flip, the resulting fault is confined to one nibble and therefore exploitable.

Note that the faults in round 9 are used for stage 0 of the cryptanalysis. For stage 1 of the cryptanalysis, faults must be injected into round 8. In the sequential version of the circuit, the three inverters are manipulated and the fault injection is controlled by reducing V_{dd} during cycle 9 (for stage 0 of the attack) or cycle 8 (for stage 1 of the attack) while nominal voltage is applied during other cycles. In the combinational version, three inverters from round 9 and another three inverters from round 8 are manipulated. Unfortunately (from the attacker's point of view) this gives raise to simultaneous excitation of faults in rounds 8 and 9. These faults violate the assumptions of the cryptanalysis procedure and are not exploitable.

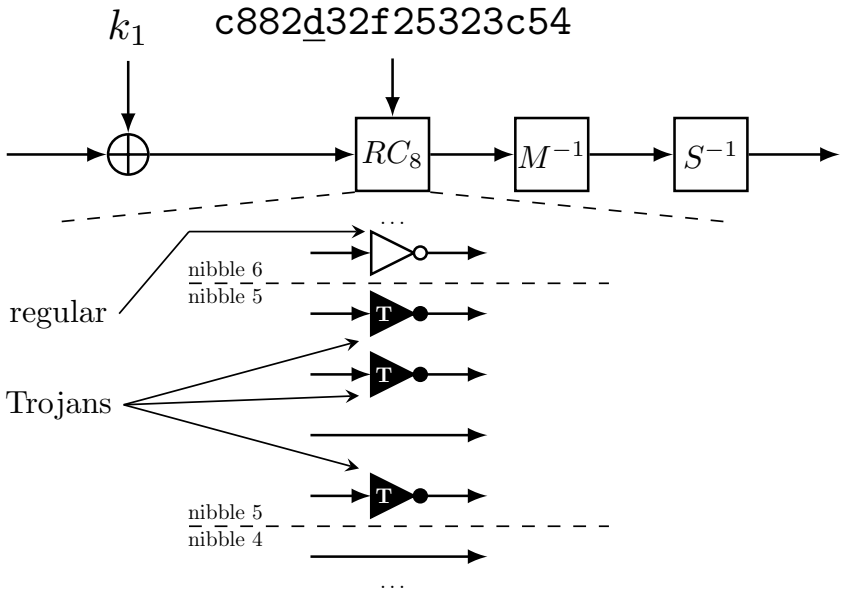


Figure 3.17. Schematics of PRINCE MAPLE Trojans

The average trigger factor of the Trojans inserted into 40 instances of PRINCE is shown in Fig 3.18. The trigger factor of the Trojans in PRINCE is lower than in LED-64, as the Trojans inserted into rounds 8 and 9 are vulnerable to simultaneous triggering and result in NE faults. Such triggering actions were discarded from cryptanalysis. As in LED-64, the trigger factor does not directly correspond to exploitable faults and the percentage of EX starts decreasing beyond a certain V_{dd}. The behavior is shown in Fig 3.19 and the best case percentages for EX is shown in Table 3.5.

3.4.2.7 Cryptanalysis results

In this part we report on the experimental results of the differential fault analysis of PRINCE. We analysed $10,000$ instances, each consisting of a data set of 50 triples (C, C', C'') where C denotes the correct and C' and C'' the faulty ciphertexts of stage 0 and 1, respectively. The ciphertexts were generated from plaintexts and keys chosen uniformly at random, but the key remained fixed for each set of 50 triples. Table 3.6 summarizes the results of the attack and shows that on average between 4 and 5 faults are necessary to successfully reconstruct the 128-bit key.

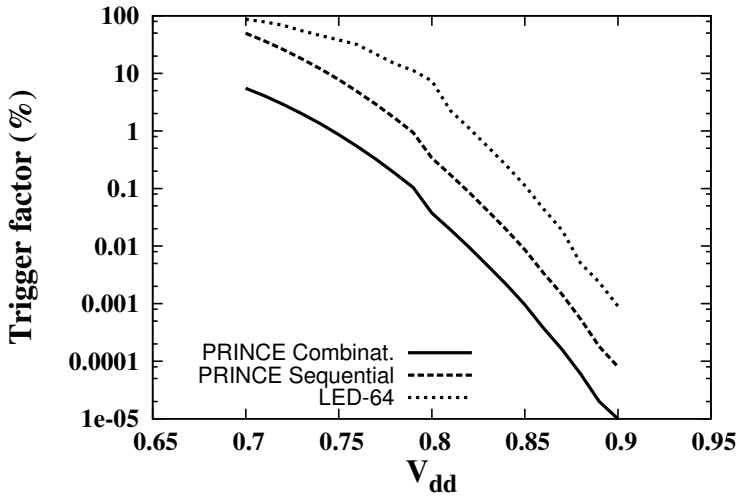


Figure 3.18. Trigger factor of the Trojan inverters inserted into LED-64 and PRINCE

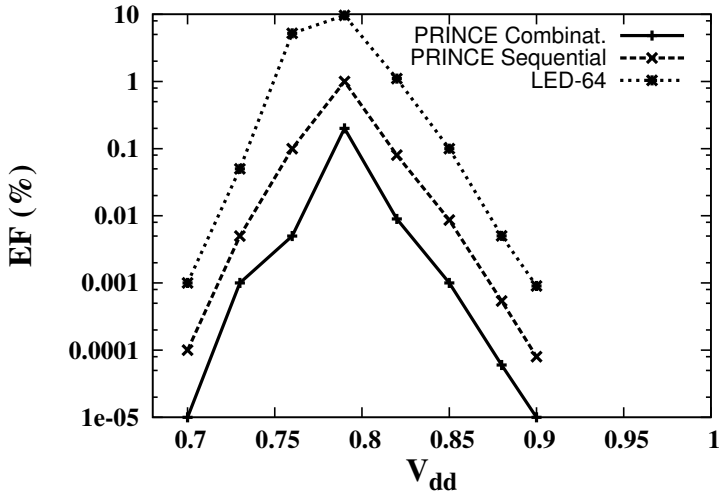


Figure 3.19. Percentage of Trojan induced exploitable faults in LED-64 and PRINCE

We set the thresholds for the multi-stage fault attack to rather low values: 2^{12} for stage 0 and 2^{16} for stage 1. In general, higher thresholds lead to less required fault injections but increase the complexity of subsequent post-processing. In our case, fault injections using Trojans or V/T manipulation are relatively easy to perform; therefore we opted for lower values and observed approximately one more required

Table 3.5. Summary of Fault-injection Attacks

Algorithm			LED-64	PRINCE	
Circuit	Type		Sequential	Combinational	Sequential
	Gate Equivalent		2256	8,320	8,540
	Clock frequency		100MHz	150 MHz	1.7 GHz
	Cycles/encryption		32	1	10
	Exploitable faults		Single nibble, round 30	Single nibble, round 8/9	
Fault effect distrib.	*V/T*	Fault-free	82.6%	49.8%	84.8%
		Exploitable	**0.4%**	**0.03%**	**0.2%**
		Not exploitable	17.0%	50.17%	15.0%
	Trojan	Fault-free	72.86%	31.0%	73.9%
		Exploitable	**9.6%**	**0.2%**	**1.1%**
		Not exploitable	17.54%	68.8%	25.0%

Table 3.6. Overview on the number of required faults

Stage	Min	Max	Avg	Median
0	2	3	2.06	2.0
1	2	11	2.84	3.0

fault on average, compared to [35]. The run time of the cryptanalysis was around 13 seconds.

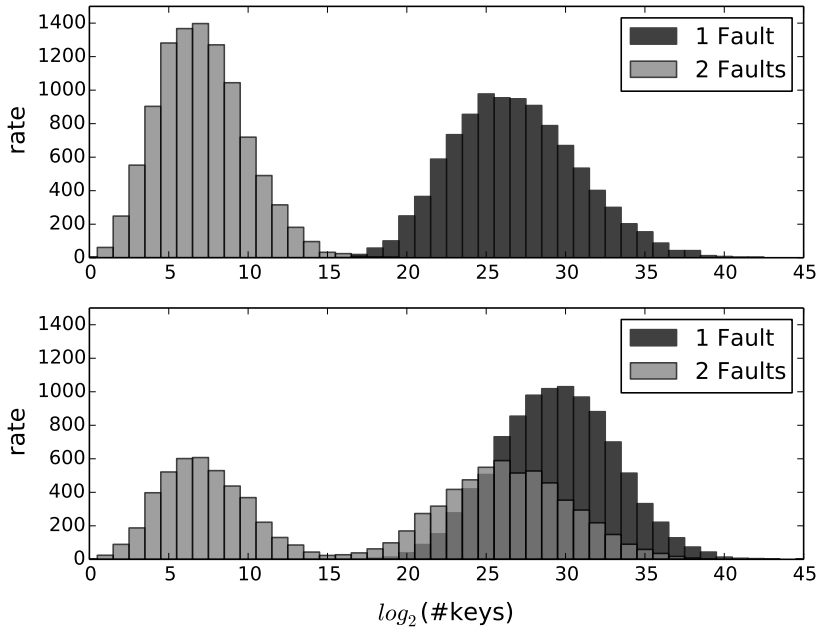


Figure 3.20. PRINCE analysis results for stages 0 (upper) and 1 (lower)

We also analysed the distributions of the number of remaining keys after one and two fault injections and compared them to the theoretical results. The distributions are shown in Fig 3.20. The upper graph shows the results for stage 0 and the lower graph for stage 1. The x-axis denotes the base-2 logarithms of the number of key candidates and the y-axis shows the (rounded) rate how often a particular number of key candidates occurred. There are cases where much more faults (up to 11) are required, but 2 faults were the minimum for every instance. As every exploitable fault requires a reasonable number ($10^4 - 10^6$) of fault injections, a complete attack using 4 – 5 exploitable faults is feasible.

CHAPTER 4

MODELING ATTACKS ON PHYSICALLY UNCLONABLE FUNCTIONS

In the previous chapter, we presented techniques for precise fault-injections into cryptographic blocks for extracting secret keys from the faulty ciphertexts using differential cryptanalysis. One of the major concerns surrounding cryptographic blocks is that the key is digitally stored and processed within the chip. Apart from fault injection attacks, the secret key can also be extracted by employing side-channel attacks. For example, by exploiting the data-dependent power information or electromagnetic radiation emanating from the device under attack, the secret key can be extracted. Moreover, the digital secrets are often stored in a non-volatile memory and they can be extremely expensive for resource constrained platforms such as RFID, FPGA etc. [86]. These constraints have served as one of the main motivations towards the development of Physically Unclonable Functions (PUFs), which offer an inexpensive way to generate unique signatures in runtime. Although initially assumed to be tolerant to various attacks, recent works have suggested otherwise.

Numerous attacks on PUFs have been reported in the literature. Most of the attacks are specific to the PUF architecture being targeted, while some of them are generic. Some of the popular attacks include modeling, side-channel, fault attacks,etc. Among this group, the most prominent one is the modeling attack with the help of a Machine Learning (ML) algorithm. ML attacks are more specific to strong PUFs, as the weak PUFs have very few CRPs and hence the CRPs won't be available for external access. The collection of various modeling attacks on some of the popular

implementations of strong PUFs can be found in [78, 79]. Attacks on generalized Arbiter, Feed-Forward, XOR Arbiter PUF constructions were first reported in [78].

Recently, attacks on PUFs using unreliable response bits have been demonstrated [17,18]. The attacks use the inherent instability in PUF responses due to device noise in [18] and the instabilities arising from voltage and temperature fluctuations in [17]. A similar work on breaking controlled Arbiter PUFs using fault- and power-side channels was reported in [8]. As the output of the Arbiter PUF is not directly available in controlled Arbiter PUFs (outputs are hashed), statistical methods employing correlation were used in [8]. Rührmair et al. have proposed a combined machine learning and side-channel attack for XOR Arbiter PUFs in [59].

In this chapter, we explore the vulnerabilities of some of the popular "strong" PUF architectures. In particular, the delay-based PUF designs are chosen as the target, as they are extremely popular because of their simplistic implementation complexities. Majority of the delay-based PUF designs can be implemented using simple logic gates and are ideal for integration into modern day embedded systems. For analyzing the vulnerabilities, the attacks are performed over the data collected from post-silicon measurements of the test chip *sugarloaf* using the methodology described in chapter 7. The vulnerabilities of delay-based PUFs are analyzed using *simple* or *standalone* machine learning algorithms (ML) using stable CRPs. However, the performance of standalone ML algorithms degrade under the presence of error-inflicted CRPs in the training set. To that extent, we propose a technique that exploits the unreliable or unstable CRPs in the training set and extracts some data-dependent information in the form of side-channels. The side-channel information is used in conjunction with a machine learning algorithm in order to improve the prediction accuracies of ML attacks. As the side-channel information is used along with a ML algorithm, they are also referred to as *hybrid attacks* further in the document. First, we describe

the architectures of the PUF targets followed by a brief description of the employed machine learning algorithms.

4.1 PUF Targets

As explained above, the delay-based PUFs were chosen as the attack target. Two popular delay-based PUF constructions, namely *arbiter* and *feed-forward arbiter* PUFs were analyzed for their vulnerabilities to standalone and hybrid attacks. The delay-based PUFs exploit the impact of process variations on the propagation delays of logic gates to generate unique signatures. The architectures of arbiter and feed-forward arbiter PUFs are explained in the next subsections.

4.1.1 Arbiter PUFs

The concept of arbiter PUFs was introduced in [51, 52, 86]. An arbiter PUF architecture is shown in Figure 4.1. Arbiter PUFs are based on delay variations in logic gates arising from process manufacturing variations in integrated circuits. The general idea is to trigger a race condition in two identically laid out paths and decide the winner among the paths using an *arbiter*. From an implementation point of view, the identical paths are built using switches or multiplexers, that accept an external challenge bit (C_i). The challenge bit decides whether the switch *passes* the input to output or *switches* it, as shown in Figure 4.1(a). The switches/multiplexers are connected in series to form a digital delay path. A switch/MUX implementation in terms of logic gates is shown in Figure 4.1(b). When a signal (say a rising pulse) is applied to the delay path, the signal undergoes different delays through every switch. At the end of the delay paths, the arbiter decides which of the paths is faster based on the instantaneous arrival times of the racing signals. If we denote the delay difference between the arrival time of racing signals at outputs (A & B) and the input trigger time as T_A and T_B respectively, then the response computation is given by equation

4.1. A common practice is to use a set-reset (SR) latch as an arbiter by connecting one of the racing paths to *set* and the other path to *reset*. It is important to note that the number of CRP pairs is exponential to the number of challenge bits. So, arbiter PUFs fall under the category of strong PUFs.

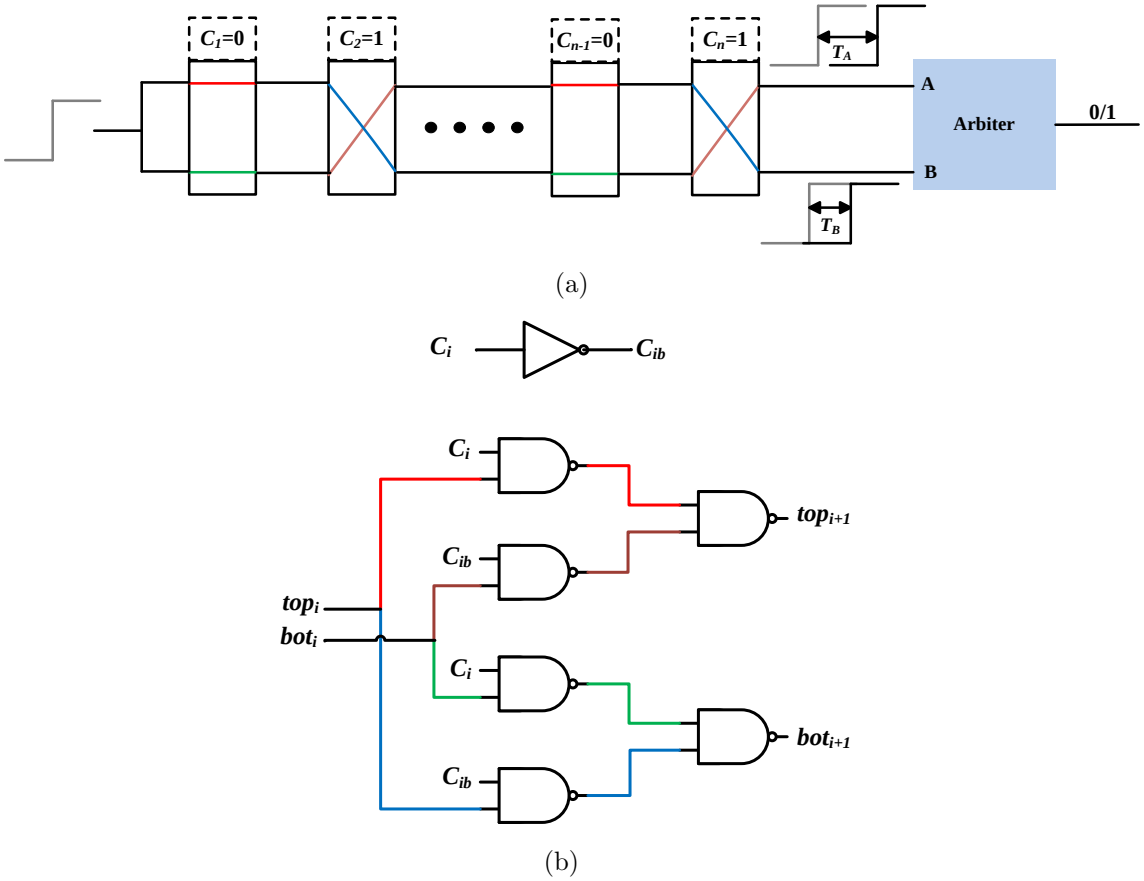


Figure 4.1. (a)Arbiter PUF architecture with n stages; (b) NAND gate based implementation of a single MUX/switch stage. The path of propagation of two signals top_i and bot_i is determined by the challenge bit C_i. If $C_i = 1$, then $top_{i+1} = top_i$ and $bot_{i+1} = bot_i$. Else, $top_{i+1} = bot_i$ and $bot_{i+1} = top_i$.

$$r = \begin{cases} 0 & \text{if } T_A > T_B \\ 1 & else \end{cases} \tag{4.1}$$

To increase the sensitivity of arbiter PUFs to process variations, it is better to use minimum sized transistors in the delay stage circuits. However, the arbiter should be

designed using up-sized transistors to tolerate process variations. The arbiter should fairly evaluate the response based on the delay difference between the racing signals (which can be positive or negative) and must not introduce any bias. To evaluate the fairness in response computation, arbiters built using D-type flip-flop and SR NAND latch were compared. D-type flip-flop arbiter in 45nm technology node has a setup time of around 20-35 ps, that introduces a bias in response computation. However, SR NAND latch arbiter has a bias of less than 3 ps because of its cross-coupled structure and it enables fair arbitration. The comparison results are shown in Figure 4.2.

4.1.2 Feed-Forward Arbiter PUFs

As the arbiter PUFs were shown to be vulnerable to modeling attacks because of its linear challenge-response behavior, non-linear PUF structures have been proposed in order to improve the PUF's resilience [51]. One such architecture is the feed-forward PUF. A feed-forward arbiter PUF exploits delay variations in CMOS gates and interconnects to extract unique signatures like an arbiter PUF [51]. However, some of the challenge bits in a feed-forward PUF are determined as a result of racing conditions at intermediate stages with the help of arbiters. The construction of a feed-forward arbiter PUF is shown in Figure 4.3. The intermediate arbiters $a_1..a_m$ generates the additional challenge bits apart from the user specified challenge $C_1..C_n$. The challenge bits $C_1..C_n$ decide the path of propagation through the delay paths whose gate-level implementation is shown in Figure 4.1(b). The path of signal propagation is determined in the same as in an arbiter PUF. The rising pulse applied at the input undergoes different delays as it gets propagated through the delay paths and the total delay difference ($\Delta t_n = T_A - T_B$) generated is sampled at the output with the help of an arbiter to generate the response bit r. The condition for response generation is similar to an arbiter PUF as demonstrated in equation 4.1.

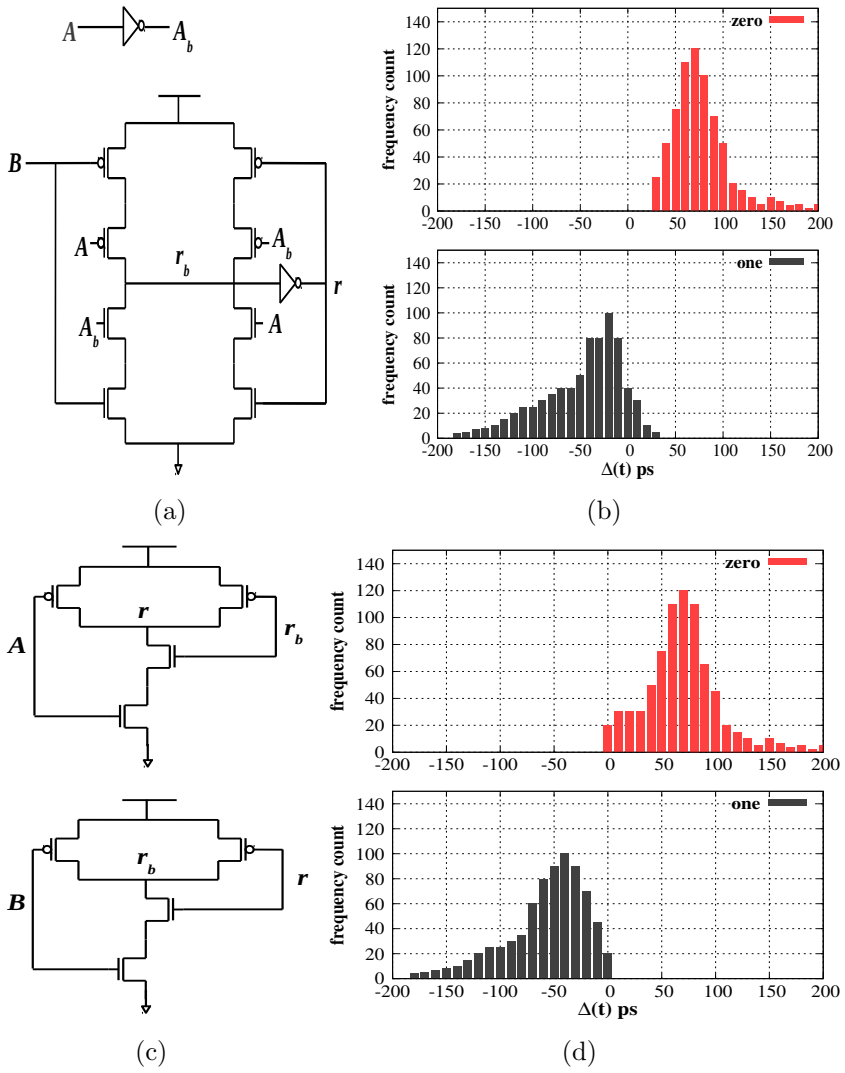


Figure 4.2. Fairness evaluation of arbiters based on bias point for choosing 0/1 output. The bias point is given by $\Delta(t) = T_A - T_B$ (a) A simple D-Type flip-flop arbiter; (b) Plot showing the bias point of d-type flip-flop arbiter for choosing 0/1 around 30 ps; (c) A simple SR-NAND latch arbiter; (d) Plot showing the bias point of SR NAND arbiter for choosing 0/1 approximately at 0 ps

4.2 PUF Target's Performance Analysis

It is important to analyze the performance metrics of the PUF circuits before mounting an attack, as they play a vital role in determining the optimal prediction accuracies. For example, if the prediction accuracy obtained from a modeling attack is below the reliability of a PUF circuit, the attack is deemed unsuccessful. So, the per-

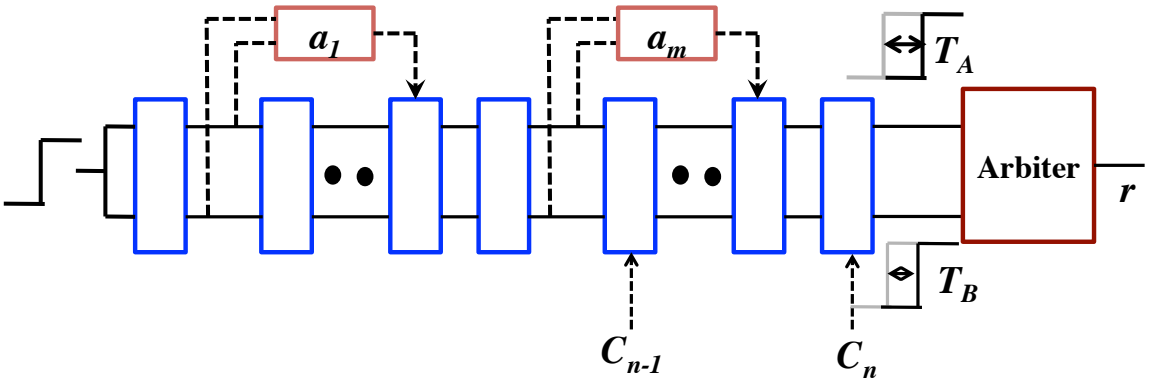


Figure 4.3. Feed-forward arbiter PUF

Table 4.1. Summary of arbiter PUF's performance metrics from statistical circuit simulations and post-silicon measurements

Type of analysis	# Stages	# PUF instances	Inter-class HD	Intra-class HD	Uniformity	Bit aliasing HD
Statistical circuit simulations	64	200	0.47	0.05	0.48	0.48
	80		0.475	0.048	0.49	0.48
	128		0.47	0.05	0.485	0.475
Post-silicon measurements	64	200	0.39	0.058	0.41	0.42
	80		0.38	0.059	0.43	0.41
	128		0.375	0.059	0.43	0.42

formance metrics of arbiter and feed-forward arbiter PUFs were analyzed as per the framework described in chapter 2. The performance metrics were obtained through statistical circuit simulations and cross-validated with post-silicon measurements. For all the analysis, 64, 80 and 128 stage PUF circuits were used.

For performing statistical circuit simulations, HSPICE was used. Different PUF instances were obtained through Monte carlo simulations as per the methodology shown in Figure 4.6. For better accuracies, the netlist of the PUF circuit was obtained using the parasitic extractor tool available in Calibre suite. Threshold voltage variations were assigned from a Gaussian distribution with 3σ deviation of 90mV (for 32nm technology). The performance metrics, namely uniqueness, reliability, uniformity and bit aliasing probability of the circuit were analyzed from circuit simulations

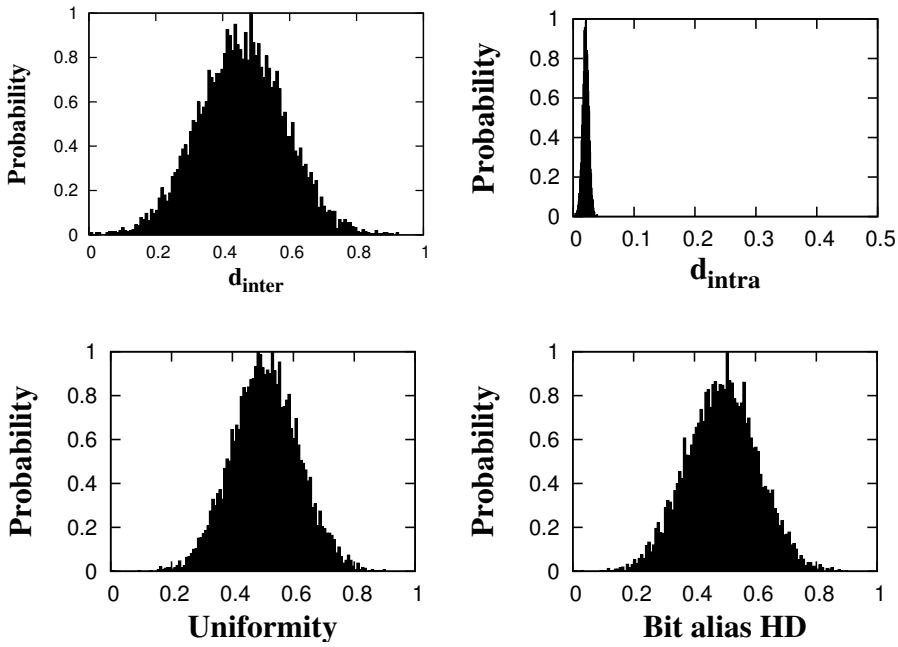


Figure 4.4. Arbiter PUF's performance metrics distribution from statistical circuit simulations

using the IBM 32nm SOI models and the results are shown in Figure 4.4. Around 100,000 CRPs were collected from 200 PUF instances for analyzing uniqueness, uniformity and bit aliasing probability. To compute reliability, the experiments were conducted at nominal and extreme operating conditions as described in chapter 2. The results are tabulated in Table 4.1. It can be observed that the arbiter PUF circuit shows excellent properties upon statistical circuit simulations and the values obtained were close to ideal values. Similar methodology was adopted to compute the performance metrics from post-silicon measurements and the results are shown in Figure 4.5. Around 200 PUF instances were analyzed over 100,000 CRPs for post-silicon validation. The post-silicon validation results agree well with statistical circuit simulation results and can be clearly seen from Table 4.1.

Similar analyses were carried out for feed-forward arbiter PUFs. However, there are different configurations possible to implement feed-forward PUFs. Some of the variable design parameters include the number of feed-forward loops, number of

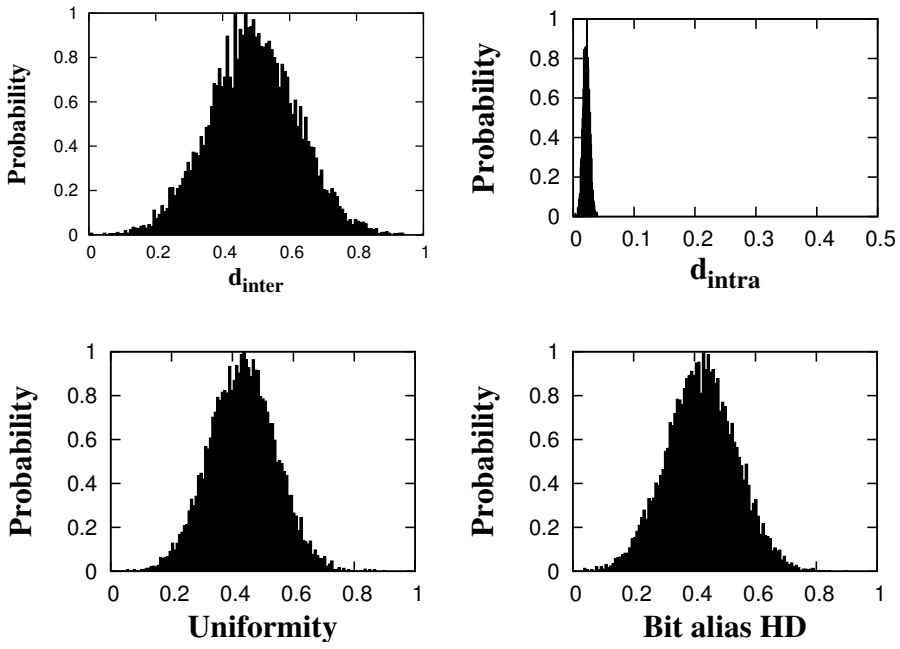


Figure 4.5. Arbiter PUF's performance metrics distribution from post-silicon measurements

stages/loop, dependency between loops, etc. Assuming the loops are symmetric, if the external challenge is k bits wide and the number of loops is l, then the total number of delay stages is $k + l$ and the number of stages/loop is k/l. As the variable design parameters impact the performance of a PUF, they were analyzed through statistical circuit simulations and post-silicon measurements. The feed-forward PUF was implemented in such a way that the number of loops and number of stages/loop can be altered using the configuration bits. The different configurations were analyzed for the different performance metrics, as per the framework described in chapter 2. Around 100,000 CRPs were collected from 200 different PUF instances for the analyses. For measuring the PUF's reliability, the experiments were conducted at nominal and extreme operating conditions and the number of bit-flips were observed. The results are shown in Figures 4.7 and 4.8 and also tabulated in Table 4.2. It can be observed that the design strategy has a significantly higher impact on reliability than uniqueness. As expected, reliability decreases with the number of loops. So, it

Table 4.2. Summary of Feed-forward arbiter PUF's performance metrics from statistical circuit simulations and post-silicon measurements

Type of analysis	# Bits in challenge (k)	# Loops (l)	# PUF instances	Inter-class HD	Intra-class HD	Uniformity	Bit aliasing HD
Statistical circuit simulations	64	6	200	0.48	0.05	0.48	0.49
		7		0.485	0.052	0.485	0.48
		8		0.47	0.055	0.47	0.485
	80	6		0.485	0.055	0.49	0.49
		7		0.48	0.0575	0.48	0.475
		8		0.47	0.059	0.49	0.48
	128	6		0.475	0.052	0.48	0.47
		7		0.48	0.056	0.47	0.48
		8		0.48	0.06	0.49	0.49
Post-silicon measurements	64	6	200	0.42	0.074	0.42	0.41
		7		0.425	0.076	0.41	0.415
		8		0.415	0.08	0.415	0.42
	80	6		0.41	0.077	0.42	0.42
		7		0.42	0.079	0.42	0.415
		8		0.415	0.0881	0.425	0.42
	128	6		0.43	0.076	0.43	0.42
		7		0.425	0.079	0.435	0.43
		8		0.415	0.0882	0.42	0.425

is highly necessary to choose the number of loops and number of stages/loop such that an optimal design is achieved in terms of reliability and unpredictability. For the PUF circuits being considered in this work (64, 80 and 128 stage PUFs), the number of loops was set to 8 ($l = 8$).

4.3 Employed Modeling Attacks

We investigated the vulnerability of the PUF circuits to three attacks namely, Logistic Regression (LR), Support Vector Machines (SVM) and Evolution Strategies (ES). Although each one of them performed well, we describe the SVM and ES attacks for the sake of brevity. Unless mentioned otherwise, all the experiments were conducted on a 32-node cluster of Intel Xeon processors.

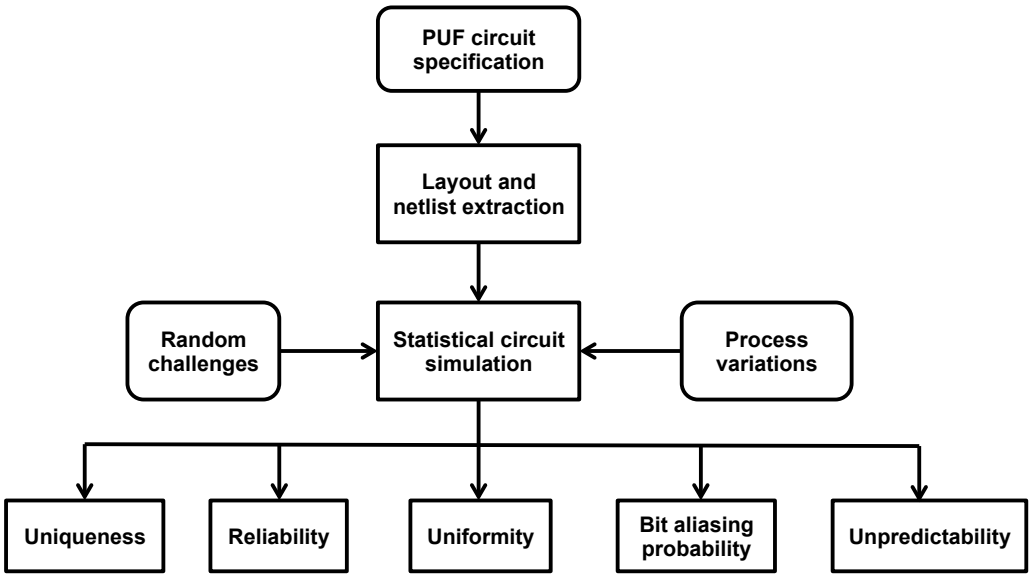


Figure 4.6. Methodology for analyzing the performance metrics of PUF circuits

4.3.1 Support Vector Machines

SVMs are well investigated learning algorithms for data classification preferably in binary form. Linear or Non-linear separating surfaces can be used for data classification depending on the input-output relationship. Some of the popular non-linear kernels include high-order polynomials, radial-basis functions, etc.

As majority of the PUF circuits have linear challenge-response relationship, we describe the linear `SVMs`. The main objective of a linear SVM is to find the hyperplane that helps to classify the datasets into a binary set as shown in Figure 4.9 [27]. The hyperplane H is given by

The hyperplane H is given by

$$w.x - b = \pm 1$$

where w is the normal vector to hyperplane H and $\frac{b}{||w||}$ is the offset of the hyperplane from the origin along w.

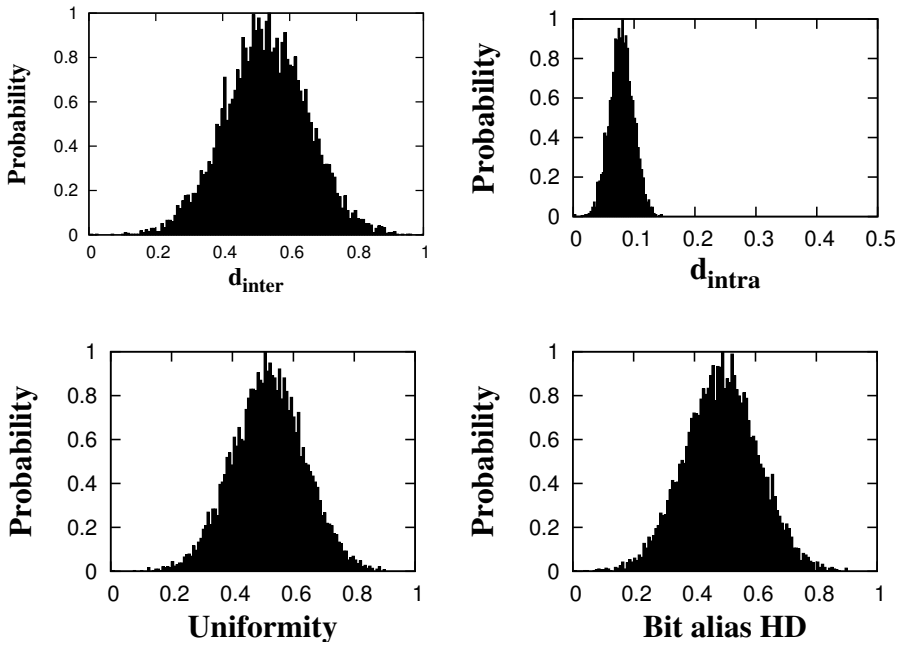


Figure 4.7. Feed-forward arbiter PUF's performance metrics distribution from statistical circuit simulations

The classification and prediction accuracy increases as the distance between the planes increases. The distance between the planes also impact the amount of mispredictions. Theoretically, the maximum distance between the hyperplanes for best classification accuracy is given by $\frac{2}{||w||}$ [27]. One of the other factors impacting the mispredictions is the non-linearity at the boundaries, that arise from the PUF circuits. Minimizing $||w||$ is not trivial and it is a constrained optimization problem better solved by Lagrangian multipliers [64]. For this purpose, we use the linear kernel of Lagrangian SVMs to find the best hyperplanes [65].

4.3.2 Evolution Strategies

Evolution Strategies are algorithms that are used for black-box optimization problems and are generally successful for PUF attacks [8, 78]. They are based on creating random models in several iterations also known as *generations* from a *parent*. The sample models are also known as off-springs. Only the off-springs that are deemed

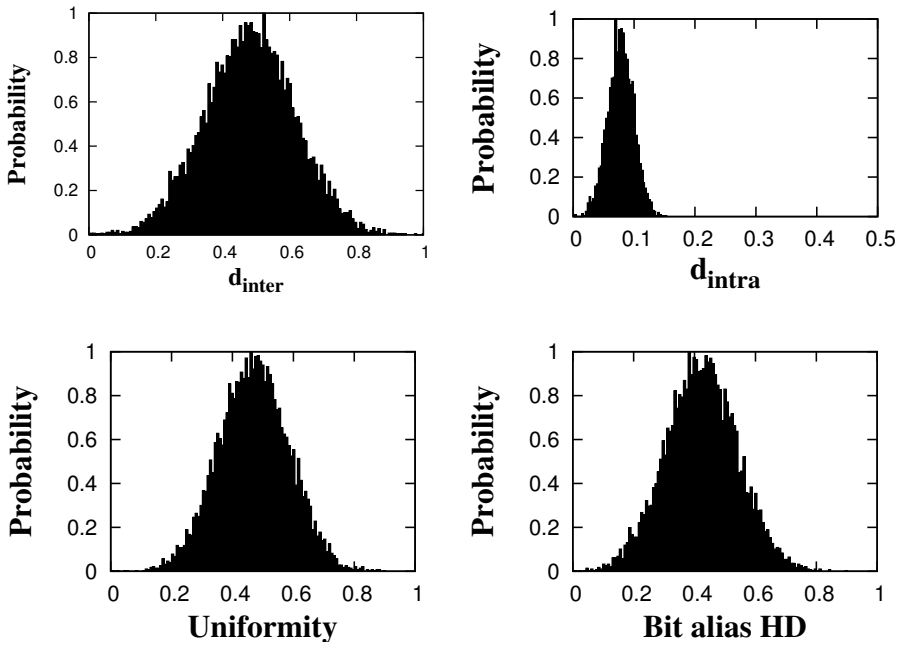


Figure 4.8. Feed-forward arbiter PUF's performance metrics distribution from post-silicon measurements

to be fit based on some fitness tests are selected and are used as parents for the next generation. The process is repeated until the convergence point (usually set by the user) is reached. The prediction accuracies for PUF attacks increase with the number of generations. The major advantage of ES is that it is completely randomized and can be parameterized [78]. ES also helps better to tackle the non-linearity from device leakage currents because of parametric nature. For example, if a weak PUF model is deemed to be fit in a generation, the convergence point will degrade in the next generation and can be discarded.

For our experiments based on ES, we set $(\mu, \lambda) = (6,36)$ and the mutation parameter $\tau = \frac{1}{\sqrt{n}}$ as in [78]. However, we evaluated the performance of attacks with and without recombination. The performance improvement with recombination ($\rho < 6$) was very marginal with respect to the attack without recombination. So, we report the attack results without recombination in this chapter.

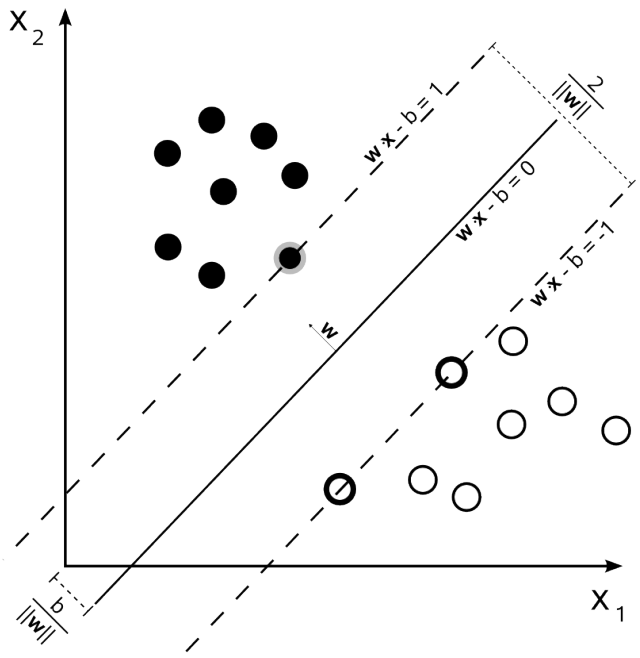


Figure 4.9. Data classification in Support Vector Machines

4.4 Modeling Delay-based PUFs

As described earlier in this chapter, the delay-based PUF designs are vulnerable to modeling attacks. We base our attack as per the framework described by Daihyun Lim in [51]. An arbiter PUF's challenge-response relationship can be expressed through an additive delay model. The top and bottom paths of an arbiter PUF shown in Figure 4.1(a) can be expressed as the sum of delays of individual stages. A direct measurement of stage delays is extremely difficult. However, they can be estimated with the help of a machine learning algorithm by observing a subset of CRPs. Each stage of an arbiter PUF can be expressed through two delay difference parameters as shown in Figure 4.10, that encode the four individual delay paths of a stage shown in Figure 4.11. The notations of individual delay components are consistent with [51]. Now, the delay parameter for C_i=1 in terms of individual delay components is shown in the following equations.

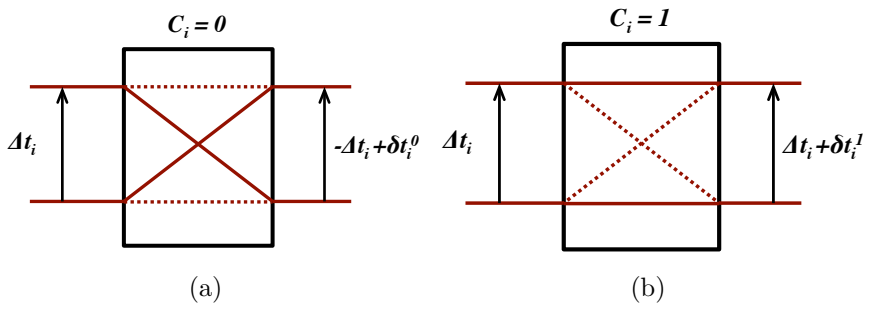


Figure 4.10. Delay difference parameters for (a) $C_i = 0$ and (b) $C_i = 1$

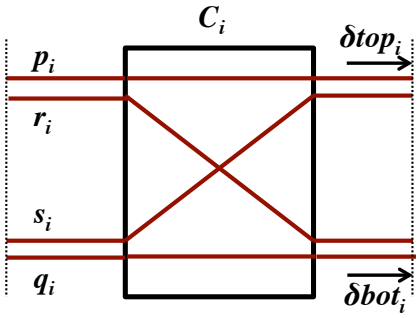


Figure 4.11. Individual delay components of a single stage of an arbiter PUF

$$\delta_{top}(i) = p_i + \delta_{top}(i-1), \delta_{bot}(i) = q_i + \delta_{bot}(i-1) \tag{4.2}$$

$$\begin{aligned} \Delta t_i &= \delta_{bot}(i) - \delta_{top}(i) \\ &= (q_i - p_i) + (\delta_{bot}(i-1) - \delta_{top}(i-1)) \\ &= \delta t_i^1 + \Delta t_{i-1} \end{aligned} \tag{4.3}$$

Similarly, the delay difference parameter for C_i=0 can be expressed.

$$\delta_{top}(i) = s_i + \delta_{bot}(i-1), \delta_{bot}(i) = r_i + \delta_{top}(i-1) \tag{4.4}$$

$$\begin{aligned}\Delta t_i &= \delta_{bot}(i) - \delta_{top}(i) \\ &= (r_i - s_i) + (\delta_{top}(i-1) - \delta_{bot}(i-1)) \\ &= \delta t_i^0 - \Delta t_{i-1}\end{aligned} \tag{4.5}$$

In equations 4.3 and 4.5, δt_i refers to the delay difference introduced by the unit stage. The delay difference parameter indicates that the delay differences (δt_i) are all that are needed to model the PUF rather than the absolute delay values (p_i, q_i, r_i, s_i). The total delay difference at the input of the arbiter denoted by Δt_n is given by equation 4.6.

$$\Delta t_n = p_0 \Delta t_0 + \sum_{i=1}^{n} p_i \delta t_i \tag{4.6}$$

In equation 4.6, $(p_0, p_1 ... p_n)$ represents the parity vector computed using equation 4.7 with p_n=1.

$$p_i = \prod_{j=i+1}^{n} C_j \tag{4.7}$$

For computing the parity vector, the challenge bits are mapped from (0,1) to (-1,1). The CRP relationship can be expressed as follows:

$$\Delta t_n \lessgtr 0 \tag{4.8}$$

The above described model serves as the basis for the modeling attacks. The results from modeling attacks are presented in the next subsection.

4.5 Modeling Attack Results for Delay-based PUFs

We evaluated the delay-based PUF targets implemented in 32nm SOI for the various ML attacks described in section 4.3. Different lengths of the PUF circuit (64, 80 and 128 stages) were analyzed in the attacks. The experiments under nominal

conditions were repeated 5 times and the stable CRPs across the iterations were separated and used for the attacks.

SVM based attacks were performed over 20 different PUF instances implemented in the test chip, For SVM based attacks, we trained the classifier with a random set of CRPs collected from the global set of CRPs (3 Million). The challenges were mapped from (0,1) to (-1,1) as in [51] and the parity vector was constructed as described in section 4.3 A subset of CRPs were picked in random for training the classifier and the rest were used to evaluating the performance of the classifier. The prediction errors as a function of the size of training CRPs for different lengths of arbiter and feed-forward arbiter PUFs are shown in Figures 4.12 and 4.13. We can observe that SVM based predictor can reach an accuracy of 99% when trained with 10000 CRPs approximately for arbiter PUFs. As SVMs are not ideal for non-linear data classification, the maximum prediction accuracy reached for feed-forward arbiter PUFs is around 85%. Better prediction accuracies for feed-forward arbiter PUFs were achieved with evolution strategies, as described in the following paragraphs.

We adopted a similar strategy as in SVM attacks for the ES attacks by picking a random set of CRPs from the global pool. In the evaluation phase, the random offsprings (count of 6) obtained from each parent were evaluated for strength based on the number of correct response bits predicted by the model. The best-fit offspring from the evaluation phase (the offspring with maximum response prediction rate for the training CRP space) was then used as a parent for the next generation and the others were discarded. The process was repeated across many generations until the prediction accuracy got saturated. The prediction errors as a function of the number of training CRPs are shown in Figures 4.14 and 4.15. From Figure 4.14, we can observe that ES attacks yield a prediction accuracy of 99% for a CRP size of 7000 approximately for arbiter PUFs. In case of feed-forward arbiter PUFs, prediction accuracies of around 98% were achieved with around 75,000 CRPs. The results of

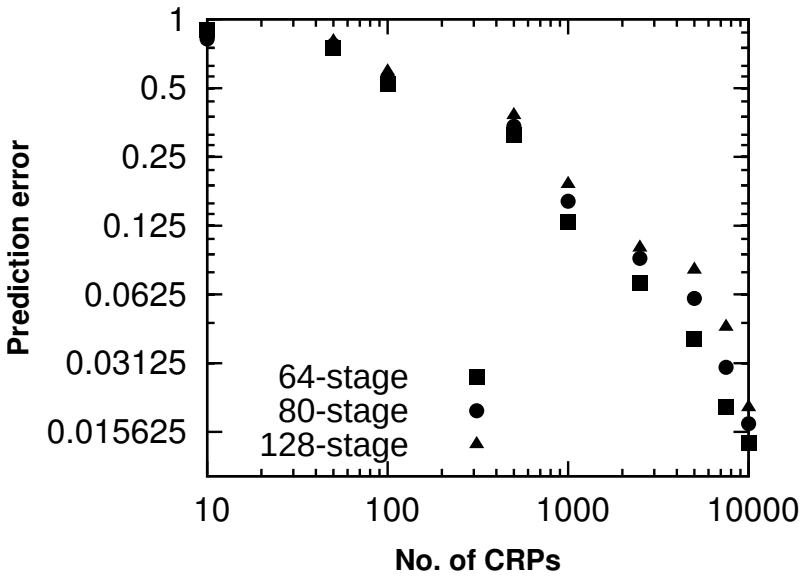


Figure 4.12. Prediction errors from SVM attacks on 64, 80 and 128 stage arbiter PUFs

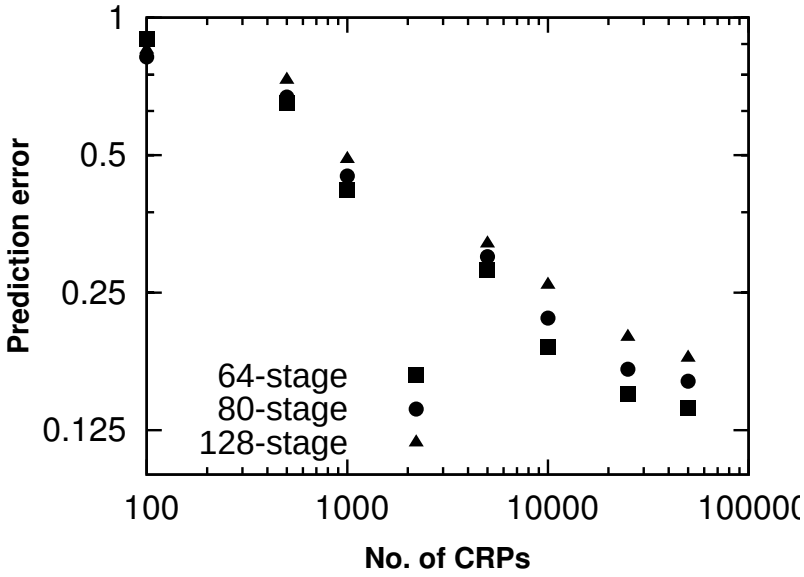


Figure 4.13. Prediction errors from SVM attacks on 64, 80 and 128 stage feed-forward arbiter PUFs

modeling attacks on arbiter and feed-forward arbiter PUFs are summarized in Tables 4.3 and 4.4, respectively. Similar to SVM based attacks, only stable CRPs were

used in the attacks. However, in real time data collection, it is highly likely that some of the CRPs are unstable. The impacts of error-inflicted CRPs on the prediction rates for arbiter and feed-forward arbiter PUFs are demonstrated in the next subsection.

Table 4.3. Summary of modeling attacks on arbiter PUFs using silicon data

Type of PUF	ML algorithm	# Stages	# Loops	# Training CRPs	Prediction accuracy (%)	Attack time
Arbiter PUF in 32nm tech.	SVM	64	-	10,000	98.8	1.2s
		80			98.5	1.2s
		128			98.2	1.2s
	ES	64		7,000	98.9	4.5s
		80			98.7	4.5s
		128			98.4	4.5s

4.5.1 Impact of Error-inflicted CRPs

The error-inflicted CRPs in the training set degrade the prediction accuracies achieved by modeling attack algorithms. The bit-flips happen when the polarity of the delay difference sampled by the arbiter is reversed. The bit-flips impact the training and prediction accuracies of the modeling algorithm. The bit-flips can happen due to

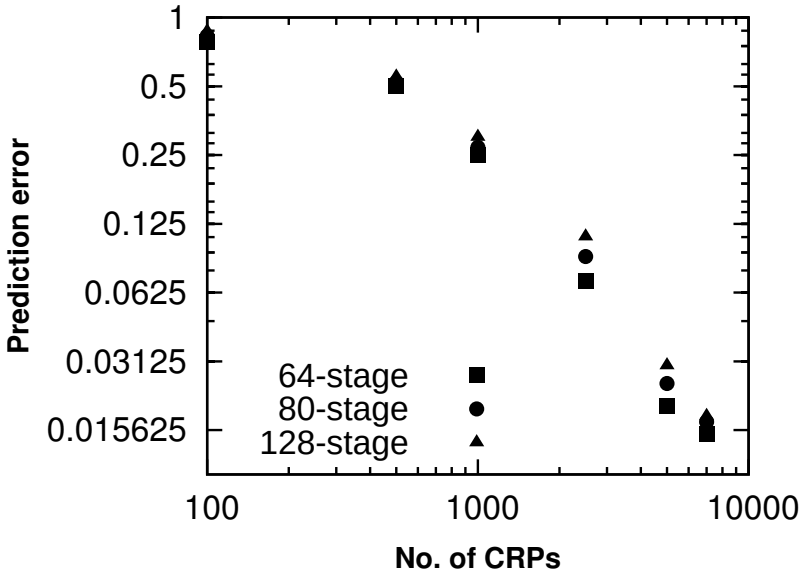


Figure 4.14. Prediction errors from ES attacks on 64, 80 and 128 stage arbiter PUFs

Table 4.4. Summary of modeling attacks on Feed-forward arbiter PUFs using silicon data

Type of PUF	ML algorithm	# Stages	# Loops	# Training CRPs	Prediction accuracy (%)	Attack time
Feed-forward PUF in 32nm tech.	SVM	64	8	70,000	84.5	7:20min
		80			85.2	7:20min
		128			85.8	7:20min
	ES	64		100,000	99.1	55:20min
		80			98.8	1:20hrs
		128			98.4	2:45hrs

internal and external noise sources. The internal noise often includes the substrate and thermal noise and are intrinsic to the ICs. On the other hand, the external noise includes the voltage and temperature fluctuations from the surrounding environment to the test setup. The amount of bit-flips from internal and external noise sources in delay-based PUF circuits are tabulated in Table 4.5. For observing the impact of internal noise, the measurements were conducted at optimal conditions (0.9V and 25° C) for 5 times and the amount of bit-flips for the same challenges were collected. We can observe that around 2.6% of the 3 Million CRPs are unreliable and have

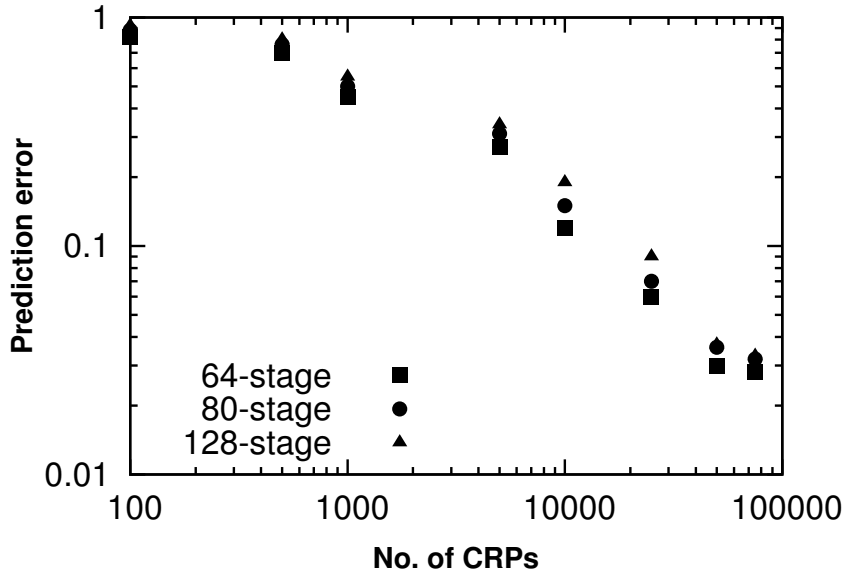


Figure 4.15. Prediction errors from ES attacks on 64, 80 and 128 stage feed-forward arbiter PUFs

higher chance to get flipped under the same measurement conditions. For observing the impact of external noise, the measurements were conducted at extreme operating conditions (1.1V and 75° C) and the bit-flips were collected. From table 4.5, we can observe that around 5.8% of the total responses from arbiter PUFs are susceptible to fluctuations in voltage and temperature conditions. In case of feed-forward arbiter PUFs, around 8.2% of the CRPs were unstable. The amount of bit-flips are shown for feed-forward PUF constructions with $l = 8$.

Table 4.5. Summary of the bit-flips measurements for delay-based PUF designs in 32nm technology

Type of PUF	**# Stages**	**Measurement**	**# Measured CRPs**	**# Unreliable CRPs**	**% Bit-flips**
Arbiter PUF	64	Intrinsic	3*10^6	78,300	2.61
	80			78,500	2.62
	128			79,120	2.63
	64	Extrinsic		175,200	5.85
	80			178,100	5.93
	128			179,200	5.97
Feed-forward Arbiter PUF	64	Intrinsic		81,200	2.7
	80			81,650	2.72
	128			80,800	2.69
	64	Extrinsic		263,900	8.8
	80			264,200	8.81
	128			264,500	8.82

The impact of the error-inflicted CRPs on the prediction accuracies for SVM and ES based attacks for arbiter PUFs are shown in Figures 4.16 and 4.17, respectively. Evolution strategies were able to handle the error-inflicted CRPs better than SVM, because of their parametric nature as observed from Fig 4.17. The degradation in prediction accuracy for 6% error-inflicted CRPs is around 4% for ES and around 11% for SVM. The impact of the error-inflicted CRPs on SVM and ES attacks for arbiter PUFs are summarized in Table 4.6.

Similar to arbiter PUFs, the impact of the error-inflicted CRPs on prediction rates for feed-forward arbiter PUFs were analyzed. The results are shown in Figures 4.18 and 4.19. Although ES based attacks are highly tolerant to errors in CRPs, there is

a slight drop in prediction accuracy (around 3%). In case of SVM based attacks, the drop in prediction accuracy is around 13%.As the error-inflicted CRPs have a reversed polarity in delay difference, the learning phase of the modeling attack is impacted.

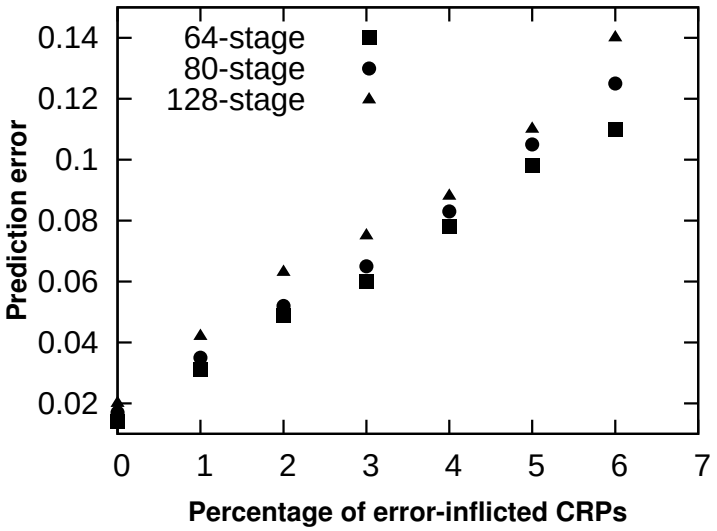


Figure 4.16. Impact of the error-inflicted CRPs on the prediction rates of SVM attacks for arbiter PUFs

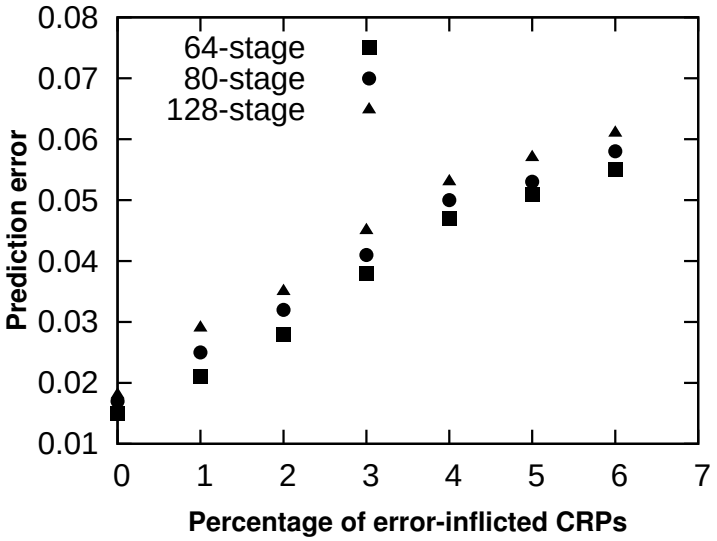


Figure 4.17. Impact of the error-inflicted CRPs on the prediction rates of ES attacks for arbiter PUFs

Table 4.6. Summary of the impacts of error-inflicted CRPs on prediction rates for arbiter PUFs

Type of PUF	ML algorithm	# Stages	% Error-inflicted CRPs			
			0	**2**	**4**	**6**
Arbiter PUF	SVM	64	98.8	95.1	92.2	89
		80	98.5	94.8	91.7	87.5
		128	98.2	93.7	91.2	86
	ES	64	98.9	97.2	95.3	94.5
		80	98.7	96.8	95	94.2
		128	98.4	96.5	94.5	93.9

However, not all the CRPs suffer from polarity reversal. Only the challenges for which the delay difference is lower than a threshold are susceptible. This threshold delay difference can leak some information and can be used along with the training set CRPs to improve the prediction accuracies. The details are described in the next section.

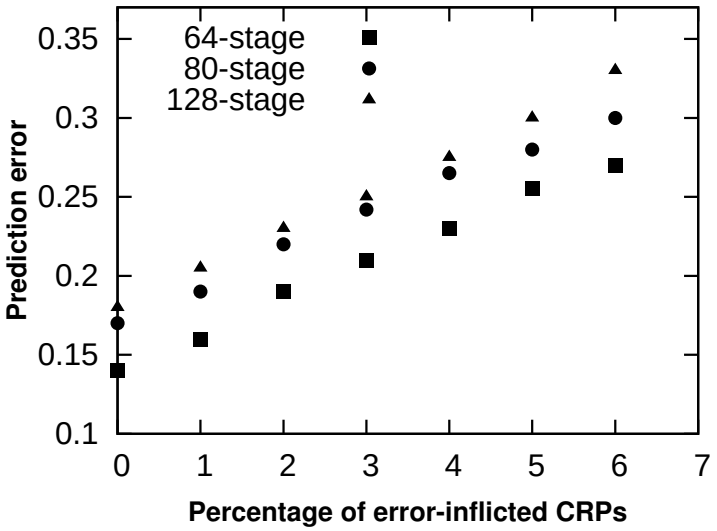


Figure 4.18. Impact of the error-inflicted CRPs on the prediction rates of SVM attacks for Feed-forward arbiter PUFs

Table 4.7. Summary of the impacts of error-inflicted CRPs on prediction rates for Feed-forward arbiter PUFs

Type of PUF	ML algorithm	# Stages	# Loops	% Error-inflicted CRPs			
				0	**2**	**4**	**6**
Feed-forward arbiter PUF	SVM	64	8	84.5	81.2	77.5	73.1
		80		85.2	78.1	73.4	70.4
		128		85.8	77.4	72.3	69.2
	ES	64		99.6	97.1	96.6	95.8
		80		98.8	96.5	95.9	95.6
		128		98.4	96.4	95.7	95.5

4.6 Hybrid attacks on delay-based PUFs

As described earlier, the unreliable or unstable CRPs can leak some information in order to improve the prediction accuracies of standalone modeling attacks. In case of delay-based PUF designs, the vulnerable delay-difference data is used to improve the accuracy of PUF models and also filter out the weak PUF models. Evolution strategies based attacks work better with the fault-injection data because of its parametric nature. As ES based attacks performed better with vulnerable delay-difference data than SVM, the hybrid attacks are based on ES framework.

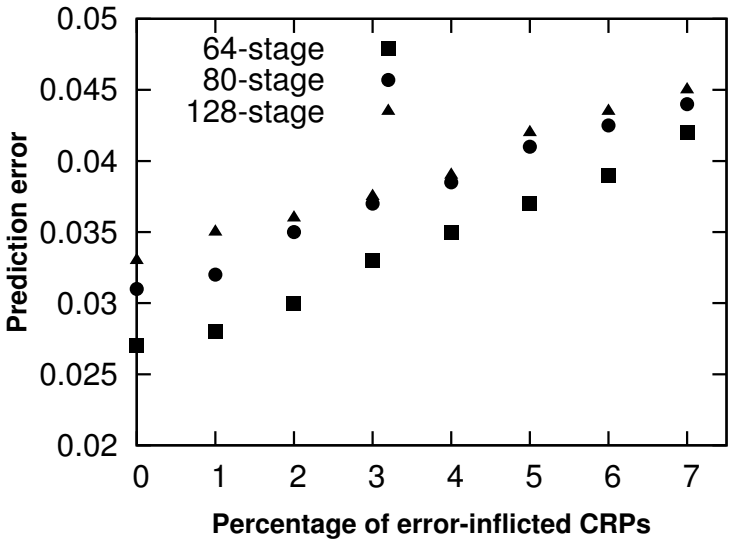


Figure 4.19. Impact of the error-inflicted CRPs on the prediction rates of ES attacks for Feed-forward arbiter PUFs

4.6.1 Scope of Fault Injection

Among the techniques to inject faults, voltage and temperature manipulation (V/T) is one of the most popular techniques. Altering the environmental conditions will make the circuit to deviate from the nominal operating conditions, thereby causing a malfunction. As explained in the previous sections, PUF circuits suffer from both intrinsic and extrinsic noises. The delay-based PUFs when operated at extreme operating conditions (1.1V and 75° C) produce more unreliable CRPs than the unreliable CRPs induced by intrinsic noise. Although this is detrimental to the PUF's operation, some data dependent information can be extracted. One such information is the amount of delay-difference that is sensitive to V/T variations.

To estimate the amount of threshold delay-difference (Δt_{min}), Monte carlo simulations were performed over the arbiter and feed-forward arbiter PUF circuits using the 32nm SOI transistor models. Threshold voltage variations were assigned from a Gaussian distribution with 3σ deviation of 90mV. Around 1 Million CRPs along with the corresponding delay-difference (Δt_n) values were collected from spice simulations. For injecting faults, experiments were conducted by changing the operating conditions (0.8 to 1V in steps of 0.05V and 25 to 75° C in steps of 25° C) and the above mentioned data were collected. The error-prone response bits under different operating conditions when compared to the nominal conditions (0.9V and 25° C) were mapped to the corresponding delay-difference values. The CRPs for which the delay-difference was close to 0s were highly vulnerable to changes in operating conditions and the behavior can be clearly observed from Figures 4.20 and 4.21. The Figures 4.20 and 4.21 show the error-free and error-prone CRPs for arbiter and feed-forward arbiter PUFs respectively. As similar characteristics were observed for different PUF sizes, the distributions for 128 stage PUF circuits are shown. It can be observed that the error-prone CRPs have Δt_n around $5ps$ and $6ps$ for arbiter and feed-forward ar-

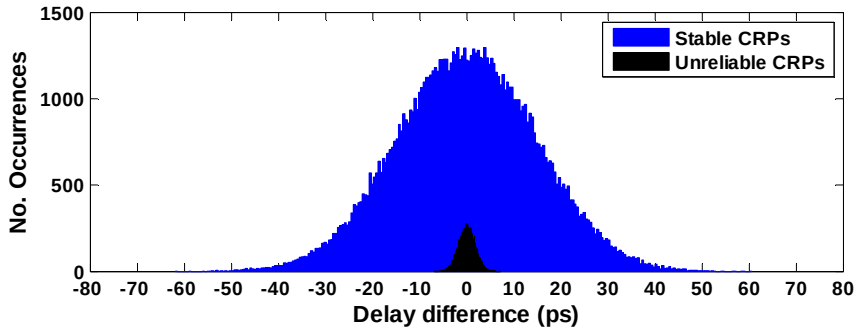


Figure 4.20. Delay-difference distributions of error-free and error-inflicted CRPs from arbiter PUFs

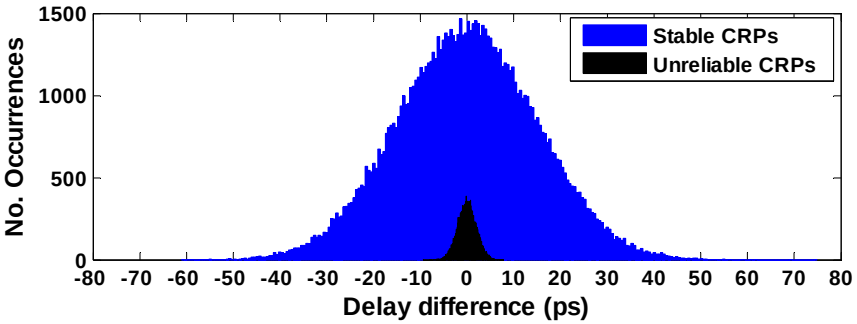


Figure 4.21. Delay-difference distributions of error-free and error-inflicted CRPs from feed-forward arbiter PUFs

biter PUFs respectively. These values ($5ps$ and $6ps$) were taken as Δt_{min} for arbiter and feed-forward arbiter PUFs.

In delay-based PUF designs, not all the error-prone CRPs are caused by fluctuating operating conditions. Some of the error-prone CRPs are caused by irregularities in the arbiter itself. So, the error-prone CRPs with $\Delta t_n < \Delta t_{min}$ induced by the impact of operating conditions on delay stages were seperated. The plot showing the amount of flipped bits with $\Delta t_n < \Delta t_{min}$ observed under different operating conditions for arbiter and feed-forward arbiter PUFs are shown in Figures 4.22 and 4.23, respectively. It can be observed that the error-prone CRPs satisfying Δt_{min} condition are slightly lower than intra-class HD shown in Table 4.5. The amount of instable

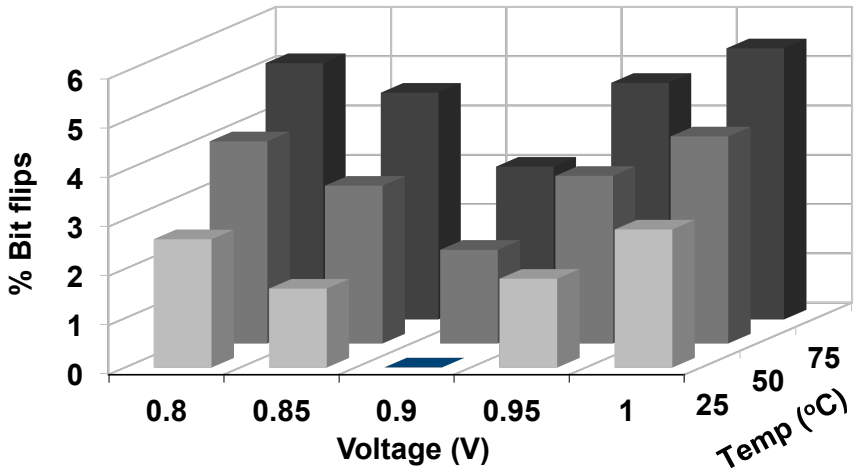


Figure 4.22. Amount of bit-flips with $\Delta t_n < \Delta t_{min}$ for arbiter PUFs

CRPs can be increased further by considering the CRPs with delay-difference higher than Δt_{min} by altering the operating conditions even further (> 1.1V). However, we set the limits to around 10% of nominal V_{dd}, as they correspond to the typical power supply noise observed in an IC. Moreover, some of the security ICs may be equipped with on-chip voltage detector to avoid over- and under-powering attacks. From the analysis shown above, it is clear that if a response bit flips, then it is most likely to have a delay-difference less than Δt_{min}. This information is used as a catalyst for aiding modeling attacks.

4.6.2 Fault-injection assisted ES attacks

In hybrid attacks, the ES algorithm with the same parameters described in section 4.3 was used. But, the threshold Δt_{min} data was used to filter out the bad PUF models. In a simple ES attack, the filtering process is done based on the number of correct response bits predicted by the model. However, in hybrid attacks, the filtering process is done based on the number of correct response bits and the extent to which the PUF model adheres to the threshold model. This is done by evaluating the correlation coefficient between a hypothesis vector and the golden model constructed

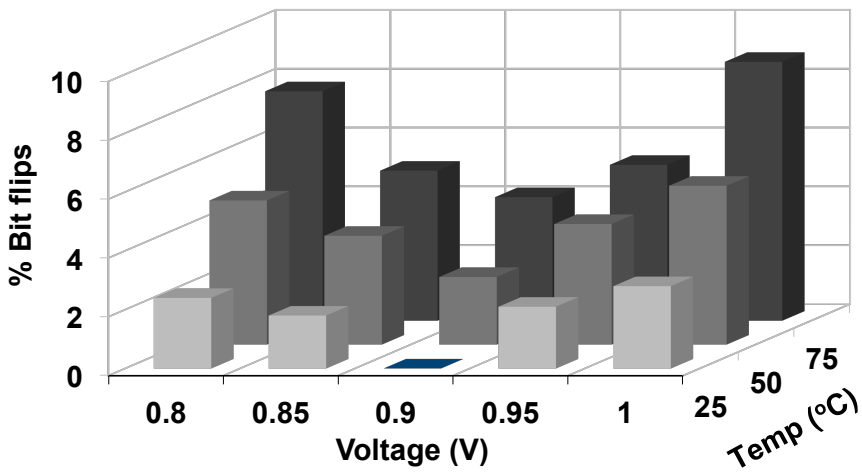


Figure 4.23. Amount of bit-flips with $\Delta t_n < \Delta t_{min}$ for feed-forward arbiter PUFs

from the observed CRPs. The hypothesis vector (F_H) is obtained from the ES model by evaluating the delay differences (Δt_H) for the training CRPs, especially for the error-prone ones. It is given by,

$$F_H = \begin{cases} 1, \text{if} |\Delta t_H| < \Delta t_{min} \\ \\ 0, \text{if} |\Delta t_H| > \Delta t_{min} \end{cases} \tag{4.9}$$

The same process is repeated to obtain the golden vector F over the measured CRPs from the test chip. If the response bit r_i got flipped under extreme conditions, then F_i is assigned to 1, else F_i is assigned to 0. By evaluating the correlation-coefficient between F and F_H, the strength of the hypothesis vector is obtained. The models which yield the best fit in terms of the number of correct response bits predicted and correlation co-efficient are passed onto the next generation. Better models were obtained by increasing the number of unstable CRPs in the training set as shown in Figure 4.24.

Hybrid attack experiments were performed over 20 PUF instances from the test chip. The attacks were performed using 100 sets of random CRPs picked from the global pool (3 million) and the results were averaged. In the training set, around 7% of

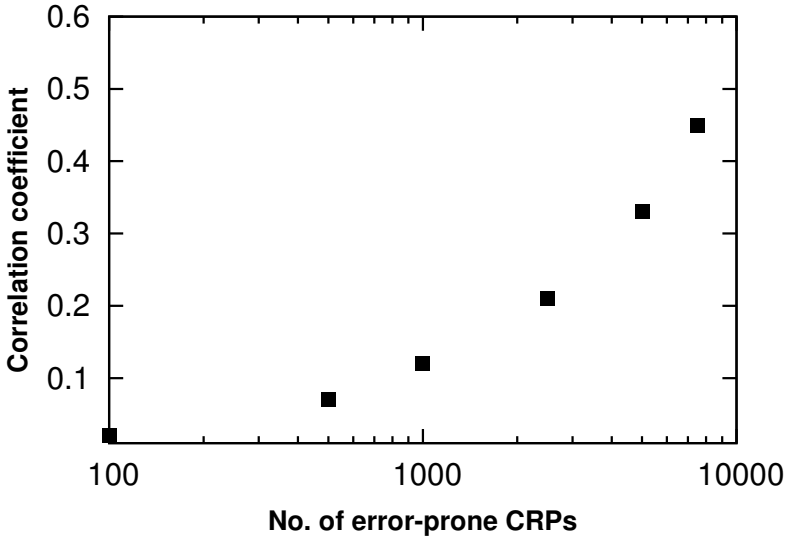


Figure 4.24. Impact of the number of error-inflicted CRPs on the strength of PUF models

the CRPs were error-inflicted which were obtained from fault-injection attacks. The performance of hybrid attacks over arbiter and feed-forward arbiter PUFs are shown in Figure 4.25 and 4.26 respectively. The best-case prediction accuracy achieved from hybrid attacks is over 99% for the delay-based PUF designs. This represents a 4% improvement in performance over the simple ES attack with error-inflicted CRPs for arbiter and feed-forward arbiter PUFs. The major advantage with hybrid attacks is that better prediction accuracies are achieved with almost the same number of CRPs as in simple ES attacks. The best-case prediction accuracies from hybrid attacks along with attack time overheads are shown in Table 4.8.

Some of the major results from modeling and hybrid attacks are summarized in Table 4.9. In this chapter, the vulnerabilities of delay-based PUFs to modeling attacks were clearly demonstrated. The impacts of error-inflicted CRPs on prediction rates of machine learning algorithms were also presented. To that extent, we presented a technique to exploit error-prone CRPs for improving the prediction accuracies of machine learning attacks under the presence of error-inflicted CRPs.The vulnerabilities

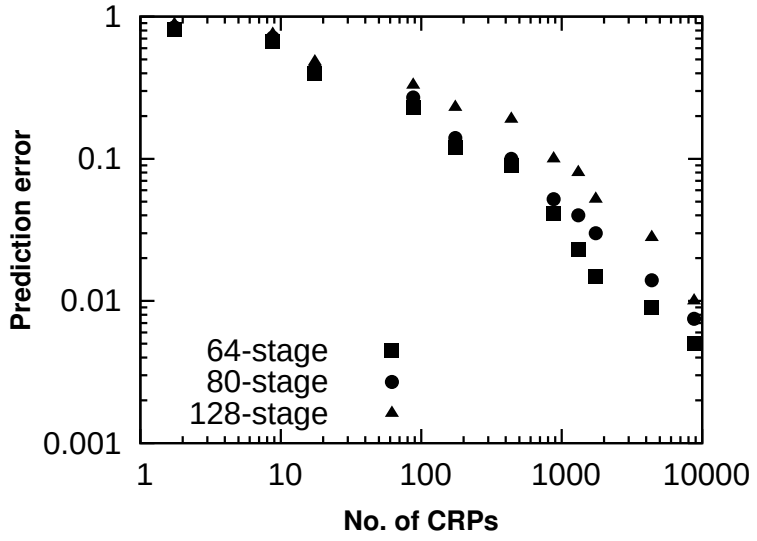


Figure 4.25. Performance of hybrid attacks on arbiter PUFs under the presence of 6% error-inflicted CRPs

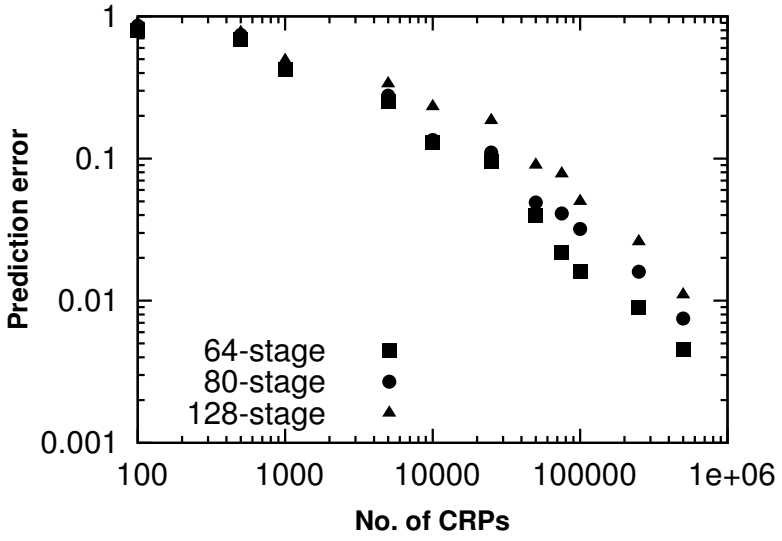


Figure 4.26. Performance of hybrid attacks on feed-forward arbiter PUFs under the presence of 7% error-inflicted CRPs

of existing delay-based PUF circuits impose a strong pressing need on the design of new secure PUF architectures. To that end, we propose a secure PUF architecture

Table 4.8. Summary of hybrid attack's performance on arbiter and feed-forward arbiter PUFs using silicon data

Type of PUF	**# Stages**	**# Loops**	**# Training CRPs**	**Prediction accuracy (%)**	**Attack time**
Arbiter PUF	64	-	9,000	99.5	7.5s
	80			99.25	7.5s
	128			99	7.5s
Feed-forward Arbiter PUF	64	8	100,000	99.4	65:10min
	80			98.9	1:30hrs
	128			98.8	3:10hrs

that is tolerant to modeling attacks. The PUF architecture along with post-silicon validation results are presented in the next chapter.

Table 4.9. Summary of some major results from modeling and hybrid attacks on delay-based PUFs

Type of PUF	**Type of attack**	**# Loops**	**# Training CRPs**	**Prediction accuracy (%)**
Arbiter PUF	SVM	-	10,000	98.2
	ES with stable CRPs		7,000	98.4
	ES with 6% error-inflicted CRPs		7,000	93.9
	Hybrid		9,000	99
Feed-forward Arbiter PUF	SVM	8	70,000	85.8
	ES with stable CRPs		100,000	98.5
	ES with 7% error-inflicted CRPs		100,000	95.5
	Hybrid		100,000	98.8

CHAPTER 5

MODELING ATTACK RESISTANT PUF DESIGN BASED ON NON-LINEAR ELEMENTS

In the previous chapter, the vulnerabilities of delay-based PUFs to modeling and hybrid attacks were presented. Since PUF based systems offer tremedous potential to replace traditional cryptography, they should be designed in such a way that they are resistant to well-known attacks. To that extent, we propose a novel modeling attack resistant PUF architecture based on non-linear computing elements [1]. As the proposed PUF is based on current mirrors, the performance of the proposed PUF is compared against a popular current-based PUF architecture. So, we take a closer look at the vulnerabilities posed by current-based PUFs [61] using the data from *sugarloaf* [2]. Modeling and hybrid attacks are used to evaluate the security of current-based PUF circuit.

Motivated by the vulnerabilities of existing current-based PUF circuits, we propose a secure PUF based on non-linear current mirrors, which is termed as "nlcPUF" further in the document. A current mirror is a basic block in analog circuits, whose function is to copy the input current in the read-node to its output node. The amount of copied current is highly dependent on the amount of device mismatch, if minimum sized transistors are used. The basic principle in nlcPUF design is to use a non-linear current mirror, that shifts the input current by some amount depending on the strength of the input current. A non-linear current mirror can be designed by

[1] This work was published in [44]

[2] This work using simulated data was published in [43]

utilizing a constant current source along with a simple current mirror, in order to introduce a threshold to the input current. The nlcPUF architecture consists of two identical non-linear current mirror chains connected at the input to a common current source. The currents through the unit blocks are then propagated through a current switching element, which accepts an external challenge bit. The construction is similar to an arbiter PUF architecture. However, in an arbiter PUF, the amount of delay introduced by a stage remains fixed for a challenge bit. In nlcPUF construction, the amount of current introduced by a stage is dependent on the input current, which is due to the non-linearity in the current mirror's transfer characteristic. Post-silicon validation of nlcPUF using 32nm CMOS SOI shows that the PUF has excellent statistical properties in terms of inter- and intra-die distances. One of the most striking features is the low information leakage demonstrated in terms of its modeling attack resistance. The attacks mounted on nlcPUF using SVM and ES show a significant increase in attack resistance almost 50x higher than other strong PUF architectures. The same holds good for hybrid attacks as well, where the error-inflicted CRPs are exploited for improving the prediction accuracy. Before describing the architecture and performance evaluation of nlcPUFs, we present the vulnerabilities of a popular current-based PUF architecture against which nlcPUFs are compared.

5.1 Current-based PUFs

In this section, we provide an overview of Current-based PUFs. Current-based PUFs were first introduced by Majzoobi et al. in 2012 [61]. The architecture of Current-based PUF is shown in Fig 5.1. Current-based PUFs are based on the addition of process variation sensitive currents. As shown in Fig 5.1, the process variation sensitive currents are generated by the current generation (CG) transistors. The transistors are of minimum size in order to maximize the impact of process variations. The gate voltage of the CG transistors are controlled by a bias voltage V_g. The external

challenge decides the set of currents to be selected and combined through the select and combine transistors (SC). A CG transistor together with the SC transistors serve as the building block of the PUF circuit as highlighted in Fig 5.1. When the challenge bits controlling the SC transistors of a unit stage ($C^a[i], C^b[i]$) are at logic high, the generated current is split in half when the transistors are matched. However, under the presence of process variations, the current split ratio deviates from 0.5. If only one of the challenge bits of a unit stage are at logic high, then the generated current directly flows through the SC whose gate voltage is at logic high. Finally, if none of the challenge bits of a unit stage are at logic high, only device leakage currents flow through the SC transistors. If we assume the current split ratio between the SC transistors of a unit stage as α_i, β_i and the leakage current as $I[i]^L$ and 'don't care' as X, the currents flowing through the SC transistors can be expressed in terms of the following equation.

$$I^a[i] = \begin{cases} I[i], & \text{if } C^a[i] = 1 \text{ and } C^b[i] = 0; \\ \alpha_i I[i], & \text{if } C^a[i] = 1 \text{ and } C^b[i] = 1; \\ I[i]^L, & \text{if } C^a[i] = 1 \text{ and } C^b[i] = X; \end{cases} \tag{5.1}$$

$$I^b[i] = \begin{cases} I[i], & \text{if } C^a[i] = 0 \text{ and } C^b[i] = 1; \\ \beta_i I[i], & \text{if } C^a[i] = 1 \text{ and } C^b[i] = 1; \\ I[i]^L, & \text{if } C^a[i] = X \text{ and } C^b[i] = 0; \end{cases} \tag{5.2}$$

The currents from all the stages are then combined and the total currents are denoted as I^a and I^b as shown in Fig 5.1. These currents are then compared using a sense amplifier to generate a single response bit *output*. The trigger signal *trig* is asserted low before the challenges are spplied such that the output nodes *output* and $output_b$ are precharged to V_{dd}. Once the trigger signal is asserted high, the challenges

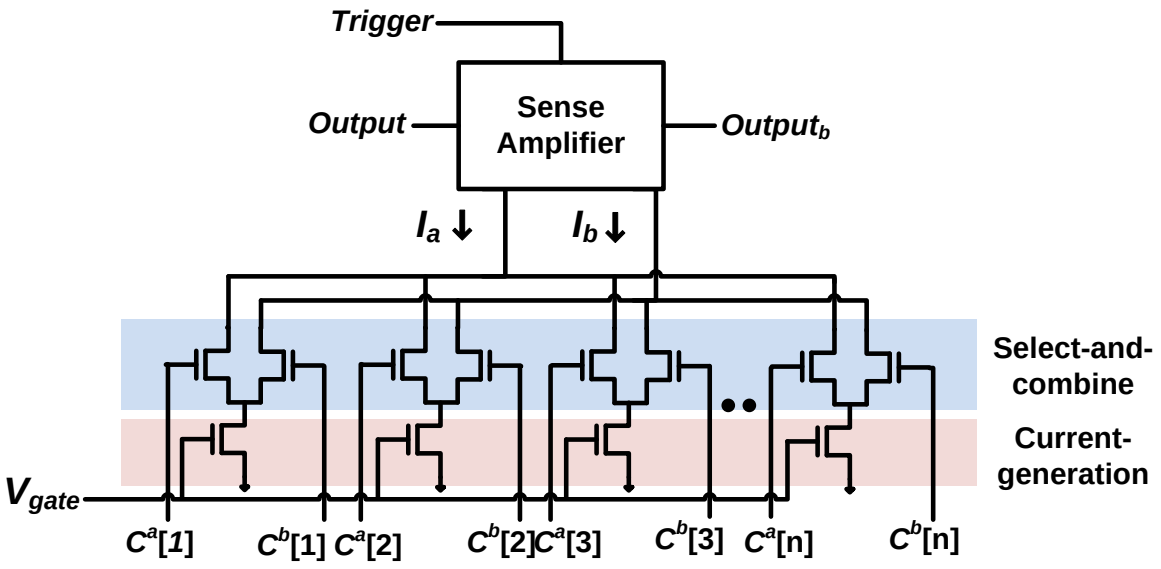


Figure 5.1. Current-based PUF architecture [61]. $C^a[i]$ and $C^b[i]$ represents the challenge bits of a single stage. The inputs to the sense amplifier are the currents I_a and I_b. $output_b$ refers to the complimentary form of the output bit.

are appled and the currents start flowing through the sense amplifier. Based on the relative strengths of the currents I^a and I^b, one of the nodes starts discharging faster and establishes a positive feedback to settle the metastability.

The current-based PUFs were evaluated against the modeling attacks presented in section 4.3. The parametric model for current-based PUFs are presented in the next section.

5.2 Attacks on Existing Current-based PUFs

In this section, we present the attack model and results obtained for the Current-based PUFs described in section 5.1. The CRPs were collected from the 32 nm test chip as discussed in chapter 7.

5.2.1 Attack Model

The Current-based PUF shown in Figure 5.1 is based on the linear combination of process-sensitive currents, where the combining function is decided by an external challenge $C = [(C^a[1], C^b[1]), (C^a[2], C^b[2]), ..., (C^a[N], C^b[N])]$, where N represents the length of the Current-based PUF. So, for a N stage PUF, the challenge is $2N$ bits wide. Moreover, the challenges are chosen such that the number of ones in the left and right input challenges are equal. If not, the responses become biased because of the fact that the number of currents that are combined are not balanced and the response can be predicted before the current difference is sampled. This is achieved if the following condition is met:

$$\sum_{i=1}^{N} C^a[i] = \sum_{i=1}^{N} C^b[i]. \tag{5.3}$$

Every unit stage of the Current-based PUF can be represented using four parameters, as shown in Figure 5.2. The parameters are nothing but the difference of the currents flowing through the SC transistors of a unit stage. Based on the attacks, we observed that the parameter where both the challenge bits of a unit stage are '0' ($\Delta I[i]^{\{0,0\}}$) does not impact the prediction accuracy very much. This is mainly due to the fact that whenever both the challenge bits are '0', only device leakage currents flow through the SC transistors and hence the difference becomes highly negligible.

Similar to attacks on arbiter PUFs as described in [51], it is necessary to express the current difference ($\Delta I[i]$) as a function of the challenge bits and the total current generated by the CG transistor. Please note that the sign of the current difference ($+$ or $-$) is all that is required to break the PUF rather than the absolute value. It is highly difficult to measure the amount of current split and the device leakage current through physical measurements. However, they can be learned through ML algorithms by collecting a subset of CRPs.

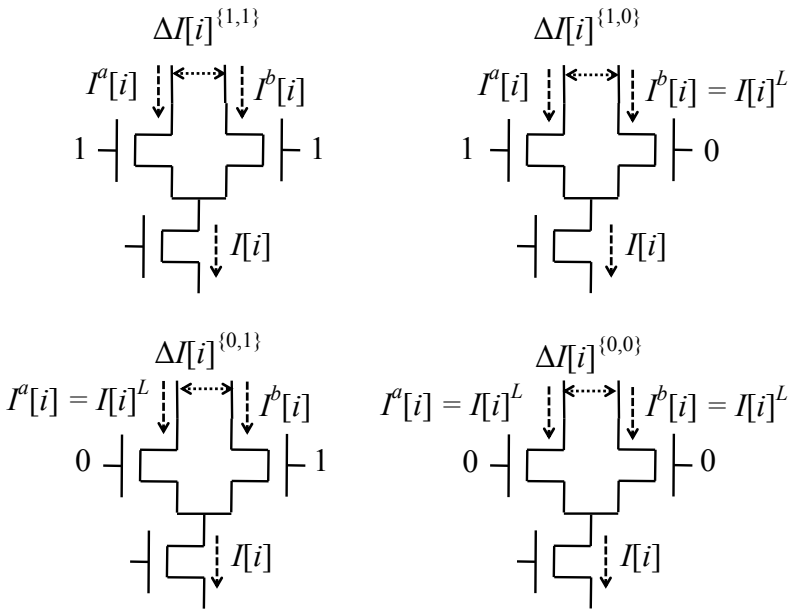


Figure 5.2. Current difference modeling parameters for Current-based PUFs

Before expressing the current difference of a unit stage formally, let us fix the notations. The current difference for a unit stage is denoted by $\Delta I[i]^{\{C^a[i],C^b[i]}\}$, where $C^a[i], C^b[i]$ are the gate inputs of the SC transistors (challenge bits). The device leakage current flowing through the SC transistor whose gate input is at logic low is denoted as $I[i]^L$. Let α_i, β_i represent the current split ratio of a unit stage such that $\alpha_i + \beta_i = 1$. The formal expressions for the current difference parameters are as follows:

$$\Delta I[i] = I^a[i] - I^b[i]$$

$$\Delta I[i]^{\{1,1\}} = (\alpha_i C^a[i] - \beta_i C^b[i]) I[i]$$

$$\begin{aligned}\Delta I[i]^{\{0,1\}||\{1,0\}} &= (I[i] - I[i]^L)(C^a[i] - C^b[i]) + \\ &\quad I[i]^L(-C^a[i] + C^b[i]) \\ &= (I[i] - 2I[i]^L)(C^a[i] - C^b[i])\end{aligned}$$

$$\Delta I[i]^{\{0,0\}} \approx 0$$

In the equations, the corresponding challenge bits can be substituted to get the current difference. For example $\Delta I[i] = (\alpha_i - \beta_i) I[i]$ when $C^a[i], C^b[i] = \{1, 1\}$. Given the current difference parameters of the unit stage, it is possible to express the total current difference ($\Delta I = I_a - I_b$) as follows:

$$\Delta I = \sum_{i=1}^{N} \Delta I[i]^{\{C^a[i], C^b[i]\}} \tag{5.4}$$

So, the CRP relationship of the Current-based PUF can be expressed as,

$$\Delta I \lessgtr 0. \tag{5.5}$$

To be more precise, the response computation is given by

$$r = \begin{cases} 1 & \text{if } \Delta I > 0 \\ 0 & else. \end{cases} \tag{5.6}$$

The above described model is used in the ML attacks and the results are demonstrated in the following subsection.

5.2.2 Attack Results

We evaluated the Current-based PUF implemented in 32nm SOI for the various ML attacks described in section 4.3. Different lengths of the PUF circuit (64, 80 and 128 stages) were analyzed in the attacks. The experiments under nominal conditions

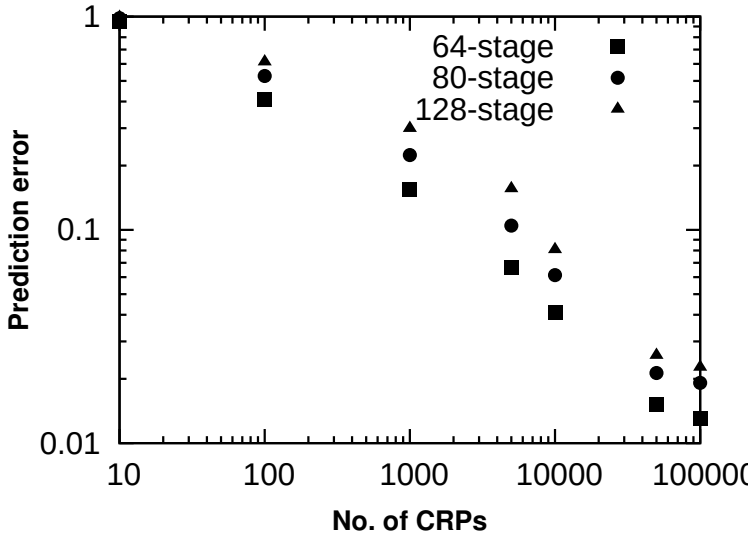


Figure 5.3. Prediction errors from SVM attacks for 64, 80 and 128 stage Current-based PUFs

were repeated 5 times and the stable CRPs across the iterations were separated and used for the attacks.

For SVM based attacks, we trained the classifier with a random set of CRPs collected from the global set of CRPs (3 Million). The challenges were mapped from (0,1) to (-1,1) as in [51] and used as a feature vector in SVM attacks. The prediction errors as a function of the size of training CRPs for different lengths of Current-based PUFs are shown in Fig 5.3. We can observe that SVM based predictor can reach an accuracy of 98% when trained with 20,000 CRPs approximately.

We adopted a similar strategy for the ES attacks by picking a random set of CRPs from the global pool. In the evaluation phase, the random offsprings (count of 6) obtained from each parent were evaluated for their strength based on the number of correct response bits predicted by the model. The best-fit offspring from evaluation phase was then used as a parent for the next generation and the others were discarded. The process was repeated across many generations until the prediction accuracy got saturated. The prediction errors as a function of the number of training CRPs are

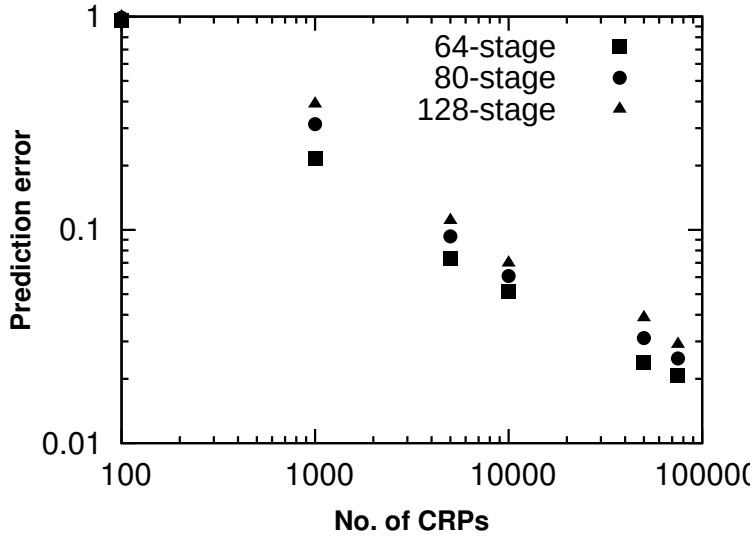


Figure 5.4. Prediction errors from ES attacks for 64, 80 and 128 stage Current-based PUFs

shown in Figure 5.4. The results are shown for a generation count of 100. For some chips, optimal prediction rates (∼99%) were obtained for a lower generation count and the maximum generation count obtained from 20 chips was around 95. So, a generation count of 100 was used for ES attacks. From Figure 5.4, we can observe that ES attacks yield a prediction accuracy of 98% for a CRP size of 70,000 approximately. Similar to SVM attacks, only the stable responses observed over 5 different measurements under the nominal conditions were used for the attack.

The results from SVM and ES based attacks are tabulated and shown in Table 5.1. The different attack parameters and the attack time overheads are also tabulated. As mentioned in the paragraphs above, the attacks employed only stable responses. However, if unreliable responses are used in attacks, the prediction accuracies degrade. The impacts of error-inflicted CRPs on modeling attacks for Current-based PUFs are demonstrated in the next section.

Table 5.1. Standalone ML attack results on current-based PUFs using stable CRPs from 32nm test chip

ML algorithm	# Stages	# CRPs	# Training CRPs	Prediction accuracy (%)
SVM	64	$3*10^6$	100,000	98.8
	80			98.4
	128			98
ES	64	$3*10^6$	80,000	98
	80			97.7
	128			97.3

5.2.3 Impact of Error-inflicted CRPs

The error-inflicted CRPs in the training set degrades the prediction accuracies achieved by modeling attack algorithms. The bit-flips happen when the polarity of the current difference sampled by the sense amplifier is reversed. So, the bit-flips impact the training and prediction accuracy of the modeling algorithm. As explained in the previous chapter, the bit-flips can happen due to internal and external noise sources. The amount of bit-flips from internal and external noise sources in the Current-based PUF circuits are tabulated in Table 5.2. For observing the impact of internal noise, the measurements were conducted at optimal conditions (0.9V and 25° C) for 5 times and the amount of bit-flips for the same challenges were collected. We can observe that around 2% of the 3 Million CRPs are unreliable and have higher chance to get flipped under the same measurement conditions. For observing the impact of external noise, the measurements were conducted at extreme operating conditions (1.1V and 75° C) and the bit-flips were collected. From table 5.2, we can observe that around 7% of the total responses are susceptible to fluctuations in voltage and temperature conditions.

The impact of the error-inflicted CRPs on the prediction accuracies for SVM and ES based attacks are shown in Figures 5.5 and 5.6, respectively. Evolution strategies were able to handle the error-inflicted CRPs better than SVM, because

Table 5.2. Bit flip measurements from 32nm Current-based PUFs. Intrinsic bit flips were observed from repeated measurements under optimal conditions, whereas extrinsic bit flips were observed by changing the operating conditions.

Measurement	**# Measured CRPs**	**# Unreliable responses**	**% Bit-flips**
Intrinsic	$3*10^6$	59,550	1.98
Extrinsic		210,010	7

of their parametric nature as observed from Fig 5.6. The degradation in prediction accuracy for a 6% error-inflicted CRPs is around 7% for ES and around 11% for SVM. The impact of the error-inflicted CRPs on SVM and ES attacks are tabulated in Table 5.3. As the error-inflicted CRPs have a reversed polarity in current difference when compared to the original response, the learning phase of the modeling attack is impacted. However, not all the CRPs suffer from the polarity reversal. Only the challenges for which the current difference is lower than a threshold are susceptible. This threshold current difference can leak some information and can be used along with the training set CRPs to improve the prediction accuracies. The details are described in the next subsection.

Table 5.3. Impact of error-inflicted CRPs on ML prediction rates for current-based PUFs

ML algorithm	**# Stages**	**% Error-inflicted CRPs**			
		0	**2**	**5**	**7**
SVM	64	98.8	95.8	93.3	90.7
	80	98.4	95.3	92.8	90.1
	128	98	94.6	91.9	89.1
ES	64	98	96.3	94.1	92.8
	80	97.7	96	93.8	92.6
	128	98.3	95.6	93.5	92.4

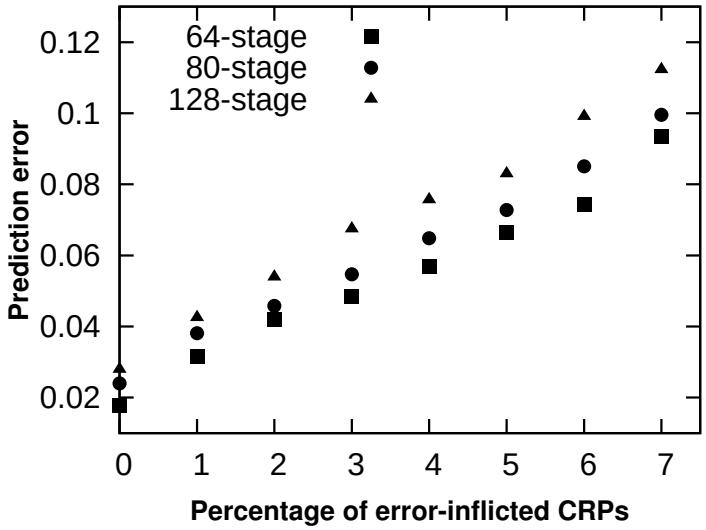


Figure 5.5. Performance of SVM attacks on Current-based PUFs with error-inflicted CRPs

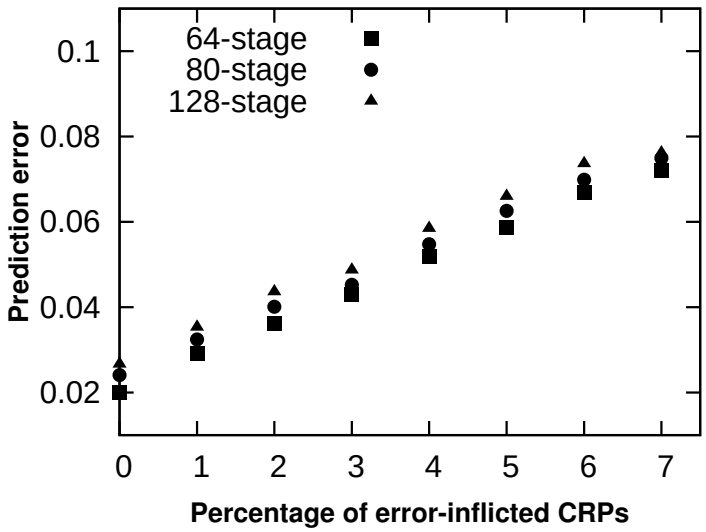


Figure 5.6. Performance of ES attacks on Current-based PUFs with error-inflicted CRPs

5.2.4 Hybrid attacks on Current-based PUFs

Similar to delay-based PUFs, the current-based PUFs were analyzed for their vulnerabilities to fault-assisted ES attacks, also termed as *hybrid attacks*. The Current-

based PUFs when operated at extreme operating conditions (1.1V and 75° C) produce more unreliable CRPs. Although this is detrimental to the PUF's operation, some data dependent information can be extracted. One such information is the amount of current difference that is sensitive to V/T variations.

To estimate the amount of threshold current difference, Monte carlo simulations were performed over the Current-based PUF circuit using the 32nm SOI transistor models. Threshold voltage variations were assigned from a Gaussian distribution with 3σ deviation of 90mV. Around 1 Million CRPs along with the corresponding current difference values were collected from spice simulations. For injecting faults, experiments were conducted by changing the operating conditions (0.8 to 1V in steps of 0.05V and 25 to 75° C in steps of 25° C) and the above mentioned data were collected. The error-prone response bits that flipped under different operating conditions when compared to the nominal conditions (0.9V and 25° C) were mapped to the corresponding current-difference values. The CRPs for which the current-difference was close to 0A were highly vulnerable to changes in operating conditions and the behavior can be clearly observed from Figure 5.7. The plot in Figure 5.7 shows the error-free and error-prone CRPs for both the supply voltage and temperature variations. As similar characteristics were observed for different PUF sizes, the distribution for 128 stage PUF circuit is shown in Figure 5.7. The plot showing the amount of flipped bits whose current-difference is less than 5 nA observed under different conditions is shown in Figure 5.8. The amount of instable CRPs can be increased further by considering the CRPs with current difference higher than 5 nA that may flip under highly extreme conditions ($>$ 1.1V). However, we set the limits to around 10% of nominal V_{dd}, as they correspond to the typical power supply noise level. Moreover, some of the security ICs may be equipped with on-chip voltage detector to avoid over- and under-powering attacks. Hence, the limits of V_{dd} were set to conservative values while mounting fault attacks. Moreover, the unreliable CRPs with current difference

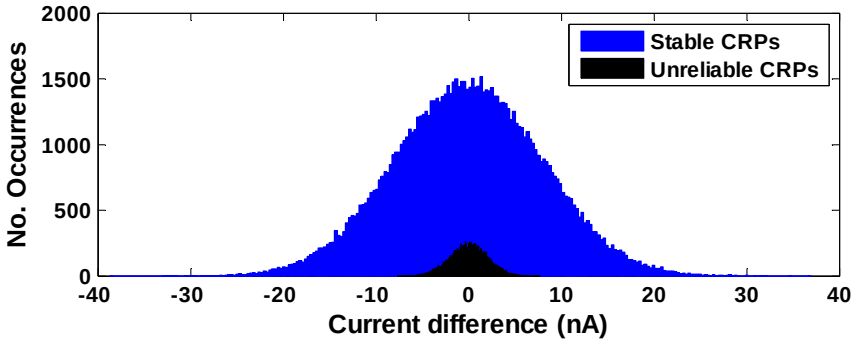


Figure 5.7. Current-difference distributions of error-free and error-prone CRPs

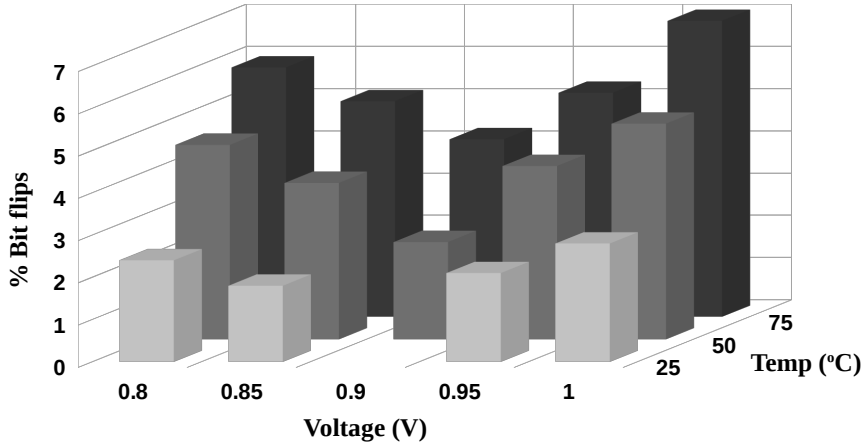


Figure 5.8. Percentage of unstable CRPs from circuit simulations whose current difference is lower than 5 nA

less than 5 nA yielded optimum results with respect to the number of generations required to reach the optimal prediction accuracy. If a response bit flips, then it is most likely to have a current difference less than 5 nA. This current difference is denoted by ΔI_{min}.

5.2.4.1 Performance of Hybrid attacks

The ES algorithm with the same parameters described in section 4.3 was used in hybrid attacks. The current-difference data (ΔI_{min}) was used to filter out the bad PUF models. As described in section 4.6, the PUF models were tested for their fitness

by evaluating the correlation-coefficient between a hypothesis vector and the golden vector obtained from measured CRPs. The hypothesis vector is obtained from the PUF model by computing the current difference for a given challenge (ΔI_H). The hypothesis vector F_H is given by,

$$F_H = \begin{cases} 1, \text{if} |\Delta I_H| < \Delta I_{min} \\ \\ 0, \text{if} |\Delta I_H| > \Delta I_{min} \end{cases} \tag{5.7}$$

The golden vector F is obtained using the same equation for the measured CRPs, i.e if the response bit r_i got flipped then F_i is assigned to 1, else F_i is assigned to 0. By evaluating the correlation-coefficient between F and F_H, the strength of the hypothesis vector is obtained. Higher correlation is observed for a PUF model which is closer to the ideal PUF model (PUF circuit). The correlation-coefficient is lesser than 1, as some of the flipped response bits are induced by instabilities in the sense amplifier rather than the current-difference. In such cases, the hypothesis shown in equation 5.7 is not valid and will impact the prediction accuracy. The correlation coefficient increases with the number of unstable CRPs obtained from the PUF circuit as shown in Figure 5.9. We show the data for 128-stage PUF and similar results were obtained for 64- and 80 stage PUFs as well.

Hybrid attack experiments were conducted on 20 different PUF instances on the test chips and the results were averaged over 100 random sets of training CRPs. In the global CRP training set (3 Million), around 210,000 error-inflicted CRPs were present as demonstrated by Table 5.2. The prediction accuracies obtained from hybrid attacks for the PUF circuits are shown in Figure 5.10. The best-case prediction accuracy obtained from the hybrid attack is close to 99.5%. This represents a significant improvement in prediction rate when compared to 94% prediction accuracy obtained from a standalone ES attack with error-inflicted CRPs. The hybrid attacks achieve significantly higher prediction rates for almost the same number of training CRPs

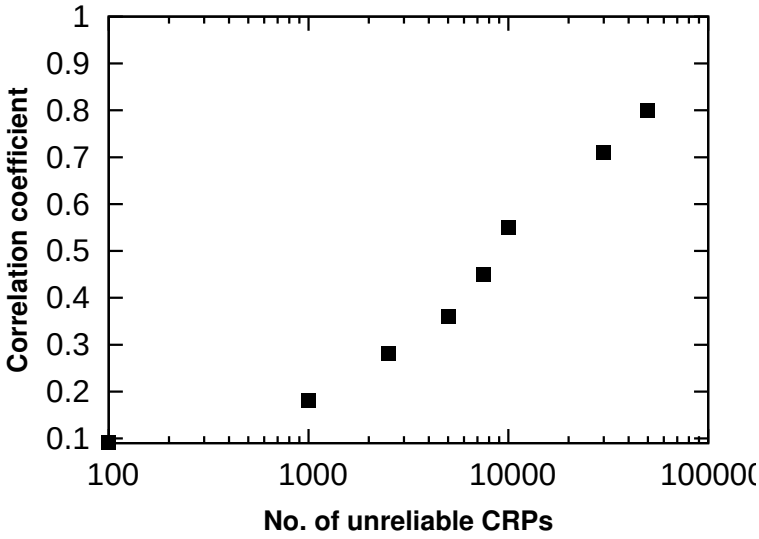


Figure 5.9. Correlation coefficient versus the number of unstable CRPs used in hybrid attacks

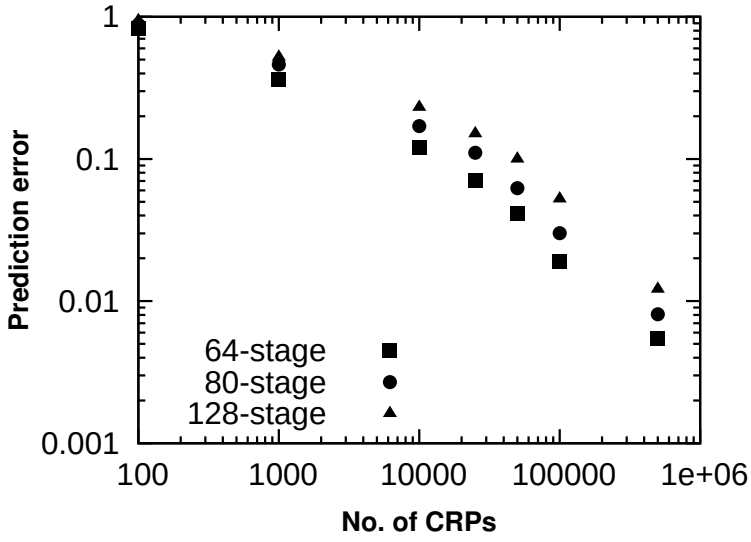


Figure 5.10. Performance of hybrid attacks with 7% error-inflicted CRPs

and a generation count. The averaged results from 20 PUF instances are summarized in Table 5.4.

Table 5.4. Performance of hybrid attacks on current-based PUFs. The results were averaged over 20 different PUF instances on test chip. Around, 7% of unreliable CRPs were present in the training set used in hybrid attacks.

ML algorithm	# Stages	# Training CRPs	# Unreliable CRPs	Prediction rate (%)
Hybrid attacks	64	500,000	34,550	99.5
	80			99.1
	128			98.8

In the above sections, the vulnerabilities of existing Current-based PUFs to modeling and hybrid attacks were clearly demonstrated. In the next section, we present the secure PUF architecture that is tolerant to modeling attacks.

5.3 Non-linear Current Mirror based PUF Architecture

In this section, we describe in detail the proposed secure PUF architecture using non-linear current mirrors (nlcPUF). First, we explain the source of non-linearity in section 5.3.1 followed by the PUF architecture in section 5.3.3. The post-silicon validation of the performance metrics of nlcPUF are presented in section 5.3.5.

5.3.1 Source of Non-linearity in nlcPUF

The proposed PUF uses non-linear current mirrors [91] as building blocks. Current mirrors act as diodes, as the output current(I_{out}) flows only for positive input currents (I_{in}). A current mirror exhibits non-linear transfer characteristics when a constant current source is used along with the current mirror. The threshold input current is decided by the amount of the constant current source. A simple circuit implementation of a non-linear current mirror is shown in Figure 5.11. As described in [91], current mirrors can be constructed in NMOS-only, PMOS-only and NMOS-PMOS combinations. However, NMOS-only current mirror is used in our PUF construction due to its simplicity. The non-linear transfer characteristic of the current mirror shown in Figure 5.11 was evaluated using circuit simulations using the IBM 32nm transistor models by setting the constant current source at $100\mu A$. The trans-

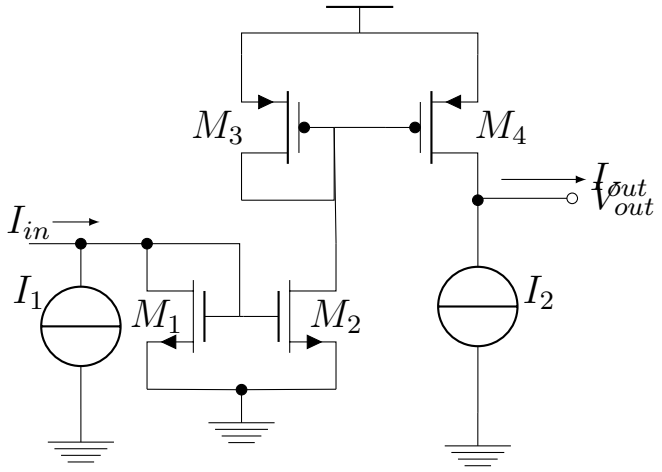


Figure 5.11. Non-linear current mirror [91]

fer characteristic of the non-linear current mirror is shown in Figure 5.12. It can be observed that the current mirror exhibits a breakpoint around $100\mu A$, beyond which a linear behavior in I_{in}-I_{out} characteristic is observed. The slope of the linear region in I_{in}-I_{out} characteristic is determined by the relative strengths of the transistors M_1 and M_2. However, due to mismatches in the transistor sizes arising from process variations, the transfer characteristics deviate from its nominal behavior. This behavior is exploited in the proposed PUF circuit.

5.3.2 Effect of Process Variations

As described in the previous subsection, the presence of device mismatches in the transistors M_1 and M_2 impacts the transfer characteristics of the current mirror. To estimate the impact of process variations, Monte carlo simulations were performed over the current mirror shown in Figure 5.11 using the IBM 32nm SOI transistor models. Threshold voltage variations were assigned from a Gaussian distribution with 3σ $\approx$90mV. The impact of process variations on the transfer characteristics is shown in Figure 5.13. Although constant current sources were assumed in this analysis (I_1=I_2=100μA), there might be some mismatches in a silicon implementation. These

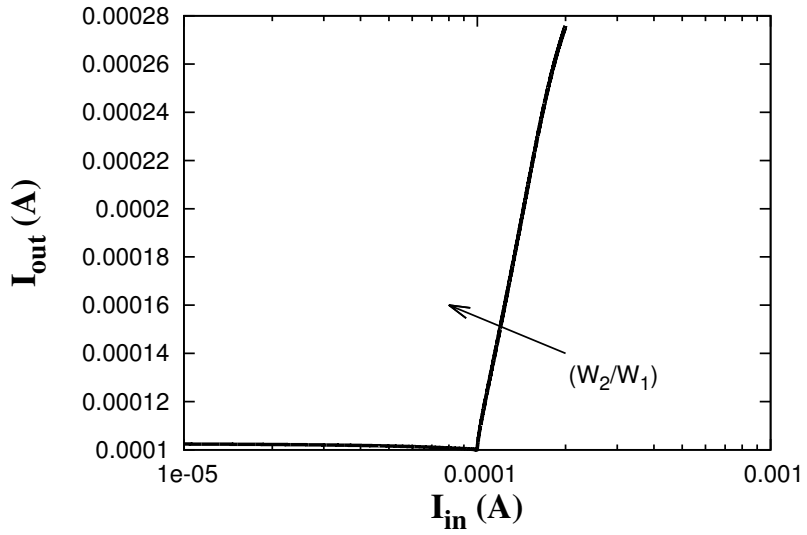


Figure 5.12. Non-linear transfer characteristic of the current mirror

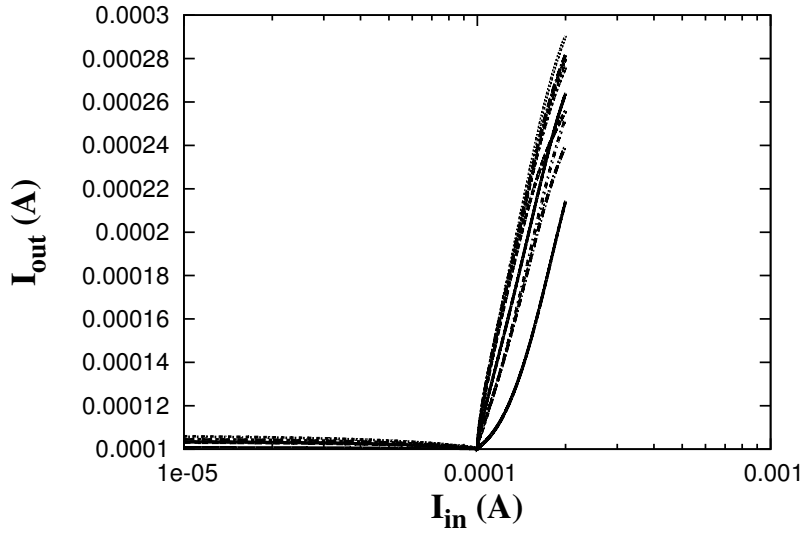


Figure 5.13. Impact of process variations on the transfer characteristic of Non-linear current mirror

uncertainties in transfer characteristics arising from process variations are exploited in nlcPUF construction. The details are described in the next subsection.

5.3.3 nlcPUF Architecture

The proposed nlcPUF is based on current propagation through two identical chains of non-linear current mirrors and is analogous in operation to an arbiter PUF [86]. Figure 5.14 shows the architecture of nlcPUF. The circuit has a multiple bit challenge C and produces a single bit response. The challenge bits determine the current propagation paths through a switch. The switch either *passes* or *switches* the input currents to the outputs, whose implementation is shown in Figure 5.15(a). In our construction, $C_i = 0$ switches the currents and $C_i = 1$ passes the currents directly to outputs. In this way, a unique propagation path is created for each challenge C. For evaluating the response bit, the inputs of the circuit are tied to a common current source whose amplitude is equal to twice the breakpoint of non-linear current mirrors. The input current is split in half and flows through the identical paths. Due to process variations in the current mirrors, the output current of a single stage can fall in one of the two regions of the transfer characteristics shown in Figure 5.13. The current shift ratio of a single stage is given by,

$$\text{Current shift ratio}_i = \frac{\text{Output current}_i}{\text{input current}_i}. \tag{5.8}$$

The output current from a stage is fed as an input to the subsequent stage, which shifts the current based on its current shift ratio. The process is repeated over all the stages and the current difference at the output ($\Delta I = I_a - I_b$ as shown in Figure 5.14) is sampled to get a single bit response.

In an arbiter PUF, the amount of delay introduced by a unit stage is fixed for a given challenge bit, i.e, the delay contribution from stage i is dependent only on the challenge bit C_i. However, in nlcPUF, the amount of current shift introduced by a single stage is dependent on the input current itself. So, the current shift ratio introduced by stage i is dependent on the challenge bit C_i, as well as the challenge bits $C_1 ... C_{i-1}$. This property enhances the unpredictability and modeling attack

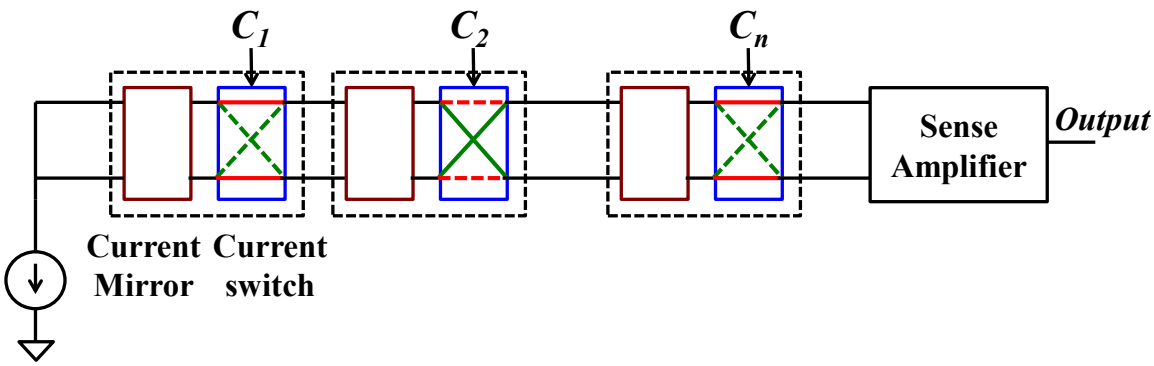


Figure 5.14. Proposed PUF Architecture. I_a and I_b are the output currents that are compared to generate the response bit.

resistance of the PUF circuit. The currents at the output I_a and I_b are compared to generate a single bit response. For comparing the currents, a latch-based sense amplifier shown in Figure 5.15(b) is used that generates the output bit based on the relative strengths of the currents. Before response generation, the *trigger* signal in the sense amplifier is pulled low which pre-charges the *out* and out_b (not shown in figure) nodes to V_{dd}. Once the *trigger* signal goes high, the challenge bits are applied. Based on the strengths of the flowing currents, one of the output nodes discharges quickly than the other node. This results in a positive feedback which settles the output nodes. The process is repeated again for generating further response bits.

As it is highly difficult to express the transfer characteristic of the entire PUF circuit in a closed from, the non-linearity injected by a single stage is shown in terms of the current shift. The current shift introduced by stages {20,40,60,80} in an 80 stage nlcPUF circuit is shown in Figure 5.16. The current shift distribution was obtained by simulating the 80 stage nlcPUF circuit using the IBM 32nm transistor models over 20,000 CRPs. The impact of process variations can be observed from the uniformly distributed current shift values. The varying means of the current shift distributions for the different stages show the presence of non-linearity (i.e. stage 80 has a lower mean than stage 40). Similar non-linear effects were observed for other stages as well.

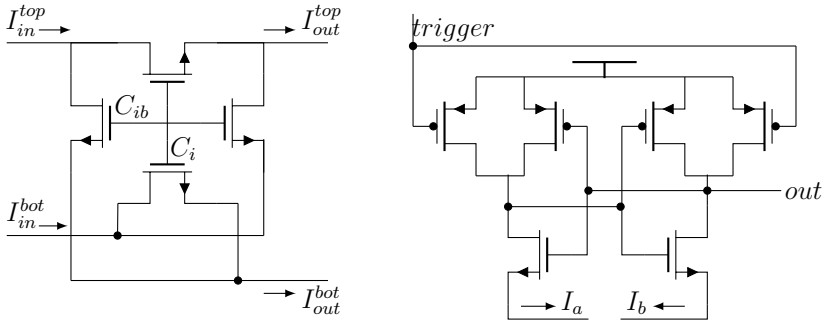


Figure 5.15. (a) Current switch, (b) Sense amplifier. The input currents to the current switch are I_{in}^{top}, I_{in}^{bot} and the output currents are I_{out}^{top} and I_{out}^{bot}. C_{ib} is the inverted challenge bit ($C_{ib} = \sim C_i$).

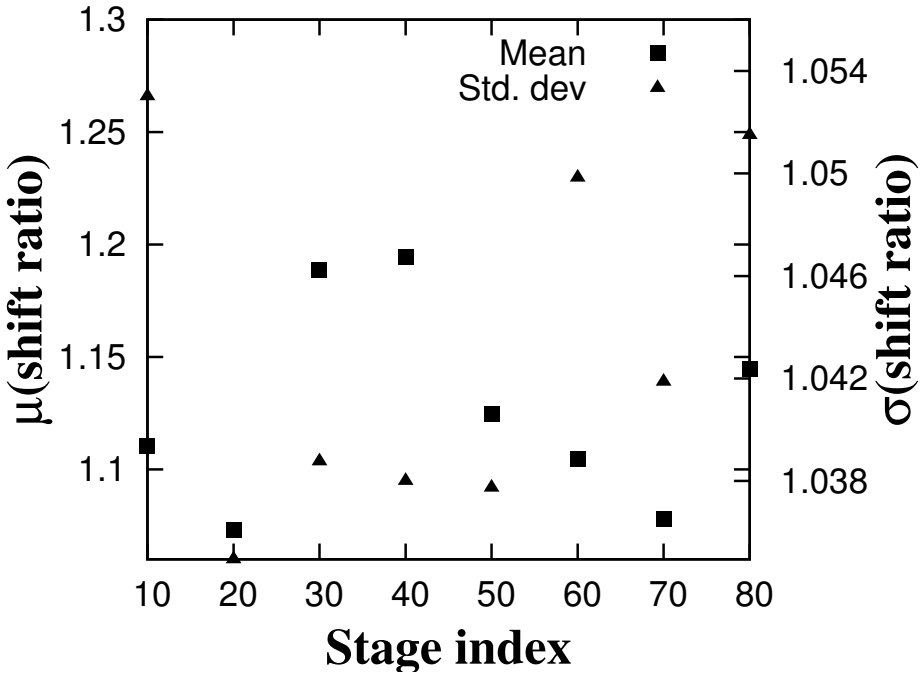


Figure 5.16. Mean and standard deviation of current shift ratios in an 80-stage nlcPUF circuit

Please note that the delay introduced by a single stage in an arbiter PUF will be a fixed value irrespective of the challenge bit, unlike the uniformly distributed current shift introduced by a single stage in nlcPUF. The impact of non-linearity introduced by individual stages on the unpredictability of responses is presented in section 5.3.6.

5.3.4 Implementation Details

The nlcPUF circuits of different lengths were designed as an array containing 32 instances. Every unit stage of the nlcPUF has around 22 transistors. The entire array containing 64-stage nlcPUFs was laid out in an area of $0.025mm^2$, with each PUF instance occupying an area of $\sim 790\mu m^2$. The areas of the nlcPUF implementations are shown in Table 5.5.

Table 5.5. Area details of a single instance of various nlcPUF circuits

# Stages	Area (μm^2)
64	790
80	986
128	1579

5.3.5 Post-silicon Validation of nlcPUF

The proposed nlcPUF circuit was fabricated using the IBM 32nm SOI process as shown in chapter 7. Around 200 instances of the nlcPUF circuit (10 instance x 20 dies) were evaluated across 3 Million CRPs. The validation was performed under nominal conditions (0.9V and 25°C) and extreme conditions (V_{dd}±0.1V and 75°C) for 5 times and the results were averaged. For evaluating the performance metrics, the framework described in chapter 2 was used. The various performance metrics analyzed include uniqueness, reliability, uniformity and bit aliasing probability. The results of the post-silicon validation are shown in Figure 5.17 and summarized in Table 5.6. It can be observed that the nlcPUF architecture exhibits excellent statistical properties upon post-silicon validation. The analysis of security vulnerabilities of nlcPUF architecture is presented in the next subsection.

5.3.6 Security Evaluation of nlcPUF architecture

In this section, we present the security evaluation results of nlcPUF architecture presented in the previous section.

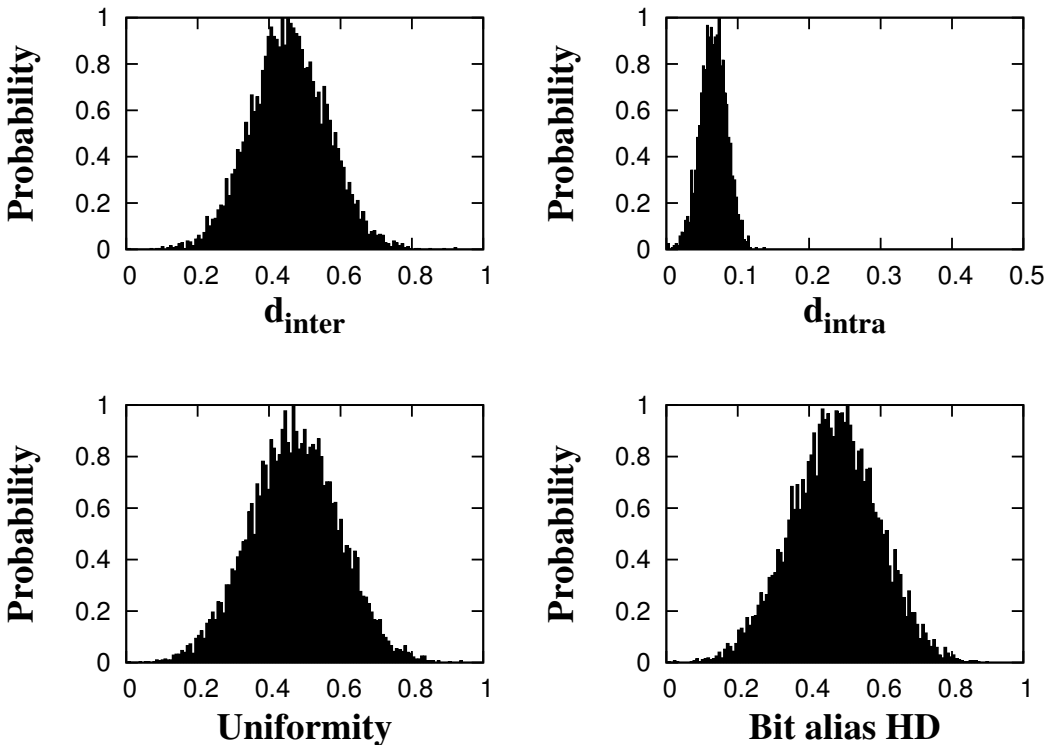


Figure 5.17. PUF Performance metrics distributions. (a) Inter-class HD (b) Intra-class HD (c) Uniformity and (d) Bit-aliasing probability

5.3.6.1 Modeling attacks validation of nlcPUF

The fabricated nlcPUFs were tested for vulnerabilities using the different modeling attacks presented in section 4.3. Like in Current-based PUFs, around 3 Million CRPs were collected through post-silicon measurements under nominal and extreme operating conditions. For simple modeling attacks using SVM and ES, the stable CRPs collected under nominal operating contions were used in modeling attacks. For this purpose, the experiments under nominal conditions were repeated 5 times and the stable CRPs across all the iterations were separated.

Similar to attacks on Current-based PUFs, the SVM and ES classifiers were trained using a random set of CRPs from the global pool. The challenges were mapped from (0,1) to (-1,1) for modeling purposes. The feature vector for the attacks were obtained using the equation:

Table 5.6. Performance validation of nlcPUF and comparison to other strong PUF architectures

Type of PUF	Tech node	# Stages	# PUF instances	dinter	dintra (intrinsic/extrinsic)	Uniformity	Bit-alias HD
Arbiter [54]	45 nm	64	80	0.38	-/0.08	-	-
		80	-	-	-	-	-
		128	-	-	-	-	-
Current based	32 nm	64		0.39	0.08	0.42	0.43
		80		0.4	0.075	0.415	0.415
		128		0.39	0.072	0.42	0.42
nlcPUF	32 nm	64		0.41	0.02/0.062	0.41	0.43
		80		0.405	0.019/0.065	0.42	0.42
		128		0.42	0.017/0.068	0.42	0.415

$$p_i = \prod_{j=i+1}^{n} C_j \tag{5.9}$$

where $(p_0, p_1..p_n)$ represents the parity vector with $p_n = 1$ and C_j refers to the j^{th} challenge bit. The prediction errors as a function of the training CRP size for nlcPUFs against SVM and ES attacks are shown in Figures 5.18 and 5.19 respectively. It can be observed that nlcPUF has almost 20% higher modeling attack resistance when compared to Current-based PUFs. For a training set size of 200,000 nlcPUF exhibits around 80% learnability, whereas the Current-based PUF exhibits 98% learnability.

The error-inflicted CRPs degrade the prediction accuracies obtained for nlcPUF circuits as in Current-based PUFs. The impacts of intrinsic and extrinsic noise on nlcPUF circuits can be found in Table 5.6. From intrinsic noise measurements, i.e. measurements repeated under nominal operating conditions, around 2.1% bit-flips were observed. On the other hand, around 6.5% bit-flips were observed under extreme operating conditions. The ML attacks were repeated with a new random set of data that include error-inflicted CRPs along with the stable CRPs. The impact of error-inflicted CRPs on the prediction accuracies of SVM and ES attacks on nlcPUF circuits are shown in Figures 5.20 and 5.21. It can be observed that the prediction accuracies degrade by 25% for 6% error-inflicted CRPs.

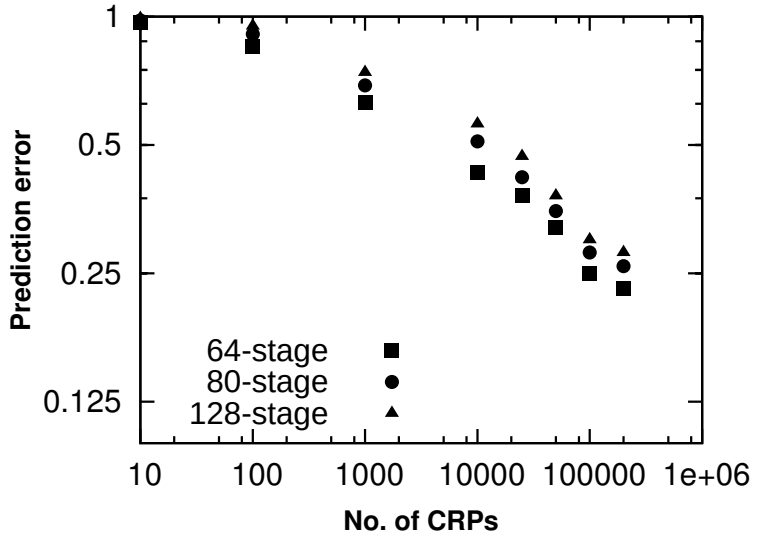


Figure 5.18. Prediction errors from SVM attacks for 64, 80 and 128 stage nlcPUF

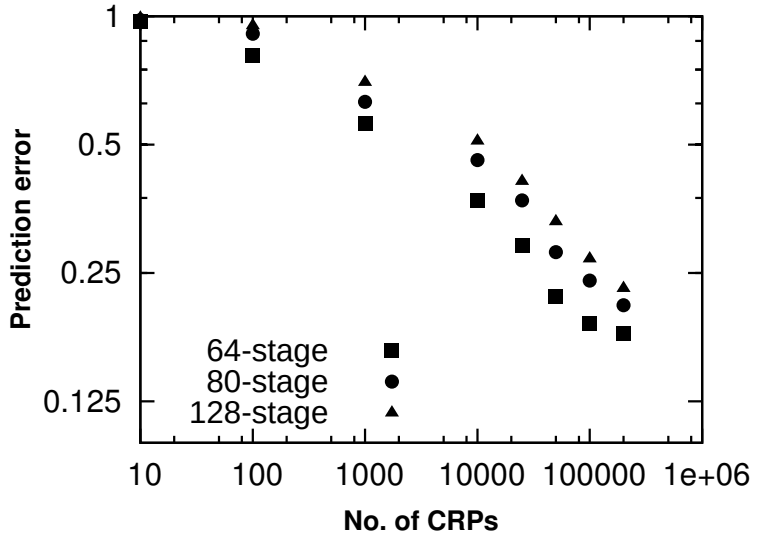


Figure 5.19. Prediction errors from ES attacks for 64, 80 and 128 stage nlcPUF

5.3.7 Hybrid Attacks on nlcPUF

The nlcPUF circuits were also evaluated for vulnerabilities using the hybrid attack methodology described in section 4.6. As the hybrid attacks require a vulnerable threshold value, the current-difference ΔI_{min} was evaluated using Spice simulations.

From 32nm statistical circuit simulations on 128 stage nlcPUF, ΔI_{min} was found to be around 8nA. The value was more or less the same for 64, and 80 stage PUF circuits as well. The threshold value was used to construct the hypothesis vector F_H as described

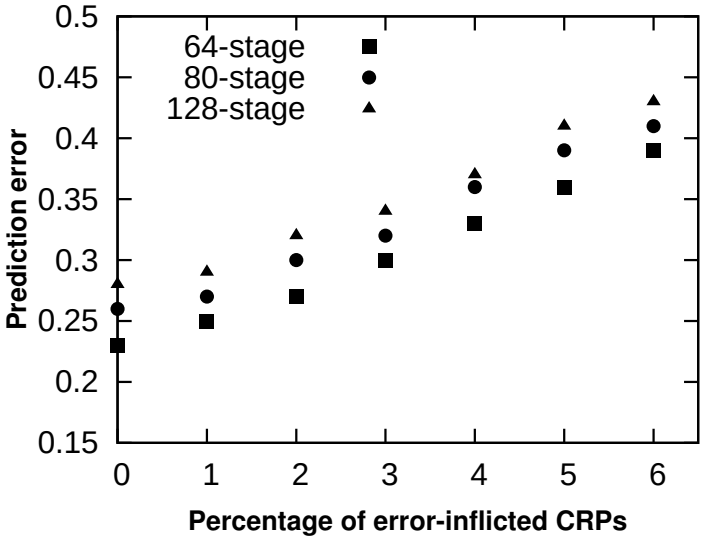


Figure 5.20. Impact of error-inflicted CRPs on SVM prediction rates for 64, 80 and 128 stage nlcPUF

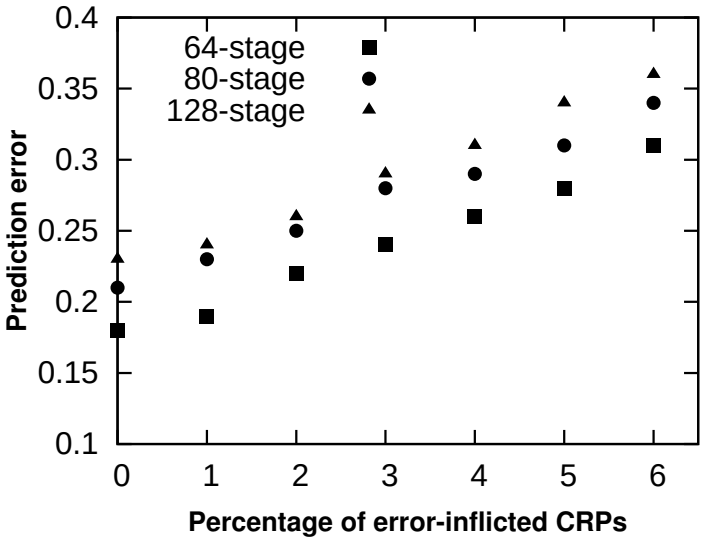


Figure 5.21. Impact of error-inflicted CRPs on ES prediction rates for 64, 80 and 128 stage nlcPUF

Table 5.7. Security Validation of nlcPUF and comparison to other strong PUFs

Type of PUF	Tech. node	Type of Attack	# Stages	# Training CRPs	Prediction accuracy(%)	Training time
Arbiter [78]	45nm	Logistic Regression	64	18,050	99.9	0.6s
Current based	32nm	ES with stable CRPs	64	75K	98	12:40min
			80		97.7	16:20min
			128		97.3	21:10min
		Hybrid	64	500K	99.5	74:10min
			80		99.1	83:12min
			128		98.8	112:40min
nlcPUF	32nm	ES with stable CRPs	64	200K	82	22:20min
			80		79.2	29:10min
			128		77.5	43:20min
		Hybrid	64	500K	86.1	78:20min
			80		82.4	88:10min
			128		80.1	119:10min

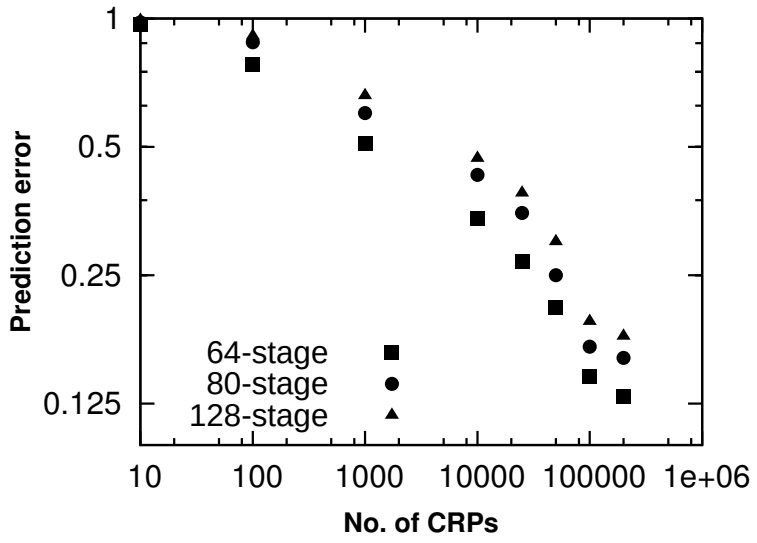


Figure 5.22. Prediction errors from hybrid attacks for 64, 80 and 128 stage Current-based PUFs

in section 4.6. Experiments were conducted on 20 different PUF instances for 100 random sets of CRPs and the results were averaged. The performance of hybrid attacks on nlcPUF circuits are shown in Figure 5.22. The nlcPUF circuit exhibits around 15x higher resistance over Current-based PUFs. As ES achieves only moderate prediction accuracies even under the presence of highly stable CRPs (∼80%), only

a marginal improvement in prediction accuracy was observed from hybrid attack framework which builds upon the ES attack.

Table 5.7 summarizes the attack results on nlcPUFs. The comparison results with Arbiter and Current-based PUFs are also shown in Table 5.7. In general, nlcPUF architecture exhibits excellent security properties measured in terms of information leakage. Please note that the nlcPUF architecture can also be modified to incorporate Feed-forward [51] and XOR operations, which can further improve the security properties.

CHAPTER 6

DESIGN STRATEGIES FOR PUF CIRCUITS AND SYSTEMS

In the previous chapter, the design and post-silicon validation results of a modeling attack resistant PUF were presented. The techniques presented were used to improve the unpredictability of PUF responses. Apart from unpredictability, uniqueness and reliability are also some of the significant metrics to be considered in commercial applications involving PUF circuits. In this chapter, techniques to improve the uniqueness of PUF based circuits are presented. In order to improve the uniqueness, fabrication/lithography aware techniques are presented[1]. Delay- and current-based circuits are used as the PUF targets to demonstrate the impacts of lithography aware design techniques. Finally, a PUF based authentication system is also presented[2].

6.1 Lithography Aware Design of Physically Unclonable Functions

In sub-wavelength lithography, the polygons/structures in the mask are printed onto a wafer using an imaging system [57]. Process variations are omnipresent during the fabrication process. These variations can be classified into "systematic" and "random" variations. It is often desirable for a PUF circuit to have random variations dominant over systematic variations. In order to make a PUF circuit more "unique", systematic variations should be suppressed.

[1]This work was published in [42, 46]

[2]This work was published in [41]

In this section, we present a generalized lithographic simulation framework adopted for improving PUF design. The main objective of the lithographic simulation framework is to enhance the sensitivity of the PUF circuit to process variations and improve uniqueness when viewed across dies and wafers. It is well known that forbidden pitches are more prominent in sub-wavelength lithography [50]. Forbidden pitches are often undesirable, as the polygons/structures at these pitches will not be printed to their maximum resolution. However, the sensitivity of critical dimension (CD) is very high to the pitch variations near the forbidden zone. This is used constructively to enhance the impact of process variations on PUF design. This is done by placing gate structures of the transistors at pitches closer to forbidden zone. Such a technique allows the circuit designers to amplify and effectively utilize various sources of lithographic variations including dose, resist thickness, lens imperfections, defocus, etc. in PUF designs. Arbiter- and Current-based PUFs were chosen as the PUF targets to demonstrate the impacts of lithography aware design technique. The PUFs were also designed using a conventional approach and compared with lithography aware designs.

We will address variations due to dose and defocus in this work. Dose variations arise from fluctuations in the intensity and the duration of light source. Focus variations or defocus arise due to changes in the relative distance between the lens and the resist. This can be due to misalignment (change in focus, wafer tilt, etc.) or changes in resist thickness due to CMP. In this paper, the former is referred to as defocus and the latter as resist variations. So, we will study the effect of dose, defocus and resist variations on the structures and hence the circuit parameters across the dice and wafers. Though all the structures suffer variations, gate or polysilicon structures suffer the most as they represent the critical dimension of a technology node. Furthermore, the variations in gate structures translate to larger variations in electrical parameters more than other structures. The focus of this work is to do physical de-

sign of a PUF circuit in such a way that the impact of manufacturing variations on the circuit is enhanced.

6.1.1 Related work

In [84], novel PUF circuits known as litho-PUFs were proposed that consider proximity effects, density effects and formation of non-rectangular gates printed during the lithography process. Though the proposed scheme in [84] also uses forbidden pitches to improve the uniqueness of PUFs, the evidence of improving inter-wafer uniqueness was not presented. Also, the impacts of different lithographic variations on polygons placed near the forbidden pitch were not evaluated in [84]. Traditionally, Optical Proximity Correction (OPC) tries to reduce variations, both systematic & random by using the geometric error between the simulated contour and the target as a cost function. In [24], a PUF-aware OPC scheme was described, that tries to reduce the systematic variations and increase random variations in the regions of the mask that contain the PUF circuit. The scheme continues to work as a traditional OPC in non-PUF regions of the mask. The proposed scheme in [24] tries to maximize the variance of the mean edge-placement error (EPE). An improvement in uniqueness of 5% and reliability of 70% compared to conventional OPC was reported in [24]. Similar work on enhancing random variations and suppressing systematic variations was presented in [23]. In order to suppress systematic variations, several layout techniques have been used. However, all the existing works in the literature focus only on improving inter-die variations. Wafer-to-wafer variations are known to be more correlated [38]. This affects the uniqueness of PUFs and has not been extensively explored in the literature.

6.1.2 Exploiting Forbidden Pitches for Improving Uniqueness

In this section, we describe the proposed scheme on improving the uniqueness of delay-based PUFs by amplifying the effect of inherent manufacturing variations

during the fabrication process. We use the arbiter and current-based PUFs shown in Figures 4.1 and 5.1 respectively for validating the proposed scheme.

In sub-wavelength lithography, the wavelength of the light source used is larger than the widths of printed structures (lines) leading to variations in printed structures. Further, the destructive interference of light waves from slits near closely placed lines leads to very small line widths, even zero line widths [84]. So, the spacing between lines and hence the pitch has an impact on the printed line width. The pitches at which the printed line width goes to zero are the so-called "*forbidden pitches*" [50]. The sensitivity of the critical dimension to the pitch variations is very high for the pitches closer to the forbidden pitch zone.

The occurrence of forbidden pitches in 45nm and 32nm nodes is shown in Figure 6.1. The forbidden pitches have been determined for the polysilicon structures, as they represent the critical dimension of a technology node. In order to compute forbidden pitches, the experimental methodology discussed in section 6.1.3.1 is adopted. The forbidden pitch is found to be 190nm (155nm) in 45nm (32nm) node. The change in printed line width (ΔCD) when a change in line spacing (Δs) occurs as a result of lithographic variations is given by,

$$\Delta CD = \frac{dCD}{ds}\Delta s \tag{6.1}$$

Where, $\frac{dCD}{ds}$ is the sensitivity of line width (CD) to line spacing (s). It can be clearly seen that the critical dimension is highly sensitive to pitches around the forbidden pitch zone from Figure 6.1 and even a small distortion due to lithography variations can lead to higher variations in critical dimension.

Given the above discussions, we propose our scheme of enhancing the manufacturing variations by placing the gate structures at pitches that are "sensitive". In this scheme, the transistors are broken to multiple fingers such that the pitch between adjacent fingers is a "sensitive pitch" near the forbidden pitch zone. This helps

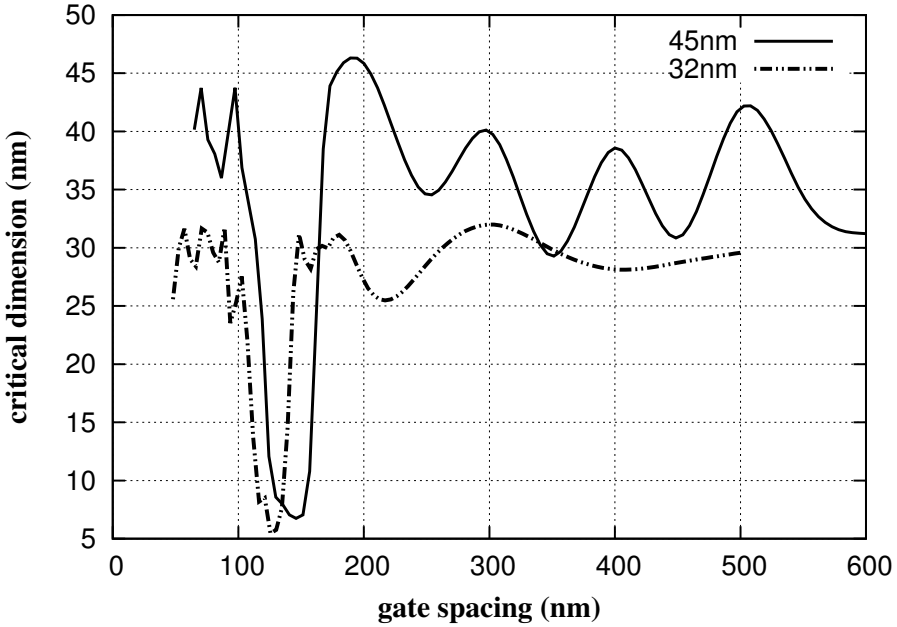


Figure 6.1. Forbidden pitches in 45nm and 32nm nodes

in improving the extent of inter-die and inter-wafer variations by leveraging various sources of lithographic variations such as dose, defocus and resist thickness. This leads to an increased variation in electrical parameters and further translates to an improvement in uniqueness of a PUF as shown in the rest of this document. It is essential to note that the proposed scheme helps to improve the uniqueness of PUFs and does not contribute much towards security/unpredictability of PUF responses.

Security of a PUF circuit arises from the PUF construction itself and not from the way it is designed. To improve security, the PUF construction has to be modified such that the challenge-response behavior cannot be learned using modeling attacks [78]. Some of the examples for improving the modeling attack resistance include feed-forward arbiter PUFs, XOR arbiter PUFs, etc. The techniques to improve the security of a PUF are not an aspect of this work and we focus only on improving the uniqueness of PUFs. However, our scheme can be used to improve the uniqueness of the modified PUF constructions as well.

6.1.2.1 Outline of the Proposed Scheme

Upon performing lithographic simulations on the fingered gate structures in the presence of variations such as dose, defocus, resist thickness, etc., the changes in the line width are computed and fitted within a distribution. Now this serves as a Process Variation (PV) model for the channel length of the transistors in the circuit simulations. So, the various steps in our scheme are:

- Enhance manufacturing variations by exploiting the high sensitivity of critical dimension to pitches near forbidden zone.
- Map the lithographic variations to circuit parameters (PV model) and use in circuit simulations.
- Validate the proposed scheme for inter-die and inter-wafer uniqueness.

6.1.3 Lithographic Simulation Results

In this section, we describe the lithographic simulations performed to obtain the process variation model. We also describe in detail the PUF validation techniques and results.

6.1.3.1 Manufacturing Aware Physical Design Framework

Lithographic simulations were performed by changing the pattern density to compute the changes in the critical dimension. Both 45nm and 32nm nodes were considered in this work. In order to obtain the image of the printed contour, a commercial simulator (Calibre©Workbench^{TM}) was used. A dipole light source (193 nm) having a radius of $0.35\mu m$ and *sigma-center* of $0.1\mu m$ was used for all the experiments as shown in Figure 6.2. The numerical aperture (NA) of the imaging system is 1.35 (1.56) for 45nm (32 nm) node [75]. The critical dimension was obtained from the coordinates of the printed contour using gate-slicing approach [83].

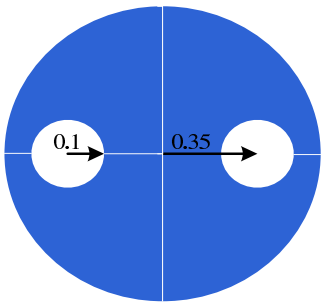


Figure 6.2. Dipole light source

The overall experimental methodology to compute uniqueness is shown in Figure 6.3 and is explained below. For all the experiments, the models from the IBM 32nm SOI library were used.

1. **Manufacturing Process (MP) models**: These describe the extent of lithographic variations such as dose, defocus and resist thickness in the manufacturing process. The MP models have different extents of variations at fine granularities.

2. **Litho Simulation**: The MP models were fed as an input to the litho-simulator along with the layout. The lithographic simulations produce print-image contours which take into account the impact of neighboring cells (proximity effects) along with the inherent variations in the cell itself.

3. **Process Variation (PV) models**: CD values were extracted from the print-image contours after an extensive set of lithographic simulations using the fine-grained MP models. The CD values were then fitted within a Gaussian distribution to form a PV model. The mean (μ_{CD}) and standard deviation (σ_{CD}) of the CD distribution are used to describe a PV model. We obtain separate inter-die and inter-wafer PV models as the corresponding MP models are different.

4. **SPICE simulation**: The obtained PV models, along with a netlist of a PUF circuit, were then simulated using a circuit simulator (HSPICE). An extensive

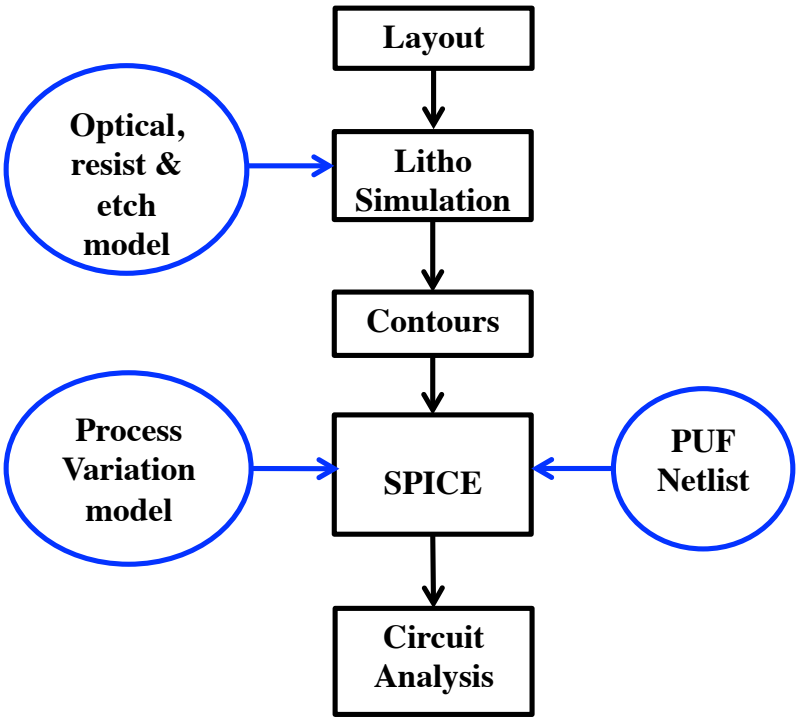


Figure 6.3. Simulation methodology to compute uniqueness

set of challenges were simulated and responses were collected from different PUF instances.

5. **Performance metrics computation**: The performance metrics of the litho-aware and conventional PUF designs were computed as per the framework described in chapter 2.

As explained earlier, the transistors in the PUF circuit were fractured into fingers and the spacing between fingers is chosen near the forbidden pitch zone. Furthermore, the pitch between the fingers in different stages was slightly varied to improve the unpredictability of responses. However, the transistors in a particular stage of a PUF circuit were replicas (designed with the same set of pitches) such that no bias is introduced into the circuit. Now the PV models are generated as follows.

6.1.3.2 Intra-die PV model

Lens aberrations are an important source of intra-die variations and are modeled using Zernike's coefficients [22]. This forms the intra-die MP model. The intra-die PV model, obtained by litho-simulation (using the intra-die MP model) and extraction of the statistics of CD distribution, is shown in Table 6.1.

Table 6.1. Intra-die PV model

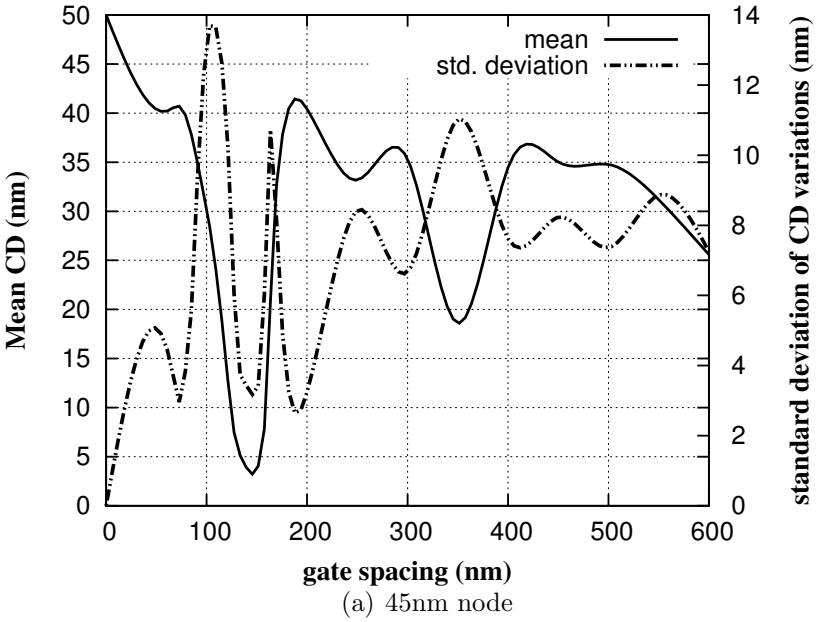
	32nm	45nm
Mean (μ_{CD}^{intra})	34.2nm	49.34nm
Std. Deviation (σ_{CD}^{intra})	1.35nm	2.4nm

6.1.3.3 Inter-die PV model

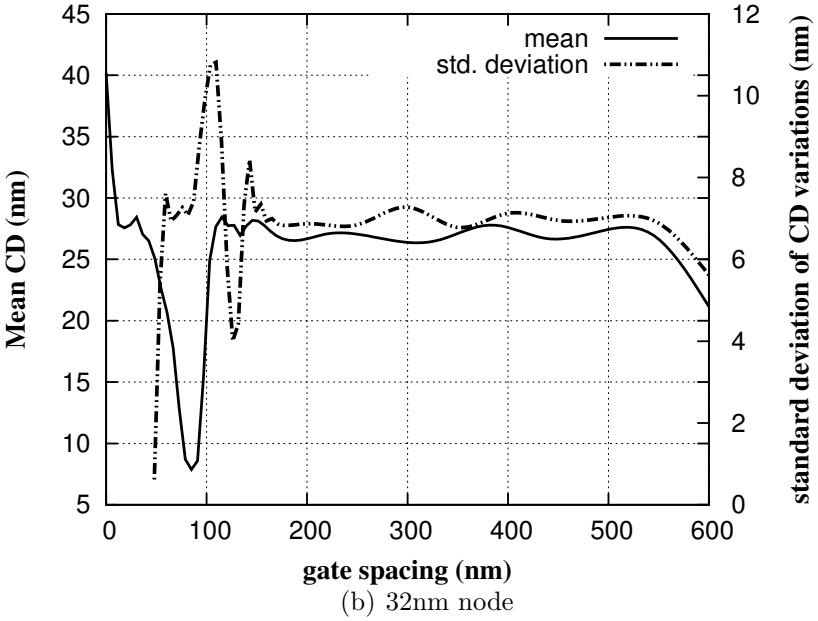
Table 6.2 gives the bounds for the Inter-die MP models. Litho-simulations were performed at various values of dose, defocus and resist thickness within the bounds specified in Table 6.2. After litho-simulations, extraction of statistics of CD distribution yields the inter-die PV model. However, we need to determine a "sensitive pitch" before the design process. Figure 6.4 shows the mean and standard deviation of CD for both 45 nm and 32 nm nodes at different gate spacing values. From Figure 6.4 , we can observe that CD is very sensitive to variations around a gate spacing value of 110 nm (95 nm) for the 45 nm (32 nm) node. Choosing these values as the gate spacing between transistor fingers, litho-simulations were performed using the inter-die MP model. Statistics of CD distribution are shown in Table 6.3 and these form the inter-die PV model.

Table 6.2. Inter-die MP model

Dose	$\pm$ 5%
Defocus	$\pm$ 5nm
Resist thickness	$\pm$ 5%



(a) 45nm node



(b) 32nm node

Figure 6.4. Sensitivity of CD to gate spacing

6.1.3.4 Inter-wafer PV model

The inter-wafer MP model has the following bounds: dose (±2%), defocus (±2%) and resist thickness ($\pm 2nm$). In addition, wafer-tilt of ($\pm 5nm$) was also considered. These values were assigned based on an observation that the amount of inter-wafer

Table 6.3. Inter-die and Inter-wafer PV models

	32nm	45nm
Mean (μ_{CD}^{die})	30.2nm	44.5nm
Std. Deviation (σ_{CD}^{die})	8.4nm	10.1nm
Mean (μ_{CD}^{wafer})	31.2nm	44.8nm
Std. Deviation (σ_{CD}^{wafer})	6.4nm	8.5nm

variations is typically one-third of inter-die variations [38]. The CD distribution at the sensitive gate spacing of 110 nm (95 nm) is obtained for 45 nm (32 nm) node through litho-simulations and the statistics are shown in Table 6.3. This constitutes the inter-wafer PV model.

6.1.3.5 Performance metrics computation

The models obtained from lithographic simulations (die- and wafer models) were used along with the PUF netlists and statistical circuit simulations were performed. For all the experiments, 128 stage PUF circuits were used. The parameters obtained from lithographic simulations (critical dimension variations) correspond to the transistor length variations. Apart from gate-length variations, threshold voltage variations were also assigned from a Gaussian distribution with 3σ deviations of 150mV and 90mV for 45nm and 32nm technology nodes, respectively. As the main objective of the proposed framework is to improve the uniqueness of PUF circuits, the inter-class HD was analyzed through circuit simulations and post-silicon measurements from *sugarloaf*. Around 100,000 CRPs were collected from 200 different PUF instances for inter-class HD computation. The results from circuit simulations and post-silicon measurements are shown in Table 6.4 and 6.5. It can be observed that the litho-aware design techniques have improved the inter-die and inter-wafer distances by almost 6% and 16% respectively for 45nm node. For 32nm, the improvements in inter-die and inter-wafer uniqueness are around 6% and 15% respectively. Post-silicon measurements also agree well with simulation results. For post-silicon measurements,

only inter-die distances are shown, as the fabricated dies were from the same wafer. Similar results were obtained for the current-based PUF circuits as well. Thus, by careful physical design, more unique signatures can be generated without making any additional changes to the PUF layout.

Table 6.4. Uniqueness validation results for litho-aware and conventional arbiter PUFs

Type of experiment	**Type of variations**	**Technology node**	**Type of PUF**	**Inter-class HD**
Statistical circuit simulations	inter-die	32nm	Reference PUF	0.46
			Litho-aware design	0.485
		45nm	Reference PUF	0.46
			Litho-aware design	0.49
	inter-wafer	32nm	Reference PUF	0.31
			Litho-aware design	0.37
		45nm	Reference PUF	0.33
			Litho-aware design	0.39
Post-silicon measurements	inter-die	32nm	Reference PUF	0.38
			Litho-aware design	0.43
	inter-wafer	-		

Apart from uniqueness analysis, the impacts of litho-aware techniques on the other performance metrics such as reliability, uniformity and bit-aliasing were observed. The results are summarized in Tables 6.6 and 6.7. The reliability of the litho-aware circuits were almost identical to the conventional designs. Only marginal improvements were observed. The marginal improvements can be attributed to the increased transistor sizes due to fracturing, which slightly minimizes the impacts of noise and environmental condition fluctuations. In case of uniformity, the litho-aware design fared better than conventional designs by as much as 7%. In case of bit-aliasing probability, improvements of around 8% were observed. These results indicate that litho-aware design helps to extract maximum process variations for better PUF designs without adding any additional structures to the layout.

Table 6.5. Uniqueness validation results for litho-aware and conventional current-based PUFs

Type of experiment	Type of variations	Technology node	Type of PUF	Inter-class HD
Statistical circuit simulations	inter-die	32nm	Reference PUF	0.45
			Litho-aware design	0.48
		45nm	Reference PUF	0.46
			Litho-aware design	0.485
	inter-wafer	32nm	Reference PUF	0.32
			Litho-aware design	0.37
		45nm	Reference PUF	0.34
			Litho-aware design	0.39
Post-silicon measurements	inter-die	32nm	Reference PUF	0.39
			Litho-aware design	0.445
	inter-wafer	-		

As explained in the earlier sections, security of a PUF circuit is determined by the architecture rather than design strategies. In ideal scenario, no differences in modeling attack resistance will be observed for litho-aware designs. Modeling attacks using support vector machines were performed over the 128-stage arbiter and current-based PUF circuits using the methodology described in earlier chapters. The prediction errors for arbiter and current-based PUF circuits are shown in Figures 6.5 and 6.6 respectively. It can be observed that the litho-aware design technique does not improve the modeling attack resistance of PUF circuits. The litho-aware PUFs

Table 6.6. Impact of litho-aware design on other performance metrics for arbiter PUFs

Type of experiment	Technology node	Type of PUF	Intra-class HD	Uniformity	Bit-aliasing HD
Statistical circuit simulations	32nm	Reference PUF	0.05	0.48	0.47
		Litho-aware design	0.048	0.49	0.485
	45nm	Reference PUF	0.055	0.47	0.46
		Litho-aware design	0.052	0.49	0.49
Post-silicon measurements	32nm	Reference PUF	0.059	0.43	0.42
		Litho-aware design	0.055	0.46	0.455
	45nm	-			

Table 6.7. Impact of litho-aware design on other performance metrics for current-based PUFs

Type of experiment	Technology node	Type of PUF	Intra-class HD	Uniformity	Bit-aliasing HD
Statistical circuit simulations	32nm	Reference PUF	0.06	0.47	0.46
		Litho-aware design	0.057	0.485	0.48
	45nm	Reference PUF	0.063	0.48	0.45
		Litho-aware design	0.06	0.49	0.48
Post-silicon measurements	32nm	Reference PUF	0.072	0.42	0.42
		Litho-aware design	0.068	0.46	0.45
	45nm	-			

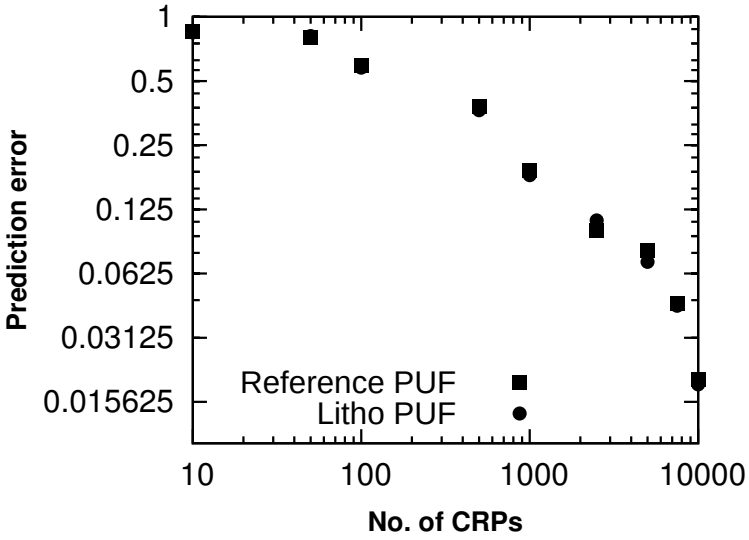


Figure 6.5. Prediction errors from SVM attacks on Litho-aware arbiter PUFs

can be broken using almost the same number of training CRPs as conventional PUF designs.

6.2 PHAP: Password based Authentication System using PUFs

In the previous section, we presented circuit design strategies for improving the performance metrics of PUFs. In this section, we present an authentication system/protocol employing PUF based circuits. One of the most promising applications of PUFs is the verification of identity of hardware devices. This is generally done by

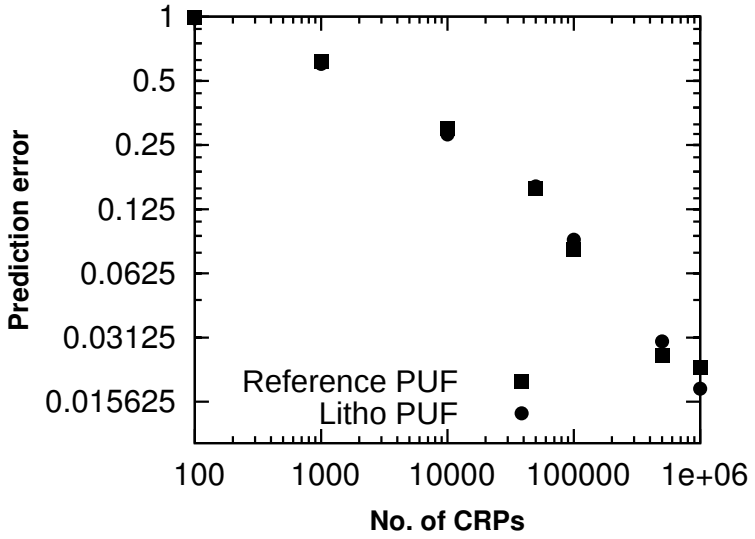


Figure 6.6. Prediction errors from SVM attacks on Litho-aware current-based PUFs

a Trusted Authority (TA), which has a database of CRPs of various PUF instances. The database will be created during the *enrollment* process by applying challenges to a PUF and storing the corresponding response(s). During the *authentication* process, TA sends a particular challenge from its database to the hardware and compares the obtained response with the one in the database [86]. If the responses match, the device will be authenticated. However, one of the setbacks in the protocol is that TA will not be able to distinguish whether the hardware is with a trusted party or an adversary.

In this section, we present Password based Hardware Authentication using PUFs (PHAP), a system in which TA will be able to determine the possession of the trusted hardware using a simple *user password*. A separate one-time shared key (also called as *session password* further in the document) between TA and the user will be mixed with the initial PUF response corresponding to the challenge sent by TA and be used as a seed for a pseudo-random number generator such as a Linear Feedback Shift Register (LFSR). The output of LFSR then serves as a new challenge for the PUF block. This challenge will be completely different from the initial challenge and ex-

tremely difficult for an adversary to predict by random guessing, thereby providing an additional layer of security. We also leverage the time difference between real time execution by a trusted party and simulation time of the system (by an adversary) for authentication purposes. In this work, we show that the time difference for an adversary can be amplified using the one-time session password, as only the trusted party can obtain the correct response within the stipulated time (t_{max}). This is a crucial property used in Simulation Possible but Laborious (SIMPL) systems [77]. However, an adversary can predict the one-time session password using a brute-force attack only to end up providing the correct response well beyond the stipulated time. Also, the usage of one-time session password helps to alleviate the need of an enrollment process, as TA will be able to compute the response after the authentication process is initiated.

6.2.1 Background and Related Work for PHAP

A very common problem in computing is secure authentication [28, 36, 70, 71], and the existing protocols have certain flaws associated with them. The problems of storing secret keys for performing cryptographic operations were discussed in the earlier chapters. To counterattack the problems involved in key storage, PUFs have been introduced in the literature in which the hardware decides the mapping of challenge C_i to response R_i. Various protocols for authentication using PUFs have been proposed in the literature [25, 72, 86]. The authentication protocol proposed in [86] focuses on comparing the response obtained from a device with the response generated during enrollment process. However, the protocol proposed in [86] fails to determine the possession of the trusted hardware. An adversary possessing the trusted hardware will be able to authenticate himself along with the hardware. A lightweight challenge response protocol proposed in [72] utilizes noisy PUFs for authentication. Various robust authentication protocol schemes have been proposed in [25], where PUFs have

been used to improve the resilience of authentication protocols. Some of the schemes employ user's password for authentication and they tend to be more resilient towards security attacks such as modeling. However, hardware level implementation and the associated issues have not been analyzed. In this work, we tend to improve the authentication protocol proposed in [25] for hardware employing a PUF as a source of security by tightly integrating an one-time session password into the authentication protocol. A closer version to the proposed work is PEAR [36], where PUFs have been used for user password maintenance. However, the scheme requires initial enrollment process, where the CRPs of PUFs are stored in a database for future authentication purposes. We tend to alleviate this problem by publishing a public description of the system and a public simulation algorithm such that the TA can compute the response once an authentication request is initiated similar to SIMPL systems.

6.2.2 SIMPL Systems

SIMPL system [77] is a public key version of Physically Unclonable Functions. They possess a certain binary description to facilitate public simulation and prediction in a slower fashion than real time execution by the hardware. Since CRPs have to be stored prior to authentication, they must be maintained secretly over the entire course of time (multiple authentication processes). SIMPL tends to overcome this problem by publishing a public algorithm Sim along with a publicly available description $D(S)$ of the hardware system S, that allows public emulation of the system. However, the public emulation takes sufficiently longer duration beyond the stipulated time (t_{max}), within which the response must arrive for authentication. The parameter t_{max} is predefined before the authentication process. According to [77], a system S can be called as a SIMPL system, if it meets the following requirements:

Table 6.8. Description of Notations

Signal	Description	Length (bits)
C_{user}	Challenge from TA	64
R_{inter}	Intermediate response from PUF block	32
R_{padded}	Intermediate response padded with shared secret key	64
C_{LFSR}	LFSR Output - New challenge to the PUF Block	64
R_{user}	Response sent to TA for authentication	32

1. A partially disordered system S upon excitement with a challenge C_i, produces a response R_i. The mapping function F_S is decided by the disorder present in the system.

2. It is possible to obtain the response R_i for a particular challenge C_i by simulating a public simulation algorithm Sim using description $D(S)$ of the system.

3. Any possible emulation or algorithm that simulates the response of S should be considerably slower than the real time behavior of system S.

4. The system S must be physically unclonable.

There exists a subtle difference between a SIMPL and PHAP system. In SIMPL, anyone with the public description $D(S)$ and Sim can simulate the system, but considerably slower than S. However, in PHAP, only an adversary simulates the system S considerably slower than the real time execution of S by a trusted party. Hence, the difficulty in simulation of PHAP is due to one-time session password prediction rather than the nature of hardware as in SIMPL systems. Moreover, the terms S, Sim and $D(S)$ have been used in the paper to maintain consistency.

6.2.3 PHAP Architecture

The architecture of the proposed authentication system is shown in Figure 6.7. Before describing the components, the notations used in the system are shown in Table 6.8.

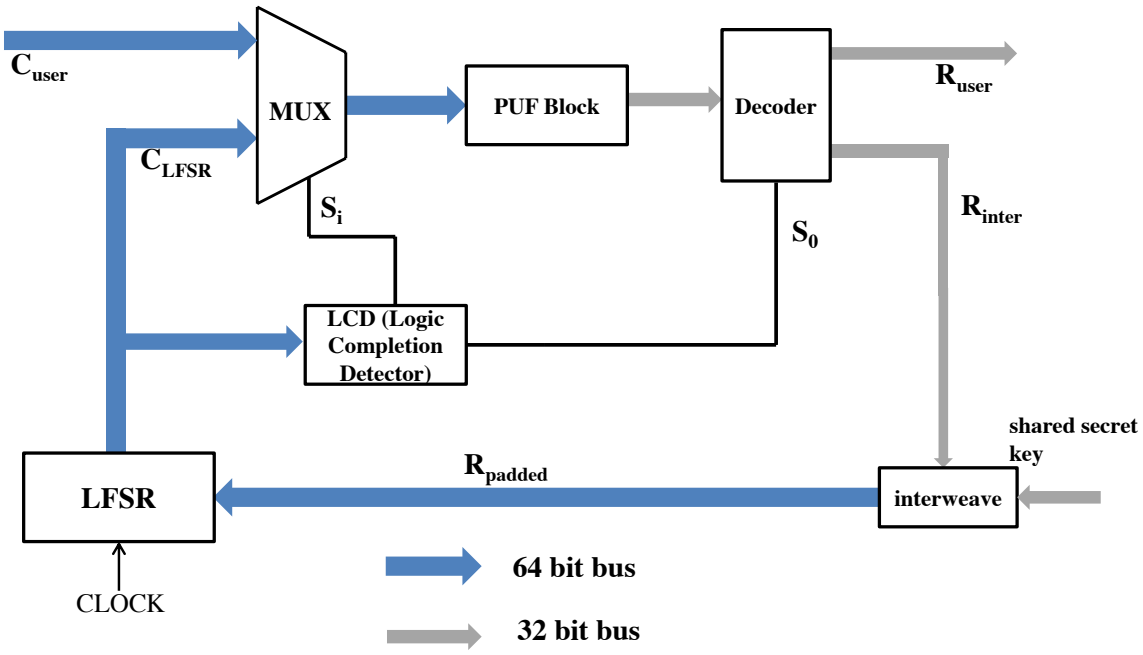


Figure 6.7. PHAP Architecture

1. **MUX**:The MUX chooses the challenge to be applied to the PUF block (S_i= 0 applies the user challenge C_{user} and $S_i = 1$ applies the LFSR output as a new challenge to the PUF block or vice versa).

2. **PUF Block**: PUF block can be a delay based PUF (Arbiter or Ring oscillator PUF) and the block contains 32 individual PUF units so as to produce a 32 bit response for a given 64 bit challenge. Other lengths can also be used. Feed-forward arbiter PUF architecture has been used in order to increase the attack resistance to modeling attacks (95 % predictability after 50000 rounds [88]). In a feed-forward arbiter PUF, some of the challenge bits are determined internally as a result of a race.

3. **DECODER/DEMUX**: This is used in order to forward the response either as an output (R_{user}) or as an input to the LFSR. The decision is based on the state of the select bit (S_o).

4. **LFSR**:Linear Feedback Shift Register is used in order to shift the seed by certain number of clock cycles (few ten thousands) so as to produce a new challenge for the PUF block. The seed is obtained after interleaving the response with the shared secret key (session password). Here, the interleaving positions can be made public and included in the description of the system S to allow public simulation.

5. **Logic Completion Detector (LCD)** - LCD is used to generate the select bit signals for MUX and DECODER. Once the LFSR computation is complete, LCD output will go to logic high, which will set S_i and S_o such that C_{LFSR} is applied to the PUF block and R_{user} is generated respectively.

6.2.4 Authentication protocol

In this section, the details of the operation of the system is presented. This, in turn represents the authentication protocol being used. Let $hash(.)$ represents one-way collision resistant hash function and ID represents the unique identifier for a PHAP system. We also make an assumption that the description of PHAP system $D(S)$ and a simulation algorithm Sim to simulate the system S are made public. Moreover, *user password* refers to the password being used for user authentication and *session password* refers to the shared secret key used for LFSR computation.

1. **Enrollment**: Even though we call the initial process as *enrollment*, it does not involve the collection of CRP pairs as in traditional enrollment process. Here, TA is responsible for generating IDs (identifier) unique for multiple instances of the system S and a set of one-time user and session passwords in the form of a password card. A sample one-time password card is shown in Table 6.9. The size of this card can be changed in accordance with user discretion. The user can opt to use a password apart from the passwords given by the trusted

Table 6.9. Password Card (either session or user password)

Pointer / Password	1	2	..	n
1	11 / $Password_{11}$	12 / $Password_{12}$	1.. / $Password_{1..}$	1n / $Password_{1n}$
2	21 / $Password_{21}$	22 / $Password_{22}$	2.. / $Password_{2..}$	2n / $Password_{2n}$
..	..1 / $Password_{..1}$	..2 / $Password_{..2}$	.... / $Password_{....}$	..n / $Password_{..n}$
n	n1 / $Password_{n1}$	n2 / $Password_{n2}$	n.. / $Password_{n..}$	nn / $Password_{nn}$

authority after first authentication. In case of usage in embedded systems, the password can be entered through a keypad, so as to allow external interface.

2. **Authentication**

- Once the user/device raises an authentication initiation to TA, a pointer is sent to the user, which corresponds to the location of the one-time user and session passwords in the password card shown in Table 6.9. User sends back the tuple $<ID$,$hash$(user password)$>$, with which the user can be authenticated as TA has a copy of the user password. The collision resistant $hash(.)$ ensures that no other user password has the same hash value as that of the required password. The ID is needed by TA to surf through the database, if it has to maintain a large number of devices in the database. A sample database of trusted authority is shown in Table 6.10.
- The user is then presented with a challenge C_{user}, whom upon evaluation of the system sends back the response R_{user} to TA. Internally, the one-time session password is interleaved with R_{inter} to obtain R_{padded}, which will be used as a seed for LFSR computation. Here, it is necessary to note that the session password is not transmitted over the network to TA.
- In the mean time, TA also computes the response R_{TA} for the challenge C_{user} using the algorithm Sim and the one-time session password.

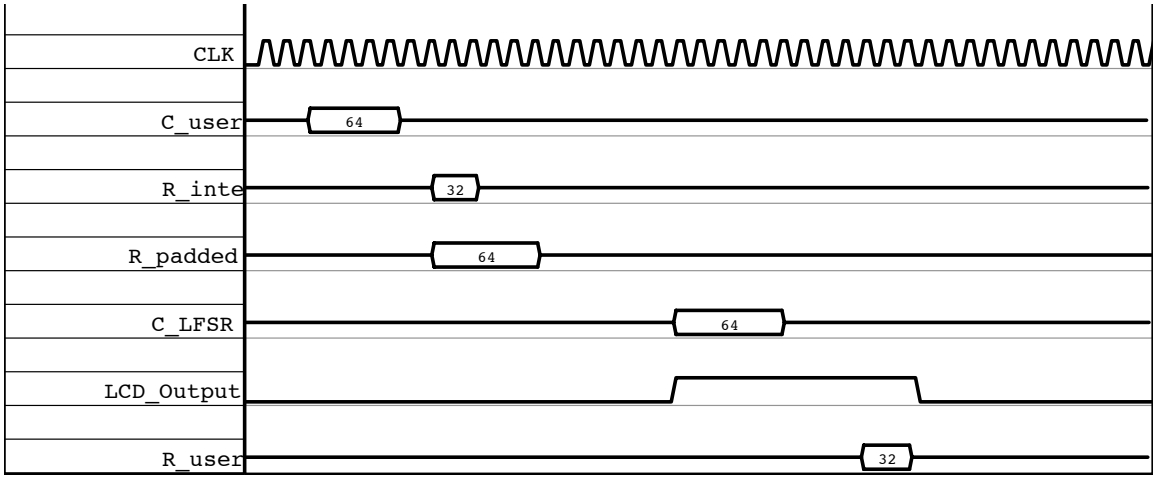


Figure 6.8. Timing diagram for PHAP

Table 6.10. Trusted Authority's Database

	Challenge	Response
$<(ID_1, hash(\text{user password}_1..\text{user password}_k)>$	$C_1.....C_k$	$R_1.....R_k$
$<(ID_2, hash(\text{user password}_1..\text{user password}_k)>$	$C_1.....C_k$	$R_1.....R_k$
...	...	...
$<(ID_n, hash(\text{user password}_1..\text{user password}_k)>$	$C_1.....C_k$	$R_1.....R_k$

- The user is authenticated if
  (a) The response R_{user} matches R_{TA}.
  (b) The time taken for the user to compute and send the response R_{user} is at most t_{max}, where t_{max} is pre-defined and is the maximum time within which the user must send the response to be authenticated.

A timing diagram to illustrate the operation of PHAP system is shown in Figure 6.8. Here logic 'high' represents the action being undertaken at that particular time instance. A formal definition of the authentication protocol being used is defined in Table 6.11.

It is known from previous chapters that delay-based PUFs can be modeled using machine learning algorithms to very high accuracies. Hence, there is a strong pressing need for better design of delay-based PUFs that are inherently resistant to modeling

Table 6.11. Authentication Protocol

Authenticate - User and Device S authentication to trusted authority
- S initiates the authentication request to TA - TA sends back the pointer to the user - User sends back the tuple <(ID,$hash$(user password))> to TA - TA sends back user authentication acknowledgment if user password is corrrect along with C_{user} - User sends back R_{user} after computation - TA computes R_{TA} using $D(S)$ and Sim - Hardware is authenticated if $R_{TA} = R_{user}$ and time taken for R_{user} computation is $\leq t_{max}$

attacks. PHAP helps to overcome this issue by using an easy-to-model delay-based PUF and adding an external password that partly determines the final challenge. Here, some or all of the bits in challenge will be unknown to the adversary. This makes the modeling attack extremely difficult, as some of the challenge bits are masked from an adversary. Also, PHAP helps to mask the session password from being transmitted directly through the network. This property can be useful in applications in which a password (typically *hash*(password)) is solely used for authentication purposes.

6.2.5 Simulation Results

In this section, we present some simulation results of the proposed system. To simulate the PUF block, statistical circuit simulations have been used as shown in Figure 4.6 and threshold voltage variations were assigned from a normal distribution with a 3σ deviation of 90mV, to be consistent with ITRS specifications [1]. The other tools used in the work include Synopsys DC Compiler, Perl and Matlab.

6.2.5.1 PUF Block

Some of the significant performance metrics of a PUF include *uniqueness* and *reliability*. Since uniqueness directly affects the security of the system, it was analyzed by computing the hamming distance distribution. In order to compute the hamming distance distribution, 64 challenges were applied over 32 PUF instances (32 bit output considered) and Monte Carlo simulations were run. The distribution is shown in

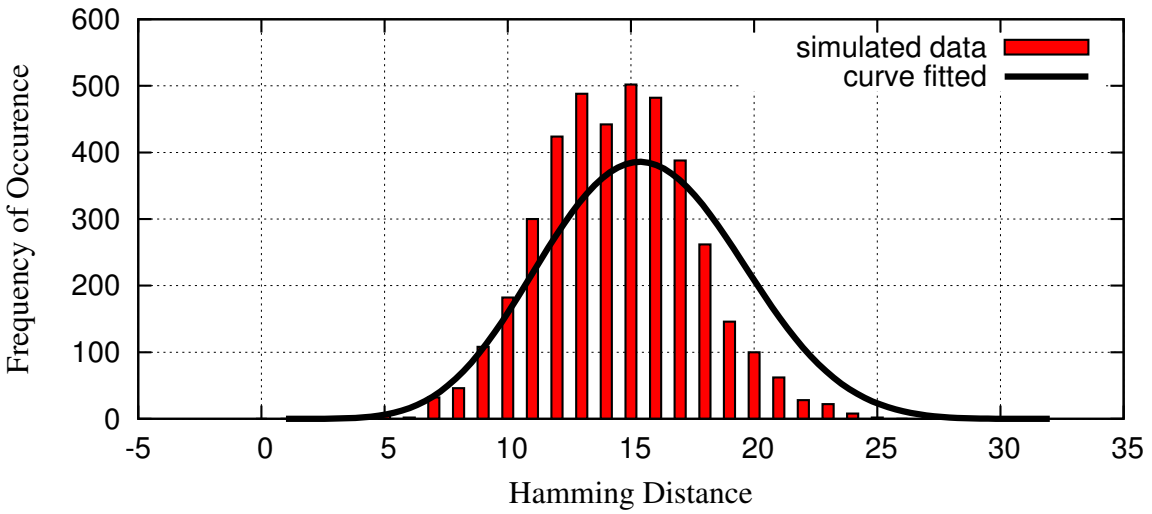


Figure 6.9. Hamming Distance distribution of the PUF block

Figure 6.9. It can be observed that the mean of the distribution is around 16, which corresponds to a uniqueness of 50%.

6.2.5.2 PHAP System

LFSR is one of the integral components in PHAP system. Fibonacci LFSR implementation was used and 50,000 cycles was set as the rounds by which the seed will be shifted. However, the number of rounds can be changed during runtime. The hamming distance distribution of the LFSR output after 50,000 shifts for around 10,000 seeds is shown in Figure 6.10. The distribution shows that the successive LFSR outputs typically have a significant hamming distance, thereby providing higher degree of randomness to the whole system.

In order to observe the uniqueness of R_{user} which corresponds to the final response used in authentication, several experiments were performed to capture various possible scenarios and hamming distance distributions were obtained. In experiment 1, around 50 random session passwords were chosen and interleaved with 40 different initial PUF responses R_{inter} corresponding to C_{user}. The corresponding hamming

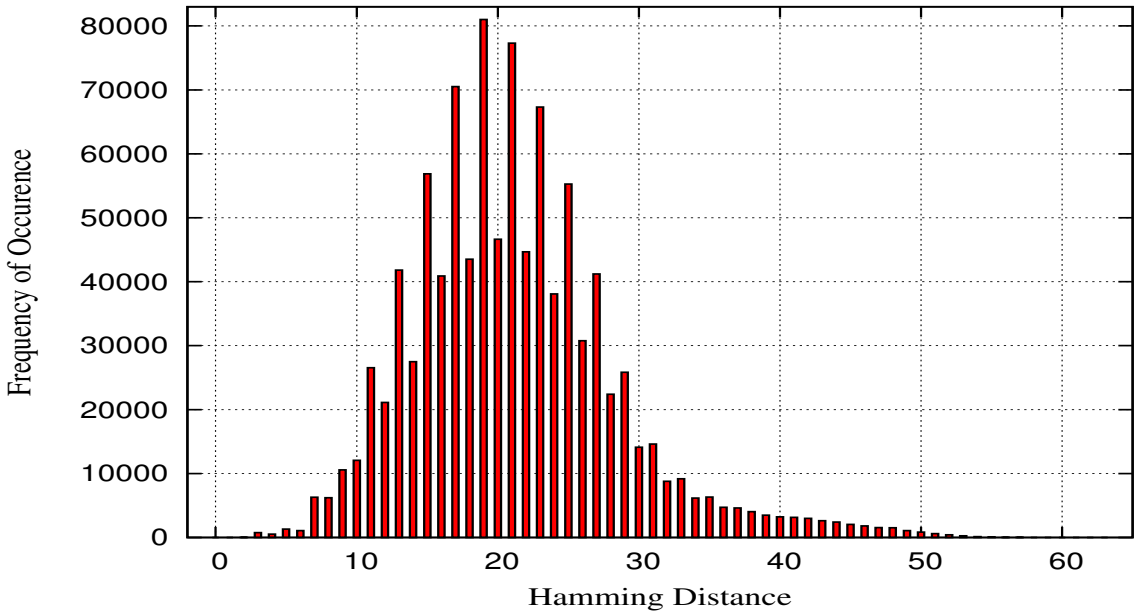


Figure 6.10. Hamming distance distribution of the LFSR output

distance distribution is shown in Figure 6.11. It can be inferred that PHAP has a uniqueness of about 41 %.

In experiment 2, the scenario in which the seed to LFSR (R_{padded}) varying from 1 to 40 bits in succession was analyzed. This was done by carefully picking C_{user} and

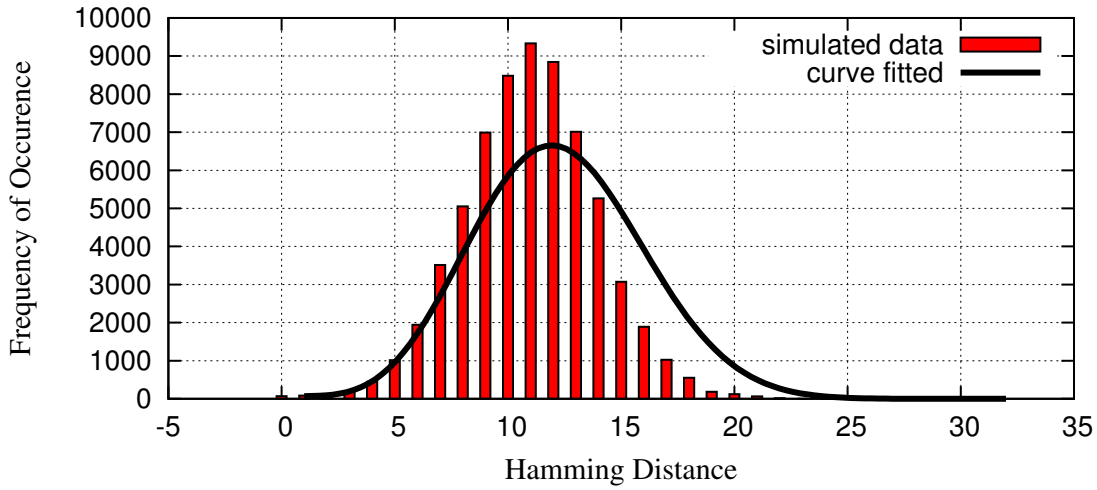


Figure 6.11. Hamming distance distribution for various session passwords experiment

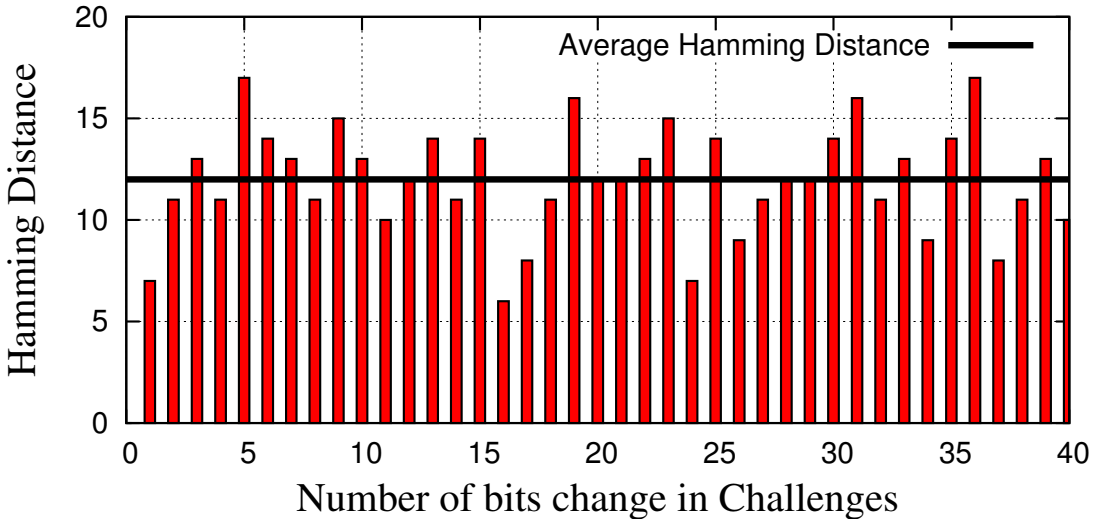


Figure 6.12. Hamming distance vs R_{padded} varying from 1 to 40 bits

session password, such that R_{padded} varied from 1 to 40 bits in succession. The plot showing the variation in hamming distance with respect to various seeds to LFSR (R_{padded}) is shown in Figure 6.12. It can be inferred that the system responds to R_{padded} varying by 1 bit in succession by producing a hamming distance of 12 bits in average over the responses (R_{user}).

In experiment 3, the scenario in which the LFSR seed R_{padded} varying by 1 bit was analysed. 32 different R_{padded} varying by 1 bit were created by manipulating C_{user} and session passwords.The hamming distance was computed for various R_{user} corresponding to C_{LFSR} and is shown in Figure 6.13. It can be inferred that for a single bit change in R_{padded}, the system responds by producing a response with a hamming distance of 13.5 in average.

These experiments provide an insight that PHAP is unique in its responses under varying conditions. The presence of LFSR helps to mask the session password by shifting it through a certain number of clock cycles, which helps in maintaining the uniqueness as well as security of the system.

6.2.6 Security Analysis of PHAP

In this section, we present a formal description and analysis of the security of PHAP system. Several lemmas are presented along with proofs by considering threats from an adversary A. The security of PHAP is compromised, if A is able to obtain or predict the user and session passwords and simulate the system within t_{max} to obtain a correct response. Some of the possible threats that PHAP faces are identical to that of PEAR [36], given the similarity in the authentication protocol used.

Lemma 1

An adversary A can attack PHAP by random session password prediction with negligible probability.

Proof: Since the session password is not transmitted over the network, an adversary A can attack the system by random session password prediction. If an attacker can predict the session password correctly, the system can be simulated using Sim and $D(S)$ to compute the response within t_{max}. However, the session password is 32 bits long and it is sufficiently difficult enough to predict the correct session password,

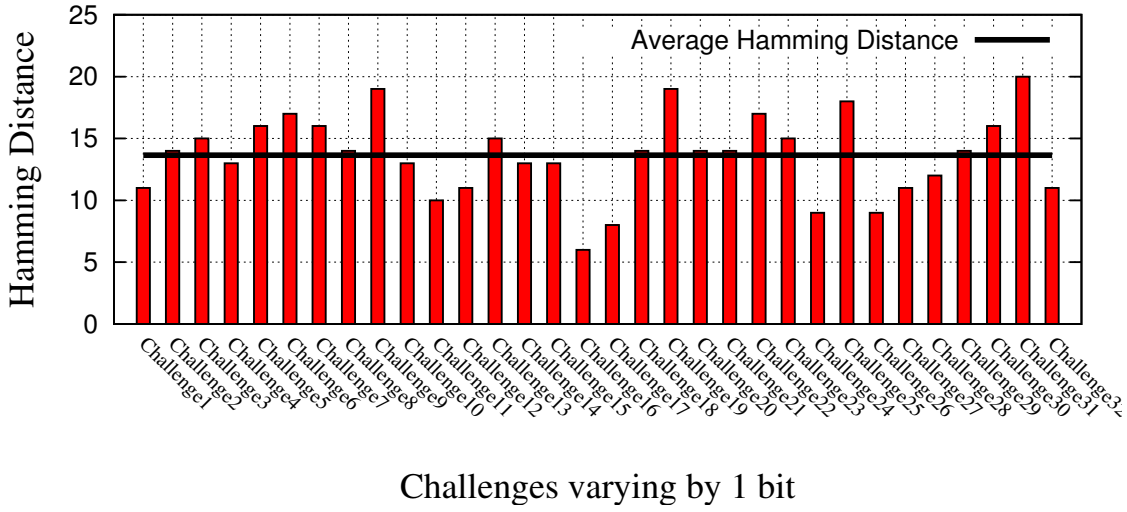


Figure 6.13. Hamming distance vs R_{padded} varying by 1 bit

due to the size of password space (2^{32}). The attacker can continue to predict the session password randomly by omitting the previously tried wrong password until the correct session password is attained. However, the simulation time corresponding to this action will be extremely high and authentication will be interrupted once t_{max} is elapsed. Hence, the adversary A will never be able to realize the instant at which the correct session password is predicted. The simulation time (t_{sim}) is computed using

$$t_{sim} = \text{time to compute initial PUF response}(R_{inter}) + \text{time for LFSR computation} + \text{time to compute } R_{user}$$

The plot showing the probability of correct password prediction by an attacker in the n^{th} round with the corresponding simulation time t_{sim} is shown in Figure 6.14. The Figure 6.14 also shows the variation in simulation time for various LFSR operation frequencies (50 MHz and 1 GHz) and different LFSR shift cycles (10,000 and 50,000 cycles) by which the seed will be shifted.

It can be inferred that the prediction probability is very low for a given t_{max} = 10 ms and that the attacker can predict the correct session password only with a probability of $\leq 10^{-6}\%$. Hence, PHAP can leverage the time difference between real time execution of S and simulation time by an adversary for authentication, even though the description of system S and a public simulation algorithm Sim is made public.

Lemma 2

An adversary A cannot obtain any useful information from the data transmitted between the hardware and trusted authority.

Proof: In case of passive eavesdropping, the attacker cannot recover any useful data communicated between the hardware device and TA. The only data transmitted by the device to TA for authentication is the tuple $<ID$,$hash$(user password)$>$. Even if

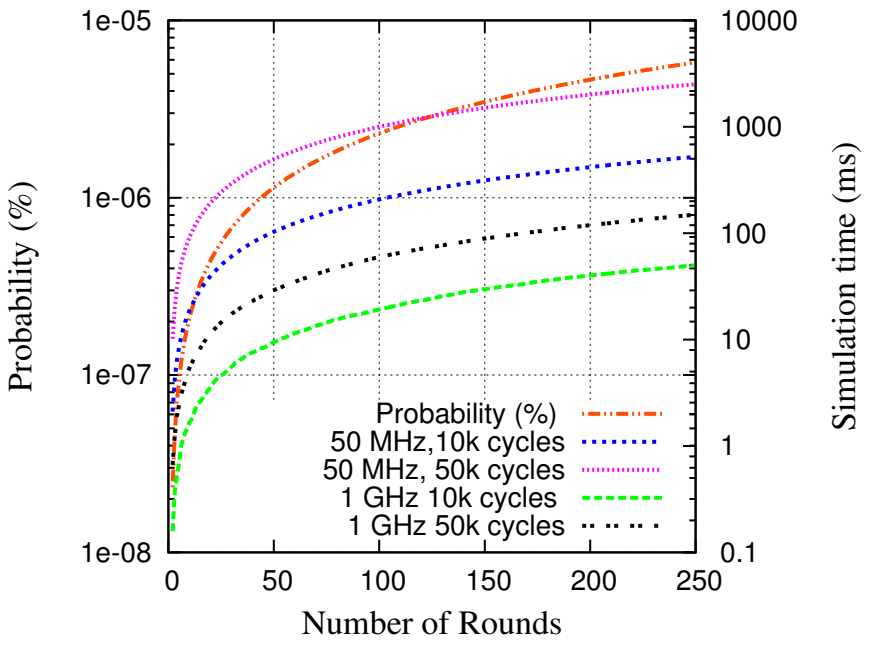


Figure 6.14. Simulation time and Probability plot

A has extensive computational and storage capabilities such that the hash values of all the user passwords are stored (typically takes about 80 GB, if SHA-0 is used), the usage of one-time user password makes user authentication by A extremely difficult. This is mainly due to the absence of user password ordering based on the pointer generated by TA during enrollment process.

Lemma 3

An adversary A cannot recover any useful data from the information transmitted between TA and hardware.

Proof: The only data transmitted from TA to the hardware is the tuple <pointer, C_{user}>, once an authentication request is initiated by the user. Since A has no clue about the order of user passwords in the password card, the pointer is of no use for A. Moreover, the presence of shared key (session password) makes the correct simulation

of responses for C_{user} by A extremely difficult.

Lemma 4

Given physical access to the device,an adversary A can impersonate a truster user to the trusted authority with a lower probability.

Proof: Since PHAP helps to determine the possession of the trusted hardware using one-time user password, A would be required to predict the correct user password based on the pointer. Even if we assume that A succeeds in predicting the user password,the attacker would have to undergo extensive execution of S by random session password prediction explained in Lemma 1 for hardware authentication. It is important to note that authentication will be interrupted once t_{max} elapses and the instant at which correct session password is attained will be unknown to A.

Lemma 5

A trusted user can authenticate himself along with the hardware to the trusted authority with higher probability.

Proof: Since a trusted user would have access to the password card issued by TA, user and hardware authentication is guaranteed with high probability. However, authentication can fail due to additional delays in the network, such that the response R_{user} reaches TA after t_{max}. In such cases, the user can initiate the authentication again and continue until the user and hardware authentication process is complete. From the trusted authority side, t_{max} can be increased a little higher if additional delays in the network are noticed.

Lemma 6

Using the previously utilized ID and user password, A can authenticate himself to the trusted authority with negligible probability.

Proof: In case of passive eavesdropping, an attacker can capture the tuple $<ID$,$hash$(user password)$>$ during trusted user's authentication. During subsequent authentication, TA will search for the incoming $hash$(user password) in the database to identify if the user password has been used already. If a match is found, user authentication will be interrupted. This prevents A from using the captured $hash$(user password) to authenticate himself to the trusted authority.

Theorem 1

PHAP based authentication protocol provides a secure means of trusted user and hardware authentication.

Proof: By Lemma 1, we can be sure that an attacker will be able to predict the response with negligible probability. Lemmas 2 and 3 ensure that data transmitted between the hardware and TA does not leak any information required to compute the response within the stipulated time. Moreover, Lemma 4 helps to ensure that an adversary A can impersonate the trusted user even while possessing the system S with lower probability. Lemma 5 shows trusted user and hardware authentication. Lemma 6 shows that the previously utilized information is of no use for an attacker and also shows how the system can protect itself against an adversary authentication. Hence, we can conclude based on the lemmas that PHAP based authentication protocol helps for secure hardware authentication by a trusted user.

CHAPTER 7

SILICON PROTOTYPING

The different PUF circuits described in the earlier chapters were fabricated in a prototype chip, named *sugarloaf* using IBM 32nm SOI technology. The implemented PUF circuits are shown in Table 7.1. Different circuits on chip include arbiter, current-based PUFs, nlcPUFs and reliability monitor for delay-based PUFs. Arbiter, feed-forward and current-based PUFs were implemented using conventional and litho-aware techniques. The unpackaged and packaged dies are shown in Figure 7.1. The packaged versions of the dies were used for post-silicon validation. Around 32 instances of each circuit were implemented on chip and the total die count was 40. The architecture of the test chip containing PUF circuits is shown in Figure 7.2. The configuration bits are used to select a particular PUF instance from the available PUF banks. The bits are loaded using a scan chain setup. The controller logic, apart from selecting a particular PUF instance, also helps to clock gate the unused PUF instances. The scan chain is 141 bits wide, which accepts the 1-bit serial data per clock cycle. The description of the configuration bits is shown in Table 7.2. The configuration bits $b_{12}..b_{10}$ selects a PUF bank from the available banks. The bits $b_9 b_8$ and $b_7..b_3$ are used to select a particular PUF configuration (64, 80 or 128-stage) and a particular PUF instance from the selected configuration in the PUF bank respectively. The *go* signal acts as a scan-enable signal when asserted low. Once *go* is asserted high, the PUF computation is initiated.

The challenges for the PUF circuit are generated using a Linear Feedback Shift Register (LFSR). Fibonacci type LFSR of different lengths (64, 80 and 128) were

Table 7.1. Available PUF circuits in *sugarloaf*

Type of PUF	No. of instances	Available configurations
Arbiter PUF	32	64, 80 and 128 stages
Feed-forward arbiter PUF	32	64, 80 and 128 stages
Current-based PUF	32	64, 80 and 128 stages
nlcPUF	32	64, 80 and 128 stages
Litho-aware arbiter PUF	32	64, 80 and 128 stages
Litho-aware feed-forward PUF	32	64, 80 and 128 stages
Litho-aware current PUF	32	64, 80 and 128 stages
Reliability monitor for delay-based PUF	32	64 and 128 stages

implemented in order to generate random challenge bits for the available PUF configurations. The seeds are loaded into the LFSR registers using the scan chain. Apart from LFSR challenge generation, user specified challenges can also be applied to the PUF circuits through bypassing the LFSR logic. This is particularly useful for the benchmarking work proposed in the dissertation. The architecture of challenge gen-

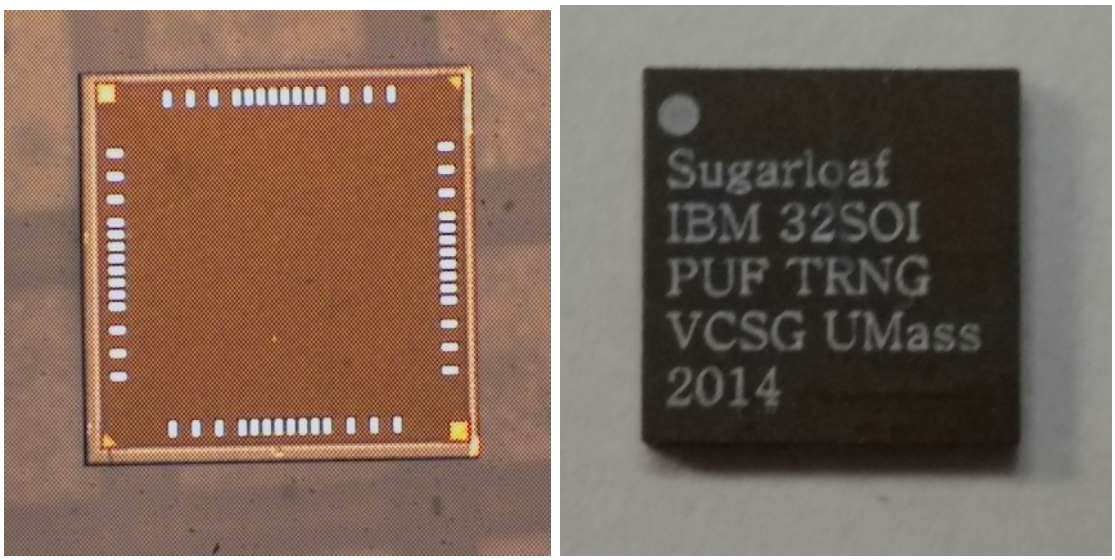


Figure 7.1. Unpackaged and Packaged *sugarloaf* die photos

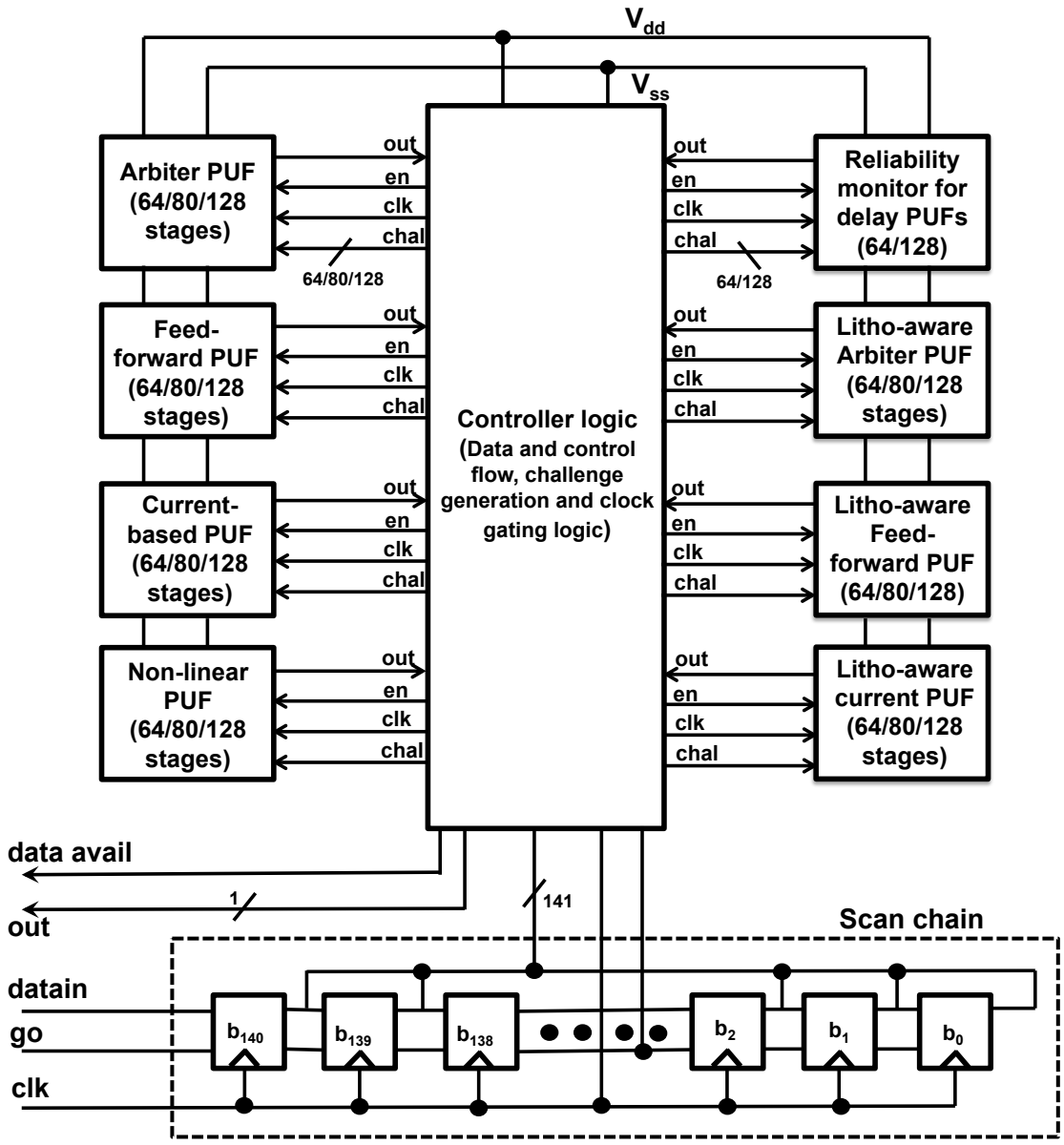


Figure 7.2. Architecture of PUF portion in *sugarloaf*

eration logic, which is a part of the controller logic is shown in Figure 7.3. The fabricated chip is 2mmx2mm in area. The area number of a single instance of different PUF circuits are shown in Table 7.3. The area numbers are shown only for the PUF circuits and the details of the controller logic are omitted. The layout snapshot of the PUF banks with controller logic is shown in Figure 7.4.

Table 7.2. Configuration bits description

Bits	Function
$b_{140}..b_{13}$	User specified challenge/LFSR feed
$b_{12}..b_{10}$	PUF bank selection
b_9, b_8	PUF configuration selection
$b_7..b_3$	PUF instance selection
b_2	Enable computation
b_1	Challenge selection
b_0	Read output

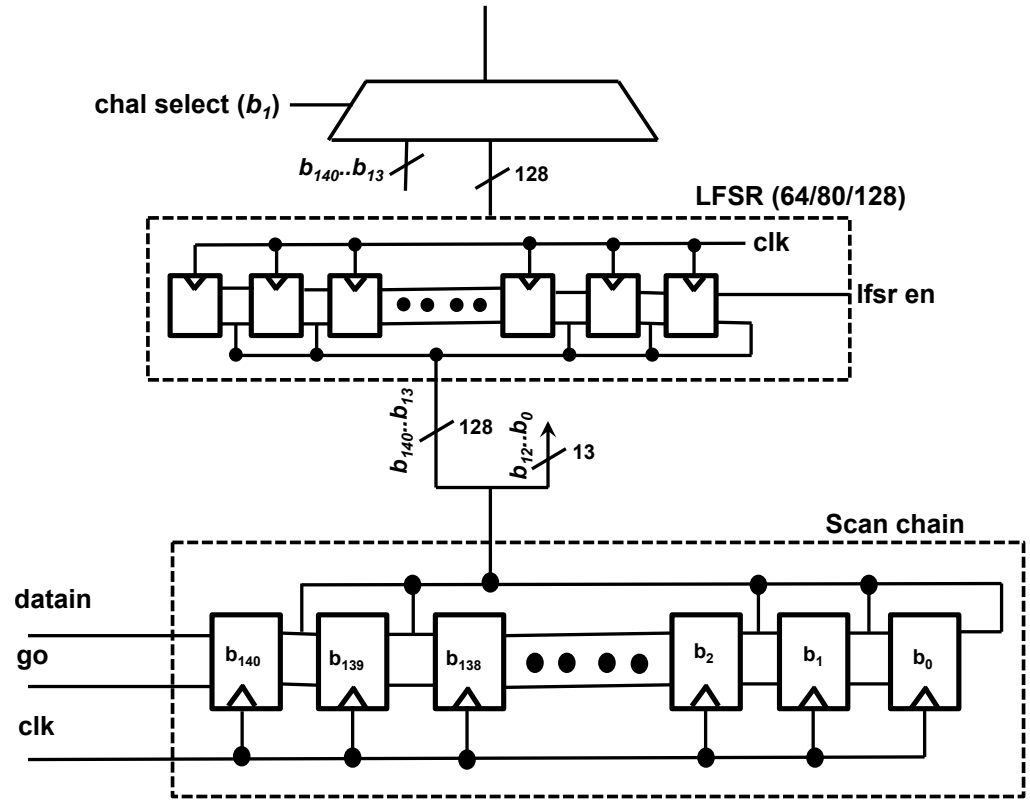


Figure 7.3. Challenge generation for PUF circuits in *sugarloaf*

7.1 Measurement setup

As explained above, the packaged versions of the dies were used for post-silicon validation. To perform post-silicon validation, an FPGA based testing environment was setup and synchronized with the test chip. The block diagram of the testing setup is shown in Figure 7.5. A standard QFN56 test socket was used to mount the

Table 7.3. Area details of a single instance of various PUF circuits

Type of PUF	# Stages	Area (μm^2)
Arbiter	64	1250
	80	1560
	128	2496
Feed-forward	64	1870
	80	2340
	128	3750
Current PUF	64	650
	80	810
	128	1290
nlcPUF	64	790
	80	986
	128	1579

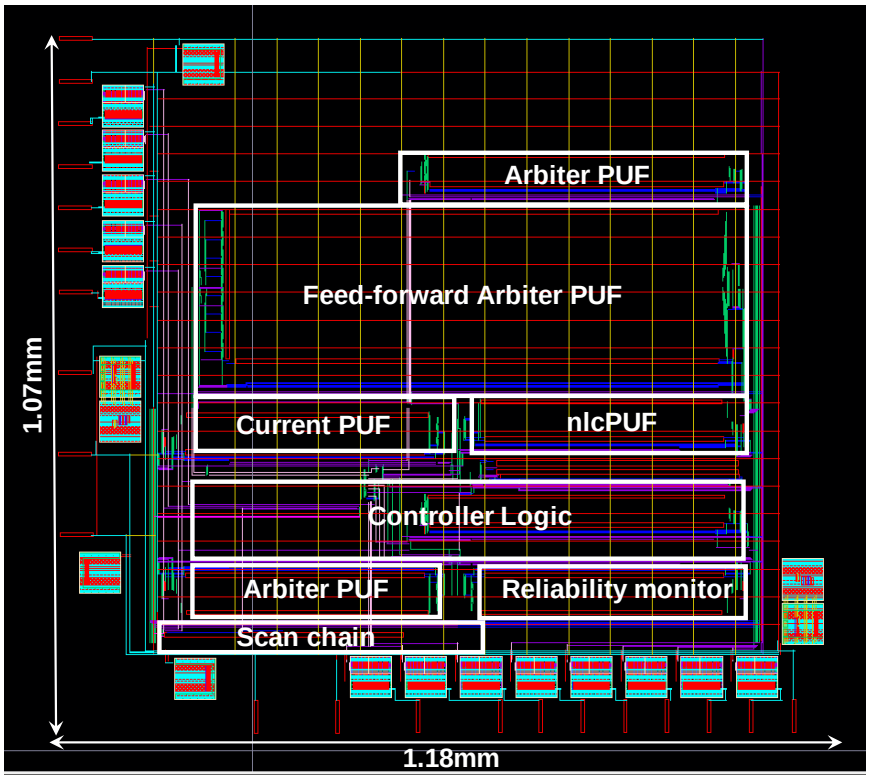


Figure 7.4. Layout snapshot of the PUF banks with controller logic

packaged die. Spartan-3E FPGA which operates at a voltage of 2.5V was used for post-silicon validation setup. An off-chip bi-directional voltage converter (TXB0104)

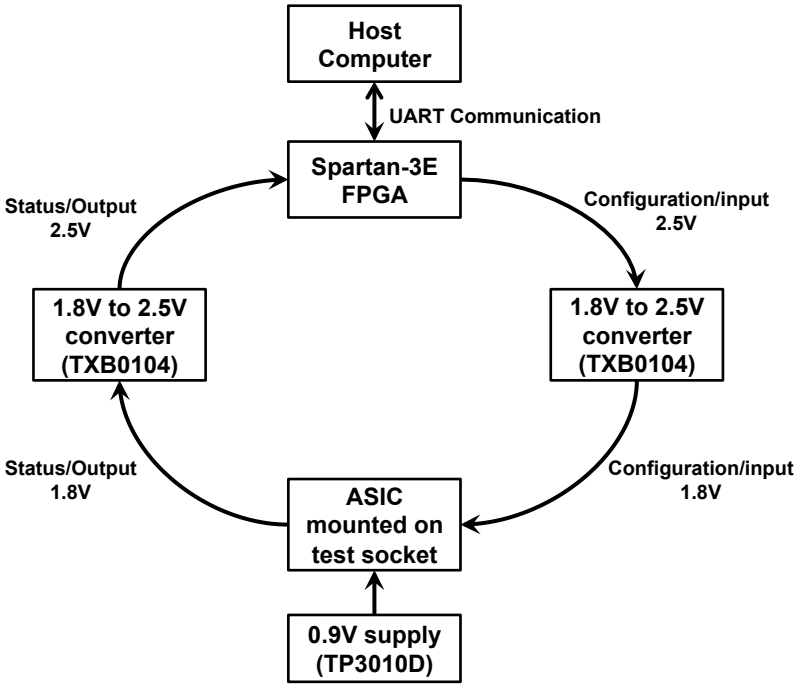


Figure 7.5. Post-silicon validation setup

between 2.5V and 1.8V was used for interfacing the FPGA and ASIC. I/O cells were available on the test chip for bi-directional voltage level translation between 1.8V and 0.9V for the signals (clock, configuration bits and output). Tekpower TP3010D DC power supply was used for generating 0.9V for the test chip. The FPGA helps in loading the configuration bits into the chip and reading out the response bits from the chip. The configuration bits are used to select the type of PUF among the other PUF implementations and also to choose a particular instance of the selected PUF circuit. Once the responses are available, the FPGA dumps the data into the host computer via UART communication.

The post-silicon validation results of the performance metrics of PUF instances can be found in the corresponding chapters.

CHAPTER 8

STATISTICAL BENCHMARKING FOR PUFS

8.1 Introduction

[1]In the earlier chapters, we presented the security vulnerabilities of cryptographic hardware blocks. We also explored the vulnerabilities of PUF circuits to various modeling and side-channel based attacks. To enhance the modeling attack resistance, we presented a novel PUF circuit based on non-linear current mirrors. To enhance the performance metrics of PUF circuits, we presented novel solutions from the perspectives of IC fabrication and protocol design. Although the PUFs are vulnerable to modeling attacks, it takes an immense effort to completely understand the vulnerability of the circuit and build a parametric model that can be scaled for various attacks. The parametric modeling requires non-trivial level understanding of machine learning algorithms. The "conceptualization phase" of a new PUF design involves substantial effort and time from the PUF designer to build the parametric model and evaluate it against possible modeling attacks. In such a scenario, the existence of a benchmark suite for "strong PUFs" may reduce the effort and time overheads involved in the security evaluation phase for new PUF designs. Such a benchmark suite can also help in fine-tuning the PUF architecture to harness maximum randomness and security from the PUF circuit. In this chapter, we present a statistical NIST based benchmark suite for characterization of "strong PUFs". In particular, we present an analysis of the impacts of varying methods of challenge generation on the performance of PUF

[1]This is a joint collaborative work with Dr.-Ing.Ulrich Rührmair, Technische Universität München.

designs. Such an analysis helps the users for choosing an optimum challenge generation method for improving the randomness of PUF designs. As the benchmark suite is based on the existing NIST suite, the security analysis does not involve effort and time-consuming parametric modeling involved in machine learning based attacks.

8.1.1 Need for Benchmarks

The enormous potential of modeling techniques makes the design of efficient "strong PUFs" with improved ML-resilience an interesting research problem. However, it is non-trivial to judge the exact ML-resilience of PUFs. The question whether a PUF can be broken efficiently by ML based attacks often depends on whether numeric "models" with certain properties exist for this PUF [2]. As described earlier in this chapter, the identification of ML algorithms and suited models is time consuming and requires a strong ML background. Furthermore, the practical failure to identify suitable ML algorithms, or a model with certain properties, is no guarantee or "proof" that such an algorithm or model indeed does not exist.

These constraints call for easily scalable and applicable benchmarks. The benchmarks, in ideal scenario, should satisfy the following properties:

1. No in-depth knowledge of ML algorithms is required.

2. Easily scalable to various "strong" PUF architectures.

3. Simple parametric models without additional properties should be used.

---

[2] A "model" of a PUF is a function F that describes the mapping of challenges to their corresponding responses. F usually takes as inputs the challenge C and the PUFs individual, random internal parameters P (for example the runtime delays in an arbiter PUF), and outputs the corresponding response R, i.e., $R = F(C;P)$. This is often referred to as a *parametric model* of the PUF. ML based attacks often require the parametric model to satisfy certain properties. For example, F must be linearly separable in order to make support vector machines applicable, or differentiable in order to allow the use of logistic regression, etc. For some PUF architectures, it might be really difficult to construct such parametric models that satisfy the desired properties required by ML algorithms.

4. Fine-grained security evaluation should be possible. For example, the suite should be able to distinguish varying PUF bit-lengths and other architectural parameters.

8.1.2 Contributions

Our major contributions include:

1. We present a statistical benchmark suite to evaluate the security levels of "strong" PUFs.

2. We evaluate the impacts of varying challenge generation methods on the performance of PUFs.

3. We present a brief analysis of response compressibility from "strong" PUFs and their correlations to the benchmark suite.

8.2 Statistical Benchmarking

As described earlier, we base our benchmarks based on NIST suite. First, the PUF target is chosen and a large sample of challenge-response pairs from the PUFs is collected. The CRPs may be from a software PUF model, statistical circuit simulations or post-silicon measurements. The PUF responses are then fed to the NIST suite to obtain their scores for various tests.

Although we tested various PUF architectures, we will present the results from XOR arbiter PUFs in this section for the sake of brevity. For the analysis, we collected a large sample of responses (around 1 Million) from software PUF model of XOR arbiter PUF structure. The PUF model is based on linear additive delay model for arbiter PUFs and was obtained from TU Munich. The model generates n arbiter PUFs and obtains the XOR value of the n bit response for each challenge. If n is set to '1', the model generated is equivalent to a simple arbiter PUF structure.

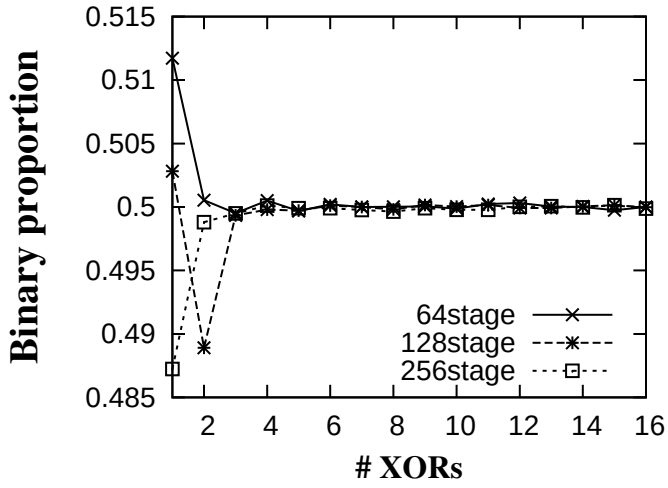


Figure 8.1. Proportion of 0's and 1's in XOR arbiter PUFs for Mersenne Twister based challenges

Varying PUF bit-lengths are used for security evaluation purposes. For this work, we evaluate up to $n = 16$. We use 64, 128 and 256 stage XOR arbiter PUFs for security evaluations using the benchmarks. For generating random challenges, we used a pseudorandom number generator (PRNG) known as Mersenne Twister (MT). Unless mentioned otherwise, the depicted results correspond to average results obtained from 5 random iterations.

One of the straightforward, albeit less effective way to evaluate randomness is measuring the number of 0's and 1's in the response streams. We evaluated the proportion of 0's and 1's in the XOR arbiter PUFs for MT based challenges and the results are shown in Figure 8.1. It can be observed that the PUF structure produces a fairly equal number of 0's and 1's across a large number of evaluations and varying XOR lengths. So, the randomness metric based on binary proportions does not yield any useful observation.

We therefore evaluated the responses for their NIST scores. NIST suite evaluates the bitstream across different tests and interested readers are referred to [7] for details on different tests. We collected the sum of different test scores and highly random

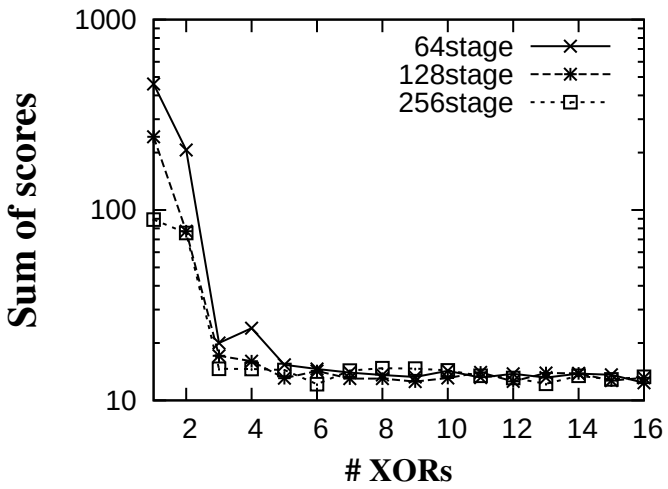


Figure 8.2. NIST scores for XOR arbiter PUFs for Mersenne Twister based challenges

stream produces the minimum score. The results from NIST suite based evaluation of XOR arbiter PUFs for MT based challenges are shown in Figure 8.2. Again we can infer that the results saturate beyond certain number of XORs and are not distinguishable for the given challenge generator. For example, 8- and 16-XOR arbiter PUFs have almost the same scores and the fine-grain randomness is not noticed by the NIST suite. One potential reason could be the randomness induced by the challenge generation mechanism itself. In such cases, the randomness induced by the challenge generator overrides the randomness in the PUF circuit. So, new methods of challenge generation have to be identified to enable fine-grain security evaluation of "strong" PUF circuits.

8.2.1 Impact of varying challenge generation methods

As the MT based challenges suppress the randomness of PUF circuits, it is essential to identify suitable challenge generation methods. The identification of suitable challenge generators with reduced entropy, but statistically random enough is an interesting research problem. Moreover, delay-based PUFs have an additional concern

that the maximum entropy is concentrated around the delay stages toward the arbiter. In other words, a single bit change in challenge's LSB bits have higher impact on the output bit when compared to a single bit change in the MSB bits. So, it is important to identify a suitable challenge generator with minimal change in LSB bits across the challenges. Upon investigating various statistically random generators, quasi-random (QR) sequences seemed to be promising enough. A quasi-random or low discrepancy sequence is "less random" than a pseudorandom number sequence, but more useful for tasks such as Monte Carlo applications and in global optimizations. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers. We used two popular QR sequences, namely Halton and Sobol. The Matlab implementations of Halton and Sobol (`haltonset` and `sobolset` respectively) support interesting options such as `scramble`, `leap` and `skip`. More details can be found in [67].

Initially, we conducted experiments using the `scramble` setting turned on for generating QR sequences. `Haltonset` supports reverse-radix scrambling and `sobolset` supports random linear scrambling. Similar to the experiments using MT generators, the XOR arbiter PUFs up to $n = 16$ were evaluated for binary proportions. The results for scrambled halton and sobol based challenge generators are shown in Figures 8.3(a) and 8.3(b) respectively. We can infer that the binary proportions do not yield any useful observations for the PUF's entropy similar to MT generators. So, we carried out experiments to evaluate the NIST sum of scores for XOR arbiter PUF responses obtained for QR sequences. The sum of scores against varying number of XORs for Halton and Sobol sequences with scrambling are shown in Figures 8.4(a) and 8.4(b) respectively. We can infer that the scores saturate beyond $n = 6$ and the scrambled sequences suppress the entropy of the PUF circuit. Upon careful observation, we observed that the scrambled generators still have considerable amount of bit flips towards LSB bits.

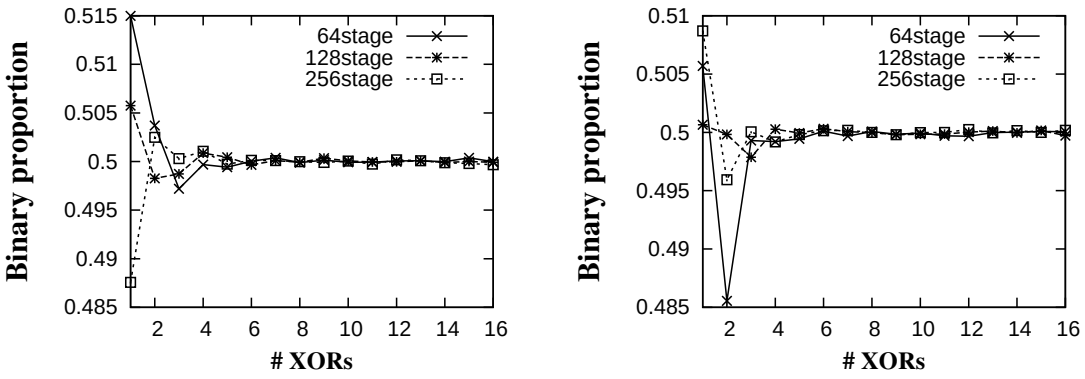


Figure 8.3. Proportion of 0's and 1's in XOR arbiter PUFs for Halton(left) and Sobol(right) based generators with scrambling

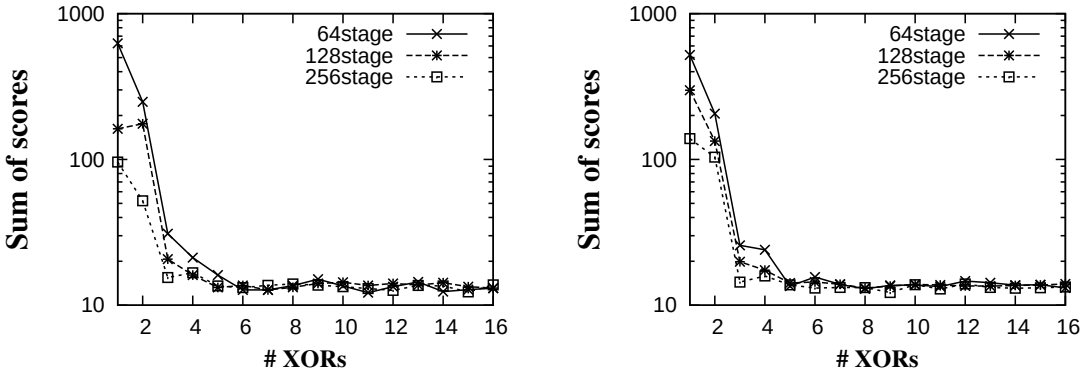


Figure 8.4. NIST sum of scores for XOR arbiter PUFs for Halton(left) and Sobol(right) based generators with scrambling

This motivated us to explore the impacts of using QR sequences without any scrambling applied during challenge generation. We set the `scramble` setting to 'off' and generated the challenges. Similar to previous experiments, we collected the binary proportions for the response streams and the results are shown in Figures 8.5(a) and 8.5(b). Similar effects to MT and scrambled QR sequences were observed with the binary proportions experiment.

As binary proportions experiment didn't yield positive results even without scrambling, we evaluated the responses obtained from QR unscrambled sequences using the NIST suite. The sum of scores against the varying number of XORs are shown in

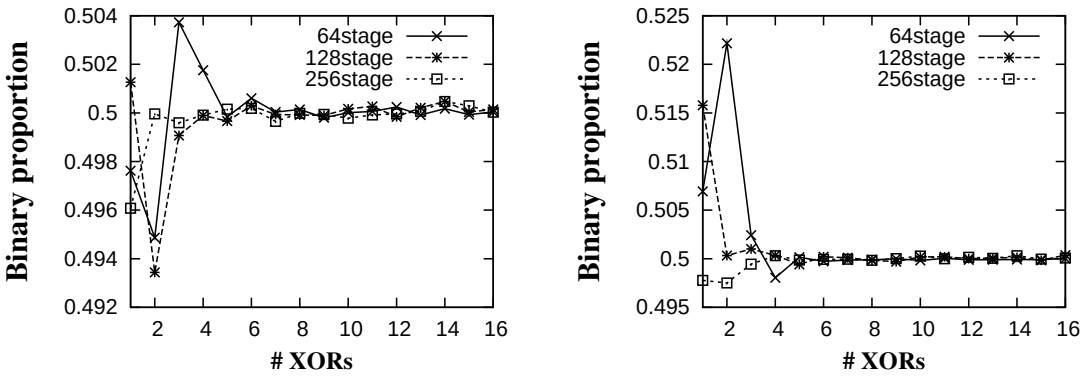


Figure 8.5. Proportion of 0's and 1's in XOR arbiter PUFs for Halton(left) and Sobol(right) based generators without scrambling

Figures 8.6(a) and 8.6(b) respectively. We can infer from the figures that Sobol unscrambled generator performs similar to the other generators explained above with the scores saturating beyond certain number of XORs. However, Halton based generator has the minimal number of bit-flips towards LSB bits and yields expected results as shown in Figure 8.6(a). The NIST based suite captures the entropy of the PUF circuit as the entropy of Halton unscrambled generator is very less. Moreover, the different versions of PUF circuits (64, 128 and 256 stages) and varying number of XORs can be clearly identified as seen in Figure 8.6(a). For example, increasing the number of XORs makes the response highly random as noted by decreasing sum of scores. The results agree well with the modeling attack results trend usually observed in PUFs. The modeling attack resistance increases exponentially with increasing number of XORs similar to the trend shown in Figure 8.6(a). This analysis shows that Halton unscrambled generator could be highly useful for predicting the modeling attack resistance of PUFs using simple NIST suite. The benchmark suite using Halton unscrambled generator also enables fine-grained security assessment such as the impacts of varying XORs, bit-lengths, etc.

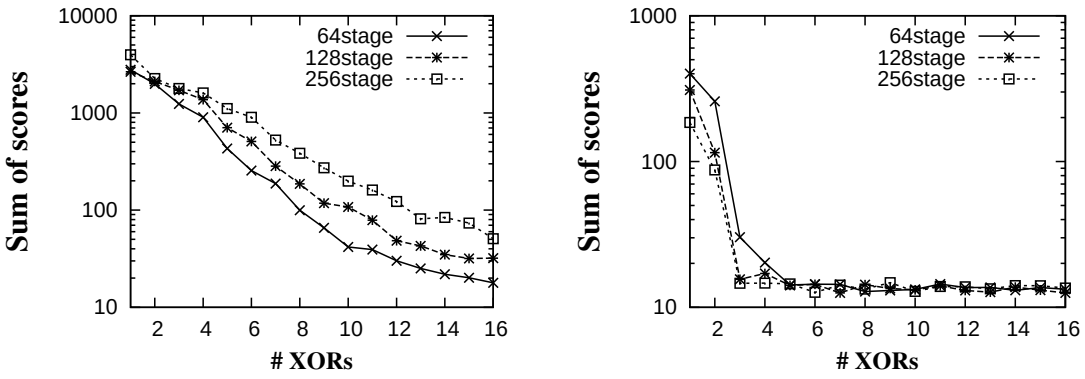


Figure 8.6. NIST sum of scores for XOR arbiter PUFs for Halton(left) and Sobol(right) based generators without scrambling

8.3 Response Compressibility Analysis

In addition to the benchmark suite analyses, we also evaluated the PUFs against response compression algorithms. If the outputs are highly random, the compression ratio should be lower and if the outputs are not random enough, the responses will be highly compressible. We used two compression algorithms with very high compression rations:

1. `7z` [2]: Lempel-Ziv-Markov chain algorithm (LZMA2) method of 7-zip compression algorithm

2. `AdvanceCOMP`: Huffman based compression algorithm. We refer to it as "Advcomp" further in the document.

Both the compression algorithms support varying levels of compression ranging from "nominal" to "extreme". Although we achieved similar result trends using both the settings, we present the results obtained using the "extreme" setting for the sake of brevity.

Initially we evaluated the XOR arbiter PUFs for data compressibilities using the response streams obtained from MT based challenge generator. The results are shown in Figures 8.7(a) and 8.7(b). We can infer that the compression ratios are almost

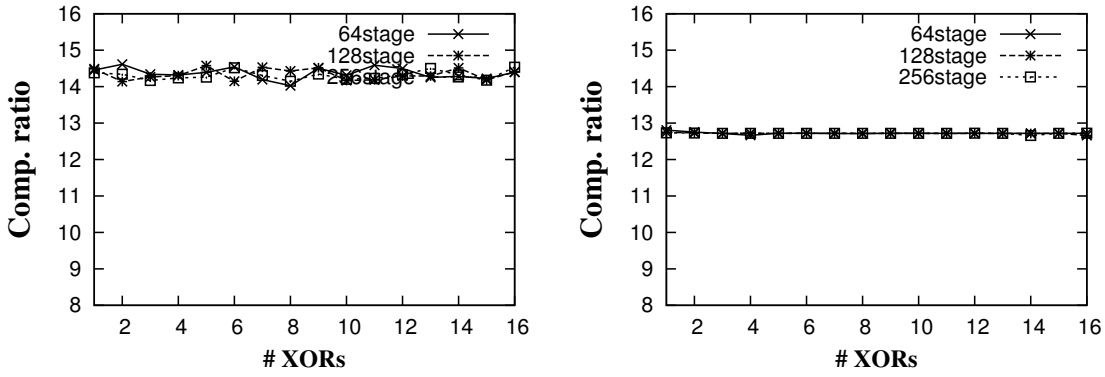


Figure 8.7. Compression ratios for MT generator based XOR arbiter PUFs using 7z(left) and Advcomp(right) algorithms

identical for varying number of XORs. This is due to the fact that the challenge generator's entropy dominates the entropy of the PUF circuit. As the challenge set is same for all the XOR PUF structures, the compression ratios are almost identical. These results agree well with the benchmark suite results obtained for MT based challenge generators.

As MT based challenge generators didn't yield satisfactory results, we evaluated the XOR arbiter PUFs for data compressibilities using QR sequences. We initially used scrambled QR generators to generate response bitstreams. The compression ratios of the response bitstreams from 7z and Advcomp algorithms are shown in Figure 8.8. We can infer that both the halton and sobol scrambled generators produce identical compression ratios against the varying number of XORs, thereby not providing an opportunity to perform fine-grained security assessments. Again, the results agree well with the benchmark suite results.

Similar to the benchmark suite experiments, a breakthrough was obtained using the unscrambled versions of QR generators. We set the `scramble` setting to "off" and generated the challenge streams. The response bitstreams obtained using the challenge streams were then evaluated against 7z and Advcomp algorithms. The compression ratios against varying number of XORs are shown in Figure 8.9. It can

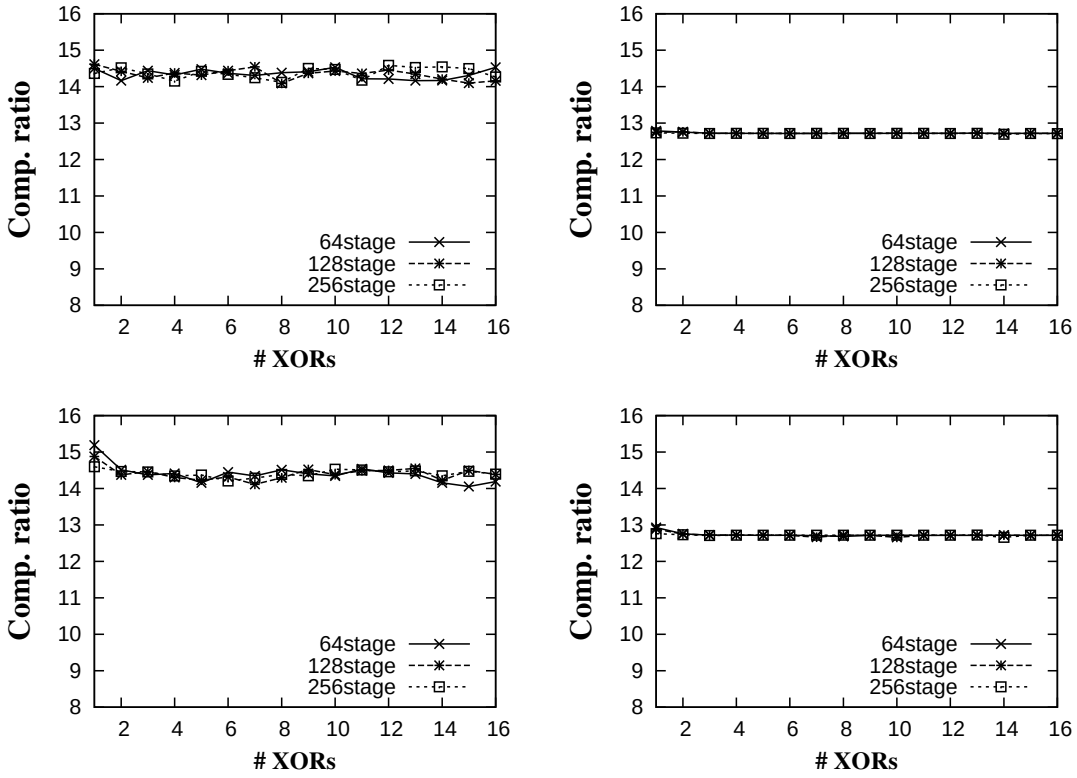


Figure 8.8. Compression ratios for scrambled halton(top) and sobol(bottom) generators based XOR arbiter PUFs using 7z(left) and Advcomp(right) algorithms

be inferred that the data compression algorithms were able to distinguish the varying number of XORs upto certain extent($n = 5$) for unscrambled QR based response streams. The compression ratios were higher for lower number of XORs and starts reducing with increasing number of XORs. This is obviously due to the fact that increasing number of XORs also increases the entropy of PUF responses. Even better results were observed for halton unscrambled generators as seen from Figure 8.9. The data compression algorithms were able to distinguish varying number of XORs and bitlengths to extents better than the unscrambled sobol generator. In particular, Advcomp yields significantly better results for unscrambled Halton generator than the sobol counterpart. The fine-grained security evaluations agree quite well with the benchmark suite results. However, the fine-grained security assessments are still not accurate enough when compared to the results obtained from benchmark suites. One

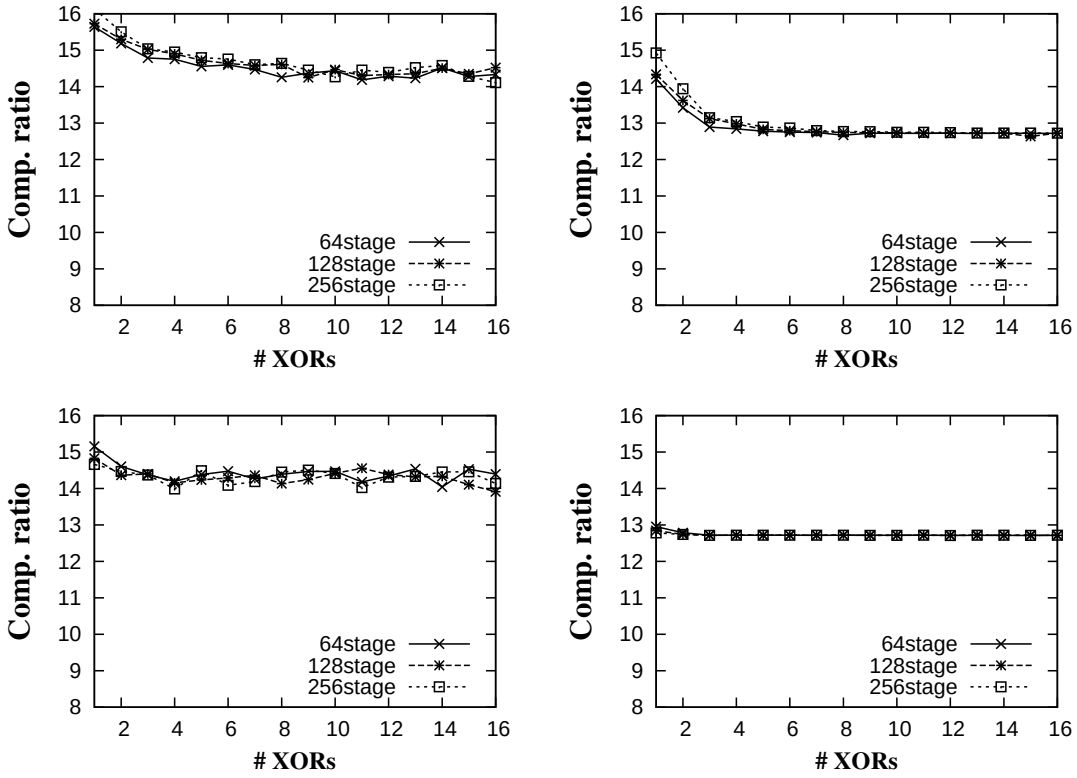


Figure 8.9. Compression ratios for unscrambled halton(top) and sobol(bottom) generators based XOR arbiter PUFs using 7z(left) and Advcomp(right) algorithms

potential reason could be the limited data compression resolutions of the compression algorithms employed. As part of our future work, we will be exploring several other data compression algorithms that enable highly accurate fine-grained security assessments.

In general, we were able to verify the authenticity of the NIST based benchmark suite using our analysis on data compressibilities. The results from compressibility analysis follow the same trend as the results from benchmark suite to certain extent. We were able to verify that the unscrambled Halton generator yields the optimum results to enable fine-grained security assessments of the XOR arbiter PUFs.

CHAPTER 9

CONCLUSION AND FUTURE WORK

Moore's law has continuously driven the semiconductor industry over the past few decades. The semiconductor technology has certainly had a major impact on significant improvements in various things ranging from day-to-day human lives to rocket-science. With the continuous improvements in wireless and mobile computing thanks to semiconductor technology and in turn Moore's law, people tend to rely on pocket-size devices for executing a gamut of day-to-day activities. Some of these activities often involve storage and processing of sensitive data, which puts security and privacy protection at forefront position. As the data is often transmitted over untrusted wireless medium, the devices are equipped with means to provide cryptographic protection to the transmitted data. As the devices are getting smaller in size and power with technology scaling, it has provided an excellent platform for advancements of lightweight cryptographic algorithms. However, the advancements in lightweight crypto- blocks face challenges from ever-increasing process variations and various forms of hardware and software level threats. In this work, we have demonstrated the vulnerabilities of lightweight cryptographic blocks to different forms of fault-injection attacks. In particular, we have presented novel hardware Trojans based techniques to inject transient faults into a cryptographic circuit. The Trojan insertion techniques are based on altering the voltage transfer characteristic (VTC) of the target gate through the manipulation of doping concentration and/or dopant area of a transistor. We demonstrated that the proposed fault-injection methods are suited for attacks on cryptographic circuits requiring high precision by mounting

attacks on state-of-art lightweight cryptographic block ciphers such as LED-64 and PRINCE. This holds for the rather simple attack on LED-64, which needs one exploitable fault in most cases, as well as for a more sophisticated two-stage attack on PRINCE where fault locations in both stages interfere with each other. Conventional tamper-resistance techniques, such as shielding or voltage-drop sensors, are largely ineffective against attacks presented in this work but may still be required to protect the circuit against other threats. It appears that effective countermeasures against parametric fault injection are on protocol level (frequent key regeneration) or information level (concurrent error detection). The MAPLE Trojans can substantially bias the fault injection towards the locations desired by the attacker, and they are nearly undetectable by all known test methods. In summary, our results suggest that the right mix of countermeasures on different abstraction levels is essential for comprehensive protection.

Although process variations are detrimental to a circuit's operation, PUFs have been proposed as a viable solution to harness the unpredictable nature of process variations for security applications. Initially, PUFs were assumed to be inherently tolerant to various attacks and threats. To analyze the vulnerabilities of PUF circuits, we designed and fabricated different PUF architectures such as arbiter, feed-forward arbiter, current-based PUFs in 32nm SOI technology available from IBM. The various modeling attacks mounted on these different PUF architectures suggest that the existing PUFs are highly vulnerable to machine learning based modeling attacks. The prediction accuracies for these PUF circuits reach more than 98% for a limited set of training CRPs. However, the performance of modeling attacks are highly impacted by the presence of error-prone CRPs in the learning and training set CRPs. To that extent, we have proposed a technique that exploits data-dependent information from the PUF target and use it in conjunction with a machine learning algorithm in order to improve the performance of modeling attack. The data-dependent information is

extracted through the use of a fault-injection attack on the PUF target. We demonstrated the effectiveness of the technique by mounting hybrid attacks on all the PUF targets mentioned above. The vulnerabilities of existing PUF circuits impose a strong pressing need on design of efficient and secure PUF designs. To that extent, we presented the design and analysis of a modeling attack resistant PUF design based on non-linear circuit elements. The circuit relies on current switching using non-linear current mirrors, where the amount of current switch is dependent on the challenge bit and the input current itself. The post-silicon validation of the circuit using IBM 32nm SOI process indicates that the PUF circuit has excellent statistical and security properties. The non-linear PUF exhibits around 10x and 20-25x improvements in modeling and hybrid attack tolerances respectively. Moreover, the silicon area numbers of the proposed PUF are at par with other strong PUF architectures.

Apart from security levels, there are other important performance metrics to be satisfied for commercial deployments. This thesis explored techniques to improve the performance of PUFs from various circuit and system level perspectives. Fabrication aware design technique that exploits forbidden pitches in sub-wavelength lithography has been proposed to improve the performance of PUF. The proposed design framework is highly generic and can be applied with minimal effort to any silicon based PUF implementation that relies on process variations. We also presented a system level technique/protocol for enhancing the performance of PUF based systems by extending the concept of trusted user and device authentication. Finally, we presented a statistical method to evaluate PUF circuits and also to predict the security levels of strong PUF architecture during the conceptualization phase. As mounting modeling attacks involve time and effort-consuming parametric modeling and deep knowledge of machine learning algorithms, the process of conceptualizing new PUF architectures is often cumbersome. However, the proposed benchmark suite builds upon the existing NIST test suite and requires a basic parametric model without ad-

ditional properties to be satisfied unlike the parametric models involved in machine learning based attacks. We also presented data compression analyses to validate the performance of statistical benchmark suite against fine-grained security assessments.

In general, secure computation in nanometer CMOS regime is highly complicated and the desire to build efficient cryptographic systems will be hard to quench. The ever-increasing process variations provide both an excellent platform and also a stumble block for varying levels of hardware security applications. Further, the evolving styles of computation logic present an excellent opportunity to identify different types of implementation strategies for hardware security blocks in order to address various forms of possible threats. Overall, the field of hardware security in nanometer CMOS promise an exciting future for commercial and research fields.

9.1 Future Work

This dissertation has touched the fields of hardware Trojans, fault-injection attacks, modeling attacks and PUFs and the common observation is that the results can be extended in the future. Some of the possible avenues for extension are provided in this section.

- **Hardware Trojans:** This work touched upon manipulation of manufacturing process parameters to insert hardware Trojans. As the Trojan gates are identical to normal gates in metal and polysilicon layers, it is extremely hard to detect them through optical inspection. Recent work shows that dopant-area manipulation can be detected through scanning electron microscopy, albeit at an increased cost when compared to metal layer detection [85]. However, no existing work shows that doping concentration based manipulation can be detected through optical inspection. The identification of detection strategies and possible countermeasures for manufacturing process Trojans is extremely im-

portant for secure computation logic. Design of cross-level protective schemes which balance security against efficiency and cost is a challenge for the future.

- **PUFs:** The last decade or so has seen an enormous amount of work put forth to make and break PUF circuits. Majority of the work are based on identifying suitable parametric models for analysis and attack purposes. Due to cost and effort constraints, majority of the work report results from simulations instead of post-silicon measurements. In such cases, extreme care must be taken to identify suitable process variation models to enable accurate characterization of PUFs. Although process variations are random enough, the presence of spatial correlations degrade the performance of PUFs. So, the process variation models should be built to incorporate the correlation component. This also paves opportunity to exploit various lithographic sources of variations in order to suppress the correlation component in process variations. This work touched upon the concept of forbidden pitches to suppress spatial correlations in process variations. However, there exist a wide-range of opportunities in sub-wavelength lithography that can be exploited for improving the performance of PUF circuits. Moreover, the most common technique to improve the reliability of PUF circuits is to employ error-correction codes (ECC). Depending on the amount of unreliability in PUFs, the size of ECC may increase and pose additional constraints for commercial deployments. As reliability of a PUF is directly related to security levels, novel techniques from circuit- and system-level perspectives are needed to improve the reliability of PUFs in order to reduce the workload on ECCs. The proposed hybrid attacks that exploit unreliable CRPs also indicate that reliability issues in PUFs should be addressed in the near future. Finally, various forms of attacks indicate that novel and efficient "strong" PUF architectures that are inherently tolerant to modeling attacks present a challenge to the future.

- **Benchmarking PUFs:** This work presented a brief description and analysis of extending NIST suite to develop benchmarks for early-stage analysis of PUF architectures. While QR sequences seem appropriate for statistical benchmarks, the results indicate that a comprehensive analysis of the impacts of various statistically random sequences on NIST based benchmark suite are needed. Although an architecture independent sequence would reduce the time and effort, it is important to identify the different sensitivity levels of the PUF target to varying random sequences. This would enable the designers to perform security assessments at extreme fine-grained levels. The identification of suitable data compression algorithms for performing fine-grained security assessments is also an exciting venue for research in PUF benchmarking field.

BIBLIOGRAPHY

[1] International Technology Roadmap for Semiconductor (ITRS), 2006.

[2] 7-zip. 7z format. `http://www.7-zip.org/7z.html`.

[3] Antoni, L., Leveugle, R., and Feher, B. Using Run-time Reconfiguration for Fault Injection in Hardware Prototypes. In *IEEE Defect and Fault Tolerance in VLSI Systems* (2000), pp. 405–413.

[4] Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., and Whelan, C. The Sorcerers Apprentice Guide to Fault Attacks. *Proceedings of the IEEE 94*, 2 (2006), 370–382.

[5] Barenghi, A., Bertoni, G., Parrinello, E., and Pelosi, G. Low Voltage Fault Attacks on the RSA Cryptosystem. In *Fault Diagnosis and Tolerance in Cryptography – FDTC 2009. 6th Workshop on* (2009), pp. 23–31.

[6] Barenghi, A., Breveglieri, L., Koren, I., and Naccache, D. Fault Injection Attacks on Cryptographic Devices: Theory, Practice and Countermeasures. *Proceedings of the IEEE 100*, 11 (2012), 3056–3076.

[7] Bassham, III, Lawrence E., Rukhin, Andrew L., Soto, Juan, Nechvatal, James R., Smid, Miles E., Barker, Elaine B., Leigh, Stefan D., Levenson, Mark, Vangel, Mark, Banks, David L., Heckert, Nathanael Alan, Dray, James F., and Vo, San. Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010.

[8] Becker, G., and Kumar, R. Active and passive side-channel attacks on delay based PUF designs. Cryptology ePrint Archive, Report 2014/287, 2014.

[9] Becker, G.T., Regazzoni, F., Paar, C., and Burleson, W. Stealthy Dopant-Level Hardware Trojans. In *Cryptographic Hardware and Embedded Systems - CHES 2013*, vol. 8086 of *LNCS*. Springer Berlin Heidelberg, 2013, pp. 197–214.

[10] Biham, E., and Shamir, A. Differential Fault Analysis of Secret Key Cryptosystems. In *Annual Int'l Cryptology Conf.* (1997), vol. 1294 of *LNCS*, pp. 513–525.

[11] Blaauw, D., Chopra, K., Srivastava, A., and Scheffer, L. Statistical timing analysis: From basic principles to state of the art. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27*, 4 (2008), 589–607.

[12] Boneh, D., DeMillo, R.A., and Lipton, R.J. On the Importance of Elimination Errors in Cryptographic Computations. *Journal of Cryptology* (2001), pp. 101–119.

[13] Borghoff, J., et al. PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications. In *Advances in Cryptology – ASIACRYPT 2012* (2012), Xiaoyun Wang and Kazue Sako, Eds., vol. 7658 of *LNCS*, Springer Berlin Heidelberg, pp. 208–225.

[14] Bösch, C., Guajardo, J., Sadeghi, A-R., Shokrollahi, J., and Tuyls, P. Efficient helper data key extractor on FPGAs. In *Cryptographic Hardware and Embedded Systems CHES 2008*, vol. 5154 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 181–197.

[15] Canivet, G., Maistri, P., Leveugle, R., Clediere, J., Valette, F., and Renaudin, M. Glitch and Laser Fault Attacks onto a Secure AES Implementation on a SRAM-Based FPGA. *Journal of Cryptology 24*, 2 (2011), 247–268.

[16] Cha, B., and Gupta, S.K. Trojan Detection Via Delay Measurements: A New Approach to Select Paths and Vectors to Maximize Effectiveness and Minimize Cost. In *Design Automation and Test in Europe* (2013), pp. 1265–1270.

[17] Delvaux, J., and Verbauwhede, I. Fault injection modeling attacks on 65nm arbiter and RO sum PUFs via environmental changes. Cryptology ePrint Archive, Report 2013/619, 2013.

[18] Delvaux, J., and Verbauwhede, I. Side channel modeling attacks on 65nm arbiter PUFs exploiting cmos device noise. In *Hardware-Oriented Security and Trust (HOST)* (June 2013), pp. 137–142.

[19] Dodis, Y., Reyzin, L., and Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 523–540.

[20] Engelke, P., Polian, I., Renovell, M., Kundu, S., Seshadri, B., and Becker, B. On Detection of Resistive Bridging Defects by Low-Temperature and Low-Voltage Testing. *IEEE Trans. on CAD 27*, 2 (February 2008), 327–338.

[21] et al., Shiyanovskii. Process Reliability Based Trojans Through NBTI and HCI Effects. In *NASA/ESA Conf. Adaptive Hardware and Systems* (2010), pp. 215–222.

[22] Farrar, N.R., Smith, A.H., Busath, D.R., and Taitano, D. In-situ measurement of lens aberrations. *Proc. of SPIE 4000* (2000), 18–29.

[23] Forte, D., and Srivastava, A. Manipulating manufacturing variations for better silicon-based physically unclonable functions. In *Proc. IEEE Computer Society Annual Symposium on VLSI* (Aug. 2012), pp. 171 –176.

[24] Forte, D., and Srivastava, A. On improving the uniqueness of silicon-based physically unclonable functions via optical proximity correction. In *IEEE/ACM Design Automation Conference* (june 2012), pp. 96 –105.

[25] Frikken, K.B., Blanton, M., and Atallah, M.J. Robust authentication using physically unclonable functions. In *Proceedings of the 12th International Conference on Information Security* (Berlin, Heidelberg, 2009), ISC '09, Springer-Verlag, pp. 262–277.

[26] Guo, J., Peyrin, T., Poschmann, A., and Robshaw, M. The LED Block Cipher. *LNCS 6917* (2011), pp. 326–341.

[27] Hearst, M. Support vector machines. In *IEEE Intelligent Systems* (1998), pp. 18–28.

[28] Heather, J., Lowe, G., and Schneider, S. How to prevent type flaw attacks on security protocols. *J. Comput. Secur. 11*, 2 (Mar. 2003), 217–244.

[29] Hojsik, M., and Rudolf, B. Differential Fault Analysis of Trivium. In *Int'l Workshop on Fast Software Encryption* (2008), vol. 5086 of *LNCS*, pp. 158–172.

[30] Holcomb, D.E., Burleson, W., and Fu, K. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security* (2007).

[31] Holcomb, D.E., and Fu, K. Bitline PUF: Building native challenge-response PUF capability into any SRAM. In *Cryptographic Hardware and Embedded Systems CHES 2014*, Lejla Batina and Matthew Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014, pp. 510–526.

[32] Hospodar, G., Maes, R., and Verbauwhede, I. Machine learning attacks on 65nm arbiter PUFs: Accurate modeling poses strict bounds on usability. In *IEEE International Workshop on Information Forensics and Security* (2012).

[33] International, SypherMedia. Circuit Camouflage Technology - SMI IP Protection and Anti-Tamper Technologies. White Paper Version 1.9.8j, 2012.

[34] Jovanovic, P., Kreuzer, M., and Polian, I. A Fault Attack on the LED Block Cipher. In *COSADE* (2012), W. Schindler and S.A. Huss, Eds., vol. 7275 of *LNCS*, Springer Berlin Heidelberg, pp. 120–134.

[35] Jovanovic, Philipp, Kreuzer, Martin, and Polian, Ilia. Multi-Stage Fault Attacks on Block Ciphers. Cryptology ePrint Archive, Report 2013/778, 2013.

[36] Kerr, S., Kirkpatrick, M.S., and Bertino, E. PEAR: a hardware based protocol authentication system. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS* (New York, NY, USA, 2010), SPRINGL '10, ACM, pp. 18–25.

[37] Kim, C.H., and Quisquater, J.-J. Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures. *LNCS 4462* (2007), pp. 215–228.

[38] Kim, D., Cho, Choongyeun, Kim, Jonghae, Plouchart, J.-O., Trzcinski, R., and Ahlgren, D. CMOS mixed-signal circuit process variation sensitivity characterization for yield improvement. In *Proc. IEEE Custom Integrated Circuits Conference* (Sept. 2006), pp. 365 –368.

[39] Kim, I., Maiti, A., Nazhandali, L., Schaumont, P., Vivekraja, V., and Zhang, H. From statistics to circuits: Foundations for future physical unclonable functions. In *Towards Hardware-Intrinsic Security*, Ahmad-Reza Sadeghi and David Naccache, Eds., Information Security and Cryptography. Springer Berlin Heidelberg, 2010, pp. 55–78.

[40] Kocher, P., Jaffe, J., and Jun, B. Differential Power Analysis. In *Annual Int'l Cryptology Conf.* (1999), vol. 1666 of *LNCS*, pp. 388–397.

[41] Kumar, R., and Burleson, W. PHAP: Password based hardware authentication using pufs. In *Microarchitecture Workshops (MICROW), 2012 45th Annual IEEE/ACM International Symposium on* (Dec 2012), pp. 24–31.

[42] Kumar, R., and Burleson, W. Litho-aware and low power design of a secure current-based physically unclonable function. In *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on* (Sept 2013), pp. 402–407.

[43] Kumar, R., and Burleson, W. Hybrid modeling attacks on current-based PUFs. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on* (Oct 2014), pp. 493–496.

[44] Kumar, R., and Burleson, W. On design of a highly secure PUF based on nonlinear current mirrors. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on* (May 2014), pp. 38–43.

[45] Kumar, R., Chandrikakutty, H.K., and Kundu, S. On improving reliability of delay based physically unclonable functions under temperature variations. In *Hardware-Oriented Security and Trust* (June 2011), pp. 142–147.

[46] Kumar, R., Dhanuskodi, S.N., and Kundu, S. On manufacturing aware physical design to improve the uniqueness of silicon-based physically unclonable functions. In *VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on* (Jan 2014), pp. 381–386.

[47] Kumar, R., Jovanovic, P., Burleson, W., and Polian, I. Parametric trojans for fault-injection attacks on cryptographic hardware. Cryptology ePrint Archive, Report 2014/783, 2014.

[48] Kumar, R., Jovanovic, P., and Polian, I. Precise fault-injections using voltage and temperature manipulation for differential cryptanalysis. In *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International* (July 2014), pp. 43–48.

[49] Kumar, R., Patil, V.C., and Kundu, S. On design of temperature invariant physically unclonable functions based on ring oscillators. In *VLSI (ISVLSI), 2012 IEEE Computer Society Annual Symposium on* (Aug 2012), pp. 165–170.

[50] Kundu, S., Sreedhar, A., and Sanyal, A. Forbidden pitches in sub-wavelength lithography and their implications on design. *Journal of Computer-Aided Materials Design 14* (Apr. 2007), 79–89.

[51] Lim, D. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2004.

[52] Lim, D., Lee, J.W., Gassend, B., Suh, E., van Dijk, M., and Devadas, S. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13*, 10 (oct. 2005), 1200 –1205.

[53] Lin, L., Holcomb, D.E., Krishnappa, D.K., Shabadi, P., and Burleson, W. Low-power sub-threshold design of secure physical unclonable functions. In *ACM/IEEE International Symposium on Low-Power Electronics and Design* (aug. 2010), pp. 43 –48.

[54] Lin, L., Srivathsa, S., Krishnappa, D.K., Shabadi, P., and Burleson, W. Design and validation of arbiter-based PUFs for sub-45-nm low-power security applications. *IEEE Transactions on Information Forensics and Security 7*, 4 (aug. 2012), 1394 –1403.

[55] Lopez-Ongil, C., Garcia-Valderas, M., Portela-Garcia, M., and Entrena, L. Autonomous Fault Emulation: A New FPGA-Based Acceleration System for Hardness Evaluation. *IEEE Transactions on Nuclear Science 54*, 1 (2007), 252–261.

[56] M. Tunstall, D. Mukhopadhyay, and S. Ali. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication* (2011), vol. 6633 of *LNCS*, pp. 224–233.

[57] Mack, C. *Fundamental Principles of Optical Lithography: The Science of Microfabrication.* Wiley-Interscience, 2007.

[58] Maes, R., and Verbauwhede, I. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, Ahmad-Reza Sadeghi and David Naccache, Eds., Information Security and Cryptography. Springer Berlin Heidelberg, 2010, pp. 3–37.

[59] Mahmoud, A., Rührmair, U., Majzoobi, M., and Koushanfar, F. Combined modeling and side channel attacks on strong PUFs. Cryptology ePrint Archive, Report 2013/632, 2013.

[60] Maiti, A., Gunreddy, V., and Schaumont, P. A systematic method to evaluate and compare the performance of physical unclonable functions. In *Embedded Systems Design with FPGAs*. Springer New York, 2013, pp. 245–267.

[61] Majzoobi, M., Ghiaasi, Golsa, Koushanfar, F, and Nassif, S.R. Ultra-low power current-based PUF. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on* (May 2011), pp. 2071–2074.

[62] Majzoobi, M., Koushanfar, F., and Potkonjak, M. Testing techniques for hardware security. In *IEEE International Test Conference* (oct. 2008), pp. 1 –10.

[63] Mangard, S., Oswald, E., and Popp, T. *Power Analysis Attacks Revealing the Secrets of Smartcards*. Springer, 2007.

[64] Mangasarian, O. L., and Musicant, David R. Lagrangian support vector machines. *J. Machine Learning Research 1* (Sept. 2001), 161–177.

[65] Mangasarian, O.L., and Musicant, D. R. LSVM Software: active set support vector machine classification software, 2000. www.cs.wisc.edu/∼musicant/lsvm/.

[66] Mathew, S.K., Satpathy, S.K., Anders, M.A., Kaul, H., Hsu, S.K., Agarwal, A., Chen, G.K., Parker, R.J., Krishnamurthy, R.K., and De, V. 16.2 a 0.19pj/b pvt-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International* (Feb 2014), pp. 278–279.

[67] Mathworks. Generating quasi-random numbers. `http://www.mathworks.com/help/stats/generating-quasi-random-numbers.html`.

[68] Moore's, Law. Wikipedia. `http://en.wikipedia.org/wiki/Moore\%27s_law`.

[69] Narasinham, S., and Bhunia, S. Hardware Trojan Detection. In *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. Springer, 2012, pp. 339–364.

[70] Needham, R.M., and Schroeder, M.D. Using encryption for authentication in large networks of computers. *Commun. ACM 21*, 12 (Dec. 1978), 993–999.

[71] Otway, D., and Rees, O. Efficient and timely mutual authentication. *SIGOPS Oper. Syst. Rev. 21*, 1 (Jan. 1987), 8–10.

[72] Öztürk, Erdinç, Hammouri, Ghaith, and Sunar, Berk. Towards robust low cost authentication for pervasive devices. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications* (Washington, DC, USA, 2008), PERCOM '08, IEEE Computer Society, pp. 170–178.

[73] Pappu, R.S. *Physical one-way functions.* PhD thesis, Massachusetts Institute of Technology, March 2001.

[74] Polian, I. Power Supply Noise: Causes, Effects, and Testing. *ASP Jour. Low-Power Electronics 6*, 2 (2010), 326–338.

[75] Ronse, K., Jansen, P., Gronheid, R., Hendrickx, E., Maenhoudt, M., Goethals, M., and Vandenberghe, G. Lithography options for the 32nm half pitch node and beyond. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE* (sept. 2008), pp. 371 –378.

[76] Rührmair, U. Oblivious transfer based on physical unclonable functions. In *Trust and Trustworthy Computing*, vol. 6101. Springer Berlin Heidelberg, 2010, pp. 430–440.

[77] Rührmair, U. SIMPL systems, or: can we design cryptographic hardware without secret key information? In *Proceedings of the 37th international conference on Current trends in theory and practice of computer science* (Berlin, Heidelberg, 2011), SOFSEM'11, Springer-Verlag, pp. 26–45.

[78] Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., and Schmidhuber, J. Modeling attacks on physical unclonable functions. In *ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 237–249.

[79] Rührmair, U., Sölter, J., Sehnke, F., Xu, Xiaolin, Mahmoud, A., Stoyanova, V., Dror, G., Schmidhuber, J., Burleson, W., and Devadas, S. PUF modeling attacks on simulated and silicon data. *Information Forensics and Security, IEEE Transactions on 8*, 11 (Nov 2013), 1876–1891.

[80] Satpathy, S., Mathew, S., Li, Jiangtao, Koeberl, P., Anders, M., Kaul, H., Chen, G., Agarwal, A., Hsu, S., and Krishnamurthy, R. 13fj/bit probing-resilient 250k PUF array with soft darkbit masking for 1.94% bit-error in 22nm tri-gate CMOS. In *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014 - 40th* (Sept 2014), pp. 239–242.

[81] Schmidt, J., and Herbst, C. A Practical Fault Attack on Square and Multiply. In *Fault Diagnosis and Tolerance in Cryptography – FDTC 2008. 5th Workshop on* (2008), pp. 53–58.

[82] Selmane, N., Guilley, S., and Danger, J.-L. Practical Setup Time Violation Attacks on AES. In *Dependable Computing Conference* (2008), pp. 91–96.

[83] Sreedhar, A., and Kundu, S. On modeling impact of sub-wavelength lithography on transistors. In *Proc. International Conference on Computer Design* (Oct. 2007), pp. 84 –90.

[84] Sreedhar, A., and Kundu, S. Physically unclonable functions for embeded security based on lithographic variation. In *Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2011), pp. 1 –6.

[85] Sugawara, T., Suzuki, D., Fujii, R., Tawa, S., Hori, R., Shiozaki, M., and Fujino, T. Reversing stealthy dopant-level circuits. In *Cryptographic Hardware and Embedded Systems CHES 2014*, Lejla Batina and Matthew Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014, pp. 112–126.

[86] Suh, G.E., and Devadas, S. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference* (New York, NY, USA, 2007), DAC '07, ACM, pp. 9–14.

[87] Tehranipoor, M, and Koushanfar, F. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computers 27*, 1 (2010), 10–25.

[88] Verbauwhede, I., and Maes, R. Physically unclonable functions: manufacturing variability as an unclonable device identifier. In *ACM Great Lakes Symposium on VLSI'11* (2011), pp. 455–460.

[89] Vivekraja, V., and Nazhandali, L. Feedback based supply voltage control for temperature variation tolerant PUFs. In *VLSI Design* (Jan 2011), pp. 214–219.

[90] Wang, W., Reddy, V., Yang, B., Balakrishnan, V., Krishnan, S., and Cao, Y. Statistical prediction of circuit aging under process variations. In *IEEE Custom Integrated Circuits Conference* (2008), pp. 13–16.

[91] Wilamowski, B.M, Ferre-Pikal, E.S, and Kaynak, O. Low power, current mode CMOS circuits for synthesis of arbitrary nonlinear functions. In *Proc. NASA Symposium on VLSI Design* (2000).

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

[93] Zussa, L., et al. Efficiency of a Glitch Detector Against Electromagnetic Fault Injection. In *Design Automation and Test in Europe* (2014).