TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Sicherheit in der Informationstechnik

Key Derivation with Physical Unclonable Functions

Matthias Hiller

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. sc. techn. Gerhard Kramer
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Georg Sigl
	2.	Prof. Dr.-Ing. Martin Bossert

Die Dissertation wurde am 04.07.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.11.2016 angenommen.

Abstract

Secure cryptographic keys are a prerequisite to protect the data that is transmitted and stored by today's and tomorrows embedded systems in the Internet of Things. For many of those systems, secure Non-Volatile Memory (NVM) is not available due to technical and cost constraints. Silicon Physical Unclonable Functions (PUFs) evaluate manufacturing variations to generate unique secrets inside Integrated Circuits (ICs) to replace the functionality of the NVM. However, PUFs are affected by noise and changes in environmental conditions, so that PUF responses cannot be directly used as cryptographic keys.

Key derivation algorithms turn a noisy PUF response into a reliable cryptographic key. Error correction is required to remove the variation caused by environmental effects and random noise, and derive stable cryptographic keys. Helper data enables error correction by mapping random PUF responses to codewords of Error-Correcting Codes (ECCs), where errors can be detected and corrected. Over the last 10 years, several helper data generation approaches were introduced and implemented. Some challenges of key derivation with PUFs are the lacking of a generic theoretical framework and optimal practical ways of generating helper data. Further, today's hardware implementations focus on small block lengths in error correction for a low complexity.

This thesis addresses these open points by showing that the practical problem of key derivation with PUFs is closely related to the information theoretical problem of key generation from compound sources. It also provides an algebraic representation that applies to a wide class of previous work on error correction for PUFs. The new representation allows to upper bound the secrecy leakage of an approach already on the algorithmic level during the design stage. The analysis shows that today's algorithms do not support maximum key rates and minimum helper data sizes, while causing no helper data leakage at the same time.

Systematic Low Leakage Coding (SLLC) is a new error correction scheme presented in this thesis that translates the properties of theoretical random coding approaches to a deterministic code generation scheme. It maximizes the size of the key and minimizes the size of the helper data without causing secrecy leakage through the helper data. In addition, implementation parameters for an extremely low-area error correction for PUFs with very low bit error probability are provided.

Not all PUF response bits are equally stable. This work shows with the information theoretical concept of typicality that the error correction overhead decreases significantly by selecting reliable PUF response bits in larger blocks. Differential Sequence Coding (DSC) is introduced as an error correction approach that creates one single block of reliable PUF response bits and thus overcomes the limitations of small block sizes of previous implementations. This work uses DSC in combination with the popular code class of convolutional codes. The DSC and Viterbi decoder error correction module reduces the size of the PUF and the helper data significantly for a popular reference

scenario for SRAM PUFs and Xilinx Spartan 3 FPGAs without increasing the slice count of the implementation.

The final comparison shows that different error correction candidates are favorable for different applications, depending on their requirements and constraints. This work expands the state of the art by providing more efficient solutions for various parameter sets.

Keywords : Physical Unclonable Functions, Key derivation, Fuzzy Extractor, Syndrome Coding, Error-Correcting Codes, Information Theory, Hardware Implementation, FPGA.

Kurzfassung

Sichere kryptografische Schlüssel sind eine Grundvoraussetzung, um Daten zu schützen, die von heutigen und zukünftigen einbegetteten Systemen im Internet der Dinge gespeichert, verarbeitet und übertragen werden. Sichere Nicht-flüchtige Speicher sind aufgrund technischer oder wirtschaftlicher Beschränkungen in vielen Systemen nicht verfügbar. Silicon Physical Unclonable Functions (PUFs) werten Fertigungsschwankungen aus, um einzigartige Geheimnisse in integrierten Schaltungen zu erzeugen und nichtflüchtige Speicher zu ersetzen. Die so-genannten PUF Responses werden von Rauschen und physikalischen Umgebungsgrößen beeinflusst, sodass sie nicht direkt als stabile kryptografische Schlüssel verwendet werden können.

Algorithmen zur Schlüsselableitung verarbeiten die PUF Response zu einem zuverlässigen kryptografischen Schlüssel. Fehlerkorrekturverfahren werden dabei benötigt, um die Abweichungen, die durch das Rauschen und die Umgebungseffekte entstanden sind, zu entfernen. Helperdaten ermöglichen die Fehlerkorrektur, indem sie die zufälligen PUF Responses auf Codewörter von fehlerkorrigierenden Codes abbilden in denen Fehler erkannt und korrigiert werden können. In den letzten 10 Jahren wurden zahlreiche Verfahren zur Helperdatenerzeugung vorgestellt und implementiert. Das Fehlen von allgemeinen theoretischen Modellen und optimalen praktischen Verfahren zur Helperdatenerzeugung, sowie das Bestreben möglichst kleine Fehlerkorrekturblöcke zu nutzen, um Hardwareimplementierungen klein zu halten sind Herausforderungen bei der Schlüsselableitung mit PUFs.

Bezüglich der theoretischen Modelle zeigt diese Arbeit, dass das informationstheoretische Problem der Schlüsselerzeugung aus Compound Quellen eng mit Schlüsselableitung aus PUFs verwandt ist. Außerdem wird eine generische algebraische Darstellung diskutiert, mit der eine große Zahl vorhandener Ansätze zur Schlüsselableitung dargestellt werden kann. Die neue Darstellung ermöglicht es, für einen Ansatz auf algorithmischer Ebene eine obere Schranke für die Schlüsselinformation, die maximal durch die Helperdaten nach Außen gelangt, anzugeben.

Systematic Low Leakage Coding (SLLC) ist ein in dieser Arbeit entwickelter neuer Fehlerkorrekturansatz, der die Eigenschaften von theoretischen Verfahren mit Zufallscodes zeigt und gleichzeitig durch deterministische Codegenerierung auch praktisch umgesetzt werden kann. Es maximiert er die Schlüsselgröße und minimiert gleichzeitig die Größe der benötigten Helperdaten. Außerdem werden Implementierungsparameter für eine Lightweight-Implementierung für PUFs mit sehr niedriger Fehlerwahrscheinlichkeit diskutiert.

Nicht alle PUF-Response-Bits sind gleich stabil. Deshalb zeigt diese Arbeit mithilfe des informationstheoretischen Konzepts typischer Sequenzen, dass die Fehlerkorrektur für große Blöcke ausgewählter PUF-Response-Bits deutlich effizienter ist. Differential Sequence Coding (DSC) ist ein in dieser Arbeit entwickelter Fehlerkorrekturansatz, der auf einem einzigen Block mit zuverlässigen PUF-Response-Bits arbeitet und deshalb

die Beschränkung kleiner Blockgrößen überwindet. In dieser Arbeit wird DSC zusammen mit Faltungscodes eingesetzt. Das Fehlerkorrekturmodul mit DSC und Viterbi-Decodierer reduziert die Anzahl benötigter PUF-Response-Bits und die Helperdategröße deutlich für ein typisches Referenzszenario mit SRAM PUF und Xilinx Spartan 3E FPGA. Dabei wird die Größe der Implementierung nur unwesentlich erhöht.

Der abschließende Vergleich zeigt, dass unterschiedliche Fehlerkorrekturansätze je nach Vorgaben und Einschränkungen für unterschiedliche Anwendungen geeignet sind. Dabei erweitert diese Arbeit den Stand der Technik um neue, effizientere, Verfahren für unterschiedliche Anwendungen und Parameter.

Schlüsselworte : Physical Unclonable Functions, Schlüsselableitung, Fuzzy Extractor, Syndrome Coding, Kanalcodierung, Informationstheorie, Hardwareimplementierung, FPGA.

Acknowledgments

The last four and a half years at TUM were full of exciting new experiences and challenges, where no year was like the one before. Moving to a new university, working on a topic for several years and having the time to look at it from several different perspectives, having deep technical discussions, establishing new courses, advising students, publishing with people from different groups and fields, going to conferences, having research stays abroad, getting settled and finding new friends in Munich, ... and all of this would not have been possible without several people who I would like to thank:

First of all I would like to thank Prof. Dr.-Ing. Georg Sigl for giving me the chance to do my PhD at his Chair of Security in Information Technology at TUM, giving me the freedom to work on the ideas that interested me most, for providing me guidance when needed, for opening doors and enabling opportunities. Collaborating with various people from other academic groups, Fraunhofer and industry was a great benefit and also being involved in teaching was an enriching personal experience.

Further, I would like to thank Prof. Dr.-Ing. Martin Bossert for being member of the dissertation committee and for carefully reading this dissertation. He was involved in several steps of my university education starting when I attended his lecture *Signale und Systeme* in Ulm in 2007 and I would also like to thank him for the collaboration over the last years.

Also, I would like to thank Prof. Dr. sc. techn. Gerhard Kramer for heading the dissertation committee and for the inspiring discussions in the PUF COM Cluster and beyond.

I would like to thank Dr.-Ing. Michael Pehl for the endless discussions, developing new ideas, collaborating closely, sharing his knowledge on algebra and statistics, and also for handling all kinds of administrative issues. I appreciate his feedback after carefully proof-reading this dissertation.

I thank my office mates Fabrizio De Santis, Michael Weiner, Florian Wilde and Johanna Baehr for the good times and collaboration and also all other members of the Chair of Security in Information Technology. I would like to express some special thanks to Marion Burhop and Harry Olm for all the big and small administrative and technical things they took care of, which supported my work in many different ways.

Further, I would like to thank Dr.-Ing. Dominik Merli for introducing me to this wonderful topic during my diploma thesis and the collaboration in the beginning, after I moved to TUM.

Over the last years, I had the chance to work with great people between San Jose and Shanghai and I would like to thank all of them for the collaboration. The three months with Dr.-Ing. Axel Poschmann and the PACE group at NTU in Singapore were very interesting to get new research perspectives and a broader view on cryptographic engineering outside of Munich. Further, the two months I spent in 2014 and 2015 with

Mandel Yu at Verayo gave precious practical insights on the PUF world and also lead several ideas and joint publications. I also thank Jeroen Delvaux and Mandel Yu for the intensive intercontinental collaboration, where we exchanged around 500 emails since early 2015.

I was lucky that I had the chance to advise some very talented and hard-working students that supported me in research and teaching, for example with implementations that will be discussed later in this work, and also contributing to several publications.

My work was funded by the German Federal Ministry of Education and Research (BMBF) through the project ARAMiS (01IS11035Y), SIBASE (01S13020A) and SMERCS (01DP12037A), and the Bavaria California Technology Center (2014-1/9).

Finally I would like to thank my family and friends for their support, patience and encouragement over the last years.

Matthias Hiller
July 2016

Contents

List of Acronyms

ASIC	Application Specific Integrated Circuit
BCH	Bose–Chaudhuri–Hocquenghem
C-IBS	Complementary Index-Based Syndrome Coding
CMOS	Complementary Metal-Oxide-Semiconductor
CO	Code-Offset
CTW	Contex-Tree Weighting
DSC	Differential Sequence Coding
ECC	Error-Correcting Code
FPGA	Field Programmable Gate Array
GCC	Generalized Code Concatenation
GE	Gate Equivalent
GMC	Generalized Multiple Concatenated
i.i.d.	independent and identically distributed
IBS	Index-Based Syndrome Coding
IC	Integrated Circuit
LUT	Lookup Table
ML	Maximum-Likelihood
NVM	Non-Volatile Memory
PUF	Physical Unclonable Function
RAM	Random Access Memory
REP	Repetition
RLE	Run-Length Encoding
RM	Reed–Muller
RO	Ring Oscillator
ROM	Read-Only Memory
ROVA	Reliability Output Viterbi Algorithm
RS	Reed–Solomon
SDML	Soft-Decision Maximum-Likelihood
SLLC	Systematic Low Leakage Coding
SRAM	Static Random Access Memory
TBD	Trace-Back-Depth
TRNG	True Random Number Generator

Chapter 1.

Introduction

The increasing availability of energy efficient and cost efficient computation, and also of wireless connectivity to the Internet allows to use connected embedded devices in more and more applications. This leads to several drivers that are changing our economy, society and everyday lives within a span of just a few years.

The Internet of Things is unfolding [AIM10] and allows various kinds of smart sensors to collect and transmit data, data centers to aggregate the data in the cloud and then control lightweight decentralized actuators or displays reacting on the data, sometimes even under real-time constraints. Moving the Internet of Things to the industrial context resembles in Industry 4.0 [BFKR14] where individual goods can now be manufactured in a large-scale industrial environment. All steps of the manufacturing processes of the future can be monitored and controlled with small connected embedded devices. In the medical and health care sector, wireless body area networks are established to connect sensors, spread over the body, to personal or even remote control instances for an individual medical treatment and also enabling a fast response if a critical condition occurs [ZRJ14].

All these trends have in common that they involve sensitive data that is processed, transmitted and stored, and thus has to be protected. Typically, cryptography is the technology of choice because it provides secure algorithms that are designed to be implemented efficiently in practice. The branch of lightweight cryptography [Pos09] takes the energy and area constraints especially into account.

Secret keys are a foundation for cryptography to give a legitimate user an advantage over an attacker. This advantage can only be preserved if it is impossible for the attacker to obtain information on the key of the legitimate user. In many practical scenarios, this requirement can be relaxed in a sense that the effort for the attacker to obtain the key has to be higher than the expected revenue or damage so that it is not interesting for an economically thinking attacker to pursue to go after the key.

Therefore, countermeasures have to be taken to secure the key and preserve the integrity of the system over the lifetime of the device. It is not sufficient to simply store the keys in weakly protected or even unprotected memories. The next section discusses the conventional approach to store keys in non-volatile memory and is followed by an introduction to key derivation with PUFs.

1.1. Non-Volatile Key Storage in Integrated Circuits

The straightforward way to provide a cryptographic key in an embedded device is to store it permanently inside the device. Storing cryptographic keys in Non-Volatile Memory (NVM) is typically a cumbersome task if a certain security level has to be met. Referring to the overview provided in [Kil15], five types of storage of particular interest are addressed in this section. More information on the state of the art and emerging NVM technologies can be found for example in [RE10, Che15].

Embedded Flash Embedded flash, also called Flash Electrically Erasable Programmable Read-Only Memory, is a non-volatile storage that is based on hot electron injection. Implementing embedded flash technology has the advantage that the memory can be written thousands of times. However, it has the downside that it requires at least ten more mask steps than standard CMOS technology during manufacturing and that it is currently only available down to a $55nm$ manufacturing process [Kil15]. For more information on the security of embedded flash, see for example [HT07].

Read-Only Memory (ROM) Hard-wired information can be placed in the mask in the form of ROM when a circuit is designed. It offers only a low security level against attacks and all devices at the same wafer position will have the same key, which is not tolerable for many security applications where devices should have unique identities.

Electrical Fuse The electrical fuse is a one-time programmable memory and is set by forcing such a high current through the fuse that the electrical connection is completely destroyed or at least severely permanently damaged. Fuse cells are relatively large and the surrounding circuit has to be protected by empty guard spaces around the fuse so that it is not damaged during the programming process. Due to its large form factor and massive interference with the material, fuses can be read out relatively easily by attackers, compared to other NVM technologies.

CMOS Floating Gate The memory cell is a MOS transistor with two gates, a floating one and a contacted one that overlap each other. The floating gate is insulated by an oxide from the top and bottom. The state of the transistor is determined by the charge of the floating gate and it can be programmed multiple times. This approach has similarities with embedded flash and also requires additional masking steps. However, it is compatible to the CMOS manufacturing process so that it can be integrated into existing processes more easily. This NVM type has a medium security level.

Antifuse The antifuse is a circuit with high capacity and resistance in the default state. In contrast to destroying a connection in the electrical fuse, the antifuse

establishes a connection, so that it changes to a low resistance state, after a high programming voltage is applied. The permanent changes inside the transistor are relatively small which makes them harder to detect for an attacker.

All presented NVM approaches have in common that they permanently change the silicon and, as a consequence, the functional behavior of the circuit. The following section present Physical Unclonable Functions. They have the advantage that they exploit existing minimal variations in the circuit so that no permanent changes in the circuit behavior can be measured.

1.2. Security from Intrinsic Manufacturing Variation

Physical Unclonable Functions (PUFs) cast in silicon emerged over the last decade as an efficient solution to increase the security level of cryptographic key storage in standard circuits to the level of the remaining circuit. As a major advantage, they can be manufactured in the same standard CMOS technology as the remaining circuit such that only additional area has to be spent on a chip but no additional processing steps are required.

PUFs evaluate manufacturing variations in the circuit to derive device-unique secrets. Analog physical measures such as doping levels or physical dimensions lead for example to different threshold voltages of transistors. Switching delays and other effects are sampled and quantized to create a digital value, called PUF response.

There are two main applications of PUFs: lightweight authentication and secure key derivation. Some PUFs have different configurations, set by an external challenge. This can be used to perform lightweight authentication protocols such as [MRK+12, YMVD14][YHD+16] to provide security without performing actual cryptographic operations that are expensive in terms of area. Focusing on secure key derivation, if applicable, a constant configuration sequence is assumed in the following to generate the same secret whenever it is requested.

While there is currently an ongoing arms race in the crypto-less lightweight protocols [YHD+16], there is a consensus in the community that key derivation with PUFs generally increases the security of CMOS circuits. Both approaches can already be found in commercial products as discussed in Section 1.4 later in this chapter.

1.3. Secure Key Derivation with PUFs

PUFs measure physical circuit properties and – like any physical measurement – they are affected by noise and varying environmental conditions. PUF responses are not

completely stable so that they cannot be directly used as cryptographic keys. At the same time, PUF responses of a specific chip are ideally completely unpredictable from the outside. Security is based on the fact, they do not have any structure but it also inhibits to directly identify and correct errors.

Therefore, additional side-information is stored that links the PUF response to a mathematical structure, and thus enables error correction. This side-information, or *helper data*, must not reveal information about the secret to still be able use the secret as key. After mapping the PUF response to a structure, typically an Error-Correcting Code (ECC) takes over to detect and correct the variation between the measured PUF response and an initial reference PUF response to reproduce the initially derived key.

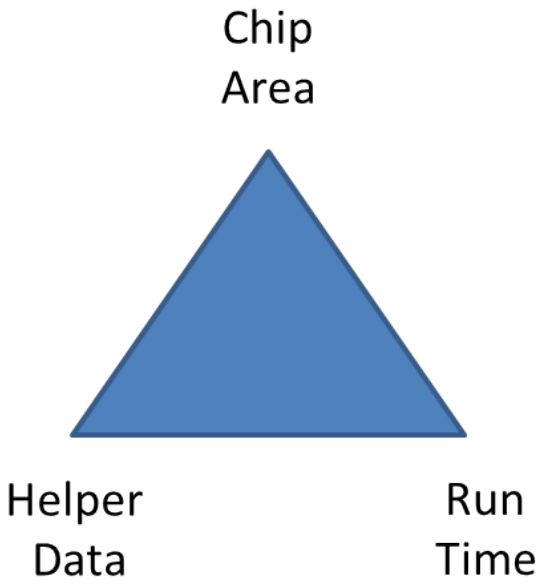

Figure 1.1.: Evaluation criteria for secure key derivation with PUFs

When implementing a PUF key derivation module, different aspects have to be taken into account. Figure 1.1 shows three main criteria for PUF key derivation, assuming that the security and reliability level of the cryptographic key are specified by the system or the cryptographic application the key is used in:

Chip Area The area on the IC that has to be spent on the PUF and the error correction translates directly to the cost of the key derivation module. Later, the error correction will optimized for two aspects. The first is to keep the number of PUF response bits as low as possible to save area. The second aspect is to reduce the complexity of the ECC. Depending on the characteristics and silicon area footprint of the PUF, different trade-offs between PUF response bits and ECC complexity are made to reduce the overall size of the key derivation block.

Helper Data The helper data has to be stored permanently and the cost of this storage can vary greatly. The range goes from expensive on-chip memory on one side to nearly free remote storage on a server on the other side of this range. Later in Chapter 6, special effort is spent to reduce the size of helper data with a compression algorithm.

Run Time Depending on the application there can be very strict or up to virtually no timing constraints. Assuming that a PUF key can be precomputed before the cryptographic algorithm runs does not pose strict time constraints. There is typically a trade-off between area and time, where this work optimizes for area.

1.4. Integration of PUFs into Commercial Products

Fundamental research is necessary to explore the feasibility of a new idea or technology and the ultimate goal in engineering is to bring this technology into everyday products. Meanwhile, hundreds of papers on PUFs were published which shows that the field was attracting broad academic interest and was maturing over the last years.

Two companies, Intrinsic-ID [Int] and Verayo[Ver], emerged out of the initial groups at Philips and the Massachusetts Institute of Technology working on PUFs which sell PUF IP and PUF-enhanced services.

A brief review of public information shows that secure key derivation with PUFs is already available as security feature inside Altera [Alt15] and Microsemi FPGAs [Mic15]. Also results of test chips of Intel processors [MSA+14] were published that use PUFs for secure key derivation. In addition, PUFs were also already introduced in a smart-card product by NXP [NXP13], which was recently certified by the German Federal Office for Information Security for the security level EAL6+ [Bun16b, Bun16a]. Other major smart-card vendors such as Infineon or Samsung also published papers on PUFs [HB10, KLC+16]. This shows that secure key derivation with PUFs is becoming commercially more and more relevant for various application scenarios.

Beyond secure key derivation, lightweight RFID tags with PUFs are used for example for Canon 60D DSLR cameras in China to mark genuine products and protect against counterfeiting [Dev].

The patent research in [SIB14] has shown that most major semiconductor companies filed patents on PUFs which strongly indicates that there is a global commercial interest in PUFs. However, the technology still has to make the way out of its niche in the next years to push further into the emerging mass markets, for example regarding the Internet of Things.

1.5. Contributions of this Thesis

The previous sections have shown that key derivation with PUFs already went a long way from fundamental research into first commercial products. However, there are still limits and shortcomings in the state of the art, and some of them will be addressed in later chapters of this thesis.

Theoretical models are important to understand the fundamental behavior and achievable limits of a problem. So far, the PUF work is lacking a theoretical model that covers the maximum key size together with the minimum helper data size over different environmental conditions. This makes it impossible to evaluate if the state of the art approaches already achieve this limits or how large the space for possible improvements is.

Error correction for PUFs seems to be manageable in practice if the block size is chosen small enough for compact hardware implementations. This contradicts with the typical information theoretical approach of increasing the block size to control the statistical properties of a drawn sequence.

Theoretical Model and new Security Criterion So far, PUF research is driven from a practical point of view but is lacking a theoretical model that shows the limits, especially when it comes to helper data. This work shows the parallels between the information theoretical model of secure key generation from a compound source and the practical problem of secure key derivation with PUFs, and highlights the common points and differences.

So far, error correction schemes were mainly represented on an algorithmic level where it is hard to see general security properties. Many key derivation approaches with PUFs can be brought into an algebraic form. The algebraic representation of the linear state-of-the art schemes are discussed and an algebraic security criterion is introduced which allows to upper bound the leakage of an error correction scheme already during the design of the algorithm.

Systematic Low Leakage Coding There exists a theoretically optimal information theoretic approach for secure generation using large random codebooks which is brought into a deterministic form in this work. Systematic Low Leakage Coding (SLLC) is introduced as first approach that combines a minimal helper data size and the possibility to achieve theoretical limits for optimal ECCs. In addition, this work discusses the parameters for a lightweight implementation to demonstrate the practical feasibility of the new scheme.

Differential Sequence Coding There are several ways to obtain reliability information on specific PUF response bits. Looking at the distribution of reliable PUF response bits

over different blocks with the information theoretical concept of typicality reveals that the reliability of the error correction increases greatly by using larger blocks. Differential Sequence Coding (DSC) is introduced as a pointer-based approach which indexes PUF response bits that are more reliable than a given reliability threshold and treats all indexed bits as one single block.

In addition, I was the first to use convolutional codes in the PUF context. Looking at the output error probability over different parameters shows that DSC is far more effective as previous work. An FPGA hardware implementation is discussed and compared to the state of the art to demonstrate the performance of DSC in a practical setting.

Evaluation The error correction approaches for PUFs have different properties so that their effectiveness also depends on the scenario they are used in. The implementations discussed in this work are compared to the state of the art for different input and output error probabilities and I present a comprehensive listing also containing execution times and FPGA slice counts.

1.6. Outline

Background information on PUF definitions, PUF primitives and evaluation criteria is given in Chapter 2. Chapter 3 discusses the state of the art algorithms and implementations for secure key derivation with PUFs.

Chapter 4 introduces new theoretical foundations on key derivation with PUFs. A first syndrome coding scheme, called Systematic Low Leakage Coding, is closely related to the presented theory and is introduced in Chapter 5. A pointer-based syndrome coding scheme, called Differential Sequence Coding, and its FPGA implementation are presented in Chapter 6.

The evaluation in Chapter 7 compares the new schemes to the state of the art to set this work into a larger context. Chapter 8 concludes this work and gives an outlook over open related problems. Additional information is provided in the Appendix.

1.7. Definitions and Notation

Notation Random variables are given in capital italic letters, e.g. X and scalars such as outcomes of random variables in small italic letter, e.g. x. Calligraphic letters $\mathcal{X}$ indicate sets and $|\mathcal{X}|$ is the cardinality of set $\mathcal{X}$. Further, a superscript over a letter, e.g. X^n, denotes a vector of n instances of X. Note that the random variables in X^n can have different probability distributions such that for example each PUF response bit can have an individual bit error probability. X_i^j selects elements i to j of vector X^n.

$\Pr[A]$ is the probability of event A. $P_X(x)$ denotes the probability distribution or, more formally, the probability mass function, of X for $x \in \mathcal{X}$. $cdf(\cdot)$ is the corresponding cumulative distribution function. Further, let $\mu(\cdot)$ be the mean operator and $\sigma(\cdot)$ the standard deviation.

Matrices are written in bold capital letters. Let $\mathbf{A}^T$ be $\mathbf{A}$ transposed. $\mathbf{I}$ is the identity matrix with ones in the main diagonal and $\mathbf{0}$ is the all zero matrix. Concatenations are indicated with square brackets $[\cdot]$.

For random variables X and Y, $X|Y$ denotes X under the condition Y. Let $H(X)$ stand for the Shannon entropy of X and $H(XY)$ for the joint entropy of X and Y, and let $I(X;Y)$ be the mutual information between X and Y [CT06, Kra07].

For a better readability, integer representations of numbers and their binary representations in $\mathbb{F}_2^n$ are both used without marking the binary representation explicitly. In cases that require special emphasis, the binary representation of i in $\mathbb{F}_2^n$, $n \in \mathbb{Z}^+$, is denoted with $b_n(i)$.

Error-Correcting Codes Typically, error-correcting codes are defined by the code length n, the code size (or number of information bits) k and the minimum distance between any two codewords d [Bos99]. Sometimes, also the minimum number of correctable errors $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ is used. Code $\mathcal{C}$ is defined as set containing all codewords C_i, $i = 1, ..., 2^k$.

According to the definition in [BW13], codes are also characterized by code length n and code size k. Instead of specifying the code distance, the maximum probability $\epsilon > 0$ of a decoding error is defined for a given channel $\mathfrak{T}$. Then, a code is characterized as (n, k, ϵ) code in the theoretical part. For asymptotic results, k is replaced by k_n. The Channel Coding Theorem [CT06] states that for channel $\mathfrak{T}$ with input X, output Y and capacity $C_{\mathfrak{T}}$, there exist (n, k_n, ϵ) codes such that

$$\lim_{n\to\infty} \frac{k_n}{n} = C_{\mathfrak{T}=I(X;Y)} \tag{1.1}$$

$$\lim_{n\to\infty} \epsilon = 0 \tag{1.2}$$

For the practical part, the (n, k, d) code parameters are used again.

Chapter 2.

Physical Unclonable Functions

As already discussed in the previous chapter, PUFs evaluate manufacturing variation to generate information in a circuit that is unpredictable from the outside. This chapter addresses three main points:

- Basic PUF definitions are covered in Section 2.1.
- Section 2.2 reviews qualitative and quantitive measures for PUF quality.
- Section 2.3 introduces a set of PUF constructions that covers the most popular physical phenomena to generate PUF response bits.

General introductions to PUFs can be found for example in [MV10, HB10, RDK11, Mae13, HYKD14].

This chapter contains a new method on the reliability evaluation of PUFs that was introduced in [HSP13] and I also contributed to the entropy estimations published in [PRPHG14, WHP14, PHG16].

2.1. Definition of PUFs

A very early PUF definition was presented by Gassend *et al.* in [GCDD02] that defines a PUF as *a function that maps challenges to responses, that is embodied by a physical device, and that verifies the following properties:*

1. *Easy to evaluate: The physical device is easily capable of evaluating the function in a short amount of time.*
2. *Hard to characterize: From a polynomial number of plausible physical measurements [...] an attacker [...] can only extract a negligible amount of information about the response to a randomly chosen challenge.*

For secure key storage, a fixed challenge schedule is applied and the responses are not published such that the second point only refers to lightweight authentication with PUFs. The PUF response bits should ideally be independent and identically distributed (i.i.d.), which is consistent with the second property.

The term *intrinsic* PUF states that the PUF response is generated from a part of an IC without external components so that several intrinsic PUFs can also be implemented on FPGAs without modifications [GKST07]. The term *intrinsic* PUF is widely used synonymously with the term *silicon PUF*. Ideally, a silicon PUF is manufactured as digital circuit in a standard CMOS process so that the design and manufacturing can be integrated easily into existing design flows and manufacturing processes.

Plaga and Merli introduced a notion of characterizing a PUF as information storage [PM15]. The stored value is defined by the manufacturing variations and this definition also can be applied to quantum devices.

Typically, PUFs are separated into two classes of PUFs: strong PUFs and weak PUFs.

Strong PUF Some PUF circuits can be configured such that one configuration of physical properties is selected from a large set of possible combinations. *Strong PUFs* have a challenge-response interface. The challenge configures the PUF and the response depends on the challenge and the physical properties of the PUF. The PUF can output a large number of response bits that ideally are hard to predict even if a large number of challenge-response pair of a particular PUF is already known [GKST07]. This facilitates the use in lightweight authentication protocols where PUF bits are exposed to the attacker.

This property is tempting in theory but also faced several practical attacks in recent days. Machine learning attacks try to create a model of a PUF from a number of public challenge-response pairs to be able to authenticate a mathematical clone of the PUF [RSS+10]. The practical security of a PUF-based lightweight authentication protocol depends on the number of challenge-response pairs that is necessary to be able to reliably predict unknown PUF responses so that one goal is to limit the number of available challenge-response pairs [YHD+16].

Weak PUF It is natural to assume that a circuit with unknown properties produces one read-out value. *Weak PUFs* [GKST07] have a response space that grows only linearly with the area. The PUF outputs a reasonable number of response bits for key derivation but it is not sufficiently high for a large number of authentication events. Weak PUFs are also known as *physically obfuscated key* [Gas03].

Note that other than the names imply, weak and strong PUFs differ in their behavior but do not necessarily lead to different security levels.

2.2. PUF Properties

Robustness The term robustness characterizes how much two responses of the same PUF differ. It is given quantitatively as one minus the relative Hamming distance

between the PUF responses and is an indicator for the reliability of a PUF [GKST07, MGS13]. The term *intra Hamming Distance* is also used in this context.

For chips that each return initial PUF responses X^n and i subsequent PUF responses Y_i^n, the intra distance HD_{intra} is defined as

$$HD_{intra} = \frac{1}{k} \sum_{i=1}^{k} \frac{HD(X^n, Y_i^n)}{n} \times 100\% \tag{2.1}$$

However, HD_{intra} only gives the expected error probability over all PUF response bits so that it is only a rough estimate. Having a good model of the reliability distribution of the PUF response bits is important to evaluate the PUF primitives themselves and later to design the error correction properly [Mae13],[HSP13].

In the remainder of this work, the reliability distribution in [MTV09b], which was obtained from real world data, will be used as reference for discussion and simulation. This has the advantage that it makes the results reproducible and also comparable to referenced work referring to the same setup. Details on the distribution are provided in Appendix A.3.

Instead of using a newly coined PUF term, I will use the bit or block error probability as measure as it common in communication theory in the following. Especially, since the systems will operate in the domain of a robustness very close to 100% where differences are only hardly visible but important. The step from 10^{-9} to 10^{-10} is more clear than $1 - 10^{-9}$ to $1 - 10^{-10}$ and especially allows logarithmic plotting.

Physical Unclonability Physical Unclonability [AMS+11] is a lofty definition from the early days of PUFs. It conveys the aim to create non-silicon PUF like optical PUFs [PRTG02], coating PUFs [TSS+06], or the PEP foil that is wrapped around a device [HSZS13] that make a device unclonable by adding some physical security layer. For silicon circuits, there are technology dependent limitations that allow for example invasive attacks with focused ion beams on any part of the circuit [NHSB13]. It is reasonable to make the PUF as secure as the surrounding circuit processing security sensitive information, but true unclonability is out of scope and capability of today's silicon PUFs manufactured in CMOS technology.

Unpredictability Unpredictability covers the aspect that a PUF response to a challenge should be hard to predict if the responses of other PUFs to this challenge are known. Further, a response should be hard to predict if a fixed number of PUF responses of the same PUF is already known. The PUF response should also be hard to predict from helper data.

The *inter Hamming distance* or *Uniqueness* [MGS13] is a popular measure to assess the unpredictability of a set of PUF responses. Again for m chips with initial PUF responses X^n, the uniqueness is defined as

$$HD_{inter} = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{HD({X_i}^n, {X_j}^n)}{n} \times 100\% \tag{2.2}$$

Applying the inter Hamming distance is a first-order approach but is not able to cover correlations and other patterns within PUF responses. *Bit Aliasing* and *Uniformity* are similar measures that also cover different properties of the PUF [MGS13].

Source coding was used for the first time by Ignatenko *et al.* in [ISS+06] to give a tighter upper bound on the entropy of PUF bits. Context-tree weighting [WST95] but also other algorithms such as Lempel-Ziv [ZL77] can be used. If analog PUF data is availalbe, principal component analysis can be applied to detect patterns in PUF data [WHP14].

In [PRPHG14, PHG16], we went one further step and looked at the entropies of different parts of structures to derive possible shortcomings in the circuit design.

2.3. PUF Primitives

Silicon PUF primitives that are manufactured as IC contain two different components: One measures an internal physical quantity that is unique for each circuit in a *measurement circuit*. This can be for example threshold voltages of transistors.

The second part is a *quantization stage* that turns the analog measurement of the property into a digital signal that is used in later processing steps. Similar definitions can also be found in [AMS+11].

2.3.1. SRAM PUF

SRAM cells have a undefined power-up state depending on the threshold voltages of the involved transistors in the inverters that are shown in Figure 2.1. The SRAM PUF [GKST07, HBF09] makes use of this phenomenon and evaluates the power-up state of the SRAM as PUF response.

SRAM cells are very highly optimized circuits such that SRAM PUFs can generate a relatively high number of PUF bits on a low area with average bit error probabilities around 15% and a high unpredictablity [KKR+12]. However, SRAM PUFs require a careful layout of the cells such that no bias is introduced as it was observed for example in [vHvdLS+13]. The SRAM cell combines the measurement and quantization circuit

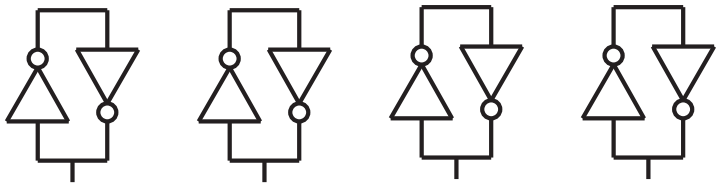

Figure 2.1.: SRAM PUF

because the inverter whose threshold voltage is reached first after power-up forces the other inverter into the inverse state.

Other examples that use a similar symmetric structure with an undefined power-up state are the Butterfly [KGM+08], Flip-Flop [MTV08], Two-Stage [HB10] or Buskeeper [SvdSvdL12] PUFs.

2.3.2. Arbiter PUF

The Arbiter PUF evaluates the cumulative delay difference of signals that propagate through two different paths in a circuit. The Arbiter PUF circuit contains of a number of multiplexers that can be configured through the external challenge as shown in Figure 2.2.

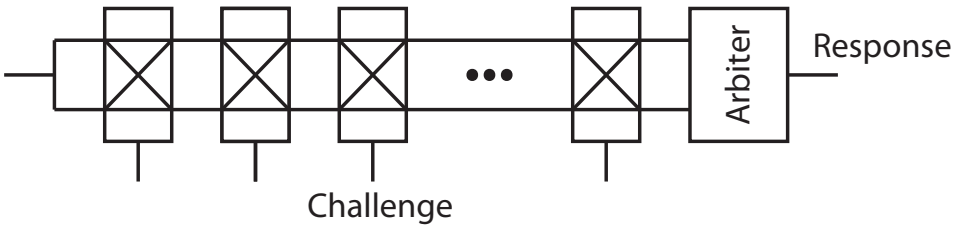

Figure 2.2.: Arbiter PUF

The first work on Arbiter PUFs was carried out by Lee *et al.* [LLG+04] and Lim *et al.* [Lim04, LLG+05].

Arbiter PUFs have the advantage that they can be implemented very compactly on ASICs and also have a challenge-response interface. This allows to generate multiple bits from a small number of PUFs, as it used e.g. by Yu *et al.* in [YHD15]. A symmetric layout of the delay path and the arbiter is important to avoid bias and achieve a high unpredictability. In practice, the results of multiple arbiter chains are XORed [SD07] to increase the resilience against machine learning attacks.

2.3.3. Ring Oscillator PUF

The Ring-Oscillator (RO) PUF by Suh and Devadas [SD07] comprises of chains with odd numbers of inverting elements as shown in Figure 2.3 as measurement circuit. Instead

of evaluating a signal that propagates through the circuit once like in the Arbiter PUF, the RO PUF switches constantly with a manufacturing variation dependent frequency. The signal propagates through the chain so that the the inverted input signal at the end of the chain is fed back to the input and propagates again through the chain. Due to the delay of the inverting elements, a periodic oscillation can be observed. Typically, one NOR gate is used as enable signal to prevent the circuit from oscillating permanently to save energy.

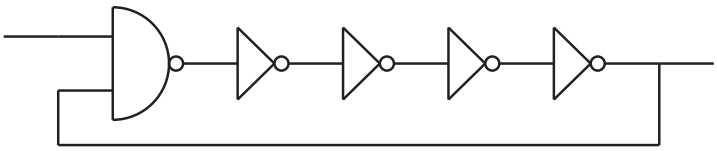

Figure 2.3.: Ring-Oscillator with 5 inverting elements

Several ROs are combined to an RO PUF, given in Figure 2.4. Counters evaluate the oscillation frequencies of the inverter chains and quantize the measured values. The ROs are typically evaluated pair-wise so that only two counters are required and the different RO chains are connected through multiplexers to the counter. More details on the implementation RO PUFs and attacks on them can be found e.g. in the dissertation of Merli in [Mer14].

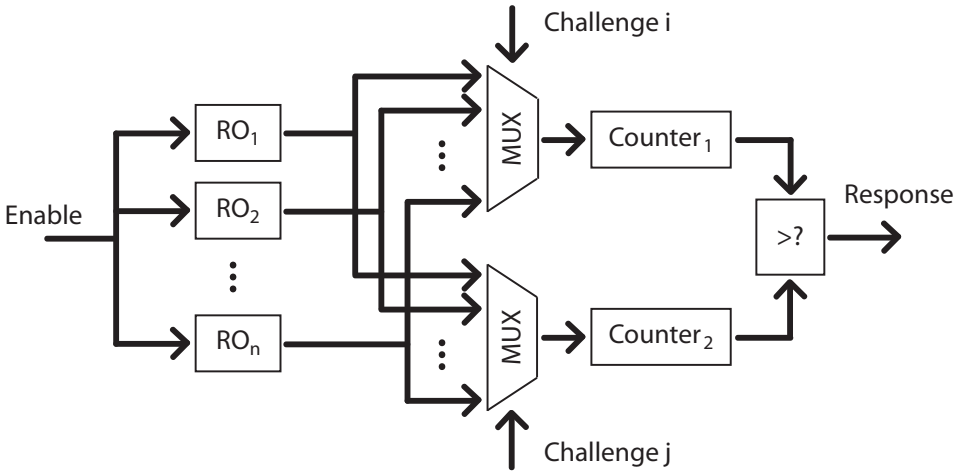

Figure 2.4.: Ring-Oscillator PUF

RO PUFs have the advantage that they can be implemented very well on FPGAs. Since they are built from standard cells, they are also well-suited for the implementation in IP blocks.

The sum-PUF by Yu and Devadas allows to mimic an Arbiter-PUF-like behavior with ROs on FPGAs [YD10a]. Several improvements on the quantization of the frequencies were proposed in [YQ10, MVHV12, YQ13, GI14]. Some of these approaches require to store helper data that can be attacked [DV14b].

2.3.4. Bistable and Twisted Bistable Ring PUF

The Bistable Ring PUF [CCL+11, CCL+12] can be seen as a mix of the three previous PUFs. It contains of a large RO with an even number of inverting elements such that it ultimately settles in a stable state like the SRAM PUF. In addition, the elements of the ring can be configured similarly to an Arbiter PUF with an external challenge to generate multiple response bits from the same physical structure.

The Twisted Bistable Ring PUF [SH14] is an extension that fixes practical shortcomings of the Bistable Ring PUF such as a strong bias over the PUF responses. This is achieved by changing the order of the elements in the ring through the challenge instead of exchanging inverters in a ring.

2.3.5. Non-Silicon PUFs

The previous PUFs all have in common that they comprise of digital logic gates that can be easily integrated into default CMOS manufacturing processes. Originally, PUFs were introduced by Pappu *et al.* in the non-silicon world [PRTG02]. This subsection mentions a few non-silicon PUFs for completeness. However, the remainder of this work focuses silicon PUFs.

Instead of using intrinsic circuit components, the Coating PUF by Tuyls *et al.* wraps a layer around the circuit to protect it [TSS+06]. The coating can contain particles that block different wavelengths of light and also metal particles leading a local variation in capacitance. This unique pattern is used to generate PUF responses. An optical coating PUF was investigated for example in [EFK+12].

The Protecting Electronic Products foil by Hennig *et al.* [HSZS13] goes one step further than the coating PUF and allows to cover larger form factors, for example an entire printed circuit board for tamper protection.

2.4. Conclusions

This chapter discussed basic definitions on PUFs to lay the foundation for the following chapters. Most silicon PUFs evaluate effects based on threshold voltages or timing behavior of transistors. The SRAM and BR PUF settle in unpredictable stable states while the RO and Arbiter PUF measure timing properties.

Different reliability and entropy measures are presented that are important for the key derivation schemes. They permit to quantify statistical properties of PUFs to layout the error correction discussed in the next chapter.

Chapter 3.

Error Correction for PUFs

This chapter addresses the state of the art in error correction for PUFs, starting with the definitions used in referenced work. It gives an overview over theoretical work in different related communities around the hardware security field. Then, previous practical schemes and their implementations are discussed. Overviews covering different aspects of error correction can be found in e.g. in [Mae13, HYKD14, DGSV15],[DGV+16]

PUF-specific definitions for error correction for PUFs are provided in Section 3.1. Section 3.2 gives an overview over related theoretical work. In Section 3.3, linear schemes are discussed as one of the two major classes of error correction schemes for PUFs. The pointer-based schemes in Section 3.4 are the second class. PUF-specific ECC implementations are given in Section 3.5.

This chapter also mentions contributions on error correction schemes where I was not the main contributor and contributions that do not fit in the flow of the main contributions of this work that will be addressed in detail in the following chapters. In [MPB+14, PMB+15, HKS+15], we analyzed and implemented generalized concatenated codes for PUFs [Bos99]. The C-IBS papers in [HMSS12], and partly [HDSMS12] are based on my diploma thesis [Hil11] and extended in [HDSMS12]. The Maximum-Likelihood (ML) symbol decoder by Yu *et al.* presented in [YHD15] interprets multiple PUF bits as higher-dimensional symbols. A high-level introduction in German can be found in [HPS15] and I also contributed to the overview and analysis by Delvaux *et al.* in [DGV+16].

3.1. Definitions

There are several synonym and overlapping definitions for the error correction that were used over the last years. This section reviews the most common ones and explains the definitions used in this work.

The term *helper data algorithm* is a very wide term and covers all error correction schemes for PUFs that use helper data.

Secure Sketch, Helper Data Architecture and Fuzzy Embedder The *secure sketch* allows to correct errors in biometric templates with the help of helper data.

To be more precise, Dodis *et al.* [DRS04] define a secure sketch S over space $\mathcal{X}$ as a randomized map $\mathsf{S} : \mathcal{X} \to \mathbb{F}_2^*$ that fulfills the following two properties:

1. There exists a deterministic *recovery procedure* Rec that recovers X from its sketch $W = \mathsf{S}(X)$ and any vector Y close to X such that $\mathsf{Rec}(Y, W) = X$ for all $X, Y \in \mathcal{X}$ with Hamming distance $hd(X, Y) \leq \lfloor \frac{d-1}{2} \rfloor$.
2. For all random variables X with output set $\mathcal{X}$ and min-entropy $H_\infty(X)$ the average min-entropy of X given W is at least m', so $\tilde{H}_\infty(X|W) \geq m'$.

A *helper data architecture* [TG04] is a less often used synonym for a secure sketch.

The *fuzzy embedder* by Buhan *et al.* [BDH+10] is a modification of the secure sketch that embeds an arbitrary, external secret into a biometric template and stores helper data. It recovers the embedded secret instead of restoring the biometric template. The term is mainly used in a biometrics context rather than in the PUF field.

Fuzzy Extractor The *fuzzy extractor* is an extension of the secure sketch. Instead of outputting the corrected biometric template, the template is hashed to produce a full-entropy cryptographic key [DRS04]. Formally, a fuzzy extractor contains the two procedures *generation* Gen and *reproduction* Rep:

1. The probabilistic *generation procedure* Gen has an input $X \in \mathcal{X}$, and outputs the extracted secret string $K \in \mathbb{F}_2^l$ and the public helper data W. Let P_U be the uniform distribution. For any $X \sim P_X$ with $H_\infty(X)$ it is required for $(K, W) \leftarrow \mathsf{Gen}(X)$ so that the statistical distance between (K, W) and (P_U, W) is bounded by an ϵ. Only the Hamming distance is used in this work, but the construction also holds for other metrics.
2. The deterministic *reproduction procedure* Rep recovers K from W and a $Y \in \mathbb{X}$ that is close to X with $hd(X, Y) \leq \lfloor \frac{d-1}{2} \rfloor$. If $(K, W) \leftarrow \mathsf{Gen}(X)$ then $\mathsf{Rep}(Y, W) = K$.

Comments: Typically, $\mathcal{X} = \mathbb{F}_2^n$. In practice, there also deterministic Gen procedures used.

According to [DRS04], a fuzzy extractor can be built from any (n, k, d) ECC.

In [DGV+16], we have discussed new, tighter, bounds to evaluate $\tilde{H}_\infty(X|W)$.

Using the same biometric template multiple times can leak key information. This issue was addressed by Boyen in [Boy04].

The fuzzy extractor can also be extended to a robust fuzzy extractor to detect tampering with the helper data [DKRS06, KR08, DKK+12].

Syndrome Coding and ECC All previous definitions have the shortcoming that they all address the helper data generation and the error correction together. I picked up the wording of [YD10b] and define the isolated helper data generation step as *syndrome coding*. This definition is inspired by the coding theoretic definition, where the syndrome is the multiplication of a vector with the parity check matrix of a code. If the syndrome is zero the vector is a codeword. For the error correction part, the existing definition for ECCs is fully precise and sufficient.

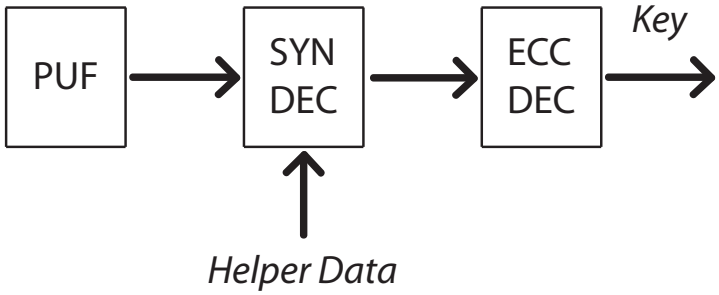

Figure 3.1.: Generic reproduction procedure

Figure 3.1 shows a generic minimalistic reproduction procedure. The PUF response and the helper data are the inputs of the syndrome decoder. Then, the output is forwarded to the ECC decoder, where the remaining errors are corrected. For some approaches, this figure is extended, for example by adding hash functions.

The definitions are mainly given for completeness to put the referenced work into context. In the following, I will mainly use the syndrome coding wording.

3.2. Theoretical Background

Error correction for PUFs touches the information theory, biometrics and cryptographic community that use slightly different methods and definitions, and also emphasize different aspects and results. Later, I will also discuss new theoretical results. This section presents an overview over the state of the art on theoretical work on PUFs to embed the new results in a larger context.

Information Theory Community Wyner's wiretap channel [Wyn75] is an old problem in information theory that started a large body of research and is currently attracting a lot of attention under the term *Physical Layer Security* [BBRM08]. Basically, two legitimate parties establish a joint secret by transmitting messages over a channel where the attacker has access to a slightly noisier version of the communication than the legitimate receiver.

The problem of key agreement from correlated sequences of a source with multiple outputs is the corresponding source coding problem. The main difference here is that the shared sequence is given by the source and cannot be chosen arbitrarily like in a channel coding problem. Early work was carried out by Ahlswede and Csiszar [AC93] and Maurer [Mau93]. Strong secrecy is a stronger security notion that bounds the absolute secrecy leakage instead of the secrecy leakage per bit [CN00].

Boche and Schäfer[1] [BW13] extended the model and have shown that strong secrecy is achievable. The authors also quantified the communication costs. This line of work is an important foundation for this thesis and more detailed information can be found in Appendix A.1.

Biometrics Community Biometrics is a popular application of the information theoretic model of secret key agreement. An early overview can be found in [LTS07]. Secret key rates of optical PUFs were addressed in early work of Ignatenko *et al.* in [ISS+06] where the approach of [Mau93] is generalized from i.i.d (independent and identically distributed) to ergodic sources, and the entropy is measured with Context-Tree-Weighting [WST95], a universal source coding algorithm.

Shielding Functions by Linnartz and Tuyls [LT03] generate secret keys for continuous distribution using quantization index modulation. The work of Buhan *et al.* [BDHV07, Buh08, BDH+10] also looks at different information embedding approaches for continuous distributions. Recent work on the quantization was also performed by Immler *et al.* in [IHKS16].

For biometrics, privacy leakage plays an equally important role as secrecy leakage. Since everyone only has a limited amount of biometric features that are potentially re-used for different security applications, the question arises how much of the unique information is revealed by each approach. Ignatenko and Willems discuss fundamental secrecy and privacy trade-offs from an information theoretic point of view in [IW09, IW10, IW12].

Revealing the confidence of a generated key might be advantageous for some applications but also opens new attack vectors as shown in [SKVdV09].

The *fuzzy vault* [JS02] is a scheme that allows to use a set of biometric in an order-invariant way to unlock stored secrets.

Crypto Community The universal composition framework [Can01] is a major research direction in cryptography that is not directly related to secure key generation. However, [BFSK11, Sch13] show that PUFs can be used in this context for cryptographic protocols such as bit commitment and oblivious transfer. Practical implications are discussed in [RvD13].

In [AMS+09b, AMS+09a], Armknecht *et al.* interpret PUFs as pseudo-random functions and discuss the properties and implications.

[1] né Wyrembelski

3.3. Linear Schemes

Linear schemes are one family of practical syndrome coding schemes. They have in common that linear operations are applied to the entire input sequence to compute the secret and the helper data. Therefore they also use linear ECCs [Bos99] with parameters (n, k, d). Some schemes can also handle non-linear codes but this option was not investigated or even implemented so far.

Fuzzy Commitment The Fuzzy Commitment [JW99] by Juels and Wattenberg stores a random input R^k with the help of the PUF in a concealed way such that it can be reproduced and used later as key S^n. R^k is encoded to a codeword $S^n = C^n = R^k\ \mathbf{G}$ of the ECC by multiplying the random sequence with the generator matrix $\mathbf{G}$ of the code. Then, it is masked with the PUF response X^n and the result is stored as public helper data W^n.

$$S^n = R^k\ \mathbf{G} \tag{3.1}$$
$$W^n = \left(R^k\ \mathbf{G}\right) \oplus X^n \tag{3.2}$$

A modified version of the Fuzzy Commitment was introduced in [TAK+05]. It uses a secret K^k which does not contain any redundancy directly as output instead of the codeword.

$$K^k = S^k = R^k \tag{3.3}$$
$$W^n = \left(R^k\ \mathbf{G}\right) \oplus X^n \tag{3.4}$$

The original Fuzzy Commitment relies on an information theoretical security argument. The Computational Fuzzy Extractor [FMR13] is a modified Fuzzy Commitment that is constructed based on a complexity theoretical argument, namely the hard problem of learning parity with noise [BKW03]. Here, the structured generator matrix of the ECC $\mathbf{G}$ is replaced by a random generator matrix that is a-priorily chosen. Since the decoding is hard under the presence of errors, a trapdoor is introduced in [HRvD+16] that basically uses erasures to mark unreliable PUF bits in Y^n dynamically during reproduction and thus reduces the decoding complexity to a practically feasible level while keeping the complexity for the attacker at the previous level.

Code-Offset Fuzzy Extractor The Code-Offset Fuzzy Extractor [DRS04] shows several parallels to the Fuzzy Commitment. The main difference is that the PUF response X^n defines the secret S^n instead of deriving it from the random number R^k. This difference causes leakage so that a hash function has to be added. Again, the helper data W^n

is computed as XOR between a random codeword $C^n = R^k\ \mathbf{G}$ and the PUF response X^n.

$$\begin{aligned} S^n &= X^n \\ K^k &= f\left(S^n\right) \\ W^n &= \left(R^k\ \mathbf{G}\right) \oplus X^n \end{aligned} \tag{3.5}$$

There exist different implementations of this approach:

An early implementation was introduced by Bösch *et al.* [Bös08, BGS+08] looking at Reed–Muller, BCH and Golay Codes [Bos99].

Maes *et al.* published a Soft-Decision Reed–Muller implementation [MTV09b, MTV09a] for an SRAM PUF on FPGAs, decoding the Reed–Muller code as Generalized Multiple Concatenated (GMC) code [Bos99].

Van der Leest *et al.* looked at an enrollment scenario with a single read-out [vdLPvdS12] also with Golay Codes and a standard-array decoder.

Recently, we introduced constructions with generalized code concatenation in [MPB+14] using Reed–Muller codes and [PMB+15] also including Reed–Solomon codes. Kürzinger implemented the approach discussed in [MPB+14] under my supervision [HKS+15][Kür14] aiming at a very compact implementation size.

Syndrome Construction Another construction introduced in [DRS04] is based on the method in [BBCS92] and requires linear codes. It stores the syndrome of the PUF response as helper data so that no extra input random number R^k is required. Note that for linear ECCs, the syndrome is precisely defined for block codes with parity check matrix $\mathbf{H}$. The syndrome is computed by multiplying the PUF response X^n with $\mathbf{H}^T$. Again, a hash function $K^k = f\left(S^n\right)$ is added to mitigate leakage. It compresses the PUF output to create a secure cryptographic key.

In channel coding, the syndrome reveals information on the error pattern, or more precisely the coset of the input word. In the general PUF context, the word syndrome is interpreted as information that facilitates error correction since it also contains information on the error pattern in the PUF response. Here, *Syndrome Construction* is used to specifically refer to the precise channel coding definition.

Suh implemented this scheme in 2005 in the AEGIS secure processor [Suh05], using a BCH code as ECC. Maes *et al.* also presented an implementation called PUFKY in [MVHV12] that uses the syndrome construction and a BCH code. It is part of a stand-alone FPGA IP core using RO PUFs.

Parity Construction The construction in [DFM98] stores the parity of the PUF response according to $W^{n-k} = X^k\mathbf{P}$. The PUF response X^k is hashed and output as key.

3.4. Pointer-Based Schemes

Instead of storing a linear function of the PUF response and random input number as helper data, the pointer-based schemes store pointers to specific PUF responses. This has the advantage that linear dependencies between the secret and the helper data are removed.

Index-Based Syndrome Coding Index-Based Syndrome Coding (IBS) is a pointer-based approach and was introduced by Yu and Devadas in [YD10b]. A secret is encoded with an ECC and the PUF response is divided into fixed-sized blocks. Within each block the PUF response bit that is equal to the codeword bit with the highest probability is indexed, so that a pointer to this bit is stored in the helper data.

IBS performs error reduction by selecting response bits with a higher than average reliability. Another advantage is that for i.i.d. PUF bits, the pointers are uncorrelated with the code sequence so that no information leaks through the helper data.

In [YMSD11], an RO sum-PUF was used to generate the PUF bits. Syndrome distribution shaping was introduced to harden the approach against machine learning attacks. In addition, an ASIC implementation was presented in [YSS+12]. The output bits of an RO sum-PUF are not fully i.i.d. so that IBS helper data can attacked with machine learning as it was shown in [BWG15].

Complementary Index-Based Syndrome Coding IBS ignores the majority of PUF response bits so that is quite inefficient in terms of used PUF bits. Only a small fraction is indexed while the rest is discarded. In my diploma thesis [Hil11], I introduced Complementary IBS (C-IBS) that adds an intermediate encoding step. The codeword bit is encoded with a repetition-like code, but with equal Hamming weight for both codewords. Then, each repetition code bit is encoded as IBS pointer. This increases the efficiency of IBS. Publications on C-IBS can be found in [HMSS12] and [HDSMS12].

Maximum-Likelihood Symbol Recovery The previous two approaches index bit-wise. In contrast, the ML Symbol Recovery we presented in [YHD15] indexes an entire PUF response block, similar to the Slender protocol for authentication with strong PUFs [MRK+12]. The PUF response is partitioned into several blocks and the secret selects one of the blocks which is published as helper data.

For decoding, the entire response sequence is read in and all blocks are compared to the helper data. The position of the block with the minimum distance is returned as secret. Note that ML decoding is applied, which has increased error correction capability compared to other decoding algorithms. However, the decoding complexity and number of PUF response bits increase exponentially with the number of embedded key bits. The implementation in [YHD15] demonstrated that this approach is especially suited for PUFs with bit error probabilities greater than 20%.

3.5. Error-Correcting Code Implementations

For most of the implementations, the ECC decoder is the largest module in the reproduction block. In contrast to most communication use cases, latency and throughput are not critical for most considered PUF applications. If the key can be precomputed before it is used, the area is the most critical optimization goal in today's implementations.

Application Specific Processors One popular implementation strategy breaks down the decoding algorithm into instructions that are executed by a processor with a reduced custom instruction set. It has the advantage that the components are generic and can be reused several times. This strategy was used to implement a *Generalized Multiple Concatenated* (GMC) code decoder [Bos99] for a Reed–Muller code in [MTV09a] and a BCH decoder in [MVHV12, VHV12]. In [Ley15], Leyh developed a core for decoding BCH codes under my supervision that will be used for reference implementation results later in this thesis.

Reed–Muller Reed-Decoder In [HKS+15], we broke down the Reed decoding of the Reed–Muller Code into very small and very similar operations and used a circular shift register to move different code bits to the desired destinations of the decoder. It is based on the Reed decoder discussed in [MS77].

3.6. Conclusions

There are two main families of syndrome coding schemes for PUFs: linear approaches and pointer-based approaches. Over the last 10 years, vivid research in the PUF community improved the efficiency of the error correction, leading to more compact and powerful implementations.

The ECC decoders are relatively large and have different requirements than in typical communication scenarios. Therefore, they have to be addressed in detail and optimized especially for this use case.

Chapter 4.

Theoretical Foundations of Key Derivation with PUFs

Chapter 3 has shown that there are several approaches for secure key derivation with PUFs and also the referenced implementation demonstrated that the approaches can be used in practice. New algorithms were designed and the limits were pushed. However the question is still open, how far the state of the art is away from ultimate limits. Information theory provides the necessary tools to answer this question because the problem of secure key agreement from correlated sources in general [AC93, Mau93] and the introduction of the compound source in specific [BW13] address this issue. I will discuss why and how this model fits to the practical problem and repeat fundamental theoretical results from [AC93, Mau93, BW13] that will be used in this work.

This chapter has two main contributions: An information theoretical model for secret key derivation with PUFs and a generic algebraic representation and security criterion to evaluate syndrome coding schemes on an algorithmic level.

Section 4.1 addresses the common points and differences between the information theoretical and the practical problem. The impact of the information theoretical limits on PUFs are discussed in Section 4.2. I introduce a unified algebraic view on key derivation with PUFs in Section 4.3 that leads to a generic security criterion, presented in Section 4.4. In Section 4.5, this criterion is applied to the linear approaches given in Section 3.3.

This chapter is based on the results published in [HYP15] and [PHS17, HPKS16].

4.1. Relation between PUFs and Compound Sources

This section shows that there is almost a 1:1 correspondence between the theoretical problem of secret key agreement from correlated compound sources and the practical problem of secret key derivation with PUFs, as shown in Figure 4.1. While some previous work only evaluated one source distribution [TG04, IW10], the compound source model introduced in [BW13] has an internal state that is able to capture different physical states of a device and output them with different source statistics. More recent work on compound sources can be found in [GBS15, TBS15].

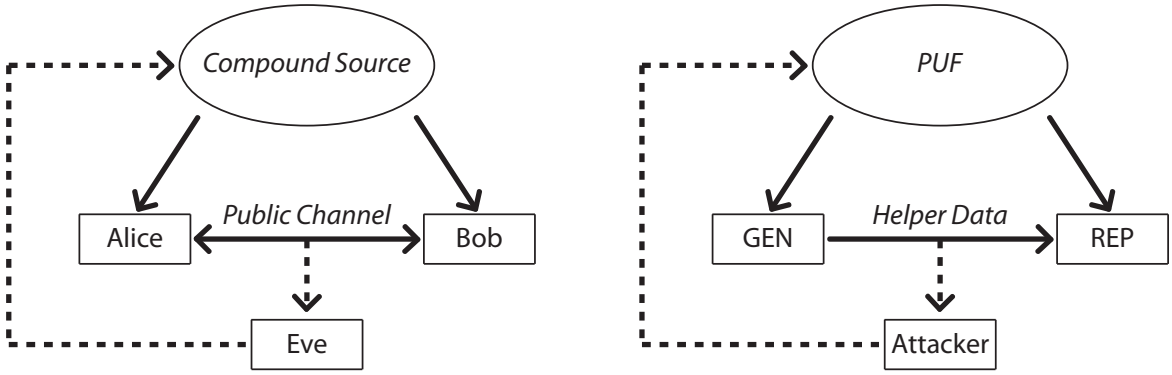

Figure 4.1.: Analogies between the key agreement from a compound source and secret key derivation with a PUF

In the theoretical scenario, Alice acts as enrollment that is performed in the secured manufacturing environment while Bob corresponds to the key reproduction procedure in the field. One difference between the theoretical and the PUF scenario is that Bob typically only derives a session key once while the PUF key generation is carried out multiple times.

The following four aspects characterize major parallels and differences between the information theoretical model and PUFs as application:

Source of Randomness A large sample of PUF ICs behaves like two nested random processes. The first random variable represents the manufacturing process that defines the physical parameters of a specific device. The PUF circuit itself is an instantiation of this random variable. The response sequence depends on the internal physical parameters of the circuit and external physical parameters, or operating conditions which are represented as the state of the source. For reproduction, noise is added which is represented as second random variable. The new response should ideally still be closely correlated with the first sequence output by this specific PUF. The response is read out in the field for each cryptographic key reproduction whenever the key is needed.

Instead of a large batch of devices that generate individual keys, two parties, Alice and Bob, want to generate session keys after each of the two observes one sequence of their common source of randomness. For exactly identical sequences, the problem is trivial. If the sequences are only correlated, Alice and Bob have to exchange messages to be able to perform error correction and extract the joint part.

Since the derivation takes place only once, the first sequence refers to a fixed state and the index is dropped in the following. Using multiple measurements during enrollment further enhances the performance of the system [GK16] but is not considered in the following.

Communication between Parties In both cases, the exchanged data enables error correction. The information theoretical scenario assumes an authenticated, noiseless, potentially bi-directional communication between Alice and Bob. However, note that the optimal protocol discussed in [BW13] only requires one-way communication.

For PUFs, the helper data enables the communication between generation and reproduction.

Influence of the Attacker In both scenarios, the attacker can set the state of the source to make the key generation as inefficient as possible for Alice and Bob. In the PUF case, the PUF operates under different environmental conditions and still has to work reliably. This represents different operation conditions and physical factors such as temperature, external voltage or age that are represented in the state of the compound source.

While the information theoretic scenario assumes an authenticated channel [BW13], the helper data can be manipulated by the attacker in the PUF case. Various attacks are discussed in [DV14a, DV14b, DGSV15] and helper data manipulation will also be addressed in Section 6.4.2.

Physical attacks such as power, electro-magnetic and photonic side-channel attacks [KS10, MSSS11, MHH+13, Mer14, TNS+14], laser fault attacks [TLG+15] and invasive attacks [NHSB13, HNT+13] are practical threats that are not addressed explicitly in this work. Here, it is also important to keep in mind that silicon PUFs are mainly aiming to secure a standard CMOS circuit where meshes and other advanced security features used in smart cards are not available. The PUF should not allow additional attacks but it is also an unrealistic desire (or promise) to solve all problems by adding a PUF.

Distance between Parties The parties involved in the key generation are separated in some sense. The information theoretical scenario typically makes the assumption that Alice and Bob are in different locations so that they communicate over a public wireless or cable connection. In contrast, two responses of the same PUF are separated in time. This has the practical limitation that no backward communication from Bob to Alice is possible. However, the optimal coding scheme in [BW13] does not require backward communication so that this limitation only has minor practical impact and the scenarios are still very similar. Assuming that the adversary has unlimited computational power, the security requirements for both key generation scenarios are identical.

For helper data manipulation, the separation over time is more advantageous for the attacker because he can spend more time to compute manipulated helper data. Interfering in a communication over space as man in the middle is more difficult because intercepting the original helper data and sending the manipulated one has to occur before a time-out signal is sent.

	Information Theoretical Model	Key Derivation with PUFs
Source of Randomness	Compound Source	Batch of PUFs
Communication between Parties	Public Communication Channel	Public Helper Data
Influence of Attacker	State of the Compound Source	Operating Conditions of PUF
Distance between Parties	Space	Time

Table 4.1.: Comparison of key derivation with a PUF and secret key agreement with a compound source

Table 4.1 briefly wraps up the previous discussion. It shows that secret key derivation with PUF is a specific case of the information theoretical problem of secure agreement from a compound source.

4.2. Review of the Information Theoretical Criteria and Limits

Figure 4.1 already gave a first, high-level overview over the problem of secret key generation with a compound source. In the following, the more detailed version in Figure 4.2 also contains the variable names. The discrete memoryless multiple compound source [BW13] with correlated output sequences has an internal state t that is element of the set $\mathcal{T}$ containing all possible states. Alice observes the outcome $x \in \mathcal{X}$ of random variable X_t with marginal probability distribution $P_{X,t}(x)$ and Bob has access to $y \in \mathcal{Y}$ drawn from random variable Y_t with marginal probability distribution $P_{Y,t}(y)$. The joint probability distribution is given by $P_{XY,t}(x, y)$. Eve is completely ignorant of X_t and Y_t but has access to the source in a sense that she can set the state t.

Alice and Bob establish keys K and L from the same key space $\mathcal{K}$, with $K, L \in \mathcal{K}$ after sending helper data W over the public channel with the goal that $H({X_t}^n|{Y_t}^n W)$ goes to zero. If this is the case, ${X_t}^n$ is fully determined by ${Y_t}^n$ and W. The secret key agreement protocols by Ahlswede and Cziszar [AC93], and Boche and Schfer [BW13] are discussed in Appendix A.1.

In theory, Alice and Bob both observe marginals of n output pairs (x, y) of the source that are drawn independently. A memoryless source that outputs independent bits is a lot easier to handle and this simplification also holds in good approximation in practice for PUFs. It was shown, e.g. in [KKR+12], that popular PUF types such as the SRAM PUF or RO PUF can have nearly independent bits if they are implemented properly.

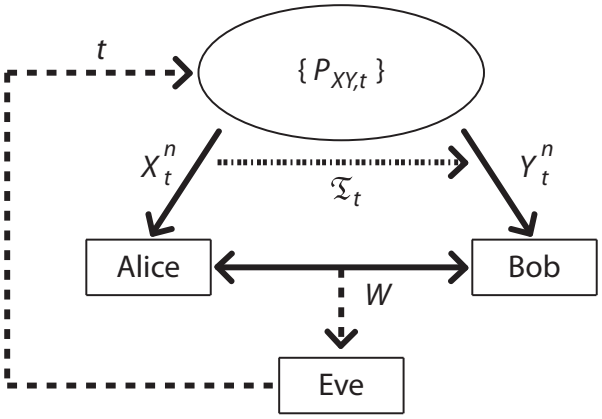

Figure 4.2.: Secret key generation with a compound source

Alice's observation X_t and Bob's observation Y_t have $I(X_t;Y_t)$ in common so that Bob only has to know $X_t|Y_t$ to be able to reproduce the correct X_t. Instead of drawing Y_t directly from the source, Y_t can be represented by the outcome of a probabilistic memoryless channel $\mathfrak{T}_t$ between Alice and Bob with input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. The channel behavior is represented with the conditioned probability distribution $P_{Y_t|X,t}$. For one key agreement, one fixed state $t \in \mathcal{T}$ is considered and the channel $\mathfrak{T}_t$ is defined by the set of the conditional probabilities for all (x, y) pairs.

$$\mathfrak{T}_t = \left\{ P_{Y|X,t}(x,y) = \frac{P_{XY,t}(x,y)}{P_{X,t}(x)} : x \in \mathcal{X}, y \in \mathcal{Y} \right\} \tag{4.1}$$

Note that in contrast to a channel coding problem such as the compound wiretap channel [LKPS09] it is not possible to optimize the properties of the transmitted sequence for the specific channel, since X_t is determined by the source and cannot be modified.

Definition of an Achievable Rate To be able to evaluate if a key agreement protocol is good or not, the following achievable rate definition was introduced in [AC93] and applied to biometrics, for example in [TG04]. The definition basically tells that the keys of Alice and Bob have to be equal with a high probability, the helper data must not leak substantial information about the key and the entire key space has to be used.

More formally, an *achievable key rate* R_{key} specifies the amount of secret information that can be derived reliably and securely from each (x, y) pair for large n. The rate is a relative measure that is defined by key size divided by the length n of the PUF response.

Ahlswede and Csiszar [AC93] defined that a key rate R_{key} is an achievable key rate if the following four conditions hold for an $\epsilon > 0$:

$$\Pr[K \neq L] < \epsilon \tag{4.2}$$

$$\frac{1}{n} I(W;K) < \epsilon \tag{4.3}$$

$$\frac{1}{n} H(K) > R_{key} - \epsilon \tag{4.4}$$

$$\frac{1}{n} \log_2 |\mathcal{K}| < \frac{1}{n} H(K) + \epsilon \tag{4.5}$$

In [BW13], Boche and Schäfer use a stronger security notion, called strong secrecy [MW00, CN08], where Condition 4.3 is replaced by

$$I(W;K) < \epsilon \tag{4.6}$$

and also extend the general notion to compound sources such that the equations have to hold for every $t \in \mathfrak{T}$. Going from Condition 4.3 to the tightened version in 4.6 is important to strengthen asymptotic results. In this work, the absolute ϵ or $n \cdot \epsilon$ value is of major interest since PUFs are operating in a relatively small, finite block-length regime.

Adapted from [BW13], the *achievable helper data rate* R_{hd} is for an achievable key rate R_{key} fulfilling Conditions 4.2 to 4.5 is given by

$$R_{hd} = \frac{\log_2(|\mathcal{W}|)}{n} \tag{4.7}$$

Interpretation of Conditions 4.2 to 4.5 Condition 4.2 quantifies that the probability of generating an incorrect key L is smaller than ϵ and ensures the reliability of a protocol. In practice, the error probability in Condition 4.2 can be quantified for example by bounding techniques, e.g. [Bos99, CT06], or Monte Carlo simulations.

Security is ensured by Condition 4.3 or the stricter version in 4.6 because it bounds the amount of key information K that leaks through the helper data W. The ϵ in Condition 4.3 is quite hard to quantify. While $H(X|W)$, analyzed in detail in [DGV+16], can serve as starting point, general properties of $I(W;K)$ will be discussed later in Section 4.3.

The rate R_{key} quantifies the performance of a scheme. To make it a meaningful quantity for the entropy of the generated key, Condition 4.4 tells that the entropy of the key must not be much smaller than rate R_{key}. The entropy $H(K)$ in Conditions 4.4 and 4.5 can be evaluated, for example with universal source coding algorithms such as context tree weighting [WST95] or the Lempel-Ziv algorithm[ZL77], or with randomness tests such as the NIST test suite for random number generators [RSN+10].

Aiming to generate cryptographically strong keys, they should use the entire key space, which is specified in Condition 4.5. It states that the key space must only be slightly larger than the space that is filled by instantiations of K.

Capacity Definitions Information theory is generally interested in specifying the ultimate achievable rates of a problem and to reach them asymptotically for block lengths going to infinity. The ultimate rate that captures the limits of a system is called capacity C and ideally an achievable rate R should approach capacity for large n with ϵ going to zero. Depending whether a capacity C is defined as upper or lower bound of a measure, it is defined as the supremum or infimum over all achievable rates R.

The supremum over all achievable key rates R_{key} gives the key capacity C_{key} of a compound source. Boche and Schäfer have proven in [BW13] that the key capacity is the minimum mutual information between both outcomes X_t and Y_t of the source over all possible states $\mathcal{T}$.

$$C_{key} = \sup_{R_{key} \text{ is achievable rate}} R_{key} \tag{4.8}$$

$$= \min_{t \in \mathcal{T}} I(X_t, Y_t) \tag{4.9}$$

The decoder knows Y_t, so it takes in average $H(X_t|Y_t)$ bits to reconstruct X_t from Y_t with joint distribution $P_{XY,t}$ and state t. However, $H(X_t|Y_t)$ varies over t. The maximum conditioned entropy over all states is the helper data capacity C_{hd} because this quantifies the minimum amount to reliably reconstruct X_t from Y_t also in the worst-case state.

$$C_{hd} = \max_{t \in \mathcal{T}} H(X_t|Y_t) \tag{4.10}$$

See again [BW13] for the proof. The capacities allow to evaluate schemes not only in comparison to previous work. They also give designers quantitative information on how far a scheme can still be improved until the absolute limits are reached and how far the state of the art is still away from these limits.

4.3. Unified Algebraic View on Secure Key Derivation with PUFs

Most helper data generation schemes discussed in Chapter 3 are given in an algorithmic description which is close to implementation. However, it is hard to see fundamental theoretical properties from this description. The unified algebraic description of the

problem, introduced in this section, reveals more information on the underlying structure, and especially facilitates to analyze the secrecy leakage in Condition 4.3 in general only by evaluating algebraic properties of the coding scheme.

The new algebraic representation can represent linear schemes as the most-widely used class covering, e.g. [DFM98, JW99, DRS04] and [HYP15]. Only very few exceptions that do not show a regular algebraic structure such as [HWRL+13, YHD15] are hard to analyze with this approach. An extended version of the algebraic representation is discussed in [PHS17] also covers IBS [YD10b] and C-IBS [HMSS12].

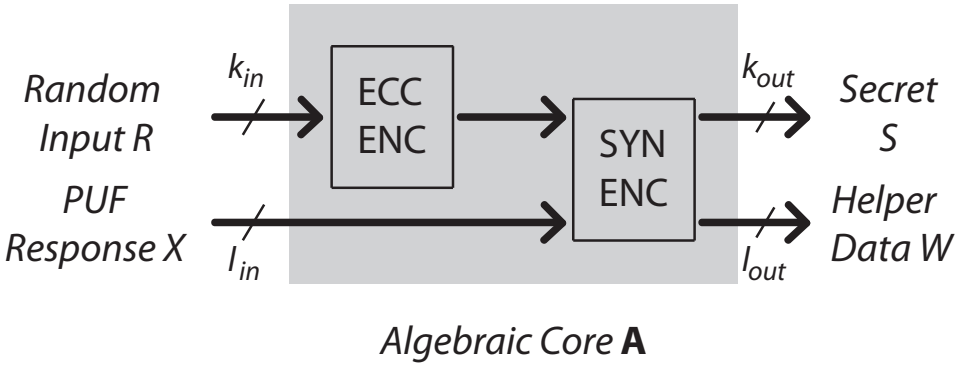

Figure 4.3.: Algebraic view on secret key and helper data generation with a PUF during enrollment

Figure 4.3 shows a generic high-level block diagram of secret key and helper data generation. This section looks at generic properties of vectors and the vector lengths are dropped in this section for simpler representation and improved readability. They are added again in the following case study to highlight overlaps and differences between the different approaches. The PUF response X is a mandatory input to bind a key to a PUF. In addition, some approaches also have a random number R as input that is only necessary for helper data generation. In principle, it can be generated externally in the secure manufacturing environment to avoid an on-chip True Random Number Generator (TRNG). The random number can be used directly as embedded key or as additional input if the PUF response is used as secret.

The binary PUF response X can be extended by reliability information, gathered from additional analog PUF response information or repeated measurements. This can be used for example to select the most stable PUF response bits with dark bit masking [AMS+09a] to reduce the number of errors that have to be corrected by the error correction. It can be also used for indexing operations to increase reliability and security at the same time for example in [YD10b, HMSS12]. These additional algorithmic steps can be represented in a *Preprocessing Matrix* $\mathbf{M}_{pre}$ or *Postprocessing Matrix* $\mathbf{M}_{post}$, discussed in detail in [PHS17].

The secret S and the helper data W are the two necessary outputs of each scheme. In addition, optional reliability information on PUF response bits X can be stored in the

helper data to enable soft-decision decoding to increase the reliability when the secret is computed later in the field [MTV09a, HMSS12].

The helper data generation schemes with linear mappings discussed in Section 3.3 have the following generic form in common with the *Algebraic Core* $\mathbf{A}$ in the center.

$$[S, W] = [R, X] \ \mathbf{M}_{pre} \ \mathbf{A} \ \mathbf{M}_{post} \tag{4.11}$$

The inputs on the left side of the matrices are the random number R with dimensions $\langle 1 \times k_{in} \rangle$ and the PUF response X with dimensions $\langle 1 \times l_{in} \rangle$. As preprocessing step such as dark bit masking [AMS+09a], the inputs are multiplied with the Preprocessing Matrix $\mathbf{M}_{pre}$. The multiplication with the Algebraic Core $\mathbf{A}$ is the most important encoding operation where all interaction between R and X is performed. No interaction between the preliminary versions of S and W are allowed to take place in the Postprocessing Matrix $\mathbf{M}_{post}$. Postprocessing steps such as the indexing operations in [YD10b, HMSS12] are captured in $\mathbf{M}_{post}$. The final results S and W are of size $\langle 1 \times k_{out} \rangle$ and $\langle 1 \times l_{out} \rangle$.

The Algebraic Core $\mathbf{A}$ reveals fundamental security properties. Therefore, $\mathbf{M}_{pre}$ and $\mathbf{M}_{post}$ in Eqn. 4.11 are removed in the following, so Eqn. 4.11 simplifies to

$$[S, W] = [R, X] \ \mathbf{A} \tag{4.12}$$

$\mathbf{A}$ is split into a left part $\mathbf{A}_L$ that outputs the secret S and a right part $\mathbf{A}_R$ that determines the helper data W. The random input R is multiplied with the upper part $[\mathbf{A}_{UL} \ \mathbf{A}_{UR}]$ of $\mathbf{A}$ and the PUF response X is multiplied with the lower part $[\mathbf{A}_{LL} \ \mathbf{A}_{LR}]$. As a result, $\mathbf{A}$ in Eqn. 4.12 decomposes to four sub-matrices of interest that will be analyzed in the remainder of this chapter.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_L & \mathbf{A}_R \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{UL} & \mathbf{A}_{UR} \\ \mathbf{A}_{LL} & \mathbf{A}_{LR} \end{bmatrix} \tag{4.13}$$

Figure 4.4 provides the dimensions of the sub-matrices of $\mathbf{A}$ given in Eqn. 4.13.

As discussed in Section 4.1, authenticated helper data cannot be guaranteed so that the helper data is prone to tampering in general. To prevent this class of attacks, described e.g. in [HWRL+13] and [DGSV15], a hash function can be used to hash the helper data and XOR it with the secret as key $K = S \oplus f(W)$ [HWRL+13]. Using $S \oplus f(W)$ instead of $f(S, W)$ has the advantage that the hash function only operates on public data, whereas $f(S, W)$ would also be an interesting target for side-channel attacks. In addition, only the diffusion property of the hash function is actually required and not the stronger one-way property so that simpler algorithms could already achieve that necessary security level.

The helper data of some schemes leak secret information. A hash function is added to compress the remaining entropy $H(S|W)$ that is spread over the entire secret S to a

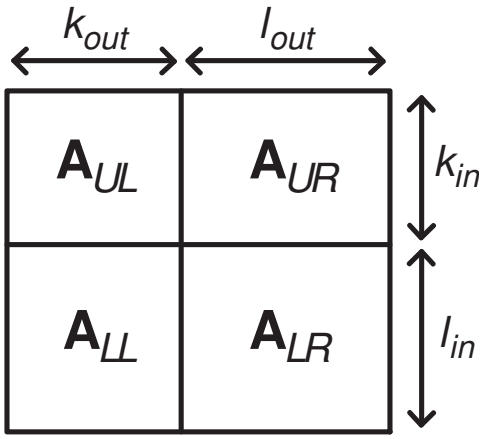

Figure 4.4.: Dimensions of the sub-matrices of the Algebraic Core

shorter vector $K = f(S)$ [DRS04] such that $H(K) > k_{out} - \epsilon$ for key size k_{out} and a small $\epsilon > 0$. It was shown in [DRS04] that information theoretically secure keys can be generated if a universal hash function is used. Typically, hardware implementations only have a regular cryptographic hash function instead of the universal hash function. Compressing S to K comes at the expense that the hash loss has to be taken into account [BDK+11] which requires a larger secret S. In addition, the hash function adds computational complexity and a possible target for side-channel attacks, such as in [MSSS11]. Therefore, one aim of the pursuit is to avoid the hashing of secret data whenever possible.

4.4. Generic Security Criterion

As given by Condition 4.3 and 4.6 in Section 4.2, the mutual information $I(S;W)$ has a critical impact on the security of the secret key. The goal is that S and W are uncorrelated such that $I(S;W) < \epsilon$ holds for a small ϵ. I will show that this is only achievable if the rank of the entire Algebraic Core $\mathbf{A}$ is equal to the sum of the ranks of the secret generating part $\mathbf{A}_L$ and the helper data generating part $\mathbf{A}_R$. In the following the *rank loss* Δ will be introduced as a measure to quantify the difference between the maximum possible rank of a matrix given by its smaller dimension and the actual rank.

Starting with some preliminaries, the size of an output space is defined by its basis. A set of row vectors of a matrix define the basis of a space. In the following, the rank of a matrix as an algebraic measure and the mutual information as an information theoretical measure are linked. The entropy of an output of a vector-matrix multiplication is upper bounded by the rank of the matrix the input is multiplied with.

Let the input sequence be $Y \in \mathbb{F}_2^n$ and the output sequence be $Z \in \mathbb{F}_2^m$. The matrix $\mathbf{A}$ has dimensions $dim(\mathbf{A}) = \langle n \times m \rangle$ and let $q = rank(\mathbf{A})$. Z is given by $Z = Y \cdot \mathbf{A}$. This multiplication is equivalent to one specific mapping of $Y \in \mathbb{F}_2^n$ into a subspace $\mathcal{Q} \subseteq \mathbb{F}_2^m$

with $|\mathcal{Q}| = 2^q$ elements. The Shannon entropy of Z is $H(Z) = q$ if all elements in space $\mathcal{Q}$ occur with the same probability and $H(Z) < q$, otherwise.

In the following, this reasoning will be applied to Eqn. 4.12. Let set $\mathcal{R}$ contain all possible input random numbers R and the set $\mathcal{X}$ contain all possible PUF responses X. On the output side, let $\mathcal{S}$ be the set with all secret keys S and helper data set $\mathcal{W}$ be the set containing all possible helper data values W. According to Eqn. 4.12 the input space $\mathcal{R} \times \mathcal{X} = \mathbb{F}_2^{k_{in}+l_{in}}$ is mapped to the output space $\mathcal{S} \times \mathcal{W} \subseteq \mathbb{F}_2^{k_{out}+l_{out}}$ through the Algebraic Core $\mathbf{A}$. $\log_2(|\mathcal{S} \times \mathcal{W}|)$ is upper bounded by the rank of $\mathbf{A}$, which is equivalent to the maximum number of linearly independent base vectors in the index set of the output space. If an element in $\mathcal{R} \times \mathcal{X}$ occurs with probability zero, the size of the output space is also reduced, so $\log_2(|\mathcal{S} \times \mathcal{W}|) < rank(\mathbf{A})$. Therefore, the maximum size of the index sets will be applied as a measure in the following.

To derive a bound on the secrecy leakage, index sets $\mathcal{I}$ of spaces are defined. An index set contains numbers of rows of a matrix that form a basis. So, the space is given as linear combination of the indexed rows.

Spaces $\mathcal{S}$ and $\mathcal{W}$ have bases formed by rows $A_{L,i}$ and $A_{R,i}$ of $\mathbf{A}_L$ and $\mathbf{A}_R$, respectively. The vectors selected by the elements in each index set $\mathcal{I}$ are linearly independent, so any linear combination of base vectors can be zero only if all coefficients λ_i are zero.

$$\mathcal{I}_L = \left\{ i \in \{1, ..., k_{in} + l_{in}\} \,\middle|\, \sum_i \lambda_i \cdot A_{L,i} = 0 \Leftrightarrow \forall i : \lambda_i = 0 \right\} \tag{4.14}$$

$$\mathcal{I}_R = \left\{ i \in \{1, ..., k_{in} + l_{in}\} \,\middle|\, \sum_i \lambda_i \cdot A_{R,i} = 0 \Leftrightarrow \forall i : \lambda_i = 0 \right\} \tag{4.15}$$

Accordingly, the index set $\mathcal{I}$ of the full Algebraic Core $\mathbf{A}$ is:

$$\mathcal{I} = \left\{ i \in \{1, ..., k_{in} + l_{in}\} \,\middle|\, \sum_i \lambda_i \cdot A_i = 0 \Leftrightarrow \forall i : \lambda_i = 0 \right\} \tag{4.16}$$

Note that, $|\mathcal{I}| \leq rank(\mathbf{A})$.

The *rank loss* Δ quantifies the difference between the maximum rank of a matrix defined by its dimensions and the maximum index set size, given by the rank. There are three rank losses of interest: Δ_L referring to $\mathbf{A}_L$, Δ_R referring to $\mathbf{A}_R$ and Δ referring to the entire core $\mathbf{A}$, so

$$\Delta_L = \min\{k_{in} + l_{in}, k_{out}\} - rank(\mathbf{A}_L) \tag{4.17}$$

Analogously, Δ_R and Δ for $\mathbf{A}_R$ and $\mathbf{A}$ are

$$\Delta_R = \min\{k_{in} + l_{in}, l_{out}\} - rank\,(\mathbf{A}_R) \tag{4.18}$$
$$\Delta = \min\{k_{in} + l_{in}, k_{out} + l_{out}\} - rank\,(\mathbf{A}) \tag{4.19}$$

The *minimal rank loss* $g_{dim}\,(k_{in} + l_{in}, k_{out}, l_{out})$ that can occur by the concatenation of two matrices, given by the dimensions, is defined as

$$\begin{aligned} g_{dim}\,(k_{in} + l_{in}, k_{out}, l_{out}) = \min\{k_{in} + l_{in}, k_{out}\} + \min\{k_{in} + l_{in}, l_{out}\} \\ - \min\{k_{in} + l_{in}, k_{out} + l_{out}\} \end{aligned} \tag{4.20}$$

Only if there are rows of $\mathbf{A}_L$ and $\mathbf{A}_R$ that form bases of two complementary vector spaces with dimensions $rank\,(\mathbf{A}_L)$ and $rank\,(\mathbf{A}_R)$, i.e., if the indices of the rows selected for the bases from $\mathbf{A}_L$ and $\mathbf{A}_R$ are different such that

$$\mathcal{I}_L \cap \mathcal{I}_R = \emptyset \tag{4.21}$$

the leakage can be brought to zero. Only then, the outputs S and W can become independent. If the spaces are not complementary, there exists a linear dependency between S and W which leads to secrecy leakage.

The union of the index sets define a basis formed by rows of $\mathbf{A}$.

Lemma: The difference between the minimal rank loss and the actual rank loss bounds the mutual information $I\,(S; W)$ between the secret and helper data.

$$I\,(S; W) \leq g_{dim}\,(k_{in} + l_{in}, k_{out}, l_{out}) \;-\; (\Delta_L + \Delta_R - \Delta) + \epsilon_0 \tag{4.22}$$
$$= rank\,(\mathbf{A}_L) + rank\,(\mathbf{A}_R) - rank\,(\mathbf{A}) + \epsilon_0 \tag{4.23}$$

ϵ_0 depends on the entropy of the input PUF data $H(X)$ and the entropy of the random number $H(R)$. As soon as there is any overlap in the index sets, S and W cannot be independent. The difference $\Delta_L + \Delta_R - \Delta$ is increased by one for each overlapping index so that the leakage of the algebraic core is increased by one accordingly.

Proof: As first step of the bounding, all possible index sets are selected which contain a maximum-sized set of linearly independent rows:

$$|\mathcal{I}_L| = rank\,(\mathbf{A}_L) \tag{4.24}$$
$$|\mathcal{I}_R| = rank\,(\mathbf{A}_R) \tag{4.25}$$

The selected index sets can be used to construct index sets for $\mathbf{A}$.

$$\mathcal{I} = \mathcal{I}_L \cup \mathcal{I}_R \tag{4.26}$$

All these sets are searched for a set $\mathcal{I}$ which is built from non-overlapping sets $\mathcal{I}_L$,$\mathcal{I}_R$. This is ensured iff Eqns. 4.24 and 4.25, Eqn. 4.26 and

$$|\mathcal{I}| = |\mathcal{I}_L| + |\mathcal{I}_R| \tag{4.27}$$

hold. If such a set $\mathcal{I}$ exists, i.e.,

$$\underset{\mathcal{I},\mathcal{I}_L,\mathcal{I}_R}{\exists} (\mathcal{I} = \mathcal{I}_L \cup \mathcal{I}_R) \wedge (|\mathcal{I}| = |\mathcal{I}_L| + |\mathcal{I}_R| = rank\,(\mathbf{A}_L) + rank\,(\mathbf{A}_R)) \tag{4.28}$$

it is ensured, that no information leaks due to the structure of the Algebraic Core. If Eqn. 4.28 cannot be fulfilled, information is leaked. Thus, we claim that Eqn. 4.28 is a necessary and sufficient condition to ensure $I\,(S;W) \leq \epsilon_0$.

Going to the mutual information as actual quantity of interest, $I(S;W)$ can be rewritten as

$$I(S;W) = H\,(S) + H\,(W) - H\,([S\ W]). \tag{4.29}$$

For sources with high entropy, the measures $H(R)$ and $H(X)$ can be set to $H(R) = k_{in} - \epsilon_{in,1}$ and $H(X) = l_{in} - \epsilon_{in,2}$, respectively. This gives ϵ, ϵ_L, and ϵ_R values in

$$\begin{aligned} H\,(S) =& H\,([R\ X]\ \mathbf{A}_L) \\ =& rank\,(\mathbf{A}_L) - \epsilon_L \\ =& \min\{k_{in} + l_{in}, k_{out}\} - \Delta_L - \epsilon_L \end{aligned} \tag{4.30}$$

$$\begin{aligned} H\,(W) =& H\,([R\ X]\ \mathbf{A}_R) \\ =& rank\,(\mathbf{A}_R) - \epsilon_R \\ =& \min\{k_{in} + l_{in}, l_{out}\} - \Delta_R - \epsilon_R \end{aligned} \tag{4.31}$$

$$\begin{aligned} H\,([S\ W]) =& H\,([R\ X]\ \mathbf{A}) \\ =& rank\,(\mathbf{A}) - \epsilon \\ =& \min\{k_{in} + l_{in}, k_{out} + l_{out}\} - \Delta - \epsilon \end{aligned} \tag{4.32}$$

The epsilon parameters equal out partially in Eqn. 4.29 giving the overall loss ϵ_0 with $\epsilon_0 = \epsilon - \epsilon_L - \epsilon_R$. As an upper bound, $\epsilon = \epsilon_{in,1} + \epsilon_{in,2}$ and $\epsilon_L = \epsilon_R = 0$ hold such that $\epsilon_0 \leq \epsilon_{in,1} + \epsilon_{in,2}$. Note that ϵ only depends on the random number and the PUF

implementation, and can be brought down close to zero with good TRNG and PUF implementations [KKR+12].

Inserting the Eqns. 4.30 to 4.32 into Eqn. 4.29 results in the mutual information

$$I(S;W) = H(S) + H(W) - H([S\ W]) \tag{4.33}$$
$$\leq rank(\mathbf{A}_R) + rank(\mathbf{A}_L) - rank(\mathbf{A}) + \epsilon_0 \tag{4.34}$$

The ranks correspond to the maximum sizes of the index sets, so

$$= |\mathcal{I}_L| + |\mathcal{I}_R| - |\mathcal{I}| + \epsilon_0 \tag{4.35}$$

Using Eqns. 4.17 to 4.19 and Eqn. 4.20 gives

$$= g_{dim}(k_{in} + l_{in}, k_{out}, l_{out}) - (\Delta_L + \Delta_R - \Delta) + \epsilon_0 \tag{4.36}$$

Eqn. 4.35 shows that, the mutual information in Eqn. 4.33 can only be brought down close to zero if Eqn. 4.28 holds.

■

4.5. Algebraic Representation and Analysis of the State of the Art

This section derives the Algebraic Cores for the linear schemes presented in Section 3.3. It analyzes the secrecy leakage and discusses which schemes approach capacity for optimal (n, k, ϵ) codes introduced in Section 1.7. The superscripts for vector lengths are picked up again to highlight the differences between the discussed approaches.

All approaches except of the Parity Construction use the same (n, k, ϵ) codes so that the results can be directly compared to each other.

Fuzzy Commitment Let $k_{in} = k_{out} = k$ and $l_{in} = l_{out} = n$. For the Fuzzy Commitment [JW99], R^k is encoded and then output as secret S^n such that $\mathbf{A}_{UL} = \mathbf{G}$ and $\mathbf{A}_{LL} = \mathbf{0}$. The product $R^k\ \mathbf{G}$ is XORed with X^n to form the helper data W^n so that $\mathbf{A}_{UR}$ is set to the generator matrix $\mathbf{G}$ while $\mathbf{A}_{UL}$ is the identity matrix $\mathbf{I}$. Let $\gamma(\cdot)$ be the decoding operation of the ECC. Then, the resulting equation in matrix form is

$$[S^n \; W^n] = \left[R^k \; X^n\right] \overbrace{\begin{pmatrix} \mathbf{G} & \mathbf{G} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}}^{\mathbf{A}} \tag{4.37}$$

$$K^k = \gamma(S^n) \tag{4.38}$$

For the modified version with $K^k = S^k = R^k$, presented in [TAK+05], $\mathbf{A}_{UL} = \mathbf{G}$ is replaced by $\mathbf{I}$, resulting in

$$\left[S^k \; W^n\right] = \left[R^k \; X^n\right] \begin{pmatrix} \mathbf{I} & \mathbf{G} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{4.39}$$

The difference between the two approaches is that [JW99] returns the codeword as secret while [TAK+05] returns the random input.

In Eqn. 4.37, $\mathbf{A}_L$ has rank k limited by the number of rows of $\mathbf{G}$ while $\mathbf{A}_R$ has rank n given by the number of columns. The full Algebraic Core has rank $k + n$ so that the index set $\mathcal{I}$ contains $k + n$ values fulfilling Eqn. 4.28. The Algebraic Core $\mathbf{A}$ of the Fuzzy Commitment in Eqn. 4.39 is an upper triangular matrix with full rank such that Eqn. 4.28 holds immediately. As a result, the secrecy leakage of the Fuzzy Commitment only depends on the joint entropy of the PUF response X^n and of the random number R^k.

If the joint entropy $H([X^n \; R^k])$ is sufficiently high, the secret S^k can directly be used as a key K^k. If $H(S^k) << k$, S^k can be compressed to a smaller key K^{k^*} with $k^* < k$ that fulfills Condition 4.5. The helper data has a fixed size of n which is larger than necessary.

For a capacity achieving (n, k_n, ϵ) code, up to k_n secret bits can be derived. For large n, approaches k_n/n approaches capacity, so

$$\lim_{n\to\infty} R_{key} = \lim_{n\to\infty} \frac{k_n}{n} = C_{key} \tag{4.40}$$

Since one helper data bit is stored for each key bit, $R_{hd} = 1$.

Code-Offset Fuzzy Extractor The Code-Offset Fuzzy Extractor [DRS04] shows several parallels to the Fuzzy Commitment. Let $k_{in} = k$ and $k_{out} = l_{in} = l_{out} = n$. Instead of using the random number as secret, the PUF response is hashed and output as key. Therefore, $S^n = X^n$, $\mathbf{A}_{UL} = \mathbf{0}$ and $\mathbf{A}_{LL} = \mathbf{I}$ while the right side of the algebraic core $\mathbf{A}$ is $\mathbf{A}_{UR} = \mathbf{G}$ and $\mathbf{A}_{LR} = \mathbf{I}$ and thus remains the same as in the Fuzzy Commitment. Eqn. 4.42 is added to compress the n-bit PUF response to a k-bit key.

$$[S^n \; W^n] = \left[R^k \; X^n\right] \begin{pmatrix} \mathbf{0} & \mathbf{G} \\ \mathbf{I} & \mathbf{I} \end{pmatrix} \tag{4.41}$$

$$K^k = f\left(S^n\right) \tag{4.42}$$

Both parts $\mathbf{A}_L$ and $\mathbf{A}_R$ of the core $\mathbf{A}$ in Eqn. 4.41 have full rank such that $\Delta_L = 0$ and $\Delta_R = 0$. However, their index sets overlap. First, the lower n rows of $\mathbf{A}_L$ are assigned to the left index set $\mathcal{I}_L$. For $\mathcal{I}_R$, k linearly independent vectors are given by $\mathbf{G}$ but the remaining $n-k$ linearly independent rows of $\mathbf{A}_R$ are already used for $\mathcal{I}_L$. So, $\Delta = n - k$ and up to $n - k$ bits leak, which is consistent to literature such as [IW12]. Since the attacker knows the helper data, the entropy of the n-bit long secret is reduced to k. As a consequence the hash function in Eqn. 4.41 has to be designed such that the remaining k bit of entropy are distributed equally to the bits of an k bit long key. Note that also the entropy loss due to hashing has to be considered to derive information theoretically secure keys [MVHV12].

For the Fuzzy Commitment, the codeword is masked with the PUF response such that it forms a secure one-time pad for a high-entropy PUF. For the Code-Offset Fuzzy Extractor, the PUF response is masked with the codeword resulting in an imperfect one time pad because by definition, not all bits in the codeword are independent. This small difference leads to a different secrecy leakage.

Similarly to the Fuzzy Commitment, k secret bits are derived. However, when taking the hash loss into account, less bits can be derived. Therefore

$$\lim_{n\to\infty} R_{key} \leq C_{key} \tag{4.43}$$

Again, one helper data bit is stored per PUF response bit, so $R_{hd} = 1$.

Syndrome Construction The the Syndrome Construction also was introduced in [DRS04]. Since no random number R^k is used, the two upper sub-matrices of $\mathbf{A}$ are set to zero. It can be seen that all PUF response bits contribute to the helper data and also to the key. The unified algebraic representation of the Syndrome Construction is given by

$$\left[S^n \; W^{n-k}\right] = [0 \; X^n] \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{H}^T \end{pmatrix} \tag{4.44}$$

$$K^k = f\left(S^n\right) \tag{4.45}$$

For PUF size n and k secret bits, this approach uses only $n - k$ bits of helper data which is the lowest possible number for a given error-correcting code and thus the best

possible solution. In Eqn. 4.44, $rank\,(\mathbf{A}_{LL}) = n$ and $rank\,(\mathbf{A}_{LR}) = n - k$. The index sets overlap fully so that $\Delta = n - k$. Therefore, the maximum leakage is equivalent to the one of the Code-Offset Fuzzy Extractor and again a hash function is required.

The key rate of the Syndrome Construction and the Code-Offset are equal, so

$$\lim_{n\to\infty} R_{key} \leq C_{key} \tag{4.46}$$

However, they differ in the helper data rate. The (n, k_n, ϵ) code has redundancy $n - k_n$, so

$$\lim_{n\to\infty} R_{hd} = \lim_{n\to\infty} \frac{n - k_n}{n} = H(X|Y) = C_{hd} \tag{4.47}$$

Parity Construction Instead of storing the syndrome, the construction in [DFM98] stores the parities of the PUF response. The entire PUF response is interpreted as information to be encoded with an ECC with systematic encoding with $\mathbf{G} = (\mathbf{I}\ \mathbf{P})$, including the parity part $\mathbf{P}$.

As for the Syndrome Construction, the secret and the helper data are computed from the PUF response while no external secret is used. Therefore, $\mathbf{A}_{UL}$ and $\mathbf{A}_{UR}$ are both set to zero again. Here, the hash function compresses the k-bit PUF response to a smaller k^*-bit key. So, the unified description is

$$\left[S^k\ W^{n-k}\right] = \left[0\ X^k\right] \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{P} \end{pmatrix} \tag{4.48}$$

$$K^{k^*} = f\left(S^k\right) \tag{4.49}$$

In Eqn. 4.48, the rank of $\mathbf{I}$ is equal to the length of the secret, so $rank\,(\mathbf{I}) = k$. $\mathbf{P}$ has rank $n - k$. As for the previous scheme, the index sets fully overlap such that $rank\,(\mathbf{A}) = k$. The mutual information is given by $I\,(S; W) = k - (n - k) + \epsilon_0 = 2k - n + \epsilon_0$ and again, $2k - n$ bits leak so that a hash function is required.

Note that in Eqn. 4.48 only $2k - n$ secret bits remain so that this approach is only suitable for small redundancies $n - k$ such that $2k - n > 0$ still holds and not the entire secret is leaked through the helper data.

In [DGV+15] it was shown that the leakage of this approach is significantly higher than the leakages of the other approaches in this comparison, so $\lim_{n\to\infty} R_{key} < C_{key}$ and $\lim_{n\to\infty} R_{hd} < C_{hd}$.

Key Derivation Scheme	$\max R_{key}$	$\min R_{HD}$	Δ	$I(S;W)$	$I(S;W)$ (perfect PUF)
Fuzzy Commitm.	C_{key}	1	0	$H(W^n) - H(X^n)$	$< \epsilon_0$
Code-Offset	$\leq C_{key}$	1	$n-k$	$H(W^n) - H(C^n)$	$< n-k+\epsilon_0$
Syndrome Constr.	$\leq C_{key}$	C_{hd}	$n-k$	$H(W^{n-k})$	$n-k$
Parity Constr.	$< C_{key}$	$< C_{hd}$	$2k-n$	$H(W^{n-k})$	$2k-n$

Table 4.2.: Key rates, helper data rates and mutual information between S and W of the state-of-the-art syndrome coding approaches for PUFs

Summary on State-of-the-Art Syndrome Decoders Linear state-of-the-art helper data generation schemes can be brought into a unified algebraic form which allows a comparison of the individual properties. Wrapping up the results of this section, Table 4.2 provides an overview over the properties of the discussed approaches.

The first two columns show whether the approaches can achieve the capacities provided in Section 4.2 for optimal ECCs.

In general, it is difficult to simplify sums of entropies. Therefore, generic leakages are given first before discussing leakages for a *nearly perfect* PUF with $H(X^n) \approx n$ that show the optimal corner case. Preprocessing can support to achieve such high entropies.

The right-most column on leakage clearly shows that the Fuzzy Commitment, which is the only one with an algebraic cores with full rank, does not leak significant secret information.

The new criterion allows to already evaluate solutions in a very early design stage and give feedback whether an algorithm can achieve zero leakage or not. The rank loss difference Δ gives an upper bound for the secrecy leakage and therefore specifies the minimum requirements for a subsequent hash function.

This work provides a generic property that allows to analyze new more complex and potentially more efficient, practical structures with less obvious leakages in future work. Especially, the currently very regular matrix structures with many identity matrices can be extended to other constructions under the constraint of keeping the rank loss low.

The state-of-the-art approaches output either R^k or X^n as S. Therefore, either $\mathbf{A}_{UL} = \mathbf{I}$ or $\mathbf{A}_{LL} = \mathbf{I}$, while the second sub-matrix of $\mathbf{A}_L$ is set to zero. Afterwards, S is either directly output as a key or fed into a hash function if $k_{out} > H(S|W)$.

4.6. Conclusions

This chapter has shown that secret key derivation with PUFs corresponds closely to the information theoretic problem of secret key agreement from a correlated source. The

new generic security criterion based on the algebraic core revealed weaknesses in several linear state-of-the-art schemes.

As a result, previous work on secure key derivation with PUFs is either able to achieve zero leakage or helper data capacity. The next chapter introduces Systematic Low Leakage Coding which is the first practical approach to combine zero leakage and a helper data size close to capacity.

Chapter 5.

Systematic Low Leakage Coding

Analyzing the state of the art has shown that previous work did not achieve helper data capacity without secrecy leakage. This chapter addresses this shortcoming and introduces Systematic Low Leakage Coding (SLLC)[1], a construction that achieves both criteria at once.

SLLC is introduced in Section 5.1. Section 5.2 addresses the theoretical properties of SLLC. An implementation sketch for a SLLC and a BCH code is discussed in Section 5.3.

This chapter is based on [HYP15] and Mandel Yu contributed especially to the implementation part where the ASIC gate counts are based on the BCH decoder implementation by Verayo that was also used in [YSS+12].

The random coding constructions by Ahlswede and Csiszar [AC93] and Boche and Schäfer [BW13], discussed in Appendix A.1, show that the theoretical bounds in Chapter 4 are achievable in principle for large block lengths and random codebooks. This work goes a step further towards practice and presents a fully linear new syndrome coding scheme that does not have to store the random codebooks anymore. The new approach can be seen as a special case of [AC93] so that the theoretical considerations and results are still valid. The random codebook generation is replaced with a deterministic procedure where all codes are derived from one parent code. Also possible hardware implementations on ASIC and FPGA are discussed.

As a prerequisite, it is assumed that (n, k, ϵ) code $\mathcal{C}$ achieves a rate of R_{Code} for the channel such that decoding errors occur with a probability $\Pr[K \neq L] < \epsilon$. It is also necessary that there exists a systematic encoding scheme for code $\mathcal{C}$ such that for all codewords $C \in \mathcal{C}$, the first k bits are equal to the information bits $c_1^k = x_1^k$. Systematic encoders exist for many popular practical code classes, such as BCH, Reed–Solomon, convolutional, low-density parity-check codes and many other code classes, see e.g. [MS77, Bos99].

[1] The SLLC construction was found independently by Hyonho *et al.* [HHK+14]. I submitted a first version of [HYP15] to CHES 2014 three weeks before the conference proceedings containing [HHK+14] were accessible on IEEE Xplore [IEE15]. I was informed about the existence of the paper in April 2015. Please note that [HHK+14] only presented the construction without the theoretical background and depth of analysis of this work.

5.1. SLLC Code Construction

As starting point, a high-level practical introduction motivates SLLC and gives a first impression before going into the theoretical details. Figure 5.1 shows the SLLC encoding. The PUF response is split into an information part $PUFi$ and mask $PUFm$. The redundancy RED is computed from PUF_I with the encoder of an ECC with systematic encoding.

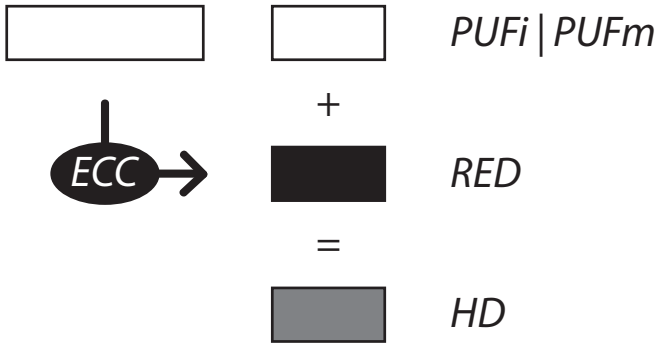

Figure 5.1.: Sketch of SLLC helper data generation

The SLLC decoding in Figure 5.2 shows that the PUF response PUF' is mapped back to the codeword with errors $PUFi'|RED'$ first, and then corrected to $PUFi$. The corrected information part is then output as key.

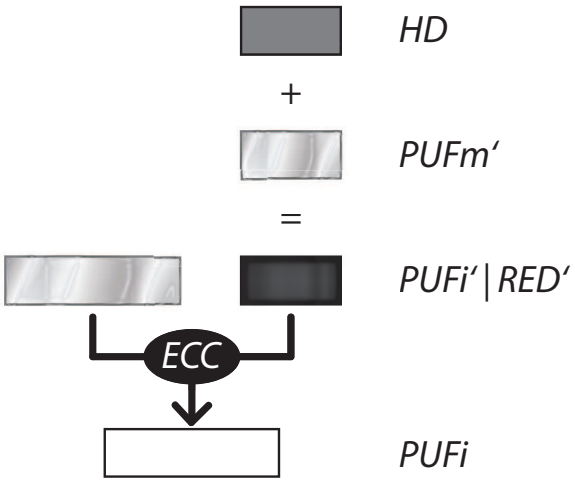

Figure 5.2.: Sketch of SLLC secret key reproduction

[AC93] serves as starting point to formally introduce SLLC. Therefore, first 2^k codebooks are generated with 2^{n-k} elements each.

In the following, one (n, k, ϵ) block code with systematic encoding creates all other codes as cosets of the basic code. For the i-th coset of the code $\mathcal{C}$, the binary representation

of i is XORed on the last $n-k$ bits of each codeword to create code $\mathcal{C}_i$. Iterating over i from 0 to $2^{n-k}-1$ assigns exactly one code $\mathcal{C}_i$ to each element in $\mathbb{F}_2^n$.

Let $\mathbf{I}_{n-k}$ be the $((n-k)\times(n-k))$ identity matrix and $\mathbf{P}$ an $(k\times(n-k))$ matrix. For systematic encoding, the linear code $\mathcal{C}$ has a parity check matrix $\mathbf{H}$ in the form [Bos99]

$$\mathbf{H} = ([\mathbf{P}^T\ \mathbf{I}_{n-k}]) \tag{5.1}$$

and a generator matrix $\mathbf{G}$ with

$$\mathbf{G} = ([\mathbf{I}_k\ \mathbf{P}]) \tag{5.2}$$

Let φ: $\mathbb{F}_2^k \mapsto \mathbb{F}_2^n$ be the encoder of code $\mathcal{C}$ and $b_k(l)$ the binary representation of l in $\mathbb{F}_2^k$. The mother code $\mathcal{C}_0$ is defined as

$$\mathcal{C}_0 = \left\{\varphi(b_k(l))\ :\ l = 0, ..., 2^k-1\right\} \tag{5.3}$$

All other codes $\mathcal{C}_i$, $i = 1, ..., 2^{n-k}-1$ are derived from $\mathcal{C}_0$ by adding a constant offset to all codewords

$$\mathcal{C}_i = \left\{\varphi(b_k(l)) \oplus [0^k\ b_{n-k}(i)]\ :\ l = 0, ..., 2^k-1\right\} \tag{5.4}$$

Given the n-bit PUF response X^n, the helper data W^{n-k} is generated by storing code index i.

$$W^{n-k} = b_{n-k}(i) = \left[\varphi(X^k) \oplus X^n\right]_{k+1}^n \tag{5.5}$$

Since for all codes with systematic encoding,

$$[\varphi(X^k)]_1^k = X_1^k \tag{5.6}$$

the $n-k$ least significant bits return the binary representation of i. This representation separates the secret key part X_1^k from the redundancy part X_{k+1}^n. The operation $\varphi(X^k) \oplus X^n$ can be interpreted as masking the redundancy $[\varphi(X^k)]_{k+1}^n$, that leaks key information, with fresh PUF bits X_{k+1}^n.

Using indices from 0 to $2^{n-k}-1$ covers the entire output space $\mathcal{X}^n$ such that there exists a code $\mathcal{C}_i$ for all $X^n \in \mathbb{F}_2^n$ such that $X^n \in \mathcal{C}_i$.

$$X^n \in \bigcup_{i=0}^{2^{n-k}-1} \mathcal{C}_i \tag{5.7}$$

holds because

$$\bigcup_{i=0}^{2^{n-k}-1} \mathcal{C}_i = \mathbb{F}_2^n. \tag{5.8}$$

Therefore, Eqn. A.2 holds with $\eta = 0$.

In contrast to the state-of-the-art approaches, the first k bits of the corrected PUF response can be used directly as secret key without hashing if $H(X^k) > k - \epsilon$ and $I(X^k; X_{k+1}^n) < \epsilon$, i.e., if the PUF is well designed. If the entropy of X^k is lower, a standard data compression function g, see e.g. [CT06], can be used to compress the corrected PUF response to a key $K = g(X^k)$ such that $H(K) > l - \epsilon$ for a small $\epsilon > 0$.

During reproduction, $\hat{Y}_t^k$ is reconstructed from Y_t^n and $W^{n-k} = b_{n-k}(i)$. Let $\gamma = {\varphi_n}^{-1}$: $\mathbb{F}_2^n \mapsto \mathbb{F}_2^k$ be the decoder of the code $\mathcal{C}$.

$$\hat{Y}_t^k = \gamma(Y_t^n \oplus [0^k \; b_{n-k}(i)]) \tag{5.9}$$

Remark In a typical communications scenario, the cosets are used to characterize the errors that occurred during transmission. All vectors in the j^{th} coset have the same error pattern $b_n(j) \in \mathbb{F}_2^n$, $j \in \{1, ..., 2^{n-1}\}$, that is added to each codeword. For bounded minimum distance decoding [Bos99], the decoder can correct errors by finding j if the Hamming weight of $b_n(j)$ is bounded by

$$wt(b_n(j)) \leq \lfloor \frac{d-1}{2} \rfloor \tag{5.10}$$

In contrast, $b_{n-k}(i)$, $i \in \{1, ..., 2^{n-k} - 1\}$, modifies the last $n - k$ bits of codewords to generate the i^{th} coset, or subcode in this context. Then, $b_{n-k}(i)$ is transmitted as side information to the decoder.

Y_t^n can be represented as sum of the initial PUF response X^n and an n-bit error pattern $b_n(j)$.

$$Y_t^n = X^n \oplus b_n(j) \tag{5.11}$$

Using $X^n = \varphi(X^k) \oplus [0^k \; b_{n-k}(i)]$ gives

$$Y_t^n = \varphi(X^k) \oplus [0^k \; b_{n-k}(i)] \oplus b_n(j) \tag{5.12}$$

Since $b_{n-k}(i)$ is known through the helper data, the decoder can correct any error as long as Eqn. 5.10 holds. The systematic encoding enables to generate the subcode without changing the first k bits. Therefore, the corrected key $\hat{Y}_t^k$ is obtained by

$$\hat{Y}_t^k = \gamma\left(\varphi(X^k) \oplus b_n(j)\right) \tag{5.13}$$

Note that Eqn. 5.9 is equivalent to Eqn. 5.13. Eqn. 5.13 leads back to the default decoding problem in the standard communications case where the decoder γ can be used.

Example In this toy example, the new scheme SLLC is used together with an ($n = 7, k = 4, d = 3$) Hamming code with systematic encoding to demonstrate underlying mechanism. 3 bits of helper data are stored and the code has the following generator matrix $\mathbf{G}$

$$\mathbf{G} = \left(\begin{array}{cccc||ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right) , \tag{5.14}$$

and parity check matrix $\mathbf{H}$ [Bos99]

$$\mathbf{H} = \left(\begin{array}{cccc||ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right) \tag{5.15}$$

Let the PUF return random sequence $x_1^7 = 1001010$. The encoder of the Hamming code encodes x_1^4 to codeword c_1^7

$$c_1^7 = x_1^4 \cdot \mathbf{G} = 1001100 \tag{5.16}$$

Storing the 3 least significant bits directly would be equivalent to the Parity Construction discussed in Chapter 3 and leak information about the key and thus violate Condition 4.3 in Section 4.2. In SLLC, fresh PUF bits mask the redundancy part to bring the leakage close to zero or eliminate it completely. The XOR between PUF response and codeword gives the code index as follows (cf. Eqn. 5.5)

$$\begin{aligned} w_1^3 &= b_3(i) && (5.17)\\ &= x_5^7 \oplus c_5^7 && (5.18)\\ &= 010 \oplus 100 && (5.19)\\ &= 110 && (5.20) \end{aligned}$$

Therefore, index 6 is stored as helper data value. The error in the received word will be labeled in bold notation in the following. Assuming $y_1^7 = 10\mathbf{1}1010$ as PUF response in the field, the syndrome decoder reconstructs

$$\begin{aligned} \tilde{c}_1^7 &= y_1^4 || (y_5^7 \oplus w_1^3) && (5.21)\\ &= 10\mathbf{1}1 || (010 \oplus 110) && (5.22)\\ &= 10\mathbf{1}1100 && (5.23) \end{aligned}$$

The Hamming decoder corrects $\tilde{c}_1^7$ to $\hat{c}_1^7 = 10\mathbf{0}1100$ which gives us $\hat{y}_1^7 = 10\mathbf{0}1010$. This example shows how to combine SLLC and an error-correcting code to correct errors such that

$$\hat{y}_1^4 = x_1^4, \text{ or in general } k = l \tag{5.24}$$

Although it is only a toy example, it was shown that SLLC permits error-tolerant secure key generation by using error-correcting codes with systematic encoding.

5.2. Evaluation

This section addresses the theoretical properties of SLLC, first to provide its Algebraic Core and demonstrate that it enables zero leakage. In addition, it has optimal asymptotic behavior for large block sizes such that the capacities can be achieved.

SLLC is currently the only deterministic scheme that achieves the secret key and the helper data capacity, and also inherently ensures information theoretic security.

In general, good PUFs have a sufficiently high entropy but do not necessarily show perfectly i.i.d. behavior. For the security proof, I therefore loosen the i.i.d. assumption to a wider assumption $H(X^n) = n - \epsilon_A$ and $H(X^k) = k - \epsilon_B$ that can also represent correlations. Further, let $H(W) = n - k - \epsilon_W$ with $\epsilon_A > \epsilon_B + \epsilon_W$.

Algebraic Core of SLLC The Algebraic Core is

$$\begin{bmatrix} S^k & W^{n-k} \end{bmatrix} = \begin{bmatrix} X_1^k & X_{k+1}^n \end{bmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{P} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{5.25}$$

Similar to the Fuzzy Commitment, the Algebraic Core $\mathbf{A}$ is an upper triangular matrix with full rank. As a result, the mutual information $I(S;W) = \epsilon_0$, i.e., no information leaks due to the structure of the algebraic core.

Achievable Rate of SLLC

Lemma 1 Rate k/n is an achievable key rate for SLLC and an (n, k, ϵ_1) code with systematic encoding where Conditions 4.2 to 4.5 are bounded by a finite ϵ.

Proof This section proves that Conditions 4.2 to 4.5 in Section 4.2 are fulfilled such that rate k/n is an achievable key rate for a compound source with channel $\mathfrak{T}_t$ using SLLC and an (n, k, ϵ_1) code $\mathcal{C}$. An ϵ_i bounds each condition and then maximizing over all four ϵ_i results in one ϵ.

Condition 4.2 Condition 4.2 is fulfilled by definition since an (n, k, ϵ_1) code is assumed as given for channel $\mathfrak{T}_t$, so

$$\Pr[K \neq L] < \epsilon_1 \tag{5.26}$$

In practice, the block error probability can be computed by bounding techniques [Bos99] or Monte Carlo simulation.

Condition 4.3 Security is addressed by Condition 4.3. It states that $I(K;W)$ has to be upper bounded by and ϵ_2.

$$\begin{aligned} I(K;W) &= H(W) + H(K) - H([W\ K]) && (5.27)\\ &= H(\varphi(X_1^k)_{k+1}^n \oplus X_{k+1}^n) + H(X_1^k) - H(X_1^n) && (5.28)\\ & && (5.29) \end{aligned}$$

According to the chain rule of entropy [CT06],

$$H(X_1^n) = H(X_1^k X_{k+1}^n) = H(X_1^k) + H(X_{k+1}^n | X_1^k) \tag{5.30}$$

This gives

$$I(K;W) = H(\varphi(X_1^k)_{k+1}^n \oplus X_{k+1}^n) - H(X_{k+1}^n | X_1^k) \tag{5.31}$$

X_{k+1}^n masks the redundancy $\varphi(X_1^k)_{k+1}^n$ such that

$$\frac{1}{n}\left(H(\varphi(X_1^k)_{k+1}^n \oplus X_{k+1}^n) - H(X_{k+1}^n | X_1^k)\right) < \epsilon_2 \tag{5.32}$$

Therefore,

$$\frac{1}{n} I(K;W) < \epsilon_2 \tag{5.33}$$

with $\epsilon_2 > \frac{1}{n}(\epsilon_A - \epsilon_B - \epsilon_W)$.

Condition 4.4 Condition 4.4 ensures that the key rate is close to the entropy of the derived key.

$$\frac{1}{n} H(K) > R_{key} - \epsilon_3 \tag{5.34}$$

with Rate $R_{key} = k/n$

$$\frac{1}{n} H(X_1^k) > \frac{k}{n} - \epsilon_3 \tag{5.35}$$

$$\frac{1}{n}(k - H(X_1^k)) < \epsilon_3 \tag{5.36}$$

$$\frac{1}{n}\epsilon_B < \epsilon_3 \tag{5.37}$$

Condition 4.5 Finally, Condition 4.5 checks that the entire key space is used by the derived key.

$$\frac{1}{n}\log_2|\mathcal{K}| < \frac{1}{n} H(K) + \epsilon_4 \tag{5.38}$$

$$\frac{k}{n} < \frac{1}{n} H(X_1^k) + \epsilon_4 \tag{5.39}$$

$$\frac{1}{n}(k - H(X_1^k)) < \epsilon_4 \tag{5.40}$$

Inserting $H(X_1^k) = k - \epsilon_B$ gives

$$\frac{1}{n}\epsilon_B < \epsilon_4 \tag{5.41}$$

Therefore, $R_{key} = k/n$ is an achievable key rate for block size n and

$$\epsilon = \max(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4) \tag{5.42}$$

■

Corollary For an ideal PUF with $\Pr[x = 0] = \Pr[x = 1] = 0.5$ and i.i.d outputs, Rate R_{key} is achievable with $\epsilon = \epsilon_1$.

Proof Due to i.i.d. PUF outputs, $H(X_{k+1}^n) = n - k$. Further, $H(X_1^k) = k$. Therefore, $\epsilon_2 \to 0, \epsilon_3 \to 0, \epsilon_4 \to 0$ which gives $\epsilon = \epsilon_1$. So, secret key rate k/n is achievable with an (n, k, ϵ_1) code.

■

Lemma 2 The secret key rate R_{key} of SLLC achieves the capacities of the source.

SLLC can achieve a rate $R_{key} = k/n$. Here, the code size is denoted with k_n to highlight that k depends on n for a given ϵ. Recall, according to Section 1.7, a capacity achieving (n, k_n, ϵ) code $\mathcal{C}$ for a channel $\mathfrak{T}_t$ has rate k_n/n such that

$$\lim_{n\to\infty} R_{key} = \lim_{n\to\infty} \frac{k_n}{n} = C_{\mathfrak{T}} = C_{key} \tag{5.43}$$

■

Lemma 3 For an ideal PUF with $\Pr[x = 0] = \Pr[x = 1] \approx 0.5$ and i.i.d outputs the helper data rate R_{hd} of SLLC, defined in in Eqn. 4.7, achieves the capacity of the source.

Proof By definition, the capacity achieving code $\mathcal{C}$ with systematic encoding has redundancy $n - k_n$.

Let $\mathcal{T}_\epsilon^n(P_X)$ be the set of all ϵ-letter typical sequences of the source. See Section 6.1.1 or [Kra07] for the definition.. For an ideal PUF,

$$\lim_{n\to\infty} \log_2 \left(|\mathcal{T}_\epsilon^n(P_X)|\right) = n \tag{5.44}$$

This is important because only is this case the union of SLLC codebooks is equal to the typical set. If Eqn. 5.44 does not hold, the codebooks contain unused values and capacity is not achieved anymore.

Therefore, for the ideal PUF,

$$\lim_{n\to\infty} R_{hd} = \lim_{n\to\infty} \frac{n - k_n}{n} \tag{5.45}$$

According to Section 1.7, for a capacity achieving code $k_n = n(I(X;Y) - \epsilon_n)$ holds for a small $\epsilon_n > 0$, so

$$\lim_{n\to\infty} R_{hd} = \lim_{n\to\infty} 1 - I(X;Y) + \epsilon_n \tag{5.46}$$

For the assumed ideal PUF with $H(X) = 1 - \epsilon_X$, $\epsilon_X > 0$ and using the identity $H(X) - I(X;Y) = H(X|Y)$

$$\lim_{n\to\infty} R_{hd} = \lim_{n\to\infty} H(X|Y) + \epsilon_X + \epsilon_n \tag{5.47}$$

Applying the limit, the code dependent part ϵ_n goes to zero resulting in

$$\lim_{n\to\infty} R_{hd} = H(X|Y) + \epsilon_X \tag{5.48}$$

$$= C_{hd} + \epsilon_X \tag{5.49}$$

Equation 5.49 shows that for capacity achieving codes and ideal PUFs with $\epsilon_X = 0$, SLLC achieves helper data capacity.

■

It can be seen that SLLC fulfills typical information theoretic requirements because the error probability and the security parameters are bounded by ϵ. In addition, the scheme is optimal in a sense that it is capacity achieving such that a maximum key size can be extracted and the required helper data size is brought down to the theoretical limit in an asymptotic setting.

5.3. Implementation

With the trend towards more reliable and secure PUFs, e.g. [HB10], PUFs with bit error probabilities of 10^{-5} and lower can be manufactured. Therefore, less powerful error correction is necessary to generate reliable keys. The PUF has only negligible bias and correlation so that it provides close to i.i.d. properties such that the last column in Table 4.2 at the end of the previous chapter holds in a good approximation.

Calculating the bit error probability with the Union Bound, e.g. [Bos99], shows in this case that a compact high-rate $(55, 43, 5)$ BCH code already leads to a key regeneration failure rate of $7.87 \cdot 10^{-11}$. This is below the Failure in Time (FIT) specification of most, if not all, popular silicon processes (typical FIT failure rate ranges from $5 \cdot 10^{-9}$ to $2 \cdot 10^{-8}$ [Xil15]).

The example uses a BCH $(55, 43, 5)$ code with systematic encoding, which is a shortened version of a BCH $(63, 51, 5)$ code [Bos99]. Running this three times results in $55 \cdot 3 = 165$

PUF bits consumed to derive $43 \cdot 3 = 129$ data bits for the key, and using $(55-43) \cdot 3 = 36$ helper data bits.

The proof-of-concept BCH decoding core is a modified version of the one used in [YSS+12] and requires 4,441 NAND2 Gate Equivalents (GE) in an ASIC implementation. The design was synthesized using Synopsys Design Compiler, comprising of 194 flip-flops and the rest conventional standard-cell combinatorial logic. It uses a serialized input and output interface. The latency is 372 clock cycles per block, and three blocks (1,116 cycles) are required to generate a 128-bit key. The decoder operates in $GF(2^6)$ with field elements constructed using the primitive polynomial $p(x) = 1 + x + x^6$. The generator polynomial used to generate the codewords for the (55,43,5) BCH code is $g(x) = ((1 + x + x^6) \cdot (1 + x + x^2 + x^4 + x^6))$. Since the code is shortened, the first 8 information bits of each 63 bit block are regarded as fixed to 0.

Note that the SLLC syndrome decoder only has a negligible impact on the overall implementation size. It can be implemented as a 6 bit counter and a comparator that decides if an input bit is within the information part of the codeword and fed directly into the BCH decoder, or if it is XORed with a helper data bit.

In addition to the BCH decoder, popular state-of-the-art methods such as the Code-Offset and Syndrome methods require an additional hash function. Compact implementations of popular lightweight hash functions like SPONGENT (256/256/16) [BKL+13] or PHOTON (256/32/32) [GPP11] require around 2,300 GE and 2,150 GE, respectively.

The rest of the work is focusing more on FPGAs than ASICs. To also provide consistent results with the remainder of this work, Julian Leyh developed and analyzed different BCH decoder implementations for FPGAs under my supervision in [Ley15]. An implementation similar to the presented ASIC design requires 232 LUTs, 155 registers, 72 slices and 692 clock cycles on a Xilinx Spartan 6 FPGA. Going to an optimized processor core, similar to the one used in previously PUFKY [MVHV12], prioritizes area strongly over time in the area time trade-off. The implementation requires 162 LUTs, 21 registers, 43 slices and 4,597 clock cycles.

Key generation from internal PUF response	Helper data size	Leakage	Hash function required	Estimated area Dec + Hash (if needed)
SLLC	36 bit	0 bit	no	$\approx 4,500$ GE
Code-Offset [DRS04]	165 bit	36 bit	yes	$\approx 6,600$ GE
Syndrome Construction [DRS04]	36 bit	36 bit	yes	$\approx 6,600$ GE

Table 5.1.: Practical comparison to related work for non-optimized implementations

Table 5.1 compares the helper data size, the secret key leakage and an estimated ASIC gate count of SLLC with the Code-Offset and the Syndrome Construction as most popular previous work. It can be seen that SLLC is the only approach that combines minimal helper data size and zero leakage through the helper data. For both other approaches $n - k = 36$ bit of the PUF response are revealed.

With SLLC 128 of the 129 data bits can be used directly as 128-bit key, without processing them through a hash function. The overhead for helper data is only 36 bits.

The areas of the Code-Offset and Syndrome Construction are estimated by adding the size of the BCH decoder and the hash function. In a modular implementation where the BCH module and the hash module are distinct, SLLC requires only the BCH module of an estimated 4500 GE. With the Code-Offset or Syndrome Construction, additional estimated 2150 GE are required for the hash, leading to a total of an estimated 6600 GE. This is an extra 47% overhead that is avoided by SLLC.

With cost of less than $5,000$ gates, 36 extra helper data bits, and the use of SLLC which eliminates the requirement for a hash function, there is an overall area reduction of 30% compared to the Code-Offset and Syndrome methods. In addition, the required helper data is cut to 24% of the Code-Offset method. Information theoretic security is achieved without having to make additional assumptions on the security of the hash function. From a practical standpoint, there is one module less to secure, e.g. against physical attacks such as the side-channel attack presented in [MSSS11].

5.4. Conclusions

This section has introduced Systematic Low Leakage Coding to demonstrate that minimal helper data size and low secrecy leakage are achievable at the same time in practice. A deterministic scheme was derived from the random coding scheme by Ahlswede and Csiszar [AC93] and it was demonstrated that the theoretical properties still hold. In addition, the parameters of a possible ASIC design using a BCH code were discussed and compared to the state of the art.

Chapter 6.

Differential Sequence Coding

After studying a generic scenario of key generation for PUFs in the last chapters, this chapter focuses on PUFs where PUF-bit specific reliability information is available, for example through multiple read-outs or multi-lettered output alphabets. Figure 6.1, which was already discussed in Chapter 3, shows that the syndrome coding and error correction block form the minimal number of data processing blocks in practical use cases. I will show with the information theoretic concept of typicality that long sequences are necessary to make precise a-priori estimates on the number of reliable PUF response bits inside of a processing block in Figure 6.1. The new syndrome coding scheme called *Differential Sequence Coding* (DSC) searches and indexes reliable PUF response bits. In contrast to previous index-based work such as IBS [YD10b] and C-IBS [HMSS12], DSC overcomes the fixed small block sizes and uses only one large block with relative pointers. Preventing helper data manipulation attacks requires adding a hash function to Figure 6.1.

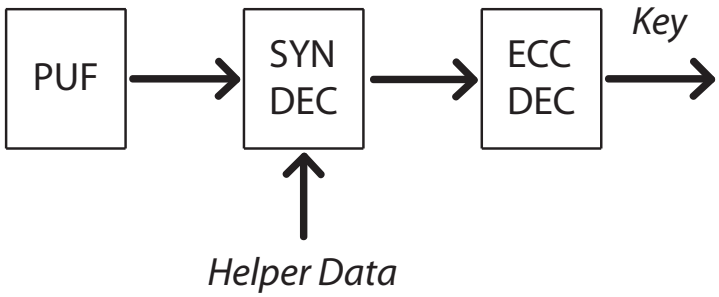

Figure 6.1.: Generic reproduction procedure

In Section 6.1, I will analyze the impact of an increasing block size with the information theoretical concept of typicality [CT06, Kra07]. DSC encoding is introduced in Section 6.2 and fundamental properties, including helper data compression, are discussed in Section 6.3. Section 6.4 covers theoretical and practical security aspects of DSC. In Section 6.5, I am the first to discuss convolutional codes in the PUF context. Then, the performance of the overall system is compared to state-of-the art approaches in Section 6.6. I will provide details on the hardware implementation in Section 6.7. Then, possible improvements of the basic DSC approach are briefly addressed in Section 6.8.

This chapter is based on the following publications: The theoretical section on typicality is based on [HYS16]. DSC was introduced first in [HWRL+13, HWS15]. The first

improvement was published by using helper data compression [HS14]. The improved Viterbi decoder was developed by Leandro Rodrigues Lima under my supervision and published in [HRLS14]. The improved SPONGENT is based the work of Maximilian Birkner [Bir13] and Leandro Rodrigues Lima and was compared to the state of the art in [JRLH14]. Putting all parts together, the final system is discussed in [HYS16]. The improvements discussed in 6.8 were analyed by Aysun Gurur Önalan and Benjamin Nolet [Nol15].

6.1. Relation between Block Size and Reliability

This section applies the information theoretical concept of typicality to demonstrate how the efficiency of syndrome coding schemes improves with an increasing block size. Averaging effects become stronger and the impact of statistical outliers is reduced. Therefore, the longer the block, the more precise predictions can be made on the distribution of symbols in a sample.

Assuming sequences of length 100, drawn from a binary source with uniform distribution, it is intuitive that most sequences have a roughly balanced number of zeros and ones. To be more precise, for example a sequence with less than 25 or more than 75 ones is only drawn with a probability around 10^{-7}. So, it is possible predictions about how many zeros and ones are in a sequence without knowing their actual positions, and the longer the sequence is, the more precise the prediction. This common case, or more precisely the set of all common sequences was defined as *typical set* by Shannon in his 1948 paper [Sha48].

For syndrome coding, the goal is to compress sequences that are drawn from a known distribution efficiently and it is most important to handle the common case efficiently. In the previous example, one could design a system that is optimized for 25 to 75 ones per sequence and only spends less effort on the other sequences with more or less ones in the sequence. Lossy compression goes even one step further and is not able to correctly process uncommon sequences at all in the ultimate case.

6.1.1. Typical Sequences in Syndrome Coding

In the following, the concept of typicality enables to analyze the effect of the block size on the distribution of reliable inputs for syndrome coding. Let X^n be a part of the overall response sequence drawn from the PUF.

Let set $\mathcal{P}$ contain the probability distributions P_{X_i} over all PUF bits X_i, $i = 1, ..., n$ with $X_i \in \{0, 1\}$ and Bernoulli probability distribution P_{X_i}. The occurrence of the different distributions P_{X_i} in $\mathcal{P}$ defines the reliability of the PUF. Examples for practical distributions can be found e.g. in [MTV09b, Mae13] and [HSP13].

For a PUF response bit X_i with expectation $\mu(X_i) \geq 0.5$, an error occurs from drawing a 0 which occurs with $\Pr[X_i = 0] = 1 - \mu(X_i)$, and for $\mu(X_i) < 0.5$ for drawing a 1 analogously. The first question is, what is the probability p that a given PUF output is reliable.

A fixed threshold $0 \leq p_{max} \leq 0.5$ defines the maximum tolerable error probability of a PUF response bit to be considered reliable. This gives

$$p = \Pr[\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}] \tag{6.1}$$

Therefore, each of the n PUF response bits X_i in X^n behaves according to a Bernoulli distributed reliability indicator Λ with parameter p. It indicates if Eqn. 6.1 holds for a specific PUF bit X_i or not. The sequence λ^n for a specific X^n indicates for each PUF bit X_i whether it is reliable ($\lambda_i = 1$) or not ($\lambda_i = 0$), $i = 1, ..., n$.

For the quantitative evaluation of typicality, the following notation is used:

- The letters $a \in \mathcal{P}_\lambda = \{0, 1\}$ indicate whether a PUF response bit is reliable or not.
- $P_\Lambda(\cdot)$ gives the precise theoretical distribution of letters occurring in sequences Λ^n.
- $N(a|\lambda^n)$ is the number of occurrences of letter a in sequence λ^n, and quantifies the empirical distribution, i.e. the number of reliable and unreliable PUF bits in X^n. Note that the distribution of $N(a|\lambda^n)$ is given by a binomial distribution with parameters $P_\Lambda(a)$ and n.
- The parameter $\epsilon > 0$ quantifies the maximum allowed deviation of the number of reliable bits $N(1|\lambda^n)$ in a given sequence from the mean number of reliable bits per sequence $n \cdot P_\Lambda(1)$ to still be part of the typical set.

According to [Kra07], a reliability indicator sequence λ^n is an *ϵ-letter typical sequence* if

$$\left| \frac{1}{n} N(a|\lambda^n) - P_\Lambda(a) \right| \leq \epsilon \cdot P_\Lambda(a) \text{ for all } a \in \mathcal{P}_\lambda \tag{6.2}$$

and the *letter typical set* $\mathcal{T}_\epsilon^n(P_\Lambda)$ is defined as set containing all sequences in $\mathcal{P}_\lambda^n$ that fulfill Eqn. 6.2.

Without loss off generality, let $P_\Lambda(0) < P_\Lambda(1)$. Otherwise $P_\Lambda(0)$s have to be replaced by $P_\Lambda(1)$s in Eqn. 6.3. The probability of drawing an ϵ-letter typical sequence is given by

$$\Pr[\Lambda^n \in \mathcal{T}_\epsilon^n(P_\Lambda)] = \sum_{i=\lceil (1-\epsilon) \cdot P_\Lambda(0) \cdot n \rceil}^{\lfloor (1+\epsilon) \cdot P_\Lambda(0) \cdot n \rfloor} \binom{n}{i} P_\Lambda(0)^i \cdot (1 - P_\Lambda(0))^{n-i} \tag{6.3}$$

Applying Hoeffding's inequality in [Kra07], Eqn. 6.3 is lower bounded by

$$\Pr[\Lambda^n \in \mathcal{T}_\epsilon^n(P_\Lambda)] \geq 1 - 4 \cdot e^{-n\epsilon^2 \min[p,1-p]} \tag{6.4}$$

and the complementary event is

$$\Pr[\Lambda^n \notin \mathcal{T}_\epsilon^n(P_\Lambda)] < 4 \cdot e^{-n\epsilon^2 \min[p,1-p]} \tag{6.5}$$

Note that the bound is relatively tight where the probability of a non-typical sequence decreases exponentially with n. In contrast, the more widely used concept of *entropy typical sequences* [Kra07] only gives a linear decrease in Eqn. 6.5 over n.

The concept of typicality allows to estimate the number of reliable bits in a PUF response sequence. The next section will take this number into account to design the error correction accordingly.

6.1.2. Analysis

An efficient error correction is designed to correct the errors that occur in typical sequences which have a controlled number of unreliable bits. One cannot make precise statements on the other sequences so that errors can occur more likely if a sequence which is not element of the typical set is drawn. Therefore, reducing the probability of drawing a non-typical sequence is a first step to reduce the overall error probability.

Figure 6.2 plots the probability of drawing a non letter-typical sequence over the block size n for $p = 0.326$ and different parameters ϵ. This p value will be used later in the implementation to reduce the average bit error probability of the reference SRAM PUF [GKST07, MTV09b] from 15% to 2.7%. An epsilon value of 0.4 corresponds to a ratio of at least $(1 - \epsilon) \cdot p = 19.5\%$ reliable PUF bits in a typical sequence, whereas for $\epsilon = 0.2$ already 26% reliable bits are guaranteed.

The solid lines show the precise computed values derived from Eqn. 6.3 while dotted lines give bounded values according to Eqn. 6.5. Note that the straight lines on the logarithmic scale correspond to an exponential behavior in n.

Figure 6.2 shows that the block size has a large impact on the probability of drawing non-typical sequences. Smaller blocks will lead to an increased key error probability because non-typical error patterns are more likely to occur. As a consequence, increasing the block size is a first requirement for an efficient usage of the reliable PUF bits. In addition, it is important to find a good trade-off between the ϵ parameter and the probability of a non-typical sequence. If a too small ϵ parameter is selected, very specific predictions on the PUF sequence can be made but there the probability of drawing such a sequence is relatively low. Otherwise, a too large ϵ results in a high probability of drawing a typical sequence but one only has less precise information about the number of reliable bits in

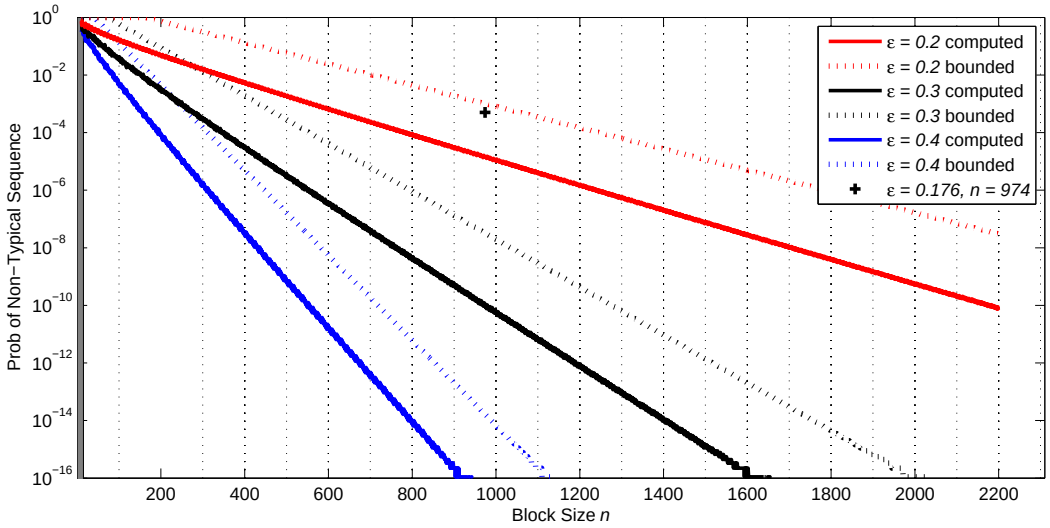

Figure 6.2.: Probability of drawing non-ϵ-letter-typical sequences, computed and upper bounded values for $p = 0.326$ and different ϵ parameters. The parameters $\epsilon = 0.176$ and $n = 974$ are used later in the implementation.

the sequence. In Figure 6.2, $p = 0.326$ such that in average roughly one third of the n PUF response bits in each sequence is reliable. Table 6.1 shows the minimum ratio of reliable PUF response bits of sequences within the ϵ-letter-typical set.

ϵ	$\frac{1}{n}N(1\|\lambda^n)$
0.2	0.26
0.3	0.23
0.4	0.20

Table 6.1.: Lowest ratio of reliable bits in an ϵ-letter-typical sequence

The reference implementations for the target scenario use block sizes between 3 and 11. This region is highlighted in gray in the far left of the plot. Several design space explorations have shown that this is the most favorable region for the conventional approaches [Bös08, MTV09b, Hil11]. However, the curves show that the probability of drawing non-typical sequences in this area is $> 10^{-1}$ so that errors through non-typical sequences have to be corrected on a regular basis. The point that will be used later in the DSC implementation is marked with the black cross. This work will operate on a single block of size 974 and probability of a non-typical sequence of $5 \cdot 10^{-4}$. ECCs are designed to correct a specific number of errors and if the errors in all typical sequences can be corrected with a high probability, successful error correction is ensured with

a high probability. By using one maximum-sized block, the probability of drawing a non-typical sequence decreases by a factor of 200.

So far, the largest block sizes of up to 256 can be found in the work of Yu *et al.* [YHD15], which is designed for a different scenario with higher PUF noise, resulting in significantly higher PUF bit / key bit ratios.

6.2. DSC Encoding

The previous section has shown that controlling the number of reliable PUF bits within each block is a prerequisite for efficient key generation with PUFs. Larger block sizes are favorable to control the number of unreliable bits per block. However, ECCs with large block sizes typically create a heavy resource overhead. The DSC approach operates on one maximally reliable block with low overhead. In particular, it is ensured beforehand that a PUF response sequence with low bit error probabilities is fed into the ECC decoder. To minimize the decoder complexity to ensure a low hardware overhead, only the reliable PUF bits from the maximally sized single block are processed while the rest is discarded.

During the generation step, the PUF provides a sequence of PUF bits X together with a reliability indicator $\mu(X)$ for each PUF response bit. Note that the reliability indicators $\mu(X)$ are unique for each chip so that they have to be obtained for each device separately. DSC reads the entire PUF response sequence X^n and marks the PUF bits that have a reliability above a predefined threshold. They point to a secret sequence C^k within the PUF response sequence. The notation is adapted from block codes where n describes the block size and k the number of embedded bits.

The PUF sequence X^n is scanned sequentially for PUF bits that are more reliable than a given error probability threshold p_{max}. When such a PUF bit is found, the distance to the last reliable PUF bit is stored as differential distance pointer U. If the expected value $\mu(X)$ of the PUF bit is closer to the corresponding code sequence bit C, the inversion bit V is set to zero. Otherwise, it is set to one. Adding the inversion bit ensures that all reliable PUF bits are used. A version without inversion bits is also possible where only PUF bits that are close to a code sequence bits C_i are indexed. However, in average only half of the reliable PUF bits are indexed such that the PUF size n has to be doubled if no inversion bits are used.

Figure 6.3 shows an example for DSC encoding. Code sequence c^4 is provided by an ECC and DSC stores one pointer for each code sequence bit. In the example, zeros are represented by white boxes and ones by black boxes. For the PUF response X^{16} and a given maximum error probability p_{max}, a white box denotes $\mu(X) \leq p_{max}$. A black box stands for $\mu(X) \geq 1 - p_{max}$ and gray boxes show the unreliable PUF response bits with $p_{max} \leq \mu(X) \leq 1 - p_{max}$.

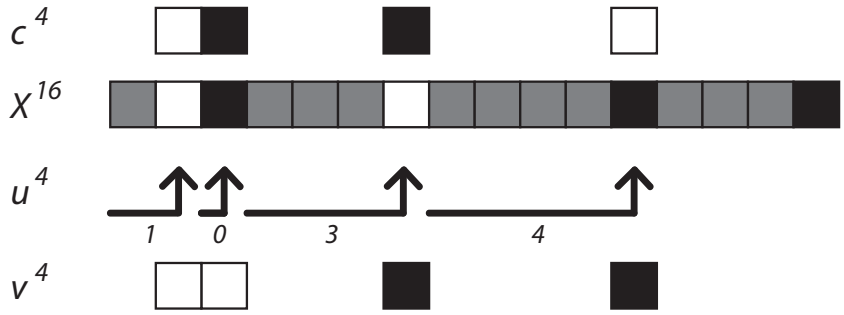

Figure 6.3.: Example for DSC encoding

The code sequence $c^4 = (0, 1, 1, 0)$ and the PUF response X^{16} are encoded to the helper data tuple $w^4 = (u, v)^4$. X_2 is the first reliable PUF response bit. The distance pointer keeps track of the unreliable PUF response bits between two reliable ones, so $u_1 = 1$. For the first inversion bit, $v_1 = 0$ since both boxes have the same color. X_3 is the next reliable PUF bit, so $u_2 = 0$ and again $v_2 = 0$. After skipping three unreliable PUF bits, X_7 is indexed by $u_3 = 3$. Since a white box is indexed for a black code bit, $v_3 = 1$. u_4 and v_4 are computed accordingly, such that $u^4 = (1, 0, 3, 4)$ and $v^4 = (0, 0, 1, 1)$.

The algorithmic description of DSC encoding is provided in Algorithm 1 with $X_i, c_i, v_i \in \mathbb{F}_2$ and $u_i \in \mathbb{F}_2^l$.

If the length of the code sequence exceeds the number of reliable PUF bits ($error\_1$), or if helper data cannot be stored ($error\_2$), an error is thrown and the algorithm fails to generate a valid set of DSC helper data.

6.3. Properties

This section addresses the probabilities of $error\_1$ and $error\_2$, leading to the yield. Further, the DSC helper data is not uniformly distributed such that it can be compressed. In the final part, this section addresses the bit error probability of DSC.

6.3.1. Yield

Previous work such as IBS and C-IBS, or the Code-Offset approach take a fixed number of PUF bits to encode a fixed number of secret bits for each block, so the average bit error probabilities can be determined and are well-controlled. However, it is not possible to determine the reliability of a specific device. As a consequence, one cannot guarantee that all devices of a batch fulfill a given minimum reliability.

In contrast, DSC already detects unreliable devices when no valid set of helper data is generated. This occurs when not enough stable PUF bits are found during manufacturing. This additional measure gives a-priori reliability information about individual

Algorithm 1: DSC Encoding

Input: X^n, c^k
Output: u^k, v^k
$o := 0$ *(The offset counter o tracks absolute the position within Λ^n)*
for $i := 1 \to k$ **do**
Search for one reliable PUF response bit for each code sequence bit.
for $j := 1 \to 2^l$ **do**
if $o + j > n$ **then**
Return *error_1 (Not enough PUF output bits within the specification)*
else if $\Pr[X_{j+o} = 0] \geq 1 - p_{max} \vee \Pr[X_{j+o} = 1] \geq 1 - p_{max}$ **then**
$u_i := j - 1$
if $\Pr[X_{j+o} = c_i] \geq \Pr[X_{j+o} = c_i \oplus 1]$ **then**
$v_i := 0$ *(No inversion)*
else
$v_i := 1$ *(Inversion)*
end if
$o := o + j$
Break
else if $j = 2^l$ **then**
Return *error_2 (Counter overflow)*
end if
end for
end for
Return u^k, v^k

devices. Therefore, we can assess the average error probability in the field over all devices and also provide a bound that the reliability of no individual device will exceed a given maximum error probability.

If the error occurs during manufacturing, the device can still be used and programmed with a different parameter set, for example, for a lower target reliability or smaller key size. In the worst case, it has to be discarded during the manufacturing process. However, the consequences are well-controlled and do not affect devices during operation out in the field. If an error would occur during operation in the field instead, the device is not able to generate the correct key and subsequent tasks cannot be performed successfully. For DSC, the probability of any individual device failing in the field is bounded. In the conventional approach, there is only a guarantee on the average failure probability.

Recall that p is defined as the probability that the error probability of a PUF response bit X is smaller than p_{max}:

$$\begin{aligned} p &= \Pr[\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}] \\ &= cdf(p_{max}) + (1 - cdf(1 - p_{max})) \end{aligned} \tag{6.6}$$

Note that some of the previous work such as [MTV09b, YD10b],[HMSS12] requires a precise estimation of $\mu(X)$ whereas DSC only operates on a binary decision whether $\mu(X_i) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}$, $i \in \{1, ..., n\}$, holds, or not. The reliability information can be obtained for example by performing multiple measurements [MTV09b] or evaluating analog PUF output values [HSP13].

As mentioned above, there are two events where helper data generation fails:

Error Event 1: Lack of reliable PUF Bits

Successful DSC encoding requires that sequence X^n contains at least k reliable PUF bits with $\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}$. If less than k reliable PUF bits are found, error event $error\_1$ is triggered. Recall the probability of drawing a non-letter-typical sequence in Eqn. 6.3. This requirement is relaxed for DSC because sequences with more than k reliable PUF response bits also pass in helper data generation. So, error $error\_1$ occurs with probability

$$e_1 = \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \tag{6.7}$$

Recall that according to the typicality discussion in Section 6.1, e_1 decreases exponentially in n for a constant ratio k to n.

Error Event 2: Helper Data Overflow

The second error event $error\_2$ occurs if the variably sized helper data does not fit into the allocated space. The distribution of the helper data size of the selected parameter set is obtained through simulation. A practical example is given later in Section 6.6.4

The cost of helper data storage varies greatly, depending on the scenario. Corner cases are on-chip NVM, where each additional byte of data is a burden. On the other side of the scale, the size is not critical if the helper data is stored on an external server that is connected over a fast interface. The FPGA scenario with off-chip helper data storage is in between.

The yield ζ is computed by the probability that neither $error\_1$ nor $error\_2$ occur. $error\_1$ and $error\_2$ are not disjoint, so

$$\zeta > 1 - (e_1 + e_2) \tag{6.8}$$

Note that e_1 depends on the worst case error probability p_{max} and the size of the PUF n, whereas e_2 is only affected by the maximum size of the helper data.

The events $error\_1$ and $error\_2$ define hard break conditions and affect the yield directly. In the following, I aim for a yield $\zeta > 99.9\%$, and thus set $e_1 \leq 5 \cdot 10^{-4}$ and also $e_2 \leq 5 \cdot 10^{-4}$.

6.3.2. Helper Data Compression with Run-Length Encoding

The helper data pointers u^k are not uniformly distributed and thus contain redundancy. This section discusses how they can be compressed to further reduce the helper data size of DSC.

Data compression, or lossless source coding, is a discipline of Information Theory. The goal of data compression is to find a shorter representation for output sequences of a given source.

Let S_U be a discrete memoryless source with output alphabet $\mathcal{U}$, probability distribution P_U and entropy $H(U)$. A source encoder maps an input sequence U^n to an output sequence Q^l with alphabet $\mathcal{Q}$ and $n, l \in \mathbb{N}$. Shannon's source coding theorem [Sha48] shows that any sequence U^n output by source S_U can be represented in average by $H(U)$ bits per symbol for $n \to \infty$. For this application, the helper data of a batch of identically manufactured PUF systems is treated as source with DSC helper data U^k of a single device as output sequence.

There are typically two approaches for lossless source coding: Either an input sequence of fixed length is mapped to a code sequence of variable length, or vice versa. The remainder of this work only considers fixed to variable length encoding. For a known source, Huffman coding [Huf52] can create output sequences with a length arbitrarily close to the entropy of the source. Universal source coding algorithms such as Lempel-Ziv coding [ZL77], are able to even compress sequences of unknown sources.

The straight-forward approach to represent a distance pointer u_i as helper data is to allocate l bits and store the binary representation $b_l(u_i)$. The pointers u are geometrically distributed with parameter p, so the probability distribution P_U is

$$P_U(u) = (1-p)^u \, p \tag{6.9}$$

This basic binary representation contains a significant amount of redundancy. Golomb's *Run-Length Encoding* (RLE) [Gol66] is a source coding algorithm designed for sequences

with geometric probability distribution. An improved version was presented by Gallager and van Voorhis [GVV75] in 1975. The basic version by Golomb [Gol66] is used in the following in this work. Note that [Gol66] treats runs of successful draws ended by an unsuccessful one and in this case, the successful draw ends a run.

RLE encodes any integer number u by a series of ones followed by a zero as delimeter and a small number of a finite alphabet $\mathcal{L}$ with elements $l_j \in \mathcal{L}$, $j = 0, ..., m-1$ and $|\mathcal{L}| = m$. For the run-length part, m determines how many unsuccessful trials are represented by every 1 and l gives the number in the remaining $u \mod m$ trials. Therefore, the compressed version $q(u)$ of u is given by

$$q(u) = \underbrace{1 \ \ldots \ 1}_{\lfloor \frac{u}{m} \rfloor \text{ times}} \ 0 \ l_{(u \bmod m)} \tag{6.10}$$

The algorithm can be interpreted as Euclidean division of u by m with different representations of the quotient and remainder. The quotient is represented in a series of ones, followed by a zero, and the remainder in the finite alphabet $\mathcal{L}$.

Algorithms [Gol66] and [GVV75] differ in the representation of l_j. Golomb [Gol66] uses an uncoded binary representation whereas Gallager and van Voorhis [GVV75] encode the length fixed part with a Huffman code [Huf52]. According to [GVV75], optimal codes can be constructed for an integer m chosen in dependency of $p \in [0, 1]$ such that

$$(1-p)^m + (1-p)^{m+1} \leq 1 < (1-p)^{m-1} + (1-p)^m \tag{6.11}$$

u	$b_4(u)$	$q(u)$, $m = 1$	$q(u)$, $m = 2$	$q(u)$, $m = 4$
0	0000	0	0 0	0 00
1	0001	1 0	0 1	0 01
2	0010	11 0	1 0 0	0 10
3	0011	111 0	1 0 1	0 11
4	0100	1111 0	11 0 0	1 0 00
5	0101	11111 0	11 0 1	1 0 01
6	0110	111111 0	111 0 0	1 0 10
7	0111	1111111 0	111 0 1	1 0 11
8	1000	11111111 0	1111 0 0	11 0 00
9	1001	111111111 0	1111 0 1	11 0 01
10	1010	1111111111 0	11111 0 0	11 0 10

Table 6.2.: Run-length encoding with $m = 1$, $m = 2$ and $m = 4$ according to [Gol66]

As an example, Table 6.2 shows RLE representations of small integers u for $m = 1$, $m = 2$ and $m = 4$. For different parameters m, the size of the fixed length part and the overall length l show large variations. Therefore, the selection of a good m value for a practical scenario will be addressed later in Section 6.6.3.

The length $l(u)$, i.e. the number of bits to represent u is given by the individual length of the run-length part and the fixed length of the finite alphabet.

$$l(u) = \left\lfloor \frac{u}{m} \right\rfloor + 1 + \log_2 m \tag{6.12}$$

Therefore, the expectation $E(l)$ is minimized, where $l(u)$ is distributed with the geometric probability distribution $P_U(u)$ given in Eqn. 6.9.

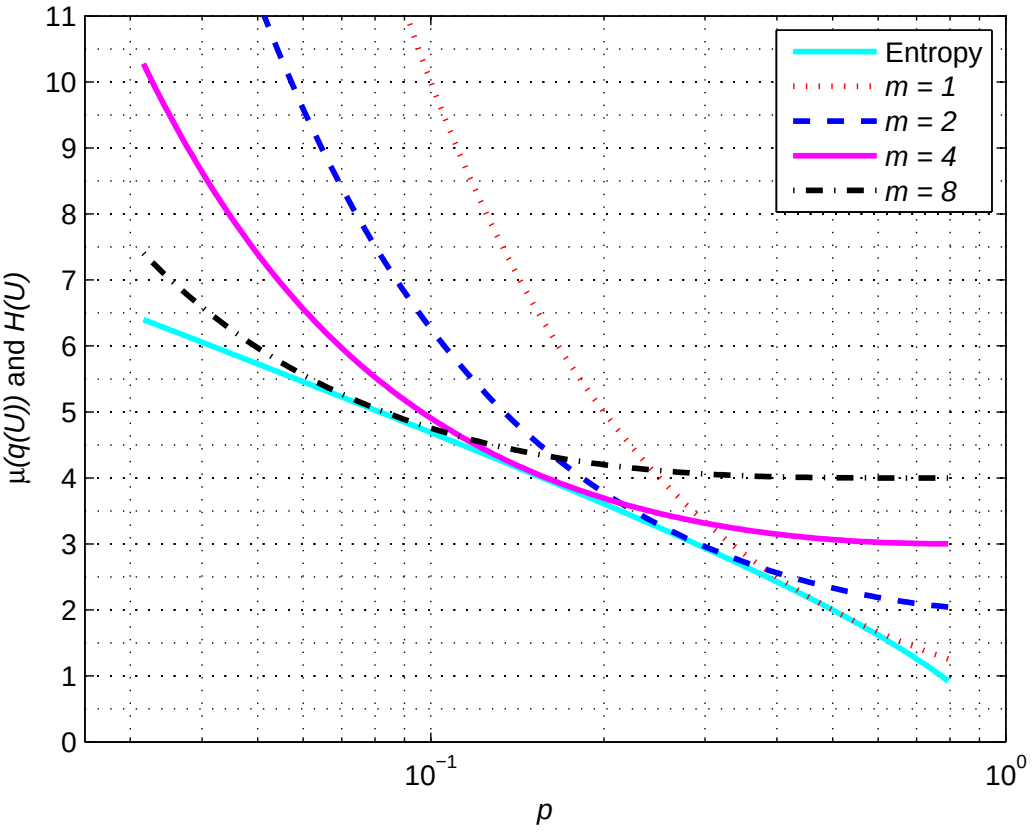

Figure 6.4.: Average RLE encoded pointer sizes $\mu(q(U))$ and entropy $H(U)$ for geometrically distributed random variables U with parameter p

The input distribution $P_U(u)$ and parameter m define the distribution and value of $q(u)$. In Figure 6.4, p is plotted on a logarithmic x-axis and the average length $\mu(q(u))$ is shown for different m on the linear y-axis. Note that the entropy (solid cyan line) is the lower bound for every lossless compression. Figure 6.4 shows that a good selection

of m gives very low overheads for different values of p such that RLE can enable optimal compression for DSC.

6.3.3. DSC Bit Error Probability

For a given PUF response distribution P_X, the output bit error probability of DSC p_{syn} can be computed analytically for a given maximum output error probability p_{max} through the integral over the error probabilities of all PUF response bits that are within the specification, normalized by their overall probability of occurrence p given by Eqn. 6.6, so

$$p_{syn} = \frac{1}{p}\left(\int_{0}^{p_{max}} P_X(x)\cdot x\ dx + \int_{1-p_{max}}^{1} P_X(x)\cdot(1-x)\ dx\right) \tag{6.13}$$

In the following, the SRAM PUF distribution presented in [MTV09b] will serve as reference to compare DSC to state-of-the-art approaches. Distributions for other PUFs can be generated for example with the help of [Mae13],[HSP13].

In Figure 6.5, the bit error probabilities for different syndrome coding schemes are shown over the number of SRAM PUF bits n for embedded bits k with the distribution given in [MTV09b] and mean error probability 15%. In this basic case, no additional ECC is considered such that the key size κ is equal to the number of indexed bits $\kappa = k$. Later, in Section 6.6, an ECC is added after the syndrome decoding, so $\kappa < k$

For DSC and key size $\kappa = k = 128$, n is chosen such that $e_1 = 5 \cdot 10^{-4}$ resulting in block size n for DSC. The average error probability of the indexed bits p_{syn} is given by Eqn. 6.13. The worst case error probability $p_{syn-max}$ covers the unlikely case that all indexed bits have error probability p_{max}, so $p_{syn-max} = p_{max}$.

Note that DSC's maximum error probabilities are in the same range as previous mean error probabilities for low n to k ratios.

Comparing the mean error probabilities, DSC is considerably more efficient than previous work for an n to k ratio of 4. This is caused by the fact that other approaches operate on very small independent blocks with varying reliability. It takes an n to k ratio between 9 and 10 for the Code-Offset Method and a repetition code with *Soft-decision Maximum-Likelihood* (SDML) decoding to approach DSC's performance. This shows that a careful selection of the 10% most reliable PUF bits still has a lower error probability as computing repetition code blocks of size 10. In addition, adding 10 bits and performing the decision whether the sum is larger than 5 or not is significantly more complex than simply forwarding one single bit.

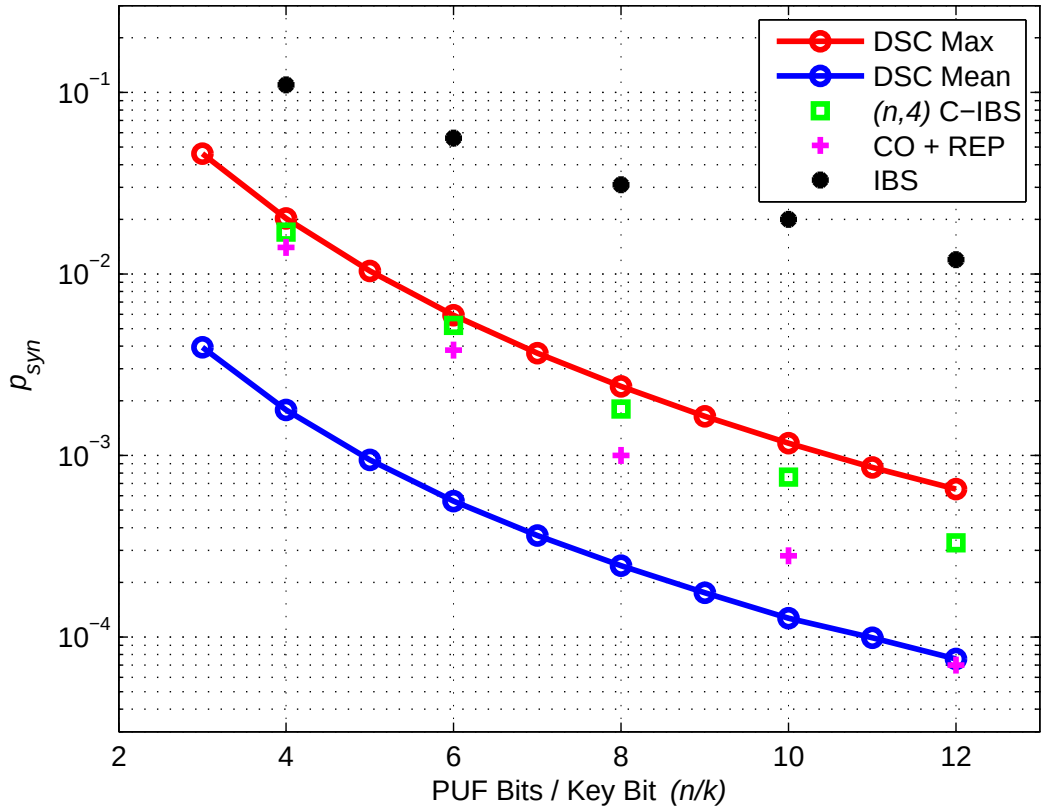

Figure 6.5.: Max and mean bit error probabilities of syndrome coding schemes without second stage ECCs for an SRAM PUF with 15% average bit error probability.

6.4. Security Analysis

This security analysis contains a theoretical and a practical part. The information theoretic analysis quantifies the amount of key information that leaks through the helper data. The helper data manipulation attack shows a vulnerability of hardware implementations of the DSC decoding algorithm and also proposes a generic countermeasure.

6.4.1. Information Theoretic Analysis

For code sequence C^k, PUF response X^n and helper data W^k, the mutual information between the code sequence and the helper data $I(C^k; W^k)$ determines the amount of

secret code information that leaks through the helper data. Let $\overline{X}^k$ be the vector of selected PUF bits in X^n. According to the definition of the mutual information,

$$I(C^k; W^k) = H(C^k) - H(C^k|W^k) \tag{6.14}$$

The helper data element W_i is computed as a function f of all previous helper data W^{i-1}, the current code sequence bit C_i and the selected PUF response bit $\overline{X}_i$.

$$W_i = f(W^{i-1}, (C_i \oplus \overline{X}_i)) \tag{6.15}$$

The distance pointers U^k are selected independently from the key, so they cannot leak any key information as long as reliability and PUF response bit are not correlated. Therefore, the leakage of the helper data W^k only depends on

$$V^k = C^k \oplus \overline{X}^k \tag{6.16}$$

Using the helper data computation in the conditioned entropy in Eqn. 6.14 gives

$$\begin{aligned} H(C^k|W^k) &= H(C^k|C^k \oplus \overline{X}^k) && (6.17)\\ &= H([C^k\ (C^k \oplus \overline{X}^k)]) - H(C^k \oplus \overline{X}^k) && (6.18)\\ &= H(C^k) + H(C^k \oplus \overline{X}^k|C^k) - H(C^k \oplus \overline{X}^k) && (6.19) \end{aligned}$$

$H(C^k \oplus \overline{X}^k|C^k) = H(\overline{X}^k)$, which removes the XOR in the joint entropy, such that

$$H(C^k|W^k) = H(C^k) + H(\overline{X}^k) - H(C^k \oplus \overline{X}^k) \tag{6.20}$$

Using this result in Eqn. 6.14 gives

$$I(C^k; W^k) = H(C^k \oplus \overline{X}^k) - H(\overline{X}^k) \tag{6.21}$$

An upper bound can be given by $H(C^k \oplus \overline{X}^k) \leq k$, so

$$I(C^k; W^k) \leq k - H(\overline{X}^k) \tag{6.22}$$

In general, correlated or biased PUF responses can lead to syndromes which leak key information if too much key information is stored. The main insight from Eqn. 6.21 is that the leakage can be reduced nearly down to zero with diligent code design such that $H(C^k \oplus \overline{X}^k) \leq H(\overline{X}^k) + \epsilon$ holds for a small $\epsilon > 0$.

As a counterexample, for codes with systematic encoding with distinct information and redundancy parts, a PUF with low entropy $H(X^n)$ cannot fully protect the information part that has maximum entropy such that information leaks through the XOR. Therefore, the codes have to be designed in such a way that the bias or correlations in the PUF are also represented in the code structure.

According to the bound in Eqn. 6.22, DSC is information theoretically secure for any code if the PUF has a high entropy such that $H(\overline{X}^k) > k - \epsilon$. According to [KKR+12], this is given for example for the SRAM PUF.

6.4.2. Helper Data Manipulation Attack

In the system model, an attacker has no access to the PUF responses or the key. However, the helper data is typically stored in an unprotected and unauthenticated NVM, so that an attacker can arbitrarily read or modify this data. Furthermore, the attacker can verify if a cryptographic operation cryptop($data\_input, key$) that uses the generated key from the PUF response shows valid behavior or not, for example by observing if a firmware decryption is successful and the system boots properly. This vulnerability and the corresponding attack strategy were found by Michael Weiner [HWRL+13].

In previous pointer based syndrome coding schemes such as IBS or C-IBS, independent blocks of PUF response bits were used for each codeword bit. This corresponds to distinct address spaces such that the attacker cannot point from a reference in one space to a reference in another. This can also be exploited for C-IBS if more than one secret bit is embedded in each block [Hil11].

DSC does not split the PUF response into blocks but uses one long sequence out of which all code sequence bits are referenced in one address space. From a security perspective, having one long sequence for all PUF response bits is problematic: it allows an attacker to compare different PUF response bits by modifying the helper data. As a result, he can learn whether PUF response bits corresponding to c_i and c_{i+1} are equal or the inverse of each other.

In a simplified scenario, no second stage error correction is used after the syndrome coding. The ECC is not required for a successful attack and the simplified scenario makes the problem accessible more easily. As a consequence, the term “codeword” is replaced by “key” for the following attack description.

Figure 6.6 shows the attack strategy, where the attacker manipulates the helper data u^4. In the changed helper data $\tilde{u}^4$, he shifts pointer i to point to position $i+1$, then modifies pointer $i+1$ to point to an unused bit between $i+1$ and $i+2$. He then finally adjusts the distance of pointer $i+2$ such that position $i+2$ and all subsequent pointers are addressed correctly again. The fact that the unused bit is not part of the key implies that its stability is below the required threshold.

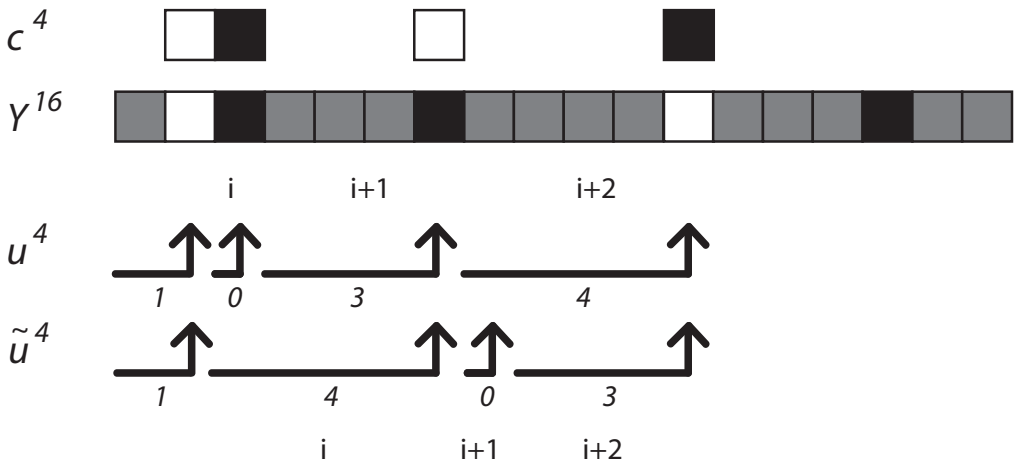

Figure 6.6.: Example for helper data manipulation attack on DSC

Therefore, the attacker can assume that if several DSC key reproductions are performed, this bit will be equal to key bit c_{i+1} in some of the reproductions. Of course, the attacker cannot observe for an individual bit in which of the reproductions this is the case; however, for those cases it becomes important whether $c_i = c_{i+1}$ holds or not. If it holds, the attacker observes that the reproduced key is equal to the original one.

The attacker can evaluate whether there exists a significant number of unchanged keys by observing the output of a cryptographic operation or by verifying whether a cryptographic operation such as firmware decryption is successful. Eventually, the attacker can repeat this procedure for every pair of subsequent bits such that in the ideal case, only one key bit remains unknown, independently of the actual key length.

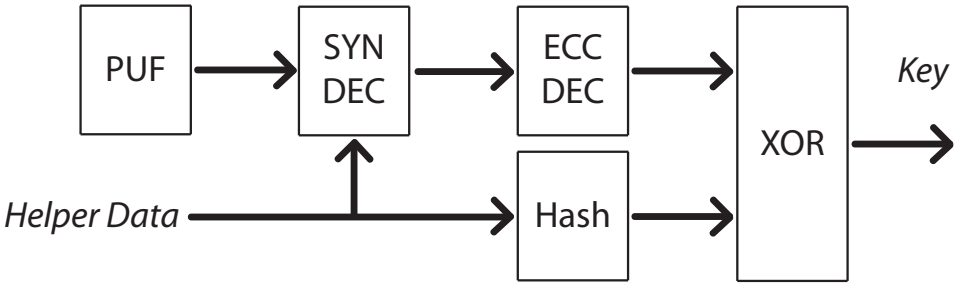

Figure 6.7.: Generic reproduction procedure with countermeasure against helper data manipulation attacks

For a successful attack, it is required to modify a small number of targeted key bits while keeping the rest at its original value. Therefore, the attack can be prevented if the attacker cannot address single key bits anymore. For this reason, a hash function is integrated into the proposed scheme as shown in Figure 6.7. The output of the ECC is XORed with the hash value of the helper data; with this addition, any change in the

helper data will on average lead to a change of 50% of the key bits. Note that in this construction, only the public helper data is fed into the hash function so that it is not threatened by physical attacks such as side-channel attacks [MSSS11].

If a second-stage ECC is present, the attack becomes more cumbersome because one has to first ensure that the ECC will not correct the induced changes by introducing additional faults. A first evaluation of such an attack strategy was carried out by Fuchs [Fuc15] under my supervision.

6.5. Convolutional Codes

Convolutional codes are a powerful and popular class of ECCs that was discovered by Elias [Eli55] in 1955 and showed a very high practical relevance throughout the seventies and eighties of the last century [CJFJ07]. Efficient decoding algorithms such as the Viterbi algorithm [Vit67], as well as other efficient decoding algorithms, are also an important prerequisite for the success of a code class.

Other than suggested in [Mae12], convolutional codes are actually very well-suited for PUFs as the remainder of this chapter demonstrates.

For all classes of channel codes, an information sequence is encoded to a longer code sequence with a code specific structure. The code sequence is exposed to an environment that stochastically changes or erases parts of the sequence. On the decoder side, errors in a given sequence are corrected by detecting distortions that contradict with the code structure and solving these contradictions.

Block codes divide the information sequence into blocks that are encoded independently into code blocks with redundancy. As discussed in Section 6.1, increasing the block length improves the error correction performance, but also increases the decoding complexity.

As one major advantage of convolutional codes, there exist very efficient decoding algorithms [Bos99]. Convolutional codes offer powerful error correction for a low hardware overhead. In this work, the focus is set on the Viterbi algorithm [Vit67] that is discussed in Appendix A.2.

6.5.1. Convolutional Encoder

Similar to a window sliding along the information sequence, every code sequence bit is a function of a constant number of consecutive information bits. This operation is defined mathematically as convolution.

The encoder of an $(2, 1, [\mu])$ convolutional code encodes one input sequence to 2 output sequences c_1, c_2 with

$$c_1 = c_{1,1}c_{1,2}c_{1,3}...$$
$$c_2 = c_{2,1}c_{2,2}c_{2,3}...$$

Output sequences c_1 and c_2 are concatenated to the code sequence c according to

$$c = c_{1_1}c_{2_1}c_{1_2}c_{2_2}...$$

The following discussion only considers $(2, 1, [\mu])$ codes with two output sequences and one input sequence. For more information about convolutional codes with a higher number of input or output sequences see, e.g. [Bos99]. The convolution operation is carried out in hardware by shifting the information sequence through a shift register of length μ, as shown in Figure 6.8.

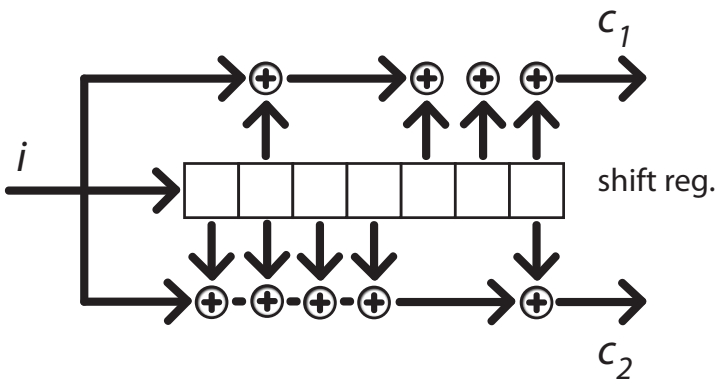

Figure 6.8.: $(2, 1, [7])$ convolutional encoder

Functions of the input bit i_j and the internal state of the decoder $(i_{j-1}, \ldots, i_{j-\mu})$ compute two output bits $(c_{1,j}, c_{2,j})$ for each input bit.

$$c_{1,j} = i_j \oplus i_{j-2} \oplus i_{j-5} \oplus i_{j-6} \oplus i_{j-7}$$
$$c_{2,j} = i_j \oplus i_{j-1} \oplus i_{j-2} \oplus i_{j-3} \oplus i_{j-4} \oplus i_{j-7}$$

As main difference to block codes, every information bit is woven into the code stream with a certain impact length instead of encoding independent blocks where every code bit might be affected by every information bit within the same block. After the last information bit, μ zeros are shifted into the encoder to terminate the sequence, so that the last information bit also has full impact length.

A larger internal state μ increases the number of code sequence bits that are affected by every information bit, which increases the error correction capability of the code,

as well as its implementation complexity. A comprehensive introduction and analysis of convolutional codes and the Viterbi algorithm that will be used in the hardware implementation can be found in [Bos99].

Convolutional codes can also be systematically encoded so that they could also be used in combination with SLLC [Bos99].

6.5.2. Bounding the Bit Error Probability of Convolutional Codes

The bit error probability is a key parameter to evaluate the performance of a code in a given scenario. However, it is computationally infeasible to calculate the precise bit error probability for most practical use cases. So, the straight forward approach is to run Monte Carlo simulations until a statistically significant number of errors is detected to calculate the average bit error probability and ideally also its confidence, see e.g. [Gra07]. For low bit error probabilities, simulations can become quite time and resource consuming.

Bounding techniques simplify a given problem so that it becomes feasible to compute a bound that gives a best or a worst case statement. This will be used in the following to asses the reliability of a large number of key derivation modules with DSC and a convolutional code.

The following upper bound for the bit error probability of convolutional codes is based on the *Bhattacharyya bound* and is discussed in [Bos99]. In the following, the bound is extended for inputs bits with varying bit error probability.

The convolutional code is a linear code. Therefore, the behavior of the entire code can be characterized from the properties of the all zeros sequence.

For the all zeros code sequence, a decoding error occurs in a sequence with Hamming weight w if e bit errors occur with

$$e > \lfloor (w-1)/2 \rfloor \tag{6.23}$$

According to the Bhattacharyya bound, the probability p_w of a decoding error in a sequence with Hamming weight w for a channel with bit error probability p_b is bounded by

$$p_w < \left(2\sqrt{p_b(1-p_b)}\right)^w \tag{6.24}$$

To analyze the mean error over the PUF response distribution, the expectation over the distribution is calculated.

$$E(p_w) < E\left(\left(2\sqrt{p_b(1-p_b)}\right)^w\right) \tag{6.25}$$

For i.i.d. error probabilities every factor can be treated independently.

$$E(p_w) < \left(E\left(2\sqrt{p_b(1-p_b)}\right)\right)^w \tag{6.26}$$

For the further bounding Jensen's inequality [CT06] is applied twice on concave functions. The concavity of the square root function permits the following two bounding steps.

$$E(p_w) < \left(2\sqrt{E\left(p_b(1-p_b)\right)}\right)^w \tag{6.27}$$

Again, $p_b(1-p_b)$ is concave, so

$$E(p_w) < \left(2\sqrt{E(p_b)(1-E(p_b))}\right)^w \tag{6.28}$$

Using $E(p_b) = p_{syn}$ computed in Eqn. 6.13, $E(p_w)$ can be bounded by

$$E(p_w) < \left(2\sqrt{p_{syn}(1-p_{syn})}\right)^w \tag{6.29}$$

Eqn. 6.29 bounds the probability of error if the given bit is part of a specific sequence with Hamming weight w. Instead of iterating over all possible code sequences, the next step iterates over the Hamming weight and assigns $E(p_w)$ to all sequences with Hamming weight w.

Conveniently, the information weight $I(w)$ of a code gives the number of sequences with Hamming weight w. The information weight spectrum of different convolutional codes can be found e.g. in [Con84] and so the output bit error probability is bounded by

$$p_{output\ err} < \sum_w I(w) \cdot E(p_w) < \sum_w I(w) \cdot \left(2\sqrt{p_{syn}(1-p_{syn})}\right)^w \tag{6.30}$$

As a result, Eqn. 6.30 bounds the bit error probability of a convolutional code and thus permits to evade laborious simulations.

6.5.3. Seesaw Viterbi Decoder Architecture

The Viterbi algorithm is a popular decoding algorithm for convolutional codes [Vit67, Bos99]. An introduction to the algorithm and an example can be found in Appendix A.2. Figure 6.9 shows the trellis diagram that is explained in detail in Appendix A.2.

There are several ways to implement a Viterbi decoder in hardware, e.g. [CS93, FG93, EDE04, TSR+05, YTA06, KwA07, SES09]. This section discusses the Viterbi decoder

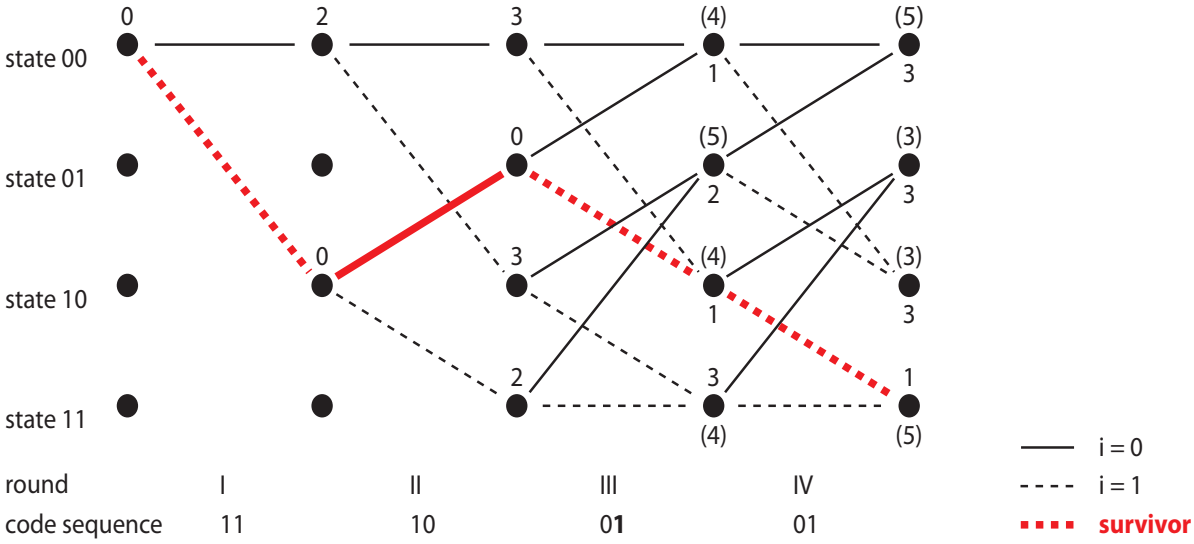

Figure 6.9.: Trellis diagram for a $(2, 1, [2])$ convolutional code

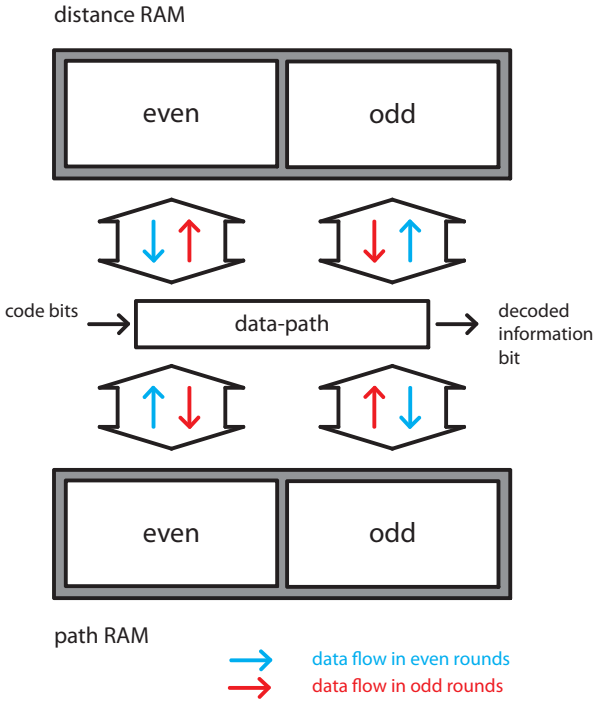

Figure 6.10.: Seesaw architecture and data flow

architecture published in [HRLS14]. Figure 6.10 shows the data flow in the architecture. Two dual-port block RAMs and the data-path in between form the core of the architecture. The *distance RAM* holds the path distances and the *path RAM* holds the previous information bits for each state.

Each RAM is partitioned in an even and an odd data section. One section provides the results of the last round and the other stores the results of the current round. Thus, the read and write accesses toggle between these sections after each round. Block RAM on FPGAs provides large, inflexible but fast chunks of storage. By storing the decoder variables (path distances and survivor paths) from last round and from current round in different memory sections (even and odd), no additional resources are allocated on the FPGA for intermediate results. This reduces the overall size of the Viterbi decoder. Note that there are also higher-density RAM structures available for ICs such that this approach is also in principle transferable to ASICs.

Distance RAM The *even* and *odd* sections of the distance RAM contain 2^{μ} path distances of the size path distance length (PDL) each. This corresponds to the numbers next to all dots in one round in Figure 6.9. The data flow in Figure 6.10 shows that in even rounds, path distances are read from the even section and written to the odd section. In odd rounds, the data flow changes such that the path distances are read from the odd section and written to the even section.

For each path distance calculation, two path distances are read from one section. The trellis in the toy example in Figure 6.9 shows that states 00 and 01, and 10 and 11 have the same possible successors. This also holds for more complex codes. Therefore, their distance values can be stored together in one memory location and accessed with a single memory read. The data-path computes the updated path distance and writes it to a location in the other memory section. Dual port block RAM on Xilinx FPGAs supports two read, or one read and one write access per clock cycle so that the entire operation is performed within one clock cycle. The path is updated in the subsequent clock cycle in the pipelined architecture.

Path RAM The path RAM stores the previous information sequence for each state of the trellis at one point in time. For each state, the surviving paths are updated in every round. After the survivor was chosen in the data-path according to the updated path-distance, the old path of the previous state is read from one memory section. Each address in the path RAM contains the hypothetical information sequence $i_j, i_{j-1}, ..., i_{j-TBD}$ for one state at the time j. The data flow is identical to the data flow of the distances RAM. In even rounds the paths are read from the even section and written to the odd section. And again, in odd rounds the data flow direction switches.

The data is shifted by one and the new information bit is added to the survivor path. Afterwards, the data is written to the corresponding address in the other memory section of the path RAM. The TBD specifies the size of the survivor for each state. Note that typical Xilinx block RAMs store up to 36 bit of data in each address [Xil11]. A TBD of 35 is recommended for the very popular $(2, 1, [7])$ code such that the entire survivor can be stored in one memory location. For a larger TBD, multiple block RAMs can be used in parallel with the same throughput or data can be spread over multiple addresses with a lower throughput and a higher control overhead.

Data-Path A small control module attached to the data-path handles the control of the other modules and the direction of the data flow for the RAMs. Aiming for a low-complexity data-path, similar operations are serialized as far as possible. In contrast to typical Viterbi decoder implementations, only one path-distance and survivor is updated at a time. The almost redundant data representation in the path RAM enables a simple update of the stored paths. A simple shift operation replaces the complex transition through the trellis, which reduces the required logic size.

The path distances only have a finite size, so an overflow handling mechanism has to be implemented for long code sequences. All paths in Figure 6.9 are derived from the same survivor path and only vary in the last rounds. Therefore, the path distance of the common survivor can be seen as a constant that can be subtracted from all path distances. The discussed mechanism can be interpreted as a repeated subtraction of the constant part from all states, as illustrated in Figure 6.11.

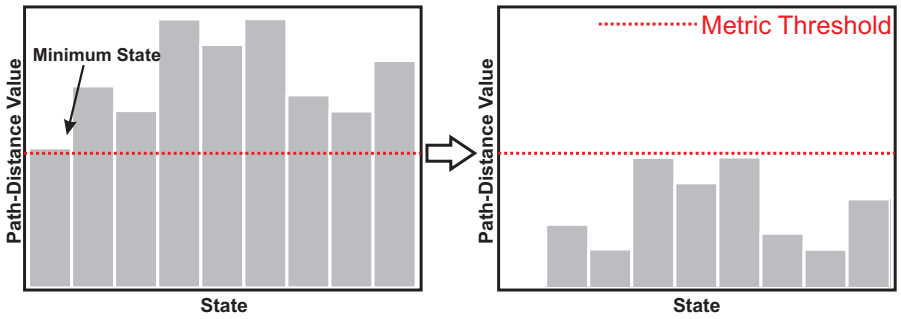

Figure 6.11.: Path-distance vverflow preventing logic

Exploiting the binary representation of the path distances permits to replace the subtraction with setting the most significant bit to zero. As soon as the MSB is set to one for all path distances, it can be set to zero for all locations again. The maximum difference between any path distances can be assumed as bounded by $2 \cdot TBD$, so that the maximum distance that has to be stored is given by $4 \cdot TBD$. As a consequence, it is sufficient to set the PDL to

$$PDL \geq \lceil \log_2 TBD \rceil + 2 \tag{6.31}$$

Both RAMs store two data sets of 2^{μ} values each. The distance RAM contains $2^{\mu+1}(\lceil \log_2 TBD \rceil + 2)$ bits according to Eqn. 6.31, and the typically larger path RAM stores $2^{\mu+1}TBD$ bits. The total amount of RAM bits can be calculated by:

$$RAM\_Bits = 2^{\mu+1}(TBD + \lceil \log_2 TBD \rceil + 2) \tag{6.32}$$

6.6. Design of a Complete Key Derivation Module

The error probabilities after source coding are not sufficient to consider the reproduced secret as reliable cryptographic key. Therefore, PUFs need a second stage of error correction. This section analyzes the performance of DSC concatenated with a convolutional code to derive a parameter set suitable for implementation.

6.6.1. Effect of the Block Size on the Typical Set

In virtually all PUF key generation schemes published to date, block-based error correction is used. This can be in the form of a BCH or Repetition code, or IBS variants. The IBS block size corresponds to 2 to the power of the index size.

The reliability of these schemes is influenced by the larger ϵ value associated with ϵ-letter typical sequences with small block sizes n. As described in the typicality analysis in Section 6.1, the probability of drawing too few reliable bits decreases exponentially with an increase in block size. By using a smaller block size, the prior approaches require more PUF bits to be used for each key bit, as shown in Figure 6.5 for the same level of p_{err}, or alternatively higher p_{err} for the same PUF bit / key bit ratio.

6.6.2. Key Bit Error Probability

An important performance criterion for PUF error correction is the number of input PUF bits that are required and the bit error probability of the output. Therefore, the relation between number of PUF bits n and output bit error probability p_{err} quantifies the efficiency of the decoder. Note that p_{syn} used in Section 6.2 referred to the error probability after syndrome decoding while p_{err} quantifies the error after both syndrome decoding and ECC error correction.

First of all, it is important to set the length μ of the shift register of the convolutional code. Figure 6.12 shows upper bounded values of the mean error probabilities for convolutional codes with memories μ from 2 to 7. The values were obtained with the bounding technique discussed Eqn. 6.30 in Section 6.5.2. For a PUF output to key bit rate of n/k, DSC is configured such that in average $n/(2k)$ bits are indexed. Aiming for a yield of 99.9% the probability of not finding enough PUF response bits is set to $e_1 = 5 \cdot 10^{-4}$ again.

It can be seen that $\mu = 7$ is required to move inside the region of less than 10 PUF response bits per key bit to be more efficient as state-of-the-art work for a mean input PUF response bit error probability of 15%. Going to memory 8 would double the number of operations in the decoder so that μ is set to 7 in the following.

Figure 6.13 compares the simulated key bit error probabilities of DSC with a $(2, 1, [7])$ convolutional code to DSC with BCH codes of various length with bounded minimum

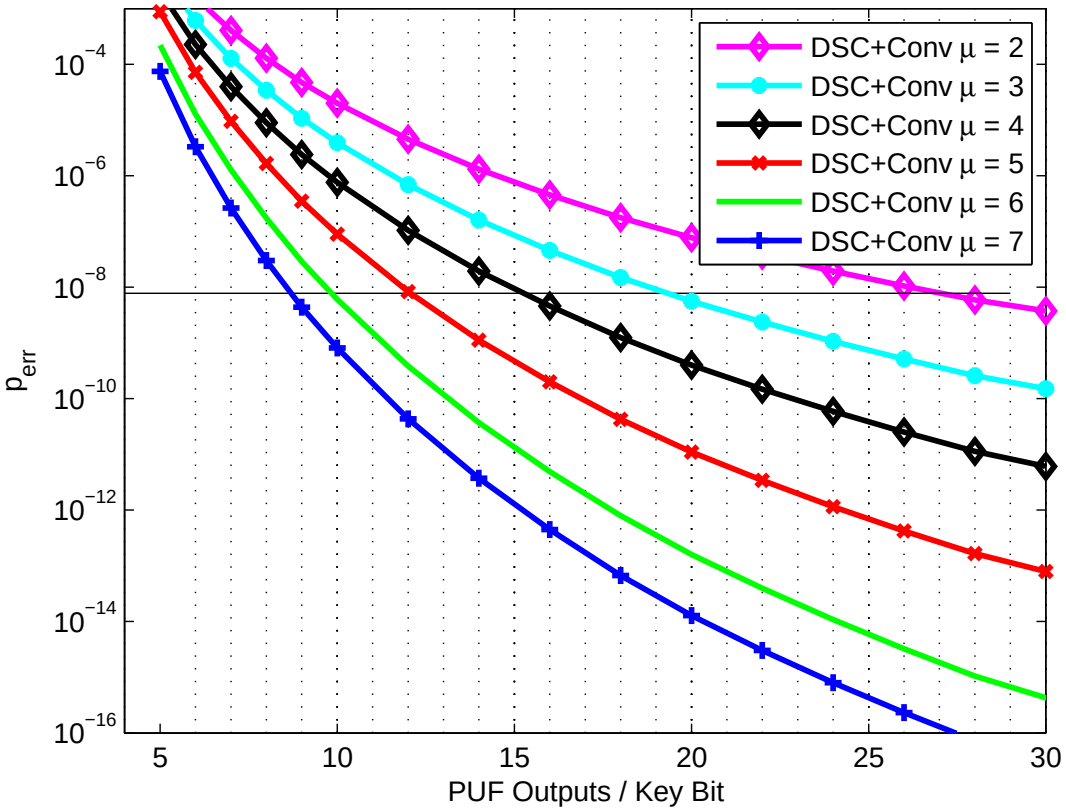

Figure 6.12.: Bounded mean key bit error probabilities of DSC concatenated with different convolutional codes for an SRAM PUF with average bit error probability 15% and $e_1 = 5 \cdot 10^{-4}$.

distance decoding [Bos99]. Looking at the number of PUF response bits per key bit at the error probability of $p_{err} = 7.81 \cdot 10^{-9}$ shows that $(2, 1, [7])$ has roughly the same performance as a $(127, 64, 21)$ BCH code. FPGA implementation sizes and run-times of the two candidates are compared in Table 6.3. The Viterbi decoder uses only 44% of the area of the BCH code such that it has a significant advantage for optimized FPGA implementations using Block RAM. As a side effect it is also 2.6× faster. Therefore, it will be the preferred decoder in the following.

	Slices	Run-Time
$(2, 1, [7])$ Seesaw Viterbi Decoder	21	3,824
$(127, 64, 21)$ BCH Decoder (simple IS) [Ley15]	48	13,952

Table 6.3.: ECC decoders synthesized for Xilinx Spartan 6 FPGAs using Block RAM

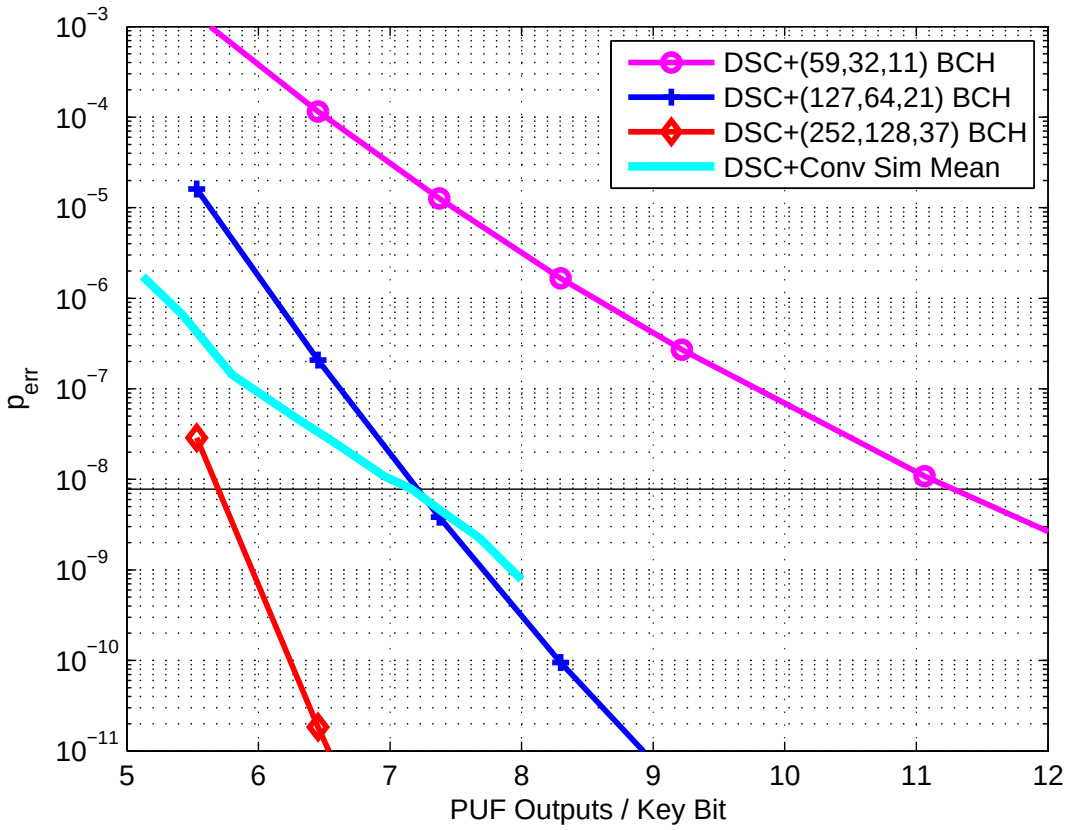

Figure 6.13.: Simulated mean key bit error probabilities of DSC concatenated with a $(2, 1, [7])$ convolutional code compared to bounded mean key bit error probabilities of rate 1/2 BCH codes

Figure 6.14 sets the DSC and convolutional coding approach in relation to previous work. The optimal point is in the lower left corner of the diagram so that a low key error probability is achieved with a low number of PUF bits. Reference values that are all coherent with the scenario in [MTV09b] were taken from [Bös08, Hil11, HMSS12]. Recall that the goal in [MTV09b] is to reproduce a 128 bit key with a key error probability smaller than 10^{-6}, which corresponds to a key bit error probability of $7.81 \cdot 10^{-9}$, from a PUF with an average bit error probability of 15% and a distribution given in [MTV09b].

It can be seen, how the field moved to the left over time. The black diamonds are the Code-Offset (CO) Fuzzy Extractor results by Bösch *et al* [BGS+08, Bös08]. Repetition (REP) codes were concatenated with relatively small BCH and Golay codes and decoded with hard decision decoding. The magenta crosses show the results by Maes *et al.* [MTV09b, MTV09a] where Repetition codes were concatenated with Reed–Muller (RM) codes and decoded with GMC and SDML soft decision decoders [Bos99]. More recent work is more efficient than the the older Bösch *et al.* results, so in general the approach

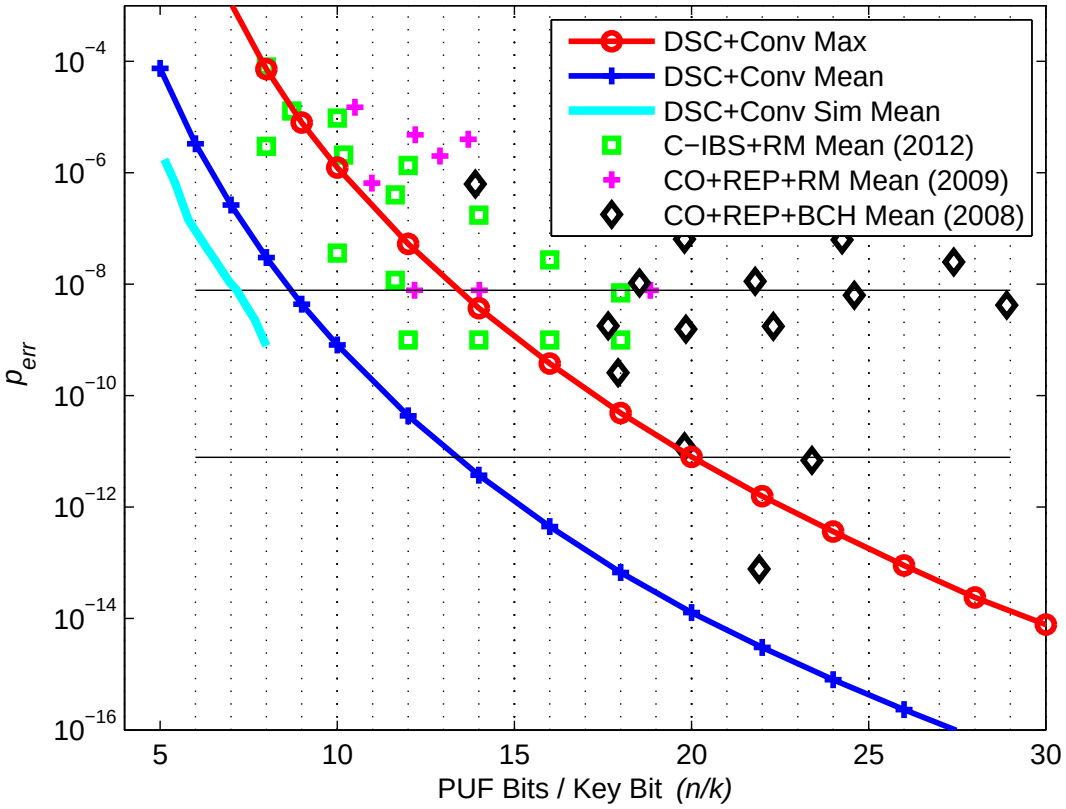

Figure 6.14.: Bounded mean and max key bit error probabilities of DSC concatenated with a $(2, 1, [7])$ convolutional code compared to the state of the art for an SRAM PUF with average bit error probability 15%. Again, $e_1 = 5 \cdot 10^{-4}$.

shows an improved performance so that less PUF bits are required to achieve the same key bit error probabilities. The C-IBS results [Hil11, HMSS12] also use IBS pointers in combination with Repetition codes and Reed–Muller codes and soft-decision decoding. The results overlap widely with the results in [MTV09b, MTV09a] with a light shift towards lower numbers of PUF response bits per key bit.

The two nearly diagonal lines across the entire diagram represent DSC concatenated with a $(2, 1, [7])$ convolutional code. As a first difference, the other approaches have fixed bit numbers that result in points in the diagram. For DSC, p_{max} can be chosen quasi-continuously as fine-grained as the digital representation of the reliability values of the PUF response bits allows. Applying again the bounding techniques discussed in Section 6.5.2 gives a maximum bit error probability for each device and also quick worst-case reference values for the mean error probability.

DSC's maximum error probability is comparable to the mean error probabilities of the state of the art whereas the bounded mean error probabilities of DSC separate the

field from the left corner. This demonstrates that the DSC and convolutional code concatenation is more efficient than previous work over the entire analyzed range.

The bounded results give a very quick and rough estimate on the performance of a scheme. The cyan line, which is the leftmost line and without explicit data points, shows Monte Carlo simulation results that are roughly 1.5 PUF bits per key bit better than the corresponding upper bounded values.

As in [GKST07], this work aims to generate a 128 bit key with an error probability smaller as 10^{-6}. This corresponds to a target bit error probability of $e_t = 7.81 \cdot 10^{-9}$, shown by the horizontal line in Figure 6.14. The simulations have shown that it is possible to reach the target bit error probability of $e_t = 7.81 \cdot 10^{-9}$, shown by the horizontal line, with $p_{max} = 0.027$ by indexing in average $p = 32.6\%$ of the available PUF bits. This specific value was measured by simulating $\nu = 1.9 \cdot 10^{11}$ PUF bits on Intel Core i7 CPUs where each CPU simulated $6.5 \cdot 10^7$ PUF bits per hour in 8 parallel threads. $1,170$ bit errors were found in total, resulting in a measured bit error probability of $e_m = 6.2 \cdot 10^{-9}$. Xiaoqing Wan and Benjamin Nolet contributed to the development of the simulation framework in their Master's Theses [Wan12] and [Nol15], both under my supervision.

In addition to the mean value e_m, confidence intervals quantify the precision of the result. Let k_σ be a scalar to give a number as multiple of standard deviations, i.e. $k_\sigma \cdot \sigma$. The confidence interval is defined as $[e_m - \Delta e, e_m + \Delta e]$. The number of errors in the Monte Carlo simulation follows a binomial distribution. To assess the confidence of the of results, Eqn. 6.33 is used [Gra07], simplified for large number of simulated PUF bits ν

$$\nu = \frac{e_m(1 - e_m) \cdot (k_\sigma)^2}{(\Delta e)^2} \tag{6.33}$$

In the following, Eqn. 6.33 will be solved for two different variables to derive different statements. $k_\sigma = 3.29$ corresponds to a 99.9% confidence interval. Solving Eqn. 6.33 for Δe gives $\Delta e = 6 \cdot 10^{-10}$. Therefore, one can say with a confidence of 99.9% that the setup has a bit error probability smaller than $6.8 \cdot 10^{-9}$.

Next, Eqn. 6.33 is solved for k_σ and Δe is set to $e_t - e_m$. Recall that e_t defines the target bit error probability of $e_t = 7.81 \cdot 10^{-9}$. As a result, the specified maximum error probability e_t has a distance of $k_\sigma = 8.9$ standard deviations from the simulated value e_m. Therefore, the specification e_t is met with a confidence[1] of $1 - 2.5 \cdot 10^{-19}$.

The corresponding number of PUF bits is 974 to embed the required 270 code sequence bits, or 128 key bits. This gives $974 \cdot p = 317.5$ reliable PUF bits in average and requires 270 reliable PUF bits to be able to index the entire code sequence. The 128 key bits are encoded to $2 \cdot 128 = 256$ code sequence bits. Termination [Bos99] requires

[1] Let $cdf_N(\cdot)$ be the *cdf* of the Normal distribution and $2k_\sigma$ the width of the confidence interval. Then, the confidence level is given by $cdf_N(k_\sigma) - cdf_N(-k_\sigma)$

another $2 \cdot 7 = 14$ bits leading to 270 code sequence bits in total. The average overhead is $1 - \frac{317.5}{270} = 17.6\%$. Therefore, letter typical sequences with an ϵ of 0.176 can be accepted. This shows that, even such a low ϵ value can be efficiently realized in practice when the block size is large enough.

6.6.3. Properties Helper Data Compression

After determining the parameters for the error correction in the last section, this section identifies a suitable helper data compression parameter m to increase the efficiency of DSC. The pointer lengths in RLE differ depending on the input and the code parameters. This section shows compression for different parameters to identify the optimal parameter for the given problem. For increasing p in geometric distributions, higher integer numbers u are selected less likely, so the average amount of information per symbol decreases. For very low p, the expression $(1-p)^u$ only decreases very slowly with increasing u. Therefore, many u have similar probabilities which results in a high entropy. For high p, small u are chosen with a high probability and $(1-p)^u$ decreases much faster. This leads to a low entropy.

Recall Figure 6.4 in Section 6.3.2. For low m, the fixed part is rather small whereas the RLE part increases rapidly with increasing source entropy. In contrast, high m have a large fixed part and only slowly increasing RLE parts.

Figure 6.15 shows that for $p = 0.326$, $m = 2$ leads to the lowest helper data size and thus achieves the best compression. With an average pointer size of 2.79, the encoded representation is only 0.03 bit higher than the entropy, so the basic RLE solution almost reaches the entropy. The more advanced RLE with Huffman coding [GVV75] is more efficient for larger alphabets $\mathcal{L}$. However, since small m are more efficient for this scenario, the basic approach by Golomb [Gol66] is analyzed and implemented in the following.

Varying m gives very low overheads for various parameters p so that RLE enables nearly optimal compression for DSC independently of the parameter p.

6.6.4. Yield Analysis

Figure 6.16 shows an empirical $(1-cdf(l))$ function, obtained by Monte Carlo simulation, that corresponds to the overflow probability e_2 in dependency of the maximum helper data size l. According to Figure 6.4, on average 2.79 helper data bits have to be stored for each distance pointer for $p = 0.326$ and $m = 2$. To handle varying helper data sizes requires to assign more helper data storage. Aiming for a yield $\zeta \geq 99.9\%$, one can tolerate overflows with a probability $e_2 \leq 5 \cdot 10^{-4}$.

The average size of the helper data can be reduced significantly compared to the $2,176$ bits of the uncompressed version without reducing the yield. At least $1,070$ helper

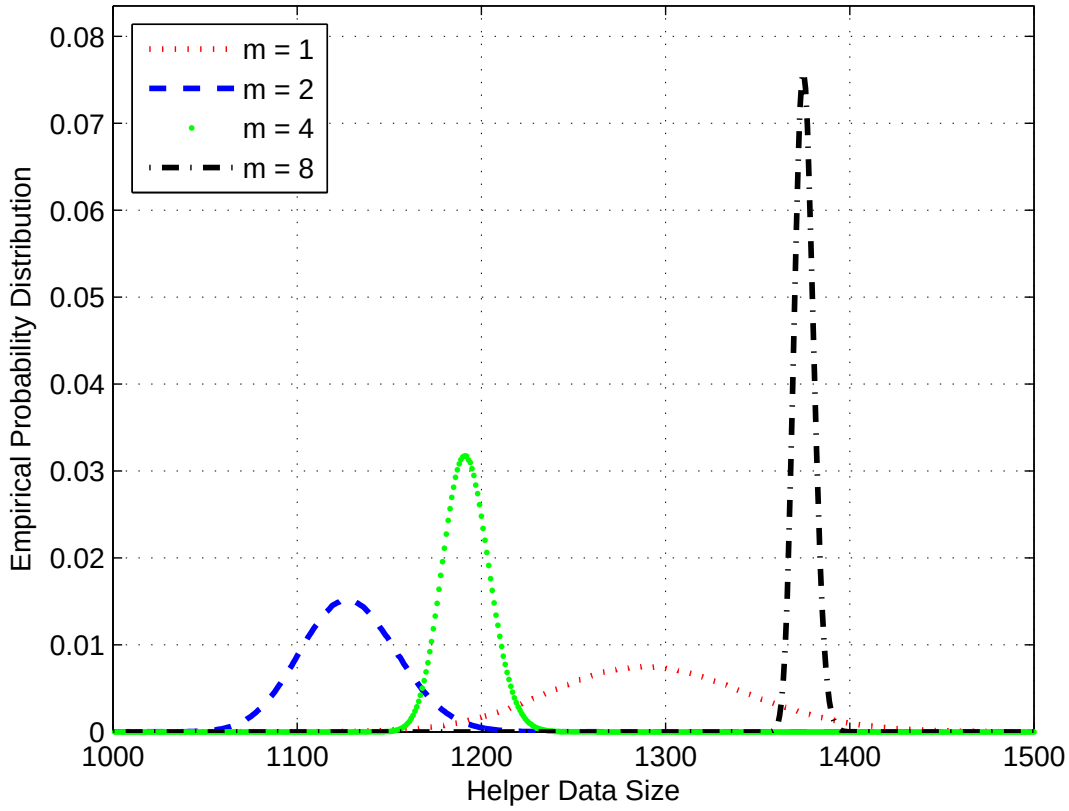

Figure 6.15.: Helper data length distribution functions based on 10^7 simulated PUFs with DSC encoding with $p = 0.326$, RLE helper data compression and a $(2, 1, [7])$ convolutional code

data bits should be assigned for a reasonable yield. However, the error probability $e_2(l)$ decreases by several orders of magnitude for spending 5% to 10% more helper data bits. As a result, $e_2(l) \leq 5 \cdot 10^{-4}$ can be achieved in practice by $l = 1,108$, which is only 8% over the entropy of the helper data.

6.6.5. Comparison with Dark Bit Masking

The DSC setup has the same error probability as dark bit masking combined with a Fuzzy Commitment [JW99] and an identical $(2, 1, [7])$ convolutional code. The average bit error probability of the distribution in [MTV09b] is varied between 10% and 20% and the parameters for a key error probability of 10^{-9} were obtained with the bounding technique discussed in Section 6.5.2. Figure 6.17 shows the average helper data sizes of DSC with helper data compression and the Fuzzy Commitment with Dark Bit Masking.

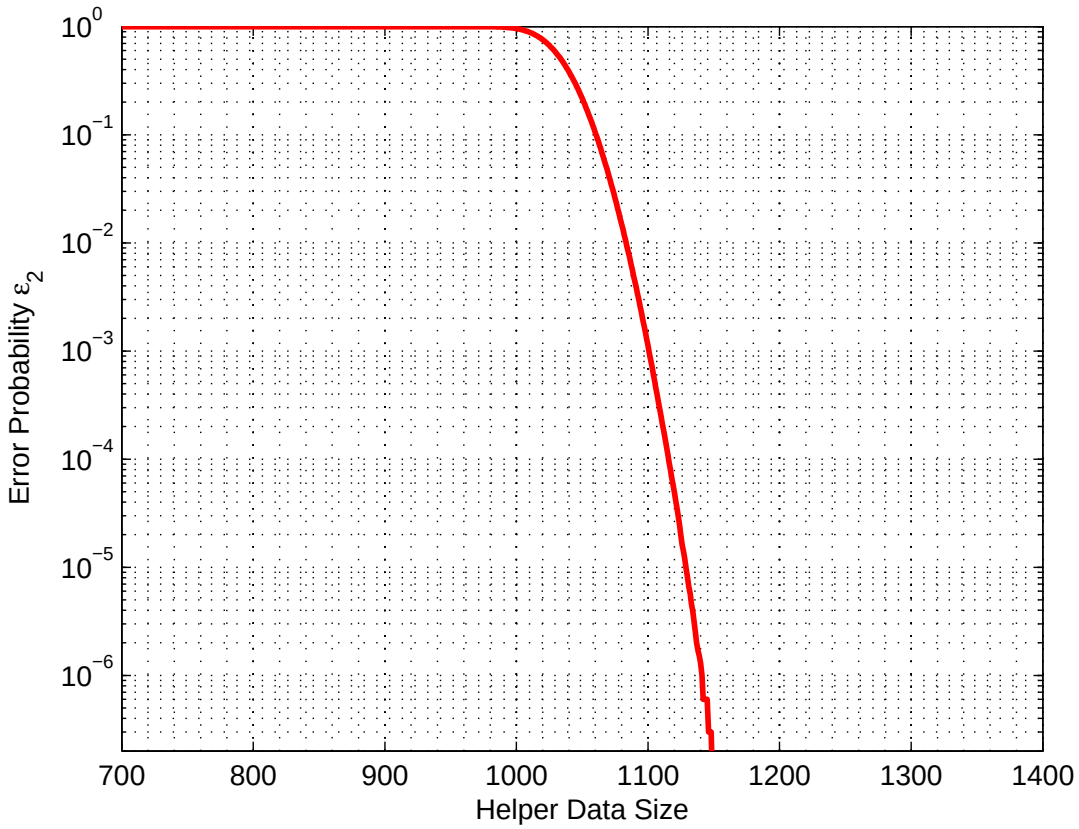

Figure 6.16.: Overflow error probabilities for different fixed helper data sizes and 10^7 simulated PUFs with DSC encoding with $p = 0.326$, helper data compression with $m = 2$ and a $(2, 1, [7])$ convolutional code

The comparison shows that DSC reduces the helper data size by up to 73% compared to the conventional approach. Therefore storing compressed differential pointers is significantly more efficient than selecting PUF bits with a bit mask when only a small fraction of PUF response bits is indexed.

6.7. Implementation

After selecting the parameters for the PUF error correction module, this section presents an overview of the hardware implementation and compares the resource consumption with the state of the art.

The encoding does not have to be part of the final implementation since it can be performed off-chip or with a different configuration bit-stream in a secure environment.

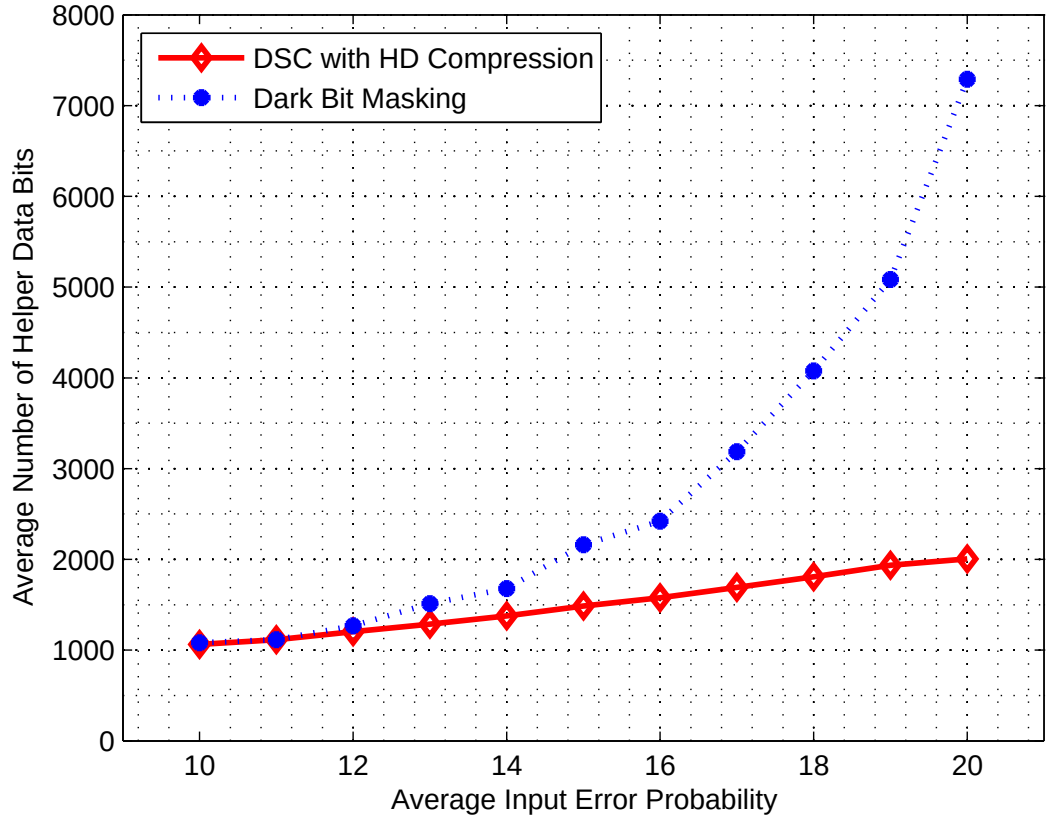

Figure 6.17.: Helper data sizes of DSC with helper data compression and dark bit masking for a key error probability of 10^{-9} and different input error probabilities

Also, the encoder modules only have a fraction of the complexity of the decoders. On a Xilinx Spartan-3E FPGA, the DSC encoder requires 15 slices (10 flip-flops and 25 LUTs) while the convolutional encoder only uses 10 slices (12 flip flops and 11 LUTs) which is roughly a factor of 10 smaller than the corresponding decoders (see Table 6.4). Therefore, the focus is set on the more important and interesting optimized decoder implementation.

For an increased robustness and flexibility, all discussed modules are protected with a double handshake shown in Figure 6.18. If valid data is on a line, the sender asserts the strobe signal *stb*. The receiver acknowledges the received data by asserting *ack* and is ready to process the next data after *ack* is set to zero again.

6.7.1. Hardware Architecture

The block diagram in Figure 6.19 shows the building blocks of the DSC and convolutional code reproduction procedure.

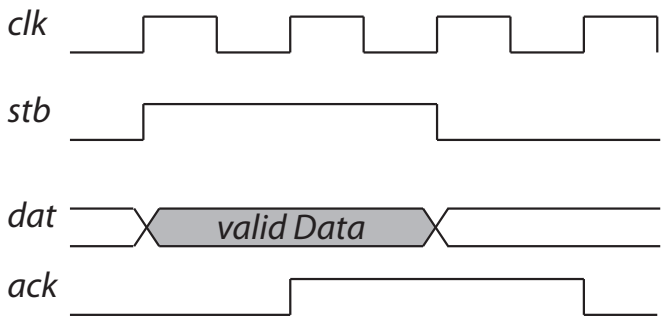

Figure 6.18.: Double handshake IO protocol

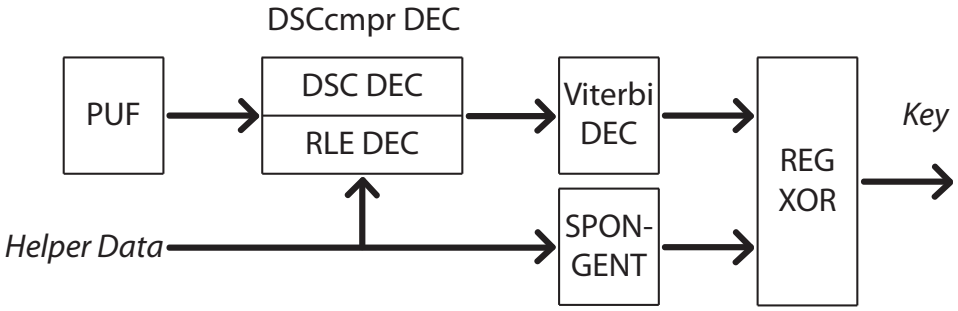

Figure 6.19.: DSC reproduction with helper data compression

The DSC decoder with helper data compression combines the functionality of decoding the helper data pointers, that are RLE encoded [Gol66], and selecting the corresponding incoming PUF response bits. The helper data and the PUF outputs are read sequentially, until the helper data signals that the current incoming PUF bit is the indexed one.

As shown in Figure 6.19, the helper data is hashed onto the output of the Seesaw Viterbi decoder to prevent helper data manipulation attacks as discussed in Section 6.4.2. SPONGENT was selected as a lightweight hash function [BKL+11]. In [JRLH14], we demonstrated that SPONGENT is well-suited for compact FPGA implementations. Therefore, I chose the implementation discussed in [JRLH14] in the smallest configuration that returns an 88-bit hash value.

The REG XOR module XORs the outputs of the Viterbi decoder and the helper data hashed in the SPONGENT module and stores the result in a register. This ensures that 88 key bits are affected by each helper data bit to corrupt the key as soon as the helper data is manipulated to prevent the attack discussed in Section 6.4.2.

6.7.1.1. Optimized SPONGENT

SPONGENT [BKL+11] is a lightweight hash function that is based on the PRESENT block cipher [BKL+07]. It can be seen as a generalization to larger block lengths.

Fig. 6.20 shows the datapath of the SPONGENT implementation used in this work. It was initially developed by Maximilian Birkner [Bir13] and later improved by Leandro Rodrigues Lima, both under my supervision. We published a comparison of this implementation with two other lightweight hash functions in [JRLH14].

To avoid large multiplexers in the design whenever possible, it contains a layer of parallel S-boxes. The three steps of the round function are processed in each round and the result is stored in the state register. No additional serialization takes place in the architecture.

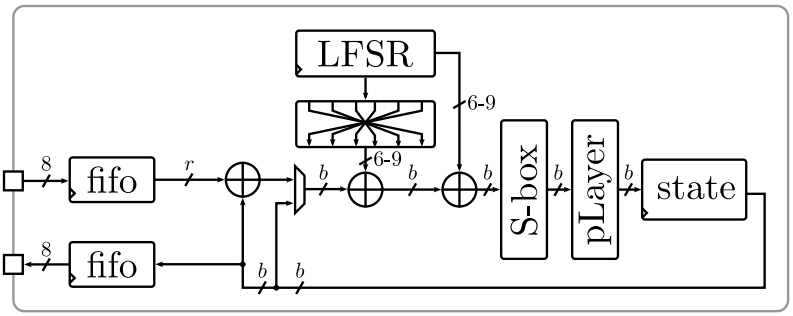

Figure 6.20.: The SPONGENT architecture.

As first step of the round function, a round constant is computed based on the internal state of an LFSR, configured with a primitive polynomial. For SPONGENT, the state is interpreted as one long vector. The state of the LFSR is XORed on the rightmost part of the state. In addition, the same data is reversed and XORed on the leftmost part of the state. Note that the XORs only refer to the right or left $6-9$ bits while the remaining b minus $6-9$ bits are directly forwarded.

For the S-box layer, the state is divided into four-bit blocks. For each mapping of a four bit block to another block of the same size, the mapping function is defined by the PRESENT S-box [BKL+07]. The S-box layer can be implemented in serial or in parallel. A serial implementation would require a large multiplexer to connect all four-bit blocks to the S-box, whereas a parallel implementation is much faster at lower area cost on an FPGA. Therefore, a parallel implementation is more favorable for this application.

The permutation layer is an extension of the (inverse) permutation in PRESENT. It is designed for ASICs, where it can be easily implemented by simple wiring. In contrast, the wiring has to be implemented with look-up tables in FPGAs which causes a slightly increased resource consumption.

6.7.2. Synthesis Results

This section compares the new DSC and Seesaw Viterbi implementation to previous work to evaluate its efficiency. Table 6.4 shows synthesis results for Xilinx Spartan 3 FPGAs and several reference implementations for the same scenario discussed in [MTV09b, MTV09a], namely an SRAM PUF with average bit error probability 15%

with distribution discussed in [MTV09b] and a desired key error probability of 10^{-6} for a 128 bit key. The DSC and convolutional code implementation is by far the most efficient one in terms of required PUF outputs and helper data bits.

	PUF Response Bits	Helper Data Bits	Slices	Block RAM Bits	Clock Cycles
Code-Offset Golay [BGS+08]	3,696	3,824	$\geq$ 907	0	$>$ 24,024
Code-Offset RM-GMC [MTV09a]	1,536	13,952	237	32,768	10,298
C-IBS RM [HMSS12]	2,304	9,216	250	0	$\sim$ 9,000
DSC Conv. Code (bounded)	1,224	2,176	262	11,264	30,846
Compr. DSC Conv. Code. (bounded)	1,224	1,224	272	11,264	33,925
Compr. DSC Seesaw (simulated)	974	1,108	249	10,752	29,243

Table 6.4.: FPGA implementations of reproduction procedures of the DSC and reference implementations synthesized for Xilinx Spartan 3E FPGAs

Tables 6.5 and 6.6 show detailed synthesis results of the architecture for Spartan-3E and Spartan-6 FPGAs. Compared the first DSC results, the Seesaw Viterbi decoder and SPONGENT implementation mainly reduces the number of not fully used block RAMs so that I was able to reduce the overall number to 2 while slightly decreasing the overall size of the top module. This was mainly achieved with a more balanced design by using the spare registers in slices that were already allocated for their LUTs. In addition, more advanced synthesis optimizations were applied to reduce the size.

Replacing the block RAM in Table 6.6 by distributed RAM increases the size of the implementation from 72 to 146 slices.

6.7.3. Evaluation

Comparing the most recent results to the early DSC results shows that using precise simulation results instead of the bounded values allows to decrease the number of PUF

	DSCcmpr Dec	Viterbi Dec	SPONGENT	REG XOR	Entire Module
Slices Total	17	68	85	58	249
Registers	9	56	117	40	247
Logic LUTs	26	75	153	104	388
Block RAM Bits	–	10,752	–	–	10,752

Table 6.5.: Detailed synthesis results of the DSC reproduction procedure implementation for Xilinx Spartan-3E FPGAs

	DSCcmpr Dec	Viterbi Dec	SPONGENT	REG XOR	Entire Module
Slices Total	7	21	24	20	72
Registers	9	55	117	33	235
Logic LUTs	18	77	85	67	251
Block RAM Bits	–	10,752	–	–	10,752

Table 6.6.: Detailed synthesis results of the DSC reproduction procedure implementation for Xilinx Spartan-6 FPGAs

bits by 20%. In addition, the size of the helper data is 9% less than the previous compressed results and 50% less than the default DSC case. The improved implementation slightly reduces the number of slices, block RAM bits and clock cycles.

These results make DSC by far the most efficient approach for this scenario in terms of PUF and helper data bits. DSC enables to generate a reliable key from 974 PUF bits and 1,108 helper data bits for $p_{max} = 0.0270$.

Figure 6.21 compares the DSC Seesaw implementation with helper data compression to the state of the art approaches discussed in Table 6.4. All results are normalized to the maximum number in the corresponding categorie in the comparison. It can be seen that all DSC results, except of the cycle count, are within the 0.4 area while all other approaches have outliers in at least two categories.

All in all, the number of PUF bits is reduced by 36% compared to [MTV09a] and the number of helper data bits by 71% compared to [GKST07], which are both the most efficient approaches for each measure with a significant drawback in the other. The required number of FPGA slices for the DSC implementation is only 5% larger than the smallest reference implementation [MTV09a]. However, optimizing rigorously for area also makes the DSC implementation the slowest investigated one in this comparison with the highest cycle counts, as shown in Table 6.4.

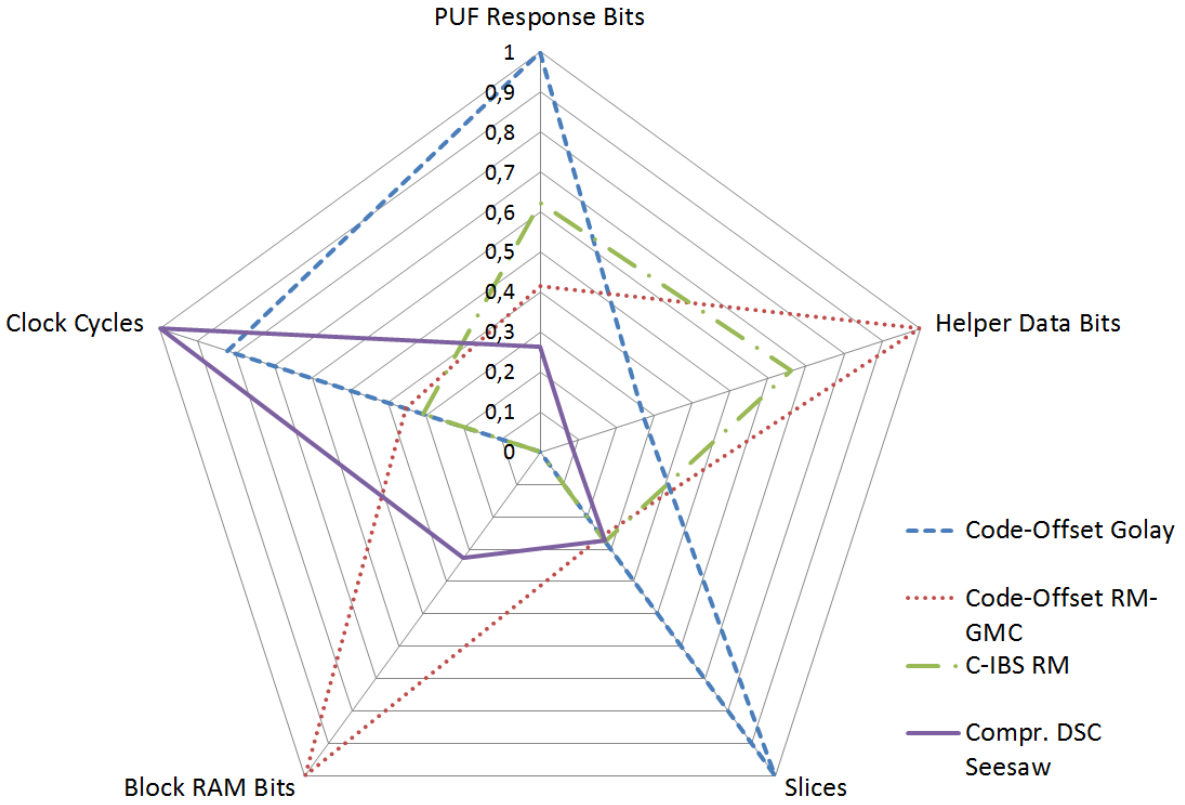

Figure 6.21.: FPGA implementations of reproduction procedures of the DSC and reference implementations synthesized for Xilinx Spartan 3E FPGAs

6.8. Further Improvements

This chapter has shown the theoretical and practical properties of DSC. In addition to the basic DSC version discussed in detail in this chapter, two incremental improvements will be addressed in brief in this section.

6.8.1. Soft-Output Viterbi

Aysun Gurur Önalan investigated in her research internship in 2015 under my supervision the option of using a Reliability Output Viterbi Algorithm (ROVA) [RB98]. The idea for this project came from Prof. Martin Bossert and the results are published in [HOSB16]. The basic idea is to read out the PUF multiple times, compute multiple keys during reproduction and select the most reliable key candidate.

The ROVA decoder can output additional reliability information that indicates how many errors were corrected to reach the given output sequence. If only a small number of errors is corrected, there are two possibilities: there was actually only a small and

well-controlled number of error events, or the transmitted sequence contained so many errors that a decoding error occurred and it was decoded to something else. However, the second event occurs only with a very small probability.

The majority of errors occurs if the result is close to the decision bound and then a wrong decision is made. The reliability output Viterbi algorithm detects cases with a low reliability. Then, the result is discarded and the decoding is carried out another time.

The simulation results have shown that this approach allows to increase p_{max} from 0.027 to 0.1 so that the approach tolerates an almost 4× higher input error probability. As a result, roughly 50% of the PUF bits could be indexed instead of currently 32%.

However this scheme might show some vulnerability against hill climbing attacks, e.g. [SKVdV09], that should be further investigated before usage of the scheme.

6.8.2. Multistage DSC

In the presented approach, DSC only uses one reliability criterion p_{max}. Benjamin Nolet investigated trade-offs for 2 and 3 criteria in his Master's Thesis [Nol15] under my supervision.

The PUF response is first scanned for PUF response bits that fulfill a very strict reliability criterion $p_{max,1}$. Due to their higher reliability, a code with a higher rate can be used to achieve the same output bit error probability. Then, the PUF bits that have reliability between the first criterion $p_{max,1}$ and a weaker criterion $p_{max,2}$ are indexed and a code with a lower rate is used to embed the next bits.

Going to multiple stages did not decrease the number of required PUF response bits but requires more helper data and 2 or 3 DSC iterations. As a consequence, we did not continue any further research in that direction.

6.9. Conclusions

This chapter quantified an algorithm-independent relationship between block size and reliability with the information theoretical concept of typicality for the first time in the PUF context.

I introduced Differential Sequence Coding (DSC), a pointer-based syndrome coding scheme that is able to skip unreliable PUF response bits and can treat the PUF response bits as a single, maximally reliable, block. I have shown its advantages from an information theoretical point of view and compared it to the state of the art.

The hardware implementation requires 36% less PUF bits and 71% less helper data bits than the best reference implementations for a popular SRAM PUF scenario.

Chapter 7.

Evaluation

Chapters 3, 5 and 6 discussed several flavors of error correction and evaluated them for specific scenarios. Different error correction schemes are favorable depending on the characteristics of the PUFs, implementation complexity constraints and the required output reliability of the key. This chapter sets them into a larger context by varying the average input bit error probability of the PUF $\mu(p_{puf})$ and designing the error correction accordingly to reach a target key error probability. SRAM PUFs with the distribution discussed in Appendix A.3 are used exemplarily as input.

Four practical implementation criteria are discussed in this evaluation for each output error probability:

- PUF bits
- Helper data bits
- Slices
- Clock cycles

It is practically not feasible to carefully design optimized implementations for all candidates and multiple parameter sets. Therefore, Section 7.1 discusses how the performance of the parameter sets that are compared in this section are estimated if no dedicated full implementation was created. Section 7.2 puts the SLLC results into context. In Section 7.3, approaches for a medium key output error probability of 10^{-6} are presented and compared. The same analysis is performed in Section 7.4 for candidates that were designed for a key error probability of 10^{-9}.

Tables 7.1 and 7.2, located at the end of the chapter, wrap up the results in a compact representation and provide the precise numbers. They contain error correction parameters for PUFs with average bit error probabilities $\mu(p_{puf})$ between 10^{-5} and 25% for target key error probabilities of 10^{-6} and 10^{-9}.

The performance of the new approaches presented in this thesis is compared to published results of state-of-the-art schemes. All results refer to Xilinx Spartan 6 FPGAs so that they are fully comparable.

7.1. Estimation of Implementation Complexity

To get a bigger picture on the performance of different approaches, this section extrapolates the implementation complexity of various parameter sets from the reference implementations by using conservative estimates. If a specific parameter set is selected for implementation, the previously discussed methods can be applied to optimize the implementation and further increase the performance to achieve more competitive results.

The required numbers of PUF response bits for DSC were obtained by applying the bounding technique presented in Section 6.5.2 except for the 15% average input error probability and 10^{-6} output key error probability data point that was discussed in detail in Section 6.6. The expected helper data size for compressed helper data refers to RLE m parameters chosen as power of 2. As shown in Section 6.6.3, a small overhead has to be added for a reasonable yield. This overhead is neglected in the resource estimation. The implementation sizes and cycle counts refer to helper data compression with $m = 2$. Values of up to $m = 16$ will cause a slight overhead in the decoder. The (2,1,[7]) Viterbi decoder with Seesaw architecture, discussed in Section 6.5.3, remains identical for all approaches. Reading a PUF response bit is assumed to take two clock cycles and a third one is assumed for the helper data handling and control overhead. Therefore, the remaining DSC cycle counts are estimated based on the reference implementation for average input error probability $\mu(p_{puf}) = 15\%$ and 10^{-6}.

The BCH code implementation results were obtained with the design discussed in [Ley15] using the advanced instruction set architecture that is optimized for area, not for speed. This implementation type has a relatively constant implementation size for different parameters. There are only minor changes in register widths while the actual instruction set is independent of the code parameters. However, the number of clock cycles increases significantly with the decoding complexity, which highly depends on the code length and code distance. For the SPONGENT-(128/128/8), a delay of $1,120$ clock cycles are estimated for the 16 bytes with 70 rounds each. The implementation discussed in [JRLH14] uses 44 FPGA slices.

For a fair comparison, all implementation sizes refer to modules with distributed RAM where no Block RAM is used.

7.2. Assessment of SLLC to the State of the Art

Most state of the art approaches were designed for mean input error probabilities $\mu(p_{puf}) > 10\%$ using more than $n = 700$ PUF bits to generate $k = 128$ key bits. Reducing n helper data bits to $n - k$ saves less than 20% of the overall helper data bits which is only a small incremental improvement.

As discussed in Section 5.3, SLLC can show its benefits best for very reliable PUFs where it becomes significantly more efficient than other approaches. Therefore, DSC

and SLLC complement each other for different scenarios instead of competing in the same range.

The SLLC data point from Section 5.3 is also shown in Section 7.4 to show this discrepancy. Due to the lacking reference implementations in this range, it is hard to compare SLLC directly to other optimized and published implementations.

7.3. Syndrome Coding and ECC Designs for Medium Key Error Probability

A key error probability of 10^{-6} is widely used in previous literature as tolerable output key error probability, e.g. in [GKST07, BGS+08, MTV09a]. Several implementations were designed and analyzes were performed for this scenario so that this section gives a comprehensive listing of previous work and sets the new work into context. Table 7.1 shows previous implementations for different average input bit error probabilities and the distribution given in [MTV09b]. The numbers of PUF and helper data bits demonstrate the effectiveness of the approaches while the implementation complexity can be seen by the number of slices and an estimate on the run time of the different approaches.

There are four candidates that will be discussed in this section:

- The Code-Offset approach with Repetition and BCH codes [Bös08] serves as base line and is the oldest candidate in this comparison. Since the paper only contains results for Spartan 3 FPGAs, new Spartan 6 results were obtained with the BCH decoder implementation proposed in [Ley15]. The old Toeplitz Hash [Kra94] was replaced with a more recent SPONGENT 128/128/8 hash function [BKL+13]. The Code-Offset Fuzzy Extractor could be replaced by SLLC which would reduce the helper data size by the size of the reproduced secret.
- The Repetition and Reed–Muller code Code-Offset Fuzzy Extractor [MTV09a, MTV09b] is a previous implementation with soft-decision decoding that requires soft input information on the reliability of specific PUF bits.
- The C-IBS syndrome coding with Reed–Muller ECC presented in [Hil11, HMSS12] is a pointer-based approach that also uses soft-decision decoding with relatively short block lengths.
- The DSC and Viterbi results were discussed in detail in Chapter 6. In contrast to the soft-decision approaches, DSC only requires a binary reliability indicator λ, whether the error probability of a specific PUF bit higher or lower than the threshold p_{max}.

The ML symbol-based approach in [YHD15] is optimized for a strong PUF that provides large numbers of PUF bits. It is designed for a different scenario, so that a direct comparison has to be done with care. In the following analysis, the mean error probability

of the PUF $\mu(p_{puf})$ serves as common reference and is always plotted as x-axis of the figures. The different y-axes visualize the different implementation measures that are also later given in columns 3 to 6 in Table 7.1, and later in Table 7.2.

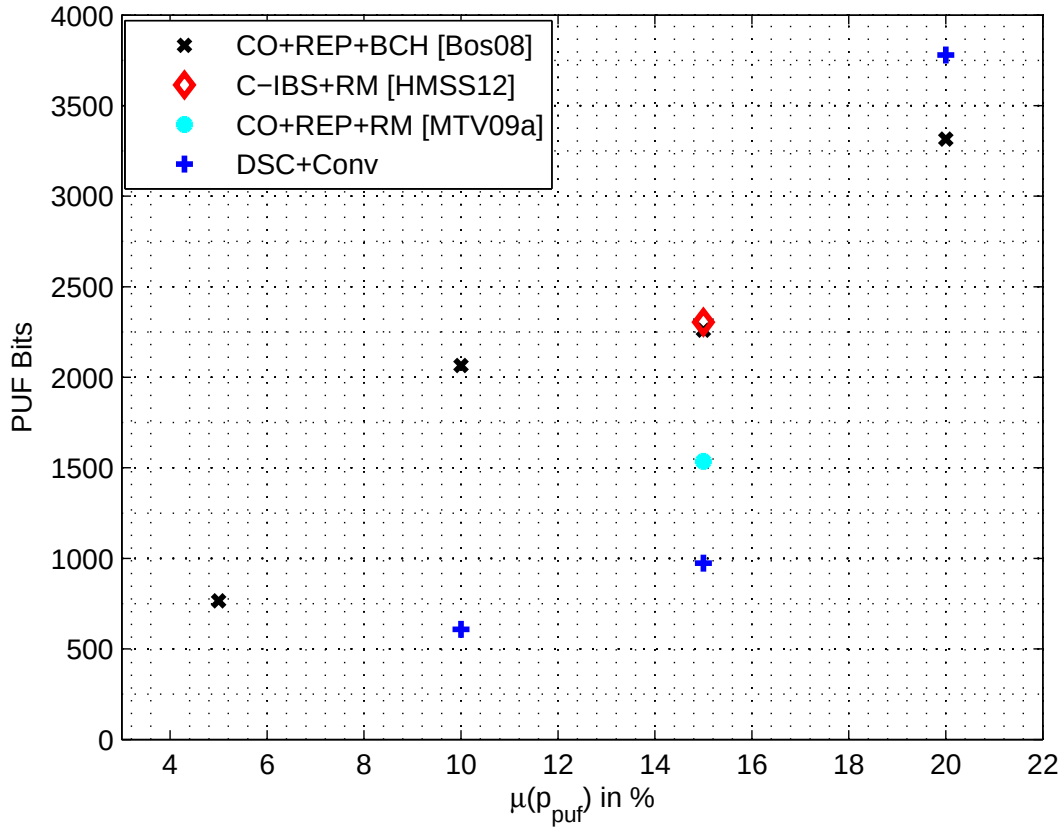

Figure 7.1.: Number of PUF bits of different syndrome coding and ECC approaches designed for a key error probability of 10^{-6}

PUF Response Bits Figure 7.1 visualizes the numbers of PUF bits of the different approaches. The results for the 15% data point were already analyzed in detail in Chapter 6. Note that the black Code-Offset BCH point lies underneath the Code-Offset RM data point. DSC shows low PUF bit counts in the middle area. The C-IBS and Code-Offset RM approaches both benefit from the soft-decision decoding. C-IBS is able to achieve the same performance as Code-Offset BCH with a significantly shorter code length, while the Code-Offset RM approach is more efficient than the other two reference points. As already discussed in Chapter 6, the DSC approach is more efficient than the three references in the 15% point.

For 10% input error, DSC requires only 30% of the PUF response bits of the Code-Offset BCH implementation. Going to 20% input error probability at the other side of

the scale, the Code-Offset BCH approach outperforms DSC at the expense of a BCH code with a long code length of 255. For DSC, it might be useful to go to a $(2, 1, [8])$ code for 20% input error probability to bring the number of PUF bits down again.

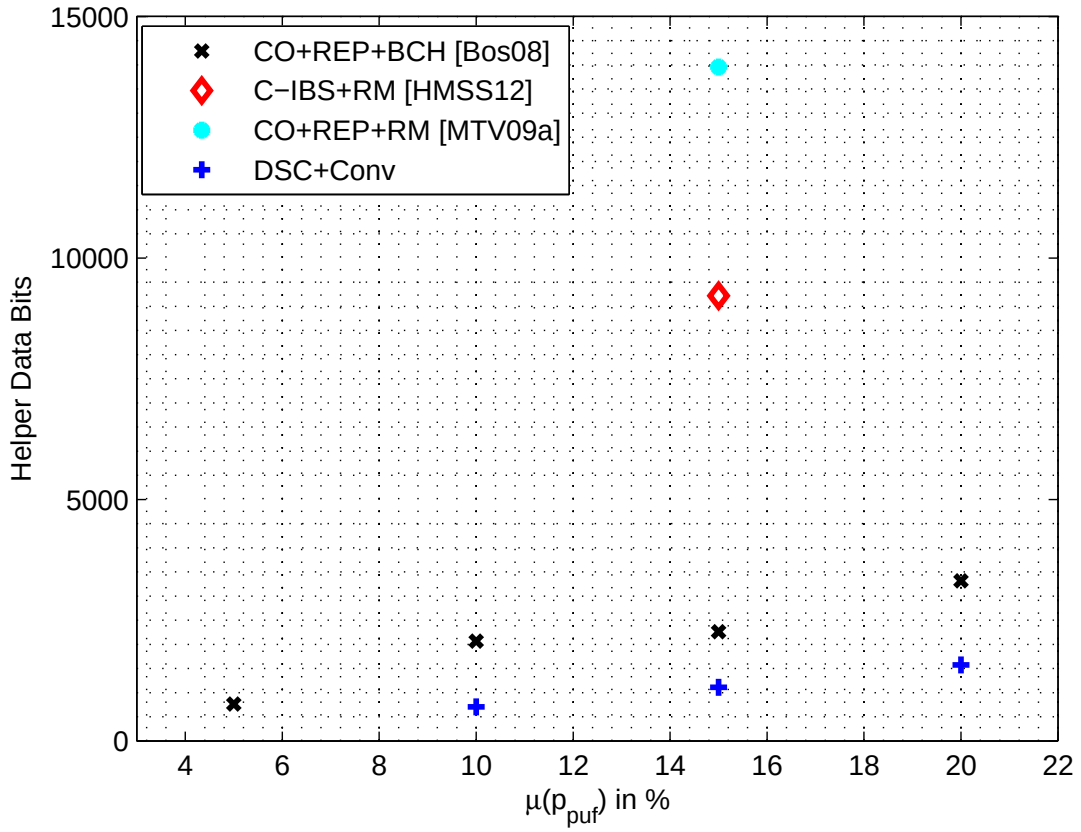

Figure 7.2.: Number of helper data bits of different syndrome coding and ECC approaches designed for a key error probability of 10^{-6}

Helper Data As expected, the helper data size results in Figure 7.2 show that the hard-decision approaches, namely the Code-Offset and BCH code construction and the DSC approach, have low helper data counts that scale almost linearly with the input error probability. As discussed in Section 6.6.5, the DSC helper data compression shows the largest impact for a high PUF bit to key bit ratio, as it is required for PUF with high mean error probabilities on the right side of the x-axis. Storing reliability information in the helper data in [MTV09a] and [HMSS12] increases the numbers by a factor of 3× and more such that these approaches have a significant disadvantage in helper data size.

Slices Figure 7.3 provides the first part of the implementation complexity results for the analyzed approaches, given by the slice counts. The synthesis results in [MTV09a]

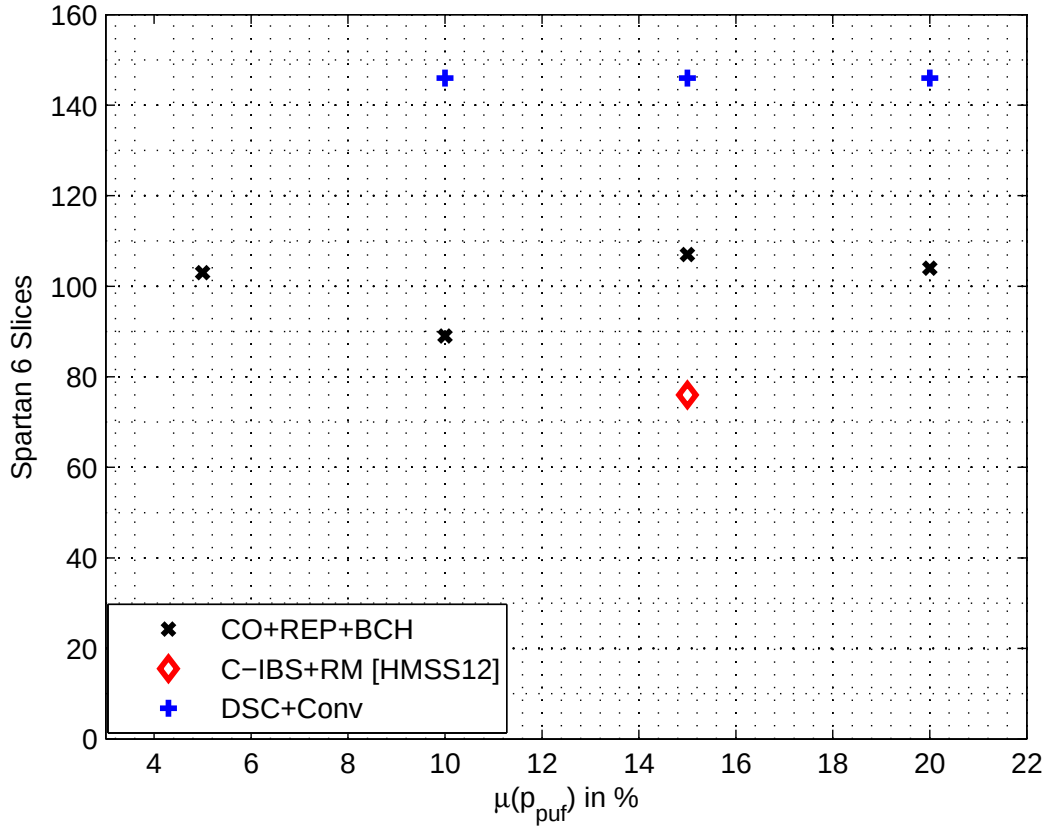

Figure 7.3.: Number of Spartan 6 slices of different syndrome coding and ECC approaches designed for a key error probability of 10^{-6}

are for a Spartan 3E FGPA so that they cannot be compared to the other approaches. The BCH and the Seesaw Viterbi decoders show a fairly constant behavior for different input error probabilities, since the block size only has a minor impact on the size of the decoder in the selected processor-based architecture [Ley15] and the $(2, 1, [7])$ convolutional code is identical over all DSC data points. Seesaw takes great advantage of Block RAM such that the slice count of the DSC and Viterbi module doubles if distributed RAM is used. In this case, the BCH decoder is roughly 25% to 30% smaller than the Seesaw Viterbi decoder. As soon as FPGA-specific optimization with Block RAM is permitted, the results in Chapter 6 hold where the DSC Seesaw is significantly more area efficient. The C-IBS RM implementation with code length 8 in [HMSS12] is small and fast with the trade-off that fine-grained reliability information is stored in the helper data. Also, a larger number of PUF bits is necessary to achieve the same reliability at the output, compared to DSC.

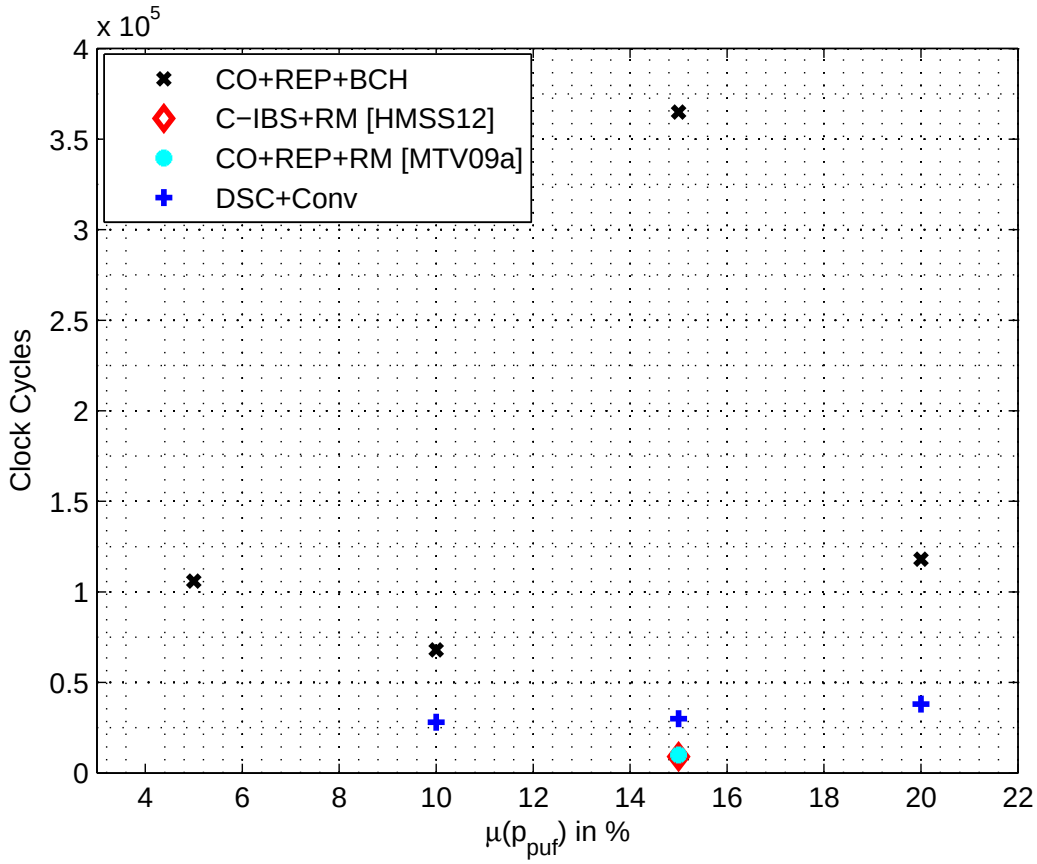

Figure 7.4.: Number of clock cycles of different syndrome coding and ECC approaches designed for a key error probability of 10^{-6}

Clock Cycles The cycle counts of the implementations are shown in Figure 7.4. They differ over several orders of magnitude due to the different code classes and architectures. Especially the cycle counts of the BCH decoders show a large variation because the number of operations is highly dependent on the code length and code rate. A lower rate leads to more syndrome equations that have to be checked in the decoder. The 15% data point is the slowest in this comparison because a length 255 BCH code with a low rate is used. The key is spread over two BCH codewords so that two time consuming BCH decoding operations are necessary. Going to a larger repetition code in the 20% data point such that only one BCH codeword is used speeds up the decoding at the expense that the number of PUF bits also goes up. DSC shows a rather linear behavior as the number of PUF bits increases since the decoder remains constant. The Reed–Muller implementations are the fastest in this comparison. However, note that the Reed–Muller decoder in the next chapter is relatively slow such that this property does not hold in general.

Considering a trade-off between all four categories, DSC shows a high performance in PUF bits, helper data bits and clock cycles with an increased implementation complexity since no block RAM is used in this comparison. The Code-Offset approach with Rep and BCH code has a high number of PUF bits and by far the highest clock cycle counts in the field. The soft decision approaches in [MTV09a] and [HMSS12] lead to a significant helper data overhead. A graphic comparison of the 15% input error probability data point can be found in Figure 6.21 in Chapter 6. The precise numbers discussed in this section can be found in Table 7.1 at the end of this chapter.

7.4. Syndrome Coding and ECC Designs for Low Key Error Probability

This section analyzes error correction schemes that were designed for a key error probability of 10^{-9}, which was used for example in [MVHV12]. The decrease in error probability can only be achieved by correcting more errors which requires a more powerful error correction.

Six implementation candidates are compared in this section:

- The SLLC and BCH code candidate was discussed in detail in Chapter 5. It was designed as lightweight solution for extremely low input error probabilities such that it is not directly comparable to the other designs in the field.
- Again, the Repetition and BCH code fuzzy extractor based on the results in [Bös08] serves as reference.
- The PUFKY design is an optimized practical implementation published in [MVHV12]. It was designed as stand-alone IP core together with an RO PUF.
- Our design and implementation of a Code-Offset and RM construction with GCC [PMB$^+$15, HKS$^+$15][Kür14] shows that the decoding complexity can be reduced by using shorter block sizes concatenated with a GCC construction.
- The Reed–Muller and Reed–Solomon code construction is the first that shows the potential of RS codes in the PUF context [PMB$^+$15]. [PMB$^+$15] also contains a more sophisticated GCC RS construction that will not be taken into account here.
- Finally, the bounded DSC and Viterbi results set the contributions provided in Chapter 6 in a larger context, also for a lower output key error probability.

PUF Response Bits Figure 7.5 has parallels to Figure 7.1. Again, DSC shows a low number of PUF bits for expected input error probabilities 10% and 15%, while the 20% data point exceeds the Code-Offset Rep and BCH code approach. In the 13% to 15% range, the DSC result is slightly better than Code-Offset results in [MVHV12] and

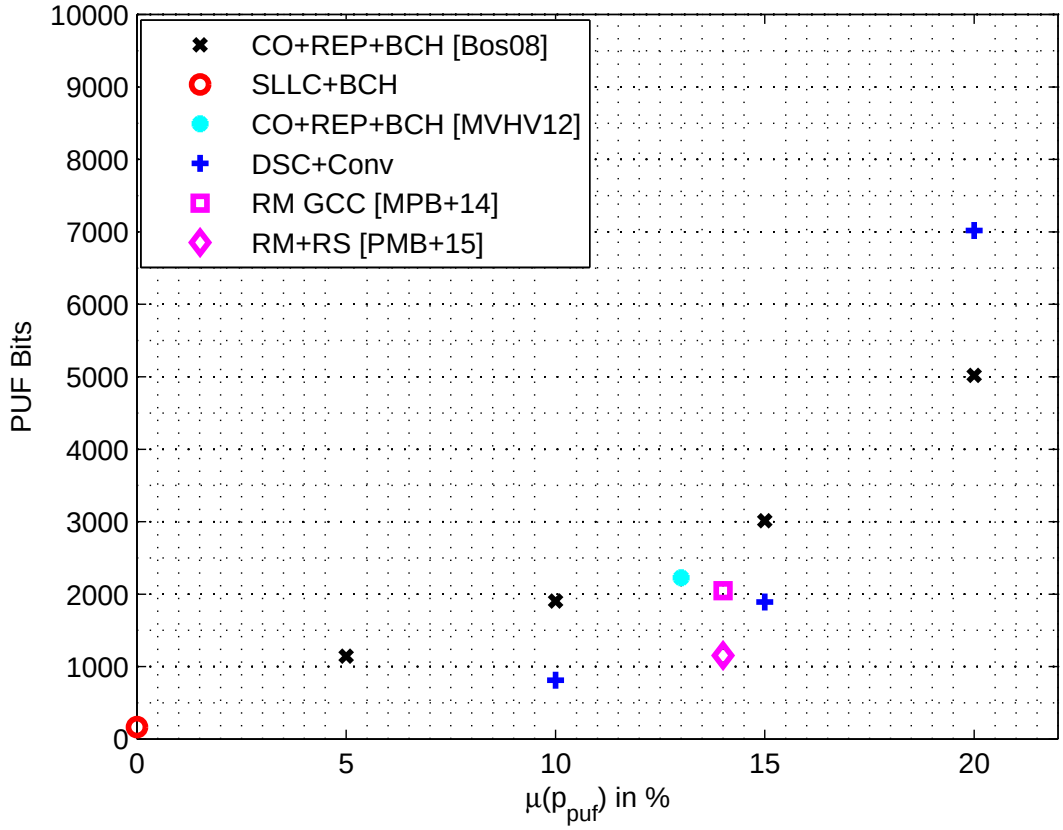

Figure 7.5.: Number of PUF bits of different syndrome coding and ECC approaches designed for a key error probability of 10^{-9}

[MPB+14]. Both require roughly two thirds of the PUF response bits of the Code-Offset Repetition and BCH implementation [Bös08]. Using only roughly half of the PUF bits of the other constructions, the Reed–Solomon construction in [PMB+15] has the best performance in this input error range.

Helper Data The helper data sizes differ significantly for the different approaches. For the Code-Offset approach, the numbers of helper data bits in Figure 7.6 are identical to the numbers of PUF response bits in Figure 7.5. For the Syndrome approach and SLLC the helper data size is given by the difference between secret size and number of PUF response bits.

The helper data compression for DSC mitigates the increase in helper data size for larger input error rates. Increasing the m parameter of the RLE encoder only leads to a slow increase in helper data size over $\mu(p_{puf})$ as discussed previously in Figure 6.4. Note that in contrast to the 10^{-6} scenario, none of the approaches relies on reliability

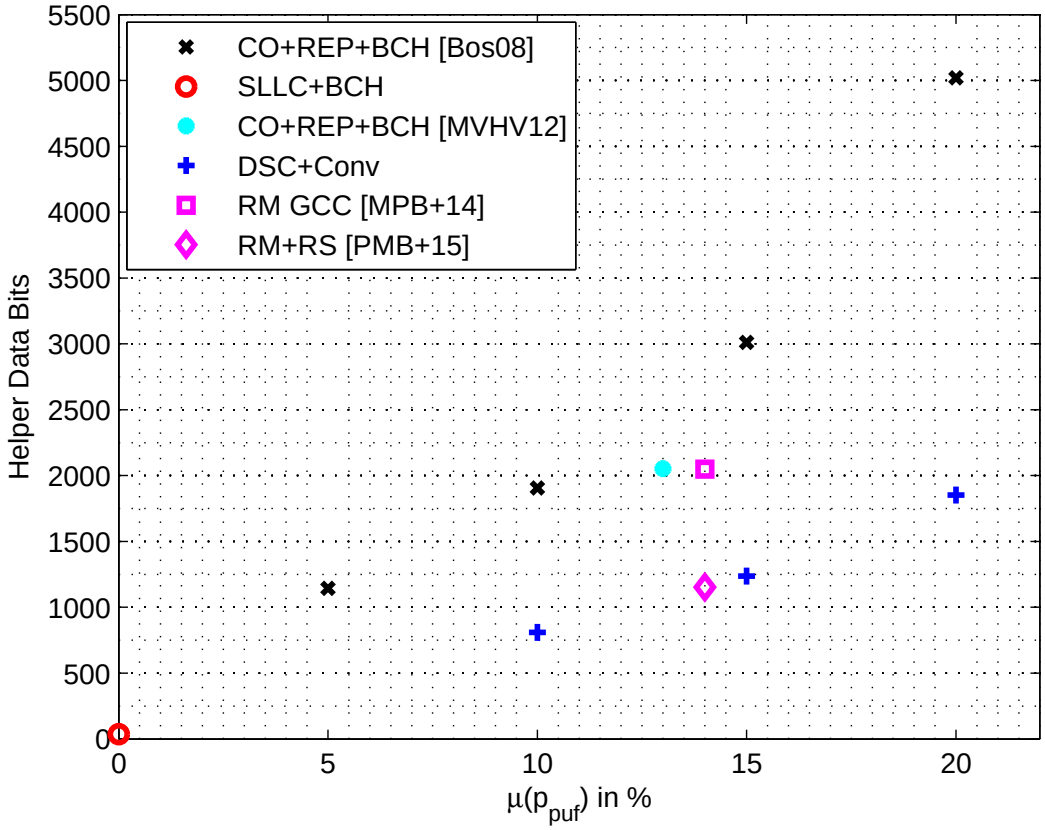

Figure 7.6.: Number of helper data bits of different syndrome coding and ECC approaches designed for a key error probability of 10^{-9}

information stored in helper data such that the maximum numbers are lower than in the 10^{-6} scenario.

Slices Figure 7.7 shows the numbers of slices of the different implementations. Due to the parallels in the implementations, again the BCH decoder presented in [Ley15] leads to a compact implementation of the Code-Offset Repetition and BCH, and the SLLC construction. The DSC slice counts are slightly higher while the implementations of [MVHV12] and [HKS+15],[Kür14] are above of 200 slices.

So far, there are no implementation results available for the Reed–Solomon construction discussed in [PMB+15]. Also note that for the implementation of the Reed–Muller Construction with GCC in [Kür14][HKS+15], the recursive decoder was replaced by a Reed decoder [MS77], which leads to a slightly increased decoder error probability.

Clock Cycles The cycle counts differ by two orders of magnitude in Figure 7.8 so that the clock cycles are plotted on a logarithmic scale. For the given compact BCH decoder,

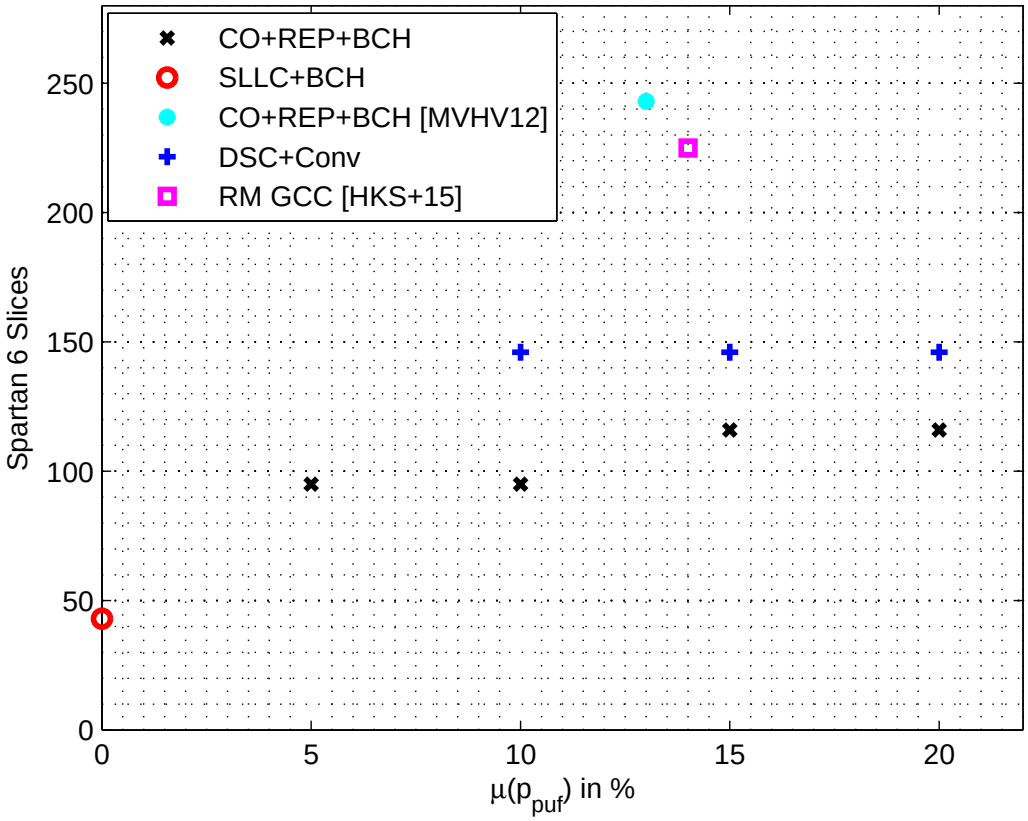

Figure 7.7.: Number of Spartan 6 slices of different syndrome coding and ECC approaches designed for a key error probability of 10^{-9}

the Code-Offset Repetition and BCH implementations are on average over one order of magnitude slower as the remaining candidates.

Combining the results in Figures 7.7 and 7.8, it is noteworthy that the DSC implementation is smaller and faster as implementations based on the Code-Offset and Syndrome constructions in [MVHV12] and [HKS+15],[Kür14]. This is mainly caused by the fact that significantly more errors have to be corrected by the ECC while DSC already reduced the average input error probability by the indexing step during helper data generation.

All in all, again DSC offers a good trade-off between PUF bits, helper data, FPGA slices and clock cycles at the price that information on the reliability of individual bits has to be obtained during generation. Other approaches are stronger in the individual categories with drawbacks in others. An compilation of the numbers plotted in the figures can be found in Table 7.2.

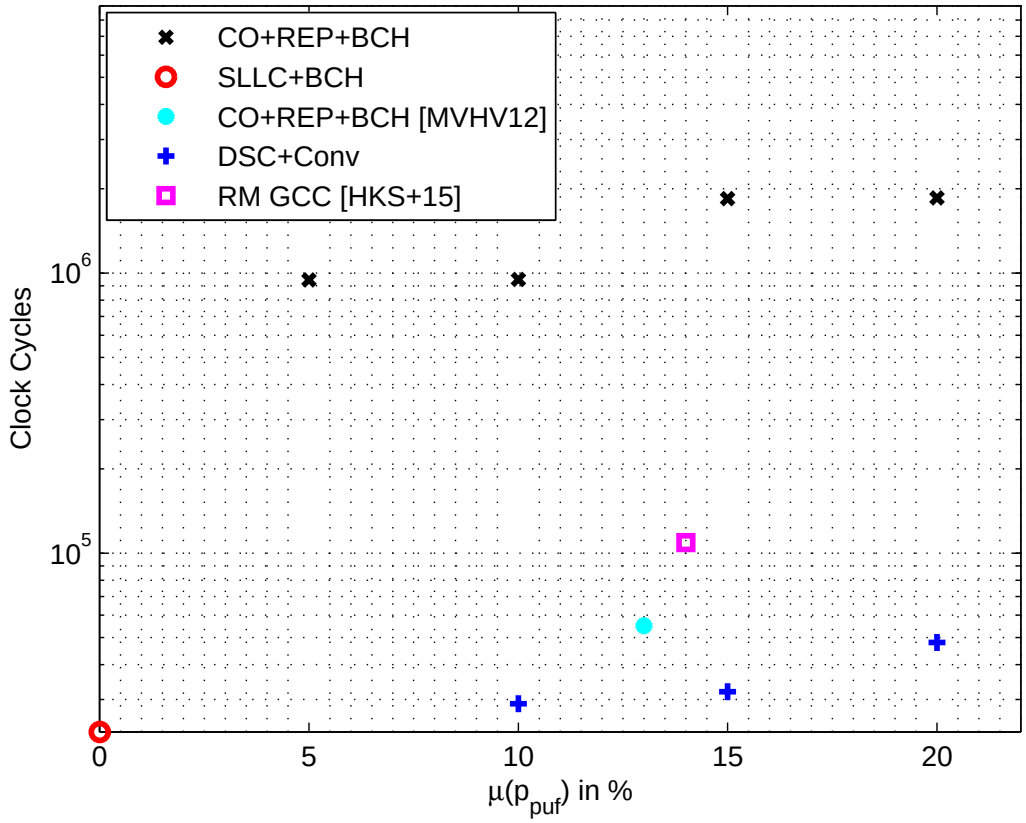

Figure 7.8.: Number of clock cycles of different syndrome coding and ECC approaches designed for a key error probability of 10^{-9}

7.5. Conclusions

The comparison has shown that there is no superb one-size-fits-all solution so that different approaches have their merit for different scenarios. The previous sections show that the DSC and convolutional code solution, presented in the previous chapter, offers a good trade-off between the different criteria and can be applied if a PUF bit specific reliability indicator is available.

PUF Bit Err. Prob.	Approach (+ Reference)	PUF Bits	HD Bits	Num Slices	Run Time Estimate	Comment
5%	CO + Rep(3,1,3) + BCH(255,171,23) [Bös08]	765	765	103	106,000	
10%	CO + Rep(7,1,7) + BCH(59,35,9) [Bös08]	2,065	2,065	89	68,000	
10%	DSC + Viterbi + SPONGENT 88	608	707	146	28,000	Binary Rely Indicator
15%	CO + Rep(5,1,5) + BCH(226,86,43) [Bös08]	2,260	2,260	107	365,000	
15%	CO + Rep(3,1,3) + RM(64,22,16) [MTV09a]	1,536	13,952	–	10,000	Soft Input Information
15%	C-IBS(9,4) + RM(8,4,4) [HMSS12]	2,304	9,216	76	9,000	Soft Input Information
15%	DSC + Viterbi + SPONGENT 88	974	1,108	146	30,000	Binary Rely Indicator
20%	CO + Rep(13,1,13) + BCH(255,171,23) [Bös08]	3,315	3,315	104	118,000	
20%	DSC + Viterbi + SPONGENT 88	3,780	1,575	146	38,000	Binary Rely Indicator
24%	ML Symbol Approach + non-binary PC [YHD15]	753,664	2,944	–	–	
25%	DSC + Viterbi + SPONGENT 88	15,120	1,977	146	72,000	Binary Rely Indicator

Table 7.1.: Comparison of different approaches with target key error probability 10^{-6} synthesized for Xilinx Spartan 6 FPGAs. Approaches where I contributed to are indicated by gray boxes.

PUF Bit Err. Prob.	Approach (+ Reference)	PUF Bits	HD Bits	Num Slices	Run Time Estimate	Comment
10^{-5}	SLLC + BCH(55,43,5)	165	36	43	23,000	
5%	CO + Rep(3,1,3) + BCH(127,57,23) [Bös08]	1,143	1,143	95	945,000	
10%	CO + Rep(5,1,5) + BCH(127,57,23) [Bös08]	1,905	1,905	95	949,000	
10%	DSC + Viterbi + SPONGENT 88	810	810	146	29,000	Binary Rely Indicator
13%	CO + Rep(7,1,7) + BCH(318,174,34) [MVHV12]	2,226	2,052	243	55,000	
14%	CO + RM(GCC) + SPONGENT 128 [PMB+15, HKS+15]	2,048	2,048	225	109,000	Err Prob 1.5×10^{-9}
14%	CO + RM + RS + SPONGENT 128 [PMB+15]	1,152	1,152	–	–	
15%	CO + Rep(3,1,3) + BCH(251,43,85) [Bös08]	3,012	3,012	116	1,843,000	
15%	DSC + Viterbi + SPONGENT 88	1,890	1,236	146	32,000	Binary Rely Indicator
20%	CO + Rep(5,1,5) + BCH(243,43,85) [Bös08]	5,020	5,020	116	1,853,000	
20%	DSC + Viterbi + SPONGENT 88	7,020	1,852	146	48,000	Binary Rely Indicator

Table 7.2.: Comparison of different approaches with target key error probability 10^{-9} synthesized for Xilinx Spartan 6 FPGAs. Approaches where I contributed to are indicated by gray boxes.

Chapter 8.

Conclusions and Outlook

The design of the error correction is a critical step to bring secure derivation storage with PUFs into practice. With the Fuzzy Commitment [JW99] and Code-Offset Fuzzy Extractor [DRS04] as starting points, several new algorithms and implementations were presented over the last years. This thesis aimed to improve this field by providing contributions in different directions. Section 8.1 briefly wraps up the major contributions of this work while Section 8.2 gives some ideas on open problems for future work.

8.1. Review of the Contributions in this Thesis

This thesis contains theoretical as well as practical contributions. Table 8.1 briefly wraps up the contributions of the different chapters of this work.

Chapter 4 addressed fundamental theoretical properties of secure key derivation with PUFs. A good theoretical model is an important prerequisite to analyze a problem. In Section 4.1, I have shown that the practical problem behaves very similar to the information theoretical problem of secret key agreement from a compound source. The mutual information between the secret key and the helper data is a critical measure for the security of all key derivation approaches with PUFs. The analysis of the rank loss in Section 4.3 demonstrated that it is sufficient to look at the algebraic properties of a scheme to provide a first upper bound on the secrecy leakage through the helper data.

I translated an information theoretical random coding approach into a practically implementable scheme in Chapter 5. Section 5.1 introduced the new scheme and analyzed its theoretical properties in Section 5.2. The implementation sketch in Section 5.3 has demonstrated that the new approach significantly reduces the gate count of an error correction module compared to its closest competitors.

Chapter 6 presented DSC, a new sequence based error correction approach that overcomes the limitations of the small block sizes of the state of the art. In Section 6.1, I applied to concept of typicality to show the shortcoming of the state of the art that small block sizes inhibit efficient error correction. The new approach DSC treats the PUF response as one long sequence as shown in Section 6.2. I also demonstrate that the helper data size can be reduced significantly by compressing the DSC distance pointer with RLE. In the security analysis in Section 6.4 I have discussed that the information leakage can be brought to zero and also that DSC is prone to helper data manipulation

	Theoretical Contributions	Practical Contributions
Chapter 4		
Section 4.1	Equivalencies between Compound Source Model and Error Correction for PUFs	
Section 4.3	Unified Algebraic Representation and Security Criterion	
Chapter 5		
Sections 5.1	SLLC Code Construction	SLLC Code Construction
Section 5.2	Theoretical Evaluation	
Section 5.3		Implementation Sketch
Chapter 6		
Section 6.1	Typicality Discussion	
Sections 6.2 & 6.4		DSC Construction, Helper Data Compression and Security Analysis
Section 6.5		Convolutional Codes for PUFs
Sections 6.6 & 6.7		Performance Analysis and FPGA Implementation
Chapter 7		
		Review of the New Approaches in Relation to the State of the Art

Table 8.1.: Overview over theoretical and practical contributions in this thesis

attacks. In addition, I also presented a countermeasure that mitigates this issue. I am the first to use convolutional codes in the PUF context, as discussed in Section 6.5. The performance analysis in Section 6.6 revealed that the combination of DSC and convolutional codes is significantly more efficient than previous work. The implementation in Section 6.7 therefore has a higher performance as the state of the art with only a small increase in resource consumption.

In Chapter 7, I set the results of this thesis in context to existing implementations for output key error probabilities of 10^{-6} and 10^{-9}. This evaluation has shown that the new DSC and SLLC schemes provide the most efficient error correction to date over a wide range of parameters.

8.2. Outlook

This thesis is one puzzle piece of the endeavor of bringing secret keys from PUFs closer to products that effect our everyday lives. Therefore some questions were answered, some were put aside, and also new ones arose. This section discusses some points of future research.

- The innovation in the PUF field focused more on syndrome coding than on channel coding. One open point is to look for more new results in coding theory that provide efficient error correction and low decoder complexities for short block lengths.
- In Section 4.3 I introduced the Algebraic Core to address the secrecy leakage of different error correction approaches. So far, most approaches are based on matrices. However, especially when BCH and Reed-Solomon codes are used, this evaluation may also be extended to polynomials in finite fields.
- Estimating the reliability of PUF response bits in practice is a hard but important issue that was not addressed in detail. This leads to a trade-off between quantization of the PUF response, number of samples and confidence of the results under the additional condition of different cost constraints.
- So far, the error correction was optimized for area. For some applications also the run time can be very critical such that research for low latency schemes is an important open problem that might require new solutions.
- In the existing work, rather ideal PUFs were assumed. Going to biased or correlated PUF bits as for example in [DGV+16] leads to new challenges that require new specially tailored error correction solutions.

Appendix A.

Supplementary Material

Chapter 4 references key agreement from compound sources and Chapter 5 introduces a deterministic approach without discussing the random coding approach. Section A.1 is intended to fill this gap and provide more background information.

Section A.2 explains the Viterbi algorithm that plays a key role in Chapter 6.

The SRAM PUF reliability distribution that is used throughout this work is provided in Section A.3.

A.1. Information Theoretical Key Agreement from Compound Sources with Random Codes

In addition to the practical key storage schemes in Chapter 3, this section discusses an information theoretic approach using random coding. Analyzing only the code construction and omitting the efficiency of encoding and decoding algorithms, or even their implementations, allows to reduce the problem to its fundamentals.

Early work on this problem was carried out by Ahlswede and Csiszar [AC93] and Maurer [Mau93]. Boche and Schäfer introduced an optimal combined syndrome coding and error correction construction for a compound source in [BW13] and later work on compound sources can be found in Grigorescu *et al.* [GBS15] and Tavangaran *et al.* [TBS15].

Figure A.1, which was already shown in Chapter 4, gives the involved components and random variables for key generation with a compound source. The compound source with state $t \in \mathcal{T}$ returns the correlated sequences ${X_t}^n$ and ${Y_t}^n$

A large number of random codebooks is created such that any output of the source ${X_t}^n$ is a codeword of one of the codes with probability close to one. The more recent approach [BW13] the old approach [AC93] differ here. The codewords in [AC93] are chosen such that they form an (n, k, ϵ) code for the channel between X and Y, while only typical sequences are considered as possible codewords in [BW13].

Alice transmits the number of the code as helper data to indicate Bob which code to use for decoding ${Y_t}^n$. The adversary Eve only knows that ${X_t}^n$ is one of the 2^k codewords of the code with the number that is transmitted. However, she received no information

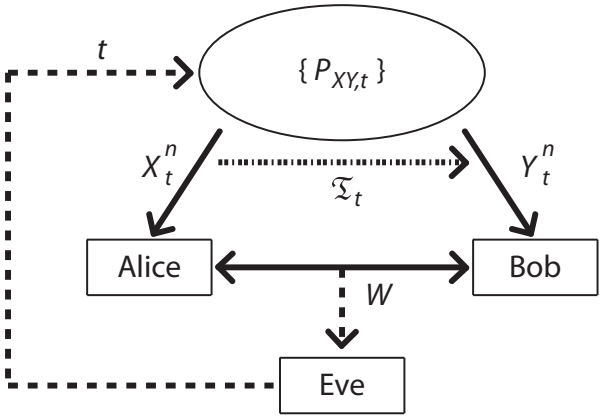

Figure A.1.: Secret key generation with a compound source

which of the 2^k codewords in the codebook is equal to ${X_t}^n$. Therefore, this information can be used as secret key.

For the sake of simplicity, this discussion is limited to the use case for PUFs where X^n is evaluated in a controlled manufacturing environment in which the state of the source is fixed. To achieve error probabilities $< \epsilon$ for all possible states and a small $\epsilon > 0$, capacity achieving (n, k, ϵ) error-correcting codes have a code rate

$$R_{Code} = \frac{k}{n} = \max_{t \in \mathcal{T}} H(X|Y_t) \tag{A.1}$$

Before running the actual key agreement process, ψ random (n, k, ϵ) codes $\mathcal{C}_i$ $(i = 0, ..., \psi - 1)$ with disjoint codewords are created. The codes $\mathcal{C}_i$ are created in sequential order, starting with code $\mathcal{C}_0$. As code generation procedure of codeword C_j $(j \in 0, ..., 2^k - 1)$ of code $\mathcal{C}_i$, an output sequence of length n and with distribution P_X is drawn from the source (PUF). If C_j is not already a codeword of the code $\mathcal{C}_i$ $(C_j \notin \mathcal{C}_i = \{C_0, ..., C_{j-1}\})$ or any other previously generated code $\mathcal{C}_0$,...,$\mathcal{C}_{i-1}$ $(C_j \notin \bigcup_{l=0}^{i-1} \mathcal{C}_l)$, the new codeword C_j is added to $\mathcal{C}_i$. Otherwise, the sequence is discarded and a new sequence is drawn. This process is continued until all ψ codes contain 2^k codewords.

The union over all codes covers the most likely output sequences X^n, so if the number of codes ψ is sufficiently large

$$\Pr[X^n \in \bigcup_{i=0}^{\psi-1} \mathcal{C}_i] > 1 - \eta \tag{A.2}$$

for a small $\eta > 0$. Note that the codebooks are public, so also accessible to the adversary Eve.

Starting with the actual key agreement, Alice draws the sequence X^n from the source. She transmits (or the PUF saves) the index i of the selected code $\mathcal{C}_i$ as helper data W such that

$$W(X^n) = \begin{cases} i, & \text{if } X^n \in \mathcal{C}_i, \text{and } i \in \{0, ..., \psi - 1\} \\ 0, & otherwise \end{cases} \tag{A.3}$$

According to Eqn. A.2, the probability of the *otherwise* case in Eqn. A.3 is bounded by η so that this helper data generation process is sufficiently reliable.

All codes are (n, k, ϵ) codes for the given channel, so Bob (or the PUF in the field) can reconstruct the correct key L from W and Y_t^n with a small error probability such that

$$\Pr[K \neq L | X^n \in \mathcal{C}_i] < \epsilon \tag{A.4}$$

Considering both error events given in Eqns. A.2 and A.4 leads to an overall error probability of

$$\Pr[K \neq L] < \epsilon + \eta \tag{A.5}$$

The proofs in [BW13] show that this approach is capacity achieving and that the security condition, Condition 4.3, in the definition of an achievable key rate in Section 4.2, is satisfied with $I(W; K) = 0$ for i.i.d. PUF response bits.

Random codes show properties of (n, k, ϵ) codes, so the error correction problem is solved in theory. However, Alice and Bob have to store all random codebooks, which makes this approach infeasible in practice, especially for PUFs when they are used in resource constrained lightweight embedded systems.

A.2. Viterbi Algorithm

As already discussed in Section 6.5, a $(2, 1, [\mu])$ convolutional encoder encodes one input sequence into two output sequences. Figure A.2 shows an encoder of a $(2, 1, [2])$ code that will be used as example in the following. This section is based on [HRLS14].

The Viterbi algorithm [Vit67, Bos99] is a powerful decoding algorithm for convolutional codes because it is a ML decoder. This means that the decoder always makes the best possible decision based on its input information.

Every state of the shift register in a convolutional encoder only depends on the previous μ information bits. The decoding complexity increases exponentially with μ, but only

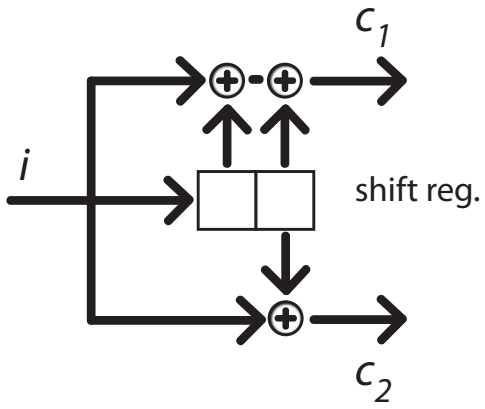

Figure A.2.: $(2, 1, [2])$ convolutional encoder

linearly in code sequence length. The Viterbi algorithm is favorable for compact implementations because it contains many small computations that can be easily serialized or parallelized depending on the constraints of the given application.

The convolutional encoder in Figure A.2 has four possible states and eight possible state transitions shown in Table A.1.

Memory State (i_{j-1}, i_{j-2})	Input Bit (i_j)	Output Bits $(c_{1_j} c_{2_j})$
00	0	00
00	1	11
01	0	11
01	1	00
10	0	10
10	1	01
11	0	01
11	1	10

Table A.1.: State transitions for a $(2, 1, [2])$ convolutional code

The Viterbi algorithm builds a history of the state transitions with the lowest number of errors from the input code sequence. For each pair of code sequence bits, the errors for all eight transitions are computed and added to the history. A trellis diagram visualizes all encoder states and possible transitions over time. Figure A.3 shows a simplified trellis diagram that only contains the state transitions and the number of errors. In each step, two input code sequence bits are evaluated to estimate the most likely internal state of the encoder, based on the previous inputs.

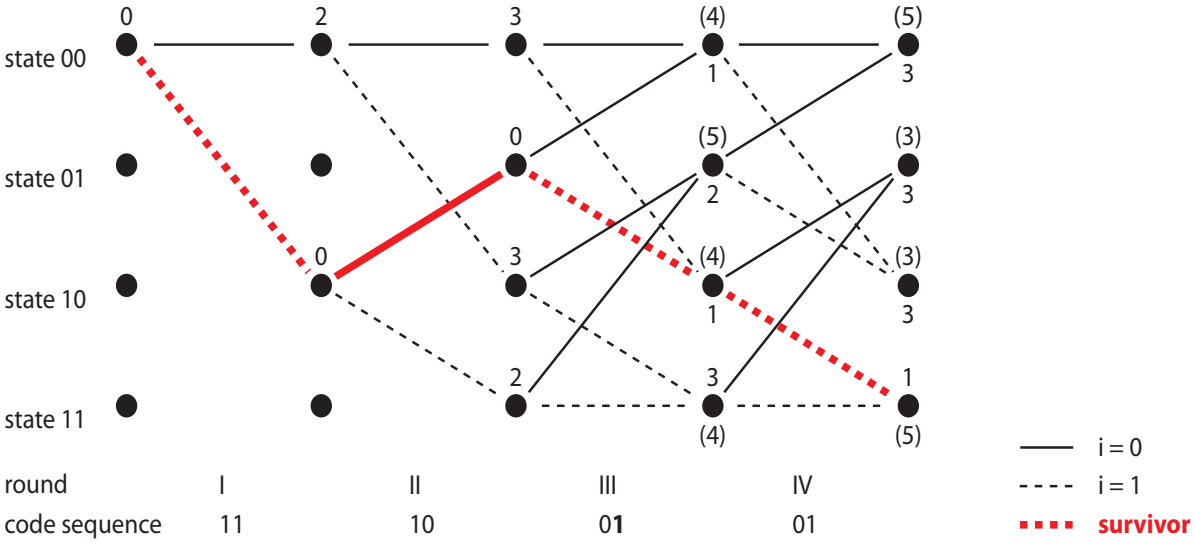

Figure A.3.: Trellis diagram for a $(2, 1, [2])$ convolutional code

In this example, the $(2, 1, [2])$ convolutional code has 4 possible memory states that are represented by dots in vertical direction. Every capital Roman number in horizontal direction represents one state transition, or decoding round, in time. The two digits below show the received code sequence bits. The path distance represents the minimum number of errors that lead to a state for the input sequence. By definition, the encoder is initialized with 00 and its path distance is set to 0 while the path distances of all other states are set to ∞.

For round I, the internal state of the encoder shown in Figure A.2 can transit to 10 if a 1 is shifted in (dotted line), or it remains in 00, if a 0 is shifted in (solid line). For both paths, the decoder computes hypothetical code sequence bits $\hat{c}_{1_1}\hat{c}_{2_1}$ based on the initial encoder state (00) and the hypothetical information bit. For both options, the Hamming distance to the actual input code sequence bits is stored as path distance. In this case 2 for 00 and 0 for 10 as shown in Figure A.3 above the black dots representing the states.

In round II, the four possible transitions from states 00 and 10 are evaluated accordingly. The $\hat{c}_{1_2}\hat{c}_{2_2}$ pairs are computed for all four paths. For every path, the Hamming distance to the two input code sequence bits is added to the path distance of the previous state. For example, one would expect code sequence bits 00 for a transition from state 00 to 00. The Hamming distance to the actual code-sequence bits is 1 which leads to an overall path distance of 3 for state 00.

The first decisions are made in round III, because now two paths merge in each state. The path distances are computed for both incoming paths and now, the decoder decides that the path with lower path distance contains less errors. Therefore it is more likely, that this path was transmitted. So, it is labeled as *survivor*. In this example, a transmission error occurred, which is marked in bold in the figure. The path distance is updated

to the path distance of the survivor and the other path, with the past distance marked with brackets in the figure, is discarded. Due to the error two states have distance 1 now.

The subsequent rounds are performed according to round III. It can be seen that the wrong path with distance 1 in the previous round has an increased distance of 3 due to the convolution, whereas the correct path still has distance 1. The bold red path with the lowest distance represents the most likely input sequence.

Figure A.3 shows that all survivors in round IV originate from the 01 state after round II. Therefore, all survivor paths contain the first two input bits as 1 and 0. They can be considered as stable and output as decoded result. The *Trace-Back-Depth* (TBD) determines the number of states that are stored before the output bits are returned. It has been shown empirically that, depending on the memory size of the encoder μ, a TBD value of 5μ is recommended [CC81].

For a hardware implementation, the path distances and paths have to be stored and updated.

Register exchange and trace back [CS93] are the two predominant approaches with several derivative and refined methods. Both have in common that the trellis is represented in the architecture similarly to the trellis diagram in Figure A.3. The TBD determines the number of rounds that are processed in the decoder. Both approaches store the predecessor for each state. The decoded bit is determined by choosing a random last state of the trellis and tracing it back to the unified survivor that is equal for all end states. The next round is computed by shifting the entire data by one step.

The approaches differ in the tracing mechanism. For register exchange, a fast but resource consuming tracing is carried out over concurrent logic with wires and multiplexers. The trace back method requires less resources but more time to translate through the trellis. Here, the trellis is searched sequentially which enables using RAM instead of registers. However, for high throughput implementations, additional effort has to be spent to parallelize the RAM update and trace back mechanism to some extent.

A.3. SRAM PUF Reliability Distribution

The SRAM PUF distribution discussed in [MTV09b] is determined by two parameters, λ_1 and λ_2. Let P_N be the probability distribution of a normal distribution and ${cdf_N}^{-1}$ the corresponding inverse cumulative distribution function.

Then the distribution of the expected values of the PUF is given by

$$P_X(x) = \frac{\lambda_1 P_N\left(\lambda_2 - \lambda_1 {cdf_N}^{-1}(x)\right)}{P_N({cdf_N}^{-1}(x)} \tag{A.6}$$

In the following, only bias-free PUFs are considered which corresponds to $\lambda_2 = 0$. Figure A.4 shows a sample probability distribution function with $\lambda_1 = 0.51$, resulting in $p_{puf} = 15\%$ that is used for example as input to evaluate the implementation candidates in Section 6.7.

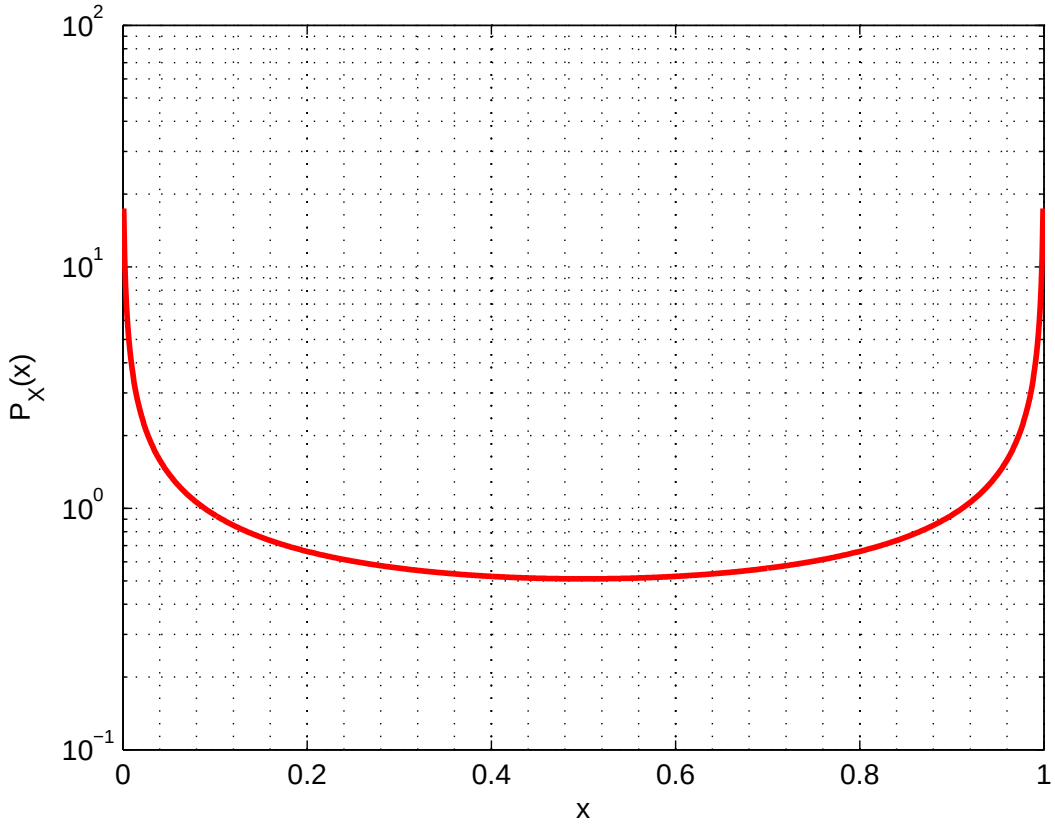

Figure A.4.: Probability distribution function of an SRAM PUF with $\lambda_1 = 0.51$, resulting in $p_{puf} = 15\%$

Sweeping over λ_1 results in the following mean error probabilities of the PUF p_{puf} provided in Table A.2.

λ_1	p_{puf}
0.16	5%
0.33	10%
0.36	11%
0.396	12%
0.433	13%
0.47	14%
0.51	15%
0.55	16%
0.592	17%
0.636	18%
0.68	19%
0.74	20%
1.01	25%

Table A.2.: Mean error probabilities of SRAM PUFs in dependency of λ_1

List of Pre-Publications

[DGV+15] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day Mandel Yu. Secure sketch metamorphosis: Tight unified bounds. *IACR eprint archive*, 2015.

[DGV+16] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day (Mandel) Yu. Efficient fuzzy extraction of PUF-induced secrets: Theory and applications. In Benedikt Gierlichs and Axel Poschmann, editors, *Conference on Cryptographic Hardware and Embedded Systems (CHES)*, volume 9813 of *LNCS*, pages 412–431. Springer Berlin / Heidelberg, 2016.

[HDSMS12] Matthias Hiller, Fabrizio De Santis, Dominik Merli, and Georg Sigl. Reliability bound and channel capacity of IBS-based fuzzy embedders. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 213–220. IEEE, 2012.

[HKS+15] Matthias Hiller, Ludwig Kürzinger, Georg Sigl, Sven Müelich, Sven Puchinger, and Martin Bossert. Low-area Reed decoding in a generalized concatenated code construction for PUFs. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2015.

[HOSB16] Matthias Hiller, Aysun Gurur Önalan, Georg Sigl, and Martin Bossert. Online reliability testing for PUF key derivation. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, pages 15–22. ACM, 2016.

[HPKS16] Matthias Hiller, Michael Pehl, Gerhard Kramer, and Georg Sigl. Algebraic security analysis of key generation with physical unclonable functions. In *Security Proofs for Embedded Systems Workshop (PROOFS)*, 2016.

[HPS15] Matthias Hiller, Michael Pehl, and Georg Sigl. Fehlerkorrekturverfahren zur sicheren Schlüsselerzeugung mit Physical Unclonable Functions. *Datenschutz und Datensicherheit (DuD)*, 39(4):229–233, 2015.

[HRLS14] Matthias Hiller, Leandro Rodrigues Lima, and Georg Sigl. Seesaw: An area-optimized FPGA Viterbi decoder for PUFs. In *Euromicro Conference on Digital System Design (DSD)*, pages 387–393. IEEE, 2014.

[HS14] Matthias Hiller and Georg Sigl. Increasing the efficiency of syndrome coding for PUFs with helper data compression. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. ACM/IEEE, 2014.

[HSP13] Matthias Hiller, Georg Sigl, and Michael Pehl. A new model for estimating bit error probabilities of ring-oscillator PUFs. In *International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2013.

[HWRL+13] Matthias Hiller, Michael Weiner, Leandro Rodrigues Lima, Maximilian Birkner, and Georg Sigl. Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, pages 43–54. ACM, 2013.

[HWS15] Matthias Hiller, Michael Weiner, and Georg Sigl. A method and an apparatus for deriving secret information from a series of response values and a method and an apparatus for providing helper data allowing to derive a secret information. *European Patent (EP 2773061B1, Assignee: Fraunhofer Gesellschaft)*, issued 2015.

[HYP15] Matthias Hiller, Meng-Day (Mandel) Yu, and Michael Pehl. Systematic low leakage coding for physical unclonable functions. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 155–166, 2015.

[HYS16] Matthias Hiller, Meng-Day (Mandel) Yu, and Georg Sigl. Cherry-picking reliable PUF bits with differential sequence coding. *IEEE Transactions on Information Forensics and Security*, 11(9):2065–2076, 2016.

[JRLH14] Bernhard Jungk, Leandro Rodrigues Lima, and Matthias Hiller. A systematic study of lightweight hash functions on FPGAs. In *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*. IEEE, 2014.

[KUM+15a] Stephan Kleber, Florian Unterstein, Matthias Matousek, Frank Kargl, Frank Slomka, and Matthias Hiller. Design of the secure execution PUF-based processor (SEPP). In *TRUDEVICE Workshop on Secure Hardware and Security Evaluation*, 2015.

[KUM+15b] Stephan Kleber, Florian Unterstein, Matthias Matousek, Frank Kargl, Frank Slomka, and Matthias Hiller. Secure execution architecture based on PUF-driven instruction level code encryption. *IACR eprint archive*, 2015.

[MPB+14] Sven Müelich, Sven Puchinger, Martin Bossert, Matthias Hiller, and Georg Sigl. Error correction for physical unclonable functions using generalized concatenated codes. In *International Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, 2014.

[PHG16] Michael Pehl, Matthias Hiller, and Helmut Graeb. Efficient evaluation of physical unclonable functions using entropy measures. *Journal of Circuits, Systems and Computers*, 25(1):1640001, 2016.

[PHS17] Michael Pehl, Matthias Hiller, and Georg Sigl. Error correction for physical unclonable functions. In Holger Boche, Ashish Khisti, H. Vincent Poor, and Rafael F. Schäfer, editors, *to appear at Information Theoretic Approaches to Security & Privacy*. Cambridge University Press, 2017.

[PMB+15] Sven Puchinger, Sven Müelich, Martin Bossert, Matthias Hiller, and Georg Sigl. On error correction for physical unclonable functions. In *International ITG Conference on Systems, Communications and Coding (SCC)*. IEEE, 2015.

[PRPHG14] Michael Pehl, Akshara Ranjit Punnakkal, Matthias Hiller, and Helmut Graeb. Advanced performance metrics for physical unclonable functions. In *International Symposium on Integrated Circuits (ISIC)*. IEEE, 2014.

[WHP14] Florian Wilde, Matthias Hiller, and Michael Pehl. Statistical security analysis of ring oscillator PUFs. In *International Symposium on Integrated Circuits (ISIC)*. IEEE, 2014.

[YHD15] Meng-Day (Mandel) Yu, Matthias Hiller, and Srinivas Devadas. Maximum likelihood decoding of device-specific multi-bit symbols for reliable key generation. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 38–43, 2015.

[YHD+16] Meng-Day (Mandel) Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight entity authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, 2016.

List of Supervised Theses

[Ali16] Syed Bilal Ali. *Software Implementation of Error Correction for PUFs.* Master's Thesis, Technical University of Munich, 2016.

[Bir13] Maximilian Birkner. *Implementation of a Lightweight Hash Function.* Bachelor's Thesis, Technical University of Munich, 2013.

[Bou16] Jason Bouroutis. *Implementation of a Hardware Abstraction Layer for an Industry 4.0 Model.* Bachelor's Thesis, Technical University of Munich, 2016.

[Fuc15] Tobias Fuchs. *Attack Schemes on DSC Helper Data.* Bachelor's Thesis, Technical University of Munich, 2015.

[Kai13] Rainer Kaiser. *Design and Implementation of a PUF + TRNG Circuit.* Bachelor's Thesis, Technical University of Munich, 2013.

[Kür14] Ludwig Kürzinger. *Analysis and Efficient Implementation of GC RM Error Correction Codes for PUFs.* Diploma Thesis, Technical University of Munich, 2014.

[Leo13] Justine Leow. *Security Analysis of PUF Data.* Bachelor's Thesis, Technical University of Munich, 2013.

[Ley15] Julian Leyh. *Lightweight BCH Decoder Architectures for PUF-Based Key Generation.* Bachelor's Thesis, Technical University of Munich, 2015.

[Nol15] Benjamin Nolet. *Differential Sequence Coding with multi-rate encoding for PUFs.* Master's Thesis, Technical University of Munich, 2015.

[Wan12] Xiaoqing Wan. *Design of a Hardware Efficient Fuzzy Embedder for PUFs on FPGAs.* Master's Thesis, Technical University of Munich, 2012.

Bibliography

[AC93] Rudolph Ahlswede and Imre Csiszar. Common randomness in information theory and cryptography - part I: Secret sharing. *IEEE Transactions on Information Theory*, 39(4):1121–1132, 1993.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[Alt15] Altera Corporation. Altera partners with Intrinsic-ID to develop worlds most secure high-end FPGA. *Press Release*, http://newsroom.altera.com/press-releases/nr-altera-intrinsic-id-security-stratix-10.htm, accessed 25.11.2015, 2015.

[AMS+09a] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In Mitsuru Matsui, editor, *Advances in Cryptology (ASIACRYPT)*, volume 5912 of *LNCS*, pages 685–702. Springer Berlin / Heidelberg, 2009.

[AMS+09b] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. PUF-PRFs: a new tamper-resilient cryptographic primitive. In Antoine Joux, editor, *Advances in Cryptology (EUROCRYPT)*, volume 5479 of *LNCS*, pages 96–102. Springer Berlin / Heidelberg, 2009.

[AMS+11] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Francois-Xavier Standaert, and Christian Wachsmann. A formal foundation for the security features of physical functions. In *IEEE Symposium on Security and Privacy (S&P)*, pages 397–412, 2011.

[BBCS92] Charles H Bennett, Gilles Brassard, Claude Crepeau, and Marie-Helene Skubiszewska. Practical quantum oblivious transfer. In Joan Feigenbaum, editor, *Advances in Cryptology (CRYPTO)*, volume 576 of *LNCS*, pages 351–366. Springer Berlin / Heidelberg, 1992.

[BBRM08] Matthieu Bloch, Joao Barros, Miguel R. D. Rodrigues, and Steven W. McLaughlin. Wireless information-theoretic security. *IEEE Transactions on Information Theory*, 54(6):2515–2534, 2008.

[BDH+10] Ileana Buhan, Jeroen Doumen, Pieter Hartel, Qian Tang, and Raymond Veldhuis. Embedding renewable cryptographic keys into noisy data. *International Journal of Information Security*, 9:193–208, 2010.

[BDHV07] Ileana Buhan, Jeroen Doumen, Pieter Hartel, and Raymond Veldhuis. Fuzzy extractors for continuous distributions. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, page 353, 2007.

[BDK+11] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, Franois-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In Phillip Rogaway, editor, *Advances in Cryptology (CRYPTO)*, volume 6841 of *LNCS*, pages 1–20. Springer Berlin / Heidelberg, 2011.

[BFKR14] Malte Brettel, Niklas Friederichsen, Michael Keller, and Marius Rosenberg. How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 perspective. *International Journal of Science, Engineering and Technology 8 (1), 37*, 44, 2014.

[BFSK11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *Advances in Cryptology (CRYPTO)*, volume 6841 of *LNCS*, pages 51–70. Springer Berlin / Heidelberg, 2011.

[BGS+08] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on FPGAs. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 5154 of *LNCS*, pages 181–197. Springer Berlin / Heidelberg, 2008.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christoph Paar, Axel Poschmann, Matt J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 4727 of *LNCS*, pages 450–466. Springer Berlin / Heidelberg, 2007.

[BKL+11] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. SPONGENT: A lightweight hash function. In Bart Preneel and Tsuyoshi Takagi, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 6917 of *LNCS*, pages 312–325. Springer, Heidelberg, 2011.

[BKL+13] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. SPONGENT: The design space of lightweight cryptographic hashing. *IEEE Transactions on Computers*, 62(10):2041–2053, 2013.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

[Bos99] Martin Bossert. *Channel Coding for Telecommunications*. John Wiley & Sons, New York, 1999.

[Bös08] Christoph Bösch. *Efficient Fuzzy Extractors for Reconfigurable Hardware*. Master's thesis, Ruhr-University Bochum, 2008.

[Boy04] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *ACM Conference on Computer and Communications Security (CCS)*, pages 82–91, 2004.

[Buh08] Ileana Buhan. *Cryptographic keys from noisy data, theory and applications*. Dissertation, University of Twente, 2008.

[Bun16a] Bundesamt für Sicherheit in der Informationstechnik. NXP Secure Smart Card Controller P6021y VB (BSI-DSZ-CC-0955, Rev. 0.93). *Security Target Lite*, 2016.

[Bun16b] Bundesamt für Sicherheit in der Informationstechnik. NXP Secure Smart Card Controller P6021y VB including IC Dedicated Software (BSI-DSZ-CC-0955-2016). *Certification Report*, 2016.

[BW13] Holger Boche and Rafael F. Wyrembelski. Secret key generation using compound sources - optimal key-rates and communication costs. In *International ITG Conference on Systems, Communications and Coding (SCC)*. IEEE, 2013.

[BWG15] Georg Tobias Becker, Alexander Wild, and Tim Gneysu. Security analysis of index-based syndrome coding for PUF-based key generation. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2015.

[Can01] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[CC81] George C. Clark and J. Bibb Cain. *Error Correction Coding for Digital Communications*. Plenum Press, New York, 1981.

[CCL+11] Qingqing Chen, György Csaba, Paolo Lugli, Ulf Schlichtmann, and Ulrich Rührmair. The bistable ring PUF: A new architecture for strong physical unclonable functions. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 134–141, 2011.

[CCL+12] Qingqing Chen, György Csaba, Paolo Lugli, Ulf Schlichtmann, and Ulrich Rührmair. Characterization of the bistable ring PUF. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1459–1462. ACM/IEEE, 2012.

[Che15] An Chen. Emerging nonvolatile memory (NVM) technologies. In *European Solid State Device Research Conference (ESSDERC)*, pages 109–113. IEEE, 2015.

[CJFJ07] David J. Costello Jr. and G. David Forney Jr. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95:1150–1177, 2007.

[CN00] Imre Csiszar and Prakash Narayan. Common randomness and secret key generation with a helper. *IEEE Transactions on Information Theory*, 46(2):344–366, 2000.

[CN08] Imre Csiszar and Prakash Narayan. Secrecy capacities for multiterminal channel models. *IEEE Transactions on Information Theory*, 54(6):2437–2452, 2008.

[Con84] Jean Conan. The weight spectra of some short low-rate convolutional codes. *IEEE Transactions on Communications*, 32(9):1050–1053, 1984.

[CS93] Robert Cypher and C. Bernard Shung. Generalized trace-back techniques for survivor memory management in the Viterbi algorithm. *Journal of VLSI signal processing systems for signal, image and video technology*, 5(1):85–94, 1993.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, New York, second edition, 2006.

[Dev] Srinivas Devadas. *Personal Website*, http://people.csail.mit.edu/devadas/, accessed 01.12.2015.

[DFM98] George I. Davida, Yair Frankel, and Brian J. Matt. On enabling secure applications through off-line biometric identification. In *IEEE Symposium on Security and Privacy (S&P)*, pages 148–157, 1998.

[DGSV15] Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. Helper data algorithms for PUF-based key generation: Overview and analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(6):889–902, 2015.

[DKK+12] Yevgeniy Dodis, Bhavana Kanukurthi, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. *IEEE Transactions on Information Theory*, 58(9):6207–6222, 2012.

[DKRS06] Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In Cynthia Dwork, editor, *Advances in Cryptology (CRYPTO)*, pages 232–250. Springer Berlin / Heidelberg, 2006.

[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 L. Camenisch, editors, *Advances in Cryptology (EUROCRYPT)*, volume 3027 of *LNCS*, pages 523–540. Springer Berlin / Heidelberg, 2004.

[DV14a] Jeroen Delvaux and Ingrid Verbauwhede. Attacking PUF-based pattern matching key generators via helper data manipulation. In Josh Benaloh, editor, *Topics in Cryptology (CT-RSA)*, volume 8366 of *LNCS*, pages 106–131. Springer International Publishing, 2014.

[DV14b] Jeroen Delvaux and Ingrid Verbauwhede. Key-recovery attacks on various RO PUF constructions via helper data manipulation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014.

[EDE04] Dalia A. El-Dib and Mohamed I. Elmasry. Modified register-exchange Viterbi decoder for low-power wireless communications. *IEEE Transactions on Circuits and Systems*, 51(2):371–378, 2004.

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

[Eli55] Peter Elias. Coding for noisy channels. *Proceedings of the Institute of Radio Engineers*, 43(3):356–356, 1955.

[FG93] Gennady Feygin and P. G. Gulak. Architectural tradeoffs for survivor sequence memory management in Viterbi decoders. *IEEE Transactions on Communications*, 41(3):425–429, 1993.

[FMR13] Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. Computational fuzzy extractors. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology (ASIACRYPT)*, volume 8269 of *LNCS*, pages 174–193. Springer Berlin / Heidelberg, 2013.

[Gas03] Blaise Gassend. *Physical Random Functions*. Master's thesis, 2003.

[GBS15] Andrea Grigorescu, Holger Boche, and Rafael F. Schaefer. Robust PUF based authentication. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2015.

[GCDD02] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *ACM Conference on Computer and Communications Security (CCS)*, pages 148–160, 2002.

[GI14] Onur Günlü and Onurcan Iscan. DCT based ring oscillator physical unclonable functions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8248–8251, 2014.

[GK16] Onur Günlü and Gerhard Kramer. Privacy, secrecy, and storage with noisy identifiers. Technical report, 2016.

[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, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 4727 of *LNCS*, pages 63–80. Springer Berlin / Heidelberg, 2007.

[Gol66] Solomon W. Golomb. Run-length encodings (corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, 1966.

[GPP11] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *Advances in Cryptology (CRYPTO)*, volume 6841 of *LNCS*, pages 222–239. Springer Berlin / Heidelberg, 2011.

[Gra07] Helmut E Graeb. *Analog design centering and sizing*. Springer, 2007.

[GVV75] Robert G. Gallager and David C. Van Voorhis. Optimal source codes for geometrically distributed integer alphabets (corresp.). *IEEE Transactions on Information Theory*, 21(2):228–230, 1975.

[HB10] Maximilian Hofer and Christoph Böhm. An alternative to error correction for SRAM-like PUFs. In Stefan Mangard and Franois-Xavier Standaert, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 6225 of *LNCS*, pages 335–350. Springer, Berlin / Heidelberg, 2010.

[HBF09] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, 2009.

[HHK+14] Kang Hyunho, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura. Cryptographic key generation from PUF data using efficient fuzzy extractors. In *International Conference on Advanced Communication Technology (ICACT)*, pages 23–26, 2014.

[Hil11] Matthias Hiller. *Optimized Fuzzy Extractor for PUFs on FPGAs*. Diplomarbeit, Ulm University, 2011.

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

[HNT+13] Clemens Helfmeier, Dmitry Nedospasov, Christopher Tarnovsky, Jan Starbug Krissler, Christian Boit, and Jean-Pierre Seifert. Breaking and entering through the silicon. In *ACM Conference on Computer & Communications Security (CCS)*, pages 733–744, 2516717, 2013.

[HRvD+16] Charles Herder, Ling Ren, Marten van Dijk, Meng-Day (Mandel) Yu, and Srinivas Devadas. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Transactions on Dependable and Secure Computing*, 2016.

[HSZS13] Maxim Hennig, Oliver Schimmel, Philipp Zieris, and Georg Sigl. Manipulationssensible kopierschutzfolie. In *D A CH Security*, 2013.

[HT07] Helena Handschuh and Elena Trichina. Securing flash technology. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 3–20. IEEE, 2007.

[Huf52] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

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

[IEE15] IEEE support. Personal communication, 2015.

[IHKS16] Vincent Immler, Maxim Hennig, Ludwig Kürzinger, and Georg Sigl. Practical aspects of quantization and tamper-sensitivity for physically obfuscated keys. In *Workshop on Cryptography and Security in Computing Systems (CS2)*, 2016.

[Int] Intrinsic-ID BV. *Company Website*, https://www.intrinsic-id.com/, accessed 25.11.2015.

[ISS+06] Tanya Ignatenko, Geert Jan Schrijen, Boris Skoric, Pim Tuyls, and Frans M. J. Willems. Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method. In *IEEE International Symposium on Information Theory (ISIT)*, pages 499–503, 2006.

[IW09] Tanya Ignatenko and Frans M. J. Willems. Biometric systems: Privacy and secrecy aspects. *IEEE Transactions on Information Forensics and Security*, 4(4):956–973, 2009.

[IW10] Tanya Ignatenko and Frans M. J. Willems. Information leakage in fuzzy commitment schemes. *IEEE Transactions on Information Forensics and Security*, 5(2):337–348, 2010.

[IW12] Tanya Ignatenko and Frans M. J. Willems. Biometric security from an information-theoretical perspective. *Foundations and Trends in Communications and Information Theory*, 7(2-3):135–316, 2012.

[JS02] Ari Juels and Madhu Sudan. A fuzzy vault scheme. In *IEEE International Symposium on Information Theory (ISIT)*, page 408. IEEE, 2002.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM Conference on Computer and Communications Security (CCS)*, pages 28–36, 1999.

[KGM+08] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, and Pim Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 67–70, 2008.

[Kil15] Kilopass Technology Inc. Comparison of embedded non-volatile memory technologies and their applications. *White Paper*, http://www.kilopass.com/download-white-paper-a-comparison-of-embedded-non-volatile-memory-technologies-and-their-applications/#wpcf7-f8864-p8865-o1, accessed 25.11.2015, 2015.

[KKR+12] Stefan Katzenbeisser, Unal 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, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 7428 of *LNCS*, pages 283–301. Springer Berlin / Heidelberg, 2012.

[KLC+16] Bohdan Karpinskyy, Yongki Lee, Yunhyeok Choi, Yongsoo Kim, Mijung Noh, and Sanghyun Lee. 8.7 physically unclonable function for secure key generation with a key error rate of 2E-38 in 45nm smart-card chips. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 158–160, 2016.

[KR08] Bhavana Kanukurthi and Leonid Reyzin. An improved robust fuzzy extractor. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *International Conference on Security and Cryptography for Networks (SCN)*, LNCS, pages 156–171. Springer Berlin / Heidelberg, 2008.

[Kra94] Hugo Krawczyk. LFSR-based hashing and authentication. In YvoG Desmedt, editor, *Advances in Cryptology (CRYPTO)*, volume 839 of *LNCS*, pages 129–139. Springer Berlin / Heidelberg, 1994.

[Kra07] Gerhard Kramer. Topics in multi-user information theory. *Foundations and Trends in Communications and Information Theory*, 4(4-5):265–444, 2007.

[KS10] Deniz Karakoyunlu and Berk Sunar. Differential template attacks on PUF enabled cryptographic devices. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2010.

[KwA07] Matthias Kamuf, Viktor wall, and John B. Anderson. Survivor path processing in Viterbi decoders using register exchange and traceforward. *IEEE Transactions on Circuits and Systems*, 54(6):537–541, 2007.

[Lim04] Daihyun Lim. *Extracting Keys from Integrated Circuits*. Master's thesis, Massachusetts Institute of Technology, 2004.

[LKPS09] Yingbin Liang, Gerhard Kramer, H. Vincent Poor, and Shlomo (Shitz) Shamai. Compound wiretap channels. *EURASIP Journal on Wireless Communications and Networking*, 2009:1–12, 2009.

[LLG+04] Jae W. Lee, Daihyun Lim, Blaise Gassend, Gookwon Edward Suh, Marten van Dijk, and Srinivas Devadas. A technique to build a secret key in

integrated circuits for identification and authentication applications. In *IEEE Symposium on VLSI Circuits (VLSIC)*, pages 176–179, 2004.

[LLG+05] Daihyun Lim, Jae W. Lee, Blaise Gassend, Gookwon Edward Suh, Marten van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.

[LT03] Jean-Paul Linnartz and Pim Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In Josef Kittler and Mark S Nixon, editors, *Internation Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA)*, volume 2688 of *LNCS*, pages 393–402. Springer Berlin Heidelberg, 2003.

[LTS07] Jean-Paul Linnartz, Pim Tuyls, and Boris Skoric. A communication-theoretical view on secret extraction. In Pim Tuyls, Boris Skoric, and Tom A.M. Kevenaar, editors, *Security with Noisy Data*, pages 57–77, London, 2007. Springer.

[Mae12] Roel Maes. *Physically Unclonable Functions: Constructions, Properties and Applications*. Dissertation, Katholieke Universiteit Leuven, 2012.

[Mae13] Roel Maes. An accurate probabilistic reliability model for silicon PUFs. In Guido Bertoni and Jean-Sbastien Coron, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, LNCS. Springer, Berlin / Heidelberg, 2013.

[Mau93] Ueli Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39:733–742, 1993.

[Mer14] Dominik Merli. *Attacking and Protecting Ring Oscillator Physical Unclonable Functions and Code-Offset Fuzzy Extractors*. Dissertation, Technical University of Munich, 2014.

[MGS13] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. In Peter Athanas, Dionisios Pnevmatikatos, and Nicolas Sklavos, editors, *Embedded Systems Design with FPGAs*, pages 245–267. Springer New York, 2013.

[MHH+13] Dominik Merli, Johann Heyszl, Benedikt Heinz, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of RO PUFs. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 19–24, 2013.

[Mic15] Microsemi Corporation. AC434: Using SRAM PUF system service in SmartFusion2 – Libero SoC v11.6. *Application Note*, http://www.microsemi.com/document-portal/doc_download/134545-ac434-using-sram-puf-system-service-in-smartfusion2-libero-soc-v11-6-application-note, accessed 25.11.2015, 2015.

[MRK+12] Mehrdad Majzoobi, Masoud Rostami, Farinaz Koushanfar, Dan S. Wallach, and Srinivas Devadas. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, pages 33–44. ACM, 2012.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*. North-Holland, 1977.

[MSA+14] Sanu K. Mathew, Sudhir K. Satpathy, Mark A. Anders, Himanshu Kaul, Steven K. Hsu, Amit Agarwal, Gregory K. Chen, Rachael J. Parker, Ram K. Krishnamurthy, and Vivek De. A 0.19pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100 In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 278–279, 2014.

[MSSS11] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-channel analysis of PUFs and fuzzy extractors. In Jonathan M. McCune, Boris Balacheff, Adrian Perrig, Ahmad-Reza Sadeghi, Angela Sasse, and Yolanta Beres, editors, *International Conference on Trust and Trustworthy Computing (TRUST)*, volume 6740 of *LNCS*, pages 33–47. Springer Berlin / Heidelberg, 2011.

[MTV08] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic PUFs from flip-flops on reconfigurable devices. In *Benelux workshop on information and system security (WISSec)*, 2008.

[MTV09a] 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, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 5747 of *LNCS*, pages 332–347. Springer Berlin / Heidelberg, 2009.

[MTV09b] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. A soft decision helper data algorithm for SRAM PUFs. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2101–2105, 2009.

[MV10] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 3–37. Springer Berlin / Heidelberg, 2010.

[MVHV12] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. PUFKY: A fully functional puf-based cryptographic key generator. In Emmanuel Prouff and Patrick Schaumont, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 7428 of *LNCS*, pages 302–319. Springer Berlin / Heidelberg, 2012.

[MW00] Ueli Maurer and Stefan Wolf. Information-theoretic key agreement: From weak to strong secrecy for free. In Bart Preneel, editor, *Advances in Cryptology (EUROCRYPT)*, volume 1807 of *LNCS*, pages 351–368. Springer Berlin Heidelberg, 2000.

[NHSB13] Dmitri Nedospasov, Clemens Helfmeier, Jean-Pierre Seifert, and Christian Boit. Invasive PUF analysis. In *Fault Diagnosis and Tolerance in Cryptography Workshop (FDTC)*. IEEE, 2013.

[NXP13] NXP Semiconductors N.V. NXP strengthens SmartMX2 security chips with PUF anti-cloning technology. *Press Release*, http://www.nxp.com/news/press-releases/2013/02/nxp-strengthens-smartmx2-security-chips-with-puf-anti-cloning-technology.html, accessed 25.11.2015, 2013.

[PM15] Rainer Plaga and Dominik Merli. A new definition and classification of physical unclonable functions. In *Workshop on Cryptography and Security in Computing Systems (CS2)*, pages 7–12, 2694807, 2015. ACM.

[Pos09] Axel Y. Poschmann. *Lightweight Cryptography - Cryptographic Engineering for a Pervasive World*. Dissertation, Ruhr-University Bochum, 2009.

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

[RB98] Arvind R. Raghavan and Carl W. Baum. A reliability output Viterbi algorithm with applications to hybrid ARQ. *IEEE Transactions on Information Theory*, 44(3):1214–1216, 1998.

[RDK11] Ulrich Rührmair, Srinivas Devadas, and Farinaz Koushanfar. Security based on physical unclonability and disorder. In Mohammad Tehranipoor and Cliff Wang, editors, *Introduction to Hardware Security and Trust*. Springer-Verlag New York Inc., 2011.

[RE10] Wolfgang Rankl and Wolfgang Effing. *Smart Card Handbook*. John Wiley & Sons, 2010.

[RSN+10] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. Special publication 800-22 revision 1a: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, National Institute of Standards and Technology, 2010.

[RSS+10] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jrgen Schmidhuber. Modeling attacks on physical unclonable functions. In *ACM Conference on Computer and Communications Security (CCS)*, pages 237–249, 2010.

[RvD13] Ulrich Rührmair and Marten van Dijk. On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols. *Journal of Cryptographic Engineering*, 3(1):17–28, 2013.

[Sch13] Heike Schröder. *Physically Uncloneable Functions in the Stand-Alone and Universally Composable Framework*. Dissertation, TU Darmstadt, 2013.

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

[SES09] Sherif W. Shaker, Salwa H. Elramly, and Khaled A. Sheriata. FPGA implementation of a reconfigurable Viterbi decoder for WiMAX receiver. In *International Conference on Microelectronics (ICM)*, pages 264–267, 2009.

[SH14] Dieter Schuster and Robert Hesselbarth. Evaluation of bistable ring PUFs using single layer neural networks. In Thorsten Holz and Sotiris Ioannidis, editors, *Trust and Trustworthy Computing (TRUST)*, volume 8564 of *LNCS*, pages 101–109. Springer International Publishing, 2014.

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell Systems Technical Journal*, pages 379–423 and 623–656, 1948.

[SIB14] SIBASE Zwischenbericht. *Project Report*, 2014.

[SKVdV09] Alex Stoinov, Tom Kevenaar, and Michiel Van der Veen. Security issues of biometric encryption. In *IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*, pages 34–39, 2009.

[Suh05] Gookwon Edward Suh. *AEGIS : A Single-Chip Secure Processor*. Dissertation, Massachusetts Institute of Technology, 2005.

[SvdSvdL12] Peter Simons, Erik van der Sluis, and Vincent van der Leest. Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 7–12, 2012.

[TAK+05] Pim Tuyls, Anton H. M. Akkermans, Tom A. M. Kevenaar, Geert-Jan Schrijen, Asker M. Bazen, and Raimond N. J. Veldhuis. Practical biometric authentication with template protection. In Takeo Kanade, Anil Jain, and NaliniK Ratha, editors, *Audio- and Video-Based Biometric Person Authentication (AVBPA)*, volume 3546 of *LNCS*, pages 436–446. Springer Berlin / Heidelberg, 2005.

[TBS15] Nima Tavangaran, Holger Boche, and Rafael F. Schaefer. Secret-key capacity of compound source models with one-way public communication. In *IEEE Information Theory Workshop (ITW)*, pages 252–256, 2015.

[TG04] Pim Tuyls and Jasper Goseling. Capacity and examples of template-protecting biometric authentication systems. In Davide Maltoni and AnilK Jain, editors, *Biometric Authentication International Workshop (BioAW)*, volume 3087 of *LNCS*, pages 158–170. Springer Berlin / Heidelberg, 2004.

[TLG+15] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. Laser fault attack on physically unclonable functions. In *Fault Diagnosis and Tolerance in Cryptography Workshop (FDTC)*. IEEE, 2015.

[TNS+14] Shahin Tajik, Dmitri Nedospasov, Jean-Pierre Seifert, Clemens Helfmeier, and Christian Boit. Emission analysis of hardware implementations. In *Euromicro Conference on Digital System Design (DSD)*, 2014.

[TSR+05] Russell Tessier, Sriram Swaminathan, Ramaswamy Ramaswamy, Dennis Goeckel, and Wayne Burleson. A reconfigurable, power-efficient adaptive Viterbi decoder. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(4):484–488, 2005.

[TSS+06] Pim Tuyls, Geert-Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, and Rob Wolters. Read-proof hardware from protective coatings. In Louis Goubin and Mitsuru Matsui, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 4249 of *LNCS*, pages 369–383. Springer Berlin Heidelberg, 2006.

[vdLPvdS12] Vincent van der Leest, Bart Preneel, and Erik van der Sluis. Soft decision error correction for compact memory-based PUFs using a single enrollment. In Emmanuel Prouff and Patrick Schaumont, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 7428 of *LNCS*, pages 268–282. Springer Berlin / Heidelberg, 2012.

[Ver] Verayo, Inc. *Company Website*, http://www.verayo.com/, accessed 25.11.2015.

[VHV12] Anthony Van Herrewege and Ingrid Verbauwhede. Tiny application-specific programmable processor for BCH decoding. In *IEEE International Symposium on System on Chip (SoC)*, pages 1–4, 2012.

[vHvdLS+13] Anthony van Herrewege, Vincent van der Leest, Andr Schaller, Stefan Katzenbeisser, and Ingrid Verbauwhede. Secure PRNG seeding on commercial off-the-shelf microcontrollers. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, pages 55–64. ACM, 2013.

[Vit67] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[WST95] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[Wyn75] Aaron D. Wyner. The wire-tap channel. *The Bell Systems Technical Journal*, 54(8):1355–1387, 1975.

[Xil11] Xilinx, Inc. Spartan-6 FPGA data sheet: DC and switching characteristics (DS162 v3.0), 2011.

[Xil15] Xilinx, Inc. Device reliability report UG116 (v10.3.1). *Technical Report*, 2015.

[YD10a] Meng-Day (Mandel) Yu and Srinivas Devadas. Recombination of physical unclonable functions. In *Government Microcircuit Applications and Critical Technology Conference (GOMACTech)*, pages 1–4, 2010.

[YD10b] Meng-Day (Mandel) Yu and Srinivas Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design & Test of Computers*, 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, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 6917 of *LNCS*, pages 358–373. Springer Berlin / Heidelberg, 2011.

[YMVD14] Meng-Day (Mandel) Yu, David M'Raihi, Ingrid Verbauwhede, and Srinivas Devadas. A noise bifurcation architecture for linear additive physical functions. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014.

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

[YQ13] Chi-En Yin and Gang Qu. Improving PUF security with regression-based distiller. In *ACM/IEEE Design Automation Conference (DAC)*, 2013.

[YSS+12] Meng-Day (Mandel) Yu, Richard Sowell, Alok Singh, David M'Raihi, and Srivinas Devadas. Performance metrics and empirical results of a PUF cryptographic key generation ASIC. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 108–115, 2012.

[YTA06] Gang Yao, Erdogan Ahmet T., and Tughrul Arslan. An efficient pre-traceback architecture for the Viterbi decoder targeting wireless communication applications. *IEEE Transactions on Circuits and Systems*, 53(9):1918–1927, 2006.

[ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[ZRJ14] Meng Zhang, Anand Raghunathan, and Niraj K. Jha. Trustworthiness of medical devices and body area networks. *Proceedings of the IEEE*, 102(8):1174–1188, 2014.

List of Figures

List of Tables

List of Symbols

$\mathbf{A}$	Algebraic Core
$b(\cdot)$	Binary representation
C/c	Codeword or code sequence
$\mathcal{C}$	Code
C_{hd}	Helper data capacity
C_{key}	Key capacity
$cdf(\cdot)$	Cumulative distribution function
d	Minimum distance of a code
$dim(\cdot)$	Dimensions of a vector or matrix
Δ	Rank loss
$E(\cdot)$	Expectation
e_1	Probability that $error\_1$ occurs
e_2	Probability that $error\_2$ occurs
$f(\cdot)$	Hash function
$g(\cdot)$	Compression function
$g_{dim}(\cdot,\cdot)$	Dimensional rank loss
$\gamma(\cdot)$	Decoding operation of an ECC
$\mathbf{G}$	Generator matrix
$H(\cdot)$	Shannon entropy
$\mathbf{H}$	Parity check matrix
$I(\cdot;\cdot)$	Mutual information
$\mathbf{I}$	Identity matrix
$\mathcal{I}$	Index set
k	Code size
k_{in}	Input size of the random number of the Algebraic Core
k_{out}	Output size of the secret of the Algebraic Core
l_{in}	Input size of PUF response of the Algebraic Core
l_{out}	Output size of helper data of the Algebraic Core
K/k	Key from generation
κ	Key size
L/l	Key from reproduction
$\mathcal{L}$	Finite-length alphabet for RLE
Λ/λ	Reliability indicator variable
m	Parameter of run-length encoding
$\mathbf{M}_{pre}$	Preprocessing Matrix

$\mathbf{M}_{post}$	Postprocessing Matrix
$\max(\cdot,\cdot)$	Maximum operator
$\min(\cdot,\cdot)$	Minimum operator
$\mu(\cdot)$	Mean value
μ	Memory length of convolutional code
n	Block length, or code length
$N(\cdot\|\cdot)$	Numbers of occurrence of a letter in a sequence
p	Probability of indexing a PUF response bit
$P(\cdot)$	Probability distribution
$\mathbf{P}$	Parity part of the generator matrix of an ECC with systematic encoding
p_{err}	Output bit error probability of the key
p_{max}	Maximum input error probability of indexed PUF response bits
p_{puf}	Output error probability of a PUF response bit
p_{syn}	Average bit error probability at output of syndrome decoder
$p_{syn-max}$	Upper bound of output bit error probability of syndrome decoder
$\Pr[\cdot]$	Probability
$\varphi(\cdot)$	Encoding operation of an ECC
$q(\cdot)$	RLE compressed version of an integer number
R/r	Input random number
R_{hd}	Achievable helper data rate
R_{key}	Achievable key rate
$rank(\cdot)$	Rank of a matrix
S/s	Output secret
$\sigma(\cdot)$	Standard deviation
$\sup(\cdot,\cdot)$	Supremum operator
T/t	Internal state of a compound source
$\mathcal{T}_{\epsilon}^{n}$	Letter typical set
$\mathfrak{T}$	Channel between X and Y
U/u	Differential distance pointer
V/v	Inversion bit
W/u	Helper data
$wt(\cdot)$	Hamming weight operator
X/x	PUF response during generation
Y/y	PUF response during reproduction
ζ	Yield of DSC helper data generation failure
$\mathbf{0}$	All zeros matrix
$\|\cdot\|$	Cardinality operator

Index