Technische Universität München

Lehrstuhl für Sicherheit in der Informationstechnik
an der Fakultät für Elektrotechnik und Informationstechnik

Attacking and Protecting Ring Oscillator Physical Unclonable Functions and Code-Offset Fuzzy Extractors

Dominik Merli

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines **Doktor-Ingenieurs (Dr.-Ing.)** genehmigten Dissertation.

Vorsitzende:		Univ.-Prof. Dr. D. Schmitt-Landsiedel
Prüfer der Dissertation:		
	1.	Univ.-Prof. Dr.-Ing. G. Sigl
	2.	Univ.-Prof. Dr.-Ing. M. Ortmanns, Universität Ulm

Die Dissertation wurde am 24.09.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 09.04.2014 angenommen.

Abstract

Many cryptographic algorithms require secure storage of secret keys. One common way to achieve this is the usage of flash memory. However, because of additional process costs, this is often not an option for low-cost devices. On the other hand, less expensive memory technologies like fuses are vulnerable against simple optical inspection attacks. Physical Unclonable Functions (PUFs) are information storage primitives, which offer a compromise storage solution by deriving device-specific information from submicron process variations of microchips. They can be beneficial in terms of higher security than fuses, but also in less process costs than flash memory. During the last decade, PUFs have increasingly gained interest in industry as well as in academia. Among the variety of proposed architectures, the Ring Oscillator PUF (RO PUF) evolved as one of the most promising PUFs. One reason for that is its implementation advantage on Field-Programmable Gate Arrays (FPGAs). Moreover, its basis, namely Ring Oscillator (RO) frequencies, is known to have good statistical properties that support its usage as a source for device-specific PUF bits. Its basic concept is to measure RO frequencies by counters and perform a relative comparison on the result values to get a secret PUF response bit. This is repeated with different ROs to obtain a unique bit sequence. One of the main applications of PUFs is key generation, where a cryptographic key is reconstructed based on noisy PUF output bits and a set of helper data that is generated during an enrollment process. The Code-Offset Fuzzy Extractor (COFE) is one of the most popular key generation algorithms. It is able to handle the noise of an RO PUF's output bits to ensure reliable key generation. Despite immense research on developing efficient PUF implementations and key derivation algorithms, one important aspect has been more or less neglected until now: PUF-based systems have to face a variety of physical attacks when out in the field.

In this thesis, I investigate the resistance of RO PUFs and COFEs against physical attacks and suggest countermeasures to protect them. First, I show how to implement RO PUFs on FPGA devices. On the one hand, designers have to achieve as identical as possible RO implementations to avoid

vulnerabilities by statistical imbalances. On the other hand, the frequency measurement circuits have to provide accurate values to allow for an efficient and reliable RO PUF. Regarding attacks, one potential target is the PUF primitive itself. As shown in this thesis, one possible physical attack vector for RO PUFs is capturing its ElectroMagnetic (EM) emission to learn about its internal secrets. This technique can be used by an adversary to observe the internally measured RO frequencies of an RO PUF and then map them to their corresponding ROs. Thereby, the RO PUF's security is broken. Further, an attacker could use side-channel attacks mounted at the key generation algorithm, e.g., a COFE, to recover internal secrets. There, physical characteristics of devices such as power consumption or EM emission are observed during cryptographic operations to derive information about the used secret key by statistical means. My work demonstrates the feasibility of several attacks on RO PUFs and COFEs and provides means to counteract these threats.

My first contribution to the attack resilience of RO PUFs is the investigation of their localized EM emission. Since localized EM measurements have to be mounted as closely as possible to the die surface, I determined the influences of chip decapsulation on RO frequencies in a beforehand analysis. I implemented 256 ROs on an FPGA and compared their characteristics before and after removing the chip's package. The results show only very slight deviations clearing the way for localized EM analyses of RO PUFs. Following, I developed and performed an attack on a standard RO PUF implementation which enables an adversary to observe RO frequencies and map them to the correct ROs. With this attack, all generated RO PUF output bits can be disclosed, which breaks the RO PUF's security. As a countermeasure, I propose an RO comparison strategy which does not allow to map emitted frequencies to their corresponding ROs and thereby eliminates the discovered attack vector. However, the gained security comes with an resource overhead, which is why I also present a way to improve an RO PUF's efficiency while maintaining its attack resistance.

During a second analysis phase, I discovered an enhanced attack which enables an attacker to extract secret PUF bits even if the protection against the previous attack is implemented. Here, the crucial point is that during each measurement run of an RO PUF, each RO frequency is measured by one distinct counter. If an attacker is able to separately observe the RO frequencies sequentially measured by each counter, he is finally only left with the complexity of testing all possible counter comparison combinations. This is a fairly simple task since the number of counters is usually very low. To perform this attack, an attacker has to be able to locally separate the EM emissions of all RO PUF measurement counters or other emitting

components of each measurement path. In this thesis, I show that this is possible after collecting EM traces over the whole die area of an RO PUF device. To protect RO PUFs from this attack, I suggest a location masking countermeasure destroying the relation between measurement locations and measured ROs. Additionally, I propose interleaved placement of RO PUF components to enhance its attack resilience against localized EM attacks.

In the last part of this thesis, I focus on the vulnerabilities of COFEs against side-channel attacks. Since attacks on key generation algorithms can be applied no matter which underlying PUF is used and how high its security level might be, they have a high relevance for embedded systems using secret keys derived from PUFs. First, I developed a differential power analysis attack exploiting side-channel leakage originating from Error-Correcting Code (ECC) modules, which are an essential part of COFEs. This differential attack can be performed if an adversary is able to manipulate helper data, which is the only input data to a COFE besides a noisy variant of the secret PUF response bits. Thereby, all PUF output bits can be disclosed, which breaks the security of the PUF-based key generation system. In a second attack on COFEs, I show that the Toeplitz hashing, which is often used as the extractor function in COFEs, is prone to simple power analysis. The input bits of this function correspond to the enrolled secret PUF response bits in COFEs and can be revealed by analyzing the operation-dependent leakage exhibited by its hardware implementation.

To protect COFEs from the described attacks, I propose a masking scheme which, in contrast to standard masking schemes, does not rely on completely random masks, but on encoded masks representing random codewords of an ECC. This enables the protection of linear ECCs without losing their fundamental error correction features. Further, the proposed masking scheme can be extended throughout the Toeplitz hashing function to protect COFEs also against the shown simple power analysis attack. The result is a consistently masked COFE, which protects the generated key until it is handed over to a cryptographic module using it in a masked or demasked fashion.

Summarizing, this thesis demonstrates attacks on RO PUFs as well as on COFEs and proposes countermeasures to protect these circuits, which is an important contribution to the attack resilience of PUF-based hardware security systems.

Acknowledgements

After years of research, successes, setbacks, and a good deal of experiences, there is only one thing left to say: thank you!

I sincerely appreciate the support, guidance, and advice of my supervisor Prof. Dr.-Ing. Georg Sigl. I also thank my second examiner Prof. Dr.-Ing. Maurits Ortmanns.

During my time at Fraunhofer AISEC, I got to know great people, some of which I meanwhile call friends. I thank Dieter Schuster, Dr.-Ing. Johann Heyszl, Benedikt Heinz, Mattias Hiller, Robert Hesselbarth, and Dr. Rainer Plaga for scientific discussions and cooperations, and Philipp Zieris for proof-reading my dissertation. Further, I thank Dr. Frederic Stumpf and Prof. Dr. Claudia Eckert for their support and trust in my work.

I thank Prof. Dr. Helia Hollmann for initially encouraging me to pursue a doctorate.

Finally, I sincerely thank my father and my mother for their endless support, and my brother, my sister, and all my friends for being there for me whenever I was in need.

Contents

List of Figures

List of Tables

Nomenclature

ASIC	Application-Specific Integrated Circuits
BR PUF	Bistable Ring PUF
C-IBS	Complementary Index-Based Syndrome
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide Semiconductor
COFE	Code-Offset Fuzzy Extractor
COSS	Code-Offset Secure Sketch
CRP	Challenge-Response Pair
DPA	Differential Power Analysis
ECC	Error-Correcting Code
EM	ElectroMagnetic
FFT	Fast Fourier Transform
FIB	Focused Ion Beam
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
IBS	Index-Based Syndrome
IC	Integrated Circuit
LFSR	Linear Feedback Shift Register
LUT	Look-Up Table

NVM	Non-Volatile Memory
OTP	One-Time Programmable
POWF	Physical One-Way Function
PUF	Physical Unclonable Function
RO	Ring Oscillator
ROM	Read-Only Memory
RO PUF	Ring Oscillator PUF
SCA	Side-Channel Analysis
SIMPL	SIMulation Possible but Laborious
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis
SRAM	Static Random-Access Memory
TBR PUF ...	Twisted Bistable Ring PUF
UCF	User Constraints File
VHDL	Very high speed integrated circuit HDL

Chapter 1

Introduction

Our society lives in a world full of electronic devices. Daily, billions of people rely on the support of a diversity of communication channels, intelligent infrastructures and an incredible number of digital services. We use cars, computers, tablets, mobile phones, credit cards and any other electronic devices as if it would be the most normal thing in the world. However, this new electrified world also entails risks. What if thieves found out which digital sequences have to be transmitted to cars to unlock them without any mechanical trace? What if adversaries were able to listen to every call and read all text messages on mobile phones from any remote location? What if credit cards could be cloned and used for illegal transactions everywhere in the world? The answer to these questions is: Then, our society would not function anymore. Therefore, it is inevitable to consider information security, i.e., the security of private, corporate and governmental data, as one of the most important and essential topics of our modern society.

The area of information security deals with a variety of applications ranging from cloud services distributed over hundreds of servers in different countries to enterprise and industrial networks prone to espionage and manipulation to secure microchips in mobile phones and smartcards. The continuous trend of increasing integration density leads to the fact that millions of networked high-performance computation devices surround us all day. Believing the vision of the Internet of Things, this will even increase in the next years. Then, not only every car, mobile phone and smartcard will be equipped with an embedded processor, but sensors and electronic devices will be everywhere around us, communicating with each other and the environment. However, many of these devices operate on private information, i.e., secrets not meant to be accessible to everyone. Therefore, the development of robust and affordable security solutions for embedded systems is a very important research topic.

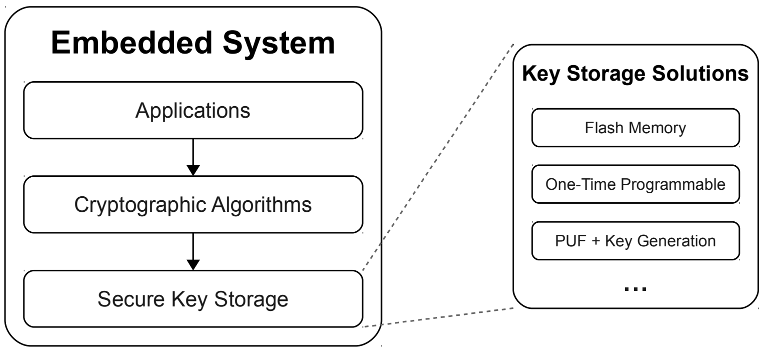


Figure 1.1: Embedded system with secret key

One fundamental requirement for many embedded security applications is that a secret cryptographic key is somehow stored on an embedded device as shown in Figure 1.1. This key can be used for encryption and decryption routines, but also for authentication protocols. The most comfortable way of realizing this key storage is by integrating a flash memory, which can be programmed with a specific key during manufacturing. However, in order to implement a flash memory, an additional cost-intensive manufacturing process is required, which is not affordable for low-cost products. On the other hand, simpler storage implementations, e.g., fuses or mask Read-Only Memories (ROMs), are relatively easy to read out by optical die inspection.

The research field of Physical Unclonable Functions (PUFs) deals with alternative information storage mechanisms, which intrinsically provide a security mechanism to impede duplication. The basic concept of every PUF can be described as shown in Figure 1.2. There, a challenge input configures a physical structure that is measured afterwards to obtain a response value that depends on the PUF's unique physical properties and their configuration. One goal of PUF research can be lower implementation costs than flash memory, while keeping the resistance against optical inspection higher than in the case of fuses. Further, PUFs can be used to enhance tamper resistance of information storage devices compared to conventional memories. Described in a more philosophical way, this means that PUFs free embedded security systems from the fetters of requiring a conventional Non-Volatile Memory (NVM) with additionally implemented, dedicated security mechanisms for storing secret key material. This might seem abstract, but it allows to combine storage and security mechanisms by measuring unique, not necessarily silicon-based features of an embedded system that can be used as a basis for generating a device-specific bit string.

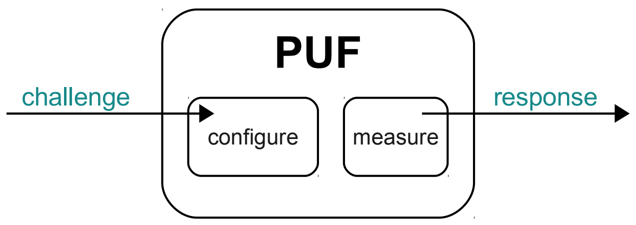


Figure 1.2: Basic PUF concept

Practical advantages of this liberation can be of financial but also of security-related value. Since PUFs based on sophisticated materials usually require development and integration of new manufacturing processes and thereby induce significant costs, they constitute a quite narrow application niche. One example application in this niche is the combination of two subsystems, e.g., a smartcard's plastic body that is physically connected to and measured by an integrated security chip [EFK+12]. There, the information storage is implemented by the binding of plastic structures and microchips, which also increases the system's security by protecting it from attacks tampering with the unique material constellation. On the other hand, silicon PUFs [GCvDD02, SD07, GKST07, KGM+08, YD10a, CCL+11] can be implemented with standard Complementary Metal Oxide Semiconductor (CMOS) processes and can even be instantiated on Field-Programmable Gate Arrays (FPGAs), resulting in a low additional costs. Hence, constructions based on CMOS component measurements, e.g., propagation delays or oscillator frequencies, were preferred by researchers as well as industry during the last decade. These PUFs enable security applications on devices where an additional manufacturing process for integrated NVM is not affordable.

A popular representative of the class of silicon PUFs is the Ring Oscillator PUF (RO PUF) [SD07]. Its basic concept is shown in Figure 1.3. It measures the frequencies of a set of identically implemented Ring Oscillators (ROs), which are considered to be randomly distributed. Since these frequencies vary depending on environmental conditions such as ambient temperature and supply voltage, the RO PUF architecture performs a relative comparison of two measured RO frequencies to compensate occurring disturbances. The RO PUF's response bit signals if the first RO frequency was higher or lower than the second one. This measurement process is repeated for several subsequently overlapping RO pairs, e.g., RO_1 and RO_2, RO_2 and RO_3, and so on, to acquire a bit string long enough for the desired application. This

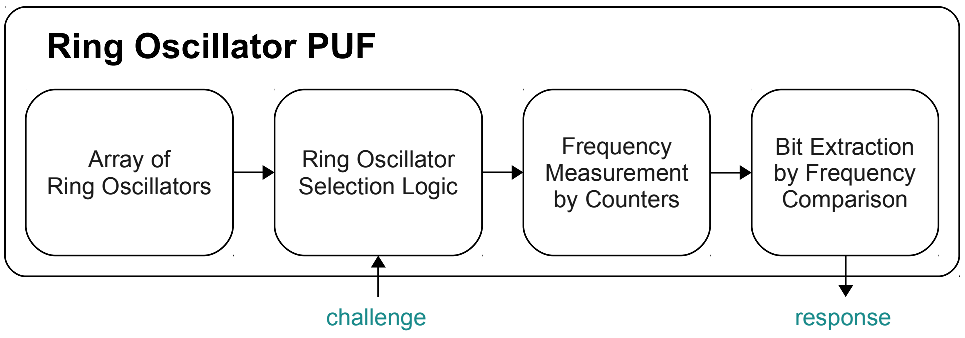

Ring Oscillator PUF
Array of Ring Oscillators
Ring Oscillator Selection Logic
Frequency Measurement by Counters
Bit Extraction by Frequency Comparison
challenge
response

Figure 1.3: RO PUF concept

architecture has been shown to be advantageous over other PUFs in terms of statistical properties [KKR+12] and does not suffer from FPGA implementation limitations as much as other PUFs do [MMS09]. In Chapter 2, I give a more detailed introduction to the field of PUFs, the most popular silicon PUFs and why I chose to use the RO PUF as one of main subjects of my thesis. In Chapter 3, I discuss how to implement RO PUFs on FPGA devices in a solid way and which details have to be handled with care.

The difference between a PUF and a conventional memory lies in its security mechanism protecting it from duplication. Therefore, PUFs are often advertised to be unclonable, or at least hard to clone. However, cloning is usually just a question of practical feasibility. Comparing PUFs to One-Time Programmables (OTPs), e.g., fuses, they have the potential to achieve a higher resilience against optical imaging attacks, because their secret information is stored in sub-micron physical structures when the PUF device is not powered. In this case, attackers need costly high-precision equipment and sophisticated non-influencing measurement methods to attack the device. However, PUF output bits are determined by measuring its unique physical characteristics, which can be observed and analyzed by an attacker much easier. As a result, the practical unclonability becomes questionable. In Chapter 4, I show that the claim of unclonability does not hold for standard RO PUFs, since they reveal their unique properties by ElectroMagnetic (EM) emissions of subsequent RO frequency comparisons, which can be captured by an attacker. Then, a clone of an RO PUF instance can be generated. I also propose an improved concept that eliminates the conceptual problem leading to the found leakage. In the second part of Chapter 4, I show that an adversary equipped with high resolution EM equipment is even able to extract the unique frequencies and the mapping to their ROs from the previously pro-

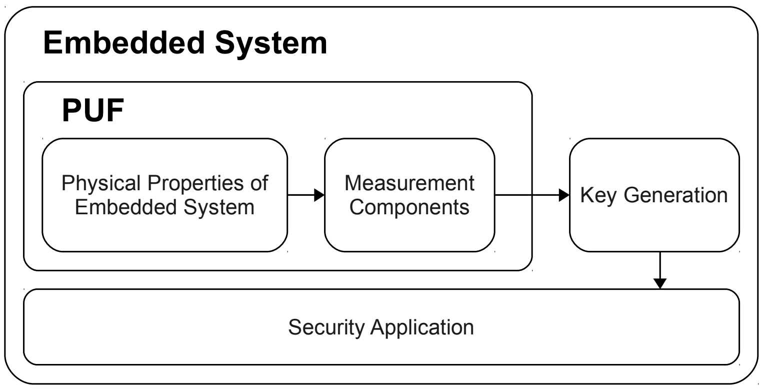


Figure 1.4: General PUF usage for key generation

tected design by localized EM measurements [HMH+12a, HMH+12b] on the die of an RO PUF chip. This, again, allows to establish a model which behaves like the original instance of an RO PUF. To protect RO PUFs against such sophisticated attacks, I propose two countermeasures. One breaks the relation between measured ROs and the correspondingly used measurement components by randomizing the RO measurement process. The other one is a practical hiding countermeasure including interleaved placement and interwoven routing of RO PUF components and signals.

The main use case for RO PUFs, and also for many other PUF constructions, is the generation of a secret cryptographic key from unique chip features, as shown in Figure 1.4. Based on physical measurements resulting in secret, device-specific data, e.g., RO frequencies, a reliable key is generated. However, physical measurements vary over time and environmental conditions like supply voltage and temperature. Because of that, ROs may, e.g., exhibit decreasing frequencies when their operating temperature is increased. This noise leads to flipping PUF output bits, which are not tolerated by cryptographic algorithms. Therefore, PUF output bits cannot be used directly, but have to be processed by key generation algorithms [DRS04, BDHV07, YD10b, PD11, HMSS12] that compensate variations in PUF output bits and can provide a reliable and still unique bit string. These algorithms are also called helper data algorithms, because, during an enrollment phase, they collect information about the noisy PUF fingerprint and generate so-called helper data. It consists of redundancy information about the internal PUF characteristics, but does not reveal a significant amount of information about the secret PUF bits to an attacker. Some of these algorithms, directly extract a device-specific bit string, a secret key,

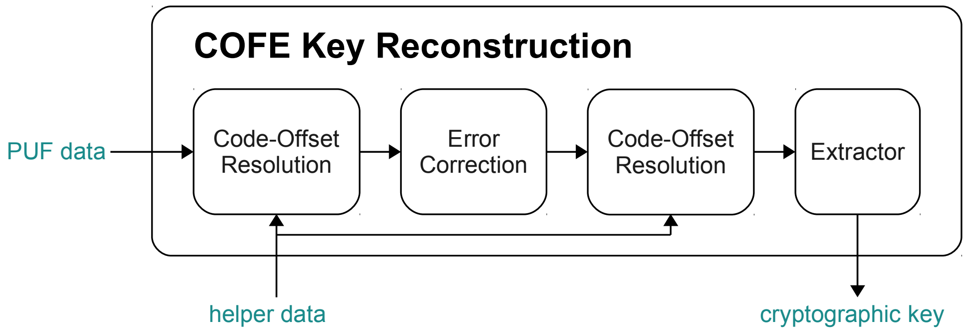


Figure 1.5: Basic COFE reconstruction concept

from PUF output bits. Others take a key as an additional input and bind it to the unique PUF fingerprint by involving it in the helper data generation. All methods have in common that, when out in the field, they are able to reconstruct the enrolled key from a noisy version of the PUF fingerprint and the stored helper data.

One of the first and still popular algorithms of this kind is called Code-Offset Fuzzy Extractor (COFE) and was proposed by Dodis et al. [DRS04]. There, the helper data consists of an XOR code-offset between all PUF output bits and a randomly chosen codeword of an Error-Correcting Code (ECC). Its basic reconstruction process is shown in Figure 1.5. There, the code-offset is resolved by XORing the helper data with the noisy PUF fingerprint, resulting in a noisy codeword that can be corrected by the corresponding ECC module, if less than a certain amount of bit flips occurred. The corrected codeword can then be XORed to the helper data recovering the original PUF output bits. Since these bits usually have a bias, i.e., are not uniformly distributed, an extractor module is necessary to convert them into a cryptographic key. A popular extractor algorithm for efficient COFE implementations [BGS$^+$08, MTV09] is the Toeplitz hashing [Kra94], which basically consists of a Linear Feedback Shift Register (LFSR) and an XOR accumulator.

PUF-based key generation can be used in a variety of devices to replace expensive NVM key storage or to enable a unique key in the first place. Smartcards, automotive and industrial control units, and many other embedded systems can make use of this technique. However, all of these devices are out in the field, where users and attackers can tamper with them and mount physical attacks. While invasive manipulations by a Focused Ion Beam (FIB), or semi-invasive laser fault injection attacks can only be performed with specialized and expensive equipment, a much more affordable attack method for adversaries is Side-Channel Analysis (SCA). Since its first publication in 1996 [Koc96], these attacks have been considered a strong threat to embedded systems. There, the principle idea is to monitor the power consumption or the timing behavior of a device during the execution of a cryptographic algorithm. Based on statistical measures, information about internally processed secret intermediate values can be gathered. There are two methods that have become very popular for the analysis of symmetric and asymmetric cryptographic algorithms during the last years and that can be performed even with low-cost equipment: Simple Power Analysis (SPA) [KJJ99] and Differential Power Analysis (DPA) [KJJ99]. For SPA, a collected EM or power trace can directly be interpreted, i.e., internal secrets can directly be extracted from a small number of traces. On the other hand, DPA observes tiny differences in intermediate values of an algorithm by collecting a rather high number of traces with different input values. With the help of statistical tools, this differential approach is able to extract secrets even from very noisy measurements.

For a PUF key generation system, all parts of the reconstruction algorithm of a COFE, as shown in Figure 1.5, can be the target of SCA. After enrollment, when the device is out in the field, it is performed every time the key is generated from the secret PUF response bits. Hence, it is highly probable that adversaries will choose to attack the key generation algorithm of PUF-based devices instead of directly tackling a PUF's information storage. Therefore, I see it as inevitable to analyze side-channel vulnerabilities of COFEs. In Chapter 5, I demonstrate two possible attacks and present a countermeasure for secure COFE implementations. The first attack is a DPA, where helper data manipulation leads to changes of intermediate values. There, the side-channel leakage of the error correction modules in COFEs is exploited. It can be used to extract the secret PUF bits by observing the power consumption or EM emission of intermediate values of ECCs. The second attack focuses on a hardware implementation of the Toeplitz hashing function, which exhibits changing power consumption depending on the function's input data, namely, the secret PUF response bits. This can be exploited by SPA. As a countermeasure for both attacks, I propose a codeword masking scheme. It enables the protection of the key generation process beginning at code-offset helper data processing over error correction and Toeplitz hashing up to the point where the key is used by a crypto module. It uses random codewords of linear ECCs to break the relation between intermediate values and helper data input bits, which can also be extended throughout the Toeplitz hashing extractor. Thereby, the essential error correction capabilities of the ECCs implemented in COFEs are preserved and protection against SPA and DPA attacks is achieved.

This thesis is organized as follows. Chapter 2 gives relevant background information on PUFs, key generation algorithms and physical attacks. In Chapter 3, implementation obstacles for RO PUFs on FPGAs are discussed. Attacks directly targeting the EM emission of RO PUFs and protective countermeasures are shown in Chapter 4. In Chapter 5, two attacks on COFE implementations are demonstrated and a codeword masking scheme is proposed as a countermeasure. Conclusions drawn from the results of this thesis and suggestions for future work are summarized in Chapter 6.

Chapter 2

Background

This chapter provides background information on the research field and applications of PUFs and related ideas. It also explains some of the most popular silicon PUF architectures, including the RO PUF. Further, PUF-based key generation algorithms, such as fuzzy extractors and secure sketches based on a code-offset, are introduced. To give an overview of relevant attack vectors, this chapter also lists physical attacks, which pose a threat to PUF-based security systems.

The chapter is organized as follows. In Section 2.1, an overview of the research field of PUFs is given, while Section 2.2 explains several proposed PUF architectures for CMOS chips and FPGAs. Section 2.3 details relevant key generation algorithms. An introduction to physical attacks is presented in Section 2.4.

2.1 Security Based on Physical Properties

This section details the development of the research field of PUFs and similar security mechanisms based on unique physical properties.

2.1.1 Unique Physical Properties

The most basic idea of using unique physical features for security purposes is to measure physical objects with external equipment and compare the measurements to earlier acquired results in order to verify the authenticity of an object. This principle is schematically shown in Figure 2.1. In practice, this approach is applicable in scenarios where products that do not possess a digital processing unit have to be protected from unauthorized cloning. Examples are banknotes, clothes, and optical data carriers. The security

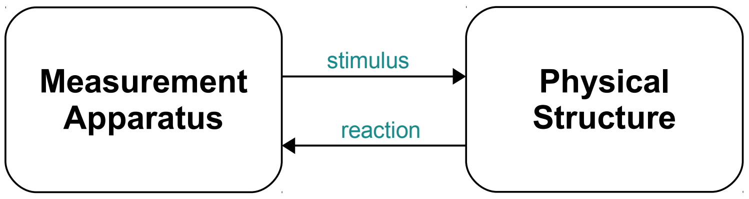


Figure 2.1: External measurement of unique physical features

of this protection scheme lies in the fact that an attacker is not able to reproduce a given physical structure with sufficient precision, even if he is able to measure its characteristics with a high resolution.

Identification methods based on unique physical device properties date back to the Cold War. In 1989, it was reported [GM89] that Bauder had proposed a technique to uniquely mark missiles by spraying a thin random layer of particles onto each weapon. Illuminating the device from different angles, light is scattered by this layer in a device-specific way. Recording the reflection patterns directly after manufacturing enables the identification of the device at a later stage. These markings were regarded as unforgeable, because even if an attacker knew the different illumination angles and could retrieve the resulting speckle patterns, he would not be able to built a particle layer that exhibits exactly the same unique features.

In 2001, Pappu et al. introduced the term Physical One-Way Function (POWF) [Pap01, PRTG02]. Their idea was to stimulate an optical token with a laser beam from one side and capture the resulting speckle pattern on the other side. The angle of the laser beam was meant to be the input of the POWF and the pattern captured by a camera was the output. The POWF, namely the light transformation, was assumed to be very complex and not invertible for an attacker, i.e., provided with a speckle pattern, it was impossible to tell the corresponding input laser angle. This idea is sometimes referred to as the first PUF construction, but, actually, it is no PUF because it relies on external analog measurements of an optical scattering structure, which are only compared to previously recorded results.

Later, devices with the described security mechanism were also called unique objects [Rüh09] or certificates of authenticity [DK07]. It was shown that these methods can also be implemented by measuring metallic structures with arrays of radio frequency antennas [DK07]. Even the physical characteristics of standard optical data carriers can be used for this purpose [HDS09].

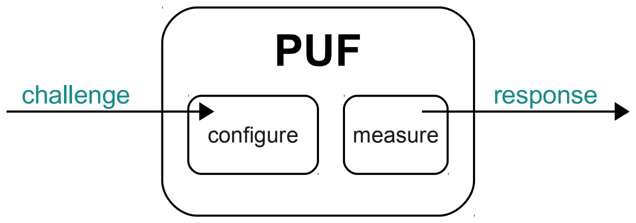


Figure 2.2: PUFs as challenge-response primitives

2.1.2 Physical Unlonable Functions

Loosely speaking, PUFs are functions which map an input value, the challenge, to an output value, the response, where each specific instance of this function depends on internally measured physical characteristics of each PUF device. The basic structure of a PUF is shown in Figure 2.2. However, this characterization does not suffice for a clear separation from similar security primitives and to stress the specific value of PUFs. In my opinion [MP13], PUFs can be defined as information storage devices which include a security mechanism that impedes their duplication and that is indivisible connected to its storage mechanism. However, until now, the PUF research community has not agreed on a comprehensive and widely accepted definition that could serve as a solid basis for certification processes and future research. Therefore, in the following, I explain the milestones of PUF history to convey a sense of what PUFs are meant to be.

In 2002, to the best of my knowledge, the first idea for using intrinsic electrical properties for identification and authentication of Integrated Circuits (ICs) was described. Layman et al. filed a patent [LCNT02] which explains that transistor manufacturing mismatch consistently influences the start-up values of memory cells. They suggested to use the resulting start-up states as a unique device signature.

In the same year, Gassend et al. [GCvDD02, GCvDD03] coined the term *physical unclonable function* because they disagreed that PUFs and one-way functions match regarding meaning and properties, as suggested by the earlier proposal of POWFs. In the same publication, they proposed the arbiter PUF architecture, which is one of the best analyzed PUF implementations to date. The arbiter PUF has a binary challenge input, which configures the delay parameters of two identically implemented signal paths. Measuring the delay difference of each delay path configuration by an arbiter element leads to a single bit PUF response. Figure 2.3 shows a basic challenge-response

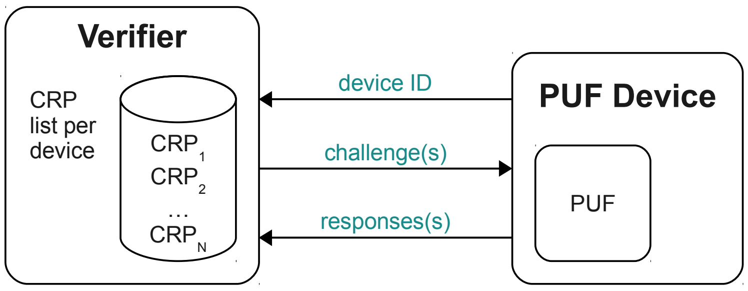


Figure 2.3: PUF-based challenge-response authentication

device authentication scheme based on PUFs, which was also described in their publication. First, a verifier collects Challenge-Response Pairs (CRPs) directly after manufacturing the PUF device and saves them in a secure database. At a later stage, when the device is out in the field, the verifier can remotely challenge the PUF with randomly selected challenges from the database and compare the acquired responses to the stored ones. If the responses match, except for a certain tolerance percentage of noise involved in PUF measurements, then the device can be regarded as authentic. Compared to conventional challenge-response algorithms, the advantage of PUF-based device authentication is that no binary secret key is available on the PUF device at all times.

In 2004, Dodis et al. [DRS04] proposed an algorithm to reliably extract a cryptographic key from noisy sources, such as, biometric features. However, this so-called fuzzy extractor also paved the way for PUF-based key generation, because PUFs are also unique but noisy sources, similar to biometric data. The similarity between PUFs and biometrics is also reflected in the term fingerprints of microchips, which is often used to explain PUFs. As shown in Figure 2.4, in order to generate a key from a PUF, during an enrollment phase, a noisy PUF response is collected and redundancy information about it is stored. Later, this information helps reliably reconstructing the originally enrolled PUF response bits without revealing enough information to enable an attack.

The U.S. company Verayo was founded in 2005 to bring the arbiter PUF to the market. They offer device authentication solutions based on arbiter PUFs, but also key generation modules based on the secret bits that can be extracted from an arbiter PUF.

The first PUF architecture designed as a countermeasure against invasive physical attacks was proposed by Tuyls et al. in 2006. The basis for

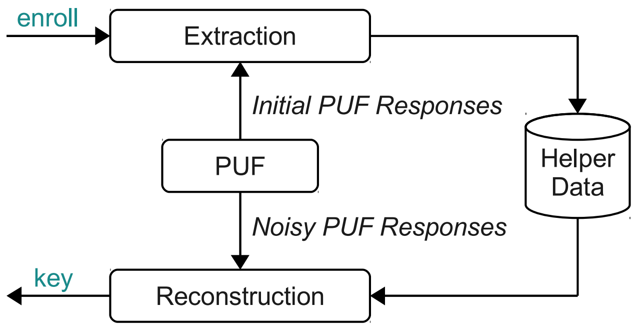


Figure 2.4: Basic key generation process

the coating PUF [TSŠ+06] is a microchip that has comb-shaped capacitive sensors distributed over its top metal layer. The attack resilience is achieved by covering the chip with a protective coating, consisting of an aluminophosphate matrix material and randomly distributed TiO_2 and TiN particles. A device-specific key can be generated based on the material-dependent capacity values measured by the sensors on top of the chip. Any invasive tampering, e.g. by a FIB, will alter the material characteristics and thereby implicitly destroy the unique device key. However, the additional process of applying the random layer to a microchip adds significant costs to the manufacturing process of a device, compared to pure silicon PUFs.

In 2007, two still very important PUF architectures, mainly used for cryptographic key generation, have been published. Suh and Devadas proposed the RO PUF [SD07], which generates PUF response bits based on relative RO frequency comparisons. This type of PUF is very popular, because, in practice, it can be properly implemented in FPGAs as well as Application-Specific Integrated Circuits (ASICs). Additionally, a PUF architecture built from Static Random-Access Memory (SRAM) cells has been proposed, which made its way to commercial availability through the 2008 founded Dutch company Intrinsic ID. It uses the power-up states of an SRAM memory to generate a device-specific key. Further, PUF proposals based on cross-coupled latches [KGM+08] or flip-flop start-up behavior [MTV08] followed the trend of providing unique physical sources for cryptographic key generation, but never reached the popularity and quality of RO PUFs and SRAM PUFs.

Afterwards, the development of PUFs with complex challenge-response behavior, which can be used for device authentication, has gained more prominence. Majzoobi et al. [MKP08a] and Rührmair et al. [RSS+10] showed that machine learning attacks can be used to model a variety of PUFs, including arbiter PUFs as well as their extensions [LLG+04, GLC+04, LLG+05, SD07, MKP08b]. This demonstrated that many of the previously proposed PUFs are not suited for secure challenge-response authentication. An interesting approach to obtain a more complex behavior is the algorithmic processing of PUF output values, e.g., calculating the sum of RO frequencies depending on the challenge input, in the case of the sum PUF [YD10a, YMSD11, YSS+12]. Another architecture exploits the fuzzy shape of glitches generated in a combinational network [SS10]. A third idea builds on the oscillation behavior of a bistable ring [CCL+11, CCL+12], which is supposed to be more complex than an arbiter PUF's signal propagation and is therefore worth further investigations.

In 2009, public key cryptography pendants of PUFs have been developed independently by Beckmann and Potkonjak as public PUFs [BP09] and by Rührmair et al. as SIMulation Possible but Laborious (SIMPL) systems [Rüh09], which were later joined by a related concept called time-bounded FPGA authentication [MEK10]. In these PUF applications, a public PUF simulation model is provided as kind of a public key, which allows to simulate the PUF behavior. It is used to obtain the correct PUF response upon a random challenge which is then compare to the hardware PUF's response. The security mechanism of this PUF application lies in the fact that the original PUF device can produce the response within a very short time span, which cannot be achieved by any simulation. The strong advantage of this idea is the fact that no secret information would be present in the device at all. However, a practical problem remains. Until now, there have been very few investigations about how to specify and evaluate the required time window between real device performance and simulation in order to reach a specific level of security.

Similar to the idea of a coating PUF, a recent proposal [EFK+12] aims at protecting smartcards from invasive attacks. Esbach, me et al. describe an architecture, where the combination of a security microcontroller and the unique material properties of the card body form a PUF. Light emitting diodes on the smartcard chip generate light, which propagates through the thin layers of material and is then captured by photo sensors. The measured sensor values are used to generate a secret key. Therefore, any material manipulation results in an altered key.

Summarizing, no matter if PUFs have been proposed for challenge-response authentication, key generation or tamper resistance, they always have the goal to combine information storage with a protective security mechanism.

2.1.3 PUF Quality Measures

The quality of a PUF strongly depends on the statistical properties of its fundamental physical information sources, e.g., RO frequencies. However, a source with a good statistical distribution does not automatically lead to a high-quality PUF. In order to assess the quality of PUFs, several measures have been proposed [HYKS10, MGS11]. However, no standardized set of tests is used and accepted by PUF researchers, yet. Also, practical issues like efficient implementation or intrinsic tamper resistance are important key facts to judge the suitability and quality of a PUF for a certain application. However, these properties are hard to capture in statistical figures.

In the following, I explain the three most important statistical measures used to judge new PUF constructions. They are essential for a basic understanding of a PUF's quality.

Intra-Device Hamming Distance

The intra-device Hamming distance is a measure for the noise of a PUF architecture. Variations of it have been called 'reliability' [MGS11] or 'steadiness' [HYKS10]. This measure is determined for every single PUF device. Looking at the intra-device Hamming distance of many PUF devices allows to judge the PUF architecture's typical noise level. $HD_{intra,i}$ of a chip i can be calculated as the mean of Hamming distances $HD(\mathbf{r}_{i,ref}; \mathbf{r}_{i,t})$ of an n-bit reference response bit vector $\mathbf{r}_{i,ref}$ at normal operating conditions and m n-bit response bit vector samples $\mathbf{r}_{i,t}$ at different times or operating conditions. The optimal value, indicating no noise, is 0 %.

$$HD_{intra,i} = \frac{1}{m}\sum_{t=1}^{m}\frac{HD(\mathbf{r}_{i,ref}; \mathbf{r}_{i,t})}{n} \times 100\% \tag{2.1}$$

Uniformity

Another property of each PUF device is its distribution of output values, captured by the measures of 'uniformity' [MGS11] and 'randomness' [HYKS10]. However, I favor the term uniformity, because it better describes that it is a measure for the uniform distribution of ones and zeros among all PUF output bits of a device. It can be determined as the mean value of all n response bits $r_{i,l}$ of a device i, where 50 % represents perfect uniformity.

$$Uniformity_i = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\% \tag{2.2}$$

Inter-Device Hamming Distance

This measure is also called 'uniqueness' [HYKS10, MGS11] because it describes how good an individual PUF can be distinguished in a whole population of PUFs. The following formula calculates the inter-device Hamming distance HD_{inter} as the average of all Hamming distances $HD(\mathbf{r_i};\mathbf{r_j})$ of two n-bit response bit vectors $\mathbf{r_i}$ and $\mathbf{r_j}$ of two PUFs on two different chips i and j. The best value of this measure is 50 %, which means that every PUF device differs in half of its response bits from any other device.

$$HD_{inter} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(\mathbf{r_i};\mathbf{r_j})}{n} \times 100\% \qquad (2.3)$$

2.2 Silicon PUFs

Using standard CMOS technology for PUF implementations [GCvDD02, SD07, GKST07, KGM+08, YD10a, CCL+11] instead of other materials is favored because of three reasons. First, no additional manufacturing process is required to produce these kind of PUFs, as it is the case for, e.g., coating PUFs [TSŠ+06]. This way, additional costs are kept low. Second, chip manufacturers already have statistics about variations and corner cases of their CMOS technology, which allows for a better insight to identify which effects can be exploited by PUFs. Third, also FPGAs can make use of PUFs that are based on simple logic gates.

This section gives an overview of silicon PUF architectures, focusing on the ones repeatedly used and analyzed in literature. It also compares all explained architectures to the RO PUF and explains why it is one of the most flexible, reliable and best-analyzed PUF architectures.

2.2.1 Ring Oscillator PUF

Since the introduction of silicon PUFs, oscillating circuits have been regarded as possible sources to extract unique properties of silicon devices [GCvDD02]. The nowadays popular RO PUF architecture, as shown in Figure 2.5, was introduced in 2007 [SD07]. Its secret information originates from inevitable silicon manufacturing variations of ROs, which lead to randomly distributed RO frequencies. These frequencies are measured by counters increasing with every rising edge of their input signal. The RO PUF's binary output bit is obtained by a relative comparison of two RO frequencies. If the first RO exhibits a higher frequency than the second one, the output will be set to

0, and otherwise to 1. This relative comparison already partly compensates for environmental influences such as temperature and supply voltage variations. This RO PUF response generation process is repeated for several subsequently overlapping RO pairs, i.e., RO_1 and RO_2, RO_2 and RO_3, and so on, to acquire a device-specific bit sequence that can be used, e.g., for key generation.

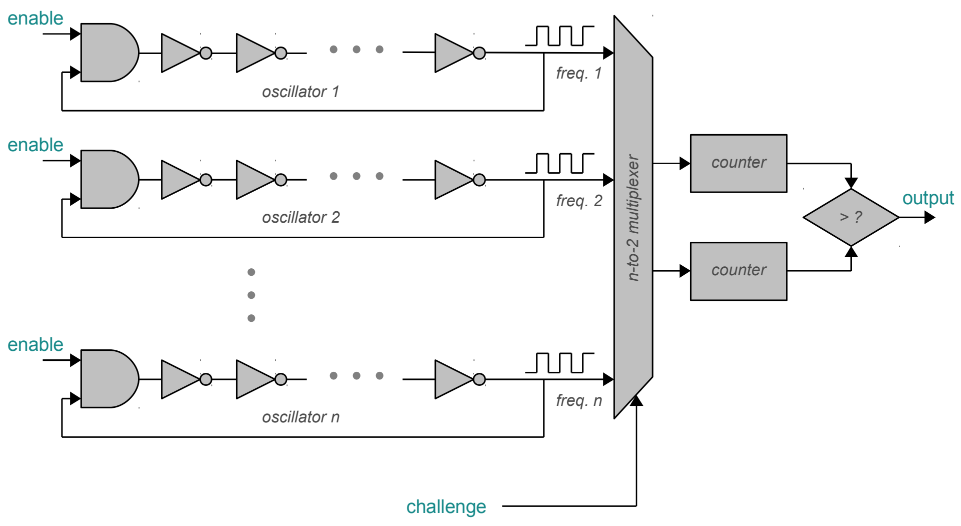


Figure 2.5: RO PUF architecture

One of the strongest advantages of RO PUFs is that they do not require symmetric placement and routing as other PUFs do [MMS09]. It is only important that identical RO instances are used, which can be achieved, e.g., by hard macros or placement constraints. This fact also enables the use of RO PUFs on FPGAs with good statistical properties [MCMS10]. Also, for ASIC implementations, RO PUFs achieve one of the best and mostly unbiased statistical distributions [KKR+12] leading to good uniqueness and uniformity.

Since the publication of the RO PUF, several variants of it have been proposed, e.g., to enhance its reliability [MS09]. Also, improvements regarding the information extraction have been published, e.g., a grouping algorithm to maximize an RO PUF's secret extraction [YQ10] and an entropy distiller for RO PUFs [YQ11].

Summarizing, the reasons for taking the RO PUF as the basis for this thesis are its broad applicability on FPGAs and ASICs, its outstanding statistical properties and its popularity in literature. These facts make the

RO PUF one of the most widespread and promising PUF architectures for future security applications.

2.2.2 Arbiter PUF

The first silicon PUF, proposed in 2002, is the arbiter PUF [GCvDD02]. Its delay-based architecture is shown in Figure 2.6. There, a sequence of multiplexer stages implements two configurable delay paths. Depending on the challenge input bits, at every stage, the multiplexers select a straight signal propagation or a path crossing. To obtain a response from a challenge configuration of an arbiter PUF instance, a signal edge is provided at the delay paths' input. At the end, an arbiter component, e.g., a latch, decides which path had a shorter propagation delay and then outputs a 0 or a 1, respectively. This architecture is currently commercially available by the company Verayo.

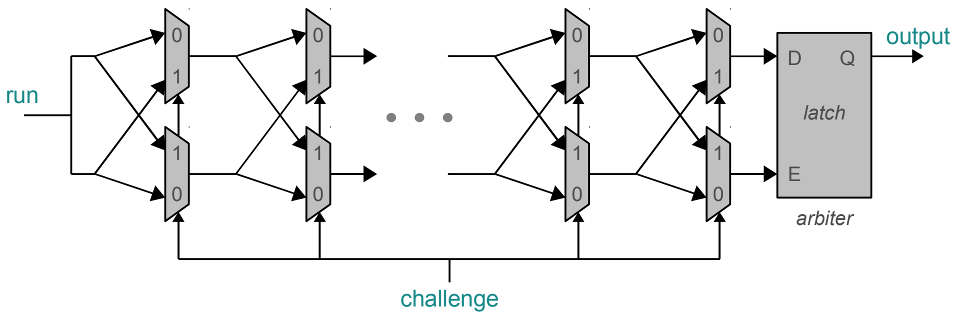


Figure 2.6: Arbiter PUF architecture

Originally, it was proposed for challenge-response authentication, but machine learning attacks [MKP08a, RSS$^+$10] have shown that the arbiter PUF model can be easily learned and simulated.

In order to counteract machine learning attacks, variants of the basic arbiter PUF have been proposed [LLG$^+$04, GLC$^+$04, LLG$^+$05, SD07, MKP08b]. XOR arbiter PUFs, feed-forward arbiter PUFs and lightweight secure PUFs have the common idea to introduce non-linearity in order to complicate the model of the standard arbiter PUF. Non-linear behavior can be achieved by the following measures:

- **PUF response XOR.** Several arbiter PUFs are measured in parallel and their responses are XORed at the end to obtain a single response bit. Since this bit depends on the non-linear XOR operation, it is harder to draw conclusions about the internal delay components.

- **Feed forward paths.** Additional arbiters are introduced at intermediate stages of an arbiter PUF. The resulting output bit of these arbiters is fed forward to control one of the following multiplexer stages.
- **Input and output network.** A lightweight secure PUF encloses several arbiter PUFs with input networks that mix challenge bits and an output network that transforms all output bits non-linearly to a smaller response string with the help of XOR operations.

These extensions improve the security of an arbiter PUF, but also increase its resource consumption. However, although exhibiting more complexity, it has been shown, that they are still prone to machine learning attacks [MKP08a, RSS$^+$10] and only very large versions reached exhaustive computation limits. Therefore, the arbiter PUF's suitability for challenge-response authentication is questionable, but it can still be used to generate bits for PUF-based key generation. However, its statistical properties [KKR$^+$12], which are relevant for key generation applications, also lie behind the ones of RO PUFs and SRAM PUFs.

Compared to RO PUFs, arbiter PUFs have a lower area requirement, but also exhibit a lower entropy [KKR$^+$12], which makes them less suitable for efficient key generation. Additionally, one requirement of arbiter PUFs is the symmetric layout of all delay components, signal lines and the arbiter component. This demand is hard to fulfill on FPGAs and requires very careful design of ASIC implementations.

2.2.3 SRAM PUF

The SRAM PUF is one of the most important PUF architectures to date. The idea of using uninitialized start-up values of memory cells was already described in a patent [LCNT02] in 2002, but the popular SRAM PUF architecture was proposed in 2007 [GKST07]. Figure 2.7 shows an SRAM cell that, in the ideal case, is completely balanced. However, in real implementations, one can observe a tendency for each cell to fall to one or the other side after power-up. An SRAM PUF exploits these device-specific start-up bits and uses them to generate a device-specific cryptographic key.

The SRAM PUF exhibits a high quality regarding statistical properties [KKR$^+$12]. However, it requires non-initialized SRAM memory, which is only available on very few FPGA devices [MTV08]. Therefore, on the statistical side, the SRAM PUF is on a similar level as the RO PUF, but

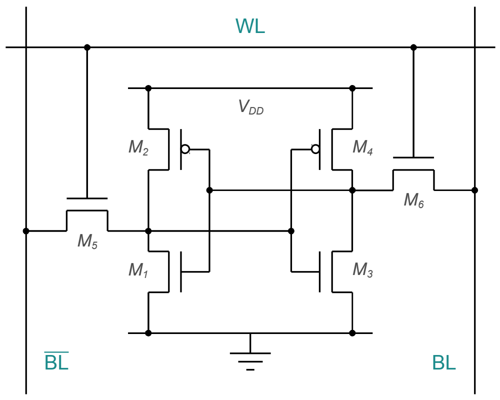


Figure 2.7: SRAM PUF cell

when it comes to practical FPGA implementation, the RO PUF shows clear advantages. But nevertheless, for ASICs or specific microcontrollers, the SRAM PUF architecture can provide additional security features. It has reached the stage of commercial availability by the company Intrinsic-ID.

2.2.4 Butterfly PUF

The butterfly PUF [KGM+08] was introduced in 2008 to enable SRAM-like cells on FPGAs.

As shown in Figure 2.8, it consists of two cross-coupled latches. To obtain a device-specific output bit, first, one of the latches is pulled to 1 and the other one to 0. After releasing the latches from this forced state, the cell oscillates and finally falls into an output state that depends on the physical characteristics of both latches, such as their signal propagation delay.

Although the butterfly PUF was designed for FPGA implementation, the required symmetric routing of the cross-coupled latch connections is hard to achieve [MMS09]. Also, this PUF architecture cannot compete with the quality and robustness of an RO PUF's statistical distributions, even when implemented on an ASIC [KKR+12].

2.2.5 Bistable Ring PUF

The Bistable Ring PUF (BR PUF) [CCL+11, CCL+12] originates from the idea of having an inverter ring with an even number of inverters, where the smallest possible ring with two inverters is very similar to an SRAM cell.

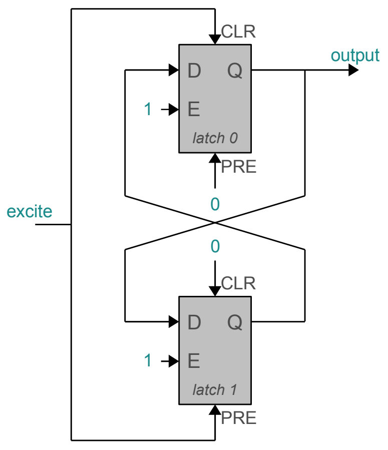


Figure 2.8: Butterfly PUF architecture

These ring structures have two stable states, where the nodes have alternating values, i.e., 010...101 and 101...010. The challenge bits of a BR PUF configure which delay element, out of two possible, is inserted into the bistable ring, as shown in Figure 2.9. The final response bit is acquired by capturing the state of one ring stage. This architecture enables a large number of different ring configurations. Therefore, the authors proposed it for device authentication and, because of less symmetric routing than for arbiter PUFs, they also emphasized its suitability for FPGA implementations.

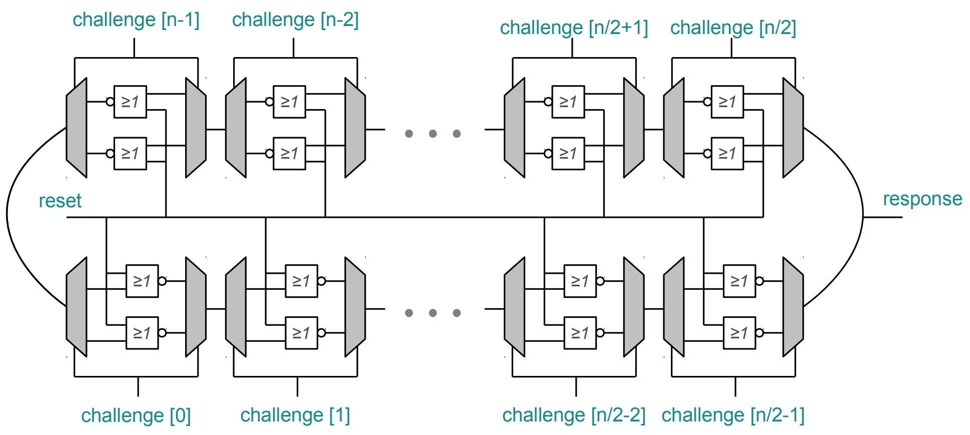
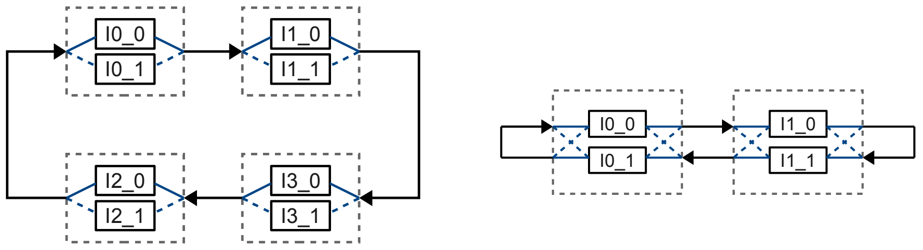


Figure 2.9: Bistable ring PUF architecture



(a) Abstract BR PUF with 8 elements for a 4-element ring and a 4-bit challenge (b) Abstract TBR PUF with 4 elements for a 4-element ring and a 2-bit challenge

Figure 2.10: Comparison between BR PUF and TBR PUF

The authors claim that the oscillating behavior leading to the PUF response might be more complex than the arbiter PUFs response measurements. However, I discovered a strong bias in its architecture, leading to a strong lack in uniformity of many BR PUF devices. The reason for that is that a flipped challenge bit replaces one delay element with another, while to original one is disregarded. If both delays are similar, a challenge change can lead to the same behavior and, consequently, to the same response. If this occurs for many challenge combinations, a strong bias in the PUF responses can be observed.

A recent patent application of me and Schuster [MS13] explains that the BR PUF architecture has the described structural flaw, which results in highly biased PUF responses for many devices. The invention describes a new ring structure, the Twisted Bistable Ring PUF (TBR PUF). There, delay elements are not plugged in and plugged out depending on the challenge bits, but change positions within the ring, leading to a twist in the ring structure that results in a balanced response distribution. The difference between the two architectures is depicted in Figure 2.10.

Drawing the structure of the TBR PUF is more difficult as in the case of the BR PUF because the building blocks of a TBR PUF have two inputs at the left and two outputs at the right (which is neglected in Figure 2.10 for simplicity). Therefore, the back path of the ring structure, i.e., the lower path, contains loops from a block's output to the next left block's input as shown in the detailed architecture in Figure 2.11.

First analyses suggest that the TBR PUF architecture is more complex than arbiter PUFs and BR PUFs and also exhibits a more uniform distribution of PUF responses, compared to the BR PUF, i.e. shows a better value for uniformity of PUF devices. However, its oscillating behavior is not completely understood yet and it has not been analyzed as much as the RO PUF.

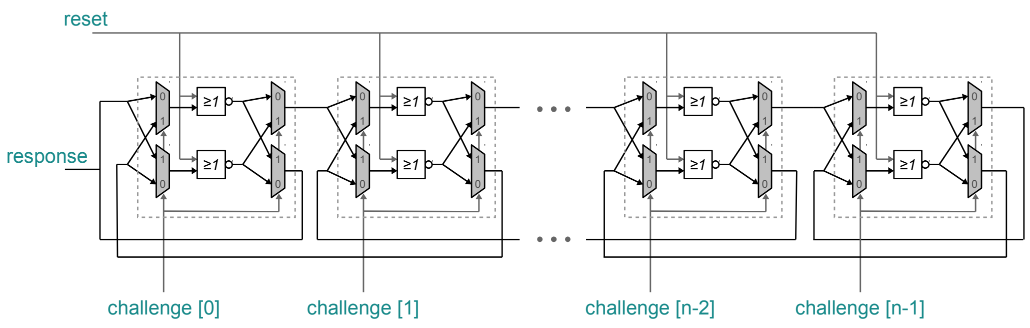


Figure 2.11: Twisted bistable ring PUF architecture

Although promising, this architecture is not at a stage where it would make sense to analyze it regarding physical attacks. Therefore, the RO PUF is favored over the TBR PUF for the analyses provided in this thesis. However, the TBR PUF can be regarded as an interesting candidate for future PUF research.

2.2.6 Sum PUF

Using recombination functions for PUFs [YD10a] is another interesting idea to provide complex and yet reliable PUF behavior. The sum PUF [YD10a, YMSD11, YSS$^+$12] is the first architecture resulting from this idea. As shown in Figure 2.12, it accumulates the differences of RO frequency pairs. Each corresponding challenge bit decides if the difference (either negative or positive) enters the sum with its original or its inverted sign. In the end, the sign of the resulting sum gives the PUF response value, while the less significant bits can be used as reliability information.

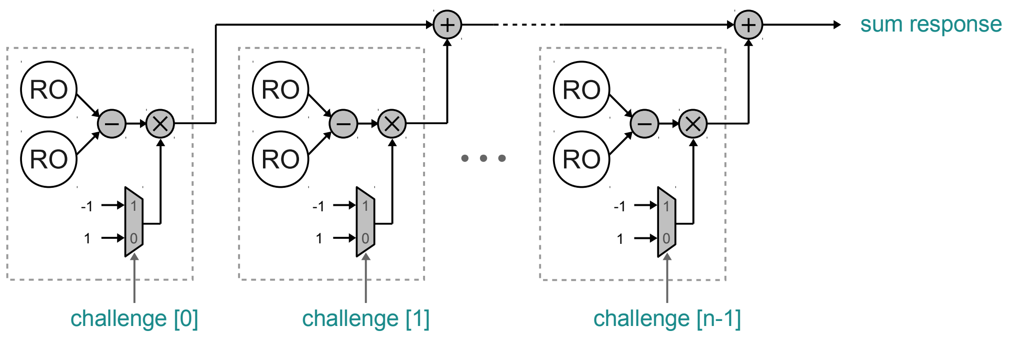


Figure 2.12: Sum PUF architecture

The sum PUF can be regarded as an extension of the RO PUF and is obviously based on the same primitives. However, it has not gained as much attention as RO PUFs and is still to be analyzed.

The reason why the sum PUF is mentioned in this collection of important silicon PUFs, is the fact that it also relies on measuring RO frequencies. Therefore, its architecture and also future developments based on ROs might be vulnerable to similar attacks as described in Chapter 4. Also, the countermeasures proposed in this thesis can be a first step towards protecting sum PUFs.

2.3 Key Generation Based on PUFs

Every physical measurement involves a certain amount of noise, so do PUFs. For device identification and authentication by CRPs, this can be handled by the number of requested CRPs and a tolerance limit at the verifier side. However, for a cryptographic key, noisy bits are unacceptable and a high reliability has to be guaranteed. This can be achieved by helper data algorithms, which generate helper data during an enrollment phase and are usually supported by error correction modules. This data can then be used to reliably reconstruct the secret defined at enrollment when the device is out in the field. During the last decade, a variety of algorithms [DRS04, BDHV07, YD10b, PD11, HMSS12] has been proposed, among which, the code-offset constructions proposed by Dodis et al. [DRS04] are the most popular ones. Also, the Index-Based Syndrome (IBS) coding scheme proposed by Yu and Devadas [YD10b] has received notable attention. This section explains the most popular helper data algorithms, focusing on their reconstruction functions, which are used in the field for key generation. These modules are necessary for reliable key recovery, but are also prone to physical attacks as shown in this thesis.

2.3.1 Code-Offset Secure Sketch

One of the most basic key generation algorithms, the secure sketch [DRS04], was proposed by Dodis et al. in 2004 and follows a similar principal as the earlier proposed fuzzy commitment scheme by Juels and Wattenberg [JW99]. It originates from the field of biometrics, where it is used to reconstruct the enrolled biometric data from a noisy version of biometric features and stored helper data, the sketch. Since PUFs operate on bits instead of features, as in the case of biometrics, the Code-Offset Secure Sketch (COSS) is the most suited variant of the secure sketch family. It can be used to reliably recover original PUF bits, but also to embed a secret key into PUF response bits. It is used in cases where the PUF output bits are uniformly distributed. In this thesis, I show how these basic building blocks of key generation can be attacked and thereupon protected.

Reconstruction of PUF Bits

A COSS can be used to recover enrolled biometric data and, therefore, in the same way, can reconstruct the original PUF bits acquired during enrollment. First, a random binary codeword $\mathbf{c} = encode_C(\mathbf{x})$ of a code C has to be generated by encoding a random bit vector $\mathbf{x}$. Then, the helper data bit vector (the sketch) $\mathbf{w} = \mathbf{c} \oplus \mathbf{r}$ is the result of an XOR operation, i.e., the code-offset, between $\mathbf{c}$ and the secret PUF response bit vector $\mathbf{r}$. If the bit flips contained in the noisy PUF response bit vector $\mathbf{r}'$ do not exceed the correction capabilities of the decoding function $decode_C$ of code C, the original PUF response bit vector $\mathbf{r}$, which can be used to generate a binary key $\mathbf{k}$, can be computed as follows:

$$\mathbf{r} = encode_C(decode_C(\mathbf{r}' \oplus \mathbf{w})) \oplus \mathbf{w} = correct_C(\mathbf{r}' \oplus \mathbf{w}) \oplus \mathbf{w} \tag{2.4}$$

I denote the combination of the decoding and encoding functions $encode_C$ and $decode_C$ as $correct_C(.) = encode_C(decode_C(.))$, for simplicity. The architecture of the reconstruction module is depicted in Figure 2.13.

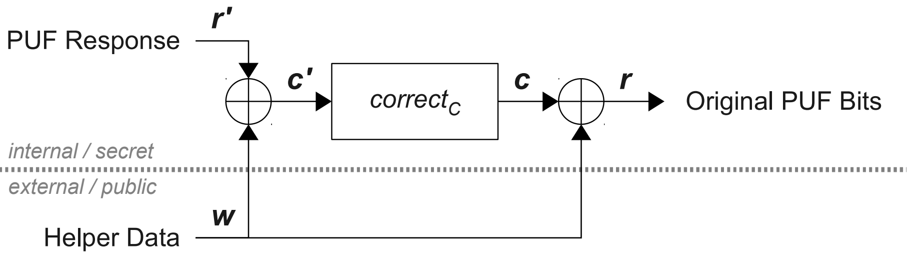


Figure 2.13: PUF response reconstruction module of a COSS

Embedding a Secret Key

The basic COSS architecture can also be used to embed a secret key into PUF output bits, which results in an even simpler architecture. There, during the enrollment phase, no random codeword is used, but an external binary key $\mathbf{k}$ is encoded to a binary codeword $\mathbf{c}$ of a code C: $\mathbf{c} = encode_C(\mathbf{k})$. The public helper data is again the code-offset between codeword $\mathbf{c}$ and the initial PUF response bit vector $\mathbf{r}$, i.e., $\mathbf{w} = \mathbf{c} \oplus \mathbf{r}$. During the reconstruction phase, the reconstruction module, as shown in Figure 2.14, is used to reliably generate the embedded key $\mathbf{k}$ from a noisy PUF response bit vector $\mathbf{r}'$:

$$\mathbf{k} = decode_C(\mathbf{r}' \oplus \mathbf{w}) \tag{2.5}$$

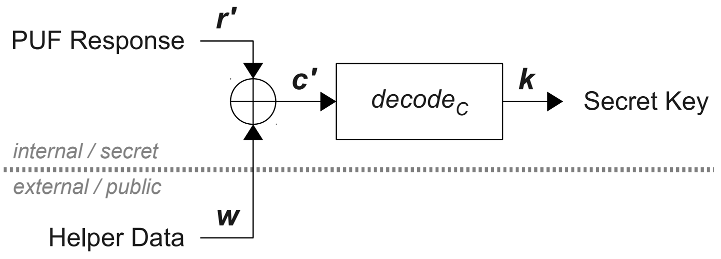


Figure 2.14: Key reconstruction module of a COSS

2.3.2 Code-Offset Fuzzy Extractor

If the underlying PUF used for key generation does not exhibit a completely random distribution of bits in the response bit vector $\mathbf{r}$, a COSS has to be extended by an extractor *Ext* to obtain a COFE as proposed by Dodis et al. [DRS04]. *Ext* can be any extractor algorithm that takes the corrected PUF response bit vector $\mathbf{r}$ as an input and compresses it to a binary key $\mathbf{k}$, where each key bit is completely random for each device and, therefore, suited for cryptographic purposes. For efficient COFE implementations, until now, Toeplitz hashing [Kra94] was a popular choice for the required extractor function.

For COFEs, the COSS architecture to extract the enrolled PUF bits, is simply extended by an extractor *Ext*. The reconstruction module, as shown in Figure 2.15, first resolves the code-offset between the helper data bit vector $\mathbf{w}$ and the noisy PUF output bit vector $\mathbf{r}'$ and then corrects the resulting noisy codeword $\mathbf{c}' = \mathbf{r}' \oplus \mathbf{w}$ to $\mathbf{c} = correct_C(\mathbf{c}')$. Afterwards, the helper data is XORed to the corrected binary codeword to obtain the enrolled PUF response bit vector $\mathbf{r} = \mathbf{c} \oplus \mathbf{w}$. Finally, $\mathbf{r}$ is compressed to a binary key $\mathbf{k}$ with a uniform distribution of ones and zeros by an extractor *Ext*:

$$\mathbf{k} = Ext(correct_C(\mathbf{r}' \oplus \mathbf{w}) \oplus \mathbf{w}) \tag{2.6}$$

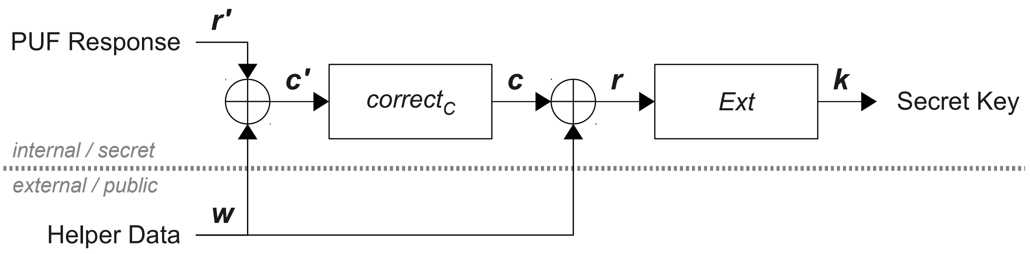


Figure 2.15: COFE reconstruction module

2.3.3 Robust Sketches and Robust Fuzzy Extractors

In 2005, Boyen et al. [BDK$^+$05] explained that an active adversary can gain information about the user's biometric data (or PUF) by maliciously manipulating the communication between a server (implementing a fuzzy extractor) and the user. As a solution, they proposed robust sketches and robust fuzzy extractors which can detect helper data manipulations.

Their proposed robust constructions use a hash function H to generate a hash value $\mathbf{h} = H(\mathbf{k}, \mathbf{w})$ of the helper data and the enrolled key. This hash value is then stored along with helper data $\mathbf{w}$. During each reconstruction, a (maybe manipulated) secret key $\hat{\mathbf{k}}$ is generated as in a standard secure sketch or fuzzy extractor, but before using $\hat{\mathbf{k}}$, the hash value $\hat{\mathbf{h}} = H(\hat{\mathbf{k}}, \hat{\mathbf{w}})$ is compared to the previously generated value $\mathbf{h}$. If the hash values match, $\hat{\mathbf{w}}$ was not manipulated and $\hat{\mathbf{k}} = \mathbf{k}$, otherwise, $\hat{\mathbf{k}}$ will be discarded.

While these constructions are secure for remote scenarios, they are still vulnerable to side-channel attacks based on helper data manipulation, as I explain later in this thesis.

2.3.4 Index-Based Syndrome Coding

The Index-Based Syndrome (IBS) [YD10b] coding scheme was developed by Yu and Devadas in 2010. It embeds a secret key into noisy PUF response bits. During enrollment, it tests each bit of a block of PUF response bits for its reliability. Depending on the key bit to embed, it then stores the index of the most reliable 1 or 0 as helper data. The stored indices are selected from the PUF bit blocks again during reconstruction, which leads to increased reliability.

An extension of IBS, Complementary-IBS (C-IBS), was developed under my supervision and published by Hiller, me et al. [HMSS12]. There, not only one index, but a set of indices is stored as helper data for each key bit. The indices point to reliable bits of the same value as the key bit, but also to complementary bits. During reconstruction, the set of indices per key bit is used to select all necessary PUF bits, which are then decoded to the corresponding key bit. Thereby, compared to IBS, the stability of key bits is further enhanced.

Usually, the IBS or C-IBS output bit vectors are still not as reliable as desired for many applications. Therefore, these methods can be supported by error correction modules to generate a highly reliable cryptographic key. However, attacks against ECCs, e.g., as shown in this thesis, may also pose a threat to IBS and C-IBS implementations, but the masking scheme I propose in Chapter 5 can also be used to protect ECCs in IBS and C-IBS modules.

2.3.5 Other Key Embedding/Extracting Algorithms

Two further key generation techniques have been presented in literature that, however, have not gained as much attention as the ones presented above. They are not relevant for this thesis and are therefore only briefly explained for completeness.

One method is based on quantization index modulation [BDHV07], a class of data hiding codes. It originates from the field of watermarking. The authors show that embedding a watermark into a host signal has close resemblance to embedding a secret key into noisy PUF responses. Therefore, they model fuzzy extractors in terms of watermarking notation and show that this enables to derive measures for the embedding rate, the reliability and the information leakage characteristics of fuzzy extractors.

Another interesting approach [PD11] publishes parts of PUF response sequences during the enrollment phase, depending on the key bits to embed. It uses efficient pattern matching techniques to identify the correct position of the published pattern within the generated PUF response bit sequence during reconstruction. The bits of the resulting position index can then be used as secret key bits. However, in contrast to the other presented algorithms, this scheme only works for PUFs that have a large challenge-response space.

2.4 Physical Attacks

During the last decade, a large number of papers have been published [Wag12] about physical attacks on smartcards, secure microcontrollers, or FPGAs. Some of these attacks also pose a threat to secure embedded systems based on PUFs. This section gives relevant background information on physical attacks on embedded systems. It supports the comprehension of the analyses described in this thesis, which demonstrate some of the first physical attacks on PUFs and key generation algorithms.

2.4.1 Secret Extraction by Physical Measurements

Measuring physical characteristics that are supposed to be secret is one of the most straight-forward attacks, but, nonetheless, a very important one in the area of PUFs. Its practical feasibility is always determined by the manufacturing technology of the device under attack as well as the precision of an attacker's equipment. These attacks have two main goals: memory content extraction and functionality reverse engineering. Optical imaging [KK99, Sko05] is one of the relevant techniques to extract information from microchips, such as secret cryptographic keys.

For PUFs, optical imaging is not as critical as for other memory structures, e.g., mask ROMs or fuses. However, PUFs are also a kind of memory, which exploit information stored in sub-micron variations. One possible way to extract a PUF's secret information, e.g., its unique RO frequencies, are EM emission measurements, as I detail in Chapter 4. Recently, it was also shown that photonic emission analysis [SNK$^+$12] can be used to analyze PUF secrets such as SRAM start-up values [HNBJP13].

One should note that these direct physical measurement attacks always target the information storage component, e.g., a memory or a PUF, and not an algorithm.

2.4.2 Side-Channel Attacks

Since their introduction in 1996 by Kocher [Koc96], the analysis of side-channels in embedded security devices has gained a lot of interest in academia as well as in industry. The basic principle is that implementations of cryptographic algorithms leak secret information through side-channels, such as execution time [Koc96], power consumption [KJJ99], and electromagnetic [GMO01, QS01] emission. Therefore, an attacker is able to extract secrets from cryptographic algorithms running on an embedded system by measuring its physical properties during execution time.

PUF-based key generation is also an algorithm, which processes a secret, namely, the secret key it generates and the secret PUF response bits. Therefore, this class of attacks is highly relevant for key embedding and extraction algorithms. In this thesis, I consider attacks on the power consumption and the EM emission of PUF devices, because these physical effects can be measured quite well with state-of-the-art laboratory equipment and still allow for strong attacks.

One class of power analysis attacks is SPA [KJJ99]. There, an attacker tries to interpret power consumption or EM emission traces, which have been recorded during the execution of an algorithm. This is possible, if the traces show a strong dependency on the secret data involved in the calculation. This is the case, e.g., if a secret is processed serially and a 0 input bit leads to a much lower/higher power consumption than a 1 input bit. Then, one can visually interpret the power consumption trace and link power consumption values to secret data bits. These attacks require a low noise level, because they usually operate only on one or a few traces. In Section 5.3, I show how the Toeplitz hashing module of a COFE can be attacked by SPA.

The most popular power attack technique is DPA [KJJ99]. It exploits the differences in power consumption of intermediate values of a cryptographic operation when provided with changing input data. Compared to SPA, it usually requires a larger number of traces to successfully extract the used secret. One of its advantages is that it allows to exploit very small power variations, even in the presence of noise disturbances. Also, with an increasing number of recorded traces with different input values, the guess for the secret data gets more and more confident. In Section 5.2, I show a DPA attack that can be mounted on error correction modules of COFEs.

For a comprehensive overview of SCA, I refer the reader to the book *Power Analysis Attacks: Revealing the Secrets of Smart Cards* of Mangard et al. [MOP07].

2.4.3 Other Physical Attacks

The attacks shown in this thesis focus on measuring power consumption and EM emission, but there are also other physical attacks that can be used to break cryptographic devices. Some of them are important to consider when designing a PUF-based security system and should be regarded as future work for PUFs. One of these methods are fault attacks [BDL97], which have been discovered by Boneh et al. in 1997. They allow to derive information about internal secrets by analyzing the effects of faults injected into cryptographic operations. This attack vector can also be relevant for key generation algorithms. Further, more sophisticated techniques within the class of semi-invasive attacks, as introduced by Skorobogatov [Sko05], can be used to extract PUF properties. When it comes to deeply invasive attacks, e.g., microprobing and FIB manipulations [KK99], pure silicon PUFs and especially their post-processing are as vulnerable as conventional integrated circuits, if no dedicated protection is implemented.

Chapter 3

Implementation of RO PUFs on FPGAs

In this chapter, I analyze the specific problems designers face when designing an RO PUF on an FPGA. This includes implementation of identical ROs and their consistent placement. Further, I discuss the usage of asynchronous counters as frequency measurement circuits. I also show a spatial frequency distribution analysis over an FPGA die, which is influenced by measurement logic placement. Parts of this chapter have been published in the paper *Improving the Quality of Ring Oscillator PUFs on FPGAs* at the 5th Workshop on Embedded Systems Security in 2010 [MSE10]. The results presented in this chapter provide a solid basis for reliable RO PUF implementations.

The chapter is organized as follows. Section 3.1 explains the functionality of ROs and how to design them for FPGAs. In Section 3.2, frequency measurements based on asynchronous counters are discussed. In Section 3.3, an analysis on spatial RO frequency distribution is shown and a design strategy for reliable RO PUFs is proposed. Section 3.4 summarizes this chapter.

3.1 Ring Oscillators

A ring of an odd number of inverters is one of the most basic oscillating circuits in digital IC design. These oscillators can be implemented in ASICs as well as in FPGAs and are a popular solution for PUFs, where their dependency on manufacturing variations is exploited. The reason for that is that these structures, compared to other PUF architectures, exhibit a relatively low-biased statistical distribution [KKR+12]. Another application for ROs in secure embedded devices are True Random Number Generators (TRNGs) [FMC85], where their random jitter is extracted.

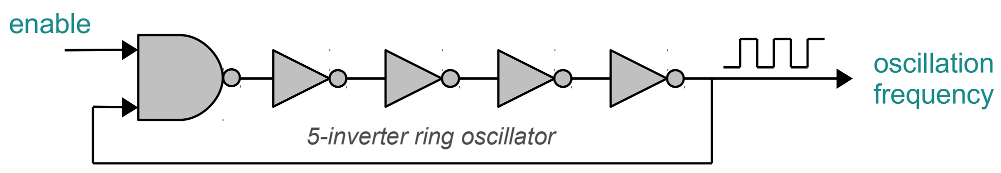


Figure 3.1: Architecture of a 5-inverter RO

3.1.1 Functionality

An RO consists of an odd number of inverters and, therefore, has no stable digital state, which is the reason why it starts oscillating after powering. Its oscillation frequency depends on the sum of propagation delays of all inverters. Since the propagation delay of each inverter depends on its unique physical characteristics, RO operation can be seen as a repetitive measurement of propagation delays. Therefore, ROs are a suitable measure to extract unique characteristics introduced during IC manufacturing.

RO implementations for PUFs and TRNGs are usually equipped with an enable signal integrated by a NAND gate in order to save power. Figure 3.1 shows the architecture of a 5-inverter RO with an enable input. Note that there are only four dedicated inverters in this implementation and the fifth inverter is integrated into the enable NAND gate.

3.1.2 Influences on RO Frequency

RO PUFs build on the uniqueness and stability of individual RO frequencies. These frequencies are subject to a set of influences, which have to be considered during RO PUF design.

- **Implementation details.** The number of inverters in an RO, the technology-specific propagation delay of each inverter, and the routing of the RO-internal signals determine the nominal frequency of an RO. In order to avoid measurement problems, this mean frequency should be designed to be lower than the maximum operating frequency of the used counter flip-flops.

- **Manufacturing variations.** The deviation of inverter delays, arising from sub-micron physical variations, occurring during manufacturing, are responsible for the uniqueness of every RO PUF. The standard deviation of RO frequencies among the population of ROs is determined by these influences. A wide distribution of RO frequencies together with a low individual RO frequency noise suggests a good quality of RO comparisons, because ROs can be differentiated better as with a narrow distribution and the same individual noise.

- **Thermal noise.** Thermal noise occurs at every transistor, therefore, also inverter delays involve this ever-present noise. This leads to a certain amount of inevitable noise contained in RO frequencies, even under stable environmental conditions.
- **Temperature variations.** Inverters in ROs are sensitive to temperature changes. Therefore, with increasing temperature, RO frequencies decrease and vice versa. This effect increases the RO frequency noise over the application temperature range. As a result, the reliability of RO PUF measurements is decreased. This effect can be partly compensated by comparing two ROs that are both subject to similar temperature influences.
- **Supply voltage variations.** Variations in the supply voltage of RO circuits also influence their behavior and lower the reliability of RO PUFs.
- **Frequency locking.** ROs are also prone to frequency locking, i.e., they can synchronize with other internal or external oscillating sources. This can be a point of attack, but it can also cause issues with other oscillating circuits implemented in the same device.

3.1.3 FPGA Implementation

Usually, digital hardware designs are developed using Hardware Description Languages (HDLs) like VHDL or Verilog. Also, the functional behavior of an RO can be described in such a language. However, one faces some difficulties when designing physical primitives like ROs:

- **Unwanted optimization.** The FPGA synthesis tool tries to optimize all described circuits and reduces an RO until it is removed completely. Of course, this is unacceptable for RO PUFs, which want to extract unique device features by measuring ROs.
- **Random placement.** Whenever changes are made to the design, the placement of the RO components can change, i.e., the absolute location on the FPGA floorplan can be altered as well as the routing distances between its inverters. This happens because the FPGA synthesis tool

searches for optimized placement and routing, e.g., regarding most efficient resource usage or shortest signal paths, which might lead to different results, if the design is changed. This is critical for RO PUFs, because they require identical ROs that have the same physical structure and nominal oscillation frequency.

The problem of unwanted optimization can be handled by configuring the synthesis tool to ignore ROs during optimization. For VHDL, this can be done by attaching the `SAVE` attribute to all RO internal signal lines. This is independent of the target device and ensures correctly synthesized ROs.

Exact placement and routing, however, always depend on the used devices and tools. In this thesis, I use Xilinx Spartan-3 [Xil12a], Spartan-3A [Xil10] and Spartan-3E [Xil12b] devices for the implementations. Xilinx FPGA devices are internally organized in Configurable Logic Blocks (CLBs). For the Spartan-3, Spartan-3A, and Spartan-3E families, each CLB groups four slices. Each slice contains two Look-Up Tables (LUTs), two storage elements, which can be used as flip-flops or latches, and miscellaneous supporting circuits.

For Xilinx FPGAs, it is possible to develop hard macros to overcome the problems explained above. The Xilinx FPGA Editor tool can be used to develop templates, called hard macros, of basic building blocks like ROs, which can then be instantiated in any HDL. In this application, the LUTs and flip-flops of slices can be configured manually and, also, the inputs and outputs of slices can be manually connected leading to specific manual routing within the hard macro. If this hard macro is instantiated, it ensures that every instance of this block has the same slice configuration and the same block-internal routing. In the case of RO PUFs, this is sufficient, because it enables to implement ROs that have identical FPGA configuration properties, although hard macros and the FPGA Editor tool do not provide exact internal FPGA layout information. To complete the implementation, it is necessary to define placement constraints for each RO hard macro instance to ensure adjacent placement of ROs in the FPGA's floorplan. For Xilinx, these constraints are written into the User Constraints File (UCF) of a project. Thereby, consistent absolute and relative placement as well as routing of RO-internal components and signals is achievable.

However, developing hard macros for RO PUFs with the FPGA Editor has some drawbacks. It can be quite inconvenient to initially design a hard macro or to change its details at a later stage. Also, porting the hard macro to a new device family usually requires a new hard macro design. An alternative to using hard macros is the extension of the RO placement constraints, which are also necessary in the hard macro case. It is possible to add the generation of detailed placement constraints for each component of an RO

FPGA implementation. This allows for more flexibility because handling text-based constraint files is easier than working with the FPGA Editor and can be adapted faster to new designs and devices. Therefore, for some of the experiments in this thesis, I used scripts that output constraints for every single LUT used in an RO. Thereby, identical placement of ROs can be achieved and the same routing will be generated implicitly because the connections of input and output ports of all used LUTs are defined. And since placement constraints have to be defined anyway, this does not represent large additional effort.

3.2 Frequency Measurement

In this section, important considerations about the counters used for frequency measurements in RO PUFs are detailed.

3.2.1 Asynchronous Counter

ROs oscillate with a unique frequency depending on the propagation delays of their gates. In order to extract this RO-specific property, RO PUFs implement counters incrementing on the positive edge of an RO's output. Synchronous counters are not suitable in this case, because these counters can produce incorrect counter values if controlled by a varying RO output signal instead of a stable and uniformly routed clock signal. Therefore, one has to use asynchronous counters to allow for reliable operation even when controlled by jittery RO oscillation signals.

In Figure 3.2, the architecture of an asynchronous counter is shown as used for RO PUFs. It consists of a series of toggle flip-flops, where the first one is clocked by an RO output signal. Each flip-flop output represents one counter bit, which is inverted and then connected to the next flip-flop's clock input to toggle the following flip-flop, if the previous one signalizes a 1-bit overflow by a negative edge.

3.2.2 Measurement Error

The runtime during which an RO frequency is measured has a significant influence on the measurement error, i.e., the measurement tolerance, caused by the counter implementation and, therefore, on the quality of an RO PUF. A first requirement is that the RO runtime is balanced between reliable RO discrimination and fast response extraction. Further, counter sizes have to be chosen wide enough to avoid overflows for ROs with high frequencies.

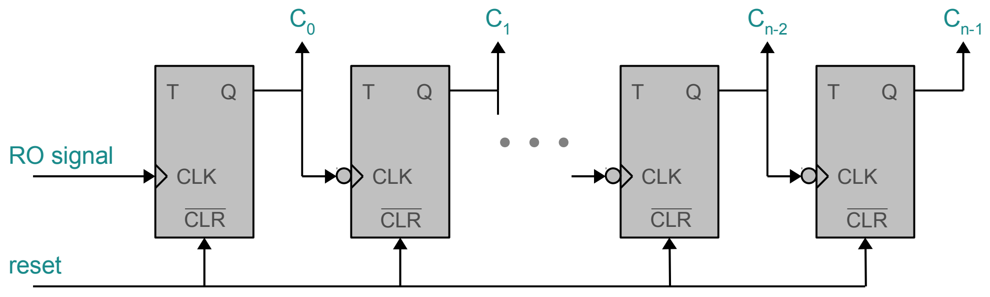


Figure 3.2: Asynchronous counter architecture

The maximum absolute error $e_{abs,max}$ resulting from a given RO measurement runtime t_{run} can be calculated as follows:

$$e_{abs,max} = \pm \frac{1}{2 \cdot t_{run}} \tag{3.1}$$

Taking the mean RO frequency value f_{mean} into account, the maximum relative error $e_{rel,max}$ can be determined as:

$$e_{rel,max} = \pm \frac{1}{2 \cdot t_{run} \cdot f_{\text{mean}}} \tag{3.2}$$

For instance, if a maximum relative error of $e_{rel,max} = \pm 0.1\,\% = 0.001$ is required for frequency measurements of ROs with a nominal mean frequency of $f_{\text{mean}} = 100\,\text{MHz}$, the minimal runtime for each RO measurement can be calculated as:

$$t_{run} = \frac{1}{2 \cdot 0.001 \cdot 100\,\text{MHz}} = 5\mu s \tag{3.3}$$

This defines the lower bound on RO comparison runtime, while the upper bound of measurement time is determined by application requirements.

However, one should also notice that other factors, such as enable signal propagation delay deviations, temperature variations, and supply voltage deviations, can cause further measurement errors.

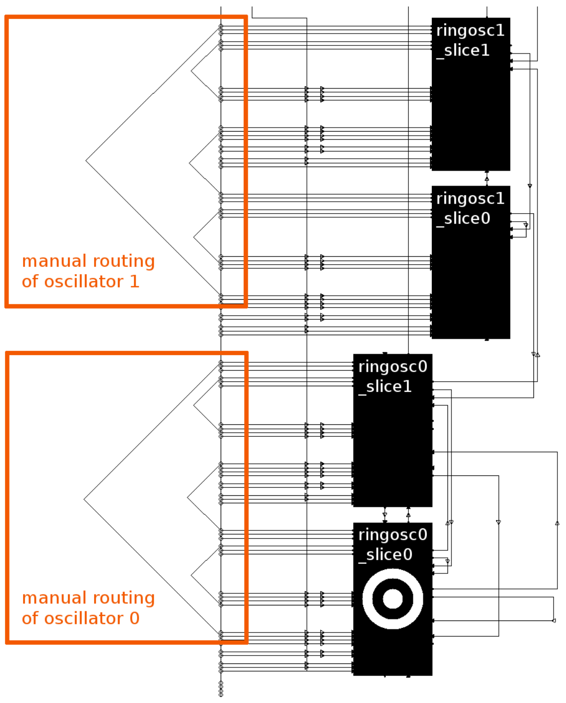


Figure 3.3: Hard macro of one CLB containing two ROs

3.3 RO Placement and Comparison

This section deals with the questions, where to preferably place ROs on an FPGA floorplan and which ROs should be compared to obtain unique and reliable RO PUF responses.

3.3.1 Spatial Ring Oscillator Frequency Analysis

Maiti and Schaumont [MS09] mentioned that RO frequencies depend on their location on an FPGA die, e.g., frequencies of ROs at the edges of an FPGA are slower than central ROs. In the following, I show that also the placement of RO measurement logic has an influence on their frequencies.

In order to analyze the spatial dependency of RO frequencies, I instantiated an array of 2712 ROs. I used the hard macro shown in Figure 3.3 to implement two ROs per CLB and covered more than half of a Xilinx Spartan-3E XC3S1200E die with ROs. Additionally, an RS232 interface and measurement counters were implemented. I synthesized four different versions with different placement of RO measurement logic.

Figure 3.4 shows the different areas of measurement logic and the spatial distribution of frequencies over the FPGA floorplan. The presented results show that the position of interfaces and read-out logic significantly influences intra-die conditions and RO frequencies.

My analysis results show that RO frequencies increase with increasing distance to the communication and read-out logic for every placement case. A clear spatial dependency of RO frequencies is shown. Possible explanations for this effect can be a local temperature increase at the continuously running measurement circuits, which might lower surrounding RO frequencies, or influences arising from increased routing length of RO output signals.

Second, a stripe pattern is visible over the entire area. This is due to the fact that the hard macro shown in Figure 3.3 consists of two oscillators, which are displayed in two different columns in the FPGA floorplan view. Alternating, one column consists of only RO_0 instances, while the next one has only RO_1 implementations. The stripe pattern is visible because these two RO implementations, although looking nearly identical in the hard macro view, still show a noticeable mean frequency difference of approximately 10 MHz, which is almost a third of the whole range of oscillator frequencies from 185 to 218 MHz.

When looking closer, the locations at the bottom borders of Figures 3.4(a) to 3.4(d) show lower instead of further increasing frequencies. This might result from higher energy densities at the die borders because of nearby output buffer locations, local heat accumulation or different physical implementation.

Comparing Figures 3.4(c) and 3.4(d), one will find that moving the logic farther away only shows little improvement. So, the idea of placing dummy cells between ROs and logic, as proposed in [SHO08], is not applicable here.

I draw two conclusions from this experiment. On the one hand, only ROs within the same spatial region, i.e., neighborhood on the die, can be compared to each other because RO frequencies exhibit a strong spatial dependency. On the other hand, even if RO implementations look quite similar in the FPGA Editor tool, as it is the case for RO_0 and RO_1 in Figure 3.3, they cannot be assumed to have the same nominal frequency. Therefore, only identical instances of ROs placed and routed in a completely equivalent way within a slice should be compared to avoid predetermined comparison results. Unfortunately, the physical structure of different slices is usually not published by FPGA manufacturers, which complicates identical RO implementation.

3.3.2 Physical Mapping and Comparison Strategy

Usually, RO PUFs extract $n-1$ bits from n ROs [MS09] by sequentially comparing overlapping RO pairs and it was shown that the quality of RO PUFs

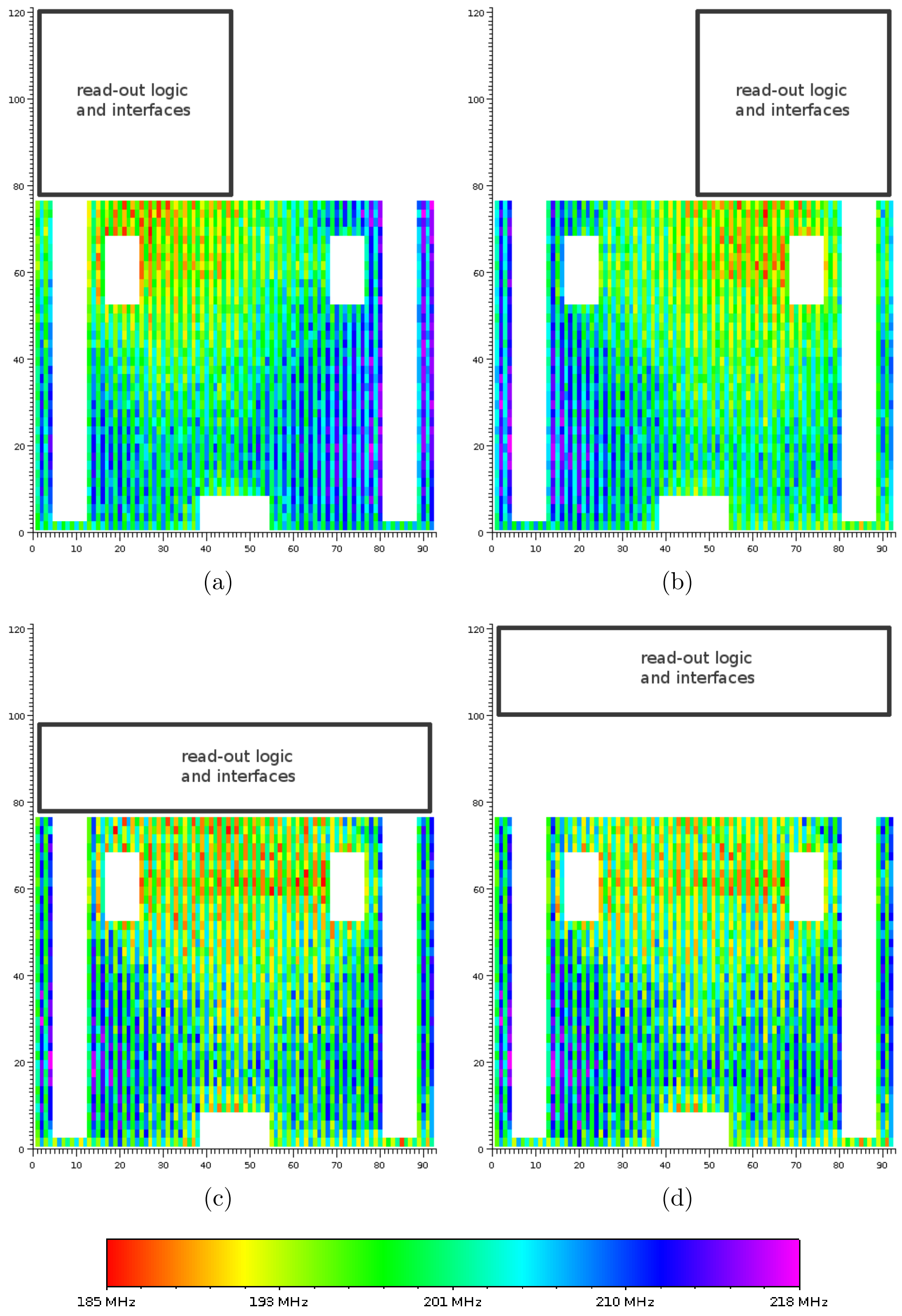
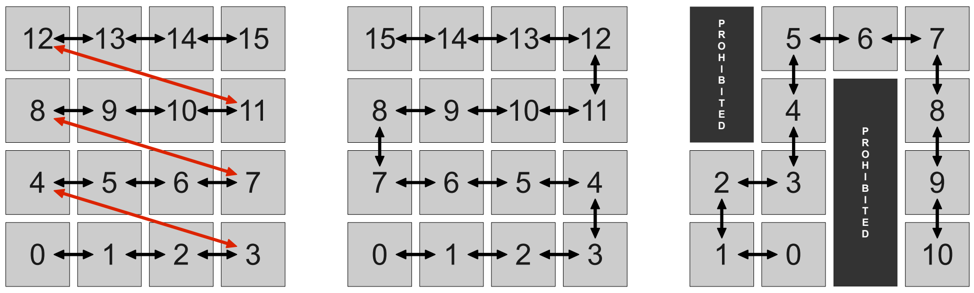


Figure 3.4: Spatial RO frequency distribution on a Xilinx Spartan-3E



(a) (b) (c)

Figure 3.5: Spatial RO mapping strategies

increases with controlled RO placement [MS09]. However, even when placed with location constraints, as shown in Figure 3.5(a), problems can occur, when comparing the last RO of the first row with the first RO of the second row. Therefore, I suggest to use a chain-like placement approach, as shown in Figure 3.5(b), to enable comparisons of only adjacent ROs because only closely located ROs are exposed to similar intra-die conditions.

Note that this approach allows to select one RO pair after the other with a simple counter feeding addresses to the RO multiplexer. Only the different physical placement is responsible for increased RO PUF quality. Even in cases with hard area constraints, where ROs have to be place in unshaped structures, the chaining method can be applied, as depicted in Figure 3.5(c).

For Xilinx FPGAs, this placement method can also be combined with the anyhow important placement constraints when implementing ROs on FPGAs. This guarantees an enhanced selection of RO pairs.

3.4 Summary

In this chapter, I showed that the design of ROs on FPGAs involves some obstacles, such as automatic optimization and random placement. However, these problems can be solved by HDL constructions, hard macros, and placement constraints. Further, I explained the necessity of using asynchronous counters for RO frequency measurement purposes and described the incurred measurement error. The last part of this chapter dealt with the spatial placement of ROs on an FPGA's floorplan, which is significant for RO PUF quality. For a flexible and powerful design flow, I suggested to implement a script-based constraint generator, which is able to handle both identical RO implementation and all related placement issues.

Chapter 4

Attacks on RO PUFs

RO PUFs are usually implemented to measure unique device features that can be used for cryptographic key generation. They represent a kind of secure memory and, therefore, also have to face physical attacks that try to extract their secret information. In contrast to classical memories, which have a quite short read-out time, RO PUFs measure ROs running for a relatively long time and, thereby, emitting EM signals correlated with their unique frequencies. Therefore, I chose to evaluate attacks on RO PUFs based on EM measurements.

In this chapter, I propose two EM attacks on RO PUFs. The first one exploits EM emissions of a standard RO PUF implementation. There, vulnerabilities arise from overlapping RO comparisons. I also propose a countermeasure to overcome this problem by avoiding overlapping comparisons. Further, I show how efficiency can be maintained for protected RO PUF architectures. The second attack exploits localized EM leakage, which helps an attacker to separate an RO PUF's measurement components. Thereby, it is possible to extract the secret RO frequencies for each RO measurement run, even for implementations protected against the first attack. I propose randomized measurement chains and interleaved placement as countermeasures against this attack. Randomizing the usage of measurement components breaks the relation between RO and measurement circuits, while interleaved circuit placement provides practical protection because of EM emission overlap and does not incur any overhead. Additionally, this chapter provides explanations of the used EM measurement setup, discusses the impact of FPGA decapsulation on RO PUFs, and shows in a pre-analysis that even a single RO can be located by state-of-the-art EM measurement equipment. Parts of this chapter have been published in the paper *Semi-invasive EM Attack on FPGA RO PUFs and Countermeasures* at the 6th Workshop on Embedded Systems Security in 2011 [MSSS11a] and the contribution *Localized Electro-*

magnetic Analysis of RO PUFs at the IEEE International Symposium on Hardware-Oriented Security and Trust in 2013 [MHH$^+$13].

This chapter is organized as follows. A brief overview of related work is given in Section 4.1. The impact of decapsulating an FPGA on its RO PUF is analyzed in Section 4.2. Section 4.3 explains the measurement setup I used, while Section 4.4 shows the practical feasibility to locate a tiny 3-inverter RO by EM analysis. In Section 4.5, the first attack exploiting RO overlap is shown and a countermeasure is proposed. The second attack based on localized EM measurements is demonstrated in Section 4.6. Section 4.7 summarizes this chapter.

4.1 Related Work

Until now, there is no related work on RO PUF secret extraction based on EM emissions besides my contributions. In the following, I will list works that adjoin and contributed to EM analysis of RO PUFs.

In 2001, Gandolfi et al. [GMO01] and Quiscater and Samyde [QS01] used the electromagnetic emission of a cryptographic device to break it. They performed a side-channel attack based on algorithmic leakage present in EM emissions. In contrast to their analysis, I exploit the leakage of physical, not algorithmic, properties that reveal an RO PUF's secret.

A first paper about using localized EM emission for chip cartography was presented by Sauvage et al. [SGM09]. They demonstrated that an FPGA exhibits different EM emissions at different die locations (mostly at the edges of the die), depending on the internally programmed design. In my attacks, I directly exploit separable EM emissions coming from different die areas corresponding to different RO PUF components.

Heyszl et al. [HMH$^+$12a, HMH$^+$12b] showed that high-resolution probes enable localized acquisition of EM emission traces over a grid of locations on the surface of a die. Each trace will capture the emissions of parts of an integrated circuit close to the measurement location. They further showed that this can be exploited for attacks on cryptographic modules. Building on these results, I use localized EM analysis for high-resolution frequency emission cartography, which allows to spatially separate components of RO PUFs.

Rührmair et al. [RSS$^+$10] demonstrated that RO PUFs can be modeled by machine learning algorithms, if an attacker is able to collect a sufficient number of CRPs. They also remark that a full read-out of RO PUFs is possible. However, these attacks become infeasible for key generation scenarios, where no public challenge-response interface is available to an attacker. Therefore, they are not relevant or comparable to the attacks I propose.

Besides extracting information from the EM emission of RO PUFs, another way to attack them is the injection of EM signals as shown by Bayon et al. [BBA+12]. There, RO frequency locking can be achieved by externally injecting frequencies. The locking effect synchronizes all ROs and, thereby, cancels the uniqueness of the RO signals. However, although these attacks can disturb the correct operation of a PUF, they do not allow the extraction of information about the original RO behavior.

4.2 Impact of RO PUF Decapsulation

A popular (marketing) argument for using PUFs instead of conventional key storage like NVM is their natural tamper resistance property [GKST07, GŠT+09, SVW10]. This assumption holds true if an attacker aims at manipulating the PUF structure itself, as for any other memory, but does not automatically protect a whole microchip and its package from tampering, at least in the case of silicon PUFs.

In this section, I show that decapsulating a Xilinx Spartan-3 XC3S200 FPGA in a VQ100 package from the backside does not have a significant influence on internal RO frequencies. As a result, attacks like on-die EM cartography are valid for RO PUFs and probably many other silicon PUFs not involving package properties into their measurements.

4.2.1 FPGA Decapsulation

For the following analysis, I followed the suggestion of Skorobogatov [Sko10] to open an FPGA package from the backside, which can be done manually and without the application of acids. First, the FPGA was soldered on a printed circuit board having a hole in the center allowing to access the backside of the package, as shown in 4.1(a). Then, a drill was used to remove the plastic packaging down to the copper lead-frame of the package, as shown in Figure 4.1(b). After cutting the corner connections of the ground-plate with a carpet cutter, one can take away the plate, exposing the backside of the die. Finally, the remaining glue was scraped off and die's surface was cleaned with acetone. Figure 4.1(c) shows the final result.

For later experiments, I also used FPGAs that were decapsulated from the frontside by fuming nitric acid, because one can achieve an even higher Signal-to-Noise Ratio (SNR) there [HMH+12b].

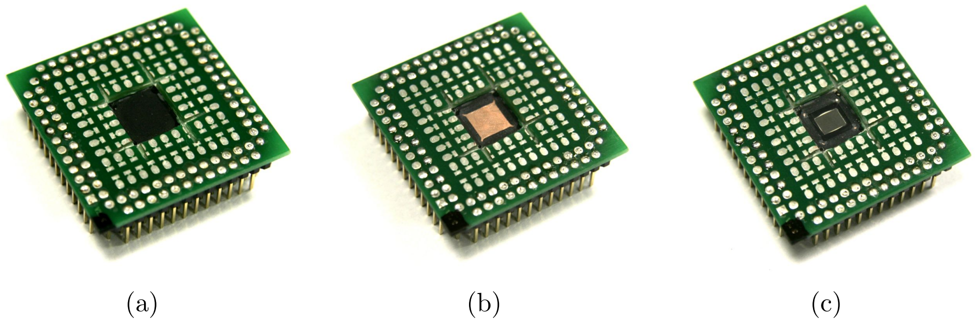


Figure 4.1: Step-by-step backside decapsulation of an FPGA

4.2.2 Analysis Before and After Decapsulation

In order to investigate the impact of chip decapsulation on FPGA RO PUFs, I designed an array of 256 oscillators on a Xilinx Spartan-3 XC3S200 FPGA occupying 256 CLBs. Additionally, a multiplexer, a 16-bit counter, and control logic were integrated to enable frequency measurements for all oscillators. Each oscillator was measured 10000 times over a period of 4096 clock cycles at 20 MHz in a temperature-controlled room to precisely characterize the given RO frequencies. The resulting maximum absolute measurement error with the measurement runtime $t_{run} = 4096 \cdot \frac{1}{20MHz} = 204.8\mu s$ can be calculated by Equation 3.1 to be ± 2441Hz.

Measurements were conducted for ROs consisting of three, five, and seven inverters before and after decapsulation. To analyze the differences before and after decapsulation, I first generated frequency maps, as shown in Figure 4.2 for the 7-inverter case, to visually compare the frequency fingerprint. Then, I also calculated the mean value f_{mean} of all RO frequencies to investigate if there was a shift in the mean frequency. Further, to simulate an RO PUF, I determined the 255-bit PUF response vector $\mathbf{r}_i, i = 1, ..., 10000$ by sequential relative comparison of all 256 measured frequencies of my implementation. From these responses, the intra-device Hamming distance, a measure for the present noise, can be calculated, as explained in Equation 2.1. Also, the mean PUF response $\mathbf{r}_{mean}$ can be determined:

$$\mathbf{r}_{mean} = \frac{1}{10000} \sum_{i=1}^{10000} \mathbf{r}_i \tag{4.1}$$

Figure 4.2 shows the frequency maps, i.e., the physical fingerprint, for the 7-inverter RO test design before and after decapsulating the FPGA. By visual inspection, no significant difference can be observed.

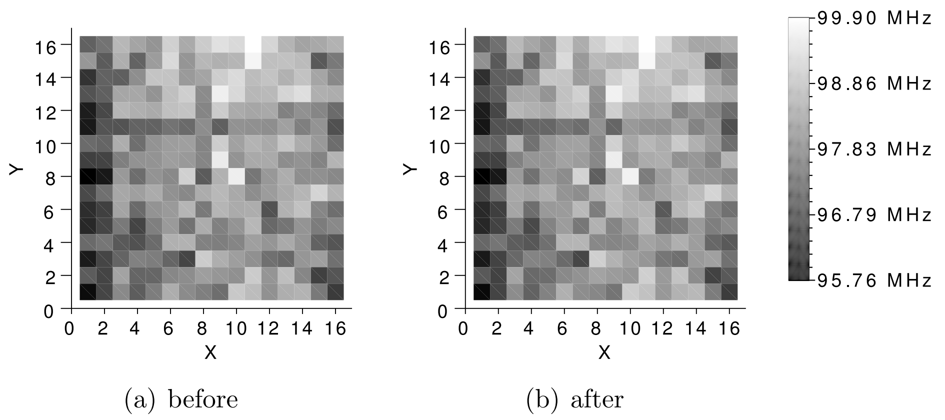


Figure 4.2: RO frequency maps for ROs with seven inverters on a Xilinx Spartan-3 XC3S200 FPGA

Table 4.1 shows the RO mean frequencies of all analyzed RO types before and after decapsulation. They indicate that opening the chip package shifts the mean frequency slightly higher. A reason for that might be lesser heat accumulation without package.

Table 4.1: Mean RO frequency before/after decapsulation

RO Type	$f_{mean,before}$	$f_{mean,after}$	$f_{mean,after} - f_{mean,before}$
3-inverter	197.96 MHz	198.07 MHz	+0.11 MHz
5-inverter	132.56 MHz	132.70 MHz	+0.14 MHz
7-inverter	98.10 MHz	98.20 MHz	+0.10 MHz

Analyzing the Hamming distances between the mean RO PUF responses $\mathbf{r}_{mean,before}$ and $\mathbf{r}_{mean,after}$, I found that the influence of decapsulation alters less than 3% of RO PUF output bits, as shown in Table 4.2. In the perfect tamper resistance case, tampering with the device should lead to a change in 50% of all response bits. Additionally, the noise contained in the RO PUF measurements of the original device causes a maximum intra-device Hamming distance $\max(HD_{intra,before})$ that is significantly higher than the change caused by the decapsulation. This noise and even additional noise from temperature and voltage variations has to be handled by key generation algorithms in order to enable reliable key extraction anyway. This means, the smaller response deviation introduced by opening the FPGA package will also be corrected by the key reconstruction algorithms, leading to no alteration of the generated key. Therefore, I conclude that decapsulation does not lead to significant changes in RO PUF behavior and does not hinder an attacker from removing an FPGAs package to mount attacks like on-die EM emission analysis.

Table 4.2: Decapsulation effect on RO PUF responses

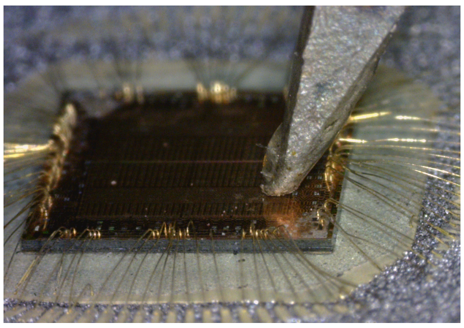
RO Type	$\max(HD_{intra,before})$	$HD(\mathbf{r}_{mean,before}, \mathbf{r}_{mean,after})$
3-inverter	11.81%	0.79%
5-inverter	7.87%	2.76%
7-inverter	5.12%	2.36%

4.3 Localized Electromagnetic Analysis

In this section, I explain the measurement setup I used for localized EM analysis over multiple locations of an FPGA die. In contrast to global EM measurements at a fixed measurement point with a coarse EM probe, an array of local measurement points enables a much more detailed analysis of a device's EM emission. For the first EM attack shown later in this thesis, the setup significantly enhances the SNR over an attack at a single measurement point without package removal. For the second attack, the specific localized leakage of RO PUFs is exploited, i.e., the attack requires to separate leakage from different RO PUF components, which is not possible with a single-point measurement. This section also gives an overview of the basic analysis measures I used for the following attacks.

4.3.1 Measurement Setup

The basic measurement equipment for the two following attacks is the same. I used a Langer ICR HH 150-6 magnetic field probe, with a 150 µm shielded horizontal coil, 6 windings, 100 µm resolution, and a 30 dB amplifier to capture the EM emission close to the die surface. Further, I used an additional Langer PA 303 30 dB amplifier and a LeCroy WavePro 715Zi oscilloscope to record measurement traces. In order to automate the position movement and the trace acquisition, I wrote a script to control the oscilloscope and an x-y-table accordingly. The distance between the EM probe and the die surface was approx. 50 µm to prevent damaging die or probe. Figure 4.3 shows two pictures of the EM probe being close to the FPGA die surface as during my measurements.


(a)

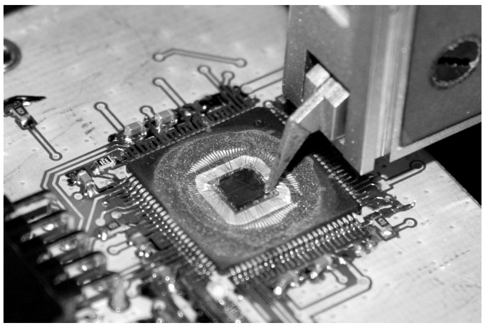

(b)

Figure 4.3: EM probe on FPGA die surface

4.3.2 Analysis Preliminaries

This section explains some basic preliminaries for the subsequent attacks.

Grid of Traces

In the following analyses, I always collect traces $\mathbf{t}_{(x,y)}$, where x and y denote coordinates of locations (x, y) in the ranges $1, ..., X$ and $1, ..., Y$ over an FPGA die location set $\mathbf{A}_{die}$. Each trace $\mathbf{t}_{(x,y)}$ consists of T samples $t_{(x,y),k}, k = 1, ..., T$.

$$\mathbf{A}_{die} = \{(x, y) \mid x = 1, ..., X; y = 1, ..., Y\} \tag{4.2}$$
$$\mathbf{t}_{(x,y)} = \{t_{(x,y),k} \mid (x, y) \in \mathbf{A}_{die}; k = 1, ..., T\} \tag{4.3}$$

The x-y-table I used automatically moved the mounted EM probe step by step through all required locations, while a script provided input data to the device under test, which then triggered the oscilloscope to record an EM trace.

Fast Fourier Transform

The secret of RO PUFs lies in their unique RO frequencies. Therefore, the attacks I present in the following sections mainly operate in the frequency domain instead of the time domain. In order to transform time series, such as EM traces, into the frequency domain, I utilize the Fast Fourier Transform (FFT) [Smi97] with a rectangular window. It allows fast decomposition of traces into frequency components, namely amplitude and phase of each frequency, which then can be visualized, e.g., as a frequency amplitude spectrum. I denote the Fourier transform $\mathcal{F}$ of a trace $\mathbf{t}_{(x,y)}$ into a complex FFT vector $\mathbf{g}_{(x,y)}$ by:

$$\mathbf{g}_{(x,y)} = \mathcal{F}(\mathbf{t}_{(x,y)}) \tag{4.4}$$
$$\mathbf{g}_{(x,y)} = \{g_{(x,y),k} \mid k = 1, ..., T\} \tag{4.5}$$

The amplitude values of this complex vector, i.e., the absolute magnitudes, are denoted as $|g_{(x,y),k}|$ for every frequency bin k. The phase value is not relevant for my analyses.

EM Maps

To visualize local properties of EM emission, I use a $X \times Y$ pixel map $\mathbf{M}$, where each map pixel $m_{(x,y)}$ was generated from the according FFT vector $\mathbf{g}_{(x,y)}$ by a function f that depends on the specific attack step.

$$\mathbf{M} = \{m_{(x,y)} \mid x = 1, ..., X; y = 1, ..., Y\} \tag{4.6}$$
$$m_{(x,y)} = f(\mathbf{g}_{(x,y)})\ \forall\ x = 1, ..., X; y = 1, ..., Y \tag{4.7}$$

Spectrum Filtering

I use the term filtering to describe the procedure of considering only a part of a frequency spectrum. I do not apply digital filter techniques, but just restrict the analysis from a T-bin FFT vector $\mathbf{g}_{(x,y)}$ to a vector $\mathbf{g}^*_{(x,y)}$ with T^* frequency bins within a frequency range from f_{lo} to f_{hi}. This filtering is denoted as:

$$\mathbf{g}^*_{(x,y)} = \mathit{filter}(\mathbf{g}_{(x,y)}, f_{lo}, f_{hi}) \tag{4.8}$$

Clustering of Data

In the second attack, where I exploit local differences of EM emissions, I aim at separating RO PUF components. Therefore, a clustering algorithm, such as the k-means [DHS01] algorithm, is required. These algorithms partition a set of n values into q clusters. I denote the clustering of values $\{x_1, ..., x_n\}$ into q clusters $Q_1, ..., Q_q$ as:

$$\{Q_1, ..., Q_q\} = \mathit{cluster}(\{x_1, ..., x_n\}) \tag{4.9}$$

4.4 EM Emission of Ring Oscillators

In this section, I answer the question whether it is feasible to identify the frequency and location of a tiny oscillating circuit, such as a single 3-inverter RO, by measuring its weak EM emission with the above mentioned state-of-the-art equipment. Also, I explain how to identify ROs in a frequency amplitude spectrum. This is a preliminary experiment to understand the characteristics of ROs' EM emission, which is implicitly required by the following attacks.

4.4.1 Test Design

For this analysis, I implemented a test design with six manually placed ROs on a Xilinx Spartan XC3S200 FPGA device, as shown in Figure 4.4. RO_1 and RO_2 consist of seven inverters, RO_3 and RO_4 of five inverters, and RO_5 and RO_6 of only three inverters. The right column of ROs was implemented in one CLB, using only local routing. In order to analyze the influences of global routing, the ROs in the left column used four, three and two CLBs, respectively. All oscillators were running simultaneously for 2048 cycles. Note that the components on the very left of the FPGA floorplan are control logic and have only negligible influence on the ROs.

4.4.2 Frequency Spectra

I recorded $54 \times 54 = 2916$ traces with 512000 samples each over the whole FPGA die and transformed them into the frequency domain by FFT.

$$\mathbf{A}_{die} = \{(x, y) \mid x = 1, ..., 54; y = 1, ..., 54\} \tag{4.10}$$

$$\mathbf{g}_{(x,y)} = \mathcal{F}(\mathbf{t}_{(x,y)}) \; \forall \; (x, y) \in \mathbf{A}_{die} \tag{4.11}$$

$$\mathbf{g}_{(x,y)} = \{g_{(x,y),k} \mid k = 1, ..., 512000\} \tag{4.12}$$

Then, I calculated the standard deviation spectrum $\mathbf{S}_{sd}$ of all amplitude spectra over all locations, yielding high values at frequencies which show a strong spatial dependency, i.e., are present at some locations and missing at others.

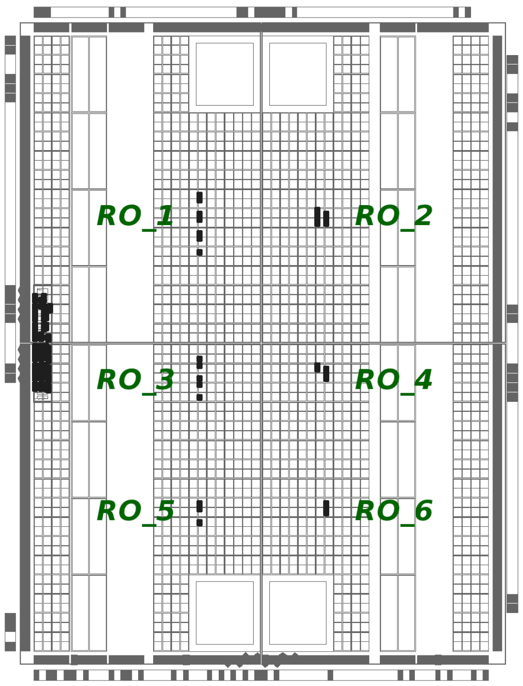


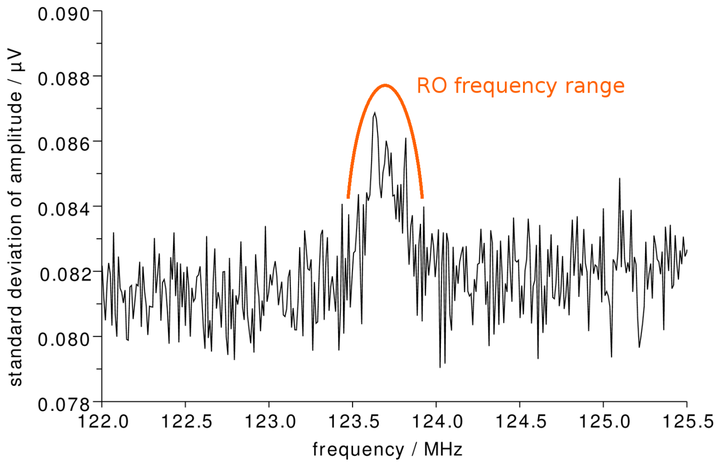
Figure 4.4: Floorplan of RO emission test design on Xilinx Spartan-3 XC3S200 FPGA

$$\mathbf{S}_{sd} = \{s_{sd,k} \mid k = 1, ..., 512000\} \tag{4.13}$$

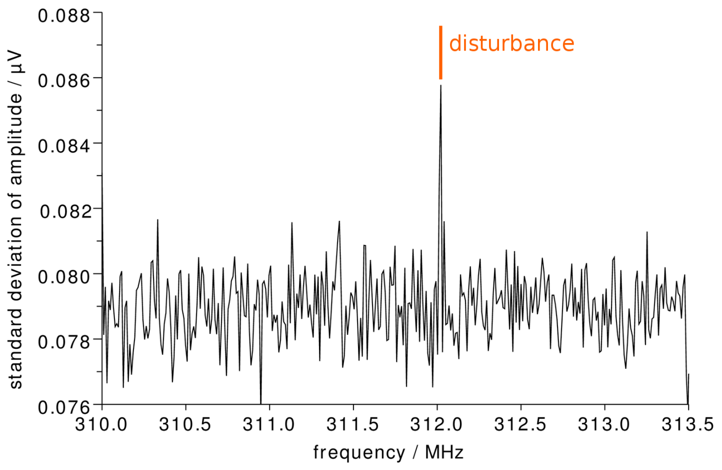
$$\mu_k = \frac{1}{2916} \sum_{x=1}^{54} \sum_{y=1}^{54} g_{(x,y),k} \ \forall \ k = 1, ..., 512000 \tag{4.14}$$

$$s_{sd,k} = \sqrt{\frac{1}{2916} \sum_{x=1}^{54} \sum_{y=1}^{54} (g_{(x,y),k} - \mu_k)^2} \ \forall \ k = 1, ..., 512000 \tag{4.15}$$

I used $\mathbf{S}_{sd}$ to identify the implemented ROs without prior knowledge about their frequencies. I found that RO emission spectra, e.g., as shown in Figure 4.5(a), are quite different from frequency spectra originating from disturbances, as shwon in Figure 4.5(b). These disturbances are introduced by the control circuits or by the measurement environment. The RO spectra show a wider frequency range, because their frequencies vary over time and measurement location. Therefore, they are represented as a “hill” in the spectrum. Disturbances often show a spike, which corresponds to the specific frequency on which the disturbance source is operating, e.g., in wireless network applications.



(a)



(b)

Figure 4.5: Exemplary amplitude standard deviation spectra of (a) a ring oscillator and (b) a disturbance frequency

Table 4.3: Identified RO frequencies

RO	Type	Freq. Range / MHz	Mean Freq. / MHz	Variation / MHz
1	7 inverters in 4 CLBs	123.55 – 123.85	123.700	±0.150
2	7 inverters in 1 CLB	142.95 – 143.10	143.025	±0.075
3	5 inverters in 3 CLBs	177.75 – 178.05	177.900	±0.150
4	5 inverters in 1 CLB	202.80 – 202.95	202.875	±0.075
5	3 inverters in 2 CLBs	298.95 – 299.25	299.100	±0.150
6	3 inverters in 1 CLB	338.80 – 339.00	338.900	±0.100

I was able to find all six ROs and identify their frequency characteristics by looking for characteristic RO signatures, as shown in Figure 4.5(a), in the spectrum $\mathbf{S}_{sd}$. Table 4.3 shows that larger ROs (more inverters, longer routing) exhibit lower frequencies, as expected. It also shows that ROs implemented in one CLB show a lower variation, i.e., occupy a narrower frequency range.

4.4.3 RO Localization

In order to locate the position of the implemented ROs, I filtered the previously obtained FFT result vectors $\mathbf{g}_{(x,y)}$, i.e., I reduced the $T = 512000$ frequency bins to T^* bins within the range of 338.80 – 339.00 MHz to select RO$_6$. Then, I generated an EM map where each pixel $m_{(x,y)}$ represents the average amplitude in the filtered spectrum at the corresponding location.

$$\mathbf{g}^*_{(x,y)} = \mathit{filter}(\mathbf{g}_{(x,y)}, 338.8, 339.0)\ \forall\ (x,y) \in \mathbf{A}_{die} \tag{4.16}$$

$$m_{(x,y)} = \frac{1}{T^*} \sum_{k=1}^{T^*} g^*_{(x,y),k}\ \forall\ (x,y) \in \mathbf{A}_{die} \tag{4.17}$$

In Figure 4.6, the local emission of RO$_6$ can clearly be identified at the lower right position, as expected from the floorplan in Figure 4.4.

Therefore, my measurements show that the localization and measurement of a single 3-inverter RO is possible. However, it has to be noted that, although measured very close to the surface of the FPGA die, even the EM emission of the smallest RO extends to a circle with a diameter of approx. 500 µm, which suggests that it is very hard to distinguish closely placed ROs. In other words, the results support interleaved placement as a method to disguise RO EM emission, because separation of ROs or other emitting components placed in close proximity becomes unlikely.

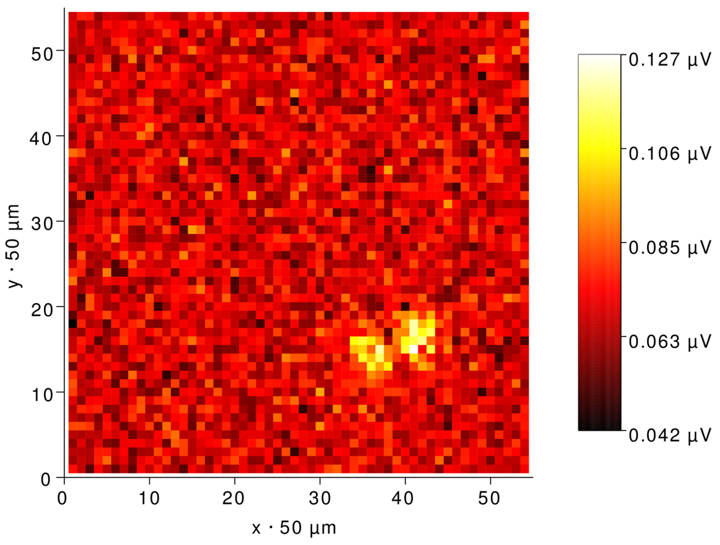


Figure 4.6: EM map revealing the position of RO_6

4.5 Attack on Standard RO PUFs

This first attack focuses on the extraction and mapping of RO frequencies based on EM measurements performed on a standard RO PUF implementation. This enables an attacker to fully characterize a given RO PUF. Compared to the second attack, no separation of RO PUF components takes place, i.e., no localized leakage is exploited. However, compared to standard power or EM analyses, the localized EM trace acquisition technique provides a higher SNR for the following analysis traces [HMH+12b], which supports this attacks.

The attack is divided into several steps. First, the frequency range of the analyzed ROs has to be determined. Then, the die area with the highest RO frequency leakage, i.e., the area where ROs frequencies can be observed best, has to be identified, which enhances the SNR. After restricting the frequency range and the die location, two distinct frequencies for each comparison of two ROs can be identified. The overlapping RO comparison, as shown in Figure 4.7, is the key to the presented attack. It allows to link the observed RO frequencies to their corresponding ROs in the correct sequence. Thereby, a full RO PUF model can be generated.

In the following, I detail the EM analysis methods for attacking a standard RO PUF implementation with two counters a and b, and $N_{comp} + 1$ ROs generating an N_{comp}-bit response from N_{comp} comparisons of two ring oscillators $RO_{a,n}$ and $RO_{b,n}$ for $n = 1, ..., N_{comp}$.

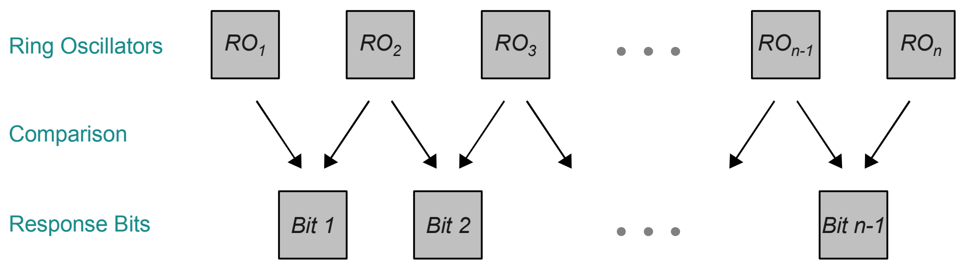


Figure 4.7: Overlapping RO comparisons

As described in Section 2.2.1, for each comparison n, the result is obtained as follows:

$$counter_a(RO_{a,n}) > counter_b(RO_{b,n}) \rightarrow 0 \tag{4.18}$$
$$counter_a(RO_{a,n}) \leqslant counter_b(RO_{b,n}) \rightarrow 1 \tag{4.19}$$

Following, I demonstrate the practical feasibility of the described attack and show how RO PUFs can be protected against the presented attack.

4.5.1 Detailed Attack Steps

In this section, I detail the necessary steps for EM emission attacks on standard RO PUFs.

RO Frequency Range

The first step in the EM analysis of standard RO PUFs is to identify the frequency range in which the ROs operate. Since clock signals, their harmonics, and other disturbances are spread over the entire frequency spectrum, it is essential to concentrate on a restricted frequency range for analysis.

My approach is based on the fact that every comparison in a standard RO PUF exhibits two specific frequencies, corresponding to the two compared oscillators. For each comparison, these frequencies change while all other frequencies resulting from irrelevant parts of the device or external sources are rather constant or noisy. The number of comparisons N_{comp}, which is important to know for this attack, is either publicly available because it is mentioned in the data sheet of an RO PUF device, or it can easily be extracted by looking at the regular RO measurement intervals present in the power side-channel of an RO PUF. Thus, I first divide the collected traces $\mathbf{t}_{(x,y)}$ into N_{comp} equal time slots $\mathbf{t}_{(x,y),n}$, one for every comparison. Then, I transform all $\mathbf{t}_{(x,y),n}$ to the frequency domain:

$$\mathbf{A}_{die} = \{(x, y) \mid x = 1, ..., X; y = 1, ..., Y\} \tag{4.20}$$
$$\mathbf{g}_{(x,y),n} = \mathcal{F}(\mathbf{t}_{(x,y),n}) \ \forall \ (x, y) \in \mathbf{A}_{die}; n = 1, ..., N_{comp} \tag{4.21}$$
$$\mathbf{g}_{(x,y),n} = \{g_{(x,y),n,k} \mid k = 1, ..., T\} \tag{4.22}$$

I accumulate the differences of frequency amplitudes $|g_{(x,y),i,k}| - |g_{(x,y),j,k}|$ between all comparisons i and j in each trace. Finally, for easier comparison, the possibly signed accumulations of amplitude differences are unified to the positive range by taking their absolute value leading to the spectrum $\mathbf{S}_{diff}$.

$$\mathbf{S}_{diff} = \{s_{diff,k} \mid k = 1, ..., T\} \tag{4.23}$$
$$s_{diff,k} = \Big|\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{i=1}^{N_{comp}-1}\sum_{j=i}^{N_{comp}} (|g_{(x,y),i,k}| - |g_{(x,y),j,k}|)\Big| \ \forall \ k = 1, ..., T \tag{4.24}$$

In $\mathbf{S}_{diff}$, high difference peaks indicate frequencies that are present in some comparisons but missing in others, i.e., these frequencies are very likely to be RO frequencies. With the highest amplitudes in this spectrum, the RO frequency range f_{lo} to f_{hi} can be identified.

Area of Leakage

I present two ways to identify the area of RO PUF frequency leakage. Common to both approaches is the filtering of the frequency domain vectors $\mathbf{g}_{(x,y),n}$, i.e., reducing them to T^* frequency bins within the before obtained frequency range f_{lo} to f_{hi}:

$$\mathbf{g}^*_{(x,y),n} = \mathit{filter}(\mathbf{g}_{(x,y),n}, f_{lo}, f_{hi}) \ \forall \ (x, y) \in \mathbf{A}_{die}; n = 1, ..., N_{comp} \tag{4.25}$$

The first method is plotting the mean difference between comparisons in the above identified frequency range in a map $\mathbf{M}$. The resulting map highlights points, where the frequencies can be distinguished. Each pixel $m_{(x,y)}$ of map $\mathbf{M}$ can be calculated as follows:

$$m_{(x,y)} = \frac{1}{T^*}\sum_{k=1}^{T^*}\Big|\sum_{i=1}^{N_{comp}-1}\sum_{j=i}^{N_{comp}} (|g^*_{(x,y),i,k}| - |g^*_{(x,y),j,k}|)\Big| \ \forall \ (x, y) \in \mathbf{A}_{die} \tag{4.26}$$

Another possibility is using the amplitudes of the leaking frequencies directly. One can plot the mean of all amplitudes in the found frequency range for every map point. This map gives a hint which area of a design discloses high amplitudes of RO frequencies.

$$m_{(x,y)} = \frac{1}{T^*} \sum_{k=1}^{T^*} \sum_{i=1}^{N_{comp}} |g^*_{(x,y),i,k}| \;\forall\; (x, y) \in \mathbf{A}_{die} \tag{4.27}$$

These methods allow to identify locations (x, y), where a high leakage is present, in order to combine them into a leakage location set $\mathbf{A}_{leak}$.

Distinct RO Frequencies

In the third step, I focus only on the $|\mathbf{A}_{leak}|$ locations (x, y) in $\mathbf{A}_{leak}$, determined in the previous analysis step. I average all spectra to obtain a noise reduced average spectrum $\mathbf{S}_{comp,n}$ for every RO frequency comparison n.

$$\mathbf{S}_{comp,n} = \{s_{comp,n,k} \mid k = 1, ..., T^*\} \tag{4.28}$$

$$s_{comp,n,k} = \frac{1}{|\mathbf{A}_{leak}|} \sum_{(x,y)\in\mathbf{A}_{leak}} |g^*_{(x,y),n,k}| \;\forall\; n = 1, ..., N_{comp}; k = 1, ..., T^* \tag{4.29}$$

Afterwards, one should be able to visually extract two distinct frequencies f_1 and f_2 (with high amplitudes) in every spectrum $\mathbf{S}_{comp,n}$ where $f_1 < f_2$. These frequencies represent the two RO signals of each comparison. A list L with elements l_1, l_2, ..., $l_{N_{comp}}$, containing the comparison n linked with the found frequencies $f_{1,n}$ and $f_{2,n}, n = 1, ..., N_{comp}$, can be generated as preparation for the last analysis step.

$$L = \{l_n = (n; f_{1,n}; f_{2,n}) \mid n = 1, ..., N_{comp}\} \tag{4.30}$$

RO PUF Modeling

Until now, an attacker only knows about the RO frequencies present during each comparison, but does not have any information about the secret bits, which can be extracted from the RO PUF. In the last step, the measured RO pairs $RO_{a,n}$ and $RO_{b,n}$ of each comparison n come into play. They constitute the missing link between the found frequencies and the corresponding ROs.

When generating a secret response from $N_{comp} + 1$ oscillators by applying a sequential comparison scheme, where $RO_{a,1} = RO_1$ and $RO_{b,1} = RO_2$,

$RO_{a,2} = RO_2$ and $RO_{b,2} = RO_3$ and so on, a response length of N_{comp} bits can be achieved. There, the rule $RO_{a,n} = RO_{b,n-1}, n = 2, ..., N_{comp}$ (second RO in one comparison will be first RO in following comparison) is meant to achieve independent RO comparisons.

However, this rule can be exploited to correctly map the measured frequencies to their corresponding ROs because it leads to the condition that one of two frequencies of a comparison, i.e., $f_{1,n}$ or $f_{2,n}$, must be common to one of the preceding comparison frequencies $f_{1,n-1}$ or $f_{2,n-1}$:

$$\begin{aligned}(f_{1,n} = f_{1,n-1}) \vee (f_{1,n} = f_{2,n-1}) \vee (f_{2,n} = f_{1,n-1}) \vee (f_{2,n} = f_{2,n-1}) \\ \forall\ n = 2, ..., N_{comp}\end{aligned} \tag{4.31}$$

Therefore, the frequency $f_{RO_{a,n}}$ of the oscillator $RO_{a,n}$ can be determined as the frequency that is equal to one of preceding frequencies $f_{1,n-1}$ or $f_{2,n-1}$:

$$f_{RO_{a,n}} = \begin{cases} f_{1,n}: & \textit{if}\ (f_{1,n} = f_{1,n-1}) \vee (f_{1,n} = f_{2,n-1}) \\ f_{2,n}: & \textit{if}\ (f_{2,n} = f_{1,n-1}) \vee (f_{2,n} = f_{2,n-1}) \end{cases} \forall\ n = 2, ..., N_{comp} \tag{4.32}$$

Clearly, $f_{RO_{b,n}}$ must be the other frequency of the two possible frequencies. The frequencies of the very first and the very last oscillator in the sequence, RO_1 and $RO_{N_{comp}+1}$ respectively, will only appear once in the list L, but knowing the mapping of the ROs compared to them, RO_1 and $RO_{N_{comp}+1}$ can be linked to the left over frequencies.

By the shown rules, it is possible to solve the comparisons and map all frequencies to their corresponding ROs. To obtain a full model of an RO PUF, the list of all $N_{comp} + 1$ ROs $RO_1, ..., RO_{N_{comp}+1}$, linked with their frequencies $f_{RO_1}, ..., f_{RO_{N_{comp}+1}}$, has to be derived from the found RO frequencies $f_{RO_{a,n}}$ and $f_{RO_{b,n}}$:

$$f_{RO_1} = f_{RO_{a,1}} \tag{4.33}$$

$$f_{RO_n} = f_{RO_{a,n}}\ \forall\ n = 2, ..., N_{comp} \tag{4.34}$$

$$f_{RO_{N_{comp}+1}} = f_{RO_{b,N_{comp}}} \tag{4.35}$$

If an attacker has reached this point, he can perform the comparisons between all ROs himself and, thereby, generate all secret PUF output bits. However, assuming that an attacker does not know if the comparison uses a $>$ or a $<$, he still has to check which of the two possibilities leads to the correct bit sequence.

Discussion on Analysis Limits

The main attack scenario of the presented methods are RO PUFs used for secret key generation. There, a fixed sequence of N_{comp} challenges is applied step by step to efficiently use the number of integrated ROs. I assume that this challenge sequence is not concealed, but can be regarded as known, as requested for good security designs. However, even in the case of a scrambled challenge sequence, $N_{comp}-1$ ROs are measured at least twice to obtain N_{comp} response bits. Therefore, the list L can be generated and sorted in a way that subsequent comparisons contain one common frequency. Afterwards, the analysis can be continued as described.

Further, the runtime of a single RO frequency comparison or the number of frequency comparisons is assumed to be known, i.e., it is possible to separate the sequence of frequency comparisons. However, even if the runtime would not be known, an attacker could estimate it by dividing the whole measurement time by the number of generated response bits or observe the regularity of RO measurements in the power side-channel of an RO PUF.

In case of very close RO frequencies in one comparison, it might not be possible to distinguish between them. Since this effect is also responsible for noise in RO PUF responses, the uncertainty of my analysis only reflects this noise. Algorithms like fuzzy extractors or error-tolerant protocols can handle a specified amount of noise anyway, which means that an attacker only has to extract the PUF response with a certain precision and eventually happening extraction errors can be seen as noise contained in the PUF response bits.

4.5.2 Case Study: FPGA Ring Oscillator PUF

For practical verification of the proposed attack method, I implemented an RO PUF prototype with chained RO comparisons to extract N_{comp} bits from $N_{comp}+1$ ROs. Step-by-step, it compares two ROs running during the measurement.

For the proof-of-concept, I implemented nine ring oscillators, each built out of seven inverters. During eight comparisons, each of them lasting 4096 cycles of the 20 MHz system clock, eight PUF response bits were generated. I used a small number of oscillators to focus on the evaluation of the basic attack while keeping the data and computational effort for trace storage and processing low. Also, designs with a higher number of ROs lead to large multiplexers, which function as a kind of amplifier and even increase the EM emission related to RO frequencies.

The design was loaded on a Xilinx Spartan XC3S200 FPGA, which was decapsulated from the backside. As I explained in Section 4.2, removing the package of an FPGA does not influence the ROs' frequencies noticeably.

I chose a grid of 50×42 measurement points over the $4.8\,mm \times 4.0\,mm$ die area to acquire 2100 EM emission traces over the FPGA die hosting my RO PUF design. The oscilloscope was set to a sampling rate of 1 *GS/s* and recorded 600000 samples per trace. The function generator for the device's clock signal was synchronized with the oscilloscope.

RO Frequency Range

The first step to disclose the implemented ROs was to identify their frequency range. I calculated the accumulated frequency amplitude difference spectrum as described in Equation 4.24. Figure 4.8 shows the resulting spectrum for my RO PUF implementation. One can find significant peaks in this spectrum that signalize that these frequency amplitudes change from one comparison to another. The largest peak is located at around 100 MHz. By closer inspection, as shown in Figure 4.9, the exact frequency range can be determined to reach from $f_{lo} = 100.3\,\text{MHz}$ to $f_{hi} = 102.1\,\text{MHz}$. Other peaks found in the spectrum correspond to harmonics of the RO frequencies or other comparison-dependent frequencies, e.g., originating from the components of the measuring counter.

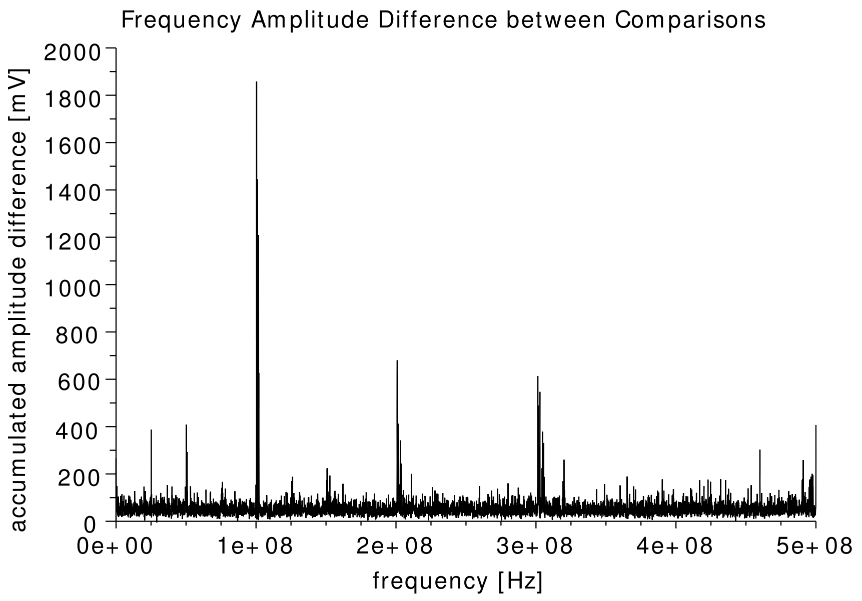


Figure 4.8: Frequency amplitude difference spectrum revealing RO frequency range around 100 *MHz*

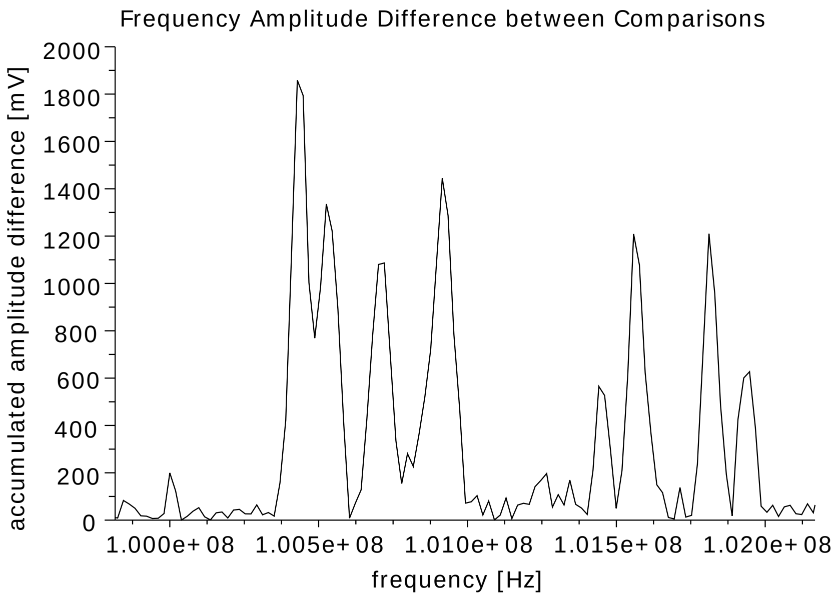


Figure 4.9: Spectrum range with highest peaks

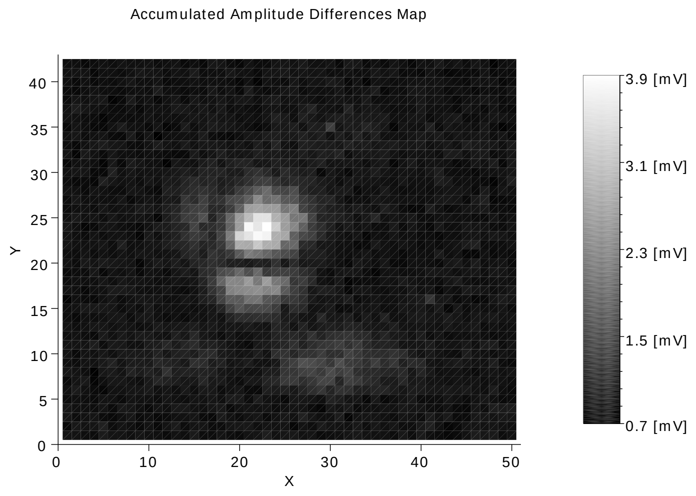
Area of Interest

Next, I generated maps of the FPGA die to find the locations where the oscillators' frequencies can be observed best. I used both proposed methods given by Equations 4.26 and 4.27. One can see light areas in Figure 4.10(a) that indicate amplitude differences between comparisons and therefore represent points of interest for an attacker. The frequency amplitude map, as depicted in Figure 4.10(b), shows similar results indicating high amplitudes of RO frequencies at bright pixels.

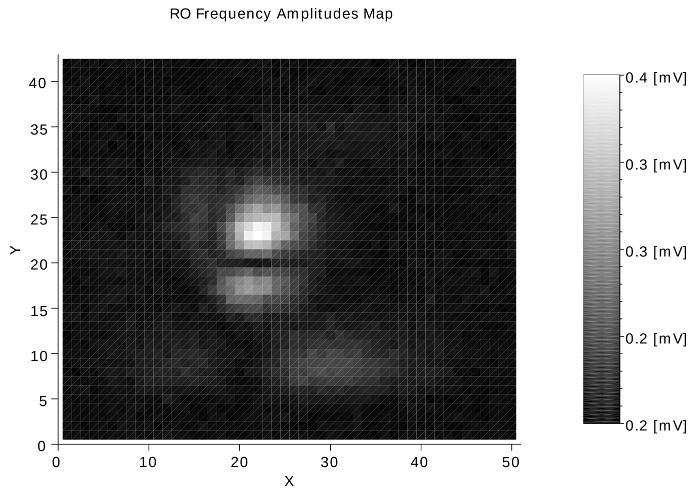
Thus, the area of main leakage can coarsely be limited to the locations within the rectangle spanning over $x = 18, ..., 28$ and $y = 15, ..., 29$. This means that the number of traces to process can be reduced from 2100 to 165 relevant ones, which accelerates the analysis and significantly enhances the quality of the final results.

RO Frequency List

As a last step, I calculated the mean frequency amplitude spectra for every comparison slot, as described by Equation 4.29, and plotted them one beneath the other, as shown in Figure 4.11. One can see that every comparison contains two distinct frequency peaks representing two RO frequencies. Table 4.4 shows an RO frequency comparison list.



(a) amplitude difference map



(b) frequency amplitude map

Figure 4.10: Maps to identify the area of leakage

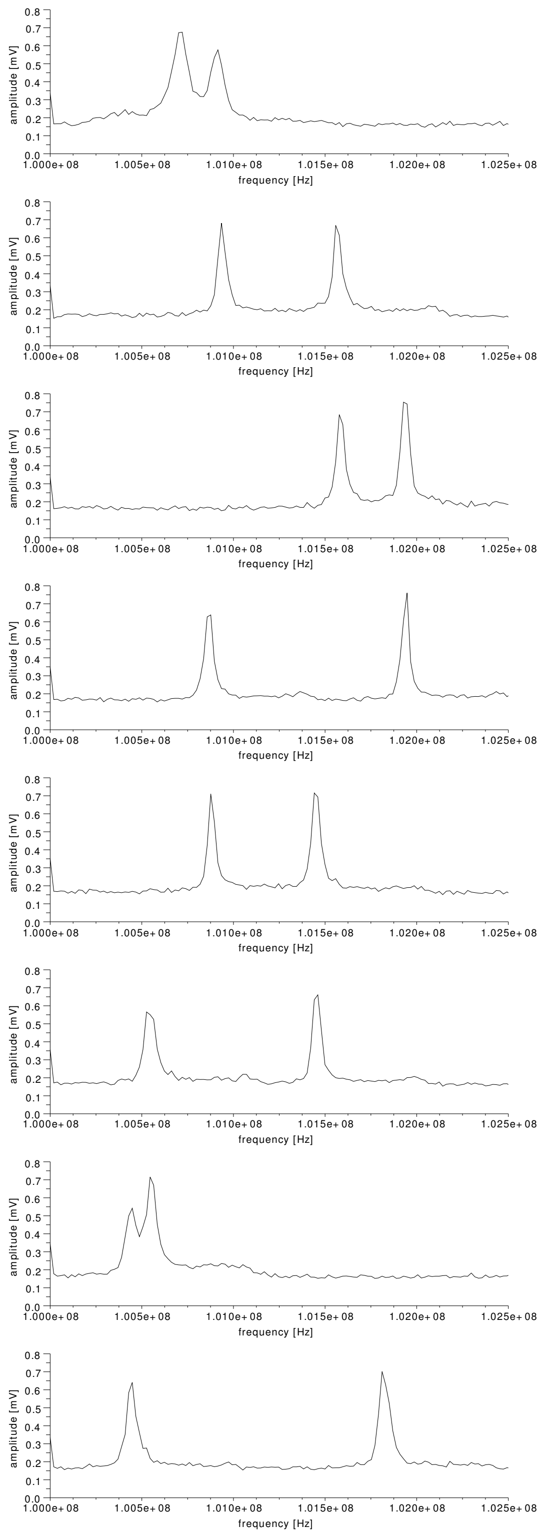


Figure 4.11: Frequency amplitude spectra for each of the 8 RO comparisons

Table 4.4: RO frequency comparison list

comparison n	f_1 / MHz	f_2 / MHz
1	100.71	100.91
2	100.93	101.56
3	101.58	101.94
4	100.87	101.94
5	100.88	101.45
6	100.55	101.45
7	100.44	100.55
8	100.44	101.82

RO PUF Model

To recover the secret response generated by my RO PUF implementation, I finally exploited the frequency overlap of RO comparisons to decode the found frequencies according to Equation 4.32. Since the chain of oscillators is evaluated pair by pair, i.e., first, RO*1* and RO*2*, then, RO*2* and RO*3*, etc., I know that, e.g., *RO2* is common to the first and the second comparison. As one can see, the frequencies 100.91 *MHz* of comparison 1 and 100.93 *MHz* of comparison 2 are very close, while the other frequencies of these comparisons are farther away. Thus, f_{RO_2} can be averaged to 100.92 *MHz*. If one continues like that, a full model of the analyzed RO PUF can be generated by a list, as shown in Table 4.5.

Table 4.5: Complete RO PUF model

RO_m	f_{RO_m} / MHz
1	100.71
2	100.92
3	101.57
4	101.94
5	100.88
6	101.45
7	100.55
8	100.44
9	101.82

On the basis of Table 4.5, an attacker is able to extract two possible secret PUF responses, depending on the comparison operator, i.e., if the RO PUF uses a $>$ or a $<$ comparison. In this case, one of the two bit sequences, which can be generated by comparison of the found frequencies, is

$(1, 1, 1, 0, 1, 0, 0, 1)$. This matches the correct response bits I collected in a log file during measurements. Therefore, this experiment successfully extracted the secret response bits of my RO PUF implementation.

4.5.3 Countermeasure

As shown above, the extraction of RO frequencies from EM measurements is feasible in practice. Therefore, countermeasures are necessary to protect RO PUFs.

The crucial point allowing an attacker to map frequencies to oscillators is the overlapping of ROs in a comparison sequence extracting N_{comp} bits from $N_{ro} = N_{comp} + 1$ ROs. A straight forward approach to prevent oscillator identification is to use every oscillator only once in a single comparison, resulting in a reduced bit extraction rate of $N_{ro}/2$ bits from N_{ro} ROs. Hence, more oscillators have to be implemented for the same number of response bits. This leads to a higher hardware footprint, but the overall extraction runtime stays the same.

Another fact facilitating EM attacks on RO PUFs is the sequential call of all comparisons, i.e., always two frequencies are present at a time during RO measurement. Therefore, the RO sequence can be exploited to map frequencies to oscillators. Comparing all $N_{comp}+1$ oscillators at the same time would be resistant against the shown attack and one could still extract N_{comp} response bits. Also, the runtime would be reduced to the measurement time of a single comparison. However, the significant drawback of this solution is the immense hardware overhead, since every oscillator would need its own counter consisting of several flip-flops and logic gates.

The ideas of non-overlapping RO comparisons and parallel RO measurements are inefficient in terms of their required hardware components if only one of them is used. However, their combination is the solution to achieve a resistant RO PUF architecture, while keeping the required overhead low.

Using a small number of measurement counters N_{cnt} in parallel allows to keep the hardware overhead low, while the measured oscillators can then be compared with overlap to extract $N_{cnt} - 1$ bits. To achieve a high number of RO PUF response bits, several measurement runs N_{run} need to be performed, generating $N_{run} \times (N_{cnt} - 1)$ bits from $N_{ro} = N_{run} \times N_{cnt}$ ROs. However, none of the ROs is allowed to be measured twice in order to consistently avoid any exploitable RO overlap. Table 4.6 shows a list of possible parameters to securely extract 128 bits from an RO PUF. Depending on the required hardware components of an RO and a counter, one could choose the optimal solution for a given device or technology.

Table 4.6: Comparison of parameters for a resistant RO PUF concept

N_{cnt}	N_{run}	$N_{run} \times (N_{cnt} - 1)$	$N_{ro} = N_{run} \times N_{cnt}$
2	128	128	256
3	64	128	192
4	43	129	172
5	32	128	160
6	26	130	156
7	22	132	154

4.6 Localized Attack on Protected RO PUFs

In this section, I detail how RO PUFs that are protected against the attack shown in Section 4.5 can still be attacked by localized EM analysis. The key to this attack is the separation of RO PUF measurement components by exploiting localized EM leakage. If an attacker is able to separately observe measured RO frequencies leaked at multiplexers, counters, or comparators, the security of an RO PUF is broken.

4.6.1 Location-Dependent Frequency Leakage

When attacking RO PUFs, an attacker aims at disclosing the mapping between ROs and their corresponding frequencies. Using localized measurements, a theoretical straight forward approach would be to record a trace at every RO position, extract the dominant frequency, and map it to the RO at this location. However, there are practical problems mitigating this threat:

1. As shown in Section 4.4, the EM emission of a single RO extends to several 100 micrometers, making differentiation of closely placed ROs very hard.
2. There can be more than 1000 oscillators with unknown locations in a PUF design, leaving an attacker with the task to precisely locate each single RO.
3. Even if all ROs can be measured, an attacker still needs to find out which ROs are compared to each other to generate the final PUF bits.

More promising and exploitable points for an practical attack can be discovered by looking at the logic and registers connected to an RO, e.g., multiplexers and counters. These resources are shared among several oscillators, so the number of locations to determine is reduced drastically. And

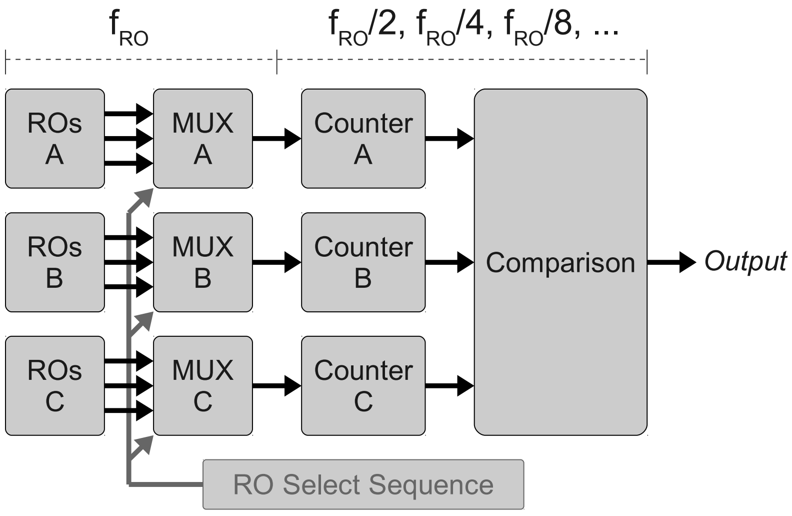


Figure 4.12: Ring oscillator measurement chain

since these circuits usually consist of larger and more complex structures, the emitting area and the emission's amplitude are increased. Also, the number of possible component combinations to obtain the PUF output bits is very much lower than possible RO combinations.

In the common case where ROs are measured by asynchronous counters, each counter flip-flop divides the original RO frequency by 2 (beginning with the least significant bit), as shown in Figure 4.12. This generates additional leakage at fractions of the ROs' frequencies, which can also be exploited.

4.6.2 Attack Steps

As shown in Section 4.5, standard RO PUF architectures are vulnerable to EM analysis. Therefore, I proposed a protected architecture, where the RO comparison overlap is eliminated. In order to enhance efficiency, I proposed the usage of more than two counters for RO measurements. Figure 4.12 shows the architecture of a protected 3-counter RO PUF with its three measurement paths.

If an attacker is able to separate the three measurement paths of an RO PUF implementation of the architecture shown in Figure 4.12, then he is able to obtain a list of RO frequencies that have been measured by each measurement counter during several measurement runs. The challenging task in this case is to locate and separate each measurement path. In this section, I explain how this can be achieved.

A set of N_{ro} oscillators is measured by N_{cnt} counters. Then, $N_{run} = \lceil \frac{N_{ro}}{N_{cnt}} \rceil$ sequential measurement runs are necessary to determine all frequencies that are subsequently used for key generation. The EM emission of an RO PUF has to be observed during this time frame. As N_{ro}, N_{cnt} and N_{run} might be found in the specification of the device, I assume that they are known to an attacker. This helps the adversary to successfully perform the following attack steps:

1. **Trace acquisition.** Record EM traces for all N_{run} measurement runs at every location on a grid over the devices' die surface.

2. **Domain transform.** For every location, divide the recorded trace into N_{run} equal time slots and transform each of them into the frequency domain. This leads to N_{run} frequency spectra at each location.

3. **Location identification.** Identify locations where information about RO frequencies is leaked, i.e., locations that show a significant standard deviation in the frequency domain over all measurement runs. Locations where the frequency spectrum does not vary over different measurement runs do not leak RO frequencies.

4. **Frequency range determination.** Determine the frequency range where ROs can be observed best in order to be able to focus on a narrow frequency band for further processing. This can be achieved by analyzing the contribution of frequencies to the standard deviation at all significant locations found in the previous step. This step might either reveal the original RO frequencies, or fractions of them.

5. **Location-to-path mapping.** Cluster all significantly contributing locations into N_{cnt} measurement paths according to their mainly present frequency in the previously found frequency range. This represents the actual separation of measurement paths, the key to this attack. It can be done by a clustering algorithm such as k-means [DHS01].

6. **RO frequency extraction.** Decide on a location (or a set of locations) for each measurement path and extract the dominant frequency for each path during every measurement run. This leads to a sequence of N_{run} frequency values for every measurement path. Then, the last information about the RO PUF an attacker does not know, is how its measurement counters are compared to each other. However, since there is only a small number of counters, he only needs to test which of the $N_{cnt}!$ possible measurement path comparison combinations is the correct one leading to the full PUF model.

Trace Acquisition and Domain Transformation

The first step is recording traces $\mathbf{t}_{(x,y)}, x = 1, ..., X; y = 1, ..., Y$ over a grid $\mathbf{A}_{die}$ of $X \times Y$ measurement points including all N_{run} measurement runs, as described in Section 4.3. Afterwards, the traces $\mathbf{t}_{(x,y)}$ are divided into N_{run} subtraces $\mathbf{t}_{(x,y),n}, n = 1, ..., N_{run}$, which are then transformed into the frequency domain:

$$\mathbf{A}_{die} = \{(x, y) \mid x = 1, ..., X; y = 1, ..., Y\} \tag{4.36}$$

$$\mathbf{g}_{(x,y),n} = \mathcal{F}(\mathbf{t}_{(x,y),n}) \; \forall \; (x, y) \in \mathbf{A}_{die}; n = 1, ..., N_{run} \tag{4.37}$$

Every transformed trace $\mathbf{g}_{(x,y),n}$ consists of T frequency bins:

$$\mathbf{g}_{(x,y),n} = \{g_{(x,y),n,k} \mid k = 1, ..., T\} \; \forall \; (x, y) \in \mathbf{A}_{die}; n = 1, ..., N_{run} \tag{4.38}$$

Location Identification

The identification of leaking locations is an important preparation step for localized attacks, because only these locations provide information about the leakage of all measurement paths. Separating the found locations is the main challenge for this attack vector.

A map $\mathbf{M}_{loc}$ with points $m_{loc,(x,y)}$ indicating the locations of leakage can be calculated as follows. The mean value $\mu_{(x,y),k}$ of each frequency amplitude $|g_{(x,y),n,k}|$ at every location (x, y) is calculated before hand.

$$\mu_{(x,y),k} = \frac{1}{N_{run}} \sum_{n=1}^{N_{run}} |g_{(x,y),n,k}| \; \forall \; (x, y) \in \mathbf{A}_{die}; k = 1, ..., T \tag{4.39}$$

$$\sigma_{(x,y),k} = \sqrt{\frac{1}{N_{run}} \sum_{n=1}^{N_{run}} (|g_{(x,y),n,k}| - \mu_{(x,y),k})^2 \; \forall \; (x, y) \in \mathbf{A}_{die}; k = 1, ..., T} \tag{4.40}$$

$$m_{loc,(x,y)} = \frac{1}{T} \sum_{k=1}^{T} \sigma_{(x,y),k} \; \forall \; (x, y) \in \mathbf{A}_{die} \tag{4.41}$$

There, locations (x, y) with a high average frequency amplitude standard deviation over all RO measurement runs represent RO-dependent EM emissions. Therefore, these leaking locations can be combined into a set $\mathbf{A}_{leak}$.

Frequency Range Determination

Focusing only on locations in $\mathbf{A}_{leak}$, the contributing frequency range can be determined. Therefore, the amplitude standard deviation spectrum $\mathbf{S}_{sd}$ is calculated as follows.

$$\mathbf{S}_{sd} = \{s_{sd,k} \mid k = 1, ..., T\} \tag{4.42}$$

$$s_{sd,k} = \frac{1}{|\mathbf{A}_{leak}|} \sum_{(x,y)\in\mathbf{A}_{leak}} \sigma_{(x,y),k} \; \forall \; k = 1, ..., T \tag{4.43}$$

From this spectrum $\mathbf{S}_{sd}$, the frequency range of interest, f_{lo} to f_{hi}, can be extracted by taking the range with the highest mean standard deviation values.

Mapping of Locations to Measurement Paths

Until now, an attacker has only gained information about where RO frequencies leak on the die surface. However, he is still not able to separate the locations into N_{cnt} parts, which is necessary to observe each measurement path independently. Therefore, this mapping step is essential for the proposed attack. All locations in $\mathbf{A}_{leak}$ have to be mapped to one of the N_{cnt} measurement paths once. In order to achieve this location mapping, only one measurement run n_{alloc} of all N_{run} measurement runs is necessary. This run n_{alloc} has to be chosen by inspecting the frequencies present during each measurement. A suitable run exhibits three well separable and distinguishable RO frequencies.

For the data of measurement run n_{alloc}, every location (x, y) in $\mathbf{A}_{leak}$ is analyzed for the mainly present frequency bin $f^{max}_{(x,y)}$, i.e., the frequency with the highest amplitude in the frequency range f_{lo} to f_{hi}. Afterwards, the locations (x, y) can be clustered into clusters $Q_1, ..., Q_{N_{cnt}}$ according to their frequencies $f^{max}_{(x,y)}$, e.g., by the k-means algorithm [DHS01]. In this case, a clustering algorithm is useful because it automates the process of separating the number of locations in $\mathbf{A}_{leak}$ according to the locally emitted frequencies. Thereby, the N_{cnt} measurement paths can be identified.

$$f^{max}_{(x,y)} = k \; : \; |g_{(x,y),n_{alloc},k)}| = max(|g^{*}_{(x,y),n_{alloc}}|) \tag{4.44}$$

$$\{Q_1, ..., Q_{N_{cnt}}\} = cluster(\{f^{max}_{(x,y)} \; \forall \; (x, y) \in \mathbf{A}_{leak}\}) \tag{4.45}$$

RO Frequency Extraction

Finally, for every cluster $Q_i \in \{Q_1, ..., Q_{N_{cnt}}\}$, i.e., every measurement path, one observation point $p_i = (x_{Q_i}, y_{Q_i})$ out of all locations in each cluster Q_i has to be fixed. This is achieved by taking the one that exhibits the highest amplitudes in the frequency spectrum from f_{lo} to f_{hi} for each cluster. Then, at these locations, the frequency bins $f_{p_i}^{max}$ can be extracted for every of the N_{run} measurement runs. The results from this analysis are N_{cnt} lists L_i with elements $l_{i,1}, ..., l_{i,N_{run}}$ of sequential RO frequencies for every measurement path $i = 1, ..., N_{cnt}$:

$$L_i = \{l_{i,n} = (n; f_{p_i,n}) \mid n = 1, ..., N_{run}\} \; \forall \; i = 1, ..., N_{cnt} \tag{4.46}$$

Since the frequencies in these lists are compared in a fixed manner within an RO PUF, the final step an attacker has to perform is to generate the relative comparison results for each of the $N_{cnt}!$ possible comparison combinations of the obtained frequency lists. One can then check which of the generated keys is the correct one by comparing it to en-/decryption data of the PUF system under attack.

Discussion on Attack Limits

For this attack, I assume that an attacker has knowledge about the number of implemented counters N_{cnt} and the number of RO measurement runs N_{run}. However, if unknown, the number of measurement runs could be identified by looking at the power consumption of the device. The number of counters might be found in the datasheet of the device, but can also be obtained by visual inspection of the frequency spectrum of the RO frequency range during a measurement run. There, depending on the SNR, clear peaks are visible representing the number of RO frequencies measured in parallel, i.e., the number of counters. Therefore, the underlying assumptions are realistic.

The crux of the matter for this attack is the feasibility to separate all measurement paths from each other. This is a practical issue that depends on the technology of the device under attack as well as the precision of the used measurement equipment. Therefore, this will be discussed after the proof-of-concept attack presented in Section 4.6.3.

4.6.3 Case Study: Protected FPGA RO PUF

In this case study, I analyze the presented attack on a protected RO PUF design that is resistant against the EM attack shown in Section 4.5. I explain

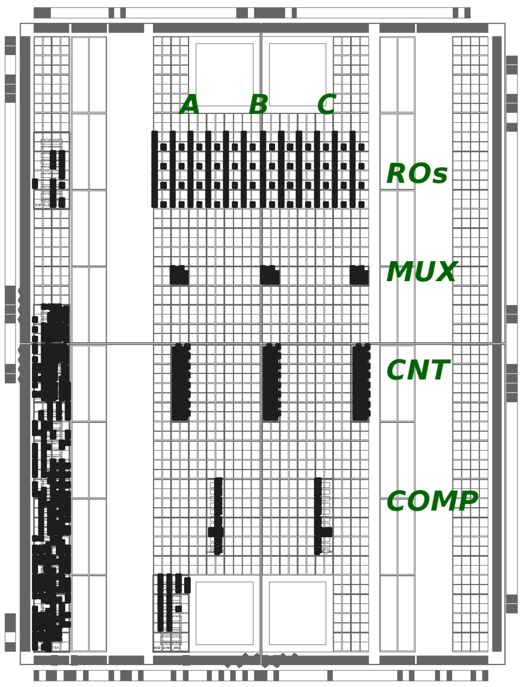


Figure 4.13: Floorplan of 3-counter RO PUF design on a Xilinx Spartan XC3S200 FPGA

the design under attack and demonstrate how to locate leaking RO PUF components. Further, I perform the identification of the leaking RO frequency range, and show the final extraction results.

Protected Implementation

The design under attack generates 32 PUF bits based on $N_{ro} = 48$ 5-inverter ROs, which are measured by $N_{cnt} = 3$ counters in $N_{run} = 16$ RO measurement runs, each lasting 1027 cycles. I implemented three measurement paths A, B, and C, each consisting of an array of 16 ROs, a multiplexer, and an asynchronous counter, as shown in Figure 4.13. I used a Xilinx Spartan XC3S200 FPGA as the device under attack. To obtain two PUF output bits, $\mathrm{CNT_A}$ is compared to $\mathrm{CNT_B}$ and $\mathrm{CNT_B}$ to $\mathrm{CNT_C}$, each comparison yielding a 1, if the first counter measured a higher frequency than the second one, and a 0 otherwise. Note that components at the very left side of the FPGA floorplan are necessary for control purposes. To verify my attack results, the generated PUF bits are logged over serial communication.

For this proof of concept, I chose to generate only 32 PUF response bits, because for designs with more generated bits (and therefore more ROs), the

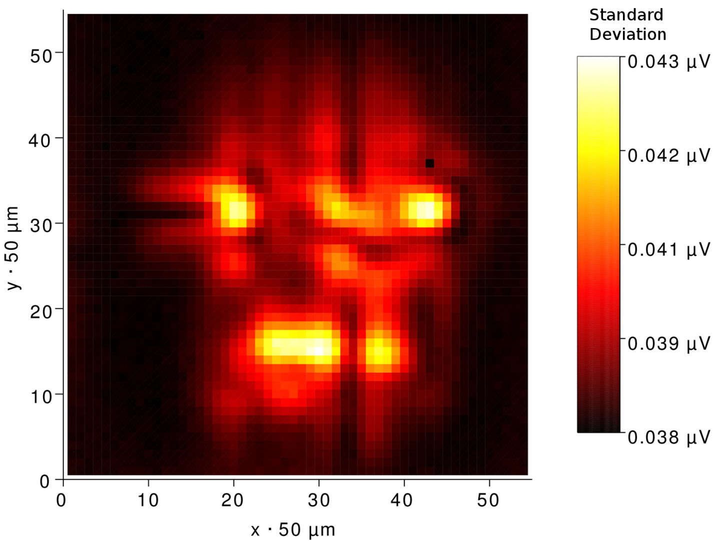


Figure 4.14: Standard deviation map indication leaking areas

multiplexers will be larger generating even higher amplitudes for leaking information about RO frequencies. Also, I deliberately placed each component within a compact area, leaving space between each part of the RO PUF. This gives me the opportunity to judge the origin and the spatial extent of EM leakage. It also allows me to determine whether closer placement would lead to strong EM signal overlap or not.

Leaking Locations and RO Frequency Range

After trace acquisition and domain transform, an attacker has to identify locations, where EM emissions reveal currently active RO frequencies over all RO runs, i.e., these locations will have a high standard deviation in the frequency spectrum over different RO measurement runs. Therefore, for every location, i.e., every recorded trace, I obtained the standard deviation over all FFT spectra of the 16 RO measurement runs. The map shown in Figure 4.14 was generated according to Equation 4.41 and depicts the mean value of the standard deviation of all frequency amplitudes over all 16 measurement runs.

Comparing Figure 4.14 to the floorplan in Figure 4.13, one can see that exploitable frequency emissions mainly occur at locations of multiplexers and between counters and comparators. Note that the multiplexers and comparators serve as a kind of amplifier for the leakage generated at ROs and counters because they consist of several gates oscillating with RO/counter frequencies,

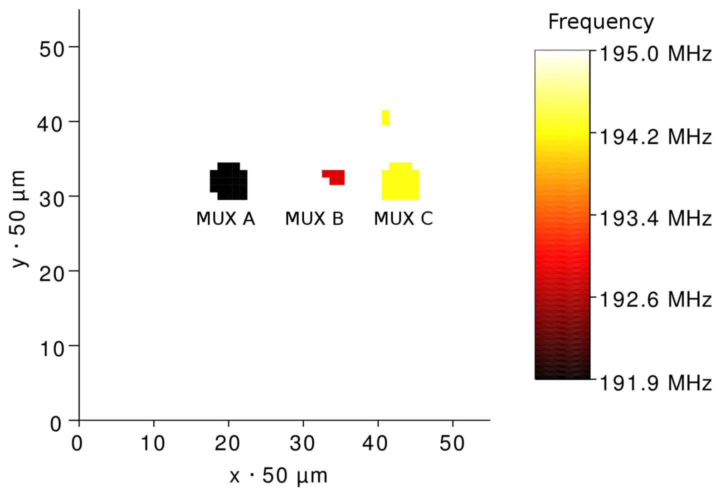


Figure 4.15: Map showing three separated areas of leakage acquired from analyzing the first measurement run

which makes them a favored target. My comparator implementation continuously compares the counter states to demonstrate their possible leakage. However, the counter signals to the comparators can also be gated to avoid comparator leakage during operation. For all further analyses, I only focus on the identified locations exhibiting significant values in Figure 4.14 (upper half of the map's value range).

To determine the ROs' frequency range, I averaged the standard deviation spectra of all identified locations as described by Equation 4.43. In the resulting spectrum, all frequencies contributing to the location-dependent leakage are represented by peaks. I found accumulations of frequencies with high values at $188.75 - 195.25$ MHz, $94.37 - 97.63$ MHz, and $47.18 - 48.82$ MHz, representing the original RO frequency range, the half, and the quarter frequency ranges, respectively.

Mapping Locations to Components

In this step, I mapped the identified locations to three measurement paths, in order to be able to measure frequencies for each path separately. I performed this mapping as described in Equation 4.45 and chose the first RO measurement run ($n_{alloc} = 1$) as a basis. I restricted my analysis to identified locations that showed a significant standard deviation in the RO frequency range of $188.75 - 195.25$ MHz. For each of these locations, I extracted the frequency value with the highest amplitude.

Then, the actual mapping was achieved by a k-means algorithm [DHS01],

Table 4.7: RO PUF analysis results

Meas. Run	RO_A / MHz	RO_B / MHz	RO_C / MHz	PUF Bits
1	191.883	192.839	194.205	00
2	193.288	190.302	191.434	01
3	192.566	191.317	193.151	01
4	193.561	190.712	189.249	11
5	189.444	189.015	189.990	01
6	190.205	188.917	189.639	01
7	191.649	189.483	191.863	01
8	189.815	190.146	190.127	10
9	192.176	190.224	193.268	01
10	193.444	190.420	193.093	01
11	192.449	192.488	192.976	00
12	192.956	193.717	193.151	10
13	191.122	191.571	190.810	10
14	192.312	190.868	192.761	01
15	192.468	189.580	192.098	01
16	191.649	191.122	192.683	01

clustering all location-frequency pairs according to their frequencies into three categories. After this step, all locations emitting similar frequencies belonged to the same cluster (measurement path). The resulting separation is shown in Figure 4.15 and indicates that also closely placed designs can be attacked, since the clusters are far apart and clearly separated. Note that counter/comparator locations are not visible, since they leak in a different frequency range.

RO Frequency Extraction

In the final step, I chose the observation locations for the three measurement path clusters to be $p_1 = (20, 32)$, $p_2 = (34, 33)$, and $p_3 = (43, 32)$, respectively, because these locations exhibited the highest values in Figure 4.14 for each cluster. I was able to obtain the three concurrently emitted RO frequencies during all measurement runs by extracting the frequency with the highest amplitude at all three locations for every run. My results are listed in Table 4.7.

Testing the $N_{cnt}! = 3! = 6$ possibilities of how the counter values can be compared, I found a bit sequence exactly matching the PUF response bits logged over serial communication during trace acquisition. Even small

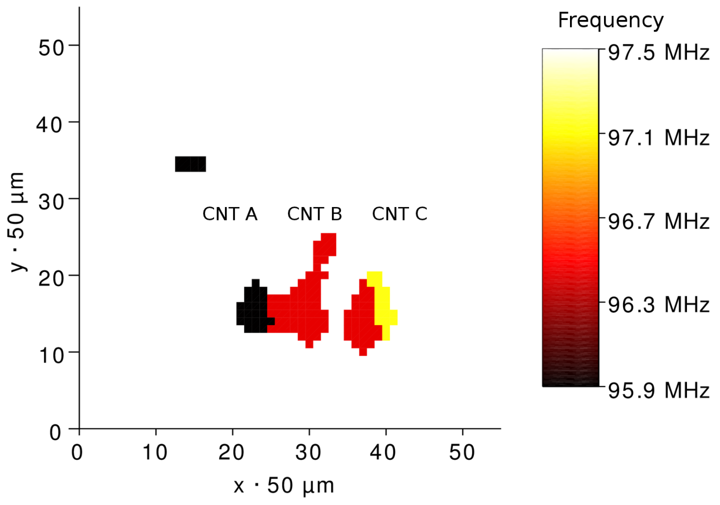


Figure 4.16: Cluster map at half frequencies

differences, e.g., in measurement run 5, lead to correct results. At this point, an attacker would have to confirm his possible bit sequences by performing a test en-/decryption with the crypto module using the PUF's key.

Counter/Comparator Leakage

Performing the clustering for the frequency range 94.375 – 97.625 MHz, i.e., the half of the original RO frequency range, I derived Figure 4.16. Thereby, I demonstrate that the attack can also be performed at fractions of the original RO frequencies that are leaked by asynchronous counters and the subsequent comparators. There, the advantage of using a clustering algorithm becomes clear because it allows to separate adjacent and different sized areas. The black rectangle in the upper left part of Figure 4.16 seems to be a side-effect present in this frequency range.

4.6.4 Countermeasures

In Section 4.6.3, I showed that it is feasible in practice to break the security of RO PUFs by localized EM analysis. In this section, I discuss two countermeasures to protect RO PUF devices: (1) randomizing the measurement paths as far as possible to break the relation between an RO and the measurement components used to determine its frequency, and (2) interleaving the placement of critical measurement components to render the practical separation harder or even impossible.

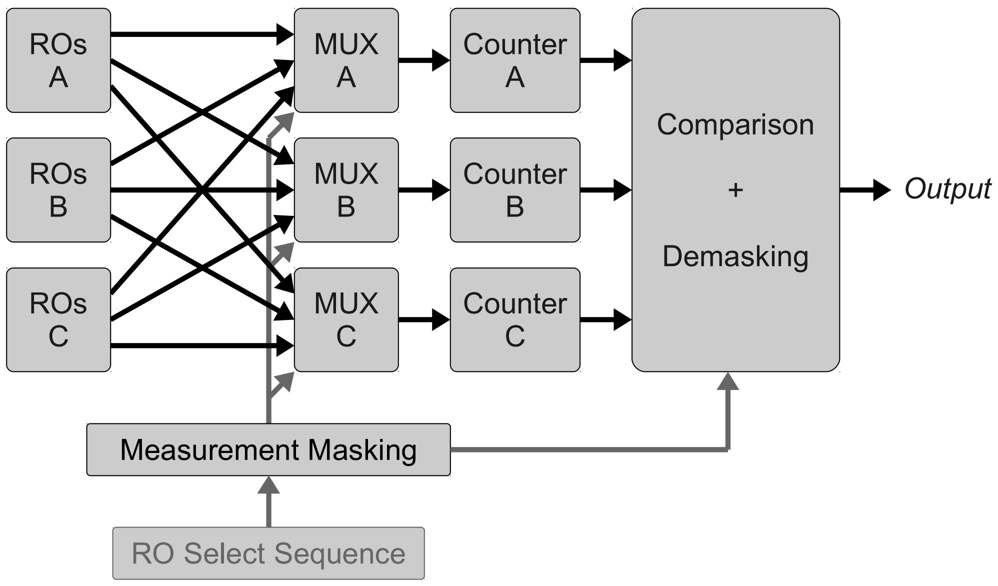


Figure 4.17: Randomized RO PUF measurement architecture

Measurement Path Randomization

The proposed attack exploits that each RO has a fixed location and a specific measurement path through a multiplexer to a counter. Clearly, the location of a unique RO cannot be randomized during runtime, but the subsequent measurement paths can be changed dynamically by a randomized RO selection logic. This yields a location masking scheme that breaks the relation between an RO and its measurement path. An attacker facing a randomized measurement scheme has no chance of mapping present frequencies to measurement paths and will therefore not succeed with the proposed attack.

To achieve this, each N_{run}-to-1 multiplexer of an RO PUF architecture, as shown in Figure 4.12, has to be replaced by an N_{ro}-to-1 multiplexer that is connected to all available ROs, as shown in Figure 4.17. Additionally, the N_{cnt} addresses of the measured ROs must be distributed to these multiplexers according to a random location mask m_l to randomly choose the measurement paths for each measurement run. After the measurement process, the masked measurement paths have to be demasked again, e.g., by N_{cnt} multiplexers controlled by the location mask m_l, which resolve the earlier random permutation performed in the masking step. Thereby, randomization of the measurement paths through multiplexers and counters is possible. However, as the RO locations cannot be randomized, the RO array stays vulnerable.

This countermeasure would also require significant overhead. For the implementation attacked in Section 4.6.3, all 16-to-1 multiplexers would have to be replaced by 48-to-1 multiplexers, resulting in approx. 200% multiplexer area overhead. Further, RO address masking and demasking logic has to be added as well as a TRNG.

Interleaved Placement

Interweaving signal routing and purposely interleaving component placement is a kind of practical hiding countermeasure [HdlTR12] that can only provide security to a certain degree depending on an attacker's facilities. Figure 4.15 shows that the leaking areas of each multiplexer are clearly defined and only adjacent placement would not provide protection from localized EM attacks. Therefore, I propose to interleave ROs, multiplexers, comparators, and counters on a register and look-up table level.

The strength of this countermeasure is that neither additional logic nor a TRNG is needed. However, careful placement and routing techniques are required to achieve a high protection level. Also, this technique can be combined with the location masking countermeasure to hide leakage originating from the RO array and the multiplexer input gates, which cannot be protected otherwise.

4.7 Summary

I presented two physical attacks on RO PUF implementations that exploit information leaked by their EM emissions.

For standard implementations, the mapping between measured frequencies and corresponding ROs can be achieved by observing which measured frequency overlaps with the subsequent comparison. Thereby, a full RO PUF model can be generated by a physical attacker. In this case, localized EM measurements allow for a higher SNR than standard power or EM measurements and, therefore, enable clear identification and interpretation of RO frequencies. I also demonstrated the practical feasibility of the described attack. As a countermeasure, I proposed completely avoiding overlapping RO comparison as well as using more than two counters for efficiency reasons.

In the second attack, location-dependent leakage of standard and protected RO PUFs is exploited. I showed the steps which an attacker has to perform in order to separate RO PUF measurement paths. Based on this separation, an adversary can sequentially observe all RO frequencies for each RO measurement path, which, again, allows him to create a functional duplicate of the RO PUF under attack. In a case study, I analyzed the practical obstacles of this attack and conclude that it is feasible with state-of-the-art measurement equipment. However, smaller technology nodes will require even more precise equipment. I proposed two countermeasures against the presented localized attack, namely, randomization of measurement paths and interleaved placement of RO PUF components.

Summarizing, I discovered two important attack vectors for RO PUFs that represent the two first physical attacks on RO PUF implementations. Further, I showed that these threats can be exploited in practice by two case studies. I proposed countermeasures against both attacks in order to protect RO PUFs. Considering a wider scope, the presented attacks and countermeasures can also serve for the development and hardening of other RO-based PUFs, e.g., the sum PUF.

Chapter 5

Attacks on Key Generation

Currently, the main application of PUFs is secret key generation, e.g., as brought to market by the companies Verayo and Intrinsic-ID. However, this application is not limited to silicon PUFs, but also used in the case of PUFs aiming for increased tamper resistance, e.g., coating PUFs [TSŠ+06] or optical smartcard PUFs [EFK+12]. Key generation algorithms, e.g., fuzzy extractors [DRS04] or IBS [YD10b], have to be used to process internally generated PUF bits in order to obtain a reliable cryptographic key. They consist of an enrollment phase and a reconstruction phase. The enrollment phase takes place in a secure environment after manufacturing. There, a specific key is fixed by embedding it into PUF response bits or extracting it directly from these bits. Also, helper data is generated from the characteristics of the noisy PUF response bits during this phase. This redundancy data does not contain enough information about the secret key to be of help to an attacker and can therefore be stored in external, unsecure memory. During the reconstruction phase, the stored helper data and a vector of noisy PUF response bits are used to regenerate the enrolled secret key. Since the reconstruction is performed again and again in the field, this part of the algorithm is prone to physical attacks.

As demonstrated in Chapter 4, PUFs can be attacked directly without any knowledge about the subsequent key generation mechanisms. However, this approach usually requires insight into the PUF architecture and sophisticated measurement equipment. In this chapter, I present two attacks on implementations of COFEs. The first one mounts a DPA attack on the error correction module of a COFE based on collected power consumption traces. The second one targets the Toeplitz hashing extractor by an SPA attack. I propose codeword masking as a countermeasure against these attacks, which enables masking of linear ECCs without affecting their error correction capabilities. This scheme can be applied to the helper data input and can be continued throughout the whole reconstruction algorithm up to the point where the masked key and its mask leave the module. Then, the subsequent modules can make use of the already masked key, enabling a consistent PUF key generation masking scheme. Parts of this chapter have been published in the paper *Side-Channel Analysis of PUFs and Fuzzy Extractors* at the 4th International Conference on Trust and Trustworthy Computing in 2011 [MSSS11b] and the preprint publication *Protecting PUF Error Correction by Codeword Masking* in 2013 [MSS13].

This chapter is organized as follows. Section 5.1 gives an overview of related work in side-channel attacks on key generation systems. Section 5.2 demonstrates a DPA attack by helper manipulation exploiting leakage from a COFE's error correction module. An SPA attack on the popular Toeplitz hashing for COFEs is shown in Section 5.3. A consistent countermeasure to protect against both attacks is described in Section 5.4. A summary of this chapter is found in Section 5.5.

5.1 Related Work

After the fundamental work about power analysis of Kocher et al. [KJJ99], a large amount of further research was conducted regarding improved signal processing, enhanced attacks, and strong protection strategies. The book *Power Analysis Attacks: Revealing the Secrets of Smart Cards* of Mangard et al. [MOP07] represents a solid basis for side-channel attacks. In this chapter, I do not aim at developing a qualitatively new power analysis method, but use well-known methodologies to demonstrate vulnerabilities of PUF-based key generation mechanisms. The difference to previous work in the field of side-channel attacks is that changes of intermediate values in key generation algorithms are achieved by helper data manipulation instead of different plain- or ciphertexts. Also, the proposed countermeasure is an extension of random masking techniques to enable protection of error correction modules that cannot be protected by standard masking.

Until now, only a few publications deal with side-channel analysis of secure sketches and fuzzy extractors. Karakoyunlu et al. [KS10] presented two attacks on software implementations of Reed-Solomon codes [Bos99] and Bose-Chaudhuri-Hocquenghem (BCH) codes [Bos99]. The authors claimed that standard software implementations of these codes show data dependent leakage that can be attacked by SPA. Further, they proposed a differential template attack, where they built templates for every possible input symbol by varying the helper data. Then, a set of distinguished templates is chosen for the analysis of the device under attack. In contrast to this earlier attack paper, I also propose a countermeasure to protect secure sketches and fuzzy extractors against the attacks I discovered.

There exists some related work about ECCs causing vulnerabilities in security critical systems. In 2009, Dai and Wang [DW09] presented a study on side-channels of ECCs used in reliability enhancing techniques for memories. Also, side-channel attacks on the McEliece public key cryptosystem [STM+08, MSSS11c] exploit leakage of ECC implementations. I show that also side-channel leakage of ECCs in PUF-based key generation algorithms has to be considered as a realistic threat to their security.

Side-channel leakage of extractors in the scenario of leakage-resilient cryptography has been investigated by Medwed and Standaert [MS11]. However, in their case, extractors implement a secret key used in every operation, whereas in the case of COFEs, the extractor function generates a secret key from the input bits of a secret source.

5.2 Attack on Error-Correcting Codes

In this section, I first discuss the fact that no standardized COSS and COFE implementations exist. Then, I demonstrate that correlation-based DPA [BCO04] is applicable to a COSS. The shown vulnerability of the ECC implementation also holds for COFEs because they are extensions of COSSs.

The basis for the following analyses are N_{tr} traces $\mathbf{t}_n$ with T samples $t_{n,k}$ obtained from power or EM measurements of a device during the key generation process.

$$\mathbf{t}_n = \{t_{n,k} \mid k = 1, ..., T\} \; \forall \; n = 1, ..., N_{tr} \tag{5.1}$$

5.2.1 Implementation Diversity of ECCs

COFEs require the use of ECCs to ensure highly reliable key reconstruction. Error correction schemes have been under research for a long time and a diversity of ECCs exists. A comprehensive overview of codes, their decoding algorithms, and efficient implementations can be found in the book *Channel Coding for Telecommunications* by Bossert [Bos99].

In the case of COFEs, one important requirement is an efficient software or hardware implementation of the used ECC. Until now, research has focused on efficient hardware implementations that keep the silicon area footprint of PUF key generation modules low. Bösch et al. [BGS+08] presented a study on the efficiency of hardware implementations of hard-decision ECCs for COFEs. They investigated repetition codes, Reed-Muller codes, Golay codes, and BCH codes. Their results indicate that, compared to the other evaluated codes, BCH codes have the strongest error correction capabilities. However, BCH codes are known to have a large hardware footprint, resulting in inefficient implementations. They proposed to use concatenated codes [For65] to achieve a compromise between error correction strength and implementation complexity. One of the shown results is the fact that ECC output word error probabilities of less than 10^{-6} can be achieved without requiring a complex BCH code, e.g., by the combination of repetition and Golay codes. On the other hand, combinations of repetition and BCH codes significantly lower the number of required PUF response bits. Therefore, no general suggestion for COFE implementations can be made. PUF system designers always have to decide between hardware implementation efficiency and required PUF bits. Another factor influencing the choice of implementation is the generated amount of helper data. Also note that the mentioned study does rely on the noise and error characteristics of SRAM PUFs with an average bit error probability of 0.15. However, optimizing a COFE for another PUF architecture might lead to results suggesting different combinations as most efficient compromise between strong error correction and low hardware resource consumption.

A further fact to consider for COFE design is that the average bit error probability is not the best error characterization technique for PUFs because, usually, some PUF bits are very stable and others are very noisy. Including these properties into error correction can gain further efficiency. For COFEs, this can be handled by using soft-decision information for error correction, as shown by Maes et al. [MTV09]. There, the authors suggest the usage of a concatenation of a repetition code and a Reed-Muller code, both using soft-decision information about the reliability of their input bits gathered from an SRAM PUF.

The bottom line is that many different COFE implementations exist because resource constraints vary over different applications and the PUF response bit error probabilities always depend on the used PUF type and technology. Therefore, possible attacks on error correction modules have to be adapted for the specific type of ECC implementation. However, the codeword masking countermeasure I propose in Section 5.4 can be applied to all linear ECCs no matter which type.

5.2.2 DPA based on Helper Data Manipulation

Differential side-channel attacks on cryptographic algorithms require the manipulation of one of the algorithm's inputs to generate changing intermediate values. Looking at Figures 2.13, 2.14, and 2.15, one finds that helper data $\mathbf{w}$ is actually the only input data that can be manipulated without invasive tampering. The noisy PUF response bit vector $\mathbf{r}'$ is determined by the unique PUF properties, therefore, manipulation of helper data $\mathbf{w}$ is the only practical way to achieve changes in internal intermediate values. In the case of code-offset constructions, the manipulation of a single bit of $\mathbf{w}$ directly flips one input bit of the analyzed decoding function $decode_C(\mathbf{r}' \oplus \mathbf{w})$, which is a desirable property for an attacker.

It is legit to assume deterministic write access to the helper data string for many embedded security applications because helper data is usually stored in external memory, which actually is a profitable advantage of PUFs. However, if the helper data is located in a memory that is hard to manipulate, as it might be the case for smartcards, DPA on PUF key generation becomes impossible or at least very hard.

In order to apply correlation-based DPA to the error correction of a COSS, first, an intermediate value of $decode_C()$ has to be chosen as an attack point. It can be assumed that an attacker knows the function $decode_C()$ from specification documents of a device with PUF-based key generation or has determined its characteristics by reverse engineering.

Then, traces $\mathbf{t}_1, ..., \mathbf{t}_W$ with T samples each have to be collected for W different helper data input vectors $\mathbf{w}_1, ..., \mathbf{w}_W$ resulting in a $W \times T$ trace matrix $\mathbf{T}$. Based on the chosen power model, for each combination of the R possible PUF response vectors $\mathbf{r}_1, ..., \mathbf{r}_R$ and the values for each possible helper data vector $\mathbf{w}_1, ..., \mathbf{w}_W$, hypothetical intermediate values have to be calculated and stored as a $W \times R$ hypothesis matrix $\mathbf{H}$.

Finally, the correlation between the T sample columns of $\mathbf{T}$ and each of the R PUF response hypothesis columns of $\mathbf{H}$ has to be computed as used in standard correlation-based DPA [BCO04, MOP07] to obtain a $R \times T$ correlation matrix $\mathbf{M}$. The maximum correlation value in $\mathbf{M}$ gives the trace sample and the best correlating PUF bit vector hypothesis. This PUF bit vector represents the extracted secret that, together with the original helper data $\mathbf{w}$, can be used to calculate the embedded key $\mathbf{k}$.

For practical attacks, the noise contained in $\mathbf{r}'$ now and then changes bits of the ECC decoder input of a secure sketch, which makes the DPA inaccurate. Therefore, PUF noise generally leads to an increased number of required traces.

5.2.3 DPA on Secure Sketch FPGA Implementation

For the investigation of a secure sketch implementation, I decided to use a concatenation of two ECCs C_1 and C_2 to achieve a compromise between im-

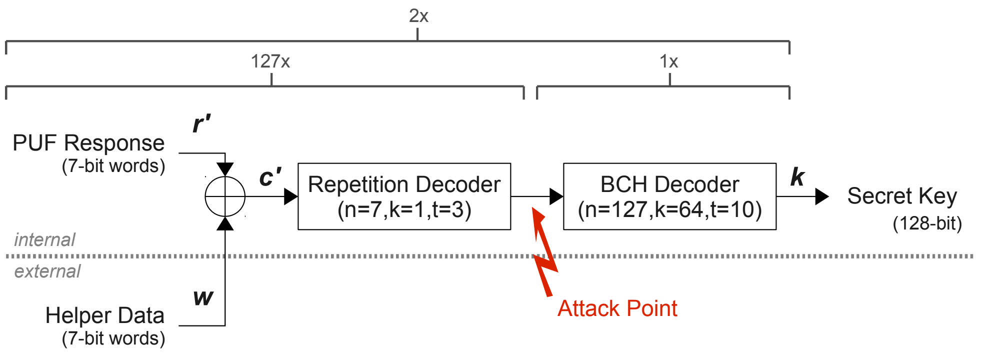


Figure 5.1: Secure sketch implementation under test

plementation complexity and error correction capabilities [BGS+08, MTV09]. During enrollment, first the encoding function of C_1 is used and then its output is further processed by the encoding function of C_2. For decoding during the reconstruction phase, the process is reversed, i.e., first C_2, then C_1.

The COSS reconstruction architecture to be attacked is shown in Figure 5.1. I chose an $(n_2{=}7, k_2{=}1, t_2{=}3)$ repetition code for C_2 because it can be implemented very efficiently. For C_1, I chose an $(n_1{=}127, k_1{=}64, t_1{=}10)$ BCH code, which has the ability to significantly lower the residual error probability, but also incurs a more complex implementation.

As shown by Bösch et al. [BGS+08], the output word error probabilities P_1 and P_2 after decoding a noisy codeword with bit error probability p_b by C_1 and C_2, respectively, can be estimated as follows. The resulting bit error probability of the output bits is known to be lower than the calculated word error probability [DMV04].

$$P_2 = \sum_{i=t_2+1}^{n_2} \binom{n_2}{i} p_b^i (1-p_b)^{n_2-i} = 1 - \sum_{i=0}^{t_2} \binom{n_2}{i} p_b^i (1-p_b)^{n_2-i} \tag{5.2}$$

$$P_1 = \sum_{i=t_1+1}^{n_1} \binom{n_1}{i} P_2^i (1-P_2)^{n_1-i} = 1 - \sum_{i=0}^{t_1} \binom{n_1}{i} P_2^i (1-P_2)^{n_1-i} \tag{5.3}$$

For the two codes I chose, P_2 can be calculated to be $0.5 \cdot 10^{-6}$, i.e., the resulting bit error probability is less then 10^{-6}, which represents a realistic requirement [BGS+08].

I embedded a 128-bit key into 1778 PUF response bits, which results in 1778 bits of helper data. My implementation has two 7-bit input interfaces for chunks of PUF response bits and helper data bits. The code-offset

XOR and the repetition decoding are implemented in combinational logic, which yields a decoded 1-bit output for each 7-bit helper data input word. After each repetition decoding, the output bit is shifted into the BCH decoder [Jam97], which bit-serially decodes 127 input bits to a stable 64-bit word. This procedure is performed twice to obtain a 128-bit key.

I did not use a real PUF implementation, but provide PUF response bits as well as helper data from a preloaded circular buffer. This is important to obtain DPA results that are not influenced by a PUF's specific noise characteristic.

I chose the output of the repetition code decoder, which is stored in the input register of the BCH decoder, as the intermediate value to attack, as shown in Figure 5.1. One reason for that is the smaller computational complexity of a repetition code decoding compared to the calculation of intermediate values of a BCH code. The second reason is that all codeword inputs of the 7-to-1 repetition decoder can be covered by manipulating the 7-bit helper data chunks, which leads to $W = 128$ traces per repetition codeword. I used a Hamming distance model hypothesis between two succeeding repetition code output bits to estimate the hypothetical power consumption under $W = 128$ different helper data manipulations and $R = 128$ possible PUF response bit vectors. For the first decoded bit, the preceding register value is assumed to be zero after reset.

I synthesized my COSS design for a Xilinx XC3S200 FPGA and analyzed its power consumption over a 10 Ohm shunt resistor with a differential probe connected to a LeCroy wavePro 715Zi oscilloscope. I recorded 128 traces per repetition codeword for every possible manipulation of the 7-bit helper data input. I focused my analysis on the first four cycles after each helper data word was provided to the key reconstruction because the repetition decoding happens during this time frame. Afterwards, I correlated the hypothetical power values with the measured traces. For all following hypothesis correlation figures, I only depicted positive hypothesis correlations because the linearity property of the analyzed COSS circuit leads to the fact that inverted PUF inputs show a 'mirrored' negative correlation, which does not provide further information.

In Figure 5.2, the maximum correlation of all 7-bit PUF hypothesis for one repetition code decoder run are shown. The values were generated with only 128 measured traces without any preprocessing. The estimated significance bound [MOP07] for a DPA attack with 128 traces is approx. $4/\sqrt{128} = 0.35$. Therefore, I am confident that my attack worked correctly, since the results show a correlation of 0.52 for the correct PUF response bits (0x43) and leave all other hypothesis below or only slightly above the significance bound (0.38, in the best case).

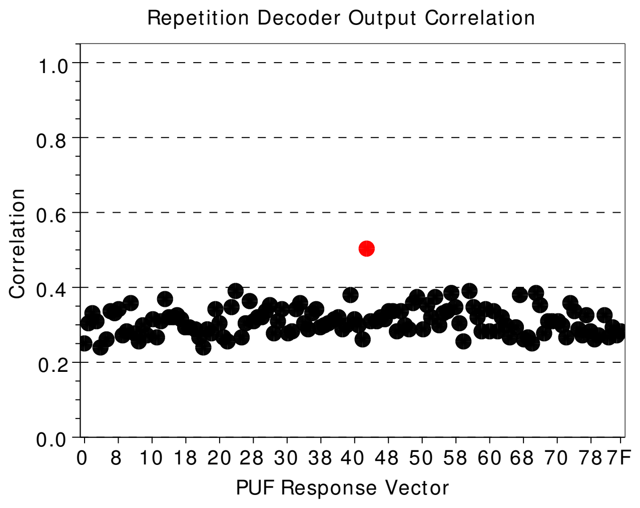


Figure 5.2: Maximum correlation of repetition decoder output

For a full attack on all 1778 PUF bits of my implementation, an attacker would have to perform the described analysis for all 254 repetition code decoder runs, which results in a minimum of $254 \times 128 = 32512$ traces to record and analyze.

5.3 Attack on Toeplitz Hashing

In this section, I present an SPA attack on the hardware implementation of Toeplitz hashing. This extractor algorithm was previously used for efficient COFE implementations. Also, I demonstrate the practical feasibility of the attack for an FPGA implementation.

5.3.1 Leakage of PUF Response Bits

Besides error correction, which is essential for COSSs and COFEs, also, an extractor algorithm is necessary for COFEs. Until now, the papers focusing on efficient COFE implementations, namely, Bösch et al. [BGS$^+$08] and Maes et al. [MTV09], make use of the LFSR-based Toeplitz hashing [Kra94]. It fulfills the requirements of redistributing the non-uniformly distributed PUF response bits [BGS$^+$08] to obtain a bit string that can be used as a cryptographic key.

Its basic hardware architecture is shown in Figure 5.3. When operating, first, an LFSR is initialized with a random start value, called helper data 2

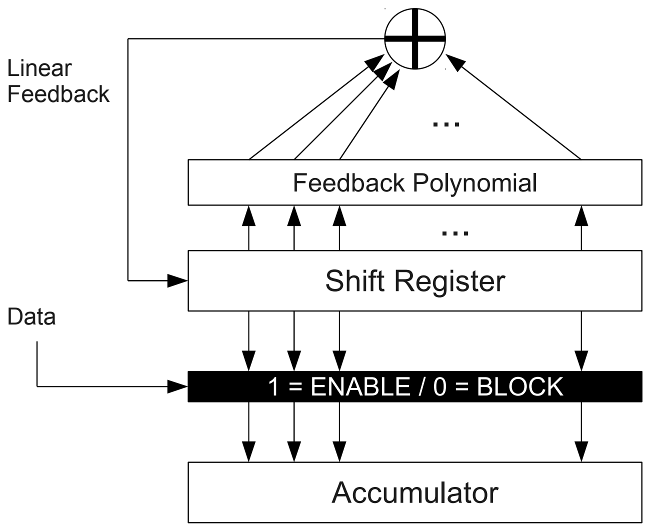


Figure 5.3: Efficient Toeplitz hashing implementation

by Bösch et al. [BGS+08]. Then, for every input bit, the LFSR is shifted once. If the input bit is a 1, then the value of the LFSR will be XORed into the accumulator register, if the input bit is 0, only the LFSR shift takes place without accumulation.

In COFEs, the input data of the Toeplitz hashing algorithm corresponds to regenerated and corrected PUF response bits $\mathbf{r}$. The resulting output bits represent a cryptographic key $\mathbf{k}$.

An attacker can exploit the fact that different operations, namely XOR accumulations and LFSR shifts, are performed by the LFSR-based hardware implementation, depending on the module's input data. This means, it is possible to extract the secret PUF response bits $\mathbf{r}$ from which the secret key $\mathbf{k}$ can be derived.

For a COFE, where an R-bit PUF response bit vector $\mathbf{r}$ is processed by a Toeplitz hashing extractor to obtain a secret key $\mathbf{k}$, the basic steps of my SPA attack are:

1. **Trace acquisition.** Collect several power consumption or EM emission traces of the Toeplitz hashing operation. One basic requirement is that the time slot of the extractor processing can be identified within the trace. Since the key reconstruction can be performed repeatedly, recording hundreds or thousands of traces of the same operation sequence is realistic.

2. **Noise cancellation.** Process the captured traces in order to obtain a low-noise trace that can be interpreted. Common processing steps include trace alignment and averaging.

3. **Point identification.** Determine the sequence of R samples that strongly depend on the input data, i.e., the secret PUF response bits.

4. **Bit extraction.** Divide the identified values into two groups, one for all points where an XOR accumulation takes place, representing a 1, and one for points without XOR accumulation, representing a 0.

Low-Noise Traces

For SPA attacks, usually, traces with a high SNR are required in order to enable their direct interpretation. In the case of COFEs, the Toeplitz hashing operation can be executed over and over again, which enables the collection of N_{tr} traces. These traces $\mathbf{t}_1, ..., \mathbf{t}_{N_{tr}}$ can then be averaged to obtain a mean trace $\bar{\mathbf{t}}$ with significantly lower noise level. However, because of several practical effects, collected traces are not perfectly aligned to achieve the best averaging result. Therefore, based on a significant trace sample $\hat{t}_n, n = 1, ..., N_{tr}$, e.g., a significant peak that has to be found in every trace, each trace has to be shifted by a value Δ_n to match the first trace.

$$\Delta_n = \hat{t}_n - \hat{t}_1 \; \forall \; n = 1, ..., N_{tr} \tag{5.4}$$

$$\bar{\mathbf{t}} = \frac{1}{N_{tr}} \sum_{n=1}^{N_{tr}} \mathit{shift}(\mathbf{t}_n, \Delta_n) \tag{5.5}$$

Peaks of Interest

If the averaged trace $\bar{\mathbf{t}}$ has a significantly high SNR, one is able to extract R trace values $v_1, ..., v_R$ corresponding to R bits sequentially processed by the Toeplitz hashing. This can be done by visual inspection of $\bar{\mathbf{t}}$, but also, e.g., by extracting the maximum value of each clock cycle during Toeplitz hashing operation.

$$\{v_1, ..., v_R\} \in \bar{\mathbf{t}} \tag{5.6}$$

PUF Response Extraction

The final step of this attack is the extraction of bits from the extracted values $v_1, ..., v_R$. For that, the threshold v_{th} is calculated as the mean of all values $v_1, ..., v_R$.

$$v_{th} = \frac{1}{R} \sum_{i=1}^{R} v_i \tag{5.7}$$

Then, all values v_i with a value above the threshold correspond to a 1 bit because a 1 at the input of the Toeplitz hashing causes additional power consumption generated by the XOR accumulation. Values below the calculated threshold indicate a 0 bit, respectively.

Thereby, the whole PUF response bit sequence $\mathbf{r} = \{r_1, ..., r_R\}$ can be extracted by inspection of the averaged trace. Having this sequence, an attacker is able to derive the secret key **k** by manually applying the Toeplitz hashing function on the obtained bits of **r**.

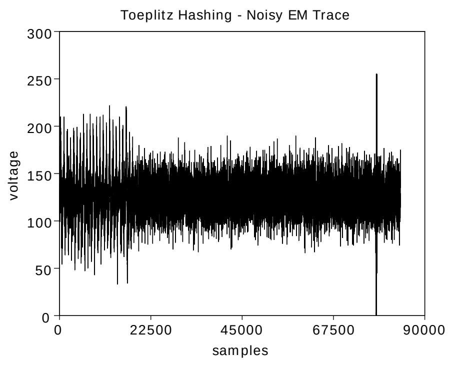
As for every attack, an attacker has to verify the correctness of the key by testing it against the real system. However, if he finds the key to be incorrect, the best way to search for the real key is to try other possible values for **r** by flipping bits r_i where the corresponding value v_i is close to the threshold v_{th}. This is reasonable because deriving bits from values close to the threshold is less reliable than for the ones further away.

5.3.2 SPA on Toeplitz Hashing FPGA Implementation

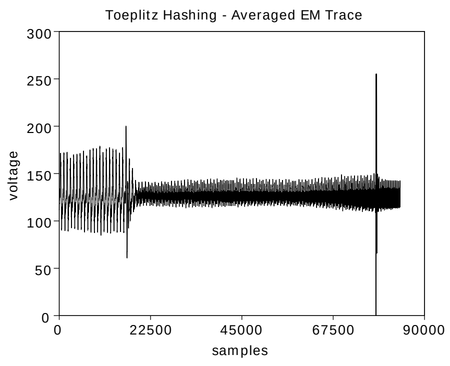
In order to verify the practical feasibility of the described SPA attack, I applied the above explained attack steps to a COFE implementation, where a 64-bit key **k** is generated from a error-corrected 74-bit PUF response vector **r**. The Toeplitz hashing implementation in the prototype under attack performs the LFSR shift and the XOR accumulation in two subsequent cycles, i.e., the compression of 74 bits takes 148 cycles.

For this experiment, the device under test was a Xilinx Spartan3E-1200 FPGA running at 50 MHz. I captured EM traces at the surface of the FPGA package with a Langer ICR HH 150 EM probe and a Langer PA 303 amplifier at a sample rate of 20 GS/s based on a LeCroy wavePro 715Zi oscilloscope. I used a trigger generated at the point in time when the key generation was finished to support this proof-of-concept attack.

Figure 5.4(a) shows a single EM trace that hardly permits to interpret anything except recognizing the trigger signal's influence, namely, the high peak in the right part of the trace, and the end of the error correction phase



(a) single noisy trace



(b) result after processing of 5000 traces

Figure 5.4: Noise reduction effect of alignment and averaging

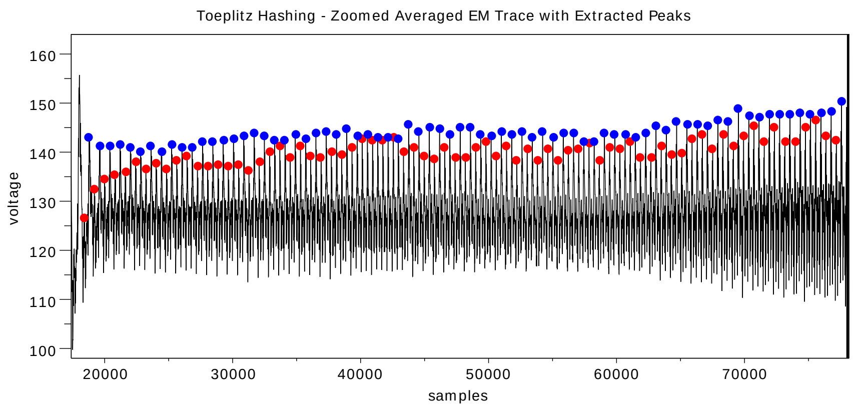


Figure 5.5: Toeplitz hashing trace with XOR (red) and shift operations (blue)

represented by the higher peaks in the left part of the trace. The Toeplitz hashing happens in between, but cannot be analyzed in this state.

After aligning 5000 traces with the help of the trigger influence peak and averaging them as described in Equation 5.5, the final trace, shown in Figure 5.4(b), exhibits a much higher SNR, which is a good basis for an SPA attack on the Toeplitz hashing.

After extracting the peak values $v_1, ..., v_R$ mentioned in Equation 5.6, shown as red dots in Figure 5.5, one can see that the recorded measurement traces show a drift towards higher values over the Toeplitz hashing operation.

This drift cannot be handled generally by the method explained in Section 5.3.1. In the case of my measurements, I was able to compensate the drift by also extracting the shift operation peaks $v_{1,ref}, ..., v_{R,ref}$, shown in blue, that are independent from the XOR operations, but are also subject to the drift. Then, for every XOR peak, I calculated the quotient $v_i/v_{i,ref}, i = 1, ..., R$, to lower the drifting effect by using the peaks caused by the shifting operation as reference values. The resulting values are depicted in Figure 5.6. The shown threshold value v_{th} was calculated according to Equation 5.7 and allows to interpret the collected values as bits of the secret PUF response vector $\mathbf{r}$.

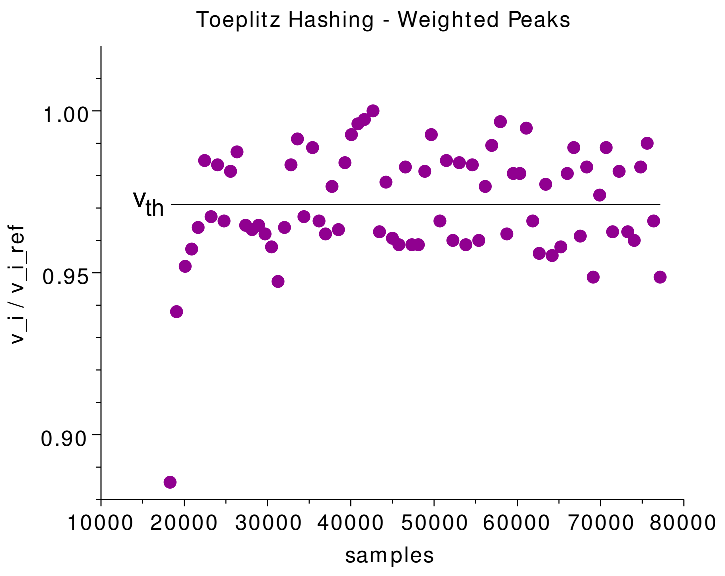


Figure 5.6: Weighted peaks, basis for bit interpretation

Table 5.1 shows that 3000 of my traces are enough to reduced their noise to a level where correct bit extraction becomes possible. However, even with only 500 traces, a bit extraction error of only 4.1% can be reached.

Table 5.1: Number of required traces for SPA attack

Traces	Bit Extraction Error
5000	0 (0%)
4000	0 (0%)
3000	0 (0%)
2000	1 (1.4%)
1000	1 (1.4%)
500	3 (4.1%)
250	5 (6.8%)
100	12 (16.2%)

Although I had to compensate a drifting effect in these measurements, other devices and implementations might not exhibit such an effect. Therefore, I cannot give a general rule how to handle these practical influences trace, but the general approach of the presented attack is valid.

5.4 Codeword Masking as a Countermeasure

A popular method to prevent DPA attacks is masking input values of an algorithm with random masks [CJRR99, GP99, MOP07] to destroy the dependency of intermediate and input values. However, this is not possible for ECCs, because a random mask would be regarded as a random error vector introducing (additional) errors to the original codeword. Therefore, trivial masking cannot be applied without loosing the ECC's essential error correction properties.

In this section, I propose a codeword masking scheme that can be applied to linear ECCs without having an impact on their error correction capabilities and is therefore not influencing the functionality of the original application, e.g., PUF key generation. First, I explain the principle of codeword masking. Then, I show how to apply it to a basic secure sketch architecture and I demonstrate its practical protection.

The linearity property [Bos99] of the functions encode_C and decode_C of a linear code C defines that the sum (XOR) of two codewords of C always results in another codeword of C. This also holds for concatenations of linear codes and represents the basis for my codeword masking scheme. I propose to mask an original codeword $\mathbf{c}$ of C by XORing it with the codeword mask $\mathbf{c_m}$, where $\mathbf{c_m}$ is a random codeword of C. The result of this boolean masking can still be decoded by decode_C to cancel bit errors, if present. Decoding $\mathbf{c_m}$ leaves the random bit vector $\mathbf{m}$ that corresponds to the 'raw' mask. This mask can then be used to demask the decoded result by XORing:

$$\mathit{decode}_C(\mathbf{c} \oplus \mathbf{c_m}) = \mathit{decode}_C(\mathbf{c}) \oplus \mathit{decode}_C(\mathbf{c_m}) = \mathit{decode}_C(\mathbf{c}) \oplus \mathbf{m} \qquad (5.8)$$

The relation between the original input codeword and the processed intermediate values is broken by this method while preserving the ECCs error correction features. Therefore, I propose this scheme to protect linear ECCs from DPA attacks like the one shown in Section 5.2.

Looking at the overhead of codeword masking, the straight forward approach would be duplicating the decoding module. However, I propose to generate a random mask $\mathbf{m}$ (used for demasking) and encode it to obtain $\mathbf{c_m}$. Thereby, only the encoding function encode_C has to be implemented, which

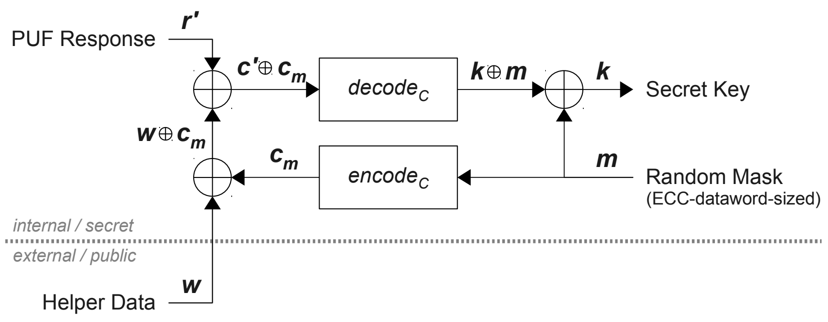


Figure 5.7: Masked secure sketch (*Rec*)

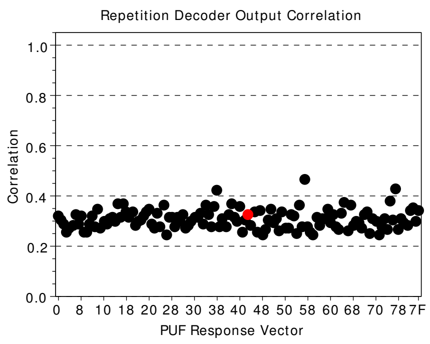
has a significantly lower implementation complexity than the decoding function *decode*$_C$ [Bos99]. Further, the encoding function is also required for the enrollment phase, which means that it can be reused for masking purposes resulting in almost no overhead.

5.4.1 Secure Sketch Protection

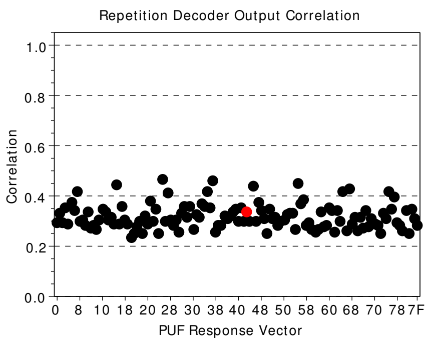
The application of the proposed technique to secure sketches is shown in Figure 5.7. There, the helper data $\mathbf{w}$ is masked by the masking codeword $\mathbf{c_m} = \mathit{encode}_C(\mathbf{m})$ of a random mask $\mathbf{m}$. The resulting ECC decoder input $\mathbf{w} \oplus \mathbf{c_m} \oplus \mathbf{r}'$ can be decoded to $\mathbf{k} \oplus \mathbf{m}$, which results in the generated key $\mathbf{k}$ after demasking with the raw mask $\mathbf{m}$.

For a practical evaluation, I extended the secure sketch implementation described in Section 5.2 by the codeword masking scheme shown in Figure 5.7. In detail, I added a 128-bit mask register (same size as key), which is first encoded by the BCH code encoder and then by the repetition code encoder, representing the encoding function for the concatenated ECC. I preloaded the mask register with random bits before each analysis run. The overhead in this proof-of-concept FPGA design sums up to 20% more slices, 45% more registers and 22% more look-up tables, which is mainly determined by the additional registers for the raw and the encoded codeword mask, but also by the additional BCH encoder. I regard these results as rather efficient, because the overhead is significantly below the numbers for AES, where masked FPGA implementations are reported to have around 60% slice overhead [KBG08]. Also, the overhead can be reduced even more, if the existing BCH encoder of the enrollment module is used for masking purposes.

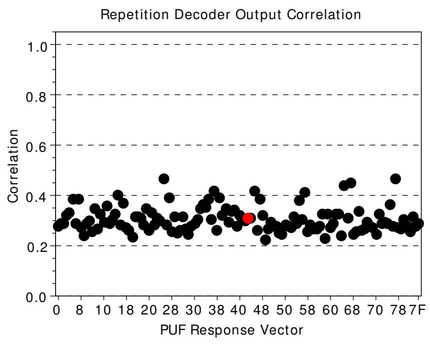
In order to evaluate the codeword masking countermeasure in practice, I performed the correlation-based first-order DPA attack described in Section 5.2 on the protected COSS implementation.



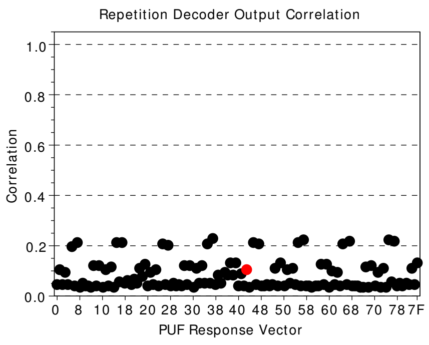
(a) 128 traces (1st dataset)



(b) 128 traces (2nd dataset)



(c) 128 traces (3rd dataset)



(d) 6400 traces

Figure 5.8: Maximum correlation of masked repetition decoder output

Figures 5.8(a), 5.8(b), and 5.8(c) show results for attacks on one repetition decoder run with three different sets of 128 traces covering all 128 possible helper data manipulations. They exhibit significant correlations, but neither for the correct PUF response vector (0x43, red mark), nor for the same value at all. An attack based on 6400 traces (50 repetitions of the 128 helper data manipulations) leads to the correlations shown in Figure 5.8(d). Although 6400 traces does not sound like a high number for DPA, note that this attack only targets 7 out of 1778 bits and, if the attack would be successful with 6400 traces, a total of $254 \times 6400 = 1625600$ traces would be required for a complete attack. However, also with 6400 traces, the correct PUF value is not distinguishable. I blame the visible patterns of significant correlations on the linearity of the repetition code implementation, but they do not reveal the secret PUF bits. This demonstrates the protective capabilities of the proposed masking scheme for PUF error correction against first-order DPA attacks. Higher-order attacks might be a threat to it, but were out-of-scope for this work.

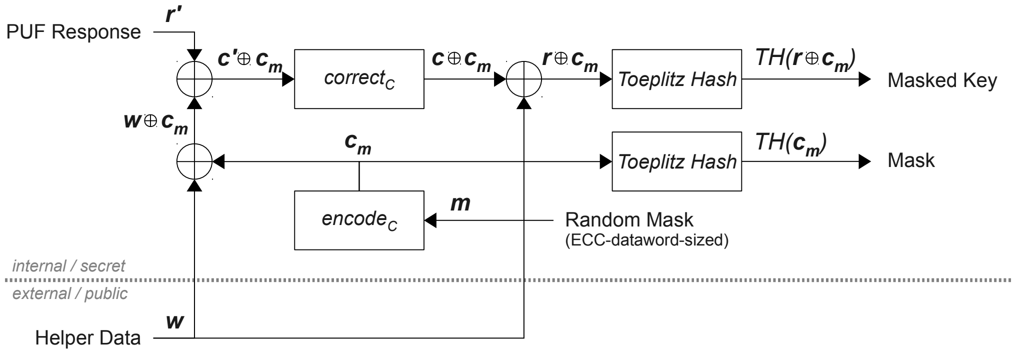


Figure 5.9: Masked COFE reconstruction module

5.4.2 COFE Masking

Since the error correction of the COFE architecture is very similar to the one of a secure sketch, it is also vulnerable to the attack shown in Section 5.2. Additionally, the second processing of helper data **w** allows to manipulate intermediate values of the extractor module *Ext* and, thereby, to mount extractor DPA attacks based on helper data manipulations. The double influence of helper data **w** is shown in the equation of a code-offset fuzzy extractor's reproduction phase:

$$\mathbf{k} = Ext(correct_C(\mathbf{r}' \oplus \mathbf{w}) \oplus \mathbf{w}) \tag{5.9}$$

With the proposed codeword masking scheme, it is possible to consistently mask and, thereby, protect fuzzy extractor architectures. The application to the error correction module is identical as for secure sketches, but in the case of a COFE, the error correction output, i.e., the masked PUF response $\mathbf{r} \oplus \mathbf{c_m}$, is fed into the Toeplitz hashing *TH*, as shown in Figure 5.9. The encoded mask is also processed by the Toeplitz hashing. In this case, the linearity property of this hash algorithm is of advantage. Its final result is a linear combination of LFSR states, because the state of the Toeplitz hashing LFSR is XORed into an accumulator depending on the algorithm's input data. This leads to a hashed masking codeword $TH(\mathbf{c_m})$ that represents the correct mask for the masked key $TH(\mathbf{r} \oplus \mathbf{c_m})$:

$$TH(\mathbf{r} \oplus \mathbf{c_m}) = TH(\mathbf{r}) \oplus TH(\mathbf{c_m}) = \mathbf{k} \oplus TH(\mathbf{c_m}) \tag{5.10}$$

The resulting pair of masked key and corresponding mask can be used to resolve the masking, if required, but can also be provided to the subsequent crypto module leading to a consistently masked COFE from the first helper data input to the subsequent cryptographic algorithm.

By implementing this masked COFE architecture, the power consumption and intermediate values of a COFE are randomized. This also protects devices from the SPA attack shown in Section 5.3 because the Toeplitz hashing processes masked PUF response bits.

5.4.3 Robust Sketches and Robust Fuzzy Extractors

Boyen et al. [BDK+05] proposed robust sketches and robust fuzzy extractors in 2005, as introduced in Section 2.3. While their constructions are secure for remote scenarios, I want to stress that this does not hold for a physical attacker who is able to observe side-channel leakage. The reason for that is that the operations of standard secure sketches and fuzzy extractors that cause exploitable side-channel leakage still need to be performed for the proposed robust architectures before a decision can be made whether the helper data was manipulated or not. Therefore, observing side-channels enables an attacker to extract information about intermediate results, even before a robust sketch or a robust fuzzy extractor detects the manipulation. As a result, also robust COSSs and COFEs need to be protected against side-channel attacks, e.g., by the proposed codeword masking.

5.4.4 Masking Other PUF Key Generation Algorithms

The main part of this chapter is based on code-offset algorithms, however, I want to emphasize that similar DPA attacks are also applicable to other PUF key generation algorithms, e.g., IBS [YD10b] and its optimized extension C-IBS [HMSS12] as shown in Figure 5.10.

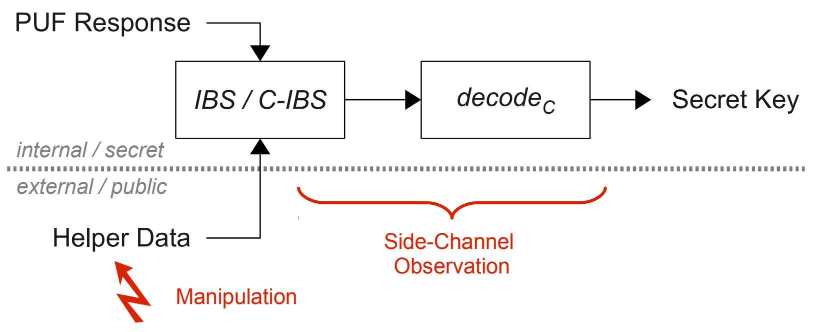


Figure 5.10: Attacking IBS-based key generation

IBS stores indices of reliable PUF output bits as helper data, which are used to select the most reliable bits during reconstruction. C-IBS additionally stores indices of reliable bits with complementary values to increase reliability. Both constructions are usually supported by an ECC implementation to achieve lower residual error probabilities.

Regarding DPA attacks, there is a slight differences compared to code-offset algorithms. While flipping a bit in code-offset helper data directly leads to a flipped bit of the ECC input data, for IBS, an attacker has to guess (from a small number of choices) a valid index of an inverted PUF bit to achieve the necessary bit flip. For C-IBS, again, an attacker can manipulate single bits easily, since exchanging original and complementary indices in the helper data deterministically causes a bit flip.

In order to protect the error correction module of IBS and C-IBS implementations, I propose to apply a random codeword mask to PUF bits selected by IBS/C-IBS before processing them by ECCs, as shown in Figure 5.11. Afterwards, the masked key can be demasked or forwarded to the subsequent crypto module.

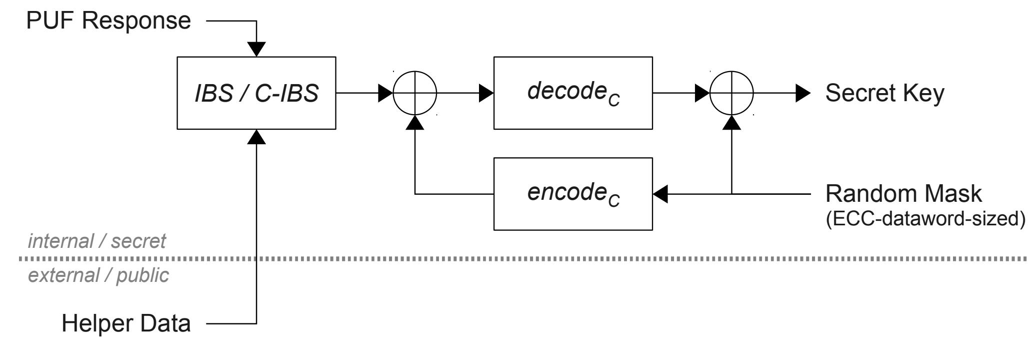


Figure 5.11: Masking of ECCs in IBS architecture

5.5 Summary

I proposed a DPA and an SPA attack on COFE implementations, which can compromise the secret key generated based on secret PUF response bits.

DPA attacks on COSSs and COFEs can be performed by manipulating the code-offset helper data, which is processed with the secret PUF bits during key reconstruction. Thereby, intermediate values of the error correction modules can be influenced leading to leakage, which can be exploited by DPA attacks. I explained the necessary steps to perform the described attack and showed that this attack vector is feasible in practice.

The Toeplitz hashing, usually used for efficient fuzzy extractor implementations, exhibits a power consumption and EM emission that depends on the secret PUF bits fed into the extractor algorithm. If an attacker is able to record several traces during the operation of this algorithm and manages to reduce the contained noise to a minimum, he can extract all PUF bit values. I described the vulnerability of Toeplitz hashing that can be exploited by attackers and demonstrated the attack for an FPGA implementation.

In order to protect COFEs from the described attacks, I proposed a codeword masking scheme that enables randomization of the power consumption of ECCs. I showed that its implementation can be efficient because the duplication of an ECC's decoder is not necessary, but only an additional encoder is needed. Also, the proposed scheme can forward the generated key in a masked fashion, which enables DPA protection from PUF response and helper data processing up to the crypto module, where the secret key is used.

Summarizing, I presented two attacks on COFEs that represent significant attack vectors posing a serious threat on PUF-based key generation. Codeword masking can be an important step towards SCA-resistant COFE implementations and, as shown, can also be applied to other key generation algorithms.

Chapter 6

Conclusions

In this thesis, I investigated vulnerabilities and protection techniques to enhance the security of RO PUFs and COFEs against physical attacks.

First, I discussed the obstacles of solid RO PUF implementations on FPGAs. I conclude that designers of RO PUFs, and PUFs in general, have to focus on best-possible physical implementations in terms of equality and local placement. Only then, reliable and secure PUFs can be created that can serve as a high-quality source for device-specific bit extraction.

As a basic step before evaluating vulnerabilities of RO PUFs, I showed that removing an FPGA's chip package does not significantly influence RO frequencies. This paves the way for any semi-invasive attack on RO PUFs. Following, I showed that standard RO PUFs are prone to EM emission analysis and proposed an improved architecture. However, with high-precision equipment even the protected implementation can be broken by exploiting localized EM emission. Therefore, I proposed a randomized measurement scheme to break the relation between ROs and used measurement counters to protect RO PUFs. Further, I suggested to used interleaved placement techniques to achieve a higher level of practical security in cases where the rather high overhead of the randomization countermeasure is not acceptable. In conclusion, RO PUFs as originally proposed are not secure when assuming a physical attacker. However, as shown in this thesis, there are ways to achieve secure RO PUF implementations even when facing physical attacks.

In the second part of this thesis, I showed that an attacker does not necessarily have to attack the underlying PUF structure, but can also mount side-channel attacks on key generation modules like COFEs. They can leak information that reveals the secret key during its generation. I showed that well-known attack concepts, like SPA and DPA, can be adapted and successfully applied to the PUF key generation use case. Further, I presented a protection scheme that enables SPA and first-order DPA protection for code-offset algorithms, but can also be applied to other key generation schemes. I conclude, that also COFE implementations show exploitable vulnerabilities. Since side-channel attacks can be performed with even low-cost measurement equipment, attacking the key generation modules is probably the first target of an attacker. Here, my work is the first approach to consistently protect PUF-based key generation against SCA from its first processing stage to its output.

I finally conclude that I discovered several vulnerabilities of RO PUFs and code-offset algorithms and proposed suitable countermeasures. Thereby, I contributed to the development of attack-resistant PUF systems.

Future Work

My work shows that RO PUFs are not secure against physical attacks. I can imagine that this is also true for other PUF architectures. Therefore, I see it as essential to perform similar analyses on other promising PUF structures, such as the SRAM PUF, the optical smartcard PUF, and the twisted bistable ring PUF. In my opinion, randomized PUF measurement processes are a key countermeasure to withstand future physical attacks.

Further, other popular key generation systems, such as IBS and C-IBS have to be investigated to understand, if they show specific vulnerabilities regarding helper data manipulation and side-channel exploitation. Also, the implementation efficiency and implementation security of codeword masking with specific hard- and soft-decision codes can be valuable future work.

A third field of future work is the development of more sophisticated attacks on COFEs, e.g., without the necessity to manipulate helper data. Also, other physical attacks, such as laser fault injection, have not been considered for PUFs yet, but might reveal further vulnerabilities which have to achieve secure PUF applications.

Bibliography

[BBA$^+$12] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In *Proceedings of the Third international conference on Constructive Side-Channel Analysis and Secure Design*, COSADE'12, pages 151–166, Berlin, Heidelberg, 2012. Springer-Verlag.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 135–152. Springer Berlin / Heidelberg, 2004.

[BDHV07] Ileana Buhan, Jeroen Doumen, Pieter Hartel, and Raymond Veldhuis. Constructing practical fuzzy extractors using QIM. Technical Report 07-52, University of Twente, CTIT, Enschede, Netherlands, 2007.

[BDK$^+$05] Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 561–561. Springer Berlin / Heidelberg, 2005.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT*, pages 37–51, 1997.

[BGS$^+$08] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on FPGAs. In *CHES '08: Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 181–197, Berlin, Heidelberg, 2008. Springer-Verlag.

[Bos99] Martin Bossert. *Channel Coding for Telecommunications.* Wiley, 1999.

[BP09] N. Beckmann and M. Potkonjak. Hardware-based public-key cryptography with public physically unclonable functions. In *Information Hiding*, pages 206–220. Springer, 2009.

[CCL$^+$11] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair. The bistable ring PUF: A new architecture for strong physical unclonable functions. In *IEEE Int. Symposium on Hardware-Oriented Security and Trust*, June 2011.

[CCL$^+$12] Qingqing Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Ruhrmair. Characterization of the bistable ring PUF. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1459 –1462, march 2012.

[CJRR99] Suresh Chari, Charanjit Jutla, Josyula Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 791–791. Springer Berlin / Heidelberg, 1999.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2 edition, November 2001.

[DK07] G. DeJean and D. Kirovski. RF-DNA: Radio-frequency certificates of authenticity. *Cryptographic Hardware and Embedded Systems (CHES)*, pages 346–363, 2007.

[DMV04] C. Desset, B. Macq, and L. Vandendorpe. Computing the word-, symbol-, and bit-error rates for block error-correcting codes. *Communications, IEEE Transactions on*, 52(6):910–921, 2004.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of

Lecture Notes in Computer Science, pages 523–540. Springer Berlin / Heidelberg, 2004.

[DW09] Jianwei Dai and Lei Wang. A study of side-channel effects in reliability-enhancing techniques. In *Proceedings of the 2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, DFT '09, pages 236–244, Washington, DC, USA, 2009. IEEE Computer Society.

[EFK$^+$12] Thomas Esbach, Walter Fumy, Olga Kulikovska, Dominik Merli, Dieter Schuster, and Frederic Stumpf. A new security architecture for smartcards utilizing PUFs. In *Proceedings of the 14th Information Security Solutions Europe Conference (ISSE'12)*. Springer Vieweg, 2012.

[FMC85] R. C. Fairfield, R. L. Mortenson, and K. B. Coulthart. An LSI random number generator (RNG). In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 203–230, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[For65] George David Forney, Jr. *Concatenated Codes*. PhD thesis, Massachusetts Institute of Technology, 1965.

[GCvDD02] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 148–160, New York, NY, USA, 2002. ACM.

[GCvDD03] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Delay-based circuit authentication and applications. In *Symposium on Applied Computing (SAC)*, 2003.

[GKST07] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.

[GLC$^+$04] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience*, 16(11):1077–1098, 2004.

[GM89] S. N. Graybeal and P. B. McFate. Getting out of the starting block. *Scientific American (USA)*, 261(6), 1989.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis the "duplication" method. In Cetin Koc and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, volume 1717 of *Lecture Notes in Computer Science*, pages 728–728. Springer Berlin / Heidelberg, 1999.

[GŠT$^+$09] Jorge Guajardo, Boris Škorić, Pim Tuyls, Sandeep S. Kumar, Thijs Bel, Antoon H. Blom, and Geert J. Schrijen. Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions. *Information Systems Frontiers*, 11(1):19–41, March 2009.

[HdlTR12] Wei He, Eduardo de la Torre, and Teresa Riesgo. An interleaved EPE-immune PA-DPL structure for resisting concentrated EM side channel attacks on FPGA implementation. In *Proceedings of the 3rd international conference on Constructive Side-Channel Analysis and Secure Design*, COSADE'12, pages 39–53, Berlin, Heidelberg, 2012. Springer.

[HDS09] G. Hammouri, A. Dana, and B. Sunar. CDs have fingerprints too. *Cryptographic Hardware and Embedded Systems (CHES)*, pages 348–362, 2009.

[HMH$^+$12a] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In Orr Dunkelman, editor, *Topics in Cryptology CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 231–244. Springer Berlin / Heidelberg, 2012.

[HMH$^+$12b] Johann Heyszl, Dominik Merli, Benedikt Heinz, Fabrizio De Santis, and Georg Sigl. Strengths and limitations of high-resolution electromagnetic field measurements for side-channel analysis. In Stefan Mangard, editor, *Smart Card Research and*

Advanced Applications, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012.

[HMSS12] M. Hiller, D. Merli, F. Stumpf, and G. Sigl. Complementary IBS: Application specific error correction for PUFs. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, pages 1 –6, june 2012.

[HNBJP13] Clemens Helfmeier, Dmitry Nedospasov, Christian Boit, and Seifert Jean-Pierre. Cloning physically unclonable functions. In *Proceedings of the IEEE Int. Symposium of Hardware-Oriented Security and Trust.* IEEE, June 2013.

[HYKS10] Yohei Hori, Takahiro Yoshida, Toshihiro Katashita, and Akashi Satoh. Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs*, RECONFIG '10, pages 298–303, Washington, DC, USA, 2010. IEEE Computer Society.

[Jam97] Ernest Jamro. The design of a VHDL based synthesis tool for BCH codecs. Master's thesis, School of Engineering, The University of Huddersfield, Sep 1997.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, CCS '99, pages 28–36, New York, NY, USA, 1999. ACM.

[KBG08] N. Kamoun, L. Bossuet, and A. Ghazel. SRAM-FPGA implementation of masked s-box based DPA countermeasure for AES. In *Design and Test Workshop, 2008. IDT 2008. 3rd International*, pages 74–77, 2008.

[KGM$^+$08] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 67–70, June 2008.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *WOST'99: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, pages 2–2, Berkeley, CA, USA, 1999. USENIX Association.

[KKR+12] Stefan Katzenbeisser, Ünal Kocabas, Vladimir Rozic, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions(PUFs) cast in silicon. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 283–301. Springer Berlin Heidelberg, 2012.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, 1996. Springer-Verlag.

[Kra94] Hugo Krawczyk. LFSR-based hashing and authentication. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–139, London, UK, 1994. Springer-Verlag.

[KS10] D. Karakoyunlu and B. Sunar. Differential template attacks on PUF enabled cryptographic devices. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1 –6, dec 2010.

[LCNT02] P. Layman, S. Chaudhry, J.G. Norman, and J.R. Thomson. Electronic fingerprinting of semiconductor integrated circuits. US Patent 6,738,294, September 2002.

[LLG+04] J-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits with identification and authentication applications. In *IEEE VLSI Circuits Symposium*, New-York, June 2004.

[LLG+05] Daihyun Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits.

Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 13(10):1200–1205, December 2005.

[MCMS10] Abhranil Maiti, Jeff Casarona, Luke McHale, and Patrick Schaumont. A large scale characterization of RO-PUF. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 66–71, 2010.

[MEK10] M. Majzoobi, A. Elnably, and F. Koushanfar. FPGA time-bounded unclonable authentication. In *Information Hiding*, pages 1–16. Springer, 2010.

[MGS11] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. Cryptology ePrint Archive, Report 2011/657, 2011. http://eprint.iacr.org/2011/657.

[MHH$^+$13] Dominik Merli, Johann Heyszl, Benedikt Heinz, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of RO PUFs. In *Proceedings of the IEEE Int. Symposium of Hardware-Oriented Security and Trust.* IEEE, June 2013.

[MKP08a] M. Majzoobi, F Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *Test Conference, 2008. ITC 2008. IEEE International*, pages 1–10, 2008.

[MKP08b] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure PUFs. In *ICCAD '08: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 670–673, Piscataway, NJ, USA, 2008. IEEE Press.

[MMS09] Sergey Morozov, Abhranil Maiti, and Patrick Schaumont. A comparative analysis of delay based PUF implementations on FPGA. Cryptology ePrint Archive, Report 2009/629, 2009. http://eprint.iacr.org/2009/629.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[MP13] Dominik Merli and Rainer Plaga. Was ist der mehrwert von physical unclonable functions in der chipsicherheit? In *13. Deutscher IT Sicherheitskongress*, 2013.

[MS09] Abhranil Maiti and Patrick Schaumont. Improving the quality of a physical unclonable function using configurable ring oscillators. In *19th International Conference on Field Programmable Logic and Applications (FPL), 2009. FPL '09.*, 2009.

[MS11] Marcel Medwed and Franois-Xavier Standaert. Extractors against side-channel attacks: Weak or strong? In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 256–272. Springer Berlin / Heidelberg, 2011.

[MS13] Dominik Merli and Dieter Schuster. Physical unclonable function. European patent application, EP13153481, 2013.

[MSE10] Dominik Merli, Frederic Stumpf, and Claudia Eckert. Improving the quality of ring oscillator PUFs on FPGAs. In *5th Workshop on Embedded Systems Security (WESS'2010)*, Scottsdale, AZ, USA, October 2010. ACM.

[MSS13] Dominik Merli, Frederic Stumpf, and Georg Sigl. Protecting PUF error correction by codeword masking. Cryptology ePrint Archive, Report 2013/334, 2013. http://eprint.iacr.org/2013/334.

[MSSS11a] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-invasive EM attack on FPGA RO PUFs and countermeasures. In *6th Workshop on Embedded Systems Security (WESS'2011)*, Taipei, Taiwan, October 2011. ACM.

[MSSS11b] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-channel analysis of PUFs and fuzzy extractors. In *4th International Conference on Trust and Trustworthy Computing (TRUST2011)*, Pittsburgh, PA, USA, June 2011. Springer.

[MSSS11c] H. Molter, Marc Stöttinger, Abdulhadi Shoufan, and Falko Strenzke. A simple power analysis attack on a McEliece cryptoprocessor. *Journal of Cryptographic Engineering*, 1:29–36, 2011.

[MTV08] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic PUFs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security (WISSec 2008)*, page 17, Eindhoven,NL, 2008.

[MTV09] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2009.

[Pap01] Ravikanth Pappu. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, 2001.

[PD11] Zdenek Sid Paral and Srinivas Devadas. Reliable and efficient PUF-based key generation using pattern matching. In *HOST*, pages 128–133. IEEE Computer Society, 2011.

[PRTG02] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, September 2002.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smard Cards. In *Smart Card Programming and Security (E-smart 2001)*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag, 9 2001.

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

[Rüh09] Ulrich Rührmair. Simpl systems: On a public key variant of physical unclonable functions. Technical report, Cryptology ePrint Archive, International Association for Cryptologic Research, 2009.

[SD07] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 9–14, 2007.

[SGM09] Laurent Sauvage, Sylvain Guilley, and Yves Mathieu. Electromagnetic radiations of FPGAs: High spatial resolution cartography and attack on a cryptographic module. *ACM Trans. Reconfigurable Technol. Syst.*, 2:4:1–4:24, March 2009.

[SHO08] Ying Su, Jeremy Holleman, and Brian P. Otis. A digital 1.6 pj/bit chip identification circuit using process variations. *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, 43(1):69–77, Jan 2008.

[Sko05] Sergei P. Skorobogatov. Semi-invasive attacks – A new approach to hardware security analysis. Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005.

[Sko10] Sergei Skorobogatov. Optical fault masking attacks. In Luca Breveglieri, Marc Joye, Israel Koren, David Naccache, and Ingrid Verbauwhede, editors, *FDTC*, pages 23–29. IEEE Computer Society, 2010.

[Smi97] Steven W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, San Diego, CA, USA, 1997.

[SNK$^+$12] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple photonic emission analysis of AES. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems, CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 41–57. Springer Berlin Heidelberg, 2012.

[SS10] D. Suzuki and K. Shimizu. The glitch PUF: A new delay-PUF architecture exploiting glitch shapes. *Cryptographic Hardware and Embedded Systems (CHES) 2010*, pages 366–382, 2010.

[STM$^+$08] Falko Strenzke, Erik Tews, H. Molter, Raphael Overbeck, and Abdulhadi Shoufan. Side channels in the McEliece PKC. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 216–229. Springer Berlin / Heidelberg, 2008.

[SVW10] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. PUF-enhanced RFID security and privacy. In *Workshop on Secure Component and System Identification (SECSI)*, 2010.

[TSŠ$^+$06] Pim Tuyls, Geert-Jan Schrijen, Boris Škorić, Jan van Geloven, Nynke Verhaegh, and Rob Wolters. Read-proof hardware from protective coatings. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, chapter 29, pages 369–383. Springer Berlin Heidelberg, 2006.

[Wag12] Mathias Wagner. 700+ attacks published on smart cards: The need for a systematic counter strategy. In *COSADE*, pages 33–38, 2012.

[Xil10] Xilinx Inc. *Spartan-3A FPGA Family Data Sheet - DS529*, v2.0 edition, 2010. http://www.xilinx.com/ support/documentation/data_sheets/ds529.pdf.

[Xil12a] Xilinx Inc. *Spartan-3 FPGA Family Data Sheet - DS099*, v3.0 edition, 2012. http://www.xilinx.com/ support/documentation/data_sheets/ds099.pdf.

[Xil12b] Xilinx Inc. *Spartan-3E FPGA Family Data Sheet - DS312*, v4.0 edition, 2012. http://www.xilinx.com/ support/documentation/data_sheets/ds312.pdf.

[YD10a] Meng-Day (Mandel) Yu and Srinivas Devadas. Recombination of physical unclonable functions. In *35th Annual GOMACTech Conference*, Reno, NV, March 2010. United States. Dept. of Defense.

[YD10b] Meng-Day (Mandel) Yu and Srinivas Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Des. Test*, 27(1):48–65, 2010.

[YMSD11] Meng-Day (Mandel) Yu, David M'Raihi, Richard Sowell, and Srinivas Devadas. Lightweight and secure PUF key storage using limits of machine learning. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems (CHES 2011)*, volume 6917 of *Lecture Notes in Computer Science*, pages 358–373. Springer Berlin Heidelberg, 2011.

[YQ10] Chi-En Yin and Gang Qu. Lisa: Maximizing RO PUF's secret extraction. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 100–105, 2010.

[YQ11] Chi-En Yin and Gang Qu. A regression-based entropy distiller for RO PUFs. Technical report, University of Maryland, Institute for Systems Research, 2011.

[YSS$^+$12] Meng-Day Yu, R. Sowell, A. Singh, D. M'Raihi, and S. Devadas. Performance metrics and empirical results of a PUF cryptographic key generation asic. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, pages 108 –115, june 2012.