KU LEUVEN

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Security Analysis of PUF-Based Key Generation and Entity Authentication

Jeroen Delvaux

Supervisors:
Prof. dr. ir. Ingrid Verbauwhede
Prof. dr. Dawu Gu
(Shanghai Jiao Tong University, China)

Dissertation presented in partial
fulfillment of the requirements for the
degree of Doctor of Engineering
Science (PhD): Electrical Engineering

June 2017

Security Analysis of PUF-Based Key Generation and Entity Authentication

Jeroen DELVAUX

Examination committee:
Prof. dr. ir. Jean Berlamont, chair
Prof. dr. ir. Ingrid Verbauwhede, supervisor
Prof. dr. Dawu Gu, supervisor
(Shanghai Jiao Tong University, China)
Prof. dr. ir. Wim Dehaene
Dr. Benedikt Gierlichs
Prof. dr. Xuejia Lai
(Shanghai Jiao Tong University, China)
Prof. dr. Shengli Liu
(Shanghai Jiao Tong University, China)
Prof. dr. Kazuo Sakiyama
(University of Electro-Communications, Japan)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2017

© 2017 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Jeroen Delvaux, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgements

I thank my promotor Prof. Ingrid Verbauwhede for welcoming me to the research group *Computer Security and Industrial Cryptography* (COSIC) of KU Leuven in October 2012. I thank my promotor Prof. Dawu Gu for welcoming me to the *Lab of Cryptology and Computer Security* (LoCCS) of *Shanghai Jiao Tong University* (SJTU) in October 2013.

I thank Sandrine Debecker for coordinating my application for admission to SJTU. I thank Leen Cuypers, Prof. Yixin Ma, and all others who were involved in establishing an agreement between KU Leuven and SJTU on granting double doctoral degrees.

I thank *Flanders Innovation & Entrepreneurship* (VLAIO), or formerly, the Flemish *Agency for Innovation by Science and Technology* (IWT), for funding my research during 48 months through a doctoral grant.

I thank my coauthors Dr. Roel Peeters and Dr. Dries Schellekens for giving expert advice on cryptographic protocols. I thank my coauthors Dr. Matthias Hiller and Mandel Yu for having numerous discussions on PUFs and for proofreading parts of this thesis. I thank Dr. Anthony Van Herrewege and Dr. Vladimir Rožić for elucidating the respective peripherals and internals of the UNIQUE ASIC. I thank Dr. Steven Delvaux for pointing out that the multiple integral in (3.17) can be bounded by an inscribed hypersphere.

I thank jury members Prof. Wim Dehaene, Dr. Benedikt Gierlichs, Prof. Xuejia Lai, Prof. Shengli Liu, and Prof. Kazuo Sakiyama for asking questions about and giving comments on my PhD thesis. I thank Prof. Bart Preneel, Prof. Hui Li, and Prof. Qiuliang Xu for reviewing my thesis as part of an SJTU procedure. I thank Prof. Jean Berlamont for serving as the chair of my public PhD defense in Leuven.

I thank Jun Ma and Shuyi Li for forwarding numerous procedures that the SJTU doctoral program requires. I thank Di Yan and Prof. Zhiqiang Liu for translating numerous forms from Chinese to English. I thank the latter also for serving as the secretary of my preliminary PhD defense in Shanghai, and for arranging most of its practicalities. I thank Prof. Lei Wang, Dr. Zheng Guo, and all others who occasionally helped me with formalities.

I thank Dr. Weijia Wang for welcoming Dr. Gierlichs at Pudong airport prior to my preliminary PhD defense. I thank Dr. Sen Xu for welcoming Prof. Verbauwhede at Hongqiao airport, and for helping me find a first apartment nearby the Minhang campus of SJTU. I thank Wei Sun for helping me register that same apartment at the local police station. I thank Dr. Shi-Feng Sun for helping me find a first bicycle, and for translating the abstract of my submission to ChinaCrypt 2014 from English to Chinese. I thank Bohan Yang for ordering the Chinese snacks that complement the reception after my public defense. I thank Sujoy Sinha Roy for letting me borrow his suit prior to my public defense.

I thank Wim Devroye for handling travel requests and booking flights. I thank secretary Péla Noë for handling numerous practicalities. I thank Elsy Vermoesen for handling various financial matters.

Jeroen Delvaux

June 2017

Abstract

No two physical objects are exactly the same, even when manufactured with a nominally identical process. For example, two sheets of paper that are indistinguishable with the naked eye, still differ considerably in their nanoscale fiber structures. Although manufacturing variability is usually undesired, the associated ability to uniquely identify a physical object, which is constrained to an *integrated circuit* (IC) in this thesis, can be leveraged for security purposes. To facilitate the registration of unique features, a so-called *physically unclonable function* (PUF) can be implemented on the IC. A PUF circuit is designed to be sensitive to process variations, i.e., challenged with a binary input, it provides a binary, device-unique response. This building block can hence be understood as the silicon equivalent of human biometrics.

PUFs can augment the security architecture of an ever-increasing number of electronic devices that access our personal data and/or represent our identities. This includes but is not limited to smartphones, credit cards, access badges, the sensors and actuators of automated home, and medical implants. PUFs usually need to team up with other building blocks, e.g., *true random number generators* (TRNGs), cryptographic algorithms, error-correcting codes, *non-volatile memory* (NVM), etc. We analyze the security of such multi-component systems in a format that allows for comparisons among proposals that have similar or identical objectives. Numerous newly revealed flaws and attacks are presented throughout this thesis. On the bright side, the lessons learned can help improve the quality of future PUF-based systems.

Samenvatting (Dutch)

Twee fysische objecten zijn nooit exact hetzelfde, zelfs wanneer ze met een identiek proces gefabriceerd worden. Bijvoorbeeld, twee vellen papier die men met het blote oog niet kan onderscheiden, verschillen desondanks in hun vezelstructuur op nanoschaal. Alhoewel zulke procesvariaties normaal gezien ongewenst zijn, de mogelijkheid om als dusdanig een fysisch object uniek te identificeren, kan gebruikt worden voor veiligheidsdoeleinden. In deze thesis focussen we op de beveiliging van een geïntegreerde schakeling. Om de registratie van unieke kenmerken te vergemakkelijken, kan een zogenaamde *fysisch onkloonbare functie*, of kortweg een PUF in het Engels, op de chip geïmplementeerd worden. Een PUF schakeling is ontworpen om gevoelig te zijn aan procesvariaties, i.e., gegeven een binaire invoer, produceert het een binaire, chip-specifieke uitvoer. Deze bouwblok kan dus begrepen worden als het silicium equivalent van menselijke biometrie.

PUFs kunnen de veiligheidsarchitectuur van een steeds toenemend aantal elektronische apparaten die toegang hebben tot onze persoonlijke gegevens en/of onze identiteiten representeren versterken. Dit omvat maar is niet beperkt tot smartphones, kredietkaarten, toegangsbadges, de sensoren en actuatoren van een geautomatiseerd huis, en medische implantaten. PUFs moeten gewoonlijk samenspelen met andere bouwblokken, e.g., toevalsgenerators, cryptografische algoritmes, foutcorrigerende codes, niet-volatiel geheugen, enzovoort. We analyseren de veiligheid van zulke systemen in een formaat waarmee voorstellen die gelijkaardige of identieke doeleinden hebben vergeleken kunnen worden. Talrijke nieuw ontdekte zwakheden en aanvallen worden gepresenteerd in deze thesis. Aan de positieve kant, de geleerde lessen kunnen helpen om de veiligheid van toekomstige PUF-gebaseerde systemen te verbeteren.

List of Acronyms

ACM	Association for Computing Machinery
AES	Advanced Encryption Standard
ASIC	application-specific integrated circuit
BCH	Bose, Ray-Chaudhuri, Hocquenghem
CDF	cumulative distribution function
CMOS	complementary metal–oxide–semiconductor
COSIC	Computer Security and Industrial Cryptography
CRP	challenge–response pair
DoS	denial-of-service
EEPROM	electrically erasable programmable read-only memory
FIB	focused ion beam
FPGA	field-programmable gate array
HDA	helper data algorithm
HMAC	keyed-hash message authentication code
i.i.d.	independent and identically distributed
IBS	index-based syndrome
IC	integrated circuit
IEEE	Institute of Electrical and Electronics Engineers
KDF	key derivation function

laser light amplification by stimulated emission of radiation

LFSR linear-feedback shift register

LHL leftover hash lemma

LoCCS Lab of Cryptology and Computer Security

MAC message authentication code

MTP multiple-time programmable

NIST National Institute of Standards and Technology

NVM non-volatile memory

OTP one-time programmable

PDF probability density function

PhD Doctor of Philosophy

PMF probability mass function

PRF pseudorandom function family

PRNG pseudorandom number generator

PUF physically unclonable function

RFID radio-frequency identification

RO ring oscillator

ROM read-only memory

RRAM resistive random-access memory

RSA Rivest, Shamir, Adleman

SDML soft-decision maximum-likelihood

SHIC super high information content

SJTU Shanghai Jiao Tong University

SMV spatial majority vote

SRAM static random-access memory

TMV temporal majority vote

TRNG true random number generator

XOR exclusive OR

List of Symbols

$\mathbf{0}$	All-zeros vector
$\mathbf{1}$	All-ones vector
$\mathbf{a}$	Output of a cryptographic algorithm
α	Lower bound
$\mathbf{b}$	Output of a cryptographic algorithm
β	Upper bound
$\mathbf{c}$	Challenge of a PUF
d	Minimum distance of a code
$\boldsymbol{\delta}$	Variability model of a strong PUF
$\mathbf{e}$	Error pattern
ϵ	Constant close to zero
η	Constant; length of a binary vector
f	Frequency of oscillation
$\mathbf{G}$	Generator matrix of a linear code
g	Number of CRPs
$\boldsymbol{\gamma}$	Challenge $\mathbf{c}$ after invertible transformation
$\mathbf{H}$	Parity-check matrix of a linear code
$\mathbf{h}$	Helper data
$\mathbf{h}_{\perp}$	Collective helper data of all previous steps

$\mathbf{i}$	Identifier
i	Index
$\mathbf{I}_q$	Identity matrix of size $q \times q$
j	Index
$\mathbf{k}$	Symmetric key
k	Message length of a block code
$\mathbf{k}_{\mathrm{app}}$	Keying material of a cryptographic application
$\mathbf{k}_{\mathrm{priv}}$	Private key
$\mathbf{k}_{\mathrm{pub}}$	Public key
κ	Size of a symmetric key $\mathbf{k}$
λ	Constant; length of a binary vector
$\mathbf{m}$	Message
m	Constant that relates to the size of a PUF
μ	Mean
$\mathbf{n}$	Nonce; seed
n	Length of a codeword $\mathbf{w}$
ω	Aggregate of variability contributions
$\mathbf{P}$	Rightmost part of a generator matrix $\mathbf{G}$ in standard form
p_{bias}	Parameter of a biased distribution
p_{corr}	Parameter of a spatially correlated distribution
p_{error}	Error rate of a response bit
p_{fail}	Failure rate of a PUF-based system
ϕ	Angle in a hyperspherical coordinate system
q	Constant
$\mathbf{r}$	Response of a PUF
ρ	Radial distance in a hyperspherical coordinate system

$\mathbf{s}$	State
σ	Standard deviation
$\mathbf{\Sigma}$	Covariance matrix
T	Temperature
t	Minimum number of errors that can be corrected
U	Uniformly distributed random variable
v	Instance of a PUF-enabled device
$\boldsymbol{\varphi}$	Syndrome of a possibly corrupted codeword $\tilde{\mathbf{w}}$
V_S	Supply voltage
$\mathbf{w}$	Codeword
$\mathbf{x}$	Concatenation of PUF responses $\mathbf{r}$
$\mathbf{y}$	Concatenated response $\mathbf{x}$ after processing
$\mathbf{z}$	Concatenated response $\mathbf{x}$ after processing
ζ	Error-correcting code

Contents

Abstract iii

List of Acronyms vii

List of Symbols ix

Contents xiii

1 Introduction 1

1.1 Securing Electronic Devices 1

1.2 Thesis Outline and Contributions 4

2 Preliminaries 7

2.1 Notation 7

2.2 Min-Entropy 8

2.3 Probability Distributions 9

2.4 Coding Theory 10

2.5 Embedded, Non-Volatile Memory 12

2.6 Cryptographic Algorithms 13

2.7 Physically Unclonable Functions 15

2.7.1 Informal 15

2.7.2 Formal 17

3 Security Analysis of PUF Circuits — Novel Methods 19

3.1 A Representative Set of PUF Circuits 20

3.1.1 Two Design Flavors: Weak and Strong PUFs 21

3.1.2 Selected Weak PUFs 22

3.1.3 Selected Strong PUFs 23

3.1.4 Machine Learning 28

3.1.5 An Accurate Variability–Noise Model 32

3.2 Min-Entropy 34

3.2.1 Overview 34

3.2.2 Black-Box Bounds 39

3.2.3 Gray-Box Bounds 41

3.2.4 White-Box Bounds 45

3.3 Physical Attacks 53

3.3.1 Related Literature 53

3.3.2 Exploitation of Noise 55

3.4 Conclusion 62

4 A Survey on PUF-Based Key Generation 65

4.1 System Overview 67

4.1.1 Reference: Key-Storage in Physically Secure NVM . . . 67

4.1.2 An HDA for PUF-Based Key Generation 68

4.1.3 Failure Rate 71

4.1.4 Min-Entropy Loss 72

4.2 Cryptographic Back-End 74

4.2.1 Encryption of an Application Key 74

4.2.2 Integrity of Helper Data 75

4.2.3 Key Derivation Function 79
4.3 Error-Correction with a Secure Sketch 80
4.3.1 Definition 80
4.3.2 Constructions using a Block Code 82
4.3.3 Repeated Execution of a Concatenated Code 87
4.3.4 Convolutional Codes 88
4.3.5 Tight Bounds on the Min-Entropy Loss 88
4.3.6 Reusability 102
4.3.7 Helper Data Manipulation 104
4.4 Other Error-Correction Schemes 106
4.4.1 Exhaustive Search 106
4.4.2 Soft-Decision Decoding 106
4.4.3 Pattern Matching 110
4.5 Computationally Straightforward Front-End 114
4.5.1 Temporal Majority Vote 115
4.5.2 Suh and Devadas 116
4.5.3 Spatial Majority Vote 120
4.5.4 Index-Based Syndrome 123
4.5.5 Von Neumann 125
4.5.6 Comparisons 128
4.6 PUFs with Helper Data Input 129
4.7 Conclusion 131

5 A Survey on PUF-Based Entity Authentication 133
5.1 Protocol Requirements 134
5.1.1 Complete Specification 134
5.1.2 Resilience against Leakage of NVM Contents 135

5.1.3 Able to Handle Noise 135
5.1.4 Resistance to Machine Learning Attacks 135
5.1.5 Expanding the Response of a Strong PUF 136
5.1.6 Low-Cost and Resource-Constrained 136
5.1.7 Easy-to-instantiate 136
5.1.8 Resistance to Protocol Attacks 137
5.1.9 Scalability: Identification Prior to Authentication 137
5.1.10 On the Mutual Authentication Order 138
5.2 Authentication via PUF-Based Key Generation 138
5.2.1 Reference Protocols 139
5.2.2 Related Literature 142
5.3 Analysis of Protocols 142
5.3.1 Basic Authentication (2001) 142
5.3.2 Controlled PUFs (2002) 147
5.3.3 Bolotnyy and Robins (2008) 149
5.3.4 Öztürk et al. (2008) 151
5.3.5 Hammouri et al. (2008) 154
5.3.6 Kulseng et al. (2010) 157
5.3.7 Super High Information Content PUFs (2010) 159
5.3.8 Sadeghi et al. (2010) 161
5.3.9 Logically Reconfigurable PUFs (2011) 162
5.3.10 Reverse Fuzzy Extractors (2012) 165
5.3.11 Converse Authentication (2012) 170
5.3.12 Lee et al. I (2012) 174
5.3.13 Jin et al. (2012) 176
5.3.14 Slender PUFs (2012) 179
5.3.15 Xu and He (2012) 181

5.3.16 He and Zou (2012) 183
5.3.17 Jung and Jung (2013) 186
5.3.18 Lee et al. II (2013) 188
5.3.19 Noise Bifurcation (2014) 191
5.3.20 System of PUFs (2014) 193
5.3.21 PUF Lockdown (2016) 196
5.4 Overview and Discussion 200
5.5 Conclusion 202

6 Conclusions and Further Work 205

6.1 PUFs versus Embedded NVM 205
6.2 Min-Entropy 206
6.3 Literature Disconnect 207

Bibliography 209

Curriculum Vitae 229

List of Publications 231

Chapter 1

Introduction

Nowadays, the term *physically unclonable function* (PUF) usually refers to an embedded electrical circuit that is designed to be prone to manufacturing variations while remaining fairly resistant to noise. From a black-box perspective, a binary input, i.e., the *challenge*, is mapped to a binary, device-specific, and slightly noisy output, i.e., the *response*. Subsequent to the manufacturing of the first silicon PUFs by Lofstrom et al. [107] and Gassend et al. [51] in the early 2000s, research efforts have vastly increased over time. Moreover, PUF-enabled products became commercially available, as exemplified by companies such as Intrinsic-ID [72] and Verayo [173]. Applications of PUFs mostly relate to the security architecture of a low-cost, resource-constrained electronic device.

The remainder of this chapter is organized as follows. Section 1.1 provides a broad introduction to the security architecture of a lightweight electronic device, and highlights the role of PUFs in this regard. Section 1.2 outlines the contributions of this PhD thesis to the analysis of presumably secure PUF-based systems, while briefly summarizing the contents of each remaining chapter.

1.1 Securing Electronic Devices

We are increasingly surrounded by miniaturized devices that collect, store, and process our personal data, thereby possibly representing our identities. Many of us carry around smartphones, tablet computers, electronic identity cards, access badges, and smart cards for public transit. Even in closer proximity to our bodies, wearable and implantable medical devices can monitor physiological signals and administer treatment accordingly. Consider for example insulin pumps that treat diabetes patients and *implantable cardioverter-defibrillators* that correct cardiac arrhythmias.

Still in our immediate vicinity, a *smart home* is equipped with a plethora of automated systems. Consider for example motion-activated light switches and a refrigerator that keeps track of its contents and corresponding expiration dates. A similar trend holds for corporate and public infrastructure. *Radio-frequency identification* (RFID) tags can facilitate the *supply chain management* of a company. Unlike a traditional barcode, tags are not required to be within the line of sight of the reader. Outdoors, spatially distributed sensors can monitor various environmental parameters in real-time. This includes air pollution in congested cities as well as radiation levels around nuclear power plants [25]. Moreover, forest fires can be detected in an early stage by measuring temperature, humidity, carbon dioxide concentrations, etc.

Many of the aforementioned electronic devices are interconnected through a wireless local network. Depending on the context, this can be referred to as a *personal area network*, a *body area network*, a *wireless sensor network*, etc. Communication standards with a range of about 1 m to 100 m usually rely on electromagnetic waves, e.g., Wi-Fi and Bluetooth. Standards with a range of several centimeters, e.g., *near-field communication*, usually rely on electromagnetic induction between two loop antennas. Individual devices and local networks as a whole may in turn be connected to the Internet, which ultimately results in the *Internet of things* [5]. Unfortunately, the networked nature inherently expands the attack surface. Cryptographic protocols are indispensable for securing the many information streams against eavesdropping and manipulation, and can also prevent identity theft.

The objectives of a cryptographic protocol can typically be broken down into fundamental security needs, as illustrated in a threefold manner. First, there is the *authentication of data*. Consider for example a tablet-enabled online banking application where any malicious modification to an otherwise genuine transaction should be detected. A second security goal, which is usually required in addition to the first, is the *confidentiality of data*. Consider for example smartphone users who might want to keep the contents of their text, audio, and video conversations secret from the outside world. A third security need is *entity authentication*. A device, and by extension also its owner, then prove their identity to another party. Consider for example an electronic access badge that is used to enter a private facility.

Cryptographic protocols achieve their goals through a set of algorithmic building blocks. This can include the use of block ciphers, stream ciphers, and cryptographic hash functions, for example. Protocol and algorithm specifications are usually considered to be public knowledge. According to Kerckhoffs' principle [82], a cryptosystem should be secure even if the attacker knows everything about the system, except the key. The latter refers to a secret binary string that is usually stored in the embedded, *non-volatile memory* (NVM) of a device, i.e., contents are preserved during the power-off state. Despite a history of broken systems, plenty of algorithmic solutions have been analyzed extensively over time by a large, international community of cryptologists and remain unbroken to date.

Unfortunately, a secure algorithm does not necessarily result in a secure implementation. Due to the mobile, distributed nature of modern electronic devices, an attacker might easily obtain physical access to the *integrated circuit* (IC). Measuring physical characteristics while the IC performs cryptographic operations could hence leak information about the key. Even with relatively affordable equipment, power consumption [86], electromagnetic radiation [45, 134], and other so-called *side-channels* can be recorded and digitized into a time series. Statistical analysis of a certain number of *traces* could suffice to recover the key. As an either alternative or complementary threat, *fault attacks* [9] actively disrupt the nominal computational behavior of the IC via physical channels in order to extract information about the key.

Regardless of the cryptographic algorithms, a more direct approach is to physically attack the NVM that stores the key. Memories usually consist of an array of identically designed cells that each store a single bit. Unauthorized reading and writing of cell contents via, e.g., optical channels or a *focused ion beam* (FIB), is feasible [90, 69, 59, 158] after having decapsulated the IC, although potentially tedious for large data sizes. The nearby *bus*, i.e., a small set of parallel, conductive wires, that connects the NVM with other components provides an alternative target. A *microprobe*, i.e., a fine-tipped needle, allows an attacker to access the bus and its serialized data streams through a single location [3, 90].

Unfortunately, perfect resistance to physical attacks does not exist. The capabilities of a FIB workstation are quite impressive, for example. This highly expensive tool allows for nanoscale modifications to the circuit, e.g., the removal and deposition of conductive material so as to cut and create electrical connections respectively. System providers usually aim to foresee a device with sufficient countermeasures in order to resist attackers with a more limited budget in terms of time and money. Unfortunately, these countermeasures impose a considerable overhead on the implementation. In the ideal case, physical attacks are rendered economically infeasible, i.e., the benefits of having performed a successful attack do not outweigh the required resources.

As an alternative to key-storage in NVM, the use of a PUF can be considered. The latter building block can greatly benefit from a custom-designed circuit and layout, which complies with an *application-specific integrated circuit* (ASIC). Nevertheless, several PUF designs circumvent the restrictions of a *field-programmable gate array* (FPGA) on the selection, placement, and routing of circuit elements reasonably well. Most frequently, the device-unique but noisy and non-uniformly distributed responses of a PUF are post-processed into a stable, uniformly distributed secret key. This approach hinges on the assumption that the PUF and its post-processing logic offer a higher resistance to physical attacks than NVM, especially during the power-off state. There also exist more lightweight proposals that avoid the detour of generating a secret key and directly leverage the device-unique functional behavior of the PUF instead. Numerous authentication protocols that adopt the latter approach have been published, for example.

1.2 Thesis Outline and Contributions

The remainder of this PhD thesis consists of five chapters. Their contents and technical contributions are as follows:

- **Chapter 2: Preliminaries**. In addition to introducing the notation, sufficient technical background is provided to make this thesis self-contained. Basic concepts from statistics, coding theory, cryptology, and other disciplines are reviewed.
- **Chapter 3: Security Analysis of PUF Circuits — Novel Methods**. We investigate PUFs as an isolated building block and develop new methods for assessing their security. A first, theoretical contribution proves that the functional behavior of several PUFs is not as uniquely tied to a given device as is typically assumed. To be precise, we derive tight upper bounds on the *min-entropy* of their responses, and these evaluate to surprisingly low numbers. This fundamentally constrains the usability of the assessed PUFs for applications such as key generation. A second, experimental contribution showcases the vulnerability of several PUFs to a new type of physical attack. Noise serves as an inherent side-channel, and its enhancement through environmental changes could result in a more efficient fault attack.
- **Chapter 4: A Survey on PUF-Based Key Generation**. We analyze the algorithmic security of numerous methods that aim to post-process the responses of a PUF into a stable, uniformly distributed secret key.

Apart from revealing unanticipated threats, it turns out that the security of most error-correction methods can only be proven for ideal PUFs. To be precise, it is assumed that responses are uniformly distributed, or slightly more general, that response bits are independent and identically distributed. Such assumptions are hard to meet in practice. Better but still problematic, several constructions based on coding theory can handle all distributions that have a relatively high min-entropy. We derive improved bounds on the min-entropy loss of the latter constructions so as to handle a large variety of distributions more efficiently.

- **Chapter 5: A Survey on PUF-Based Entity Authentication**. We analyze the security and usability of 21 PUF-based authentication protocols. In order to treat all proposals equally, we rigorously apply a newly developed framework of custom-tailored protocol requirements. Apart from revealing numerous unanticipated threats, it turns out that many proposals again rely on ideal PUFs. To be precise, it is often assumed that responses are noiseless, that challenges and responses are both vectors of arbitrary length, and/or that challenge-response pairs are independent. Such assumptions are not realistic in practice. We also contribute to the design of a new protocol that aims to resolve the issues of two existing protocols.
- **Chapter 6: Conclusions and Further Work**. We revisit the main contributions of this thesis and make suggestions for further research.

Chapter 2

Preliminaries

This chapter is organized as follows. Section 2.1 introduces the notation that will be used throughout this thesis. Sections 2.2 to 2.6 review preliminary knowledge of statistics, coding theory, technologies for embedded NVM, and cryptology. Section 2.7 provides a conceptual introduction to PUFs.

2.1 Notation

The following notational system is used consistently throughout this thesis:

- **Variables**. Vectors are denoted by a bold-faced, lowercase character, e.g., $\mathbf{x} = (x_1\, x_2)$. All vectors are row vectors. All-zeros and all-ones vectors are denoted by $\mathbf{0}$ and $\mathbf{1}$ respectively. Matrices are denoted by a bold-faced, uppercase character, e.g., $\mathbf{X}$. The $q \times q$ *identity matrix* is denoted by $\mathbf{I}_q$. A random variable is denoted by an uppercase character, e.g., X. A set, often but not necessarily referring to all possible outcomes of a random variable, is denoted by an uppercase, calligraphic character, e.g., $\mathcal{X}$. The set of all λ-bit row vectors is denoted by $\{0,1\}^\lambda$. If the length λ is variable rather than constant, we denote this set by $\{0,1\}^\star$. Variables that change over time are occasionally denoted with an additional counter between round brackets, e.g., $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, etc.
- **Operations**. Assignment operations are denoted by an arrow, e.g., $x \leftarrow 0$. For binary vectors and binary matrices, addition, e.g., $\mathbf{x} \oplus \mathbf{y}$, and multiplication, e.g., $\mathbf{X}\,\mathbf{y}$ are performed modulo 2. Bitwise inversion is

denoted by an overline, e.g., $\overline{(1\ 0\ 1)} = (0\ 1\ 0)$. The cardinality of a set $\mathcal{X}$, i.e., the number of elements of $\mathcal{X}$, is denoted by $|\mathcal{X}|$. Custom-defined procedure names are printed in a sans-serif font, e.g., Hamming weight $\mathsf{HW}(\mathbf{x})$ and Hamming distance $\mathsf{HD}(\mathbf{x}, \mathbf{y})$. The concatenation of row vectors is denoted by a double vertical bar, e.g., $\mathbf{x} \| \mathbf{y}$.

- **Probabilities**. The probability of an event is denoted by $\mathbb{P}(\cdot)$. The expected value of a random variable X is denoted by $\mathbb{E}_{x \leftarrow X}[X]$. The *probability density function* (PDF) of a continuous random variable and the *probability mass function* (PMF) of a discrete random variable are both denoted by $\mathsf{f}(\cdot)$. Likewise, the *cumulative distribution function* (CDF) of a random variable is denoted by $\mathsf{F}(\cdot)$.

2.2 Min-Entropy

The *min-entropy* [136] of a discrete random variable X as defined in (2.1) is a worst-case predictability measure. The probability that an attacker guesses a secret $x \leftarrow X$ first time right is quantified on a logarithmic scale. It holds that $0 \leq \mathbb{H}_\infty(X) \leq \log_2(|\mathcal{X}|)$; the upper bound corresponds to a uniform distribution. For binary vectors, i.e., $\mathcal{X} = \{0,1\}^\lambda$, it holds that $0 \leq \mathbb{H}_\infty(X) \leq \lambda$.

$$\mathbb{H}_\infty(X) \overset{\text{def}}{=} -\log_2\Big(\max_{x \in \mathcal{X}} \mathbb{P}(X = x)\Big). \tag{2.1}$$

Consider now a pair of possibly correlated random variables: (X, Y). The *conditional min-entropy* [39] of X given Y is as defined in (2.2). Terms of the expectation with $\mathbb{P}(Y = y) = 0$ are evaluated as 0. The definition still quantifies the probability that an attacker guesses the secret x first time right, on a logarithmic scale. It holds that $0 \leq \tilde{\mathbb{H}}_\infty(X|Y) \leq \mathbb{H}_\infty(X)$; the upper bound corresponds to cases where X and Y are independent.

$$\tilde{\mathbb{H}}_\infty(X|Y) \overset{\text{def}}{=} -\log_2\Big(\mathbb{E}_{y \leftarrow Y}\big[\max_{x \in \mathcal{X}} \mathbb{P}((X = x)|(Y = y))\big]\Big). \tag{2.2}$$

A proven property [39] of conditional min-entropy is given in (2.3). In absence of random variable Y_2, the inequalities simplify to $\tilde{\mathbb{H}}_\infty(X|Y_1) \geq \mathbb{H}_\infty((X, Y_1)) - \log_2(|\mathcal{Y}_1|) \geq \mathbb{H}_\infty(X) - \log_2(|\mathcal{Y}_1|)$.

$$\begin{aligned} \tilde{\mathbb{H}}_\infty(X|(Y_1, Y_2)) &\geq \tilde{\mathbb{H}}_\infty((X, Y_1)|Y_2) - \log_2(|\mathcal{Y}_1|) \\ &\geq \tilde{\mathbb{H}}_\infty(X|Y_2) - \log_2(|\mathcal{Y}_1|). \end{aligned} \tag{2.3}$$

Although we consistently adopt the notion of min-entropy in this thesis, it is worth mentioning that *Shannon entropy* is frequently used for the analysis of PUF-based systems as well. The latter notion, as is defined in (2.4), quantifies the amount of information in a discrete random variable X. To be precise, $\mathbb{H}(X)$ comprises a lower bound on the expected number of bits that a lossless compression algorithm may output for a given $x \in \mathcal{X}$. In terms of security, Shannon entropy quantifies the average-case predictability of X, and is hence less conservative than min-entropy. It can be seen easily that $\mathbb{H}_\infty(X) \leq \mathbb{H}(X)$, regardless of the distribution of X.

$$\mathbb{H}(X) \stackrel{\text{def}}{=} -\mathbb{E}_{x \leftarrow X}\Big[\log_2\big(\mathbb{P}(X = x)\big)\Big]. \tag{2.4}$$

Likewise, *conditional Shannon entropy*, as is defined in (2.5), is less conservative than conditional min-entropy. After applying Jensen's inequality to (2.2), i.e., $\log_2(\mathbb{E}[\cdot]) \geq \mathbb{E}[\log_2(\cdot)]$, given that the logarithm is a concave function, it can be seen that $\tilde{\mathbb{H}}_\infty(X|Y) \leq \tilde{\mathbb{H}}(X|Y)$, regardless of the distribution of (X, Y).

$$\tilde{\mathbb{H}}(X|Y) \stackrel{\text{def}}{=} -\mathbb{E}_{y \leftarrow Y}\Big[\sum_{x \in \mathcal{X}} \mathbb{P}((X = x)|(Y = y)) \log_2\big(\mathbb{P}((X = x)|(Y = y))\big)\Big]. \tag{2.5}$$

2.3 Probability Distributions

Random variables are described by a probability distribution. The PDF and CDF of a normally distributed random variable $X \sim N(\mu, \sigma^2)$ with mean μ and variance σ^2 are given in (2.6). For a standard normal distribution $N(0,1)$, the notation simplifies to $\mathsf{f}_{\text{norm}}(x)$ and $\mathsf{F}_{\text{norm}}(x)$ respectively.

$$\mathsf{F}_{\text{norm}}(x; \mu, \sigma) \stackrel{\text{def}}{=} \int_{-\infty}^{x} \mathsf{f}_{\text{norm}}(y; \mu, \sigma)\ \mathrm{d}y \stackrel{\text{def}}{=} \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\ \mathrm{d}y. \tag{2.6}$$

The PDF of a λ-variate normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\Sigma}$ the symmetric covariance matrix, is given in (2.7). If $\mathbf{y} = \boldsymbol{\tau} + \mathbf{x}\,\mathbf{B}$ is an affine transformation of $X \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then $Y \sim N(\boldsymbol{\tau} + \boldsymbol{\mu}\,\mathbf{B}, \mathbf{B}^T\,\boldsymbol{\Sigma}\,\mathbf{B})$.

$$\mathsf{f}_{\text{norm}}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \stackrel{\text{def}}{=} \frac{1}{\sqrt{(2\pi)^\lambda \det(\boldsymbol{\Sigma})}} \exp\Big(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})^T\Big). \tag{2.7}$$

The PMF and CDF of a binomial distribution $B(\lambda, p)$ with λ independent Bernoulli trials that each have success probability p is given in (2.8). Its mean and variance are λp and $\lambda p(1-p)$ respectively.

$$\mathsf{F}_{\text{bino}}(\eta; \lambda, p) \overset{\text{def}}{=} \sum_{i=0}^{\eta} \mathsf{f}_{\text{bino}}(i; \lambda, p) \overset{\text{def}}{=} \sum_{i=0}^{\eta} \binom{\lambda}{i} p^i (1-p)^{\lambda - i}. \tag{2.8}$$

More generally, the Poisson binomial distribution captures the number of successes for independent Bernoulli trials that are not necessarily identically distributed. The PMF and CDF are given in (2.9).

$$\begin{aligned} \mathsf{F}_{\text{poisbino}}(\eta; p_1, p_2, \ldots, p_\lambda) &\overset{\text{def}}{=} \sum_{i=0}^{\eta} \mathsf{f}_{\text{poisbino}}(i; p_1, p_2, \ldots, p_\lambda) \\ &\overset{\text{def}}{=} \sum_{i=0}^{\eta} \sum_{\substack{\mathcal{J} \subseteq \{1,2,\ldots,\lambda\} \\ \text{such that } |\mathcal{J}| = i}} \prod_{j \in \mathcal{J}} p_j \prod_{j \in \{1,2,\ldots,\lambda\} \setminus \mathcal{J}} (1 - p_j). \end{aligned} \tag{2.9}$$

The *statistical distance* [55] between two discrete random variables X_1 and X_2 with a joint set of outcomes $\mathcal{X}$ is as defined in (2.10).

$$\mathsf{SD}(X_1, X_2) \overset{\text{def}}{=} \frac{1}{2} \sum_{x \in \mathcal{X}} |\mathbb{P}(X_1 = x) - \mathbb{P}(X_2 = x)|. \tag{2.10}$$

2.4 Coding Theory

A *binary code* ζ is a bijection from a set of messages $\mathcal{M}$ to a set of codewords $\mathcal{W} \subseteq \{0,1\}^n$. The *minimum distance* d is the minimum number of bits in which any two distinct codewords differ. An encoding procedure maps a message $\mathbf{m} \in \mathcal{M}$ to a codeword $\mathbf{w} \in \mathcal{W}$, i.e., $\mathbf{w} \leftarrow \mathsf{Encode}(\mathbf{m})$. Most error-correction procedures can handle $t = \lfloor (d-1)/2 \rfloor$ errors for any noise-corrupted codeword $\tilde{\mathbf{w}} = \mathbf{w} \oplus \mathbf{e}$, i.e., $\mathsf{Correct}(\tilde{\mathbf{w}}) = \mathbf{w}$ if $\mathsf{HW}(\mathbf{e}) \leq t$. In some cases, more than t errors can be corrected. For convenience, we also define a decoding procedure that returns the corresponding message instead, i.e., $\hat{\mathbf{m}} \leftarrow \mathsf{Decode}(\tilde{\mathbf{w}})$. Equation (2.11) expresses the Hamming bound [108]. The equality holds for *perfect codes* only; it then holds that any vector in $\{0,1\}^n$ is within distance t of a codeword. All other codes are subject to the inequality.

$$\sum_{i=0}^{t} \binom{n}{i} |\mathcal{M}| \leq 2^n. \tag{2.11}$$

A binary $[n, k, d]$ *block code* ζ restricts the message length $k = \log_2(|\mathcal{M}|)$ to an integer. For linear block codes, which are discussed next, any linear combination of codewords is again a codeword. A $k \times n$ *generator matrix* $\mathbf{G}$, having full rank, can then implement the encoding procedure, i.e., $\mathbf{w} = \mathbf{m}\,\mathbf{G}$. For any translation $\boldsymbol{\tau} \in \{0,1\}^n$ and linear code ζ, the set $\{\boldsymbol{\tau} \oplus \mathbf{w} : \mathbf{w} \in \mathcal{W}\}$ is referred to as a *coset*. Two cosets are either disjoint or coincide [108]. Therefore, the set $\{0,1\}^n$ is fully covered by 2^{n-k} cosets. The minimum weight vector $\mathbf{e}$ in a coset is called the *coset leader*. In case of a conflict, i.e., a common minimum $\mathsf{HW}(\mathbf{e}) > t$, an arbitrary leader can be selected. The minimum distance d of a linear code equals the minimum Hamming weight of its nonzero codewords. A linear code ζ is *cyclic* if every circular shift of a codeword results again in a codeword belonging to ζ.

A generator matrix is in *standard form* if $\mathbf{G} = (\mathbf{I}_k\,\mathbf{P})$, i.e., the first k bits of a codeword equal the message, followed by $n - k$ redundancy bits. A *parity-check matrix* $\mathbf{H}$, with dimensions $(n-k) \times n$, determines the so-called *syndrome* $\boldsymbol{\varphi} = \tilde{\mathbf{w}}\,\mathbf{H}^T$ of a possibly corrupted codeword $\tilde{\mathbf{w}}$. The syndrome captures all the information necessary for decoding $\tilde{\mathbf{w}}$. For each codeword $\mathbf{w}$, it holds that $\mathbf{w}\,\mathbf{H}^T = \mathbf{0}$. Therefore, the syndrome can be rewritten as $\boldsymbol{\varphi} = \mathbf{e}\,\mathbf{H}^T$. It can be seen easily that two vectors $\tilde{\mathbf{w}}_1$ and $\tilde{\mathbf{w}}_2$ have the same syndrome $\boldsymbol{\varphi}$ if and only if they belong to the same coset. Generator and parity-check matrices can be derived from each other, e.g., for a generator matrix in standard form, $\mathbf{H} = (\mathbf{P}^T\ \mathbf{I}_{n-k})$.

A binary (η, λ) *convolutional code*, where η is the output data rate and λ is the *constraint length*, has an arbitrary message length k. The encoder slides a window of length λ over a message $\mathbf{m} \in \{0,1\}^k$ and XORs its contents into η codeword bits at any given position. If both sides of message $\mathbf{m}$ are padded with 0s, then the length n of a codeword $\mathbf{w}$ equals $(k + \lambda - 1)\eta$. Convolutional codes are linear and could for a given message length k hence be represented by a generator matrix $\mathbf{G}$. Despite the small minimum distance $d \leq \eta\,\lambda$, on average, a number of errors that is considerably larger than $t = \lfloor (d-1)/2 \rfloor$ can be corrected [174].

2.5 Embedded, Non-Volatile Memory

Some of the variables that are used in cryptographic protocols require permanent storage in a miniaturized electronic device. Table 2.1 lists three categories of embedded, *non-volatile memory* (NVM). The more flexibility in terms of write cycles, the higher the manufacturing cost, and the higher the required resources such as the silicon area per bit. The first category is hard-wired *read-only memory* (ROM). Its mask-defined contents are shared by each manufactured device.

Table 2.1: Embedded NVM.

Type	**Write cycles**	**Mature technologies**
ROM	1×, pre-manufacturing	hard-wired
OTP	1×, post-manufacturing	fuses, antifuses
MTP	$\infty\times$, post-manufacturing	EEPROM, Flash, battery-backed SRAM

The second category is *one-time programmable* (OTP) NVM. Technologies are based on either fuses or antifuses, where a conductive path is irreversibly broken and created respectively, and are either foundry-specific or foundry-independent. Several *intellectual property* providers offer foundry-independent OTP NVMs that require neither additional masks nor additional manufacturing steps with respect to a regular *complementary metal–oxide–semiconductor* (CMOS) process. This includes eMemory [43], Kilopass [84], NSCore [128], Sidense [154], and SST [155]. A limited number of write cycles can be emulated via the partitioning of a large OTP NVM. Writing involves disabling and enabling the current and next partition respectively.

The third category is true *multiple-time programmable* (MTP) NVM, supporting a number of write cycles that is virtually unlimited with respect to a typical lifetime of a device. Unfortunately, traditional technologies are expensive. *Electrically erasable programmable read-only memory* (EEPROM) and its successor Flash require floating-gate transistors, which results in additional masks and processing steps. Battery-backed *static random-access memory* (SRAM) relies on a CMOS-compatible volatile memory, but batteries are considered to be expensive. There are cell structures for MTP NVM that do not further complicate the manufacturing process, as provided by, e.g., eMemory [43]. However, these initiatives should not be considered as a commodity yet. Furthermore, the storage density is typically lower than for EEPROM/Flash.

2.6 Cryptographic Algorithms

The bulk of this thesis relates to *symmetric-key cryptography*. It is then assumed that two or more parties have access to a shared secret $\mathbf{k}$ that is chosen uniformly at random from $\{0,1\}^{\kappa}$. *Brute-force attacks* that enumerate all 2^{κ} possible values of the key should be computationally infeasible. A frequently recommended *key size* nowadays is $\kappa = 128$; the advent of *quantum computers* can be anticipated by choosing $\kappa = 256$ [127]. Below, we review seven efficiently-implementable primitives. For abstraction purposes, these primitives are often modeled by an ideal specification that cannot be realized in practice. Another side note is that there exist generic constructions to convert one primitive into another one.

- A *block cipher* consists of two paired algorithms: encryption $\mathbf{a} \leftarrow \mathsf{Encrypt}(\mathbf{m};\mathbf{k})$ and decryption $\mathbf{m} \leftarrow \mathsf{Decrypt}(\mathbf{a};\mathbf{k})$, with $\mathbf{m},\mathbf{a} \in \{0,1\}^{\lambda}$ a plaintext and a ciphertext respectively. For any given key $\mathbf{k} \in \{0,1\}^{\kappa}$, encryption comprehends a permutation of the set $\{0,1\}^{\lambda}$ and decryption comprehends the inverse permutation. A well-known cipher instance is the *Advanced Encryption Standard* (AES) [35], issued by the United States' *National Institute of Standards and Technology* (NIST) in 2001. Although the *block size* λ is a constant, several modes of operation provide confidentiality for arbitrary-length messages $\mathbf{m} \in \{0,1\}^{\star}$. An ideal, non-realizable block cipher would comprise a list of 2^{κ} permutations that are selected independently, randomly, and uniformly from the set of all possible permutations of $\{0,1\}^{\lambda}$.

- A *cryptographic hash function* $\mathbf{a} \leftarrow \mathsf{Hash}(\mathbf{m})$ maps an arbitrary-length input $\mathbf{m} \in \{0,1\}^{\star}$ to a fixed-length *digest* $\mathbf{a} \in \{0,1\}^{\lambda}$. The following three security properties are of interest. First, *pre-image resistance* implies that for a given digest $\mathbf{a}$, it should be computationally infeasible to find any input $\mathbf{m}$ such that $\mathbf{a} = \mathsf{Hash}(\mathbf{m})$. Furthermore, *second pre-image resistance* implies that for a given input $\mathbf{m}_1$, the attacker cannot find a second input $\mathbf{m}_2$, with $\mathbf{m}_1 \neq \mathbf{m}_2$, such that $\mathsf{Hash}(\mathbf{m}_1) = \mathsf{Hash}(\mathbf{m}_2)$. Finally, *collision resistance* implies that the attacker cannot find two inputs $\mathbf{m}_1$ and $\mathbf{m}_2$, with $\mathbf{m}_1 \neq \mathbf{m}_2$, such that $\mathsf{Hash}(\mathbf{m}_1) = \mathsf{Hash}(\mathbf{m}_2)$. Due to the *birthday paradox*, a brute-force attacker is expected to find a first collision by evaluating roughly $2^{\lambda/2}$ inputs. A standardized instance is the *Secure Hash Algorithm 3* (SHA-3), as released by NIST in 2015 [42]. An ideal, non-realizable hash function would behave as a *random oracle*, i.e., a deterministic mapping from inputs $\mathbf{m}$ to responses $\mathbf{a}$ that are selected independently, randomly, and uniformly from $\{0,1\}^{\lambda}$.

- A *message authentication code* (MAC) is the fixed-length output $\mathbf{a} \in \{0,1\}^\lambda$ of a MAC algorithm $\mathbf{a} \leftarrow \mathsf{MAC}(\mathbf{m};\mathbf{k})$, also referred to as a *keyed cryptographic hash function*, and is usually appended to the arbitrary-length input $\mathbf{m} \in \{0,1\}^\star$ so as to enable verification of its authenticity. For a secure *unforgeable* MAC, an attacker cannot compute the MAC value $\mathbf{a}_1$ of a given message $\mathbf{m}_1$ without knowledge of the key $\mathbf{k}$, even when given the MAC value $\mathbf{a}_2$ of many adaptively chosen messages $\mathbf{m}_2$ with $\mathbf{m}_1 \neq \mathbf{m}_2$. Although instances are not necessarily based on cryptographic hash functions, a well-known, standardized proposal is the *keyed-hash message authentication code* (HMAC) [93].

- A *pseudorandom function family* (PRF) $\mathbf{a} \leftarrow \mathsf{PRF}(\mathbf{m};\mathbf{k})$, with key $\mathbf{k} \in \{0,1\}^\kappa$, consists of 2^κ functions that each map an input $\mathbf{m} \in \{0,1\}^\lambda$ to an output $\mathbf{a} \in \{0,1\}^\eta$. For a secure PRF, it should be computationally infeasible to distinguish between a function that is chosen uniformly at random from the family and a random oracle, the latter of which is non-realizable. This holds even if the attacker is given access to many adaptively chosen input–output pairs $(\mathbf{m},\mathbf{a})$.

- A *synchronous stream cipher* expands a key $\mathbf{k} \in \{0,1\}^\kappa$ into an arbitrary-length pseudorandom stream $\mathbf{a} \in \{0,1\}^\star$ and typically performs encryption and decryption through XORing, i.e., $\mathbf{b} \leftarrow \mathbf{m} \oplus \mathbf{a}$ and $\mathbf{m} \leftarrow \mathbf{b} \oplus \mathbf{a}$ respectively. This comprehends an approximation of the *information-theoretically* secure but impractical cipher that is referred to as a *one-time pad*. It then holds that $\mathbf{k},\mathbf{a},\mathbf{b} \in \{0,1\}^\kappa$, i.e., key size κ tends to be prohibitively large. For both stream ciphers and one-time pads, keys are not to be reused, as this would imply that $\mathbf{b}_1 \oplus \mathbf{b}_2 = \mathbf{m}_1 \oplus \mathbf{m}_2$.

- A cryptographically secure *pseudorandom generator* [41] expands a random seed $\mathbf{n} \in \{0,1\}^\lambda$ into an output $\mathbf{a} \in \{0,1\}^\star$ such that it is computationally infeasible to distinguish $\mathbf{a}$ from true randomness. We, however, refer to a *pseudorandom number generator* (PRNG) as an algorithm that does not necessarily satisfy this property. The most notable inclusion is a *linear-feedback shift register* (LFSR); it is then straightforward to recover the λ-bit state from λ output bits.

- A *key derivation function* (KDF) [92] converts a secret $\mathbf{x} \in \{0,1\}^\lambda$ that is not necessarily uniformly distributed into one or more secret keys $\mathbf{k} \in \{0,1\}^\kappa$. Formally, $(\mathbf{k}_1,\mathbf{k}_2,\ldots) \leftarrow \mathsf{KDF}(\mathbf{x},\mathbf{n})$, where the optional argument $\mathbf{n} \in \{0,1\}^\eta$ is a uniformly distributed seed.

A fundamental limitation of symmetric-key cryptography is that parties require a secure channel for the initial exchange of a secret key $\mathbf{k}$. Depending on the application, this can be an either tolerable or prohibitive constraint. *Public-*

key cryptography provides an alternative in the latter case, at the cost of computational efficiency. Each party then owns a pair of mathematically related keys: a public key $\mathbf{k}_{\text{pub}}$ and a private key $\mathbf{k}_{\text{priv}}$, both of which are typically larger than a symmetric key $\mathbf{k}$ in order to achieve the same security level against brute-force attacks. An example of a widely adopted algorithm is RSA [139], which is named after the initials of its coauthors Rivest, Shamir, and Adleman. We review two popular uses of public-key cryptosystems:

- *Public-key encryption* allows an arbitrary sender to transmit confidential data by using the public key $\mathbf{k}_{\text{pub}}$ of a recipient, i.e., $\mathbf{a} \leftarrow \mathsf{Encrypt}(\mathbf{m}; \mathbf{k}_{\text{pub}})$, with $\mathbf{m}$ the cleartext and $\mathbf{a}$ the ciphertext. Only the recipient possesses the corresponding private key $\mathbf{k}_{\text{priv}}$ and can hence decrypt the ciphertext, i.e., $\mathbf{m} \leftarrow \mathsf{Decrypt}(\mathbf{a}; \mathbf{k}_{\text{priv}})$.
- A *digital signature* $\mathbf{a}$ is appended to a message $\mathbf{m}$ in order to protect its authenticity. In contrast to a MAC, it also provides *non-repudiation*, i.e., the sender cannot deny having signed the message $\mathbf{m}$. The private key $\mathbf{k}_{\text{priv}}$ of the sender is used in the signing procedure $\mathbf{a} \leftarrow \mathsf{Sign}(\mathbf{m}; \mathbf{k}_{\text{priv}})$. An arbitrary recipient can verify the signature using the public key $\mathbf{k}_{\text{pub}}$ of the sender, i.e., $b \leftarrow \mathsf{Verif}(\mathbf{m}, \mathbf{a}; \mathbf{k}_{\text{pub}})$, with bit b denoting either acceptance or rejection. For messages $\mathbf{m}$ of considerable size, it is usually more efficient to sign and verify a digest $\mathsf{Hash}(\mathbf{m})$ instead. This also precludes certain forgeries of RSA signatures.

2.7 Physically Unclonable Functions

Physically unclonable functions (PUFs) comprehend the central theme of this thesis. This primitive lacks a formal, unambiguous definition that is widely accepted and adopted by the research community. Most authors hence opt for an informal description through the enumeration of compelling properties that comply with most practical designs. We first provide such an enumeration, and elaborate our vision on formal definitions afterwards.

2.7.1 Informal

- A PUF captures the inherent manufacturing variability of a physical object through a set of *challenge–response pairs* (CRPs). The term *object* distinguishes PUFs from the related field of human *biometrics*. For the majority of recent applications, the object comprehends an IC. Process variations such as *random dopant fluctuation* imply that each transistor exhibits unique characteristics in terms of currents and voltages. Most silicon PUFs allow us to query CRPs within the perimeter of the IC itself, thereby removing the need for external actuation and measurement equipment. Furthermore, challenges and responses are usually binary vectors rather than analog signals so as to facilitate interaction with other building blocks of the IC. We limit the scope of this thesis to the latter type of silicon PUF and do not further consider designs that rely on, e.g., randomly distributed particles in optical tokens [131] and magnetic media [71].

- Unfortunately, the same design techniques that make a PUF sensitive to process variations cause it to pick up noise as well. Variability and noise refer to deviations from the nominal behavior that are reproducible and irreproducible respectively. While the former is defined through the structural rigidity of the solid materials that constitute an IC, the latter is a time-dependent phenomenon that originates from randomly moving particles. For example, Johnson-Nyquist noise comprises the unavoidable vibrations of electrons and other thermally agitated charge carriers. Even in perfectly stable laboratory conditions, a PUF's response to a given challenge is hence not perfectly reproducible.
- Perturbations of the IC's environment further deteriorate the reproducibility. For example, the waveform of an externally provided supply voltage could deviate from its nominal counterpart. Moreover, the outside temperature continuously changes, especially if there are other heat-generating elements in the IC's vicinity. Even within the perimeter of the IC, various building blocks could interfere with the behavior of the PUF. Depending on the magnitude of the previously listed perturbations, the expected error rate of a response bit with respect to a previously recorded reference usually ranges from 5% to 20%. Apart from a few PUF designs where the reference response is either formed [33] or reinforced [18] by applying high physical stresses after the manufacturing of the IC, reported error rates below 5% typically do not include perturbations in the experimental setup. Note, however, that consumer products are supposed to remain functional in spite of environmental changes.
- Aging effects alter the nominal behavior of a PUF. Strong electric fields slowly but steadily cause structural damage to an IC. For example, *electromigration* causes the raster of ions that constitutes a metallic wire to gradually break down due to the mechanical impact of colliding electrons. Moreover, *negative-bias temperature instability* alters the current-voltage characteristics of a transistor. Given that reference responses are usually only prerecorded once, their reproducibility progressively deteriorates in the long term. A considerable obstacle for the release of commercial

products with an expected lifetime of several years is that the exact increase in error rate is hard to predict. Accelerated aging experiments that put a PUF-enabled device under high physical stresses for a short period of time provide an approximation at best.

2.7.2 Formal

An *ideal* PUF can be defined with relative ease, as exemplified in Definition 1 for silicon targets.

Definition 1 (Ideal PUF). *For a given manufacturing process of an IC with a given layout, an ideal PUF is a manufactured building block that realizes a function with domain* $\mathcal{C} \subseteq \{0,1\}^{\lambda}$ *and codomain* $\mathcal{R} \subseteq \{0,1\}^{\eta}$*, where each instantiated function depends on process variations and is modeled as a random oracle. For each challenge* $\mathbf{c} \in \mathcal{C}$*, a given function instance then deterministically returns a response* $\mathbf{r}$ *that is chosen uniformly at random from* $\mathcal{R}$*, regardless of environmental changes and aging. All relevant environmental parameters are bounded, e.g., supply voltage* $V_S \in [\alpha_V, \beta_V]$ *and temperature* $T \in [\alpha_T, \beta_T]$*, and a maximum lifespan* β_L *is defined. The evaluation time of any given function instance has an upper bound* β_E*. Tamper-evidence is imposed as an additional requirement, i.e., fully invasive attacks either damage or alter the functional behavior.*

A crucial problem with Definition 1 and various other definitions of ideal PUFs [131, 49, 53] is that practical realizations are non-existent [144, 110]. Or at the very least, most designs that are generally regarded as a PUF do not meet the constraints. Less restrictive definitions are considerably more difficult to devise and are even more prone to personal opinions. Several authors [26, 110] proposed such a definition, none of which gained a widespread following, presumably in part because PUF literature already proliferated into chaos, i.e., authors are used to informal, do-it-yourself descriptions that avoid non-strictly required notational complexities. We nevertheless aim to capture a large variety of non-ideal designs in Definition 2, which is tailored for PUF-based key generation. To be precise, there is compatibility with so-called *secure sketches* and *fuzzy extractors* [39], both of which will be defined in Chapter 4. Note that the 'F' in PUF is a misnomer, given that a mathematical *function* is supposed to be deterministic.

Definition 2 (Non-ideal PUF). *For a given manufacturing process of an IC with a given layout, a non-ideal PUF is a manufactured building block that realizes a non-deterministic mapping from a set* $\mathcal{C} \subseteq \{0,1\}^{\lambda}$ *to a set* $\mathcal{R} \subseteq \{0,1\}^{\eta}$*, where the distribution of each random variable* R_i*, with* $i \in [1, |\mathcal{C}|]$*, depends on process* *variations, noise, environmental variables, and aging. All relevant environmental parameters are bounded, e.g., supply voltage* $V_S \in [\alpha_V, \beta_V]$ *and temperature* $T \in [\alpha_T, \beta_T]$*, and a maximum lifespan* β_L *is defined. The evaluation time of any given PUF has an upper bound* β_E*. Let* $\mathbf{x}$ *denote the result of concatenating a PUF's responses* $\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_g$ *to a list of publicly known challenges* $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_g$*. Considering the infinite set of PUFs in their nominal environment, i.e., supply voltage* $V_S = v_{\text{nom}}$*, where* $v_{\text{nom}} \in [\alpha_V, \beta_V]$*, temperature* $T = t_{\text{nom}}$*, where* $t_{\text{nom}} \in [\alpha_T, \beta_T]$*, etc., at the beginning of their life cycle, it should hold that the min-entropy* $\mathbb{H}_\infty(X_{\text{nom}}) \geq \alpha_X$*. Finally, it should hold that the* α_H*-th percentile of* $\mathsf{HD}(X_{\text{nom}}, X_{\text{rec}})$*, where reproduced response* X_{rec} *complies with environmental and lifespan constraints, has an upper bound* t*.*

Chapter 3

Security Analysis of PUF Circuits — Novel Methods

We develop new methods for the security analysis of a stand-alone PUF circuit. A first contribution is the derivation of tight upper bounds on the min-entropy of several PUFs, i.e., Ring Oscillator Sum, Loop, Arbiter, Feed-Forward Arbiter, S-ArbRO, and XOR PUFs. This constrains their usability for the fuzzy extraction of a secret key, as an alternative to storing keys in NVM. For example, it is shown that ideally manufactured Arbiter PUFs with 64 stages cannot provide more than 197 bits of min-entropy. Van Herrewege et al. [171] previously assumed that 1785 bits of min-entropy can be extracted, which renders their 128-bit key generator instantly insecure. We also derive upper bounds that comply with non-ideally manufactured PUFs, as is the case in practice for silicon realizations. As a side contribution, we refute the claim that S-ArbRO PUFs are highly resistant to machine learning attacks.

Second, we are the first to experimentally demonstrate that the noisiness of the responses of a PUF can be leveraged as a side-channel. By repeatedly measuring the response bit r to an identical challenge $\mathbf{c}$, the one and only internal non-linearity of both Ring Oscillator Sum and Arbiter PUFs is bypassed. Therefore, a predictive model can be constructed by solving a system of linear equations, i.e., the attacker is faced with trivial mathematics. Moreover, we experimentally demonstrate that the noisiness of a PUF, and hence also the efficiency of the attack, can be increased by applying environmental changes to the IC. Although the attack remains less efficient than state-of-the-art machine learning attacks, follow-up work by other authors turned the tables. A hybrid attack that combines noise measurements with a machine learning algorithm turns out to be the most effective and efficient method for modeling XOR PUFs to date, for example.

Version History. The version history of this chapter is as follows. Our publication at the *1st Asian IEEE symposium on Hardware-Oriented Security and Trust* (AsianHOST 2016) is considerably extended. Our contributions to the *6th IEEE symposium on Hardware-Oriented Security and Trust* (HOST 2013) and the *IEEE Transactions on Circuits and Systems I* (TCAS-I 2014) are summarized more briefly, given the improvements by other authors. The author of this PhD thesis is the main contributor to all three publications.

Organization. The remainder of this chapter is organized as follows. Section 3.1 specifies a representative set of PUF circuits that is used for illustrative purposes throughout this thesis. Section 3.2 derives upper bounds on the min-entropy of several of the selected PUFs. Section 3.3 presents the experimental results of our noise-based physical attacks on two of the selected PUFs. Section 3.4 concludes this chapter.

3.1 A Representative Set of PUF Circuits

A staggering number of PUF circuits has already been published and new designs keep being proposed with no end in sight. Luckily, the underlying mechanisms are fairly similar, and there is hence no urgent need for exhaustive listings here. We present a small but representative set of PUF circuits that merely serves to illustrate our main technical contributions later-on in this thesis. Our set includes the three most well-known and well-studied PUFs, i.e., Arbiter PUFs [100, 105], SRAM PUFs [98, 53, 68], and Ring Oscillator PUFs [161, 183], as well as some of their variations.

Every PUF circuit comprises an analog core that is encapsulated by a digital shell. The waveform of analog signals within the core, i.e., currents and voltages, deviates from its nominal counterpart due to both process variations and noise. A conversion from analog to digital quantities is needed such that the process variations are captured without amplifying the noise. The selected PUF designs all perform this conversion by comparing the waveform of two nominally identical signals. Environmental perturbations and process drift might affect both signals equally and hence cancel out to some extent. A naive design where the waveform of a single signal is compared against a hardcoded reference is not only less portable across technologies, but also more susceptible to environmental changes.

3.1.1 Two Design Flavors: Weak and Strong PUFs

Given that silicon area is a scarce resource, two opposing design topologies emerge according to the number of CRPs that a PUF provides. For so-called *weak* and *strong PUFs*, the number of unique challenges $\mathbf{c}$ scales polynomially and exponentially with the circuit area respectively. Polynomial coefficients are as such that when given unrestricted access to a reasonably sized weak PUF, one can exhaustively query and tabulate all CRPs in relatively limited time and hence construct a software clone. For strong PUFs of sufficient size, however, it is infeasible to enumerate all CRPs. This entails a world of difference for cryptographic applications.

The most straightforward weak PUF design consist of an array of autonomous, identically laid-out cells. Each cell produces a single bit, or occasionally, a few bits. The address of a single cell, or alternatively, a cluster of cells serves as the challenge $\mathbf{c}$, i.e., cardinality $|\mathcal{C}|$ scales linearly with the circuit area. The primary use of weak PUFs in a security architecture is the generation of a device-unique secret key $\mathbf{k} \in \{0,1\}^{\kappa}$. To be precise, a read-out of the complete array of cells, hereby enumerating all possible challenges $\mathbf{c} \in \mathcal{C}$ in, e.g., counter mode, provides a noisy secret $\mathbf{x} \in \{0,1\}^{\lambda}$, with $\lambda > \kappa$, that can be post-processed into a stable secret key $\mathbf{k}$. As cells are intended to operate autonomously, the distribution of the secret X has the potential to be fairly uniform with respect to the infinite set of manufactured devices.

Based on their virtually unlimited number of challenges $\mathbf{c}$, strong PUFs are a seemingly more versatile component in the security architecture of an electronic device. The other side of the coin, however, is that their CRPs are highly correlated. One manifestation thereof is that machine learning algorithms allow for the construction of an accurate predictive model, given a relatively small training set of CRPs $\{(\mathbf{c}_1, \mathbf{r}_1), (\mathbf{c}_2, \mathbf{r}_2), \ldots, (\mathbf{c}_g, \mathbf{r}_g)\}$. Commonly used learning techniques include *artificial neural networks*, *logistic regression*, and *evolution strategies*. The resistance of a given strong PUF design is usually quantified by both the number of training CRPs, i.e., g, and the algorithm runtime that are needed to accurately model an instance.

There does not exist a single well-validated design of a strong PUF that is known to be fully resistant to machine learning attacks, i.e., given unrestricted access to the CRPs, the construction of a software clone is always feasible in practice. In 2012, Maes [110] questioned whether the design of a resistant strong PUF is at all possible, and also several years later, it remains an open problem. Occasionally, the introduction of a new design is accompanied by sensational resistance claims. However, analysis by independent parties tends to refute such claims, sooner or later. Just like their weak cousins, strong PUFs usually would have to team up with cryptographic algorithms in order to provide excellent security guarantees.

Three reasons limit our optimism for a true breakthrough in terms of modeling resistance. First, there are no success stories yet, although Gassend et al. [50] mapped a strong PUF to silicon in 2002 already, and although the quest for a resistant design remained an active research topic ever since. Second, strong PUFs extract their enormous number of CRPs from a limited silicon area only, hereby incorporating a relatively small number of circuit elements. A highly correlated structure that reuses elementary contributions to the overall manufacturing variability is the unavoidable consequence. Third, unlike cryptographic algorithms, strong PUF designs are highly constrained in their use of *confusion* and *diffusion* techniques. Otherwise, contributions from local noise sources would accumulate and render the response bits prohibitively unstable.

As a side note, the above subdivision into weak and strong PUFs comprehends a popular adaptation of more restrictive notions that were proposed by Guajardo et al. [53] in 2007. Originally, weak and strong PUFs were also required to be *tamper evident*, i.e., physical invasion is supposed to permanently alter the functional behavior of a PUF. Moreover, the knowledge of an arbitrary CRP $(\mathbf{c}_1, \mathbf{r}_1)$ may only provide a negligible amount of information on the response $\mathbf{r}_2$ to any challenge $\mathbf{c}_2$, with $\mathbf{c}_1 \neq \mathbf{c}_2$. The latter requirement seems hard to meet for strong PUFs in particular. Although relevant as the specification of an ideal PUF, we stick to the more recent notions as these adequately subdivide the designs that are realizable and in use today.

3.1.2 Selected Weak PUFs

We describe several weak PUFs.

SRAM and Other Memory-Based PUFs

An unmodified SRAM can be adopted as a PUF, as first proposed by Layman et al. [98], and later also by Guajardo et al. [53] and Holcomb et al. [68]. Its initial state after power-up provides a device-unique fingerprint. Note that on transistor-level, an SRAM cell can be understood as pair of cross-coupled inverters, i.e., NOT gates. Random process variations cause one inverter to operate faster than the other, and depending thereupon each cell exhibits a preference to initialize as either a 1 or a 0. Functional behavior is inherently foreseen, i.e., the memory address serves as the challenge $\mathbf{c}$ and the corresponding cell contents serve as the response $\mathbf{r}$.

Other bistable cells that consist of a positive feedback loop are known to exhibit PUF behavior as well. Various latches and flip-flops, which operate in a synchronous and asynchronous manner respectively, have been used. The cell of Su et al. [160] comprises a set-reset latch that consists of two cross-coupled NOR gates. Likewise, the cell of Maes et al. [113] comprises of a single D-type flip-flop. Unfortunately, its responses turn out to be highly biased [113, 104, 89, 81], i.e., 0s and 1s are not equally likely to occur. This can presumably be attributed to an asymmetry in the circuitry. Each cell of the Butterfly PUF of Kumar et al. [97] comprises two cross-coupled D-type flip-flops. The cell of Simons et al. [156] comprises a bus-keeper, i.e., a weak latch that consists of two cross-coupled inverters.

Ring Oscillator PUFs

A *ring oscillator* (RO), i.e., a self-oscillating loop that consist of an odd number of inverters, is a commonly used building block for crafting a PUF. Its frequency of oscillation f, i.e., the reciprocal of the total propagation delay, depends on random process variations. Counting the number of rising and/or falling edges within a fixed time span provides a digital estimate thereof.

Consider an array of frequencies: $f_1, f_2, \ldots, f_m$. The following three PUF designs generate response bits r via pairwise comparisons $f_i \lesseqgtr f_j$, with $i \neq j$. In simplified form, the design of Yu et al. [183] partitions the ROs into $|\mathcal{C}| = \lfloor m/2 \rfloor$ disjunct pairs that each generate one response bit r. Maiti and Schaumont [116] subdivide the ROs into $|\mathcal{C}| = (m-1)$ overlapping pairs instead. Suh and Devadas [161] allow for the selection of every possible pair, i.e., $|\mathcal{C}| = \binom{m}{2} = m(m-1)/2$.

3.1.3 Selected Strong PUFs

We describe several strong PUF designs that are based on the propagation delay of logic gates, or better, the variability thereof. Formally, we specify a function $r \leftarrow \mathsf{PUF}(\mathbf{c}, \boldsymbol{\delta})$, where $\boldsymbol{\delta} \in \mathbb{R}^{\eta}$ collects the relevant delay-based quantities. This abstraction of an ideally manufactured PUF assumes that propagation delays and corresponding frequencies are representable by a single constant. It is hence neglected that, in practice, rising and falling edges are sloping rather than instantaneous. Moreover, noise is not yet taken into account. We also specify for each design an invertible transformation that maps a challenge $\mathbf{c} \in \{0,1\}^\lambda$ to a more convenient representation $\boldsymbol{\gamma} \in \{-1,0,1\}^\eta$.

RO Sum PUFs

As is the case with weak PUFs, several designs of a strong PUF are again RO-based. A first design is the RO Sum PUF of Yu and Devadas [186]. Consider the measurement of m pairwise frequency differences, i.e., $\boldsymbol{\delta} = \big(f_2 - f_1 \;\; f_4 - f_3 \;\; \cdots \;\; f_{2m} - f_{2m-1}\big)$. The challenge $\mathbf{c}$ determines for each pair $i \in [1,m]$ whether either δ_i or its opposite value $-\delta_i$ is accumulated to a variable ω. Written as a dot product, $\omega = \boldsymbol{\delta}\,\boldsymbol{\gamma}^T$, where $\boldsymbol{\gamma} \in \{-1,1\}^m$. A comparison $\omega \lessgtr 0$ results in a single response bit r. Recall that the number of challenges, i.e., $|\mathcal{C}| = 2^m$, scales exponentially with the circuit area.

S-ArbRO PUFs

Similarly, S-ArbRO-2 PUFs were proposed by Ganta and Nazhandali [46]. This design generalizes an RO Sum PUF such that only a subset of q oscillator pairs contributes to ω, where $q \in [1,m]$ is a constant. The selection of q pairs is part of the challenge $\mathbf{c}$. Written as a dot product, $\omega = \boldsymbol{\delta}\,\boldsymbol{\gamma}^T$, where $\boldsymbol{\gamma} \in \{-1,0,1\}^m$ and $\sum_{i=1}^{m} |\gamma_i| = q$. The number of unique challenges $\mathbf{c}$ equals $\binom{m}{q}2^q$.

The same authors also proposed S-ArbRO-4 PUFs. The set of m RO pairs, with m even, is then subdivided into $m/2$ partitions that contain two pairs each. Only a subset of q partitions contributes to ω, where $q \in [1, m/2]$ is a constant. The selection of q partitions is part of the challenge $\mathbf{c}$. Within each contributing partition, either one out of two frequency differences δ is accumulated to ω, without flipping signs. Written as a dot product, $\omega = \boldsymbol{\delta}\,\boldsymbol{\gamma}^T$, where $\boldsymbol{\gamma} \in \{0,1\}^m$, $\sum_{i=1}^{m} \gamma_i = q$, and $\forall i \in [1, m/2], \gamma_{2i} + \gamma_{2i-1} \in \{0,1\}$. The number of unique challenges $\mathbf{c}$ equals $\binom{m/2}{q}2^q$.

Loop PUFs

Gassend et al. [51] were the first to map a strong PUF to silicon. Their design consists of a single reconfigurable RO, where challenge-driven multiplexers allow to locally switch among parallel path segments. A sequential measurement of the frequency f for two different configurations eventually results in a response bit r. As an alternative, the authors suggest comparing the frequencies f of two identically laid-out, reconfigurable ROs. There is a virtually endless number of combinations in which a reconfigurable RO can be constructed from buffers, inverters, non-inverting multiplexers, inverting multiplexers, and other logic gates, as reflected by the large number of slightly differing proposals [99, 116, 184, 32, 47, 138].

One such proposal is the so-called Loop PUF of Rioul et al. [138], which considerably simplifies the challenge sequencer of the original, affiliated design by Cherif et al. [32]. The term *loop* refers to the reconfigurable oscillator that is shown in Figure 3.1. Let δ_i, with $i \in [1, m]$, denote the difference between the propagation delays of stage i when configured with challenge bits $c_i = 1$ and $c_i = 0$ respectively. Given the dot product $\omega = \boldsymbol{\delta}\,\boldsymbol{\gamma}^T$, where $\boldsymbol{\gamma} \in \{-1, 1\}^m$, the corresponding response bit r is generated through a comparison $\omega \lessgtr 0$.

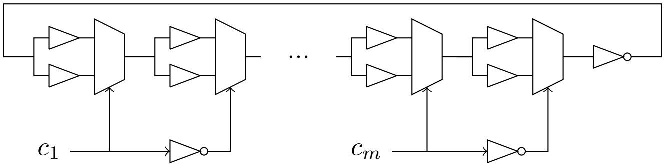


Figure 3.1: A Loop PUF with m stages, as proposed by Rioul et al. [138]. The frequencies of the oscillator when configured with a challenge $\mathbf{c} \in \{0, 1\}^m$ and its additive inverse $\bar{\mathbf{c}}$ respectively are compared in order to generate a response bit r.

Arbiter PUFs

The Arbiter PUF of Lee et al. [100, 105] is shown in Figure 3.2. A rising edge propagates through two reconfigurable paths with nominally identical delays. Because of process variations however, there is a delay difference ω between both paths. An arbiter element decides which path 'wins' the race through a comparison $\omega \lessgtr 0$, and generates a response bit r.

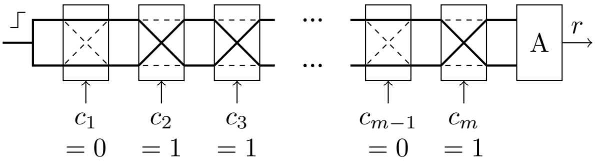


Figure 3.2: An Arbiter PUF with m stages, as proposed by Lee et al. [100, 105].

The two paths are constructed from a series of m switching elements. Challenge $\mathbf{c}$ determines for each stage whether path segments are either crossed or uncrossed. As shown in Figure 3.3, each stage has a unique contribution to the time difference ω, depending on the value of its challenge bit c. Equation (3.1) writes ω as a dot product [105]. The number of unique challenges $\mathbf{c}$ equals 2^m.

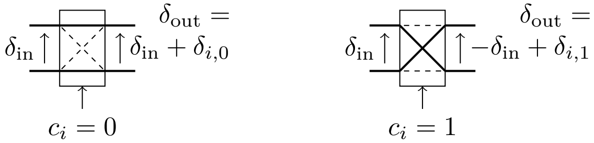


Figure 3.3: The delay behavior of a single stage of an Arbiter PUF.

$$\omega = \boldsymbol{\delta}\,\boldsymbol{\gamma}^T, \quad \text{with } \boldsymbol{\delta} = \boldsymbol{\delta}_\star\,\mathbf{B},$$

$$\boldsymbol{\delta}_\star = \begin{pmatrix}\delta_{1,0} & \delta_{1,1} & \delta_{2,0} & \delta_{2,1} & \dots & \delta_{m,0} & \delta_{m,1}\end{pmatrix},$$

$$\mathbf{B} = \frac{1}{2}\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & -1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 1 \end{pmatrix}^T, \tag{3.1}$$

$$\text{and } \boldsymbol{\gamma} = \begin{pmatrix} (1-2c_1)(1-2c_2)\dots(1-2c_m) \\ (1-2c_2)\dots(1-2c_m) \\ \vdots \\ (1-2c_m) \\ 1 \end{pmatrix}^T.$$

Feed-Forward Arbiter PUFs

To improve the resistance to machine learning attacks, Lee et al. [100] also proposed a feed-forward variant of their Arbiter PUF, as shown in Figure 3.4. Each out of $q \geq 1$ additional arbiter elements then compares the accumulated time difference after a stage $i \in [1, m-1]$ against a threshold 0, and drives the challenge bit c_j of a stage $j \in [i+1, m]$. Note that $|\mathcal{C}| = 2^{m-q}$ challenges remain selectable. In practice, unfortunately, noise effects are amplified by the additional arbiter elements, i.e., an upper bound on q is imposed. Therefore, machine learning attacks remain feasible [118, 145].

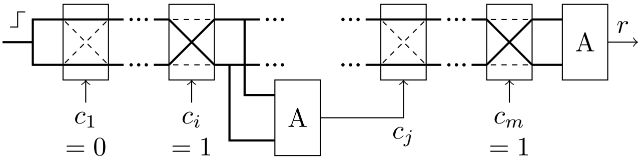


Figure 3.4: A Feed-forward Arbiter PUF with m stages and $q = 1$ loop.

XOR PUFs

As proposed by Suh and Devadas [161], the response bits of a strong PUF can be XORed before release so as to improve the resistance to machine learning attacks. The original single-sentence specification is equally ambiguous as ours, but Rührmair et al. [145] popularized the following interpretation. The XOR PUF comprises $q > 1$ identically designed circuits of a strong PUF that are laid-out in parallel. An identical challenge **c** is fed to all q component PUFs, and their respective response bits r_j, with $j \in [1, q]$, are XORed in order to obtain the final response bit, i.e., $r = r_1 \oplus r_2 \oplus \ldots \oplus r_q$. Although initially proposed for Arbiter PUFs, the use of this technique has later also been suggested for other strong PUFs, e.g., Bistable Ring PUFs [178]. The larger the number of component PUFs, the harder to learn the input–output behavior of the XOR PUF. In practice, unfortunately, the corresponding increase in noisiness imposes an upper bound on q.

Majzoobi et al. [118, 117] proposed a more extensively parameterized alternative to the previously described XOR PUF. Given again the use of q parallel component PUFs, a first crucial difference is that the outgoing XOR network may allow for the extraction of multiple response bits. An example instance with $q = 5$, which is graphically represented by the authors [118, 117], defines the response **r** of the XOR PUF as given in (3.2). However, as pointed out by Sahoo et al. [151], an attacker can XOR well-chosen bits of response **r** so as to obtain easier-to-learn targets. For the given example, an attacker can learn all 2-component combinations, i.e., $r_1 \oplus r_2$, $r_1 \oplus r_3$, …, $r_4 \oplus r_5$, which also implies knowledge of all 4-component combinations.

$$\mathbf{r} = \begin{pmatrix} r_1 \oplus r_2 \oplus r_3 \oplus r_4 \\ r_2 \oplus r_3 \oplus r_4 \oplus r_5 \\ r_3 \oplus r_4 \oplus r_5 \oplus r_1 \\ r_4 \oplus r_5 \oplus r_1 \oplus r_2 \end{pmatrix}^T . \tag{3.2}$$

A second crucial difference is that all q component PUFs do not necessarily evaluate the same challenge $\mathbf{c}$. Majzoobi et al. [118, 117] point out for the Arbiter PUF in particular that its response r is not equally sensitive to a change in value of each of its challenge bits c. As can be derived from (3.1), inverting the value of challenge bits c_1 and c_m causes response r to flip with a probability of nearly 0 and nearly 1 respectively, for example. As a resolution, the authors suggest feeding challenge $\mathbf{c}$ and its reversed counterpart $\begin{pmatrix} c_m & c_{m-1} & \dots & c_1 \end{pmatrix}$ to component PUFs with index $j \in [1, q]$ odd and even respectively. Rührmair et al. [145] later confirmed for XOR PUFs with a single response bit, i.e., $r = r_1 \oplus r_2 \oplus \cdots \oplus r_q$, that the resistance to machine learning attacks is indeed improved this way.

Yu et al. [190, 188] suggest feeding a completely different challenge to each component PUF. This variation of the XOR PUF assumes that challenge bits are produced by a PRNG, which now has to increase its production with a factor q, i.e., $q\,m$ challenge bits c are needed for every response bit r. Control over the applied challenge bits is limited this way. In the hypothetical case of absolute control, an attacker could fix the challenge of $q - 1$ component PUFs to a constant value and gather CRPs so as to model the remaining component PUF individually. However, given the presence of a well-designed PRNG, the resistance of the proposed XOR PUF to machine learning attacks is observed to be higher than for the original proposal of Suh and Devadas [161].

Another XOR PUF variation of Yu et al. [188] consists of a single component PUF only. Its serialized response $\begin{pmatrix} r_1 & r_2 & \cdots & r_q \end{pmatrix}$ to q subsequent challenges is then XORed into a single response bit r. The use of a well-designed PRNG as challenge generator should again ensure that the attacker has limited control. Serialized XORing can be combined with the previously described parallel techniques.

3.1.4 Machine Learning

In spite of Definition 1, realizations of strong PUF do not behave as a random oracle, as is clear from the small-sized variability models $\boldsymbol{\delta}$ in Section 3.1.3. One manifestation thereof is that the *strict avalanche criterion* [176] is not satisfied. Similar to what has already been mentioned for Arbiter PUFs in particular, other strong PUF designs also have challenge bits c that, when flipped, do not necessarily cause the response bit r to flip with a probability of 50%. Although a more detailed quantization of this effect can be insightful [118], we focus on another manifestation of the inherent functional dependencies: machine learning algorithms allow for the construction of an accurate predictive model. Given a relatively small training set of CRPs, i.e., $\{(\mathbf{c}_1, r_1), (\mathbf{c}_2, r_2), \ldots, (\mathbf{c}_g, r_g)\}$, the response r_{g+1} to an unseen challenge $\mathbf{c}_{g+1}$ is predicted with a higher success rate than random guessing would.

The accuracy of a predictive model is formalized in (3.3), where N is the uniformly distributed seed of the possibly randomized training algorithm. It is assumed that challenges $\mathbf{c}_i$ are sampled independently and randomly from a common set $\mathcal{C}$. Monte Carlo experiments allow for an approximate evaluation of the accuracy. Each expectance operator then deteriorates to an average of a relatively small number of random samples. Normally, the function $\mathsf{Accuracy}(g)$ increases monotonically with the number of training CRPs g, and its codomain is $[{}^{1}\!/_{2}, 1]$.

$$\begin{aligned} \mathsf{Accuracy}(g) = \mathbb{E}_{\mathbf{c}_1 \leftarrow C_1}\Big[\mathbb{E}_{\mathbf{c}_2 \leftarrow C_2}\Big[\ldots \mathbb{E}_{\mathbf{c}_{g+1} \leftarrow C_{g+1}}\Big[\mathbb{E}_{\mathbf{n} \leftarrow N}\Big[\mathbb{E}_{\boldsymbol{\delta} \leftarrow \Delta}\Big[\\ \mathbb{P}\big(\mathsf{Predict}(\mathbf{c}_{g+1}; \mathbf{c}_1, r_1, \mathbf{c}_2, r_2, \ldots, \mathbf{c}_g, r_g, \mathbf{n}) = r_{g+1}\big)\Big]\Big]\Big]\ldots\Big]\Big], \\ \text{with } \forall i \in [1, g+1], r_i \leftarrow \mathsf{PUF}(\mathbf{c}_i, \boldsymbol{\delta}). \end{aligned} \tag{3.3}$$

The internal mechanisms of RO Sum, S-ArbRO-2, S-ArbRO-4, Loop, and Arbiter PUFs are fairly similar. There is a linear part, i.e., the dot product $\omega = \boldsymbol{\delta}\,\boldsymbol{\gamma}^T$, followed by a non-linear part, i.e., the threshold operation $\omega \lesseqgtr 0$. Machine learning is greatly facilitated by using pairs $(\boldsymbol{\gamma}_i, r_i)$ as training data instead of pairs $(\mathbf{c}_i, r_i)$, given that this enhances the inherent linearity of the input–output behavior. Out of many techniques that can tackle this fairly straightforward learning problem, we opt for linear regression [56]: a *least squares* solution $\hat{\boldsymbol{\delta}}$ of the usually overdetermined system of linear equations in (3.4) is then computed. This training method is fully deterministic, i.e., the set of seeds $\mathcal{N} = \varnothing$. A response bit r_{g+1} is predicted through a comparison $\big(\boldsymbol{\gamma}_{g+1}\ \ 1\big)\hat{\boldsymbol{\delta}} \lesseqgtr {}^{1}\!/_{2}$. Bias, i.e., an imbalance between the expected number of 0s and 1s, can be captured by making the models affine rather than linear. Except for the variability model of an Arbiter PUF, which already incorporates an intercept, transformed challenges $\boldsymbol{\gamma}$ are therefore extended with a constant 1.

$$\begin{pmatrix} \gamma_1 & 1 \\ \gamma_2 & 1 \\ \vdots & \vdots \\ \gamma_g & 1 \end{pmatrix} \boldsymbol{\delta} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_g \end{pmatrix}. \tag{3.4}$$

Figure 3.5 plots machine learning results for all five PUF designs. Ideal RO Sum PUFs and ideal Loop PUFs are mathematically equivalent, so we only simulate one. S-ArbRO-2 and S-ArbRO-4 PUFs [46] were claimed to be highly resistant to machine learning attacks, especially in comparison with an Arbiter PUF. Our results clearly refute this claim. The authors wrongly assumed that it is necessary to learn a separate model for all possible selections of q contributing stages.

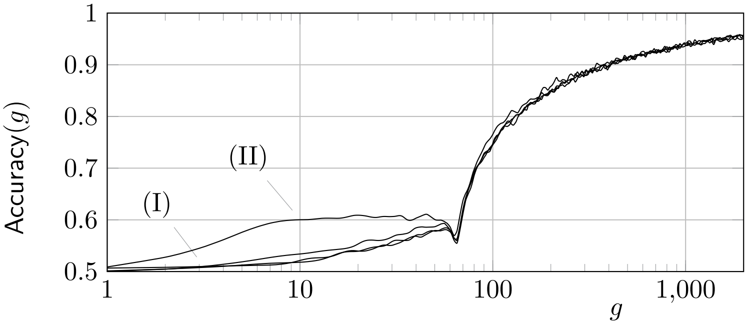


Figure 3.5: The modeling accuracy of ideal (I) RO Sum, S-ArbRO-2, Arbiter, and (II) S-ArbRO-4 PUFs as a function of g, i.e., the number of training CRPs. All designs are instantiated with $m = 64$ stages. For S-ArbRO-2 and S-ArbRO-4 PUFs in particular, we choose $q = 32$ and $q = 16$ contributing stages respectively, as this maximizes their number of unique challenges $\mathbf{c}$. Pairwise frequency and delay differences Δ_i are assumed to be *independent and identically distributed* (i.i.d.) normal random variables. For improved visibility, we did fit a *smoothing spline* through the raw Monte Carlo results.

Three out of four learning curves approximately coincide. The main anomaly, apart from a sudden dip in performance around $g = m = 64$, is that S-ArbRO-4 PUFs are relatively easy to model if the system of equations in (3.4) is underdetermined, i.e., $g < m$. The root cause of this anomaly is a newly revealed issue: bias. Even when ideally manufactured, the instances of an S-ArbRO-4 PUF are considerably in favor of producing either 0s or 1s. As is clear from Figure 3.6, the probability $p_{\text{bias}} = \mathbb{P}(R_i = 1)$ for a given instance tends to differ considerably from 1/2. Given the high number of competing, bias-free PUF designs, it seems unlikely to us that a system provider would be willing to deal with bias-induced complications.

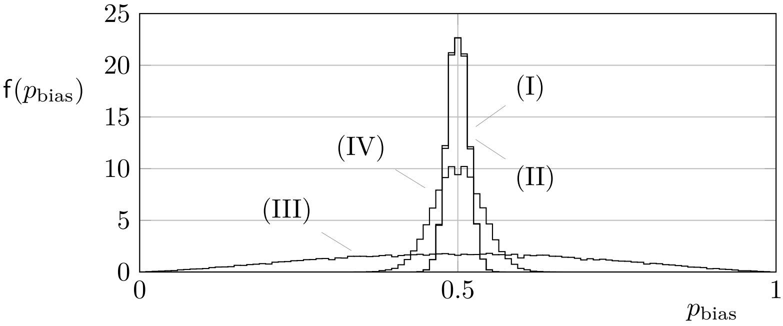


Figure 3.6: An estimated PDF of the bias ratio P_{bias} for ideal (I) RO Sum PUFs, (II) S-ArbRO-2 PUFs, (III) S-ArbRO-4 PUFs, and (IV) Arbiter PUFs. Pairwise frequency and delay differences Δ_i are assumed to be i.i.d. normal random variables. All designs are instantiated with $m = 64$ stages. For S-ArbRO-2 and S-ArbRO-4 PUFs in particular, we choose $q = 32$ and $q = 16$ contributing stages respectively. For 10^5 randomly generated instances of each PUF design, we evaluate the response r to 1000 randomly generated challenges $\mathbf{c}$. Each normalized histogram consists of 101 bins.

The fact that each out of m frequency differences δ of an S-ArbRO-4 PUF always contributes with the same sign to ω is conjectured to be responsible for the bias. Note that the same holds for the two delay differences δ that comprise the last stage of an Arbiter PUF. This results in a milder version of the bias issue, given that the signs of the first $2(m-1)$ delay differences δ are still randomized by the challenge. The bias characteristics of RO Sum and S-ArbRO-2 PUFs are the most favorable, i.e., the behavior of a random oracle is approximated.

Recall that XOR PUFs and Feed-forward PUFs are harder to learn than their ancestors. Their learning curves are not reproduced in this thesis, and we hence refer to the related literature [145, 110, 118, 165]. Another side note is that PUF realizations on an ASIC or an FPGA might be somewhat harder to learn than their ideal abstractions. A first reason is that part of the internal, real-world mechanisms might not have been captured by the variability model $\boldsymbol{\delta}$. A second reason is that both training and testing data are polluted by noise and possibly also environmental changes. However, a *temporal majority vote* (TMV) [37] can to some extent 'undo' the noisiness. As formalized in (3.5) for an odd number of votes q, each challenge $\mathbf{c}$ is then evaluated q times, and the most frequently occurring value of the corresponding response bit r is retained.

$$r = \begin{cases} 1, & \text{if } \hat{r}^{(1)} + \hat{r}^{(2)} + \ldots + \hat{r}^{(q)} > \dfrac{q-1}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

As has been suggested earlier on, the ability to construct a predictive model does not necessarily result in a security threat. The use of a strong PUF can still be considered in multi-component systems where access to the CRPs is highly restricted. The most notable inclusion is PUF-based key generation. In Section 3.2, we therefore still aim to quantify the min-entropy that can be provided by the given strong PUF designs.

3.1.5 An Accurate Variability–Noise Model

In practice, the previously discussed PUFs produce response bits r_i that are not perfectly reproducible. So as to provide insights into this phenomenon, we describe an accurate and well-validated variability–noise model [157, 112, 109]. We will frequently use this model throughout the remainder of this thesis, given that it can be applied to many PUF designs, especially when each individual response bit r_i is generated through the comparison of two nominally identical signals. Prior to the elucidation of further details, a warning against the use of oversimplified models is provided.

Naive, Homogeneous

For a manufactured PUF, the reproducibility is generally observed to differ considerably among the response bits r_i. While some response bits r_i are relatively stable, others are relatively unstable. A naive, homogeneous model might nevertheless assume that all response bits r_i are equally reproducible, given that this simplifies the mathematical analysis of error-correction methods for PUFs. The average-case behavior of several methods has been approximated this way [21, 22, 103, 88], but care is obviously advised. In this thesis, we rely on a more accurate, heterogeneous model instead.

Accurate, Heterogeneous

We describe an accurate, heterogeneous variability–noise model that was first introduced by Škorić et al. [157], and later refined by Maes et al. [112, 109] so as to account for bias. There are earlier uses in the related field of human biometrics [106]. There are numerous works in literature, including our aforementioned IEEE papers, where at least part of the model is experimentally validated. The most complete validation is published by Maes [109]. The cornerstone of the model is that the output variables of a highly complicated physical process tend to be normally distributed, according to the *central limit theorem.* The contributions of both variability and noise are the result of complex underlying dynamics and are hence approximated this way. It is further assumed that both effects are independent and additive. For ease of notation, we assume that the evaluation of a given challenge $\mathbf{c}$ results in a single response bit r only.

The model consists of a parameterized probability distribution for the j-th evaluation of response bits $\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_\lambda$, given a list of corresponding challenges $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_\lambda$. Two random variables are incorporated in (3.6). First, there are the variability contributions Ω_{var}, which are defined with respect to the IC's reference environment. For example, reference values for the external temperature T and the supply voltage V_S could be 25 °C and 1.2 V respectively. A random variate $\boldsymbol{\omega}_{\text{var}} \in \mathbb{R}^\lambda$ is drawn only once for any given PUF. The off-diagonal elements of the covariance matrix $\boldsymbol{\Sigma}_{\text{var}}$ account for spatial and functional correlations. For example, the respective responses r_1 and r_2 of two neighboring cells of an SRAM PUF might tend to be equal and could hence exhibit a positive covariance, i.e., $\Sigma_{\text{var},1,2} > 0$. Similarly for Arbiter PUFs, the responses r_1 and r_2 to challenges $\mathbf{c}_1$ and $\mathbf{c}_2$ that differ in their last bit c_m exclusively, exhibit a negative covariance, i.e., $\Sigma_{\text{var},1,2} < 0$.

$$\forall i \in [1, \lambda], \forall j \in [1, \infty), \hat{r}_i^{(j)} = \begin{cases} 1, & \text{if } \left(\omega_{\text{var},i} + \hat{\omega}_{\text{noise},i}^{(j)}\right) > \omega_{\text{thres}}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

$$\text{where } \Omega_{\text{var}} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_{\text{var}}) \text{ and } \hat{\Omega}_{\text{noise}}^{(j)} \sim N\left(\boldsymbol{\mu}_{\text{noise}}^{(j)}, \boldsymbol{\Sigma}_{\text{noise}}^{(j)}\right).$$

The second random variable $\hat{\Omega}_{\text{noise}}^{(j)}$ comprises the contributions of additive noise. Its distribution depends on the environment of the IC at the given time j. Note that the properties of noise sources inside the IC might be influenced by the environment, e.g., Johnson-Nyquist noise in metallic wires and transistors increases with the temperature T. A random variate $\hat{\boldsymbol{\omega}}_{\text{noise}} \in \mathbb{R}^\lambda$ is drawn for each evaluation j of the response $\hat{\mathbf{r}}$. If the environment at time j happens to be identical to the reference, then the offset $\boldsymbol{\mu}_{\text{noise}} = \mathbf{0}$. We often assume i.i.d. contributions of the noise, i.e., $\hat{\Omega}_{\text{noise}} \sim N\big(\mu_{\text{noise}}\,\mathbf{1}, \sigma^2_{\text{noise}}\,\mathbf{I}_\lambda\big)$. In the ideal case, the threshold ω_{thres} is zero, but nonzero values can incorporate bias, i.e., an imbalance between the expected number of 0s and 1s.

Every model has limitations, and also here, certain physical effects might not have been incorporated adequately. Nevertheless, the chosen model provides an excellent trade-off between accuracy and mathematical complexity. Moreover, given that in related literature, the mathematical analysis of error-correction methods for PUFs is often reliant on the homogeneous model, the use of a heterogeneous model entails an improvement in terms of accuracy.

3.2 Min-Entropy

PUFs are most frequently used for the generation of a device-unique secret key $\mathbf{k} \in \{0,1\}^\kappa$. Unfortunately, concatenating responses $\mathbf{r}_1, \mathbf{r}_2, \ldots$ to a list of publicly known challenges $\mathbf{c}_1, \mathbf{c}_2, \ldots$ does not immediately result in a secret key $\mathbf{k}$. Apart from the noisiness, the concatenated string $\mathbf{x} \in \{0,1\}^\lambda$ is most likely non-uniformly distributed. As will be discussed in Chapter 4, a fuzzy extractor [39] provides an *information-theoretically secure* mechanism to transform $\mathbf{x}$ into a stable secret key $\mathbf{k}$. However, in order to guarantee that K is indeed uniformly distributed over $\{0,1\}^\kappa$, a tight lower bound on the min-entropy of X needs to be derived.

3.2.1 Overview

Unfortunately, developing proven solutions to the previously stated problem might very well be infeasible for any silicon PUF [110]. Under the assumption of an ideally manufactured design though, it is fairly trivial for several weak PUFs to express the min-entropy of X in an exact manner, which is even better than a lower bound. Our newly developed theory focuses on the more difficult case of a strong PUF instead; tight upper bounds on the min-entropy of X are derived for several designs. Although upper bounds are not usable for the provably secure instantiation of a fuzzy extractor, it nevertheless results in novel insights. Our theory demonstrates that designers of PUF-based systems are often too optimistic about the min-entropy provided.

Weak PUFs

Several weak PUF designs consists of an array of identically designed cells that each produce a single bit r. As exemplified in Section 3.1.2, this includes the SRAM PUF [98, 53, 68] and other memory-based designs [160, 113, 97, 156], as well as the RO PUF of Yu et al. [183]. The most prominent entropy-degrading effects for such PUFs are bias and spatial correlations. Bias comprehends an imbalance between the expected number of 0s and 1s. Spatial correlations implicate that neighboring cells might influence each other.

The latter non-uniformities are frequently observed for real-world PUFs [81, 89, 113, 104, 7], i.e., designs that are fabricated on either an ASIC or an FPGA. Although a tight lower bound on the min-entropy of X cannot be evaluated in a proven manner, the derivation of an upper bound tends to be a more feasible alternative. To the best of our knowledge, feeding response bits into a lossless compression algorithm is the only tool available so far [70, 104]. The average number of retained bits converges to an upper bound on the min-entropy. Katzenbeisser et al. [81] apply this technique to all six PUFs that comprise the 65 nm CMOS ASIC of the European research project UNIQUE [89].

Under the assumption of an ideally manufactured SRAM-like PUF, the concatenated response $\mathbf{x} \in \{0,1\}^{\lambda}$ would be uniformly distributed, i.e., $\mathbb{H}_{\infty}(X) = \lambda$. Statistical tests that detect non-uniformities can indicate to which extent such an assumption is realistic. NIST specified a test suite [147] that is frequently used worldwide. However, tests are designed to operate on a one-dimensional bitstream. This complies with a TRNG, for example, but for two-dimensional SRAM-like PUFs it is somewhat less suitable. Concatenating either rows or columns into a one-dimensional bitstream may obfuscate spatial correlations in the perpendicular direction. The latter remark also applies to lossless compression algorithms that operate on a one-dimensional bitstream.

Another uniformity metric that is popular for PUFs in particular is the *inter-device distance*. Given two PUF-enabled devices v_1 and v_2, the Hamming distance between the respective responses $\mathbf{r}_1, \mathbf{r}_2 \in \{0,1\}^{\eta}$ to an identical challenge $\mathbf{c}$ is averaged according to (3.7). Note that for SRAM-like PUFs, an exhaustive evaluation of all challenges $\mathbf{c} \in \mathcal{C}$ is within reach. For an ideal PUF, which behaves as a random oracle according to Definition 1, the random variable D_{inter} exhibits a binomial distribution $B(\eta, 1/2)$ with respect to the infinite set of device pairs. It is a common practice to tabulate both the sample mean and the sample standard deviation of D_{inter} [110], as can be computed for a limited number of manufactured devices. The obtained values should be compared against the ideal values $\eta/2$ and $\sqrt{\eta}/2$ respectively; differences can be attributed to bias and correlations respectively. To enable comparison among different PUF architectures, one can divide the inter-device distance d_{inter} by the length η and hence normalize the codomain to $[0,1]$.

$$\begin{gathered} d_{\text{inter}}(v_1, v_2) \overset{\text{def}}{=} \mathbb{E}_{\mathbf{c} \leftarrow C}\big[\mathsf{HD}(\mathbf{r}_1, \mathbf{r}_2)\big], \\ \text{with } \forall i \in \{1,2\}, \mathbf{r}_i \leftarrow \mathsf{PUF}(\mathbf{c}, \boldsymbol{\delta}_i) \text{ and } \boldsymbol{\delta}_i \leftarrow \Delta. \end{gathered} \tag{3.7}$$

Recall that, in practice, SRAM-like PUFs are often observed to exhibit bias and/or spatial correlations. In order to estimate the min-entropy of X, one could define a non-uniform distribution that closely resembles the experimental data according to statistical tests. Consider for example a scenario where spatial correlations seem to be absent, but there is a bias that remains approximately constant with respect to the spatial coordinates of the two-dimensional array. Formally, one could then assume that response bits X_i, with $i \in [1, \lambda]$, are i.i.d. and that $\mathbb{P}(X_i = 1) = p_{\text{bias}}$, where constant $p_{\text{bias}} \in [0,1]$. The min-entropy is then $\mathbb{H}_\infty(X) = -\lambda \log_2(\max(p_{\text{bias}}, 1 - p_{\text{bias}}))$. Although this again comprehends an unproven technique, its usage is worthy of consideration due to the lack of proven alternatives.

Katzenbeisser et al. [81] estimate the min-entropy of a single SRAM cell, conditioned on the knowledge of the physically neighboring cell contents. Formally, $\widetilde{\mathbb{H}}_\infty(X_{i,j} | (X_{i-1,j}, X_{i+1,j}, X_{i,j-1}, X_{i,j+1}))$ is determined, where the indices are coordinates in a two-dimensional array. The numerical results, however, are irrelevant for the fuzzy extraction [39] of a PUF-based key, and merely serve as a statistical test for detecting spatial correlations. Recall that the min-entropy of a long multi-bit secret X should be determined.

Not all weak PUFs consist of an array of autonomous cells that each produce a single bit r. Even under the assumption of an ideally manufactured design, the distribution of X is then not necessarily straightforward. Consider for example the RO-based PUFs of Suh and Devadas [161] and Maiti and Schaumont [116]. The analysis of their min-entropy relies on methods that can also be used for strong PUFs, as discussed hereafter.

Strong PUFs

Techniques that work well for SRAM-like PUFs are not necessarily suitable for strong PUFs. When applying the NIST non-uniformity tests and/or a lossless compression algorithm to the concatenated response $\mathbf{x} \in \{0,1\}^\lambda$ of a strong PUF, the challenges $\mathbf{c}_i$ are once again ignored. However, given that challenges $\mathbf{c}_i$

no longer correspond to spatial coordinates, functional correlations among CRPs are completely lost. Katzenbeisser et al. [81] nevertheless observe a considerable compression rate for the 65 nm CMOS Arbiter PUFs that were manufactured in the European research project UNIQUE. This can mainly be attributed to bias though, and hence not to functional correlations.

We are the first to derive tight upper bounds on the min-entropy of X for various strong PUFs, hereby incorporating the highly correlated functional behavior. Although techniques are not necessarily limited thereto, we focus on RO Sum, S-ArbRO-2, S-ArbRO-4, Loop, Arbiter, Arbiter Feed-forward, and XOR PUFs. Three newly developed methods produce bounds via a black-box, gray-box, and white-box approach respectively, i.e., the internal mechanisms of the PUF are disregarded, easily incorporated, and extensively analyzed respectively. A brief summary of each method is given below:

- The black-box approach relies on easy-to-obtain machine learning results in order to evaluate the bound. PUFs simulated in software as well as hardware implementations on either an FPGA or an ASIC can be modeled and are therefore supported. The more efficient the learning algorithm, the tighter the upper bound on the min-entropy of X. It is assumed that challenges C_i are i.i.d. random variables.
- The white-box approach requires mathematical analysis of the PUF internals. We rely on the variability models $\boldsymbol{\delta}$ in order to render this approach feasible, and assume that their elements Δ_i are i.i.d. random variables. Although hardware implementations do not necessarily comply with the latter assumptions, it still reflects the ideal-case behavior of the PUF as intended by its designers. Compared to the black-box approach, bounds on the min-entropy are further improved.
- The gray-box approach represents the middle ground. Just like the white-box approach, PUF internals are incorporated, but no extensive analysis is required. The derivation of the bounds is somewhat similar to the black-box approach. The assumptions of both the black-box and the white-box approach are inherited.

Numerical results for all three methods are summarized in Table 3.1. It demonstrates that designers are often too optimistic about the min-entropy provided. The PUF-based key generator of Van Herrewege et al. [171] does not meet its intended 128-bit security level, for example.

Table 3.1: Numerical results for three newly developed methods that produce upper bounds on the min-entropy $\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda))$ that is provided by a strong PUF. So as to enable comparison among these methods, all data relates to ideally manufactured PUFs and challenges $\mathbf{c}_i$ that are selected independently, randomly, and uniformly from the set of all possible challenges $\mathcal{C}$.

	Black-box	**Gray-box**	**White-box**
Assumptions	independently and randomly selected challenges /	independently and randomly selected challenges ideally manufactured PUF	/ ideally manufactured PUF
RO Sum PUF with $m = 64$	≈ 248 bit if $\lambda = 500$ ≈ 381 bit if $\lambda = 1000$ ≈ 493 bit if $\lambda = 1500$	≈ 221 bit if $\lambda = 500$ ≈ 303 bit if $\lambda = 1000$ ≈ 360 bit if $\lambda = 1500$	≈ 193 bit
Loop PUF with $m = 64$			
S-ArbRO-2 PUF with $m = 64$ and $q = 32$			?
S-ArbRO-4 PUF with $m = 64$ and $q = 16$			64 bit
Arbiter PUF with $k = 64$			≈ 197 bit
Feed-forward Arbiter PUF with $k = 64$	?	?	
RO Sum XOR PUF with $m = 64$ and $q = 2$	?	≈ 416 bit if $\lambda = 500$ ≈ 597 bit if $\lambda = 1000$ ≈ 703 bit if $\lambda = 1500$	≈ 387 bit

3.2.2 Black-Box Bounds

A first proven method to evaluate upper bounds on the min-entropy of X adopts a black-box approach. We rely on machine learning results exclusively, regardless of the internals of the PUF circuit. The method can therefore be applied to simulated PUFs and/or FPGA/ASIC implementations. The performance of machine learning techniques is always suboptimal in practice, i.e., $\mathsf{Accuracy}(g)$ in (3.3) is not the theoretical maximum with respect to a given number of training CRPs g. This implies that an upper bound on the min-entropy of X is the best achievable result, i.e., a lower bound or an exact value is inherently out of reach.

Theory

The min-entropy of X obviously depends on the given list of hardcoded challenges, i.e., $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_\lambda$. It is our goal, however, to assess the quality of a given PUF design in a universal manner, regardless of the implementation details of a particular challenge generator. We therefore resort to the conditional min-entropy in (3.8), where the success rate of an attacker's best guess is averaged over the set of all possible challenge generators. It is fair to assume that challenges C_i are i.i.d. uniform random variables. Challenge generators are typically designed to produce pseudorandom bitstreams whose properties approximate the properties of truly random sequences.

$$\begin{aligned}\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \dots, C_\lambda)) = -\log_2\Bigg(\mathbb{E}_{\mathbf{c}_1 \leftarrow C_1}\Big[\mathbb{E}_{\mathbf{c}_2 \leftarrow C_2}\Big[\dots \mathbb{E}_{\mathbf{c}_\lambda \leftarrow C_\lambda}\Big[\\ \max_{x \in \mathcal{X}} \mathbb{P}\big((X = \mathbf{x})|((C_1 = \mathbf{c}_1) \cap (C_2 = \mathbf{c}_2) \cap \dots \cap (C_\lambda = \mathbf{c}_\lambda))\big)\Big] \dots\Big]\Big]\Bigg).\end{aligned} \tag{3.8}$$

Consider a possibly randomized procedure $\hat{\mathbf{x}} \leftarrow \mathsf{Guess}(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_\lambda, \mathbf{n})$, where N is a uniformly distributed seed, used by the attacker to make an educated guess of the concatenated response $\mathbf{x}$. An upper bound on its success rate is defined by the optimal guessing procedure, which deterministically returns the most likely value of X with respect to the given set of challenges. A performance evaluation of a non-ideal realization of Guess hence results in an upper bound on the min-entropy that is produced by the PUF, as shown in (3.9).

$$\begin{gathered}\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda)) \leq -\log_2\Bigg(\mathbb{E}_{\mathbf{c}_1 \leftarrow C_1}\Big[\mathbb{E}_{\mathbf{c}_2 \leftarrow C_2}\Big[\\ \ldots \mathbb{E}_{\mathbf{c}_\lambda \leftarrow C_\lambda}\Big[\mathbb{E}_{\mathbf{n} \leftarrow N}\Big[\mathbb{P}\big((X = \mathsf{Guess}(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\lambda, \mathbf{n}))| \\ ((C_1 = \mathbf{c}_1) \cap (C_2 = \mathbf{c}_2) \cap \ldots \cap (C_\lambda = \mathbf{c}_\lambda))\big)\Big]\Big] \ldots\Big]\Big]\Bigg).\end{gathered} \tag{3.9}$$

Consider the following realization of Guess, making use of a machine learning algorithm. The first g bits of the concatenated response $\mathbf{x}$, where $g \in [1, \lambda]$ is a constant, are guessed at random, which involves sampling a random variable U that is uniformly distributed over $\{0, 1\}^g$ whenever Guess is evaluated. The success probability thereof is $(1/2)^g$, regardless of the distribution of X. The resulting set of g hopefully correct CRPs comprehends the input of a possibly randomized training algorithm that outputs a predictive model of the PUF. A procedure Predict can then provide the last $(\lambda - g)$ bits of prospective response $\hat{\mathbf{x}}$. The upper bound on the min-entropy produced by the PUF reduces to (3.10).

$$\begin{gathered}\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda)) \leq -\log_2\Bigg(\mathbb{E}_{\mathbf{c}_1 \leftarrow C_1}\Big[\\ \mathbb{E}_{\mathbf{c}_2 \leftarrow C_2}\Big[\ldots \mathbb{E}_{\mathbf{c}_\lambda \leftarrow C_\lambda}\Big[\mathbb{E}_{\mathbf{n} \leftarrow N}\Big[\mathbb{E}_{\boldsymbol{\delta} \leftarrow \Delta}\Big[\Big(\frac{1}{2}\Big)^g \\ \prod_{i=g+1}^{\lambda} \mathbb{P}(X_i = \mathsf{Predict}(\mathbf{c}_i; \mathbf{c}_1, x_1, \mathbf{c}_2, x_2, \ldots, \mathbf{c}_g, x_g, \mathbf{n}) \\ \Big]\Big]\Big] \ldots\Big]\Big]\Bigg), \quad \text{with } \forall i \in [1, \lambda], x_i \leftarrow \mathsf{PUF}(\mathbf{c}_i, \boldsymbol{\delta}).\end{gathered} \tag{3.10}$$

Given that challenges C_i are independent, the modeling accuracy as defined in (3.3) reappears. We hence obtain the bound in (3.11). For a given response length λ, the value of g that results in the tightest bound is of primary interest. We emphasize that the min-entropy of a sizable multi-bit response X is being bounded here, as this has immediate application to a subsequent fuzzy extractor [39]. Katzenbeisser et al. [81] and Maes [110] previously considered a less complicated but also less useful scenario where the min-entropy of a single response bit is bounded, i.e., $\tilde{\mathbb{H}}_\infty(R_{g+1}|(C_1, R_1, \ldots, C_g, R_g, C_{g+1})) \leq -\log_2(\mathsf{Accuracy}(g))$.

$$\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda)) \leq g - (\lambda - g)\log_2(\mathsf{Accuracy}(g)). \tag{3.11}$$

Numerical Results

Figure 3.7 plots upper bounds on the min-entropy of ideal, simulated Arbiter PUFs. This result clearly undermines the security of the 128-bit key generator of Van Herrewege et al. [171], which relies on an Arbiter PUF with $m = 64$ stages as well. Their 1785-bit response X was assumed to be uniformly distributed, while our bound implies that the min-entropy does not exceed 550 bits. Note that we consider simulated PUFs, which differs from the 65 nm CMOS ASIC that was manufactured during the research project UNIQUE. Nevertheless, the conclusion hardly changes when plugging in the learning curve of the latter [110]. Upper bounds on the min-entropy of ideal RO Sum PUFs, ideal Loop PUFs, ideal S-ArbRO-2 PUFs, and ideal S-ArbRO-4 PUFs are approximately the same, given that the learning curves in Figure 3.5 approximately coincide.

3.2.3 Gray-Box Bounds

Unlike the previously derived black-box bounds, the evaluation of gray-box bounds requires knowledge of the PUF internals. We limit ourselves to ideally manufactured PUFs, although it might be feasible to incorporate certain non-idealities that occur in hardware implementations. Frequencies F_i and delay differences Δ_i are assumed to be i.i.d. normal random variables. Furthermore, we assume that frequencies can be measured with infinite precision, which differs from the digital counters used in hardware.

Finally, it is assumed that the concatenated response $\mathbf{x}$ of a PUF can be evaluated in a noiseless manner, which is not necessarily a far-fetched assumption in practice. As will be discussed in Chapter 4, a reference value of $\mathbf{x}$, which in turn determines the value of the key $\mathbf{k}$, is usually drawn during the enrollment of a PUF-enabled device, where the environment can be ultra-stable. Environmental changes that later occur during the in-the-field deployment of a device hence do not affect the min-entropy of X and K. Moreover, TMV as previously formalized in (3.5) can suppress noise during the enrollment.

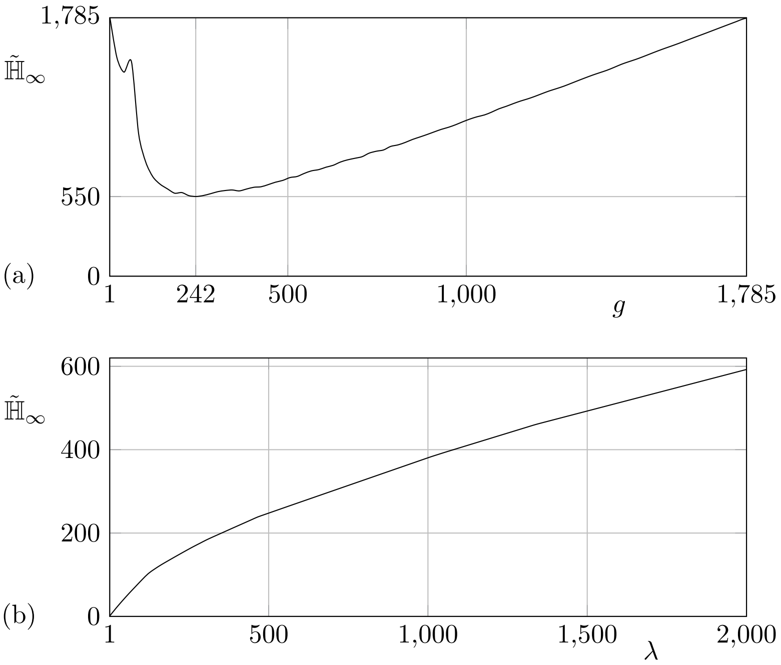


Figure 3.7: An upper bound on the min-entropy $\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda))$ of a simulated Arbiter PUF with $m = 64$ stages. (a) For $\lambda = 1785$ challenges, the best bound is approximately 550 bits, hereby using $g = 242$ CRPs as training set. (b) For each response length λ, the optimal number of training CRPs g can be selected so as to obtain the best possible bound.

Theory

The variability model of each strong PUF in Section 3.1.3 is instantiated by a list of independent, zero-centered normal random variables $\Delta_i \sim N(0, \sigma_i^2)$, with $i \in [1, \eta]$. As illustrated in Figure 3.8, we partition the domain of each random variable Δ_i into 2^q equiprobable intervals, where $q \in \{1, 2, 3, \ldots\}$ is a constant. Within each interval $[\alpha_{i,j}, \beta_{i,j}]$, with $j \in [1, 2^q]$, an arbitrary center value $\overline{\delta}_{i,j}$ is subsequently selected. We opt for the expected value of Δ_i, conditional on $\alpha_{i,j} < \Delta_i < \beta_{i,j}$. As elaborated in (3.12), the expected value of a truncated normal distribution can be computed easily.

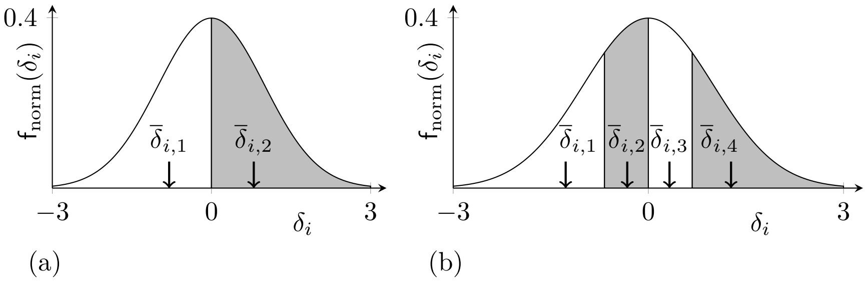


Figure 3.8: Partitioning the domain of a standard normal distribution $\Delta_i \sim N(0,1)$ into 2^q equiprobable intervals, with (a) $q = 1$ and (b) $q = 2$. The center value $\overline{\delta}_{i,j}$ of each interval $j \in [1, 2^q]$ is indicated by an arrow.

$$\overline{\delta}_{i,j} = \frac{2^q}{\sigma_i} \int_{\alpha_{i,j}}^{\beta_{i,j}} \delta \, \mathsf{f}_{\text{norm}}\Big(\frac{\delta}{\sigma_i}\Big) \, \mathrm{d}\delta = 2^q \sigma_i \Bigg(\mathsf{f}_{\text{norm}}\Big(\frac{\alpha_{i,j}}{\sigma_i}\Big) - \mathsf{f}_{\text{norm}}\Big(\frac{\beta_{i,j}}{\sigma_i}\Big) \Bigg), \quad \text{with } \alpha_{i,j} = \mathsf{F}^{-1}_{\text{norm}}\Big(\frac{j-1}{2^q}\Big)\sigma_i, \beta_{i,j} = \mathsf{F}^{-1}_{\text{norm}}\Big(\frac{j}{2^q}\Big)\sigma_i, \text{and } j \in [1, 2^q]. \tag{3.12}$$

Suppose that we modify a randomly instantiated PUF such that each δ_i is replaced by the center value $\overline{\delta}_{i,j}$ that belongs to the same interval. The respective accuracy as defined in (3.13) can be evaluated easily via Monte Carlo experiments. Normally, $\mathsf{Accuracy}(q)$ increases monotonically with the quantization parameter q.

$$\mathsf{Accuracy}(q) = \mathbb{E}_{\mathbf{c} \leftarrow C}\Big[\mathbb{E}_{\boldsymbol{\delta} \leftarrow \Delta}\Big[\mathbb{P}\big(\mathsf{PUF}(\mathbf{c}, \boldsymbol{\delta}) = \mathsf{PUF}(\mathbf{c}, \overline{\boldsymbol{\delta}})\big)\Big]\Big], \quad \text{with } \forall i = [1, m], \exists j = [1, 2^q] \text{ such that } \delta_i, \overline{\delta}_{i,j} \in [\alpha_{i,j}, \beta_{i,j}]. \tag{3.13}$$

Similar to our derivation of the black-box bounds, we consider a randomized procedure $\hat{\mathbf{x}} \leftarrow \mathsf{Guess}(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\lambda, \mathbf{n})$, where N is a uniformly distributed seed, used by the attacker to make an educated guess of the concatenated response $\mathbf{x}$. The following two-step instantiation of Guess is adopted. First, an interval $[\alpha_j, \beta_j]$ is selected uniformly at random for each random variable Δ_i. Hopefully, $\alpha_j \leq \delta_i \leq \beta_j$ so that center value $\overline{\delta}_j$ could serve as an accurate estimate. The probability that all intervals are guessed correctly equals $(1/2^{m\,q})$. Second, the variability model of the PUF instantiated with all m center values allows for the evaluation of response bits.

Given that challenges are independent, we obtain the upper bound on the min-entropy given by (3.14). For a given response length λ, the value of q that results in the tightest bound is of primary interest.

$$\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda)) \leq m\,q - \lambda \log_2(\mathsf{Accuracy}(q)). \tag{3.14}$$

Numerical Results

Figure 3.9 plots upper bounds on the min-entropy of ideal, simulated RO Sum PUFs. Numerical results for ideal Arbiter PUFs, ideal loop PUFs, ideal S-ArbRO-2 PUFs, and ideal S-ArbRO-4 PUFs are approximately the same.

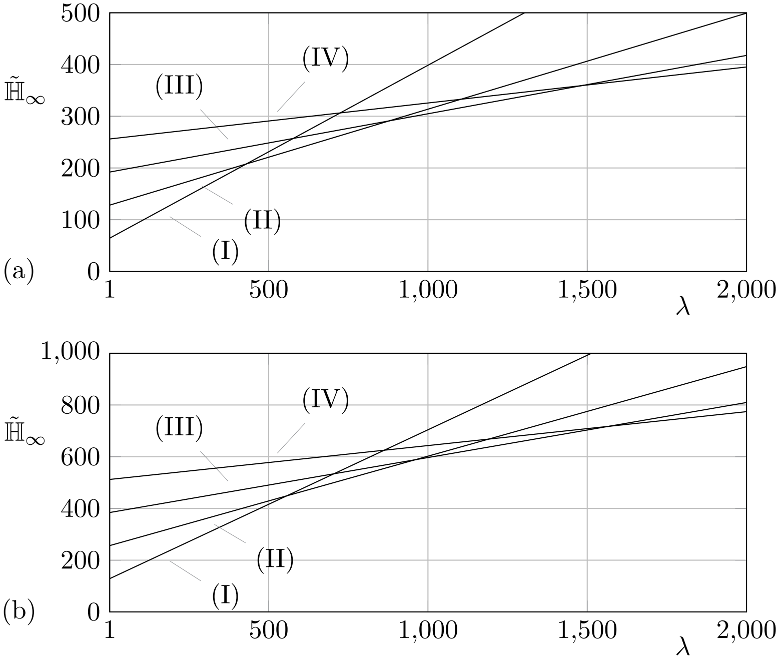


Figure 3.9: Gray-box bounds on the min-entropy $\tilde{\mathbb{H}}_\infty(X|(C_1, C_2, \ldots, C_\lambda))$ of (a) an RO Sum PUF with $m = 64$ stages, and (b) an RO Sum XOR PUF with 2 chains of $m = 64$ stages each. The domain of each variable Δ_i is subdivided into 2^q equiprobable intervals with (I) $q = 1$, (II) $q = 2$, (III) $q = 3$, and (IV) $q = 4$. Note that the bounds are linear functions of the number of challenges λ.

3.2.4 White-Box Bounds

The white-box approach requires extensive analysis of the PUF internals, but can result in tighter bounds and improved insights. We derive upper bounds on the min-entropy of the gigantesque $|\mathcal{C}|$-bit concatenated response X, obtained by evaluating all possible challenges $\mathbf{c}$. The bound is hence valid for all possible challenge generators, typically producing a small subset of $\mathcal{C}$ only. The min-entropy of a random variable, in this case X, can only decrease or remain equal when part of its bits are discarded. As for the gray-box bounds, we limit ourselves to ideally manufactured PUFs.

Weak, RO-Based PUFs

The response bits r of the RO-based PUF of Suh and Devadas [161] are clearly correlated. For example, knowledge of $f_1 < f_2$ and $f_2 < f_3$ implies $f_1 < f_3$. Assume that the response bits r to all challenges $\mathbf{c} \in \mathcal{C}$ are concatenated into a multi-bit secret $\mathbf{x}$ of length $m(m-1)/2$. The authors conclude that the min-entropy is $\mathbb{H}_\infty(X) = \log_2(m!)$, given that all $m!$ frequency orderings, i.e., permutations, are equally likely. For a challenge generator that produces a subset of $\mathcal{C}$ only, the previously derived expression is to be considered as an upper bound, i.e., $\mathbb{H}_\infty(X) \leq \log_2(m!)$.

Similarly, the RO-based PUF of Maiti and Schaumont [116] generates a secret response $\mathbf{x}$ of length $(m-1)$. In contradiction to the authors' claim, the response bits are not independent. This has been pointed out by Yin and Qu [181] through an intuitive counterexample with $m = 3$ ROs. We considerably extend their analysis by evaluating the min-entropy as a function of m. Consider the following Monte Carlo experiment on small-scale PUFs. For any given $m \in [2, 20]$, we evaluate the $(m-1)$-bit response $\mathbf{x}$ of 10^5 randomly generated PUF instances. Test results indicate that the alternating sequences $\mathbf{x} = \begin{pmatrix} 0 & 1 & 0 & 1 & \ldots \end{pmatrix}$ and $\mathbf{x} = \begin{pmatrix} 1 & 0 & 1 & 0 & \ldots \end{pmatrix}$ are the most likely to occur, which corresponds to $f_1 < f_2 > f_3 < \ldots$ and $f_1 > f_2 < f_3 > \ldots$ respectively. The min-entropy of X is hence upper-bounded as shown in (3.15), regardless of whether m is small or large. We do not formally exclude that another outcome is more likely to occur and therefore claim to have derived an upper bound rather than an exact result.

$$\mathbb{H}_\infty(X) \leq -\log_2\Big(\mathbb{P}\big(X = \begin{pmatrix} 0 & 1 & 0 & 1 & \ldots \end{pmatrix}\big)\Big). \tag{3.15}$$

Theory on Euler numbers and alternating permutations [159] allows for an exact evaluation of (3.15). We obtain (3.16), where $2\,\mathsf{eul}(m)$ equals the number of alternating permutations of size m. There exist more efficient methods to compute the given recurrence relation for $\mathsf{eul}(m)$, but we do not further elaborate this matter.

$$\mathbb{H}_\infty(X) \leq \log_2\left(\frac{m!}{\mathsf{eul}(m)}\right),$$
$$\text{with } \mathsf{eul}(m) = \begin{cases} \displaystyle\sum_{i=0}^{m-1} \binom{m-1}{k} \mathsf{eul}(i)\mathsf{eul}(m-1-i) & \text{if } m \geq 2, \\ 1 & \text{if } m \in \{0,1\}. \end{cases} \tag{3.16}$$

Figure 3.10 plots the min-entropy of both weak, RO-based PUFs. It can be seen that the incorrect result $\mathbb{H}_\infty(X) = (m-1)$ of Maiti and Schaumont [116] is overly optimistic.

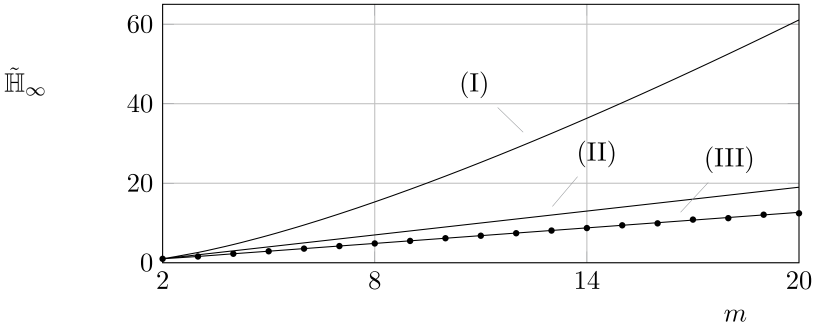


Figure 3.10: The min-entropy of two weak, RO-based PUFs that consist of m oscillators. (I) The exact result $\mathbb{H}_\infty(X) = \log_2(m!)$, which holds for the design of Suh and Devadas [161]. (II) The invalid result $\mathbb{H}_\infty(X) = (m-1)$ of Maiti and Schaumont [116]. (III) Our newly derived upper bound $\mathbb{H}_\infty(X) = \log_2(m!/\mathsf{eul}(m))$ shows a considerable discrepancy with the latter result. Each dot corresponds to a Monte-Carlo experiment of size 10^5 that verifies the correctness of (3.16).

RO Sum PUFs

We carry-out the following Monte Carlo experiment on small-scale RO Sum PUFs. For a given $m \in [1, 5]$, i.e., the number of stages, we evaluate the 2^m-bit concatenated response $\mathbf{x}$ of 10^6 randomly generated PUF instances. Test results indicate that the $2m$ outcomes where exactly one δ_i dominates are the most likely to occur. For instance, $\delta_1 > |\delta_2| + |\delta_3| + \ldots + |\delta_m|$, degenerating the thresholding procedure to $\gamma_1\, \delta_1 \lessgtr 0$. The min-entropy of X is hence upper-bounded as shown in (3.17), regardless of whether m is small or large. For ease of notation, the pairwise frequency differences are assumed to be standard normal random variables, i.e., $\Delta_i \sim N(0, 1)$, with $i \in [1, m]$. The standard deviation can be chosen arbitrarily due to thresholding with 0. As before, we do not formally exclude that another outcome is more likely to occur and therefore claim to have derived a bound rather than an exact result.

$$\begin{gathered}\mathbb{H}_\infty(X) \leq -\log_2\Bigg(\mathbb{P}\Bigg(\Delta_1 > \sum_{i=2}^{m} |\Delta_i|\Bigg)\Bigg) = \\ -\log_2\Bigg(\int_0^\infty \mathsf{f}_{\mathrm{norm}}(\delta_1) \int_0^{\delta_1} 2\mathsf{f}_{\mathrm{norm}}(\delta_2) \int_0^{\delta_1-\delta_2} 2\mathsf{f}_{\mathrm{norm}}(\delta_3) \\ \ldots \int_0^{\delta_1-\delta_2-\ldots-\delta_{m-1}} 2\,\mathsf{f}_{\mathrm{norm}}(\delta_m)\,\mathrm{d}\delta_m \ldots \mathrm{d}\delta_3\,\mathrm{d}\delta_2\,\mathrm{d}\delta_1\Bigg).\end{gathered} \tag{3.17}$$

Unfortunately, for large m, the nested integrals in (3.17) cannot be evaluated in a convenient numerical or analytical manner. The distribution of the sum of zero-truncated normally distributed random variables is non-trivial. We therefore integrate over a subregion only, which remains consistent with an upper bound on $\mathbb{H}_\infty(X)$. As illustrated in Figure 3.11 for $m = 3$, the complete integration domain for δ_2 to δ_m is bounded by a simplex with vertices $\begin{pmatrix}\delta_1 & 0 & \ldots & 0\end{pmatrix}$ to $\begin{pmatrix}0 & \ldots & 0 & \delta_1\end{pmatrix}$. The most straightforward reduction comprehends integrating in a hypercube with vertex $\begin{pmatrix}\delta_1/(m-1) & \delta_1/(m-1) & \ldots & \delta_1/(m-1)\end{pmatrix}$. The resulting expression in (3.18) can be evaluated easily.

$$\begin{gathered}\mathbb{P}\Bigg(\Delta_1 > \sum_{i=2}^{m} |\Delta_i|\Bigg) \geq \int_0^\infty \mathsf{f}_{\mathrm{norm}}(\delta_1) \\ \Bigg(\mathsf{F}_{\mathrm{norm}}\Big(\frac{\delta_1}{m-1}\Big) - \mathsf{F}_{\mathrm{norm}}\Big(-\frac{\delta_1}{m-1}\Big)\Bigg)^{m-1} \mathrm{d}\delta_1.\end{gathered} \tag{3.18}$$

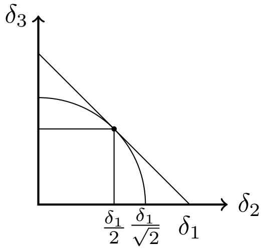


Figure 3.11: Reducing the integration domain of (3.17), illustrated for $m = 3$. Both an inscribed hypercube and an inscribed hypersphere are represented.

A tighter upper bound on the min-entropy $\mathbb{H}_\infty(X)$ is obtained by integrating over the inscribed hypersphere instead. A transformation from Cartesian coordinates $\begin{pmatrix}\delta_2 & \delta_3 & \dots & \delta_m\end{pmatrix}$ to hyperspherical coordinates $\begin{pmatrix}\rho & \phi_1 & \dots & \phi_{m-2}\end{pmatrix}$ is defined in (3.19).

$$\begin{aligned} \delta_2 &= \rho \cos(\phi_1), \\ \delta_3 &= \rho \sin(\phi_1)\cos(\phi_2), \\ &\vdots \\ \delta_{m-1} &= \rho \sin(\phi_1)\dots\sin(\phi_{m-3})\cos(\phi_{m-2}), \\ \delta_m &= \rho \sin(\phi_1)\dots\sin(\phi_{m-3})\sin(\phi_{m-2}), \end{aligned} \tag{3.19}$$

$$\text{with } \rho \in \mathbb{R}^+, \phi_{m-2} \in [0, 2\pi), \text{ and } \phi_1, \cdots, \phi_{m-3} \in [0, \pi).$$

As can be derived from the Jacobian determinant, the volume element for integration is given in (3.20).

$$\begin{aligned} \mathrm{d}\delta_2\,\mathrm{d}\delta_3 \cdots \mathrm{d}\delta_m = \rho^{m-2}\sin^{m-3}(\phi_1)\sin^{m-4}(\phi_2)\cdots \\ \sin(\phi_{m-3})\,\mathrm{d}\rho\,\mathrm{d}\phi_1\,\mathrm{d}\phi_2\cdots\mathrm{d}\phi_{m-2}. \end{aligned} \tag{3.20}$$

We hence obtain (3.21), which is valid for $m \geq 3$.

$$\mathbb{P}\Bigg(\Delta_1 > \sum_{i=2}^{m} |\Delta_i|\Bigg) \geq \int_0^{\pi} \sin^{m-3}(\phi_1)\,\mathrm{d}\phi_1$$

$$\int_0^{\pi} \sin^{m-4}(\phi_2)\,\mathrm{d}\phi_2 \dots \int_0^{\pi} \sin(\phi_{m-3})\,\mathrm{d}\phi_{m-3} \int_0^{2\pi} \mathrm{d}\phi_{m-2} \tag{3.21}$$

$$\int_0^{\infty} \mathsf{f}_{\mathrm{norm}}(\delta_1)\Bigg(\int_0^{\frac{\delta_1}{\sqrt{m-1}}} \frac{\rho^{m-2}}{(\sqrt{2\pi})^{m-1}} \exp\Big(-\frac{\rho^2}{2}\Big)\,\mathrm{d}\rho\Bigg)\mathrm{d}\delta_1.$$

The latter product of integrals can be fully elaborated. First, observe the recurrence relation in (3.22).

$$\int_0^{\pi} \sin^i(\phi)\,\mathrm{d}\phi = \begin{cases} 2 & \text{if } i = 1, \\ \dfrac{\pi}{2} & \text{if } i = 2, \\ \dfrac{i-1}{i} \displaystyle\int_0^{\pi} \sin^{i-2}(\phi)\,\mathrm{d}\phi & \text{if } i > 2. \end{cases} \tag{3.22}$$

For m odd, (3.23) hence holds.

$$\prod_{i=1}^{m-3}\Bigg(\int_0^{\pi} \sin^i(\phi)\,\mathrm{d}\phi\Bigg) = \frac{\pi^{(m-3)/2}}{(\frac{m-3}{2})!}. \tag{3.23}$$

Combined with repeated partial integration, we obtain the bound in (3.24), valid for m odd.

$$\mathbb{H}_\infty(X) \leq -\log_2\Bigg(\frac{1}{2}\Bigg(1 - \sqrt{\frac{m-1}{m}} \sum_{i=0}^{(m-3)/2} \frac{(2i)!}{(i!)^2(4m)^i}\Bigg)\Bigg). \tag{3.24}$$

Fig. 3.12 plots upper bounds on min-entropy using the inscribed hypercube and the inscribed hypersphere respectively. As an additional validation of the latter case, we compute (3.21) and (3.24) with MATLAB and Maple respectively, i.e., the same numerical results are produced by two different tools.

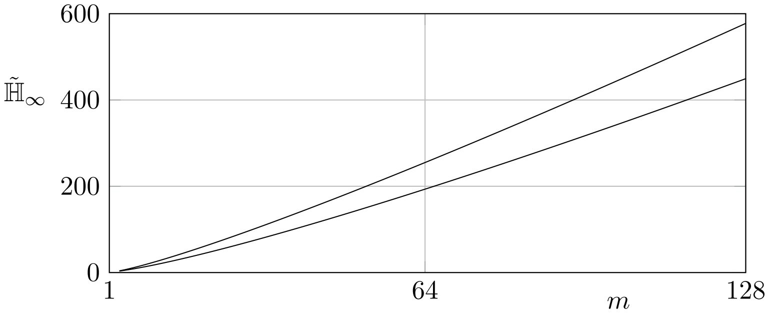


Figure 3.12: Upper bounds on the min-entropy of an ideal RO Sum PUF as a function of the number of stages m. Top and bottom curves correspond to integration via an inscribed hypercube and hypersphere respectively.

Loop PUFs

The mathematical abstractions of an m-stage Loop PUF and an m-stage RO Sum PUF are identical. Therefore, the previously derived upper bound (3.24) still applies. Another result of interest that applies to both PUF designs is proven by Rioul et al. [138]: for a well-chosen set of m challenges, it is possible to extract a uniformly distributed response, i.e., $\mathbb{H}_\infty(X) = m$. To be specific, the challenges should correspond to the rows of an $m \times m$ Hadamard matrix. A remarkable consequence of this derivation is that given $2m$ non-reconfigurable oscillators, an RO Sum PUF and the weak RO-based PUF of Yu et al. [183] are both able to provide an m-bit uniformly distributed response.

Arbiter PUFs

The upper bound in (3.24) on the min-entropy of an RO Sum PUF with $m+1$ stages applies to an Arbiter PUF with m stages equally well. As has been mentioned in Section 2.3, if $\boldsymbol{\delta} = \boldsymbol{\delta}_\star \mathbf{B}$ is a linear transformation of $\Delta_\star \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then $\Delta \sim N(\boldsymbol{\mu}\mathbf{B}, \mathbf{B}^T \boldsymbol{\Sigma} \mathbf{B})$. With $\mathbf{B}$ defined in (3.1) and $\Delta_\star \sim N(\mathbf{0}, \mathbf{I}_{2m})$, it hence holds that $\Delta \sim N(\mathbf{0}, \mathsf{diag}(1/2, 1, \ldots, 1, 1/2))$. As all $m+1$ variables are independent, we evaluate a lower bound on the probability $\mathbb{P}(\Delta_2 > |\Delta_1| + |\Delta_3| + |\Delta_{m+1}|)$ similar to (3.21). Compared to the black-box approach in Figure 3.7, the upper bound on the min-entropy is considerable improved. The bound on the 1785-bit response of an arbiter PUF with $m = 64$ stages is reduced from approximately 550 bits to approximately 197 bits.

Feed-forward Arbiter PUFs

The upper bound on the min-entropy of an Arbiter PUF with m stages applies to its feed-forward variant with an equal number of stages m stages as well, regardless of the number of loops q and the corresponding tap positions. The main insight is that the feed-forward variant covers only a subset of 2^{m-q} out of 2^m transformed challenges $\boldsymbol{\gamma}$, i.e., part of the gigantesque concatenated response $\mathbf{x}$ can be discarded. Despite being more resistant to machine learning attacks [145], the min-entropy hence does not increase, which highlights the superiority of the white-box approach to the black-box approach. This also implies that the feed-forward variant is not very suitable for the fuzzy extraction of a secret key, given that for the same min-entropy, it consumes more resources while providing less stable responses.

S-ArbRO-4 PUFs

There is a trivial upper bound on the min-entropy of an S-ArbRO-4 PUF with m RO pairs, i.e., $\mathbb{H}_\infty(X) \leq m$. This corresponds to the probability that the signs of all m frequency differences δ_i are either all negative or all positive, making concatenated response $\mathbf{x}$ either $\mathbf{0}$ or $\mathbf{1}$. Despite outputting up to $\binom{m/2}{q}2^q$ response bits, the produced min-entropy is hence not larger than for the weak RO-based PUF of Yu et al. [183], given an equal number of $2m$ ROs.

XOR PUFs

Consider all XOR PUFs that condense the responses of q parallel and identically laid-out component PUFs into a single bit, regardless of the chosen strong PUF design, and regardless of whether component challenges are identical [161, 145], permuted [118, 117], or virtually independent [188]. As shown in (3.25), a generic upper bound on the min-entropy $\mathbb{H}_\infty(X_1 \oplus X_2 \oplus \ldots \oplus X_q)$ of an XOR PUF can be derived from an upper bound on the min-entropy $\mathbb{H}_\infty(X_j)$ of its component PUFs.

$$\mathbb{H}_\infty\big(X_j\big) \leq \beta \implies \mathbb{H}_\infty\big(X_1 \oplus X_2 \oplus \ldots \oplus X_q\big) \leq q\,\beta. \tag{3.25}$$

The reasoning for the case of identical component challenges [161, 145] is given in (3.26). A lower bound α on the probability of occurrence of the most likely response $\mathbf{x}$ of a component PUF can be raised to the power of q for the XORed counterpart.

$$\begin{aligned}&\forall \mathbf{x} \in \mathcal{X}, \mathbb{P}\big(X_1 = \mathbf{x}\big) = \mathbb{P}\big(X_2 = \mathbf{x}\big) = \ldots = \mathbb{P}\big(X_q = \mathbf{x}\big) \geq \alpha \\ \implies &\mathbb{P}\left(X_1 \oplus X_2 \oplus \ldots \oplus X_q = \begin{cases}\mathbf{x} & \text{if } q \in \{1,3,5,\ldots\} \\ \mathbf{0} & \text{if } q \in \{2,4,6,\ldots\}\end{cases}\right) \geq \alpha^q.\end{aligned} \tag{3.26}$$

A more general reasoning for permuted [118, 117] and independent [188] component challenges is given in (3.27), with responses $\mathbf{x}$ having length $|\mathcal{C}|$ and $|\mathcal{C}|^q$ respectively.

$$\begin{aligned}&\forall \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q \in \mathcal{X}, \\ &\mathbb{P}\big(X_1 = \mathbf{x}_1\big) = \mathbb{P}\big(X_2 = \mathbf{x}_2\big) = \ldots = \mathbb{P}\big(X_q = \mathbf{x}_q\big) \geq \alpha \\ \implies &\mathbb{P}\big(X_1 \oplus X_2 \oplus \ldots \oplus X_q = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_q\big) \geq \alpha^q.\end{aligned} \tag{3.27}$$

For specific instances of an XOR PUF, a tighter upper bound may be derived. Consider for example an m-stage RO Sum XOR PUF with identical component challenges and q even. Recall that $2m$ responses of an RO Sum PUF were identified to be equally likely and we derived a lower bound α on their identical probability of occurrence in (3.21). There are hence $(2m)^{q/2}$ combinations that produce with probability α^q an XORed response $X_1 \oplus X_2 \oplus \ldots \oplus X_q = \mathbf{0}$ in (3.26) rather than only one. The improved bound is given in (3.28).

$$\begin{aligned}&\mathbb{H}_\infty\big(X_1 \oplus X_2 \oplus \ldots \oplus X_q\big) \leq q\,\beta - \frac{q}{2}\Big(1 + \log_2(m)\Big), \\ &\text{with } \beta \geq -\log_2\Big(\mathbb{P}\big(\Delta_1 > |\Delta_2| + |\Delta_3| + \ldots + |\Delta_m|\big)\Big).\end{aligned} \tag{3.28}$$

In fact, the looser bound in (3.25) already reveals that XOR PUFs are not very suitable for the fuzzy extraction of a secret key. In terms of min-entropy, it is more efficient to omit the XOR operation and let each out of q identically laid-out component PUFs autonomically provide a fraction $1/q$ of the response bits. This also reduces the error rates, which results in further savings.

3.3 Physical Attacks

PUFs were initially praised for their resistance to physical attacks. This in part relies on the intuitive insight that invasion of the IC might permanently alter the behavior of the PUF and hence destroy the secret. Furthermore, in contrast to NVM, the power-off state is presumed to be highly secure because the secret is hidden in nanoscale process variations. However, it would be a stretch to conclude that PUFs offer comfortable security guarantees by default. Side-channel analysis is non-invasive and hence cannot destroy the secret. Likewise, fault attacks may alter the behavior of a PUF temporarily rather than permanently. Finally, invasive techniques do not necessarily damage the elements of the PUF that harvest process variations.

From the early 2010s onwards, i.e., roughly a decade after the first silicon PUFs were manufactured [107, 51], the first successful physical attacks on PUFs were being reported. We summarize the current state-of-the-art. Afterwards, our contribution is elaborated in more detail. We were the first to experimentally demonstrate that noise can be exploited as a side-channel. This fundamentally differs from the side-channel analysis of cryptographic algorithms, where noise is usually a prohibiting factor rather than an exploit. After our work was published, we received word [146] that Rührmair et al. previously pitched the core idea of our attack as part of a single paragraph [145]. The respective novelty of our work is hence to be understood as an extensive elaboration and a first proof of concept, hereby relying on experimental data. Moreover, a newly proposed extension to fault attacks is experimentally validated as well.

3.3.1 Related Literature

Several physical attacks on PUF circuits have been reported. Below, we summarize them according to the primary physical channel that is being exploited. Evidently, all attacks have limitations, e.g., the feasibility of most invasive attacks has only been demonstrated for older technologies, and various countermeasures have been proposed, but we do not discuss these further.

- **Electromagnetic emissions**. First theoretically [122] and later experimentally [121, 120], Merli et al. demonstrated that RO-based PUFs can be prone to electromagnetic analysis. From the IC's frontside [120] or backside [121] and in a semi-invasive manner, oscillator frequencies are captured and response bits are hence revealed. As a side note, Bayon et al. [11] later used a similar exploit for RO-based TRNGs.

- **Photonic emissions**. From the IC's backside, Helfmeier et al. [58] capture photons that are emitted by transistors in saturation. The contents of an SRAM PUF are obtained this way, in a semi-invasive manner. Likewise, Tajik et al. [163] use photons to measure the propagation delay of logic gates. Arbiter PUFs, RO-based PUFs, and other delay-based designs are hence at risk as well.

- **Laser stimulation**. From the IC's backside and in a semi-invasive manner, Nedospasov et al. [124] stimulate the inverters of an SRAM PUF cell with a near-infrared laser. This causes local heating, which in turn triggers an electric current that can be detected through the power supply and hence reveal the cell contents. Likewise, Tajik et al. [164] use a laser to iteratively disable all but one arbiter chains that constitute an XOR PUF. Machine learning of each individual chain eventually results in a predictive model of the XOR PUF. This attack is performed on a rebootable FPGA-like device so that induced faults can be undone. The authors are also able to disable the individual oscillators of an RO-based PUF.

- **Focused ion beam**. Helfmeier et al. [58] either trim or remove transistors with an FIB workstation so that the possibly unknown initial state of an SRAM PUF can be overwritten. If an initial state has been retrieved via other means, this would also allow for the construction of a physical clone. As a side note, the so-called Coating PUF of Tuyls et al. [168] is claimed to be resistant to FIB attacks. This design harvests process variations from a custom-designed coating directly above the metal layers. The authors performed an experimental attack from the IC's frontside that inadvertently implants ions into the coating and hence alters the PUF characteristics.

- **Power consumption**. Becker and Kumar [14] and Rührmair et al. [146] measure power traces in order to model Arbiter and Arbiter XOR PUFs respectively.

- **Remanence Decay**. Zeitouni et al. [191] experimentally recover the initial state of an SRAM PUF via remanence decay. Despite its volatile nature, cell-specific data remanence effects that decay over time are observed to be present.

The primary focus of the aforementioned attacks is the collection or alteration of otherwise inaccessible CRPs. For weak PUFs, this corresponds to a direct characterization of the secret. For strong PUFs, an additional machine learning step might be required for the attack to be successful.

3.3.2 Exploitation of Noise

We describe our work on the exploitation of noise.

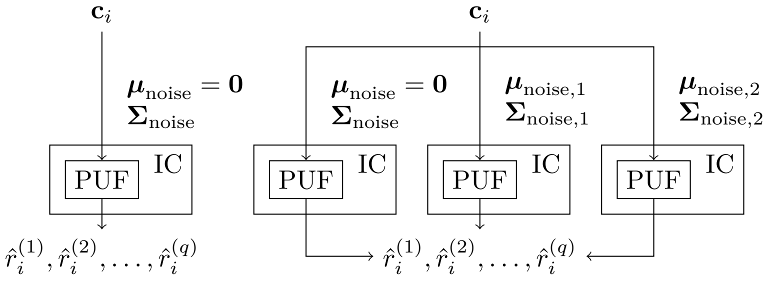
Idea

Given direct access to the interface of a strong PUF, an attacker can obtain the noisy response $\hat{r}_i$ to any challenge $\mathbf{c}_i$. We point out that by measuring the reproducibility of r_i, which involves a repeated evaluation of $\mathbf{c}_i$, the attacker can recover an internal, analog secret that facilitates the learning problem. Consider the heterogeneous variability–noise model of Section 3.1.5 where all response bits are equally affected by noise. As is clear from (3.29), reproducibility measurements allow for an estimate of the internal variability aggregate ω_i. An attacker can hence bypass the threshold operation of a PUF circuit, i.e., a non-linearity that increases the resistance to modeling is eliminated.

$$\mathbb{P}(\hat{R}_i = 1) = \mathsf{F}_{\text{norm}}((\omega_i - \omega_{\text{thres}})/\sigma_{\text{noise}}). \tag{3.29}$$

We experimentally demonstrate the exploit for both Arbiter and RO Sum PUFs. The threshold operation is then the only non-linearity. When bypassed, an attacker can hence construct a predictive model by solving a system of linear equations. For our proof-of-concept attack, randomly chosen challenges $\mathbf{c}_i$ are evaluated q times each, and we retain the ones with rather noisy responses, i.e., $\big(\hat{r}_i^{(1)} + \hat{r}_i^{(2)} + \cdots + \hat{r}_i^{(q)}\big)/q \in [0.1, 0.9]$. The most stable bits are discarded, given that small errors on $\mathbb{P}(\hat{R}_i = 1)$ are then amplified to large errors on the variability component ω_i.

The noisier the PUF, the higher the retention ratio p_{ret} for CRPs, and hence the more efficient the attack. For this reason, we consider the two measurement setups in Figure 3.13. Given that the attacker needs the obtain a predictive model of the PUF under nominal conditions, the most intuitive approach is to gather CRPs with the IC in its nominal environment. However, by changing the environment across the q evaluations, the retention ratio p_{ret} can be increased. So as to maintain a high modeling accuracy under nominal conditions, perturbations are symmetrical with respect to the nominal environment.


$\mathbf{c}_i$
$\boldsymbol{\mu}_{\text{noise}} = \mathbf{0}$
$\boldsymbol{\Sigma}_{\text{noise}}$
PUF
IC
$\hat{r}_i^{(1)}, \hat{r}_i^{(2)}, \ldots, \hat{r}_i^{(q)}$
$\mathbf{c}_i$
$\boldsymbol{\mu}_{\text{noise}} = \mathbf{0}$
$\boldsymbol{\Sigma}_{\text{noise}}$
$\boldsymbol{\mu}_{\text{noise},1}$
$\boldsymbol{\Sigma}_{\text{noise},1}$
$\boldsymbol{\mu}_{\text{noise},2}$
$\boldsymbol{\Sigma}_{\text{noise},2}$
PUF
IC
PUF
IC
PUF
IC
$\hat{r}_i^{(1)}, \hat{r}_i^{(2)}, \ldots, \hat{r}_i^{(q)}$

(a) side-channel (b) fault injection

Figure 3.13: Two measurement setups for the modeling a strong PUF via physical noise. The response r_i to a given challenge $\mathbf{c}_i$ is measured $q > 1$ times. (a) In the first setup, responses are passively recorded with the IC in its nominal environment. Noise is hence exploited as a side-channel. (b) In the second setup, we actively change the IC's environment so that responses are less stable. This classifies as a fault attack, which aims to facilitate the exploitation of noise as a side-channel.

Target Device

Our experiments are performed on a 65 nm CMOS ASIC that was manufactured during the European research project UNIQUE. A considerable bias has previously been reported for its Arbiter PUFs [89, 81, 110]. As is clear from the gate-level schematics [141], an asymmetry in the arbiter element seems to be responsible. A buffering inverter for reading out the response bit r causes one half of an otherwise symmetric latch to have a higher capacitive load. The addition of a dummy inverter would have restored the symmetry and hence improved the bias characteristics. The missing replica is not really a problem for our purposes though. On the contrary, it demonstrates that our newly developed modeling attacks can handle bias.

The ASIC also houses a batch of 4096 identically designed ROs. Each oscillator consists of 80 inverters and one NAND gate, which is able to start and stop the oscillation. For improved testability, the approximate number of oscillations was designed to be reconfigurable. All oscillations stop as soon as the counter of one dedicated RO, which consists of 64 inverters and one NAND gate, reaches a reconfigurable threshold. This feature allows for a flexible trade-off between latency and noise, as experimentally confirmed in Figure 3.14. The longer an RO oscillates, the more stable the measurement of its number of oscillations becomes. As our newly developed attacks exploit noise, it would be convenient to keep the counter values low. However, such testability features are not expected to be a part of a final market product. We hence adopt the perspective of a system provider instead and fix the number of oscillations to approximately 1000.

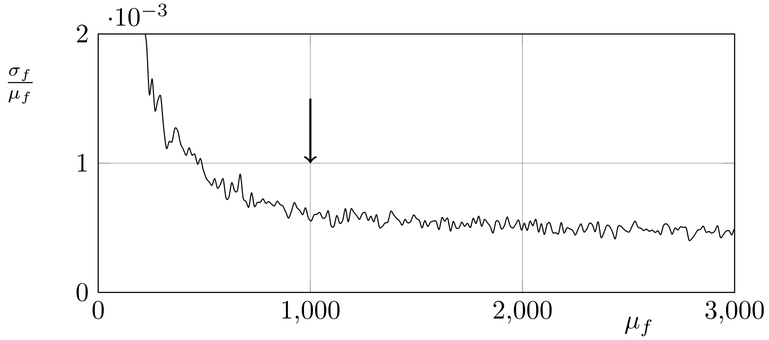


Figure 3.14: The trade-off between latency and noise for the ROs of the 65 nm CMOS ASIC that was designed during the European research project UNIQUE [89, 81, 110]. For various threshold values and for each out of eight ROs, the mean μ_f and the standard deviation σ_f of a sample of 20 measured counter values is computed. For improved visibility, we average the sample means μ_f and the normalized sample standard deviations σ_f/μ_f over all eight ROs before plotting. As indicated by the arrow, we fix the number of oscillations to approximately 1000.

Sensitivity to the Environment

We investigate the sensitivity of our target PUFs to the environment. In this work, the environment is defined by the supply voltage V_S and the outside temperature T. Their nominal values are $V_S = 1.2\,\mathrm{V}$ and $T = 20\,^{\circ}\mathrm{C}$ respectively. Other environmental influences such as electromagnetic interference are not investigated here. For the supply voltage V_S, we perform a sweep from $0.95\,\mathrm{V}$ to $1.45\,\mathrm{V}$ in steps of $0.05\,\mathrm{V}$, while keeping the temperature T at its nominal value. Likewise, we use a TestEquity Half Cube temperature chamber to perform a sweep from $-20\,^{\circ}\mathrm{C}$ to $60\,^{\circ}\mathrm{C}$ in steps of $10\,^{\circ}\mathrm{C}$, while keeping the supply voltage V_S at its nominal value. The ASIC is observed to remain operable and undamaged throughout our experiments. Results for the Arbiter and RO-based PUFs are shown in Figures 3.15 and 3.16 respectively.

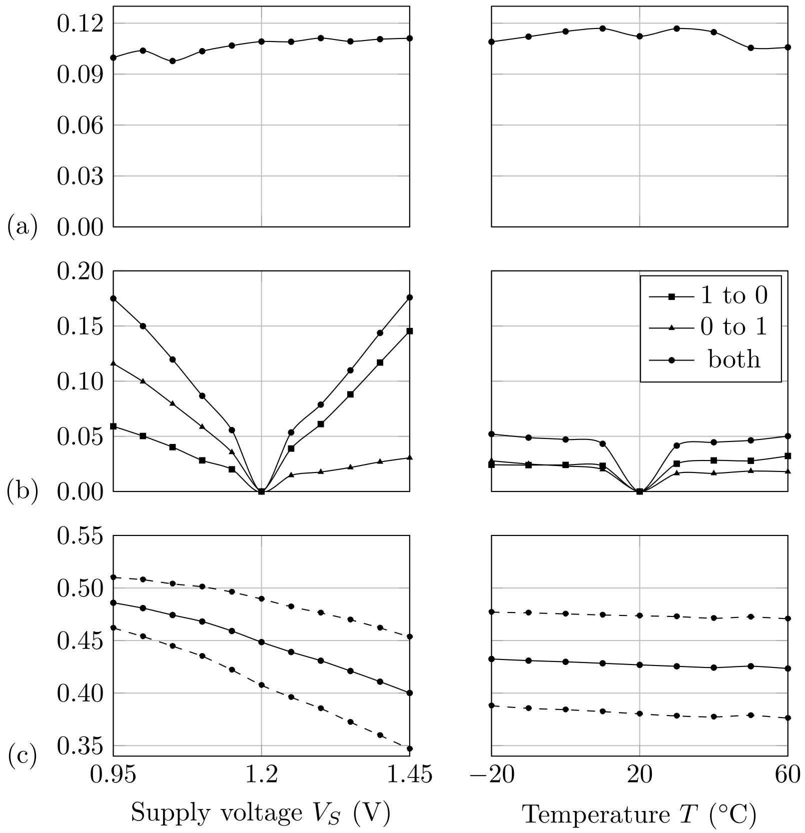


Figure 3.15: The sensitivity of our target Arbiter PUFs to the environment. The responses $\hat{r}_i$ of 32 PUF instances to $\lambda = 2000$ randomly chosen challenges $\mathbf{c}_i$ are evaluated $q = 15$ times each. (a) The fraction p_{ret} of CRPs that is deemed to be noisy, i.e., $\big(\hat{r}_i^{(1)} + \hat{r}_i^{(2)} + \cdots + \hat{r}_i^{(q)}\big)/q \in [0.1, 0.9]$. (b) The fraction of post-TMV responses r_i that flips with respect to the nominal environment. (c) The probability $p_{\text{bias}} = (r_1 + r_2 + \cdots + r_\lambda)/\lambda)$ of the PUF instances: both the sample mean and the ±1 sample standard deviation interval are shown.

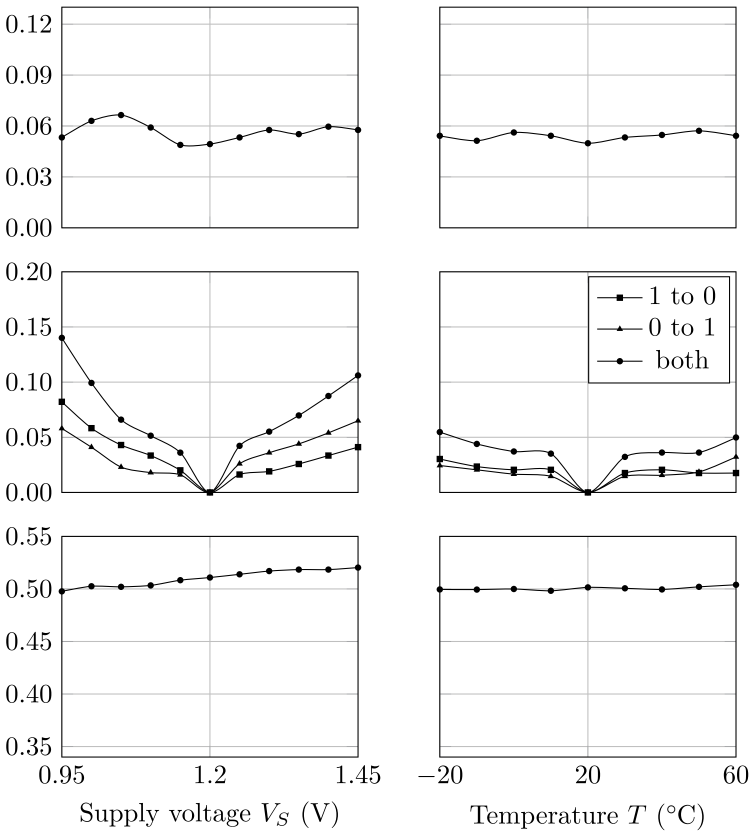


Figure 3.16: The sensitivity of our target ROs to the environment. The responses $\hat{r}_i$ of $\lambda = 2048$ randomly chosen oscillator pairs are evaluates $q = 15$ times each. (a) The fraction p_{ret} of CRPs that is deemed to be noisy, i.e., $\big(\hat{r}_i^{(1)} + \hat{r}_i^{(2)} + \cdots + \hat{r}_i^{(q)}\big)/q \in [0.1, 0.9]$. (b) The fraction of post-TMV responses r_i that flips with respect to the nominal environment. (c) The bias $p_{\text{bias}} = (r_1 + r_2 + \cdots + r_\lambda)/\lambda)$ of the PUF cells.

We conclude that both PUFs are considerably more sensitive to changes of the supply voltage V_S than to changes of the outside temperature T. Therefore, we accelerate our modeling attacks by changing V_S exclusively.

Numerical Results

Tables 3.2 and 3.3 summarize the performance of our noise-based modeling attacks on Arbiter and RO Sum PUFs respectively, where the response $\hat{r}$ to each challenge $\mathbf{c}_i$ is evaluated $q = 3$ times each. The more to the right in either table, the larger the perturbations of the supply voltage V_S. We observe no significant decrease in modeling accuracy with respect to the side-channel case, given that perturbations are chosen to be symmetrical around the nominal value. For the largest perturbations, the retention ratio p_{ret} and hence also the efficiency of our attacks increases with a factor $17.9/7.5 \approx 8.7/3.7 \approx 2.4$ for both Arbiter and RO Sum PUFs.

Even when applying changes to the IC's environment, the retention ratio p_{ret} remains relatively small. Compared to state-of-the-art machine learning attacks, which usually retain all CRPs as training data, more evaluations of an Arbiter or RO Sum PUF are hence required so as to obtain a given modeling accuracy. In the concluding sections of our IEEE publications, we did suggest joining efforts rather than competition though, i.e., the development of a hybrid attack that combines noise measurements with a machine learning algorithm was proposed as further work. It has later been demonstrated by other authors that such a hybrid attack can indeed outperform stand-alone machine learning attacks.

Follow-up Work

We consider three categories of follow-up work:

- First, hybrid attacks that combine noise measurements with a machine learning algorithm have been developed. Kumar and Burleson [95, 96] applied a hybrid attack to both Feed-forward Arbiter PUFs and *Leakage Current*-based PUFs. Furthermore, Becker [13] applied a hybrid attack to Arbiter XOR PUFs that considerably outperforms stand-alone machine learning attacks.
- Second, Becker [12] demonstrated that with the stability information of responses exclusively, i.e., disregarding whether the nominal values are either 1 or 0, an accurate predictive model of an Arbiter PUF can

Table 3.2: The performance of our noise-based modeling attacks with the Arbiter PUFs of the UNIQUE ASIC as target. The responses $\tilde{r}$ to randomly generated challenges $\mathbf{c}$ are evaluated $q = 3$ times each until g unstable CRPs can be retained. Accuracies are computed through a test set of 3000 randomly chosen CRPs that are collected under nominal conditions, i.e., $V_S = 1.20\,\text{V}$. TMV with 15 votes is applied to the test responses. Accuracies are computed for 32 PUF instances: both the sample mean and the ±1 sample standard deviation interval is given.

	Side-channel	Fault injection				
V_S	1.20 V	1.15 V 1.20 V 1.25 V	1.10 V 1.20 V 1.30 V	1.05 V 1.20 V 1.35 V	1.00 V 1.20 V 1.40 V	0.95 V 1.20 V 1.45 V
p_{ret}	7.5%	8.4%	10.3%	12.9%	15.4%	17.9%
100	85.1% ±9.9%	87.4% ±6.9%	88.7% ±3.8%	89.3% ±2.6%	88.3% ±2.5%	87.4% ±3.2%
200	93.6% ±1.5%	93.8% ±1.4%	93.9% ±1.2%	93.7% ±1.0%	93.6% ±1.0%	93.4% ±1.3%
300	94.6% ±1.2%	94.7% ±0.9%	95.0% ±0.7%	95.0% ±0.7%	95.0% ±0.7%	94.8% ±0.7%
400	95.3% ±0.8%	95.3% ±0.7%	95.6% ±0.7%	95.6% ±0.6%	95.6% ±0.5%	95.3% ±0.5%
500	95.6% ±0.6%	95.7% ±0.6%	95.9% ±0.6%	95.7% ±0.6%	95.9% ±0.5%	95.7% ±0.5%
g	Accuracy of predictive model					

still be constructed. This comprehends a novel threat for PUF-based systems that hide response bits but not their respective stabilities from the attacker [171].

- Third, PUFs and PUF-based systems that aim to mitigate noise-related threats have recently been proposed. Consider for example the Multiplexer PUFs of Sahoo et al. [150] and the second so-called lockdown protocol of Yu et al. [188].

Table 3.3: The performance of our noise-based modeling attacks with the RO Sum PUFs of the UNIQUE ASIC as target. The responses $\tilde{r}$ to randomly generated challenges $\mathbf{c}$ are evaluated $q = 3$ times each until g unstable CRPs can be retained. Accuracies are computed through a test set of 5000 randomly chosen CRPs that are collected under nominal conditions, i.e., $V_S = 1.20\,\text{V}$. TMV with 15 votes is applied to the test responses.

	Side-channel	Fault injection				
V_S	1.20 V	1.15 V 1.20 V 1.25 V	1.10 V 1.20 V 1.30 V	1.05 V 1.20 V 1.35 V	1.00 V 1.20 V 1.40 V	0.95 V 1.20 V 1.45 V
p_{ret}	3.7%	4.4%	5.7%	6.9%	7.5%	8.7%
100	96.1%	94.5%	95.2%	96.1%	95.4%	92.4%
200	98.0%	98.3%	97.7%	97.6%	97.8%	96.4%
300	98.4%	98.6%	98.4%	98.5%	98.2%	97.4%
400	98.8%	99.0%	98.6%	98.7%	98.6%	98.0%
500	98.8%	98.9%	98.8%	99.0%	98.9%	98.2%
g	Accuracy of predictive model					

3.4 Conclusion

The security analysis of a PUF is a multifaceted and currently non-standardized discipline. We demonstrated the severity of threats that were quasi unknown and hence unexplored before. A first contribution proves that the functional behavior of several strong PUFs is not as uniquely tied to a given device as is often assumed. Three methods for deriving upper bounds on the min-entropy of a PUF were developed. Although they cannot guarantee the secure instantiation of future key generators, they can show that instantiations from the past are certainly insecure. We also showed that S-ArbRO PUFs are an easy target for machine learning attacks, despite the proposing authors' claim.

Our second contribution comprehends the first experimental validation that noise can be used as a side-channel for constructing a predictive model of a strong PUF. An attacker who measures the noisiness of responses can recover an internal analog secret that lowers the complexity of the learning problem.

Although our attacks were initially not optimized for speed, other authors later confirmed our hypothesis that combining noise measurements with machine learning techniques is highly efficient.

Chapter 4

A Survey on PUF-Based Key Generation

Cryptographic systems usually rely on reproducible, uniformly distributed secret keys. Affordable, physically secure key-storage in embedded NVM is hard to obtain though. Harvesting entropy from PUFs presents an alternative that lowers the vulnerability during the power-off state. Unfortunately, response bits are corrupted by noise and non-uniformities are bound to occur. Furthermore, a certain level of control on the key might be required, while response bits are determined by uncontrollable variations during the manufacturing process. A *helper data algorithm* (HDA) might nevertheless allow for the extraction of a reproducible, uniformly distributed, and controllable key. Helper data is public and reveals information about the response bits though; the system provider should hence quantify how much min-entropy remains. We contribute to the analysis of two categories of HDAs:

- *Fuzzy extractors* guarantee the correction of a predefined number of errors and are information-theoretically secure for all distributions of which the min-entropy exceeds a lower bound. The building block for handling noisiness, i.e., the *secure sketch*, provides error-correction with most frequently a binary $[n, k, d]$ block code. For instantiation purposes, the conservative, universal $(n - k)$ upper bound on the min-entropy loss is usually applied. Unfortunately, for non-uniform distributions, the residual min-entropy is underestimated, which implies that more response and helper bits than necessary have to be used. Expensive die area is hence blocked by circuits that are not strictly required to obtain the desired security level, i.e., symmetric key length. Moreover, given that considerable error rates have to be supported, it is often infeasible to instantiate a secure key generator for distributions that have a relatively low min-entropy.

We derive new, improved bounds on the min-entropy loss of a secure sketch for various PUF-induced distributions with practical relevance. Our bounds are easy-to-evaluate and can hence be used to improve the implementation efficiency. This is showcased for two predominant PUF imperfections, i.e., independent but potentially biased response bits as well as spatially correlated response bits, although a large variety of distributions could be supported. Moreover, a variety of commonly used codes is covered, e.g., Bose–Chaudhuri–Hocquenghem (BCH), Golay, and Reed–Muller codes, regardless of their algebraic complexity. Our newly developed theory is also used to disprove the common belief that secure sketches are *reusable*, i.e., we refute the claim that repeated helper data exposure does not result in additional min-entropy loss. This is bad news for the *reverse fuzzy extractor*, a lightweight mode of operation that is gaining momentum in the design of PUF-based protocols.

- Numerous other HDAs, which do not satisfy the definition of a fuzzy extractor, have been proposed. If there is a security proof, it usually applies to ideal-case distributions only, i.e., uniformly distributed responses, or slightly more general, independent but biased response bits. Although these assumptions are not necessarily realistic in practice, we complete and review derivations of the min-entropy loss accordingly. We show for example that bias is amplified for selection schemes that retain and discard the most and least stable response bits respectively. Moreover, we point out that the min-entropy loss of the *soft-decision decoding* scheme has been underestimated by its authors.

  Given that fuzzy extractors remain the default solution for PUF-based key generation, our improvements upon the $(n - k)$ bound allow for fair comparisons with these alternative methods. We illustrate this explicitly for the *index-based syndrome* (IBS) and von Neumann debiasing schemes, both of which are tailored for i.i.d. response bits. The existential motivation of both schemes hinges on the assumption that a stand-alone secure sketch cannot handle biased distributions. We eliminate the need for an educated guess that originates from the extrapolation of repetition code insights and/or the application of the overly conservative $(n - k)$ bound. It is concluded that for highly biased distributions, both debiasing schemes still have reason to exist.

  Another contribution is the development of a new category of helper data manipulation attacks. Designers of HDAs usually assume that an attacker has read-access to the helper data, and analyze the min-entropy loss accordingly, but often overlook that an attacker may also have write-access. By applying well-chosen manipulations, and comparing the corresponding failure rates for key reconstruction, an attacker may gain additional information about the secret responses. It turns out that most HDAs are vulnerable to some extent. In the worst-case scenario, it can even result in complete key-recovery, as is the case for certain instances of the *pattern matching* and soft-decision decoding schemes, for example. Luckily, cryptographic solutions can detect manipulations of the helper data, and hence counter the presented attacks. It is nevertheless important to raise awareness, so that future implementations are protected adequately.

Version History. The version history of this chapter is as follows. Our publication in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (TCAD 2015) is updated, improved, and considerably extended. Our publication at the *18th Conference on Cryptographic Hardware and Embedded Systems* (CHES 2016) is considerably extended. Our contributions to both the *Cryptographers' Track at the RSA Conference* (CT-RSA 2014) and the *17th Design, Automation & Test in Europe Conference & Exhibition* (DATE 2014) are summarized more briefly. The author of this PhD thesis is the main contributor to all four publications; Matthias Hiller and Mandel Yu extensively reviewed the CHES paper.

4.1 System Overview

4.1.1 Reference: Key-Storage in Physically Secure NVM

Figure 4.1 shows the conventional security architecture of an electronic device, where the embedded OTP NVM is assumed to be physically secure. There are two consecutive phases subsequent to the manufacturing of a device: one-time enrollment and in-the-field deployment, which take place in a trusted and untrusted environment respectively. During the enrollment, a cryptographic key $\mathbf{k}_{\text{app}}$ is programmed into the device. This key can be symmetric, but may equally well comprise the private and/or public key of an asymmetric system. Note that a public key requires integrity protection only, i.e., its confidentiality is of no concern. The procedure KeyGen that generates $\mathbf{k}_{\text{app}}$ relies on the presumably uniformly distributed output $\mathbf{n}$ of a *true random number generator* (TRNG). While for symmetric systems, it could be presumed that $\mathbf{k} = \mathbf{n}$, asymmetric systems require a more complicated procedure, as exemplified by the primality test of RSA [139].

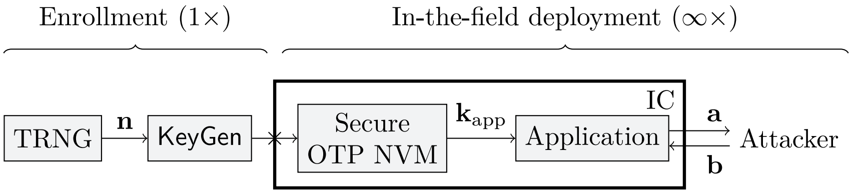


Figure 4.1: Key-storage in physically secure OTP NVM. The symbol × on the boundary of the IC denotes a one-time interface that is disabled after the enrollment.

The programming interface of the IC is permanently disabled after the enrollment. While the device is deployed in the field, a cryptographic application that interacts with the outside world requests the key $\mathbf{k}_{\text{app}}$ whenever needed. For applications that do not require an a priori shared secret with other parties, it could be beneficial to generate $\mathbf{k}_{\text{app}}$ within the device perimeter. Although the TRNG and the procedure KeyGen then have to be implemented on a resource-constrained IC, such an encapsulation lowers the degree of trust that is needed during the enrollment. All public-key systems where a unique key pair is assigned to each device are compatible with this approach, but also symmetric-key applications are not to be excluded. Consider for example a self-encrypting, password-protected Flash drive that uses AES internally.

4.1.2 An HDA for PUF-Based Key Generation

The previously described architecture relies on embedded OTP NVM that is assumed to be physically secure. As this is hard to obtain in practice, the implementation of an alternative PUF-based architecture as shown in Figure 4.2 can be considered. During the enrollment, helper data $\mathbf{h}$ is generated via a possibly randomized procedure $\mathbf{h} \leftarrow \mathsf{Gen}(\mathbf{x}, \mathbf{n})$, where X is the reference response of the PUF and N is a uniformly distributed secret. As before, Gen can be implemented either off-chip or on-chip. The OTP NVM that stores the helper data $\mathbf{h}$ can be assumed to be fully R/W-insecure. The problem is in fact shifted to several newly introduced building blocks that all require a physically secure implementation instead.

The first building block is a PRNG with a hardcoded seed that generates a list of publicly known challenges $\mathbf{c}$. This way, the respective responses $\mathbf{r}$ of the PUF can be concatenated into a lengthy secret $\mathbf{x}$. In the conventional case where a weak PUF rather than a strong PUF is used for key generation purposes, the challenge generator may be as simple as a counter. The consecutive steps of the

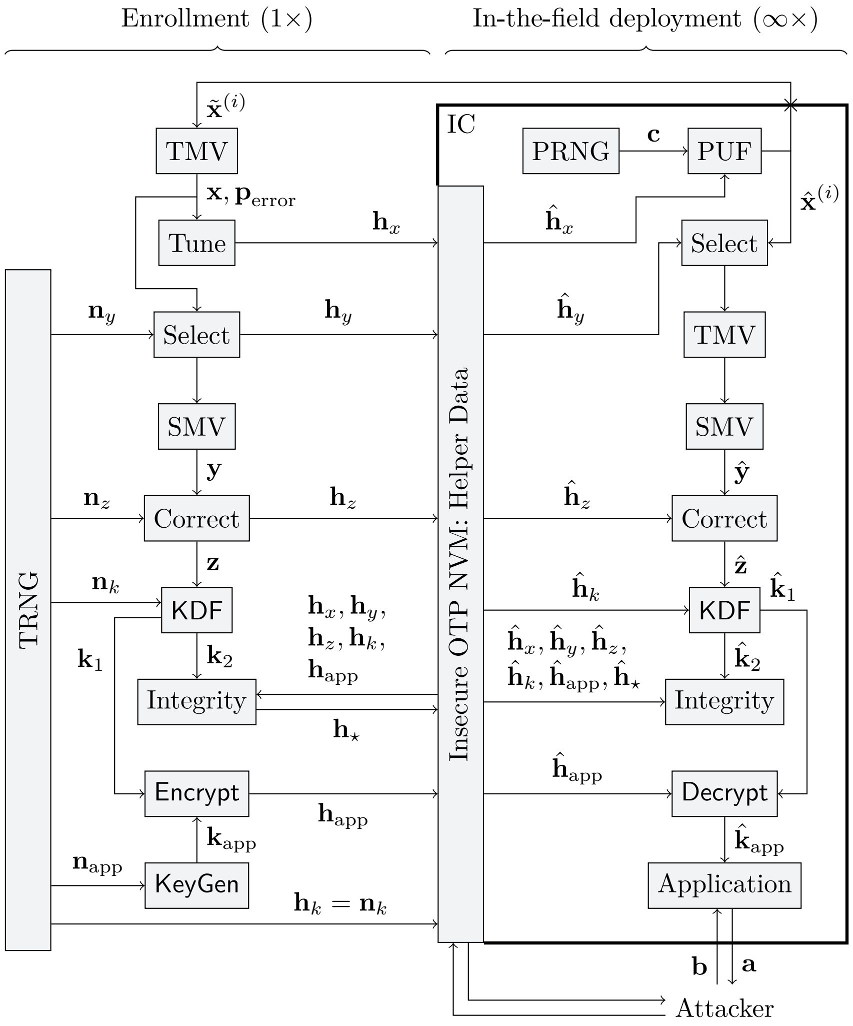

Enrollment (1×)
In-the-field deployment (∞×)
$\tilde{\mathbf{x}}^{(i)}$
TMV
$\mathbf{x}, \mathbf{p}_{\text{error}}$
Tune
$\mathbf{h}_x$
TRNG
$\mathbf{n}_y$
Select
$\mathbf{h}_y$
SMV
$\mathbf{y}$
$\mathbf{n}_z$
Correct
$\mathbf{h}_z$
$\mathbf{z}$
$\mathbf{n}_k$
KDF
$\mathbf{k}_1$
$\mathbf{k}_2$
$\mathbf{h}_x, \mathbf{h}_y, \mathbf{h}_z, \mathbf{h}_k, \mathbf{h}_{\text{app}}$
Integrity
$\mathbf{h}_\star$
Encrypt
$\mathbf{h}_{\text{app}}$
$\mathbf{k}_{\text{app}}$
$\mathbf{n}_{\text{app}}$
KeyGen
$\mathbf{h}_k = \mathbf{n}_k$
Insecure OTP NVM: Helper Data
IC
PRNG
$\mathbf{c}$
PUF
$\hat{\mathbf{x}}^{(i)}$
$\hat{\mathbf{h}}_x$
Select
$\hat{\mathbf{h}}_y$
TMV
SMV
$\hat{\mathbf{y}}$
$\hat{\mathbf{h}}_z$
Correct
$\hat{\mathbf{z}}$
$\hat{\mathbf{h}}_k$
KDF
$\hat{\mathbf{k}}_1$
$\hat{\mathbf{k}}_2$
$\hat{\mathbf{h}}_x, \hat{\mathbf{h}}_y, \hat{\mathbf{h}}_z, \hat{\mathbf{h}}_k, \hat{\mathbf{h}}_{\text{app}}, \hat{\mathbf{h}}_\star$
Integrity
$\hat{\mathbf{h}}_{\text{app}}$
Decrypt
$\hat{\mathbf{k}}_{\text{app}}$
Application
$\mathbf{b}$
$\mathbf{a}$
Attacker

Figure 4.2: A generic PUF-based key generator. Not necessarily all steps of the HDA are present in a practical implementation. The symbol × on the boundary of the IC denotes a one-time interface that is disabled after the enrollment.

HDA, which are extensively analyzed throughout this chapter, are summarized below. Not necessarily all steps have to be present; typical implementations of a full-fledged key generator [115, 19, 66] comprise a more limited subset.

- Despite our prior specifications in Chapters 2 and 3 that the challenge $\mathbf{c}$ is the single input of a PUF, several RO-based designs [182, 181] also accept helper data $\mathbf{h}$. By choosing the value of $\mathbf{h}$ for each manufactured instance of the PUF design individually, the noise and/or uniformity characteristics of the challenge-response behavior can be improved.
- Only a subset of the bits of the concatenated response $\mathbf{x}$ is selected for further processing. By retaining and discarding stable and unstable bits respectively, the reproducibility can be improved. Alternatively, a selection procedure may aim to improve the uniformity characteristics instead. Regardless of the objective, indices of either retained or discarded bits need to be stored as helper data $\mathbf{h}$.
- A *temporal majority vote* (TMV) involves a repeated evaluation of each ingoing bit, as formalized in (3.5). This favors the reproducibility. No helper data is required. In contrast to all other steps, TMV can be implemented during the enrollment exclusively, during the reconstruction exclusively, or during both phases independently.
- A *spatial majority vote* (SMV) enhances the reproducibility by aggregating several ingoing bits into a single output bit. The term *spatial* assumes that the PUF consists of an array of autonomous cells, which complies with an SRAM PUF for example. For, e.g., Arbiter PUFs, techniques still apply but the terminology is less intuitive.
- A final error-correction step produces a reproducible but usually non-uniformly distributed secret $\mathbf{z}$. Most frequently, but not necessarily, this involves a construction that relies on coding theory.
- A KDF converts the usually non-uniformly distributed secret $\mathbf{z}$ into one or more symmetric keys $\mathbf{k}$. The seed is stored as helper data $\mathbf{h}$. As a side note, $\mathbf{k}$ is occasionally referred to as a *physically obfuscated key* [49] in related work.
- PUFs inherently allow for the extraction of a symmetric key $\mathbf{k}$ that is determined by process variations. In order to gain full control over the keying material, an arbitrary application key $\mathbf{k}_{\text{app}}$ can be encrypted by a symmetric PUF-derived key $\mathbf{k}$. This also enables the use of public-key cryptography.

- An attacker may manipulate the helper data $\mathbf{h}$ of all previous steps so as to obtain information about the key $\mathbf{k}$. Another issue, which is of a non-malicious nature, is that excess noise can occasionally cause the reproduced key $\hat{\mathbf{k}}$ to differ from its enrolled counterpart $\mathbf{k}$. Therefore, it is recommended to perform an integrity check on the helper data $\hat{\mathbf{h}}$ and/or the key $\hat{\mathbf{k}}$.

In related literature, the term *syndrome* occasionally refers to the helper data $\mathbf{h}$ of all kinds of initial steps that deal with errors. Somewhat confusingly, one frequently used error-correction scheme happens to rely on the syndrome $\boldsymbol{\varphi}$ of a possible corrupted codeword $\tilde{\mathbf{w}}$, as is defined by a parity-check matrix $\mathbf{H}$. In order to avoid ambiguity, we preserve the term syndrome for the latter usage exclusively.

4.1.3 Failure Rate

The application key $\mathbf{k}_{\text{app}}$ of an enrolled device v is supposed to be reproducible. In-the-field reproduction usually succeeds on the condition that the newly generated response $\hat{\mathbf{x}}$ is sufficiently close to its enrolled counterpart $\mathbf{x}$. Each enrolled device v nevertheless exhibits a certain failure rate p_{fail} as is defined in (4.1), i.e., the probability that its application key $\mathbf{k}_{\text{app}}$ cannot be recovered due to an excessively noisy evaluation of its PUF. A typical, desired value for the failure rate is 10^{-6}. Although it is not impossible that an erroneous reproduction $\hat{\mathbf{z}} \neq \mathbf{z}$ nevertheless results in the correct application key $\hat{\mathbf{k}}_{\text{app}} = \mathbf{k}_{\text{app}}$, this term can safely be neglected.

$$\begin{aligned} p_{\text{fail}} = \mathbb{P}\big((\hat{K}_{\text{app}} \neq \mathbf{k}_{\text{app}}) | (K_{\text{app}} = \mathbf{k}_{\text{app}})\big) &\approx \mathbb{P}\big((\hat{K} \neq \mathbf{k}) | (K = \mathbf{k})\big) \\ &\approx \mathbb{P}\big((\hat{Z} \neq \mathbf{z}) | (Z = \mathbf{z})\big). \end{aligned} \tag{4.1}$$

The distribution of P_{fail} is usually non-trivial, but under the assumption of i.i.d. response bits X_i, exact formulas can be derived for the expectance $\mathbb{E}_{v \leftarrow V}[P_{\text{fail}}]$, i.e., the failure rate averaged over the infinite set of enrolled devices $\mathcal{V}$. Ultimately, $\mathbb{E}_{v \leftarrow V}[P_{\text{fail}}]$ can be expressed in terms of $\mathbb{E}_{v \leftarrow V}[P_{\text{error},x}]$, i.e., the expected error rate of the i.i.d. response bits X_i. The crucial insight is that all analyzed selection, TMV, and SMV methods produce i.i.d. output bits given i.i.d. input bits. Expected values of the bit error rate hence propagate through the HDA and are updated with every step. That is until the final error-correction step, where $\mathbb{E}_{v \leftarrow V}[P_{\text{fail}}]$ can be computed.

We define the initial error rate $p_{\text{error},x}$ as the probability that a regenerated response bit $\hat{x}^{(i)}$ differs from its enrolled, post-TMV counterpart x. The application of TMV during the enrollment can significantly reduce the expected bit error rate $\mathbb{E}_{v \leftarrow V}[P_{\text{error},x}]$. A formalization of this one-time procedure for an odd number of votes q has already been given in (3.5). Depending on the testing infrastructure of a given IC manufacturing process, the required counters may even be implemented externally rather than on each individual device, which would correspond to an economical head start in the reduction of the expected failure rate $\mathbb{E}_{v \leftarrow V}[P_{\text{fail}}]$. Moreover, given that the error rate $p_{\text{error},x}$ of each individual response bit x might have to be estimated anyway in order to enable the enrollment of other HDA steps [187, 112, 157], TMV can piggyback.

Under the assumption of i.i.d. response bits X_i and the heterogeneous variability–noise model in Section 3.1.5, the expected post-TMV error rate $\mathbb{E}_{v \leftarrow V}[P_{\text{error},x}]$ as a function of the number of votes q is given in (4.2). For $q \to \infty$ votes, the integrand simplifies to $\mathsf{f}_{\text{norm}}(\omega) p^{\infty}_{\text{error}}(\omega)$. For the degenerate case of $q = 1$ vote, the integrand simplifies to $2\mathsf{f}_{\text{norm}}(\omega) p^{\infty}_{\text{error}}(\omega)(1 - p^{\infty}_{\text{error}}(\omega))$. As is shown in Figure 4.3, the expected error rate $\mathbb{E}_{v \leftarrow V}[P_{\text{error},x}]$ in the former case is reduced with approximately a factor 0.7 compared to the latter case.

$$\begin{aligned}\mathbb{E}_{v \leftarrow V}\left[P_{\text{error},x}\right] = \int_{\omega=-\infty}^{\infty} \mathsf{f}_{\text{norm}}(\omega) \Bigg(& p^{\infty}_{\text{error}}(\omega)\, \mathsf{F}_{\text{bino}}\Big(\frac{q-1}{2}; q, p^{\infty}_{\text{error}}(\omega)\Big) \\ & + \Big(1 - p^{\infty}_{\text{error}}(\omega)\Big)\Big(1 - \mathsf{F}_{\text{bino}}\Big(\frac{q-1}{2}; q, p^{\infty}_{\text{error}}(\omega)\Big)\Big)\Bigg)\, \mathrm{d}\omega, \qquad (4.2) \\ \text{with } p^{\infty}_{\text{error},x}(\omega) = \lim_{q \to \infty} & \left[p_{\text{error},x}(\omega)\right] = \mathsf{F}_{\text{norm}}\left(-\frac{|\omega - \omega_{\text{thres}}|}{\sigma_{\text{noise}}}\right).\end{aligned}$$

4.1.4 Min-Entropy Loss

Procedures for estimating the min-entropy of concatenated response X have been discussed in Chapter 3. Unfortunately, this does not yet incorporate the min-entropy losses that are to be expected in every step of the HDA, as formalized in (4.3). The integrity check is omitted for simplicity reasons. For the most part, the inequalities are to be understood as the repercussions of monotonically decreasing the expected bit error rate. Given, e.g., a uniformly distributed response X with an expected bit error rate $\mathbb{E}_{v \leftarrow V}[P_{\text{error},x}] = 0.1$, it

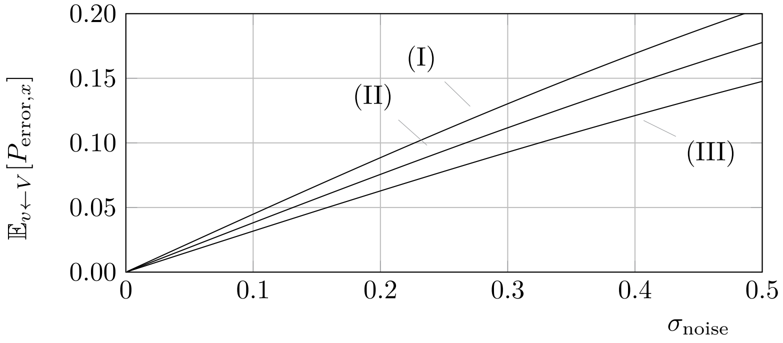


Figure 4.3: The expected error rate $\mathbb{E}_{v\leftarrow V}[P_{\text{error},x}]$ for reproducing i.i.d., post-TMV response bits X_i, given (I) $q = 1$, (II) $q = 3$, and (III) $q \to \infty$ votes respectively. The curves correspond to an unbiased PUF, i.e., $\omega_{\text{thres}} = 0$.

is not unusual that thousands of response bits as well as thousands of helper bits would be needed in order to obtain, e.g., a uniformly distributed 128-bit key K that can be reproduced with an expected failure rate $\mathbb{E}_{v\leftarrow V}[P_{\text{fail}}] = 10^{-6}$.

$$\begin{aligned} &\tilde{\mathbb{H}}_\infty(X|H_x) \geq \tilde{\mathbb{H}}_\infty(Y|(H_x, H_y)) \geq \tilde{\mathbb{H}}_\infty(Z|(H_x, H_y, H_z)) \\ &\geq \tilde{\mathbb{H}}_\infty(K|(H_x, H_y, H_z, H_k)) \geq \tilde{\mathbb{H}}_\infty(K_{\text{app}}|(H_x, H_y, H_z, H_k, H_{\text{app}})). \end{aligned} \tag{4.3}$$

In order to guarantee that key $\mathbf{k} \in \{0,1\}^\kappa$ is uniformly distributed, even for those who observe all helper data $\mathbf{h}$, the losses in terms of min-entropy should be quantified. With the notable exception of a fuzzy extractor [39], it turns out that this is only feasible for ideal-case distributions of the concatenated response $\mathbf{x} \in \{0,1\}^\lambda$. The most popular abstraction [112, 187, 62, 111, 170] involves i.i.d. response bits X_i, with $i \in [1, \lambda]$, and is parameterized by a single constant $\mathbb{P}(X_i = 1) = p_{\text{bias}}$. For an unbiased PUF, i.e., $p_{\text{bias}} = 1/2$, this corresponds to a uniform distribution over $\{0,1\}^\lambda$. Under the assumption of the heterogeneous variability–noise model in Section 3.1.5, the post-TMV constant p_{bias} as a function of the number of votes q is given in (4.4). For $q \to \infty$, the value of p_{bias} converges to $\mathsf{F}_{\text{norm}}(-\omega_{\text{thres}})$. The bias slightly increases with the number of votes q, i.e., there is a larger deviation from the ideal value $p_{\text{bias}} = 1/2$.

$$p_{\text{bias}} = \int_{\omega=-\infty}^{\infty} \mathsf{f}_{\text{norm}}(\omega)\mathsf{F}_{\text{bino}}\left(\frac{q-1}{2}; q, \mathsf{F}_{\text{norm}}\left(\frac{\omega_{\text{thres}} - \omega}{\sigma_{\text{noise}}}\right)\right) \mathrm{d}\omega. \tag{4.4}$$

Given the initial min-entropy $\mathbb{H}_\infty(X) = -\lambda \log_2\bigl(\max(p_{\text{bias}}, 1 - p_{\text{bias}})\bigr)$, we analyze the losses that are incurred by each HDA step. Although a single, unregulated exposure of helper data $\mathbf{h}$ is our primary focus, an attacker could have access to additional information and hence induce higher losses. For example, the attacker could force a device to reproduce its key $\mathbf{k}$ numerous times so as to obtain an estimate of the failure rate p_{fail}. This induces additional losses if P_{fail} and the concatenated response X are correlated. Likewise, a system provider might discard all devices with, e.g., $p_{\text{fail}} > 10^{-5}$, prior to deployment, which modifies the in-the-field distribution of X. Moreover, several PUF-based protocols [171, 110, 7] release helper data for not one but multiple noisy versions of the concatenated response X. HDA steps of which the min-entropy loss does not depend on the number of exposures are referred to as *reusable*.

4.2 Cryptographic Back-End

The final steps of our generic HDA in Figure 4.2 all involve cryptographic algorithms.

4.2.1 Encryption of an Application Key

The cryptographic application dictates the requirements for the final keying material $\mathbf{k}_{\text{app}}$. PUFs inherently allow for the extraction of a stable, uniformly distributed secret $\mathbf{k} \in \{0,1\}^\kappa$, which complies with symmetric-key algorithms such as AES. Unfortunately, public-key cryptography is not inherently supported: a key pair $(\mathbf{k}_{\text{priv}}, \mathbf{k}_{\text{pub}})$ needs to satisfy certain mathematical constraints, as exemplified by the primality test of RSA [139]. Even symmetric-key applications might occasionally face issues if the value of key $\mathbf{k}$ is determined by the inherent randomness of a PUF. Consider for example two PUF-enabled devices that are supposed to interact with each other and that are hence required to generate an identical key $\mathbf{k}$. Another example is the potentially required ability to replace a malfunctioning device while preserving the value of its key $\mathbf{k}$.

A universal solution [99, 152] is to use the PUF-derived key $\mathbf{k}$ for the encryption of an arbitrary application key $\mathbf{k}_{\text{app}}$, where the corresponding ciphertext is published as helper data $\mathbf{h}_{\text{app}}$. While deployed in the field, a device can then perform a decryption so as to recover $\mathbf{k}_{\text{app}}$. An *authenticated encryption* scheme, which efficiently provides both data confidentiality and data integrity, could be considered for implementation purposes. We refer to the proposals of the currently ongoing CAESAR competition [27]. A more traditional *Encrypt-then-MAC* [16] strategy nevertheless remains attractive. The helper data $(\mathbf{h}_x, \mathbf{h}_y, \mathbf{h}_z, \mathbf{h}_k)$ of all previous HDA steps requires integrity protection only, i.e., a separate MAC algorithm has to be implemented anyway.

There exist alternatives to the encryption of an arbitrary application key $\mathbf{k}_{\text{app}}$. For public-key systems, a PUF-derived key $\mathbf{k}$ can be adopted as the random seed of a procedure KeyGen that generates the key pair $(\mathbf{k}_{\text{priv}}, \mathbf{k}_{\text{pub}})$ [162]. No helper data $\mathbf{h}_{\text{app}}$ is required, but the computational workload during the reconstruction phase is expected to be high. For symmetric-key systems, one could consider that several selection and error-correction schemes [74, 187, 62] inherently allow for the extraction of a secret key $\mathbf{k}$ of which the value does not depend on the value of the concatenated response $\mathbf{x}$. Care is advised though: configuring two devices with an identical key $\mathbf{k}$, for example, can cause additional min-entropy loss for each. Although the concatenated responses $\mathbf{x}$ differ, an attacker is given more helper data $\mathbf{h}$ for the recovery of an identical secret $\mathbf{z}$, as is formalized in (4.5).

$$\tilde{\mathbb{H}}_\infty(Z|(H_{x,1}, H_{y,1}, H_{z,1})) \geq \tilde{\mathbb{H}}_\infty(Z|(H_{x,1}, H_{x,2}, H_{y,1}, H_{y,2}, H_{z,1}, H_{z,2})). \quad (4.5)$$

4.2.2 Integrity of Helper Data

The need for PUF-based key generation hinges on the absence of read-secure OTP NVM, i.e., a system provider cannot simply store $\mathbf{k}_{\text{app}}$ according to the architecture of Figure 4.1. It should hence be assumed that an attacker has read-access to the helper data $\{\mathbf{h}_x, \mathbf{h}_y, \mathbf{h}_z, \mathbf{h}_k, \mathbf{h}_{\text{app}}, \mathbf{h}_\star\}$, which is stored in OTP NVM instead. A stronger and commonly adopted motivation for PUFs hinges on the assumption that OTP NVM is not only read-insecure but also write-insecure. The alternative assumption that physical attacks are unidirectional seems artificial to us. Moreover, the stronger model allows the system provider to store helper data off-chip, where it is possibly managed by another party. Finally, several PUF-based authentication protocols [171, 110, 7] require the transfer of helper data over a fully insecure communication channel, where it can easily be intercepted and manipulated by an attacker. We consider three categories of threats that rely on the manipulation of helper data:

- An attacker might be able to trigger a modification to the reconstructed key $\hat{\mathbf{k}}$ that is unknowingly accepted by the cryptographic application. In the worst-case scenario, $\hat{\mathbf{k}}$ is partially known or even reduced in size and hence easy to recover completely via subsequent brute-force methods. If only the mathematical relationship to its original value $\mathbf{k}$ is known, a *related-key attack* on the application might still be feasible.

- An attacker might be able to measure the failure rate p_{fail} for the reconstruction of key $\mathbf{k}$ after some well-chosen helper data manipulation has been performed. In its simplest form, such an experiment can reveal the value of a single bit of $\mathbf{k}$, or alternatively, whether or not two bits of $\mathbf{k}$ are equal. An increase in the number of experiments can result in the recovery of the full key.
- Physical attacks on the reconstruction logic of the HDA can be facilitated by performing certain manipulations of the helper data.

The feasibility of the first two categories of threats depends on the cryptographic application. We distinguish between two types of applications, as summarized in Table 4.1. For simplicity, it is assumed that $\mathbf{k}_{\text{app}} = \mathbf{k}$. The first type of application, which is referred to as *interactive*, is more vulnerable as it allows an attacker to distinguish between any pair of keys. Consider for example an authentication protocol where a token proves its identity to another party by computing the MAC value of a received nonce $\mathbf{n}$. The second type of application, which is referred to as *silent*, is less vulnerable. An attacker can only check whether the initial key $\mathbf{k}$ is reproduced correctly. Consider for example a *secure boot* mechanism, where the integrity of processor code is verified by recomputing a hardcoded signature $\mathbf{b}$.

Table 4.1: Two types of cryptographic applications, differing in their vulnerability to helper data manipulation. The second column abstracts the information obtained by an attacker as queries to an oracle.

Type	**Oracle**	**Example**
Interactive	$\hat{\mathbf{k}} \overset{?}{=} \mathbf{k}$, $\hat{\mathbf{k}}^{(1)} \overset{?}{=} \hat{\mathbf{k}}^{(2)}$	$\mathbf{a} \leftarrow \mathsf{MAC}(\mathbf{n}; \mathbf{k})$
Silent	$\hat{\mathbf{k}} \overset{?}{=} \mathbf{k}$	$a \leftarrow \begin{cases} 1, & \text{if } \mathsf{MAC}(\mathbf{n}; \mathbf{k}) = \mathbf{b} \\ 0, & \text{otherwise} \end{cases}$

A scheme that with high probability detects helper data manipulation can be implemented on each device. There is a thin border with schemes that detect non-malicious failures, as can be attributed to an excessively noisy evaluation of the PUF. If either event is detected, operations can be aborted prematurely. Afterwards, a retrial may be launched. Four schemes are represented in Table 4.2. Variations can be applied to each scheme, but the main purpose here is to provide an overview of the design flavors and their associated issues. All schemes require the implementation of a cryptographic algorithm. Note, however, that resources can be shared with the cryptographic application and/or other steps of the HDA.

Boyen et al. [24, 40] propose the use of three protective mechanisms. First, helper data $\mathbf{h}$ is fed into a MAC-like algorithm where the PUF-derived secret $\mathbf{z}$ serves as a key, and the output $\mathbf{h}_\star$ is stored as helper data. The given realization relies on a cryptographic hash function that is modeled as a random oracle. Second, an upper bound t on the number of correctable errors is imposed. Third, only if the latter two checks are passed, a key $\hat{\mathbf{k}}$ of which the value directly depends on the helper data $\hat{\mathbf{h}}$ is computed. In practice, most malicious modifications to $(\mathbf{h}, \mathbf{h}_\star)$ will be detected. However, it is not excluded that for some well-chosen manipulations of $\mathbf{h}$, an attacker could force a device to reproduce a partially or fully known value $\hat{\mathbf{z}}$. Despite detecting most algorithmic helper data attacks as well as non-malicious failures, there is no protection against the enhancement of physical attacks. By the time that a device can verify its MAC $\mathbf{h}_\star$, side-channel traces already have been gathered.

Kevenaar et al. [83] adopt a simplified version where only the key $\mathbf{k}$ is fed into a cryptographic hash function. The digest is stored as helper data, i.e., $\mathbf{h}_\star \leftarrow \mathsf{Hash}(\mathbf{k})$. The proposed alternative offers less protection against algorithmic helper data attacks though. For several key-recovery attacks described later-on, potentially assuming a *silent* application, the attacker only needs to know whether or not a certain manipulation of the helper data results in a failure to reconstruct the key $\mathbf{k}$. It does not matter whether the occurrence of a failure is observed via either the application or the detection scheme.

Hiller et al. [66] only maintain the last out of three protective mechanisms of Boyen et al. [24, 40] and modify it as follows. The helper data $\mathbf{h}$ is fed into a cryptographic hash function and the resulting digest is XORed with the PUF-derived secret $\mathbf{z}$ so as to obtain the secret key $\mathbf{k}$. We observe that the proposed scheme is potentially insecure against related-key attacks; any modification of the helper data $\mathbf{h}$ that maintains the value of secret $\hat{\mathbf{z}}$ results in a related key $\hat{\mathbf{k}}$ with a known additive XOR pattern. The original computation of $\mathbf{k} \leftarrow \mathsf{Hash}(\mathbf{z}, \mathbf{h})$ is deemed more secure and also eliminates the implicit assumption that Z given H is uniformly distributed.

Tuyls et al. [168] adopt public-key cryptography. Let $(\mathbf{k}_{\text{priv}}, \mathbf{k}_{\text{pub}})$ denote the key pair of a trusted party. During the enrollment of a given device, helper data $(\mathbf{h}_x, \mathbf{h}_y, \mathbf{h}_z, \mathbf{h}_k, \mathbf{h}_{\text{app}})$ is signed with the private key $\mathbf{k}_{\text{priv}}$, and the result is published as helper data $\mathbf{h}_\star$. Given that every device stores the public key $\mathbf{k}_{\text{pub}}$ in write-secure ROM, the signature $\hat{\mathbf{h}}_\star$ can be verified in the field. A unique feature of the proposed scheme is that physical attacks on the reconstruction

Table 4.2: Four schemes for the detection of helper data manipulation and/or non-malicious failures. In the random oracle model and when applied to a secure sketch, the scheme of Boyen et al. [24, 40] is provably secure against algorithmic attacks. The label 'medium' accounts for cases where the definition of a secure sketch is not satisfied.

	Boyen et al. [40]	Kevenaar et al. [83]	Hiller et al. [66]	Tuyls et al. [168]
Enrollment	$\mathbf{h}_\star \leftarrow \mathsf{Hash}_1(\mathbf{z},\mathbf{h})$, $\mathbf{k} \leftarrow \mathsf{Hash}_2(\mathbf{z},\mathbf{h})$	$\mathbf{h}_\star \leftarrow \mathsf{Hash}(\mathbf{k})$	$\mathbf{k} \leftarrow \mathbf{z} \oplus \mathsf{Hash}(\mathbf{h})$	$\mathbf{h}_{\star,1} \leftarrow \mathsf{Sign}(\mathbf{h};\mathbf{k}_{\mathrm{priv}})$, $\mathbf{h}_{\star,2} \leftarrow \mathsf{Sign}(\mathsf{Hash}(\mathbf{k});\mathbf{k}_{\mathrm{priv}})$
Reproduction	$\hat{\mathbf{h}}_\star \overset{?}{=} \mathsf{Hash}_1(\hat{\mathbf{z}},\hat{\mathbf{h}})$, $\mathsf{HD}(\tilde{\mathbf{y}},\hat{\mathbf{y}}) \overset{?}{\leq} t$, $\hat{\mathbf{k}} \leftarrow \mathsf{Hash}_2(\hat{\mathbf{z}},\hat{\mathbf{h}})$	$\hat{\mathbf{h}}_\star \overset{?}{=} \mathsf{Hash}(\hat{\mathbf{k}})$	$\hat{\mathbf{k}} \leftarrow \hat{\mathbf{z}} \oplus \mathsf{Hash}(\hat{\mathbf{h}})$	$\mathsf{Verif}(\hat{\mathbf{h}},\hat{\mathbf{h}}_{\star,1};\mathbf{k}_{\mathrm{pub}})$, $\mathsf{Verif}(\mathsf{Hash}(\hat{\mathbf{k}}),\hat{\mathbf{h}}_{\star,2};\mathbf{k}_{\mathrm{pub}})$
Assumptions				W-secure ROM
Detection of non-malicious failures	✓	✓	×	✓
Resistance to algorithmic attacks	Medium	Low	Low	High
Prevention of enhanced side-channel analysis	×	×	×	✓

logic cannot be enhanced via helper data manipulation. A potential drawback is that the scheme hinges on the assumption that ROM is more write-secure than embedded OTP NVM.

4.2.3 Key Derivation Function

A KDF converts the stable but usually non-uniformly distributed secret Z into one or more secret keys $\mathbf{k} \in \{0,1\}^{\kappa}$. For the derivation of a single uniformly distributed key K, a *strong (randomness) extractor* [126, 39] as is formalized in Definition 3 can efficiently instantiate the KDF. For brevity, we did omit the adjective *average-case* in *average-case strong extractor*. For our generic HDA in Figure 4.2, the seed $\mathbf{n}$ is published as new helper data $\mathbf{h}_k$ and $\mathbf{h}_{\perp} = (\mathbf{h}_x, \mathbf{h}_y, \mathbf{h}_z)$. The constant ϵ is usually negligibly small, e.g., $\epsilon = 2^{-\kappa}$.

Definition 3 (Strong Extractor). *A strong extractor is a function* $\mathbf{k} \leftarrow \mathsf{Ext}(\mathbf{z}, \mathbf{n})$*, where* $\mathbf{z} \in \{0,1\}^{\lambda}$*,* $\mathbf{n} \in \{0,1\}^{\eta}$*, and* $\mathbf{k} \in \{0,1\}^{\kappa}$*. The evaluation time is polynomial in* λ *and* η*. For any distribution of* $(Z, H_{\perp})$ *with* $\tilde{\mathbb{H}}_{\infty}(Z|H_{\perp}) \geq \alpha$*, where* α *is a constant, it holds that* $\mathsf{SD}((K, H_{\perp}, N), (U, H_{\perp}, N)) \leq \epsilon$*, where* U *is uniformly distributed over* $\{0,1\}^{\kappa}$*,* N *is uniformly distributed over* $\{0,1\}^{\eta}$*, and* ϵ *is a constant.*

Universal hash functions [29] are excellent randomness extractors, according to the *leftover hash lemma* (LHL) [55]. Their min-entropy loss $\alpha - \kappa = 2\log_2(1/\epsilon)$ is the best that can be achieved by any extractor, but remains nevertheless substantial. As is clear from Table 4.3, the extraction of a 128-bit key K requires input Z given $H_{\perp}$ to have a min-entropy of at least 384 bits. The *seed* $\mathbf{h}$, which comprehends a randomly selected member from a family of hash functions, is of substantial length and therefore poses another practical burden. The Toeplitz universal hash function is a popular choice in practice due to its convenient LFSR-based architecture. Nevertheless, implementations by Bösch et al. [22] and Maes et al. [114] short-cut the min-entropy loss to zero for efficiency reasons.

Table 4.3: The min-entropy loss $\alpha - \kappa$ of a randomness extractor.

Hash function	**Theory**	**Min-entropy loss**
Universal	LHL [55]	2κ for $\epsilon = 2^{-\kappa}$
	Generalized LHL [8]	κ for $\epsilon = 2^{-\kappa}$
Cryptographic	Random oracle	0

For a wide range of applications, the min-entropy loss can be reduced according to the generalized LHL [8]. The most notable omissions are PRFs and stream ciphers. For the extraction of a 128-bit key K, it is then required that input Z given $H_\perp$ has a min-entropy of at least 256 bits. Also the length η of the seed $\mathbf{h}$ can be reduced. As the overall overhead is still substantial, practical key generators often rely on a cryptographic hash function that is assumed to behave as a *random oracle*. The latter idealized heuristic results in zero min-entropy loss as well as the absence of helper data $\mathbf{h}$. Consider for example the SPONGENT cryptographic hash function [20], as implemented by Maes et al. [115].

As a side note, the previous hashing step is occasionally referred to as *privacy amplification*. The latter term might be more relevant to its original context where the biometric features of a user are stored for future authentication purposes [106]. The privacy of the users remains intact even if an attacker obtains the hashed file contents. The seed value provides an elegant *salting* mechanism, similar to existing best practices for storing textual passwords.

If not a single but multiple keys $\mathbf{k}$ need to be derived, the sizes of the PUF, the strong extractor, and all intermediate building blocks could be scaled accordingly. A more efficient solution, however, is the implementation of an *extract-then-expand* approach [92]. This includes, for example, the concatenation of a strong extractor and a PRF.

4.3 Error-Correction with a Secure Sketch

Error-correction, which is also referred to as *information reconciliation* in the related literature, is most frequently performed by a *secure sketch*. The popularity of this primitive originates from the fact that for any distribution of input Y, it provides a lower bound on the error-correcting capabilities, and simultaneously also an upper bound on the min-entropy loss.

4.3.1 Definition

A secure sketch restores a noisy version $\tilde{\mathbf{y}}$ of the reference input $\mathbf{y}$ to its original value, given that both vectors are sufficiently close to each other. The original definition of Dodis et al. [39] supports various *metric spaces* $\mathcal{Y}$ and distance functions dist that are all relevant to the field of human biometrics. Our recapitulation in Definition 4 is tailored for PUFs and hence restricts the scope to binary vectors $\mathbf{y} \in \{0,1\}^n$ and the Hamming distance HD. For brevity, the adjective *average-case* in *average-case secure sketch* has been omitted, given that all realizations that will be discussed in this chapter allow for the inclusion of the random variable $H_\perp$. Note that for our generic HDA in Figure 4.2, $H_\perp = (H_x, H_y)$.

Definition 4 (Secure Sketch). *A secure sketch consists of a pair of efficient and possibly randomized procedures: the sketching procedure* $\mathbf{h} \leftarrow \mathsf{SSGen}(\mathbf{y}, \mathbf{n})$*, where* N *is a uniformly distributed seed, and the recovery procedure* $\hat{\mathbf{z}} \leftarrow \mathsf{SSRep}(\tilde{\mathbf{y}}, \mathbf{h})$*. Two properties hold:*

- *(Correctness). If* $\mathsf{HD}(\mathbf{y}, \tilde{\mathbf{y}}) \leq t$*, where* t *is a constant, correctness of reconstruction is guaranteed, i.e.,* $\hat{\mathbf{z}} = \mathbf{z}$*. If* $\mathsf{HD}(\mathbf{y}, \tilde{\mathbf{y}}) > t$*, there is no guarantee whatsoever.*
- *(Security). For all distributions of* $(Y, H_\perp)$ *with* $\tilde{\mathbb{H}}_\infty(Y|H_\perp) \geq \alpha_Y$*, where* α_Y *is a constant, a corresponding lower bound on the residual min-entropy holds, i.e.,* $\tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) \geq \alpha_Z$*, where* α_Z *is a constant.*

For convenience, we have generalized the original definition of a secure sketch such that its nominal output $\mathbf{z}$ is not necessarily equal to $\mathbf{y}$. The prior notion of *fuzzy commitment* [74] can hence be supported as well. We then commit to an arbitrary secret $\mathbf{z}$ by binding it to $\mathbf{y}$. One may decommit given a noisy version $\tilde{\mathbf{y}}$ that is sufficiently close to $\mathbf{y}$. Constructions that return a substring $\mathbf{z}$ of $\mathbf{y}$ [76] are supported as well. For all discussed realizations, α_Z can easily be determined for any given α_Y. This can be attributed to a universal upper bound on the min-entropy loss, i.e., for all distributions of $(Y, H_\perp)$ it holds that $\tilde{\mathbb{H}}_\infty(Y|H_\perp) - \tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) \leq \beta$, where β is a constant.

As is clear from the following thought experiment [28], a secure sketch can only be guaranteed to handle inputs of which the min-entropy is relatively high. Consider all distributions of $(Y, H_\perp)$ such that $\mathcal{H}_\perp = \varnothing$ and such that only $\lceil 2^{\alpha_Y} \rceil$ outcomes $\mathbf{y}$ in the set $\mathcal{Y}$ have a nonzero probability of occurrence. Evidently, these probabilities cannot exceed $2^{-\alpha_Y}$, given that $\mathbb{H}_\infty(Y) \geq \alpha_Y$ holds. If there exist distributions such that all $\lceil 2^{\alpha_Y} \rceil$ possible outcomes fit within a hypersphere of radius t, then an attacker can recover $\mathbf{y}$ by executing SSRep on the center of the hypersphere, which implies $\tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) = 0$. The lower bound on α_Y in (4.6) ensures that no such distributions exist.

$$\begin{aligned} \alpha_Y &> \log_2\Big(\textstyle\sum_{i=0}^{t} \binom{n}{i}\Big) \\ &> -n\Big(\tfrac{t}{n}\log_2\big(\tfrac{t}{n}\big) + \big(1 - \tfrac{t}{n}\big)\log_2\big(1 - \tfrac{t}{n}\big)\Big) - \tfrac{1}{2}\log_2(n) - \tfrac{1}{2}. \end{aligned} \tag{4.6}$$

A slightly modified notion brings us to the *fuzzy extractor* [39] in Definition 5. Output K is then guaranteed to be nearly-uniform, even for those who observe helper data $(H, H_\perp)$, and can hence be used as a symmetric key. The constant ϵ is usually chosen to be negligibly small, e.g., $\epsilon = 2^{-\kappa}$. There is a proven standard method to craft a fuzzy extractor from a secure sketch: a *strong randomness extractor*, as previously discussed in Section 4.2.3, could derive a key from the sketch output, i.e., $\mathbf{k} \leftarrow \mathsf{Ext}(\mathbf{z}, \mathbf{n}_k)$, where N_k is a uniformly distributed seed.

Definition 5 (**Fuzzy Extractor**). *A fuzzy extractor consists of a pair of efficient and possibly randomized procedures: the sketching procedure* $\mathbf{h} \leftarrow \mathsf{FEGen}(\mathbf{y}, \mathbf{n})$*, where* N *is a uniformly distributed seed, and the recovery procedure* $\hat{\mathbf{k}} \leftarrow \mathsf{FERep}(\hat{\mathbf{y}}, \mathbf{h})$*. Two properties hold:*

- *(Correctness). If* $\mathsf{HD}(\mathbf{y}, \hat{\mathbf{y}}) \leq t$*, where* t *is a constant, correctness of reconstruction is guaranteed, i.e.,* $\hat{\mathbf{k}} = \mathbf{k}$*. If* $\mathsf{HD}(\mathbf{y}, \hat{\mathbf{y}}) > t$*, there is no guarantee whatsoever.*
- *(Security). For all distributions of* $(Y, H_\perp)$ *with* $\tilde{\mathbb{H}}_\infty(Y|H_\perp) \geq \alpha_Y$*, where* α_Y *is a constant, the output* $\mathbf{k} \in \{0,1\}^\kappa$ *is guaranteed to be nearly uniform, even for those who observe the helper data* $(\mathbf{h}_\perp, \mathbf{h})$*. Adopting the statistical distance as defined in* (2.10) *as a metric of uniformity, it holds that* $\mathsf{SD}((K, H_\perp, H), (U, H_\perp, H)) \leq \epsilon$*, where* U *is uniformly distributed over* $\{0,1\}^\kappa$ *and* ϵ *is a constant.*

4.3.2 Constructions using a Block Code

Table 4.4 specifies seven realizations of a secure sketch that all rely on a binary code ζ with minimum distance d. Correctness of the reconstruction is guaranteed if $\mathsf{HD}(\mathbf{y}, \tilde{\mathbf{y}}) \leq t$, where $t = \lfloor (d-1)/2 \rfloor$ is the error-correcting capability of ζ. The expected failure rate for key reconstruction is given in (4.7), under the assumption of i.i.d. response bits Y_i. Although a decoding algorithm might occasionally be able to handle more than t errors, for the sake of simplicity, it is assumed that $\mathsf{HD}(\mathbf{y}, \tilde{\mathbf{y}}) > t$ always results in a failure.

$$\begin{aligned}\mathbb{E}_{v \leftarrow V}\left[P_{\text{fail}}\right] &= 1 - \mathbb{E}_{v \leftarrow V}\left[\mathsf{F}_{\text{poisbino}}(t, P_{\text{error},1}, P_{\text{error},2}, \cdots, P_{\text{error},n})\right] \\ &= 1 - \mathsf{F}_{\text{bino}}\big(t; n, \mathbb{E}_{v \leftarrow V}[P_{\text{error}}]\big).\end{aligned} \tag{4.7}$$

We prove that all seven constructions exhibit an identical min-entropy loss. The equivalency proofs are established in a pairwise manner, as guided by Figure 4.4. For a given distribution of $(Y|H_\perp)$, any two constructions, when

Table 4.4: Seven secure sketch constructions. Correctness of the reconstruction is guaranteed if $\mathsf{HD}(\mathbf{y}, \tilde{\mathbf{y}}) \leq t$. A similar construction of Davida et al. [37] is not considered, given that it provides a less favorable trade-off between min-entropy loss and error-correcting capabilities [74].

Construction	$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{H}\rvert$	$\mathbf{h} \leftarrow \mathsf{SSGen}(\mathbf{y}, \mathbf{n})$	$\lvert\mathcal{Z}\rvert$	$\hat{\mathbf{z}} \leftarrow \mathsf{SSRep}(\tilde{\mathbf{y}}, \mathbf{h})$	$\mathbf{z}$
Code-offset, Juels et al. [74]	2^n	$\lvert\mathcal{M}\rvert$	2^n	$\mathbf{h} \leftarrow \mathbf{y} \oplus \mathsf{Encode}(\mathbf{n})$	2^n	$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{y}} \oplus \mathbf{h}$ $\hat{\mathbf{z}} \leftarrow \mathsf{Correct}(\tilde{\mathbf{w}})$	$\mathbf{w}$
Code-offset, Dodis et al. [39]					2^n	$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{y}} \oplus \mathbf{h}$ $\hat{\mathbf{z}} \leftarrow \mathbf{h} \oplus \mathsf{Correct}(\tilde{\mathbf{w}})$	$\mathbf{y}$
Code-offset, Tuyls et al. [166]					$\lvert\mathcal{M}\rvert$	$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{y}} \oplus \mathbf{h}$ $\hat{\mathbf{z}} \leftarrow \mathsf{Decode}(\tilde{\mathbf{w}})$	$\mathbf{m}$
Syndrome, Bennett et al. [17]	2^n	0	2^{n-k}	$\mathbf{h} \leftarrow \mathbf{y}\,\mathbf{H}^T$	2^n	$\boldsymbol{\varphi} \leftarrow \tilde{\mathbf{y}}\,\mathbf{H}^T \oplus \mathbf{h}$ $\hat{\mathbf{z}} \leftarrow \tilde{\mathbf{y}} \oplus \hat{\mathbf{e}}$	$\mathbf{y}$
Systematic Yu [185]	2^n	0	2^{n-k}	$\mathbf{h} \leftarrow (y_1 \cdots y_k)\,\mathbf{P}$ $\oplus(y_{k+1} \cdots y_n)$	2^n	$\hat{\mathbf{w}} \leftarrow \mathsf{Correct}(\tilde{\mathbf{y}} \oplus (\mathbf{0} \Vert \mathbf{h}))$ $\hat{\mathbf{z}} \leftarrow \hat{\mathbf{w}} \oplus (\mathbf{0} \Vert \mathbf{h})$	$\mathbf{y}$
Systematic Kang et al. [76]					2^k	$\hat{\mathbf{z}} \leftarrow \mathsf{Decode}(\tilde{\mathbf{y}} \oplus (\mathbf{0} \Vert \mathbf{h}))$	$(y_1 \cdots y_k)$
Multi-code [1]	2^n	2^k	2^{n-k}	$\mathbf{h} \leftarrow j$ s.t. $\mathbf{y} \in \zeta_j$	2^k	$\hat{\mathbf{z}} \leftarrow \mathsf{Decode}(\tilde{\mathbf{y}}; \zeta_j)$	$\mathbf{m}$

instantiated with an identical code ζ, are shown to result in the same residual min-entropy $\tilde{\mathbb{H}}_\infty(Z|(H_\perp, H))$, as defined in (4.8). Pairwise equivalencies that are established in other works [175, 63] consider fewer methods and/or impose unnecessary restrictions on the distribution of Y. We hence make progress in terms of completeness and generality. We emphasize that the min-entropy loss does not depend on the decoding method, simply because the helper data $\mathbf{h}$ is not affected.

$$\tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) \overset{\text{def}}{=} -\log_2\Big(\mathbb{E}_{(\mathbf{h}_\perp,\mathbf{h})\leftarrow(H_\perp,H)}\Big[\max_{\mathbf{z}\in\mathcal{Z}} \mathbb{P}\big((Z=\mathbf{z})|((H_\perp=\mathbf{h}_\top)\cap(H=\mathbf{h}))\big)\Big]\Big). \tag{4.8}$$

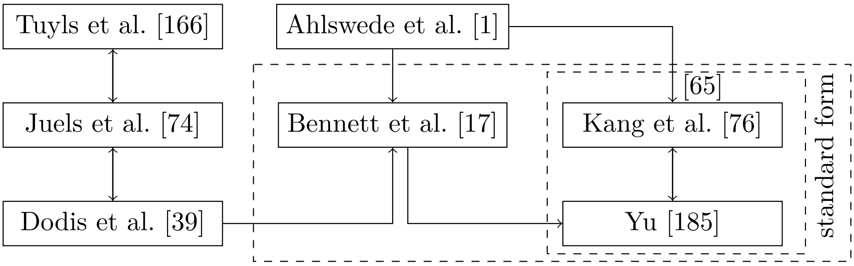


Figure 4.4: Pairwise min-entropy loss equivalencies among seven sketches, as indicated by the arrows. Transitive relations apply when following the arrows. E.g., the schemes of Dodis et al. [39] and Kang et al. [76] are equivalent, given that both are instantiated with a linear code in standard form.

For $[n, k, d]$ block codes in particular, the well-known $(n-k)$ upper bound on the min-entropy loss holds, i.e., $\tilde{\mathbb{H}}_\infty(Y|H_\perp) - \tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) \leq (n-k)$, as proven by Dodis et al. [39]. More generally, this extends to $\tilde{\mathbb{H}}_\infty(Y|H_\perp) - \tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) \leq n - \log_2(|\mathcal{M}|)$. Due to the equivalence among secure sketch constructions, the bound apply to all seven.

Code-Offset Methods

The code-offset method of Juels et al. [74], as represented in Table 4.4, requires a code ζ that is not necessarily linear. Even more, it is not required be a block code either. Linear codes (BCH, Golay, repetition, etc.) remain the most frequently used though due to their efficient decoding algorithms [108]. Table 4.4 also represents a modification where Rep returns sketch input $\mathbf{y}$ rather than codeword $\mathbf{w}$, as proposed by Dodis et al. [39]. For the latter, it was proven that the $(n-k)$ upper bound on the min-entropy loss holds, given a block code. Table 4.4 also lists another minor modification where Rep returns message $\mathbf{m}$, as suggested by Tuyls et al. [166]. This necessitates an implementation of Decode rather than Correct.

All three code-offset methods produce the same helper data $\mathbf{h}$ but differ in their reconstructed output $\mathbf{z}$. Nevertheless, we argue that the residual min-entropy in (4.8) evaluates to the same value for all three methods. This follows from an underlying one-to-one correspondence, given in (4.9). The algorithm Encode comprehends a bijection between the set of all messages $\mathcal{M}$ and the set of all codewords $\mathcal{W}$. Furthermore, for given helper data $\mathbf{h}$, there is a bijection between $\mathcal{W}$ and a reduced set of responses $\mathcal{Y}' = \{\mathbf{h} \oplus \mathbf{w} \mid \mathbf{w} \in \mathcal{W}\} \subseteq \mathcal{Y}$. Note that $|\mathcal{M}| = |\mathcal{W}| = |\mathcal{Y}'|$.

$$\begin{aligned}\forall(\mathbf{h}_\perp, \mathbf{h}, \mathbf{m}) \in (\mathcal{H}_\perp \times \mathcal{H} \times \mathcal{M}), \mathbb{P}\Big((M = \mathbf{m}) | \big((H_\perp, H) = (\mathbf{h}_\perp, \mathbf{h})\big)\Big)\\ = \mathbb{P}\Big((W = \mathsf{Encode}(\mathbf{m})) | \big((H_\perp, H) = (\mathbf{h}_\perp, \mathbf{h})\big)\Big)\\ = \mathbb{P}\Big((Y = \mathsf{Encode}(\mathbf{m}) \oplus \mathbf{h}) | \big((H_\perp, H) = (\mathbf{h}_\perp, \mathbf{h})\big)\Big).\end{aligned} \tag{4.9}$$

Min-entropy loss can be understood as a *one-time pad* imperfection. The sketch input $\mathbf{y}$ is masked with a random codeword $\mathbf{w}$, the latter of which is not uniformly distributed: $\mathbb{H}_\infty(W) = \log_2(|\mathcal{M}|) < n$. For linear codes in particular, we highlight a convenient interpretation using cosets. The helper data $\mathbf{h}$ then reveals in which coset reference $\mathbf{y}$ resides. It can be seen easily that $\mathbf{h}$ is equal to a random vector in the same coset as $\mathbf{y}$. The residual min-entropy in (2.2) hence reduces to (4.10) for linear codes, where $\mathbf{e}$ denotes a coset leader.

$$\begin{aligned}\tilde{\mathbb{H}}_\infty(Z|(H_\perp, H)) = -\log_2\Big(\mathbb{E}_{(\mathbf{h}_\perp, \mathbf{e}) \leftarrow (H_\perp, E)}\Big[\\ \max_{\mathbf{w} \in \mathcal{W}} \mathbb{P}\Big((Y = \mathbf{e} \oplus \mathbf{w}) | \big((H_\perp, E) = (\mathbf{h}_\perp, \mathbf{e})\big)\Big)\Big]\Big).\end{aligned} \tag{4.10}$$

Syndrome Method

The syndrome method of Bennett et al. [17], as specified in Table 4.4, was initially proposed as part of a *quantum oblivious transfer* protocol, but maps quite easily to the secure sketch framework of Dodis et al. [39]. The method requires a linear code ζ with parity-check matrix $\mathbf{H}$. The well-known $(n-k)$ upper bound on the min-entropy loss holds, as proven by Dodis et al. [39]. This is a trivial consequence from the universally valid expression in (2.3), given that the helper data $\mathbf{h}$ is limited to $(n-k)$ bits.

The syndrome method of Bennett et al. [17] and the code-offset method of Dodis et al. [39] both reconstruct $\mathbf{z} = \mathbf{y}$. Furthermore, for both methods, helper data $\mathbf{h}$ reveals in which coset $\mathbf{y}$ resides. For the syndrome method, this is a trivial consequence from the one-to-one correspondence between cosets and syndromes. Note that the code-offset method is being instantiated with a linear code, given that the syndrome method is restricted to this case. The residual min-entropy of both methods can hence be written as shown in (4.10).

Systematic Methods

The method of Yu [185], as specified in Table 4.4, requires a linear code $\mathcal{C}$ with a generator matrix in standard form, i.e., $\mathbf{G} = (\mathbf{I}_k \, \mathbf{P})$. We observe that the $(n-k)$ upper bound on the min-entropy loss holds due to (2.3), given that helper data $\mathbf{h}$ is limited to $(n-k)$ bits. Table 4.4 also represents a slightly modified method where Rep returns $(y_1 \, y_2 \, \dots \, y_k)$ rather than $\mathbf{y}$. This was first proposed by Kang et al. [76] and independently also by Hiller et al. [65]. Nevertheless, (4.11) indicates that the residual min-entropy in (4.8) is identical. The main insight is that $(y_1 \, y_2 \, \dots \, y_k)$ and $\mathbf{h}$ fully determine $(y_{k+1} \, y_{k+2} \, \dots \, y_n)$.

$$\begin{aligned} &\forall(\mathbf{h}_\perp, \mathbf{h}, \mathbf{y}) \in (\mathcal{H}_\perp \times \mathcal{H} \times \mathcal{Y}), \\ &\mathbb{P}\Big(\big((Y_1, Y_2, \cdots, Y_k) = (y_1, y_2, \cdots, y_k)\big) \big| \big((H_\perp, H) = (\mathbf{h}_\perp, \mathbf{h})\big)\Big) \\ &= \mathbb{P}\Big(\big(Y = (y_1 \, y_2 \, \dots \, y_k)\mathbf{G} \oplus (\mathbf{0} \, \mathbf{h})\big) \big| \big((H_\perp, H) = (\mathbf{h}_\perp, \mathbf{h})\big)\Big). \end{aligned} \tag{4.11}$$

The methods of Bennett et al. [17] and Yu [185] both reconstruct the sketch input, i.e., $\mathbf{z} = \mathbf{y}$. We are the first to observe though that the helper data $\mathbf{h}$ is identical as well, as proven in (4.12). Of course, this assumes a generator matrix in standard form, i.e., $\mathbf{G} = (\mathbf{I}_k \, \mathbf{P})$, given that Yu's method is restricted to this case.

$$\mathbf{h} = \mathbf{y}\,\mathbf{H}^T = \mathbf{y} \begin{pmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{pmatrix} = (y_1 \, y_2 \, \dots \, y_k)\,\mathbf{P} \oplus (y_{k+1} \, y_{k+2} \, \dots \, y_n). \tag{4.12}$$

Multi-Code Method

The method of Ahlswede et al. [1], as specified in Table 4.4, was initially proposed for secret key transport with *correlated sources*. It nevertheless maps quite easily to our framework of interest, as observed by Hiller et al. [65]. A distinguishing feature is the use of multiple codes ζ_j, covering mutually disjoint sets of codewords. We restrict our attention to $[n,k,d]$ block codes with $j \in [0, 2^{n-k}-1]$. Every $\mathbf{y} \in \mathcal{Y}$ then coincides with exactly one codeword, guaranteeing correctness. Furthermore, the $(n-k)$ upper bound on the min-entropy loss holds due to (2.3), given that helper data $\mathbf{h} = j$ is limited to $(n-k)$ bits.

Hiller et al. [65] proposed an efficient implementation where all codes are derived from a single parent code $\mathcal{C}_0$. In particular, ζ_0 is a linear code in standard form, i.e., $\mathbf{G} = (\mathbf{I}_k\,\mathbf{P})$, and all other codes ζ_j are cosets: $\mathcal{W}_j = \{\mathbf{w} \oplus (\mathbf{0}\|\mathbf{h}) \mid \mathbf{w} \in \mathcal{W}_0\}$. This turns out to be fully equivalent with the method of Kang et al. [76] in Table 4.4, i.e., helper data $\mathbf{h}$ and reconstructed output $\mathbf{z}$ are identical. We consider a slightly more general case. In particular, a linear code ζ_0 that is not necessarily in standard form, as required by the method of Bennett et al. [17] as well. All child codes ζ_j are again formed as the cosets of ζ_0. Therefore, helper data $\mathbf{h} = j$ still reveals in which coset $\mathbf{y}$ resides and (4.10) holds once again. The one-to-one correspondence of output $\mathbf{z}$ in (4.13) finalizes our proof.

$$\begin{aligned}\forall(\mathbf{h}_\perp,\mathbf{h},\mathbf{y}) \in (\mathcal{H}_\perp \times \mathcal{H} \times \mathcal{Y}), \mathbb{P}\Big((Y=\mathbf{y})|\big((H_\perp,H)=(\mathbf{h}_\perp,\mathbf{h})\big)\Big)\\ = \mathbb{P}\Big(\big(M = \mathsf{Decode}(\mathbf{y};\mathcal{C}_\mathbf{h})\big)|\big((H_\perp,H)=(\mathbf{h}_\perp,\mathbf{h})\big)\Big).\end{aligned} \tag{4.13}$$

4.3.3 Repeated Execution of a Concatenated Code

Optimized implementations of a secure sketch often rely on a concatenated code $\zeta_2 \circ \zeta_1$ that processes q non-overlapping blocks of response bits independently [22, 115]. The inner code ζ_2 is usually a small $[n_2, k_2 = 1, d_2 = n_2]$ repetition code, with n_2 odd, allowing to support a high bit error rate $\mathbb{E}[P_\text{error}]$. A large $[n_1, k_1, d_1]$ outer code ζ_1, e.g., a BCH code [108], is faced with a considerably lower bit error rate so that its min-entropy loss can be relatively small. The size of ζ_1 is nevertheless limited due to the implementation footprint of Correct/Decode. Response $\mathbf{y}$ hence needs to be partitioned into q blocks in order to generate a key $\mathbf{k}$ of sufficient length.

It is convenient to encapsulate the operation $q \times [n_2, k_2, d_2] \circ [n_1, k_1, d_1]$ in a single umbrella block code with $[n = q \cdot n_1 \cdot n_2, k = q \cdot k_1, d = d_1 \cdot d_2]$. Only a maximum of $t = t_1(t_2 + 1) \leq \lfloor (d-1)/2 \rfloor$ errors is guaranteed to provide correctness of reconstruction. The average-case behavior is more optimistic though, as reflected by the expected failure rate in (4.14) under the assumption of i.i.d. response bits. The $(n - k)$ upper bound on the min-entropy loss can be applied to the umbrella block code.

$$\mathbb{E}_{v \leftarrow V}\big[P_{\text{fail}}\big] = 1 - \Big(\mathsf{F}_{\text{bino}}\big(t_1, n_1, 1 - \mathsf{F}_{\text{bino}}(t_2, n_2, \mathbb{E}_{v \leftarrow V}[P_{\text{error}}])\big)\Big)^q. \quad (4.14)$$

4.3.4 Convolutional Codes

Despite the popularity of binary $[n, k, d]$ block codes, Hiller et al. [66] implement a secure sketch with a binary (η, λ) *convolutional code.* Used in conformity with, e.g., the code-offset method in Table 4.4, it is clear that the $(n - k)$ upper bound on the min-entropy loss still applies. For example, the authors adopt an $(\eta = 2, \lambda = 7)$ convolutional code as an $[n = 268, k = 128, d = 8]$ block code, exhibiting a min-entropy loss of at most 140 bits.

4.3.5 Tight Bounds on the Min-Entropy Loss

System providers who implement a secure sketch currently rely on the $(n - k)$ upper bound on the min-entropy loss [115]. Unfortunately, this bound often leads to an overly conservative design when instantiating security parameters accordingly. We aim to improve this situation through the development of a graphical framework that produces tight bounds on the residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H)$ for typical distributions of SRAM-like PUFs. Previous HDA steps are assumed to be absent, i.e., $Y = X$ and $\mathcal{H}_\perp = \varnothing$. The critical *first-order* effects of bias and spatial correlations are captured. Both lower and upper bounds are supported. The lower bounds are of primary interest for a conservative system provider, who evidently entertains the worst-case scenario. Considerable improvements upon the $(n - k)$ bound, i.e., the leftmost inequality in (4.15), are obtained. Therefore, one is even able to handle low-entropy distributions that are deemed impossible according to (4.6).

$$\begin{aligned}\underbrace{\max\bigl(\mathbb{H}_\infty(Y) - (n - \log_2(|\mathcal{M}|)), 0\bigr)}_{\text{worst-case}} &\leq \tilde{\mathbb{H}}_\infty(Y|H) \\ &\leq \underbrace{\min\bigl(\log_2(|\mathcal{M}|), \mathbb{H}_\infty(Y)\bigr)}_{\text{best-case}}.\end{aligned} \tag{4.15}$$

We also improve upon the rather trivial upper bounds [39] that comprehend the rightmost inequality in (4.15). Our lower and upper bounds combined define a relatively narrow interval in which the exact value of the residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H)$ is enclosed. We cover a variety of codes, regardless of their algebraic complexity, as well as a variety of distributions. Our bounds are easy-to-evaluate and are able to handle large codes. Although derived for the code-offset sketch of Dodis et al. [39] in particular, the bounds hold for all seven constructions in Table 4.4 due to the previously established equivalencies in Section 4.3.2.

Distributions

Our derivation is generic in the sense that a large variety of distributions of response Y could be covered. We only require that the set of all possible responses $\mathcal{Y} = \{0,1\}^n$ can be partitioned into a limited number of subsets $\mathcal{Y}_j$, with $j \in [1, \lambda]$, such that all elements of $\mathcal{Y}_j$ have the same probability of occurrence p_j. Formally, $\mathbb{P}(Y = \mathbf{y}) = p_j$ if and only if $\mathbf{y} \in \mathcal{Y}_j$. These probabilities are strictly monotonically decreasing, i.e., $p_j > p_{j+1}$, with $j \in [1, \lambda - 1]$. Occasionally, $p_\lambda = 0$. The ingoing min-entropy is easily computed as $\mathbb{H}_\infty(Y) = -\log_2(p_1)$.

We evaluate bounds on the residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H)$. The runtime of the corresponding algorithms is roughly proportional to the number of subsets λ. The crucial observation is that a small λ might suffice to capture realistic models of an SRAM-like PUF. Below, we describe a parameterized distribution of Y for both biased and spatially correlated PUFs. Both distributions are to be considered as proof-of-concept models, used in showcasing the feasibility of a new research direction. If a given PUF is not approximated accurately enough, one can opt for an alternative and possibly more complicated *second-order* distribution. As long as λ is limited, bounds can be evaluated in milliseconds to minutes on a standard desktop computer.

- *Biased distribution.* We assume that response bits Y_i, with $i \in [1, n]$, are i.i.d. such that $\mathbb{P}(Y_i = 1) = p_{\text{bias}}$, where $p_{\text{bias}} \in [0, 1]$ is a real-valued constant. For $p_{\text{bias}} = 1/2$, this corresponds to a uniform distribution.

- *Correlated distribution.* We assume that responses are distributed such that $\mathbb{P}(Y_i = Y_{i+1}) = p_{\text{corr}}$, with $i \in [1, n-1]$ and a real-valued constant $p_{\text{corr}} \in [0, 1]$. This extends to (4.16) for larger neighborhoods. There is no bias, i.e., $\mathbb{P}(Y_i = 1) = 1/2$. For $p_{\text{corr}} = 1/2$, the latter model corresponds to a uniform distribution. Although spatial correlations are frequently encountered in experimental work, e.g., byte-level dependencies for the SRAM PUFs in [172, 7], these are often neglected in information-theoretic work due to their complexity. We hope that our results may help turn the tide on this.

$$p_{\text{corr},i,j} = \mathbb{P}(Y_i = Y_j) = \sum_{u=0}^{\lfloor |i-j|/2 \rfloor} \mathsf{f}_{\text{bino}}(2u; |i-j|, 1 - p_{\text{corr}}), \qquad \text{with } i, j \in [1, n]. \tag{4.16}$$

Table 4.5 specifies the subsets $\mathcal{Y}_j$ for both proof-of-concept distributions. For the biased distribution, we partition the set $\mathcal{Y}$ according to $\mathsf{HW}(\mathbf{y})$. The probability that a random sample $\mathbf{y}$ belongs to $\mathcal{Y}_j$ can be described by a binomial distribution with $j-1$ successes for n Bernoulli trials that each have success probability $p_\star = \min(p_{\text{bias}}, 1 - p_{\text{bias}})$. For the correlated distribution, we partition the set $\mathcal{Y}$ according to $\sum_{i=1}^{n-1} \mathsf{HD}(y_i, y_{i+1})$, i.e., the number of transitions in $\mathbf{y}$. The responses in subset $\mathcal{Y}_j$ exhibit $j-1$ transitions and obey either one out of two forms, i.e., $\mathbf{y} = (\mathbf{0\,1\,0} \cdots)$ or $\mathbf{y} = (\mathbf{1\,0\,1} \cdots)$. A related observation is that if $\mathbf{y} \in \mathcal{Y}_j$, then so is its additive inverse, i.e., $\overline{\mathbf{y}} \in \mathcal{Y}_j$. This explains the factors 2 and $1/2$ everywhere. The cardinalities $|\mathcal{Y}_j|$ are further determined with *stars and bars* combinatorics [44]. To be precise, we separate n indistinguishable stars into j distinguishable bins by adding $j-1$ out of $n-1$ bars.

We treat the degenerate case $p_{\text{bias}} = p_{\text{corr}} = 1/2$, i.e., a uniform distribution, separately. There is only one set then. Formally, $\lambda = 1$, $|\mathcal{Y}_1| = 2^n$, and $p_1 = 1/2^n$. As proven by Reyzin [137], the min-entropy loss of a secure sketch is maximal for a uniformly distributed input, which makes this a case of special interest.

Generic Bounds

We derive generic bounds that can be applied to any distribution of response Y where the number of subsets λ is limited. Equation (4.17) holds for the code-offset construction of Dodis et al. [39], given that a codeword $\mathbf{w}$ is selected uniformly at random during the enrollment.

Table 4.5: Subsets $\mathcal{Y}_j$ for both the biased and spatially correlated distribution. We define $p_\star = \min(p_{\text{bias}}, 1 - p_{\text{bias}})$ and $p_\star = \min(p_{\text{corr}}, 1 - p_{\text{corr}})$ respectively.

Biased distribution			Correlated distribution		
j	$\lvert\mathcal{Y}_j\rvert$	p_j	j	$\lvert\mathcal{Y}_j\rvert$	p_j
1	1	$(1-p_\star)^n$	1	2	$\frac{1}{2}(1-p_\star)^{n-1}$
2	n	$p_\star(1-p_\star)^{n-1}$	2	$2(n-1)$	$\frac{1}{2}p_\star(1-p_\star)^{n-2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
j	$\binom{n}{j-1}$	$(p_\star)^{j-1}(1-p_\star)^{n-j+1}$	j	$2\binom{n-1}{j-1}$	$\frac{1}{2}(p_\star)^{j-1}(1-p_\star)^{n-j}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	n	$(p_\star)^{n-1}(1-p_\star)$	$n-1$	$2(n-1)$	$\frac{1}{2}(p_\star)^{n-2}(1-p_\star)$
$n+1$	1	$(p_\star)^n$	n	2	$\frac{1}{2}(p_\star)^{n-1}$

$$\mathbb{P}\big((H=\mathbf{h})|(Y=\mathbf{y})\big) = \begin{cases} 1/|\mathcal{M}|, & \text{if } \exists\, \mathbf{w}, \mathbf{h} = \mathbf{y} \oplus \mathbf{w}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.17}$$

Equation (4.18) applies Bayes' rule to the definition of conditional min-entropy in (4.8) and fills in (4.17). The 0 case is resolved by switching variables for the max operator. A direct exhaustive evaluation of the resulting formula requires up to $2^n|\mathcal{M}|$ operations.

$$\begin{aligned} \tilde{\mathbb{H}}_\infty(Y|H) &= -\log_2\left(\sum_{\mathbf{h}\in\mathcal{H}} \cancel{\mathbb{P}(H=\mathbf{h})} \max_{\mathbf{y}\in\mathcal{Y}} \frac{\mathbb{P}(Y=\mathbf{y})\mathbb{P}((H=\mathbf{h})|(Y=\mathbf{y}))}{\cancel{\mathbb{P}(H=\mathbf{h})}}\right) \\ &= -\log_2\left(\frac{1}{|\mathcal{M}|}\sum_{\mathbf{h}\in\mathcal{H}} \max_{\mathbf{w}\in\mathcal{W}} \mathbb{P}(Y=\mathbf{h}\oplus\mathbf{w})\right). \end{aligned} \tag{4.18}$$

For linear codes, the workload can be reduced substantially. With a similar derivation as before, we rewrite (4.10) as shown in (4.19). Up to 2^n operations suffice. Nevertheless, direct evaluation is only feasible for small codes. In order to handle large codes, as is typically the case for a practical key generator, further steps are required.

$$\tilde{\mathbb{H}}_\infty(Y|H) = -\log_2\Big(\sum_{\mathbf{e}\in\mathcal{E}} \max_{\mathbf{w}\in\mathcal{W}} \mathbb{P}(Y = \mathbf{e} \oplus \mathbf{w})\Big). \tag{4.19}$$

Observe that (4.18) iterates over all helper vectors $\mathbf{h}$ and selects each time the most likely response $\mathbf{y}$ that is within range, via the addition of a codeword $\mathbf{w} \in \mathcal{W}$. We now reverse the roles, as is shown in Figure 4.5. We iterate over all responses $\mathbf{y}$, from most likely to least likely, i.e., from $\mathcal{Y}_1$ to $\mathcal{Y}_\lambda$. Within a certain subset $\mathcal{Y}_j$, the order of the $\mathbf{y}$'s may be chosen arbitrarily. Subsequently, we assign helper vectors $\mathbf{h}$ to each $\mathbf{y}$, as represented by the black squares, until the set $\mathcal{H}$ of size 2^n is depleted. For each assigned $\mathbf{h}$, we assume that the corresponding $\mathbf{y}$ is the most likely vector, according to (4.18). Let $s_j^{\mathbf{h}}$ denote the number of black squares that are assigned to subset $\mathcal{Y}_j$. The residual min-entropy is then easily computed as in (4.20).

$$\tilde{\mathbb{H}}_\infty(Y|H) = -\log_2\left(\frac{1}{|\mathcal{M}|}\sum_{j=1}^{\lambda} s_j^{\mathbf{h}} p_j\right). \tag{4.20}$$

Both linear and non-linear codes are supported by the former graphical representation. Nevertheless, we elaborate linear codes as a special case due to their practical relevance. Figure 4.6 swaps the order of iteration in (4.19). Only one row suffices, i.e., each column of helper data vectors $\mathbf{h}$ in Figure 4.5 is condensed to a single square. Black and white squares are now assigned to cosets, as represented by their coset leaders $\mathbf{e}$. Let $s_j^{\mathbf{e}}$ denote the number of black squares that are assigned to subset $\mathcal{Y}_j$. The residual min-entropy is then easily computed as in (4.21), hereby dropping denominator $|\mathcal{M}|$ compared to (4.20), given that $s_j^{\mathbf{h}} = 2^k\, s_j^{\mathbf{e}}$.

$$\tilde{\mathbb{H}}_\infty(Y|H) = -\log_2\left(\sum_{j=1}^{\lambda} s_j^{\mathbf{e}} p_j\right). \tag{4.21}$$

In the worst-case scenario, the most likely responses $\mathbf{y}$ all map to unique helper vectors $\mathbf{h}$, without overlap, resulting in a lower bound on the residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H)$. For a linear code, this would be the case if the first 2^{n-k} responses $\mathbf{y}$ all belong to different cosets. In the best-case scenario, our sequence of $\mathbf{y}$'s exhibits maximum overlap in terms of helper data $\mathbf{h}$, resulting in an upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$. For a linear code, this would be the case if the first 2^k $\mathbf{y}$'s all map to the same coset, and this repeated for all 2^{n-k} cosets. Algorithms 1 and 2 comprehend a literal transcript of Figure 4.5 and compute

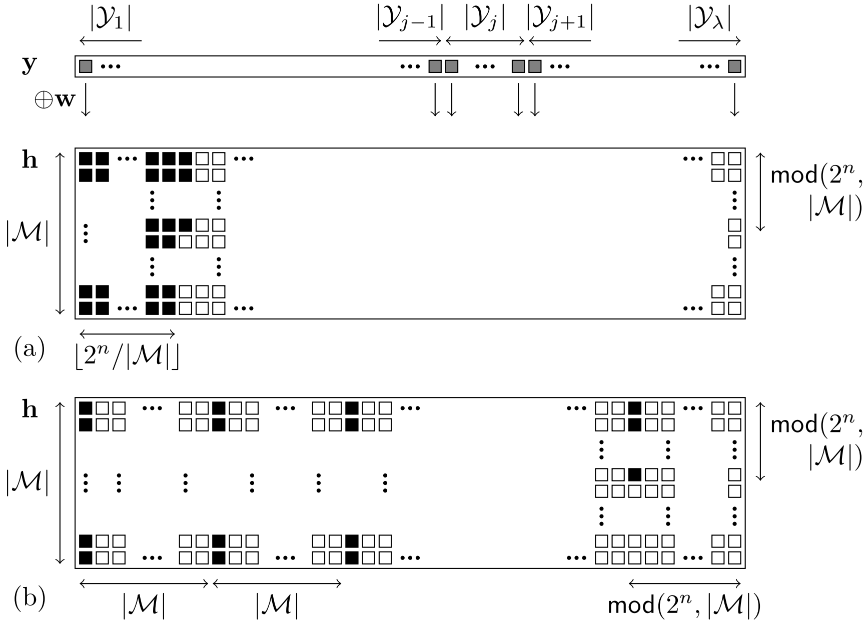


Figure 4.5: Reversal of the roles in (4.18). (a) A lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$. (b) An upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$. Black squares represent terms that contribute to $\tilde{\mathbb{H}}_\infty(Y|H)$, one for each $\mathbf{w} \in \mathcal{W}$. White squares represent non-contributing terms, overruled by the max operator. In general, there are few black squares but many white squares, 2^n versus $(|\mathcal{M}| - 1)2^n$ to be precise. For block codes, i.e., $|\mathcal{M}| = 2^k$, the last column of black squares is completely filled.

the lower bound and upper bound respectively. Auxiliary variables $s^{\mathbf{h}}$ and $s^{\mathbf{y}}$ accumulate black and gray squares respectively. We abstain from special case algorithms for linear codes, although it would result in a few simplifications.

Algorithms 1 and 2 may now be applied to a variety of distributions. For a uniform distribution, the lower and upper bound both evaluate to $\tilde{\mathbb{H}}_\infty(Y|H) = log_2(|\mathcal{M}|)$, regardless of other code specifics. Or simply k, for block codes in particular. The min-entropy loss is hence exactly $(n - k)$ bits, given that $\mathbb{H}_\infty(Y) = n$. Reyzin's proof [137] therefore implies that the universal $(n - k)$ bound cannot be tightened any further. Although numerical results are fairly presentable already for the biased and correlated distributions, we further tighten their bounds first.

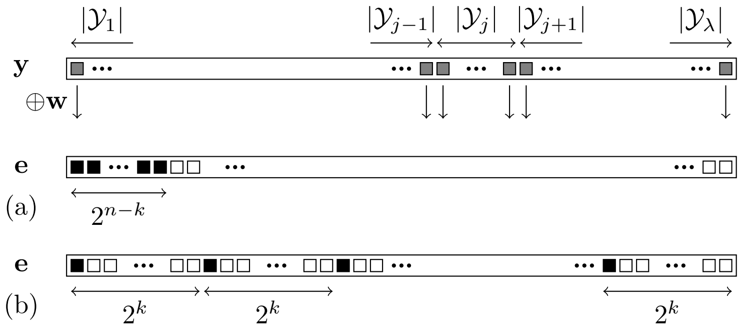


Figure 4.6: Reversal of the roles in (4.19), as applied to linear codes. (a) A lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$. (b) An upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$. Black squares represent terms that contribute to $\tilde{\mathbb{H}}_\infty(Y|H)$, one for each $\mathbf{e} \in \mathcal{E}$. White squares represent non-contributing terms, overruled by the max operator.

Algorithm 1: BoundWorstCase

```
Input: List $\langle |\mathcal{Y}_j|, p_j \rangle$
Output: Lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$
$j, p, s^{\mathbf{h}} \leftarrow 0$
while $s^{\mathbf{h}} < 2^n$ do
    $j \leftarrow j + 1$
    $s_j^{\mathbf{h}} \leftarrow \min(|\mathcal{Y}_j||\mathcal{M}|, 2^n - s^{\mathbf{h}})$
    $s^{\mathbf{h}} \leftarrow s^{\mathbf{h}} + s_j^{\mathbf{h}}$
    $p \leftarrow p + s_j^{\mathbf{h}} p_j$
$\tilde{\mathbb{H}}_\infty(Y|H) \leftarrow -\log_2(p/|\mathcal{M}|)$
```

Algorithm 2: BoundBestCase

```
Input: List $\langle |\mathcal{Y}_j|, p_j \rangle$
Output: Upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$
$j, p, s^{\mathbf{h}}, s^{\mathbf{y}} \leftarrow 0$
while $s^{\mathbf{h}} < 2^n$ do
    $j \leftarrow j + 1$
    $s^{\mathbf{y}} \leftarrow s^{\mathbf{y}} + |\mathcal{Y}_j|$
    $s_j^{\mathbf{h}} \leftarrow \lceil (s^{\mathbf{y}} - s^{\mathbf{h}})/|\mathcal{M}| \rceil |\mathcal{M}|$
    $s_j^{\mathbf{h}} \leftarrow \min(\max(s_j^{\mathbf{h}}, 0), 2^n - s^{\mathbf{h}})$
    $s^{\mathbf{h}} \leftarrow s^{\mathbf{h}} + s_j^{\mathbf{h}}$
    $p \leftarrow p + s_j^{\mathbf{h}} p_j$
$\tilde{\mathbb{H}}_\infty(Y|H) \leftarrow -\log_2(p/|\mathcal{M}|)$
```

Tighter Bounds

Tighter bounds can be obtained by leveraging code and distribution properties more effectively. Algorithms 3 and 4 generalize Algorithms 1 and 2 respectively. In the former case, an additional input imposes an upper bound on the accumulated number of black squares, i.e., $\forall j, (s_1^{\mathbf{h}} + s_2^{\mathbf{h}} + \ldots + s_j^{\mathbf{h}}) \leq (\beta_1^{\mathbf{h}} + \beta_2^{\mathbf{h}} + \ldots + \beta_j^{\mathbf{h}})$. In the latter case, an additional input imposes a lower bound on the accumulated number of black squares, i.e., $\forall j, (s_1^{\mathbf{h}} + s_2^{\mathbf{h}} + \ldots + s_j^{\mathbf{h}}) \geq (\alpha_1^{\mathbf{h}} + \alpha_2^{\mathbf{h}} + \ldots + \alpha_j^{\mathbf{h}})$. We now provide several examples.

Algorithm 3: BoundWorstCase2

```
Input: List $\langle |\mathcal{Y}_j|, p_j, \beta_j^{\mathbf{h}} \rangle$
Output: Lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$
$j, p, s^{\mathbf{h}}, \beta^{\mathbf{h}} \leftarrow 0$
while $s^{\mathbf{h}} < 2^n$ do
    $j \leftarrow j + 1$
    $\beta^{\mathbf{h}} \leftarrow \beta^{\mathbf{h}} + \beta_j^{\mathbf{h}}$
    $s_j^{\mathbf{h}} \leftarrow \min(|\mathcal{Y}_j||\mathcal{M}|, \beta^{\mathbf{h}} - s^{\mathbf{h}})$
    $s_j^{\mathbf{h}} \leftarrow \min(s_j^{\mathbf{h}}, 2^n - s^{\mathbf{h}})$
    $s^{\mathbf{h}} \leftarrow s^{\mathbf{h}} + s_j^{\mathbf{h}}$
    $p \leftarrow p + s_j^{\mathbf{h}} p_j$
$\tilde{\mathbb{H}}_\infty(Y|H) \leftarrow -\log_2(p/|\mathcal{M}|)$
```

Algorithm 4: BoundBestCase2

```
Input: List $\langle |\mathcal{Y}_j|, p_j, \alpha_j^{\mathbf{h}} \rangle$
Output: Upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$
$j, p, s^{\mathbf{h}}, s^{\mathbf{y}}, \alpha^{\mathbf{h}} \leftarrow 0$
while $s^{\mathbf{h}} < 2^n$ do
    $j \leftarrow j + 1$
    $s^{\mathbf{y}} \leftarrow s^{\mathbf{y}} + |\mathcal{Y}_j|$
    $\alpha^{\mathbf{h}} \leftarrow \alpha^{\mathbf{h}} + \alpha_j^{\mathbf{h}}$
    $s_j^{\mathbf{h}} \leftarrow \lceil (s^{\mathbf{y}} - s^{\mathbf{h}})/|\mathcal{M}| \rceil |\mathcal{M}|$
    $s_j^{\mathbf{h}} \leftarrow \max(s_j^{\mathbf{h}}, \alpha^{\mathbf{h}} - s^{\mathbf{h}}, 0)$
    $s_j^{\mathbf{h}} \leftarrow \min(s_j^{\mathbf{h}}, 2^n - s^{\mathbf{h}})$
    $s^{\mathbf{h}} \leftarrow s^{\mathbf{h}} + s_j^{\mathbf{h}}$
    $p \leftarrow p + s_j^{\mathbf{h}} p_j$
$\tilde{\mathbb{H}}_\infty(Y|H) \leftarrow -\log_2(p/|\mathcal{M}|)$
```

We further tighten the lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ for the correlated distribution. The improvement applies to linear codes that have the all-ones vector $\mathbf{1}$ of length n as a codeword. This includes Reed–Muller codes of any order [108]. This also includes many BCH, Hamming and repetition codes, on the condition that these are cyclic and having d odd, as easily proven hereafter. Consider an arbitrary codeword with Hamming weight d. XORing all 2^n circular shifts of this codeword results in the all-ones codeword, which ends the proof. As mentioned before, each set $\mathcal{Y}_j$ of the correlated distribution can be partitioned into pairs $\{\mathbf{y}, \overline{\mathbf{y}}\}$, with $\overline{\mathbf{y}}$ the additive inverse of $\mathbf{y}$. Paired inputs belong to the same coset, i.e., maximum overlap in terms of helper data $\mathbf{h}$. Therefore, we impose the cumulative upper bound in (4.22).

$$\beta_j^{\mathbf{h}} = |\mathcal{M}| \frac{|\mathcal{Y}_j|}{2} = 2^{k-1} |\mathcal{Y}_j|. \tag{4.22}$$

For instance, consider linear/cyclic $[n, k = 1, d = n]$ repetition codes, i.e., having generator matrix $\mathbf{G} = \mathbf{1}$, with n odd. Algorithms BoundWorstCase2 and BoundBestCase then converge to the exact result $\tilde{\mathbb{H}}_\infty(Y|H) = 1$, not depending on parameter p_{corr}. This is the best-case scenario, given the universal bound $\tilde{\mathbb{H}}_\infty(Y|H) \leq k$. Figure 4.7 illustrates the former with squares for $n = 5$. The result also holds if the repetition code is neither linear/cyclic nor odd. As long as $\mathbf{w}_1 \oplus \mathbf{w}_2 = \mathbf{1}$, the elements of each $\mathcal{Y}_j$ can be paired into cosets. Although the term coset is usually preserved for linear codes, translations of a non-linear repetition code are either disjunct or coincide and still partition the space $\{0,1\}^n$. As a side note, the result offers a refutation of the *repetition code pitfall* of Koeberl et al. [87], a work that overlooks that $(n - k)$ is an upper bound only.

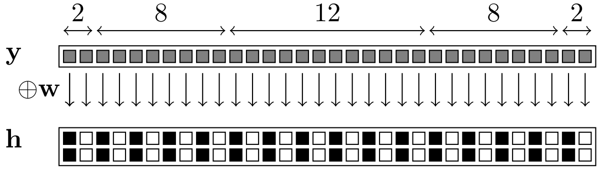


Figure 4.7: The exact residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H)$ for the correlated distribution and an $[n = 5, k = 1, d = 5]$ repetition code.

We improve the upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ for both the biased and correlated distribution. In particular, we take minimum distance d into account. The main insight is that two slightly differing inputs $\mathbf{y}_u \neq \mathbf{y}_v$ do not overlap in terms of helper data $\mathbf{h}$. More precisely, if $\mathsf{HD}(\mathbf{y}_u, \mathbf{y}_v) \in [1, d-1]$, then $\{\mathbf{y}_u \oplus \mathbf{w} : \mathbf{w} \in \mathcal{W}\} \cap \{\mathbf{y}_v \oplus \mathbf{w} : \mathbf{w} \in \mathcal{W}\} = \varnothing$. For the biased distribution, the following holds: $\mathsf{HD}(\mathbf{y}_u, \mathbf{y}_v) \in [1, d-1]$ if $\mathbf{y}_u \neq \mathbf{y}_v$ and $\mathbf{y}_u, \mathbf{y}_v \in (\mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \ldots \cup \mathcal{Y}_{t+1})$. Or stated otherwise, the elements of the first $t+1$ sets all result in unique $\mathbf{h}$'s. Therefore, we can impose the constraint given in (4.23). Figure 4.8 depicts the squares.

$$\alpha_j^{\mathbf{h}} = \begin{cases} |\mathcal{Y}_j||\mathcal{M}|, & \text{if } j \in [1, t+1], \\ 0, & \text{otherwise.} \end{cases} \tag{4.23}$$

There is an interesting observation for perfect codes in particular. As clear from the Hamming bound in (2.11), all unique $\mathbf{h}$'s are covered by the first $t+1$ sets exclusively. BoundWorstCase and BoundBestCase2 hence produce the same output, implying that the residual min-entropy is evaluated exactly, as further simplified in (4.24). For $[n, k = 1, d = n]$ repetition codes with n odd, the result provides a refutation of the *repetition code pitfall* [87]. Maes et al. [111] later presented a similar contribution, differing in its use of Shannon entropy rather than min-entropy.

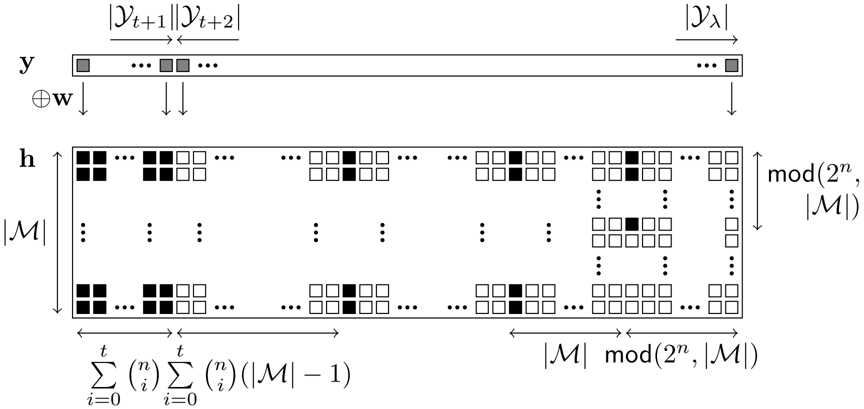

$|\mathcal{Y}_{t+1}|$
$|\mathcal{Y}_{t+2}|$
$|\mathcal{Y}_{\lambda}|$
$\mathbf{y}$
$\oplus\mathbf{w}$
$\mathbf{h}$
$|\mathcal{M}|$
$\mathsf{mod}(2^n, |\mathcal{M}|)$
$\sum_{i=0}^{t}\binom{n}{i}$
$\sum_{i=0}^{t}\binom{n}{i}(|\mathcal{M}|-1)$
$|\mathcal{M}|$
$\mathsf{mod}(2^n, |\mathcal{M}|)$

Figure 4.8: A tightened upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ for the biased distribution, hereby making use of (4.23).

$$\begin{aligned}\tilde{\mathbb{H}}_\infty(Y|H) &= -\log_2\Bigl(\textstyle\sum_{j=1}^{t+1}|\mathcal{Y}_j|\,p_j\Bigr)\\ &= -\log_2\Bigl(\mathsf{F}_{\text{bino}}\bigl(t;n,\min(p_{\text{bias}},1-p_{\text{bias}})\bigr)\Bigr).\end{aligned}\tag{4.24}$$

For codes that do not happen to be perfect, there is still margin for improvement. We inject some promising thoughts but abstain from numerical results later-on. Consider a linear code of which the Hamming weight distribution of the coset leaders $\mathbf{e}$ is well-understood. Let $|\mathcal{E}_h|$ denote the number of cosets such that $h = \mathsf{HW}(\mathbf{e})$. Clearly, $|\mathcal{E}_h| = \binom{n}{h}$ for $h \in [0,t]$. Our interest concerns $|\mathcal{E}_h|$ for $h > t$, all of which are exactly known in the ideal case, as in [30] for certain BCH codes. The largest h for which $|\mathcal{E}_h| > 0$ is also referred to as the *covering radius* h_{cr} of the code. For a bias $p_{\text{bias}} < 1/2$, (4.25) comprehends the exact residual min-entropy. The latter expression extends to $p_{\text{bias}} > 1/2$ in case the all-ones vector $\mathbf{1}$ is a codeword. This includes Reed–Muller codes as well as cyclic codes with d odd, as has been argued earlier on. If only bounds on $|\mathcal{E}_h|$ and/or h_{cr} are known, one might still be able to further tighten the bounds on $\tilde{\mathbb{H}}_\infty(Y|H)$ correspondingly.

$$\tilde{\mathbb{H}}_\infty(Y|H) = -\log_2\left(\frac{1}{|\mathcal{M}|}\sum_{h=0}^{h_{\mathsf{cr}}}|\mathcal{E}_h|\,|\mathcal{M}|\,p_{h+1}\right) = -\log_2\left(\sum_{h=0}^{h_{\mathsf{cr}}}|\mathcal{E}_h|\,p_{h+1}\right).\tag{4.25}$$

For instance, consider $[n, k = 1, d = n]$ repetition codes with n even. These form the non-perfect and therefore less popular counterpart of n odd. Inputs $\mathbf{y}$ belonging to $\mathcal{Y}_j$ and $\mathcal{Y}_{n+2-j}$ are still paired in order to form the cosets. Unlike n odd, there is a central set $\mathcal{Y}_{t+2}$ that contains both members of each pair. Therefore, $h_{\mathsf{cr}} = t + 1$ and $|\mathcal{E}_{t+1}| = |\mathcal{Y}_{t+2}|/2$. As argued before, the operational principles of cosets extend to non-linear repetition codes. Fig. 4.9 depicts the squares for $n = 4$. Equation (4.26) evaluates the residual min-entropy.

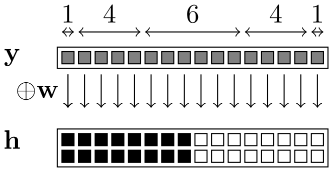


Figure 4.9: The exact residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H)$ for the biased distribution and an $[n = 4, k = 1, d = 4]$ repetition code.

$$\begin{aligned}\tilde{\mathbb{H}}_\infty(Y|H) = -\log_2\Bigg(&\mathsf{F}_{\text{bino}}\big(t; n, \min(p_{\text{bias}}, 1 - p_{\text{bias}})\big)\\ &+\frac{1}{2}\binom{n}{\frac{n}{2}}\big(p_{\text{bias}}(1 - p_{\text{bias}})\big)^{\frac{n}{2}}\Bigg).\end{aligned} \tag{4.26}$$

Also for the correlated distribution, the distance d might be incorporated to tighten the upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$. First of all, we assign $|\mathcal{M}|$ unique $\mathbf{h}$'s to one out of two elements in $\mathcal{Y}_1$. For ease of understanding, assume $\mathbf{y} = \mathbf{0}$, comprehending the first case in (4.27). For each set $\mathcal{Y}_j$, with $j \in [2, n]$, we then count the number of inputs $\mathbf{y} \in \mathcal{Y}_j$ such that $h = \mathsf{HW}(\mathbf{y}) \leq t$. The latter constraint guarantees all assigned $\mathbf{h}$'s to be unique. We distinguish between two forms, $\mathbf{y} = (\mathbf{0}\|\mathbf{1}\|\mathbf{0}\| \ldots)$ and $\mathbf{y} = (\mathbf{1}\|\mathbf{0}\|\mathbf{1}\| \ldots)$, resulting in two main terms. For each form, we apply *stars and bars* combinatorics twice. In particular, we assign h indistinguishable stars, i.e., ones, to distinguishable bins and independently also for $n - h$ zeros. Note that $\alpha_j^{\mathbf{h}} = 0$ for $j > 2t + 1$. To ensure formula correctness, one may verify numerically that $\alpha_1^{\mathbf{h}} + \alpha_2^{\mathbf{h}} + \ldots + \alpha_{2t+1}^{\mathbf{h}}$ equals the left hand side of the Hamming bound in (2.11).

$$\alpha_j^{\mathbf{h}} = \begin{cases} |\mathcal{M}|, & \text{if } j = 1,\\ |\mathcal{M}|\Big(\sum_{h=\lfloor j/2\rfloor}^{t} \binom{h-1}{\lfloor j/2\rfloor-1}\binom{n-h-1}{\lceil j/2\rceil-1} & \\ \quad + \sum_{h=\lceil j/2\rceil}^{t} \binom{h-1}{\lceil j/2\rceil-1}\binom{n-h-1}{\lfloor j/2\rfloor-1}\Big), & \text{otherwise.}\end{cases} \tag{4.27}$$

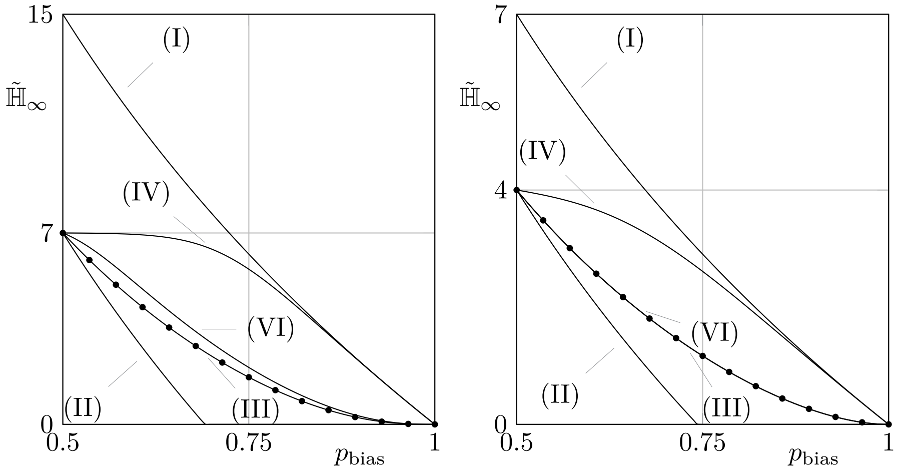
Numerical Results

Figure 4.10 presents numerical results for various BCH codes. We focus on small codes, given that these allow for an exact exhaustive evaluation of the residual min-entropy using (4.18) and/or (4.19). This way, the tightness of various bounds can be assessed adequately. Figure 4.10(d) nevertheless demonstrates that our algorithms are able to support large codes equally well, in compliance with a practical key generator. Note that only half of the bias interval $p_{\text{bias}} \in [0, 1]$ is depicted. The reason is that all curves mirror around the vertical axis of symmetry $p_{\text{bias}} = {}^{1}/_{2}$. The same holds for the correlated distribution with parameter p_{corr}.

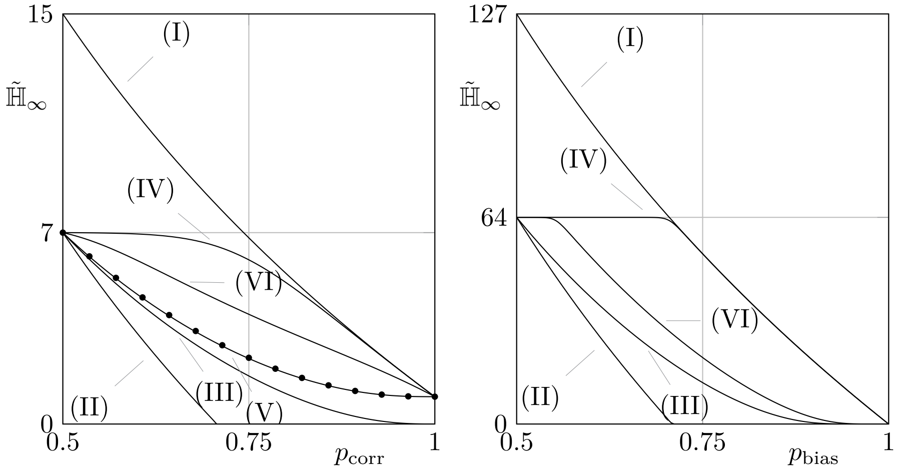
Especially the lower bounds perform well, which benefits a conservative system provider. The best lower bounds in Figure. 4.10(a), (b) and (c) visually coincide with the exact result. The gap with the $(n-k)$ bound is the most compelling around $p_{\text{bias}}, p_{\text{corr}} \approx 0.7$, where the corresponding curves hit the horizontal axis $\widetilde{\mathbb{H}}_\infty(Y|H) = 0$. Also our upper bounds are considerably tighter than their more general alternatives in (4.15). Nevertheless, the latter bounds remain open for further improvement, with the exception of Figure 4.10(b). An $[n = 7, k = 4, d = 3]$ code is perfect and lower and upper bounds then converge to the exact result for the biased distribution.

Table 4.6 quantifies the reduction in implementation footprint for a fuzzy extractor that produces a 128-bit key from a biased PUF. A concatenated code $\zeta_2 \circ \zeta_1$ is independently applied to q non-overlapping blocks of PUF response bits. We consider all 70 BCH codes ζ_1 with $n_2 \leq 255$ and all 7 repetition codes ζ_2 with $n_2 \leq 13$ and n_2 odd. The degenerate case $n_2 = 1$ ensures that our search space of 490 codes includes stand-alone BCH codes. Given a bias p_{bias} and an expected bit error rate $\mathbb{E}_{v \leftarrow V}[P_{\text{error},y}]$, we retain the code that minimizes the number of response bits n while satisfying the following two constraints. First, the residual min-entropy $\widetilde{\mathbb{H}}_\infty(Y|H) \geq 128$. Due to i.i.d. response bits Y_i, algorithm BoundWorstCase can be applied to an $[n_1 \cdot n_2, k_1, d_1 \cdot d_2]$ umbrella code and the residual min-entropy thereof is multiplied by q. A second constraint states that the expected failure rate for key reconstruction $\mathbb{E}_{v \leftarrow V}[P_{\text{fail}}] \leq 10^{-6}$. Due to i.i.d. response bits Y_i, we can compute $\mathbb{E}_{v \leftarrow V}[P_{\text{fail}}]$ as given in (4.14).

According to the $(n-k)$ bound, a modest bias is highly detrimental already. Most notably, for $p_{\text{bias}} = 0.56$, there is no code within the search space that satisfies all the design constraints. According to the newly derived bound, PUFs with a considerable bias can be supported. We emphasize that a carefully balanced PUF circuit with a custom-designed layout tends to have a low bias. Notable cases of a high bias can typically be attributed to an asymmetry in either the PUF circuit or its layout, e.g., the D flip-flop PUF in [113, 104, 89,



(a) Bias; $[n = 15, k = 7, d = 5]$. (b) Bias; $[n = 7, k = 4, d = 3]$.



(c) Correlation; $[n = 15, k = 7, d = 5]$. (d) Bias; $[n = 127, k = 64, d = 21]$.

Figure 4.10: The min-entropy loss of a secure sketch for various BCH codes. Dots correspond to an exact exhaustive evaluation of (4.18)/(4.19). The legend of the curves is as follows. (I) The ingoing min-entropy $\mathbb{H}_\infty(Y) = -\log_2(p_1)$. (II) The lower bound $\tilde{\mathbb{H}}_\infty(Y|H) = \max(\mathbb{H}_\infty(Y) - (n-k), 0)$. (III) The lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ according to BoundWorstCase. (IV) The upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ according to BoundBestCase. (V) The lower bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ according to BoundWorstCase2. (VI) The upper bound on $\tilde{\mathbb{H}}_\infty(Y|H)$ according to BoundBestCase2.

Table 4.6: The implementation footprint of a practical fuzzy extractor, using the $(n-k)$ bound and the BoundWorstCase algorithm respectively. We assume i.i.d. response bits Y_i with bias p_{bias} and noise component $\sigma_{\text{noise}} = 0.325$. Error-correction relies on the concatenation of a BCH code ζ_1 and a repetition code ζ_2, with size $n_1 \in \{7, 15, 31, 63, 127, 255\}$ and $n_2 \in \{1, 3, 5, 7, 9, 11, 13\}$ respectively. Each row specifies the concatenated code that minimizes the number of PUF response bits n while satisfying the constraints, i.e., a residual min-entropy $\tilde{\mathbb{H}}_\infty(Y|H) \geq 128$ and an expected failure rate $\mathbb{E}[P_{\text{fail}}] \leq 10^{-6}$. The helper data size of the code-offset sketch equals n also and is hence minimized simultaneously.

Bound	p_{bias}	$\mathbb{E}[P_{\text{error}}]$	$q \times [n_2, k_2, d_2] \circ [n_1, k_1, d_1]$	$\tilde{\mathbb{H}}_\infty(Y\|H)$	PUF size n	$\mathbb{E}[P_{\text{fail}}]$
$n-k$	0.50	$\approx 10.0\%$	$2 \times [5, 1, 5] \circ [127, 64, 21]$	128	1270	$\approx 3.26\text{E}{-8}$
	0.52	$\approx 10.0\%$	$3 \times [3, 1, 3] \circ [255, 87, 53]$	≈ 131.1	2295	$\approx 1.44\text{E}{-8}$
	0.54	$\approx 9.96\%$	$10 \times [5, 1, 5] \circ [255, 155, 27]$	≈ 134.4	12750	$\approx 5.56\text{E}{-7}$
	0.56	$\approx 9.90\%$	No code within the search space satisfies the constraints.			
new theory	0.50	$\approx 10.0\%$	$2 \times [5, 1, 5] \circ [127, 64, 21]$	128	1270	$\approx 3.26\text{E}{-8}$
	0.52	$\approx 10.0\%$	$1 \times [5, 1, 5] \circ [255, 163, 25]$	≈ 134.3	1275	$\approx 4.27\text{E}{-7}$
	0.54	$\approx 9.96\%$	$2 \times [3, 1, 3] \circ [255, 99, 47]$	≈ 132.5	1530	$\approx 5.35\text{E}{-7}$
	0.56	$\approx 9.90\%$	$3 \times [3, 1, 3] \circ [255, 87, 53]$	≈ 131.3	2295	$\approx 9.90\text{E}{-9}$
	0.58	$\approx 9.81\%$	$2 \times [5, 1, 5] \circ [255, 163, 25]$	≈ 130.0	2550	$\approx 4.85\text{E}{-7}$
	0.60	$\approx 9.71\%$	$3 \times [5, 1, 5] \circ [255, 155, 27]$	≈ 129.5	3825	$\approx 6.96\text{E}{-8}$
	0.62	$\approx 9.58\%$	$4 \times [5, 1, 5] \circ [255, 163, 25]$	≈ 130.4	5100	$\approx 4.42\text{E}{-7}$
	0.64	$\approx 9.42\%$	$10 \times [3, 1, 3] \circ [255, 99, 47]$	≈ 132.8	7650	$\approx 3.87\text{E}{-7}$
	0.66	$\approx 9.24\%$	$17 \times [3, 1, 3] \circ [255, 99, 47]$	≈ 129.7	13005	$\approx 3.28\text{E}{-7}$

81] with $p_{\text{bias}} > 0.7$. For low-bias PUFs, with $p_{\text{bias}} \in [0.42, 0.58]$, a stand-alone secure sketch will later turn out to be competitive with state-of-the-art debiasing schemes [187, 62, 104, 170, 111].

4.3.6 Reusability

A secure sketch is occasionally claimed to be reusable, as exemplified by the so-called reverse fuzzy extractor protocols [171, 110, 7], but this is incorrect. Our counterexamples show that repeated helper data exposure may result in additional min-entropy loss. The revealed flaw is attributed to the misuse of a reusability proof of Boyen [23]. For the code-offset construction with linear codes, the exposure of $\mathbf{h}^{(1)} \leftarrow \mathsf{SSGen}(\mathbf{y})$ and $\mathbf{h}^{(2)} \leftarrow \mathsf{SSGen}(\mathbf{y} \oplus \mathbf{e})$, where the perturbation $\mathbf{e}$ is known and fully determined by the attacker, is indeed provably equivalent. The latter helper data reveals that $\mathbf{y}$ belongs to an identical coset $\{\mathbf{h}^{(1)} \oplus \mathbf{w} : \mathbf{w} \in \mathcal{W}\} = \{\mathbf{h}^{(2)} \oplus \mathbf{e} \oplus \mathbf{w} : \mathbf{w} \in \mathcal{W}\}$. However, the perturbation $\mathbf{e}$ is determined by the inherent noisiness of a PUF rather than by the attacker and its release hence reveals new information.

Given a sequence of q exposures, the attacker can estimate all individual bit error rates $p_{\text{error},i}$ with $i \in [1, n]$, as well as the coset to which reference $\mathbf{y}$ belongs. For this purpose, the attacker collects helper data $\mathbf{h}^{(j)} \leftarrow \mathsf{SSGen}(\mathbf{y} \oplus \mathbf{e}^{(j)})$, with $j \in [1, q]$. The difference vector among each pair of noisy responses can be recovered as long as its Hamming weight does not exceed t; consider a non-redundant set $(\mathbf{e}^{(1)} \oplus \mathbf{e}^{(j)})$ with $j \in [2, q]$. For $q \to \infty$, the estimates in (4.28) converge to their exact counterpart.

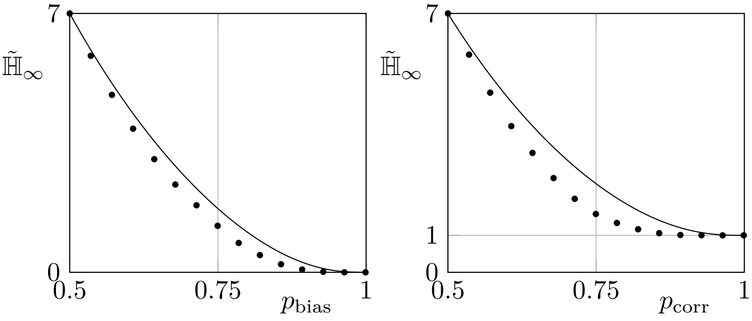
$$\lim_{q \to \infty} \left[\begin{cases} (0, p_i), & \text{if } p_i < 1/2, \\ (1, 1 - p_i), & \text{otherwise} \end{cases} \right] = (e_i^{(1)}, p_{\text{error},i}) \quad \text{with } p_i = \sum_{j=2}^{q} \frac{e_i^{(1)} \oplus e_i^{(j)}}{q - 1}. \tag{4.28}$$

For ease of understanding, assume $Y = X$. The exposure of a bit error rate $p_{\text{error},i} = \mathsf{F}_{\text{norm}}(-|\omega_i - \omega_{\text{thres}}|/\sigma_{\text{noise}})$ then implies knowledge of the threshold discrepancy $|\omega_i - \omega_{\text{thres}}|$. The residual min-entropy of reference response X is captured by (4.29).

$$\begin{aligned}&\tilde{\mathbb{H}}_\infty(X|(H, P_{\text{error},1}, \dots, P_{\text{error},n}))\\ &= -\log_2\left(\mathbb{E}_{v\leftarrow V}\left[\frac{\max_{\mathbf{w}\in\mathcal{W}}\mathbb{P}(\Omega = \boldsymbol{\omega}_{\mathbf{w}})}{\sum_{\mathbf{w}\in\mathcal{W}}\mathbb{P}(\Omega = \boldsymbol{\omega}_{\mathbf{w}})}\right]\right),\end{aligned} \tag{4.29}$$

with $\omega_{\mathbf{w},i} = \omega_{\text{thres}} + (1 - 2w_i)(\omega_i - \omega_{\text{thres}})$ and $i \in [1, n]$.

Figure 4.11 quantifies the residual min-entropy of X with the exclusion and inclusion of revealed bit error rates $p_{\text{error},i}$ respectively. In the latter case, we rely on a Monte Carlo evaluation of (4.29), as enabled by choosing a small $[n = 15, k = 7, d = 5]$ BCH code, given that an analytical approach is not straightforward. For both the biased and correlated distribution, it turns out that repeated helper data exposure results in additional min-entropy loss.



(a) Bias; $[n = 15, k = 7, d = 5]$. (a) Correlation; $[n = 15, k = 7, d = 5]$.

Figure 4.11: The additional min-entropy loss attributed to revealed bit error rates. Solid lines represent $\tilde{\mathbb{H}}_\infty(X|H)$, as computed with BoundWorstCase2; Figure 4.10 previously confirmed the visual overlap with the exact result. Dots correspond to $\tilde{\mathbb{H}}_\infty(X|(H, P_{\text{error},1}, \dots, P_{\text{error},n}))$, hereby relying on Monte Carlo evaluations of size 10^6, i.e., the number of samples $v \leftarrow V$.

The crucial insight for the biased distribution is that majority and minority bits tend to exhibit lower and higher error rates respectively. Note that $\mathbb{E}_{v\leftarrow V}[P_{\text{error},x}|X_i = 1] < \mathbb{E}_{v\leftarrow V}[P_{\text{error},x}|X_i = 0]$ if $p_{\text{bias}} > 1/2$ and vice versa otherwise. The attacker obtains the bit-specific bias $p_{\text{bias},i} = \mathbb{P}(X_i = 1|P_{\text{error},i} = p_{\text{error},i})$ in (4.30), which is more informative than $p_{\text{bias}} = \mathbb{P}(X_i = 1)$.

$$p_{\text{bias},i} = \frac{\mathsf{f}_{\text{norm}}(\omega_{\text{thres}} + |\omega_i - \omega_{\text{thres}}|)}{\mathsf{f}_{\text{norm}}(\omega_{\text{thres}} + |\omega_i - \omega_{\text{thres}}|) + \mathsf{f}_{\text{norm}}(\omega_{\text{thres}} - |\omega_i - \omega_{\text{thres}}|)}. \tag{4.30}$$

The crucial insight for the correlated distribution is that correlation among Ω_i and Ω_j, with $i, j \in [1, n]$, implies correlation among $P_{\text{error},i}$ and $P_{\text{error},j}$. We compute $\mathbb{P}(\Omega = \boldsymbol{\omega}_{\mathbf{w}}) = \mathsf{f}_{\text{norm}}(\boldsymbol{\omega}_{\mathbf{w}}, \mathbf{0}, \boldsymbol{\Sigma})$ in (4.29) with $\Sigma_{i,j} = \sin(\pi(c_{i,j} - 1/2))$ and $c_{i,j}$ defined in (4.16). The latter relation can be proven by integrating (4.31) in polar coordinates. The diagonal elements $\Sigma_{i,i} = 1$.

$$c_{i,j} = 2 \int_0^\infty \int_0^\infty \mathsf{f}_{\text{norm}}\left(\begin{pmatrix}\omega_i & \omega_j\end{pmatrix}; \begin{pmatrix}0 & 0\end{pmatrix}, \begin{pmatrix}1 & \Sigma_{i,j} \\ \Sigma_{i,j} & 1\end{pmatrix}\right) \mathrm{d}\omega_i \, \mathrm{d}\omega_j. \tag{4.31}$$

Observe in Figure 4.11 that the overly conservative $(n-k)$ bound compensates for the additional, unanticipated min-entropy loss. This observation, however, does not necessarily hold for every possible distribution. Moreover, the use of the $(n-k)$ bound in part undermines the lightweight intentions of a reverse fuzzy extractor [171, 110, 7]. Further theoretical work may determine to which extent and at which cost reverse fuzzy extractors can be repaired. A potential resolution already exists for biased distributions, as will be discussed in Section 4.5.5.

4.3.7 Helper Data Manipulation

Out of seven secure sketch constructions in Table 4.4, the three code-offset methods generate the most helper data $\mathbf{h}$, i.e., 2^n rather than 2^{n-k} bits, and are hence presumed to be the most vulnerable to manipulation attacks. We therefore focus on the code-offset case exclusively, and list four types of threats below:

First, consider the code-offset method of Tuyls et al. [166], instantiated with a linear code, and applied to a nearly-uniform input Y. The output $Z = M$ is then nearly-uniform as well, which allows the system provider to omit the subsequent hash function, i.e., a key $\mathbf{k} = \mathbf{z}$ is generated [65]. Under the assumption of an interactive application, as defined in Table 4.1, this unfortunately enables related-key attacks. XORing the helper data with a codeword $\mathbf{w}$ results in a related key $\hat{\mathbf{k}} = \mathbf{k} \oplus \mathbf{m}$.

Second, manipulations of the helper data allow an attacker to estimate the error rates $p_{\text{error},i}$ of individual response bits. As is showcased in Section 4.3.6, the release of this information can increase the min-entropy loss. For the code-offset methods, the attacker flips an arbitrary bit h_i of the helper data. Even under the assumption of a silent application, as defined in Table 4.1, an attacker can evaluate the changes in failure rate for the code-offset method of Tuyls et al. [166]. For the code-offset method of Dodis et al. [39], an interactive application would be required.

Third, given that a large response $\mathbf{y}$ is usually partitioned into smaller blocks that are processed independently, divide-and-conquer attacks might apply. We assess the possibilities for launching a brute-force search on each block independently. Linear codes, which are omnipresent in the literature, are found to be secure. Non-linear codes with three or more codewords $\mathbf{w}$, which have not been used in the literature so far, are not necessarily secure. Algorithm 5 can then retrieve the response bits of each partition. Regardless of whether $\mathbf{0} \in \mathcal{W}$, set $\mathcal{I}$ is expected to contain a single element in the end. Depending on the code, false positives might occur in the occasional case that the sum of three codewords is again a codeword. This can easily be resolved via an extension of the algorithm.

Algorithm 5: ATTACK ON THE CODE-OFFSET CONSTRUCTION OF DODIS ET AL. [39]

```
Input: Codewords $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_{|\mathcal{M}|}$ of the non-linear code $\zeta$
Input: Helper data $\mathbf{h}$
Output: Response section $\mathbf{y}$
$\mathcal{I} \leftarrow \{1, 2, \ldots, |\mathcal{M}|\}$
for $i \leftarrow 1$ to $|\mathcal{M}|$ do
    $j \leftarrow \mathsf{mod}(i, |\mathcal{M}|) + 1$
    Modify helper data: $\hat{\mathbf{h}} \leftarrow \mathbf{h} \oplus \mathbf{w}_i \oplus \mathbf{w}_j$
    if key reconstruction fails then
        $\mathcal{I} \leftarrow \mathcal{I} \backslash \{i, j\}$
$\mathbf{y} \leftarrow \mathbf{h} \oplus \mathbf{w}_{\mathcal{I}}$
```

Codes and decoding algorithms with some sort of non-uniformity for the error-correcting capability t might be in danger as well. Consider the following examples. First, the capability to correct 1 → 0 and 0 → 1 errors might differ. Second, the number of errors that can be corrected might differ per codeword and/or might depend on the position of the errors. In the former cases, an attacker can iterate again over all potential codewords and test hypotheses via the failure rate p_{fail}.

Fourth, the manipulation of helper data can facilitate physical attacks. Merli et al. [122] and Karakoyunlu and Sunar [77] both target the code-offset method, collect power traces for various helper vectors, and successfully recover the secret PUF response. Merli et al. [123] later proposed a countermeasure for linear codes where the sketch input is masked, i.e., XORed, with a randomly chosen codeword. This requires the implementation of a physically secure TRNG.

4.4 Other Error-Correction Schemes

We discuss several error-correction schemes that do not satisfy the definition of a secure sketch.

4.4.1 Exhaustive Search

A PUF-enabled device could perform an exhaustive search for the error pattern $\mathbf{e} = \mathbf{y} \oplus \hat{\mathbf{y}} \in \{0,1\}^{\lambda}$ [130]. An integrity check as given in Table 4.2 serves as a stopping criterion. This approach is only practical if the expected bit error rate $\mathbb{E}[P_{error,y}]$ is sufficiently low, e.g., due to the application of a prior bit selection scheme and/or TMV, and if λ is limited. The availability of individual bit error rates could accelerate the search. In fact, the workload of the most efficient search is quantified by the Shannon entropy $\mathbb{H}(E)$, as given in (4.32) under the assumption of i.i.d. response bits Y_i.

$$\begin{aligned}\mathbb{H}(E) = -\lambda\big(&\mathbb{E}_{v\leftarrow V}[P_{error,y}]\log_2(\mathbb{E}_{v\leftarrow V}[P_{error,y}])\\ +&(1-\mathbb{E}_{v\leftarrow V}[P_{error,y}])\log_2(1-\mathbb{E}_{v\leftarrow V}[P_{error,y}])\big).\end{aligned} \tag{4.32}$$

4.4.2 Soft-Decision Decoding

Maes et al. [112] were the first to apply *soft-decision decoding* to the recovery procedure SSRep of a secure sketch, and implemented a PUF-based key generator accordingly [114]. The main benefit is that the error-correcting capabilities of the secure sketch are improved with respect to traditional hard-decision decoding. A considerable drawback is that the error rates $p_{\mathrm{error},i}$ of all ingoing response bits y_i need to be published as helper data $\mathbf{h}_{\mathrm{error}}$, in addition to the helper data $\mathbf{h}$ that is inherent to the secure sketch itself. For the three code-offset methods in Table 4.4, a *soft-decision maximum-likelihood* (SDML) decoder recovers the randomly selected codeword $\mathbf{w}$ according to (4.33). Given that an exhaustive search over all $|\mathcal{M}|$ candidate codewords $\hat{\mathbf{w}}$ is involved, only small codes can be used. So as to obtain a key $\mathbf{k}$ of sufficient length, the response $\mathbf{y}$ should hence be subdivided into q partitions that are each processed independently.

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}\in\mathcal{W}} \prod_{i=1}^{n} (1-p_{\mathrm{error},i})^{h_i \oplus \tilde{y}_i \oplus w_i} (p_{\mathrm{error},i})^{h_i \oplus \tilde{y}_i \oplus w_i \oplus 1}. \tag{4.33}$$

For implementation purposes, it is usually more efficient to use the log-likelihood in (4.34) instead. A product is then converted into a sum. The logarithms can be computed in advance, e.g., by publishing $(\log(1 - p_{\text{error},i}) - \log(p_{\text{error},i}))$ as helper data $\mathbf{h}_{\text{error}}$ instead of $p_{\text{error},i}$.

$$\hat{\mathbf{w}} = \underset{\mathbf{w} \in \mathcal{W}}{\arg\max} \sum_{i=1}^{n} (-1)^{h_i \oplus \tilde{y}_i \oplus w_i} \big(\log(1 - p_{\text{error},i}) - \log(p_{\text{error},i})\big). \tag{4.34}$$

Nevertheless, the computational burden remains substantial. For convolutional codes, Viterbi's decoding algorithm [174] provides a more efficient alternative. For Reed–Muller codes, the *generalized multiple concatenated* (GMC) decoding algorithm improves the runtime, although it does not guarantee maximum-likelihood.

Min-Entropy Loss

As previously elaborated in Section 4.3.6, the exposure of bit error rates $p_{\text{error},i}$ can result in additional min-entropy loss. Maes et al. [112] have proven the seemingly more optimistic result in (4.35), valid for i.i.d. response bits Y_i with $\mathbb{P}(Y_i = 1) = p_{\text{bias}}$ and under the assumption of the heterogeneous variability–noise model in Section 3.1.5. It is also assumed that noise is suppressed during the enrollment, which complies with a joint effort for majority voting and the estimation of bit error rates.

$$\mathbb{H}_\infty(Y) = \tilde{\mathbb{H}}_\infty(Y | (P_{\text{error},1}, P_{\text{error},2}, \ldots, P_{\text{error},n})). \tag{4.35}$$

An outline of the proof is given in (4.36). For the bit-specific bias $p_{\text{bias},i} = \mathbb{P}((Y_i = 1) | (P_{\text{error},i} = p_{\text{error},i}))$ given in (4.30), it holds that $p_{\text{bias},i} \in (0.5, 1)$ if $p_{\text{bias}} \in (0.5, 1)$ and $p_{\text{bias},i} \in (0, 0.5)$ if $p_{\text{bias}} \in (0, 0.5)$. An attacker's best guess is hence still either $\mathbf{y} = \mathbf{0}$ or $\mathbf{y} = \mathbf{1}$.

$$\begin{aligned} &\tilde{\mathbb{H}}_\infty(Y | (P_{\text{error},1}, P_{\text{error},2}, \ldots, P_{\text{error},n})) \\ &= -n \log_2\Big(\mathbb{E}_{\omega \leftarrow \Omega}\big[\max(p_{\text{bias},i}(\omega), 1 - p_{\text{bias},i}(\omega))\big]\Big) \\ &= -n \log_2\Big(\max\big(\mathbb{E}_{\omega \leftarrow \Omega}\big[p_{\text{bias},i}(\omega)\big], 1 - \mathbb{E}_{\omega \leftarrow \Omega}\big[p_{\text{bias},i}(\omega)\big]\big)\Big) \\ &= -n \log_2(\max(p_{\text{bias}}, 1 - p_{\text{bias}})) = \mathbb{H}_\infty(Y). \end{aligned} \tag{4.36}$$

Unfortunately, the helper data $\mathbf{h}$ of the code-offset method has not been incorporated. Although the exposure of bit error rates $p_{\mathrm{error},i}$ does not induce min-entropy loss by itself, it can amplify the min-entropy loss that is attributed to $\mathbf{h}$, as suggested in (4.37). Section 4.3.6 quantified the additional losses for the biased distribution in particular.

$$\tilde{\mathbb{H}}_\infty(Y|H) \geq \tilde{\mathbb{H}}_\infty(Y|(H, P_{\mathrm{error},1}, P_{\mathrm{error},2}, \ldots, P_{\mathrm{error},n})). \tag{4.37}$$

Helper Data Manipulation

Maes et al. [114] mention that the integrity of helper data should be verified, but implement a key generator that lacks such protective measures. We observe that, depending on the specifics of the implementation, and even for *silent* applications as defined in Table 4.1, manipulation of the published error rates might result in a complete recovery of the secret key $\mathbf{k}$.

Consider an SDML decoder that is applied to q partitions of response $\mathbf{y}$ independently. A crucial insight is that the decisive impact of any response bit $\hat{y}_i$ on the likelihood computation in (4.33) can be nullified by setting the corresponding error rate $p_{\mathrm{error},i}$ to $1/2$. For ease of understanding, assume that the response bits are noiseless, i.e., $\hat{y}_i = y_i$. As illustrated in Figure 4.12 for a small-sized $[n, k, d]$ BCH code, an attacker who manipulates the helper data k times and observes the corresponding failure rates p_{fail} for key reconstruction, might be able to recover the secret codeword $\mathbf{w}$ of any given partition. By repeating the same mechanism for each partition, the secret key $\mathbf{k}$ is eventually recovered.

Also $[n, k = 1, d = n]$ repetition codes, which are usually adopted as part of a concatenated code $\zeta_2 \circ \zeta_1$, are vulnerable to manipulation attacks. Consider an attacker who sets the published error rates $\mathbf{p}_{\mathrm{error}}$ of any given partition to $\begin{pmatrix} 1/2 & 1/2 & \cdots & 1/2 \end{pmatrix}$. Codewords $\mathbf{w}_1$ and $\mathbf{w}_2$ are then equally likely, according to (4.33). A given implementation of an SDML decoder might, for example, deterministically return codeword $\mathbf{w}_1$ in case of such a tie. A single observation of the failure rate p_{fail} for key reconstruction hence suffices to recover the secret codeword $\mathbf{w}$ of the given partition.

Not a Secure Sketch

Although Maes et al. [112] build upon the code-offset method, their scheme as a whole should not be regarded as a secure sketch. First of all, the authors analyzed the min-entropy loss for i.i.d. response bits Y_i exclusively, while a

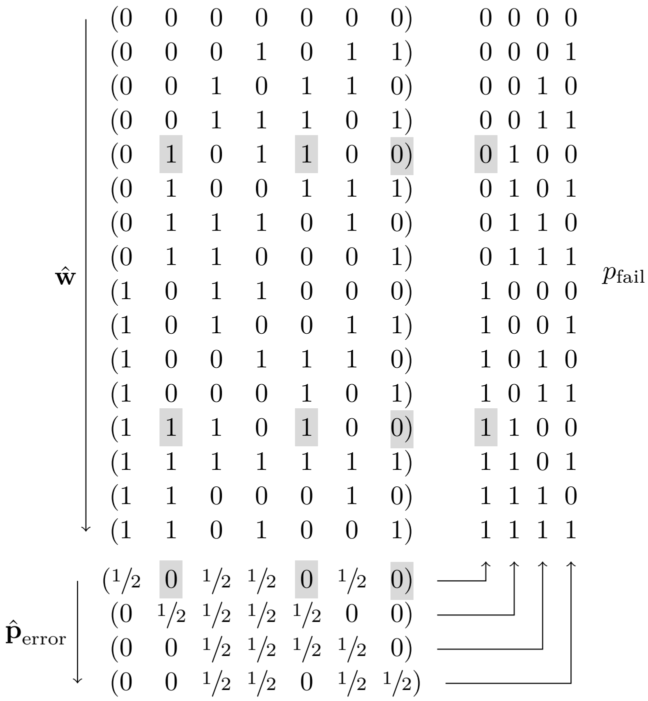


Figure 4.12: A helper data manipulation attack on the SDML decoding scheme of Maes et al. [112, 114]. The working principle is illustrated for an $[n = 7, k = 4, d = 3]$ BCH code, where the exhaustive search for the most likely codeword $\hat{\mathbf{w}}$ is performed in the given order, from top to bottom. In case of a maximum-likelihood tie, it is assumed that the first codeword $\hat{\mathbf{w}}$ that takes part in the tie is returned. By observing the failure rate p_{fail} for key reconstruction, given $k = 4$ well-chosen helper data patterns $\hat{\mathbf{h}}_{\text{error}}$, an attacker can uniquely identify the enrolled codeword $\mathbf{w}$.

secure sketch can handle all distributions of Y with a given lower bound on the min-entropy. Moreover, it is not straightforward to impose a meaningful lower bound t on the number of errors that can be corrected. In theory, the errors could all occur at indices i where $p_{\text{error},i}$ is relatively low, and hence mislead the soft-decision decoder.

Computing Bit Error Rates On-The-Fly

Van der Leest et al. [103] later proposed an alternative scheme where the bit error rates $p_{\text{error},i}$ are estimated on-the-fly during the reconstruction phase rather than stored as part of helper data. This variation relies again on the code-offset method, instantiated with a concatenated code $\zeta_2 \circ \zeta_1$ that processes q non-overlapping blocks of response bits independently. In addition to a hard-decision output, the decoder of the repetition code ζ_2 provides a level of confidence that can be used by the subsequent soft-decision decoder of ζ_1. The $(n-k)$ upper bound on the min-entropy loss still applies; the previously discussed manipulation attack is precluded.

4.4.3 Pattern Matching

Paral and Devadas [132, 133] were the first to adopt *pattern matching* as an error-correction technique for PUFs, followed by Majzoobi et al. [119, 140]. During the enrollment, a substring $\mathbf{w} \in \{0,1\}^{\eta}$ of the response $\mathbf{y} \in \{0,1\}^{\lambda}$, the former of which is referred to as a pattern, is selected uniformly at random and published as helper data $\mathbf{h}$. The corresponding index $z \in [1,q]$ serves as keying material. The following three variations for the selection of a substring have been proposed. First, the response $\mathbf{y}$ can be partitioned into non-overlapping substrings, i.e., $\lambda = q\,\eta$. Second, the substrings of $\mathbf{y}$ can be allowed to overlap, i.e., $\lambda = q + \eta - 1$. Third, the response $\mathbf{y}$ can be considered as cyclic while having overlapping substrings, i.e., $\lambda = q$.

The reconstruction involves shifting the helper data pattern $\mathbf{h}$ along a regenerated response $\hat{\mathbf{y}}$ so that the Hamming distance at each index $\hat{z} \in [1,q]$ can be computed. The following two variations for the retrieval of the enrolled index z have been proposed. First, the condition $\mathsf{HD}(\mathbf{h}, \hat{\mathbf{h}}) \leq t$, where threshold t is a constant, is expected to be satisfied at the index $\hat{z} = z$ while at all other indices $\hat{z} \neq z$ it is not. We assume that the reconstruction fails if zero indices satisfy this condition. We also assume that either the first or the last matching index is deterministically returned in case of a tie. A second, non-parametric recovery procedure outputs the index $\hat{z}$ that minimizes the Hamming distance instead. Again, we assume that a tie is resolved via the deterministic selection of either the first or the last matching index.

In order to obtain a key $\mathbf{k}$ of sufficient length, the previously described enrollment and reproduction steps are performed on g non-overlapping responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_g$ independently. Binary representations of indices $z_1, z_2, \ldots, z_g$ are concatenated to obtain the outgoing secret $\mathbf{z}$.

So far, the proposal complies with both weak and strong PUFs. For the latter type of PUF in particular, machine learning attacks are counteracted by the secret decoupling of challenges and responses. As an additional protective measure, the secret index z of each round can *fork* the next round of the challenge generator [132]. Stated otherwise, the CRP stream of each round depends on the secret indices z of all previous rounds. As a consequence, the link between challenges and responses becomes less and less traceable with each round. As an alternative additional countermeasure [140], each published pattern $\mathbf{h}$ can be padded with random bits.

Min-Entropy Loss

Regardless of all the variations and under the assumption of i.i.d. response bits Y_i with bias p_{bias}, the concatenation $\mathbf{z}$ of substring indices is uniformly distributed even for an attacker who observes the helper data $\mathbf{w}$. However, the proof-of-concept implementation relies on an Arbiter XOR PUF and correlations among the response bits Y_i are hence to be expected.

Failure Rate

We analyze the expected failure rate under the assumption of a uniformly distributed response Y and non-overlapping patterns $\mathbf{h}$. An exact expression for the first recovery procedure, which has a parameter t, is given in (4.38). Counter i represents the number of indices $\hat{z} \neq z$ that satisfies the Hamming distance condition in addition to $\hat{z} = z$.

$$\mathbb{E}[P_{\text{fail}}] = 1 - \left(\mathsf{F}_{\text{bino}}\big(t, \eta, \mathbb{E}[P_{\text{error},y}]\big) \sum_{i=0}^{q-1} \frac{\mathsf{f}_{\text{bino}}\big(i, q-1, \mathsf{F}_{\text{bino}}(t, \eta, 0.5)\big)}{i+1} \right)^{g}. \tag{4.38}$$

The expected failure rate for the second, non-parametric recovery procedure is given in (4.39). Counter i represents the number of indices $\hat{z} \neq z$ that joins index $\hat{z} = z$ in a minimum Hamming distance tie.

$$\mathbb{E}[P_{\text{fail}}] = 1 - \left(\sum_{t=0}^{\eta} \mathsf{f}_{\text{bino}}\big(t, \eta, \mathbb{E}[P_{\text{error},y}]\big) \right.$$
$$\left. \sum_{i=0}^{q-1} \frac{\binom{q-1}{i} \big(\mathsf{f}_{\text{bino}}(t, \eta, 0.5)\big)^{i} \big(1 - \mathsf{F}_{\text{bino}}(t, \eta, 0.5)\big)^{q-1-i}}{i+1} \right)^{g}. \tag{4.39}$$

As is clear from Figure 4.13, the second recovery procedure performs better than the first one. This ratifies the intuitive insight that the first recovery procedure does not properly distinguish among indices $\hat{z}$ that exhibit a low Hamming distance.

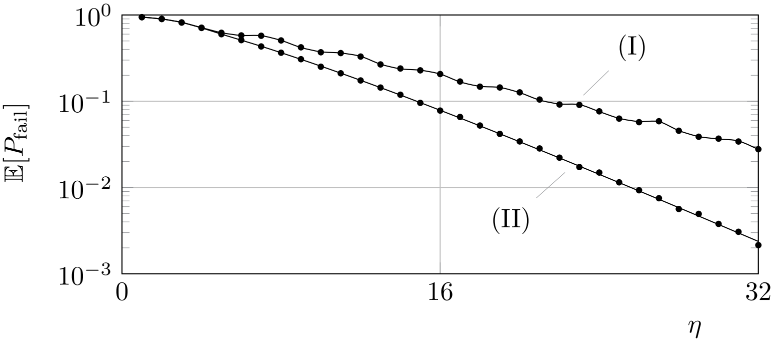


Figure 4.13: The expected failure rate $\mathbb{E}[P_{\text{fail}}]$ for pattern matching with $q = 32$ non-overlapping patterns of length $\eta \in [1, 32]$. The averaged bit error rate $\mathbb{E}[P_{\text{error},y}] = 0.1$ and only $g = 1$ round is considered. Curve (I) corresponds the first recovery procedure as given in (4.38), with parameter t chosen so as to minimize the expected failure rate. Curve (II) correspond to the second, non-parametric recovery procedure as given in (4.39). Each dot corresponds to a Monte Carlo experiment of size 10^5 that verifies the correctness of the analytic results.

Helper Data Manipulation

Depending on the chosen variation, pattern matching might be vulnerable to helper data manipulation attacks. Consider for example basic pattern matching with overlapping substrings where the key is used by an interactive application as defined in Table 4.1. As illustrated in Figure 4.14, an attacker can then iteratively shift the given pattern **h** along response **y**, and determine the value of one previously unrevealed bit with every move.

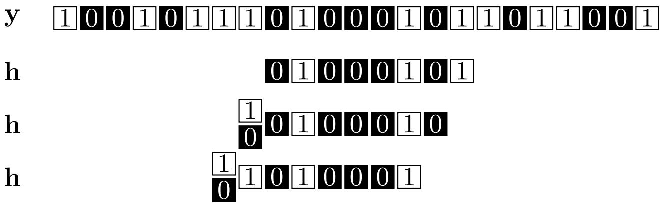


Figure 4.14: A first helper data manipulation attack on a basic pattern matching key generator with overlapping substrings. Only a single round is depicted.

Regardless of the chosen recovery procedure, the key at any given index is more and less stable with a correctly and incorrectly guessed helper bit respectively. With every move, the attacker hence needs to measure the failure rate for key reconstruction twice. A large number of measurements might be required in order to statistically distinguish both failure rates with sufficient confidence. In order to accelerate this test, the attacker can temporarily flip a fraction of the previously revealed bits in both patterns, i.e., both failures rates can be increased to a more favorable range.

If the response **y** is non-cyclic, then the attacker observes a sudden increase of both failure rates when the first (or last) index is exceeded. Therefore, the index of the original pattern is revealed, and by repeating the same mechanism for all rounds, the original secret key can hence be recovered. This is not the case for a cyclic response **y**, although the attacker still retrieves previously unrevealed response bits, which might in turn facilitate an alternative attack vector.

Even under the assumption of a silent application as defined in Table 4.1, some variations of pattern matching are prone to key-recovery through helper data manipulation. We refer to our CT-RSA 2014 publication for more details.

4.5 Computationally Straightforward Front-End

The front-end of our generic HDA in Figure 4.2 consists of three consecutive steps that require straightforward computations only, i.e., subset selection, TMV, and SMV. These steps are usually instantiated in a coordinated manner, and we therefore discuss them in a joint effort. The primary objective of the proposals in Table 4.7 is a reduction in error rate, i.e., $\mathbb{E}[P_{\text{error},y}] < \mathbb{E}[P_{\text{error},x}]$, and/or a reduction in bias, i.e., $|\mathbb{P}(Y_i = 1) - {}^1\!/_2| < |\mathbb{P}(X_i = 1) - {}^1\!/_2|$. Often, a reduction in either one is at the expense of the other. Moreover, response bits are usually lost in the process, and the size of the helper data $\mathbf{h}$ can be considerable.

Table 4.7: Proposed methods for the computationally straightforward front-end of a PUF-based key generator.

	Consecutive steps			Properties		
Proposal	Subset	TMV	SMV	Bias	p_{error}	Choose $\mathbf{y}$
Bolotnyy et al. [21]		✓			↓	×
Škorić et al. [157]	✓			↑	↓	×
Suh and Devadas [161]	✓			↑	↓	×
Maes et al. [113]			✓	↓	↑	×
van der Leest et al. [103]			✓	↑	↑↓	×
Koeberl et al. [88]	✓		✓	↑	↓	×
von Neumann [111]	✓			↓	↑	×
IBS [187, 62]	✓		✓	↓	↑↓	✓

A byproduct of TMV/SMV comprises the estimates of individual bit error rates p_{error}. The enables the enrollment of several forms of subset selection, as well as the use of the previously discussed soft-decision scheme of Maes et al. [112], where error rates are stored as helper data $\mathbf{h}$. Alternatively, the error rates required for soft-decision decoding can be estimated on-the-fly during the reconstruction phase, as proposed by van der Leest et al. [103]. Finally, an exhaustive search for the error pattern, as discussed in Section 4.4.1, can be accelerated if the bit error rates are known.

4.5.1 Temporal Majority Vote

TMV, as previously discussed for the enrollment in Sections 4.1.3 and 4.1.4, can be performed during the reconstruction phase as well [37, 21, 113, 183]. However, then it is not a one-time effort anymore, and additional circuitry is indispensable, i.e., counters that could previously be implemented externally now have to be implemented on the PUF-enabled device always.

Bit Error Rates

Under the assumption of i.i.d. response bits X_i, the expected post-TMV error rate $\mathbb{E}[P_{\text{error},y}]$ is given in (4.40), where $p^{\infty}_{\text{error}}(\omega)$ is as defined in (4.2). We assume that an equal number of votes q is used during both the enrollment and the reconstruction. Errors on relatively stable response bits are successfully suppressed, but for the relatively unstable bits, high error rates unfortunately remain high.

$$\begin{split}\mathbb{E}_{v \leftarrow V}\left[P_{\text{error},y}\right] = 2\int_{\omega=-\infty}^{\infty} \mathsf{f}_{\text{norm}}(\omega)\, \mathsf{F}_{\text{bino}}\Big(\frac{q-1}{2}; q, p^{\infty}_{\text{error}}(\omega)\Big) \\ \bigg(1 - \mathsf{F}_{\text{bino}}\Big(\frac{q-1}{2}; q, p^{\infty}_{\text{error}}(\omega)\Big)\bigg)\, \mathrm{d}\omega, \quad \text{with } q \text{ odd.}\end{split} \tag{4.40}$$

As is clear from the heterogeneous variability–noise model in Section 3.1.5, TMV is never sufficient by itself if the IC's environment in the field differs from during the enrollment, i.e., $\mu_{\text{noise}} \neq 0$. Subsequent and/or preceding error-correction steps have to be implemented as well.

Neither Min-Entropy Loss nor Helper Data Manipulation

The absence of helper data $\mathbf{h}$ is a considerable benefit. Most notably, attacks that rely on the manipulation of helper data do not apply. Moreover, given that we consider the application of TMV during the reconstruction phase, the enrolled secret $\mathbf{x}$ remains unaffected, and there is hence no min-entropy loss. This differs from the application of TMV during the enrollment: under the assumption of i.i.d. response bits X_i, it was shown that bias is slightly amplified according to (4.4).

4.5.2 Suh and Devadas

The scheme of Škorić et al. [157], which has later been adopted by several others [161, 4, 67, 66, 19], outputs a string $\mathbf{y} \in \{0, 1\}^{\eta}$ where the most and least reliable bits of response $\mathbf{x} \in \{0, 1\}^{\lambda}$ have been retained and discarded respectively. The fraction of retained bits, i.e., $p_{\text{ret}} = \eta/\lambda$, is a parameter. The smaller the retention ratio, the smaller the expected bit error rate $\mathbb{E}[P_{\text{error},y}]$, but the more response bits x that are lost in the process. Suh and Devadas [161] later generalized the scheme such that response $\mathbf{x}$ is first subdivided into partitions of size λ, and subsequently, the η most stable bits of each partition are retained. Figure 4.15 provides an illustration.

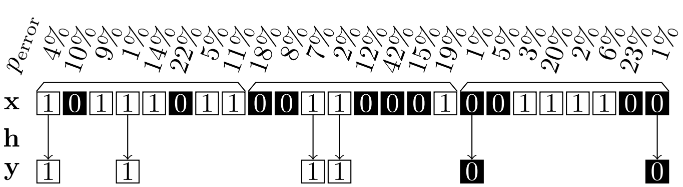


Figure 4.15: The η-out-of-λ selection scheme of Suh and Devadas [161], with $\eta = 2$ and $\lambda = 8$. Only the enrollment is depicted.

Several formats for the storage of helper data $\mathbf{h}$ have been proposed, given that there is a trade-off between its size and its in-the-field interpretation effort. Bhargava and Mai [19] assign a dedicated helper bit h to each response bit x, i.e., helper data $\mathbf{h}$ serves as a mask. This format is size-efficient if the retention ratio $p_{\text{ret}} \approx 1/2$. Alternatively, for $p_{\text{ret}} < 1/2$ and $p_{\text{ret}} > 1/2$, it could be more efficient to store the indices of the retained and discarded bits respectively, especially if λ is small. Hiller et al. [66] suggest storing the distances between consecutive retained/discarded response bits x instead. Representing these distances with a fixed number of bits is not necessarily efficient though. The same authors therefore proposed the use of a *run-length encoding* scheme.

As pointed out by Armknecht et al. [4], there is a good symbiosis between the η-out-of-λ selection scheme and TMV. First of all, estimations of individual bit error rates $p_{\text{error},x}$, which enable the enrollment of the selection scheme, happen to be a natural byproduct of TMV. Moreover, the response bits x for which TMV is effective and ineffective respectively are retained and discarded respectively.

Bit Error Rate

Even under the assumption of i.i.d. response bits X_i, it is not straightforward to evaluate the expected error rate $\mathbb{E}[P_{\text{error},y}]$ in an exact, analytical manner. The joint PDF $\mathsf{f}\big(\omega_{(1)}, \omega_{(2)}, \cdots, \omega_{(\lambda)}\big) = \lambda!\,\mathsf{f}\big(\omega_{(1)}\big)\mathsf{f}\big(\omega_{(2)}\big)\cdots\mathsf{f}\big(\omega_{(\lambda)}\big)$ of order statistics $\omega_{(1)} < \omega_{(2)} < \ldots < \omega_{(\lambda)}$ [2] results in nested integrals that do not simplify well. Nevertheless, relatively simple formulas arise for two extreme cases that together bound the behavior of all other instances. A first, asymptotic case $\lambda \to \infty$ is elaborated in (4.41), where $p^{\infty}_{\text{error},x}(\omega)$ is as defined in (4.2). For convenience, the retention ratio p_{ret} is mapped to an offset $\delta \in [0, \infty)$ from the threshold value ω_{thres}.

$$\begin{cases} \lim\limits_{\lambda\to\infty}\big[p_{\text{ret}}\big] = 1 - \mathsf{F}_{\text{norm}}(\omega_{\text{thres}} + \delta) + \mathsf{F}_{\text{norm}}(\omega_{\text{thres}} - \delta), \\ \lim\limits_{\lambda\to\infty}\Big[\mathbb{E}[P_{\text{error},y}]\Big] = \lim\limits_{\lambda\to\infty}\big[p_{\text{ret}}\big]\bigg(\displaystyle\int_{-\infty}^{\omega_{\text{thres}}-\delta} \mathsf{f}_{\text{norm}}(\omega)p^{\infty}_{\text{error},x}(\omega)\,\mathrm{d}\omega \\ \qquad + \displaystyle\int_{\omega_{\text{thres}}+\delta}^{\infty} \mathsf{f}_{\text{norm}}(\omega)p^{\infty}_{\text{error},x}(\omega)\,\mathrm{d}\omega\bigg). \end{cases} \tag{4.41}$$

A second extreme case is the selection of only $\eta = 1$ bit per partition. Equation (4.42) is valid for the non-degenerate case $\lambda \geq 2$.

$$\begin{aligned} &\mathbb{E}[P_{\text{error},y}] = \\ &\int_{-\infty}^{\omega_{\text{thres}}} \int_{\omega_{(1)}}^{2\omega_{\text{thres}}-\omega_{(1)}} \mathsf{f}\big(\omega_{(1)}, \omega_{(\lambda)}\big)p^{\infty}_{\text{error},x}\big(\omega_{(1)}\big)\,\mathrm{d}\omega_{(\lambda)}\,\mathrm{d}\omega_{(1)} \\ &+ \int_{\omega_{\text{thres}}}^{\infty} \int_{2\omega_{\text{thres}}-\omega_{(\omega)}}^{\omega_{(\lambda)}} \mathsf{f}\big(\omega_{(1)}, \omega_{(\lambda)}\big)p^{\infty}_{\text{error},x}\big(\omega_{(\lambda)}\big)\,\mathrm{d}\omega_{(1)}\,\mathrm{d}\omega_{(\lambda)}, \\ &\text{with } \mathsf{f}\big(\omega_{(1)}, \omega_{(\lambda)}\big) = \lambda(\lambda-1)\mathsf{f}_{\text{norm}}\big(\omega_{(1)}\big)\mathsf{f}_{\text{norm}}\big(\omega_{(\lambda)}\big) \\ &\Big(\mathsf{F}_{\text{norm}}\big(\omega_{(\lambda)}\big) - \mathsf{F}_{\text{norm}}\big(\omega_{(1)}\big)\Big)^{\lambda-2} \text{if } \ \omega_{(1)} < \omega_{(\lambda)}. \end{aligned} \tag{4.42}$$

Figure 4.16 shows numerical results for (4.41) and (4.42). All possible instances of η-out-of-λ selection reside in the banana-shaped region that is bounded by either curve. Subdividing response $\mathbf{x}$ into partitions is clearly not Pareto-optimal.

An intuitive explanation thereof is that the bits of response $\mathbf{x}$ with the lowest error rates p_{error} are not necessarily equally spread over the various partitions. Nevertheless, partitioning might allow for smaller helper data $\mathbf{h}$, so the analysis of this approach remains relevant. The expected bit error rate $\mathbb{E}[P_{\text{error},y}]$ can be made arbitrarily low by discarding more and more response bits x. This way, Bhargava and Mai [19] were able to manufacture a reliable key generator on a 65 nm CMOS ASIC, without relying on error-corrections methods like for instance a secure sketch.

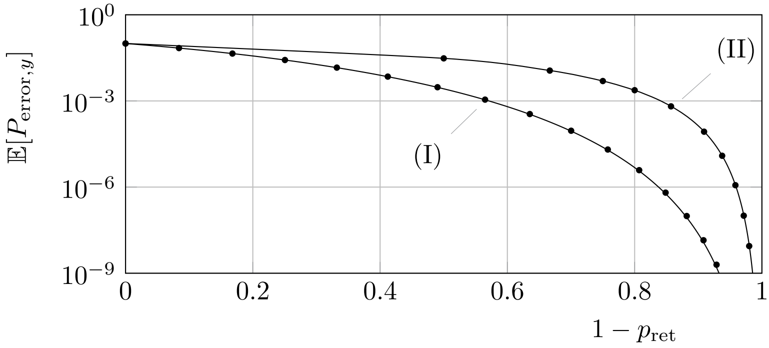


Figure 4.16: Trade-off between the retention ratio p_{ret} and the expected bit error rate $\mathbb{E}[P_{\text{error},y}]$ for the η-out-of-λ selection scheme of Suh and Devadas [161] with (I) $\lambda \to \infty$, and (II) $\eta = 1$. The response X is assumed to be uniformly distributed. The Gaussian noise is chosen to have a standard deviation $\sigma_{\text{noise}} = 0.325$ so that the ingoing bit error rate $\mathbb{E}[P_{\text{error},x}] \approx 10\%$ after a perfect majority vote, i.e., $q \to \infty$. Solid lines correspond to the analytic formulas, which are validated by Monte Carlo experiments that produce nearly coinciding dots. For 10^6 randomly generated responses $\mathbf{x}$ of size λ and with variability components $\boldsymbol{\omega}$ drawn from $\Omega \sim N(\mathbf{0}, \mathbf{I}_\lambda)$, the error rates of the retained bits are averaged. We approximate the asymptotic curve (I) by choosing $\lambda = 2048$.

Min-Entropy Loss

In general, $\mathbb{H}_\infty(X)/\lambda \neq \tilde{\mathbb{H}}_\infty(Y|H)/\eta$, i.e., the ratio of the min-entropy to the number of bits is not preserved. There is for instance an amplification of bias. Again under the assumption of i.i.d. response bits X_i, the outgoing bias $\mathbb{P}(Y_i = 1) = p_{\text{bias},y}$ is given in (4.43) and (4.44) for the extreme cases $\lambda \to \infty$ and $\eta = 1$ respectively. Figure 4.17 quantifies the corresponding min-entropy loss for retention ratios $p_{\text{ret}} \in \{^1/_2, ^1/_8\}$.

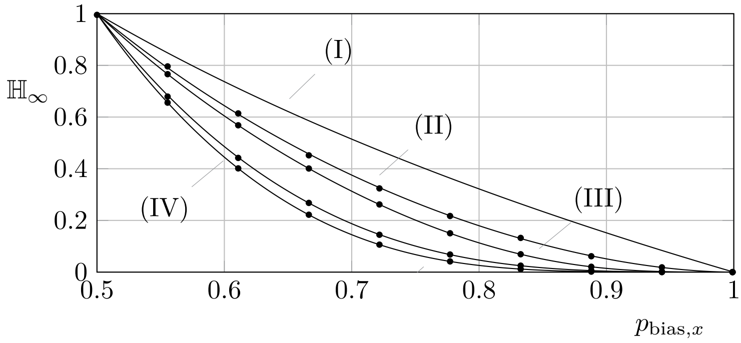


Figure 4.17: The min-entropy loss for the η-out-of-λ selection scheme of Suh and Devadas [161], as applied to i.i.d. response bits X_i with $\mathbb{P}(X_i = 1) = p_{\text{bias},x}$. The (I) ingoing min-entropy $\mathbb{H}_\infty(X_i) = -\log_2(p_{\text{bias},x})$ is considerably higher than the outgoing min-entropy $\mathbb{H}_\infty(Y_i) = -\log_2(p_{\text{bias},y})$ for parameter values (II) $\eta = 1$ and $\lambda = 2$, (III) $p_{\text{ret}} = 1/2$ and $\lambda \to \infty$, (IV) $\eta = 1$ and $\lambda = 8$, and (V) $p_{\text{ret}} = 1/8$ and $\lambda \to \infty$. Each dot corresponds to a Monte Carlo experiment where the estimate $\hat{p}_{\text{bias},y}$ for 10^5 randomly generated responses $\mathbf{x}$ of size λ is averaged. We approximate curves (III) and (V) of the asymptotic case $\lambda \to \infty$ by choosing $\lambda = 2048$.

$$\lim_{\lambda \to \infty} \left[p_{\text{bias},y}\right] = \frac{1 - \mathsf{F}_{\text{norm}}(\omega_{\text{thres}} + \delta)}{1 - \mathsf{F}_{\text{norm}}(\omega_{\text{thres}} + \delta) + \mathsf{F}_{\text{norm}}(\omega_{\text{thres}} - \delta)}. \tag{4.43}$$

$$p_{\text{bias},y} = \int_{\omega_{\text{thres}}}^{\infty} \int_{2\omega_{\text{thres}} - \omega_{(\lambda)}}^{\omega_{(\lambda)}} \mathsf{f}\big(\omega_{(1)}, \omega_{(\lambda)}\big) \, \mathrm{d}\omega_{(1)} \, \mathrm{d}\omega_{(\lambda)}. \tag{4.44}$$

Helper Data Manipulation

As pointed out by Hiller et al. [64], the η-out-of-λ selection scheme is highly vulnerable to helper data manipulation. Even for *silent* applications as defined in Table 4.1, the unpartitioned case can result in a quasi-complete recovery of the secret key $\mathbf{k}$. Assume that each retained bit y of response $\mathbf{x}$ has discarded neighbors to both its left and right. By shifting the index of such a retained bit y to an otherwise discarded neighbor, while observing the difference in the failure rate p_{fail} for key reconstruction, it is revealed whether or not both bits are equal. Eventually, the response $\mathbf{x}$ is completely revealed, except for one degree of freedom that corresponds to a bit-wise inversion of $\mathbf{x}$. Also for the partitioned case with $\eta \geq 2$, the attack is fairly effective, given that at most 1 bit of min-entropy remains within each partition.

For the partitioned case with $\eta = 1$, the previously described attack is less effective. Nevertheless, exposure of all equalities and inequalities within each individual partition can still result in additional min-entropy loss. For example, when the response bits X_i are biased, an attacker can determine whether a retained bit y is part of the either the presumed minority or the presumed majority.

4.5.3 Spatial Majority Vote

Koeberl et al. [88] proposed three schemes for the execution of a *spatial majority vote* (SMV), referred to with Roman numerals I, II, and III in Figure 4.18. The SMV I and SMV II schemes require helper data, while the SMV III scheme does not. As a side note, Maes et al. [113] previously proposed an SMV-like debiasing scheme that is not further discussed in this thesis.

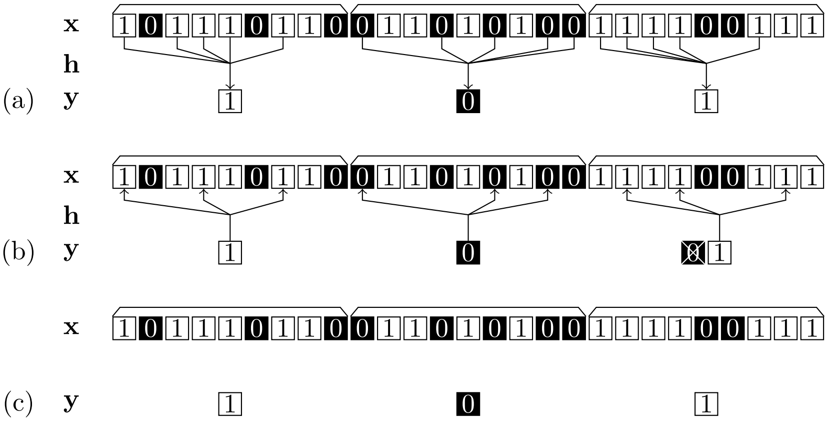


Figure 4.18: The schemes of Koeberl et al. [88], where SMV is applied to subsets of η-out-of-λ response bits. The enrollment is depicted for three variations of the scheme that are instantiated with partitions of size $\lambda = 9$: (a) the SMV I scheme, which allows for the correction of at most $t = 2$ errors among $\eta = 5$ reproduced bits, (b) the SMV II scheme with $\eta = 3$, which allows for the correction of at most $t = 1$ error among η reproduced bits, and (c) the SMV III scheme, which allows for the correction of a variable number of errors $t \in [0, \lambda - 1]$ among λ reproduced bits.

The SMV I scheme retains the first $\eta = (\lambda+1)/2$ majority bits within partitions of size λ exclusively, where $\lambda \in \{5, 9, 13, \ldots\}$. Helper data indicates whether a given response bit x is either retained or discarded. Up to $t = (\eta - 1)/2$ errors can be corrected among each selection of η reproduced bits. The SMV II scheme initially draws each outgoing bit y randomly and uniformly from $\{0, 1\}$. We then randomly select $\eta \in \{3, 5, 7, \ldots\}$ out of $\lambda \in \{2\eta, 2\eta{+}1, 2\eta{+}2, \ldots\}$ response bits x that are equal to y. If an insufficient number of bits is available, the complement of y is enrolled and reproduced instead. Again, up to $t = (\eta - 1)/2$ errors can be corrected. The SMV III scheme outputs the most occurring outcome among $\lambda \in \{3, 5, 7, \ldots\}$ response bits during both the enrollment and reconstruction phase. The number of errors t that can be corrected varies.

Min-entropy Loss

Under the assumption of i.i.d. but potentially biased response bits X_i, Koeberl et al. [88] derived the residual min-entropy of an outgoing bit Y_i as is given in (4.45). All three schemes amplify bias, as represented more clearly in Figure 4.19. The SMV II scheme suffers less from this issue in comparison with the SMV I and SMV III schemes.

$$\tilde{\mathbb{H}}_\infty(Y_i|H) = -\log_2(\max(p_{\text{bias},y}, 1 - p_{\text{bias},y})), \quad \text{with } p_{\text{bias},y} =$$
$$\begin{cases} 1 - \mathsf{F}_{\text{bino}}\Big(\dfrac{\lambda - 1}{2}; \lambda, p_{\text{bias},x}\Big) & \text{for SMV I, III} \\ 1 - \dfrac{\mathsf{F}_{\text{bino}}(\eta - 1; \lambda, p_{\text{bias},x}) + \mathsf{F}_{\text{bino}}(\lambda - \eta; \lambda, p_{\text{bias},x})}{2} & \text{for SMV II.} \end{cases} \tag{4.45}$$

The SMV I scheme is specified to retain the first η majority bits [88], but this is presumed to be an oversight. Clearly, the η retained indices should be selected uniformly at random among the majority bits, similar to the SMV II scheme. Otherwise, the attacker gains an additional advantage for biased PUFs in distinguishing whether y is either 0 or 1. E.g., for $p_{\text{bias}} > 1/2$, the indices of the retained bits would tend to be evenly distributed and clustered towards the start for $y = 0$ and $y = 1$ respectively.

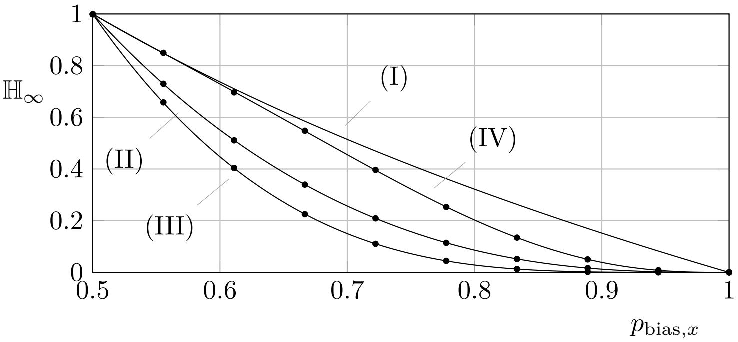


Figure 4.19: The SMV schemes of Koeberl et al. [88] amplify bias, as is quantified for i.i.d. response bits X_i. (I) The ingoing min-entropy $\mathbb{H}_\infty(X_i) = -\log_2(\max(p_{\text{bias},x}, 1 - p_{\text{bias},x}))$. The residual min-entropy $\tilde{\mathbb{H}}_\infty(Y_i|H)$ for (II) the SMV I and SMV III schemes with $\lambda = 5$, (III) the SMV I and SMV III schemes with $\lambda = 9$, and (IV) the SMV II scheme with $\lambda = 9$ and $\eta = 3$. Each dot corresponds to a Monte Carlo evaluation of size 10^6, so as to verify the correctness of (4.45).

Bit Error Rate

Koeberl et al. [88] do not take into account that the conditional bit error rates $\mathbb{E}[P_{\text{error},x=0}]$ and $\mathbb{E}[P_{\text{error},x=1}]$ are not necessarily equal, as is the case for a biased PUF. A more accurate reliability analysis under the assumption of i.i.d. response bits relies on (4.46).

$$\mathbb{E}[P_{\text{error},y}] = p_{\text{bias},y}\mathbb{E}[P_{\text{error},y=1}] + (1 - p_{\text{bias},y})\mathbb{E}[P_{\text{error},y=0}]. \tag{4.46}$$

The expression for $\mathbb{E}[P_{\text{error},y=1}]$ is given in (4.47). The expression for $\mathbb{E}[P_{\text{error},y=0}]$ is similar and hence not explicitly listed here.

$$\mathbb{E}[P_{\text{error},y=1}] = \begin{cases} 1 - \mathsf{F}_{\text{bino}}\Big(\dfrac{\lambda-1}{4}; \dfrac{\lambda+1}{2}, \mathbb{E}[P_{\text{error},x=1}]\Big) & \text{for SMV I,} \\[2ex] 1 - \mathsf{F}_{\text{bino}}\Big(\dfrac{\eta-1}{2}; \eta, \mathbb{E}[P_{\text{error},x=1}]\Big) & \text{for SMV II,} \\[2ex] \displaystyle\sum_{i=(\lambda+1)/2}^{\lambda} \mathsf{f}_{\text{bino}}(i; \lambda, p_{\text{bias},x}) & \\[2ex] \displaystyle\sum_{j=0}^{\lambda-i} \mathsf{f}_{\text{bino}}(j, \lambda-i, \mathbb{E}[P_{\text{error},x=0}]) & \\[2ex] \mathsf{F}_{\text{bino}}(i-j, i, 1-\mathbb{E}[P_{\text{error},x=1}]) & \text{for SMV III.} \end{cases} \tag{4.47}$$

Helper Data Manipulation

Although Koeberl et al. [88] mention that helper data can possibly be stored off-chip, its manipulation by an attacker has not been considered. For the SMV I and SMV II schemes, there is a similar exploit as for the 1-out-of-λ selection scheme of Suh and Devadas [161] in Section 4.5.2.

4.5.4 Index-Based Syndrome

The *index-based syndrome* (IBS) scheme of Yu and Devadas [187], where the term *syndrome* is a synonym for helper data, serves a debiasing purpose. To be precise, i.i.d. but potentially biased response bits X_i are transformed into a pre-chosen, uniformly distributed secret Y. We further consider the generalized IBS scheme of Hiller et al. [62], which has two parameters instead of only one. As is shown in Figure 4.20, the response $\mathbf{x}$ is first subdivided into partitions of size λ, and subsequently, η somewhat stable bits are retained for each partition. The main difference with the scheme of Suh and Devadas [161] in Section 4.5.2 is that the retained bits x should reconstruct a pre-chosen secret $\mathbf{y}$.

Although the secret Y could be uniformly distributed, a joint optimization with the subsequent secure sketch limits its set of outcomes $\mathcal{Y}$ to the codewords $\mathbf{w} \in \mathcal{W}_1$ of a block code ζ_1. In conjunction with the codewords $\mathbf{w} \in \mathcal{W}_2$ of the IBS scheme, a concatenated code $\zeta_2 \circ \zeta_1$ appears. Compared to the sketch constructions in Table 4.4, the enrollment reduces to the selection of a random

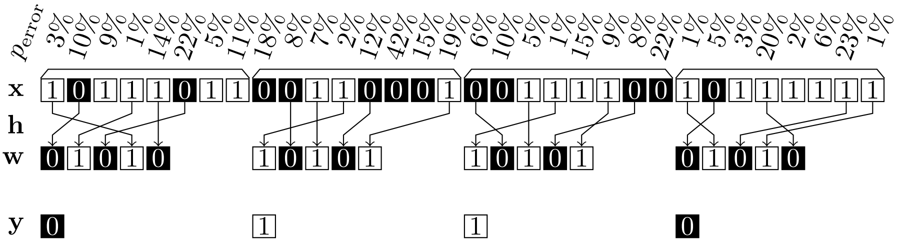


Figure 4.20: The generalized IBS scheme of Hiller et al. [62], where $\eta = 5$ response bits x are selected for each partition of size $\lambda = 8$. During the enrollment, a secret bit Y_i is chosen uniformly at random for each partition and encoded to either one out of two codewords $\mathbf{w}$ that have approximately the same Hamming weight, e.g., $\mathbf{w} \in \{(0\,1\,0\,1\,0), (1\,0\,1\,0\,1)\}$. Helper data $\mathbf{h}$ locally rearranges part of the response bits x such that codewords $\mathbf{w}$ can be reconstructed in the most reliable manner possible. If all nominal ones within a certain partition are depleted, the least reliable zeros serve as a replacement, and vice versa. During the reconstruction phase, decoding operations $\hat{y} \leftarrow \mathsf{Decode}(\tilde{w})$ can handle up to $(\eta - 1)/2 = 2$ errors for each partition.

codeword $\mathbf{w} \in \mathcal{W}_1$, i.e., no helper data $\mathbf{h}$ needs to be stored in addition to the index pointers. Likewise, the reconstruction reduces to either $\hat{\mathbf{y}} \leftarrow \mathsf{Correct}(\tilde{\mathbf{y}})$ or $\hat{\mathbf{z}} \leftarrow \mathsf{Decode}(\tilde{\mathbf{y}})$.

Min-Entropy Loss

For stand-alone IBS, and under the assumption of i.i.d. response bits X_i, the pre-chosen secret Y remains uniformly distributed, even for those who observe the index pointers H. For the joint optimization with a subsequent secure sketch, it holds that Z remains uniformly distributed. As a side note, Yu et al. [189] applied the IBS scheme to the concatenated response X of a RO Sum PUF. As discussed in Chapter 3, the latter design suffers from functional correlations, and the assumption of i.i.d. response bits X_i is hence not necessarily valid. Some theory, but not a rigid security proof, was developed in support. According to Becker et al. [15], the suggested instances exhibit a larger entropy loss than what is claimed.

Bit Error Rate

Even under the assumption of i.i.d. response bits X_i, it is not straightforward to evaluate the expected bit error rate $\mathbb{E}[P_{\text{error},y}]$ in an exact, analytical manner. As for the scheme of Suh and Devadas [161] in Section 4.5.2, order statistics result in nested integrals that do not simplify well. For ease of analysis, we only consider instances of the generalized IBS scheme where $\eta = 1$ bit is selected for each partition of size λ, i.e., the single-parameter version of Yu and Devadas [187]. An exact formula is then given in (4.48), where $p_{\text{error}}^{\infty}(\omega)$ is as defined in (4.2).

$$\begin{aligned}\mathbb{E}_{v \leftarrow V}[P_{\text{error},y}] = {} & \frac{1}{2}\int_{-\infty}^{\infty} \mathsf{f}\big(\omega_{(1)}\big) p_{\text{error}}^{\infty}\big(\omega_{(1)}\big)\,\mathrm{d}\omega_{(1)} \\ & + \frac{1}{2}\int_{-\infty}^{\infty} \mathsf{f}\big(\omega_{(\lambda)}\big) p_{\text{error}}^{\infty}\big(\omega_{(\lambda)}\big)\,\mathrm{d}\omega_{(\lambda)}, \\ \text{with } \mathsf{f}\big(\omega_{(\lambda)}\big) = {} & \lambda\,\mathsf{f}_{\text{norm}}\big(\omega_{(\lambda)}\big)\big(\mathsf{F}_{\text{norm}}\big(\omega_{(\lambda)}\big)\big)^{\lambda-1} \\ \text{and } \mathsf{f}\big(\omega_{(1)}\big) = {} & \lambda\,\mathsf{f}_{\text{norm}}\big(\omega_{(1)}\big)\big(1 - \mathsf{F}_{\text{norm}}\big(\omega_{(1)}\big)\big)^{\lambda-1}.\end{aligned} \tag{4.48}$$

4.5.5 Von Neumann

Several recently proposed schemes that are based on von Neumann's [125] debiasing algorithm from 1951 transform i.i.d. response bits X_i into a uniform distribution Y, even for those who observe the newly introduced helper data H. This was first proposed by van der Leest et al. [104], and later also by Van Herrewege [170] and Maes et al. [111]. Again, a considerable fraction of the response bits x is discarded in order to restore the balance between the expected number of 0s and 1s. A crucial difference with IBS [187, 62] in Section 4.5.4 is that the outgoing secret Y is determined by the response X rather than pre-chosen.

Figure 4.21 specifies several variations of the von Neumann-based schemes. All versions start with a biased response $\mathbf{x} \in \{0,1\}^{\lambda}$ that is partitioned into pairs of bits. For the most basic version, the first bit of pairs 01 and 10 is retained, while pairs 00 and 11 are discarded as a whole. Under the assumption of i.i.d. response bits X_i, the number of retained bits obeys a binomial distribution $B(\lfloor \lambda/2 \rfloor, 2p_{\text{bias}}(1 - p_{\text{bias}}))$. A second pass of the algorithm on the decimated discarded pairs increases the expected number of retained bits. Three or more passes can be performed, but the gain in retention ratio drops rapidly with each additional pass. The outgoing secret Y is uniformly distributed, under the assumption of i.i.d. response bits X_i, and is usually fed into a subsequent secure sketch. Recall that the min-entropy loss of all sketch constructions in Table 4.4 is exactly $(n - k)$ bits for a uniformly distributed Y.

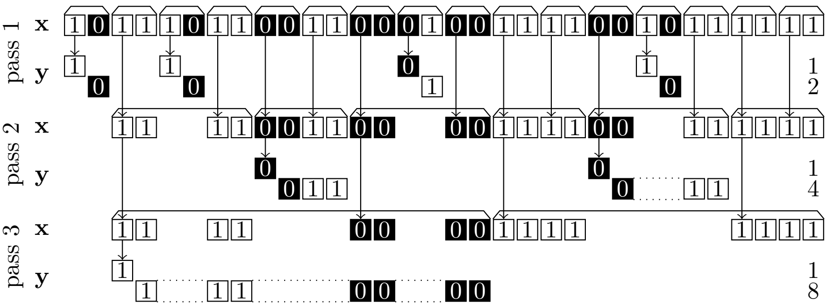


Figure 4.21: Several variations of the von Neumann-based debiasing schemes [104, 170, 111]. Only the enrollment is depicted. In the first pass, 01 and 10 sequences are retained, while 00 and 11 sequences are discarded. Optionally, a second pass can retain previously discarded 0011 and 1100 sequences, but therefore not the 0000 and 1111 sequences. A third pass retains previously discarded 00001111 and 11110000 sequences, but therefore not the 00000000 and 11111111 sequences. Originally, only the first bit of each retained sequence contributes to $\mathbf{y}$. Under the assumption of i.i.d. response bits X_i, secret Y is then uniformly distributed. A joint optimization with the repetition code ζ_2 of a subsequent secure sketch allows to retain sequences as a whole, i.e., 2, 4, and 8 bits are retained in the first, second and third pass respectively.

Maes et al. [111] improved the retention ratio for concatenated codes $\zeta_2 \circ \zeta_1$ that embed an $[n_2, k_2 = 1, d_2 = n_2]$ repetition code ζ_2 with n_2 even. Undecimated sequences are retained as a whole and shuffled such that each sequence in $\mathbf{y}$ remains within the boundaries of a single repetition code. There is no additional min-entropy loss, given that a repetition code reveals all pairwise equalities among its corresponding response bits anyway. The number of passes imposes a lower bound on the size of the repetition code, e.g., $n_2 \geq 8$ if three passes are performed.

Finally, we highlight that the von Neumann debiasing scheme in Figure 4.22 was claimed to be reusable [111]. This claim holds, despite overlooking the misuse of Boyen's proof and stating that a stand-alone sketch is reusable. An unintended side effect of introducing placeholder pairs is that individual bit error rates cannot be estimated anymore. Helper data only allows for the estimation of pairwise error rates. The scheme is considerably less efficient than other von Neumann variants though, showing that reusability comes at a price.

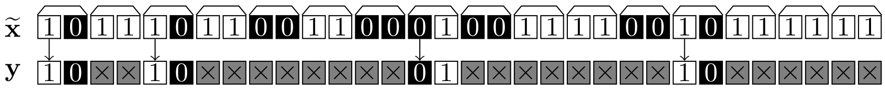


Figure 4.22: A reusable von Neumann debiasing scheme that allows for the enrollment of an unlimited number of noisy PUF responses $\widetilde{\mathbf{x}}$. There is a single pass that retains 01 and 10 sequences as a whole. The 00 and 11 sequences merely serve as placeholders, contributing to neither the enrollment nor the reproduction, i.e., only part of $\widetilde{\mathbf{x}} \oplus \mathbf{w}$ is released as helper data. The $[n_2, k_2 = 1, d_2 = n_2]$ repetition code with n_2 even is virtually shortened due to local placeholder pairs.

Min-Entropy Loss

If only the first bit of each sequence in $\{01, 10, 0011, 1100, \cdots\}$ is retained, the resulting secret Y is uniformly distributed under the assumption of i.i.d. response bits X_i. Alternatively, if sequences are retained as a whole, Y is not uniformly distributed anymore, but the residual min-entropy of the subsequent secure sketch remains the same.

Failure Rate

Under the assumption of i.i.d. response bits X_i, the expect bit error rate $\mathbb{E}[P_{\text{error},y}]$ for all von Neumann-based debiasing schemes is given in (4.49).

$$\mathbb{E}_{v \leftarrow V}[P_{\text{error},y}] = \frac{1}{2}\mathbb{E}_{v \leftarrow V}[P_{\text{error},x} | X_i = 0] + \frac{1}{2}\mathbb{E}_{v \leftarrow V}[P_{\text{error},x} | X_i = 1]. \quad (4.49)$$

However, for the variation of Maes et al. [111] where sequences in {01, 10, 0011, 1100, $\cdots$} are retained as a whole, the outgoing bit error rates $P_{\text{error},y}$ are not i.i.d. anymore. The given expression is hence not compatible with the failure rate of a concatenated code in (4.14). The correct expression for the expected failure rate of the repetition code ζ_2 is given in (4.50). It is assumed that this failure rate equals $1/2$ whenever $n_2/2$ errors are detected.

$$\begin{aligned}\mathbb{E}_{v\leftarrow V}[P_{\text{fail},\zeta_2}] = 1 - \sum_{j=0}^{t} &\mathsf{f}_{\text{bino}}\Big(j; \frac{n_2}{2}, \mathbb{E}_{v\leftarrow V}[P_{\text{error},x}|X_i = 0]\Big)\\ &\mathsf{F}_{\text{bino}}\Big(t-j; \frac{n_2}{2}, \mathbb{E}_{v\leftarrow V}[P_{\text{error},x}|X_i = 1]\Big)\\ -\frac{1}{2}\sum_{j=0}^{t+1} &\mathsf{f}_{\text{bino}}\Big(j; \frac{n_2}{2}, \mathbb{E}_{v\leftarrow V}[P_{\text{error},x}|X_i = 0]\Big)\\ &\mathsf{f}_{\text{bino}}\Big(t+1-j; \frac{n_2}{2}, \mathbb{E}_{v\leftarrow V}[P_{\text{error},x}|X_i = 1]\Big).\end{aligned} \tag{4.50}$$

Helper Data Manipulation

In the worst case, helper data manipulation attacks can result in the complete retrieval of the secret $\mathbf{k}$. Consider the variation of Maes et al. [111] where sequences in $\{01, 10, 0011, 1100, \cdots\}$ are retained as a whole. Given that the helper data allows to shuffle sequences, an attacker can swap the order of two equal-length sequences. By observing the failure rate p_{fail} for reconstructing the key $\mathbf{k}$, it is known whether they are equal or complementary. This mechanism is applied iteratively.

4.5.6 Comparisons

A frequently reoccurring problem with the PUF literature is that newly proposed schemes often lack a proper comparison to what already exists. This remark certainly applies to the four previously discussed debiasing schemes: skewed SMV [113], pattern matching [132, 133], IBS [187, 62], and the von Neumann adaptations [104, 170, 111]. Moreover, for the latter two schemes in particular, it is conjectured that a stand-alone secure sketch cannot handle biased distributions well. This corresponds to an educated guess, originating from the extrapolation of repetition code insights and/or the application of the $(n-k)$ bound. Our newly developed bounds in Section 4.3.5 can resolve this motivational uncertainty, as elaborated next.

Table 4.8 quantifies the implementation footprint for a fuzzy extractor that produces a 128-bit key from a biased PUF. For IBS, the expected failure rate for reconstructing repetition codewords, i.e., $\mathbb{E}_{v\leftarrow V}[P_{\text{fail},\mathcal{C}_2}]$, is approximated via Monte Carlo simulations of size 10^6, given that an exact evaluation via joint order statistics is not straightforward. A complication for the von Neumann schemes is that the length of $\mathbf{y}$ varies with $\mathbf{x}$. Therefore a yield is defined, i.e., the probability that sufficient bits can be provided for the subsequent secure sketch. An exact analytical evaluation of the retention ratio is computationally intensive from 3 passes onwards [111], so we rely on Monte Carlo simulations of size 10^6 instead.

It turns out that a stand-alone sketch is competitive in the low-bias region, e.g., $p_{\text{bias}} \in [0.42, 0.58]$. For high-bias situations, either a redesign of the PUF or a debiasing scheme is needed. It is beneficial to couple the discussed debiasing schemes to a sketch that has a k-bit rather than an n-bit output. The uniformly distributed output is then directly usable as a key $\mathbf{k}$, thereby eliminating the need for a randomness extractor Ext.

4.6 PUFs with Helper Data Input

Several RO-based PUFs have, in addition to a challenge $\mathbf{c}$, a constant helper data input $\mathbf{h}$. The latter input aims to improve the reliability and/or uniformity characteristics of the challenge-response behavior. The constructions that we have analyzed are highly vulnerable to helper data manipulation attacks. We provide two examples in this thesis, and refer to our DATE 2014 publication for more details.

- Yin and Qu [182] proposed an algorithm for pairing ROs such that the respective differences of their frequencies f exceed a certain threshold. For an array of m ROs, up to $\lfloor m/2 \rfloor$ disjunct pairs that each generate a response bit r are selected during the enrollment. The oscillator indices of each pair are stored as helper data $\mathbf{h}$. Compared to the static pairs of Yu et al. [183], the reproducibility of the response bits improves. However, even under the assumption of a silent application, as defined in Table 4.1, an attacker can recover the key $\mathbf{k}$ through helper data manipulation. For example, by swapping the order of two pairs, while observing the failure rate p_{fail} for key reconstruction, an attacker can determine whether or not the respective response bits r are equal. After a sufficient number of iterations, the concatenated response $\mathbf{x}$ is completely revealed, except for one degree of freedom that corresponds to a bit-wise inversion of $\mathbf{x}$.
- After plotting the topology of a two-dimensional array of experimentally obtained frequencies $f_{i,j}$, undesired trends are often visible. Systematic process variations might cause frequencies to be relatively low and high in opposite corners, for example. One potential resolution is the pairing of

Table 4.8: The implementation footprint of a practical fuzzy extractor, using the generalized IBS and von Neumann debiasing schemes respectively. The setting is identical to Table 4.6. A residual min-entropy $\tilde{\mathbb{H}}_\infty(X|H) \geq 128$ and an expected failure rate $\mathbb{E}[P_{\text{fail}}] \leq 10^{-6}$ are imposed. All BCH codes with $n_1 \in \{7, 15, 31, 63, 127, 255\}$ are considered. For IBS and the von Neumann schemes, we consider all repetition codes with $n_2 \in \{1, 3, 5, 7, 9, 11, 13\}$ and $n_2 \in \{8, 10, 12, 14\}$ respectively. For IBS in particular, we impose the constraint $\lambda \leq 16$.

	p_{bias}	$\mathbb{E}[P_{\text{error}}]$	$\mathbb{E}[P_{\text{error}}\|0]$	$\mathbb{E}[P_{\text{error}}\|1]$	Parameters	Retention	$q \times [n_2, k_2, d_2] \circ [n_1, k_1, d_1]$	$\tilde{\mathbb{H}}_\infty(X\|W)$	PUF size n	$\mathbb{E}[P_{\text{fail},\zeta_2}]$	$\mathbb{E}[P_{\text{fail}}]$
Generalized IBS	0.50	≈ 10.0%	≈ 10.0%	≈ 10.0%	$\lambda = 7$	≈ 71.4%	$2 \times [5, 1, 5] \circ [127, 64, 21]$	128	1778	≈ 1.01E−2	≈ 1.70E−7
	0.54	≈ 9.96%	≈ 10.6%	≈ 9.40%	$\lambda = 7$	≈ 71.4%	$2 \times [5, 1, 5] \circ [127, 64, 21]$	128	1778	≈ 1.12E−2	≈ 4.57E−7
	0.58	≈ 9.81%	≈ 11.2%	≈ 8.79%	$\lambda = 7$	≈ 71.4%	$1 \times [5, 1, 5] \circ [255, 131, 37]$	131	1785	≈ 1.41E−2	≈ 6.46E−9
	0.62	≈ 9.58%	≈ 11.8%	≈ 8.18%	$\lambda = 8$	62.5%	$2 \times [5, 1, 5] \circ [127, 64, 21]$	128	2032	≈ 1.17E−2	≈ 7.09E−7
	0.66	≈ 9.24%	≈ 12.5%	≈ 7.56%	$\lambda = 8$	62.5%	$1 \times [5, 1, 5] \circ [255, 131, 37]$	131	2040	≈ 1.83E−2	≈ 3.59E−7
	0.70	≈ 8.80%	≈ 13.2%	≈ 6.92%	$\lambda = 9$	≈ 77.8%	$1 \times [7, 1, 7] \circ [255, 131, 37]$	131	2295	≈ 1.90E−2	≈ 6.27E−7
	0.74	≈ 8.24%	≈ 13.9%	≈ 6.27%	$\lambda = 11$	≈ 81.8%	$1 \times [9, 1, 9] \circ [255, 131, 37]$	131	2805	≈ 1.62E−2	≈ 5.72E−8
	0.78	≈ 7.57%	≈ 14.6%	≈ 5.58%	$\lambda = 13$	≈ 84.6%	$1 \times [11, 1, 11] \circ [255, 131, 37]$	131	3315	≈ 1.65E−2	≈ 7.32E−8
	0.82	≈ 6.76%	≈ 15.4%	≈ 4.85%	$\lambda = 16$	≈ 68.8%	$1 \times [11, 1, 11] \circ [255, 131, 37]$	131	4080	≈ 1.66E−2	≈ 7.57E−8
	0.86	≈ 5.80%	≈ 16.4%	≈ 4.07%	$\lambda = 16$	≈ 81.3%	$2 \times [13, 1, 13] \circ [255, 71, 59]$	142	8160	≈ 3.57E−2	≈ 2.85E−8
	0.90	≈ 4.64%	≈ 17.5%	≈ 3.21%	$\lambda = 16$	≈ 81.3%	$3 \times [13, 1, 13] \circ [255, 45, 87]$	135	12240	≈ 7.51E−2	≈ 6.42E−7
von Neumann	0.50	≈ 10.0%	≈ 10.0%	≈ 10.0%	3 passes multi-out ($n_2 \geq 8$) retention yield 99%	≈ 83.4%	$4 \times [8, 1, 8] \circ [63, 36, 11]$	144	2418	≈ 2.73E−3	≈ 9.85E−8
	0.54	≈ 9.96%	≈ 10.6%	≈ 9.40%		≈ 81.6%	$4 \times [8, 1, 8] \circ [63, 36, 11]$	144	2471	≈ 2.72E−3	≈ 9.73E−8
	0.58	≈ 9.81%	≈ 11.2%	≈ 8.79%		≈ 77.0%	$4 \times [8, 1, 8] \circ [63, 36, 11]$	144	2617	≈ 2.71E−3	≈ 9.37E−8
	0.62	≈ 9.58%	≈ 11.8%	≈ 8.18%		≈ 70.7%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	2675	≈ 8.70E−4	≈ 9.81E−7
	0.66	≈ 9.24%	≈ 12.5%	≈ 7.56%		≈ 63.6%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	2971	≈ 8.52E−4	≈ 9.05E−7
	0.70	≈ 8.80%	≈ 13.2%	≈ 6.92%		≈ 56.2%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	3365	≈ 8.29E−4	≈ 8.12E−7
	0.74	≈ 8.24%	≈ 13.9%	≈ 6.27%		≈ 48.6%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	3885	≈ 8.00E−4	≈ 7.06E−7
	0.78	≈ 7.57%	≈ 14.6%	≈ 5.58%		≈ 41.4%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	4567	≈ 7.65E−4	≈ 5.91E−7
	0.82	≈ 6.76%	≈ 15.4%	≈ 4.85%		≈ 33.5%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	5650	≈ 7.23E−4	≈ 4.72E−7
	0.86	≈ 5.80%	≈ 16.4%	≈ 4.07%		≈ 26.1%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	7237	≈ 6.73E−4	≈ 3.55E−7
	0.90	≈ 4.64%	≈ 17.5%	≈ 3.21%		≈ 18.5%	$3 \times [10, 1, 10] \circ [63, 45, 7]$	135	10212	≈ 6.13E−4	≈ 2.45E−7

physically neighboring ROs [116]. Alternatively, Yin and Qu [181] suggest storing the coefficients of a two-dimensional polynomial that models the undesired trends. In the field, a substraction can then be performed. However, key-recovery through helper data manipulation might be feasible for interactive applications, as defined in Table 4.1. An attacker can superimpose steep linear or quadratic terms that, depending on how ROs are paired, might render random process variations insignificant. Most of the reproduced bits $\hat{r}$ then have known values, which enables a brute-force search for the few unknown values.

4.7 Conclusion

Secure sketches are the main workhorse of modern PUF-based key generators. The min-entropy loss of most sketches is upper-bounded by $(n - k)$ bits and designers typically instantiate system parameters accordingly. However, the latter bound tends to be overly pessimistic, resulting in an unfortunate implementation overhead. We showcased the proportions for a prominent category of PUFs, with bias and spatial correlations acting as the main non-uniformities. New considerably tighter bounds were derived, valid for a variety of popular but algebraically complex codes. These bounds are unified in the sense of being applicable to seven secure sketch constructions. Deriving tighter alternatives for the $(n - k)$ bound counts as unexplored territory and we established the first significant stepping stone. New techniques may have to be developed in order to tackle more advanced *second-order* distributions. Elaborating a wider range of applications would be another area of progress. We hope to have showcased the potential by debunking the main security claim of the reverse fuzzy extractor.

Besides from secure sketches and fuzzy extractors, we analyzed the security of various other HDAs. Our survey showcases that a proven expression on the min-entropy loss can only be derived for ideal-case distributions, most notably, i.i.d. response bits. In practice, PUFs typically do not comply with this assumption. Regardless of whether a universe of ideal-case distributions is relevant, we at least avoid comparing apples and oranges. It would be unfair to draw performance comparisons with a secure sketch based on the $(n - k)$ upper bound on the min-entropy loss, given that better bounds can be obtained for ideal-case distributions. By using our newly developed bounds, we are the first to provide proper existential motivation for debiasing schemes. Moreover, we put a spotlight on the often overlooked helper data manipulation threats.

Comparing failure rates for the key reconstruction with respect to well-chosen manipulations turns out to be highly effective: most schemes are vulnerable to some extent.

Chapter 5

A Survey on PUF-Based Entity Authentication

We consider a common authentication scenario between two parties: a low-cost, resource-constrained token and a resource-rich server. Practical instantiations of a token include the following: RFID tags, smart cards, and the nodes of a wireless sensor network. Unilateral, i.e., one-way, or possibly mutual, i.e., two-way, entity authentication is the primary objective. Occasionally, the authentication protocol is required to preserve the privacy of a token. The server usually has secure computing and storage at its disposal, while tokens are exposed to physical attacks. We review 21 PUF-based authentication protocols in chronological order, starting from the advent of silicon PUFs in the early 2000s, and ending with the 2016 state-of-the-art. Numerous security and practicality issues are revealed. The assessment is aided by a unified notation and a transparent framework of protocol requirements.

Version History. The version history of this chapter is as follows. A first, preliminary version of our survey was presented at the *16th Conference on Cryptographic Hardware and Embedded Systems* (CHES 2014). The exact same version was also presented at ChinaCrypt 2014, which is a national conference without proceedings. A second, journal version was published in the *ACM Computing Surveys*. This chapter comprises a third version, where we once again update, extend, and improve our contributions. The author of this PhD thesis is the main contributor to the aforementioned publications; Roel Peeters and Dries Schellekens have given expert advice. As a side contribution, the PRNG of

Mandel Yu's newly proposed protocol in the *IEEE Transactions on Multi-Scale Computing Systems* (TMSCS) has been designed by the author of this PhD thesis.

Organization. The remainder of this chapter is organized as follows. Section 5.1 lists the requirements of a PUF-based authentication protocol. Section 5.2 discusses authentication via PUF-based key generation, which is to be considered as the default solution. Section 5.3 comprises our analysis of 21 alternative solutions that most frequently make use of a strong PUF. Section 5.4 provides a summary of our assessment. Section 5.5 concludes this chapter.

5.1 Protocol Requirements

PUFs require a special flavor of protocol design. We are the first to explicitly list an extensive set of protocol requirements. Sections 5.1.1 to 5.1.10 describe a mixture of PUF-induced requirements and more conventional concerns. The list is tailored to the vast majority of PUF-based protocols, where the authenticity of a token and/or the server is verified, but can be mapped quite easily to other security objectives. The urgency of individual items ranges from strictly required to highly desired, as nuanced hereafter.

5.1.1 Complete Specification

A protocol should be specified in a complete, unambiguous manner. Although this seems obvious, we observe that many proposals do not comply. Furthermore, the use of a particular PUF design should be suggested, at least for protocols of which the security and/or functional behavior is highly reliant on, e.g., its inherent resistance to machine learning attacks. As a side note, there is considerable advantage in having a graphical representation of a protocol, clearly detailing all computations and exchanged messages. This facilitates the analysis considerably. We observe that many proposals focus on a text-based description, with only a minimal graphical representation as support. In this work, we specify each protocol in a detailed graphical manner, while using a unified notation. For ease of understanding, we make abstraction of non-essential refinements regarding the two-party setting, e.g., multiple servers, a trusted issuer to aid the enrollment, a distinction between RFID readers and the back-end server, etc.

5.1.2 Resilience against Leakage of NVM Contents

A PUF-based protocol should remain secure even when the contents of a token's OTP/MTP NVM, if present, are compromised. The reasoning is as follows. OTP NVM allows storing a random variate of a uniformly distributed secret in a noiseless manner, so when presumed to be physically secure, it outclasses the noisy, non-uniformly distributed responses of a weak, SRAM-like PUF. Likewise, a random variate stored in OTP MTP can instantiate a noiseless, device-unique function that is harder to learn than a strong PUF. Consider for example a challenge-response mapping $\mathbf{r} \leftarrow \mathsf{MAC}(\mathbf{c}; \mathbf{k})$, where the key $\mathbf{k}$ is stored in physically secure NVM. An ultra-lightweight MAC algorithm might already suffice to improve upon the modeling resistance of a typical strong PUF design. Recall that due to the presence of noise, the PUF designs in Section 3.1.3 are constrained to relatively simple operations only, i.e., addition, substraction, XORing, and comparisons with a threshold 0.

5.1.3 Able to Handle Noise

A PUF-based protocol should remain functional despite the noisiness of the responses of its PUF. Recall that the expected bit error rate with respect to a prerecorded reference response is typically $1\% - -20\%$, largely depending on aging and changes of the IC's environment. The lowest noise levels apply to laboratory settings where the environment is ultra-stable. Higher noise levels apply to market products, which are typically supposed to remain functional in a wide range of temperatures, for example.

5.1.4 Resistance to Machine Learning Attacks

As previously discussed in Section 3.1.1, strong PUFs are too fragile for unprotected exposure. This is demonstrated by a history of machine learning attacks [100, 118, 145]. So far, there does not exist a single well-validated design of a strong PUF that is both practical and fully resistant to modeling. Existing designs usually need to team up with a cryptographic algorithm in order to fully mitigate this threat. Several proposals opt for more lightweight logic, such as XORing, a PRNG, a TRNG, etc. Mostly, this offers partial protection only, as becomes clear later-on.

5.1.5 Expanding the Response of a Strong PUF

So as to counteract brute-force and random guessing attacks on a PUF-based protocol, the responses should be of sufficient length. Unfortunately, practical realizations of a strong PUF tend to provide a short response only, most frequently, a single bit r, as is the case for all designs in Section 3.1.3. Two methods are frequently used for concatenating single-bit responses r into a long string $\mathbf{x} \in \{0,1\}^{\lambda}$. First, λ replicas of a PUF circuit can evaluate the same challenge $\mathbf{c}$ in parallel [80, 85], which imposes a considerable area overhead. Second, a single PUF circuit can evaluate a list of λ challenges, which imposes a considerably latency overhead. Most frequently, a PRNG or a cryptographic algorithm is used to expand a single master challenge into a list of secondary challenges [38, 130, 54, 132, 171, 119, 190, 91]. We argue that the expansion method should be specified as an explicit part of the protocol. In addition to the importance of properly reflecting the consumed resources, it turns out that some expansion methods enable new attacks.

5.1.6 Low-Cost and Resource-Constrained

We evaluate the PUF lightweight premise. Low-cost manufacturing is mainly an issue for proposals that rely on MTP NVM. However, as practically every protocol building block requires a physically secure implementation, it might extend to a more general concern, largely depending on the selected countermeasures. Constraints in available resources are mainly an issue for proposals that rely on cryptographic algorithms.

5.1.7 Easy-to-instantiate

Ideally, one should not make exacting assumptions about the PUF so that the protocol can be instantiated easily. There is considerable advantage in the design of generic protocols that are compatible with most practical PUF designs. First, efficiency and performance characteristics differ for every PUF design in terms of speed, area, power, noisiness, etc. Flexibility of choice hence allows for better optimization with respect to a given set of constraints. Second, the recent proliferation of physical attacks on PUFs promotes to envision them as easy-to-replace building blocks. For all surveyed protocols, it is simply assumed that physical attacks on the PUF are infeasible. Recall that PUF-based key generation is compatible with quasi every weak or strong PUF. The proposals under review are usually more specific and often require a strong PUF. Furthermore, additional constraints are imposed, most frequently with respect to the modeling resistance.

5.1.8 Resistance to Protocol Attacks

An authentication protocol should be resistant to conventional impersonation and *denial-of-service* (DoS) attacks. In order to distinguish these from the previously discussed machine learning attacks, this part of our analysis assumes that the functional behavior of a PUF is ideal, i.e., behaving as a random oracle. For DoS attacks, the scope is limited to non-invasive, remote exploits such as the desynchronization of a state. Acts of vandalism where a token is, for example, burned or crushed are out of scope and would be trivial to perform anyway under the common assumption that an attacker has physical access. Additional security claims such as token privacy [60] should be validated as well. For protocols that claim resilience to the leakage of a token's NVM contents, the feasibility of all previously listed attacks should be reevaluated. It even makes sense to reevaluate the feasibility of DoS attacks, as it could have a benefit to perform such an attack with delay, in a remote, non-mechanical manner.

The assumed capabilities of an attacker should be realistic with respect to the intended application. Unfortunately, it is an occasional practice to mitigate realistic threats with unrealistic assumptions. Luckily, most proposals stick to a fully insecure communication channel between token and server. Moreover, physical attacks that retrieve the state of a token are usually assumed to be feasible. It is realistic to assume that a token and its PUF are still functional after having performed such an attack. Otherwise, an attacker cannot interfere with a genuine protocol run anymore, and the benefit of the resilience claim would be limited. Note that server impersonation and DoS attacks are irrelevant for a mechanically destroyed token. Only the privacy of past protocol runs and to some extent also token impersonation attacks would still be relevant.

5.1.9 Scalability: Identification Prior to Authentication

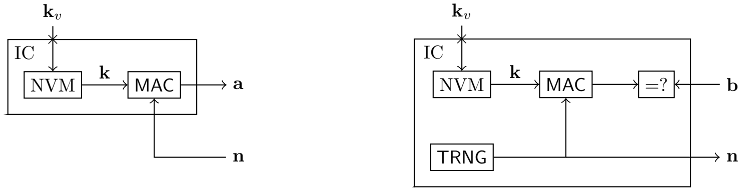
Ideally, the execution time of a protocol should not increase with the number of registered tokens. Recall that the server might have to interact with thousands to millions of tokens, depending on the use case. It is essential to distinguish between the authentication and the identification of a token. The former refers to the secure verification of an identity while the latter is limited to an unverified claim. Many of the surveyed protocols instantly provide token authentication. Unfortunately, identification prior to authentication is more practical. Otherwise, the server would have to iterate over all registered tokens in order to establish a match. There is a simple solution if privacy is of no concern. Each token can then store a unique identifier **i** in insecure OTP NVM and transfer its value to the server at the start of a protocol run. Alternatively, one could opt for a noisy PUF-generated identifier [107]; matching noisy identifiers is a fairly straightforward operation. We will not further comment on the absence of an identification step, except for proposals that simultaneously claim to preserve the privacy of a token.

5.1.10 On the Mutual Authentication Order

An attacker's potential to freely query a token is a major security concern. Therefore, it is a good practice to establish server authenticity first, in the case of mutual authentication [61]. The corresponding reduction in attack surface inherently benefits the security and privacy objectives. This might even be the reason to provide mutual authentication in the first place, rather than providing token authenticity only. Although several protocols adopt the opposite authentication order, we do not further discuss this matter, as it comprehends a guideline rather that a stringent requirement.

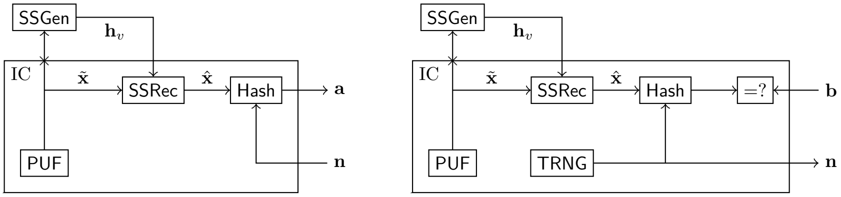
5.2 Authentication via PUF-Based Key Generation

The most straightforward method for crafting a PUF-based authentication protocol is as follows: select a conventional, preferably well-validated authentication protocol from the literature and replace key-storage in OTP NVM with PUF-based key generation, given that the latter is often believed to be more physically secure. Although such concatenations are not the primary interest of this chapter, they establish a baseline in terms of security, extensibility, and efficiency. A well-validated protocol combined with a well-understood mechanism for PUF-based key generation, e.g., a fuzzy extractor [39], might provide excellent security guarantees. Likewise, the extensibility to security objectives other than entity authentication is outstanding and might benefit from a large body of work. Efficiency might be a major concern though, given that cryptographic algorithms and an error-correcting code are expected to be involved. In order to facilitate making comparisons with the surveyed PUF-based protocols later-on, we now embody the common baseline through four reference authentication protocols.



(a) Reference I-A.

(b) Reference I-B.



(c) Reference II-A.

(d) Reference II-B.

Figure 5.1: The hardware of a token for four reference authentication protocols. Intermediary registers and control logic are not drawn. The symbol × on the boundary of the IC denotes a one-time interface that is disabled after the enrollment.

5.2.1 Reference Protocols

We first specify two unilateral authentication protocols that rely on physically secure OTP NVM, i.e., an assumption that does not necessarily hold in practice. Reference protocol I-A verifies the authenticity of a token and is specified in Figure 5.2; a block diagram of the token hardware is shown in Figure 5.1(a). Each token stores a device-unique secret key $\mathbf{k}$ in OTP NVM. A keyed cryptographic algorithm performs the authentication. Among several possibilities, we opt for a MAC algorithm. If the number of genuine authentications is large with respect to the length of nonce $\mathbf{n}$, the repercussions of the *birthday paradox* are to be considered when instantiating the protocol for a given security level.

Reference protocol I-B verifies the authenticity of the server instead and is specified in Figure 5.3; a block diagram of the token hardware is shown in Figure 5.1(b). Both reference protocols could be intertwined in order to provide mutual authentication. Intertwinement is not to be confused with the less secure alternative where both unilateral authentication protocols are executed independently from each other.

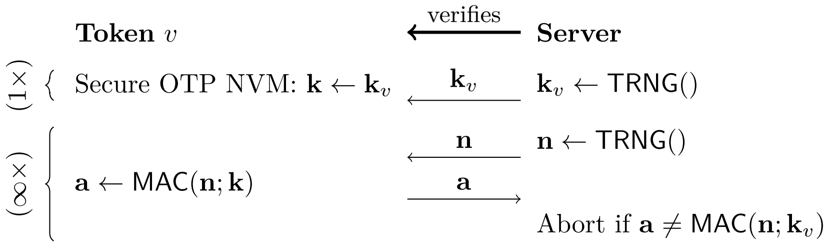

Token v
verifies
Server
$(1\times)$
Secure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_v$
$\mathbf{k}_v$
$\mathbf{k}_v \leftarrow \mathsf{TRNG}()$
$(\infty\times)$
$\mathbf{n}$
$\mathbf{n} \leftarrow \mathsf{TRNG}()$
$\mathbf{a} \leftarrow \mathsf{MAC}(\mathbf{n};\mathbf{k})$
$\mathbf{a}$
Abort if $\mathbf{a} \neq \mathsf{MAC}(\mathbf{n};\mathbf{k}_v)$

Figure 5.2: Reference protocol I-A. The symbol ∞ denotes that the number of protocol runs is virtually unlimited rather than truly infinite.

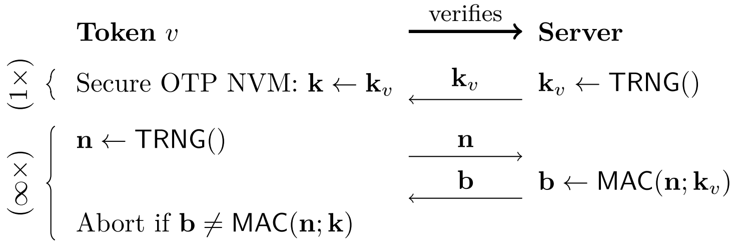

Token v
verifies
Server
$(1\times)$
Secure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_v$
$\mathbf{k}_v$
$\mathbf{k}_v \leftarrow \mathsf{TRNG}()$
$(\infty\times)$
$\mathbf{n} \leftarrow \mathsf{TRNG}()$
$\mathbf{n}$
$\mathbf{b}$
$\mathbf{b} \leftarrow \mathsf{MAC}(\mathbf{n};\mathbf{k}_v)$
Abort if $\mathbf{b} \neq \mathsf{MAC}(\mathbf{n};\mathbf{k})$

Figure 5.3: Reference protocol I-B.

Given that physically secure OTP NVM is hard to obtain, we also specify two unilateral authentication protocols that rely on PUF-based key generation instead. Reference protocol II-A verifies the authenticity of a token and is specified in Figure 5.4; a block diagram of the token hardware is shown in Figure 5.1(c). In principle, a fuzzy extractor could transform the noisy, non-uniformly distributed secret X into a stable, uniformly distributed key K. We perform an optimization by reusing the second stage of the fuzzy extractor, i.e., a cryptographic hash function that is modeled as a random oracle, for authentication purposes. Public helper data $\mathbf{h}$ is stored by the server, or alternatively, at the token side in physically insecure OTP NVM. Not only in the former but also in the latter case, we are highly in favor of implementing an integrity check on the helper data, as is clear from the numerous manipulation attacks described in Chapter 4. Such a check has been omitted for simplicity, given that the protocol merely serves as a tool for drawing conceptual comparisons.

Reference protocol II-B verifies the authenticity of the server instead and is specified in Figure 5.5; a block diagram of the token hardware is shown in Figure 5.1(d). Again, both reference protocols could be intertwined in order to provide mutual authentication.

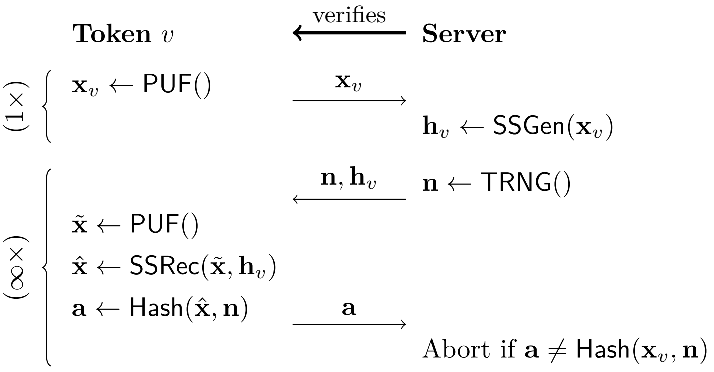

Token v
verifies
Server
$(1\times)$
$\mathbf{x}_v \leftarrow \mathsf{PUF}()$
$\mathbf{x}_v$
$\mathbf{h}_v \leftarrow \mathsf{SSGen}(\mathbf{x}_v)$
$(\infty\times)$
$\mathbf{n}, \mathbf{h}_v$
$\mathbf{n} \leftarrow \mathsf{TRNG}()$
$\tilde{\mathbf{x}} \leftarrow \mathsf{PUF}()$
$\hat{\mathbf{x}} \leftarrow \mathsf{SSRec}(\tilde{\mathbf{x}}, \mathbf{h}_v)$
$\mathbf{a} \leftarrow \mathsf{Hash}(\hat{\mathbf{x}}, \mathbf{n})$
$\mathbf{a}$
Abort if $\mathbf{a} \neq \mathsf{Hash}(\mathbf{x}_v, \mathbf{n})$

Figure 5.4: Reference protocol II-A. Either a weak or a strong PUF could be used. Its responses $\mathbf{r}$ to a list of publicly known challenges $\mathbf{c}$ are concatenated into a lengthy secret $\mathbf{x}$. For a weak PUF, logic for generating a list of challenges might be as simple as a counter. In case of a strong PUF, an LFSR with a hardcoded seed could be used, or alternatively, the resources of the cryptographic hash function could be reused.

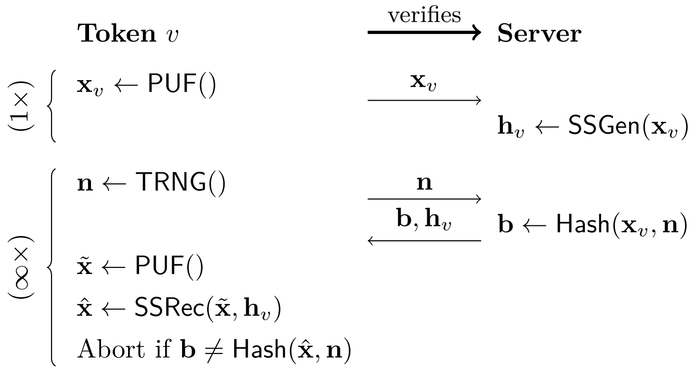

Token v
verifies
Server
$(1\times)$
$\mathbf{x}_v \leftarrow \mathsf{PUF}()$
$\mathbf{x}_v$
$\mathbf{h}_v \leftarrow \mathsf{SSGen}(\mathbf{x}_v)$
$(\infty\times)$
$\mathbf{n} \leftarrow \mathsf{TRNG}()$
$\mathbf{n}$
$\mathbf{b}, \mathbf{h}_v$
$\mathbf{b} \leftarrow \mathsf{Hash}(\mathbf{x}_v, \mathbf{n})$
$\tilde{\mathbf{x}} \leftarrow \mathsf{PUF}()$
$\hat{\mathbf{x}} \leftarrow \mathsf{SSRec}(\tilde{\mathbf{x}}, \mathbf{h}_v)$
Abort if $\mathbf{b} \neq \mathsf{Hash}(\hat{\mathbf{x}}, \mathbf{n})$

Figure 5.5: Reference protocol II-B.

5.2.2 Related Literature

Also in the related literature, several authors have specified a PUF-based authentication protocol where the role of the PUF is limited to the generation of a secret key. An early proposal, which bears a resemblance to reference protocol II-A, is the use of a *physically obfuscated key* by Gassend [49]. Alternatively, Tuyls et al. [167] use public-key cryptography in order to mitigate the need for a shared secret between token and server, although their proposal might not be lightweight. There is the protocol of Bassil et al. [10], which has been badly broken by Safkhani et al. [149]. The protocol of Kardaş et al. [79] does not fully fit our scope, given that it aims to defend against the compromise of an RFID reader.

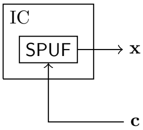
5.3 Analysis of Protocols

We describe and analyze 21 PUF-based authentication protocols in chronological order. As some protocols rely on similar design concepts, we also considered categorizing them. However, given that none of our devised classifications does *fly all the way*, we opt for the history of development as the main theme instead. Sections 5.3.1 to 5.3.21 can be read in arbitrary order, although we recommend getting acquainted with the basic protocol first.

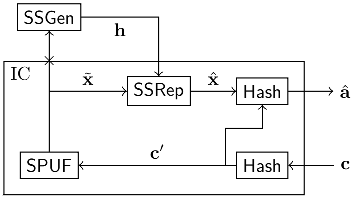
5.3.1 Basic Authentication (2001)

The first and most basic authentication method is proposed by Pappu in his PhD thesis [131]. Although originally intended for optical PUFs, the same protocol has later been applied to silicon PUFs [135, 161, 38, 34]. The protocol is specified in Figure 5.7; a block diagram of the token hardware is shown in Figure 5.6(a). During the enrollment phase and for each token v, the server collects the response $\mathbf{x}$ to g challenges $\mathbf{c}$ that are selected independently, randomly, and uniformly from the set $\mathcal{C}$. In the field, a genuine token should be able to reproduce the prerecorded response $\mathbf{x}$ to each selected challenge $\mathbf{c}$. Only an approximate match is required, i.e., a threshold ϵ on the Hamming distance $\mathsf{HD}(\mathbf{x}, \tilde{\mathbf{x}})$ takes the noisiness of the PUF into account.

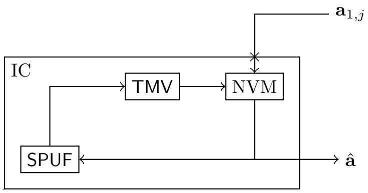
The server should discard each of its prerecorded CRPs after use. Otherwise, even a passive, eavesdropping attacker would be able to replay an old protocol run and impersonate a token. An unfortunate side effect of this countermeasure is that both the storage requirements and the time needed for the enrollment scale proportionally with the number of protocol runs g. Furthermore, the



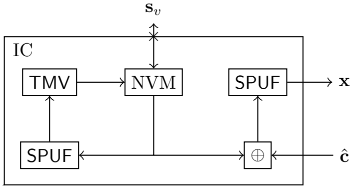
(a) Basic [131].



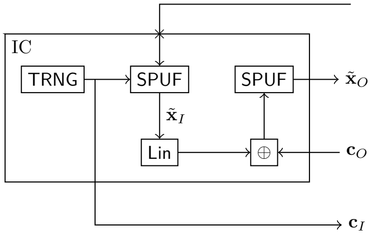
(b) Controlled PUF [51].



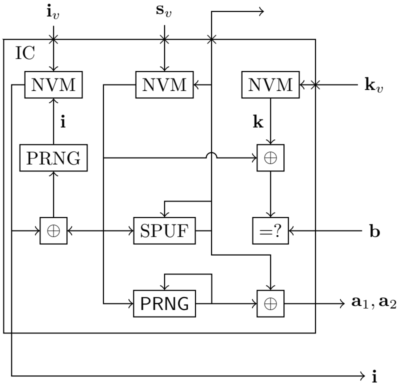
(c) Bolotnyy and Robins [21].



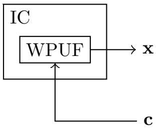
(d) Öztürk et al. [130].



(e) Hammouri et al. [54].

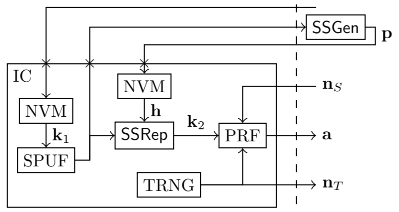


(f) Kulseng et al. [94].

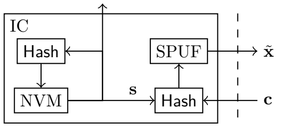


(g) SHIC PUF [143]

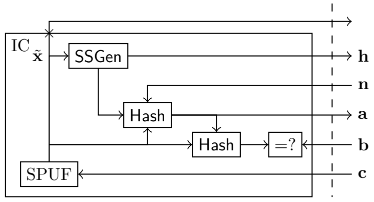
Figure 5.6: The hardware of a token for all surveyed authentication protocols. Intermediary registers and control logic are not drawn. The symbol × on the boundary of the IC denotes a one-time interface that is disabled after the enrollment.



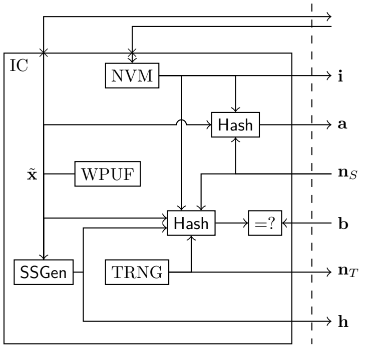
(h) Sadeghi et al. [148].



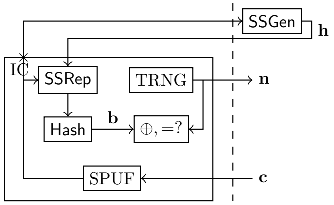
(i) Logically reconfigurable PUF [80].



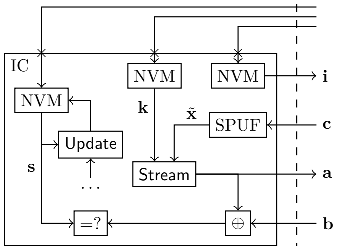
(j) Reverse fuzzy extractor [171].



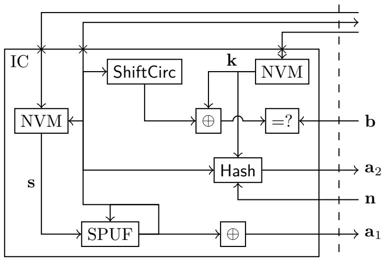
(k) Revised reverse FE [110].



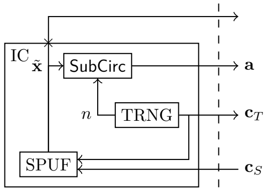
(l) Converse [85].



(m) Lee et al. I [101].

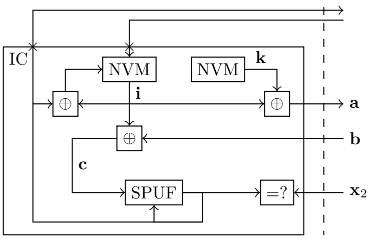


(n) Jin et al. [73].

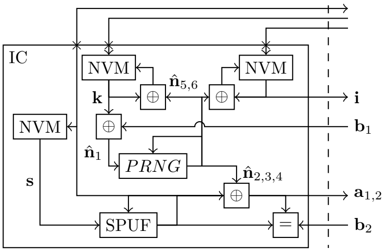


(o) Slender PUF [119].

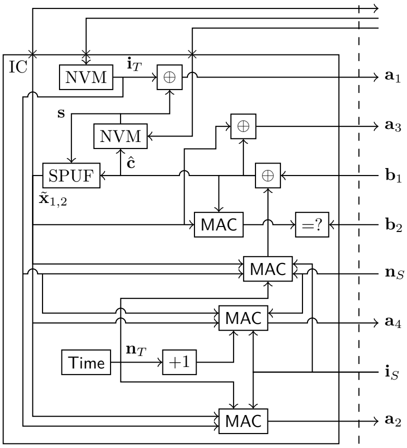
Figure 5.6: The hardware of a token (continued).



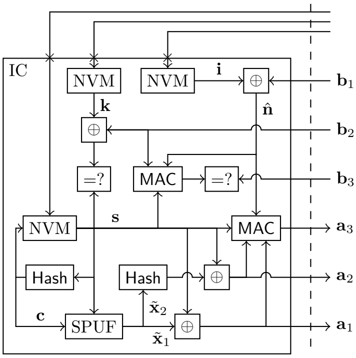
(p) Xu and He [179].



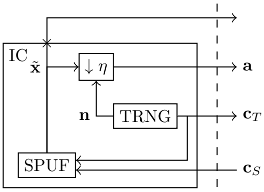
(q) He and Zou [57].



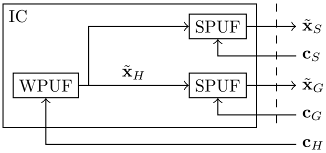
(r) Jung and Jung [75].



(s) Lee et al. II [102].

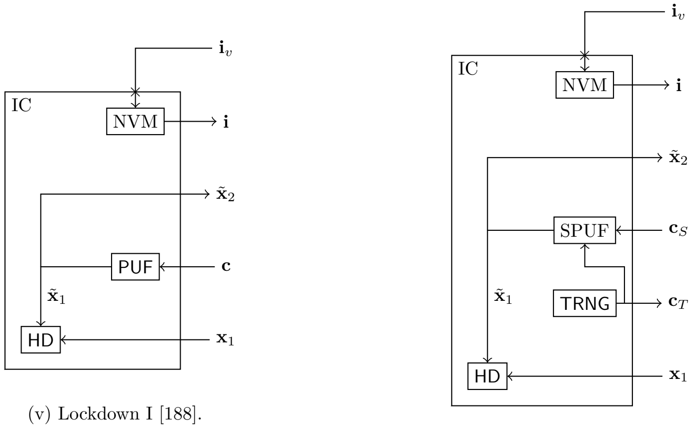


(t) Noise bifurcation [190].



(u) System of PUFs [91].

Figure 5.6: The hardware of a token (continued).



(v) Lockdown I [188].

(w) Lockdown II [188].

Figure 5.6: The hardware of a token (continued).

$$\begin{array}{cllll}
 & \textbf{Token } v & \xleftarrow{\text{verifies}} & \textbf{Server} & \\
\multirow{3}{*}{$(1\times)\left\{\right.$} & & \xleftarrow{\mathbf{c}_{i,v}} & \mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}() & \multirow{2}{*}{$\left.\right\}\ \forall i \in [1,g]$} \\
 & \mathbf{x}_{i,v} \leftarrow \mathsf{SPUF}(\mathbf{c}_{i,v}) & \xrightarrow{\mathbf{x}_{i,v}} & & \\
 & & & g_v \leftarrow g & \\
\multirow{6}{*}{$(g\times)\left\{\right.$} & & & i \leftarrow g_v & \\
 & & & (\mathbf{c},\mathbf{x}) \leftarrow (\mathbf{c}_{i,v},\mathbf{x}_{i,v}) & \\
 & & \xleftarrow{\mathbf{c}} & g_v \leftarrow g_v - 1 & \\
 & \tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c}) & \xrightarrow{\tilde{\mathbf{x}}} & & \\
 & & & \text{Abort if } \mathsf{HD}(\mathbf{x},\tilde{\mathbf{x}}) > \epsilon &
\end{array}$$

Figure 5.7: The most basic PUF-based authentication protocol. Devadas et al. [38] instantiate the strong PUF with an Arbiter PUF. Moreover, they first feed challenge **c** into an LFSR so that the 1-bit responses r to an expanded list of challenges can be concatenated into a lengthy string $\mathbf{x}$.

need for a strong PUF with, e.g., $|\mathcal{C}| = 2^{128}$ unique challenges, is evident. Otherwise, an attacker with physical access to a token could exhaustively collect and tabulate all CRPs. If g is large compared to $|\mathcal{C}|$, the repercussions of the birthday paradox might have to be considered as well. Similarly, the length of concatenated response $\mathbf{x}$ should be, e.g., 128 bits. It is then extremely unlikely that a blind guess of $\mathbf{x}$ happens to be correct.

Observe that the protocol is represented in a strictly server-initiated manner. In the hypothetical case that it was device-initiated instead, an attacker might exhaustively query and deplete the CRP database of the server, which results in DoS. Although corresponding countermeasures could be implemented at the server side, e.g., time-out mechanisms, we do not further discuss this matter.

Machine Learning Attacks (#4)

The proposal does not offer any protection against machine learning attacks. Given physical access to a token, an attacker can construct a predictive model of its PUF so as to allow for impersonation later-on. Becker [13] experimentally demonstrated the feasibility of this attack for a commercial product that relies on Arbiter XOR PUFs. Given that there is no secure instantiation of the protocol due to the lack of an appropriate strong PUF, its practical value is rather limited. We emphasize that this observation applies to silicon PUFs in particular. Optical PUFs [131] are often believed to be more resistant to machine learning attacks, but they are unfortunately incompatible with low-cost, integrated applications.

5.3.2 Controlled PUFs (2002)

In order to counter machine learning attacks, Gassend et al. [51, 50, 49, 52] introduced the rather generic concept of controlled PUFs. We limit ourselves to its most well-known instantiation. The corresponding protocol is specified in Figure 5.8; a block diagram of the token hardware is shown in Figure 5.6(b). A first cryptographic hash function, which precedes the strong PUF, prevents an attacker from choosing challenges arbitrarily. However, it is only the second cryptographic hash function, which succeeds the strong PUF, that effectively counters machine learning attacks. Pre-image resistance implies that an attacker cannot recover the raw, exploitable response $\mathbf{x}$ from a given digest $\mathbf{a}$.

Unfortunately, the second cryptographic hash function necessitates an additional error-correction module. Otherwise, even a single error in the reproduced response $\tilde{\mathbf{x}}$ would be sufficient to trigger an authentication failure, given that cryptographic hash functions are designed to satisfy the *strict avalanche*

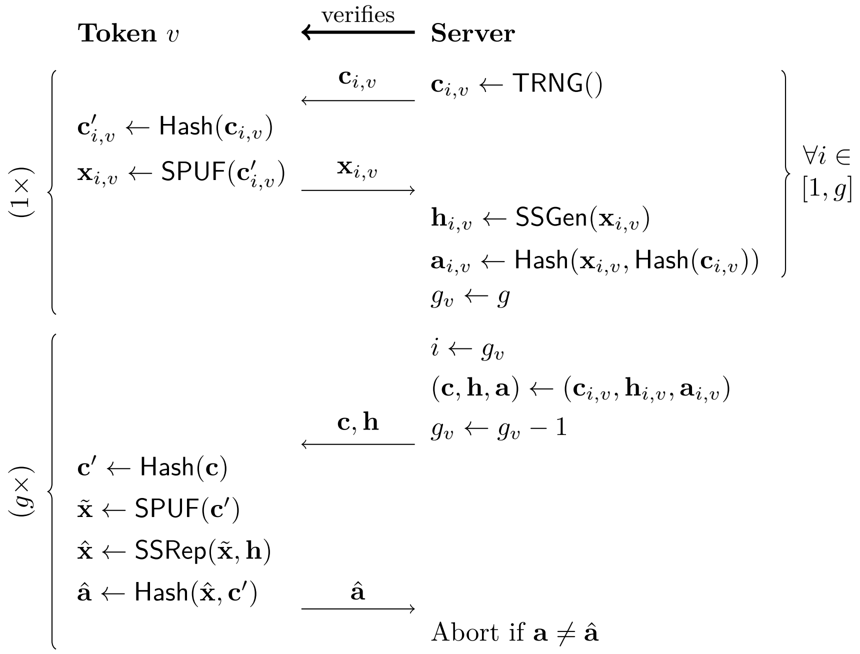

Token v
verifies
Server
$(1\times)$
$\mathbf{c}_{i,v}$
$\mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}()$
$\mathbf{c}'_{i,v} \leftarrow \mathsf{Hash}(\mathbf{c}_{i,v})$
$\mathbf{x}_{i,v} \leftarrow \mathsf{SPUF}(\mathbf{c}'_{i,v})$
$\mathbf{x}_{i,v}$
$\forall i \in [1, g]$
$\mathbf{h}_{i,v} \leftarrow \mathsf{SSGen}(\mathbf{x}_{i,v})$
$\mathbf{a}_{i,v} \leftarrow \mathsf{Hash}(\mathbf{x}_{i,v}, \mathsf{Hash}(\mathbf{c}_{i,v}))$
$g_v \leftarrow g$
$(g\times)$
$i \leftarrow g_v$
$(\mathbf{c}, \mathbf{h}, \mathbf{a}) \leftarrow (\mathbf{c}_{i,v}, \mathbf{h}_{i,v}, \mathbf{a}_{i,v})$
$\mathbf{c}, \mathbf{h}$
$g_v \leftarrow g_v - 1$
$\mathbf{c}' \leftarrow \mathsf{Hash}(\mathbf{c})$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c}')$
$\hat{\mathbf{x}} \leftarrow \mathsf{SSRep}(\tilde{\mathbf{x}}, \mathbf{h})$
$\hat{\mathbf{a}} \leftarrow \mathsf{Hash}(\hat{\mathbf{x}}, \mathbf{c}')$
$\hat{\mathbf{a}}$
Abort if $\mathbf{a} \neq \hat{\mathbf{a}}$

Figure 5.8: Authentication with controlled PUFs, as proposed by Gassend et al. [51]. The authors were also the first to propose and manufacture a strong PUF for a silicon medium. Their design is based on a reconfigurable RO.

criterion [176]. A secure sketch is used for this purpose, and the server stores accompanying helper data. Moreover, as an optional reinforcement against machine learning attacks, the authors suggest using the PUF in feedback mode. The error-corrected PUF output is then fed back as a PUF input, and this for multiple rounds.

Incomplete Specification (#1)

Although the given strong PUF design provides a single response bit r only, the authors do not suggest an expansion method to obtain a long response $\mathbf{x}$. The relatively small-sized digest $\mathbf{c}'$ of the first cryptographic hash function contains, unlike a PRNG-generated stream, for example, insufficient bits for a list of challenges for which the responses could hypothetically be concatenated. Moreover, if the PUF is used in feedback mode, an expansion procedure would have to be performed multiple times.

Inefficiencies (#6)

In terms of efficiency, authentication via controlled PUFs seems to be inferior to reference protocol II-A, or stated otherwise, inferior to the authors' own concept of a *physically obfuscated key* [49]. First, server storage requirements scale linearly with g, i.e., the number of authenticati+ons, in contrast to constant-size. For the same reason, the enrollment phase is expected to take more time as well. Second, the proposal seems to be overprotected against machine learning, i.e., the feedback mechanism could be omitted. Third, helper data is transmitted with every protocol run and cannot efficiently be stored by a token, if desired.

Nevertheless, controlled PUFs might be superior in terms of physical security. As has been pointed out by the authors, PUF-based key generation relies on the secrecy of a relatively limited number of bits, i.e., the key $\mathbf{k}$ and the associated PUF response $\mathbf{x}$. With controlled PUFs, there is no single condensed secret that, if revealed, undermines the security of the whole system. For impersonation purposes, an attacker would have to extract sufficient CRPs via physical means in order to enable a subsequent machine learning step. Depending on the modeling resistance of the strong PUF, this corresponds to an increased number of executions of a certain physical attack, but it is not to be confused with stopping physical attacks. Becker and Kumar [14] defeated controlled Arbiter PUFs with a hybrid physical–learning attack, for example. Moreover, given a reasonable number of protocol runs g, the helper data $\mathbf{h}$ is too large to be stored in the embedded OTP NVM of a token and is hence trivial to manipulate during transfer, if this would enhance a certain physical attack.

5.3.3 Bolotnyy and Robins (2008)

The protocol of Bolotnyy and Robins [21] is specified in Figure 5.9; a block diagram of the token hardware is shown in Figure 5.6(c). The authors claim that tokens are authenticated in a private manner, under the assumption of a passive, eavesdropping attacker. Furthermore, the strong PUF is assumed to be resistant to machine learning. A *temporal majority vote* (TMV) is suggested as an error-correction mechanism, during both the enrollment and the reconstruction phases, as previously discussed in Sections 4.1.3 and 4.5.1 respectively.

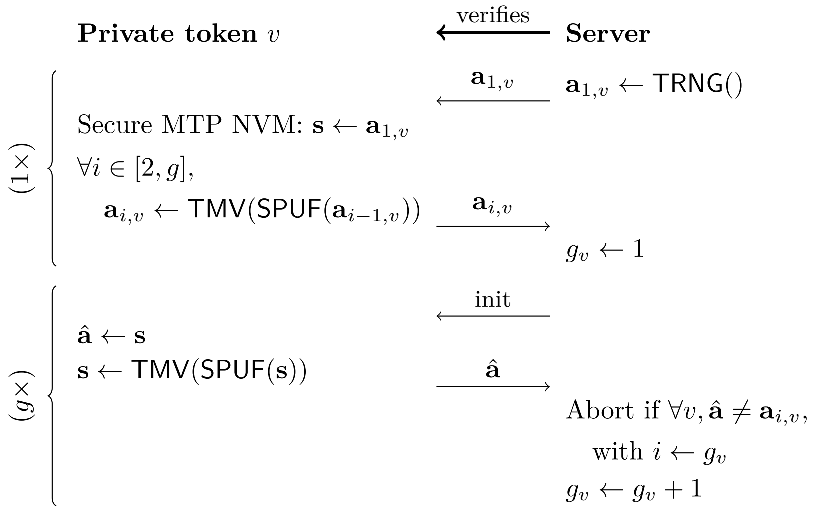

Private token v
verifies
Server
$\mathbf{a}_{1,v}$
$\mathbf{a}_{1,v} \leftarrow \mathsf{TRNG}()$
Secure MTP NVM: $\mathbf{s} \leftarrow \mathbf{a}_{1,v}$
$\forall i \in [2, g]$,
$\mathbf{a}_{i,v} \leftarrow \mathsf{TMV}(\mathsf{SPUF}(\mathbf{a}_{i-1,v}))$
$\mathbf{a}_{i,v}$
$g_v \leftarrow 1$
$(1\times)$
init
$\hat{\mathbf{a}} \leftarrow \mathbf{s}$
$\mathbf{s} \leftarrow \mathsf{TMV}(\mathsf{SPUF}(\mathbf{s}))$
$\hat{\mathbf{a}}$
Abort if $\forall v, \hat{\mathbf{a}} \neq \mathbf{a}_{i,v}$,
with $i \leftarrow g_v$
$g_v \leftarrow g_v + 1$
$(g\times)$

Figure 5.9: The authentication protocol of Bolotnyy and Robins [21].

Incomplete Specification (#1)

The authors do not clearly specify a strong PUF that can be used to instantiate the protocol, although the Arbiter PUF [100] is cited as part of the paper's introduction. Furthermore, the authors do not suggest a method for expanding a single-bit response r into a long string $\mathbf{a}$.

Physically Secure NVM Undermines PUF Benefit (#2)

The need for physically secure NVM undermines the main benefit of a strong PUF, given that the former could hypothetically be used to craft a noiseless, device-unique function that is harder to learn than the latter. An attacker who obtains read-access to the state $\mathbf{s}$ of a token via physical means would be able to impersonate it during the next protocol run. Likewise, there is a privacy breach as well. An attacker who obtains write-access to the state $\mathbf{s}$ of a token via physical means can undermine the system security completely.

Non-Functional due to PUF Noisiness (#3)

The noisiness of the PUF has not been taken into account properly, which makes the protocol non-functional. The authors wrongly assume that each response bit exhibits a small, identical error rate, e.g., constantly 2%. This would allow to make the overall failure rate of the protocol negligibly small, even with a relatively limited number of votes. However, in practice, there is a continuous spectrum from highly stable to highly unstable response bits, as previously elaborated in Section 3.1.5. TMV is effective and ineffective for relatively stable and relatively noisy response bits respectively, as previously discussed in Section 4.5.1. Moreover, if a token's environment in the field differs from during the enrollment, TMV is never sufficient by itself.

Machine Learning Attacks (#4)

It is unrealistic to assume that strong PUFs are resistant to machine learning attacks, which further limits the practical value of the protocol. There is hence no secure instantiation of the protocol, due to the lack of an appropriate strong PUF.

Overly Restricted Attacker Capabilities, Denial-of-Service (#8)

As has been acknowledged in the article, the assumption of an eavesdropping attacker mitigates a DoS threat. Note that an active attacker only needs to query a token in order to cause a desynchronization with the server. However, we argue that realistic threats should not be mitigated with unrealistic assumptions. The protocol is proposed for RFID systems in particular, i.e., an excellent environment for mounting an active attack. Sending an authentication request to a token is sufficient for an active attacker to cause desynchronization of the state $\mathbf{s}$. Either blocking or modifying a token's response $\mathbf{a}$ has a similar effect.

5.3.4 Öztürk et al. (2008)

The protocol of Öztürk et al. [130] is specified in Figure 5.10; a block diagram of the token hardware is shown in Figure 5.6(d). The proposed system relies on two strong PUFs for which a predictive model is constructed during the enrollment phase. The outer strong PUF provides a challenge-response authentication mechanism, similar to the basic protocol that has been discussed in Section 5.3.1.

However, in order to prevent machine learning attacks, its challenge is obfuscated by a secret internal state $\mathbf{s}$. During each protocol run, state $\mathbf{s}$ is updated by the inner strong PUF.

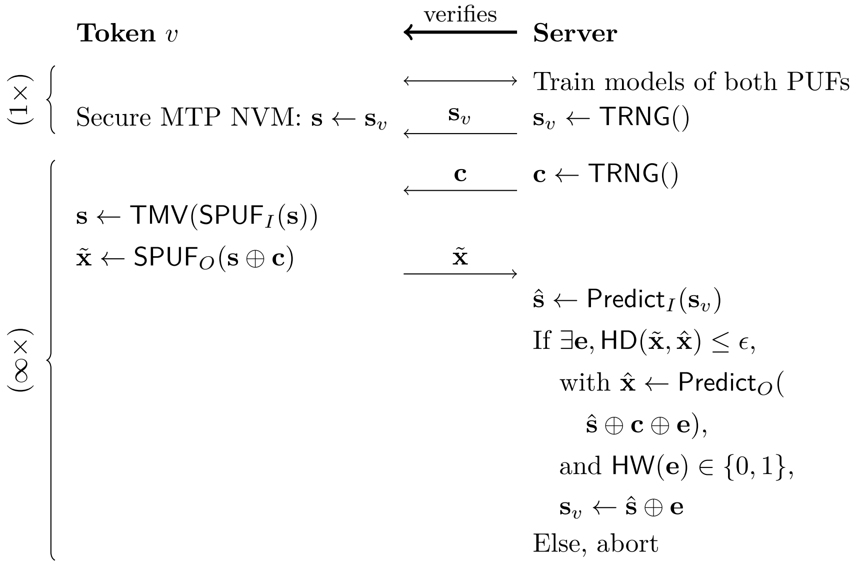

Token v
verifies
Server
$(1\times)$
Train models of both PUFs
Secure MTP NVM: $\mathbf{s} \leftarrow \mathbf{s}_v$
$\mathbf{s}_v$
$\mathbf{s}_v \leftarrow \mathsf{TRNG}()$
$(\infty\times)$
$\mathbf{c}$
$\mathbf{c} \leftarrow \mathsf{TRNG}()$
$\mathbf{s} \leftarrow \mathsf{TMV}(\mathsf{SPUF}_I(\mathbf{s}))$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}_O(\mathbf{s} \oplus \mathbf{c})$
$\tilde{\mathbf{x}}$
$\hat{\mathbf{s}} \leftarrow \mathsf{Predict}_I(\mathbf{s}_v)$
If $\exists \mathbf{e}, \mathsf{HD}(\tilde{\mathbf{x}}, \hat{\mathbf{x}}) \leq \epsilon$,
with $\hat{\mathbf{x}} \leftarrow \mathsf{Predict}_O($
$\hat{\mathbf{s}} \oplus \mathbf{c} \oplus \mathbf{e})$,
and $\mathsf{HW}(\mathbf{e}) \in \{0, 1\}$,
$\mathbf{s}_v \leftarrow \hat{\mathbf{s}} \oplus \mathbf{e}$
Else, abort

Figure 5.10: The authentication protocol of Öztürk et al. [130]. Both the inner and the outer strong PUFs are instantiated with an Arbiter PUF. The challenge $\mathbf{s}^{(t)} \in \{0,1\}^\lambda$ of the inner strong PUF is fed into a permutation-based PRNG such that the single-bit responses r to an expanded list of challenges can be concatenated into long string $\mathbf{s}^{(t+1)}$. To be precise, $\forall i \in [1, \lambda], s_i^{(t+1)} \leftarrow \mathsf{SPUF}_I(\pi^{i-1}(\mathbf{s}^{(t)}))$, where π is a hardcoded permutation.

A temporal majority vote, as previously discussed in Section 4.5.1, is suggested as an error-correction mechanism for the inner strong PUF. It is therefore assumed that desynchronization of the state $\mathbf{s}$ seldom occurs, and if so, that the Hamming distance is exactly one bit. A recovery procedure at the server side iteratively flips one bit of the state $\mathbf{s}$ until the authentication succeeds.

Incomplete Specification (#1)

The authors do not propose a method to expand the single-bit response of the outer strong PUF into a long string. Furthermore, the permutation π that expands the single-bit response of the inner strong PUF should be chosen carefully, as elaborated later-on, but no constraints are imposed.

Physically Secure NVM Undermines PUF Benefit (#2)

The need for physically secure NVM undermines the main benefit of the outer strong PUF. Observe that the former could hypothetically be used to craft a noiseless, device-unique function that is harder to learn than the latter, at least for an attacker in the field, while being trivial to model for the server during the enrollment phase. The motivation for the inner strong PUF is questionable as well. A well-designed PRNG could update the state equally well, i.e., $\mathbf{s}^{(t+1)} \leftarrow \mathsf{PRNG}(\mathbf{s}^{(t)})$. Although the specification of its algorithm would have to be public, the value of the initial state already provides a device-unique secret, the temporal majority vote could be discarded, and it would allow for a feedback loop with improved statistical properties.

PUF Noisiness Underestimated (#3)

The synchronization effort attributed to the noisiness of the inner strong PUF is somewhat underestimated, especially in light of environmental perturbations. As previously pointed out for the protocol of Bolotnyy and Robins [21] in Section 5.3.3, a temporal majority vote is ineffective for relatively noisy response bits. Errors of the state $\mathbf{s}$ comprehend a frequent rather than a seldom event, and are not necessarily limited to a single bit. Although the exhaustive recovery procedure at the server side could be generalized to account for multiple erroneous bits, the computational effort rapidly increases to infeasible levels, as has been discussed in Section 4.4.1.

Denial-of-Service (#8)

There is a simple DoS attack. An attacker only needs to send a challenge $\mathbf{c}$ to a token in order to cause a desynchronization of the state $\mathbf{s}$. Either blocking or modifying the response $\tilde{\mathbf{x}}$ has the same effect.

PRNG Weaknesses (#8)

A first issue with the permutation-based PRNG is that all produced challenges have the same Hamming weight. Alternative lightweight solutions such as an LFSR approximate a truly random sequence more effectively. A second issue is that certain permutations result in a security hazard, but no constraints are imposed. Consider for example the order $q \geq 1$ of a λ-bit permutation, which is the minimum number of repeated iterations that produces an identity function, i.e., $\forall \mathbf{s} \in \{0,1\}^{\lambda}, \pi^{i}(\mathbf{s}) = \pi^{i+q}(\mathbf{s})$. A low order causes the state $\mathbf{s}$ to consist of reappearing substrings, which opposes its presumed uniformity. An exact formula [177] for the probability that a randomly chosen permutation has order q is given in (5.1). Figure 5.11 shows that an order $q < \lambda$ occurs relatively frequently.

$$\mathsf{f}(q) = \sum_{d_1 | q} \mu\left(\frac{q}{d_1}\right) \left[x^n\right] \sum_{i=1}^{\infty} \frac{\left(\sum_{d_2|d_1} \frac{x^{d_2}}{d_2}\right)^i}{i!}, \tag{5.1}$$

with μ the Möbius function and $\left[x^n\right]$ the coefficient of x^n.

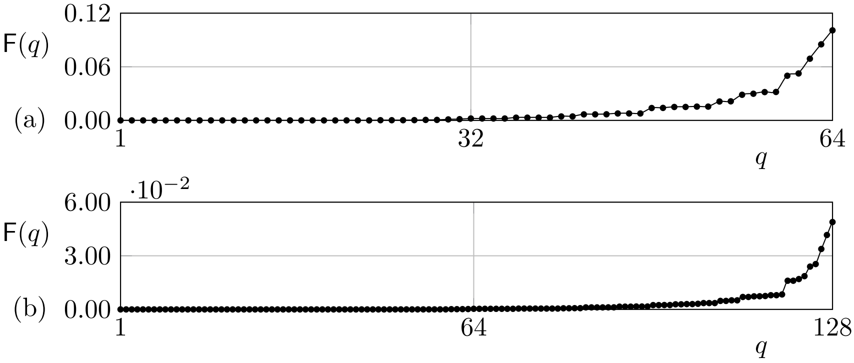


Figure 5.11: The CDF of the order q of an λ-bit permutation, with (a) $\lambda = 64$ and (b) $\lambda = 128$. Solid curves correspond to an accumulation of the exact formula in (5.1). Each dot corresponds to a Monte Carlo experiment of size 10^5 that verifies the correctness of this formula.

5.3.5 Hammouri et al. (2008)

The protocol of Hammouri et al. [54] is specified in Figure 5.12; a block diagram of the token hardware is shown in Figure 5.6(e). The proposed system relies on two strong PUF for which a predictive model is constructed during the enrollment phase. The outer strong PUF provides a challenge-response authentication mechanism similar to the basic protocol that was discussed in Section 5.3.1. However, in order to counter machine learning attacks, its challenge is obfuscated by the response $\tilde{\mathbf{x}}_I$ of the inner strong PUF. Two strong PUFs cannot simply be cascaded without inducing an amplification of the noise. In order to make this effect bearable, the authors introduce an additional XOR network Lin. As a result, two responses $r_1 \leftarrow \mathsf{SPUF}_O(\mathsf{Lin}(\mathbf{c}_1) \oplus \mathbf{c}_O)$ and $r_2 \leftarrow \mathsf{SPUF}_O(\mathsf{Lin}(\mathbf{c}_2) \oplus \mathbf{c}_O)$ would tend to be equal if the Hamming distance $\mathsf{HD}(\mathbf{c}_1, \mathbf{c}_2)$ between both challenges is small, i.e., the same transfer characteristics as for an RO Sum PUF are obtained.

Machine Learning Attacks are not Excluded (#4)

It is highly unlikely that the overall system is sufficiently resistant to machine learning attacks. Observe that there is rather delicate balance between the security of the overall system and the security of both component PUFs. Individually, the two strong PUFs should be easy-to-model, as this enables the enrollment, but in cascaded form, the construction of a predictive model should suddenly be infeasible. To date, no proof-of-concept attack has been published though. We conjecture that the obtained result is somewhat similar to the use of XOR PUFs: the overall system is harder to learn than each of its components, but it is still feasible given that noise imposes an upper bound on the overall complexity.

Overcomplicated Enrollment with Yield Issues (#7)

Reading out the responses $\tilde{\mathbf{x}}_I$ of the inner strong PUF via a one-time interface would have made the enrollment easy. This would have allowed to model both strong PUFs independently from each other. Instead, the authors devised a rather complicated procedure where the responses $\tilde{\mathbf{x}}_I$ of the inner strong PUF are recovered through the responses $\tilde{\mathbf{x}}_O$ of the outer strong PUF. The latter PUF is therefore required to operate in a highly linear manner, which is not expected to benefit the modeling resistance of the overall system. Note that a one-time interface is still required in order to seed the LFSR with fixed points.

Moreover, the inner strong PUF is instantiated with a rather far-fetched design. The effectiveness of its custom-designed modeling procedure hinges on assumptions about the layout of its circuit and has not been tested on either simulated or experimental data. One could, for example, have opted for a more generic solution $\tilde{r} \leftarrow \mathsf{SPUF}_I(\mathsf{LFSR}(\mathbf{c}_I))$, where the design of the inner strong PUF and its learning algorithm can be chosen arbitrarily and where fixed points of the LFSR are blocked. Although we would recommend using a one-time interface for reading-out the responses $\tilde{\mathbf{x}}_I$ of the inner strong PUF during the enrollment, more in line with the original proposal, one could opt to disable the state updates of the LFSR so that each challenge $\mathbf{c}_I$ is evaluated λ times and $\tilde{\mathbf{x}}_I \in \{\mathbf{0}, \mathbf{1}\}$, apart from potential noisiness.

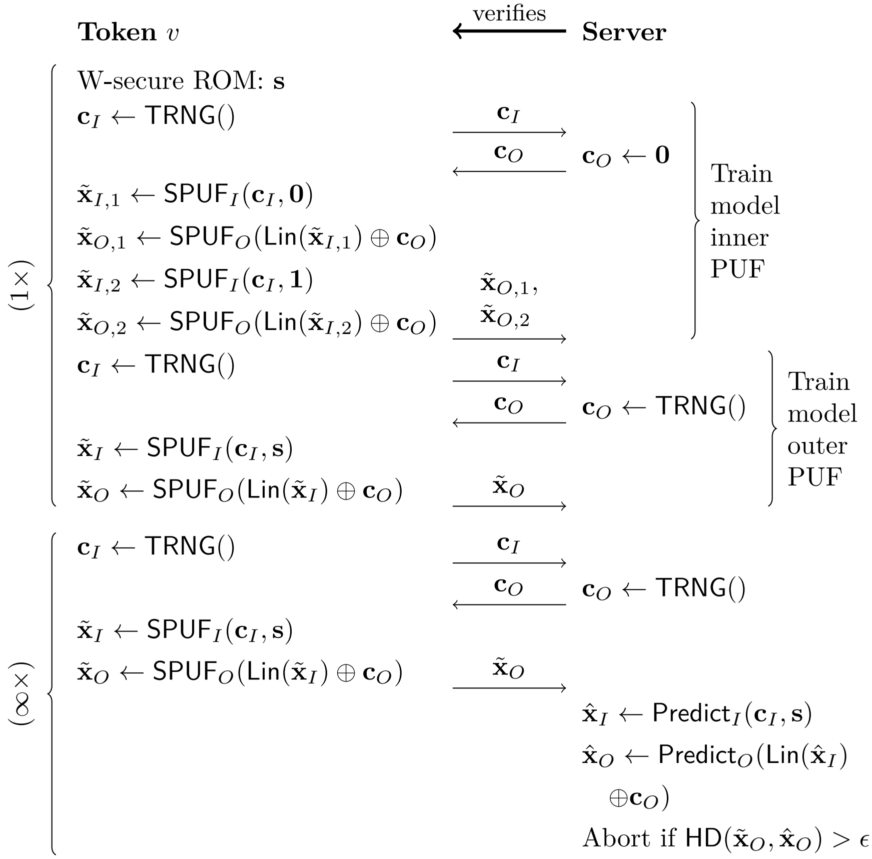

Token v
verifies
Server
$(1\times)$
W-secure ROM: $\mathbf{s}$
$\mathbf{c}_I \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_I$
$\mathbf{c}_O$
$\mathbf{c}_O \leftarrow \mathbf{0}$
$\tilde{\mathbf{x}}_{I,1} \leftarrow \mathsf{SPUF}_I(\mathbf{c}_I, \mathbf{0})$
$\tilde{\mathbf{x}}_{O,1} \leftarrow \mathsf{SPUF}_O(\mathsf{Lin}(\tilde{\mathbf{x}}_{I,1}) \oplus \mathbf{c}_O)$
$\tilde{\mathbf{x}}_{I,2} \leftarrow \mathsf{SPUF}_I(\mathbf{c}_I, \mathbf{1})$
$\tilde{\mathbf{x}}_{O,2} \leftarrow \mathsf{SPUF}_O(\mathsf{Lin}(\tilde{\mathbf{x}}_{I,2}) \oplus \mathbf{c}_O)$
$\tilde{\mathbf{x}}_{O,1}$, $\tilde{\mathbf{x}}_{O,2}$
Train model inner PUF
$\mathbf{c}_I \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_I$
$\mathbf{c}_O$
$\mathbf{c}_O \leftarrow \mathsf{TRNG}()$
$\tilde{\mathbf{x}}_I \leftarrow \mathsf{SPUF}_I(\mathbf{c}_I, \mathbf{s})$
$\tilde{\mathbf{x}}_O \leftarrow \mathsf{SPUF}_O(\mathsf{Lin}(\tilde{\mathbf{x}}_I) \oplus \mathbf{c}_O)$
$\tilde{\mathbf{x}}_O$
Train model outer PUF
$(\infty\times)$
$\mathbf{c}_I \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_I$
$\mathbf{c}_O$
$\mathbf{c}_O \leftarrow \mathsf{TRNG}()$
$\tilde{\mathbf{x}}_I \leftarrow \mathsf{SPUF}_I(\mathbf{c}_I, \mathbf{s})$
$\tilde{\mathbf{x}}_O \leftarrow \mathsf{SPUF}_O(\mathsf{Lin}(\tilde{\mathbf{x}}_I) \oplus \mathbf{c}_O)$
$\tilde{\mathbf{x}}_O$
$\hat{\mathbf{x}}_I \leftarrow \mathsf{Predict}_I(\mathbf{c}_I, \mathbf{s})$
$\hat{\mathbf{x}}_O \leftarrow \mathsf{Predict}_O(\mathsf{Lin}(\hat{\mathbf{x}}_I) \oplus \mathbf{c}_O)$
Abort if $\mathsf{HD}(\tilde{\mathbf{x}}_O, \hat{\mathbf{x}}_O) > \epsilon$

Figure 5.12: The authentication protocol of Hammouri et al. [54]. The outer strong PUF is instantiated with an m-stage Arbiter PUF. Its single-bit response r is expanded into a long string $\tilde{\mathbf{x}}_O \in \{0,1\}^\lambda$ by running the protocol λ times, i.e., λ challenge pairs $(\mathbf{c}_I, \mathbf{c}_O)$ are exchanged. The non-linearities that are attributed to the switching behavior are eliminated through a challenge transformation $\mathsf{Lin}(\mathbf{c}) = (c_1 \oplus c_2 \; c_2 \oplus c_3 \; \cdots \; c_{m-1} \oplus c_m \; c_m)$. The inner strong PUF is instantiated with a custom-tailored design that interleaves two Arbiter PUFs into a single circuit. Its single-bit responses r to a list of challenges $\left(\mathbf{c}_I, \mathbf{s}^{(i)}\right)$, where $i \in [1, \lambda]$, are concatenated in order to obtain a long string $\tilde{\mathbf{x}}_I \in \{0,1\}^\lambda$. States $\mathbf{s}^{(i)}$ are provided by an LFSR that starts cycling from a hardcoded seed $\mathbf{s}^{(1)}$. During the enrollment, seed value $\mathbf{s}^{(1)}$ is set to fixed points $\mathbf{0}$ and $\mathbf{1}$ instead so that responses $\tilde{\mathbf{x}}_I \in \{\mathbf{0}, \mathbf{1}\}$, apart from potential noisiness. Both values of $\tilde{\mathbf{x}}_I$ can be distinguished, given that an Arbiter PUF is likely to produce opposite responses r for challenges $\mathsf{Lin}(\mathbf{0}) = \mathbf{0}$ and $\mathsf{Lin}(\mathbf{1}) = (0\,0 \cdots 0\,1)$.

Finally, the authors overlook that their enrollment procedure does not always succeed in practice. It is wrongly assumed that the error rate of response bits r can be described through a single constant p_{error}. As is clear from the heterogeneous variability–noise model in Section 3.1.5, the responses r_i to different challenges $\mathbf{c}_i$ exhibit different error rates $p_{\text{error},i}$. If for the Arbiter PUF of a given device, variability aggregate $\omega \approx 0$ for challenges $\mathbf{c} \in \{\mathbf{0}, (0\,0 \cdots 0\,1)\}$, then the responses $\tilde{\mathbf{x}}_I$ of the inner strong PUF cannot be recovered.

5.3.6 Kulseng et al. (2010)

The protocol of Kulseng et al. [94] is specified in Figure 5.13; a block diagram of the token hardware is shown in Figure 5.6(f). In order to preserve the privacy of a token, its public identifier $\mathbf{i}$ is updated with every protocol run. An attacker can hence only track a token in between protocol runs. Given that an attacker may desynchronize the state $(\mathbf{s}, \mathbf{i})$ by blocking the last message $(\mathbf{a}_1, \mathbf{a}_2)$, recovery logic is foreseen to mitigate this DoS threat.

Incomplete Specification (#1)

Contradictive information is provided regarding the update of identifier $\mathbf{i}$: either the current state $\mathbf{s}$ or its successor $\tilde{\mathbf{x}}$ may be used. Although both possibilities result in an equally insecure system, we align our interpretation with prior cryptanalysis of Kardaş et al. [78] and opt for the current state $\mathbf{s}$. Moreover, the authors of the proposed protocol do not suggest a method to expand the single-bit response r of the strong PUF into a long string $\tilde{\mathbf{x}}$.

Physically Secure NVM Undermines PUF Benefit (#2)

The need for physically secure NVM undermines the benefit of the strong PUF, given that the former could hypothetically be used to craft a noiseless, device-unique function that is harder to learn than the latter.

Non-Functional due to PUF Noisiness (#3)

The noisiness of the PUF response $\tilde{\mathbf{x}}$ has not been taken into account, which makes the protocol non-functional. We make abstraction of this issue in the remainder of our analysis.

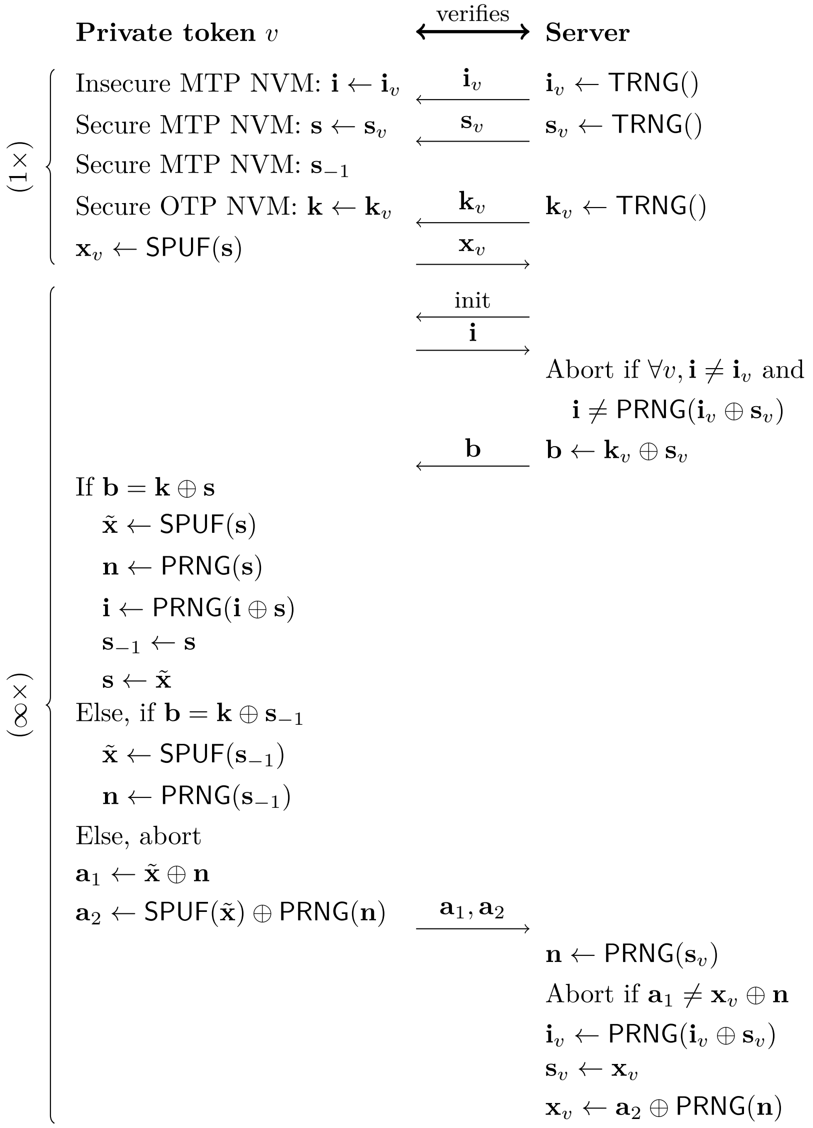

Private token v
$\xleftrightarrow{\text{verifies}}$
Server
$(1\times)$
Insecure MTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$
$\xleftarrow{\mathbf{i}_v}$
$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
Secure MTP NVM: $\mathbf{s} \leftarrow \mathbf{s}_v$
$\xleftarrow{\mathbf{s}_v}$
$\mathbf{s}_v \leftarrow \mathsf{TRNG}()$
Secure MTP NVM: $\mathbf{s}_{-1}$
Secure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_v$
$\xleftarrow{\mathbf{k}_v}$
$\mathbf{k}_v \leftarrow \mathsf{TRNG}()$
$\mathbf{x}_v \leftarrow \mathsf{SPUF}(\mathbf{s})$
$\xrightarrow{\mathbf{x}_v}$
$(\infty\times)$
$\xleftarrow{\text{init}}$
$\xrightarrow{\mathbf{i}}$
Abort if $\forall v, \mathbf{i} \neq \mathbf{i}_v$ and
$\mathbf{i} \neq \mathsf{PRNG}(\mathbf{i}_v \oplus \mathbf{s}_v)$
$\xleftarrow{\mathbf{b}}$
$\mathbf{b} \leftarrow \mathbf{k}_v \oplus \mathbf{s}_v$
If $\mathbf{b} = \mathbf{k} \oplus \mathbf{s}$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{s})$
$\mathbf{n} \leftarrow \mathsf{PRNG}(\mathbf{s})$
$\mathbf{i} \leftarrow \mathsf{PRNG}(\mathbf{i} \oplus \mathbf{s})$
$\mathbf{s}_{-1} \leftarrow \mathbf{s}$
$\mathbf{s} \leftarrow \tilde{\mathbf{x}}$
Else, if $\mathbf{b} = \mathbf{k} \oplus \mathbf{s}_{-1}$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{s}_{-1})$
$\mathbf{n} \leftarrow \mathsf{PRNG}(\mathbf{s}_{-1})$
Else, abort
$\mathbf{a}_1 \leftarrow \tilde{\mathbf{x}} \oplus \mathbf{n}$
$\mathbf{a}_2 \leftarrow \mathsf{SPUF}(\tilde{\mathbf{x}}) \oplus \mathsf{PRNG}(\mathbf{n})$
$\xrightarrow{\mathbf{a}_1, \mathbf{a}_2}$
$\mathbf{n} \leftarrow \mathsf{PRNG}(\mathbf{s}_v)$
Abort if $\mathbf{a}_1 \neq \mathbf{x}_v \oplus \mathbf{n}$
$\mathbf{i}_v \leftarrow \mathsf{PRNG}(\mathbf{i}_v \oplus \mathbf{s}_v)$
$\mathbf{s}_v \leftarrow \mathbf{x}_v$
$\mathbf{x}_v \leftarrow \mathbf{a}_2 \oplus \mathsf{PRNG}(\mathbf{n})$

Figure 5.13: The authentication protocol of Kulseng et al. [94]. The strong PUF is instantiated with an Arbiter PUF; the PRNG is instantiated with an LFSR. We assume that the current state $\mathbf{s}$ and hence not its successor $\tilde{\mathbf{x}}$ is used to update the identifer $\mathbf{i}$, in light of the somewhat ambiguous specification.

Server Impersonation (#8)

The proposed desynchronization recovery logic allows an attacker to impersonate the server. Replaying the message $\mathbf{b}$ of the last, genuine protocol run suffices.

Denial-of-Service (#8)

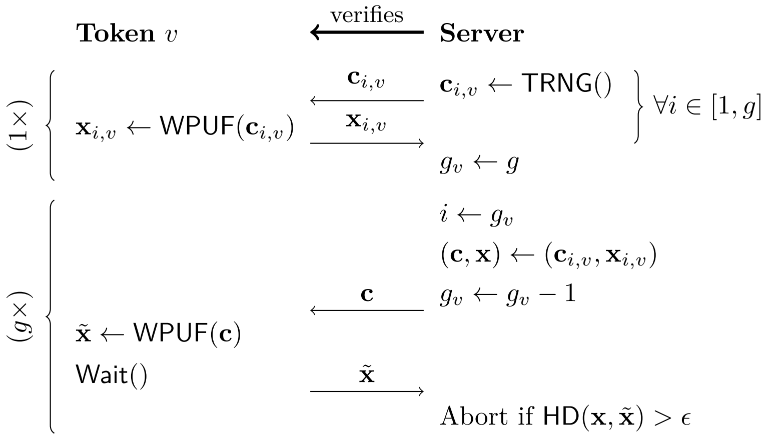
As pointed out by Kardaş et al. [78], there is a simple DoS attack. An attacker can interfere with a genuine protocol run, i.e., modify the output $\mathbf{a}_2$ of a token to an arbitrary value, and hence cause desynchronization with the server.

Token/Server Impersonation and Privacy Breach (#8)

Kardaş et al. [78] previously described an attack that results in full system disclosure. The initial λ-bit state of an LFSR can easily be recovered from an arbitrary λ-bit output sequence. Given two consecutive identifiers, i.e., $\mathbf{i}^{(1)}$ and $\mathbf{i}^{(2)} = \mathsf{LFSR}(\mathbf{i}^{(1)} \oplus \mathbf{s}^{(1)})$, an attacker can hence recover the state $\mathbf{s}^{(1)}$ as well as all other secret variables. Our analysis reveals the existence of an alternative attack, which exploits the linearity of the LFSR. Given again two consecutive identifiers, i.e., $\mathbf{i}^{(1)}$ and $\mathbf{i}^{(2)}$, an attacker can recover $\mathbf{n}^{(1)} = \mathbf{i}^{(2)} \oplus \mathsf{LFSR}(\mathbf{i}^{(1)})$ as well as all other secret variables.

5.3.7 Super High Information Content PUFs (2010)

Authentication with so-called *super high information content* (SHIC) PUFs, as proposed by Rührmair et al. [143], is specified in Figure 5.14. A block diagram of the token hardware is shown in Figure 5.6(g). Both the number of cells and the read-out latency of the memory-based weak PUF are exceptionally high by design. Therefore, the time required for a hypothetical read-out of the entire cell contents exceeds the lifetime of a token, or alternatively, it exceeds the maximal access time of an attacker. The latter figures can vary from days to years, depending on the use case. Presumably under the assumption of uniformly distributed cell contents, the protocol is claimed to be secure against a computationally unrestricted adversary.



Token v
verifies
Server
$(1\times)$
$\mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_{i,v}$
$\mathbf{x}_{i,v} \leftarrow \mathsf{WPUF}(\mathbf{c}_{i,v})$
$\mathbf{x}_{i,v}$
$\forall i \in [1, g]$
$g_v \leftarrow g$
$(g\times)$
$i \leftarrow g_v$
$(\mathbf{c}, \mathbf{x}) \leftarrow (\mathbf{c}_{i,v}, \mathbf{x}_{i,v})$
$\mathbf{c}$
$g_v \leftarrow g_v - 1$
$\tilde{\mathbf{x}} \leftarrow \mathsf{WPUF}(\mathbf{c})$
$\mathsf{Wait}()$
$\tilde{\mathbf{x}}$
Abort if $\mathsf{HD}(\mathbf{x}, \tilde{\mathbf{x}}) > \epsilon$

Figure 5.14: Authentication with SHIC PUFs, as proposed by Rührmair et al. [143]. This type of weak PUF consists of a large memory that is required to operate slowly. Preferentially, the high latency is inherent to the memory itself, although alternatively, a dedicated timer could induce a constant delay. Suggested values for the data size and the read-out throughput are 10 Gbit and $100\,\mathrm{bit\,s^{-1}}$ respectively. Given uninterrupted access to a token, approximately three years would hence be needed to read-out the entire cell contents. Emerging high-density technologies such as *resistive random-access memory* (RRAM) are preferred to SRAM.

Inefficiencies (#6)

Although tokens require the implementation of a single building block only, the proposed protocol is relatively inefficient. Using modern 10 nm features, the authors estimate that the size of the RRAM is in the mm^2 to cm^2 range, depending on the use case. Despite using an older 65 nm technology, Bhargava and Mai [19] actually manufactured a full-fledged PUF-based key generator as well as a cryptographic algorithm with a total size of only $119\,\mathrm{\mu m}^2$. Moreover, the response time for performing an authentication would be in the µs range only. Compared to an SHIC PUF with a $100\,\mathrm{bit\,s^{-1}}$ throughput, the stringent latency requirements of modern communication standards are hence easier to meet. Yu et al. [188] recently eliminated the inefficiencies of the SHIC protocol through a subtle modification, as will be discussed in Section 5.3.21.

Random Guessing Attacks (#8)

Given that the SHIC protocol lacks computations, we agree that there is no benefit for an attacker in being computationally unrestricted. However, we argue that the protocol does not scale to a comfortable security level. A 10 Gbit memory allows for 2^{28} challenges $\mathbf{c}$ that have a 28-bit response $\mathbf{x}$ each. An attacker can hence correctly guess the response $\mathbf{x}$ to any unseen challenge $\mathbf{c}$ with a non-negligible probability of 2^{-28}. Likewise, an attacker who previously queried a token with a single, randomly selected challenge $\mathbf{c}$, can replay the collected response $\mathbf{x}$ in the next server-initiated protocol run with a probability of 2^{-28}. Actually, the security level is slightly lower than 28 bit, given that the responses of the SHIC PUF are noisy in practice. The protocol of Yu et al. [188] easily scales to a comfortable 128-bit security level.

5.3.8 Sadeghi et al. (2010)

The protocol of Sadeghi et al. [148] is specified in Figure 5.15; a block diagram of the token hardware is shown in Figure 5.6(h). The privacy of a token is preserved at any time. Physically invasive attacks that recover the NVM contents $(\mathbf{k}_1, \mathbf{h})$ of a token are assumed be *destructive*, i.e., afterwards the given token cannot circulate in the system anymore. The authors use a strong PUF and a fuzzy extractor in order to generate a cryptographic key $\mathbf{k}_2$.

Incomplete Specification (#1)

The authors do not suggest the use of a particular strong PUF. Moreover, the need to expand its usually single-bit response r into a long string $\mathbf{x}$ has not been acknowledged either.

Inefficiencies (#6)

Two inefficiencies have not been addressed. First, the initial key $\mathbf{k}_1$ and its OTP NVM can be omitted. A hardcoded challenge generator suffices for either a weak or a strong PUF to generate the key $\mathbf{k}_2$. Second, joint optimization of the fuzzy extractor and the authentication logic has not been explored, unlike reference protocol II-A.

$$\begin{array}{lcl}
\textbf{Private token } v & \xleftarrow{\text{verifies}} & \textbf{Server}
\end{array}$$

$$(1\times)\left\{\begin{array}{lcl}
\text{Insecure OTP NVM: } \mathbf{k}_1 \leftarrow \mathbf{k}_{1,j} & \xleftarrow{\mathbf{k}_{1,j}} & \mathbf{k}_{1,j} \leftarrow \mathsf{TRNG}() \\
\mathbf{x}_v \leftarrow \mathsf{SPUF}(\mathbf{k}_1) & \xrightarrow{\mathbf{x}_v} & \mathbf{k}_{2,j} \leftarrow \mathsf{Hash}(\mathbf{x}_v) \\
\text{Insecure OTP NVM: } \mathbf{h} \leftarrow \mathbf{h}_v & \xleftarrow{\mathbf{h}_v} & \mathbf{h}_v \leftarrow \mathsf{SSGen}(\mathbf{x}_v)
\end{array}\right.$$

$$(\infty\times)\left\{\begin{array}{lcl}
 & \xleftarrow{\mathbf{n}_S} & \mathbf{n}_S \leftarrow \mathsf{TRNG}() \\
\mathbf{n}_T \leftarrow \mathsf{TRNG}() & & \\
\hat{\mathbf{x}} \leftarrow \mathsf{SSRep}(\mathsf{SPUF}(\mathbf{k}_1), \mathbf{h}) & & \\
\hat{\mathbf{k}}_2 \leftarrow \mathsf{Hash}(\hat{\mathbf{x}}) & & \\
\mathbf{a} \leftarrow \mathsf{PRF}(\mathbf{n}_T, \mathbf{n}_S; \hat{\mathbf{k}}_2) & \xrightarrow{\mathbf{n}_T, \mathbf{a}} & \\
 & & \text{Abort if } \forall v, \\
 & & \quad \mathbf{a} \neq \mathsf{PRF}(\mathbf{n}_T, \mathbf{n}_S; \mathbf{k}_{2,j})
\end{array}\right.$$

Figure 5.15: The authentication protocol of Sadeghi et al. [148]. Unlike the original proposal, we represent the fuzzy extractor as an explicit part of the protocol. We argue that implicit usage does not properly reflect the related security and efficiency concerns.

Privacy without Identification (#9)

The proposal does not scale in the number of tokens, which can cause unacceptable latencies in practical use cases. To conclude a protocol run, the server needs to exhaustively evaluate a PRF member for each registered token.

5.3.9 Logically Reconfigurable PUFs (2011)

Authentication with so-called logically reconfigurable PUFs, as proposed by Katzenbeisser et al. [80], is specified in Figure 5.16. A block diagram of the token hardware is shown in Figure 5.6(i). The authors extend the basic authentication protocol of Section 5.3.1 to make tokens recyclable. Instead of replacing a token, an internal state $\mathbf{s}$ is reconfigured, if the device is handed over to a different user, for example. The goal is to reduce the amount of electronic waste and its associated disposal costs. Moreover, the authors aim to prevent an attacker from tracking a token across updates of the state $\mathbf{s}$. Observe that this corresponds to a rather limited form of privacy only; other proposals claim to prohibit tracking either completely [148] or across runs of the authentication protocol [21, 94, 73]. An attacker is allowed to have read-access to the current state $\mathbf{s}$; there is no direct write-access though.

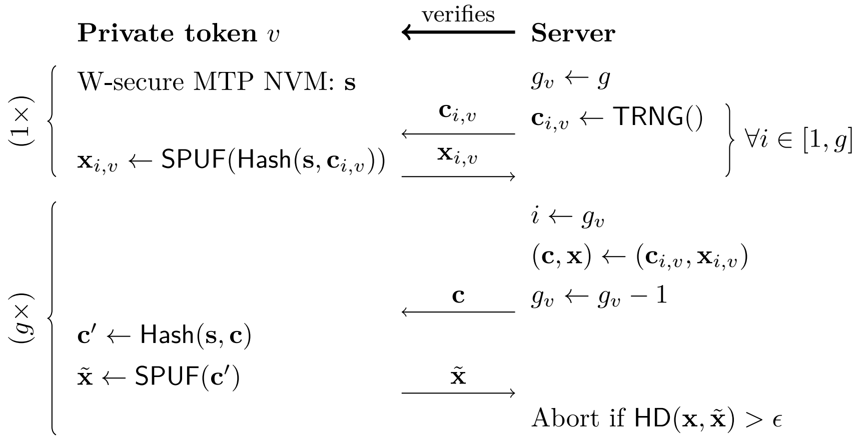

verifies
Private token v
Server
$(1\times)$
W-secure MTP NVM: $\mathbf{s}$
$g_v \leftarrow g$
$\mathbf{c}_{i,v}$
$\mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}()$
$\forall i \in [1, g]$
$\mathbf{x}_{i,v} \leftarrow \mathsf{SPUF}(\mathsf{Hash}(\mathbf{s}, \mathbf{c}_{i,v}))$
$\mathbf{x}_{i,v}$
$(g\times)$
$i \leftarrow g_v$
$(\mathbf{c}, \mathbf{x}) \leftarrow (\mathbf{c}_{i,v}, \mathbf{x}_{i,v})$
$\mathbf{c}$
$g_v \leftarrow g_v - 1$
$\mathbf{c}' \leftarrow \mathsf{Hash}(\mathbf{s}, \mathbf{c})$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c}')$
$\tilde{\mathbf{x}}$
Abort if $\mathsf{HD}(\mathbf{x}, \tilde{\mathbf{x}}) > \epsilon$

Figure 5.16: Authentication with logically reconfigurable PUFs, as proposed by Katzenbeisser et al. [80]. An attacker is given read-access to the state $\mathbf{s}$. Tracking a token is claimed to be infeasible across updates of the state, i.e., $\mathbf{s} \leftarrow \mathsf{Hash}(\mathbf{s})$; the server enrolls new CRPs after every update. The strong PUF is instantiated with an Arbiter PUF. Two methods are suggested for expanding its single-bit response r into a long string $\tilde{\mathbf{x}} \in \{0,1\}^\lambda$. First, λ identically laid-out arbiter PUFs can evaluate the challenge $\mathbf{c}'$ in parallel. Second, a single Arbiter PUF can evaluate a sequence of λ challenges, i.e., $\mathbf{c}' = \mathsf{Hash}(\mathbf{s}, \mathbf{c}, i)$, with $i \in [1, \lambda]$. Schneider and Schröder [153] proposed a variation on the update mechanism of the state, i.e., $\mathbf{s} \leftarrow \mathsf{TRNG}()$.

Incoherent Specification (#1)

According to the authors, the server does not necessarily need to collect new CRPs after an update of the state $\mathbf{s}$. The server would collect and store CRPs $(\mathbf{c}', \mathbf{x})$ of the underlying strong PUF instead, and recompute them into device-level CRPs $(\mathbf{c}, \mathbf{x})$ according to the current value of the state $\mathbf{s}$. However, this does not comply with authors' assumption that the cryptographic hash function is collision-resistant, i.e., the suggested transformation cannot be performed. Moreover, the authors do not specify a mechanism to make their somewhat privacy-preserving protocol scalable in the number of registered tokens. This issue could potentially be resolved by uniquely initializing the state $\mathbf{s}$ during the enrollment and transferring its possibly updated value to the server at the start of each protocol run.

Machine Learning Attacks (#4)

The strong PUF is assumed to be resistant to machine learning attacks. Moreover, extensively elaborated security proofs [80, 153] all rely on the assumption that its CRPs are unpredictable. In practice, however, there is no secure instantiation of the protocol due to lack of an appropriate strong PUF. The proof-of-concept implementation, for example, relies on an Arbiter PUF, for which a predictive model can be trained easily. The value of the challenge $\mathbf{c}'$ cannot be chosen arbitrarily due to the pre-image resistance of the cryptographic hash function, but this hardly counts as a defense. Most machine learning attacks in literature rely on a set of randomly chosen challenges. Somewhat after the fact, the authors suggest a crossover with the controlled PUFs of Gassend et al. [51, 50] that were previously discussed in Section 5.3.2, but neither a detailed specification nor an implementation of this resolution is provided.

Questionable Bill of Materials (#6)

The proposal aims to the make tokens recyclable so as to reduce electronic waste and its associated disposal costs. However, unlike what is suggested, access rights are not necessarily bound to a specific token, and might be flexibly managed by the server instead. For example, a token with a controlled PUF [51, 50] can be handed over for the purpose of authenticating a new user, and the server might grant new access rights accordingly. Moreover, the proposal induces the need for write-secure MTP NVM. So-called disposable but low-cost tokens are hence transformed into so-called recyclable but more expensive tokens, i.e., there is no guarantee for financial benefit.

Denial-of-Service (#8)

There is a simple denial-of-service attack. An attacker can update the state $\mathbf{s}$ of a token and hence invalidate the corresponding CRPs that are stored by the server. The authors do not describe an authentication mechanism for the authority that reconfigures the state, which would be required to stop this attack from happening.

5.3.10 Reverse Fuzzy Extractors (2012)

The original reverse fuzzy extractor protocol, as proposed by Van Herrewege et al. [171], is specified in Figure 5.17. A block diagram of the token hardware is shown in Figure 5.6(j). A lightweight alternative to authentication via PUF-based key generation is provided. To be precise, the recovery procedure SSRep of a secure sketch is executed by the resource-rich server exclusively, and hence not by the resource-constrained tokens. Recall that SSRep usually involves the error-correction of a corrupted codeword, i.e., a fairly expensive operation. The procedure SSGen, which tends to be lightweight, is implemented on each token instead. This raises the concern of repeated helper data exposure. An attacker is given access to helper data $\mathbf{h}$ that corresponds to not one but multiple noisy versions of a given response $\mathbf{x}$. The authors prove that there is no additional min-entropy loss with respect to a single exposure only.

Maes, who co-authored the original proposal, later proposed a revised version in his PhD thesis [110]. The corresponding protocol is specified in Figure 5.18. A block diagram of the token hardware is shown in Figure 5.6(k). A reversal of the authenticity checks is stated to be the main difference, i.e., the identity of the server is verified first in the revised version. There seem to be other fundamental changes though, all of which happen to improve the security, as clarified hereafter.

Inefficiencies (#6)

There are inefficiencies in the original protocol of Van Herrewege et al. [171]. Even when the server would register a single tuple $(\mathbf{c}, \mathbf{x})$ only, i.e., $g = 1$, the protocol still allows for a virtually unlimited number of authentications. The replies of a token are already guaranteed to be fresh via the use of nonce $\mathbf{n}$, i.e., the registration of multiple CRPs does not contribute to the elimination of replay attacks. The use of $g = 1$ would result in several simplifications of the protocol, as has been acknowledged by the authors. Most notably, the challenge $\mathbf{c}$ can be hardcoded and does not need to be transferred with every protocol run. Moreover, the storage requirements of the server are relaxed this way. Optionally, this would also enable the use of a weak PUF.

The authors nevertheless promote the use of $g > 1$, which supposedly offers an increased resistance to side-channel attacks. We question the effectiveness of this countermeasure. For the purpose of impersonating a token in the general case that $g > 1$, an attacker only would have to extract sufficient CRPs via physical means so that a subsequent machine learning step can be performed. Compared to $g = 1$, there could be an increased number of executions of a

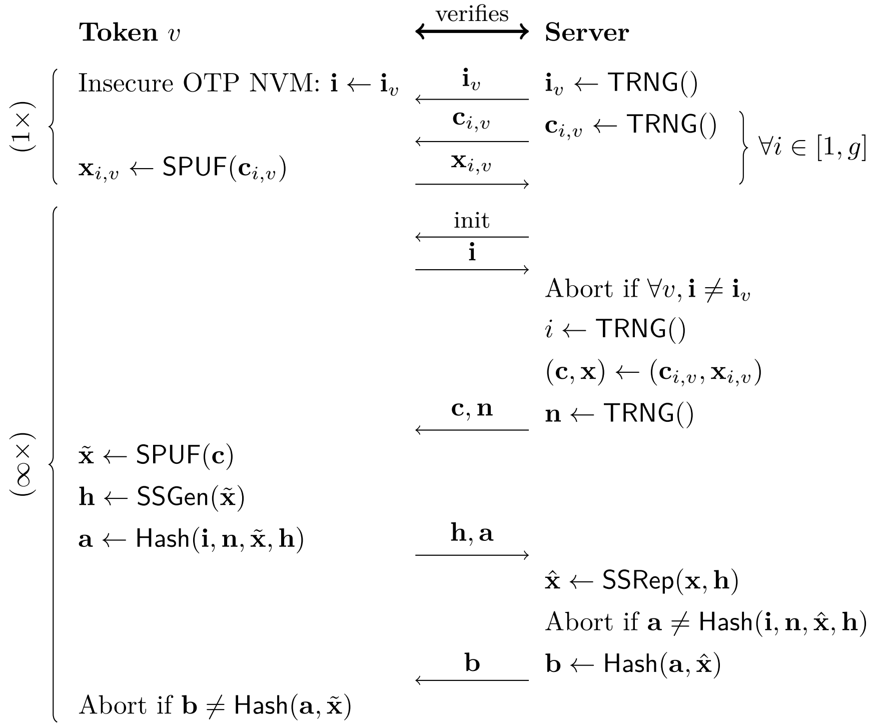

Token v
verifies
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$
$\mathbf{i}_v$
$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_{i,v}$
$\mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}()$
$\forall i \in [1, g]$
$\mathbf{x}_{i,v} \leftarrow \mathsf{SPUF}(\mathbf{c}_{i,v})$
$\mathbf{x}_{i,v}$
$(\infty\times)$
init
$\mathbf{i}$
Abort if $\forall v, \mathbf{i} \neq \mathbf{i}_v$
$i \leftarrow \mathsf{TRNG}()$
$(\mathbf{c}, \mathbf{x}) \leftarrow (\mathbf{c}_{i,v}, \mathbf{x}_{i,v})$
$\mathbf{c}, \mathbf{n}$
$\mathbf{n} \leftarrow \mathsf{TRNG}()$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c})$
$\mathbf{h} \leftarrow \mathsf{SSGen}(\tilde{\mathbf{x}})$
$\mathbf{a} \leftarrow \mathsf{Hash}(\mathbf{i}, \mathbf{n}, \tilde{\mathbf{x}}, \mathbf{h})$
$\mathbf{h}, \mathbf{a}$
$\hat{\mathbf{x}} \leftarrow \mathsf{SSRep}(\mathbf{x}, \mathbf{h})$
Abort if $\mathbf{a} \neq \mathsf{Hash}(\mathbf{i}, \mathbf{n}, \hat{\mathbf{x}}, \mathbf{h})$
$\mathbf{b}$
$\mathbf{b} \leftarrow \mathsf{Hash}(\mathbf{a}, \hat{\mathbf{x}})$
Abort if $\mathbf{b} \neq \mathsf{Hash}(\mathbf{a}, \tilde{\mathbf{x}})$

Figure 5.17: The original reverse fuzzy extractor protocol, as proposed by Van Herrewege et al. [171]. For its proof-of-concept implementation, the strong PUF is instantiated with a 64-stage Arbiter PUF. The corresponding challenge $\mathbf{c}$ is first fed into an LFSR so that the 1-bit responses r to an expanded list of challenges can be concatenated into a 1785-bit string $\mathbf{x}$. The authors impose a lower bound on the noisiness of this response, so as to avoid the impersonation of a token via replay. Moreover, the secure sketch is instantiated with the syndrome construction of Dodis et al. [39]. An $[n = 255, k = 21, d = 111]$ BCH code is applied to each out of 7 partitions of the response $\mathbf{x}$.

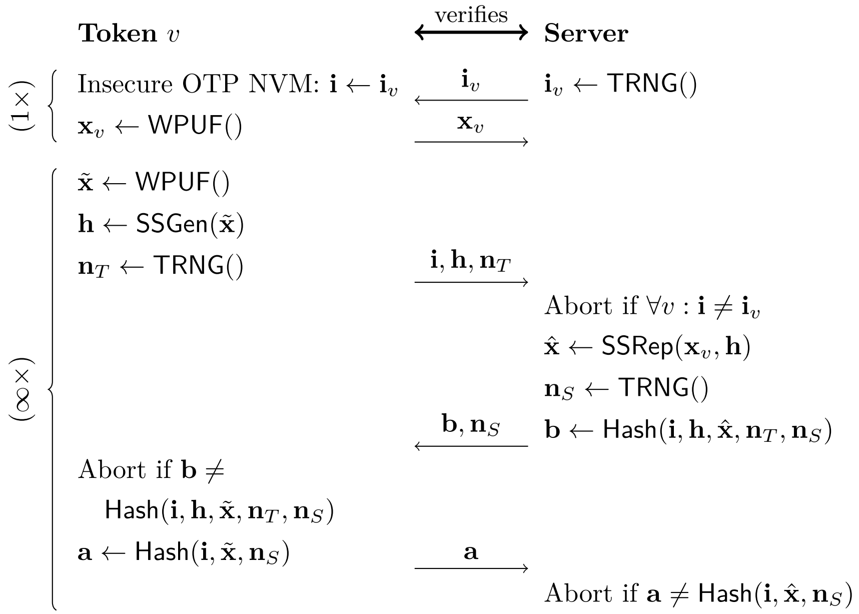

Token v
verifies
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$
$\mathbf{i}_v$
$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
$\mathbf{x}_v \leftarrow \mathsf{WPUF}()$
$\mathbf{x}_v$
$(\infty\times)$
$\tilde{\mathbf{x}} \leftarrow \mathsf{WPUF}()$
$\mathbf{h} \leftarrow \mathsf{SSGen}(\tilde{\mathbf{x}})$
$\mathbf{n}_T \leftarrow \mathsf{TRNG}()$
$\mathbf{i}, \mathbf{h}, \mathbf{n}_T$
Abort if $\forall v : \mathbf{i} \neq \mathbf{i}_v$
$\hat{\mathbf{x}} \leftarrow \mathsf{SSRep}(\mathbf{x}_v, \mathbf{h})$
$\mathbf{n}_S \leftarrow \mathsf{TRNG}()$
$\mathbf{b}, \mathbf{n}_S$
$\mathbf{b} \leftarrow \mathsf{Hash}(\mathbf{i}, \mathbf{h}, \hat{\mathbf{x}}, \mathbf{n}_T, \mathbf{n}_S)$
Abort if $\mathbf{b} \neq \mathsf{Hash}(\mathbf{i}, \mathbf{h}, \tilde{\mathbf{x}}, \mathbf{n}_T, \mathbf{n}_S)$
$\mathbf{a} \leftarrow \mathsf{Hash}(\mathbf{i}, \tilde{\mathbf{x}}, \mathbf{n}_S)$
$\mathbf{a}$
Abort if $\mathbf{a} \neq \mathsf{Hash}(\mathbf{i}, \hat{\mathbf{x}}, \mathbf{n}_S)$

Figure 5.18: The revised reverse fuzzy extractor protocol, as proposed by Maes [110]. The weak PUF can be instantiated with an SRAM PUF, for example.

certain physical attack, but this is not be confused with actually stopping the attack. The aforementioned hybrid physical–modeling attack of Becker and Kumar [14], for example, is not necessarily limited to controlled PUFs [51, 50]. The suggested countermeasure might hence not outweigh its missed advantages, and there could be numerous more rewarding countermeasures. Maes [110] does not discuss the foregoing matter, although he uses $g = 1$.

Still in their battle against replay attacks, Van Herrewege et al. [171] avoid the use of a TRNG at the token side and impose a lower bound on the noisiness of response $\tilde{\mathbf{x}}$ instead. The presumed reasoning is that the omission of a building block should improve the overall efficiency. In practice, however, the obtained gains are counterbalanced by losses. Given that more conventional design efforts to stabilize PUFs are obstructed, response $\tilde{\mathbf{x}}$ and helper data $\mathbf{h}$ are required to be of a larger size so as to obtain a given security level. The need to withstand environmental changes amplifies this opposition. An attacker who tries to impersonate the server via replay, might immerse the token of interest in an extremely stable environment, i.e., minimize the bit error rates. For genuine use, however, large environmental changes are to be anticipated, and system dimensions should be chosen accordingly. Maes [110] does not discuss the foregoing matter, although he uses a TRNG at the token side.

Min-Entropy Deficiencies Attributed to the Strong PUF (#8)

The proof-of-concept implementation of Van Herrewege et al. [171] suffers from a deficit in min-entropy. Without putting forward any evidence, the 64-stage Arbiter PUF is simply assumed to provide a 1785-bit uniformly distributed secret, i.e., $\mathbb{H}_\infty(X) = 1785$. However, as we have previously derived in Section 3.2, the min-entropy of an ideally manufactured, 64-stage Arbiter PUF cannot exceed 197 bits, i.e., $\mathbb{H}_\infty(X) < 197$. The min-entropy of the given strong PUF is hence overestimated with at least a factor 9.

Due to the release of helper data $\mathbf{h}$, the secure sketch induces an additional min-entropy loss, i.e., $\tilde{\mathbb{H}}_\infty(X|H) < \mathbb{H}_\infty(X)$. After updating the value of $\mathbb{H}_\infty(X)$, application of the $(n-k)$ upper bound on the min-entropy loss results in the worst-case scenario $\tilde{\mathbb{H}}_\infty(X|H) \geq 0$. Note, however, that the $(n-k)$ bound is not very tight for non-uniformly distributed inputs. At this point, it is hence hard to quantify the magnitude of the min-entropy deficit. Moreover, despite the presence of a min-entropy deficit, the execution of an accelerated brute-force search is not necessarily straightforward. The design of an algorithm that exhaustively iterates over all possible values of response $\mathbf{x}$, given helper data $\mathbf{h}$, seems far from trivial. We suggest this analysis as future work.

Refuted Reusability Claim (#8)

The proposals of Van Herrewege et al. [171] and Maes [110] both rely on the reusability of their secure sketch. Unfortunately, in Section 4.3.6, we have refuted this reusability claim.

PRNG Weaknesses (#8)

The PRNG that is used in the proof-of-concept implementation of Van Herrewege et al. [171] might enable several impersonation threats. First of all, the authors do not provide an explicit warning that fixed points of the LFSR, e.g., $\mathbf{c} = \mathbf{0}$, should be refused. In absence of this check, it would be trivial for an attacker to successfully impersonate the server a virtually unlimited number of times. A fixed point makes the PUF evaluate a list of identical challenges. Disregarding the noisiness of the corresponding response bit, the concatenated response would be either $\tilde{\mathbf{x}} = \mathbf{0}$ or $\tilde{\mathbf{x}} = \mathbf{1}$ for any given token. An attacker is hence able to predict digest $\mathbf{b}$ with a success rate that is $1/2$ the first time, and 1 from the second time onwards. The attack still works if the given response bit is slightly noisy. Helper data $\mathbf{h}$ is released, so an attacker computes $\tilde{\mathbf{x}} \in \{\mathsf{SSRep}(\mathbf{0}, \mathbf{h}), \mathsf{SSRep}(\mathbf{1}, \mathbf{h})\}$.

A second, less obvious LFSR-related threat is elaborated next. For ease of understanding, first consider the simplified case of a perfectly noiseless PUF. Although we cover the use of the syndrome construction of Dodis et al. [39] exclusively, a similar threat applies to all other sketch constructions in Table 4.4. The release of helper data $\mathbf{h} \in \{0,1\}^{n-k}$ for any given concatenated response $\mathbf{x} \in \{0,1\}^n$ provides the attacker with an underdetermined system of $n-k$ linear equations in n unknowns. Combined with the use of an LFSR, this allows an attacker to obtain helper data $\mathbf{h}$ for q gradually shifted responses $\begin{pmatrix} r_1 & r_2 & \cdots & r_n \end{pmatrix}, \begin{pmatrix} r_2 & r_3 & \cdots & r_{n+1} \end{pmatrix}, \ldots, \begin{pmatrix} r_q & r_{q+1} & \cdots & r_{n+q-1} \end{pmatrix}$. All data is subsequently collected in a single system of equations, as given in (5.2).

$$\mathbf{A}\begin{pmatrix} r_1 & r_2 & \cdots & r_{n+q-1} \end{pmatrix}^T = \begin{pmatrix} \mathbf{h}^{(1)} & \mathbf{h}^{(2)} & \cdots & \mathbf{h}^{(q)} \end{pmatrix}^T,$$

$$\text{with } \mathbf{A} = \left(\begin{array}{cccccccccc}
\multicolumn{5}{c|}{\mathbf{H}} & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \cdots & \mathbf{0}^T \\ \hline
\multicolumn{1}{c|}{\mathbf{0}^T} & \multicolumn{5}{c|}{\mathbf{H}} & \mathbf{0}^T & \mathbf{0}^T & \cdots & \mathbf{0}^T \\ \hline
\vdots & \ddots & & & & & & & \ddots & \vdots \\ \hline
\mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \cdots & \multicolumn{1}{c|}{\mathbf{0}^T} & \multicolumn{5}{c}{\mathbf{H}}
\end{array}\right). \tag{5.2}$$

Even for a relatively small number of iterations q, the system in (5.2) contains more equations than unknowns. It is not necessarily solvable though, given that dependencies among equations have not yet been considered. If the linear code happens to be cyclic, as is the case for the BCH code of the proof-of-concept implementation, it actually turns out that $\mathsf{rank}(\mathbf{A}) = n - k + q - 1$. Regardless of the number of iterations q, there are hence always k degrees of freedom. A complete proof is not provided here, but follows from the fact that elementary row operations on parity check matrix $\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_n \end{pmatrix}$ can produce a shifted version $\begin{pmatrix} \mathbf{h}_n & \mathbf{h}_1 & \cdots & \mathbf{h}_{n-1} \end{pmatrix}$.

Although the previous attack might work for non-cyclic codes exclusively, there is an alternative for cyclic codes. For large q, a divide-and-conquer machine learning attack can be performed instead, given that response $\mathbf{x}$ is partitioned in smaller sections due to to code size constraints. An Arbiter PUF can be modeled with only a few thousand CRPs, so consider, e.g., $q = 10^4$, which includes both training and test data. The correct combination of unknowns (only $2^k = 2^{21}$ possibilities) would result in the observable event of a high modeling accuracy.

We emphasize that the previously described attacks do not apply to the revised protocol of Maes [110], given that there is no PRNG anymore. Moreover, after our attacks have first been published, Aysu et al. [6] implemented the original protocol of Van Herrewege et al. [171] with an SRAM PUF, which again eliminates the PRNG. If the strong PUF would be maintained, they suggest replacing the LFSR by a block cipher-based PRF.

Exploitation of Bit Error Rates (#8)

As a follow-up to our work, Becker [12] was able to mount a modeling attack on the protocol of Van Herrewege et al. [171] that does not depend on the chosen PRNG. He points out that the functional behavior of an Arbiter PUF can be learned through the error rates of the response bits exclusively, i.e., without knowing the values of the response bits themselves. Note that the bit error rates can easily be estimated via the repeated helper data exposure.

5.3.11 Converse Authentication (2012)

The so-called converse authentication protocol of Kocabaş et al. [85], which is a revised version of the original proposal by Das et al. [36], is specified in Figure 5.19. A block diagram of the token hardware is shown in Figure 5.6(l). The term *converse* highlights the somewhat unconventional unilateral setting where tokens authenticate the server. For a given token-generated nonce $\mathbf{n}$, the server is supposed to select two challenges $\mathbf{c}$ for which the XORed difference vector of the hashed responses $\mathbf{b}$ equals $\mathbf{n}$. The restriction $\mathbf{n} \neq \mathbf{0}$ prevents an attacker from impersonating the server by replaying a previously seen tuple $(\mathbf{c}, \mathbf{h})$ in a duplicated fashion, i.e., $\mathbf{c}_i = \mathbf{c}_j$ and $\mathbf{h}_i = \mathbf{h}_j$.

The authors impose several restrictions on the capabilities of an attacker. First of all, physically invasive analysis of a token is not allowed. Moreover, the authors assume a passive, eavesdropping attacker, who tries to impersonate the server after having obtained a bounded number of genuine protocol transcripts. It is argued that with an active adversary, who may manipulate protocol traffic, the authentication will fail with overwhelming probability.

Unclear and Overly Restricted Attacker Capabilities (#1)

In addition to their contradictory specification, the capabilities of the attacker are overly restricted to be practical. First of all, it is unclear whether an attacker is allowed to gain physical access to a token and perform side-channel analysis,

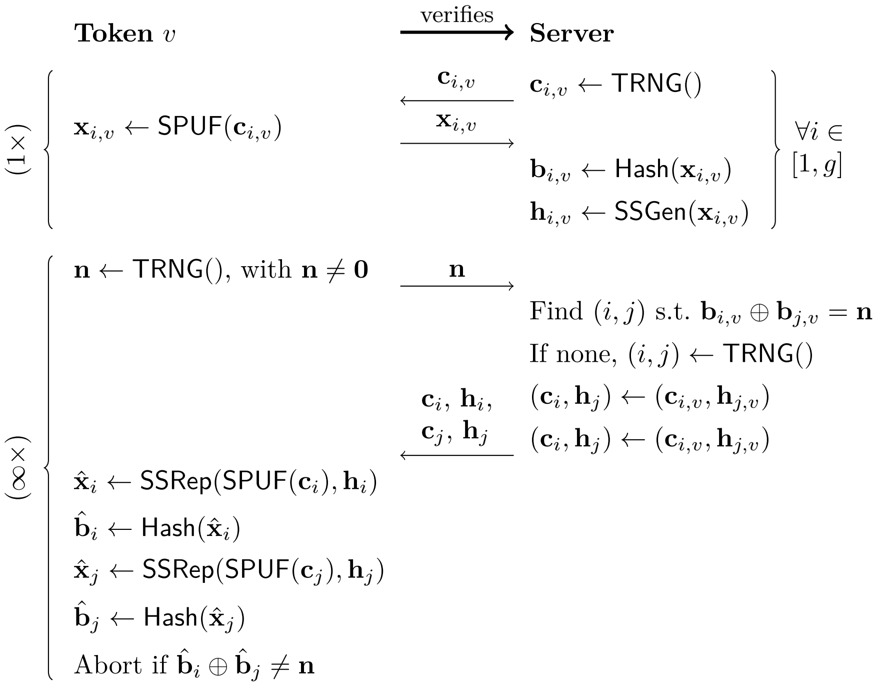

Token v
$\xrightarrow{\text{verifies}}$
Server
$(1\times)$
$\mathbf{c}_{i,v}$
$\mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}()$
$\mathbf{x}_{i,v} \leftarrow \mathsf{SPUF}(\mathbf{c}_{i,v})$
$\mathbf{x}_{i,v}$
$\forall i \in [1, g]$
$\mathbf{b}_{i,v} \leftarrow \mathsf{Hash}(\mathbf{x}_{i,v})$
$\mathbf{h}_{i,v} \leftarrow \mathsf{SSGen}(\mathbf{x}_{i,v})$
$(\infty\times)$
$\mathbf{n} \leftarrow \mathsf{TRNG}()$, with $\mathbf{n} \neq \mathbf{0}$
$\mathbf{n}$
Find (i, j) s.t. $\mathbf{b}_{i,v} \oplus \mathbf{b}_{j,v} = \mathbf{n}$
If none, $(i, j) \leftarrow \mathsf{TRNG}()$
$(\mathbf{c}_i, \mathbf{h}_j) \leftarrow (\mathbf{c}_{i,v}, \mathbf{h}_{j,v})$
$(\mathbf{c}_i, \mathbf{h}_j) \leftarrow (\mathbf{c}_{i,v}, \mathbf{h}_{j,v})$
$\mathbf{c}_i$, $\mathbf{h}_i$, $\mathbf{c}_j$, $\mathbf{h}_j$
$\hat{\mathbf{x}}_i \leftarrow \mathsf{SSRep}(\mathsf{SPUF}(\mathbf{c}_i), \mathbf{h}_i)$
$\hat{\mathbf{b}}_i \leftarrow \mathsf{Hash}(\hat{\mathbf{x}}_i)$
$\hat{\mathbf{x}}_j \leftarrow \mathsf{SSRep}(\mathsf{SPUF}(\mathbf{c}_j), \mathbf{h}_j)$
$\hat{\mathbf{b}}_j \leftarrow \mathsf{Hash}(\hat{\mathbf{x}}_j)$
Abort if $\hat{\mathbf{b}}_i \oplus \hat{\mathbf{b}}_j \neq \mathbf{n}$

Figure 5.19: The converse authentication protocol of Kocabaş et al. [85]. The authors suggest instantiating the strong PUF with an Arbiter PUF. In order to expand its single-bit response r into a long string $\mathbf{x} \in \{0, 1\}^{\lambda}$, it is suggested that λ replicas evaluate the same challenge $\mathbf{c}$ in parallel. A server-maintained database of size $g = 2^{25}$ is supposed to be practical. The protocol can optionally be extended with the establishment of a session key, i.e., $\mathbf{k} \leftarrow \mathsf{Hash}(\mathbf{b}_i, \mathbf{b}_j)$.

which is non-invasive by definition. If it is allowed, then the prohibition of physically invasive techniques would be a fully artificial constraint that a real-life attacker happily ignores. In the alternative scenario where an attacker is not allowed to obtain physical access to a token, the need for PUFs would be questionable. Then one might equally well store a key in physically insecure OTP NVM and execute reference protocol I-B, which seems to be a more efficient solution.

The motivation for prohibiting active attacks on the protocol is unclear as well. By manipulating messages, it is indeed trivial to make an otherwise genuine protocol run fail, but this statement holds for virtually every protocol that has been proposed in the history of cryptology. Protocol designers usually only consider DoS attacks where the failure is permanent, as attributed to, e.g., the desynchronization of a state. Moreover, it is unclear whether an attacker is allowed to freely query the server, given that the protocol is initiated by a token.

Inefficiencies (#6)

The proposed solution is inferior to reference protocol II-B in terms of efficiency. Most notably, the storage requirements for the server are high in comparison. For each token, e.g., $g = 2^{25}$ tuples $(\mathbf{c}, \mathbf{b}, \mathbf{h})$ need to stored instead of only a single tuple $(\mathbf{x}, \mathbf{h})$. The computational requirements for the server are even higher. An efficient method to find a matching pair of tuples was not discussed and seems far from obvious. For each protocol run, we hence assume the need for an exhaustive search among all $g(g-1)/2$ pairs of tuples. For $g = 2^{25}$ tuples, there are approximately 2^{49} pairs. Moreover, for each protocol run, a token needs to evaluate its PUF twice instead of only once. The same holds for the execution of both its recovery procedure SSRep and its cryptographic hash function. Finally, the converse authentication protocol is more restricted in the choice of its PUF. Only strong PUFs are part of the optimization domain.

Brute-Force Attacks (#8)

In practice, the proposed protocol does not scale to a comfortable security level. First, observe the probability that a genuine run of the authentication protocol succeeds in (5.3), with $\mathbf{b} \in \{0, 1\}^{\eta}$. This formula expresses the availability of a randomly generated nonce $\mathbf{n}$ among a randomly generated database of g tuples $(\mathbf{c}, \mathbf{b}, \mathbf{h})$, where the function $\mathsf{Hash}(\mathsf{SPUF}(\cdot))$ is assumed to behave as a random oracle. We believe that a similar formula of Kocabaş et al. [85] slightly overestimates the success rate. As is clear from Figure 5.20, we should choose system parameters such that $\log_2(g) \approx \eta/2 + 2$. So even with $g = 2^{25}$ tuples, or equivalently, $\approx 2^{49}$ pairs of tuples, the hashed responses are only of size $\eta \approx 46$.

$$p_{\text{success}} = 1 - \left(1 - \frac{1}{2^{\eta} - 1}\right)^{\frac{g(g-1)}{2}}. \tag{5.3}$$

We neglect the unclear and overly restricted attacker model of Kocabaş et al. [85], and consider the more realistic constraints that are adopted by most competing proposals instead. Given that a token initiates the protocol, an attacker should then be able to exhaustively query the server and construct a personal database of 2^{η} tuples $(\mathbf{n}, \mathbf{c}_i, \mathbf{h}_i, \mathbf{c}_j, \mathbf{h}_j)$. This would allow for unlimited impersonation of the server. Session keys $\mathbf{k}$, if present, cannot be retrieved. If

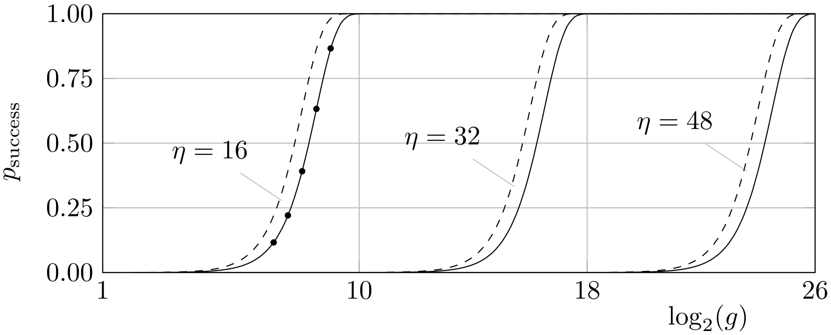


Figure 5.20: The probability that a genuine run of the converse authentication protocol succeeds, given a database of g hashed responses $\mathbf{b}$ of length $\eta \in \{16, 32, 48\}$. The solid lines correspond to the exact formula in (5.3). For verification purposes, each dot corresponds to a Monte Carlo experiment where the availability of a match for 10^5 randomly generated nonces $\mathbf{n}$ is checked. Experiments are performed for $\eta = 16$ only, due to the computational requirements. Dashed lines correspond to the incorrect formula of Kocabaş et al. [85].

the protocol would have been server-initiated instead, a brute-force attacker still has a non-negligible probability of $1/2^\eta$ to pass the authenticity check for each random guess.

Issues of the Original Version

Although the protocol of Kocabaş et al. [85] strongly resembles its original version by Das et al. [36], it is not described in terms of modifications. We observe six crucial updates. First, the secure sketch is acknowledged to require helper data. Second, the responses of the strong PUF are reinforced by a cryptographic hash function. In its absence, an attacker might be able to impersonate the server if $\mathsf{HW}(\mathbf{n}) < t$, where t denotes the number of errors that SSRep is guaranteed to correct. Consider an arbitrary response $\mathbf{x} = \mathsf{SSRep}(\mathsf{SPUF}(\mathbf{c}), \mathbf{h})$. For both the syndrome and code-offset constructions of Dodis et al. [39], an attacker might be able to produce a given $\mathbf{n}$:

$$\begin{array}{ll} \text{Code-offset construction:} & \mathbf{x} \oplus \mathbf{n} = \mathsf{SSRep}(\mathsf{SPUF}(\mathbf{c}), \mathbf{h} \oplus \mathbf{n}). \\ \text{Syndrome construction:} & \mathbf{x} \oplus \mathbf{n} = \mathsf{SSRep}(\mathsf{SPUF}(\mathbf{c}), \mathbf{h} \oplus \mathbf{n}\,\mathbf{H}^T). \end{array}$$

Third, the nonce $\mathbf{n}$ is generated by a TRNG and hence not by a PRNG. To avoid impersonation of the server via replay, the latter construction would require a state in physically secure MTP NVM, which undermines the benefits of using a PUF. Fourth, the restriction $\mathbf{n} \neq \mathbf{0}$ is introduced. However, an explicit check could be omitted given its negligible probability of occurrence. Fifth, the protocol is initiated by a token and hence not by the server. Sixth, additional constraints are imposed on the capabilities of an attacker.

5.3.12 Lee et al. I (2012)

The first protocol of Lee et al. [101] is specified in Figure 5.21; a block diagram of the token hardware is shown in Figure 5.6(m). An eavesdropping attacker can easily retrieve the current state $\mathbf{s}$, i.e., $\mathbf{s} = \mathbf{a} \oplus \mathbf{b}$. The authors therefore suggest the use of an update mechanism.

Incomplete Specification (#1,#8)

The update mechanism of the state $\mathbf{s}$ is not further specified, but should be chosen carefully. It is for example crucial that the secret constant $\mathbf{k}$ is involved in the update, i.e., $\mathbf{s} \leftarrow \mathsf{Update}(\mathbf{s}, \mathbf{k})$ rather than $\mathbf{s} \leftarrow \mathsf{Update}(\mathbf{s})$. Otherwise, an attacker could easily impersonate the server. After having eavesdropped on a genuine protocol run, the attacker replies with an arbitrary challenge $\mathbf{c}^{(2)}$ and computes $\mathbf{b}^{(2)} \leftarrow \mathbf{a}^{(2)} \oplus \mathsf{Update}(\mathbf{a}^{(1)} \oplus \mathbf{b}^{(1)})$. Moreover, the authors do not specify a countermeasure against DoS attacks. Blocking the last message could be sufficient for an attacker to desynchronize the state $\mathbf{s}$. Finally, the authors do not suggest the use of a particular strong PUF. A method to expand its usually single-bit response r into a long string $\mathbf{x}$ is not suggested either.

Physically Secure NVM Undermines PUF Benefit (#2)

The use of physically secure NVM undermines the benefit of using a PUF. Under the assumption that a token can store its key $\mathbf{k}$ in a secure manner, it is seemingly more efficient to discard the PUF and use a joint version of reference protocol I-A and reference protocol I-B instead. Tokens do not necessarily need a TRNG so as to provide freshness; the update of a physically secure state $\mathbf{s}$ would be an alternative.

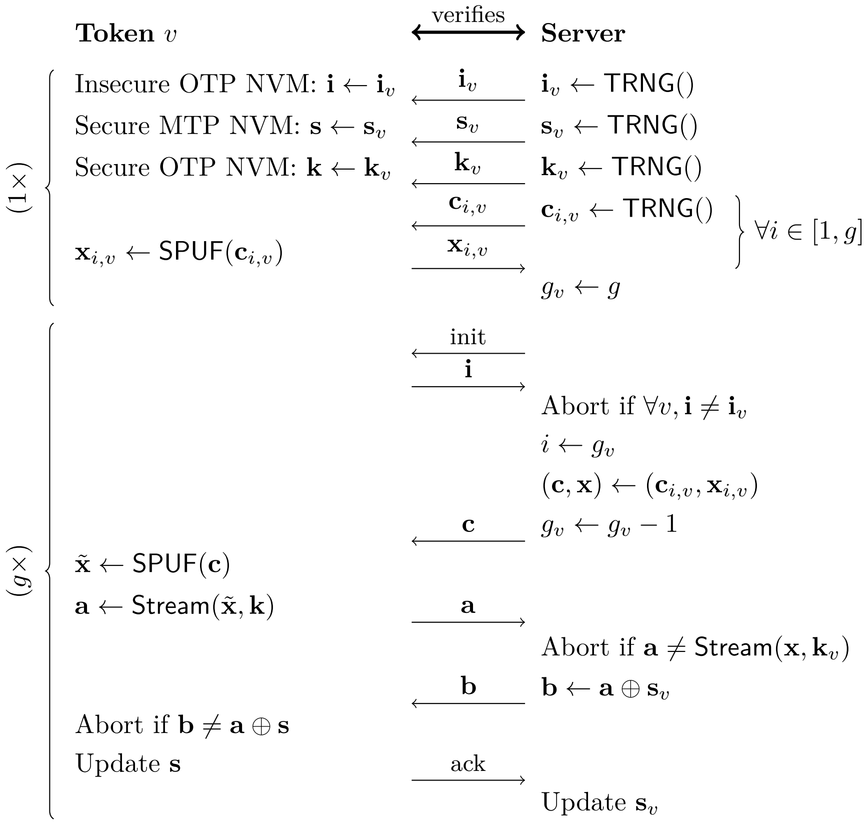

Token v
verifies
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$
$\mathbf{i}_v$
$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
Secure MTP NVM: $\mathbf{s} \leftarrow \mathbf{s}_v$
$\mathbf{s}_v$
$\mathbf{s}_v \leftarrow \mathsf{TRNG}()$
Secure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_v$
$\mathbf{k}_v$
$\mathbf{k}_v \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_{i,v}$
$\mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}()$
$\forall i \in [1, g]$
$\mathbf{x}_{i,v} \leftarrow \mathsf{SPUF}(\mathbf{c}_{i,v})$
$\mathbf{x}_{i,v}$
$g_v \leftarrow g$
$(g\times)$
init
$\mathbf{i}$
Abort if $\forall v, \mathbf{i} \neq \mathbf{i}_v$
$i \leftarrow g_v$
$(\mathbf{c}, \mathbf{x}) \leftarrow (\mathbf{c}_{i,v}, \mathbf{x}_{i,v})$
$\mathbf{c}$
$g_v \leftarrow g_v - 1$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c})$
$\mathbf{a} \leftarrow \mathsf{Stream}(\tilde{\mathbf{x}}, \mathbf{k})$
$\mathbf{a}$
Abort if $\mathbf{a} \neq \mathsf{Stream}(\mathbf{x}, \mathbf{k}_v)$
$\mathbf{b}$
$\mathbf{b} \leftarrow \mathbf{a} \oplus \mathbf{s}_v$
Abort if $\mathbf{b} \neq \mathbf{a} \oplus \mathbf{s}$
Update $\mathbf{s}$
ack
Update $\mathbf{s}_v$

Figure 5.21: The first authentication protocol of Lee et al. [101]. The stream cipher is instantiated with the affiliated design NLM-128, where $\tilde{\mathbf{x}}$ and $\mathbf{k}$ act as the key and the initialization vector respectively.

Non-Functional due to PUF Noisiness (#3)

The authors acknowledge that the responses of a PUF are noisy and mention the existence of fuzzy extractors [39] as well. However, neither the procedures SSGen and SSRep nor the transfer of helper data are an explicit part of the proposed protocol. We therefore assume the complete absence of error-correction techniques, which makes the protocol non-functional. It is unclear whether the authors may have intended to use a fuzzy extractor in an implicit manner, but even if so, its related security and efficiency concerns would not have been reflected properly. We make abstraction of the noisiness issue in the remainder of our analysis.

Cryptanalysis of NLM-128 (#8)

Orumiehchiha et al. [129] published a cryptanalytic attack on the stream cipher NLM-128, so another algorithm may have to be considered.

5.3.13 Jin et al. (2012)

The protocol of Jin et al. [73] is specified in Figure 5.22; a block diagram of the token hardware is shown in Figure 5.6(n). The privacy of a token is claimed to be preserved, i.e., an attacker can only track a token in between protocol runs. Moreover, the authors claim resistance to physical attacks that recover the state $\mathbf{s}$, i.e., future runs of the protocol are supposed to remain secure. Finally, it is acknowledged that an attacker may desynchronize the state $\mathbf{s}$ by blocking the last message $\mathbf{b}$. In order to mitigate this DoS threat, recovery logic is foreseen at the server side.

Incomplete Specification (#1)

The custom-designed variation of the Feed-forward Arbiter PUF is so vaguely specified that we were unable to interpret it as a workable design. Moreover, a method to expand its presumably single-bit response r into a long string $\mathbf{x}$ has not been proposed either.

Physically Secure NVM Undermines PUF Benefit (#2)

The use of physically secure NVM undermines the benefit of using a PUF. Although the current value of the state $\mathbf{s}$ may leak, there is no such claim for the key $\mathbf{k}$. Under the assumption that tokens can store their key $\mathbf{k}$ in a secure manner, it is seemingly more efficient to discard the PUF and execute a joint version of reference protocol I-A and reference protocol I-B instead. It can also be noted that physically secure OTP NVM would allow for the instantiation of a noiseless, device-unique function that is harder to learn than a typical strong PUF.

Non-Functional due to PUF Noisiness (#3)

The responses of the PUF are not acknowledged to be noisy, which makes the protocol non-functional. We make abstraction of this issue in the remainder of our analysis.

$$
\begin{array}{l l c l}
 & \textbf{Private token } v & \xleftrightarrow{\text{verifies}} & \textbf{Server} \\
(1\times) & \text{Insecure MTP NVM: } \mathbf{s} \leftarrow \mathbf{s}_v & \xleftarrow{\mathbf{s}_v} & \mathbf{s}_v \leftarrow \mathsf{TRNG}() \\
 & \text{Secure OTP NVM: } \mathbf{k} \leftarrow \mathbf{k}_v & \xleftarrow{\mathbf{k}_v} & \mathbf{k}_v \leftarrow \mathsf{TRNG}() \\
 & \mathbf{x}_{1,v} \leftarrow \mathsf{SPUF}(\mathbf{s}) & \xrightarrow{\mathbf{x}_{1,v}} & \\
(\infty\times) & & \xleftarrow{\mathbf{n}} & \mathbf{n} \leftarrow \mathsf{TRNG}() \\
 & \tilde{\mathbf{x}}_1 \leftarrow \mathsf{SPUF}(\mathbf{s}) & & \\
 & \tilde{\mathbf{x}}_2 \leftarrow \mathsf{SPUF}(\tilde{\mathbf{x}}_1) & & \\
 & \mathbf{a}_1 \leftarrow \tilde{\mathbf{x}}_1 \oplus \tilde{\mathbf{x}}_2 & & \\
 & \mathbf{a}_2 \leftarrow \mathsf{Hash}(\mathbf{k} \oplus \tilde{\mathbf{x}}_2 \oplus \mathbf{n}) & \xrightarrow{\mathbf{a}_1, \mathbf{a}_2} & \\
 & & & \text{If } \exists v, \mathbf{a}_2 = \\
 & & & \quad \mathsf{Hash}(\mathbf{k}_v \oplus \mathbf{x}_{2,v} \oplus \mathbf{n}), \\
 & & & \quad \text{with } \mathbf{x}_{2,v} \leftarrow \mathbf{a}_1 \oplus \mathbf{x}_{1,v}, \\
 & & & \quad \mathbf{b} \leftarrow \mathbf{k}_v \oplus \mathsf{ShiftCirc}(\mathbf{x}_{2,v}) \\
 & & & \quad \mathbf{s}_v \leftarrow \mathbf{x}_{1,v} \\
 & & & \quad \mathbf{x}_{1,v} \leftarrow \mathbf{x}_{2,v} \\
 & & & \text{Else if } \exists v, \mathbf{a}_2 = \\
 & & & \quad \mathsf{Hash}(\mathbf{k}_v \oplus \mathbf{x}_{2,v} \oplus \mathbf{n}), \\
 & & & \quad \text{with } \mathbf{x}_{2,v} \leftarrow \mathbf{a}_1 \oplus \mathbf{s}_v, \\
 & & & \quad \mathbf{b} \leftarrow \mathbf{k}_v \oplus \mathsf{ShiftCirc}(\mathbf{x}_{2,v}) \\
 & & \xleftarrow{\mathbf{b}} & \text{Else, abort} \\
 & \text{Abort if } \mathbf{b} \neq \mathbf{k} \oplus \mathsf{ShiftCirc}(\tilde{\mathbf{x}}_2) & & \\
 & \mathbf{s} \leftarrow \tilde{\mathbf{x}}_1 & &
\end{array}
$$

Figure 5.22: The authentication protocol of Jin et al. [73]. The function ShiftCirc rotates its input over half the length, i.e., $\mathsf{ShiftCirc}\big((r_1\ r_2\ \ldots\ r_{2\lambda})\big) = (r_{\lambda+1}\ \ldots\ r_{2\lambda}\ r_1\ \ldots\ r_\lambda)$. The strong PUF is instantiated with a custom-designed variation of the Feed-forward Arbiter PUF.

Server Impersonation (#8)

After having eavesdropped on a single protocol run, an attacker can impersonate the server. For this purpose, the attacker sends an arbitrary nonce $\mathbf{n}^{(2)}$ to the token of interest and replies with $\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(1)} \oplus \mathsf{ShiftCirc}\big(\mathbf{a}_1^{(2)}\big)$.

Denial-of-Service (#8)

Given that the inputs of the cryptographic hash function are XORed, an attacker is able to desynchronize the state $\mathbf{s}$. We describe two variations of this DoS threat. According to the first variant, the attacker interferes with an otherwise genuine protocol run and modifies two messages as follows: $\mathbf{n} \leftarrow \mathbf{n} \oplus \mathbf{e}$ and $\mathbf{a}_1 \leftarrow \mathbf{a}_1 \oplus \mathbf{e}$, where the error vector $\mathbf{e} \neq \mathbf{0}$ can be chosen arbitrarily. Although the token of interest rejects the server, the latter inadvertently corrupts its state. According to the second variation, the attacker first eavesdrops on a genuine protocol run. During a subsequent run of the protocol, the attacker sends $\mathbf{a}_2^{(2)} \leftarrow \mathbf{a}_2^{(1)}$ and $\mathbf{a}_1^{(2)} \leftarrow \mathbf{n}^{(1)} \oplus \mathbf{n}^{(2)}$. Again, the server inadvertently corrupts its state. As a side note, the previously described DoS attacks also result in a conflict with widely accepted privacy definitions. A single token that is persistently rejected by the server may easily be distinguished from its neighboring devices.

Token Impersonation after Leakage of the State (#8,#2)

Leakage of the state $\mathbf{s}$ allows an attacker to repeatedly impersonate a token. Moreover, there is a concurrent DoS threat that incapacitates the genuine token. The attacker first eavesdrops on a genuine protocol run and obtains $\mathbf{a}_1^{(1)}$. Subsequently, the state $\mathbf{s}^{(2)}$ is obtained via physical channels. The response $\tilde{\mathbf{x}}_1^{(2)} \leftarrow \mathbf{a}_1^{(1)} \oplus \mathbf{s}^{(2)}$ can hence be computed. A value for the response $\tilde{\mathbf{x}}_2^{(2)}$ can be chosen arbitrarily, i.e., the PUF can be replaced by an arbitrary function from now onwards. The attacker sends the corresponding messages $\mathbf{a}_1^{(2)}$ and $\mathbf{a}_2^{(2)}$ to the server.

Privacy without Identification (#9)

The proposal does not scale in the number of registered tokens, which can cause unacceptable latencies in practical use cases. During each protocol run and for each registered token, the server may have to evaluate a cryptographic hash function twice.

5.3.14 Slender PUFs (2012)

Authentication with so-called slender PUFs, as proposed by Majzoobi et al. [119], is specified in Figure 5.23. A block diagram of the token hardware is shown in Figure 5.6(o). The authors extend the basic strong PUF protocol, as previously discussed in Section 5.3.1, with a countermeasure to machine learning attacks, i.e., tokens obfuscate the link between the mutually determined challenges and the released response bits. To be precise, only a randomly selected substring **a** of the complete, concatenated response **x** is released. Unlike an attacker, the server can easily de-obfuscate the link between challenges and responses via pattern matching, given that a predictive model of the strong PUF has been constructed during the enrollment phase. As a side note, the proposal draws inspiration from the PUF-based key generator of Paral and Devadas [132], which has previously been discussed in Section 4.4.3.

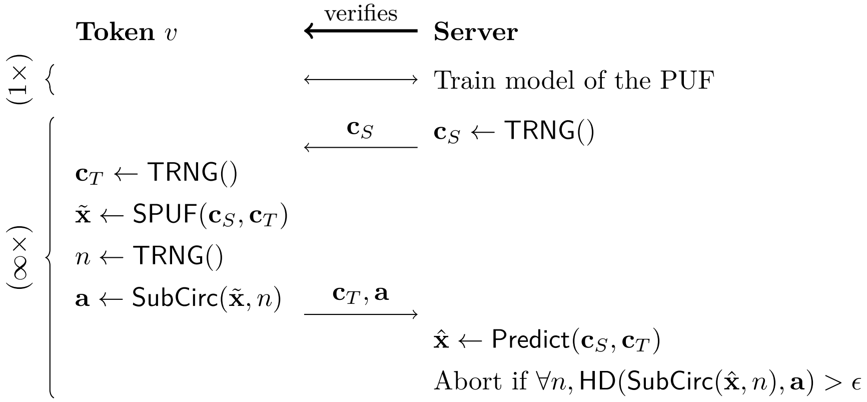

Token v
verifies
Server
$(1\times)$
Train model of the PUF
$\mathbf{c}_S$
$\mathbf{c}_S \leftarrow \mathsf{TRNG}()$
$(\infty\times)$
$\mathbf{c}_T \leftarrow \mathsf{TRNG}()$
$\tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c}_S, \mathbf{c}_T)$
$n \leftarrow \mathsf{TRNG}()$
$\mathbf{a} \leftarrow \mathsf{SubCirc}(\tilde{\mathbf{x}}, n)$
$\mathbf{c}_T, \mathbf{a}$
$\hat{\mathbf{x}} \leftarrow \mathsf{Predict}(\mathbf{c}_S, \mathbf{c}_T)$
Abort if $\forall n, \mathsf{HD}(\mathsf{SubCirc}(\hat{\mathbf{x}}, n), \mathbf{a}) > \epsilon$

Figure 5.23: Authentication with slender PUFs, as proposed by Majzoobi et al. [119]. The procedure SubCirc selects an η-bit substring from its λ-bit circular input, i.e., $\mathsf{SubCirc}((r_1\, r_2\, \cdots\, r_\lambda), n) = (r_n\; r_{(n \bmod \lambda)+1}\; \cdots\; r_{(n+\eta-2 \bmod \lambda)+1})$, with $n \in [1, \lambda]$. The strong PUF is instantiated with an Arbiter XOR PUF where challenges are permuted across chains. The challenge $(\mathbf{c}_S, \mathbf{c}_T)$ is first fed into a PRNG so that the single-bit responses r to an expanded list of challenges can be concatenated into a long string $\mathbf{x}$. The proof-of-concept implementation uses $\mathsf{PRNG}(\mathbf{c}_S \| \mathbf{c}_T) = \mathsf{LFSR}(\mathbf{c}_S) \oplus \mathsf{LFSR}(\mathbf{c}_T)$.

More recently, Rostami et al. [140] proposed an extension of their countermeasure to machine learning attacks. To be precise, substring **a** is padded with random bits before transmission, i.e., $\mathbf{a} \leftarrow \mathsf{ShiftCirc}(\mathbf{a} \| \mathbf{n}_{\text{pad}}, n_2)$, where $\mathbf{n}_{\text{pad}} \leftarrow \mathsf{TRNG}()$ and $n_2 \leftarrow \mathsf{TRNG}()$. The procedure ShiftCirc rotates its circular λ-bit input, i.e., $\mathsf{ShiftCirc}((r_1\, r_2\, \cdots\, r_\lambda), n_2) = (r_{n_2}\; r_{(n_2 \bmod \lambda)+1}\; \cdots\; r_{(n_2-2 \bmod \lambda)+1})$, with

$n_2 \in [1, \lambda]$. Moreover, the authors propose the establishment of a session key $\mathbf{k}$ as an optional extension of the protocol. This relies on the concatenation of secret indices n and n_2. Tuples $(\mathbf{c}_T, \mathbf{c}_S, \mathbf{a})$ then have to be exchanged several times so as to obtain a key $\mathbf{k}$ of sufficient length.

Machine Learning Attacks are not Excluded (#4)

Although more secure than the basic strong PUF protocol in Section 5.3.1, machine learning attacks are not excluded. First of all, observe the following contradiction. On the one hand, the strong PUF should be easy-to-model, as this enables the enrollment. On the other hand, the same PUF should be hard-to-model, as this counters attacks. For a monolithic PUF, this would be an extremely difficult balancing exercise. For composite PUFs, such as the suggested XOR construction, it works out though. During the enrollment, the individual component PUFs can be learned separately, i.e., the XOR operation is bypassed, but for an attacker this is not possible.

Nevertheless, compared to other proposals, it is hard to securely instantiate the protocol. A system provider should rely on specialized attacks that estimate a lower bound on the number of chains that needs to be XORed. For example, Becker [12] defeats instances of both the original and the extended protocol using simulated Arbiter XOR PUFs with up to four chains. Unfortunately, the system provider cannot exclude the possibility that an alternative, unpublished attack might perform better. Moreover, the XOR operation amplifies the noisiness of the PUF, i.e., an upper bound on the number of chains is imposed as well.

There is another contradiction for the extended protocol with key establishment. On the one hand, Rostami et al. [140] avoid using cryptographic algorithms in order to save resources. On the other hand, the establishment of a key is only useful if there is a cryptographic algorithm to make use of it. We hence argue that the latter keyed cryptographic algorithm might have been reused for providing reinforcement against attacks. This includes but is not limited to machine learning attacks. For example, the functionality of the LFSR-based PRNG could be provided by the keyed cryptographic algorithm as well.

PRNG Weaknesses (#8,#5)

The PRNG that is used in the proof-of-concept implementation might allow an attacker to impersonate a token. Although feedback polynomials have not been specified, we presume that an identical polynomial is used for both LFSRs. If they were supposed to be different, it seems fair that an explicit statement should have been provided. Moreover, implementation results (Table III in [119] and Table 8 in [140]) strongly suggest that there are two replicas of an identical LFSR.

Upon receipt of the server-generated challenge $\mathbf{c}_S$, the attacker chooses an identical token-generated challenge $\mathbf{c}_T$, i.e., $\mathbf{c}_T \leftarrow \mathbf{c}_S$. Under the assumption of identical feedback polynomials, the output stream of the PRNG is then equal to $\mathbf{0}$. The server hence evaluates its predictive model of the strong PUF for a list of identical challenges. The concatenated response is therefore either $\hat{\mathbf{x}} = \mathbf{0}$ or $\hat{\mathbf{x}} = \mathbf{1}$. Choosing substring $\mathbf{a}$ accordingly, the attacker passes the authenticity check with a success probability of $^1/_2$. In case the attacker has previously eavesdropped on a genuine protocol run, the success probability increases to 1 for that particular token. The attacker chooses $\mathbf{c}_T^{(2)} \leftarrow \mathbf{c}_S^{(2)} \oplus \mathbf{c}_S^{(1)} \oplus \mathbf{c}_T^{(1)}$ and $\mathbf{a}^{(2)} \leftarrow \mathbf{a}^{(1)}$. For the extended protocol of Rostami et al. [140], old sessions keys can be replayed as well. We emphasize that a careful redesign of the PRNG could provide a resolution for the previously described attack.

5.3.15 Xu and He (2012)

The protocol of Xu and He [179] is specified in Figure 5.24; a block diagram of the token hardware is shown in Figure 5.6(p). In order to preserve the privacy of a token, its identifier $\mathbf{i}$ is updated with every protocol run. An attacker can hence only track a token in between protocol runs. The authors acknowledge that an attacker may desynchronize the state $\mathbf{i}$ by blocking the last message $\mathbf{x}_2$. To mitigate this DoS threat, recovery logic is foreseen at the server side.

Incomplete Specification (#1)

The authors provide only a vague, informal description of the logic that is used to recover from a desynchronized state $\mathbf{i}$. It is stated that the server retains an old identifier $\mathbf{i}_{-1,v}$, but no further details are provided. We filled in the blanks to the best of our insights and exclude this part from cryptanalysis. Moreover, the authors do not suggest the use of a particular strong PUF. A method to expand its usually single-bit response r into a long string $\mathbf{x}$ is not suggested either.

$$
\begin{array}{llcll}
 & \textbf{Private token } v & \xleftrightarrow{\text{verifies}} & \textbf{Server} & \\
 & \text{Secure MTP NVM: } \mathbf{i} \leftarrow \mathbf{i}_v & \xleftarrow{\mathbf{i}_v} & \mathbf{i}_v \leftarrow \mathsf{TRNG}() & \\
 & \text{Secure ROM: } \mathbf{k} & & & \\
 & & \xleftarrow{\mathbf{c}_{i,v}} & \mathbf{c}_{i,v} \leftarrow \mathsf{TRNG}() & \\
(1\times)\Big\{ & \mathbf{x}_{1,i,v} \leftarrow \mathsf{SPUF}(\mathbf{c}_{i,v}) & & & \Big\}\ \forall i \in [1, g] \\
 & \mathbf{x}_{2,i,v} \leftarrow \mathsf{SPUF}(\mathbf{x}_{1,i,v}) & \mathbf{x}_{1,i,v},\ \mathbf{x}_{2,i,v}, & & \\
 & \mathbf{x}_{3,i,v} \leftarrow \mathsf{SPUF}(\mathbf{x}_{2,i,v}) & \xrightarrow{\mathbf{x}_{3,i,v}} & & \\
 & & & g_v \leftarrow g & \\
 & & \xleftarrow{\text{init}} & & \\
 & \mathbf{a} \leftarrow \mathbf{i} \oplus \mathbf{k} & \xrightarrow{\mathbf{a}} & & \\
 & & & \text{If } \exists v, \mathbf{i}_v = \mathbf{a} \oplus \mathbf{k} & \\
 & & & \quad i \leftarrow g_v & \\
 & & & \quad \mathbf{b} \leftarrow \mathbf{i}_v \oplus \mathbf{c}_{i,v} & \\
 & & & \text{Else, if } \exists v, \mathbf{i}_{-1,v} = \mathbf{a} \oplus \mathbf{k} & \\
 & & & \quad i \leftarrow g_v & \\
 & & & \quad \mathbf{b} \leftarrow \mathbf{i}_{-1,v} \oplus \mathbf{c}_{i,v} & \\
 & & & \text{Else, abort} & \\
 & & \xleftarrow{\mathbf{b}} & g_v \leftarrow g_v - 1 & \\
(g\times)\Big\{ & \hat{\mathbf{c}} \leftarrow \mathbf{b} \oplus \mathbf{i} & & & \\
 & \tilde{\mathbf{x}}_1 \leftarrow \mathsf{SPUF}(\hat{\mathbf{c}}) & \xrightarrow{\tilde{\mathbf{x}}_1} & & \\
 & & & \text{Abort if } \tilde{\mathbf{x}}_1 \neq \mathbf{x}_{1,i,v} & \\
 & & & \text{If } \mathbf{i}_v = \mathbf{a} \oplus \mathbf{k} & \\
 & & & \quad \mathbf{i}_{-1,v} \leftarrow \mathbf{i}_v & \\
 & & & \quad \mathbf{i}_v \leftarrow \mathbf{i}_v \oplus \mathbf{x}_{3,i,v} & \\
 & & \xleftarrow{\mathbf{x}_2} & \mathbf{x}_2 \leftarrow \mathbf{x}_{2,i,v} & \\
 & \tilde{\mathbf{x}}_2 \leftarrow \mathsf{SPUF}(\tilde{\mathbf{x}}_1) & & & \\
 & \text{Abort if } \tilde{\mathbf{x}}_2 \neq \mathbf{x}_2 & & & \\
 & \tilde{\mathbf{x}}_3 \leftarrow \mathsf{SPUF}(\mathbf{x}_2) & & & \\
 & \mathbf{i} \leftarrow \mathbf{i} \oplus \tilde{\mathbf{x}}_3 & & &
\end{array}
$$

Figure 5.24: The authentication protocol of Xu and He [179]. The key $\mathbf{k} \leftarrow \mathsf{TRNG}()$ is shared by all tokens. The logic for recovering from a desynchronized state $\mathbf{i}$ is reconstructed to the best of our insights, given its incomplete specification.

Physically Secure NVM Undermines PUF Benefit (#2)

The use of physically secure NVM undermines the benefit of using a PUF, given that the former could hypothetically be used to craft a noiseless, device-unique function that is harder to learn than the latter. Moreover, all eggs are put in the same basket by having a system key $\mathbf{k}$ that is shared by all tokens. Only a single token needs to be compromised for an attacker to easily obtain the identifier $\mathbf{i}$ of any given token, i.e., $\mathbf{i} = \mathbf{a} \oplus \mathbf{k}$.

Non-Functional due to PUF Noisiness (#3)

The responses of the PUF are not acknowledged to be noisy, which makes the protocol non-functional. We make abstraction of this issue in the remainder of our analysis.

Machine Learning Attacks (#4)

The authors do not acknowledge that strong PUFs are prone to machine learning attacks and do not offer any protection either. Most notably, a CRP $(\mathbf{x}_1, \mathbf{x}_2)$ is transferred in the clear. Moreover, an additional CRP $(\mathbf{x}_2, \mathbf{x}_3)$ can be obtained given that $\mathbf{x}_3^{(1)} = \mathbf{a}^{(1)} \oplus \mathbf{a}^{(2)}$. In practice, an upper bound on the number of protocol runs g should hence be imposed. This would be of no help though if the system key $\mathbf{k}$ has been compromised. An attacker would then be able to obtain a virtually unlimited number of CRPs $(\mathbf{c}, \mathbf{x}_1)$ from any given token.

Server Impersonation (#8)

An attacker only needs to eavesdrop on a single protocol run in order to successfully impersonate the server a virtually unlimited number of times. The authenticity check is passed by sending $\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(1)} \oplus \mathbf{a}^{(1)} \oplus \mathbf{a}^{(2)}$ and $\mathbf{x}_2^{(2)} \leftarrow \mathbf{x}_2^{(1)}$.

5.3.16 He and Zou (2012)

The protocol of He and Zou [57] is specified in Figure 5.25; a block diagram of the token hardware is shown in Figure 5.6(q). In order to preserve the privacy of a token, its identifier $\mathbf{i}$ is updated with every protocol run. An attacker can hence only track a token in between protocol runs. The authors claim resistance to physical attacks that recover the state $(\mathbf{i}, \mathbf{k}, \mathbf{s})$, i.e., future runs of the protocol are supposed to remain secure. Moreover, it is acknowledged that an attacker may desynchronize the state by blocking the last message $(\mathbf{a}_1, \mathbf{a}_2)$. To mitigate this DoS threat, the use of recovery logic is suggested.

Incomplete Specification (#1)

The authors provide only a vague, informal description of the logic that is used to recover from a desynchronized state $(\mathbf{i}, \mathbf{k}, \mathbf{s})$. It is stated that tokens should retain their previous state, but no further details are provided. This by itself can never work, as the server might receive either an old or a new identifier $\mathbf{i}$. Therefore, we are not able to fill in the blanks, and exclude this part from cryptanalysis. However, it should be noted that recovery logic, when not carefully designed, often leads to one or more protocol flaws. Although of lesser importance, the initialization of the state $(\mathbf{i}, \mathbf{k}, \mathbf{s})$ is not explicitly covered either. Moreover, the authors do not suggest a method to expand the single-bit response r of the Arbiter PUF into a long string $\mathbf{x}$.

Non-Functional due to PUF Noisiness (#3)

The authors wrongly assume that the noisiness of the PUF's responses is negligible, which makes the protocol non-functional in practice. We make abstraction of this issue in the remainder of our analysis.

Denial-of-Service (#8)

An attacker who interferes with an otherwise genuine protocol run can easily corrupt the state, i.e., cause DoS. It suffices to modify $\mathbf{a}_2$ to an arbitrarily chosen value. The server then inadvertently corrupts its saved response $\mathbf{x}_{2,j}$. Exploiting the linearity of the LFSR, there is a slightly more complicated alternative as well. An attacker could perform the following modifications: $\mathbf{b}_1 \leftarrow \mathbf{b}_1 \oplus \mathbf{e}$ and $\mathbf{b}_2 \leftarrow \mathbf{b}_2 \oplus \mathsf{LFSR}(\mathbf{e})$, where the error vector $\mathbf{e} \neq \mathbf{0}$ can be chosen arbitrarily. The given token then inadvertently corrupts $(\mathbf{i}, \mathbf{k})$. As an optional extension of the latter attack, the update $\mathbf{a}_1 \leftarrow \mathbf{a}_1 \oplus \mathsf{LFSR}(\mathsf{LFSR}(\mathbf{e}))$ could be performed as well, i.e., both token and server inadvertently corrupt their state.

$$
\begin{array}{lcl}
 & & \\
\textbf{Private token } v & \xleftrightarrow{\text{verifies}} & \textbf{Server}
\end{array}
$$

$$
(1\times)\left\{
\begin{array}{lcl}
\text{Insecure MTP NVM: } \mathbf{i} \leftarrow \mathbf{i}_v & \xleftarrow{\mathbf{i}_v} & \mathbf{i}_v \leftarrow \mathsf{TRNG}() \\
\text{Insecure MTP NVM: } \mathbf{k} \leftarrow \mathbf{k}_v & \xleftarrow{\mathbf{k}_v} & \mathbf{k}_v \leftarrow \mathsf{TRNG}() \\
\text{Insecure MTP NVM: } \mathbf{s} \leftarrow \mathbf{s}_v & & \\
\mathbf{x}_{1,v} \leftarrow \mathsf{SPUF}(\mathbf{s}) & & \\
\mathbf{x}_{2,v} \leftarrow \mathsf{SPUF}(\mathbf{x}_{1,v}) & \xrightarrow{\mathbf{x}_{1,v}, \mathbf{x}_{2,v}} &
\end{array}
\right.
$$

$$
(\infty\times)\left\{
\begin{array}{lcl}
 & \xleftarrow{\text{init}} & \\
 & \xrightarrow{\mathbf{i}} & \\
 & & \text{Abort if } \forall v, \mathbf{i} \neq \mathbf{i}_v \\
 & & \mathbf{n}_1 \leftarrow \mathsf{TRNG}() \\
 & & \mathbf{n}_2 \leftarrow \mathsf{PRNG}(\mathbf{n}_1) \\
 & & \mathbf{b}_1 \leftarrow \mathbf{k}_v \oplus \mathbf{n}_1 \\
 & \xleftarrow{\mathbf{b}_1, \mathbf{b}_2} & \mathbf{b}_2 \leftarrow \mathbf{x}_{1,v} \oplus \mathbf{n}_2 \\
\hat{\mathbf{n}}_1 \leftarrow \mathbf{b}_1 \oplus \mathbf{k} & & \\
\hat{\mathbf{n}}_2 \leftarrow \mathsf{PRNG}(\hat{\mathbf{n}}_1) & & \\
\tilde{\mathbf{x}}_1 \leftarrow \mathsf{SPUF}(\mathbf{s}) & & \\
\text{Abort if } \tilde{\mathbf{x}}_1 \neq \mathbf{b}_2 \oplus \hat{\mathbf{n}}_2 & & \\
\forall i \in \{2,3\}, \tilde{\mathbf{x}}_i \leftarrow \mathsf{SPUF}(\tilde{\mathbf{x}}_{i-1}) & & \\
\forall i \in [3,6], \hat{\mathbf{n}}_i \leftarrow \mathsf{PRNG}(\hat{\mathbf{n}}_{i-1}) & & \\
(\mathbf{a}_1, \mathbf{a}_2) \leftarrow (\tilde{\mathbf{x}}_2 \oplus \hat{\mathbf{n}}_3, \tilde{\mathbf{x}}_3 \oplus \hat{\mathbf{n}}_4) & & \\
(\mathbf{i}, \mathbf{k}, \mathbf{s}) \leftarrow (\mathbf{i} \oplus \hat{\mathbf{n}}_5, \mathbf{k} \oplus \hat{\mathbf{n}}_6, \tilde{\mathbf{x}}_1) & \xrightarrow{\mathbf{a}_1, \mathbf{a}_2} & \\
 & & \mathbf{n}_3 \leftarrow \mathsf{PRNG}(\mathbf{n}_2) \\
 & & \hat{\mathbf{x}}_2 \leftarrow \mathbf{a}_1 \oplus \mathbf{n}_3 \\
 & & \text{Abort if } \hat{\mathbf{x}}_2 \neq \mathbf{x}_{2,v} \\
 & & \forall i \in [4,6], \mathbf{n}_i \leftarrow \mathsf{PRNG}(\mathbf{n}_{i-1}) \\
 & & \hat{\mathbf{x}}_3 \leftarrow \mathbf{a}_2 \oplus \mathbf{n}_4 \\
 & & (\mathbf{i}_v, \mathbf{k}_v, \mathbf{x}_{1,v}, \mathbf{x}_{2,v}) \leftarrow \\
 & & \quad (\mathbf{i}_v \oplus \mathbf{n}_5, \mathbf{k}_v \oplus \mathbf{n}_6, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3)
\end{array}
\right.
$$

Figure 5.25: The authentication protocol of He and Zou [57]. The strong PUF is instantiated with a noiseless Arbiter PUF. The PRNG is instantiated with an LFSR. The logic for recovering from a desynchronized state $(\mathbf{i}, \mathbf{k}, \mathbf{s})$ is not represented due to its incomplete specification.

Token/Server Impersonation, Denial-of-Service, and Privacy Breach (#8)

The use of an LFSR results in full system disclosure. After having eavesdropped on a genuine protocol run, the attacker queries a token for its identifier $\mathbf{i}^{(2)}$. This allows for the retrieval of nonce $\mathbf{n}_5^{(1)} = \mathbf{i}^{(1)} \oplus \mathbf{i}^{(2)}$. Due to the predictability of an LFSR, nonces $\mathbf{n}_1^{(1)}$, $\mathbf{n}_2^{(1)}$, $\mathbf{n}_3^{(1)}$, $\mathbf{n}_4^{(1)}$, and $\mathbf{n}_6^{(1)}$ can be computed as well. From this point onwards, all other secret variables of the first two protocol runs can be retrieved straightforwardly.

Full System Disclosure after Leakage of the State (#8)

Regardless of the chosen PRNG, leakage of the state results in full system disclosure. First, the attacker obtains the current value of the state $(\mathbf{i}, \mathbf{k}, \mathbf{s})$ via physical channels. Note that identifier $\mathbf{i}$ could have been obtained through a simple query as well. Subsequently, the attacker eavesdrops on a genuine protocol run. This allows for the retrieval of $\mathbf{n}_1 = \mathbf{b}_1 \oplus \mathbf{k}$. Repeated evaluation of the PRNG results in nonces $\mathbf{n}_2$, $\mathbf{n}_3$, $\mathbf{n}_4$, $\mathbf{n}_5$, and $\mathbf{n}_6$. From this point onwards, the retrieval of all other secret variables is again straightforward.

5.3.17 Jung and Jung (2013)

The protocol of Jung and Jung [75] is specified in Figure 5.26; a block diagram of the token hardware is shown in Figure 5.6(r). In order to preserve the privacy of a token, its state $\mathbf{s}$ is updated with every protocol run. An attacker can hence only track a token in between protocol runs. The authors claim resistance to physical attacks that recover the state $\mathbf{s}$, i.e., future runs of the protocol are supposed to remain secure. We presume that this claim extends to the leakage of token identifier $\mathbf{i}_T$, given that an attacker may query $\mathbf{a}_1 = \mathbf{i}_T \oplus \mathbf{s}$. Finally, note that both token and server make use of timestamps.

Incomplete Specification (#1)

The authors do not specify an implementation of the stamping procedure Time. It is hence unclear whether the stamps are supposed to be *snapshots* of either a synchronized clock or an autonomous local clock. Observe that neither is trivial to implement on a resource-constrained token. One could also opt for a monotonic counter that is incremented with every protocol run, but this would require the implementation of write-secure MTP NVM instead. However, given that neither party verifies the other party's timestamp, not even to check

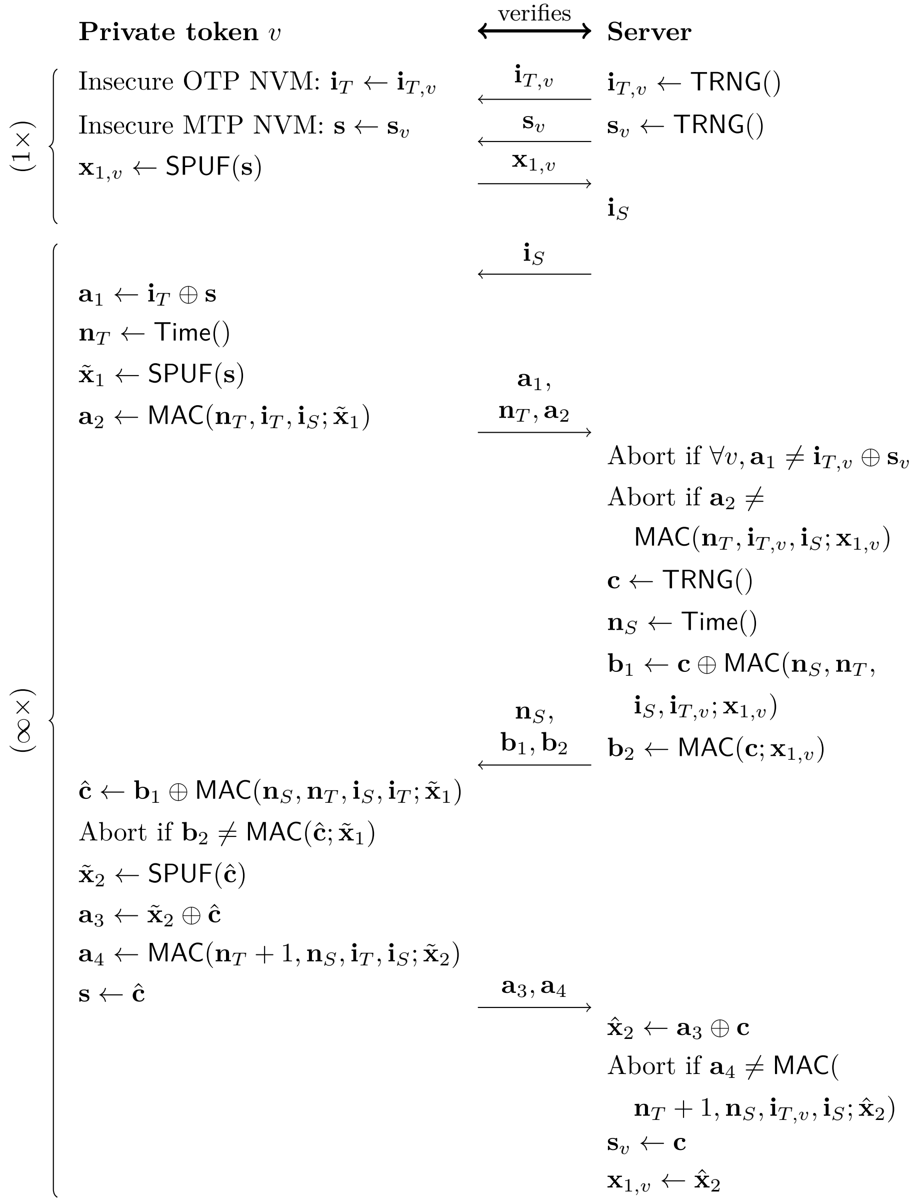

Private token v
verifies
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i}_T \leftarrow \mathbf{i}_{T,v}$
$\mathbf{i}_{T,v}$
$\mathbf{i}_{T,v} \leftarrow \mathsf{TRNG}()$
Insecure MTP NVM: $\mathbf{s} \leftarrow \mathbf{s}_v$
$\mathbf{s}_v$
$\mathbf{s}_v \leftarrow \mathsf{TRNG}()$
$\mathbf{x}_{1,v} \leftarrow \mathsf{SPUF}(\mathbf{s})$
$\mathbf{x}_{1,v}$
$\mathbf{i}_S$
$(\infty\times)$
$\mathbf{i}_S$
$\mathbf{a}_1 \leftarrow \mathbf{i}_T \oplus \mathbf{s}$
$\mathbf{n}_T \leftarrow \mathsf{Time}()$
$\tilde{\mathbf{x}}_1 \leftarrow \mathsf{SPUF}(\mathbf{s})$
$\mathbf{a}_2 \leftarrow \mathsf{MAC}(\mathbf{n}_T, \mathbf{i}_T, \mathbf{i}_S; \tilde{\mathbf{x}}_1)$
$\mathbf{a}_1, \mathbf{n}_T, \mathbf{a}_2$
Abort if $\forall v, \mathbf{a}_1 \neq \mathbf{i}_{T,v} \oplus \mathbf{s}_v$
Abort if $\mathbf{a}_2 \neq \mathsf{MAC}(\mathbf{n}_T, \mathbf{i}_{T,v}, \mathbf{i}_S; \mathbf{x}_{1,v})$
$\mathbf{c} \leftarrow \mathsf{TRNG}()$
$\mathbf{n}_S \leftarrow \mathsf{Time}()$
$\mathbf{b}_1 \leftarrow \mathbf{c} \oplus \mathsf{MAC}(\mathbf{n}_S, \mathbf{n}_T, \mathbf{i}_S, \mathbf{i}_{T,v}; \mathbf{x}_{1,v})$
$\mathbf{b}_2 \leftarrow \mathsf{MAC}(\mathbf{c}; \mathbf{x}_{1,v})$
$\mathbf{n}_S, \mathbf{b}_1, \mathbf{b}_2$
$\hat{\mathbf{c}} \leftarrow \mathbf{b}_1 \oplus \mathsf{MAC}(\mathbf{n}_S, \mathbf{n}_T, \mathbf{i}_S, \mathbf{i}_T; \tilde{\mathbf{x}}_1)$
Abort if $\mathbf{b}_2 \neq \mathsf{MAC}(\hat{\mathbf{c}}; \tilde{\mathbf{x}}_1)$
$\tilde{\mathbf{x}}_2 \leftarrow \mathsf{SPUF}(\hat{\mathbf{c}})$
$\mathbf{a}_3 \leftarrow \tilde{\mathbf{x}}_2 \oplus \hat{\mathbf{c}}$
$\mathbf{a}_4 \leftarrow \mathsf{MAC}(\mathbf{n}_T + 1, \mathbf{n}_S, \mathbf{i}_T, \mathbf{i}_S; \tilde{\mathbf{x}}_2)$
$\mathbf{s} \leftarrow \hat{\mathbf{c}}$
$\mathbf{a}_3, \mathbf{a}_4$
$\hat{\mathbf{x}}_2 \leftarrow \mathbf{a}_3 \oplus \mathbf{c}$
Abort if $\mathbf{a}_4 \neq \mathsf{MAC}(\mathbf{n}_T + 1, \mathbf{n}_S, \mathbf{i}_{T,v}, \mathbf{i}_S; \hat{\mathbf{x}}_2)$
$\mathbf{s}_v \leftarrow \mathbf{c}$
$\mathbf{x}_{1,v} \leftarrow \hat{\mathbf{x}}_2$

Figure 5.26: The authentication protocol of Jung and Jung [75]. The server has a unique identifier $\mathbf{i}_S \leftarrow \mathsf{TRNG}()$. The MAC algorithm is instantiated with the HMAC standard [93]; responses of the strong PUF serve as its key.

whether its current value exceeds its previous value, one could equally well opt for a TRNG. Moreover, the authors do not suggest the use of a particular strong PUF. A method to expand its usually single-bit response r into long string $\mathbf{x}$ has not been suggested either.

Non-Functional due to PUF Noisiness (#3)

The authors do not acknowledge that the responses of the PUF are noisy, which makes the protocol non-functional in practice. We make abstraction of this issue in the remainder of our analysis.

Denial-of-Service (#8)

There is a trivial DoS attack. Either blocking or modifying the last message $(\mathbf{a}_3, \mathbf{a}_4)$ suffices for an attacker to desynchronize the state $\mathbf{s}$.

Full System Disclosure after Leakage of the State (#8)

Leakage of the state results in full system disclosure. After having eavesdropped on a single protocol run, the attacker obtains token identifier $\mathbf{i}_T$ and state $\mathbf{s}^{(2)}$ via physical channels. The attacker can hence retrieve the secret response $\mathbf{x}_2^{(1)} = \mathbf{x}_1^{(2)} = \mathbf{a}_3^{(1)} \oplus \mathbf{s}^{(2)}$. From this point onwards, it is trivial to take full control: impersonate the server or a token, mount a DoS attack, or breach a token's privacy.

5.3.18 Lee et al. II (2013)

The second protocol of Lee et al. [102] is specified in Figure 5.27; a block diagram of the token hardware is shown in Figure 5.6(s). The authors claim resistance to physical attacks that recover the stored data $(\mathbf{i}, \mathbf{s}, \mathbf{k})$ of a token, i.e., future runs of the protocol are supposed to remain secure.

	Token v	$\xleftrightarrow{\text{verifies}}$	**Server**
$(1\times)$	Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$	$\xleftarrow{\mathbf{i}_v}$	$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
	Insecure MTP NVM: $\mathbf{s} \leftarrow \mathbf{s}_v$	$\xleftarrow{\mathbf{s}_v}$	$\mathbf{s}_v \leftarrow \mathsf{TRNG}()$
	Insecure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_v$	$\xleftarrow{\mathbf{k}_v}$	$\mathbf{k}_v \leftarrow \mathsf{TRNG}()$
	$\mathbf{x}_v \leftarrow \mathsf{SPUF}(\mathbf{s})$	$\xrightarrow{\mathbf{x}_v}$	
			$\mathbf{x}_v \leftarrow \mathsf{Hash}(\mathbf{x}_v)$
$(\infty\times)$			$\mathbf{n} \leftarrow \mathsf{TRNG}()$
		$\xleftarrow{\mathbf{b}_1}$	$\mathbf{b}_1 \leftarrow \mathbf{i}_v \oplus \mathbf{n}$
	$\tilde{\mathbf{x}}_1 \leftarrow \mathsf{SPUF}(\mathbf{s})$		
	$\mathbf{c} \leftarrow \mathsf{Hash}(\mathbf{s})$		
	$\tilde{\mathbf{x}}_2 \leftarrow \mathsf{SPUF}(\mathbf{c})$		
	$\hat{\mathbf{n}} \leftarrow \mathbf{b}_1 \oplus \mathbf{i}$		
	$\mathbf{a}_1 \leftarrow \tilde{\mathbf{x}}_1 \oplus \mathbf{s}$		
	$\mathbf{a}_2 \leftarrow \mathsf{Hash}(\tilde{\mathbf{x}}_2) \oplus \mathbf{s}$		
	$\mathbf{a}_3 \leftarrow \mathsf{MAC}(\mathbf{a}_1, \mathbf{a}_2, \hat{\mathbf{n}}; \mathbf{s})$	$\xrightarrow{\mathbf{a}_1,\ \mathbf{a}_2, \mathbf{a}_3}$	
			Abort if $\mathbf{a}_3 \neq$ $\mathsf{MAC}(\mathbf{a}_1, \mathbf{a}_2, \mathbf{n}; \mathbf{s}_v)$
			$\hat{\mathbf{x}}_1 \leftarrow \mathbf{a}_1 \oplus \mathbf{s}_v$
			Abort if $\mathbf{x}_v \neq \mathsf{Hash}(\hat{\mathbf{x}}_1)$
			$\mathbf{b}_2 \leftarrow \mathbf{k}_v \oplus \mathbf{s}_v$
			$\mathbf{b}_3 \leftarrow \mathsf{MAC}(\mathbf{b}_2, \mathbf{n}; \mathbf{s}_v)$
			$\mathbf{x}_v \leftarrow \mathbf{a}_2 \oplus \mathbf{s}_v$
		$\xleftarrow{\mathbf{b}_2, \mathbf{b}_3}$	$\mathbf{s}_v \leftarrow \mathsf{Hash}(\mathbf{s}_v)$
	Abort if $\mathbf{b}_3 \neq \mathsf{MAC}(\mathbf{b}_2, \hat{\mathbf{n}}; \mathbf{s})$		
	Abort if $\mathbf{k} \neq \mathbf{b}_2 \oplus \mathbf{s}$		
	$\mathbf{s} \leftarrow \mathbf{c}$		

Figure 5.27: The second authentication protocol of Lee et al. [102]. We presume that state $\mathbf{s}$ is also used as the key for computing MAC value $\mathbf{b}_3$, given the authors' incomplete specification.

Incomplete Specification (#1)

The authors do not specify the key that is used for computing the MAC value $\mathbf{b}_3$. There is seemingly no objection against the use of state $\mathbf{s}$ for this purpose. Moreover, the authors do not suggest the use of a particular strong PUF. A method to expand its usually single-bit response r into a long string $\mathbf{x}$ has not been suggested either.

Non-Functional due to PUF Noisiness (#3)

The responses of the PUF are not acknowledged to be noisy, which makes the protocol non-functional in practice. We make abstraction of this issue in the remainder of our analysis.

No Guidance on Resources (#6)

The authors claim that their protocol is efficient, given that a token computes only three hash values and two MAC values during a genuine run. Although we actually count two instead of three hash values, we want to point a different matter here. Neither the cryptographic hash function nor the MAC algorithm has been instantiated, but in order to save silicon area, it might be advisable to derive the MAC algorithm from the hash function, for example.

Denial-of-Service (#8)

There is a simple DoS attack. Either blocking or modifying the last message $(\mathbf{b}_2, \mathbf{b}_3)$ suffices for an attacker to desynchronize the state $\mathbf{s}$.

Full System Disclosure after Leakage of Stored Data (#8)

Leakage of the stored data of a token results in full system disclosure. After having eavesdropped on a genuine protocol run, the attacker obtains $\mathbf{i}$, $\mathbf{s}^{(2)}$ and $\mathbf{k}$ via physical channels. The attacker can now successfully impersonate the server. It suffices to first generate an arbitrary nonce $\mathbf{n}^{(2)}$ and then compute $\mathbf{b}_1^{(2)}$, $\mathbf{b}_2^{(2)}$, and $\mathbf{b}_3^{(2)}$ accordingly. At the same time, this also causes desynchronization between the given token and the real server. Alternatively, interference with a second, otherwise genuine protocol run allows an attacker to impersonate the given token from the third run onwards. On receipt of $\mathbf{b}_1^{(2)}$, the attacker retrieves nonce $\mathbf{n}^{(2)} = \mathbf{b}_1^{(2)} \oplus \mathbf{i}$. The attacker replaces the PUF by an arbitrarily chosen function and outputs $\tilde{\mathbf{x}}_1$; the values of $\mathbf{a}_2$ and $\mathbf{a}_3$ are modified accordingly. The value of $\mathbf{a}_1$ is not modified. At the same time, this also causes desynchronization between the real token and the server.

5.3.19 Noise Bifurcation (2014)

The so-called noise bifurcation protocol of Yu et al. [190] is specified in Figure 5.28. A block diagram of the token hardware is shown in Figure 5.6(t). The proposal can be understood as a variation on slender PUFs, which were discussed in Section 5.3.14. Again, machine learning attacks on the basic strong PUF protocol in Section 5.3.1 are countered by obfuscating the link between mutually determined challenges and released response bits. To be precise, only a randomly decimated version $\mathbf{a}$ of a complete, concatenated response $\mathbf{x}$ is released. This effect is referred to as *learning noise*, which not to be confused with physical noise. Unlike an attacker, the server can de-obfuscate the link between challenges and responses, given that a predictive model of the strong PUF has been constructed during the enrollment phase.

Incomplete Specification (#1)

The authors do not commit to a specific implementation of the LFSR-based PRNG. Therefore, we are unable to properly assess the security and efficiency of the proposed protocol. Recall that the LFSR-based PRNG of both the reverse fuzzy extractor protocol [171] in Section 5.3.10 and the slender PUF protocol [119, 140] in Section 5.3.14 was found to be insecure.

Machine Learning Attacks are not Excluded (#4)

Although more secure than the basic strong PUF protocol in Section 5.3.1, machine learning attacks are not excluded. First of all, observe the following contradiction. On the one hand, the strong PUF should be easy-to-model, as this enables the enrollment. On the other hand, the same PUF should be hard-to-model, as this counters attacks. For a monolithic PUF, this would be an extremely difficult balancing exercise. For composite PUFs, such as the suggested XOR construction, it works out though. During the enrollment, the individual component PUFs can be learned separately, i.e., the XOR operation is bypassed, but for an attacker this is not possible.

$$
\begin{array}{llcl}
 & \textbf{Token } v & \xleftarrow{\text{verifies}} & \textbf{Server} \\
(1\times)\,\{ & & \longleftrightarrow & \text{Train model of the PUF} \\
 & & \xleftarrow{\mathbf{c}_S} & \mathbf{c}_S \leftarrow \mathsf{TRNG}() \\
 & \mathbf{c}_T \leftarrow \mathsf{TRNG}() & & \\
 & \tilde{\mathbf{x}} \leftarrow \mathsf{SPUF}(\mathbf{c}_S, \mathbf{c}_T) & & \\
 & \mathbf{n} \leftarrow \mathsf{TRNG}() & & \\
(\infty\times)\,\{ & \mathbf{a} \leftarrow \mathsf{Decimate}(\tilde{\mathbf{x}}, \mathbf{n}) & \xrightarrow{\mathbf{c}_T, \mathbf{a}} & \\
 & & & \hat{\mathbf{x}} \leftarrow \mathsf{Predict}(\mathbf{c}_S, \mathbf{c}_T) \\
 & & & \mathcal{I} \leftarrow \mathsf{Mask}(\hat{\mathbf{x}}) \\
 & & & \hat{\mathbf{a}} \leftarrow \mathsf{Decimate}(\hat{\mathbf{x}}, \mathbf{0}) \\
 & & & \text{Abort if } \sum_{i \in \mathcal{I}} \mathsf{HD}(a_i, \hat{a}_i) > \epsilon
\end{array}
$$

Figure 5.28: The noise bifurcation protocol of Yu et al. [190]. The strong PUF is instantiated with an Arbiter XOR PUF where all chains receive a different challenge $\mathbf{c}$. The seed $(\mathbf{c}_S, \mathbf{c}_T)$ is fed into an LFSR-based PRNG so that the single-bit responses r to an expanded list of challenges $\mathbf{c}$ can be concatenated into a long string $\mathbf{x}$. The procedure Decimate subdivides concatenated response $\mathbf{x}$ into q partitions of size λ and retains a randomly selected bit x for each partition, i.e., $\mathsf{Decimate}(\mathbf{x}, n_1, n_2, \cdots, n_q) = (x_{n_1}\, x_{n_2+\lambda} \cdots x_{n_q+(q-1)\lambda})$, where $\forall i \in [1, q]$, N_i is selected randomly, uniformly, and independently from $[1, \lambda]$. The procedure Mask identifies all partitions that contain identical response bits, i.e., $\mathsf{Mask}(\hat{\mathbf{x}}) = \{i \in [1, q] \mid \hat{x}_{(i-1)\lambda+1} = \hat{x}_{(i-1)\lambda+2} = \cdots = \hat{x}_{i\,\lambda}\}$.

Nevertheless, compared to other proposals, it is hard to securely instantiate the protocol. A system provider should rely on specialized attacks that estimate a lower bound on the number of chains that needs to be XORed. Yu et al. [190] were able to defeat instances of their own protocol using simulated Arbiter XOR PUFs with up to four chains. Tobisch and Becker [165] later performed a slightly more effective attack. Unfortunately, a system provider cannot exclude the possibility that an alternative, unpublished attack might further improve the performance. Moreover, the XOR operation amplifies the noisiness of the PUF, i.e., an upper bound on the number of chains is imposed as well.

5.3.20 System of PUFs (2014)

The so-called System of PUFs of Konigsmark et al. [91] is specified in Figure 5.29; a block diagram of the token hardware is shown in Figure 5.6(u). The authors propose a two-level authentication scheme. The second level, which consists of the so-called *secure* PUF, is identical in nature to the basic authentication method in Section 5.3.1. Although assumed to be resistant to machine learning attacks by itself, the authors nevertheless extend the protocol with an insecure first level. This succession of a so-called *hidden* PUF and a so-called *guard* PUF is easy-to-model by an attacker, given that the server is faced with an identical effort during the enrollment. The need for the seemingly superfluous first level hinges on four reasons:

- The time penalty that is incurred by an attacker for modeling the first level is claimed to disable many threats in real-life use cases.
- The first level is claimed to disable a DoS attack on an otherwise self-sufficient second level. According to the authors, the basic authentication method in Section 5.3.1 allows an attacker to deplete the database of the server, given that it is infeasible for the latter to collect and store an exponentially large number of CRPs during the enrollment.
- It is claimed that a breach of the first level can be recognized by the system. An attacker who repeatedly replies with correct responses $\tilde{\mathbf{r}}_G$ to respective challenges $(\mathbf{c}_H, \mathbf{c}_G)$, but incorrect responses $\tilde{\mathbf{r}}_S$ to respective challenges $\mathbf{c}_S$, is detected. Once all challenges $\mathbf{c}_S$ are depleted, breach recovery may be initiated.
- Although a predictive model for response $\mathbf{x}_G$ can easily be obtained by both the server and an attacker, the latter might actually need an additional model for the hidden response $\mathbf{x}_H$ in order to attempt modeling the second level.

Incomplete Specification (#1)

The suggested, RO-based designs for instantiating the *hidden* PUF do not inherently produce stable response bits r, and in fact require the use of an HDA. Suh and Devadas [161] apply the η-out-of-λ selection scheme that was previously discussed in Section 4.5.2, while Yu and Devadas [187] apply the IBS scheme, as previously discussed in Section 4.5.4. The security and efficiency concerns of storing and transferring helper data $\mathbf{h}$ are not reflected in the protocol. Consider,

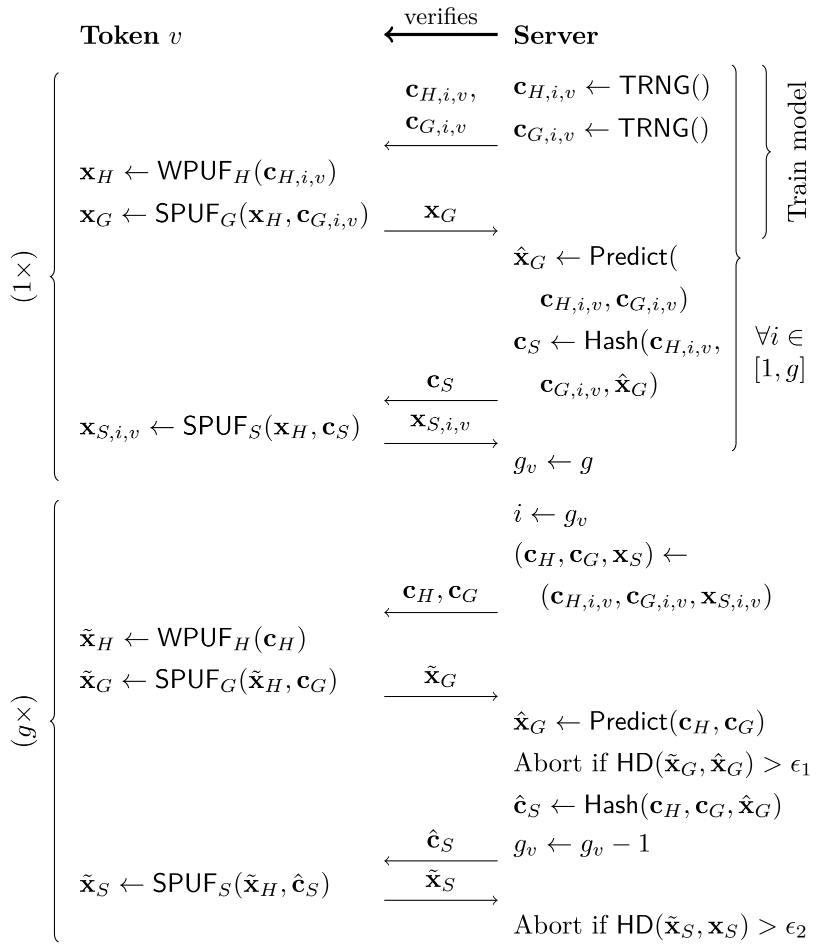

verifies
Token v
Server
$\mathbf{c}_{H,i,v}$, $\mathbf{c}_{G,i,v}$
$\mathbf{c}_{H,i,v} \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_{G,i,v} \leftarrow \mathsf{TRNG}()$
Train model
$\mathbf{x}_H \leftarrow \mathsf{WPUF}_H(\mathbf{c}_{H,i,v})$
$\mathbf{x}_G \leftarrow \mathsf{SPUF}_G(\mathbf{x}_H, \mathbf{c}_{G,i,v})$
$\mathbf{x}_G$
$(1\times)$
$\hat{\mathbf{x}}_G \leftarrow \mathsf{Predict}(\mathbf{c}_{H,i,v}, \mathbf{c}_{G,i,v})$
$\mathbf{c}_S \leftarrow \mathsf{Hash}(\mathbf{c}_{H,i,v}, \mathbf{c}_{G,i,v}, \hat{\mathbf{x}}_G)$
$\forall i \in [1, g]$
$\mathbf{c}_S$
$\mathbf{x}_{S,i,v} \leftarrow \mathsf{SPUF}_S(\mathbf{x}_H, \mathbf{c}_S)$
$\mathbf{x}_{S,i,v}$
$g_v \leftarrow g$
$i \leftarrow g_v$
$(\mathbf{c}_H, \mathbf{c}_G, \mathbf{x}_S) \leftarrow (\mathbf{c}_{H,i,v}, \mathbf{c}_{G,i,v}, \mathbf{x}_{S,i,v})$
$\mathbf{c}_H, \mathbf{c}_G$
$\tilde{\mathbf{x}}_H \leftarrow \mathsf{WPUF}_H(\mathbf{c}_H)$
$\tilde{\mathbf{x}}_G \leftarrow \mathsf{SPUF}_G(\tilde{\mathbf{x}}_H, \mathbf{c}_G)$
$\tilde{\mathbf{x}}_G$
$(g\times)$
$\hat{\mathbf{x}}_G \leftarrow \mathsf{Predict}(\mathbf{c}_H, \mathbf{c}_G)$
Abort if $\mathsf{HD}(\tilde{\mathbf{x}}_G, \hat{\mathbf{x}}_G) > \epsilon_1$
$\hat{\mathbf{c}}_S \leftarrow \mathsf{Hash}(\mathbf{c}_H, \mathbf{c}_G, \hat{\mathbf{x}}_G)$
$\hat{\mathbf{c}}_S$
$g_v \leftarrow g_v - 1$
$\tilde{\mathbf{x}}_S \leftarrow \mathsf{SPUF}_S(\tilde{\mathbf{x}}_H, \hat{\mathbf{c}}_S)$
$\tilde{\mathbf{x}}_S$
Abort if $\mathsf{HD}(\tilde{\mathbf{x}}_S, \mathbf{x}_S) > \epsilon_2$

Figure 5.29: The authentication protocol of Konigsmark et al. [91] comprises a system of three PUFs. The so-called *hidden* PUF is required to produce highly stable response bits r, and is instantiated with a weak, RO-based design [161, 187] that has a 16-bit challenge. The so-called *guard* PUF is instantiated with an Arbiter PUF that has 32 stages. The so-called *secure* PUF is assumed to be resistant to machine learning attacks, and is instantiated with an Arbiter XOR PUF that has $q = 4$ chains of $m = 64$ stages each. Challenges $\mathbf{c}$ are fed into a 16-bit, 32-bit, and 64-bit LFSR respectively so that the single-bit responses r can be concatenated into 16-bit, 32-bit, and 64-bit strings $\mathbf{x}$ respectively.

for example, the prevalence of helper data manipulation attacks. Moreover, given that helper data **h** needs to be generated first, either a one-time interface or an additional module on the device should be implemented.

Machine Learning Attacks (#4)

Although the authors seemingly suggest the opposite, there does not exist a single well-validated design of a strong PUF that is resistant to machine learning attacks. Stated otherwise: the *secure* PUF, and hence also the protocol, cannot be instantiated in a secure manner. Note that the suggested 4-chain XOR PUF is relatively easy-to-model, even with techniques that predate the proposal [145]. This issue only got worse with the more effective, hybrid attack of Becker [13].

Inefficiencies (#6)

Playing along with the assumption of a *secure* PUF, the first level still seems superfluous. The reasoning behind each of the authors' arguments is either flawed or lacking nuance:

- It is highly unlikely that the time an attacker needs for modeling the first level significantly enhances the security in practical scenarios. Given the high throughput for querying CRPs in modern devices [13], as well as the low latencies that learning algorithms might obtain [145, 13], the total elapsed time might be in the milliseconds to seconds range. In most use cases, an attacker with physical access is expected to have more time than that. If less time is given, it would not be feasible to mount a physically invasive attack on embedded NVM either, which undermines the need for PUFs. Moreover, the higher the modeling resistance of the first level, the higher the burden that is imposed on the enrollment. Note that an attacker may only need a model for a single device in order to succeed, while the server needs to obtain a model for every manufactured device.
- It is unfair to state that the basic authentication protocol in Section 5.3.1 is prone to DoS attacks by default. It all depends on which party initiates the protocol, i.e., an aspect that has not been covered. Figure 5.7 represents the *common sense* version where the server acts as sole initiator. For use cases that require a PUF-enabled device to initiate the protocol, *time-out* mechanisms could be foreseen. In fact, once the attacker obtains a predictive model for the first level of the newly proposed protocol, the inherent resistance to DoS attacks is not any better than for the basic protocol.

- The potential for recognizing breaches of the first level is much lower than what is claimed. The authors seemingly assume that an attacker who modeled the first level blindly engages in a protocol run with the server. However, playing along with the assumption of a truly *secure* PUF, this would be a useless effort, given that response $\mathbf{x}_S$ is of sufficient length such that random guessing is of no concern. In real life, where the *secure* PUF happens to be insecure, an attacker would be inclined to successfully model both levels before attempting to engage in a protocol run with the server.

- Although an attacker seemingly needs a predictive model for the hidden response $\mathbf{x}_H$, while the server does not, the corresponding enhancement of the system security might be marginal. We conjecture that state-of-the-art learning techniques can handle the given problem directly, but even if not, there are at least three potential shortcuts. First, the suggested length of challenge $\mathbf{c}_H$ is only 16 bits. This implies that an attacker could attempt learning 2^{16} predictive models for the second level, i.e., one for each element of $\mathcal{C}_H$. Second, given that the *hidden* PUF requires helper data in practice, its manipulation could enable a shortcut for retrieving $\mathbf{x}_H$. Third, the use of LFSRs enables an orthogonal line of threats. For example, by feeding the target device a fixed point $\mathbf{c}_H = \mathbf{0}$, the hidden response $\mathbf{x}_H$ is known to be either $\mathbf{0}$ or $\mathbf{1}$.

5.3.21 PUF Lockdown (2016)

Yu et al. [188], which includes the author of this PhD thesis, specified two so-called lockdown protocols. The first, most basic protocol is specified in Figure 5.30. A block diagram of its token hardware is shown in Figure 5.6(v). The design prevents an attacker from querying a token for CRPs that have not yet been disclosed during a prior, genuine protocol run. This way, the issues of either one out of two previously discussed, unilateral authentication protocols are resolved. We distinguish between the use of a weak and a strong PUF:

- When instantiated with a weak PUF, the lockdown protocol provides a more efficient alternative to the SHIC proposal of Rührmair et al. [143] in Section 5.3.7. The number of cells can be reduced considerably, no constraints are imposed on the read-out throughput, and a comfortable 128-bit security level can easily be achieved. Under the assumption of uniformly distributed cell contents, the system is still secure against a computationally unrestricted adversary. Moreover, there is no need for emerging high-density technologies like RRAM; a more conventional SRAM PUF can be used.

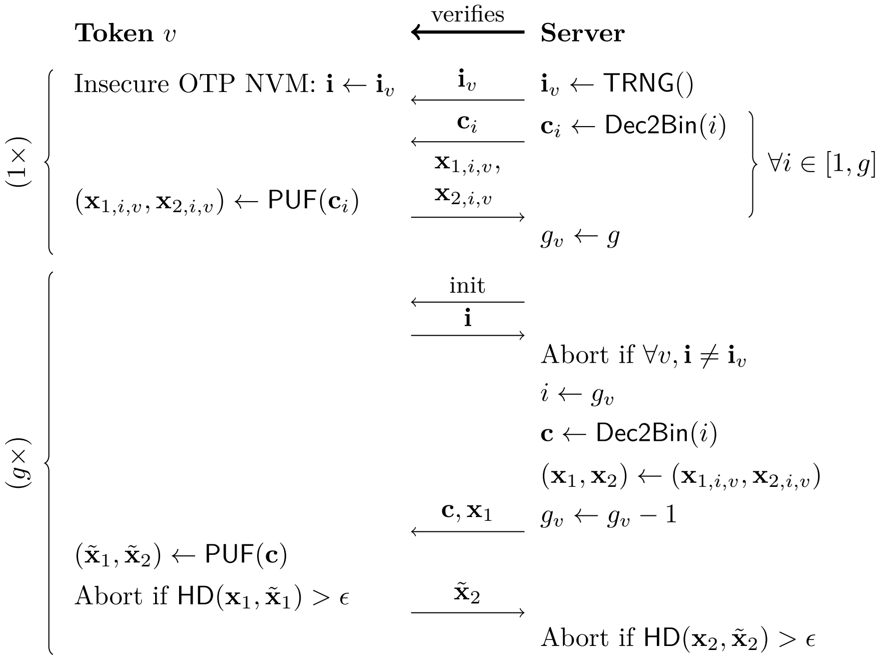

Token v
verifies
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$
$\mathbf{i}_v$
$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
$\mathbf{c}_i$
$\mathbf{c}_i \leftarrow \mathsf{Dec2Bin}(i)$
$\forall i \in [1, g]$
$(\mathbf{x}_{1,i,v}, \mathbf{x}_{2,i,v}) \leftarrow \mathsf{PUF}(\mathbf{c}_i)$
$\mathbf{x}_{1,i,v},$
$\mathbf{x}_{2,i,v}$
$g_v \leftarrow g$
$(g\times)$
init
$\mathbf{i}$
Abort if $\forall v, \mathbf{i} \neq \mathbf{i}_v$
$i \leftarrow g_v$
$\mathbf{c} \leftarrow \mathsf{Dec2Bin}(i)$
$(\mathbf{x}_1, \mathbf{x}_2) \leftarrow (\mathbf{x}_{1,i,v}, \mathbf{x}_{2,i,v})$
$\mathbf{c}, \mathbf{x}_1$
$g_v \leftarrow g_v - 1$
$(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) \leftarrow \mathsf{PUF}(\mathbf{c})$
Abort if $\mathsf{HD}(\mathbf{x}_1, \tilde{\mathbf{x}}_1) > \epsilon$
$\tilde{\mathbf{x}}_2$
Abort if $\mathsf{HD}(\mathbf{x}_2, \tilde{\mathbf{x}}_2) > \epsilon$

Figure 5.30: The first lockdown protocol of Yu et al. [188]. Despite the presence of two Hamming distance checks, the authentication is unilateral, given the lack of token-generated freshness. Either a weak or a strong PUF can be used, e.g., an SRAM or Arbiter XOR PUF respectively. We emphasize that the challenge $\mathbf{c}$ is small-sized in either case, i.e., $|\mathcal{C}| = g$. For example, if the number of authentications g is limited to 1024, then the encoding procedure $\mathsf{Dec2Bin}$ outputs challenges $\mathbf{c}$ with a length of 10 bit only. In the case of a strong PUF, the challenge $\mathbf{c}$ is first fed into an LFSR-based PRNG so that the 1-bit responses r to an expanded list of challenges can be concatenated into a long string $(\mathbf{x}_1, \mathbf{x}_2)$.

- When instantiated with a strong PUF, the lockdown protocol provides a more secure alternative to the basic authentication proposal [131, 161, 38] in Section 5.3.1. An attacker cannot freely query a token for a virtually unlimited number of CRPs anymore; an upper bound on the number of training CRPs is imposed. A heuristic security analysis is performed, based on state-of-the-art machine learning techniques.

Side-channel analysis, possibly paired with machine learning techniques in the case of strong PUF, is a major concern in practice. Observe that an attacker can force a token to evaluate the response $(\mathbf{x}_1, \mathbf{x}_2)$ to a fixed challenge $\mathbf{c}$ a virtually unlimited number of times. First of all, averaging the repeated measurements of side-channel traces can bring the signal-to-noise ratio to workable levels. This is illustrated by the photonic emission analysis of Tajik et al. [163] on delay-based PUFs. Moreover, the noise level of each individual bit in a previously unlocked response $\mathbf{x}_2$ can be estimated, which is a side-channel by itself. Recall that Becker [13] successfully paired noise measurement and machine learning techniques for Arbiter XOR PUFs.

The second lockdown protocol, as is specified in Figure 5.31, aims to prevent the latter types of attacks. A block diagram of its token hardware is shown in Figure 5.6(w). The main difference with the first protocol is that tokens now generate a fresh random challenge $\mathbf{c}_T$ during each run. Although not advertised as such, the previously discussed slender PUF [119, 140] and noise bifurcation [190] proposals adopt a similar countermeasure. Even if the latter two proposals are competitors within their own category, they could individually join forces with the lockdown protocol. A combined protocol could not only obfuscate the link between challenges and responses, but also impose an upper bound on the number of exposed CRPs.

For both protocols, the PRNG consists of a maximum-length LFSR, having a state $\mathbf{s} \in \{0,1\}^\lambda$. Initialized with a nonzero, λ-bit seed value, the LFSR starts cycling through a subset out of $2^\lambda - 1$ states, i.e., one challenge bit per state is produced until the long response pair $(\mathbf{x}_1, \mathbf{x}_2)$ is completely evaluated. For the first protocol, the seed value is $\mathbf{s} = \mathbf{a}||\mathbf{c}$, where the initialization vector $\mathbf{a}$ is a hardcoded constant. For the second protocol, the seed value is $\mathbf{s} = \mathbf{a}||\mathbf{c}_S||\mathbf{c}_T$, where $\mathbf{a}$ is again identical for all fabricated tokens. Given that both LFSRs start cycling from a partially identical seed, the output of the first few cycles may be discarded in order to enhance the uniformity characteristics of the challenge stream. The main danger is that sequences of states, i.e., gray-colored arcs in Figure 5.32, might overlap across protocol runs. If the corresponding offset turns out to be an integer multiple of the number of challenge bits of the PUF, part of a previously disclosed response pair $(\mathbf{x}_1, \mathbf{x}_2)$ could be replayed. We eliminate all forms of overlap as follows:

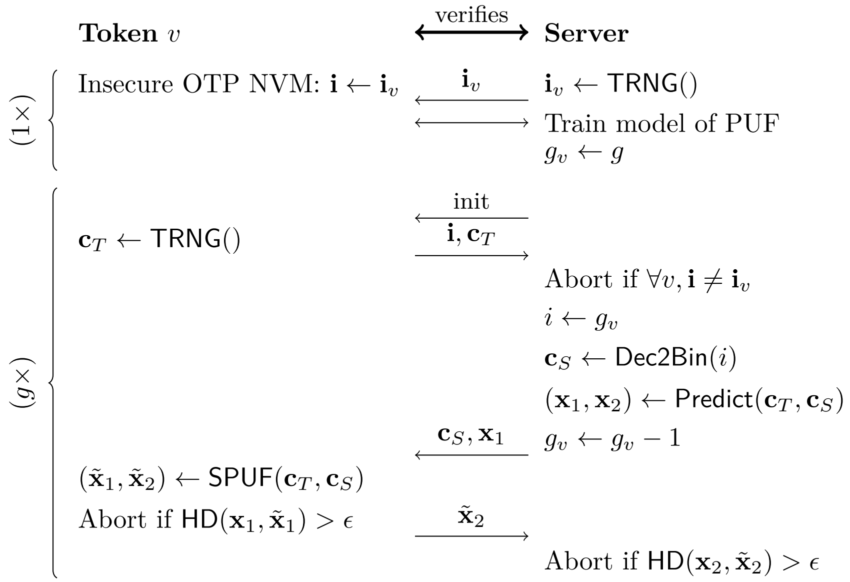
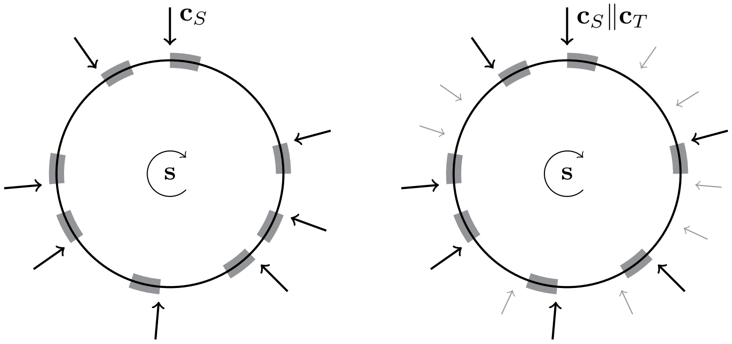

Token v
verifies
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_v$
$\mathbf{i}_v$
$\mathbf{i}_v \leftarrow \mathsf{TRNG}()$
Train model of PUF
$g_v \leftarrow g$
$(g\times)$
init
$\mathbf{c}_T \leftarrow \mathsf{TRNG}()$
$\mathbf{i}, \mathbf{c}_T$
Abort if $\forall v, \mathbf{i} \neq \mathbf{i}_v$
$i \leftarrow g_v$
$\mathbf{c}_S \leftarrow \mathsf{Dec2Bin}(i)$
$(\mathbf{x}_1, \mathbf{x}_2) \leftarrow \mathsf{Predict}(\mathbf{c}_T, \mathbf{c}_S)$
$\mathbf{c}_S, \mathbf{x}_1$
$g_v \leftarrow g_v - 1$
$(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) \leftarrow \mathsf{SPUF}(\mathbf{c}_T, \mathbf{c}_S)$
Abort if $\mathsf{HD}(\mathbf{x}_1, \tilde{\mathbf{x}}_1) > \epsilon$
$\tilde{\mathbf{x}}_2$
Abort if $\mathsf{HD}(\mathbf{x}_2, \tilde{\mathbf{x}}_2) > \epsilon$

Figure 5.31: The second lockdown protocol of Yu et al. [188]. The authentication is mutual rather than unilateral. The strong PUF is instantiated with an Arbiter XOR PUF. The server-generated challenge $\mathbf{c}_S$ is small-sized, i.e., $|\mathcal{C}_S| = g$. For example, if the number of authentications g is limited to 1024, then the encoding procedure $\mathsf{Dec2Bin}$ outputs challenges $\mathbf{c}_S$ with a length of 10 bit only. The token-generated challenge $\mathbf{c}_T$ is considerably larger, e.g., 128 bit. A challenge pair $(\mathbf{c}_T, \mathbf{c}_S)$ is fed into an LFSR-based PRNG so that the 1-bit responses r to an expanded list of challenges can be concatenated into a long string $(\mathbf{x}_1, \mathbf{x}_2)$.

- For protocol I, we suggest to run an exhaustive check during an early design phase. For a given initialization vector $\mathbf{a}$ and for all g entry points, the designer checks whether or not an entry point reappears in the course of the LFSR run. Stated otherwise: if, after the seed value $\mathbf{s} = \mathbf{a}||\mathbf{c}$ is applied, a state that begins with $\mathbf{a}$ reappears, that particular LFSR design is disqualified. This can be remedied by an alternate value for $\mathbf{a}$, a different feedback polynomial, etc. We impose the constraint $\mathbf{a} \neq \mathbf{0}$ in order to disable the fixed point $\mathbf{s} = \mathbf{0}$. For a maximum-length LFSR, which has an even number of taps, $\mathbf{s} = \mathbf{1}$ cannot be a fixed point.
- For protocol II, an exhaustive check during the design phase is no longer feasible, but we can use a run-time check instead. If, after the seed value $\mathbf{s} = \mathbf{a}||\mathbf{c}_S||\mathbf{c}_T$ has been applied, a state that begins with $\mathbf{a}$ reappears in

the course of the LFSR run, the protocol run should be aborted. The run-time check is implemented both at the token side and at the server side. We impose again the constraint $\mathbf{a} \neq \mathbf{0}$.



(a) Lockdown protocol I. (b) Lockdown protocol II.

Figure 5.32: The LFSR-based PRNG of (a) lockdown protocol I and (b) lockdown protocol II. The circle represents the complete sequence of $2^\lambda - 1$ states $\mathbf{s} \in \{0,1\}^\lambda \setminus \{\mathbf{0}\}$, traversed through in clockwise direction. Each gray-colored arc highlights a sequence of states that ultimately results in a response pair $(\mathbf{x}_1, \mathbf{x}_2)$. Points via which the circular stream can be entered are indicated by an arrow. These comprise only a very small fraction of the total number of states, i.e., the attack surface is minimized by design.

5.4 Overview and Discussion

Table 5.1 provides an overview of our analysis in Section 5.3. We adopt the perspective of an interested system provider who aims to select a protocol for implementation purposes. Proposals that do not offer any resistance to both noise and machine learning attacks are discarded. The same holds for proposals that are vulnerable to conventional protocol attacks. We do not object that the discarded proposals might contain worthwhile concepts. The remaining proposals are marked bold for further consideration and are subdivided into two categories:

- The first category of retained protocols comprehends all forms of PUF-based key generation, where the authentication is performed with a keyed cryptographic algorithm. This approach might provide an

Table 5.1: The symbol ✓ denotes 'yes'. The symbol × denotes 'no'. The symbol ∼ denotes the middle ground. An empty cell means 'non-applicable'. A grayed-out cell means 'irrelevant due to a catastrophic issue'.

	Token authenticity	Server authenticity	Token privacy	Leakage resilience	# Authentications	Resistance to noise	Modeling resistance	Token authenticity	Server authenticity	Token privacy	DoS prevention	Leakage resilience	Scalability
Reference II-A	✓	×	×		∞	✓	✓	✓			✓		✓
Reference II-B	×	✓	×		∞	✓	✓		✓		✓		✓
Basic [38]	✓	×	×		*g*	▨	×	▨			▨		▨
Controlled [51]	✓	×	×		*g*	✓	✓	✓			✓		✓
Bolotnyy et al. [21]	✓	×	✓	×	*g*	×	×	▨		▨	×		▨
Öztürk et al. [130]	✓	×	×	×	∞	∼	▨	▨			×		▨
Hammouri et al. [54]	✓	×	×		∞	▨	×	▨			▨		▨
Kulseng et al. [94]	✓	✓	✓	×	∞	×	▨	×	×	×	×		▨
SHIC [143]	✓	×	×		*g*	✓	✓	∼			✓		✓
Sadeghi et al. [148]	✓	×	✓	∼	∞	✓	✓	✓		✓	✓	∼	×
Reconfiguration [80]	✓	×	∼	∼	*g*	▨	×	▨		▨	×	▨	▨
Reverse I [171]	✓	✓	×		∞	✓	∼	∼	∼		✓		✓
Reverse II [110]	✓	✓	×		∞	✓	✓	✓	✓		✓		✓
Converse [85]	×	✓	×		∞	▨	▨		×		▨		▨
Lee et al. I [101]	✓	✓	×	×	*g*	×	▨	▨	▨		×		▨
Jin et al. [73]	✓	✓	✓	∼	∞	×	▨	▨	×	▨	×	×	×
Slender [140]	✓	×	×		∞	✓	∼	∼			✓		✓
Xu and He [179]	✓	✓	✓	×	*g*	×	∼	▨	×	▨	▨		▨
He and Zou [57]	✓	✓	✓	✓	∞	×	▨	×	×	×	×	×	▨
Jung and Jung [75]	✓	✓	✓	✓	∞	×	▨	▨	▨	▨	×	×	▨
Lee et al. II [102]	✓	✓	×	✓	∞	×	▨	▨	▨		×	×	▨
Noise bifur. [190]	✓	×	×		∞	✓	∼	✓			✓		✓
System of PUFs [91]	✓	×	×		*g*	▨	×	▨			▨		▨
Lockdown I [188]	✓	×	×		*g*	✓	∼	✓			✓		✓
Lockdown II [188]	✓	✓	×		*g*	✓	∼	✓	✓		✓		✓
Protocol			Claims						Our analysis				

excellent resistance to both machine learning and protocol attacks, but is unfortunately not lightweight. Within this category, reference protocol II-A can be understood as the weak PUF variant of controlled PUFs [51]. The protocol of Sadeghi et al. [148] is also retained, despite the restrictions on the physical compromise of its NVM contents. The revised reverse fuzzy extractor protocol [110] is tentatively retained, given that the repercussions of the refuted reusability claim might not be catastrophic.

- Retained protocols of the second category avoid the use of cryptographic algorithms and error-correcting codes in order to facilitate a lightweight implementation. This includes the slender PUF [140] and noise bifurcation [190] protocols. The former proposal is retained despite the exploitation of its PRNG, given that a redesign of this building block might easily offer a fix. Unfortunately, the protective mechanisms against machine learning attacks rely on the use of physically secure TRNG and are hard to validate. The security of the PUF lockdown protocol [188] is somewhat easier to validate, given that there is an upper bound on the number of CRPs that an attacker can obtain. Moreover, it resolves the issues of the SHIC protocol [143].

Despite having expanded the scope of our survey from 8 to 19 to 21 PUF-based authentication protocols, it is hard to *catch 'em all.* A comprehensive analysis of the missing proposals, six of which are introduced next, is suggested as further work. There is the protocol of Tuyls and Škorić [169], which has later been revised by Rührmair et al. [142] so as to resolve a security issue. The protocol of Aysu et al. [7] relies on a reverse fuzzy extractor [171, 110]. Unfortunately, it inherits again the fallout of the refuted reusability claim in Section 4.3.6. The protocols of Ye et al. [180] and Gao et al. [48] bear a resemblance to the TRNG-based obfuscation protocols [119, 140, 190] in our survey. Unfortunately, the latter proposal imposes the restriction of a passive, eavesdropping attacker, which is in conflict with the distributed, wireless nature of its intended RFID-like applications. Finally, there is the protocol of Che et al. [31], which again avoids the use of a cryptographic algorithm.

5.5 Conclusion

We analyzed the security and usability of 21 PUF-based authentication protocols. In order to treat all proposals equally, we devised a unified notation as well as a transparent framework of protocol requirements. Numerous security and practicality issues have been revealed. In fact, only the minority of the proposals is deemed suitable for consideration and potential implementation by a system provider. PUF-based key generation, where a keyed cryptographic algorithm performs the authentication, remains the preferred approach for devices that can afford the resources. Such protocols can also be extended easily to security needs other than entity authentication, e.g., message confidentiality and integrity. Nevertheless, we also selected existing designs and even contributed to a new design of a PUF-based authentication protocol for devices that cannot afford cryptographic algorithms. Hopefully, the observations and lessons learned in this chapter can facilitate future protocol design.

Chapter 6

Conclusions and Further Work

We revisit several contributions of this PhD thesis while providing suggestions for further work. Three crucial themes that run across Chapters 3, 4, and 5 are elaborated.

6.1 PUFs versus Embedded NVM

The motivation for using PUFs hinges on the assumption that embedded OTP/MTP NVM is physically insecure. The problem is in fact shifted to various other building blocks that are all assumed to be physically secure instead. However, as is clear from the numerous physical attacks on both PUFs and HDAs that have been published from the early 2010s onwards, the latter two primitives are not exactly immune either. Regardless of our own experimental work on the exploitation of both noise and environmental perturbations in Chapter 3, the sheer scale of the attacks questions the existential motivation of PUF technology. Further studies would be required in order to properly compare the inherent resistance of PUF–HDA combos and NVM to physical attacks. In the ideal case, the footprint and effectiveness of their respective countermeasures would be analyzed more frequently as well.

Moreover, we observe that all surveyed authentication protocols in Chapter 5 stand or fall by the physical security of their PUF, to the extent that another newly revealed threat does not yet *spoil the party*. The development of a *two-*

factor protocol, which remains secure if either its PUF or its NVM is comprised, is hence suggested as further work. We also point out in Chapter 4 that numerous HDAs are vulnerable to the manipulation of their helper data, which has largely been overlooked in the literature. It seems artificial to us that an attacker would be able to physically obtain read-access to the NVM storing helper data, but therefore no write-access. Moreover, several protocols in Chapter 5 transmit helper data over an insecure communication channel and hence facilitate its manipulation.

6.2 Min-Entropy

A staggering number of PUF circuits has been proposed in the literature. The uniqueness of their challenge-response behavior is typically evaluated through statistical tests that detect non-uniformities, and which require straightforward computations only. Most frequently, this includes the inter-device distance, and occasionally also the NIST test suite. Unfortunately, numerical results are hard to interpret and compare, and correlations may easily go undetected, especially for strong PUFs. Entropy would be the ultimate uniqueness metric, and has direct application to the fuzzy extraction of a secret key, but its estimation is hard and therefore mostly neglected. In Chapter 3, we propose three novel methods that derive proven upper bounds on the min-entropy of several strong PUFs. These bounds evaluate to surprisingly low numbers, which implies that the CRPs are not as uniquely tied to a given device as is often assumed. The authors of the original reverse fuzzy extractor protocol in Chapter 5, for example, overestimate the min-entropy of their Arbiter PUF with at least a factor 9.

Despite our newly developed theory, estimating the entropy of a PUF circuit remains fairly unexplored territory. Other authors might improve upon our bounds and expand the scope of the analysis to a larger set of PUF designs. Moreover, the proven existence of an entropy deficiency does not automatically provide the attacker with an accelerated brute-force search. We invite other authors to help bridging this gap.

On top of the inherent entropy deficiencies of a PUF, HDAs cause an additional loss. Our survey in Chapter 4 points out that the losses of most HDAs can only be derived for nearly ideal distributions, in casu, i.i.d. response bits. Although this assumption might approximately hold for top-quality SRAM-like PUFs, most designs and implementations are not covered. Even within an ideal universe of i.i.d. response bits, security claims are not necessarily satisfied. For example, the original soft-decision decoding scheme where bit error rates are stored as helper data exhibits a larger min-entropy loss than what has been acknowledged by its authors. Fuzzy extractors are not restricted to i.i.d. response bits and can handle all distributions of which the min-entropy is sufficiently high. We derived improved bounds on their min-entropy loss such that a large variety of low-entropy distributions can be handled more effectively. The gains were demonstrated for SRAM-like PUFs with either i.i.d. or spatially correlated response bits. We invite other authors to help improving the bounds and to investigate distributions that our theory does not support.

6.3 Literature Disconnect

There is a considerable disconnect among the individual proposals of highly related PUF-based systems. Most notably, the 21 surveyed publications that propose a PUF-based authentication protocol in Chapter 5 hardly refer and draw comparisons to one another. There are exceptions to the rule, but in most cases, proper motivation that urges the need for a new protocol is lacking. Moreover, the opportunity to learn from mistakes that happened in the past is missed. Regardless of their technical merits and regardless of whether their security claims will stand the test of time, this situation improves for a few more recent proposals that are motivated by our survey results, e.g., the protocols of Aysu et al. [7] and Yu et al. [188]. The latter proposal, for which the author of this PhD thesis was a contributor, actually aims to resolve the issues of two previously published protocols.

Also for the achievement of other goals, there is a never ending stream of remarkably similar proposals that often lack an in-depth comparison to what already exists. Consider for example the large number of slightly differing PUFs that are all based on a reconfigurable RO, as cited in Chapter 3. The debiasing schemes in Chapter 4 comprehend another example: skewed SMV, pattern matching, IBS, and the von Neumann adaptations were all proposed in an autonomous fashion. It is also more than once that the retention and disposal of respectively stable and unstable response bits has been proposed as a novel HDA. In a research field where many designs are proposed, but only few are thoroughly analyzed and compared by independent parties, we aimed to restore the balance. For future work, we encourage others to only propose a new design, if for a given set of constraints, and after exhaustive analysis of the state-of-the-art, it turns out that an overall more favorable alternative does not yet exist.

Although duplicates and motivational issues cause no harm, this differs for the designers of PUF-based systems who do not understand and/or acknowledge what exactly a PUF is. Wrong and unrealistic assumptions seemingly spread like wildfire and stall the progress of research. We are primarily referring here to the protocols in Chapter 5 where the PUF is at least partially assumed to behave as a random oracle. There are three notable differences though. First, the responses of a PUF are noisy rather than deterministic. Second, the responses are not sampled from a uniform distribution, and considerable correlations among CRPs may exist. For strong PUFs, this implies the existence of machine learning attacks. Third, challenges and responses are not of arbitrary size. Most strong PUF designs produce a single response bit only, for example.

Bibliography

[1] R. Ahlswede and I. Csiszár. Common randomness in information theory and cryptography – Part I: secret sharing. *IEEE Transactions on Information Theory*, 39(4):1121–1132, 1993. DOI: `10.1109/18.243431`.

[2] M. Ahsanullah, V. Nevzorov, and M. Shakil. *Distributions of order statistics.* In *An Introduction to Order Statistics.* Atlantis Press, 2013, pages 15–21. DOI: `10.2991/978-94-91216-83-1_2`.

[3] R. J. Anderson and M. G. Kuhn. Tamper resistance - a cautionary note. In *2nd USENIX Workshop on Electronic Commerce (EC 1996)*, pages 1–11. USENIX, Nov. 1996.

[4] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th Conference on the Theory and Application of Cryptology and Information Security*, volume 5912 of *Lecture Notes in Computer Science*, pages 685–702. Springer, Dec. 2009. DOI: `10.1007/978-3-642-10366-7_40`.

[5] L. Atzori, A. Iera, and G. Morabito. The Internet of things: A survey. *Computer Networks*, 54(15):2787–2805, Oct. 2010. DOI: `10.1016/j.comnet.2010.05.010`.

[6] A. Aysu, E. Gulcan, D. Moriyama, and P. Schaumont. Compact and low-power ASIP design for lightweight PUF-based authentication protocols. *IET Information Security*, 10(5):232–241, Sept. 2016. DOI: `10.1049/iet-ifs.2015.0401`.

[7] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, and M. Yung. End-to-end design of a PUF-based privacy preserving authentication protocol. In T. Güneysu and H. Handschuh, editors, *17th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015)*, volume 9293 of *Lecture Notes in Computer Science*, pages 556–576. Springer, Sept. 2015. DOI: `10.1007/978-3-662-48324-4_28`.

[8] B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. Leftover hash lemma, revisited. In P. Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, volume 6841 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Aug. 2011. DOI: `10.1007/978-3-642-22792-9_1`.

[9] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, Feb. 2006. DOI: `10.1109/JPROC.2005.862424`.

[10] R. Bassil, W. El-Beaino, A. I. Kayssi, and A. Chehab. A PUF-based ultra-lightweight mutual-authentication RFID protocol. In *6th Conference for Internet Technology and Secured Transactions (ICITST 2011)*, pages 495–499. IEEE, Dec. 2011. DOI: `10.13140/2.1.1965.8882`.

[11] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer. Electromagnetic analysis on ring oscillator-based true random number generators. In *International Symposium on Circuits and Systems (ISCAS 2013)*, pages 1954–1957. IEEE, May 2013. DOI: `10.1109/ISCAS.2013.6572251`.

[12] G. T. Becker. On the pitfalls of using arbiter-PUFs as building blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1295–1307, Aug. 2015. DOI: `10.1109/TCAD.2015.2427259`.

[13] G. T. Becker. The gap between promise and reality: on the insecurity of XOR arbiter PUFs. In T. Güneysu and H. Handschuh, editors, *17th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015)*, volume 9293 of *Lecture Notes in Computer Science*, pages 535–555. Springer, Sept. 2015. DOI: `10.1007/978-3-662-48324-4_27`.

[14] G. T. Becker and R. Kumar. Active and passive side-channel attacks on delay based PUF designs. IACR Cryptology ePrint Archive, Report 2014/287, Apr. 2014. `http://eprint.iacr.org/2014/287`.

[15] G. T. Becker, A. Wild, and T. Güneysu. Security analysis of index-based syndrome coding for PUF-based key generation. In *8th Symposium on Hardware Oriented Security and Trust (HOST 2015)*, pages 20–25. IEEE, May 2015. DOI: `10.1109/HST.2015.7140230`.

[16] M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, Dec. 2000. DOI: `10.1007/3-540-44448-3_41`.

[17] C. H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. Practical quantum oblivious transfer. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 351–366. Springer, Aug. 1991. DOI: `10.1007/3-540-46766-1_29`.

[18] M. Bhargava and K. Mai. A high reliability PUF using hot carrier injection based response reinforcement. In G. Bertoni and J.-S. Coron, editors, *15th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2013)*, volume 8086 of *Lecture Notes in Computer Science*, pages 90–106. Springer, Aug. 2013. DOI: `10.1007/978-3-642-40349-1_6`.

[19] M. Bhargava and K. Mai. An efficient reliable PUF-based cryptographic key generator in 65nm CMOS. In *Design, Automation & Test in Europe Conference & Exhibition (DATE 2014)*, pages 1–6. IEEE, Mar. 2014. DOI: `10.7873/DATE.2014.083`.

[20] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede. SPONGENT: the design space of lightweight cryptographic hashing. *IEEE Transactions on Computers*, 62(10):2041–2053, Oct. 2013. DOI: `10.1109/TC.2012.196`.

[21] L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in RFID systems. In *5th Conference on Pervasive Computing and Communications (PerCom 2007)*, pages 211–220. IEEE, Mar. 2007. DOI: `10.1109/PERCOM.2007.26`.

[22] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on FPGAs. In E. Oswald and P. Rohatgi, editors, *10th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2008)*, volume 5154 of *Lecture Notes in Computer Science*, pages 181–197. Springer, Aug. 2008. DOI: `10.1007/978-3-540-85053-3_12`.

[23] X. Boyen. Reusable cryptographic fuzzy extractors. In *11th Conference on Computer and Communications Security (CCS 2004)*, pages 82–91. ACM, Oct. 2004. DOI: `10.1145/1030083.1030096`.

[24] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 147–163. Springer, May 2005. DOI: `10.1007/11426639_9`.

[25] A. Brown, P. Franken, S. Bonner, N. Dolezal, and J. Moross. Safecast: successful citizen-science for radiation measurement and communication after Fukushima. *Journal of Radiological Protection*, 36(2):82–101, June 2016. DOI: `10.1088/0952-4746/36/2/S82`.

[26] C. Brzuska, M. Fischlin, H. Schröder, and S. Katzenbeisser. Physically uncloneable functions in the universal composition framework. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011: 31st Annual Cryptology Conference*, volume 6841 of *Lecture Notes in Computer Science*, pages 51–70. Springer, Aug. 2011. DOI: `10.1007/978-3-642-22792-9_4`.

[27] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. `http://competitions.cr.yp.to/caesar.html`. Accessed: 2017-05-17.

[28] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith. Reusable fuzzy extractors for low-entropy distributions. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 117–146. Springer, May 2016. DOI: `10.1007/978-3-662-49890-3_5`.

[29] L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. DOI: `10.1016/0022-0000(79)90044-8`.

[30] P. Charpin, T. Helleseth, and V. A. Zinoviev. The coset distribution of triple-error-correcting binary primitive BCH codes. *IEEE Transactions on Information Theory*, 52(4):1727–1732, 2006. DOI: `10.1109/TIT.2006.871605`.

[31] W. Che, M. Martin, G. Pocklassery, V. K. Kajuluri, F. Saqib, and J. Plusquellic. A privacy-preserving, mutual PUF-based authentication protocol. *Cryptography*, 1(1), Nov. 2016. DOI: `10.3390/cryptography1010003`.

[32] Z. Cherif, J.-L. Danger, S. Guilley, and L. Bossuet. An easy-to-design PUF based on a single oscillator: the loop PUF. In *15th Euromicro Conference on Digital System Design (DSD 2012)*, pages 156–162. IEEE, Sept. 2012. DOI: `10.1109/DSD.2012.22`.

[33] K.-H. Chuang, E. Bury, R. Degraeve, B. Kaczer, G. Groeseneken, I. Verbauwhede, and D. Linten. Physically unclonable function using CMOS breakdown position. In *55th International Reliability Physics Symposium (IRPS 2017)*, pages 4C-1.1–4C-1.7. IEEE, Apr. 2017. DOI: `10.1109/IRPS.2017.7936312`.

[34] P. F. Cortese, F. Gemmiti, B. Palazzi, M. Pizzonia, and M. Rimondini. Efficient and practical authentication of PUF-based RFID tags in supply chains. In *Conference on RFID-Technology and Applications (RFID-TA 2010)*, pages 182–188. IEEE, June 2010. DOI: `10.1109/RFID-TA.2010.5529941`.

[35] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002, pages 1–238. DOI: `10.1007/978-3-662-04722-4`.

[36] A. Das, Ü. Kocabaş, A.-R. Sadeghi, and I. Verbauwhede. PUF-based secure test wrapper design for cryptographic SoC testing. In *15th Conference & Exhibition on Design, Automation & Test in Europe (DATE 2012)*, pages 866–869. IEEE, Mar. 2012. DOI: `10.1109/DATE.2012.6176618`.

[37] G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *19th Symposium on Security and Privacy*, pages 148–157. IEEE, May 1998. DOI: `10.1109/SECPRI.1998.674831`.

[38] S. Devadas, G. E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. Design and implementation of PUF-based "unclonable" RFID ICs for anti-counterfeiting and security applications. In *2nd Conference on RFID*, pages 58–64. IEEE, Apr. 2008. DOI: `10.1109/RFID.2008.4519377`.

[39] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, Mar. 2008. DOI: `10.1137/060651380`.

[40] Y. Dodis, L. Reyzin, and A. Smith. *Fuzzy extractors*. In *Security with Noisy Data*. Springer, 2007, pages 79–99. DOI: `10.1007/978-1-84628-984-2`.

[41] Y. Dodis, A. Shamir, N. Stephens-Davidowitz, and D. Wichs. How to eat your entropy and have it too: optimal recovery strategies for compromised RNGs. *Algorithmica*:1–37, Nov. 2016. DOI: `10.1007/s00453-016-0239-3`.

[42] M. J. Dworkin. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical report Federal Information Processing Standards 202, NIST, Aug. 2015. DOI: `10.6028/NIST.FIPS.202`.

[43] eMemory Technology Inc. `http://www.ememory.com.tw/html/index.php`. Accessed: 2017-05-17.

[44] W. Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1, 3rd Edition*. 1968.

[45] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: concrete results. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *3rd Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001)*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, May 2001. DOI: `10.1007/3-540-44709-1_21`.

[46] D. Ganta and L. Nazhandali. Easy-to-build arbiter physical unclonable function with enhanced challenge/response set. In *14th International Symposium on Quality Electronic Design (ISQED 2013)*, pages 733–738. IEEE, Mar. 2013. DOI: `10.1109/ISQED.2013.6523692`.

[47] M. Gao, K. Lai, and G. Qu. A highly flexible ring oscillator PUF. In *51st Design Automation Conference (DAC 2014)*, 89:1–89:6. IEEE, June 2014. DOI: `10.1145/2593069.2593072`.

[48] Y. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavehei, D. Abbott, and D. C. Ranasinghe. Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices. In *Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6. IEEE, Mar. 2016. DOI: `10.1109/PERCOMW.2016.7457162`.

[49] B. Gassend. *Physical Random Functions.* Master's thesis, Massachusetts Institute of Technology, 2003.

[50] B. Gassend, D. E. Clarke, M. v. Dijk, and S. Devadas. Controlled physical random functions. In *18th Annual Computer Security Applications Conference (ACSAC 2002)*, pages 149–160. IEEE, Dec. 2002. DOI: `10.1109/CSAC.2002.1176287`.

[51] B. Gassend, D. E. Clarke, M. v. Dijk, and S. Devadas. Silicon physical random functions. In *9th Conference on Computer and Communications Security*, pages 148–160. ACM, Nov. 2002. DOI: `10.1145/586110.586132`.

[52] B. Gassend, M. v. Dijk, D. E. Clarke, E. Torlak, S. Devadas, and P. Tuyls. Controlled physical random functions and applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(4), Jan. 2008. DOI: `10.1145/1284680.1284683`.

[53] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In P. Paillier and I. Verbauwhede, editors, *9th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, Sept. 2007. DOI: `10.1007/978-3-540-74735-2_5`.

[54] G. Hammouri, E. Öztürk, and B. Sunar. A tamper-proof and lightweight authentication scheme. *Pervasive and Mobile Computing*, 4(6):807–818, Dec. 2008. DOI: `10.1016/j.pmcj.2008.07.001`.

[55] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, Aug. 1999. DOI: `10.1137/S0097539793244708`.

[56] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[57] Z. He and L. Zou. High-efficient RFID authentication protocol based on physical unclonable function. In *8th Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2012)*, pages 1–4. IEEE, Sept. 2012. DOI: `10.1109/WiCOM.2012.6478536`.

[58] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert. Cloning physically unclonable functions. In *6th Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, pages 1–6. IEEE, June 2013. DOI: `10.1109/HST.2013.6581556`.

[59] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert. Breaking and entering through the silicon. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *20th Conference on Computer and Communications Security (CCS 2013)*, pages 733–744. ACM, Nov. 2013. DOI: `10.1145/2508859.2516717`.

[60] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A new RFID privacy model. In V. Atluri and C. Diaz, editors, *16th European Symposium on Research in Computer Security (ESORICS 2011)*, volume 6879 of *Lecture Notes in Computer Science*, pages 568–587. Springer, Sept. 2011. DOI: `10.1007/978-3-642-23822-2_31`.

[61] J. Hermans, R. Peeters, and J. Fan. IBIHOP: proper privacy preserving mutual RFID authentication. In C. Ma and J. Weng, editors, *Workshop on Radio Frequency Identification System Security (RFIDSec Asia 2013)*, volume 11 of *Cryptology and Information Security Series*, pages 45–56. IOS Press, Nov. 2013. DOI: `10.3233/978-1-61499-328-5-45`.

[62] M. Hiller, D. Merli, F. Stumpf, and G. Sigl. Complementary IBS: application specific error correction for PUFs. In *5th Symposium on Hardware-Oriented Security and Trust (HOST 2012)*, pages 1–6. IEEE, June 2012. DOI: `10.1109/HST.2012.6224310`.

[63] M. Hiller, M. Pehl, G. Kramer, and G. Sigl. Algebraic security analysis of key generation with physical unclonable functions. In *5th Workshop on Security Proofs for Embedded Systems (PROOFS 2016)*. Springer, Aug. 2016.

[64] M. Hiller, M. Weiner, L. R. Lima, M. Birkner, and G. Sigl. Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes. In *3rd Workshop on Trustworthy Embedded Devices (TrustED 2013)*, pages 43–54. ACM, Nov. 2013. DOI: `10.1145/2517300.2517304`.

[65] M. Hiller, M.-D. Yu, and M. Pehl. Systematic low leakage coding for physical unclonable functions. In *10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS 2015)*, pages 155–166. ACM, Apr. 2015. DOI: `10.1145/2714576.2714588`.

[66] M. Hiller, M.-D. Yu, and G. Sigl. Cherry-picking reliable PUF bits with differential sequence coding. *IEEE Transactions on Information Forensics and Security (TIFS)*, 11(9):2065–2076, Sept. 2016. DOI: `10.1109/TIFS.2016.2573766`.

[67] M. Hofer and C. Böhm. An alternative to error correction for SRAM-like PUFs. In S. Mangard and F.-X. Standaert, editors, *12th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2010)*, volume 6225 of *Lecture Notes in Computer Science*, pages 335–350. Springer, Aug. 2010. DOI: `10.1007/978-3-642-15031-9_23`.

[68] D. E. Holcomb, W. P. Burleson, and K. Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, Sept. 2009. DOI: `10.1109/TC.2008.212`.

[69] A. Huang. Hacking the PIC 18F1320. `http://www.bunniestudios.com/blog/?page_id=40`, July 2005. Accessed: 2017-05-17.

[70] T. Ignatenko, G.-J. Schrijen, B. Škorić, P. Tuyls, and F. Willems. Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method. In *International Symposium on Information Theory (ISIT 2006)*, pages 499–503. IEEE, July 2006. DOI: `10.1109/ISIT.2006.261765`.

[71] R. S. Indeck and M. W. Muller. Method and apparatus for fingerprinting magnetic media, Nov. 1994. US Patent 5365586.

[72] Intrinsic-ID. `https://www.intrinsic-id.com/`. Accessed: 2017-05-17.

[73] Y. Jin, W. Xin, H. Sun, and Z. Chen. PUF-based RFID authentication protocol against secret key leakage. In Q. Z. Sheng, G. Wang, C. S. Jensen, and G. Xu, editors, *14th Asia-Pacific Web Conference on Web Technologies and Applications (APWeb 2012)*, volume 7235 of *Lecture Notes in Computer Science*, pages 318–329. Springer, Apr. 2012. DOI: `10.1007/978-3-642-29253-8_27`.

[74] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *6th Conference on Computer and Communications Security (CCS 1999)*, pages 28–36. ACM, Nov. 1999. DOI: `10.1145/319709.319714`.

[75] S. W. Jung and S. Jung. HRP: a HMAC-based RFID mutual authentication protocol using PUF. In *27th International Conference on Information Networking (ICOIN 2013)*, pages 578–582. IEEE, Jan. 2013. DOI: `10.1109/ICOIN.2013.6496690`.

[76] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura. Cryptographic key generation from PUF data using efficient fuzzy extractors. In *16th International Conference on Advanced Communication Technology (ICACT 2014)*, pages 23–26. IEEE, Feb. 2014. DOI: `10.1109/ICACT.2014.6778915`.

[77] D. Karakoyunlu and B. Sunar. Differential template attacks on PUF enabled cryptographic devices. In *2nd Workshop on Information Forensics and Security (WIFS 2010)*, pages 1–6. IEEE, Dec. 2010. DOI: `10.1109/WIFS.2010.5711445`.

[78] S. Kardaş, M. Akgün, M. S. Kiraz, and H. Demirci. Cryptanalysis of lightweight mutual authentication and ownership transfer for RFID systems. In *Workshop on Lightweight Security Privacy: Devices, Protocols and Applications (LightSec 2011)*, pages 20–25. IEEE, Mar. 2011. DOI: `10.1109/LightSec.2011.10`.

[79] S. Kardaş, S. ÇElik, M. Yildiz, and A. Levi. PUF-enhanced offline RFID security and privacy. *Journal of Network and Computer Applications*, 35(6):2059–2067, Nov. 2012. DOI: `10.1016/j.jnca.2012.08.006`.

[80] S. Katzenbeisser, Ü. Kocabaş, V. v. d. Leest, A.-R. Sadeghi, G.-J. Schrijen, and C. Wachsmann. Recyclable PUFs: logically reconfigurable PUFs. *Journal of Cryptographic Engineering*, 1(3):177–186, Sept. 2011. DOI: `10.1007/s13389-011-0016-9`.

[81] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. PUFs: myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon. In E. Prouff and P. Schaumont, editors, *14th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 283–301. Springer, Sept. 2012. DOI: `10.1007/978-3-642-33027-8_17`.

[82] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–83, Jan. 1883.

[83] T. A. M. Kevenaar, G.-J. Schrijen, A. H. M. Akkermans, M. Damstra, P. Tuyls, and M. v. d. Veen. Robust and secure biometrics: some application examples. In *ISSE 2006 – Securing Electronic Busines Processes, Highlights of the Information Security Solutions Europe 2006 Conference*, pages 196–203. Springer, Oct. 2006. DOI: `10.1007/978-3-8348-9195-2_21`.

[84] Kilopass. `http://www.kilopass.com/`. Accessed: 2017-05-17.

[85] Ü. Kocabaş, A. Peter, S. Katzenbeisser, and A.-R. Sadeghi. Converse PUF-based authentication. In S. Katzenbeisser, E. R. Weippl, L. J. Camp, M. Volkamer, M. K. Reiter, and X. Zhang, editors, *5th Conference on Trust and Trustworthy Computing (TRUST 2012)*, volume 7344 of *Lecture Notes in Computer Science*, pages 142–158. Springer, June 2012. DOI: `10.1007/978-3-642-30921-2_9`.

[86] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, Aug. 1999. DOI: `10.1007/3-540-48405-1_25`.

[87] P. Koeberl, J. Li, A. Rajan, and W. Wu. Entropy loss in PUF-based key generation schemes: the repetition code pitfall. In *7th Symposium on Hardware-Oriented Security and Trust (HOST 2014)*, pages 44–49. IEEE, May 2014. DOI: `10.1109/HST.2014.6855566`.

[88] P. Koeberl, J. Li, and W. Wu. A spatial majority voting technique to reduce error rate of physically unclonable functions. In R. Bloem and P. Lipp, editors, *5th International Conference on Trusted Systems (INTRUST 2013)*, volume 8292 of *Lecture Notes in Computer Science*, pages 36–52. Springer, Dec. 2013. DOI: `10.1007/978-3-319-03491-1_3`.

[89] P. Koeberl, R. Maes, V. Rožić, V. v. d. Leest, E. Van der Sluis, and I. Verbauwhede. Experimental evaluation of physically unclonable functions in 65 nm CMOS. In *38th European Solid-State Circuits Conference (ESSCIRC 2012)*, pages 486–489. IEEE, Sept. 2012. DOI: `10.1109/ESSCIRC.2012.6341361`.

[90] O. Kömmerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. In *1st USENIX Workshop on Smartcard Technology (Smartcard 1999)*, pages 9–20. USENIX, May 1999.

[91] S. T. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. F. Wong. System-of-PUFs: multilevel security for embedded systems. In *12th Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2014)*, pages 1–10. IEEE, Dec. 2014. DOI: `10.1145/2656075.2656099`.

[92] H. Krawczyk. Cryptographic extraction and key derivation: the HKDF scheme. In T. Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, Aug. 2010. DOI: `10.1007/978-3-642-14623-7_34`.

[93] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. Technical report, 1997.

[94] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan. Lightweight mutual authentication and ownership transfer for RFID systems. In *29th Conference on Computer Communications (INFOCOM 2010)*, pages 251–255. IEEE, Mar. 2010. DOI: `10.1109/INFCOM.2010.5462233`.

[95] R. Kumar and W. P. Burleson. Hybrid modeling attacks on current-based PUFs. In *32nd International Conference on Computer Design (ICCD 2014)*, pages 493–496. IEEE, Oct. 2014. DOI: `10.1109/ICCD.2014.6974725`.

[96] R. Kumar and W. P. Burleson. Side-channel assisted modeling attacks on feed-forward arbiter PUFs using silicon data. In S. Mangard and P. Schaumont, editors, *11th Workshop on Radio Frequency Identification: Security and Privacy Issues (RFIDsec 2015)*, volume 9440 of *Lecture Notes in Computer Science*, pages 53–67. Springer, June 2015. DOI: `10.1007/978-3-319-24837-0_4`.

[97] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. Extended abstract: the butterfly PUF protecting IP on every FPGA. In *1st Symposium on Hardware-Oriented Security and Trust (HOST 2008)*, pages 67–70. IEEE, June 2008. DOI: `10.1109/HST.2008.4559053`.

[98] P. A. Layman, S. Chaudhry, J. G. Norman, and J. R. Thomson. Electronic fingerprinting of semiconductor integrated circuits, May 2004. US Patent 6738294.

[99] D. E. Lazich and M. Wuensche. Protection of sensitive security parameters in integrated circuits. In *Mathematical Methods in Computer Science (MMICS 2008) - Essays in Memory of Thomas Beth*, pages 157–178, Dec. 2008. DOI: `10.1007/978-3-540-89994-5_13`.

[100] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. v. Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits*, pages 176–179. IEEE, June 2004. DOI: `10.1109/VLSIC.2004.1346548`.

[101] Y. S. Lee, T. Y. Kim, and H. J. Lee. Mutual authentication protocol for enhanced RFID security and anti-counterfeiting. In *26th Conference on Advanced Information Networking and Applications Workshops (WAINA 2012)*, pages 558–563. IEEE, Mar. 2012. DOI: `10.1109/WAINA.2012.12`.

[102] Y. S. Lee, H. J. Lee, and E. Alasaarela. Mutual authentication in wireless body sensor networks (WBSN) based on physical unclonable function (PUF). In *9th International Wireless Communications and Mobile Computing Conference (IWCMC 2013)*, pages 1314–1318. IEEE, July 2013. DOI: `10.1109/IWCMC.2013.6583746`.

[103] V. v. d. Leest, B. Preneel, and E. v. d. Sluis. Soft decision error correction for compact memory-based PUFs using a single enrollment. In E. Prouff and P. Schaumont, editors, *14th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 268–282. Springer, Sept. 2012. DOI: `10.1007/978-3-642-33027-8_16`.

[104] V. v. d. Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls. Hardware intrinsic security from D flip-flops. In *5th Workshop on Scalable Trusted Computing (STC 2010)*, pages 53–62. ACM, Oct. 2010. DOI: `10.1145/1867635.1867644`.

[105] D. Lim. *Extracting Secret Keys from Integrated Circuits.* Master's thesis, Massachusetts Institute of Technology, May 2004.

[106] J.-P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In J. Kittler and M. S. Nixon, editors, *4th Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA 2003)*, volume 2688 of *Lecture Notes in Computer Science*, pages 393–402. Springer, June 2003. DOI: `10.1007/3-540-44887-X_47`.

[107] K. Lofstrom, W. R. Daasch, and D. Taylor. IC identification circuit using device mismatch. In *2000 International Solid-State Circuits Conference (ISSCC)*, pages 372–373. IEEE, Feb. 2000. DOI: `10.1109/ISSCC.2000.839821`.

[108] F. J. MacWiliams and N. J. A. Sloane. *The theory of error correcting codes.* 1977.

[109] R. Maes. An accurate probabilistic reliability model for silicon PUFs. In G. Bertoni and J.-S. Coron, editors, *15th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2013)*, volume 8086 of *Lecture Notes in Computer Science*, pages 73–89. Springer, Aug. 2013. DOI: `10.1007/978-3-642-40349-1_5`.

[110] R. Maes. *Physically Unclonable Functions: Constructions, Properies and Applications.* PhD thesis, KU Leuven, Aug. 2012.

[111] R. Maes, V. v. d. Leest, E. v. d. Sluis, and F. Willems. Secure key generation from biased PUFs: extended version. *Journal of Cryptographic Engineering*, 6(2):121–137, 2016. DOI: `10.1007/s13389-016-0125-6`.

[112] R. Maes, P. Tuyls, and I. Verbauwhede. A soft decision helper data algorithm for SRAM PUFs. In *International Symposium on Information Theory (ISIT 2009)*, pages 2101–2105. IEEE, June 2009. DOI: `10.1109/ISIT.2009.5205263`.

[113] R. Maes, P. Tuyls, and I. Verbauwhede. Intrinsic PUFs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security (WISSec 2008)*, pages 1–17, Nov. 2008.

[114] R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In C. Clavier and K. Gaj, editors, *11th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2009)*, volume 5747 of *Lecture Notes in Computer Science*, pages 332–347. Springer, Sept. 2009. DOI: `10.1007/978-3-642-04138-9_24`.

[115] R. Maes, A. Van Herrewege, and I. Verbauwhede. PUFKY: a fully functional PUF-based cryptographic key generator. In E. Prouff and P. Schaumont, editors, *14th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 302–319. Springer, Sept. 2012. DOI: `10.1007/978-3-642-33027-8_18`.

[116] A. Maiti and P. Schaumont. Improved ring oscillator PUF: an FPGA-friendly secure primitive. *Journal of Cryptology*, 24(2):375–397, Apr. 2011. DOI: `10.1007/s00145-010-9088-4`.

[117] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Techniques for design and implementation of secure reconfigurable PUFs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2(1), Mar. 2009. DOI: `10.1145/1502781.1502786`.

[118] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *International Test Conference (ITC 2008)*, pages 1–10. IEEE, Oct. 2008. DOI: `10.1109/TEST.2008.4700636`.

[119] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In *1st Security and Privacy Workshops (SPW 2012)*, pages 33–44. IEEE, May 2012. DOI: `10.1109/SPW.2012.30`.

[120] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl. Localized electromagnetic analysis of RO PUFs. In *6th Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, pages 19–24. IEEE, June 2013. DOI: `10.1109/HST.2013.6581559`.

[121] D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Semi-invasive EM attack on FPGA RO PUFs and countermeasures. In *6th Workshop on Embedded Systems Security (WESS 2011)*, 2:1–2:9. ACM, Oct. 2011. DOI: `10.1145/2072274.2072276`.

[122] D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Side-channel analysis of PUFs and fuzzy extractors. In J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, editors, *4th Conference on Trust and Trustworthy Computing (TRUST 2011)*, volume 6740 of *Lecture Notes in Computer Science*, pages 33–47. Springer, June 2011. DOI: `10.1007/978-3-642-21599-5_3`.

[123] D. Merli, F. Stumpf, and G. Sigl. Protecting PUF error correction by codeword masking. Cryptology ePrint Archive, Report 2013/334, 2013. `http://eprint.iacr.org/2013/334`.

[124] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit. Invasive PUF analysis. In *10th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2013)*, pages 30–38. IEEE, Aug. 2013. DOI: `10.1109/FDTC.2013.19`.

[125] J. v. Neumann. Various techniques used in connection with random digits. In *Monte Carlo Method.* Volume 12, Applied Mathematics, pages 36–38. National Bureau of Standards Series, 1951.

[126] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, Feb. 1996. DOI: `10.1006/jcss.1996.0004`.

[127] NSA. Commercial national security algorithm suite and quantum computing FAQ. `https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf`. Accessed: 2017-05-17.

[128] NSCore. `http://www.nscore.com/`. Accessed: 2017-05-17.

[129] M. A. Orumiehchiha, J. Pieprzyk, and R. Steinfeld. Practical attack on NLM-MAC scheme. *Information Processing Letters*, 114(10):547–550, Oct. 2014. DOI: `10.1016/j.ipl.2014.04.010`.

[130] E. Öztürk, G. Hammouri, and B. Sunar. Towards robust low cost authentication for pervasive devices. In *6th Conference on Pervasive Computing and Communications (PerCom 2008)*, pages 170–178. IEEE, Mar. 2008. DOI: `10.1109/PERCOM.2008.54`.

[131] R. S. Pappu. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, Mar. 2001.

[132] Z. S. Paral and S. Devadas. Reliable and efficient PUF-based key generation using pattern matching. In *4th Symposium on Hardware-Oriented Security and Trust (HOST 2011)*, pages 128–133. IEEE, June 2011. DOI: `10.1109/HST.2011.5955010`.

[133] Z. S. Paral and S. Devadas. Reliable PUF value generation by pattern matching, July 2012. United States Patent Application 20120183135.

[134] J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, Sept. 2001. DOI: `10.1007/3-540-45418-7_17`.

[135] D. C. Ranasinghe, D. W. Engels, and P. H. Cole. Security and privacy: modest proposals for low-cost RFID systems. In *Auto-ID Labs Research Workshop*, 2004.

[136] A. Rényi. On measures of entropy and information. In *4th Berkeley Symposium on Mathematical Statistics and Probability*, pages 547–561, 1960.

[137] L. Reyzin. Entropy Loss is Maximal for Uniform Inputs. Technical report BUCS-TR-2007-011, Department of Computer Science, Boston University, Sept. 2007, pages 1–3.

[138] O. Rioul, P. Solé, S. Guilley, and J.-L. Danger. On the entropy of physically unclonable functions. In *Symposium on Information Theory (ISIT 2016)*, pages 2928–2932. IEEE, July 2016. DOI: `10.1109/ISIT.2016.7541835`.

[139] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978. DOI: `10.1145/359340.359342`.

[140] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas. Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching. *IEEE Transactions on Emerging Topics in Computing*, 2(1):37–49, Mar. 2014. DOI: `10.1109/TETC.2014.2300635`.

[141] V. Rožić. *Circuit-Level Optimizations for Cryptography.* PhD thesis, KU Leuven, Sept. 2016.

[142] U. Rührmair, C. Jaeger, and M. Algasinger. An attack on PUF-based session key exchange and a hardware-based countermeasure: erasable PUFs. In G. Danezis, editor, *Financial Cryptography and Data Security - 15th International Conference, FC 2011*, volume 7035 of *Lecture Notes in Computer Science*, pages 190–204. Springer, Feb. 2011. DOI: `10.1007/978-3-642-27576-0_16`.

[143] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba. Applications of high-capacity crossbar memories in cryptography. *IEEE Transactions on Nanotechnology*, 10(3):489–498, May 2011. DOI: `10.1109/TNANO.2010.2049367`.

[144] U. Rührmair, J. Sölter, and F. Sehnke. On the foundations of physical unclonable functions. Cryptology ePrint Archive, Report 2009/277, June 2009. `http://eprint.iacr.org/2009/277`.

[145] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, Nov. 2013. DOI: `10.1109/TIFS.2013.2279798`.

[146] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. P. Burleson. Efficient power and timing side channels for physical unclonable functions. In L. Batina and M. Robshaw, editors, *16th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2014)*, volume 8731 of *Lecture Notes in Computer Science*, pages 476–492. Springer, Sept. 2014. DOI: `10.1007/978-3-662-44709-3_26`.

[147] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, and L. E. Bassham. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical report Special Publication 800-22 Revision 1a, NIST, Apr. 2010.

[148] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. Enhancing RFID security and privacy by physically unclonable functions. In *Towards Hardware-Intrinsic Security - Foundations and Practice*, pages 281–305. Springer, Oct. 2010. DOI: `10.1007/978-3-642-14452-3_13`.

[149] M. Safkhani, N. Bagheri, and M. Naderi. Security analysis of a PUF based RFID authentication protocol. *IACR Cryptology ePrint Archive*, 2011:704, Dec. 2011. `http://eprint.iacr.org/2011/704`.

[150] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen. A multiplexer based arbiter PUF composition with enhanced reliability and security. Cryptology ePrint Archive, Report 2016/1031, Nov. 2016. `http://eprint.iacr.org/2016/1031`.

[151] D. P. Sahoo, P. H. Nguyen, D. Mukhopadhyay, and R. S. Chakraborty. A case of lightweight PUF constructions: cryptanalysis and machine learning attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 34(8):1334–1343, Aug. 2015. DOI: `10.1109/TCAD.2015.2448677`.

[152] D. Schellekens. *Design and Analysis of Trusted Computing Platforms*. PhD thesis, KU Leuven, 2013. Bart Preneel (promotor).

[153] J. Schneider and D. Schröder. Foundations of reconfigurable PUFs. In T. Malkin, V. Kolesnikov, A. B. Lewko, and M. Polychronakis, editors, *13th Conference on Applied Cryptography and Network Security (ACNS 2015)*, volume 9092 of *Lecture Notes in Computer Science*, pages 579–594. Springer, June 2015. DOI: `10.1007/978-3-319-28166-7_28`.

[154] Sidense Corp. `https://www.sidense.com/`. Accessed: 2017-05-17.

[155] Silicon Storage Technology, Inc. (SST). `http://www.sst.com/`. Accessed: 2017-05-17.

[156] P. Simons, E. v. d. Sluis, and V. v. d. Leest. Buskeeper PUFs, a promising alternative to D flip-flop PUFs. In *5th Symposium on Hardware-Oriented Security and Trust (HOST 2012)*, pages 7–12. IEEE, June 2012. DOI: `10.1109/HST.2012.6224311`.

[157] B. Škorić, P. Tuyls, and W. Ophey. Robust key extraction from physical uncloneable functions. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *3rd Conference on Applied Cryptography and Network Security (ACNS 2005)*, volume 3531 of *Lecture Notes in Computer Science*, pages 407–422. Springer, June 2005. DOI: `10.1007/11496137_28`.

[158] S. P. Skorobogatov. Semi-invasive attacks - A new approach to hardware security analysis. Technical report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, Apr. 2005.

[159] R. P. Stanley. A survey of alternating permutations. arXiv ePrint Archive, Report 0912.4240v1, 2009.

[160] Y. Su, J. Holleman, and B. P. Otis. A 1.6pJ/bit 96% stable chip-ID generating circuit using process variations. In *2007 International Solid-State Circuits Conference (ISSCC)*, pages 406–611. IEEE, Feb. 2007. DOI: `10.1109/ISSCC.2007.373466`.

[161] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *44th Design Automation Conference (DAC 2007)*, pages 9–14. IEEE, June 2007. DOI: `10.1145/1278480.1278484`.

[162] G. E. Suh, C. W. O'Donnell, and S. Devadas. Aegis: A single-chip secure processor. *IEEE Design & Test of Computers*, 24(6):570–580, Dec. 2007. DOI: `10.1109/MDT.2007.179`.

[163] S. Tajik, E. Dietz, S. Frohmann, H. Dittrich, D. Nedospasov, C. Helfmeier, J.-P. Seifert, C. Boit, and H.-W. Hübers. Photonic side-channel analysis of arbiter PUFs. *Journal of Cryptology*:1–22, Apr. 2016. DOI: `10.1007/s00145-016-9228-6`.

[164] S. Tajik, H. Lohrke, F. Ganji, J.-P. Seifert, and C. Boit. Laser fault attack on physically unclonable functions. In *12th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2015)*, pages 85–96. IEEE, Sept. 2015. DOI: `10.1109/FDTC.2015.19`.

[165] J. Tobisch and G. T. Becker. On the scaling of machine learning attacks on PUFs with application to noise bifurcation. In S. Mangard and P. Schaumont, editors, *RFIDSec 2015: Radio Frequency Identification*, volume 9440 of *Lecture Notes in Computer Science*, pages 17–31. Springer, June 2015. DOI: `10.1007/978-3-319-24837-0_2`.

[166] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G.-J. Schrijen, A. M. Bazen, and R. N. J. Veldhuis. Practical biometric authentication with template protection. In T. Kanade, A. Jain, and N. K. Ratha, editors, *5th Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA 2005)*, volume 3546 of *Lecture Notes in Computer Science*, pages 436–446. Springer, July 2005. DOI: `10.1007/11527923_45`.

[167] P. Tuyls and L. Batina. RFID-tags for anti-counterfeiting. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 115–131. Springer, Feb. 2006. DOI: `10.1007/11605805_8`.

[168] P. Tuyls, G.-J. Schrijen, B. Škorić, J. v. Geloven, N. Verhaegh, and R. Wolters. Read-proof hardware from protective coatings. In L. Goubin and M. Matsui, editors, *8th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2006)*, volume 4249 of *Lecture Notes in Computer Science*, pages 369–383. Springer, Oct. 2006. DOI: `10.1007/11894063_29`.

[169] P. Tuyls and B. Škorić. Strong authentication with physical unclonable functions. In M. Petkovic and W. Jonker, editors, *Security, Privacy, and Trust in Modern Data Management*, Data-Centric Systems and Applications, pages 133–148. Springer, 2007. DOI: `10.1007/978-3-540-69861-6_10`.

[170] A. Van Herrewege. *Lightweight PUF-based Key and Random Number Generation.* PhD thesis, KU Leuven, 2015. Ingrid Verbauwhede (promotor).

[171] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. Reverse fuzzy extractors: enabling lightweight mutual authentication for PUF-enabled RFIDs. In A. D. Keromytis, editor, *16th Conference on Financial Cryptography and Data Security (FC 2012)*, volume 7397 of *Lecture Notes in Computer Science*, pages 374–389. Springer, Feb. 2012. DOI: `10.1007/978-3-642-32946-3_27`.

[172] A. Van Herrewege, V. v. d. Leest, A. Schaller, S. Katzenbeisser, and I. Verbauwhede. Secure PRNG seeding on commercial off-the-shelf microcontrollers. In *3rd Workshop on Trustworthy Embedded Devices (TrustED 2013)*, pages 55–64. ACM, Nov. 2013. DOI: `10.1145/2517300.2517306`.

[173] Verayo. `http://www.verayo.com/`. Accessed: 2017-05-17.

[174] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, Apr. 1967. DOI: `10.1109/TIT.1967.1054010`.

[175] Y. Wang, S. Rane, S. C. Draper, and P. Ishwar. A theoretical analysis of authentication, privacy, and reusability across secure biometric systems. *IEEE Transactions on Information Forensics and Security*, 7(6):1825–1840, Dec. 2012. DOI: `10.1109/TIFS.2012.2210215`.

[176] A. F. Webster and S. E. Tavares. On the design of S-boxes. In H. C. Williams, editor, *Advances in Cryptology - CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 523–534. Springer, Aug. 1985. DOI: `10.1007/3-540-39799-X_41`.

[177] Wikipedia. Random permutation statistics. `https://en.wikipedia.org/wiki/Random_permutation_statistics`. Accessed: 2017-05-17.

[178] X. Xu, U. Rührmair, D. E. Holcomb, and W. P. Burleson. Security evaluation and enhancement of bistable ring PUFs. In S. Mangard and P. Schaumont, editors, *11th Workshop on RFID Security (RFIDsec 2015)*, volume 9440 of *Lecture Notes in Computer Science*, pages 3–16. Springer, June 2015. DOI: `10.1007/978-3-319-24837-0_1`.

[179] Y. Xu and Z. He. Design of a security protocol for low-cost RFID. In *8th Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2012)*, pages 1–3. IEEE, Sept. 2012. DOI: `10.1109/WiCOM.2012.6478482`.

[180] J. Ye, Y. Hu, and X. Li. RPUF: physical unclonable function with randomized challenge to resist modeling attack. In *1st Asian Hardware Oriented Security and Trust Symposium (AsianHOST 2016)*, pages 1–6. IEEE, Dec. 2016. DOI: `10.1109/AsianHOST.2016.7835567`.

[181] C.-E. Yin and G. Qu. Improving PUF security with regression-based distiller. In *50th Design Automation Conference (DAC 2013)*, pages 1–6. IEEE, May 2013. DOI: `10.1145/2463209.2488960`.

[182] C.-E. Yin and G. Qu. LISA: maximizing RO PUF's secret extraction. In *3th Symposium on Hardware-Oriented Security and Trust (HOST 2010)*, pages 100–105. IEEE, June 2010. DOI: `10.1109/HST.2010.5513105`.

[183] H. Yu, H. W. Leong, H. Hinkelmann, L. Möller, M. Glesner, and P. Zipf. Towards a unique FPGA-based identification circuit using process variations. In *Conference on Field Programmable Logic and Applications (FPL 2009)*, pages 397–402. IEEE, Aug. 2009. DOI: `10.1109/FPL.2009.5272255`.

[184] H. Yu, P. H. W. Leong, and Q. Xu. An FPGA chip identification generator using configurable ring oscillators. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(12):2198–2207, 2012. DOI: `10.1109/TVLSI.2011.2173770`.

[185] M.-D. Yu. Turn FPGAs into "key" players in the cryptographics field. `http://electronicdesign.com/fpgas/turn-fpgas-key-players-cryptographics-field`, July 2009. Electronic Design Magazine.

[186] M.-D. Yu and S. Devadas. Recombination of physical unclonable functions. In *35th Government Microcircuit Applications and Critical Technology Conference (GOMACTech 2010)*, pages 1–4, Mar. 2010.

[187] M.-D. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design & Test of Computers*, 27(1):48–65, Jan. 2010. DOI: `10.1109/MDT.2010.25`.

[188] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems (TMSCS)*, 2(3):146–159, July 2016. DOI: `10.1109/TMSCS.2016.2553027`.

[189] M.-D. Yu, D. M'Raïhi, R. Sowell, and S. Devadas. Lightweight and secure PUF key storage using limits of machine learning. In B. Preneel and T. Takagi, editors, *13th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2011)*, volume 6917 of *Lecture Notes in Computer Science*, pages 358–373. Springer, Sept. 2011. DOI: `10.1007/978-3-642-23951-9_24`.

[190] M.-D. Yu, D. M'Raïhi, I. Verbauwhede, and S. Devadas. A noise bifurcation architecture for linear additive physical functions. In *7th Symposium on Hardware-Oriented Security and Trust (HOST 2014)*, pages 124–129. IEEE, May 2014. DOI: `10.1109/HST.2014.6855582`.

[191] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koeberl, and A.-R. Sadeghi. Remanence decay side-channel: the PUF case. *IEEE Transactions on Information Forensics and Security*, 11(6):1106–1116, June 2016. DOI: `10.1109/TIFS.2015.2512534`.

Curriculum Vitae

The education and work experience of Jeroen Delvaux is as follows:

2013 - 2017	**Doctoral Researcher**. Shanghai Jiao Tong University, China. Research group LoCCS.
2012 - 2017	**Doctoral Researcher**. KU Leuven, Belgium. Research group COSIC. PhD grant by the Flemish government during 48 months.
2011 - 2012	**Postgraduate in Biomedical Engineering**. KU Leuven, Belgium. *Magna cum laude.*
2010 - 2012	**Research & Methods Engineer**. KLA-Tencor, ICOS division, Belgium. Design and testing of image processing algorithms.
2008 - 2010	**Master in Electrical Engineering**. Option: Integrated Electronics. KU Leuven, Belgium. *Magna cum laude.*
2005 - 2008	**Bachelor in Electrical Engineering**. Minor: Computer Science. KU Leuven, Belgium. *Cum laude.*

List of Publications

International Journals

[1] J. Delvaux, D. Gu, D. Schellekens, and I. 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, June 2015. DOI: `10.1109/TCAD.2014.2370531`.

[2] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede. A survey on lightweight entity authentication with strong PUFs. *ACM Computing Surveys*, 48(2):26:1–26:42, Oct. 2015. DOI: `10.1145/2818186`.

[3] J. Delvaux and I. Verbauwhede. Fault injection modeling attacks on 65nm arbiter and RO sum PUFs via environmental changes. *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS I)*, 61(6):1701–1713, June 2014. DOI: `10.1109/TCSI.2013.2290845`.

[4] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems (TMSCS)*, 2(3):146–159, July 2016. DOI: `10.1109/TMSCS.2016.2553027`.

International Conferences and Workshops

[1] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Secure lightweight entity authentication with strong PUFs: mission impossible? In L. Batina and M. Robshaw, editors, *16th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2014)*, volume 8731 of *Lecture Notes in Computer Science*, pages 451–475. Springer, Sept. 2014. DOI: `10.1007/978-3-662-44709-3_25`.

[2] J. Delvaux, D. Gu, and I. Verbauwhede. Upper bounds on the min-entropy of RO sum, arbiter, feed-forward arbiter, and S-ArbRO PUFs. In *1st Asian Hardware Oriented Security and Trust Symposium (AsianHOST 2016)*, pages 1–6. IEEE, Dec. 2016. DOI: `10.1109/AsianHOST.2016.7835572`.

[3] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M.-D. Yu. Efficient fuzzy extraction of PUF-induced secrets: theory and applications. In B. Gierlichs and A. Y. Poschmann, editors, *18th Conference on Cryptographic Hardware and Embedded Systems (CHES 2016)*, volume 9813 of *Lecture Notes in Computer Science*, pages 412–431. Springer, Aug. 2016. DOI: `10.1007/978-3-662-53140-2_20`.

[4] J. Delvaux and I. Verbauwhede. Attacking PUF-based pattern matching key generators via helper data manipulation. In J. Benaloh, editor, *Topics in Cryptology - CT-RSA 2014, The Cryptographers' Track at the RSA Conference 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 106–131. Springer, Feb. 2014. DOI: `10.1007/978-3-319-04852-9_6`.

[5] J. Delvaux and I. Verbauwhede. Key-recovery attacks on various RO PUF constructions via helper data manipulation. In *17th Design, Automation & Test in Europe Conference & Exhibition (DATE 2014)*, pages 1–6. IEEE, Mar. 2014. DOI: `10.7873/DATE.2014.085`.

[6] J. Delvaux and I. Verbauwhede. Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In *6th Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, pages 137–142. IEEE, June 2013. DOI: `10.1109/HST.2013.6581579`.

[7] S. Ghosh, J. Delvaux, L. Uhsadel, and I. Verbauwhede. A speed area optimized embedded co-processor for McEliece cryptosystem. In *23rd Conference on Application-Specific Systems, Architectures and Processors (ASAP 2012)*, pages 102–108. IEEE, July 2012. DOI: `10.1109/ASAP.2012.16`.

[8] E. J. Marinissen, Y. Zorian, M. Konijnenburg, C.-T. Huang, P.-H. Hsieh, P. Cockburn, J. Delvaux, V. Rožić, B. Yang, D. Singelée, I. Verbauwhede, C. Mayor, R. V. Rijsinge, and C. Reyes. IoT: source of test challenges. In *21th European Test Symposium (ETS 2016)*, pages 1–10. IEEE, May 2016. DOI: `10.1109/ETS.2016.7519331`.

National Conferences without Proceedings

[1] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Secure lightweight entity authentication with strong PUFs: mission impossible? In *ChinaCrypt 2014*, pages 99–119, Aug. 2014.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COMPUTER SECURITY AND INDUSTRIAL CRYPTOGRAPHY
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
jeroen.delvaux@esat.kuleuven.be
https://www.esat.kuleuven.be/cosic/