2009-06-02

Cryptographic Primitives from Physical Variables

Ghaith Hammouri
Worcester Polytechnic Institute

Cryptographic Primitives from Physical Variables

by
Ghaith Hammouri

A Dissertation
Submitted to the Faculty
of the

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the
Degree of Doctor of Philosophy
in
Computer Engineering

May, 2009

Approved:

Prof. Berk Sunar ECE Department Dissertation Advisor	Prof. Christof Paar ECE Department Ruhr-University of Bochum
Prof. William J. Martin Department of Mathematical Sciences	Prof. Daniel J. Dougherty Department of Computer Science
Prof. David Cyganski ECE Department	Prof. Fred J. Looft ECE Department Head

For my parents,
my wife
and my son.

I love you all.

Abstract

In this dissertation we explore a new paradigm emerging from the subtleties of cryptographic implementations and relating to theoretical aspects of cryptography. This new paradigm, namely physical variables (PVs), simply describes properties of physical objects designed to be identical but are not due to manufacturing variability.

In the first part of this dissertation, we focus our attention on scenarios which require the unique identification of physical objects and we show how Gaussian PVs can be used to fulfill such a requirement. Using this framework we present and analyze a new technique for fingerprinting compact discs (CDs) using the manufacturing variability found in the length of the CDs' lands and pits. Although the variability measured is on the order of 20 nm, the technique does not require the use of microscopes or any advanced equipment. Instead, the electrical signal produced by the photo-detector inside the CD reader will be sufficient to measure the desired variability. We thoroughly investigate the new technique by analyzing data collected from 100 identical CDs and show how to extract a unique fingerprint for each CD.

In the second part, we shift our attention to physically parameterized functions (PPFs). Although all the constructions we provide are centered around delay-based physically unclonable functions (PUFs), we stress that the use of the term PUF could be misleading as most circuits labeled with the term PUF are in reality clonable on the protocol level. We argue that using a term like PPFs to describe functions parameterized by a PV is a more accurate description. Herein, we thoroughly analyze delay-PUFs and use a mathematical framework to construct two authentication protocols labeled PUF-HB and HB+PUF. Both these protocols merge the known HB authentication family with delay-based PUFs. The new protocols enjoy the security reduction put forth by the HB portion of the protocol and at the same time maintain a level of hardware security provided by the use of PUFs. We present a proof of concept implementation for HB+PUF which takes advantage of the PUF circuit in order to produce the random bits typically needed for an HB-based authentication scheme. The overall circuit is shown to occupy a few thousand gates. Finally, we present a new authentication protocol that uses 2-level PUF circuits and enables a security reduction

which, unlike the previous two protocols, stems naturally from the usage of PVs.

Acknowledgments

First and foremost, I want to express my gratitude towards my Lord (Allah). I praise Him for His greatness and His infinite mercy. Where I stand in life today is a mere example of the countless bounties He has bestowed upon me. All I can say is "AlHamdulilLah".

The fact that this dissertation is finally being submitted is largely due to the support of my advisor Professor Berk Sunar. After spending the past few years with him, I can easily describe Professor Sunar as the best advisor a student can wish for. On a personal level, it even feels strange to call him my advisor for I have truly grown to view him as a close friend and an elder brother. The way I view research and write papers today has mostly been shaped by his continuous advice. My appreciation toward and for Professor Sunar is beyond my ability to describe. I can only hope that I have lived up to his expectations and continue to do so in the future.

I would also like to thank the members of my committee for agreeing to be part of this process. I really appreciate the time they took to read, discuss and significantly improve my dissertation. On a personal note each of the four members on my committee has had a notable impact on my research career. Professor William Martin defines my understanding of what it means to be rigorous. I hope that one day I will be able to impress him. Professor Dan Daugherty was one of the first people to methodically introduce me to theoretical computer science. I can only say that he is the reason I have developed a love and a passion for this topic. Professor David Cyganski helped me on one of the earliest research problems I worked on. He spent a considerable amount of time listening to my farfetched solutions before shooting them down. His ability to merge theory and experiment is quite inspiring. Professor Christof Paar defines working on cryptography as an engineer. His numerous papers in the field are probably one of the main reasons I find myself doing research in cryptography.

A considerable amount of my time at WPI was funded through being a TA. I thank Professor Fred Looft, the ECE-department captain for keeping me on board for all these years. The rest of my PhD years were funded by three generous NSF grants ANI-0133297, CNS-0831416 and CNS-0716306.

I thank the staff at the ECE office, Cathy Emmerton, Brenda McDonald, and Colleen Sweeney for their constant smile and for the infinite supply of chocolate they provided

us with. I also want to thank my lab mates and coauthors whom I prefer to call my academic siblings: Jens-Peter Kaps, Gunnar Gaubtz, Selçuk Baktır, Erdinç Öztürk, Deniz Karakoyunlu, Kahraman Akdemir, Berk Birand, Yin Hu and Chenguang Yang. I specifically want to thank Gunnar Gaubatz for allowing me to use his LaTex dissertation file to write mine.

The opportunity to come to this country would not have been a reality without the people at the University of Hartford. President Walter Harrison created the King Hussein scholarship which I was the first recipient of. I thank him for that and for his warm hospitality and for all his help during all my years at Hartford. A lot of what I have learned is due to several professors at the University of Hartford. I would like to thank Professors: Hisham Al-Najjar, Jonathan Hill, Jim McDonald, Virginia Noonburg and Professor Raymond McGivney for reminding me that I should follow "My Bless".

Going a little bit further back in time, my high school years at the Jubilee school greatly shaped my educational path. Muhammad Islam, my math teacher there, was one of the first people to ignite my excitement for mathematics. I can never forget the sentence at the top of all his tests, "Exam is a tool of learning." Indeed, his tests have taught me a lot and as a result I find it quite proper to thank him here.

I also want to thank all my loved ones. The WIC community, I thank them for being a family to me during my Worcester years. Mazen Ramadan, I thank him for being a true friend over the past few years. I specially thank him for proof reading parts of this dissertation and for always being there when I needed help. Ahmed, Sameh, Jareer, Zaid, Muneeb, the Raja'ees and Salman, I thank them for a friendship which has spanned miles and lived for years. My siblings, Raghad, Tariq, Qusai and their families, I thank them for their unconditional love.

Last but certainly not least I want to thank my parents, my wife and my son.

My father Mohammad has always believed in me even when I did not believe in my own self. Who I am today is a reflection of what he has taught me throughout my life. I look up to him today more than I ever have before. Any accomplishment I achieve can only become meaningful through the hope that it might make him proud of me. "I hope that who I am today is who you wanted me to be."

My mother Najah is the source of light in my life. Hearing her voice and feeling her love

has kept me going over the past 10 years living away from home. I long for the time when I can see her everyday. I truly do not know how to ever repay her, for whatever I do I know that her love will be endless. "You raised me to be who I am and to you I owe myself. I wish to one day make up for any loneliness you have ever felt."

My son Muhammad is the constant of my life. No matter what happens in my day, his smile always reminds me that he means the world to me. I hope that I will one day be as good of a father to him as my father has been to me.

My wife Reem is my soul mate. She is my best friend, my love and my family. Her patience with my forgetfulness and my busy schedule during the past few years is beyond description. As each day passes, I realize how fortunate I am to have her in my life and I realize how much I need her to be in my life. "Everything I achieve has been really achieved by you. I love you today more than you will ever know."

Worcester, MA, June 2009

Ghaith Hammouri

Contents

List of Tables

List of Figures

Chapter 1

Why Physical Variables?

Over the past few years an increasing number of results have targeted the use of physical variables in cryptography. These variables are in essence very similar to biometric modalities in humans. While biometric modalities are typically used to uniquely identify specific individuals, physical variables can be used to uniquely identify physical objects such as circuits, compact discs (CDs), papers, etc. In this work, the term physical variables (PVs) refers to variables which describe certain properties of a physical object. Due to manufacturing variability, even identical objects are expected to have a slightly different value for their PV. A simple example of a PV is the length of wires inside an integrated circuit. Even though identical circuits are designed to have the same exact wire length, in reality two circuits will most likely differ in their wire length. Different PVs have been used to uniquely identify various objects ranging from expensive merchandise to integrated circuits [28, 38, 19, 17, 15]. These new exciting results open the door to the exploration of various cryptographic applications which could benefit from the use of PVs.

In this dissertation we examine four advantages that can be gained from using PVs in cryptographic applications.

Unique identification of physical objects: The ability to achieve a secure

method of identifying physical objects would significantly limit counterfeiting; a problem which, according to the US chamber of commerce, costs the US economy USD 250 billion yearly. A number of recent results have used PVs to produce unique fingerprints for different physical objects (see, for example, [38, 19, 15]). Although these results are quite recent, using PVs to produce unique access cards was proposed decades ago by Bauder and Simmons from Sandia National Labs [5]. More recently, the work by Ravikanth Pappu on physical one-way functions [83] has reignited this area of research and stimulated a large number of related results. Another notable work in the field is that of Tuyls et al. in [96]. In their book, these authors present a comprehensive collection of articles on noisy data which include a number of results relating to the unique identification of physical objects. One of our interests in this dissertation will be to quantify the identification capability of PVs under specific assumptions on their distribution. We will also present a specific example where PVs can be used to uniquely identify compact discs (CDs).

Lightweight implementations: Although PVs are typically used for their ability to uniquely identify physical objects, they also seem to offer a cheap resource that can be utilized for cryptographic purposes. The area of lightweight cryptographic design has seen quite an impressive boom in the past few years (see, e.g., [23, 52] and the references therein). One of the most popular techniques used to reduce the size of hardware implementations is to serialize classical cryptographic protocols [24, 51, 82, 66]. Although these results are exciting, the approach itself seems to be inherently limited. In general, most classical cryptographic protocols were designed with little attention paid to their hardware implementation. Indeed, not all such protocols lend themselves to serialization; alternative approaches are required. From a broader perspective, the notion of taking classical protocols and attempting to squeeze them into smaller circuits seems to eventually come up against natural limitations. A different, yet equally exciting approach, is to explore new protocols which

are designed to be lightweight in nature. A few examples of such new lightweight constructions are given in [11, 70, 45, 90]. These proposals are mostly block ciphers which focus on reducing the gate count but do not address tamper-resilience or computational hardness. However, the approach itself seems promising in principle. As we will see in later chapters, PVs can offer a natural approach to lightweight and secure implementations. In essence, one can build cryptographic primitives which use the PV value as a parameter. As PVs assume real values, they facilitate the construction of functions which are parameterized with real values, without having to incur the expensive cost of storing real values. All that would be needed to utilize the PV in some cryptographic function is to find a mechanism to extract the PV value. In this dissertation we explore this approach and present a number of cryptographic primitives parameterized by PVs and which can be implemented with a significantly low number of gates.

Tamper resilience: While typical cryptographic protocols might be very secure in theory, one has to take into account attacks which exploit so called side-channel attacks [63, 62] directly targeting the implementation. These attacks are roughly classified into two groups. Passive attacks solely observe side-channels (e.g. computation time, power consumption, electromagnetic emanation, temperature fluctuation, etc., i.e. PVs) to deduce internal secrets from leaked side-channel profiles. In contrast, with active attacks the attacker may also inject faults during the computation [64] or manipulate the hardware in other ways. Not surprisingly, active attacks are more powerful and are more difficult to prevent. A tamper-resilient hardware design is vital in securing devices from both passive and active side-channel attacks. To detect active side-channel attacks a number of linear [30, 31] and non-linear robust coding techniques [32] have been proposed. While linear coding techniques bear little overhead, they provide protection only against random errors. A more able attacker, who can choose errors, can only be stopped by non-linear robust codes. Unfortunately,

these methods also incur significant overhead, i.e. roughly more than 100%. At the time of this writing, robust codes seem to be the best candidate for error detection. As such, in this work we highlight some of the PVs benefits in preventing active attacks against hardware.

Computational security: Any cryptographic primitive is in need of some security assurance. These assurances usually come in the form of security reduction to known computationally difficult problems. The problem, however, is that cryptographic primitives which do posses these security assurances are usually not natural to the hardware and are therefore quite demanding in terms of resources. In constrained environments, where only lightweight implementations are allowed, building these primitives becomes a challenge. The question we try to answer here is whether there exists computationally difficult problems that are naturally supported by hardware implementations. A number of recent results have provided quite elegant solutions to this problem (see, for example, [46, 50, 33, 53]). In this work we further develop these results and at the same time present new cryptographic primitives which utilize PVs in order to provide hard problems which can be efficiently implemented.

1.1 Dissertation Outline

In this dissertation we strive to achieve two main goals.

- First, to explore the ability of PVs to uniquely identify devices under certain assumptions.
- Second, to explore new cryptographic applications of PVs and to highlight the advantages gained from using a PV in these applications.

Each of these goals will be addressed in one of the two parts making up this dissertation.

1.1.1 Physical Variables and Fingerprinting

The main theme of this part of the dissertation will be a high level view of PVs with an emphasis on their ability to uniquely identify or fingerprint physical objects.

In Chapter 2 we review some previous work on PVs and on fingerprinting. Furthermore, we show when a Gaussian PV can be used to identify physical objects. We also review how to generate a repeatable fingerprint for each physical object under certain assumptions.

In Chapter 3 we introduce a new technique for extracting unique fingerprints from identical CDs. The proposed technique takes advantage of manufacturing variability found in the length of the CDs' lands and pits which will be the PV used in this chapter. Although the variability measured is on the order of 20 nm, the technique does not require the use of microscopes or any advanced equipment. Instead, the electrical signal produced by the photo detector inside the CD reader will be sufficient to measure the desired variability. We thoroughly investigate this new technique by analyzing data collected from 100 identical CDs and show how to extract a unique fingerprint from each CD. We also give specific parameters and code constructions to convert the derived fingerprints into 128-bit cryptographic keys. Finally, we outline a simple protocol which utilizes the CD fingerprint to provide IP protection for the software stored on the CD.

1.1.2 Applications of Physically Parameterized Functions

In this part of the dissertation we shift our attention to physically parameterized functions (PPFs). Simply stated, PPFs are functions that use PVs as parameters. A special case of PPFs are physically unclonable functions (PUFs) which will be the center of our discussion in this part of the dissertation.

In Chapter 4 we throughly explore delay-based PUFs and their different constructions. The main job of this chapter will be to set the stage for PUFs before we can

use them in a number of cryptographic schemes.

In Chapter 5 we merge the delay-PUF along with the HB based authentication protocol to produce PUF-HB; a hybrid protocol that enjoys the advantages of both schemes while improving the level of security. The proposed authentication scheme observes a level of tamper resilience, while at the same time being provably secure against active attacks in the detection based model. In addition, the presented protocol resists the man-in-the-middle attacks proposed so far for the HB^+ scheme. From the PUF perspective, PUF-HB is the first PUF based authentication scheme with a security reduction. From the HB perspective, PUF-HB is the first hardware HB implementation with tamper resilience properties.

In Chapter 6 we present a proof of concept implementation for HB+PUF, a variant of the PUF-HB. The HB+PUF protocol enjoys the same properties of PUF-HB with a much simpler security reduction. Our implementation takes advantage of the PUF circuit in order to produce the random bits typically needed for an HB-based authentication scheme. Note that the existence of a random number generator (RNG) is assumed in all HB-based protocols without taking into account the complexity of such device. The overall circuit is shown to occupy on the order of a few thousand gates. Note that this is considered a benchmark for lightweight implementations. This small gate count is achieved by using the tristate PUF and by serializing the operations.

In Chapter 7 we show that delay-PUFs alone can be used to create a secure challenge response authentication scheme. We present a new primitive, namely 2-level PUFs which allow for a reduction to a special class of a threshold of majority gates under the uniform distribution. Furthermore, we explore extensions of PUFs to produce efficient n-to-n mappings.

It is important to note that our discussion on PVs will not be specifically concerned with aging effects. While this is quite an important topic to be studied when

addressing PVs it be beyond the scope of this dissertation.

1.2 Editorial Note

Each chapter will include its own related work. We will be using standard notation for all our work, any specific terms or concepts will be explained when they first appear in a chapter.

Many of the results in this dissertation have previously appeared in published papers. All the results are joint work with my adviser, Professor Berk Sunar. Chapter 2 contains some results which have not been submitted for review. The results of Chapter 3 are joint work with professor Aykutlu Dana from the institute of material science and technology at Bilkent University and will appear in [40]. The results of Chapters 4 and 7 are joint work with Erdinç Öztürk and have appeared in [78, 79, 42]. The results of Chapter 6 are joint work with Berk Birand and Erdinç Öztürk and have appeared in [41]. Finally the results of Chapter 5 have appeared in [43]. Other related work of mine, which is not part of this dissertation, includes results on the usage of PUFs in finite state machines which is joint work with Kahraman Akdemir [39].

Part I

Physical Variables and Fingerprinting

Chapter 2

PVs and Fingerprinting

Physical variables (PVs) can be viewed as a description of some property in a physical object. The essential idea behind PVs is that although some physical objects are manufactured to be identical, some of their properties will be different due to manufacturing variability. When these properties can be described using a numeric value, we can model the value of the PV as a random variable sampled from some probability distribution. Our goal in this chapter will be to show when certain types of PVs can be used to uniquely identify physical objects. In order to achieve this goal, we will restrict our attention to PVs with values sampled from a Gaussian distribution[1]. We will also model noise in the measured PV value as additive Gaussian noise. This view of a Gaussian PV with Gaussian noise should be quite useful and common for two main reasons. First, Gaussian distributions are typically used to describe natural phenomena. Second, the Central Limit Theorem states that any variable which is the sum of a large number of independent and identically distributed variables is likely to be a Gaussian [80]. In fact, all the PVs which we will address in this dissertation will be treated as independent Gaussian random variates with measurements subject to independent additive Gaussian noise.

[1]We will use the terms Gaussian distribution and Normal distribution interchangeably.

Another important concept which we review in this chapter is that of fuzzy extractors. Even when it is possible to uniquely identify physical objects, it is not clear if it will also be possible to generate a unique fingerprint for the physical object. Every time a PV is measured, its value will be accompanied by a level of noise. Therefore, a fingerprint generated from a PV value is also expected to change every time the PV is measured. To solve this problem, fuzzy extractors were introduced. The main job of a fuzzy extractor is to use the PV value to generate a fingerprint which can be regenerated from a noisy version of the PV value. Therefore, fuzzy extractors can guarantee the generation of a fixed fingerprint for each physical object.

The remainder of this chapter is organized as follows. In the next section we discuss some of the previous work related to PVs and fingerprinting. Section 2.2 introduces PVs and sets the stage for using them to fingerprint physical objects. Section 2.3 addresses discretization techniques for PVs and shows when Gaussian PVs can be used to uniquely identify physical objects. Finally, in Section 2.4, we review fuzzy extractors.

2.1 Previous Work

Over the past years, several research papers have focused on using biometrics to uniquely identify humans, a lot of the techniques introduced in these papers can be applied for usage in fingerprinting physical devices. A thorough study on these various techniques and connections has been provided by Tuyls et. al in their book "Security with Noisy Data" [96]. In their book, the authors explore various techniques used in biometrics and physical devices in order to prevent counterfeiting and illegal access to secret information. As our interest is mainly directed towards physical objects, we will focus our attention on work done in that particular area.

One of the earlier work related to (PVs) was presented by Ravikanth Pappu in his

PhD thesis [83]. The main focus of his work was to define and explore physical one-way functions (POWF). The motivation for POWF is to provide a one-way function which depends on PVs and measurement probes such that the retrieval of the PV value or the probe's configuration is difficult to carryout using computational means or physical interactions.

Another concept which is also dependent upon PVs is that of physically unclonable functions (PUFs) [29, 71]. PUFs are a natural successor of POWFs and are therefore very similar to them. The idea behind PUFs is to build physical devices which compute a function such that an adversary cannot clone the device or deduce the implemented function. Unfortunately, due to the popularity of the name, several physical functions which do not satisfy the unclonability property have been labeled as PUFs. In this work we argue that the term physically parameterized function (PPFs) is a more accurate description for these functions. In Part II of this dissertation we will talk extensively about PUFs.

One of the central motivations to build physical functions is to be able to uniquely identify different physical objects. Some recent proposals directly use PVs to generate a unique fingerprint for different devices. In [3] a circuit fingerprinting technique was introduced. The technique exploits manufacturing variability in integrated chips to detect Trojan circuits inserted during the manufacturing process. The PV used for this technique is the power consumption of the fingerprinted circuit. Another secure fingerprinting technology named RF-DNA was developed by Microsoft Research [19]. The RF-DNA technology provides unique and unclonable physical fingerprints based on the subtleties in the device's reaction when subjected to an electromagnetic wave. The PV used here is the device's response when subjected to electromagnetic waves. The fingerprints are used to produce a cryptographic certificate of authenticity (COA), which when associated with a high value good, may be used to verify the authenticity of the goods and to distinguish it from counterfeit goods. Another

application of manufacturing variability is fingerprinting paper objects. In [17] the authors propose laser surface authentication where a high resolution laser microscope is used to capture the paper texture, from which a PV is derived and a fingerprint is developed. In a more recent proposal, a cheap commodity scanner was used to identify paper documents [15]. While most of the results cited above were developed in the last decade, the idea of using physical fingerprints to obtain security primitives is not new at all. Access cards based on physically unclonable properties of media were proposed decades ago by Bauder and Simmons from Sandia National Labs [5]. In Chapter 3 we will present a technique to fingerprint CDs from the length of the lands and pits used to store the information on the CD.

These examples use different types of PVs to identify physical objects. For the remainder of this chapter we will not be concerned with what the PV represents but will rather focus our attention on the PV distribution.

2.2 PVs

As we mentioned in the beginning of this chapter PVs can be viewed as a description of some property in a physical object. The idea behind PVs is that although some physical objects are manufactured to be identical, some of their properties will be different due to manufacturing variability. Any manufacturing process will undergo a number of external effects which typically cause variation in the physical objects produced. In fact, trying to eliminate this type of variability is a challenging task which has received much attention from the research community (see, for example, [81, 1]). A simple example of manufacturing variability can be seen in the length of wires inside any integrated-chip (IC). As these wires are made out of a number of atoms, it is common that the same wire (at the design level) will have a different number of atoms (and therefore different length) from one chip to another. Of course

when the variance in the wire's length is low, its effect on the circuit's behavior is unnoticeable. The manufacturing process research is usually concerned with large manufacturing variability which can have detrimental results on the overall performance of the physical object. In this dissertation we are mainly interested in the type of manufacturing variability which is not harmful to the functionality of the physical object. Therefore, we work under the assumption that the physical object is functional.

In order to be able to reason about PVs we model a PV as a random variable $X \in \mathbb{R}$ with some probability distribution $\mathcal{H}$ over the reals. Every physical object manufactured from some design D will have a different PV value. Therefore, we view a physical object as a sample from the distribution $\mathcal{H}$. We use $\psi(D)$ to denote the set of all physical objects which are manufactured from some design D, and we say that X is a PV for $\psi(D)$. Now every physical object $S_j \in \psi(D)$ will have a an ideal PV value $\hat{X}_j$. We will slightly abuse the notation and use $S_j \in_{\mathcal{H}} \psi(D)$ to denote drawing a physical object S_j from $\psi(D)$ such that $\hat{X}_j$ is chosen according to $\mathcal{H}$. We also use $S_1, \ldots, S_k \in_{\mathcal{H}} \psi(D)$ to denote that $\hat{X}_1, \ldots, \hat{X}_k$ were drawn independently according to $\mathcal{H}$. Here, we used $\hat{X}_j$ to denote the ideal value of X in S_j. However, as the measurement of X will contain noise we use X_j to denote a measurement of $\hat{X}_j$. We also model $X_j \in \mathbb{R}$ as a random variable with some probability distribution $\mathcal{D}_j$ over the reals. Notice that every physical object S_j will determine the ideal value of the PV and the noise distribution $\mathcal{D}_j$.

A natural extension of the above model is for the physical object to have multiple PVs. With this view we treat X as a vector in $\mathbb{R}^n$. Each physical object $S_j \in \psi(D)$ will have an ideal PV value $\hat{X}_j \in \mathbb{R}^n$. The i^{th} entry (coordinate) of $\hat{X}_i$ will have an ideal value $\hat{X}_{ij} \in \mathbb{R}$ and can be modeled as a random variable following the distribution $\mathcal{H}_i$. Here, we are making the assumption that the distributions $\mathcal{H}_i$ are independent. In reality there will always be a level of dependency between these dis-

tributions. However, we make this assumption in order to simplify our analysis. The measurement of the i^{th} entry of $\hat{X}_j$ on some physical object S_j will be an independent sample from the distribution $\mathcal{D}_{ij}$. We will be using a single subscript to mean a vector containing the values of all the entries of X in a specific physical object (eg. X_j), and double subscripts X_{ij} to be the value of the i^{th} entry of X_j. We also call n the size of the PV. Now when we use the notation $S_1, \ldots, S_k \in_{\mathcal{H}} \psi(D)$ we mean that the i^{th} entry of each vector $\hat{X}_1, \ldots, \hat{X}_k$ was drawn independently according to $\mathcal{H}_i$. Because the distributions $\mathcal{H}_i$ are independent, the multivariate distribution $\mathcal{H}$ over $\mathbb{R}^n$ will be $\mathcal{H} = \prod_{i=1}^{n} \mathcal{H}_i$.

With this model we can now reason about distributions rather than physical objects. It is important to note here that the above is an idealized model. For example, when we consider continuous distributions we are implying that $\psi(D)$ contains an infinite number of objects which is clearly not the case. Although we will be using an idealized model to derive some of our equations, we will see in the next chapter that the model closely predicts some of the experimental results.

Before we finish this section we discuss what we mean by the ability to uniquely identify physical objects. We first introduce the following definition. We say that a function Δ on some set $\mathcal{S}$ is a semi-pseudo distance if $\Delta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+ = [0, \infty)$ and for any elements $a, b \in \mathcal{S}$, $\Delta(a, b) = \Delta(b, a)$ and $\Delta(a, a) = 0$. The reader should note that what we refer to as a semi-pseudo distance function is a distance metric without the triangle inequality and with a weaker notion of identity. In general, there are a number of functions which are weaker than a distance metric, yet they share some of the properties of a distance metric [89]. The semi-pseudo distance is a mix between a semi-metric which does not require the triangle inequality and a pseudo metric which allows non-identical elements to have distance 0. One other technical term which we need is that of negligible functions. We say that a function $f(n) : \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial $p(x)$ there exists some $N_0 \in \mathbb{N}$ such that for all $n > N_0$

we have $f(n) < \frac{1}{p(n)}$ [36].

Now let $X_1, X_1' \in \mathbb{R}^n$ be two vectors where the i^{th} entry of each vector is an independent sample from the distribution $\mathcal{D}_{i1}$. Similarly, let $X_2, X_2' \in \mathbb{R}^n$ be two vectors where the i^{th} entry of each vector is an independent sample from the distribution $\mathcal{D}_{i2}$. The ideal values for X_1, X_1' and X_2, X_2' will be $\hat{X}_1$ and $\hat{X}_2$ respectively, both of which are independent samples from the distribution $\mathcal{H}$. Now let $\mathcal{I}$ be some extraction algorithm which takes in values from $X \in \mathbb{R}^n$ and produces output over some set $\mathcal{S}$. For X to be useful in uniquely identifying physical objects we require it to satisfy the following definition.

Definition 2.2.1. (Identifying Physical Variable (IPV)) *Let $\mu : \mathbb{N} \rightarrow \mathbb{R}^+$. A PV $X \in \mathbb{R}^n$ for $\psi(D)$ is said to be an identifying physical variable if there exists an algorithm $\mathcal{I}$ with running-time polynomial in n such that for any two objects $S_1, S_2 \in_{\mathcal{H}} \psi(D)$, $\mathcal{I}$ can achieve*

$$\Pr[\Delta(\mathcal{I}(X_1), \mathcal{I}(X_1')) < \Delta(\mathcal{I}(X_1), \mathcal{I}(X_2))] \geq 1 - \mu(n) \ ,$$

where $\mu(n)$ is a negligible function and $\Delta(a, b)$ is some semi-pseudo distance over $\mathcal{S}$. The probability is taken over any pair of objects in $\psi(D)$ sampled independently according to $\mathcal{H}$.

We state this definition for Identifying PVs in order to express and quantify the notion of the ability to uniquely identity physical objects. The definition above captures the notion that under our idealized assumptions the measured PV values from some object S_1 will with a very high probability be closer to each other than the measured PV values from a different object S_2.

2.3 Discretization and Identification

In this section we will restrict our attention to Gaussian distributions for $\mathcal{H}_i$ and for $\mathcal{D}_{ij}$. We will also formalize our assumptions in order to explore when Gaussian PVs

can be considered as IPVs. Recall that a Gaussian ($N(M, \Sigma^2)$) random variable X has a probability distribution function

$$\frac{e^{(X-M)^2/2\Sigma^2}}{\sqrt{2\pi\Sigma^2}} \quad ,$$

where M is the mean and Σ^2 is the variance.

In order for X to identify a large number of objects there needs to be a sufficient amount of entropy in $\mathcal{H}$. In general the entropy of a Gaussian distribution of variance Σ^2 will be $\log\left(\Sigma\sqrt{2\pi e}\right)$ which is the maximum entropy among all real-valued distributions with a given mean and standard deviation [16].[2] This quantity is limited, and will require a large Σ in order to allow the identification of a large number of objects. This is why we will need to consider PVs with n values larger than 1.

In some sense our goal will be to extract the entropy from X and use it to generate the fingerprint. This task is made harder with the noise contaminating the ideal values of the PV. Recall that X assumes values from the reals. It is typically easier to reason about the identification ability of binary fingerprints. Due to this reason we will start our extraction process by imposing a discretization technique on X. This discretization process will have significant impact on both the noise level and the identification ability of X. In this section we present two discretization techniques, the halfspace technique and the threshold technique. In the coming two sections we will elaborate on each of these techniques and then derive closed expression formulas for their error and collision probability. Different discretization techniques can be found in [96].

Before we start, we will list the assumptions that we have stated and which we will use in the next two sections. We will use $\mathcal{H}$ to denote that multivariate Gaussian made of all n independent Gaussian distributions $\mathcal{H}_i = N(M_i, \Sigma^2)$. Where M_i is the mean of the i^{th} entry in X over all the physical objects in $\psi(D)$. Note that by using

[2]Note that unless we mention otherwise, all logarithms in this dissertation are with respect to base 2.

Σ for all the entries of X we are assuming that the variability in all the entries of X is equal. The measurement of the i^{th} entity of X on some physical object S_j will be a sample X_{ij} from $\mathcal{D}_{ij} = N(\mu_{ij}, \sigma^2)$. We assume that $\mathcal{D}_{ij}$ will be additive noise to the ideal PV value $\hat{X}_{ij}$. Furthermore, to simplify our derivations we assume that the noise has a mean value of 0, and therefore $\mu_{ij} = \hat{X}_{ij}$. In our derivations we use μ_{ij} to avoid confusion. Also, by fixing the variance in all the measurement distributions to σ^2 we are making the assumption that the measurement devices on any physical object in $\psi(D)$ will introduce the same level of noise. Following we formalize these assumptions.

Assumption 2.3.1. *Let $X \in \mathbb{R}^n$ be a PV for $\psi(D)$. We assume,*

1. *For any physical object $S_j \in \psi(D)$ the measurement of the i^{th} entry of X_j is an independent sample from the distribution $\mathcal{D}_{ij} = N(\mu_{ij}, \sigma^2)$.*

2. *For any given physical object $S_j \in_{\mathcal{H}} \psi(D)$ the mean of the measurement on the i^{th} entry of X which is μ_{ij} will be an independent sample from the distribution $\mathcal{H}_i = N(M_i, \Sigma^2)$.*

3. *The distributions $\mathcal{H}_i = N(M_i, \Sigma^2)$ are independent for $i \in [1..n]$.*

The above captures our idealized model of a PV.

2.3.1 Halfspace Technique

The idea here is to extract a single bit from each entry in X. This can be achieved by simply breaking the space of X_{ij} in half, such that over all possible values of X_{ij} the probability of extracting a 0 or 1 is equal. This approach was first proposed in [94] and [73]. In order to extract a single bit we will use the mean of X_{ij} overall physical objects in $\psi(D)$. The following equation captures the technique.

$$Z_{ij} = \mathrm{HT}(X_{ij}) = \left\{ \begin{array}{ll} 1, & X_{ij} > M_i \\ 0, & X_{ij} \leq M_i \end{array} \right\}. \tag{2.1}$$

It should be clear that since $\mathcal{H}_i$ is symmetric around M_i the above will yield a 0 or a 1 with equal probability. Which means that Z_{ij} for two different objects will collide with probability 0.5, we label this probability as P_c. This guarantees the extraction of a single bit of entropy. With Assumption 2.3.1, the n entries of X will result in n bits of entropy. Therefore, if we consider $Z_j = [Z_{1j}, \ldots, Z_{nj}] \in \{0,1\}^n$ to be the fingerprint for each object, two physical objects $S_1, S_2 \in_{\mathcal{H}} \psi(D)$ will have an expected Hamming distance of $\frac{n}{2}$ between Z_1 and Z_2. However, with the existence of noise the Hamming distance between Z_1 and Z_2 will be affected. The next proposition will address the issue of noise. In specific, the proposition quantifies the probability of Z_{ij} yielding different values after two different reads. We use Z_{ij} and Z'_{ij} to denote two different extracted bits from the i^{th} entry of a PV on the same physical object. This proposition was proven in [57]. We restate the following proposition using our notation.

Proposition 2.3.2. ([57]) *let $X \in \mathbb{R}^n$ be a PV under Assumption 2.3.1, and $S_j \in_{\mathcal{H}} \psi(D)$ be a physical object. Also, let Z_{ij} and Z'_{ij} be the result of applying the halfspace technique of Equation 2.1 to the output of two independent measurements of X_{ij}. The error probability*

$$P_e = \Pr[Z_{ij} \neq Z'_{ij}] = \frac{1}{2} - \frac{1}{\pi} \arctan\left(\frac{\Sigma}{\sigma\sqrt{2 + \frac{\sigma^2}{\Sigma^2}}} \right) . \tag{2.2}$$

It will be useful to rewrite the equation above using the ratio $R = \frac{\sigma}{\Sigma}$ as

$$P_e = \frac{1}{2} - \frac{1}{\pi} \arctan\left(\frac{1}{R\sqrt{2 + R^2}} \right) .$$

Now we know the collision and error probability. Recall from Definition 2.2.1 that we need a semi-pseudo distance such that except with a negligible probability the

distance between Z_1 and Z_1' is smaller than the distance between Z_1 and Z_2. As the generated fingerprints are binary it will be natural to use the Hamming distance as our metric. For two binary vectors $a, b \in \{0,1\}^n$ we use $d(a,b) = |a \oplus b|$ to denote the Hamming distance, where $|a|$ is the Hamming weight[3] of a. Now we can prove the following theorem.

Theorem 2.3.3. *let $X \in \mathbb{R}^n$ be a PV under Assumption 2.3.1, and $S_1, S_2 \in_{\mathcal{H}} \psi(D)$. If $n = \Omega(R^5)$ then $d(Z_1, Z_1') < d(Z_1, Z_2)$ except with a negligible probability in n. Therefore, X is an IPV.*

Proof. Let $d_1 = d(Z_1, Z_1')$ and $d_{12} = d(Z_1, Z_2)$. For each bit in the Z_j the probability of Z_{i1} and Z_{i2} not being equal is $P_{nc} = 1 - P_c = \frac{1}{2}$, while the probability of an error (different bit) between Z_{i1} and Z_{i1}' is $P_e = \frac{1}{2} - \frac{1}{\pi} \arctan\left(\frac{1}{R\sqrt{2+R^2}}\right)$. We need to find a specific Hamming distance which separates d_1 and d_{12} except with a negligible probability. We set this Hamming distance at $n \cdot \varepsilon$ where $\varepsilon = \frac{1}{2}(P_{nc} - P_e)$. Now using Hoeffding Inequality [44] we have

$$\Pr\left[|d_1 - nP_e| \geq n\varepsilon\right] < 2e^{-2\varepsilon^2 n}$$

Similarly,

$$\Pr\left[|d_{12} - nP_{nc}| \geq n\varepsilon\right] < 2e^{-2\varepsilon^2 n}$$

We know that $P_e \leq P_{nc}$, so with probability at least $1 - 4e^{-2\varepsilon^2 n}$ we have

$$\begin{aligned} d_{12} &> nP_{nc} - n\varepsilon \quad \text{and} \\ n\varepsilon + nP_e &> d_1 \ . \end{aligned}$$

Therefore, $d_{12} > d_1$. We now need to prove that with our choice of R the failure probability $P_f = 4e^{-2\varepsilon^2 n}$ is negligible. We have,

$$\varepsilon = \frac{1}{2}(P_{nc} - P_e) = \frac{1}{2\pi} \arctan\left(\frac{1}{R\sqrt{2+R^2}}\right)$$

[3]The Hamming weight of a binary vector is the number of ones in the vector.

One can easily show that $\arctan(x) \geq \frac{x}{x+1}$. Therefore,

$$\frac{1}{2\pi}\left(\frac{1}{1+R\sqrt{2+R^2}}\right) \leq \varepsilon \quad .$$

With our choice of n we get,

$$\varepsilon = \Omega\left(\frac{1}{n^{2/5}}\right)$$

and finally,

$$P_f = 4e^{-2\varepsilon^2 n} = 4e^{-\Omega\left(n^{1/5}\right)} \quad ,$$

which is negligible in n. This satisfies the conditions for an IPV (Definition 2.2.1). □

The theorem above proves that even with a significant noise σ compared to the variation in the PV Σ it is possible to use X as an IPV. Of course, using asymptotes we are implicitly assuming that n can be made arbitrarily large. Also, we assume a dependence between R and n. To get exact numbers for the failure probability and the Hamming distances one can directly plug numbers into the equations above. We next shift our attention to a different technique.

2.3.2 Threshold Technique

The halfspace technique was quite simple and straight forward to apply. It also enabled us to easily quantify the extracted entropy. However, the problem with the halfspace technique is that there might be more than a single bit of entropy in each of the entries in X. Restricting our attention to a single bit might be wasteful in certain cases. Another problem with the halfspace technique is that it requires knowing the mean M_i for every entry of X. This might not be possible for all applications. These two problems motivate the threshold technique.

In the threshold technique the extracted binary string is simply the binary representation of X_{ij}. The essence of this technique is really in the distance function defined. It should be clear that using a Hamming distance will not be natural to the

readings which come from the reals. Two values X_{ij} and X'_{ij} which are close to each other (using the L_1-distance), will not necessarily be close when using the Hamming distance over their binary representation. Our focus in this section will be on the distance function which will enable us to use the binary representation of X_{ij} as our fingerprint. Also, keep in mind that our goal is to place conditions on σ and Σ such that X can become an IPV.

For two real values a_i and b_i we define the threshold distance which is a semi-pseudo distance as follows. For any positive τ,

$$d^{\tau}(a_i, b_i) = \left\{ \begin{array}{ll} 0, & |a_i - b_i| \leq \tau \\ 1, & |a_i - b_i| > \tau \end{array} \right\}. \tag{2.3}$$

For two real vectors $a = [a_1, \ldots, a_n] \in \mathbb{R}^n$ and $b = [b_1, \ldots, b_n] \in \mathbb{R}^n$ the distance is defined as

$$d^{\tau}(a, b) = \sum_{i=1}^{n} d^{\tau}(a_i, b_i)$$

We now start by quantifying the probability of an error occurring (threshold distance 1) when measuring the i^{th} entry of X on the same physical object. Before we start, we recall the definition of the error function which is

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad .$$

Proposition 2.3.4. *let $X \in \mathbb{R}^n$ be a PV under Assumption 2.3.1, and $S_j \in_{\mathcal{H}} \psi(D)$ be a physical object. The probability of having a threshold distance of 1 between X_{ij} and X'_{ij} is*

$$P_e = \Pr[d^{\tau}(X_{ij}, X'_{ij}) = 1] = 1 - \mathrm{erf}\left(\frac{\tau}{2\sigma}\right) \quad . \tag{2.4}$$

Proof. The reading X_{ij} will be drawn from $\mathcal{D}_{ij} = N(\mu_{ij}, \sigma)$. We quantify the probability that two independent samples from $\mathcal{D}_{ij}$ are a distance τ apart. We will quantify the probability that no error occur which will be easier to carryout. We use dummy

variables x and z in the derivation. Without loss of generality, since both X_{ij} and X'_{ij} come from the same distribution we shift the mean to 0. $\Pr[|X_{ij} - X'_{ij}| \leq \tau] =$

$$\begin{aligned}
&= \int_{-\infty}^{\infty} \frac{e^{-z^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} dz \int_{z-\tau}^{z+\tau} \frac{e^{-x^2/2\sigma^2}}{2\sqrt{2\pi\sigma^2}} dx \\
&= \frac{1}{2\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \left[\operatorname{erf}\left(\frac{z+\tau}{\sqrt{2\sigma^2}}\right) - \operatorname{erf}\left(\frac{z-\tau}{\sqrt{2\sigma^2}}\right) \right] e^{-z^2/2\sigma^2} dz \\
&= \frac{1}{2\sqrt{2\pi\sigma^2}} \left[\int_{-\infty}^{\infty} e^{-z^2/2\sigma^2} \operatorname{erf}\left(\frac{z+\tau}{\sqrt{2\sigma^2}}\right) dz - \int_{-\infty}^{\infty} e^{-z^2/2\sigma^2} \operatorname{erf}\left(\frac{z-\tau}{\sqrt{2\sigma^2}}\right) dz \right] \\
&= \frac{1}{2} \left[\operatorname{erf}\left(\frac{\tau}{2\sigma}\right) - \operatorname{erf}\left(\frac{-\tau}{2\sigma}\right) \right] \\
&= \operatorname{erf}\left(\frac{\tau}{2\sigma}\right)
\end{aligned}$$

where the third line was done using

$$\int_{-\infty}^{\infty} e^{-(\alpha z+\beta)^2} \operatorname{erf}(\gamma z + \delta)\, dz = \frac{\sqrt{\pi}}{\alpha} \operatorname{erf}\left(\frac{\alpha\delta - \beta\gamma}{\sqrt{\alpha^2+\gamma^2}}\right) . \tag{2.5}$$

Now we have,

$$P_e = \Pr[d^\tau(X_{ij}, X'_{ij}) = 1] = 1 - \Pr[|X_{ij} - X'_{ij}| \leq \tau] = 1 - \operatorname{erf}\left(\frac{\tau}{2\sigma}\right)$$

□

Next, we need to quantify the probability of a collision (threshold distance 0) occurring when measuring the i^{th} entry of X on two physical objects.

Proposition 2.3.5. *let $X \in \mathbb{R}^n$ be a PV under Assumption 2.3.1, and $S_1, S_2 \in_{\mathcal{H}} \psi(D)$. The probability of having a threshold distance of 0 between X_{i1} and X_{i2} is*

$$P_c = \Pr[d^\tau(X_{i1}, X_{i2}) = 0] = \operatorname{erf}\left(\frac{\tau}{2\sqrt{\sigma^2+\Sigma^2}}\right) \quad . \tag{2.6}$$

Proof. The measurement of X_{ij} will be drawn from $\mathcal{D}_{ij} = N(\mu_{ij}, \sigma)$. However, the mean μ_{ij} will be drawn from $\mathcal{H}_i = N(M_i, \Sigma^2)$. For a collision to happen we need $|X_{i1} - X_{i2}| \leq \tau$. To compute this probability we integrate over all values of the mean

of the first object taken from $\mathcal{H}_i$ then we integrate over all possible values of X_{i1} taken from $\mathcal{D}_{i1}$. This will cover every possible reading coming from the X_{i1}. We will still have to integrate over all the mean values of the second object taken from $\mathcal{H}_i$ and finally we integrate over all the values of X_{i2} coming from $\mathcal{D}_{i2}$ and which are at a distance τ from X_{i1}. Similar to the previous proof we will without loss of generality shift the mean of $\mathcal{H}_i$ to 0 as both X_{i1} and X_{i2} come from the same distribution. We use dummy variables z, x, t and l in the derivation. Now we have $\Pr[|X_{i1} - X_{i2}| \leq \tau] =$

$$
\begin{aligned}
&= \int_{-\infty}^{\infty} \frac{e^{-z^2/2\Sigma^2}}{\sqrt{2\pi\Sigma^2}} dz \int_{-\infty}^{\infty} \frac{e^{-(x-z)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} dx \int_{-\infty}^{\infty} \frac{e^{-t^2/2\Sigma^2}}{\sqrt{2\pi\Sigma^2}} dt \int_{x-\tau}^{x+\tau} \frac{e^{-(l-t)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} dl \\
&= \int_{-\infty}^{\infty} \frac{e^{-z^2/2\Sigma^2}}{2\sqrt{2\pi\Sigma^2}} dz \int_{-\infty}^{\infty} \frac{e^{-(x-z)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} dx \cdot \\
&\quad \int_{-\infty}^{\infty} \frac{e^{-t^2/2\Sigma^2}}{\sqrt{2\pi\Sigma^2}} \left[\operatorname{erf}\left(\frac{-x+\tau+t}{\sqrt{2\sigma^2}} \right) - \operatorname{erf}\left(\frac{-x-\tau+t}{\sqrt{2\sigma^2}} \right) \right] dt \\
&= \int_{-\infty}^{\infty} \frac{e^{-z^2/2\Sigma^2}}{2\sqrt{2\pi\Sigma^2}} dz \int_{-\infty}^{\infty} \frac{e^{-(x-z)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} \left[\operatorname{erf}\left(\frac{-x+\tau}{\sqrt{2(\sigma^2+\Sigma^2)}} \right) - \operatorname{erf}\left(\frac{-x-\tau}{\sqrt{2(\sigma^2+\Sigma^2)}} \right) \right] dx \\
&= \int_{-\infty}^{\infty} \frac{e^{-z^2/2\Sigma^2}}{2\sqrt{2\pi\Sigma^2}} dz \left[\operatorname{erf}\left(\frac{z+\tau}{\sqrt{2(2\sigma^2+\Sigma^2)}} \right) - \operatorname{erf}\left(\frac{z-\tau}{\sqrt{2(2\sigma^2+\Sigma^2)}} \right) \right] dx \\
&= \frac{1}{2} \left[\operatorname{erf}\left(\frac{\tau}{\sqrt{2(2\sigma^2+2\Sigma^2)}} \right) - \operatorname{erf}\left(\frac{-\tau}{\sqrt{2(2\sigma^2+2\Sigma^2)}} \right) \right] dx \\
&= \operatorname{erf}\left(\frac{\tau}{2\sqrt{\sigma^2+\Sigma^2}} \right)
\end{aligned}
$$

where all the integrations were carried out using Equation 2.5. □

With the last two proposition we can now bound the value of n which allows X to be used as an IPV under the threshold distance. As before we use n entries for the PV X.

Theorem 2.3.6. *let $X \in \mathbb{R}^n$ be a PV under Assumption 2.3.1, and $S_1, S_2 \in_{\mathcal{H}} \psi(D)$. If $n = \Omega(R^5)$ and $T = \frac{\tau}{2\sigma} = \mathcal{O}(1)$, then $d^\tau(X_1, X_1') < d^\tau(X_1, X_2)$ except with a negligible probability in n. Therefore, X is an IPV.*

Proof. The theorem proceeds similar to Theorem 2.3.3. We define $P_{nc} = 1 - P_c$. Note that for the halfspace technique we had $P_{nc} = P_c$. Now set, $R = \frac{\sigma}{\Sigma}$ as before and $T = \frac{\tau}{2\sigma}$.

$$P_c = \text{erf}\left(\frac{\tau}{2\sqrt{\sigma^2 + \Sigma^2}}\right) = \text{erf}\left(\frac{\tau}{\frac{2\sigma}{R}\sqrt{1+R^2}}\right) = \text{erf}\left(T\frac{R}{\sqrt{1+R^2}}\right)$$

Let $C = \frac{R}{\sqrt{1+R^2}} \leq 1$. Next we set $\varepsilon = \frac{1}{2}(P_{nc} - P_e)$. We can now show that that $\varepsilon \geq 0$.

$$\varepsilon = \frac{1}{2}(\text{erf}(T) - \text{erf}(TC)) \quad ,$$

since the error function erf() is increasing for $T \geq 0$ and $C \leq 1$ therefore $\varepsilon \geq 0$ and $P_{nc} \geq P_e$. We can now use the same steps as Theorem 2.3.3 to deduce that $d^\tau(X_1, X_1') < d^\tau(X_1, X_2)$ except with probability $P_f = 4e^{-2\varepsilon^2 n}$. All we need to do now is to show that with our choice of R the failure probability P_f is negligible. We have,

$$\varepsilon \;=\; \frac{1}{2}(\text{erf}(T) - \text{erf}(TC)) = \frac{1}{\sqrt{\pi}}\int_{TC}^{T} e^{-u^2}du \geq \frac{e^{-T^2}}{\sqrt{\pi}}(T - TC)$$

We also have,

$$1 - C = 1 - \frac{R}{\sqrt{1+R^2}} = \Omega\left(\frac{1}{R^{2+\epsilon}}\right) \quad ,$$

for any $\epsilon > 0$.

$$\varepsilon = \Omega\left(\frac{1}{R^{2+\epsilon}}\right) = \Omega\left(\frac{1}{n^{(2+\epsilon)/5}}\right) \quad .$$

Substituting into P_f we get,

$$P_f = 4e^{\Omega(n^{(1-2\epsilon)/5})} \quad .$$

By choosing $\epsilon < 0.1$, where 0.1 was chosen as an arbitrary small value, the above becomes negligible in n. Therefore, X is an IPV. □

The significance of the theorem above is that even with an R which is quite significant there will always be a way to use X as an IPV using the threshold distance.

In terms of handling different R values there is barely any difference between the halfspace and the threshold techniques. This should be expected since we are using asymptotes. What is interesting about the threshold technique is that it does not require prior knowledge of the mean M_i. The above shows that when using the binary representation of X_{ij} as the fingerprint it is possible to use X as an IPV.

2.4 Fuzzy Extractors

In this section we review fuzzy extractors and quickly discuss their relation to the previous discretization techniques. In the next chapter we will utilize fuzzy extractors in a specific application. Loosely speaking a fuzzy extractor is a technique to extract randomness from a given string such that it is possible to reproduce the same output string from a noisy version of the input string. In [20] the authors show how a fuzzy extractor can be built using an error correcting code along with a universal hashing function. Their construction requires that the output of the fingerprint (the biometric data in their language) be represented as an element of $\mathcal{F}^n$ for some field $\mathcal{F}$ and an integer n which represents the size of the fingerprint. Moreover, it is naturally assumed that the noise experienced by the fingerprint is upper bounded by a constant distance from the original fingerprint in order to guarantee identical reproduction of the extracted key.

In some sense fuzzy extractors assume the existence of IPVs in order to provide some guarantee on the distance. It is also natural to think of fuzzy extractors as the natural next step after discretization. To a great extent our review here will follow [20]. We start by quoting the following definition from [20]. We will use the following terms. The definition of min entropy is $H_\infty(A) = -\log(\max_a \Pr[A = a])$. The conditional min-entropy is $H_\infty(X|I)$ (meaning X conditioned on I)[4]. Also, recall

[4]Typically we use the | operator to mean concatenation. This will be the only part of this dissertation where it will have a different meaning.

the definition of the statistical distance between two probability distributions A and B, $\mathbf{SD}(A,B)= \frac{1}{2}\sum_v |\Pr(A=v) - \Pr(B=v)|$. U_ℓ is the uniform distribution over $\{0,1\}^\ell$. Finally, $\mathrm{dis}(a,b)$ is the distance between a and b using some distance function.

Definition 2.4.1. ([20]) *A $(\mathcal{S}, m, \ell, t, \epsilon)$-average-case fuzzy extractor is a pair of randomized procedures, generate (*Gen*) and reproduce (*Rep*), with the following properties:*

1. *The generation procedure* Gen *on input $x \in \mathcal{S}$ outputs an extracted string $Z \in \{0,1\}^\ell$ and a helper string $P \in \{0,1\}^*$.*

2. *The reproduction procedure* Rep *takes an element $x \in \mathcal{S}$ and a bit string $P \in \{0,1\}^*$ as inputs. The correctness property of fuzzy extractors guarantees that if $dis(x,x') \leq t$ and R, P were generated by $(R,P) \leftarrow \mathrm{Gen}(x)$, then $\mathrm{Rep}(x',P) = R$. If $dis(x,x') > t$, then no guarantee is provided about the output of* Rep.

3. *The security property guarantees that for any distribution X on $\mathcal{S}$ of min-entropy m, the string R is nearly uniform even for those who observe P: if $(R,P) \leftarrow \mathrm{Gen}(X)$, then $\mathbf{SD}((R,P),(U_\ell,P)) \leq \epsilon$. Furthermore, if $H_\infty(X|I) \geq m$, then $\mathbf{SD}((R,P,I),(U^\ell,P,I)) \leq \epsilon$ for any auxiliary random variable I.*

A fuzzy extractor is efficient if Gen *and* Rep *run in expected polynomial time.*

In this chapter we will be interested in a specific class of fuzzy extractors, namely the code offset fuzzy extractors. The next theorem stated and proven in [20] outlines this class.

Theorem 2.4.1. ([20]) *Given any $[n,k,2t+1]_{\mathcal{F}}$ code $\mathcal{C}$ and any m, ϵ, there exists an average-case $(M, m, \ell, t, \epsilon)$-fuzzy extractor, where $\ell = m + kf - nf - 2\log(\frac{1}{\epsilon}) + 2$. The generation algorithm* GEN *and the recovery algorithm* REP *are efficient if C has efficient encoding and decoding.*

We explain the parameters in the theorem by outlining a specific construction. This construction was proposed in [20] and further detailed in [38]. As stated in the theorem, $\mathcal{C}$ is an error correcting code over the field $\mathcal{F}$, where $f = \log(|\mathcal{F}|)$. For the construction we will also need a family of universal hashing functions[5] H . The generation algorithm GEN takes the fingerprint $x \in \mathcal{F}^n$ as input and outputs the triplet (k, w, v). Here, x is drawn from some distribution X over $\mathcal{F}^n$ which has min-entropy m. GEN starts by computing $w = x + c$ for a randomly chosen code word $c \in \mathcal{C}$ and then computes the key $k = h_v(x) \in \{0,1\}^\ell$ for some string v chosen uniformly at random such that $h_v \in H$. The recovery algorithm REP takes in the *helper data* (w, v) along with x' a noisy version of the fingerprint x and returns the key k. REP starts by computing $c' = w - x'$ which is a noisy version of c. If the Hamming distance between x and x' is less than t then so will be the Hamming distance between c and c' and therefore using the error correcting code REP can reproduce c from c'. Next, REP can compute $x = w - c$ and consequently compute $k = h_v(x)$ which will conclude the recovery algorithm.

With this construction we will have a clear way to build a fuzzy extractor. However, the key size ℓ and the security parameter ϵ will both depend on m and the code used. Moreover, the code will depend on the noise rate in the fingerprint. We finish this section by relating the min-entropy and the error rate of the fingerprint. Recall, that x is required to have a min-entropy of m and at the same time using the construction above, x will have n symbols from $\mathcal{F}$. To merge these two requirements we define the average min-entropy in every symbol $\delta = m/n$. We also define ν to be the noise rate in the fingerprint x and $F = |\mathcal{F}|$. With these definitions we can now prove the following simple bound relating the noise rate and the min-entropy rate δ/f. We will first need the definition of F-ary entropy function.

$$H_F(p) = p \log_F \left(\frac{F-1}{p}\right) + (1-p) \log_F \left(\frac{1}{1-p}\right) \quad .$$

[5]For details on universal hashing the reader is referred to [14].

Proposition 2.4.2. *For the fuzzy extractor construction of Theorem 2.4.1, and for any meaningful security parameters of* $\epsilon < 1$ *and* $\ell > 2$ *we have* $H_F(\nu) < \frac{\delta}{f}$.

Proof. From Theorem 2.4.1 we now that $\ell = m + kf - nf - 2\log(\frac{1}{\epsilon}) + 2$. Let $A = \ell + 2\log(\frac{1}{\epsilon}) - 2 = m + kf - nf$. From the conditions above we now that $A > 0$ and therefore $m + kf - nf > 0$. Let $R = k/n$ which yields $(\delta + Rf - f)n > 0$ and therefore $R > 1 - \delta/f$. Using the sphere packing bound where $k/n = R \leq 1 - H_F(\nu)$ we immediately get $H_F(\nu) < \frac{\delta}{f}$. □

As it is impossible to calculate the min-entropy for a physical source we will estimate this quantity over the symbols of x. The bound given above will give us an idea whether the min-entropy in the symbols of x will be sufficient to handle the measured noise rate. This will be useful in the next chapter when we deal with a real physical source.

We finish this section be relating the review above to the extraction techniques discussed in the previous section. For the halfspace technique the output will be a binary string. Using Assumption 2.3.1 we showed that for any two physical objects $S_1, S_2 \in_{\mathcal{H}} \psi(D)$ we have $d(Z_1, Z_1') < d(Z_1, Z_2)$ except with a negligible probability. We also know that the entropy in the output of the halfspace technique will be n. By using any reading from a physical object to form the template w produced by Gen and by setting the distance separating $d(Z_1, Z_1')$ from $d(Z_1, Z_2)$ to t, the fuzzy extractor in the above construction will work properly except with a negligible probability.

To show how the threshold technique can be properly used with a fuzzy extractor is slightly more complicated. For clarity we will postpone this discussion to Section 3.4.1 where we will be working with a real application.

2.5 Summary

In this chapter we studied Gaussian PVs and discussed discretization techniques which can enable using PVs as IPVs. We thoroughly derived the error and collision probability for the new threshold technique. We also provided a review of fuzzy extractors which will become quite useful in the next chapter.

Chapter 3

CD Fingerprinting

In this chapter we introduce a new technique for extracting unique fingerprints from identical CDs. The proposed technique takes advantage of manufacturing variability found in the length of the CD lands and pits which will be the PV used in this chapter. Although the variability measured is on the order of 20 nm the technique does not require the use of microscopes or any advanced equipment. Instead, the electrical signal produced by the photo detector inside the CD reader will be sufficient to measure the desired variability. We thoroughly investigate the new technique by analyzing data collected from 100 identical CDs and show how to extract a unique fingerprint for each CD. We also give specific parameters and code constructions to convert the derived fingerprints into 128-bit cryptographic keys. Finally, we outline a simple protocol which utilizes the CD fingerprint to provide IP protection for the software stored on the CD without taking away from the users ownership.

The remainder of this chapter is organized as follows. In the next section we present some background information. In Section 3.2, we discuss the physical aspects of CD storage, the sources of manufacturing variability and the theoretical model capturing the CD variability. Section 3.3 presents experimental data to verify our theoretical model. In Section 3.4 we discuss the fingerprint extraction technique and

determine the parameters necessary for key generation. We discuss the robustness of the fingerprint in Section 3.5 and we propose a simple protocol for usage in Section 3.6.

3.1 Background

According to a study by the Business Software Alliance about 35% of the global software market, worth $141 Billion, is counterfeit. Most of the counterfeit software is distributed in the form of a compact disc CD or a digital video disc DVD which is easily copied and sold in street corners all around the world but mostly in developing countries. Given the severity of the problem in hand, a comprehensive solution taking into account the manufacturing process, economical implications, ease of enforcement, and the owner's rights, needs to be developed. While this is an enourmous undertaking requiring new schemes at all levels of implementation, in this work, we focus on secure fingerprinting techniques for optical media.[1]

To address this problem the SecuRom technology was introduced by Sony DADC. SecuROM uses the location information of hidden markers (faulty bits) to uniquely identify a key. The technology links the identifiers produced to executable files which may only be accessed when the CD is placed in the reader. The main advantage of this technology is that it can be used with existing CD readers and writers. While the specifics of the scheme are not disclosed, in practice, the technology seems to be too fragile, i.e. slightly overused CDs become unidentifiable. Another problem is at the protocol level. The digital rights management (DRM) is enforced too harshly, therefore significantly curtailing the rights of the CD owner.

In this chapter we take advantage of CD manufacturing variability in order to generate unique CD fingerprints. As discussed in the previous chapter the approach

[1]We shall refer to all optical discs including compact discs, digital video discs (DVDs), and Sony©Blue-Ray discs as CDs for the remainder of this chapter.

of using manufacturing variability to fingerprint a device or to build cryptographic primitives has been applied in several contexts.

3.2 Pits and Lands

In this section we briefly describe how data is stored on a CD. We also present our model to describe the dimensions of the CD features. On a typical CD data is stored as a series of lands and pits formed on the surface of the CD. The pits are bumps separated by the lands to form a spiral track on the surface of the CD. The spiral track starts from the center of the CD and spirals outward. It has a width of about 0.5 μm and a 1.6 μm separation. The length of the land or pit determines the stored data. The encoding length can assume only one of nine lengths with minimum value in the range 833 to 972 nm up to a maximum of 3054 to 3563 nm with increments ranging from 278 to 324 nm. Note that the range is dependent on the speed used while writing the CD. To read the data on the CD the reader shines a laser on the surface of the CD and collects the reflected beam. When the laser hits the pits it will reflect in a diffused fashion thus appearing relatively dark compared to the the lands. Upon the collection of the incoming beam, the reader can deduce the location and length of the lands and pits which results in reading the data on the CD.

CDs are written in two ways, pressing and burning. In pressed CDs a master template is formed with lands and pits corresponding to the data. The master template is then pressed into blank CDs in order to form a large number of copies. In burned CDs, the writing laser heats the dye layer on the CD-R to a point where it turns dark, thus reflecting the reading laser in a manner consistent with physical lands. Note that the burned CDs will not have physical lands and pits but will act as if they had these features. Figures 3.1 and 3.2 show the lands and pits of a pressed CD. We captured Figure 3.1 using an optical microscope and Figure 3.2 using a scanning

electron microscope.

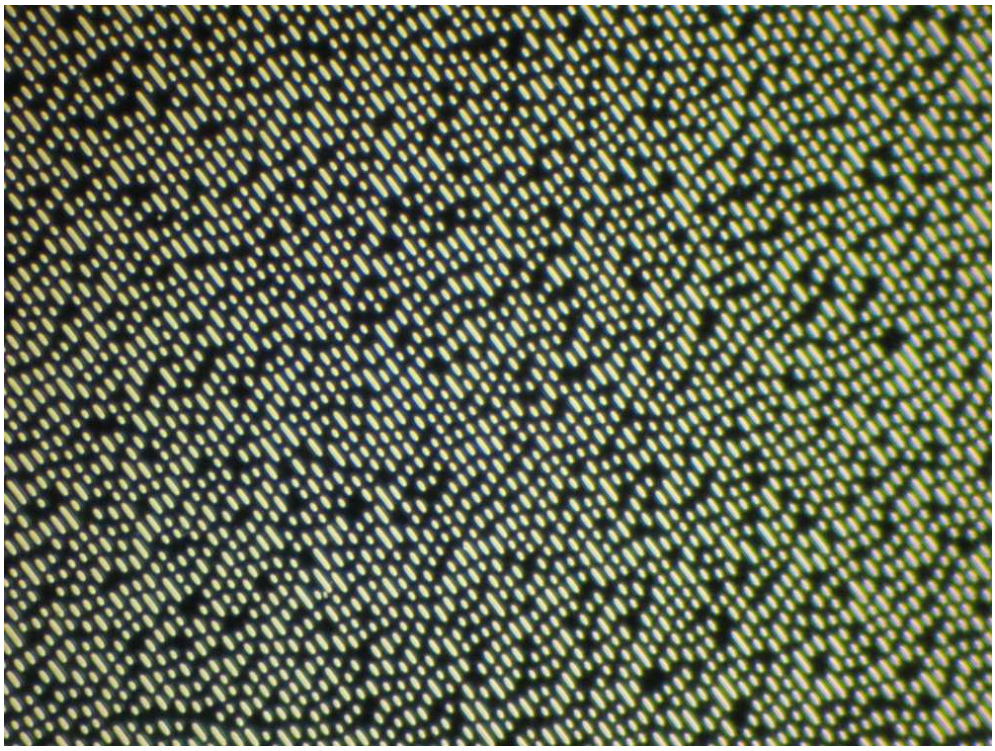


Figure 3.1: Lands and pits (Optical Microscope)

3.2.1 Source of Variation

Similar to any physical process, during the writing process CDs will undergo manufacturing variation which will directly affect the length of the lands and pits. For burned CDs this variability will be a direct result of the velocity of the CD while writing takes place. This velocity is assumed to be at a fixed rate between 1.2 and 1.4 m/s where the velocity variation during writing should be within $\pm 0.01 m/s$ [22]. Pressed CDs are manufactured by molding thermoplastics from a micro or nanostructured master prepared by lithographic methods. The molding process itself is optimized for replication fidelity and speed, and typical replication variations are on

the order of tens of nanometers [88]. The molding process involves contacting the thermoplastic with the master slightly above the glass transition temperature of the material, with a preset pressure for a brief amount of time, cooling the master and the thermoplastic to below the glass transition temperature and demoulding. Local variations of polymer material's mechanical and thermal properties, local variations of the temperature and pressure all potentially lead to variations in the imprinted structures. The thermal stresses induced during cooling and demoulding also potentially lead to variations. In this chapter we aim at using the small variation in the length of lands and pits in order to form a unique fingerprint for each CD. In the next section we characterize the length features of lands and pits.

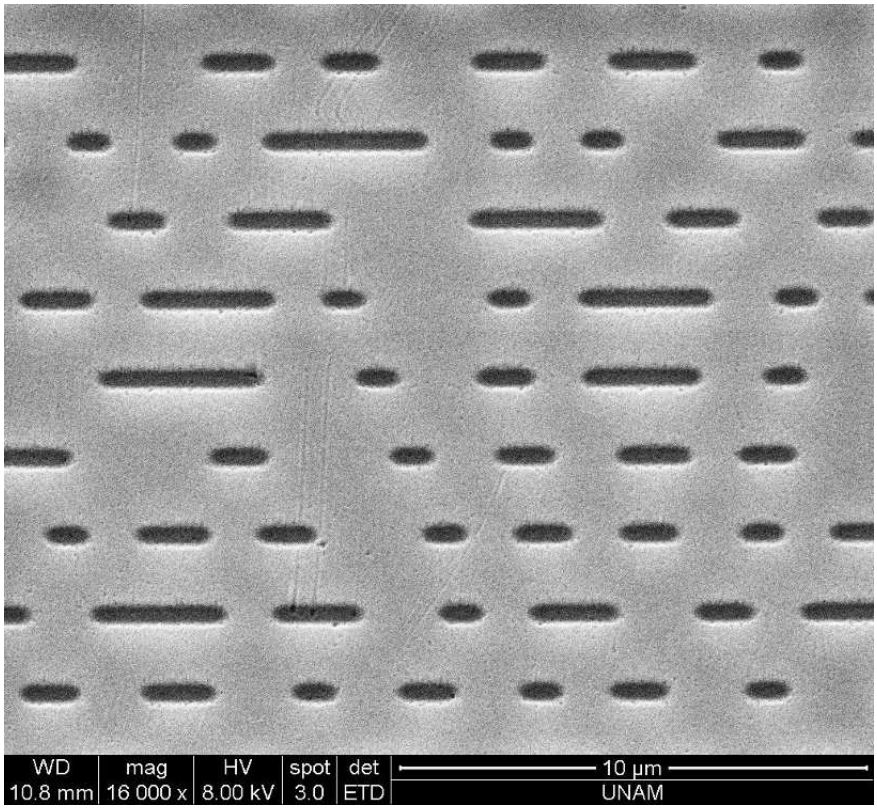


Figure 3.2: Lands and pits (SEM capture)

3.2.2 Single Location Characterization

Together lands and pits form the full spiral track. Therefore, it makes sense to fingerprint only lands or pits. The length of both lands and pits will follow similar distributions which is why we will simply use the term *location* to refer to either of them. We label the lengths of k consecutive locations by starting from a reference point on the track, as $L_1, L_2, \ldots, L_n$. In the ideal setting $L_i = c_i \cdot L$ for a small constant integer $c_i \in [3, 4, \ldots, 11]$ and $L \approx 300\ nm$. However, due to the subtle variations we talked about in the previous section we expect $L_i = c_i \cdot L + \ell_i$. The variable ℓ_i is expected to be quite small compared to L_i, and therefore difficult to measure precisely. Still our measurements should be centered around the ideal length. Hence, quite naturally across all identical CDs we model L_i as a random variable drawn from a Gaussian distribution $\mathcal{H}_i = N(M_i, \Sigma^2)$ where $M_i = c_i \cdot L$ and Σ denotes the mean and the standard deviation respectively. The length L will be the PV of this chapter. Note that in the language of the previous chapter, the length of each location (out of n CD locations) can be treated as an entry in the PV.

Here we are assuming that regardless of the location the standard deviation Σ will be the same. This is a quite a realistic assumption since Σ is essentially capturing the manufacturing variability which should affect all locations similarly. The more precise the manufacturing process is, the less of a standard deviation we would expect $\mathcal{H}_i$ to have. A perfect manufacturing process would have $\Sigma = 0$ and would therefore give all CDs the same exact length of a specific location across all identical CDs. On the other hand, for better identification of CDs we would like $\mathcal{H}_i$ to have a relatively large Σ.

In a typical CD reader, the reading laser is reflected from the CD surface back into a photo diode which generates an electrical signal that depends on the intensity of the reflected laser. Therefore, the electrical signal is expected to depict the shape of the CD surface. If these electrical signals are used to measure the length of any given

location, we expect these measurements to have a certain level of noise following a Gaussian distribution. So for location i on CD_j we denote this distribution by $\mathcal{D}_{ij} = N(\mu_{ij}, \sigma^2)$. The noise in the length measurements is captured through the standard deviation σ. Since this quantity mainly depends on the readers noise we assume that its the same for all CDs and CD locations. Contrary to Σ, to identify different CDs using the length information of CD locations we would like to see a relatively small σ. The theoretical model we use here is similar to Assumption 2.3.1. Therefore, we will be using some of the equations derived in the previous chapter.

3.3 Experimental Validation

To validate the theoretical model, we conducted an extensive experiment on a number of CDs. We directly probed into the electrical signal coming out of the photo diode constellation inside the CD reader. The intensity of this signal will reflect the CD surface geometry, and therefore can be used to study the length of the CD locations. To sample the waveform we used a 20 GHz oscilloscope. The data was read from the same locations on the same CD. Each CD was read a number of times in order to get an idea of the actual $\mathcal{D}$ distribution. Similarly, multiple identical CDs (about 100 CDs) were read from the same locations in order to generate the $\mathcal{H}$ distribution. Each collected trace required about 100 MB of storage space. Moreover, synchronizing the different traces to make sure that the data was captured from the same location of the CD was quite a challenge. We had to assign a master trace which represented the locations we were interested in studying and then ran the other traces through multiple correlation stages with the master to finally extract synchronized signals from the same locations on different CDs. Automating the process in order to accurately capture this massive amount of data was a time consuming challenge. However, we note that all this work would be almost trivially eliminated if we had access to the

internal synchronization signals of the CD reader chip. The captured signals were then further processed, using Matlab, in order to extract the location lengths and obtain the distributions. After processing, we extracted the length of 500 locations (lands) on the CDs. We used commercially pressed CDs for all the experiments reported in this chapter[2].

Figure 3.3 shows the histogram of lengths extracted from 550 reads for a randomly chosen location on one CD. The mean length of the histogram is about $\mu_{ij} = 958$ nm. This histogram captures the $\mathcal{D}$ distribution. The other locations observe similar distributions with different mean lengths which will depend on the encoded information. When considering data coming from different locations and different CDs we obtain a Gaussian distribution with $\sigma = 20\ nm$ (with an average standard deviation of 2 nm on σ). This will be a good estimate for the noise observed during probing of the electrical signals. These results verify the assumption that the noise in the electrical signal can be approximated as Gaussian noise. Note that with Gaussian noise simple averaging can be used to substantially reduce the noise level.

As we are interested to see the behavior of the location lengths across different CDs we next shift our attention to two CDs before we look at a larger batch of CDs. Figure 3.4 captures a histogram for the length of the same location on two identical CDs. What is important here is the distance between the two Gaussian distributions. The larger this distance becomes the easier it is to identify CDs. Our basic thesis for fingerprinting CDs is that the length of a single location will vary across multiple identical CDs. As pointed out earlier, this behavior can be modeled with the Gaussian distribution $\mathcal{H}_i$. The histogram in Figure 3.4 captures this for two CDs. To generalize these results we need a larger sample space to estimate the $\mathcal{H}_i$ distribution. The major problem here is that each data point needs to come from

[2]We have verified a similar behavior for burned CDs. Not surprisingly, the data from the burned CDs had a much larger variation and was easier to analyze.

a different CD. Therefore, to obtain a histogram which clearly depicts a Gaussian we would need to test on the order of 500 CDs. This was not possible as each CD required substantial time, computing power and storage space in order to produce

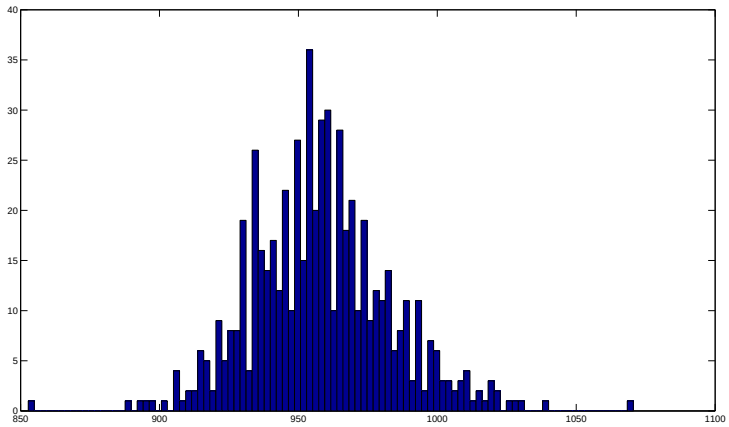


Figure 3.3: Histogram of reads coming from the same location on the same CD

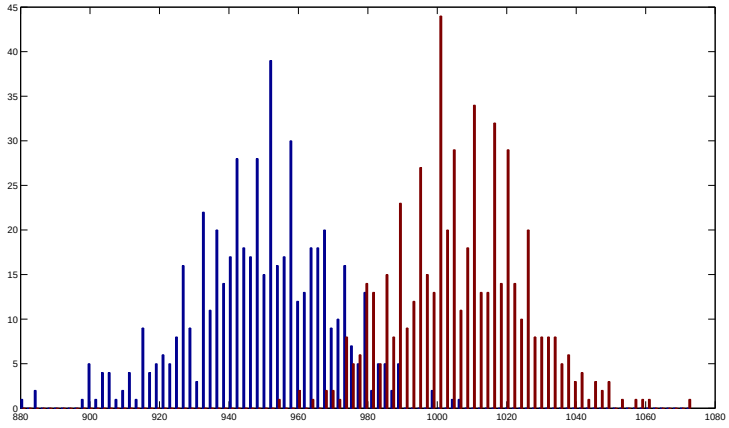


Figure 3.4: Histograms for reads coming from the same location on two CDs with identical data

final data points. However, we were able to carry out this experiment for about 100 CDs. Each CD was read about 16 times to reduce the noise. Finally, we extracted the lengths of 500 locations for each of the CDs. Figure 3.5 depicts the histogram over 100 CDs for a randomly chosen location out of the 500 extracted locations.

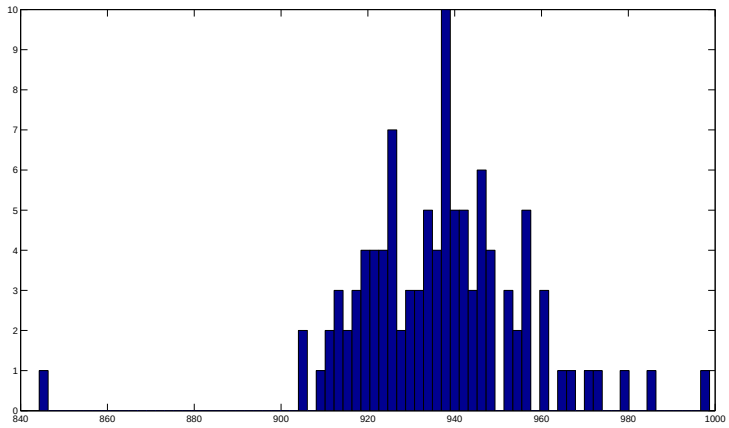


Figure 3.5: Histograms of reads coming from the same location on 100 identical CDs

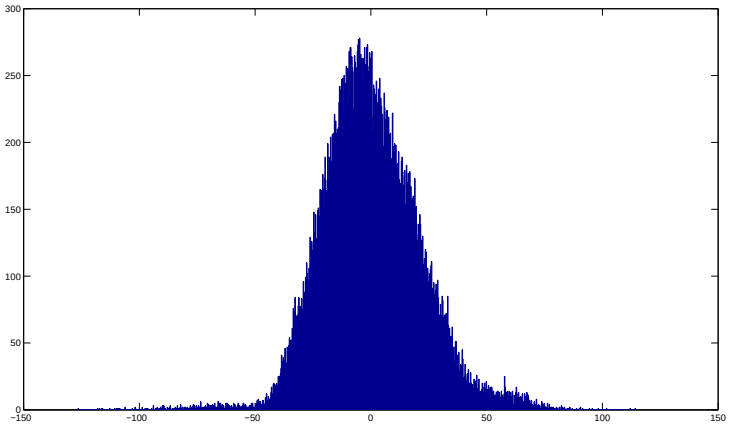


Figure 3.6: Histograms of reads coming from the 500 locations on 100 identical CDs

The histogram in Figure 3.5 has a mean of about 940 nm. As mentioned earlier, for all locations Σ had a mean of 21 nm with a standard deviation of 1.8 nm. The previous histogram looks similar to a Gaussian distribution generated from 100 data points. However, it would be interesting to get a confirmation that with more data points the above histogram would actually yield a Gaussian. To do so, we normalized the lengths of each location by subtracting the average length for that particular location. Since the distribution for each location had roughly the same Σ the normalization process effectively made all these distributions identical with a mean of 0 and a standard deviation of Σ. We then collected all these data points (on the order of 50,000 points) and plotted the corresponding histogram. This is shown in Figure 3.6. The histogram of Figure 3.6 strongly supports our thesis of normally distributed location lengths across different CDs. One might observe a slight imbalance on the positive side of the Gaussian. This behavior seems to be a result of some of the CDs having a DC offset. Fortunately, this will not pose a problem for our fingerprinting technique as we will be normalizing each batch of data to have a mean of zero, thus removing any DC components.

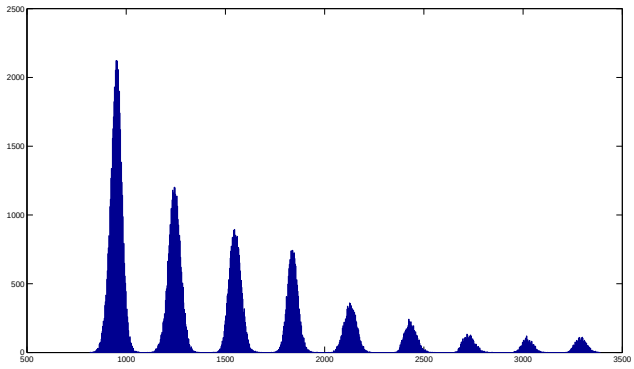


Figure 3.7: Histogram of location lengths using the electrical signal

We finish this section by showing the histogram in Figure 3.7. The main purpose

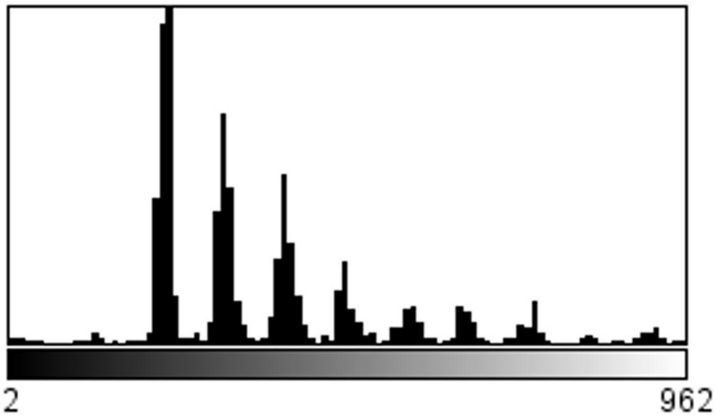


Figure 3.8: Histogram of location lengths using microscope images

of this histogram is to confirm that what we are studying is in fact the length of data locations written on the CD. We elaborated earlier that the data on a CD is stored in discrete lengths ranging from about 900 nm to about 3300 nm taking 9 steps in increments of about 300 nm. We build the histogram in Figure 3.7 using the data collected from 500 locations over the 100 CDs without normalizing each location's length to zero. In Figure 3.8 we show the same histogram with data extracted from scanning electron microscope images via image processing.

3.4 CD Fingerprinting

There are many challenges in deriving a robust and secure fingerprint. One important issue is the reading noise. Similar to a human fingerprint, we saw in the previous section that the readings used to extract the CD fingerprint are inherently noisy. The extraction of a deterministic and secure fingerprint from noisy data has been previously studied in the literature [49, 48, 20]. Most relevant to our work is the fuzzy extractor technique [20] which we have discussed in the previous chapter. For the remainder of this section we will discuss how fuzzy extractors can be used in the

CD setting. Moreover, we will discuss the experimental results and present various bounds needed to achieve high levels of security.

3.4.1 Fingerprint Extraction

In the previous section we described how the empirical data suggests that every CD has unique location lengths. These location lengths as can be seen from Figure 3.7 will have different values depending on the encoded information. Moreover, we discussed earlier that the raw data measured from the electrical signal will sometimes have different DC offsets. Therefore, it is important to process the data before the different locations can be combined together in order to produce the final fingerprint x. The first step in processing the data coming from every location on every CD is to remove the signal noise. To achieve this, the length of every location on a CD is averaged over a number of readings. Since we are assuming Gaussian noise, the noise level σ will scale to $\sigma/\sqrt{a}$ where a is the number of readings used for averaging. Next, we normalize the data using the ideal average of each location. As the ideal location lengths are discretized it becomes easy to find the ideal length for every location and subtract it from the measured lengths. This will guarantee that all location lengths have similar distributions as we saw in Figure 3.6. Finally, to remove the DC component we need a second normalization step. We subtract the mean of the reading coming from different locations of the same CD. Figure 3.9 and 3.10 show the lengths of 500 locations for two identical CDs after being averaged and normalized. Each figure contains two traces each originating from the same CD but read at different times. The vertical axis represents the variation in nanometers from the ideal length of that location. It should be easy to see that identical CDs will have different fingerprints.

We still need to outline a technique to extract a final fingerprint. Even after the previous averaging and normalization steps we will still have errors in the length read-

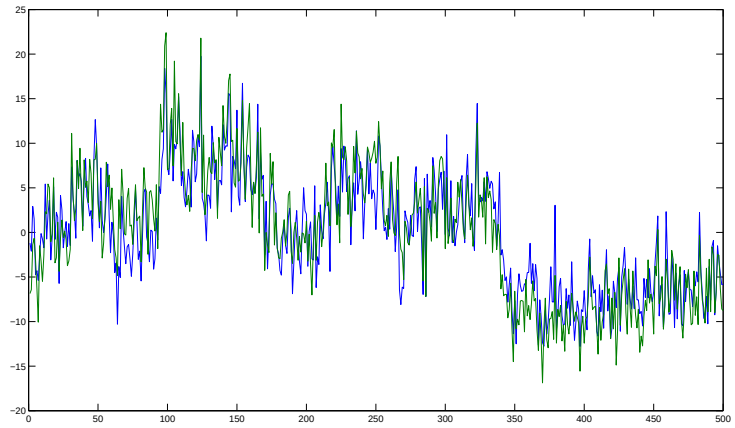


Figure 3.9: Length variation, 500 locations, CD1

ings. Although we will be using a fuzzy extractor to correct the errors, the biggest challenge towards achieving an efficient extraction technique will be the nature of this error. The noise is Gaussian over the real values of the lengths. This means that even when the data is discretized the error will manifest itself more as a shift error from the ideal length rather than a bit flip error. Unfortunately, the hamming metric does not naturally accommodate for this kind of errors. Moreover, if we assume that every location length of the CD will be a symbol in the extracted fingerprint, then the error rate would be very high as it is very difficult to get the same exact length for the CD locations. A more natural distance metric in this situation would be the Lee metric [67]. However, this might require substantial changes to the fuzzy extractor construction outlined earlier. To solve this problem we proposed the *threshold* technique in the previous chapter. Here we will elaborate on how the threshold scheme can be combined with the fuzzy extractor of Theorem 2.4.1. A formulation of the threshold scheme is shown in Table 3.4.1.

The scheme works as follows. We label the normalized readings from n locations on a CD (or any source of these variables) by $z \in \mathbb{R}^n$, where $z = z_n \dots z_1$ and the

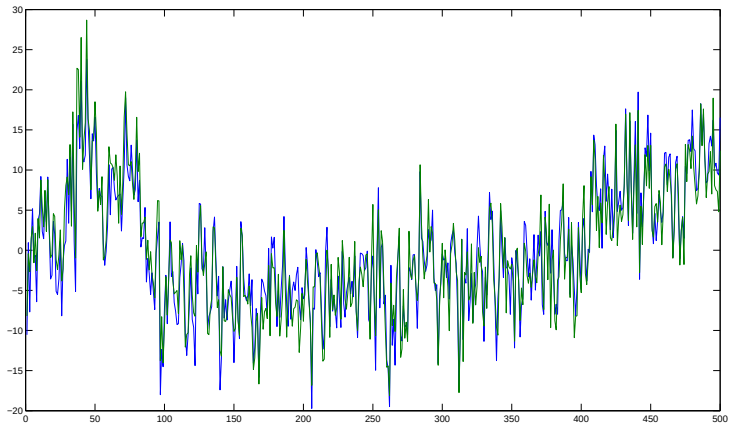


Figure 3.10: Length variation, 500 locations, CD2

symbols of z are bounded such that $-2^{u-1} \leq z_i \leq 2^{u-1} - 1$ for some integer u. It is also possible to scale the data to fit into a certain region. However, this might lead to a blow up in the error rate. Each z_i is shifted to the range $0 \leq z_i \leq 2^u - 1$ and then rounded to integers and discretized using simple unsigned binary encoding. If there is a need to use l of the bits in the fractional part then the readings are first multiplied by 2^l before rounding. Now $z_i \in \{0,1\}^{u+l}$. As explained earlier new readings of z_i will result in a noisy version z_i' which is a shift from the original z_i. If z_i is defined to be a symbol then any slight noise in z_i' will result in counting that symbol as an error. This is where the threshold scheme comes to play. To allow some room for the reading to move without being considered as an error we define the threshold $\tau = 2^s$ where $s < u + l - 1$. Any reading z_i' is considered to be an error only if $|z_i - z_i'| > \tau$ (threshold distance of 1). This seems pretty useful and straightforward as we can now treat errors natural to the Lee metric as errors in the hamming metric. However, it is not clear how we can get the fuzzy extractor of Theorem 2.4.1 to correct the cases when $|z_i - z_i'| \leq \tau$. To get this scheme to work we need to sacrifice some of the bits in z_i. Specifically we break z_i into two parts. The template part $z_{1,i} \in \{0,1\}^{u+l-s-1}$

which contains the most significant $(u + l - s - 1)$ bits of z_i. And the noisy part $z_{2,i}$ containing the remaining bits of z_i where , so $z_i = z_{1,i}|z_{2,i}$. The field used in the error correcting code $\mathcal{C}$ is only defined over the template part of z_i. Therefore, $\mathcal{F} = \{0,1\}^{u+l-s-1}$ and $f = u + l - s - 1$. Using the fuzzy extractor explained in the previous chapter for the fingerprint $x = x_n \dots x_1$ we will have $x_i = z_{1,i}$. However, the noisy part of z_i will also need to be kept in order to help correct for the noise. To generate (k, w, v) the GEN algorithm now works as follows. It takes in $x_i|z_{2,i}$ for $i \in [1, \dots, n]$ and then choses a random code word $c = c_n \dots c_1 \in \mathcal{F}^n$ and computes $w = w_n \dots w_1$ where $w_i = (x_i|z_{2,i}) + (c|\tau)$. As before v is chosen uniformly at random and $k = h_v(x)$ where $h \in H$ is a universal hash function. The recovery algorithm REP will now take in a noisy version of z which is z'. To recover x, REP computes $C_i = w_i - z_i = (x_i|z_{2,i}) + (c|\tau) - (z'_{1,i}|z'_{2,i})$. Now since $x_i = z_{1,i}$ we can rewrite this as $C_i = (z_i - z'_i) + (c_i|2^s)$. If $z_i - z'_i < -2^s$ or if $z_i - z'_i > 2^s - 1$ then there will be no carry from the template part to the noisy part of w_i, and therefore $c'_i = c_i$. In any other case c_i will change therefore resulting in an error in c'_i. Upon retrieval of c' the error correcting code will be able to correct it back to c given that less than t of the symbols were out of the threshold bound. Following is a formulation of the above discussion.

The threshold τ solves the error correcting problem with respect to the Lee distance. In particular, τ helps control the error rate which arises when treating the real values as symbols over some field. Without a threshold scheme ($\tau = 0$), the error rate will be very high. On the other hand if τ grows too large then the error rate will be low. However, the collision rate between the CDs will start to increase thus decreasing distinguishability between CDs. We have already computed the error rate and the collision rate of the threshold technique in Equations 2.4 and 2.6. These equations will hold for the CD case if Assumption 2.3.1 holds.

To verify the error probabilities given in Equations 2.4 and 2.6 we used the data

Threshold Scheme: (GEN,REP) parameterized by $M, m, \ell, t, \epsilon, l, \mathcal{C}, H, \tau = 2^s$
GEN: $(k, w, v) \leftarrow \text{GEN}(\text{CD}j)$
1. Obtain (a) samples for the length of each of the n locations on CDj.
2. Generate $z = z_n \dots z_1$:
a. Average the lengths over a samples,
b. Subtract the ideal mean from the averaged reads,
c. Normalize the sequence to have a zero mean and set that to z.
3. Find u such that $-2^{u-1} \leq z_i \leq 2^{u-1} - 1$ for all i, then shift z_i to $0 \leq z_i \leq 2^u - 1$.
4. Take the binary representation of z_i, shift left by l bits and round to an integer.
5. Form $z_{2,i}$, the lowest $s+1$ bits of z_i, and set $x_i = z_{1,i}$ to be the remaining bits of z_i.
6. Set $x = x_n \dots x_1$ to be the fingerprint template.
7. Choose a random code word $c \in \mathcal{C}$, such that $c = c_n \dots c_1$.
8. Compute $w_i = (x_i \| z_{2,i}) + (c | \tau)$ and form $w = w_n \dots w_1$.
9. Randomly choose v to compute $k = h_v(x)$ where $h_v \in H$, and output (k, w, v).
REP: $k \leftarrow \text{REP}(\text{CD}j, w, v)$
1. Generate z' as in GEN.
2. Set c'_i to be the the highest $u + l - s - 1$ bits of $w_i - z'_i$.
3. Use $\mathcal{C}$ to correct $c' = c'_n \dots c'_1$ to $c = c_n \dots c_1$.
4. Compute $x_i = w_i - c_i$.
5. Form $x = x_n \dots x_1$ and return $k = h_v(x)$.

Table 3.1: Formulation of the threshold technique for CD fingerprint extraction

collected from 100 CDs. In this data there was no averaging over the samples and therefore σ was not changed. The data we have uses 500 locations on each CD. Under the assumption that these errors will occur independently across the location then the expected value of the number of errors should be $500P_e$. Similarly, the number of collisions observed assuming the independence of the extracted symbols should be $500P_c$. Figure 3.11 plots the expected theoretical (circle) and the observed (square) number of errors both as a function of $s = \log(\tau)$. One can see that the theoretical model does a good job predicting the behavior of the experimental data. Also, this is a verification on the error independence assumption. Figure 3.12 shows the expected theoretical (circle) and the observed (square) number of collisions between the extracted strings as a function of s. Naturally the theoretical model expects a slightly lower number of collisions due to the independence assumption. As we will discuss in the coming section there will always be a level of dependency between different locations in the CD which will result in a higher collision rate.

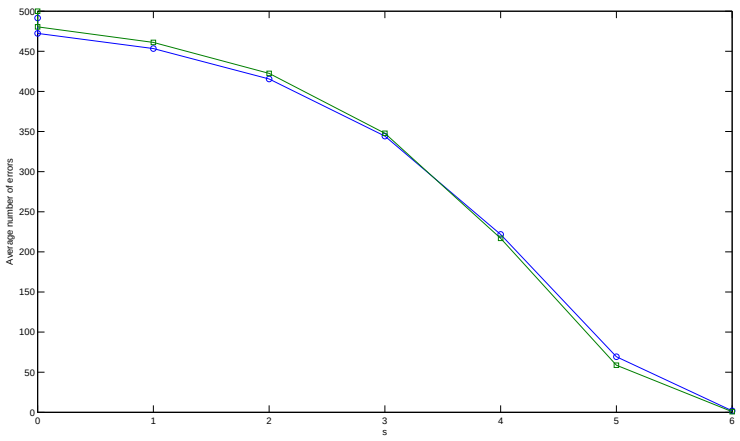


Figure 3.11: Number of errors

To have useful fingerprints, $1 - P_c$ has to to be significantly larger than P_e, otherwise the CDs will start looking alike. Without averaging, σ results in a high error

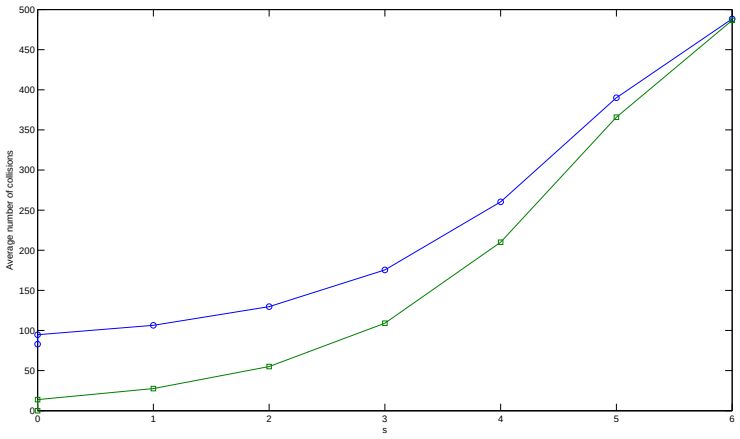


Figure 3.12: Number of collisions

rate which is larger than $1-P_c$. Therefore it becomes important to average before extraction. Also we saw in Proposition 2.4.2 that the entropy of the error rate imposed a lower bound on the amount of entropy needed to have a realistic fuzzy extractor scheme. This is another reason to push for a low error rate. In the next section we discuss entropy and error rate estimations.

3.4.2 Entropy Estimation and 128-bit Security

The previous sections dealt with the theoretical aspects of extracting the CD fingerprint. In this section we take more of an experimental approach where we are more interested in computing actual parameters. The most important parameters that we need to estimate are the entropy of the source (the CD variability) and the noise level. The first and hardest task here will be to decide the amount of entropy generated by the source. In [38] and [47] the authors use a universal source coding algorithm in order to estimate the secrecy rate. In particular it was proposed to use the Context-Tree Weighting Method (CTW) [100]. What is quite useful about

the CTW algorithm is that in [99] it was shown that for any binary stationary and ergodic source X, the compression rate achieved by CTW is upper bounded by the min-entropy $H_\infty(X)$ as the length of the input sequence approaches infinity. This is a good indication about the entropy produced by the source provided enough bits are fed to the algorithm. To apply this algorithm to our setting we start by using the data coming from the 100 CDs. On each CD we collected data from 500 locations and processed the data with a threshold value of $\tau = 2^2$. The final data came out to be in the range $[0, 2^5 - 1]$ and we did not use any fractional bits so $l = 0$. With these parameters the size of the symbols was $f = 2$. This means that every CD produced 1000 bits. The data was fed into the CTW algorithm which resulted in a compression rate of about 0.83 bits of entropy per extracted bit. Recall here that these samples were not averaged over multiple reads. Therefore the error rate is quite high. When we averaged over 16 samples the combined entropy became 0.71. This is expected since the noise will add to the entropy. In order to get a more precise estimate for the min entropy we decided to average over 225 reads. With this many reads we had to restrict our samples to only 14 CDs as the amount of data quickly becomes large. With the new sample the compression rate of the CTW algorithm was about 0.675 which seemed to be a good estimate of our min-entropy. For this sample the average error rate is $P_e = 8\%$. That is, on average about 40 of the 500 locations will fall out of the threshold region. On the other hand the collision probability is about 46%.

We can use the result in Proposition 2.4.2 which suggests that for a noise rate of 0.08 and $f = 2$ the entropy of the source should be at least 0.40 which translates in $\delta = 0.8 < 1.35$, and therefore we conclude that we have enough entropy in our source. However, with this level of entropy we are placing stringent conditions on R, i.e. the rate of the error correcting code.[3] To relax the restriction on the code

[3]Recall from the prof of Proposition 2.4.2 that $R \geq A/nf + (1 - \delta/f)$ for a security level of at least $A = \ell + 2\epsilon - 2$.

rate we took a closer look at our source bits. Ideally the two bits would have the same entropy. However looking at Figure 3.9 and 3.10 and multiple similar figures we clearly see that there is a degree of dependency between the adjacent locations. There is a low probability of a sharp change of the length variability from one location to its neighbor. With this observation we would suspect that the most significant bit will have less entropy as it is less likely to change across adjacent locations. To verify this we applied the CTW algorithm to each of the two extracted bits separately. For the most significant bit the entropy for the above three cases of no averaging, averaging over 16 reads, and averaging over 225 reads we found $1, 0.9, 0.6$-bits of entropy, respectively. When we repeated this process for the least significant bit we obtained $1, 1, 0.98$-bits of entropy, respectively. Clearly, we have more entropy in the least significant bit. It seems reasonable to only use the least significant bit to form the fingerprint and the final key. This would effectively increase the entropy of our source while very slightly affecting the error rate and the collision rate. For this least significant bit scheme we obtained $P_e = 8\%$ and $P_c = 0.46\%$.

With the parameters we have, $P_e = 0.08$ and $\delta = 0.98$ and $f = 1$ we can build a fuzzy extractor which can extract secure keys from CD fingerprints. For a 128-bit key we set $\ell = 128$. Similarly, to achieve a fuzzy extractor output which reveals very little information about the fingerprint we set $\epsilon = 64$. Using the equation of Theorem 2.4.1 we require that the error correcting code in the fuzzy extractor should satisfy $k \geq 190 + 0.02n$. Note that although $P_e = 0.08$, this is the expected error rate. For a practical scheme we require the fuzzy extractor to correct around a 0.17 error probability. These parameters can now be satisfied using a binary BCH code of $[255, 45, 88]$. More specifically, we define a code word containing 7 code words of this BCH code, which will make $n = 1785$. With this construction the failure probability[4] P_{fail} will be on the order of 10^{-6}. Note here that treating the 7 code words separately

[4]Here, $P_{fail} = 1 - \left(1 - \sum_{i=0}^{t=43} \binom{n}{i} P_e^i (1 - P_e)^{n-i}\right)^7$.

to generate separate parts of the key would significantly decrease ϵ but will decrease the failure probability. Therefore, in our failure probability we treat the 7 code words as a single entity. As we noted earlier, our data suffers from higher error rates due to the external connections which we used. With an on-chip process we can expect the error rate to drop significantly.

3.5 Robustness of the Fingerprint

We mentioned earlier that a CD fingerprint can be used to tie software licenses to individual CDs where the software is stored. Under this use scenario it becomes important to address the robustness of the fingerprint. In all our experiments the data collected came from locations in the same sector of the CD. In a real application readings would typically be collected from different sectors. Thus ensuring that a scratch or any physical damage to a specific location will not render the CD fingerprint useless.

Another important concern regarding the robustness of the fingerprint is that of aging. Although no quantitative estimate of fingerprint durability can be given within the scope of this work, mechanisms related to viscoelastic relaxation in optical disc patterns need to be discussed briefly. Optical discs are printed on polymeric substrates, which have glass transition temperatures typically above 150 C. The viscosity of such materials are temperature dependent and governed by an Arrhenius type exponential temperature dependence, described by an activation energy defined by the glass transition temperature. In its simplest form, the Arrhenius model assumes that the rate of change is proportional to $e^{\frac{-E_a}{kT}}$ where E_a is the activation energy, k is the Boltzmann constant (an invariant physical parameter), T is the absolute temperature (temperature in degrees Kelvin). Even at lower temperatures (natural operating and storage temperature range of the optical disc), viscosity of the polymer remains

finite. During the molding process, most of the internal stresses are relieved upon cooling, therefore resulting in fluctuations in the nanoscale structure of the bit patterns. The pressed discs have a thin metal coating, which is typically coated on to the polymer disc by evaporation or sputter coating, that results in the increase of the surface temperature by up to 50 C. This process is also likely to be a source of local thermoelastic stress buildup which relaxes over the lifetime of the CD. In a first order approximation, the disc material can be thought of as a Kelvin-Voigt material, and creep relaxation can be approximated by a single time-constant exponential behavior. In such a case, most of the viscoelastic relaxation will occur at the early stages of disc production, and latter time scales will have less of an effect. It may be speculated that the fingerprints due to length fluctuations of 25 nm upon 300nm characteristic bit length will persist within at least 10% of the CD lifetime, which is predicted to be 217 years at 25C 40% Relative Humidity conditions. This gives an estimated 20 year lifetime for the fingerprint [91]. Due to the exponential dependence of the relaxation on time, by recording the signature on a slightly aged optical disc (months old), the persistance of the signature can be increased.

3.6 License Distribution Protocol

To prevent unauthorized copies from installing, many popular software titles employ a simple license checking scheme. In the simple license scheme the license key is delivered in printed form (on the CD or its case). The offline license distribution scheme can be summarized as follows: identical copies of an installation software are pressed on CDs. A unique license key is printed on the cover of each CD. The CDs are sold to customers. Customers run the installation software on their machines. During installation each customer enters the license key manually to the installer. The installer checks the validity of the license. If valid, the installer copies the software

to the target machine.

This scheme is convenient as no network connection is required to complete the installation process. However, the offline scheme has a number of flaws. The licenses are checked for membership in a the set of valid licenses. This means that licenses are *not* tied to CDs. In other words, the software on an authentic CD, can be trivially copied for a fraction of a dollar to another CD, which will install under any valid license key. One way to curb rampant CD counterfeiting is to require an online license registration step, where a central database is contacted after verification of the license which checks whether the number of valid installations are exceeded. If not, the software is installed and the central database is updated accordingly. While this simple online scheme is effective, it is inconvenient as it requires an online connection. Furthermore, the restriction on the number of installations is highly restrictive and unfair to paying owners of the CD. The main difficulty in binding a key to a CD is that CDs are pressed with one template since the production of the template bears significant cost. This means that CDs coming off a production line necessarily share an identical digital image. Here we propose to use the physical fingerprints of the CDs as an identifier. We bind this identifier to the license and thereby achieve unique licenses that are intrinsically tied to CDs without changing the manufacturing process.

The proposed protocol works in a manner similar to the offline distribution protocol. The license key here will be the helper data (w, v) along with the encrypted version of the key $E_K(k)$, where $E_K()$ is an encryption function. The secret key K is known to the installer and k is the unique key extracted from the CD. In this protocol the CD reader is expected to provide the fingerprint x from publicly known addresses on the CD. The user then feeds the installer with $(w, v, E_K(k))$. The installer runs the REP algorithm to retrieve k' and then checks whether $E_K(k') = E_K(k)$ is satisfied. This protocol is very simple and can clearly be bypassed if an attacker can force the reader to return fingerprints of his choosing. While this attack is simple it requires

modification to the reader of any user who wishes to use copied software. We finish here by pointing out that more secure protocols are possible using the CD fingerprint.

3.7 Summary

In this chapter we showed how to generate unique fingerprints for any CD. The proposed technique works for pressed and burned CDs, and in theory can be used for other optical storage devices. We tested the proposed technique with a 100 identical CDs and characterized the variability across the studied CDs. We also gave specific parameters and showed how to extract a 128-bit cryptographic keys. This work opens a new door of research in the area of CD IP-protection.

Part II

Applications of Physically Parameterized Functions

Chapter 4

PUFs

In this part of the dissertation we will shift our attention to physically parameterized functions (PPFs). PPFs are functions that use PVs as parameters. A special case of PPFs are physically unclonable functions (PUFs) which are the natural successor of POWFs. When PUFs were first introduced by Gassend et. al in [28] the authors suggested that the name POWF was misleading and argued that the examples given for POWFs did not really match the standard meaning of one-way functions, and therefore they suggested that these physical functions should really be called physical random functions or PRFs. However, the acronym PRF is historically known to stand for pseudo-random functions in cryptography. This lead the authors to use the term PUF to stand for physically unclonable function.[1] Unfortunately, due to the popularity of the name, several physical functions which do not satisfy unclonability have been labeled as PUFs. We argue that the term PPF is a more accurate description of most known PUF circuits. However, the problem we face in this part of the dissertation is that most of our discussion is focused on a specific PPF which is known as a delay-based PUF. This clearly causes confusion between the names used and the actual functionality. On one hand, the term PUF is not accurate as a description. On

[1] Another reason mentioned in [28] is that the acronym PUF is easier to pronounce.

the other hand, one needs to be consistent with the existing literature. As most of the work presented in this part of the dissertation has been originally written before we had a clear classification of PPFs, therefore we will keep the original notation which happens to be consistent with the existing literature.

In this chapter we will throughly explore delay-based PUFs and their constructions. The main job of this chapter will be to set the stage for PUFs before we can use them in a number of cryptographic schemes with the goal of creating a more secure PUF. The rest of this chapter is organized as follows. The next section provides a review of major PUF proposals. We briefly explain the idea of each proposal and why most of them do not qualify as unclonable functions. Section 4.2 describes delay-based PUFs in details in addition to reviewing its mathematical model and properties. Finally, Section 4.3 presents a variant implementation of delay-based PUFs built with tristate buffers.

4.1 Previous Work

Multiple designs have been proposed in the literature in an attempt to realize PUFs. Coating PUFs were introduced in [95]. The main idea there is to use a random mixture of particles with different dielectric properties in order to build a protective coating layer around the chip. Beneath the surface, a number of sensors are installed in order to measure the capacitance at different locations of the layer. Using the capacitance reading, which will depend on the chemical structure of the layer, a secret key specific to the hardware can be derived. The capacitance of the coating layer can be seen as a PV. The capacitance sensors will act as measurements of the PV. One might argue that it is difficult for an external attacker to replicate the coating layer, or provide a mathematical model governing the input/output behavior. However, with a limited number of sensors the behavior of the entire circuit can be captured in a

small lookup table. This last observation highlights the fact that coating PUFs are not really unclonable. Although the devices might be used to uniquely identify the circuit, the fact that the entire behavior of the circuit can be replicated undermines its ability to be an unclonable function. One great advantage of coating PUFs is that any attempt to reverse engineer the circuit is faced with the coating layer. The downside however, is that building the coating layer can be quite a costly task.

A different PUF proposal is the FPGA SRAM PUF which was introduced in [38]. This device uses the startup values of the SRAM cells in an FPGA circuit. Upon startup, the inverters inside an SRAM will have small voltage mismatches due to manufacturing variability. This mismatch will be amplified upon startup, thus resulting in a random startup value for the device. The PV here will be the small voltage mismatch inside the SRAM inverters. The startup of the device will act as a measurement producing the startup values of the SRAM cells. Similar to the coating PUF the small (polynomial) size of the SRAM cells enables an attacker to build a lookup table capable of simulating the SRAM PUF behavior. Therefore, SRAM PUFs cannot be considered unclonable. In [65] the authors propose butterfly PUFs which generalize the SRAM PUF idea to work on any FPGA even no intrinsic SRAM unit is available. The butterfly PUF offers the same level of security as SRAM-PUFs. Therefore, they also cannot be considered unclonable.

Delay-based PUFs also known as silicon PUFs were introduced and analyzed in [28, 27, 29, 71]. This specific PUF will be the center of our discussion in this chapter. Therefore, we postpone discussing the details of delay PUFs to the next section.

The usage of PUFs has typically been focused on the claim that they are unclonable. The relation between the challenges (input) and the responses (output) is usually assumed to be a property of the hardware that cannot be cloned. This is not always the case as we have seen. We will further elaborate on this point when we talk about delay-based PUFs. Even if we assume that the input/output behavior is not

clonable, this would mean that the user of the PUF needs to collect a large number of input/output pairs to be able to authenticate the circuit which is impractical. On some level there seems to be a contradiction here. On one hand, we would like for the input/output relation of a PUF to be un-modelable, while on the other hand, we do not want to store a massive input/output database. The solution as we will explore in future chapters will be in resorting to computationally hard problems. The next chapter will further elaborate on this point.

4.2 Delay-based PUFs

4.2.1 How PUFs Work

A delay-based PUF [27] is a $\{0,1\}^n \rightarrow \{0,1\}$ mapping, that takes an n-bit challenge (c) as an input and produces a single bit output (r). Figure 4.1 shows a multiplexer delay-based PUF circuit consisting of n stages of switching gates. Each switch has two input and two output bits in addition to a control bit. If the control bit of the switch is 0, the two inputs are directly passed to the outputs through a straight path. If on the other hand, the control bit to the switch is 1, the two input bits are switched before being passed as output bits. Therefore, the control bit of each switch will decide the path taken by the input signals. The n switches are serially connected so that the output of each switch is connected to the input of the following switch. The two outputs of the last switch are connected to a flip-flop, which is called the *arbiter*. The two inputs to the first switch are connected to each other, and then connected to a pulse generator.

From this point we will refer to delay-based PUFs as PUFs. We briefly describe the operation of PUFs as follows. When a challenge c is received each of its n bits is used as the control bit to one of the PUF switches. Next, a pulse is generated and sent to the first switch. The pulse will break into two pulses entering the upper

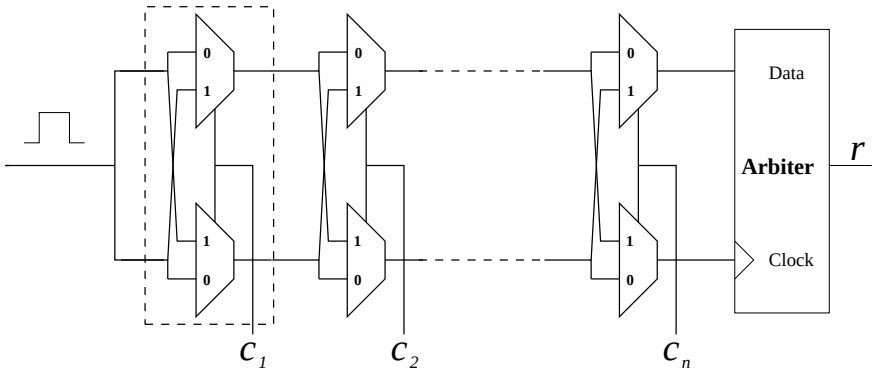


Figure 4.1: A multiplexer delay-based PUF circuit

and lower inputs of the first switch. Depending on the control bit of the first switch the two signals will either travel in a straight path or they will switch locations. Although the paths taken by the signals have been designed to have the same length, due to manufacturing variability the two paths will have a small mismatch. This mismatch will have a different value for each of the two possible paths of the two signals. Therefore, the two pulses will acquire a time delay between them which is dependent on the control bit of the first switch. The same argument applies for the rest of the n switches. Each challenge c will impose a different path on the n switches. Consequently, the total delay mismatch between the two signals will be a function of c. The job of the arbiter at the end of the switch chain is to indicate which signal arrives first. Recall that a flip-flop has two inputs, the data input and the clock input. When the clock input observes a rising edge the data input is captured at the output of the flip-flop. In a PUF setting, if the signal connected to the data input of the arbiter arrives first, the output of the arbiter will be 1. Otherwise, the output will be 0.

The output of the circuit will depend on the delay mismatch in each of the stages. The delay mismatch will be a small quantity which will be quite sensitive to external effects. A major advantage of this sensitivity is to prevent physical attacks on the system. Trying to tap into the circuit will cause a capacitance change therefore

changing the output of the PUF circuit. Removing the outer layer of the chip will have a permanent effect on these circuit variables and again, it will change the output of the PUF circuit. Briefly, we can say that a well-built PUF device will be physically tamper-resilient up to its sensitivity level.

4.2.2 Mathematical Model

In this section we derive a linear model for the delay-based PUF. In specific we show that sufficient challenge-response pairs one can build a mathematical model approximating the PUF behavior to a high degree. This will essentially undermine the idea of PUFs being an unclonable function.

A similar but slightly different linear model was derived earlier in [27]. In their model, $n + 1$ variables are needed to describe the challenge-response relation. In the model we derive, we show that this is only possible when certain assumptions are made about the inter-switch versus the intra-switch delay variations. We start by examining the behavior of the pulses passing through each of the n switches.

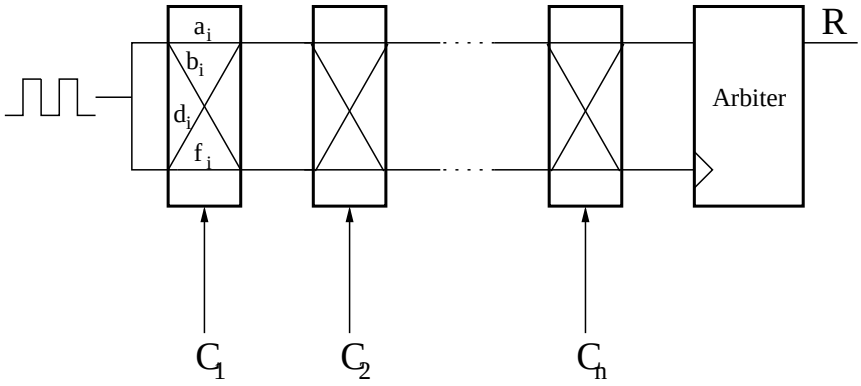


Figure 4.2: A block diagram of a delay-based PUF circuit

At each of the n stages, each of the two pulses propagating through the PUF has two paths to take. We label the two paths which the upper signal can take as a_i and b_i, and for the lower signal as d_i and f_i. This is illustrated in Figure 4.2 which

captures a block diagram of the PUF. Paths a_i and f_i are chosen when the challenge bit c_i is 0, whereas b_i and d_i are chosen when the challenge bit is 1. To compute the total delay, the path of the upper signal is followed from the initial input pulse. The signal will start traveling in a separate path to get to the first switch. Let us label this initial incurred delay as a_0. For the signal going through the lower path we label this delay as f_0. In the first switch, the signal will incur a delay that is dependent on the control signal of the corresponding switch. The delay will be $(\overline{c_1}a_1 + c_1b_1)$, where $\overline{c_1}$ is the complement of c_1. The delay in the second switch will depend on whether the signals switched paths in the first stage or not. If the signals changed paths in the first switch, then the possible delays it will incur in the second switch will be d_2 or f_2. Otherwise, the delay will be a_2 or b_2. The delay of the second stage can be written as $\overline{c_1}(\overline{c_2}a_2 + c_2b_2) + c_1(\overline{c_2}f_2 + c_2d_2)$. To carry the analysis to the rest of the switches, we introduce the variable $x_i = c_1 \oplus c_2 \oplus \ldots \oplus c_{i-1}$ where $x_1 = 0$. This variable x_i represents the parity of the first $i-1$ challenge bits, and will signify if the signal starting at the upper path stays in that path or moves to the lower path after $i-1$ switches. The delay for the i^{th} switch can now be denoted as $\overline{x_i}(\overline{c_i}a_i + c_ib_i) + x_i(\overline{c_i}f_i + c_id_i)$. The total delay in the pulse that starts in the upper path will therefore be

$$D_1 = a_0 + \sum_{i=1}^{n} \overline{x_i}(\overline{c_i}a_i + c_ib_i) + x_i(\overline{c_i}f_i + c_id_i) \ .$$

Similarly, the delay for the pulse initiated in the lower path can be derived to be equal to

$$D_2 = f_0 + \sum_{i=1}^{n} \overline{x_i}(\overline{c_i}f_i + c_id_i) + x_i(\overline{c_i}a_i + c_ib_i) \ .$$

The difference between these two delays δ is the main variable for our model. This difference will decide whether the output of the PUF is 0 or 1. Since we do not know which of the two signals will end up in the upper path, we will need to incorporate

the parity of all the challenge bits which we label P. The difference between the two delays becomes

$$\begin{aligned}\delta &= (-1)^P(D_1 - D_2) \\ &= (-1)^P\left(\sum_{i=1}^{n}(\overline{x_i} - x_i)(\overline{c_i}(a_i - f_i) + c_i(b_i - d_i)) + (a_0 - f_0)\right) \\ &= \sum_{i=1}^{n}(-1)^{P\oplus x_i}(\overline{c_i}(a_i - f_i) + c_i(b_i - d_i)) + (-1)^P(a_0 - f_0)\ .\end{aligned}$$

Finally, we define $u_i = \frac{(a_i-d_i)+(b_i-f_i)}{2}$ and $v_i = \frac{(a_i+d_i)-(b_i+f_i)}{2}$ for $i = 1 \ldots n$, and $v_0 = a_0 - f_0$. We define the parity of the challenge bits from a reverse order as $p_i = P \oplus x_i = c_i \oplus c_{i+1} \oplus \ldots \oplus c_n$. The delay equation becomes:

$$\begin{aligned}\delta &= \sum_{i=1}^{n}(-1)^{p_i}(u_i + (-1)^{c_i}v_i) + (-1)^P v_0 \\ &= \sum_{i=1}^{n}(-1)^{p_i}u_i + (-1)^{p_i\oplus c_i}v_i + (-1)^P v_0\ .\end{aligned}$$

In an actual implementation, there are only two paths connecting the consecutive switches, the four possible paths mentioned take place inside the switches (see Figure 4.1). In reality, a_i and d_i share the same path between the i^{th} switch and $i+1$ st switch. However, these two delays differ in their paths within the i^{th} switch. The same thing can be said about f_i and b_i. In the PUF circuit layout, the switches will be positioned at a significant distance from each other. Therefore, the internal delay variation in the i^{th} switch u_i will have a much smaller magnitude than the delay variation v_i in the path between the i^{th} switch and the $i+1$ switch. Due to this reason, we can drop the variable u_i from the previous equation without having a noticeable effect on the challenge-response relation.[2] Note that $p_1 = P$, and that

[2]This is the assumption needed to obtain a model similar to that in [27]. However, this assumption can be eliminated if we use tristate buffers to build the delay-based PUF, this is explained in Section 4.3. The model derived there can essentially be described using Equation (4.1) with c_i used instead of p_i.

$p_i \oplus c_i = c_{i+1}$. We also define $y_i = v_{i-1}$. The final delay equation becomes

$$\delta \;=\; \sum_{i=1}^{n}(-1)^{p_i}y_i + y_{n+1}\ . \tag{4.1}$$

Equation 4.1 uses only $n+1$ rather than $2n+1$ variables to describe the delay between the upper and the lower signal paths. This reduction is very effective in terms of efficiency, and at the same time slightly affects the error rate that can be produced from such a model. It is important to note that the delay variations y_i will depend on the fabrication process of the PUF circuit. Therefore, one would expect these variables to follow a normal distribution. In particular, the y_i values will follow a Gaussian distribution of mean zero, and a fixed variance [72, 71]. Without loss of generality, we can normalize these values and assume they belong to a normal distribution of mean 0 and variance 1. Note that the y_i variables will be the entries of the PV used in this section.

Now that we have a linear model capturing the behavior of the signals inside a PUF circuit, we derive the input output relation. Invoking the arbiter conditions for the response bit R we get,

$$\begin{aligned}\delta - T_S &> 0 \rightarrow R = 1\\ \delta - T_S &\leq 0 \rightarrow R = 0\ ,\end{aligned}$$

where T_S is the setup time for the flip flop utilized as an arbiter. To make this more concise, we relate the delay value to the response bit as $(-1)^R(\delta - T_S) < 0$. Finally, we can use Equation 4.1 to write the response equation[3]

$$(-1)^R\left(\sum_{i=1}^{n}(-1)^{p_i}y_i + y_{n+1}\right) < 0\ \ . \tag{4.2}$$

Next, we show how to learn the y_i values from the hardware implementation. For this purpose, note that Equation 4.2 is an inequality relating the challenge C which

[3]For brevity the setup time T_S can be merged with the last delay variable y_{n+1}.

consists of n input bits, c_i, to the output bit R. This inequality has $n+1$ variables which characterize the PUF circuit. Passively observing the challenge-response pairs $(C^{(j)}, R^{(j)})$ for a single PUF, we may form the following linear equation

$$\delta_j = (-1)^{p_1^{(j)}} y_1 + (-1)^{p_2^{(j)}} y_2 + \ldots + (-1)^{p_n^{(j)}} y_n + y_{n+1} \ .$$

Using Equation (4.2), we may write the following linear inequality:

$$(-1)^{R^{(j)}} \left[\ (-1)^{p_1^{(j)}} \quad (-1)^{p_2^{(j)}} \quad \ldots \quad (-1)^{p_n^{(j)}} \quad 1 \ \right] Y < 0$$

where $Y = [\ y_1, \quad y_2, \quad \cdots \quad , y_n \quad , y_{n+1}\ ]^T$. With Ns challenge-response pairs we can produce a system of linear inequalities and solve for Y using standard *linear programming* methods [98, 86]. In order to test our mathematical model, we ran a number of tests using MATLAB. Our tests were performed for three sizes of a PUF circuit $n = 32, 64$ and 128. For each n we generated the $n+1$ variables characterizing the PUF using a Gaussian distribution, with a mean of 0 and a standard deviation of 1. For each n we generated a number of random challenges N_s, and then calculated the response of the PUF according to the model presented. We then used the challenge-response pairs generated in order to solve for the $n+1$ variables which were randomly chosen. In order to solve these equations we used the medium scale algorithm supported by MATLAB [18], this algorithm is a variation of the known Simplex algorithm [4]. In order to test the viability of the model we tested the noiseless case. We generated 500,000 random points and compared the response produced by our linear model. Table 4.1 shows the percentage error we obtained. It is important to note that the percentage error was calculated over the entire space of possible challenges. However, if we constrain the challenges to the ones which were used to build the linear model, then the linear model will perfectly match the actual response. Hence, the linear model will essentially be a compressed version of the challenge-response database. From the previous argument it might be tempting to conclude that any level of accuracy can be achieved when attempting to model a PUF

	Number of challenge-response pairs N_s									
n	32	64	128	256	512	1024	2048	4096	8192	16384
32	40.33	33.03	21.18	11.63	5.70	2.61	1.37	0.61	0.28	0.14
64	45.77	40.62	32.44	21.27	11.30	5.64	2.60	1.11	0.57	0.27
128	47.66	45.30	40.08	32.02	21.38	12.13	5.94	2.89	1.27	0.59

Table 4.1: Percentage error for each (n, N_s) pair

circuit. However, this is not the case for an actual implementation. In [27, 68, 72], results of modeling a PUF circuit implemented in FPGA and in ASIC are reported. The lowest error rate achieved among these results was about 3%. Note that these results were achieved using machine learning algorithms. These results suggest a lower limit on the achievable accuracy from modeling an actual PUF circuit. This lower limit can be explained by the sensitivity of the circuit. For certain challenges the circuit goes into a metastable state where the the two pulses enter a race condition. In a metastable state the time difference between the two pulses will be smaller that the resolution of the arbiter. Therefore, the PUF output will be 0 or 1 with almost equal probability. These metastable states are not predictable, and will correspond to different challenges at different times due to the changes in external interferences, such as temperature, voltage and electromagnetic emanation. In [77], the authors use metastability to help design a true random number generator. They predict that about 0.1% of all the challenges will fall into a metastable state. Even in the presence of noise it will be possible to solve for the Y values. In [8], the authors introduce a polynomial time algorithm for solving noisy linear threshold functions similar to the one implemented in a PUF. It can be said that the PUF circuit presented in this section is completely modelable.

With the above we can now state the following definition of a PUF. The definition models a delay-based PUF as a function with some error parameter which accounts

for the noise and the modeling errors.

Definition 4.2.1. *A PUF is a function characterized by the vector* $Y \in \mathbb{R}^{n+1}$ *and the noise parameter* $\epsilon \in \left(0, \frac{1}{2}\right)$*, where* $Y = [y_1, \ldots, y_{n+1}]$ *and* $y_i \in N(0,1)$*.* $N(\mu, \sigma^2)$ *being the normal distribution with mean* μ *and variance* σ^2*.* $PUF_{Y,\epsilon}(c) : \{0,1\}^n \rightarrow \{0,1\}$ *such that*

$$PUF_{Y,\epsilon}(c) = sign\left(\sum_{i=1}^{n}(-1)^{p_i}y_i + y_{n+1}\right) \oplus \nu \quad . \tag{4.3}$$

where $sign(x) = 1$ *if* $x > 0$*, and* 0 *if* $x \leq 0$*. Also,* $\nu = 1$ *with probability* ϵ *and* $\nu = 0$ *with probability* $1 - \epsilon$*. Finally* $p_i = c_i \oplus c_{i+1} \oplus \ldots \oplus c_n$*, where* c_i *is the* i^{th} *bit of* c*.*

Note that the relation between $P = [p_1 \ldots p_n]$ and $C = [c_1 \ldots c_n]$ can be described using the equation $(P = UC)$. The strings C and P are represented as column vectors, U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2.

From the definition above it is clear that a PUF is a PPF. And from the earlier discussion it is also clear that PUFs are clonable. When PUFs were proposed in [27] the authors noted that PUFs may be vulnerable to modeling and differential attacks in the original configuration. Therefore, in [28] the authors propose to apply a cryptographic hash function to the output of the PUF circuit to eliminate such attacks. However, recall from Chapter 1 that one of our goals is to achieve our goals in a lightweight fashion. The introduction of the cryptographic hash function adds significantly to the footprint and power consumption. A hash function is a challenging cryptographic primitive to serialize. Therefore, reducing the gate count might not be feasible. In [27], feed forward arbiters were proposed to introduce non-linearity into the PUF scheme. This approach seems promising. However, since the feed forward PUF are not known to be modelable, one would need to collect a large number of challenge-response pairs for every PUF circuit and store them in a database. This is the classical approach proposed in the early literature on PUF. This solution becomes

less practical when the number of devices deployed becomes massive. We will attempt to address this problem in the next two chapters.

In the next section we will explore some properties of PUFs based on the above model.

4.2.3 Properties

In this section we will be concerned with one PUF property. The following lemma characterizes the correlation between two PUF outputs obtained from different input challenges. The proposition uses Definition 4.2.1 with the noise level $\epsilon = 0$.

Proposition 4.2.1. *Given two n-bit strings $a^{(1)}$ and $a^{(2)}$ chosen independently and uniformly at random from $\{0,1\}^n$, and $Y_0 = [y_1, \ldots, y_{n+1}]$ where $y_i \in N(0,1)$ for $i \in [1..n]$ and $y_{n+1} = 0$, and let $z^{(1)} = PUF_{Y_0}(a^{(1)})$ and $z^{(2)} = PUF_{Y_0}(a^{(2)})$. Assuming that the y_i values are independent then the probability that $z^{(1)}$ and $z^{(2)}$ are equal is*

$$Pr[z^{(1)} = z^{(2)}] = F_n(d) = 1 - \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{n+1-d}}\right). \tag{4.4}$$

Where d is the Hamming distance between $P^{(1)}$ and $P^{(2)}$ and ($P^{(i)} = Ua^{(i)}$). The strings $a^{(i)}$ are represented as column vectors, U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2.

Proof. First recall from the last section that for any string $c = [c_1, \ldots c_n]$ the PUF response is

$$\mathrm{PUF}_Y(c) = \mathrm{sign}\left(\sum_{i=1}^{n} (-1)^{p_i} y_i + y_{n+1}\right),$$

where $p_i = c_i \oplus \ldots \oplus c_n$ and $P = [p_1, \ldots, p_n]$, as noted above ($P = U \cdot c$). This mapping is a linear bijection and will therefore preserve uniformity and independence. Since $a^{(1)}$ and $a^{(2)}$ were chosen uniformly at random from $\{0,1\}^n$, the same can be said about the distribution of the corresponding P vectors $P^{(1)} = [p_1^{(1)}, \ldots p_n^{(1)}]$ and $P^{(2)} = [p_1^{(2)}, \ldots p_n^{(2)}]$. Let $Z(P^{(j)}, Y_0) = \sum_{i=1}^{n} (-1)^{p_i^{(j)}} y_i + y_{n+1}$, and let d be the

Hamming distance between $P^{(1)}$ and $P^{(2)}$. Without loss of generality we may assume that the different bits of $P^{(1)}$ and $P^{(2)}$ are in the first d bit positions. Also, let $k = n + 1$. Now we can write

$$\begin{aligned} Z(P^{(1)}, Y_0) &= \sum_{i=1}^{d} (-1)^{p_i^{(1)}} y_i + \sum_{i=d+1}^{k} (-1)^{p_i^{(1)}} y_i = D_d + S_{k-d} \\ Z(P^{(2)}, Y_0) &= -D_d + S_{k-d} \ , \end{aligned}$$

where $D_d = \sum_{i=1}^{d} (-1)^{p_i^1} y_i$ and $S_{k-d} = \sum_{i=d+1}^{k-1} (-1)^{p_i^1} y_i + y_k$. Note that since $z^{(j)} = \text{sign}(Z(P^{(j)}, Y_0))$, then for $z^{(1)}$ to be equal to $z^{(2)}$ we need $Z(P^{(1)}, Y_0)$ and $Z(P^{(2)}, Y_0)$ to have the same sign. For this, D_d needs to have a smaller magnitude than S_{k-d} which means $|D_d| < |S_{k-d}|$. Therefore

$$\Pr[z^{(1)} = z^{(2)}] = \Pr[|S_{k-d}| - |D_d| > 0] = \Pr[R_d > 0],$$

where $R_d = |S_{k-d}| + (-|D_d|)$. Let $f_R(R_d), f_D(D_d)$ and $f_S(S_{k-d})$ represent the probability distribution function (PDF) for each of the random variables R_d, D_d and S_{k-d} respectively. Each term in the summations making up D_d and S_{k-d} involves one of the bits $p_i^{(1)}$ and the real value y_i. Since $y_i \in N(0, 1)$ has mean zero and is symmetric around the y-axis, we can easily see that multiplying with $(-1)^{p_i^{(1)}}$ will not affect the normal distribution and therefore $(-1)^{p_i^{(1)}} y_i \in N(0, 1)$. Now each of the variables D_d and S_{k-d} is a summation of respectively d and $k-d$ independent normal distributions $N(0, 1)$. Thus, $f_D(D_d) = N(0, d)$ and $f_S(S_{k-d}) = N(0, k - d)$.[4] We are interested in the PDF of $-|D_d|$ and $|S_{k-d}|$. These can easily be calculated as follows:

$$f_{|S|}(x) = \left\{ \begin{array}{ll} 2N(0, k-d) & x > 0 \\ 0 & x \leq 0 \end{array} \right\} = \left\{ \begin{array}{ll} 2\frac{e^{\frac{-x^2}{2(k-d)}}}{\sqrt{2\pi(k-d)}} & x > 0 \\ 0 & x \leq 0 \end{array} \right\} ,$$

and

$$f_{-|D|}(x) = \left\{ \begin{array}{ll} 0 & x > 0 \\ 2N(0, d) & x \leq 0 \end{array} \right\} = \left\{ \begin{array}{ll} 0 & x > 0 \\ 2\frac{e^{\frac{-x^2}{2d}}}{\sqrt{2\pi d}} & x \leq 0 \end{array} \right\} .$$

[4] This is a straightforward application of the Central Limit Theorem. It is also very easy to derive directly from the PDF of a normal random variable.

Now we can calculate the desired probability

$$\begin{aligned}
\Pr[z_{(1)} = z_{(2)}] &= \Pr[R_d > 0] \\
&= \int_0^\infty f_R(w)\ dw \\
&= \int_0^\infty f_{-|D|}(w) * f_{|S|}(w)\ dw \\
&= \int_0^\infty \frac{4}{2\pi\sqrt{d(k-d)}} \cdot \\
&\qquad \left[\int_{-\infty}^\infty e^{\frac{-x^2}{2(k-d)}} \cdot U(x) \cdot e^{\frac{-(w-x)^2}{2d}} \cdot U(x-w)\ dx\right] dw
\end{aligned}$$

where $*$ denotes the convolution operator and $U(x)$ is the unit step function. By rearranging the terms we obtain

$$\begin{aligned}
\Pr[z_{(1)} = z_{(2)}] &= \frac{4}{\sqrt{2\pi(k)}} \int_0^\infty \left(\frac{1}{\sqrt{2\pi\sigma^2}} \int_w^\infty e^{\frac{-(x-\mu)^2}{2\sigma^2}}\ dx\right) e^{\frac{-w^2}{2k}}\ dw \\
&= 2\int_0^\infty \frac{1}{\sqrt{2\pi k}} \left[1 - \operatorname{erf}(\alpha w)\right] e^{\frac{-w^2}{2k}}\ dw \\
&= 2\left[\int_0^\infty \frac{e^{\frac{-w^2}{2k}}}{\sqrt{2\pi k}}\ dw - \int_0^\infty \frac{e^{\frac{-w^2}{2k}} \cdot \operatorname{erf}(\alpha w)}{\sqrt{2\pi k}}\ dw\right] \\
&= 2\left[\frac{1}{2} - \frac{1}{\pi}\arctan\left(\alpha\sqrt{2k}\right)\right]
\end{aligned}$$

where $\sigma^2 = \frac{d(k-d)}{k}$, $\mu = \frac{\sigma^2 w}{d}$, $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\ dt$ and $\alpha = \sqrt{\frac{d}{2k(k-d)}}$. The first of the two integrations is just a Gaussian over half of the space. As for the second integration this is a known definite integral. Finally, substituting back with the original variables n and d we obtain

$$\Pr[z^{(1)} = z^{(2)}] = F_n(d) = 1 - \frac{2}{\pi}\arctan\left(\sqrt{\frac{d}{n+1-d}}\right)\ .$$

□

From Proposition 4.2.1 we can conclude that the only property which affects the correlation between two different PUF responses is the Hamming distance between

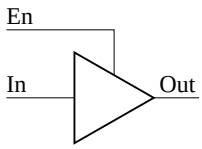
the corresponding challenges. Note that the derivation of the above equation was possible due to the independence assumption made on the components of Y. This assumption is quite important and should be one of the main metrics judging any PUF implementation. With large dependencies between y_i's, the entropy inside the PUF decreases thus limiting its identification capability.

4.3 PUFs Using Tristate Buffers

The original PUF circuit utilized multiplexers (MUXs) to implement the switches. Each delay unit includes two MUX gates. Instead of having two interleaved delay paths, we propose a circuit with two separate delay paths. We construct our PUF circuit utilizing the delay variances of tristate buffers. As we will see this construction will save on the number of logic gates used.

4.3.1 Tristate Buffers

A generic tristate buffer as shown in Figure 4.3 has three states: logic 1, logic 0 and high impedance (referred to as Hi-Z). If the gate is enabled, the input will be reflected at the output. If not enabled, the output will become Hi-Z. The truth table of a tristate buffer is shown in Figure 4.3. The outputs of two tristate buffers can be connected together to behave as a multiplexer gate. However, when both of the tristate buffers are enabled, the circuit is under risk of being damaged. If the inputs of the two tristate buffers are complement values of each other, and the enable inputs are both high, then the circuit will start heating up and eventually will be damaged.



En	In	Out
0	X	Hi-Z
1	0	0
1	1	1

Figure 4.3: A generic tristate buffer

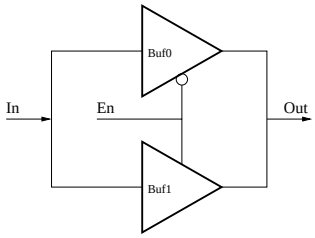


Figure 4.4: Delay unit built with tristate buffers

4.3.2 Delay Unit

A PUF circuit is based on the delay characteristics of the logic design. We build our tristate buffer based on the delay unit shown in Figure 4.4. Each delay unit is constructed by respectively connecting the input and output ports of the two tristate buffers. The enable ports of these tristate buffers are connected to each other, with one of the buffers having an inverted enable input. This assures that only one of the tristate buffer gates will be enabled for a particular enable input value. When a pulse is applied at the input of this delay unit, it will pass through either Buf0 or Buf1. The enable input will determine which tristate buffer passes the pulse. This will change the amount of the delay that this unit contributes according to the value of the enable signal.

4.3.3 PUF Circuit

The delay units are cascaded to construct a serial delay path. For our PUF circuit, we need two separate delay paths. These two delay paths are connected as shown in Figure 4.5. The inputs of these delay paths are connected together and the outputs

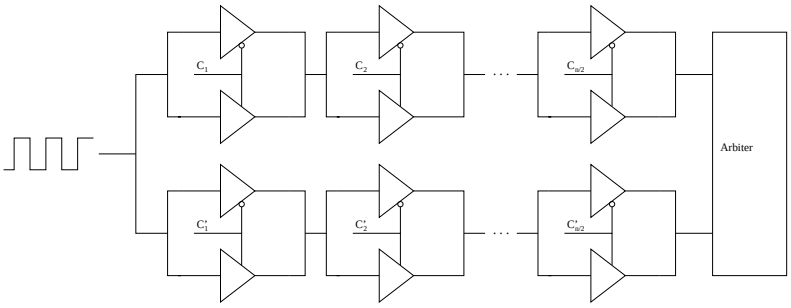


Figure 4.5: PUF architecture built with tristate buffers

are fed to an arbiter. The arbiter will capture the faster path by producing logic 1 or 0 at the output. Assume the arbiter is built with a positive edge-triggered flip-flop. The upper path is connected to the data input of the flip-flop and the lower path is connected to its clock input. If the lower path is faster, the rising edge of the clock signal will arrive before the rising edge of the data signal, therefore producing a logic 0 at the arbiter output. Alternatively, if the upper path is faster we observe the opposite effect, i.e. the output of the arbiter becomes a logic 1.

4.3.4 Mathematical Model

In this section we show that a tristate PUF is equivalent to a switch-based PUF in terms of their mathematical model. In particular we show that both implementations can be modeled using a similar linear model. In the tristate implementation of a PUF there are two groups of consecutive delay units; the upper and the lower. If we consider the upper path we notice that in each delay unit there are two possible paths. Let us call the delay in each of these two paths a_i and b_i, where i is the stage

index. Let $H_i = \frac{a_i+b_i}{2}$ and $y_i = \frac{a_i-b_i}{2}$, which means that $a_i = H_i + y_i$ and $b_i = H_i - y_i$. Depending on whether the challenge bit is 0 or 1, the signal going through the upper units will be delayed by $H_i + (-1)^{c_i} y_i$ where c_i is the i^{th} challenge to the upper delay units. Assuming that the signal has $n/2$ stages to propagate through, the total delay in the upper delay units becomes equal to

$$D_H = \sum_{i=1}^{n/2} H_i + (-1)^{c_i} y_i = \tau_H + \sum_{i=1}^{n/2} (-1)^{c_i} y_i \quad , \tag{4.5}$$

where $\tau_H = \sum_{i=1}^{n/2} H_i$. Similarly, one can derive the delay equation for the lower delay units. The two delays in each of the paths of the lower delay units can be named d_i and f_i. Similarly one can make $L_i = \frac{d_i+f_i}{2}$ and $u_i = \frac{d_i-f_i}{2}$. Now if we let c_i' be the challenge bit to the i^{th} stage of the lower units, the delay in each unit becomes $L_i + (-1)^{c_i'} u_i$. Therefore, the total delay in the lower path becomes:

$$D_L = \sum_{i=1}^{n/2} L_i + (-1)^{c_i'} u_i = \tau_L + \sum_{i=1}^{n/2} (-1)^{c_i'} u_i \quad . \tag{4.6}$$

The two signals traveling through the upper and lower delay units will interact through the arbiter. The condition on the output bit becomes:

$$D_H < D_L + T_S \rightarrow R = 1$$
$$D_H > D_L + T_S \rightarrow R = 0 \quad ,$$

where T_S is the setup time for the arbiter latch. One can make the two relations more concise by relating the response bit, $(-1)^R(D_L - D_H + T_s) < 0$. If we incorporate the original equations for D_L and D_H, we get the following response equation:

$$(-1)^R(\sum_{i=1}^{n/2} (-1)^{c_i'} u_i - (-1)^{c_i} y_i + T) < 0 \quad , \tag{4.7}$$

where $T = \tau_L - \tau_H + T_S$. Note that T, y_i and u_i are variables dependent on the circuit. These variables are dependent on the circuit itself and will vary from one PUF circuit

to another. For simplicity we assume that n is even and we define $Y = [y_1, \ldots, y_{n+1}]$ where $y_i = u_i$ and $y_{\frac{n}{2}+i} = v_i$ for $i = 1 \ldots \frac{n}{2}$, with $y_{n+1} = T$. Similarly we rename the challenge bits as $a_i = c_i$ and $a_{\frac{n}{2}+i} = c'_i$ for $i = 1 \ldots \frac{n}{2}$. We can compute the output of the PUF using the following function,

$$\mathrm{PUF}_Y(a) = \mathrm{sign}\left(\sum_{i=1}^{n}(-1)^{a_i}y_i + y_{n+1}\right) \quad .^5 \tag{4.8}$$

Where $\mathrm{sign}(x) = 1$ if $x \geq 0$, and 0 if $x < 0$. Note that this is almost identical to the equation used in Definition 4.2.1. The only difference is that the input bits are immediately used without the need for the initial transformation U. Of course Equation 4.8 needs a noise parameter in order to accurately model the actual circuit. It should be clear that the same modeling that we carried out for MUX-PUFs can be used here. We will usually refer to the MUX and the tristate PUFs as PUFs or delay-based PUFs and only specify the implementation type when it is relevant.

4.3.5 Implementation Results

We implemented the proposed tristate PUF and compared it against an implementation of a switch-based PUF built using multiplexers. The I/O for both circuits consisted of 64-bit challenge inputs, a single bit pulse input and a response output. Both designs were developed into Verilog modules and synthesized using the Synopsys tools Design Compiler and Power Compiler with the TSMC 0.13 μm ASIC library. The results are shown in Table 4.2.

The implementation results clearly display the superiority of the tristate implementation over the typical switch-based implementation in terms of design footprint and power consumption. These are attractive features for low-power devices such as RFIDs, smart cards and sensor networks. Note that the area (and the power) consumption of the tristate PUF may be further reduced by custom CMOS design.

[5] This model is almost identical to the model derived for MUX-based PUFs

	Total Power(uW)		Area(Gates)
	10MHz	100MHz	
Tristate PUF	18.78	152.93	351
MUX-PUF	23.14	193.67	450

Table 4.2: Synthesis results for tristate PUFs

4.4 Summary

In this chapter we explored delay-based PUFs and their constructions. In particular, we rederive a mathematical model for delay PUFs. We also present a cheaper implementation of delay PUFs using tristate buffers. And finally, we highlight one major property of the delay PUF. The main job of this chapter was to set the stage for PUFs before we can use them in a number of cryptographic schemes with the goal of creating a secure PUF.

Chapter 5

PUF-HB

We saw in the last chapter the various attractive properties of PUFs[1]. However, PUFs were modelable. In this chapter we will take a new approach in an attempt to get closer to an actually unclonable function. Our main idea is to merge the typical PUF circuit along with simple cryptographic primitives which can provide some level of computational hardness. This will prove to be a fruitful approach as it will maintain the PUF advantages including the advantages of having a model for the PUF behavior, while at the same time providing a level of provable security.

The cryptographic primitive we choose for merger with PUFs is an authentication family labeled HB. In complement to PUFs the HB-based authentication schemes provide a security reduction. In [46] Hopper and Blum (HB) proposed the first HB authentication scheme. The HB protocol is indeed promising for simplifying the authentication process and significantly reducing the power consumption. An additional major advantage of the HB protocol is that its security is based on the hard problem learning parity with noise (LPN) which is known to be NP-complete [7]. Unfortunately, as pointed out in [46] the HB scheme is weak against active adversaries. In [50], Juels and Weis proposed a hardened version of the HB protocol labeled HB$^+$

[1]As mentioned in Chapter 4 we will use PUF to stand for a delay-based PUF.

which resists active attacks in the detection based model.[2] HB^+ was shown to be secure [50, 53] in the detection based model. In [34], the authors demonstrated a man-in-the-middle attack for breaking HB^+.

In this chapter we merge the PUF authentication scheme along with the HB-based authentication protocol to produce a hybrid protocol which enjoys the advantages of both schemes while improving the level of security. The proposed authentication scheme enjoys a level of tamper resilience, while at the same time being provably secure against active attacks in the detection based model. In addition, the presented protocol resists the man-in-the-middle attacks proposed so far for the HB^+ scheme. From the PUF perspective, PUF-HB is the first PUF based authentication scheme which is provably secure. From the HB perspective PUF-HB is the first hardware HB implementation with tamper resilience properties. For our security proof we closely follow the proof presented in [53].

The remainder of this chapter is organized as follows. In the next section we give a review of the previously proposed HB-based authentication protocols. In Section 5.2 we define our notation and describe our protocol. The security reduction is presented in Section 5.3. In Section 5.4 we describe the known man-in-the-middle attack against HB^+ and show that the proposed protocol resists it. In Section 5.5 we discuss hardware implementation and tamper resilience properties of the proposed scheme.

5.1 The HB Family

The HB authentication schemes base their security on the hardness of the LPN problem. In this section we give a quick review of the LPN problem and the different HB authentication schemes, focusing on HB and HB^+. For a certain $\epsilon \in \left(0, \frac{1}{2}\right)$ the LPN problem is denoted by LPN_ϵ, and defined as follows.

[2] In this model, a flag is raised whenever a tag fails to authenticate for a large number of times.

Definition 5.1.1 ([53]). *Given n random binary k-bit strings $a_{(j)}$ and the bits $z_{(j)} = a_{(j)} \cdot x \oplus \nu$ for some $x \in \{0,1\}^k$, where $a \cdot b$ denotes the binary inner product between a and b, $\nu = 1$ with probability ϵ and 0 with probability $1 - \epsilon$, then find the binary string x.*

The LPN problem is known to be NP-hard [7]. In [46], the authors show that the LPN problem is log-uniform and even hard to approximate within a ratio of 2. Kearns proved in [55] that the LPN problem is hard in the statistical query model. The best known algorithm to solve the LPN problem is the BKW algorithm [9]. However, there has been a number of improvements on the algorithm with the best running time of $2^{O(k/\log k)}$ [26, 69, 74].

In the HB protocol [46], the tag and the reader share a k-bit secret string x. To authenticate the tag, the reader starts sending randomly generated k-bit challenge strings $a_{(j)}$. The tag responds with the bit $z_{(j)} = a_{(j)} \cdot x \oplus \nu$ where the variables are as defined in the LPN problem. The tag and the reader repeat the same step for multiple challenges. Finally, the reader checks to see if the number of errors in the tag's response matches the noise level, and decides to accept or reject accordingly. Note that if the tag's response did not contain noise, then a passive attacker would easily be able to deduce x after collecting k challenge-response pairs using Gaussian elimination. In [46], the authors prove that given an algorithm that predicts $z_{(j)}$ for a random $a_{(j)}$ with some advantage, then this algorithm can be used to solve the LPN_ϵ problem. However, HB is only secure against passive attacks. An active attacker can easily repeat the same challenge multiple times, effectively eliminating the noise and reducing the problem to Gaussian elimination.

To secure the HB protocol against an active attacker the HB^+ protocol was proposed in [50]. In HB^+ the tag and the reader share two k-bit strings x and y. The tag starts the authentication session by sending a random k-bit string $b_{(j)}$. The reader then responds with $a_{(j)}$ just like the HB protocol. Finally the tag responds with

$z_{(j)} = a_{(j)} \cdot x \oplus b_{(j)} \cdot y \oplus \nu$, where ν is defined as above. The protocol is proven to be secure against an active attack on the tag (excluding man-in-the-middle attacks). In such an adversary model an attacker is not allowed to obtain final decisions from the reader on whether this authentication session was successful or not. In [50] and [53] the authors show that in this adversary model breaking the HB$^+$ protocol is equivalent to solving the LPN problem. However, as we pointed out earlier, a simple man-in-the-middle attack was demonstrated on the HB$^+$ protocol in [34]. Note that in a detection based model this attack will not be successful.

In addition to HB and HB$^+$, there has been a number of other variations such as HB^{++} [13], HB-MP [76] and HB* [21]. All these proposals attempt to prevent man-in-the-middle attacks. In a more recently proposed scheme, i.e. HB$^\#$ [33], the authors propose a modified version of HB$^+$ which uses Toeplitz matrices rather than vectors for a shared secret. Under a strong conjecture the scheme is proven secure against a class of man-in-the-middle attacks. In this adversary model which is referred to as *GRS-MIM-model*, the attacker can only modify data transmission from the reader to the tag but not the other way around. This model will protect against the previously mentioned attack.

5.2 The PUF-HB Authentication Protocol

We start by defining basic notation. In the remainder of this chapter we reserve k to denote the security variable. $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ denotes the tag used in the PUF-HB protocol, where $x \in \{0,1\}^k$, $s \in \{0,1\}^{2\times n}$ where we treat s_j as a 2-bit vector. $Y = [y_1, y_2, \ldots, y_{k+1}]$ such that $y_i \in N(0,1)$ and $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. The noise parameter is $\epsilon \in \left(0, \frac{1}{2}\right)$ and n denotes the number of rounds required for authentication. We denote the reader by $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$, where all the variables are as defined for the tag except l and u which are integers in the range

$[0, n]$ such that $l \leq \epsilon \cdot n \leq u$. We also use the following modification to Definition 4.2.1 of Chapter 4. For any bit v,

$$\mathrm{PUF}_{Y,v}(a) = \left\{ \begin{array}{ll} \mathrm{sign}\left(\sum_{i=1}^{k}(-1)^{p_i}y_i + y_{k+1}\right), & v = 0 \\ \mathrm{sign}\left(\sum_{i=1}^{k}(-1)^{\overline{p_i}}y_i + y_{k+1}\right), & v = 1 \end{array} \right\} \quad , \tag{5.1}$$

where $\overline{p_i}$ is the complement of p_i.[3] Note that we have eliminated the noise parameter as we are interested in the mathematical behavior. This notation will simplify the security proof.

With this notation we describe the basic authentication step. In the j^{th} round of the protocol, $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ outputs a randomly generated vector $b_{(j)} \in \{0,1\}^k$ and sends it to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$. The reader responds with the vector $a_{(j)} \in \{0,1\}^k$ and $e_{(j)} \in \{0,1\}^2$. Finally the tag computes $z_{(j)} = b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)}\cdot s_j}(a_{(j)}) \oplus \nu$, where $\cdot$ is the binary inner product, s_j are the j^{th} two bit vector of s and $\nu = 1$ with probability ϵ and $\nu = 0$ with probability $1-\epsilon$. For authentication, this step is repeated n times. In each round the reader checks to see if the tag's response is equal to $b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)}\cdot s_j}(a_{(j)})$. If not, the reader marks the response as wrong. At the end of the n^{th} round, the reader authenticates the tag if and only if the number of wrong responses is in the range $[l, u]$.

In general, any entity can interact with the reader and try to impersonate an honest tag. To capture such interaction, let $\mathcal{E}$ be any entity trying to authenticate itself to the reader $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$. Then $\text{PUF-HB}\big(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$ iff $\mathcal{E}$ is authenticated by the reader, and is equal to 0 otherwise. The following protocol formalizes this interaction:

Provided that $\mathcal{E}$ has no information about x, s or Y, the best probability of being authenticated by the reader will be $\epsilon_s = 2^{-n}\sum_{i=l}^{u}\binom{n}{i}$. This probability represents the soundness error in the algorithm. As for an honest tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ one can see that with a very high probability $\text{PUF-HB}\big(\mathcal{T}_{(n,x,Y,s,\epsilon)}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$. However,

[3]The complement may be realized by simply XOR-ing the most-significant bit of a with v.

Protocol 1: PUF-HB$\big(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big)$

1. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ sets $c = 0$ and $j = 1$.
2. $\mathcal{E}$ sends $b_{(j)} \in \{0,1\}^k$ to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$.
3. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ choses $a_{(j)} \in \{0,1\}^k$ and $e_{(j)} \in \{0,1\}^2$ uniformly at random and sends it to $\mathcal{E}$.
4. $\mathcal{E}$ sends $z_{(j)}$ to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$.
5. If $z_{(j)} \neq b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)})$ then $c = c + 1$.
6. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ increments j and repeats steps 2 through 5 until $j = n$.
7. If $l \leq c \leq u$ then PUF-HB$\big(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$, otherwise it equals 0.

Table 5.1: PUF-HB Protocol

the tag's choice to set $\nu = 1$ with probability ϵ is independent in each round of the authentication. Therefore, it will be possible for the tag to introduce a number of errors which is outside the range $[l, u]$. This will result in a failed authentication session. We denote the probability of this incident by ϵ_c, i.e. the completeness error. If we set $l = 0$ then using the Chernoff bound we can produce the following bound, $\epsilon_c < e^{-(\epsilon n)(u/\epsilon n - 1)^2/4}$.

Protocol (1) would work identically if we run it in a parallel fashion. In this case, the n different $b_{(j)}$ vectors sent in Step 2 would be sent in a single step and similarly all the $a_{(j)}$ vectors and $e_{(j)}$ bits sent in Step 3 would also be sent in a single step. Finally all the $z_{(j)}$ bits returned in Step 4 would be sent in a single step. In general, the PUF-HB protocol is almost identical to the HB$^+$ protocol. The main difference introduced in the PUF-HB protocol is to substitute the inner product $a \cdot y$ where $y \in \{0,1\}^k$ with $\text{PUF}_{Y,s}(a)$. As we will see in the proof of Theorem 5.3.3 this substitution will not affect the security features introduced by the HB$^+$ protocol. However, as indicated in the previous sections this substitution will make the tag tamper-resilient, and will simultaneously help resist the known man-in-the-middle attack on the HB$^+$ protocol

[34].

5.3 Security Against Active Attacks

In this section, we reduce the security of the PUF-HB protocol in the active attacker model (which does not include man-in-the-middle attacks) to solving the LPN problem. Note that as we pointed out earlier, the proof here closely follows the proof of Katz et al. on the security of the HB$^+$ protocol [53]. However, due to the nature of the PUF circuit, a very simple part of the original proof in [53], becomes much more complex in our protocol. Such a difference is a reflection of the change from a simple binary inner product to a PUF operation. For a more elaborate explanation of the proof see [53] where the authors prove security for the parallel execution case with $\epsilon < 1/4$. Also see [54] for a similar proof when $\epsilon < 1/2$. Moreover, in the original paper where the HB$^+$ protocol was proposed [50] the authors provide an elegant proof of security against active attacks. The proof in [50] can easily be modified to prove the security of the PUF-HB protocol. However, for simplicity and completeness we use the proof in [53].

We start by quoting the following definitions directly from [53]. Let $A_{x,\epsilon}$ denote an LPN$_\epsilon$ oracle which outputs a $k+1$ bit string such that x is chosen uniformly at random from $\{0,1\}^k$ and $\epsilon \in \left(0, \frac{1}{2}\right)$. The output of the oracle is the string

$$(a, a \cdot x \oplus \nu) \ .$$

Where a is chosen uniformly at random from $\{0,1\}^k$, and $\nu = 1$ with probability ϵ and $\nu = 0$ with probability $1 - \epsilon$. We also define U_{k+1} to be the uniform oracle with outputs from the uniform distribution over $\{0,1\}^{k+1}$. We say that an algorithm M can (t, q, δ) solve the LPN$_\epsilon$ problem if

$$\Pr\left[M^{A_{x,\epsilon}}(1^k) = x\right] \geq \delta \ ,$$

provided that M runs in time t and uses q queries to the oracle $A_{x,\epsilon}$. The main theorem of this chapter relies on the following lemma originally due to Regev [85] and reproven in [53].

Lemma 5.3.1. ([53]) *If there exists an algorithm D making q oracle queries, and running in time t, such that*

$$\left|Pr\left[D^{A_{x,\epsilon}}(1^k)=1\right]-Pr\left[D^{U_{k+1}}(1^k)=1\right]\right|\geq\delta\ ,$$

then there exist an algorithm M making $q'=O(q\cdot\delta^{-2}\log k)$ oracle queries and running in time $t'=O(t\cdot k\delta^{-2}\log k)$, such that

$$Pr\left[M^{A_{x,\epsilon}}(1^k)=x\right]\geq\delta/4\ .$$

Before we prove the main theorem of the chapter we try to give some intuition on why the proof in [53] can be applied to prove the security of the PUF-HB protocol. In addition, we state two technical lemmas which are needed for the proof of the main theorem.

First, note that the function computed in the HB$^+$ protocol is $z=b\cdot x\oplus a\cdot y$ whereas the function computed in the PUF-HB protocol is $z=b\cdot x\oplus PUF_{Y,s}(a)$. Moreover, Theorem 5.3.3 states that if there exists an algorithm $\mathcal{A}$ which starts by impersonating a reader to interact with an honest tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ (learning phase), and then impersonates a tag to interact with an honest reader (impersonation phase) therefore achieving PUF-HB$\big(\mathcal{A},\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big)=1$ with high probability, the algorithm $\mathcal{A}$ can also be used to distinguish between an LPN oracle and a uniform oracle. As implied by Lemma 5.3.1, then algorithm $\mathcal{A}$ can be used to solve the LPN problem.

In the learning phase, the algorithm $\mathcal{A}$ will expect to interact with an honest tag. In the proof, we impersonate such a tag, and use the oracle outputs to substitute the $b\cdot x$ part of a tag's response. Note that this part of the response is in common between the HB$^+$ protocol and the PUF-HB protocol. Now since we will use an oracle

for part of the response, this means that we will have no control over x which will be determined by the oracle. At the same time we will have full control over Y, s (y in the HB$^+$ protocol).

In the impersonation phase $\mathcal{A}$ will take the role of a tag interacting with an honest reader, and will therefore attempt to correctly compute the responses $z_{(i)}$. However, the responses of $\mathcal{A}$ will depend on the interaction that took place in the learning phase. If we were successful in providing outputs which were consistent with some x, then $\mathcal{A}$ will be able to provide correct responses in the impersonation phase. This will be the case if the used oracle was an LPN oracle. On the other hand, if the oracle was the uniform oracle, then we will have failed in providing consistent outputs to $\mathcal{A}$. Consequently, the responses $z_{(i)}$ produced by $\mathcal{A}$ in the impersonating phase will be random. While this presents a technique to distinguish between an LPN oracle and a uniform oracle, nevertheless, we do not have a way to know if the responses $z_{(i)}$ were correct or not. Primarily because we have no knowledge of x. To resolve this issue we run the algorithm and acquire the responses for a set of challenges $\{a^1_{(i)}\}$ along with the $\{e^1_{(i)}\}$ bits, then we rewind the algorithm and acquire a second set of responses for a different set of challenges $\{a^2_{(i)}\}$ and the $\{e^2_{(i)}\}$ bits. Adding the two responses cancels out the effect of x and retains the effect of $\mathrm{PUF}_{Y,e^1_{(i)}\cdot s_i}(a^1_{(i)}) \oplus \mathrm{PUF}_{Y,e^2_{(i)}\cdot s_i}(a^2_{(i)})$ (this would be $a^1_{(i)} \cdot y \oplus a^2_{(i)} \cdot y$ for the HB$^+$ protocol). With this trick we will have complete knowledge over the remaining variables. Therefore, we can check whether the responses retuned by $\mathcal{A}$ were correct or not.

What remains to be shown is that given the inputs $a^1_{(i)}$ and $a^2_{(i)}$ and no other information, the algorithm $\mathcal{A}$ will not be able to predict the output bits $\mathrm{PUF}_{Y,e^1_{(i)}\cdot s_i}(a^1_{(i)})$ and $\mathrm{PUF}_{Y,e^2_{(i)}\cdot s_i}(a^2_{(i)})$. This is akin to asking the question: How much can be inferred about the output, by only knowing the input. In the HB$^+$ protocol this becomes a question of linear independence. However, in the case of PUF-HB the answer becomes much more complicated. This question was addressed in Proposition 4.2.1 and will be

further detailed in Lemma 5.3.2. In general, one can see that the function $\mathrm{PUF}_{Y,s}(a)$ (compared to $a \cdot y$) only becomes relevant toward the end of the proof of Theorem 5.3.3. In fact, it should be clear that there is a large family of functions that could be used in place of $\mathrm{PUF}_{Y,s}(a)$ or $a \cdot y$ while maintaining the security of the protocol. However, our choice of $\mathrm{PUF}_{Y,s}(a)$ was mainly motivated by hardware simplicity and tamper resilience.

The following lemma connects Proposition 4.2.1 to the main theorem.

Lemma 5.3.2. *Let $\mathcal{A}$ be an adversary who is given n strings $\{a_{(i)}\}_{i=1}^{n}$, where $a_{(i)}$ is chosen independently and uniformly at random from $\{0,1\}^k$. $\mathcal{A}$ also knows that s is chosen uniformly at random form $\{0,1\}^n$ and that $Y_0 = [y_1, \ldots, y_{k+1}]$ where $y_i \in N(0,1)$ for $i = 1, \ldots, k$ and $y_{k+1} = 0$. Let $z_{(i)} = PUF_{Y_0,s_i}(a_{(i)})$ then the bits $z_{(i)}$ will be uniform and independent (from the point of view of $\mathcal{A}$).*

Proof. To show that the bits $\{z_{(i)}\}_{i=1}^{n}$ are uniform and independent, we need to show that the probability of $z_{(i)} = 0$ for any $i \in [1, n]$ is 0.5 and that the probability of $z_{(i)} = z_{(j)}$ for any $i, j \in [1, n]$ is 0.5. It is clear that when $y_{k+1} = 0$ the output of a PUF will be balanced. Therefore, it is straightforward to see that when $\{a_{(i)}\}_{i=1}^{n}$ are independent and chosen uniformly at random the bits $\{z_{(i)}\}_{i=1}^{n}$ will also be uniformly distributed. What remains to show is that there is no correlation between the bits $\{z_{(i)}\}_{i=1}^{n}$.

From Proposition 4.2.1 and as can be seen from Equation 4.4 the probability of any two PUF outputs being equal (or not equal) depends on the Hamming distance d between $P_{(i)} = Ua_{(i)}$ and $P_{(j)} = Ua_{(j)}$ and not the specific values of $a_{(i)}$ and $a_{(j)}$. Furthermore, we can deduce from Equation 4.4 that

$$\overline{F}_k(d) = \Pr[z_{(i)} \neq z_{(j)}] = 1 - F_k(d) = \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{k-d}}\right) .$$

The two probability distributions $F_k(d)$ and $\overline{F}_k(d)$ are reflections of each other around $d = \frac{k}{2}$. Therefore, $\overline{F}_k(d) = F_k(k-d)$. This means that when the probability distribu-

tion of the Hamming distance between $P_{(i)}$ and $P_{(j)}$ is symmetrized around $\frac{k}{2}$, then $z_{(i)}$ and $z_{(j)}$ will be uncorrelated. To prove the rest of the lemma, we only need to show that the probability distribution of the Hamming distances between the different $P_{(i)}$ strings will be symmetric around $\frac{k}{2}$.

The n bit strings $\{a_{(i)}\}_{i=1}^{n}$ given to $\mathcal{A}$ will induce $\frac{n(n-1)}{2}$ different Hamming distances between the corresponding $P_{(i)}$ strings. Recall from the lemma that $z_{(i)} = PUF_{Y_0,s_i}(a_{(i)})$. Therefore, as indicated by Equation 5.1 the PUF circuit will invert the $P_{(i)}$ strings based on the value of the bit s_i. From $\mathcal{A}$'s perspective s is chosen uniformly at random from $\{0,1\}^n$. Therefore, each of the $P_{(i)}$ strings will be inverted with probability 0.5. For any two strings $P_{(i)}$ and $P_{(j)}$, if both of the strings or neither of them are inverted the Hamming distance d will not be affected. On the other hand, if only one of the two strings is inverted then the Hamming distance d will become $k-d$. Therefore, the Hamming distance between any two strings $P_{(i)}$ and $P_{(j)}$ will be d with probability 0.5 and will be $k-d$ with probability 0.5. This distribution is symmetric around $\frac{k}{2}$. □

We are now ready to prove the main theorem.

Theorem 5.3.3 (Compare to Theorem 3 in [53])**.** *If there exists an adversary $\mathcal{A}$ interacting with a tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ in at most q executions of the PUF-HB protocol (possibly concurrently), running in time t such that*

$$Pr\left[\mathcal{A}^{\mathcal{T}_{(n,x,Y,s,\epsilon)}}(1^k) : PUF\text{-}HB\left(\mathcal{A}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\right) = 1\right] \geq \delta \quad ,$$

then these exist an algorithm D making $q \cdot n$ oracle queries, running in time $O(t)$, and such that

$$\left|Pr\left[D^{A_{x,\epsilon}}(1^k) = 1\right] - Pr\left[D^{U_{k+1}}(1^k) = 1\right]\right| \geq \delta^2 - 2^{-n/2}\sum_{i=0}^{2u}\binom{n/2}{i} - e^{-\frac{n}{8}}$$

Therefore, for any $\epsilon < \frac{1}{8}$ there is an appropriate choice of n, u such that the last two terms become negligible, and thus we can conclude that the PUF-HB protocol is secure

assuming the hardness of the LPN$_\epsilon$ problem.[4]

Proof. To prove the theorem we show a construction of the algorithm D. As stated by the theorem, D is given access to an oracle returning $(k+1)$-bit strings which can be broken to $(\bar{b}, \bar{z})$, where $\bar{b} \in \{0,1\}^k$ and $\bar{z} \in \{0,1\}$. D proceeds as follows:

1. D starts by choosing vectors Y_0 and s such that y_{k+1} is set to 0, and then the k remaining y_i values are chosen from $N(0,1)$. The bit-string s is chosen uniformly at random from $\{0,1\}^{2\times n}$. D runs the algorithm $\mathcal{A}$ which will expect to interact with a PUF-HB tag. In order to impersonate a real tag, D does the following to simulate a basic authentication step: D starts by obtaining a $k+1$ bit string $(\bar{b}_{(i)}, \bar{z}_{(i)})$ from the oracle, and then sends $\bar{b}_{(i)}$ to $\mathcal{A}$ as the initial b in Protocol 1. $\mathcal{A}$ will reply with a challenge $\bar{a}_{(i)}$ and the bits $\bar{e}_{(i)}$. Next, D computes $z_{(i)} = \bar{z}_{(i)} \oplus \mathrm{PUF}_{Y_0, \bar{e}_{(i)} \cdot s_i}(\bar{a}_{(i)})$ and sends $z_{(i)}$ back to $\mathcal{A}$. D repeats this step $q \cdot n$ times.

2. In the second phase of the algorithm, $\mathcal{A}$ tries to impersonate an honest tag. Looking at the parallel execution of PUF-HB, $\mathcal{A}$ starts by sending $b_{(1)}, \ldots, b_{(n)} \in \{0,1\}^k$ to a reader. Next, D randomly choses $a^1_{(1)}, \ldots, a^1_{(n)} \in \{0,1\}^k$ and $e^1_{(1)}, \ldots, e^1_{(n)} \in \{0,1\}^2$ and send them back to $\mathcal{A}$, which will in turn respond with the bits $z^1_{(1)}, \ldots, z^1_{(n)}$. D then rewinds $\mathcal{A}$ and sends randomly chosen $a^2_{(1)}, \ldots a^2_{(n)} \in \{0,1\}^k$ and $e^2_{(1)}, \ldots, e^2_{(n)} \in \{0,1\}^2$. $\mathcal{A}$ will respond with $z^2_{(1)}, \ldots, z^2_{(n)}$. Note that since the algorithm was rewound the same b values will be sent by $\mathcal{A}$.

3. D calculates $z^{\oplus}_{(i)} = z^1_{(i)} \oplus z^2_{(i)}$ and lets $Z^{\oplus} = (z^{\oplus}_{(1)}, \ldots, z^{\oplus}_{(n)})$. D also computes $\hat{z}_{(i)} = \mathrm{PUF}_{Y_0, e^1_{(i)} \cdot s_i}(a^1_{(i)}) \oplus \mathrm{PUF}_{Y_0, e^2_{(i)} \cdot s_i}(a^2_{(i)})$ and lets $\hat{Z} = (\hat{z}_{(1)}, \ldots, \hat{z}_{(n)})$. D outputs 1 iff $Z^{\oplus}$ and $\hat{Z}$ differ in at most $2u$ positions.

[4]Note that by parametering the length ℓ of the strings $s_i \in \{0,1\}^\ell$ and $e_{(i)} \in \{0,1\}^\ell$ we may achieve some flexibility in the parameters, i.e. $\epsilon < 0.25 - 2^{-(\ell+1)}$.

Now we analyze D:

Case 1: If the oracle used by D was the uniform oracle U_{k+1}, then the outputs $\bar{z}$ given to D in step 1 were uniformly distributed and independent of everything. This means that the bits $z_{(i)}$ which D sent back to $\mathcal{A}$ in step 1 were also uniformly distributed and independent of everything. Therefore, by the end of step 1 $\mathcal{A}$ has no information about either Y_0 or s. All $\mathcal{A}$ receives in the second step are the inputs $\{a^1_{(i)}\}_{i=1}^n$, $\{e^1_{(i)}\}_{i=1}^n$ and $\{a^2_{(i)}\}_{i=1}^n$, $\{e^2_{(i)}\}_{i=1}^n$. All of these inputs are uniformly and independently distributed. As shown in Lemma 5.3.2 each of the two calculated output strings $\{\mathrm{PUF}_{Y_0,e^1_{(i)}\cdot s_i}(a^1_{(i)})\}_{i=1}^n$ and $\{\mathrm{PUF}_{Y_0,e^2_{(i)}\cdot s_i}(a^2_{(i)})\}_{i=1}^n$ will be uniform over $\{0,1\}^n$ (from the point of view of $\mathcal{A}$). However, when we add these two variables and obtain $\hat{Z}$ the individual bits of the output will not always be independent. The affect of the s_i bits will actually cancel out from both terms when $e^1_{(i)} = e^2_{(i)}$ with probability 0.25. To simplify the proof, we assume in this case that the outputs are completely dependent. Using the Chernoff approximation we bound the probability of observing more than $n/2$ dependent output bits by $e^{-\frac{n}{8}}$. The probability that $Z^{\oplus}$ and $\hat{Z}$ differ in at most $2u$ positions is exactly $2^{-n/2}\sum_{i=0}^{u}\binom{n/2}{i}$. We conclude that D outputs 1 in this case with probability at most $2^{-n/2}\sum_{i=0}^{2u}\binom{n/2}{i} + e^{-\frac{n}{8}}$.

Case 2: If D is using the oracle $A_{x,\epsilon}$ for some random x, the simulation provided by D in the first phase will be perfect and therefore $\mathcal{A}$ will be able to impersonate an honest tag with probability at least δ. Let w be the randomness used in the first phase of running $\mathcal{A}$, then we denote the probability that $\mathcal{A}$ succeeds in impersonating an honest tag for a fixed choice of w by δ_w. Now since we rewind the algorithm, the probability that $\mathcal{A}$ succeeds in both rounds is δ_w^2. Let $E(\delta_w^2)$ denote the expected value of δ_w^2 over all possible randomness w, then we have

$$E(\delta_w^2) \geq E(\delta_w)^2 = \delta^2,$$

where the square is taken out of the expected value operator using Jensen's inequality.

Now assuming that $\mathcal{A}$ succeeds in impersonating an honest tag for both rounds, then we would expect each of the response strings $z^1_{(1)}, \ldots, z^1_{(n)}$ and $z^2_{(1)}, \ldots, z^2_{(n)}$ to have at most u errors. Therefore $Z^{\oplus}$ will in turn have at most $2u$ errors.[5] When we add $z^1_{(i)}$ and $z^2_{(i)}$ to generate $z^{\oplus}_i$ we are canceling the effect of $b \cdot x$ and therefore we are left with $z^{\oplus}_{(i)} = \mathrm{PUF}_{Y_0, e^1_{(i)} \cdot s_i}(a^1_{(i)}) \oplus \mathrm{PUF}_{Y_0, e^2_{(i)} \cdot s_i}(a^2_{(i)})$. With the exception of the $2u$ errors in $Z^{\oplus}$, the strings $\hat{Z}$ and $Z^{\oplus}$ are calculating the same output. We conclude that D outputs 1 in this case with probability at least δ^2. □

This concludes our security proof, and shows that the PUF-HB protocol is secure against an active attacker provided that the LPN problem is hard to solve.

5.4 Man-in-the-Middle Attacks

The main weakness of the HB$^+$ protocol is the man-in-the-middle attack proposed in [34]. Briefly summarized, in this attack an adversary replaces all the challenges $\{a_{(j)}\}_{j=1}^n$ sent from the reader in a single authentication session by $\{a_{(j)} \oplus w\}_{j=1}^n$ where $w \in \{0,1\}^k$. The attacker knows that the challenges will interact with the secret y through $a_{(j)} \cdot y$. At the end of the n rounds, if the reader authenticates the tag, the adversary can deduce with very high probability that his changes did not affect the responses of the tag, and therefore $w \cdot y = 0$. On the other hand, if the reader rejects the tag, then the adversary will know with a very high probability that $w \cdot y = 1$. Repeating the same attack k times will allow the adversary to collect k linear equations in y. The adversary can use Gaussian elimination to recover the secret y. Similarly, the same attack can be carried out to deduce the other secret string x.

In the PUF-HB scheme this particular man-in-the-middle attack will not succeed due to the non-linearity of the PUF function. From Proposition 4.2.1 we know that

[5] This worst case happens when the u errors in $z^1_{(i)}$ and $z^2_{(i)}$ affect completely different bits.

the only type of correlations that the attacker can exploit are those related to the Hamming distance between the different input strings a. However, we saw in Lemma 5.3.2 that with the secret string s the Hamming distance information is masked for a single authentication session. It is still possible that an adversary can exploit Hamming distances between different sessions to launch an attack. Another potential point of weakness is the linearity in the $b \cdot x$ portion of the PUF-HB protocol. To protect against simple attacks exploiting this linearity, a second PUF circuit can be used with the b vector as its input. We label such a protocol PUF^2-HB, since it will essentially be identical to Protocol 1, with the only difference in the z bit calculated by the tag, which becomes

$$z_{(i)} = b_{(i)} \cdot x \oplus PUF_{Y_1,s_i}(a_{(i)}) \oplus PUF_{Y_2}(b_{(i)}) \oplus \nu \tag{5.2}$$

where the shared secret becomes (x, Y_1, Y_2, s).

5.5 Hardware Security

In the previous section we discussed the security of the proposed scheme under abstract security models. However, in recent years we have seen numerous side-channel attacks which directly target the hardware implementation. The PUF paradigm was aimed at protecting against active side-channel attacks. In the PUF-HB protocol there are only two strings that are to be stored by the tag: x and s. The secret Y is not really stored since it is part of the characteristics of the circuit itself. In Chapter 4 we discussed the resilience of PUF circuits against hardware attacks. In particular, PUF prevents an attacker from measuring the y_i parameters directly via a physical measurement. Any major changes to the surrounding temperatures or voltage levels, or any attempt to forcefully read the value of these registers will induce a change to the PUF, therefore changing the identity of the tag.

What is more impressive about the PUF circuit is that it even protects neighboring

components. This is achieved by placing all registers containing the secret strings x and s sufficiently close to the PUF circuit. Such a level of security ensures that even when the tag itself is compromised, an attacker cannot impersonate this tag by extracting the secrets from the hardware. We can not quantify this ability of a PUF.

Naturally, one might be concerned about how that will affect the modelability of the PUF in the pre-deployment phase. Before deployment of the tag, the registers are initialized to their secret values. Afterward, with the knowledge of the secret vectors the reader can develop an accurate model for the PUF circuit. Note that modeling the PUF is even possible in the existence of noise [8]. Hence, the sensitivity of the PUF does not prevent the owner from modeling it.

Finally we would like to underline that PUF-HB is not inherently protected against passive attacks, e.g. Simple Power Analysis and Differential Power Analysis. Although not trivial, side-channel profiles may be utilized to recover the secret values. If passive side-channel attacks are a concern, standard IC level power balancing techniques [92, 93, 84] must be employed. Although effective, these techniques tend to incur significant area overhead. An alternative approach would be to modify the implementation to balance the power consumption.

5.6 Summary

In this chapter we merged the PUF authentication scheme with the HB-based authentication protocol, with the goal of producing a hybrid protocol which enjoys the advantages of both schemes while improving the level of security. The main contribution of this chapter is the proposed PUF-HB authentication scheme which is tamper resilient, and at the same time provably secure against active attacks in the detection based model. In addition, the proposed protocol resists the known man-in-the-middle attacks against the HB^+ scheme. Although this proposal enjoys a number

of advantages, it still raises a number of concerns preventing it from being considered an unclonable function. First, the man-in-the-middle security is not proven for all attackers. Second, the side-channel security is also a concern as it is not clear whether there is information leakage. In the next chapter we will present another variant of PUF-HB which targets a simple security reduction with an efficient hardware implementation.

Chapter 6

HB+PUF

In this chapter we present a proof of concept implementation for HB+PUF, a variant of PUF-HB. The HB+PUF protocol enjoys the same properties of PUF-HB with a much simpler security reduction. Our implementation takes advantage of the PUF circuit in order to produce the random bits typically needed for an HB-based authentication scheme. Note that the existence of a random number generator (RNG) is assumed in all HB-based protocols without taking into account the complexity of such a device. The overall circuit is shown to occupy a few thousand gates. This small gate count is achieved by using the tristate PUF and by serializing the operation. We note here that although the HB+PUF proposal seems stronger than PUF-HB in terms of resisting man-in-the-middle attacks it will not change the general security claims of PUF-HB.

The remainder of this chapter is organized as follows. In Section 6.1 we define our notation and describe our proposed protocol. The security analysis is presented in Section 6.2. In Section 6.3 we describe our RNG construction and present analysis of its randomness. Section 6.4 describes the proof of concept implementation of the entire circuit. Finally, we present the summary in Section 6.5.

6.1 New Authentication Family: HB+PUF

In this section we present the proposed protocol. We will use $\mathcal{R}$ to denote the reader and $\mathcal{T}$ to denote the tag. n_1 and n_2 will be our security parameters. $\mathcal{T}$ and $\mathcal{R}$ are both characterized by the set of variables $(k, s_1, s_2, x, Y, \epsilon_P, \epsilon, u)$ where $s_1, x \in \{0,1\}^{n_1}$, $s_2 \in \{0,1\}^{n_2}$ and $Y = [y_1, y_2, \ldots, y_{n_1+1}]$ such that $y_i \in N(0,1)$ where $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. The noise parameters are $\epsilon, \epsilon_P \in \left(0, \frac{1}{2}\right)$. We use k to denote the number of rounds required for authentication. The last variable u is an integer in the range $[0, n]$ such that $\epsilon_f k \leq u$, where $\epsilon_f = \epsilon_P + \epsilon - 2\epsilon_p\epsilon$ denotes the total noise in the scheme.

Next, we introduce an enhancement to the delay-based PUF. We use an n-bit non-zero binary string x to implement a linear bijection on the input challenge before it is fed into the PUF. Let a be the challenge string sent to the PUF. To produce the actual challenge string a' we treat x and a as elements of a finite field $GF(2^n)$ and compute the product $a' = xa \in GF(2^n)$. We next define a new PUF equation which takes this enhancement as well as the error in modeling the PUF into consideration [1]

$$\mathrm{PUF}_{Y,x,\epsilon_p}(a) = \mathrm{PUF}_Y(xa) \oplus \nu \quad , \tag{6.1}$$

where $\nu = 1$ with probability ϵ_P and $\nu = 0$ with probability $1 - \epsilon_P$. The field multiplication may be implemented with low footprint using a simple linear feedback shift register (LFSR) based serial modular polynomial multiplier circuit. The choice of generating polynomial makes no difference in terms of the properties of the PUF device. Hence, for efficiency, low weight polynomials (e.g. trinomials) may be used in the implementation [6]. Keep in mind that in this chapter we will use tristate PUFs.

With this notation we describe the basic authentication step. In every round, $\mathcal{T}$ randomly generates $b \in \{0,1\}^{n_2}$ and sends it to $\mathcal{R}$. Upon reception $\mathcal{R}$ replies with

[1] While this enhancement does not prevent modeling attacks, it will indeed have an affect on the man-in-the-middle attacks as we will see in Section 6.2.

Protocol 2 (HB+PUF): $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle$
1. $\mathcal{R}_\sigma$ sets the counter $c = 0$
2. $\mathcal{E}$ sends $b \in \{0,1\}^{n_2}$ to $\mathcal{R}_\sigma$
3. $\mathcal{R}_\sigma$ choses $a \in \{0,1\}^{n_1}$ uniformly at random and sends it to $\mathcal{E}$
4. $\mathcal{E}$ sends z to $\mathcal{R}_\sigma$
5. if $z \neq a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,0}(a)$ then $c = c + 1$
6. Steps 2 through 5 are repeated for k iterations
7. If $c \leq u$ then $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle = 1$, otherwise it equals 0.

Table 6.1: HB+PUF Protocol

the challenge $a \in \{0,1\}^{n_1}$. Finally, $\mathcal{T}$ computes

$$z = a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,\epsilon_p}(a) \oplus \nu \ , \tag{6.2}$$

where $\nu = 1$ with probability ϵ and 0 with probability $1 - \epsilon$. Notice that this is very similar to the basic authentication step in HB$^+$. The only difference is that here we add a PUF operation. In order for $\mathcal{R}$ to authenticate $\mathcal{T}$, the same basic authentication step is repeated for k rounds. In every round $\mathcal{R}$ checks to see if $\mathcal{T}$'s response is equal to $(a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,0}(a))$. If the response is not equal to this term, $\mathcal{R}$ marks the response wrong. At the end of the k^{th} round, $\mathcal{R}$ authenticates $\mathcal{T}$ if and only if the number of wrong responses is less than u.

In general, any entity can interact with the reader and try to impersonate an honest tag. To capture such interaction, let $\mathcal{E}$ be any entity trying to authenticate itself to the reader $\mathcal{R}_\sigma$ characterized by $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$. Following the notation in [53] we define $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle := 1$ iff $\mathcal{E}$ is authenticated by the reader, and is equal to 0 otherwise. The following protocol formalizes this interaction:

6.2 Security Analysis

In this section we show that the proposed protocol HB+PUF is at least as secure as the HB$^+$ protocol. We also discuss security against man-in-the-middle attacks. Finally, we consider the parameter selection to obtain a secure implementation. The reduction from HB+PUF to HB$^+$ is in fact very simple. As can be seen from Equation 6.2, the HB+PUF protocol utilizes all the terms of HB$^+$, and only adds a PUF operation. Therefore, it should be expected that the HB+PUF protocol can not be less secure than the HB$^+$ protocol. We now formalize this intuition by showing that any algorithm capable of successfully attacking the HB+PUF protocol can be used to successfully attack HB$^+$. The HB$^+$ protocol uses a tag $\mathcal{T}_\tau^+$ and a reader $\mathcal{R}_\tau^+$ both of which can be characterized by the string of variables $\tau = (k, s_1, s_2, \epsilon, u)$. The variables in τ are defined as we have done for the HB+PUF variables in Section 6.1. We also use $\langle \mathcal{E}, \mathcal{R}_\tau^+ \rangle$ to indicate an authentication session between any entity $\mathcal{E}$ and an HB$^+$ reader $\mathcal{R}_\tau^+$ using the HB$^+$ protocol. Similar to Protocol 1, $\langle \mathcal{E}, \mathcal{R}_\tau^+ \rangle = 1$ when the reader authenticates and 0 otherwise. This notation mostly follows the work presented in [53]. Recall from the previous section that in the HB+PUF protocol we use a tag $\mathcal{T}_\sigma$ and a reader $\mathcal{R}_\sigma$ both of which can be characterized by the string of variables $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$. We next prove the reduction in the active attacker model used to prove the security of the HB$^+$ protocol. In this model the attacker interacts with the tag in a learning session before he attempts to impersonate as the tag to an honest reader[2].

Theorem 6.2.1. *Let $\mathcal{A}$ be an algorithm which interacts with an honest HB+PUF tag $\mathcal{T}_\sigma$ for q authentication sessions to achieve* $\Pr[\langle \mathcal{A}, \mathcal{R}_\sigma \rangle = 1] > \delta$, *where $\mathcal{R}_\sigma$ is an honest HB+PUF reader. Then, there exists an algorithm $\mathcal{A}'$ which can interact with any HB$^+$ tag $\mathcal{T}_\tau^+$ for q authentication sessions to achieve* $\Pr[\langle \mathcal{A}', \mathcal{R}_\tau^+ \rangle = 1] > \delta$, *where*

[2]We only need HB$^+$ to prove the reduction to the LPN problem. However, HB$^+$ is reduced to the LPN problem under the same attacker model used here.

$\mathcal{R}_\tau^+$ is an honest HB$^+$ reader.

Proof. The basic operation of $\mathcal{A}'$ is to map a given $\tau = (k^+, s_1^+, s_2^+, \epsilon^+, u^+)$ characterizing the HB$^+$ tag and reader to $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$ used to characterize an HB+PUF tag and reader. Note that all the variables in the HB$^+$ protocol are still used in the same manner in the HB+PUF protocol. Therefore, we can create $\sigma^+ = \left(k = k^+, s_1 = s_1^+, s_2 = s_2^+, x, Y, \epsilon = \epsilon^+, \epsilon_p = 0, u = u^+\right)$. The variable x is chosen randomly to be any string in $\{0,1\}^{n_1}$. The $(n_1 + 1)$ real vector Y is chosen such that $y_i \in N(0,1)$. $\mathcal{A}'$ runs as follows: It initializes $\mathcal{A}$ and allows it to carry its communication with $\mathcal{T}_\tau^+$. In particular, $\mathcal{A}'$ pass the vector b sent by $\mathcal{T}_\tau^+$ to $\mathcal{A}$ which will reply with the vector a. Again $\mathcal{A}$ passes a back to $\mathcal{T}_\tau^+$. Finally, when $\mathcal{T}_\tau^+$ returns its response z, $\mathcal{A}'$ returns $\hat{z} = z \oplus \mathrm{PUF}_{Y,x,0}(a)$ to $\mathcal{A}$. The same step is followed for all q authentication rounds between $\mathcal{T}_\tau^+$ and $\mathcal{A}$. When $\mathcal{A}'$ wants to authenticate itself to $\mathcal{R}_\tau^+$, it again runs $\mathcal{A}$ in its authentication phase. $\mathcal{A}$ will start by sending the random string $b^{(i)}$. $\mathcal{A}'$ will pass the string directly to $\mathcal{R}_\tau^+$ which will respond with the vector $a^{(i)}$. $\mathcal{A}'$ passes $a^{(i)}$ back to $\mathcal{A}$. Finally, when $\mathcal{A}$ returns its response $z^{(i)}$, $\mathcal{A}'$ returns $\hat{z}^{(i)} = z^{(i)} \oplus \mathrm{PUF}_{Y,x,0}(a^{(i)})$ to $\mathcal{R}_\tau^+$. The algorithm $\mathcal{A}'$ repeats these steps for all k rounds of the authentication session, such that $i = 1 \ldots k$.

To see why this will actually work, notice that in the first q rounds $\mathcal{A}$ is getting the response $\hat{z}$ which is effectively responses from a tag $\mathcal{T}_{\sigma^+}$. This means that at the end of the q authentication sessions $\mathcal{A}$ will have effectively communicated with $\mathcal{T}_{\sigma^+}$. In the authentication phase, when $\mathcal{A}'$ tries to authenticate itself to $\mathcal{R}_\tau^+$, it uses the algorithm $\mathcal{A}$ which will be trying to authenticate itself to $\mathcal{R}_{\sigma^+}$. Assuming that $\mathcal{A}$ will succeed in impersonating $\mathcal{T}_{\sigma^+}$ with probability larger than δ, then the responses returned by $\mathcal{A}$ which are $z^{(i)}$ will match the responses of an honest tag with probability larger than δ. However, this immediately implies that the responses returned by $\mathcal{A}'$ to $\mathcal{R}_\tau^+$ which are $\hat{z}^{(i)}$ and which differ from $z^{(i)}$ with the term $\mathrm{PUF}_{Y,x,0}(a^{(i)})$ will match the responses of an honest tag with probability larger than δ. Therefore, $\mathcal{A}'$ should also

succeed in impersonating a tag $\mathcal{T}_\tau^+$ with probability larger than δ. □

We point out that in the PUF-HB scheme the security reduction is much more involved since the secret s_1 is replaced by the PUF operation. Theorem 6.2.1 poses an immediate question of how the HB+PUF protocol behaves in relation to the the known man-in-the-middle attack against HB$^+$ [34]. As we pointed out earlier, one of the main reasons for such an attack to work is the linearity of the inner product operation. In our scheme the challenges $a^{(j)}$ are not only subjected to the inner product operation $a^{(j)} \cdot s_1$, but also to a PUF operation $a^{(j)} \cdot s_1 \oplus \text{PUF}_{Y,x,\epsilon_p}(a^{(j)})$. With both operations being used, an adversary will need to find a way to modify the challenges such that he can deduce information about each of the two operations separately. To see why a PUF operation will help against the man-in-the-middle attacks, notice that on one hand the PUF is inherently non-linear due to the sign operation. Therefore, it will prevent against any simple man-in-the-middle attack trying to explore linearity, such as the attack in [34]. On the other hand, it has been shown in Chapter 4 that the probability distribution of two different challenges $a^{(1)}$ and $a^{(2)}$ yielding the same output from a PUF operation, will only depend on the Hamming distance between $a^{(1)}$ and $a^{(2)}$. This means that any successful man-in-the-middle attack would have to exploit the Hamming distances between different challenges. However, recall from the end of Section 6.1 that the PUF circuit used in HB+PUF implements a field multiplication over $GF\,(2^{n_1})$ with the secret string x. This multiplication will essentially obfuscate the Hamming distance between different challenges. Therefore, the attacker's ability to deduce any correlation between the inputs and the outputs of the PUF is eliminated. In conclusion, we claim that the addition of a PUF to the authentication scheme will in effect render a man-in-the-middle attack quite difficult.

Note that here we are talking with respect to the GRS-MIM model introduced in [33]. To protect against the most general class of man-in-the-middle attacks, we

suggest adding a second PUF circuit to operate on the $b^{(j)}$ strings sent by the tag. In such a scheme the response of the tag would be

$$z = a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y_1,x_1,\epsilon_{p1}}(a) \oplus \mathrm{PUF}_{Y_2,x_2,\epsilon_{p2}}(b) \oplus \nu \ . \tag{6.3}$$

The suggested scheme will be more demanding in terms of hardware and power. However, we predict that it could be resilient against man-in-the-middle attacks.

We finish this section by discussing security parameters for an implementation of the design. As shown by Theorem 1 our protocol is at least as secure as the HB$^+$ protocol, which in turn is at least as hard as solving the LPN problem. All with respect to the active attacker model. In [69] the authors give a careful study of the BKW algorithm for solving the LPN problem. They conclude that the parameters first introduced for the HB$^+$ protocol by [50] and then by [53] do not provide sufficient security. In our implementation we follow the new parameters suggested by [69] and later adopted by [33]. To achieve 80-bits of security we choose $n_1 = 80, n_2 = 512$, $\epsilon_f = 0.15$ and $k = 200$. ϵ_f is not a separate parameter but rather a result from ϵ_p and ϵ. In our implementation we will have $\epsilon_p = 0.15$ and $\epsilon = 0$.

6.3 PUF-based RNG

In Chapter 4 we discussed the inherent metastability in a PUF circuit. As we pointed out earlier, these metastable states result from either environmental fluctuations, or race conditions which occur between the two propagating signals inside a PUF. In this section, we outline how metastability could be used to generate random bits. We note here that using a PUF circuit as an RNG is not a new idea. It has been previously proposed in [77]. In their design the authors use an LFSR to generate a stream of challenges. Each challenge is fed to the PUF multiple times in order to decide whether the challenge is metastable or not. Finally, the metastable outputs are used to extract randomness. In our approach, we take advantage of a PUF feedback

setting. This approach will essentially remove any need to separately check each challenge for metastability. Therefore, decreasing the control logic, and increasing the throughput.

Our RNG design is based on a shift register feeding a PUF circuit in parallel. As we have concluded in Section 6.2 the size of the PUF and thus the size of the shift register will be 80 bits. The register is initialized to a random bit string. At every clock cycle the output of the PUF is fed back to the most significant bit of the shift register, while the least significant bit is discarded. This mode of operation will ensure a continuous stream of bits. Without metastability no randomness is expected to come out of this construction. Therefore, to assess the generated randomness we need to get a good estimate on the ratio of metastable points.

In order to get an estimate for the metastability ratio, we implemented the PUF circuit on a Xilinx XC2VP30 FPGA. In typical PUF implementations, extra precautions are taken to prevent metastability. However, we are interested in having a high level of metastability. This is the case, since we use the PUF in a *noisy* authentication scheme, and as an RNG. To help induce a higher level of metastability we allow close adjacency between the PUF circuit and other parts of the implementation. We carried out a restart test by collecting 1000 different bit streams. Each bit stream was collected after the system was reseted and initialized to the same state. In a completely stable system, these bit streams would have to be identical. However, in a metastable system, every time a metastable point occurs these streams are expected to break into two groups. With each group following a different choice of the metastable point. After tracking all the bit streams we found that after 6400 bits all the 1000 streams were in completely different states, therefore suggesting the occurrence of 1000 metastable points. This yields an overall metastability ratio of about 15%. With this ratio, we can insure that the output always contains a metastable point by XOR-ing every 8 consecutive bits and using the result as the output of the

Test Name	Proportion
Frequency	100%
Frequency within block	100%
Longest run of ones in block	95%
Cumulative sum	100%
Runs	100%
Discrete Fourier Transform	100%
Non-overlapping template matching	95%
Overlapping template matching	97.5%
Maurer's Universal	100%
Approximate Entropy	97.5%
Serial	97.5%
Lempel-Ziv Complexity	100%

Table 6.2: NIST suite results for PUF-RNG

RNG.

To verify the statistical quality of the RNG output, we collected a large number of bit streams and analyzed them with the NIST statistical test suite. As recommended by the NIST tools, every bit stream contained $20,000$ points. The test suite reports a *proportion* value, which reflects the ratio of bit streams which actually passed this particular test. The final results we obtained are shown in Table 6.2. The NIST tools return a statistical result where even a true random number generator could fail in some of the runs. We can conclude from the shown results that the proposed RNG is a reasonably good generator.

6.4 Implementation

The authentication scheme presented in Section 6.1 is implemented as shown in Figure 6.1. The PUF circuit is positioned at the center of the implementation to ensure tamper-resilience for the entire circuit. As we have verified from our FPGA implementations of a PUF circuit, any change in the surrounding hardware to the PUF circuit will result in changing the PUF's internal variables. We point out here that

a PUF can easily protect against active side-channel attacks. However, for passive side-channel attacks a designer might have to resort to standard power balancing techniques [92, 93, 84]. Although effective, these technique will incur about $200 - 400\%$ area overhead. A cost too high for lighweight implementations. Our authentication architecture runs in two different modes of operation during the entire protocol.

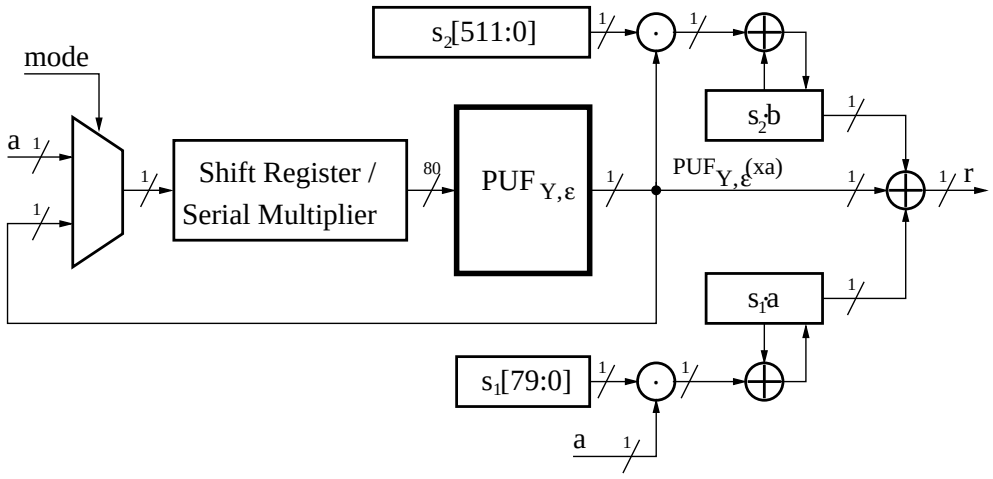


Figure 6.1: HB+PUF Authentication Scheme

RNG Mode: In this mode the PUF circuit acts as a random number generator. As explained in Section 6.3 the random string $b \in \{0,1\}^{512}$ is achieved using a shift register along with the PUF circuit. This shift register is initialized with the initialization value (IV) stored in an $80 - bit$ ROM structure. The shift register will be serially initialized to (IV) in RNG mode. For the remainder of the RNG operation the serial input of the shift register will be fed back by the output of the PUF. Conveniently enough, we do not need to store the entire random string b generated by the PUF. As b is generated 1 bit at a time, we can serially compute the inner product $b \cdot s_2$, and at the same time serially transmit b to the reader. It is important to point out that in the RNG mode, the system will not be able to detect any active side-channel attacks.

With a stream of random bits, the attacker's effects are gone undetectable. This will not be a major problem since any invasive attack on the circuit will permanently affect the PUF circuit. Therefore, as soon as the circuit is back to PUF mode, the attack can be detected. In the case where more gates are dedicated for security purposes, two separate PUF circuit can be used for authentication and random number generation.

PUF Mode: In this mode we perform the serial field multiplication xa which will be the input of the PUF. The hardware component *Shift Register/Serial Multiplier* shown in Figure 6.1 is used for this multiplication. The serial input of this shift register comes from the input a which is serially received. The field multiplication is realized through an LFSR serial multiplier, and is carried out in parallel with the inner product operation $a \cdot s_1$. These two operations will operate serially and will take about 80 cycles. The result of the field multiplication xa is fed to the PUF as the challenge input. The response bit of the PUF is then XOR-ed with the inner products $a \cdot s_1$ and $b \cdot s_2$. Finally, the response of the entire circuit r is transmitted to the reader. Note from the last section that the ratio of metastability was about 15%. This matches the overall desired noise. Therefore, there will be no need for an added noise parameter ϵ.

To estimate the gate count, HB+PUF was developed into a Verilog modules and synthesized using the Synopses Design Compiler. In the synthesis we used the TSMC 0.13 μm ASIC library. The circuit components and their gate count are explained as follows:

-**ROM Structure:** The IV for the RNG and the private keys s_1 and s_2 are stored inside the hardware and they are unique for each tag. Instead of utilizing flip-flops to store these values, we designed a ROM structure for low-area storage. To minimize the area usage, we used a design similar to a look-up table. Separate architectures

are needed to store s_1, s_2 and IV. Since s_2 is 512 bits, s_1 and IV are 80 bits, we have 672 bits of total ROM area. Synthesis results show that 110 gates are required for this storage.

-**PUF Circuit:** In our authentication scheme, we utilize an 80-bit PUF circuit. As pointed out in Section 6.1 we use the tristate PUF implementation presented in Chapter 4. This particular PUF implementation is of interest due to its low-power and low-area features. When we used the tristate PUF design our synthesis results showed the area of the PUF to be 450 gates. However, with custom circuit design, where each tristate buffer utilized about a single gate this number was reduced to 160 gates.

-**Shift register/Serial Multiplier:** The shift register has a total size of 80 bits. The structure also contains a number of XOR gates used to achieve the field multiplication. In addition, 2-1 multiplexers are used to decide which inputs feed the individual flip-flops of the register. Synthesis results show that a total equivalent of 500 gates is needed for this structure.

-**Serial inner products:** The AND and XOR components shown in Figure 6.1 are utilized for serial inner product operations. The boxes labeled as $s_2 \cdot b$ and $s_1 \cdot a$ are single flip-flops storing the results of the inner products $s_2 \cdot b$ and $s_1 \cdot a$ of Protocol 1. They work as accumulators. In each clock cycle, one bit of s_2 and one bit of b pass through an AND gate and the result is XORed with the value in the accumulator flip-flop. The same procedure is repeated for s_1 and a. At the end, the results in the accumulator registers $s_2 \cdot b$ and $s_1 \cdot a$ are XOR-ed with the result of the $\mathrm{PUF}_{Y,\epsilon}(xa)$ and the result is sent to the output as r. The area for these operations is estimated at 50 gates.

-**Control logic:** The control logic for this scheme is quite simple. For the ROM structures storing s_1 and s_2, a single 9-bit counter is needed. Since the inner products for s_1 and s_2 are operated in parallel, a single counter is enough for both operations.

For the RNG a 3-bit counter is needed to track the XOR-ing of each 8 consecutive bits. This can be interleaved with the inner product operation $s_2 \cdot b$. The architecture has only 2 modes of operation. Therefore, a single flip-flop would suffice to track the state of the hardware. The total area of the control block is estimated at about 150 gates.

The total area of the authentication hardware is 970 gates. This is below 1K gates, a typical threshold for the RFID's footprint allotted for security [82].

6.5 Summary

In this chapter we proposed a scheme which seems resilient against certain man-in-the-middle attacks. We also demonstrated an efficient method for generating the random bits needed for our proposed protocol. This was done without incurring significant overhead to the hardware, and by reutilizing parts of the authentication circuit. As can be seen the HB+PUF proposal is very similar to the PUF-HB of the previous chapter. However, the proposal of this chapter targets takes a more practical approach and focuses on the implementation. One of the minor yet important contributions here is the introduction of an enhanced version of PUFs which incorporates a field multiplication (see Equation 6.1). Although this enhancement does not prevent modeling PUFs, it does have the potential to improve its behavior under different inputs.

Chapter 7

Two Level PUFs

In the previous two chapters we saw how to improve the security of a simple PUF by combining it with HB. In this chapter we show that PUFs alone can be used to create a secure challenge response authentication scheme. In specific, we present a new primitive which is the 2-level PUF. Learning the proposed scheme in the passive attacker model is reduced to learning a special class of a threshold of majority gates under the uniform distribution. Furthermore, we explore extensions of PUFs to produce efficient n-to-n mappings. So far, PUFs have only been used as a boolean function. It should be interesting to see different ways of creating n-to-n mappings.

In Section 7.1, we define new PUF-based n-to-n mappings which we use in our proposed scheme. In Section 7.2 we define the proposed protocol and discuss how it can be modeled and deployed. In Section 7.3 we present the security analysis of the proposed protocol. We finally conclude with a summary in Section 7.4.

7.1 Beyond a Single PUF

In this section we will introduce some variations of the simple PUF scheme. In Chapter 4, we pointed out that attempting to model a PUF circuit will always have some

inaccuracy which we refer to as *noise*. In this section we explore methods for constructing a $\{0,1\}^n \to \{0,1\}^n$ mapping using a PUF circuit. The most straightforward way to produce such a mapping is to concatenate n different PUF circuits. We refer to this construction as a *PUF-box*. Note that these definitions will be variants of Definition 4.2.1.

Definition 7.1.1. *A noisy PUF-box is a function characterized by the matrix* $M \in \mathbb{R}^{n+1\times n}$ *and the noise parameter* $\epsilon \in \left(0, \frac{1}{2}\right)$*, such that* M*'s* (i,j) *entry* $m_{i,j} \in N(0,1)$*.* $PUFb_{M,\epsilon}(C) : \{0,1\}^n \to \{0,1\}^n$*,* $Z = \mathrm{PUFb}_{M,\epsilon}(C)$ *where*

$$z_j = \operatorname{sign}\left(\sum_{i=1}^{n}(-1)^{p_i} m_{i,j} + m_{n+1,j}\right) + \nu \quad . \tag{7.1}$$

The vector Z *is defined as* $Z = [z_i \dots z_n]$*, and the variables* ν *and* p_i *are defined as in Definition 4.2.1.*

The notion of a PUF-box has not been previously defined in this manner. However, there has been schemes proposed where the PUF is assumed to be an n-to-n mapping without an explanation of how this would be achieved [12]. In reality, a PUF-box is inefficient in terms of gate count. It would require $O(n^2)$ gates, which can become large even for a low security parameter. Our definition of a PUF-box is mainly used as a transitional step in order to define a more realistic n-to-n PUF-based mapping.

We next introduce a *Simulated PUF-Box (SPB)*. We first explain the intuition behind such a construction. A PUF-box is characterized by $n(n+1)$ variables. Each of these variables is only used in one of the PUFs inside the PUF-box. This large number of variables is the source of inefficiency. To solve this problem in the SPB, we only use two PUF circuits. In particular, we use a random n-bit string S to choose a path which utilizes different parts of the two PUF circuits. In the i^{th} stage of the SPB, the signal either goes through the first or the second PUF depending on the corresponding bit s_i. If we label the delay mismatch in the i^{th} stage of the first and second PUF as $y_{i,0}$ and $y_{i,1}$ respectively, then s_i will basically be choosing to use

$y_{i,0}$ or $y_{i,1}$ in the delay equation. This means that with only $2(n+1)$ variables, we can generate 2^n different PUFs, each corresponding to a different S string. A large number of these PUFs will have a similar behavior. However, if the S's are chosen to have a large pairwise hamming distance, the resulting PUFs will have a sufficiently different output behavior.

A single S string will result in having a single PUF. Therefore, we need n different S strings in order to simulate a PUF-box. The storage of n different S strings would be as costly as the PUF-box. To solve this problem, we introduce a simple LFSR [6]. The LFSR will behave as a schedule for the n different S strings. To run the SPB, the LFSR is initialized to a randomly pre-chosen state S^0. The challenge is then fed to the switches of the two internal PUF circuits. As indicated earlier, the string S^0 will induce a PUF with delay variables chosen out of the two existing PUF circuits. When a pulse is sent down the switch chain, a single output bit will be generated. Next, the LFSR runs for n cycles, therefore generating the new state S^1. This new state will induce a second PUF circuit with different delay variables, and will therefore produce the second output bit. Continuing in this fashion for n iterations, we obtain an n-bit output for the n-bit challenge input. At the end of the n^{th} iteration the LFSR is reset to the original state S^0. Note that while this construction saves on the number of gates used, it will require $O(n^2)$ clock cycles before generating an n-bit output. Our interest in this chapter is directed towards a smaller size circuit. Therefore, the time/size trade off presented by the SPB will be ideal. We now formally define a noisy SPB.

Definition 7.1.2. *A noisy SPB is a function characterized by the vectors* $Y^0, Y^1 \in \mathbb{R}^{n+1}$*, the noise parameter* $\epsilon \in \left(0, \frac{1}{2}\right)$*, the initial bit string* $S^0 \in \{0,1\}^n$ *and an LFSR function* L*. Such that* $Y^j = [y_{1,j}, \ldots, y_{n+1,j}]$ *and* $y_{i,j} \in N(0,1)$*.* $SPB_{Y^0,Y^1,S^0,L,\epsilon}(C)$:

$\{0,1\}^n \rightarrow \{0,1\}^n$ *such that* $Z = \mathrm{SPB}_{Y^0,Y^1,S^0,L,\epsilon}(C)$ *where*

$$z_j = \mathrm{sign}\left(\sum_{i=1}^{n}(-1)^{p_i} y_{i,s_i^j} + y_{n+1,s_n^j}\right) + \nu \quad . \tag{7.2}$$

The vector Z *is defined as* $Z = [z_1 \dots z_n]$, $S^j = [s_1^j \dots s_{n+1}^j]$ *and* $S^j = L^n(S^{j-1})$ *such that* L^n *denotes* n*-iterations of the LFSR function. The variables* ν *and* p_i *are defined as in Definition 4.2.1.*

We utilize these definitions in the following sections in order to derive a new authentication protocol.

7.2 A 2-Level Noisy PUF Authentication Scheme

Standalone PUF circuits may not provide adequate protection against active attacks. Suppose a PUF-based authentication protocol is used as the security measure for an RFID tag. An attacker can easily send challenges to the tag and receive the response from it, without being detected. Utilizing only a small number of challenge-response pairs, the attacker can develop a linear model of the PUF circuit with the technique outlined in Chapter 4. Adding some level of noise to the output of the PUF circuit by flipping a fraction of the response bits before sending it back to the reader will make it harder for an attacker to obtain the linear model. However, the system will still be modelable.

In [28], the authors note that PUFs may be vulnerable to modeling and differential attacks in the original configuration. Therefore, the authors propose to apply a cryptographic hash function to the output of the PUF circuit to eliminate such attacks. However, the introduction of the cryptographic hash function adds significantly to the footprint and power consumption. A hash function is a challenging cryptographic primitive to serialize. Therefore, reducing the gate count might not be feasible. In [27], feed forward arbiters were proposed to introduce non-linearity into

the PUF scheme. This approach seems promising. However, since the feed forward PUF are not known to be modelable, one would need to collect a large number of challenge-response pairs for every PUF circuit and store them in a database. This is the classical approach proposed in the early literature on PUF. This solution becomes less practical when the number of devices deployed becomes massive. In this section we attempt to present a scheme which addresses these three concerns: security, circuit size and scalability.

As pointed out earlier, a PUF device is inherently noisy due to the small variations in the circuit's physical properties. In the original PUF scheme, this noise is eliminated by implementing majority voting. Even with such a scheme, there will be an unavoidable level of noise in the system. This noise will not make the PUF more secure. In [8], the authors introduce a polynomial time algorithm for solving noisy linear threshold functions similar to the one implemented in a PUF. In addition, there has been a considerable amount of work in the area of learning threshold functions. These results show that most variations of the linear threshold functions are learnable in polynomial time. However, there are a number of elegant results showing that a 2-level threshold function would be hard to learn under certain assumptions. We will thoroughly discuss these results in Section 7.3. In this section it suffices to point out that these results are the main motivation for the following scheme.

We propose using a noisy SPB with the output connected to a noisy PUF. In short we will refer to this noisy PUF as the output PUF. Note that the overall system will be a mapping $\{0,1\}^n \rightarrow \{0,1\}$. For security purposes we control the input of the system. In particular, the input of the SPB is generated by a True Random Number Generator (TRNG). This generator will produce a uniformly random n-bit string B which will be fed to the SPB. Recall from Definition 1 that the PUF first transforms its input C into the vector P where $P = UC$. This property of the PUF can cause a large error propagation. Therefore, we choose to implement U^{-1} before sending the

output of the SPB T into the output PUF. Fortunately, U^{-1} has a very simple form, and can be implemented using $n-1$ XOR gates. In particular, let $T' = U^{-1}T$, then $t'_i = t_i \oplus t_{i+1}$, and $t'_n = t_n$[1]. Finally T' is XOR-ed with the incoming challenge C to generate the input of the output PUF which produces the response bit R. The proposed scheme is shown in Figure 7.1. We now formally define the scheme.

Definition 7.2.1. *A 2-level authentication function f is a function characterized by the vectors $Y^0, Y^1, Y^{out} \in \mathbb{R}^{n+1}$, the noise parameter $\epsilon \in \left(0, \frac{1}{2}\right)$, the initial bit string $S^0 \in \{0,1\}^n$ and the LFSR function L. Such that $y_{i,0}, y_{i,1}, y_i^{out} \in N(0,1)$. Where $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}(B,C) : \{0,1\}^{2n} \to \{0,1\}$ such that*

$$f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}(B,C) = \mathrm{PUF}_{Y^{out},\epsilon}\left(U^{-1}\left(\mathrm{SPB}_{Y^0,Y^1,S^0,L,\epsilon}(B)\right) \oplus C\right) \quad . \tag{7.3}$$

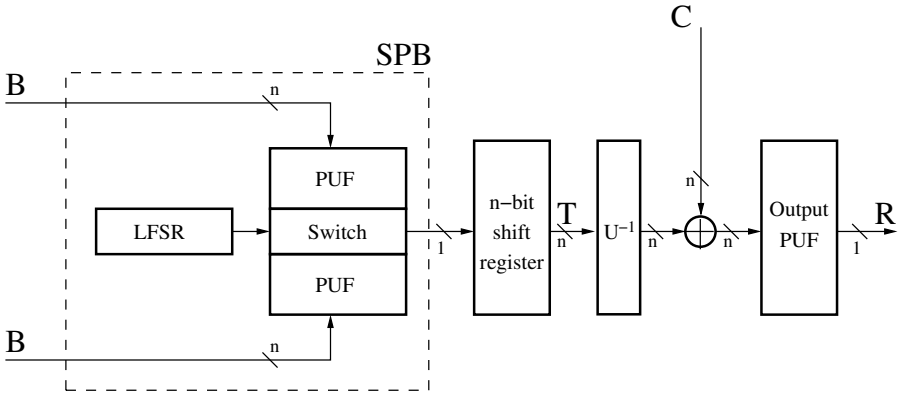


Figure 7.1: A 2-level Authentication Circuit

To assess the proposed scheme, we note that the size of the implementation will be significantly smaller than that of a cryptographic hash function. It is not hard to see the simplicity of the components involved in such an implementation. We leave the security assessment of this scheme to Section 7.3. This leaves us with the final criterion in our assessment: the ability to build an accurate model of the system.

[1]This issue can be avoided by using a tristate buffer implementation of a PUF given in Chapter 4.

This is a very sensitive aspect, since we would like the owner of the device to be able to model the system while at the same time preventing an attacker from achieving the same goal.

As we will see in Section 7.3, it will be hard to model the presented scheme. However, we provide an alternative means which would allow the owner of the PUF to initially model the system before completely eliminating such ability. In particular, we require a set and a reset control on the flip-flops inside the LFSR of the SPB. Using these inputs we can set the LFSR to start in one of three possible states: the all zeros state 0^n, the all ones state 1^n and the initial state S^0. Note that the 0^n state is a fixed point for any LFSR function, meaning that the state will not change regardless of the LFSR operation. Now, if we chose the LFSR function L so that the 1^n state is also a fixed point for L, we will be able to model the circuit. Making the 1^n state a fixed point of L is very simple. All we have to do is to require the L function to have an odd number of terms in its connection polynomial. To model the system we start by setting the state of the LFSR to the 0^n state. This would reduce the SPB into a circuit implementing the same PUF function $\mathrm{PUF}_{Y^0,\epsilon}(B)$ for all n rounds. Recall that B is a random input fed into the SPB. Now, since the SPB is implementing the same PUF function for the n rounds, the n output bits of the SPB will be the same except for about ϵn bits affected by the noise. This output T will be XOR-ed with the challenge C. However, since the challenge is externally fed to the circuit, we can choose $C = 0^n$ for all the modeling rounds. This means that the challenge input to the output PUF will be T.

Although we do not know the specific parameter Y^{out} which describes the output PUF, we do know that its input T will either contain $(1-\epsilon)n$ zeros if the output of $\mathrm{PUF}_{Y^0,\epsilon}(B) = 0$, or $(1-\epsilon)n$ ones if the output of $\mathrm{PUF}_{Y^0,\epsilon}(B) = 1$. The two types of possible inputs will have a Hamming distance larger than $(1-2\epsilon)n$ and will therefore with a very high probability produce an opposite output on the output

PUF.[2] This observation will allow us to collect a number of noisy challenge-response pairs for $\text{PUF}_{Y^0,\epsilon}$. Using the algorithm presented in [8], we can easily solve for Y^0. Repeating the same process, only this time using the 1^n state for the LFSR, we can similarly solve for Y^1. The LFSR function L and the state S^0 were both pre-chosen. Now, with the knowledge of Y^0 and Y^1 we will have a model for the SPB. Next, we set the state of the LFSR back to S^0, and then permanently disable the set/reset logic for the LFSR. Finally, since the output of the SPB is modelable, we can collect challenge-response pairs for the output PUF, therefore solving for Y^{out}. With this last step we will have completely modeled the 2-level authentication scheme.

Fortunately, an attacker will not be able to carry out the same process. When we disable the set/reset logic we are essentially logically isolating the system. Any changes that an attacker wishes to introduce to the system will have to physically target the implementation. With three PUF circuits inside the implementation, we can be assured that any tampering will drastically change the response of the authentication circuit. Even the register storing the state S^i will be located between the two PUF circuits. Any attempts to affect the register will also affect the internal delays of the PUF circuits. Hence, we can assume the register to be tamper-proof. The properties of the PUF circuit renders the authentication circuit as a black-box to the attacker. The only information available about the system will be the triple $(B, C, f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}(B, C))$.

Any 2-level authentication circuit we wish to enroll in the network will have to go through the modeling process. Once the device is enrolled and deployed in the field, the 2-level PUF authentication protocol given in Table 7.1 is used to authenticate the device. In the protocol, P represents the PUF enabled tag which is characterized by $(Y^0, Y^1, Y^{out}, S^0, L, \epsilon)$ as in Definition 7.2.1, D represents the authentication server such as a card reader, and finally ϵ_f denotes the allowable error in the authentication

[2] See Proposition 4.2.1 for a rigorous statement of this fact.

Protocol 3: 2-level noisy PUF Authentication
1. P randomly generates the n-bit string B, and sends it to D.
2. D picks a random n-bit string C and sends it to P.
3. P receives the challenge C, and then calculates $R = f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}(B, C)$.
4. P sends the bit R to D.
5. Steps 1 through 4 are repeated for k iterations.
6. D accepts P iff the number of errors in the responses R is not more than $\epsilon_f k$.

Table 7.1: 2-level-PUF authentication

protocol[3].

7.3 Security Analysis

In the previous sections we pointed out that a PUF-based circuit is highly sensitive. This property makes it almost impossible to access the internals of the circuit without causing a change to the parameters. However, we saw in Chapter 4 that collecting a number of challenge-response pairs can render a PUF circuit modelable (or learnable), even in the presence of noise [8]. Our goal in this section is to show that under certain assumptions the authentication scheme proposed in this chapter can be secure. In order to do so, we review a number of results on the learnability of threshold functions.

We start by defining a number of terms. A *Linear Threshold Function (LTF)* or a *halfspace* is essentially a boolean function $f(x) = \text{sign}(\sum_{i=1}^{n}(-1)^{x_i}y_i + y_{n+1})$. Note that this definition is equivalent to Equation (4.3) without the noise parameter ν. Therefore, we can say that a noiseless PUF is essentially an LTF. Just like a

[3]This value will depend on ϵ and will be derived in Lemma 7.3.4 of Section 7.3.

PUF the LTF is characterized by a real vector $Y \in \mathbb{R}^{n+1}$, where the threshold is the y_{n+1} variable. For our purposes we encoded the output of an LTF into a binary $\{0,1\}$ rather than $\{-1,1\}$ as done in the literature. We will use the term LTF and halfspace interchangeably to mean the same thing. We say that f is an n-dimensional halfspace if it takes inputs from $\{0,1\}^n$. Learning such a function is one of the oldest solved problems in machine learning [2]. As we pointed out earlier, it is possible to learn halfspaces even in the presence of simple classification noise [8]. A *light* halfspace is a halfspace in which the sum of the magnitudes of the weights y_i is bounded by a polynomial in n. Note that since PUFs are halfspaces which arise from natural phenomena they will be light halfspaces. A *majority* function or gate is an LTF with threshold equal to 0 and with all the y_i values equal to 1. The output of a majority gate will be equal to one iff the majority of the input bits are one. Otherwise, the output will be zero. When the input is an n-bit string the majority gate can sometimes act on a subset of these n bits. Therefore, the majority gates can be described by the vector $X \in \{0,1\}^n$, where $x_i = 1$ means that the i^{th} input bit is acted on by the majority gate. We call the vector X a *descriptor* for the majority function. So for example, if $n = 5$ and the majority gate descriptor is $X = 01101$, then the majority gate will only compute the majority of bits $2, 3$ and 5. The *intersection* of halfspaces is the logical AND of the outputs of a number of LTFs. We say that a boolean function $f : \{0,1\}^n \to \{0,1\}$ is *PAC-learnable* (Probably and Approximately Correct) [97] if there is an algorithm A that when given $\epsilon \in (0,1)$ and $\text{poly}(n, \frac{1}{\delta})$ samples $(x, f(x))$ where $x \in \{0,1\}^n$ and is drawn according to a probability distribution $\mathcal{D}$, A produces the function h, such that $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] < \delta$ where A does not know f. If $\mathcal{D}$ is the uniform distribution, we say that the algorithm A PAC-learns f under the uniform distribution. We say that an algorithm can learn in the presence of ϵ classification noise if it can learn from samples $(x, f(x) + \nu)$ where $\nu = 1$ with probability ϵ and zero otherwise. One last term to introduce is *proper*

learning, in which the algorithm A is required to output a function h that has the same form as the unknown function f. When this condition is not posed, we refer to the learning process as *non-proper* learning. It should be clear that proper learning is a harder constraint on the learning algorithms. Therefore, hardness results for non-proper learning are in general considered as stronger results.

Our main focus is on 2-level LTF circuits. This problem has been well studied in the literature. In [10] it is shown that properly learning a two level threshold function with two input nodes that are LTFs and one output node which is also an LTF is NP-complete. In [56] the authors show that in general non-proper learning of a d-level threshold function is as hard as breaking the RSA algorithm, detecting quadratic residues and factoring Blum integers. These results are with respect to an unknown distribution. However, in [58], the author strengthens the results to hold under the uniform distribution. In [59], the authors show an exponential time algorithm to learn a two level threshold circuit under the uniform distribution. When the second level is any n-to-1 boolean function, the algorithm runs in complexity exponential in k (the number of nodes of the first level). For special cases when the input nodes take as inputs disjoint sets of all the n inputs, and the second level functions are majority or AND functions, the proposed algorithm runs in exponential time in $\log(k)$. In [25] the authors show that under the assumption that the Ajtai-Dwork cryptosystem [37] is secure, then there is no weak non-proper PAC-learning algorithm for polynomial-sized majority gate circuits of depth 2. A similar result was obtained in [60] in which the authors show that non-proper learning of a polynomial-sized 2-level majority gate circuit is at least as hard as the Shortest Vector Problem (SVP) and the Shortest Independent Vector Problem (SIVP) for an approximation factor of $\widetilde{O}(n^{1.5})$. Note that these two problems are believed to be hard for this particular approximation factor [85]. The results obtained apply for a special distribution that is imposed by the reduction. The authors also show that non-proper learning of the intersection

of n^ϵ light n-dimensional halfspaces for $\epsilon > 0$ is at least as hard as the $\widetilde{O}(n^{1.5})$-SVP and $\widetilde{O}(n^{1.5})$-SIVP problems. Finally in [61] the authors show that in the statistical query model [55] any algorithm for non-proper learning of the intersection of $\sqrt{n}$ halfspaces which are n-dimensional will require at least $2^{\sqrt{n}}$ queries. To understand the importance of the statistical query model, note that in [55] the author shows that any class of functions efficiently learnable from statistical queries is also efficiently learnable with classification noise.

Motivated by these results we make the following initial assumption.

Assumption 7.3.1. *There does not exist a randomized polynomial-time algorithm which can PAC-learn any threshold of n n-dimensional halfspaces under the uniform distribution in the presence of ϵ classification noise, where $\epsilon \in (0, \frac{1}{2})$.*

When we say threshold of halfspaces, we basically mean a threshold of thresholds. Note that the majority and the AND functions are both weaker than a threshold function [35]. Assumption 7.3.1 has only been proven under strong cryptographic assumptions for certain distributions. Therefore, the main aspect to be proven about the assumption becomes learning under uniform distribution. In [61] the authors report progress on this particular problem. Also note that most of these results do not take the noise into account. The setup in the assumption is in fact identical to the 2-Level authentication function of Definition 4, except that we need to use a PUF-box rather than an SPB. In particular, we can say that a 2-Level authentication scheme using a PUF-box for its first level is a threshold of n n-dimensional halfspaces. The only difference is that the assumption does not specify the distribution on the thresholds. It seems that when the thresholds are Gaussian, learning under the uniform distribution will look like learning under the Gaussian distribution. This would mean that any bounds used in the Gaussian case can be applied to the PUF case. Note that the problem will still be hard in this case [57].

Recall that we are interested in lightweight solutions. Therefore, we will use a

stronger assumption. We start with the following definition.

Definition 7.3.1. *Let $\mathcal{G}_1, \ldots, \mathcal{G}_n$ be n different majority gates, such that $X_i \in \{0,1\}^n$ is the descriptor for $\mathcal{G}_i$. Let $\mathcal{X} \in \{0,1\}^{n^2}$ where $\mathcal{X} = [X_1 \ldots X_n]$ be the combined descriptor of the n majority gates. We say that the n majority gates described by $\mathcal{X}$ have a Linear Complexity[4] description of n, if there exists an LFSR L of length n which generates $\mathcal{X}$.*

The definition captures the complexity of describing n-majority gates. It is clear that such a description would require n^2 bits as seen in the definition. However, we are interested in the cases when the description can be compressed such that an LFSR can generate the entire sequence. Now we state the following assumption.

Assumption 7.3.2. *There does not exist a randomized polynomial-time algorithm which can PAC-learn any threshold of n-majority functions having a linear complexity description of n, where the algorithm learns under the uniform distribution in the presence of ϵ classification noise, with $\epsilon \in (0, \frac{1}{2})$.*

It should be clear that the added linear complexity condition on the majority gates is added to serve the SPB. This condition can be relaxed if we allow the LFSR to be of length $\frac{n^2}{2}$ [87]. This assumption is stronger than Assumption 7.3.1. Recall from the previous results [59] that when a number of majority gates act on disjoint inputs, there exist quasi polynomial-time algorithms to learn the AND and the majority functions of these majorities. Of course the restriction we add in Assumption 7.3.2 is much weaker than the restriction required to obtain these quasi polynomial-time algorithms. Also we need to keep in mind that a 2-level polynomial-size majority gate circuit is in general hard to learn. Another aspect of Assumption 7.3.2 is that it takes classification noise into consideration. In general, our assumption is based on

[4]The *Linear Complexity (LC)* of a bit sequence X, is the length of the shortest LFSR which generates X [75].

the general hardness of learning 2-level noisy majority and LTF functions. Next, we prove a reduction from our scheme to the problem in Assumption 7.3.2.

Theorem 7.3.3. *Say there exists a polynomial-time randomized algorithm A that can PAC-learn any 2-level authentication function $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$ as in Definition 4, with $\epsilon \in (0, \frac{1}{8})$, where A is a passive[5] algorithm which learns under the uniform distribution in the presence of ϵ_f classification noise. Then there exists a randomized polynomial-time algorithm A' which can PAC-learn any threshold of n-majority functions having a linear complexity description of n and denoted by $M_{Y^M,\mathcal{X},\epsilon_f} : \{0,1\}^n \to \{0,1\}$, such that $Y^M \in \mathcal{R}^{n+1}$ characterizes the threshold function of M, and $\mathcal{X} = [X_1 \dots X_n]$ where $X_i \in \{0,1\}^n$ is the descriptor for the i^{th} majority function in M, where A' learns under the uniform distribution in the presence of ϵ_f classification noise.*

Proof. We describe how to use A to produce the algorithm A'. When the algorithm A tries to learn the function $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$ it expects to receive a polynomial number of triplets $(B_i, C_i, R_i = f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}(B_i, C_i))$ where B_i is chosen uniformly at random. Let $(H_i, M_{Y^M,\mathcal{X}}(H_i))$ denote the samples presented to the algorithm A' in order to learn $M_{Y^M,\mathcal{X},\epsilon_f}$. Now, A' starts running A by simulating the triplet A needs to receive. The way A' does this is by setting all the challenges $C_i = 0^n$, it then feeds the algorithm A with $(H_i, 0^n, R_i = M_{Y^M,\mathcal{X}}(H_i))$. Note that H_i will be uniformly random since we are trying to learn under the uniform distribution. Now A' simply takes the solution function returned by A and uses it as the solution function for $M_{Y^M,\mathcal{X},\epsilon_f}$.

To show that this will actually work, we demonstrate that $M_{Y^M,\mathcal{X},\epsilon_f}$ is in fact an instance of $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$. Set $Y^{out} = Y^M$, $Y^0 = 0^{n+1}$ and $Y^1 = [1^n\, 0]$ where we made $y^1_{n+1} = 0$. Using the Berlekamp-Massey algorithm [75] generate the n-bit LFSR function L' and the n-bit initial state S' which produces the bit string

[5] *Passive* means that A only sees transcripts of the authentication session, rather than being able to interact with the protocol, in which case it would be able to choose C.

$\mathcal{X} = [X_1 \dots X_n]$. Such an LFSR and an initial state exist from the assumption that $\mathcal{X}$ has a linear complexity of n. Now set $L = L'$ and $S^0 = S'$. After every n iterations the LFSR will output X_i, which will in turn control the switches inside the SPB. Note that because of the way Y^0 and Y^1 are setup, in every iteration the SPB will be implementing the majority gate described by X_i. Therefore the output fed to the output PUF will be that of the n-majority gates described by $\mathcal{X}$. The output PUF will be identical to the threshold function characterized by Y^M. Finally, the noise of the entire system ϵ_f will depend on the noise parameter ϵ. As will be shown in Lemma 7.3.4 for $\epsilon < \frac{1}{8}$ we have $\epsilon_f \in (0, \frac{1}{2})$. Therefore $M_{Y^M,\mathcal{X},\epsilon_f} = f_{0^{n+1},[1^n\,0],Y^M,S',L',\epsilon}$. □

It should be clear that the above is a worst case reduction. The last thing we need to do in order to complete this section, is to show how the error ϵ in each of the 3 PUFs inside the 2-level authentication function $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$ will propagate to the final output R. To quantify the error propagation in the 2-level authentication function we assume the error that is produced by the SPB to be independent between the different rounds. This a reasonable assumption since the error depends on environmental variations that are independent of the round number. Similarly, we assume the error produced by the SPB to be independent of the error produced by the output PUF. With these assumptions we can now prove the following lemma.

Lemma 7.3.4. *The expected overall classification noise in the 2-level authentication function $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$ of Definition 4 will be $\epsilon_f < \frac{4}{\pi} \arctan\left(\sqrt{\frac{\epsilon}{1-\epsilon}}\right)$.*

Proof. Let the output of the SPB inside $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$ be $T \in \{0,1\}^n$. Because we assumed the error occurs independently, the number of errors which are expected to be found in T will belong to a Binomial distribution $B(n,\epsilon)$. Recall that in Definition 4 we use the transformation U^{-1} before injecting the output of the SPB to the output PUF. Therefore, we will have $P = T$.[6] Now we can use Proposition 4.2.1, with d as

[6] See the definition of P in Proposition 4.2.1.

a random variable following $B(n, \epsilon)$. Assume for now that the output PUF has 0% noise. In such a case the expected noise in the output of the entire circuit will be

$$\begin{aligned}\epsilon_s &= 1 - \sum_{i=0}^{n} \binom{n}{i} (\epsilon)^i (1-\epsilon)^{n-i} \left(1 - \frac{2}{\pi} \arctan\left(\sqrt{\frac{i}{n+1-i}}\right)\right) \\ &\leq 1 - F_n(\epsilon n) \\ &\leq 1 - F_{n-1}(\epsilon n) = \frac{2}{\pi} \arctan\left(\sqrt{\frac{\epsilon}{1-\epsilon}}\right)\end{aligned}$$

Now we take into consideration the noise ϵ which will occur in the output PUF. An overall error will appear on the output of $f_{Y^0,Y^1,Y^{out},S^0,L,\epsilon}$ iff the error caused by the output PUF and the error caused by the noise in the input to the output PUF do not coincide. As we assumed earlier the error in T and the error caused by the output PUF are two independent events. Therefore we have

$$\epsilon_f = \epsilon(1-\epsilon_s) + \epsilon_s(1-\epsilon) < 2\,\epsilon_s = \frac{4}{\pi} \arctan\left(\sqrt{\frac{\epsilon}{1-\epsilon}}\right)$$

□

This lemma explains why in Theorem 7.3.3 we needed the error $\epsilon \in (0, \frac{1}{8})$ in order to have $\epsilon_f < \frac{1}{2}$. Note that $\epsilon_f = \frac{1}{2}$ is the information theoretic limit on the amount of allowable noise. In general, the choice of a noise parameter will control the number of rounds k used in an authentication session. A reasonable parameter would be for ϵ_f to be around 0.25. To get a sense of the error propagation in general, for the typical noise values achieved in a PUF circuit we have $\epsilon = 3\%$ which would yield an upper limit of $\epsilon_f < 23\%$ on the 2-level authentication function noise.

7.4 Summary

We presented a tamper-proof and lightweight challenge-response authentication scheme based on 2-level noisy PUFs. The proposed authentication scheme may be implemented in a very small footprint and deployed in applications with stringent power

limitations. Furthermore, the inherent properties of PUFs provide strong tamper-resilience – a feature crucial for applications where the device is in the hand of potential attackers. From a computational security point to view the 2-level PUF scheme is weaker than PUF-HB and HB+PUF. The real advantage of the 2-level scheme is that all the secrets are PVs which makes the possibility of detecting any tampering more realistic.

Chapter 8

Summary

The work in this dissertation mainly revolved around exploring applications of PVs. Following is a summary of our contributions.

- The limits of using Gaussian PVs as identifying PVs are explored. As the most natural form of a PV, Gaussian PVs provide a convenient vehicle to rigorously prove limits on their identification capability.

- A new CD fingerprinting technique which utilizes the manufacturing variability in the length of the CDs' lands and pits is presented. The novelty of the approach is in the use of the electrical signal produced by the photo-detector inside the CD reader.

- A mathematical framework for delay-based PUFs is re-derived. This framework has been previously presented in a slightly different format [27]. The work here simplifies these derivations and quantifies the probability of observing a collision in the output of the delay-based PUF.

- A new implementation of the delay-based PUF, i.e. the tristate PUF, is proposed. The new PUF implementation is shown to be equivalent to the MUX-based delay PUF by deriving the corresponding mathematical model.

- Two authentication protocols, PUF-HB and HB+PUF, are presented. These protocols build on the HB authentication family and combine them with PUFs in order to improve on the security of the typical HB protocol.
- A proof of concept implementation for HB+PUF is presented and shown to require a few thousand gates.
- An authentication protocol based on 2-level PUFs is presented.

Bibliography

[1] Abe, S., Hashimoto, M., Onoye, T., Clock skew evaluation considering manufacturing variability in mesh-style clock distribution. In: Proceedings of the 9th International Symposium on Quality Electronic Design (ISQED 2008), 17-19 March 2008, San Jose, CA, USA, pp. 520-525, IEEE Computer Society, Los Alamitos, CA, USA (2008).

[2] Agmon, S., The relaxation method for linear inequalities. Canadian Journal of Mathematics, 6(3), pp. 382-392, (1954).

[3] Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., Sunar, B., Trojan detection using IC fingerprinting. In: Proceedings of the IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA, pp. 296-310, IEEE Computer Society, Los Alamitos, CA, USA (2007).

[4] Andersen, E.D., Andersen, K.D., Presolving in linear programming. Mathematical Programming, 71(2), pp. 221-245, (1995).

[5] Bauder, D.W., An anti-counterfeiting concept for currency Systems. Research Report PTK-11990, Sandia National Labs, Albuquerque, NM, USA (1983).

[6] Berlekamp, E.R., Algebraic Coding Theory. McGraw-Hill, New York, USA, 1968.

[7] Berlekamp, E.R., Mceliece, R.J., van Tilborg, H.C., On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory, 24(3), pp. 384-386, (1978).

[8] Blum, A., Furst, M., Jackson, J., Kearns, M., Mansour, Y., Rudich, S., A polynomial-time algorithm for learning noisy linear threshold functions. Algorithmica, 22(1/2), pp. 35-52, (1998).

[9] Blum, A., Kalai, A., Wasserman, H., Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM, 50(4), pp. 506-519, (2003).

[10] Blum, A., Rivest, R.L., Training a 3-node neural network is NP-complete. In: Hanson, S.J., Remmele, W., Rivest, R.L., (eds.) Proceedings of the First Annual Workshop on Computational Learning Theory (COLT 1988), pp. 9-18, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988).

[11] Bogdanov, A., Leander, G., Knudsen, L.R., Paar, P., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C., PRESENT - an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I., (eds.) Proceedings of the 9th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007), LNCS, vol. 4727, pp. 450-466, Springer-Verlag, Heidelberg, Germany (2007).

[12] Bolotnyy, L., Robins, G., Physically unclonable function-based security and privacy in RFID systems. In: Proceedings of the International Conference on Pervasive Computing and Communications (PerCom 2007), New York, USA, 19-23 March 2007, pp. 211-220, IEEE Computer Society, Los Alamitos, CA, USA (2007).

[13] Bringer, J., Chabanne, H., Dottax, E., HB^{++}: a lightweight authentication protocol secure against some attacks. In: Proceedings of the Second International

Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2006), June 2006, Lyon, France, pp. 28-33, IEEE Computer Society, Los Alamitos, CA, USA (2006).

[14] Carter, L., Wegman, M., Universal hash functions. Journal of Computer and System Sciences, 18(2), pp. 143-154, (1979).

[15] Clarkson, W., Weyrich, T., Finkelstein, A., Heninger, N., Halderman, J.A., Felten, E.W., Fingerprinting blank paper using commodity scanners. In: Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P 2009), Oakland, CA, May 2009, IEEE Computer Society, Los Alamitos, CA, USA, 2009 (to appear).

[16] Cover, T.M., Thomas, J.A., Elements of Information Theory. Wiley-Interscience, New York, NY, USA, 1991.

[17] Cowburn, R.P., Buchanan, J.D.R., Verification of authenticity. US Patent Application 2007/0028093, July 27th (2006).

[18] Dantzig, G.B., Orden, A., Wolfe, P., The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific Journal of Mathematics, 5(2), pp. 183-195, (1955).

[19] DeJean, G., Kirovski, D., RF-DNA: radio-frequency certificates of authenticity. In: Paillier, P., Verbauwhede, I., (eds.) Proceedings of the 9th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007), LNCS, vol. 4727, pp. 346-363, Springer-Verlag, Heidelberg, Germany (2007).

[20] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A., Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing, 38(1), pp. 97-139, (2008).

[21] Duc, D.N., Kim, K., Securing HB^+ against GRS man-in-the-middle attack. In: Proceedings of the 2007 Symposium on Cryptography and Information Security (SCIS 2007), Sasebo, Japan, 23-26 January 2007, pp. 23-26, IEICE, Tokyo, JAPAN (2007).

[22] European Computer Manufacturers' Association. Standard ECMA-130: Data interchange on read-only 120mm optical data disks (CD-ROM) (2nd ed.). ECMA, Geneva, Switzerland, 1996.

[23] Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L., A survey of lightweight cryptography implementations. IEEE Design & Test of Computers – Special Issue on Secure ICs for Secure Embedded Computing, 24(6), pp. 522-533, (2007).

[24] Feldhofer, M., Dominikus, S., Wolkerstorfer, J., Strong authentication for RFID systems using the AES algorithm. In: Joye, M., Quisquater, J.J., (eds.) Proceedings of the 6th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), LNCS, vol. 3156, pp. 357-370, Springer-Verlag, Heidelberg, Germany (2004).

[25] Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K., New results for learning noisy parities and halfspaces. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, pp. 563-574, IEEE Computer Society, Los Alamitos, CA, USA (2006).

[26] Fossorier, M.P.C., Mihaljevic, M.J., Imai, H., Cui, Y., Matsuura, K., An algorithm for solving the LPN problem and its application to security evaluation of the HB protocols for RFID authentication. In: Barua, R., Lange, T., (eds.) Proceedings of the 7th International Conference on Cryptology in India, Progress in

Cryptology (INDOCRYPT 2006), LNCS, vol. 4329, pp. 48-62, Springer-Verlag, Heidelberg, Germany (2006).

[27] Gassend, B., Lim., D., Clarke, D.E., van Dijk, M., Devadas, S., Identification and authentication of integrated circuits. Concurrency - Practice and Experience, 16(11), pp. 1077-1098, (2004).

[28] Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S., Silicon physical random functions. In: Atluri, V., (ed.) Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), pp. 148-160, ACM, New York, NY, USA (2002).

[29] Gassend, B.L.P., Physical Random Functions. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.

[30] Gaubatz, G., Savas, E., Sunar, B., Sequential circuit design for embedded cryptographic applications resilient to adversarial faults. IEEE Transactions on Computers, 57(1), pp. 126-138, (2008).

[31] Gaubatz, G., Sunar, B., Robust finite field arithmetic for fault-tolerant public-key cryptography. In: Breveglieri, L., Koren, D., Naccache, D., Seifert, J.P., (eds.) Proceedings of the 3rd International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2006), LNCS, vol. 4236, pp. 196-210, Springer-Verlag, Heidelberg, Germany (2006).

[32] Gaubatz, G., Sunar, B., Karpovsky, M., Non-linear residue codes for robust public-key arithmetic. In: Breveglieri, L., Koren, D., Naccache, D., Seifert, J.P., (eds.) Proceedings of the 3rd International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2006), LNCS, vol. 4236, pp. 173-184, Springer-Verlag, Heidelberg, Germany (2006).

[33] Gilbert, H., Robshaw, M.J.B., Seurin, Y., HB$^{\#}$: increasing the security and efficiency of HB^{+}. In: Smart, N.P., (ed.) Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2008), LNCS, vol. 4965, pp. 361-378, Springer-Verlag, Heidelberg, Germany (2008).

[34] Gilbert, H., Robshaw, M., Sibert, S., An active attack against HB^{+} - a provably secure lightweight authentication protocol. IEE Electronic Letters, 41(21), pp. 1169-1170, (2005).

[35] Goldmann, M., Håstad, J., Razborov, A.A., Majority gates vs. general weighted threshold gates. Computational Complexity, 2(4), pp. 277-300, (1992).

[36] Goldreich, O., Foundations of Cryptography: Volume 1, Basic Tools. Cambridge University Press, Cambridge, UK, 2004.

[37] Goldreich, O., Goldwasser, S., Halevi, S., Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In: Kaliski, B.S., (ed.) Proceedings of the 17th Annual International Cryptology Conference, Advances in Cryptology (CRYPTO 1997), LNCS, vol. 1294, pp. 105-111, Springer-Verlag, Heidelberg, Germany (1997).

[38] Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P., FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I., (eds.) Proceedings of the 9th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007), LNCS, vol. 4727, pp. 63-80, Springer-Verlag, Heidelberg, Germany (2007).

[39] Hammouri, G., Akdemir, K., Sunar, B., Novel PUF-based error detection methods in finite state machines. In: Lee, P.J., Cheon, J.H., (eds.) Proceedings of the 11th International Conference on Information Security and Cryptology (ICISC

2008), LNCS, vol. 5461, pp. 235-252, Springer-Verlag, Heidelberg, Germany (2008).

[40] Hammouri, G., Dana, A., Sunar, B., CDs Have Fingerprints Too. In: Clavier, C., Gaj, K., (eds.) Proceedings of the 11th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2009), Springer-Verlag, Heidelberg, Germany, 2009 (to appear).

[41] Hammouri, G., Öztürk, E., Birand, B., Sunar, B., Unclonable lightweight authentication scheme. In: Chen, L., Ryan, M.D., Wang, G., (eds.) Proceedings of the 10th International Conference on Information and Communications Security (ICICS 2008), LNCS, vol. 5308, pp. 33-48, Springer-Verlag, Heidelberg, Germany (2008).

[42] Hammouri, G., Öztürk, E., Sunar, B., A tamper-proof and lightweight authentication scheme. Pervasive and Mobile Computing, 4(6), pp. 807-818, (2008).

[43] Hammouri, G., Sunar, B., PUF-HB: a tamper-resilient HB based authentication protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M., (eds.) Proceedings of the 6th International Conference on Applied Cryptography and Network Security (ACNS 2008), LNCS, vol. 5037, pp. 346-365, Springer-Verlag, Heidelberg, Germany (2008).

[44] Hoeffding, W., Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301), pp. 13-30, (1963).

[45] Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S., HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M., (eds.) Proceedings of the 8th Workshop on Cryptographic Hardware and Embedded

Systems (CHES 2006), LNCS, vol. 4249, pp. 46-59, Springer-Verlag, Heidelberg, Germany (2006).

[46] Hopper, M.J., Blum, M., Secure human identification protocols. In: Boyd, C., (ed.) Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2001), LNCS, vol. 2248, pp. 52-66. Springer-Verlag, Heidelberg, Germany (2001).

[47] Ignatenko, T., Schrijen, G.J., Skoric, B., Tuyls, P., Willems, F., Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method. In: Proceedings of the IEEE International Symposium on Information Theory (ISIT 2006), 9-14 July, 2006, Seattle, Washington, USA, pp. 499-503, IEEE, Washington, DC, USA (2006).

[48] Juels, A., Sudan, M., A fuzzy vault scheme. Designs, Codes and Cryptography, 38(2), pp. 237-257, (2006).

[49] Juels, A., Wattenberg, M., A fuzzy commitment scheme. In: Tsudik, G., (ed.) Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS 1999), pp 28-36, ACM, New York, NY, USA (1999).

[50] Juels, A., Weis, S.A., Authenticating pervasive devices with human protocols. In: Shoup, V., (ed.) Proceedings of the 25th Annual International Cryptology Conference, Advances in Cryptology (CRYPTO 2005), LNCS, vol. 3621, pp. 293-308, Springer-Verlag, Heidelberg, Germany (2005).

[51] Kaps, J.P., Sunar, B., Energy comparison of AES and SHA-1 for ubiquitous computing. In: Zhou, X. et al., (eds.) Proceedings of the Workshop on Emerging Directions in Embedded and Ubiquitous Computing (EDEUC 2006), LNCS, vol. 4097, pp. 372-381, Springer-Verlag, Heidelberg, Germany (2006).

[52] Kaps, J.P., Gaubatz, G., Sunar, B., Cryptography on a speck of dust. Computer, 40(2), pp. 38-44, (2007).

[53] Katz, J., Shin, J.S., Parallel and concurrent security of the HB and HB$^+$ protocols. In: Vaudenay, S., (ed.) Proceedings of the 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2006), LNCS, vol. 4004, pp. 73-87, Springer-Verlag, Heidelberg, Germany (2006).

[54] Katz, J., Smith, A., Analyzing the HB and HB$^+$ protocols in the "large error" case. Cryptology ePrint Archive, Report 2006/326, 2006. http://eprint.iacr.org/

[55] Kearns, M., Efficient noise-tolerant learning from statistical queries. Journal of the ACM, 45(6), pp. 983-1006, (1998).

[56] Kearns, M., Valiant, L., Cryptographic limitations on learning boolean formulae and finite automata. Journal of the ACM, 41(1), pp. 67-95, (1994).

[57] Kelkboom, E.J.C., Molina, G.G., Kevenaar, T.A.M., Veldhuis, R.N.J., Jonker, W., Binary biometrics: an analytic framework to estimate the bit error probability under Gaussian assumption. In: Proceedings of the IEEE Second International Conference on Biometrics: Theory, Applications and Systems (BTAS 2008), 29 Sept. - 1 Oct. 2008, Washington, DC, USA, pp. 1-6, IEEE Computer Society, Los Alamitos, CA, USA (2008).

[58] Kharitonov, M., Cryptographic lower bounds for learnability of boolean functions on the uniform distribution. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT 1992), July 27-29, 1992, Pittsburgh, PA, USA, pp. 29-36, ACM, New York, NY, USA, (1992).

[59] Klivans, A.R., O'Donnell, R., Servedio, R.A., Learning intersections and thresholds of halfspaces. Journal of Computer and System Sciences, 68(4), pp. 808-840, (2004).

[60] Klivans, A.R., Sherstov, A.A., Cryptographic hardness for learning intersections of halfspaces. Journal of Computer and System Sciences, 75(1), pp. 2-12, (2009).

[61] Klivans, A.R., Sherstov, A.A., Unconditional lower bounds for learning intersections of halfspaces. Machine Learning Journal, 69(2-3), pp. 97-114, (2007).

[62] Kocher, P.C., Jaffe, J., Jun, B., Differential power analysis. In: Wiener, M.J., (ed.) Proceedings of the 19th Annual International Cryptology Conference, Advances in Cryptology (CRYPTO 1999), LNCS, vol. 1666, pp. 388-397, Springer-Verlag, Heidelberg, Germany (1999).

[63] Kocher, P.C., Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N., (ed.) Proceedings of the 16th Annual International Cryptology Conference, Advances in Cryptology (CRYPTO 1996), LNCS, vol. 1109, pp. 104-113, Springer-Verlag, Heidelberg, Germany (1996).

[64] Kulikowski, K.J., Karpovsky, M.G., Taubin, A., DPA on faulty cryptographic hardware and countermeasures. In: Breveglieri, L., Koren, D., Naccache, D., Seifert, J.P., (eds.) Proceedings of the 3rd International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2006), LNCS, vol. 4236, pp. 211-222, Springer-Verlag, Heidelberg, Germany (2006).

[65] Kumar, S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P., The butterfly PUF protecting IP on every FPGA. In: Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST 2008), 9th June 2008, San Francisco, CA, USA, pp. 67-70, IEEE, Washington, DC, USA (2008).

[66] Leander, G., Paar, C., Poschmann, A., Schramm, K., New lightweight DES variants. In: Biryukov, A., (ed.) Proceedings of the 14th International Workshop on Fast Software Encryption (FSE 2007), LNCS, vol. 4593, pp. 196-210, Springer-Verlag, Heidelberg, Germany (2007).

[67] Lee, C., Some properties of nonbinary error-correcting codes. IRE Transactions on Information Theory, 4(2), pp. 77-82, (1958).

[68] Lee, J.W., Daihyun, L., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S., A technique to build a secret key in integrated circuits for identification and authentication applications. In: Digest of Technical Papers in the Symposium on VLSI Circuits, 16 - 19 June, 2004, Honolulu, HI, USA, pp. 176-179, IEEE, Washington, DC, USA (2004).

[69] Levieil, E., Fouque, P.A., An improved LPN algorithm. In: De Prisco, R., Yung, M., (eds.) Proceedings of the 5th International Conference on Security and Cryptography for Networks (SCN 2006), LNCS, vol. 4116, pp. 348-359, Springer-Verlag, Heidelberg, Germany (2006).

[70] Lim, C., Korkishko, T., mCrypton - a lightweight block cipher for security of low-cost RFID tags and sensors. In: Song, J.S., Kwon, T., Yung, M., (eds.) Proceedings of the 6th International Workshop on Information Security Applications (WISA 2005), LNCS, vol. 3786, pp. 243-258, Springer-Verlag, Heidelberg, Germany (2006).

[71] Lim, D., Extracting Secret Keys from Integrated Circuits. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2004.

[72] Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S., Extracting secret keys from integrated circuits. IEEE Transactions on VLSI Systems, 13(10), pp. 1200-1205, (2005).

[73] Linnartz, J.P., Tuyls, P., New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Kittler, J., Nixon, M.S., (eds.) Proceedings of the 4th International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA 2003), LNCS, vol. 2688, pp. 393-402, Springer-Verlag, Heidelberg, Germany (2003).

[74] Lyubashevsky, V., The parity problem in the presence of noise, decoding random linear codes, and the subsetsum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L., (eds.) Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques: Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2005) and Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM 2005), LNCS, vol. 3624, pp. 378-389, Springer-Verlag, Heidelberg, Germany (2005).

[75] Massey, J., Shift-register synthesis and BCH decoding. IEEE Transactions on Information Theory, 15(1), pp. 122-127, (1969).

[76] Munilla, J., Peinado, A., HB-MP: a further step in the HB-family of lightweight authentication protocols. Journal of Computer Networks, 51(9), pp. 2262-2267, (2007).

[77] O'Donnell, C.W., Suh, G.E., Devadas, S., PUF-based random number generation. Technical Report 481, MIT CSAIL, Cambridge, USA (2004).

[78] Öztürk, E., Hammouri, G., Sunar, S., Physical unclonable function with tristate buffers. In: Proceedings of the International Symposium on Circuits and Sys-

tems (ISCAS 2008), 18-21 May 2008, Seattle, Washington, USA, pp. 3194-3197, IEEE, Washington, DC, USA (2008).

[79] Öztürk, E., Hammouri, G., Sunar, S., Towards robust low cost authentication for pervasive devices. In: Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), 17-21 March 2008, Hong Kong, pp. 170-178, IEEE Computer Society, Los Alamitos, CA, USA (2008).

[80] Papoulis, A., Pillai, S.U., Probability, Random Variables, and Stochastic Processes. McGraw-Hill, New York, NY, USA, 2002.

[81] Pentillä, M.J., Reducing Variability in a Semiconductor Manufacturing Environment. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2005.

[82] Poschmann, A., Leander, G., Schramm, K., Paar, C., New light-weight crypto algorithms for RFID. In: Proceedings of the International Symposium on Circuits and Systems (ISCAS 2007), 20-27 May 2007, New Orleans, Louisiana, USA, pp. 1843-1846, IEEE, Washington, DC, USA (2007).

[83] Ravikanth, P.S., Physical One-Way Functions. PhD thesis, Department of Media Arts and Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.

[84] Regazzoni, F., Badel, S., Eisenbarth, T., Großschädl, J., Poschmann, A., Deniz, Z.T., Macchetti, M., Pozzi, L., Paar, C., Leblebici, Y., Ienne, P., A simulation-based methodology for evaluating the DPA-resistance of cryptographic functional units with application to CMOS and MCML technologies. In: Blume, H., Gaydadjiev, G., Glossner, C.J., Knijnenburg, P.M.W., (eds.) Proceedings

of the 2007 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2007), pp. 209-214, IEEE, Washington, DC, USA (2007).

[85] Regev, O., On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R., (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005), pp. 84-93, ACM, New York, NY, USA (2005).

[86] Roos, C., Terlaky, T., Vial, J.P., Interior Point Methods for Linear Optimization (2nd ed.). Springer-Science, New York, USA, 2005.

[87] Rueppel, R.A., Linear complexity and random sequences. In: Pichler, F., (ed.) Proceedings of the 4th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 1985), LNCS, vol. 219 , pp. 167-188, Springer-Verlag, Heidelberg, Germany (1986).

[88] Schift, H., David, C., Gabriel, M., Gobrecht, J., Heyderman, L.J., Kaiser, W., Köppel, S., Scandella, L., Nanoreplication in polymers using hot embossing and injection molding. Microelectronic Engineering, 53(1-4), pp. 171-174, (2000).

[89] Smyth, M.B., Semi-metrics, closure spaces and digital topology. Theoretical Computer Science, 151(1), pp. 257-276, (1995).

[90] Standaert, F.X., Piret, G., Gershenfeld, N., Quisquater, J.J., SEA: a scalable encryption algorithm for small embedded applications. In: Ferrer, J.D., Posegga, J., Chreckling, D., (eds.) Proceedings of the 7th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications (CARDIS 2006), LNCS, vol. 3928 , pp. 222-236, Springer-Verlag, Heidelberg, Germany (2006).

[91] Stinson, D., Ameli, F., Zaino, N., Lifetime of Kodak writable CD and photo CD media. Eastman Kodak Company, Digital & Applied Imaging, NY, USA (1995).

[92] Tiri, K., Akmal, M., Verbauwhede, I., A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC 2002), 24-26 September 2002, Florence, Italy, pp. 403-406, IEEE, Washington, DC, USA (2002).

[93] Toprak, Z., Leblebici, Y., Low-power current mode logic for improved DPA-resistance in embedded systems. In: Proceedings of the International Symposium on Circuits and Systems (ISCAS 2005), 23-26 May 2005, Kobe, Japan, pp. 1059-1062, IEEE, Washington, DC, USA (2005).

[94] Tuyls, P., Akkermans, A.H.M., Kevenaar, T.A.M., Schrijen, G., Bazen, A.M., Veldhuis, R.N.J., Practical biometric authentication with template protection. In: Kanade, T., Jain, A.K., Ratha, N.K., (eds.) Proceedings of the 5th International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA 2005), LNCS, vol. 3546, pp. 436-446, Springer-Verlag, Heidelberg, Germany (2005).

[95] Tuyls, P., Schrijen, G.J., Skoric, B., van Geloven, J., Verhaegh, N., Wolters, R., Read-proof hardware from protective coatings. In: Goubin, L., Matsui, M. (eds.) Proceedings of the 8th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2006), LNCS, vol. 4249, pp. 369-383, Springer-Verlag, Heidelberg, Germany (2006).

[96] Tuyls, P., Skoric, B., Kevenaar, T., (eds.), Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting. Springer-Verlag, London, UK, 2007.

[97] Valiant, L. A theory of the learnable. Communications of the ACM, 27(11), pp. 1134-1142, (1984).

[98] Vanderbei, R.J., Linear Programming: Foundations and Extensions (3rd ed.). Springer-Verlag, New York, USA, 2008.

[99] Willems, F.M.J., The context-tree weighting method: extensions. IEEE Transactions on Information Theory, 44(2), pp. 792-798, (1998).

[100] Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J., The context-tree weighting method: basic properties. IEEE Transactions on Information Theory, 41(3), pp. 653-664, (1995).