Familiarize yourself with the concept of Physical Unclonable Functions (PUFs). Pay special attention to memory (SRAM) based PUFs. Analyze existing measurements of initial SRAM contents in Atmel ATmega1284 devices (supplied by the thesis supervisor). Based on your analysis, suggest a method for constructing a PUF based on such initial SRAM contents and using it for device identification and possibly other purposes.

Czech Technical University in Prague
Faculty of Information Technology
Department of Computer Systems

Master's thesis

SRAM-Based Physical Unclonable Function on an Atmel ATmega Microcontroller

Mikhail Platonov

Supervisor: Ing. Josef Hlaváč, Ph.D

1st May 2013

Acknowledgements

I would like to thank my supervisor Josef Hlaváč for his guidance throughout this graduation project. I am also grateful to Vilena Bryleva for proofreading my thesis and providing me with useful feedback. I thank my friends for their moral and material support of my achievements. I also thank all my teachers involved in the Czech Technical University for providing me with the knowledge and thus allowing me to improve myself.

Lastly, I would like to express my deep gratitude for my family. Their daily support I feel throughout my entire study period even a world away.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60(1) of the Act.

In Prague on 1st May 2013 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Platonov, Mikhail. *SRAM-Based Physical Unclonable Function on an Atmel ATmega Microcontroller*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2013.

Abstract

Secret keys are usually stored in a nonvolatile memory, which can be hard to secure. An alternative is to generate the keys "on-the-fly" by using the inherent uniqueness of a device based on the manufacturing process variations. This is realized by Physical Unclonable Functions (PUFs).

A promising approach is to construct an intrinsic PUF based on the Static Random Access Memory, since many electronic devices have embedded SRAM. However, using a SRAM as a PUF requires stability of the SRAM cells under a wide variety of conditions, moreover the SRAM output must be unique.

The aim of this thesis is to design the PUF concept based on one of the popular Atmel AVR microcontrollers. During the work the evaluation of PUF properties of the test chips is discussed. Proposed PUF-design is suitable for using in two application scenarios: chip identification and key generation. The SRAM output is not stable enough for that various post- and pre-processing techniques are analyzed to make the concept suitable for key generation scenarios.

The major contributions of the current research is the new measurements performed on an Atmel AVR microcontroller, many researches have focused mostly on FPGA so far. This work estimates the entropy of a PUF in a 16Kbyte SRAM output in order to uniquely identify each of Atmel ATmega1284P microcontrollers. Furthermore, after applying postprocessing error correction, proposed SRAM-based PUF can be used to generate a stable 4Kbit key.

Keywords Hardware security, Physical Unclonable Function, SRAM PUF, Atmel AVR microcontroller, ATmega1284, key generation, chip identification, Hamming distance, mean value, bit error rate, error correction codes.

Abstrakt

Tajné klíče jsou zpravidla uloženy v paměti, která je nezávislá na napájení. Obsah této paměti je však obtížné zabezpečit. Alternativou je generovat klíče za běhu s využitím jedinečných vlastností konkrétního zařízení, jež jsou závislé na odchylkách ve výrobním procesu. K tomu se používají fyzicky neklonovatelné funkce – označované zkratkou PUF (Physical Unclonable Function). Slibný je přístup, kdy se ke konstrukci fyzicky neklonovatelné funkce použije statická paměť RAM, protože mnoho elektronických zařízení dispo-nuje vestavěnou pamětí SRAM. Nicméně k použití SRAM jako PUF je třeba, aby buňky paměti SRAM byly stabilní ve velkém rozsahu pracovních podmínek, výstup musí být navíc jedinečný. Cílem této práce je navrhnout PUF na mikrokontroléru z oblíbené řady Atmel AVR. V práci je diskutováno vyhodnocení vlastností testovaných čipů s ohledem na PUF. Navržená PUF je vhodná pro použití ve dvou aplikačních scénářích: pro identifikaci čipů a pro generování klíčů. Výstup získaný ze SRAM není dostatečně stabilní, proto jsou analyzovány různé techniky předzpracování a následného zpracování, aby byla metoda pouv zitelní i pro generování klíčů. Hlavním přínosem prezentovaného výzkumu jsou nová měření provedená na mikrokontroléru Atmel AVR. Výzkum se doposud zaměřoval povětšinou na obvody FPGA. V práci je odhadnuta entropie funkce PUF využívající 16Kbyte SRAM pro účely jednoznačné identifikace jednotlivých kusů mikrokontrolérů Atmel ATmega1284P. Po použití následné korekce chyb lze navrženou koncepci PUF použít i pro generování stabilního 4Kbit klíče.

Klíčová slova Bezpečnost hardwaru, fyzicky neklonovatelná funkce, PUF, SRAM, mikrokontrolér ATmega1284, generování klíčů, identifikace čipů, Hammingova vzdálenost, aritmetický průměr, bitová chybovost, samoopravné kódy.

Contents

List of Figures

List of Tables

Introduction

Motivation

Microcontrollers gain their popularity because of low cost and small size. They are widely used in electronic devices from household appliances to spacecrafts. Microcontrollers are small enough to be even built in implantable medical devices.

Security requirements are the same for all devices in the network be it a full-size computer or a microcontroller. For example data transmission over public channels, especially wireless links, requires encryption with a secret key. It means keys should be generated and stored in a secure way. Usually, keys are stored in a flash, which is a well-known security issue [1, 2].

Microcontroller, as an intelligent device, requires configurable software which can be stored in a nonvolatile memory. Keeping the configuration in plaintext mode makes it the target of cloning attacks. As soon as an adversary is able to dump the memory, the whole system can be cloned. And that is a serious problem to Intellectual Property (IP) developers up to the present day. Encryption is one solution to this problem. Encrypted software is stored in nonvolatile memory and when the microcontroller is powered up, the software is decrypted with the secret key. This protects against the counterfeiting threat but causes another problem called private key storage.

There are various application scenarios where the key storage problem is the essential part. For example, applications with smart cards, banking applications, pay television, etc. In all these cases at least a master key has to be stored securely.

On-chip flash memory in a microcontroller can be hardly considered as secure storage. One solution is physical unclonable functions (PUFs).

PUFs take advantage of manufacturing process variations to generate a unique key directly out of the chip circuitry, without the need to store it explicitly.

Proposed by Gassend et al. [3] in 2002, PUFs have become a "hot" research topic in recent years. The idea of using physical phenomena of manufacturing process variations was later applied to SRAM [4, 5]. It figured out that initial state of the SRAM cells can be used as a PUF.

The fact is that manufacturing variabilities cause small mismatches between the P-channel and N-channel transistors in a 6T-SRAM cell. As a result, each SRAM cell is biased towards '0' or '1' during powering up depending on the nature of the mismatch. Some cells never have a stable state if the mismatch is very small. Such cells define the noise of a PUF and it is the main disadvantage of SRAM-based PUFs.

There are some techniques to provide a stable output [2]: some of them require an access on hardware level, some correct errors by pre- and post-selection methods. But the most feasible approach that allows to generate a stable output is applying error correction codes on the SRAM PUF output [2, 6].

This thesis analyzes statistical properties of the power-up SRAM state of one of the popular Atmel AVR microcontrollers. Two concepts of SRAM PUF implementation are suggested. One concept is used for chip identification. The other concept uses error correction techniques in order to generate strong keys. The conclusions of the work are confirmed by practical measurements of ten ATMega 1284P microcontrollers under a wide variety of conditions.

Research goals

The current PUF researches focus on FPGA implementation [1, 4, 7–17]. The goal of the present thesis is discovering of a possibility to implement a PUF on a simple AVR-series microcontroller. According to the research, which is done on FPGA, an appropriate approach is using power-up SRAM state of microcontrollers as a PUF.

In this way, the main goals of the thesis are:

1. Determining the existing concepts of Physical Unclonable Functions. Discovering the possibilities of adapting them on the given Atmel's microcontrollers. with special attention to the literature overview of the SRAM-based PUFs implementations. Identifying disadvantages

of the existing approaches in order to avoid them in constructing our own PUF.

2. Analyzing supplied measurements of the initial SRAM contents in Atmel ATmega1284 microcontrollers. Proposing a statistical template for evaluating the PUF properties and suitability for using a device as a PUF.

3. Suggesting a method for constructing a PUF on power-up SRAM contents. Evaluating the suitability of using proposed PUF-design in main application scenarious, such as device identification, key generation and other possible purposes.

Thesis overview

The thesis is organized as follows: the first part of the work is oriented towards a literature study concerning physical unclonable functions, particular emphasis is placed on SRAM-based physical unclonable functions. The latter part contains the results of the measurements of ten ATMega 1284P microcontrollers. The thesis concludes with considering a possibility of using the ten test chips in the two PUF application scenarious: key generation and device identification. The references and the author's publications are presented at the end of the masters thesis.

Chapter two starts with explaining the basic concept of PUF and the main PUF properties. Then different PUF approaches are classified and general advantages and disadvantages of these approaches are discussed. Chapter is continued with an overview of basic applications, as well as their use cases in practice. Then a brief overview of possible attacks is given. The second part of the chapter deals with the SRAM-based PUFs concept. It covers the internal structure of a SRAM cell on six transistors and known PUF-designs on SRAM cells. Then the section on patents concerning PUFs is given. This chapter concludes with a brief overview of Atmel AVR microcontrollers followed by specification of given ATMega 1284P microcontrollers.

The third chapter deals with practical measurements and provides the results of the measurements. It contains statistical analysis of the power-up SRAM state of ten Atmel ATmega1284P microcontrollers. Since the SRAM output does not meet the PUF requirements, two following sections are devoted to error correction and preselection techniques that allows to adapt SRAM output for the PUF purpose. The chapter concludes with con-

structing PUF design for the application scenarious such as key generation and device identification.

In conclusion, the thesis summarizes the concept of the SRAM-based physical unclonable functions, recapitulates the results of the measurements, mentioning the SRAM PUF-design on an Atmel ATmega1284P microcontroller and provides suggestions for further research.

Chapter 1

State of the Art

1.1 Physical Unclonable Functions

1.1.1 PUF in general

Different definitions of a physical unclonable function (PUF) can be found in the newer literature [18–20]. In general, a PUF is a function with internal random nature that uses manufacturing variability to generate an unpredictable repsponse. Therefore the PUF response of a chip is called fingerprint.

The concept of PUF was described by Pappu in his doctoral dissertation in 2001 [21]. He used the term physical one-way function (POWF) and defined it as "a function which is easy to compute but hard to invert". Later, another interpretation of the same term became popular - physical random functions [3]. Nowadays, it is generally accepted and widely used as *physical unclonable functions (PUFs)*.

The concept of PUFs is based on the local mismatches, differences between physical components of a device due to production variability. Since the manufacturing process can not be controlled, the local mismatches of produced components are random therefore a PUF can not be replicated. Thus unclonable property is achieved.

As it follows from the name, a PUF should have some features of the functions. It means according to the input (or challenge) the function generates the output (or response). But these are not true mathematical functions since they are able to produce several output values from one input or vice-versa several input values can result in one output value. It would be better to say that a PUF is an engineering function which acts upon a physical system.

Typically, a PUF input is called a challenge and the PUF output is the response. A mapping of applied challenge to its resulting response is called a challenge-response pair (or CRP). A PUF performs this CRP mapping.

$$response = PUF(challenge)$$

A PUF implementation consists of two distinct phases. During the first phase - initialization (or enrollment), a PUF is queried and number of CRPs are stored in a database. In the second phase, generally called identification (or verification), a challenge from the database is applied to the PUF and the corresponding response is compared with the entity in the database for the initial response.

Thus, in general a PUF is not a function in the mathematical sense, and can be defined as follows:

A PUF is a physical entity which produces an output value at least in dependence of physical structures which are hard to clone [2].

1.1.2 PUF properties

The following section 1.1.3 will show how wide the collection of the known PUF approaches is today. In order to identify PUF performance of an instantiation, some general intrinsic parameters must be specified. The first four properties in the list below are necessary but not sufficient for the identification approach as a PUF suitable [22]. The rest of the properties allow to estimate the quality and the strength of the PUF:

1. *Evaluability.* If a PUF and input(challenge) are given, then the corresponding response(output) is evaluated easily. In a theoretical perspective, "easy" means within polynomial time, while in a practical perspective it refers to as little overhead as possible, in constraints of time, cost, area, power, and energy of an integrated chip.

2. *Reproducibility.* The same challenge and the same PUF should produce the same response by querying the PUF multiple times. A possible error in the response should be small enough to correct it. The property differentiates PUFs from true random number generators (TRNGs).

3. *Physical unclonability.* Physical unclonability is the core PUF property. A PUF, which can not be reproduced in the manufacture way, is called manufacturer resistant. There are no two PUFs with identical challenge-response behavior.

4. *Mathematical unclonability.* There is no such mathematical apparatus which allows to construct probabilistic algorithm in order to predict the PUF response or to map the known challenges to the known responses.

5. *Opaqueness.* It shall be hard to analyze the physical structure of the PUF in order to replicate a PUFs challenge-response behavior.

6. *Identity.* The property identity is followed by physical unclonability. A statistical measurement which characterizes the identity (or uniqueness) of PUFs is an inter-distance histogram, summarized by its average value.

7. *Unpredictability.* Given a PUF and a set of challenge-response pairs for this PUF don't allow to construct algorithm in order to obtain correct response to any random challenge. Associated with mathematical unclonability.

8. *One-wayness.* Challenge can not be constructed for a given PUF and its response. Similar to the core property of the HASH functions.

9. *Tamper-evidentness.* Any small change of the PUF leads to different challenge response behavior. It could be integrity destruction of a physical entity or tampering at logical level.

1.1.3 PUF classification

Many PUF instantiations were proposed during last 30 years. Early studies have not been labeled a PUF by their authors but they have certain PUF-like properties. Possibly because the concept of PUF was systematized only in 2001 by Ravikanth S. Pappu [21].

The following list of PUF constructions does not purport to be a comprehensive overview of all well-known PUFs. This is just an attempt to classify PUFs by their main design properties and usage.

Early studies

The concepts of using the uniqueness of fiber structure of paper [23,24] were one of the first approaches of identification by objects that have unique manufacturing process variations. Later they made up the category "Paper PUFs" [25,26] which is mainly considered as an anti-counterfeiting strategy for currency notes.

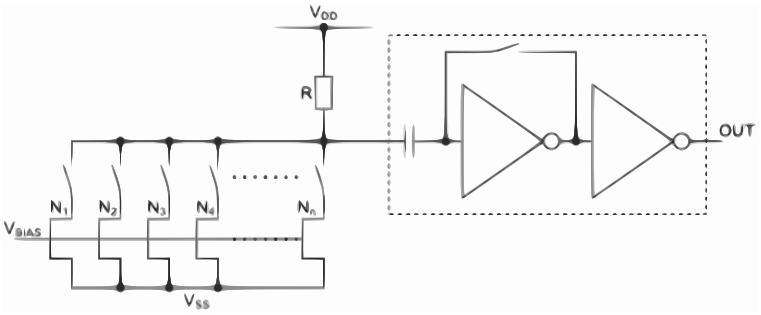

Figure 1.1: Approach from Lofstrom [1]

The idea of preventing credit card fraud by using the inherent uniqueness of the particle patterns in magnetic swipe cards was proposed by Indeck et al. in 1994 [27].

In the year 1998 Posch demonstrates a method to embed a unique signature into a coating material was used in a smart card or in the covering material of some other secure hardware device. The method bases on the impossibility of exactly reproducing a specific piece of plastic or other material used to cover the secure hardware [28].

2000, Lofstrom proposed a concept of extracting the inherent uniqueness of chips from uncontrollable manufacturing variations by comparing the drain currents of two transistors. The circuit in Fig. 1.1 contains a comparator for measuring the difference between two nominally identical transistors connected to a resistor. The output is biased toward V_{DD} or V_{SS} depending on the mismatch.

Optical PUFs

The first PUF-like concept based on random optical reflection patterns was introduced by Tolk et al. in 1992 [29]. The goal of this instantiation was the identification of strategic arms in arms control treaties. Later, proposed by Pappu et al. in 2002 [30] concept of optical PUF was based on transparent optical medium. The idea behind this approach is very simple - when a laser beam shines on the material, a random and unique speckle pattern will arise. The resulting interference pattern is an uncontrolled process and the nature of interaction between the laser and the particles is very complex. Fig. 1.2 illustrates the approach. Practical experiments [21, 30]

[1] Retrieved from [2]

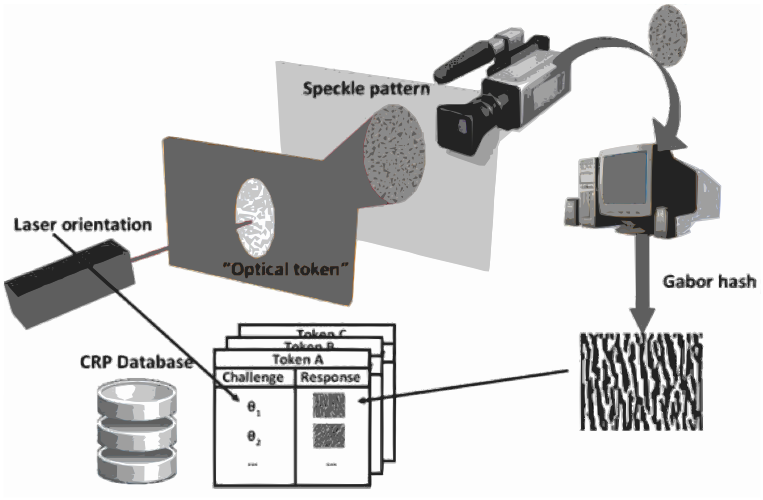

Figure 1.2: Basic operation of an optical PUF [2]

show that the optical PUF is practically unclonable. Unfortunately such PUF instantiation is hard to implement.

Ring-Oscillator PUFs

Ring-oscillator PUFs were introduced in the year 2002 [3, 31]. The schema of the approach is shown in Fig. 1.3. It turns out that the nominally identical ring-oscillators have different characteristic frequency caused by manufacturing variations and environmental conditions. Ring-Oscillator PUF is supposed to use this small random delay deviations in frequency.

Odd number of inverters make a loop where the output of a predecessor is the input of a successor. Each inverter adds an output delay to the resulting delay of a square wave. Counters in Fig. 1.3 contain all the details of the desired measure and are considered as a PUF output. Ring-oscillators are widely used as sensors to measure voltage and temperature.

Arbiter PUFs

Arbiter-based PUFs belong to the category delay PUFs as well as Ring-Oscillator PUFs. Such PUFs are based on comparing the delay between two digital paths and figuring out which of them provide the smaller delay. The originator of the approach is Lee et al. [32, 33] Production process has an effect on the physical parameters determining the exact delay of each

[2] Retrieved from [22]

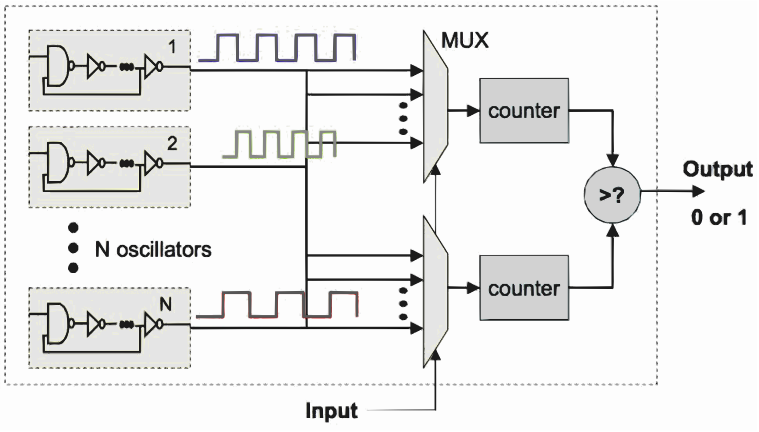

Figure 1.3: Ring Oscillator PUF [3]

path. It is expected that all devices have their own specific delay that explains the PUF concept of such approach. PUF determines which digital path is fastest and returns either a zero or one. If a delay is too small the arbiter circuit will produce unstable output depending on the environment mostly. Such phenomenon is called noise or metastability of the arbiter.

Coating PUFs

The idea of using chip-specific data was utilized by Posch in 1998 who introduced the approach to add a mixture of isolating and conducting material (coating) to the chip [28]. This idea was used in [35] by Tuyls et al. when the coating PUFs were introduced. The basic implementation and operation of a coating PUF is shown in Fig. 1.4.

Approaches discussed before were relying mostly on the effects of uncontrollable manufacturing variations. In case of coating PUF the randomness is achieved by spraying a passive dielectric coating directly on top of the integrated circuit (IC). The coating PUF obtains a response from the comb-shaped sensors in the top metal layer of an IC by measuring their capacitance. Additionally, proposed PUF concepts provide chips with the strong protection against such physical attacks as tampering. Since this coating is opaque and chemically inactive, the tampering attack would change the capacitance of the coating and as a result the original unique identifier will be destroyed.

[3] Retrieved from [34]

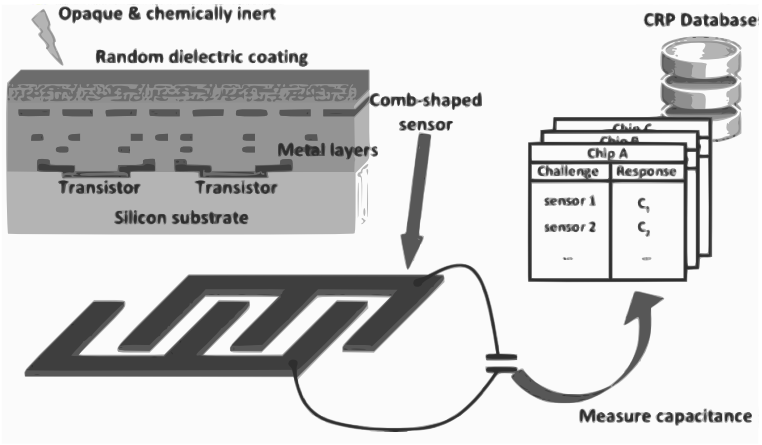

Figure 1.4: Coating PUF [4]

SRAM PUFs

A SRAM cell can be used as a PUF. Since the SRAM consists of a few transistors, the local mismatch between them is large. This is a big advantage for a PUF implementation. The idea of the SRAM PUF implementation, properties and parameters are given in more detail in Sec. 1.2.

Latch PUF

An electronic circuit which consists of two cross-coupled NOR-gates is used as a latch PUF. Such a simple NOR latch circuit is able to hold the logic value. The technique is proposed in [36]. The piniciple behind is very similar to SRAM cells. But SRAM cells are in a logically unknown state before they are set in a stable state. Whereas Latch IC is staying in stable state until it is triggered by asserting a reset signal. After which it starts converging to a stable state again. The result of convergence depends on the internal mismatch between the electronic components that have a random nature. Logical schema of a Latch PUF cell is shown in Fig. 1.5).

Flip-flop PUFs

Another memory-based intrinsic PUF was proposed by Maes et al. in 2008 as a replacement for the SRAM PUF on FPGA boards [37]. Equivalently to SRAM PUFs, Flip-Flop PUFs are based on the power-up behavior of memory but instead of SRAM cells D-flip-flops are used.

[4]Retrieved from [22]

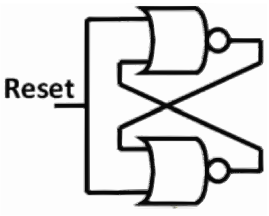

Figure 1.5: Logical circuit of a Latch PUF cell

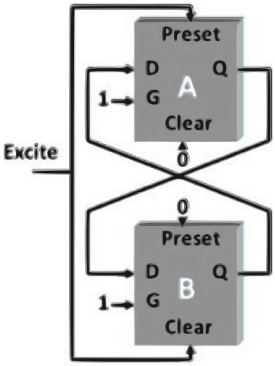

Figure 1.6: Schematical circuit of a Butterfly PUF cell[5]

Butterfly PUFs

Several drawbacks of SRAM PUFs implementation led to introducing butterfly PUFs [13]. There are some FPGAs where SRAM PUFs implementation is impossible due to the fact that all SRAM cells are automatically reseted to zero directly at power-up and hence can not be used anymore because all randomness is lost. The butterfly PUF cell construction is schematically shown in Fig. 1.6).

The Butterfly PUF uses the internal matrix of the FPGA to uniquely identify it based on the intrinsic physical characteristics of the integrated circuits. Experimental results show that it is very stable to environmental and other FPGA operating parameter variations [13].

Other instantiations

PUFs have become a hot topic. Many new researches of PUFs instantiations have appeared in the last few years. In the year 2009, Guajardo et al. described concept of LC PUF, that acts as a capacitor and constructed as a small glass plate with a metal plate on each side [38]. In [39], Helinsky et al. proposed Power Distribution PUFs based on the resistance variations in the power grid of a chip. SIMPL Systems and PPUFs were published by Ruhrmair in 2009 [40]. Reconfigurable PUFs or rPUFs were introduced in [41]. In 2011 Majzoobi et al. published an ultra-low power current-based approach [14]. Ganta et al. introduced a higly stable leakage-based approach in 2011 [42]. A current starved inverter chain PUF

[5]Retrieved from [22]

was proposed by Kumar et al [43]. [44] demonstrates an idea of dynamic physically unclonable functions using device aging to alter delay characteristics according to user instructions. Sensor based PUFs are introduced in [45]. SRAM PUF approach for key generation in wireless sensor nodes was published by Selims et al. in [46]. Lithographic variations of physical unclonable functions are described in [47]. Suzuki et al. introduced some kind of delay based PUF on glitch shapes in [48]. Some improvements of above-mentioned RO-PUFs are suggested in [49] and [12].

Performance of different instantiations

Table 1.1 summarizes the performance of some of the published PUF approaches. The performance of PUF instantiations can be described by two statistical parameters: inter- and intra- chip Hamming distance. In the table the temperature variations were taken into account because it is the source of error in PUF output. Another source of error is the power consumption which is not measured for all presented approaches thus it is not listed in the table.

Approach	Ref	Temp. (°C)	HD_{intra} ,%	HD_{inter},%
ID Cell	[50]	-25..125	5	50
Ring-oscillator PUF 1	[51]	20..120	0.48	46.15
Ring-oscillator PUF 1	[8]	25..65	<2	47.31
Arbiter	[51]	20..120	9	23
SRAM PUF1	[4]	-20..80	12	50
SRAM PUF2	[5]	25	5	bias to 1
Latch PUF	[36]	0..80	5.5	50
Inverter-based PUF	[52]	20..125	0.4	50
Butterfly PUF	[13]	-20..80	6	50
D-flip-flop PUF1	[37]	25	<5	50
D-flip-flop PUF2	[53]	-40..80	10	35
Glitch-based PUF	[48]	0..80	<8	40

Table 1.1: Summarizing of different approaches [6]

[6]Retrieved from [22]

1.1.4 Advantages and disadvantages of PUFs

Two important PUFs advantages could be emphasized: cost reducing and security raising. Some part of applications e.g. identification requires generated unique ID. Usually, ID is stored in a nonvolatile memory. The chips with nonvolatile memory cost more than without it. Because some additional resources are needed in order to produce such chips and therefore some additional expenses. The chips without internal nonvolatile memory require to generate an ID and keep it on external storage. Later the ID is transfered back which causes additional costs again.

A PUF implementation on a chip does not require nonvolatile storage because it is assumed that a chip already has an ID. Therefore a PUF is able to generate unique ID "on-the-fly". Thus the task of identification is reduced to the searching of suitable PUF that has a large enough entropy in order to identify the number of required chips.

A PUF implementation allows to increase the security level. First, the PUFs resist reverse engineering attacks. Second, PUFs don't require to generate and store secret keys on external storage and than transfer it. These steps are very pricey for the chips without PUF to guarantee the security.

A PUF can easily reach security standards because confidential data are generated directly on the chip. Moreover it is unnecessary to store it since it can be easily reproduced at any time. Therefore according to the public key cryptography principles, the secret key is needless to transfer.

The main problem of the PUFs is the noisy output (see Tab. 1.1). All PUFs produce unstable response, some of instantiations are very sensitive to the environmental conditions and as a result are very noisy. These errors can be random or deterministic. Random errors are caused by circuit noise. The deterministically generated errors are caused by temperature variations, aging effect, voltage or everything that effects on the local mismatch of the internal components.

For some applications e.g. key generation or authentication, such PUF output is not accepted. So, some techniques for the response stabilization should be applied. It will take additional cost and as a result reduce the advantages. For example, in order to correct errors in the PUF output, error correction codes should be implemented on the chip. The ECC require a nonvolatile memory available.

But such applications as identification can be implemented without additional error correction techniques as long as the distance (usually Hamming distance is calculated) between the IDs is large enough. Therefore even if relatively big errors occur, the chip can be identified correctly.

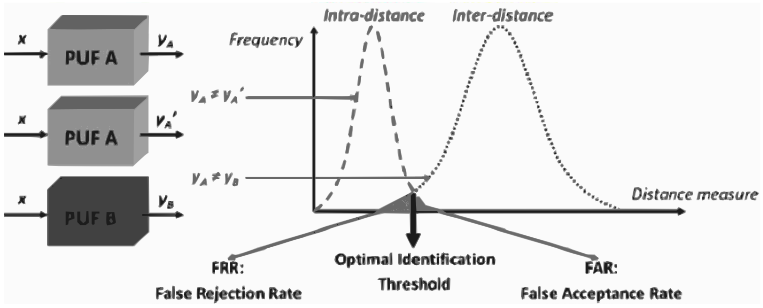

Figure 1.7: Identification process [7]

1.1.5 The basic PUF applications

Identification, authentication and key generation are three the basic PUF applications [54]. In this section an overview of all of them are given.

Device identification

Identification is the simplest application scenario that can be applied on PUF without additional techniques. The idea of PUF identification is widely used in anti-counterfeiting technologies. A biometrical identification scheme works very similarly.

Process of identification consists of two phases: initialization and identification. During the first phase, a PUF is queried many times and the challenge-response pairs are stored in a database. The identification phase allows to find an entity in the database that contains current challenge to the PUF and the corresponding response. The result of the whole identification process is a chip ID which is assigned to the CRP in the database.

The decision on the matching of the observed response and the enitity in the database is usually made by Hamming distance calculation. In a pure PUF without error correction techniques the responses on the same challenge usually slightly differ. In such cases Hamming distance is the most suitable method to measure the tollerance of binary output data. Therefore if there is an entity in the database where the Hamming distance between this entity and the observed response is close enough then chips are identified, otherwise the process fails.

The acceptance level used to decide on a positive identification depends on the intersection of the intra-chip Hamming distance and the inter-chip Hamming distance distributions (see Fig. 1.7).

[7] Retrieved from [22]

The intra-chip Hamming distance histogram represents the bit error rate distribution of the same chip during the set of queries. The inter-chip Hamming distance histogram shows the distance between the different chips. Obviously errorless identification is possible if both diagrams do not intersect. Then the threshold is somewhere in the middle of these diagrams(see Fig. 1.7). In case of overlapping of these diagrams some errors in identification are possible: either the wrong chip is identified (false-acceptance rate), or no chip is identified (false-rejection rate). In order to choose an appropriate accepted level it is recommended to minimize the sum of FAR and FRR.

Key generation

A PUF can be used in secret key generations [55]. The main requirement for such application scenario is the stable PUF output. As we seen in Table 1.1 all PUFs are quite noisy that is intrinsic property of a PUF. In order to produce exactly the same PUF response over the number of readouts, some error correction techniques should be applied.

ECC requires an auxiliary data which should be stored in nonvolatile memory. It causes cost raising (if in-chip memory is used) or decrease in security (if external storage is used) so the result reduces both PUF advantages 1.1.4.

The scheme here again consists of two phases. The initialization phase where the chip is read out, the PUF output with auxiliary data (helper data) is stored in an database. During a key generation phase a PUF is queried again and current readout is combined with the helper data from the database. An algorithm uses them to extract the same secret key as in the initialization phase. This problem is known as secret key extraction. The result of the algorithm is a shared secret key. Helper data should not be kept secret, therefore it can be stored in external storage without possible secure issue [10, 56].

Authentication

Authentication procedure is identification plus identity verification step. A very nice practical example is a card-reader at entry 1.8. A request is sent to the smart card to be authenticated [57]. The card replies back. The sender compares the card's response with the entity in the database. If they match, the card is authenticated.

If PUFs are used in response-generation scenarios, then it can be done in two types [2]: by using hardware-based CRPs or software-based CRPs:

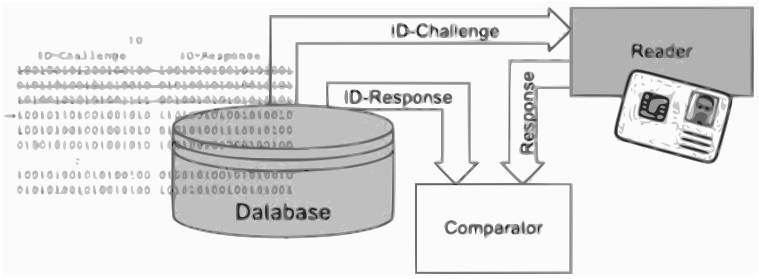

Figure 1.8: Principle of authentication using challenge-response pairs [8]

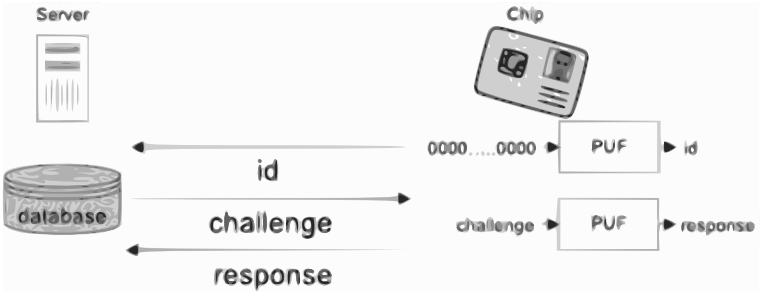

Figure 1.9: Principle of authentication with hardware-based CRPs generation [9]

1. The concept behind the authentication with hardware-based CRPs is shown in Fig. 1.9. Here PUFs should work like a HASH function: take an input and generate a corresponding output. Therefore PUFs have HASH function properties, namely unpredictability, even an extensive set of CRPs are known; uniform distribution, the responses are uniformly distributed over a set of all possible output values; stability of the response apart from possible noise at the output.

   The scheme of the authentication with hardware-based CRPs can be represented as follows: The PUF is queried by zero-vector. The output of the PUF is defined as the chip ID. Again, the output is usually noisy. So replied ID can not be found in the database. Therefore the server uses the intra- and inter-chip Hamming distance histograms in order to find out the appropriate entity in the database. After this step the chip is identified.

[8] Retrieved from [2]
[9] Retrieved from [2]

Then the server sends the new request to the PUF containing the challenge of the corresponding CRP. If the PUF response matches the expected value from the database, the chip is authenticated.

Obvious advantages of the authentication with hardware-based CRPs are that neither ECC nor nonvolatile memory is needed and it is algorithmically simple. The disadvantages are that it is hard to implement, requires external storage, CRPs must be in secure environment.

2. The idea behind the approach of authentication with software-based CRPs is that the challenge and the response are checked by means of an encryption algorithm\dot{c}itehm2012physical. The approach requires the error correction code implementation and consists of two phases. At the beginning the ID of the entity should be obtained ether from an external storage or from internal memory or generated "on-the-fly". Later on public-key or symmetric-key cryptography should be used to generate the response. Advantages of the approach are simple implementation, usually no CRPs has to be stored. Disadvantages are ECC implementation.

1.1.6 Use cases

The three main aforementioned application scenarios can be used in a wide range of use cases of PUFs. In this section the most important of them are covered.

Product identification

The idea of identification by using a PUF as an integral part of an RFID tag is shown in [58]. It is supposed that every goods item has a sticked RFID tag therefore unique PUF is assigned to it.

RFID tags are considered an improvement on bar codes. Bar codes have a limited usage. They allow to match a product only to the category. Whereas RFID tag can be used for the product identification that allows to carry more information about the product during the whole production-customer chain. But RFID tags have a bid disadvantage in comparison with bar codes - it has production cost in contrast to free printed bar codes.

Concept of PUF-RFID usage. During the manufacturing process the vendor glues a PUF RFID on a product box and inputs the information about the product into a database (e.g. destination, expiration date, etc.). A logistics company takes the product from the warehouse and distributes it to the shop. The process can be mechanized by using RFID-readers that

will help to sort goods by the destination address. Later, in a shop, the RFID is read again at the check-out. The last step in the chain is a fridge with internal RFID reader at the consumer's home. The fridge can notify consumer about approaching expiration date or even suggest some possible recipes for cooking.

The business model is original and promising. The only discouraging factor is the high initial cost.

A more realistic example is smart cards usage. In order to get an access to a room an authentication system needs to read the data from the smart card. A PUF implementation on the smart cards makes them resistant to clone attacks. A lost and found card can be reused by attackers, but a renewed card will fix this problem.

Secure key storage

A PUF can be successfully used in services which require to store unique secret key, e.g. Pay-TV, DRM, file encryption. Usually private keys are stored in nonvolatile memory that causes various attacks such as side-channel attacks and software exploits [59]. Using a PUF in applications for storing the key does not require to keep it in the chip permanently. A PUF allows to generate the unique key "on-the-fly" by using intrinsic physical properties of the PUFs.

With combination of an asymmetric cryptographic approach, the public key can be generated and used for channel encrypting. The scheme is very robust against possible attacks.

Night-shift problem

Nowadays business models are distributed over the world. And production tasks can be outsourced to foreign companies or even foreign countries. Companies that make production request often fear that another company will produce more than the ordered number of items and resell the rest of it illegally. Such problem is called the night-shift problem.

One approach to solve this problem is a PUF implementation on the chip. This PUF can be reread many times by everyone. A legally used chip has the same PUF output as the CRP record in a company's database. Illegally used chip is going to be rejected during identification phase of getting access to the service. In this case company should have the full database of produced items. In another approach a company can activate the chip just before selling. All activations are stored in database and used in identification phase.

Anti-counterfeiting

Counterfeit money is a well known product of forgery. But there are many other forgeries that cause even worse damage in the market. According to the statistics from the Organisation for Economic Co-operation and Development (OECD), the top market fields suffered from counterfeting are: drugs, electronics, software piracy, food and auto parts.

Nowdays the usual approach to protect products from forgery is an authenticity mark adding. But the problem of such approach is that all the marks are the same. When an attacker succeeds in cloning one, the product is easily forged.

PUFs properties allow to construct very simple but very reliable anti-counterfeiting schemes. Every valuable object can be equipped with a near field communication (NFC) tag including a PUF and an authentication algorithm. NFC is a form of RFID which supports two-way communication between endpoints. This technique will ensure the genuineness of a product by requesting the database with IDs of legal devices. Since a PUF is unclonable and unique it is imppossible just to clone a genuine product.

The cost of implementation is small compared to possible losses of profits and reputation. The infrustructure is also not so hard to implement. Only power consumption can cause some difficulties since the public-key cryptography algorithms take energy which is a problem in terms of passive RFID tags [60].

IP protection

FPGAs, as the reconfigurable and programmable integrated circuits have become very popular and have a wide field of applications because of functionality and cost. Therefore the problem of intellectual property in the chips is raising.

There are many approaches to obtaining the software from the existing FPGAs in order to clone the product. It can be done ether through reverse engineering, industrial espionage or unreliable outsourcing companies [13]. After the device is powered-up the software is loaded into the internal memory. An adversary can dump the memory and recover the software. Since the software is known, an adversary starts to produce full clones of the product. Sold clones are cheaper because the IP owner is not paid for research or development cost.

There are some procedures to make the software recovering more difficult [61]. But they are not very satisfactory solutions. In [16] an approach of keeping the software encrypted was proposed. The configuration is stored

on external memory and during the powering-up it is loaded into internal memory of the FPGA. With the use of a secret key from the embedded nonvolatile memory the software is decrypted and ready to use. Suggested concept requires key storing in plain-text format on internal nonvolatile memory. This key can be generated from an intrinsic PUF, for example a DFF [37] or SRAM PUF [1]. Usually PUFs are noisy therefore the error correction data should be generated. Such data can be saved on external storage since they don't include any secret information on the key.

So no internal memory is needed. A secret key is generated by PUF using physical intrinsic properties of the FPGA. The error correction data and encrypted software are both stored outside the FPGA. In addition, an cryptographic encoder should be implemented on the FPGA. Different implementations are suggested in the literature [4, 13, 17].

1.1.7 Attacks

A PUF looks very reliable and secure because of its intrinsic unique properties which are obtained due to manufacturing process variations. But there are some attacks already known. In this section they are briefly explained.

Lack of entropy

The obvious solution to the problem of product counterfeiting is to produce a clone. The cloned device should have a PUF with the exact same intrinsic properties as the original one.

Suppose that an attacker is able to repeat the whole production process of devices or has an access to the original physical environment. The question is when the attacker achieves success in the PUF cloning. This question has a statistical meaning and depends on the PUF's entropy. The larger the entropy of designed PUF the more instances are needed in order to successfully create an identical PUF.

If the entropy of the PUF is b bits, then it is theoretically possible to obtain 2^b number of unique identificators from it. All these identificators are uniformly distributed, this means the probability of occuring an ID is equal. But because of the birthday paradox the population of all possible identificators is two times less $2^{b/2}$.

Obviously such clone attack makes sense if and only if the resulting profit should be more than current operating expenses:

$$p \cdot s > a \cdot c,$$

where p is the profit of getting a counterfeit instance, s is the number of cloned instances, a is the number of instances produced by attacker and c is the expenses on the clone creating.

The number of successfully cloned PUFs instances can be expressed as:

$$s = \frac{a}{2^b}$$

Therefore, the clone attack becomes unprofitable if:

$$b > log\frac{p}{c}$$

It shows how important it is to design the PUF with the large entropy.

Modeling attacks

Modeling attacks are well described in [62] by Ruhrmair et al. They are based on the learning algorithms when some CRPs are known. Attacks are suggested for Arbiter and Ring-Oscillator PUFs [63]. E.g. an attacker knows a set of CRPs and is able to construct a model of PUF which gives the result with high probability.

Therefore PUFs must be resistant against modelling attacks, the response of the PUFs should not depend on extensive set of CRPs. However it was shown in [64] that a very accurate model (90%) of PUF can be constructed by having 500 CRPs only. The error correction techniques makes the modeling attacks even simpler. Since the ECC can easily correct 10% bit error rate in the PUF output.

Side-channel attacks

As we know the attacker does not have an access to the internal structure of the PUF, the result of such intervention cause the dysfunction of the CRP behaviour. But the attacker is able to measure some external effects of the PUF functioning such as electromagnetic radiation, time various computations, sound, and power consumption.

The researches show the susceptibility of the PUFs to the side-channel attacks [4,65]. The PUF output can be recoverd by investigating the power leakage occurring in the error correction phase [4]. But such attacks are hard to implement.

Invasive methods

Invasive attacks start with the depackaging of the chip in order to get direct access to its inner components. Because of the intrinsic properties

of the PUFs such attacks are currently not sophisticated enough to be a success. The removal of chip layers cause the destroying the unique chip fingerprint [66]. However, [15] proposes successful semi-invasive attacks based on EM cartography.

A well-designed PUF should meet the requirements in order to be resistant against described attacks:

1. *Temperature resistance.* The temperature variation effects the delay of internal PUF components [67].

2. *Voltage ramp-up time.* Decreasing the supply voltage will slow down the circuit. [68].

3. *Ageing.* Electrical components are prone to degradation over time. The underlying reasons for it are time-dependent dielectric breakdown, electro migration, hot carrier injection and negative bias temperature instability [53,69]. Therefore, ageing tests should be provided to measure the PUF behaviour under temperature variation and supply voltage.

4. *Entropy.* Entropy is directly connected with the length of the generated keys and the population of unique PUF fingerprints. The PUF's entropy should be large enough to guarantee resistance against attacks [70].

1.2 SRAM PUF

It turns out that the initial power-up values of the SRAM cells behave randomly and independently. Moreover a SRAM cell is a bistable system where uncontrolled production properties affect resulting value in a cell - either "0" or "1". These facts allow to use initial power-up state of SRAM for a PUF constructing.

1.2.1 SRAM cell

A usual SRAM cell consists of six transistors as shown in Fig. 1.10. A cell contains two cross-coupled inverters (load transistors $P1$, $P2$, $N1$ and $N2$) and two access transistors ($N3$ and $N4$) connected to the data bitlines ($\overline{WL}$ and WL) based on the wordline signal (V_{ss}). Each inverter is a p-junction transistor ($P1$, $P2$) and an n-junction transistor ($N1$, $N1$).

A key characteristic of a SRAM cell in terms of the PUF characteristics is bistability. Fig. 1.11 demonstrates the basic functionality of a bistable

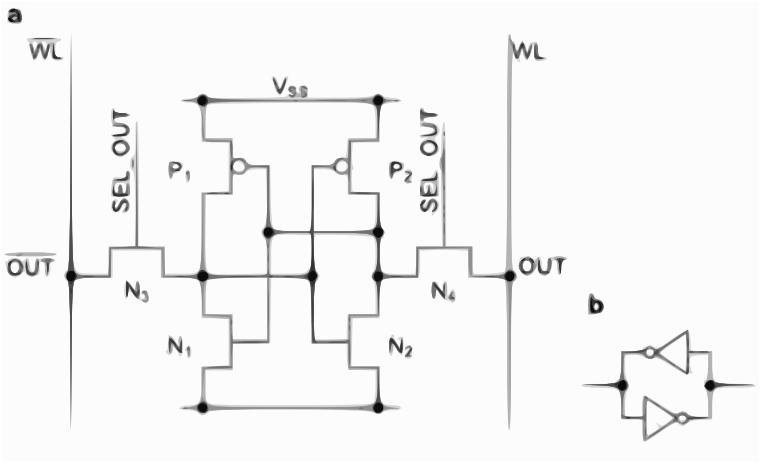

Figure 1.10: a) Common six transistor SRAM cell b)Equivalent circuit [10]

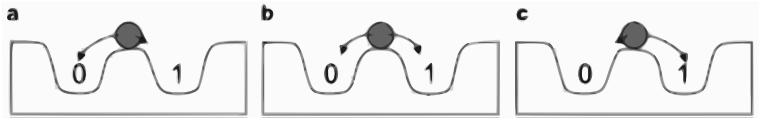

Figure 1.11: Bistable system: a) Bias toward "0"; b)Balanced system; c) Bias toward "1" [11]

system. A totally balanced system is shown in Fig. 1.11b. The probability that the ball falls down to the left is the same as the probability of its falling down to the right. Neither 1.11a nor 1.11 are balanced systems since the probabilities are unequal. In the left case the system is biased toward "0", in the right case - toward "1". The same behaviour inheres in a SRAM cell(see Fig. 1.12). But instead of a ball it is the power-up voltage that decides a case.

The reason of the process of biasing toward one of the stable state is the static-noise margin (SNM). Increasing the SNM causes a cell to become more stable, thus a higher voltage is needed to flip the state [71]. The different SRAM cells have different SNMs caused by intrinsic parameter fluctuations in a CMOS SRAM cell [72]. Such fluctuations are known and they do not affect the reading/writing process of the cell.

During power-up, the SRAM cell's cross-coupled inverters Fig. 1.10

[10]Retrieved from www.springerimages.com
[11]Retrieved from [2]

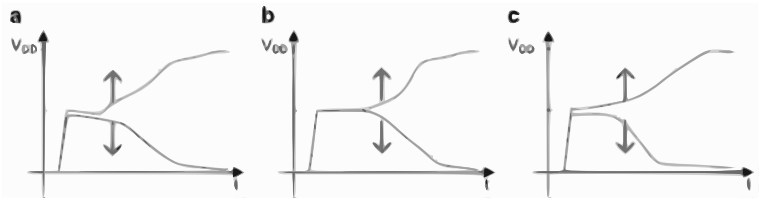

Figure 1.12: Bistable SRAM circuit: a) Bias toward "0"; b)Balanced system; c) Bias toward "1" [12]

are "floating" [4]. Therefore, the different SNM in the transistors cause the different resulting value in a cell - either "0" or "1", depending on the intrinsic characteristiscs of the cells. It was shown that the same SRAM cell tends to have the same state after power-up whereas different SRAM cells will behave randomly and independently from each other [4].

Since the bias cannot be controlled during production and the cells behave randomly and independently, all these facts are key properties of a PUF and thus the start-up values of SRAM memory can be used for a PUF constructing.

In detail a SRAM cell behavior during power-up looks as follow [2]: During powering-up the circuit V_{DD} increases toward its final value (e.g., 1.35V). As soon as the first PMOS transistor (either $P1$ or $P2$) starts to provide much current (i.e., V_{TH} of the transistor is reached) the corresponding node Q/Q_n starts to increase voltage which at the same time decreases V_{GS} of the second PMOS transistor and increases V_{GS} of the corresponding NMOS. Thus a latching effect occurs and the cell either outputs V_{SS} or V_{DD}. If the difference between the two transistors $P1$ and $P2$ is very small, both start providing current nearly at the same time. Thus, Q and Q_n increase in voltage until the V_{TH} of one of the transistor $N1$ and $N2$ is reached. Now, the latching effect is defined by the NMOS transistors. It can be seen that in most cases the output of a SRAM PUF cell is defined by the threshold voltage difference ΔV_{THP} of the PMOS transistors. Only in case of very small ΔV_{THP} NMOS transistors' threshold voltage difference ΔV_{THN} dominates the decision.

The cells that possess the same value after the power-up, are called stable cells and represent the fingerprint (or identity) of the SRAM PUF. Whereas the unstable cells are defined by changeable state from one power-up to the other. The set of the unstable cells is called noise of the PUF.

[12]Retrieved from [2]

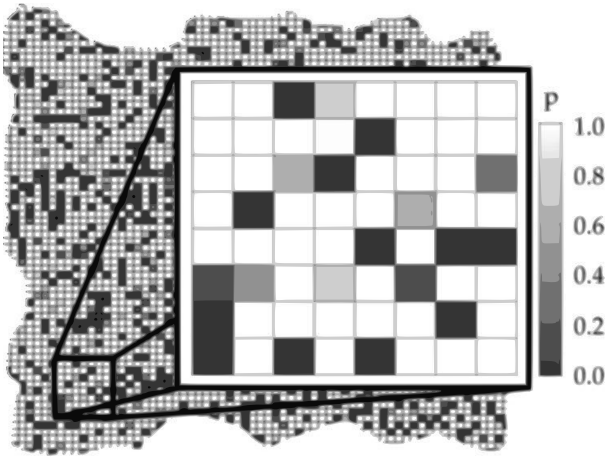

Figure 1.13: Fingerprint of the SRAM content [13]

SRAM PUFs are usually very noisy (see Tab. 1.1). An example of the SRAM heat-map diagram is shown in Fig. 1.13.

Obviously only the stable cells are suitable for the PUF output generation. There are some approaches [5] that allow to mark stable cells before the PUF is used for the first time. The rest of SRAM cells will not be used in the PUF output but can be used for storing auxiliary data (e.g., helper data of an error correction code or a cryptographic encoder).

The question is how to select stable cells. It could be done by performing readouts repeatedly and choose only the cells with stable output values. But this approach is not suitable for various reasons. First, additional measurements produce an impact on the product costs. Second, measurements should be done over the wide range of experiments, e.g. temperature and voltage variations, aging measurements.

The approach from [36] says that stable cells decide faster. The concept is shown in Fig. 1.14. By measuring time the fast flipping cells can be detected, to be selected for the PUF constructing.

In Fig. 1.14 all the cells with the decision time under t_{useful} and lie under a lower threshold or above upper threshold are marked as useful. All other cells are not selected. Simulations show that the decision time of the cells is a function of the temperature. Therefore, during the selection phase a stable temperature should be provided.

The approach proposed in [5] suggests to select the cells which provide a mismatch exceeding a certain threshold. NMOS transistors must be taken

[13] Retrieved from [5]

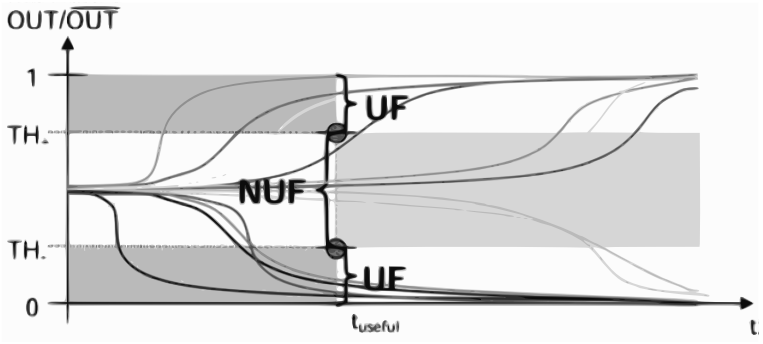

Figure 1.14: Measurement of decision time (UF: useful; NUF: notuseful)[14]

into account and the mismatch should be above a certain value. The good chosen the threshold is balanced trade-off between the ratio of the useful PUF cells and all PUF cells. A method to measure mismatch is to use an analog to digital converter (ADC).

If the pre-selection methods for searching useful cells are not possible to apply or the cost of their implementation are very high, then some post-selection methods could give better result. The post-selection techniques deal with errors which occur due to unstable cells. Such unstable cells represent a noise of the PUF output. Thus the post-selection techniques correct the errors in the PUF output. A very effective post-selection technique is based on the error correction codes.

1.2.2 Statistical analysis of SRAM PUFs

The hardware properties of a SRAM PUF can be characterized by the statistical parameters and measurements such as: mean value, Hamming distance, autocorrelation function, false acceptance rate and false rejection rate, binomial distribution. These parameters allow to estimate how far the current PUF is from an ideal one.

Mean value

The mean value parameter shows distribution of the output values “1” and “0” in a PUF. The PUF should have unpredictable output therefore the values are distributed equally. Thus an ideal PUF has the mean value parameter that equals to 0.5.

[14]Retrieved from [5]

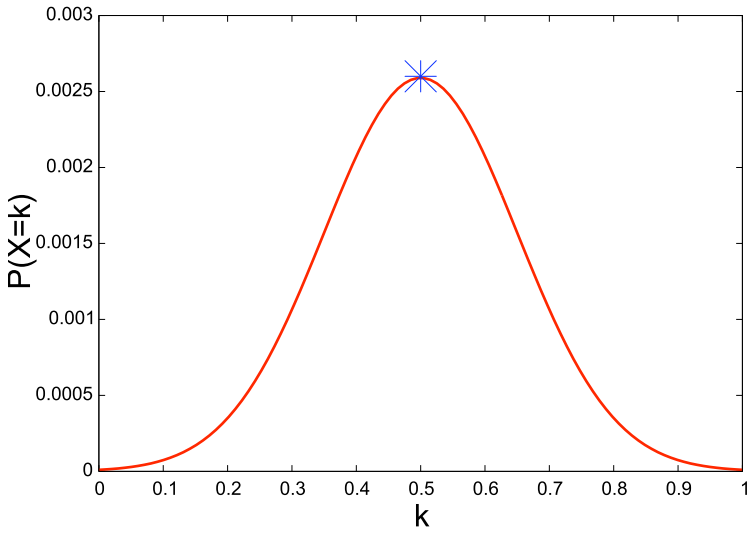

Figure 1.15: Binomial PDF

The probability density function(pdf) of a binomial distribution describes the distribution of the mean values of SRAM cells. Therefore the mean value of an ideal PUF is situated around 0.5 on top of *pdf* (Fig. 1.15). The deviation from the ideal value either to 0 or to 1 is called bias. The bias can effect the predictability since the number of likely combinations reduces. Unfortunately, the strong bias toward "1" are confirmed by many measurements of the SRAM PUF based approaches [2, 5].

Intra-chip Hamming distance

The key PUF property is reproducibility (see Sec. 1.1.2). It requires to guarantee the constant PUF response within a predefined region of operation. Changes in bits between different runs cause an error. The error can be estimated by the bit error rate parameter which is measured using the Hamming distance usually. BER is the Hamming distance between the current PUF output and the predefined one (e.g. the PUF output during the first readout is chosen usually).

An ideal PUF has the BER value is 0. But due to the environmental conditions, noise, temperature variations, power supply the BER exceeds 0. Studies show that the SRAM PUF output is very erroneous(Tab. 1.1).

Correlation

Autocorrelation function allows to detect correlation between cells of the same SRAM chip. The chips with detected correlation are very sensitive to brute-force attacks since the number of unique combinations reduces. An ideal PUF should have no correlation between bits.

In this measurement the Low periodic autocorrelation and Good aperiodic autocorrelation [73] are suitable functions:

$$\textit{Periodic autocorrelation: } C_a(\tau) = \sum_{i=0}^{N-1} (-1)^{a_i + a_{i+\tau}}, 0 \leq \tau \leq N-1 \quad (1.1)$$

$C_a(\tau) = \pm 1$ says about the correlation between bits and in contrast, $C_a(\tau) = 0$ does not show any correlation at lag τ.

$$\textit{Aperiodic autocorrelation: } A_a(\tau) = \sum_{i=0}^{N-1-\tau} (-1)^{a_i + a_{i+\tau}}, 0 \leq \tau \leq N-1 \quad (1.2)$$

Inter-chip Hamming distance

Physical unclonability of the PUFs is represented by the property identity(see Sec. 1.1.2). The identity (or uniqueness) can be characterized by estimating the Hamming distance between the PUF outputs of different chips. This statistical parameter is called inter-chip Hamming distance. An ideal PUF should have $HD_{inter} = 50\%$ between all possible chip pairs. This value confirms that no correlation between chips exist.

Hamming distance is the preferred parameter for the correlation detection between chips. But if the PUF produces the same output on each of the chips then the Hamming distance is not able to detect correlation. Therefore another approach should be chosen. It can be done by calculating each cell's mean over all chip population.

Size

Another statistical parameter is the size of the PUF which is expressed in $\frac{m^2}{bit}$ and characterized by area. It shows the technology of the device production and useful in estimating of the entropy of the PUF.

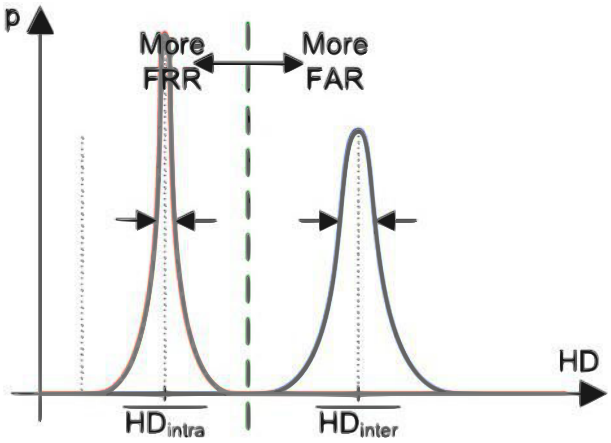

Figure 1.16: False acceptance rate (FAR) and false rejection rate (FRR) [15]

Speed

Speed can be considered as a good parameter for evaluating PUFs. There are some approaches [5] where stable cells are selected by their speed decision. It was shown that stable cells decide faster. Speed is a very useful parameter in pre-processing approaches. The measurement unit for speed is $\frac{bits}{s}$.

FAR and FRR

False acceptance rate (FAR) and false rejection rate (FRR) are shown in Fig. 1.16. They are key parameters in the identification scenario. The combination of the inter- and intra-chip Hamming distance histograms shows how well the PUF can identify different chips.

If the both curves do not intersect then the PUF is able to identify the chip without error. If some intersection area is observed then error can occur:

- a wrong chip can be identified. False Acceptance.
- a chip can not be identified. False Rejection.

Correctly chosen threshold minimizes the probability of error occurrence. Usually the intersection of curves FAR and FRR sets the threshold.

[15] Retrieved from [2]

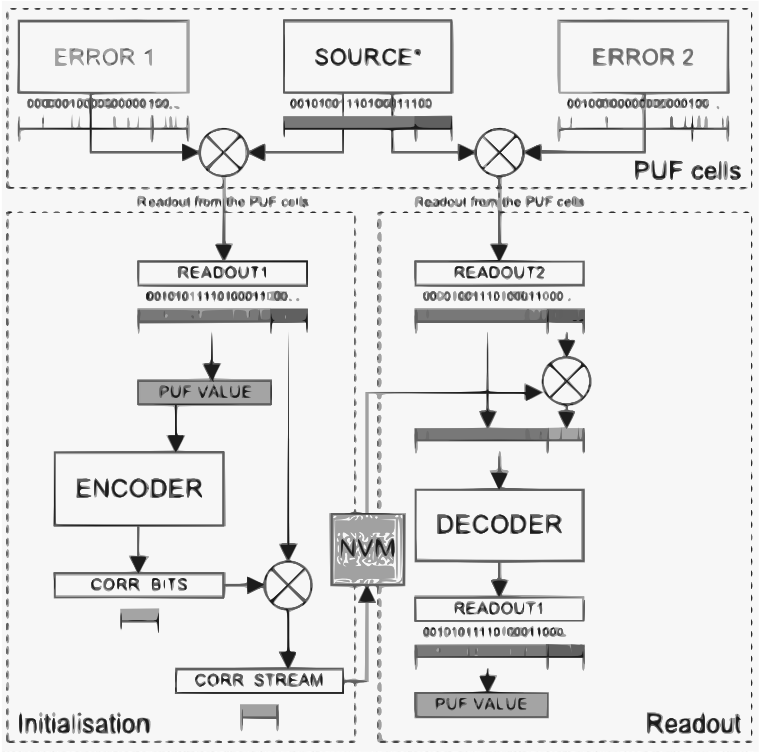

Figure 1.17: PUF with the ECC scheme [16]

FRR and FAR can be expressed algebraically as follows:

$$FRR = \frac{1}{\sigma\sqrt{2\pi}} \int_{HD_{max}}^{\infty} e^{-\frac{1}{2}\left(\frac{x-\mu_{intra}}{\sigma_{intra}}\right)^2} dx * 100\% \tag{1.3}$$

$$FAR = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{HD_{max}} e^{-\frac{1}{2}\left(\frac{x-\mu_{inter}}{\sigma_{inter}}\right)^2} dx * 100\% \tag{1.4}$$

1.2.3 SRAM PUF implementation

Fig. 1.17 contains the general ECC scheme implemented on a PUF. The procedure of the PUF design consists of two phases: the initialization phase and the working (readout) phase. During the initialization phase an auxiliary data are generated that are used later to correct errors in the chip response. This data should be saved in nonvolatile memory. But generated by ECC the auxiliary data can not be stored in NV memory as is. Since

[16] Retrieved from [2]

the NV memory is exposed to attacks by hackers then retrieved bits can be used for recovering the key generated by PUF. Thus some helper data are needed in order to hide the correction bit string. Then the bit code generated by the error correction codes encoder is XORed with the helper data retrieved from the output of SRAM cells. The result is stored in NV memory. Encoder generates the correction sequence from the part of the first readout.

The result of the readout phase is to obtain the same readout value that was during the initialization phase. The error correction codes or their combination help to reach this goal. The ECC were analyzed in detail in [74]. Some are used for correcting small bit error rates and some are very computationally complex. There are even concepts of using combination of several ECC [10].

According to the scheme in Fig. 1.17 retrieved from the nonvolatile memory correction sequence in combination with the readout of SRAM cells is used by error correction code generator. The result of this operation is the original readout obtained during the initialization phase.

As it has been already mentioned and was shown in the table of experiments (Tab. 1.1) the SRAM based PUFs show high bit error rates. And as it can be seen from the table the error rates are even higher than in others PUF implementations. To reduce the error rate different types of error correction codes (ECC) can be used. Not all the ECCs are suitable: some are very complex to implement in a microcontroller, others are not able to correct such high error rates.

In [74] different types of ECCs were analyzed: linear error-correcting codes (Hamming code), cyclic codes (BCH code) and repetition code. It was shown that only the repetition codes are capable to correct the bit error rate higher than 10%. In addition they are easy to implement on the microcontrollers. Therefore the ECC with repetition factor is preferable technique for the PUF output stabilizing by the error correction.

1.2.4 Patents

During the last decade some patents concerning the PUFs implementation have appeared. The list is shown in the following section it is not a complete list of all available patents, but just a brief overview of topics where the patents are granted. All patents are taken out in the USA and grouped by the main application scenarios (see Sec. 1.1.5) – identification, key generation and authentication.

Identification schemes

Methods of the identification by the PUF were discussed in Sec. 1.1.5. The idea behind is the manufacturing process variations producing unique fingerprint that characterizes the device itself. The fingerprint can be slightly different under environmental conditions. But usually it is enough to identify the device. Early proposed patents take into account local mismatches between internal components and suggest to generate unique number. Here are some of them:

- *ICID 1999.* In the year 1999 Lofstrom filed the first patent [75]. Local mismatch between different cells is used for identification.

- *Microvision 2000.* Patent suggests to reread the output several times in order to identify stable bits. The marked bits are used to generate the identification number of the chip.

- *Hitachi 2001.* Another chip identification procedure, based on production variability, was filed by Marunaka in 2001 [76]. Hitachi suggests to use the voltage mismatches between local components.

- *STM 2003.* The patent is based on the resistors mismatches [77]. In the year 2003 ST Microelectronics filed a patent.

- *PIXIM 2004.* One more version of using the intrinsic mismatches in voltage of circuit elements (NMOS and PMOS transistors, resistors, capacitors, inductors, light-detecting pixel elements, memory cells, amplifiers). Filed by PIXIM in 2004 [78]. First mention of SRAM PUF.

- *Infineon 2007.* Some kind of improvements of $ICID1999$ patent by Heiko Koerner from Infineon Technologies [79].

- *National Semiconductor 2008.* The patent describes an SRAM PUF that is used to generate a unique number with logic states only without going into application details [80].

- *National University Corporation Tohoku University, Advantest Corporation 2008.* Filed by Okayasu et al. in the year 2008 [81] the patent describes aging effect of the device by voltage variations over time.

Key generation schemes

Key generation approaches require stable PUF output in order to obtain the same key over the whole region of operations. Therefore the patents should contain some error correction techniques.

- *STM 2002.* One of the first patent describing the key generation procedure by using process variations was filed by ST Microelectronics in 2002 [82].
- *LSI 2005.* The patent [83] describes the procedure of obtaining the key but without using any storage outside the chip. An error correction technique are also suggested.
- *Xilinx 2005.* The patent implements mechanisms of public key cryptography. An external storage is used and transfer channel is encrypted by a key [84].
- *MIT 2006.* In the year 2006 the Massachusetts Institute of Technology describes in the patent [85] the procedure of the PUFs usage in key generation for cryptographic purposes.
- *Irdeto 2009.* The patent [86] prevents the cloning attacks by the PUFs.

Authentication schemes

In authentication schemes the verification phase is added to the identification schemes.

- *ICID 2002.* Horng et al. use the position of unstable bits for the authentication process [87].
- *MIT, Intrinsic ID 2003.* In this patent [88] a multi-bit input to a PUF causes a device specific output.
- *Philips 2004.* Another authetication procedure, based on production variability, was filed by Pim Tuyls et al. in 2004 [89].

1.3 Atmel AVR microcontrollers

The Atmel AVR^{TM}is a family of 8-bit RISC microcontrollers produced by Atmel. The AVR architecture was designed by two students at the Norwegian Institute of Technology (NTH) Alf-Egil Bogen and Vegard Wollan and developed by Atmel in 1996.

Microcontrollers produced by *Atmel* became attractive due to their advantages:

- *Moderateness of prices.* Major distributors are ready to sell inexpensive components (under $5) in small quantity.
- *Low barriers to entry.* Well documented. Can be programmed in *C*.
- *Stability.*
- *Built-in functionality.* Have all you need for basic tasks. Built-in RAM, FLASH, EEPROM, ADC.
- *Low power consumption.* This advantage extends field of utilization.
- *Cross-platform support.* Can be developed under any platforms with free tools.

Atmel manufactures 3 variations of 8-bit microcontrollers: TinyAVR, MegaAVR, XmegaAVR. These variations differ in the physical size, memory size, number of inbuilt peripherals and applications.

1.3.1 Atmel ATmega1284P

$Atmel ATmega1284P$ [90] is the high-performance Atmel 8-bit AVR RISC-based microcontroller which is equipped with 128KB ISP flash memory, 4KB EEPROM, 16KB SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a real time counter, three flexible timer/counters with compare modes and PWM, two USARTs, a byte oriented 2-wire serial interface, an 8-channel 10-bit A/D converter with optional differential input stage with programmable gain, programmable watchdog timer with internal oscillator, SPI serial port, a JTAG (IEEE 1149.1 compliant) test interface for on-chip debugging and programming, and six software selectable power saving modes. The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.

Chapter 2

Realisation

This chapter focuses on the investigation of the PUFs hardware properties in SRAM cells. In the first section of this chapter particular attention will be placed on the statistical parameters estimation. The following parameters will be determined and explained: Mean value, error rate, correlation between bits, correlation between given chips, false acceptance rate and false rejection rate. Further several post-processing and pre-processing techniques of the error correction will be proposed. Finally the PUF design process will be presented and PUF proposal based on test chips will be given.

2.1 Measurement Results.

It is important to clarify the environmental conditions in which the measurements are done. As we know [7, 91, 92] environment influences the PUFs output a lot. The chips are tested at room temperature (20°C). More than thousands of measurements are done with ten ATmega1284 microcontrollers under different conditions (various power-off times from 1 second to 1 hour and output can be pre-initialized). Each time SRAM cells were dumped and saved into log files using hexadecimal format. Statistical analysis of the log files will be given in this section. Therefore, the response binary sequence is 131072 bits which is retrieved from 16KByte memory of the chip. The initial settings are room temperature, 30 min power-off time, random state of memory (unchanged before powering on). These settings were used to determine the response. The results of further measurements will be compared with the initial measurement and error rate will be estimated.

2.1.1 Mean value

The unpredictability of the output is one of the main PUF properties. Thus the values of the SRAM cells "0" and "1" should be distributed uniformly, it means the amount of 0's and the amount of 1's should be equal. In an ideal case the arithmetical mean value of the SRAM cells should be around 0.5 and defined as:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2.1}$$

where n is the number of cells and x_i is i^{th} bit in the response string.

Mean value of the ten test chips are calculated with *mean.pl* script (see the enclosed CD) and the results are represented in Tab. 2.1.

Chip	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
μ	0.71	0.71	0.72	0.72	0.71	0.72	0.73	0.76	0.71	0.72

Table 2.1: Mean value of ten test chips

If the output values "1" and "0" of the SRAM cells are distributed equally then the mean value of different chips is in a binomial distribution. The probability density function for the binomial distribution is defined as

$$\binom{n}{k} p^k (1-p)^{n-k}, \tag{2.2}$$

where n is the number of cells in SRAM (n = 131072), k is the number of occurring "1" (0..n), and p is the probability that "1" occurs (p = 0.5).

Fig. 2.1 shows the results of ten test chips on the binomial probability density function. Since the response of the chips is distributed binomially, the mean values are placed on top of probability density function.

All mean values are close to each other but a bit far from the ideal value $\mu = 0.5$. A strong bias toward "1" can be noticed. The bias can effect the predictability. Further measurements will allow to understand the bias and exclude or minimize the effect. But even such deviation can be accepted for some applications and it depends on what the data are used for.

It would be good to figure out the binomial proportion confidence interval (CI). It can help us have a better understanding of how strong the bias is and how far is the mean from the ideal value. CI relies on approx-

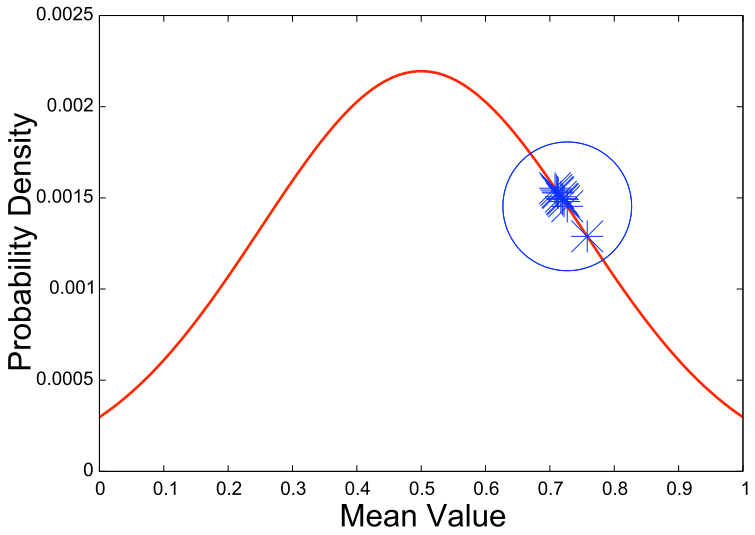

Figure 2.1: Mean values plotted against the binomial density probability curve. The measured mean values (in the circled area) are somewhat far away from the ideal value of 0.5.

imating the binomial distribution with the normal distribution and can be determined as follows:

$$CI = p \pm z_{1-\alpha/2}\sqrt{\frac{p(1-p)}{n}}, \tag{2.3}$$

where p is the probability of occurrence "1", n is the number of bits in the bit string (n = 131072), $z_{1-\alpha/2}$ is the $1-\alpha/2$ percentile of a standard normal distribution, α is the error percentile.

A 95% confidence interval can be considered as the accepted level therefore the error α is 5%, and $1-\alpha/2 = 0.975$. Using the table of Laplace transforms $z_{1-\alpha/2} = 1.96$. Since the expected probability $p = 0.5$, Formula 2.3 gives us:

$$CI = 0.5 \pm 1.96\sqrt{\frac{0.5(1-0.5)}{131072}} \approx 0.5 \pm 0.003 \tag{2.4}$$

Therefore, all the mean values of the chips are far from the confident interval.

2.1.2 Error rate

One of the main PUF requirement is a stable response under a wide range of conditions. Every request/response procedure should produce the same output, or at least have a negligible deviation. The error rate, commonly expressed as the Bit Error Rate (BER), can be estimated by calculating the Hamming distance (number of different bits between a pair of values).

The measured value is called intra-chip Hamming distance HD_{intra}. We suggest to use the first PUF output as the basis of comparision with all futher PUF outputs. Hamming distance here is the difference between the current output and the first one. An ideal PUF should have $HD_{intra} = 0$, but from practical measurements we can expect the HD_{intra} between 5 and 20 percent over the whole region of operations [4, 22]. Our measurements are done at room temperature, power-off time is 30 min, memory is used as it is. Tab. 2.2 contains the results:

Chip	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
min HD_{intra}	2%	2%	2%	4%	2%	2%	2%	2%	2%	1%
max HD_{intra}	2%	2%	2%	4%	2%	2%	2%	2%	25%	2%
avg HD_{intra}	2%	2%	2%	4%	2%	2%	2%	2%	2%	2%

Table 2.2: Intra-chip Hamming distance of ten test chips

From Tab. 2.2 we can observe very stable output for all test chips. HD_{intra} is near 2%. Only once max HD_{intra} reaches a 25% level which could be connected with a data transfer error.

Another measurement that could be interesting for estimating the error rate of the SRAM PUF is HD_{intra} calculated for all test chips in the whole region of operation, even when including variations of the power-off time (from 2 sec to 1 hour), pre-initialization of the memory cells with zeros, and small temperature variations. This measurement should show how stable the output of SRAM cells is. And if there is any noise influence on it. More than 500 tests are done for each chip and the results are shown in Tab. 2.3:

Chip	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
min HD_{intra}	2%	2%	1%	1%	2%	1%	1%	1%	2%	1%
max HD_{intra}	2%	25%	9%	21%	2%	9%	20%	9%	9%	26%
avg HD_{intra}	2%	4%	8%	11%	2%	7%	11%	9%	3%	10%

Table 2.3: Intra-chip Hamming distance of ten test chips over all range of operation

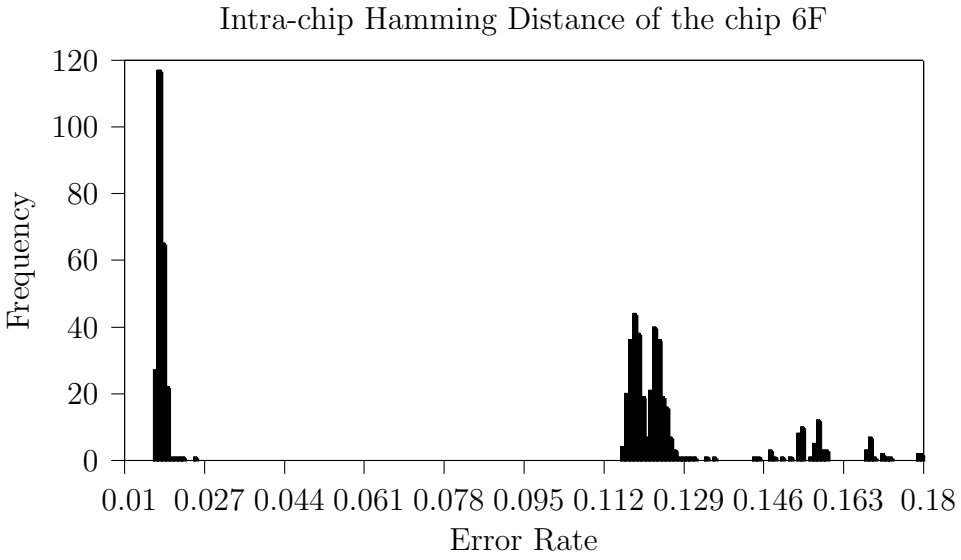

Figure 2.2: Error rate of PUFs. Intra-chip Hamming distance of chip 6F

The more tests the bigger dispersion of the output bit strings. The average values of all chips are in and around the accepted range of 5-20% [4, 22]. But the maximal values are quite high sometimes. From the frequency distribution of the values we can observe how often it happens.

Fig. 2.2 shows the frequency distribution of HD_{intra} values of the chip 6F. It was chosen because a rather high value is observed in Tab. 2.3. We can see that one group of values is around 2% level and the other one is around 11-18%. It may have happened because of the environment influence but which parameter had an effect on it will be figured out in the further experiments. Nevertheless all the parameters are in the accepted range.

The error rate is an important parameter of PUFs properties which is used in the selection of the suitable error correction codes in the next section.

2.1.3 Correlation between bits

Detectable correlation between cells of the same SRAM chip would result in reducing the number of unique combinations and would decrease the security resistance to brute-force attacks. In an ideal PUF cells must not affect each other. Here we analyze the PUF output as binary sequence for the Low periodic autocorrelation and Good aperiodic autocorrelation [73].

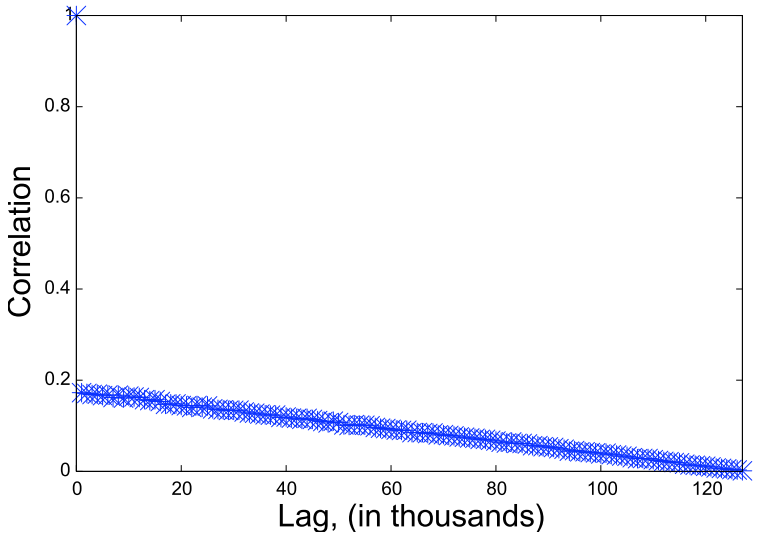

Figure 2.3: Autocorrelation of the SRAM PUF chip 6F

The periodic autocorrelation of a binary sequence a=$\{a_i\}$ of period N is defined by

$$C_a(\tau) = \sum_{i=0}^{N-1} (-1)^{a_i + a_{i+\tau}}, 0 \leq \tau \leq N-1 \tag{2.5}$$

$C_a(\tau) = \pm 1$ says about the correlation between bits and in contrast, $C_a(\tau) = 0$ does not show any correlation at lag τ.

The aperiodic autocorrelation of a binary sequence a=$\{a_i\}$ of length N is defined by

$$A_a(\tau) = \sum_{i=0}^{N-1-\tau} (-1)^{a_i + a_{i+\tau}}, 0 \leq \tau \leq N-1 \tag{2.6}$$

In the design of binary sequences, the low aperiodic autocorrelation is known to be much more difficult to achieve than the low periodic autocorrelation [93]. But anyway we should analyze both of them.

We may observe slow decreasing function with respect to the lag. The bits are correlated only at lag $\tau = 0$, which is obviously the bit correlated with itself. For small lag values the correlation is around 0.19, and for big lag values the correlation is even less and is equal to 0. This is true for all the test chips. So we may conclude that the correlation is negligible [22] and the chips are suitable.

2.1.4 Memory effect

The measurement shows how previously written data affect the output at the next readout. Two tests were prepared. In the first one SRAM PUF cells were pre-initialized. After two seconds the chip was powered on and memory state was dumped. The second test is the same as the previous one but in the first step the memory is untouched. Compare the results of the two tests in Tab. 2.4.

Chip	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
test A, avg μ,%	71	73	74	73	72	73	74	76	73	73
test B, avg μ,%	70	73	74	74	71	72	74	75	72	73
A, avg HD_{inra},%	2	4	4	4	2	4	4	4	3	3
B, avg HD_{inra},%	2	3	4	4	3	3	4	3	3	4

Table 2.4: Memory effect measurements

Memory effect is not observed in Tab. 2.4. Neither the bit error rate nor the mean values of SRAM PUFs have influence at room temperature.

2.1.5 Aging effect

Studies [9, 46] show the impact of aging on different type of circuit components. We perform power-off time analysis to observe the effect of aging on SRAM PUF. A test chip is taken and HD_{intra} and μ are measured for different values of power-off time (from 2 sec to 30min).

The parameters are stable enough all the power-off time round (see Fig. 2.4). There is no influence of power-off time. Perhaps there is some influence [6] but in micro-seconds dimension which can not be measured due to lack of technical resources.

2.1.6 Correlation between the chips

As we already know from the theoretical part of the work any PUF has two main properties. The first one is the stable response and has no correlation within internal structure. This property was discovered in the previous measurements. The second main property of PUF is uniqueness. PUF has to produce such output which should definitely identify PUF. Roughly speaking there is no the same PUF output.

The correlation between different chips can be analyzed by calculating inter-chip Hamming distance HD_{inter}. Ideal PUF should have HD_{inter}=50%.

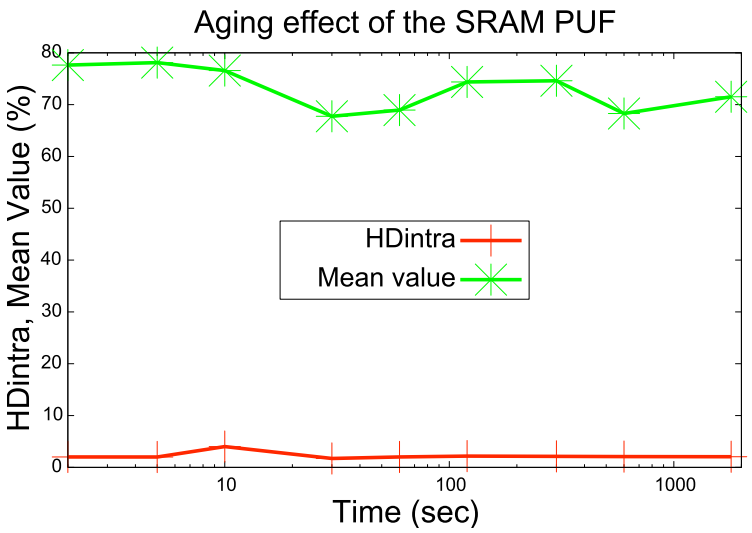

Figure 2.4: SRAM PUF on power-off time scale chip 6F

There is no correlation between chips if the mean HD_{inter} of all possible chip pairs is 50%.

The measured HD_{inter} of the ten test chips is shown in Tab. 2.5.

Chip	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
4F		41%	41%	41%	41%	41%	40%	40%	40%	41%
5F			40%	40%	40%	40%	40%	39%	40%	40%
6F				40%	40%	40%	39%	38%	40%	40%
7F					40%	40%	39%	38%	40%	40%
8F						41%	40%	39%	40%	40%
9F							40%	39%	40%	40%
AF								38%	40%	40%
CF									39%	39%
DF										40%
EF										

Table 2.5: Inter-chip Hamming distance of ten test chips

The mean value of all HD_{inter} is 40% which is a bit far from the optimal value of 50%.

An alternative approach to detect whether correlation between chips occurs is to calculate mean values over all available chips [2]. The previously shown method is more common and preferable. But it does not identify the

same output over all chips. Since we suspect the test chips of the same response in some bits it would be good to choose another approach for inter-chip correlation detection. A suitable approach is to calculate the mean value of each bit in the response over all test chips:

$$\mu(j) = \frac{1}{N} \sum_{i=1}^{N} x_{ji}, \tag{2.7}$$

where N is the amount of chips, j is a sequence number of bit in the PUF response. The results are represented as a heat-map diagram in Fig. 2.5.

In our test scenario, 2.64% cells (black) have the same output over all chips and over all operations. There are no cells that have never had "1". And just a thousandth of one percent have had "1" a few times. Most of SRAM cells changed their state to "1" in more than 50% of the tests. It confirms the results of the previous measurements about a strong bias toward "1". Frequency distribution of the PUF cell with respect to bitwise correlation is shown on Fig. 2.6.

We may notice a strong bias toward "1" again. It confirms the results of the previous measurements. There are no cells that have never been used. A thousandth of one percent have been used just a few times. Most of SRAM cells are accessed in more than 50% of the tests. From the previous measurements we know that 70% of SRAM cells are pre-initialized with "1" after powering up. At the best they should be around 50% and figure with Inter-chip bitwise correlation should be like density function of binomial distribution.

2.1.7 FAR and FRR

Assume we have a database on a server. The database stores the chip identities (fingerprints). These fingerprints were obtained during enrollment phase when all chips are tested. During the identification phase a chip sends request to the server. The server compares ID from the request with database entries. The chip is identified if there is an entry in the database when the Hamming distance between its value and request ID is below predefined values. If there is no such entry in the database the server will reply with rejection.

Some errors can happen:

1. if the accepted level is too high then a wrong chip will be identified. False Acceptance.

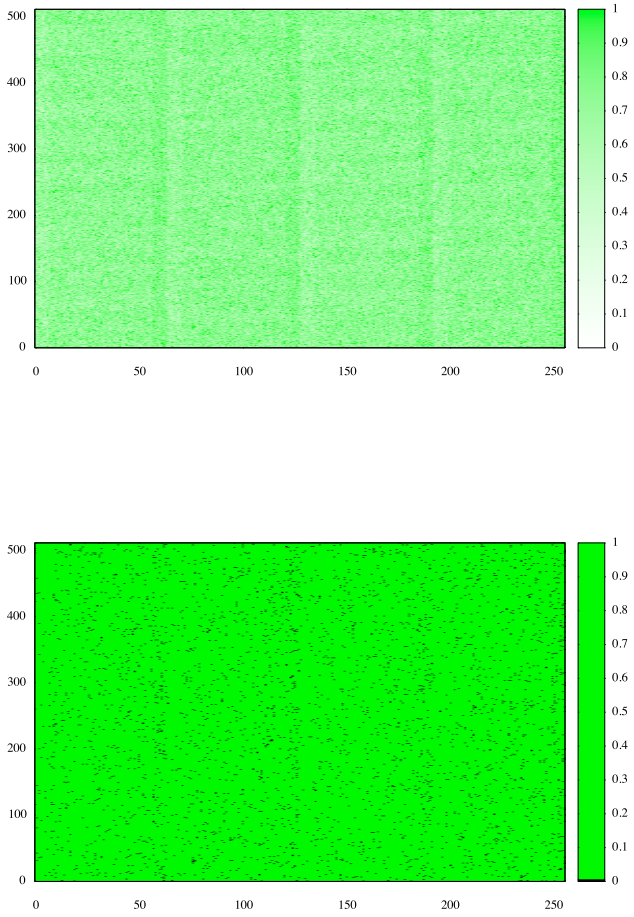

Figure 2.5: Inter-chip bitwise correlation of SRAM PUF cells. a) Intensity of writing b) Stable cells over all test chips

2. if the accepted level is too low then no entries will be found. False Rejection.

Different approaches can be found in literature [22,33] False Acceptance Rate, False Detection Rate and False Rejection Rate. Performing these measurements help to describe identification performance of a system [2].

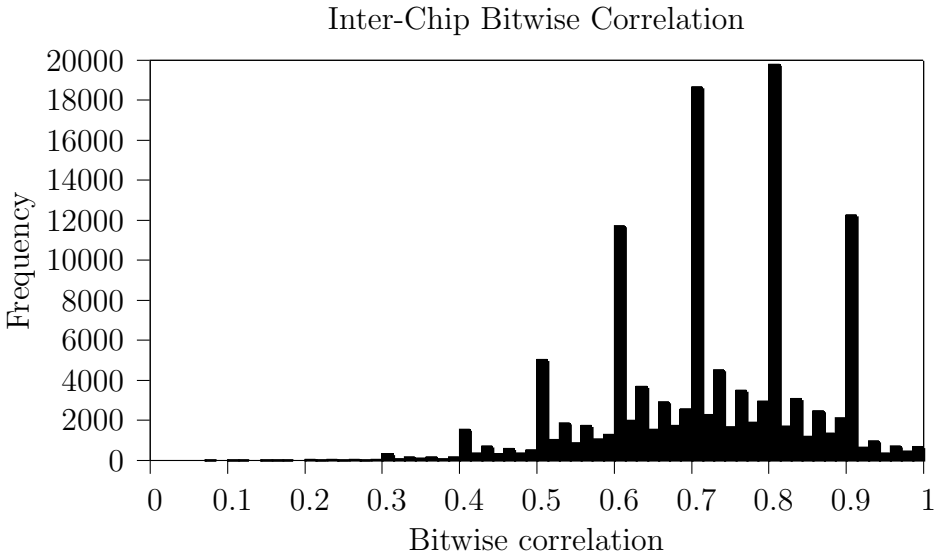

Figure 2.6: Inter-chip bitwise correlation of SRAM PUF cells

False Acceptance Rate:

$$FAR = \frac{NFA}{NAA} * 100\%, \tag{2.8}$$

where NFA is the number of false accepted attempts, NAA is the number of attacker attempts.

False Rejection Rate:

$$FRR = \frac{NFR}{NIA} * 100\%, \tag{2.9}$$

where NFR is the number of false rejections, NIA is the number of identification attempts.

The relationship between FAR, FRR and Hamming distance is shown in Fig. 2.7.

The green doted line represents the threshold for correct FAR and FRR. Moving the line to the left: wrong IDs may not be accepted but the number of rejections increases. Moving the line to the right: any ID may be accepted.

In our case the relationship between FAR and FRR is shown in Fig. 2.8

[17]Retrieved from [2]

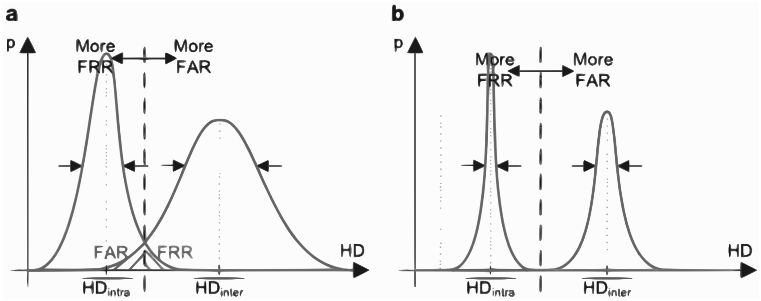

Figure 2.7: FAR and FRR: a) Small number of SRAM PUF cells b) Large number of SRAM PUF cells [17]

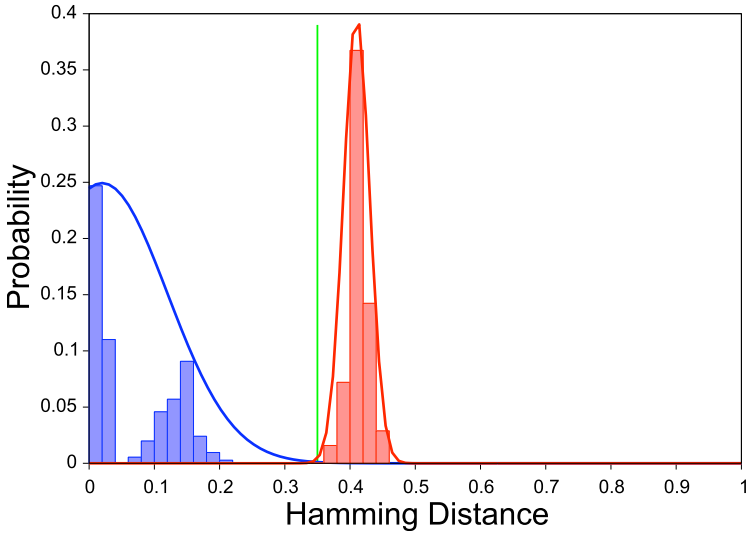

Figure 2.8: Intra-chip and Inter-chip Hamming distance. The vertical line represents the Equal Error Rate

According to the measurements the threshold is around 35%. This point is called the Equal Error Rate. It means the server can accept chip ID with 35% deviation in Hamming distance value. The values can be obtained algebraically. Suppose we deal with Gaussian distribution then the following parameters can be derived from test data:

$HD_{intra} : \mu_{intra} = 6.741\%; \sigma_{intra} = 5.957;$

$HD_{inter} : \mu_{inter} = 41.467\%; \sigma_{inter} = 1.416;$

FRR and FAR can be expressed as follows:

$$FRR = \frac{1}{\sigma\sqrt{2\pi}} \int_{HD_{max}}^{\infty} e^{-\frac{1}{2}\left(\frac{x-\mu_{intra}}{\sigma_{intra}}\right)^2} dx * 100\% \tag{2.10}$$

$$FAR = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{HD_{max}} e^{-\frac{1}{2}\left(\frac{x-\mu_{inter}}{\sigma_{inter}}\right)^2} dx * 100\% \tag{2.11}$$

For $HD_{max} = 35\%$ FAR and FRR can be derived from 2.10 and 2.11 by substitution variables $\mu_{intra}, \mu_{inter}, \sigma_{intra}, \sigma_{inter}$: $FAR \approx 0.0002$ and $FRR \approx 0.0001$.

2.1.8 Summary

The summary table A.1 in Appendix represents the measurement result from the 10 test chips. According to the results, the initial SRAM state of chips is suitable to be used as a PUF. All measured statistical properties are within feasible ranges. The results of the correlation between bits and correlation between chips turned out to be very good. The response output of chips is quite stable since intra-chip Hamming distance stays small. The results are not influenced by power-off time or memory pre-initialization before power-off. However, a strong bias on the output toward "1" is observed that can effect the predictability since the number of likely combinations reduces.

Close similarity of the results show that all ten chips were designed in the same way.

The following section covers Error Correction techniques that allow correct errors in the PUFs responses and make them more stable.

2.2 Error correction

The output of SRAM PUF is very noisy. Depending on the environment PUF cells show error rate of 10% and more [36]. This fact is confirmed by scientific researches [74] and our measurements in the previous section (BER is around 3% with the maximum level of 16%). Since we can uniquely identify each chip by estimating its Hamming distance only thus for some applications e.g. identification, such noisy PUF output is more than enough. For the most part of applications e.g. key generation, authentication, such PUF output can not be accepted. Thus some techniques for the response stabilization should be applied.

2.2.1 Approach: Using more than one PUF cell for the error reduction

Here we suggest to compress the response of 3 SRAM cells into 1 PUF bit. The output should be determined by the majority decision. It means the states of three cells: 000, 001, 010, 100 will produce "0" at the output. Otherwise, "1" is generated for the states: 111, 110, 101, 011. Simulation results will answer the question: does this method reduce errors in the output. The measurements are done for the combining of 3, 33 and 99 SRAM cells. The results are listed in Tab. 2.6

bits	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
1	2.0	2.0	8.3	10.5	1.9	7.0	10.4	8.8	1.9	9.9
3	2.0	2.1	9.0	10.4	1.9	9.0	13.4	9.0	1.9	10.0
33	5.0	4.1	13.3	16.0	5.1	11.7	17.3	11.2	4.2	14.8
99	9.2	8.5	16.1	19.8	7.1	16.6	21.2	14.2	8.5	17.1

Table 2.6: Average BER in %, before and after majority decision

The results in Tab. 2.6 show that there is no improvement in the bit error rate after combining SRAM cells. Always the bit error rate increases after the majority decision. Moreover the more bits are combined the worse results are achieved. The results for the 99 bits are even worse than for 33 and much more worse than for 3 bits combining.

The situation can be described algebraically to prove or disprove the simulation results. Assume that p is the probability of error occurring in one bit. Then the probability of error occurring after 3-bit combining has to be sum of probabilities:

$$p = \frac{1}{2}p_0 + \frac{1}{2}p_1, \tag{2.12}$$

where p_0 is the probability of error occurring at nominal "0" output, p_1 - at nominal "1". Since $p_0 = p_1$ then $p = p_0$

For p_0 is true:

$$\begin{aligned} p_0 &= p(error\ in\ 000, 001, 010, 100) = \\ &= p(error\ in\ 000) + p(error\ in\ 001, 010, 100) = \\ &= \frac{1}{4}\left(3(pp(1-p)) + ppp\right) + \frac{3}{4}\left(2p(1-p)(1-p) + pp(1-p) + ppp\right) = ... \\ &= \frac{3}{2}p - \frac{3}{2}p^2 + p^3 \end{aligned} \tag{2.13}$$

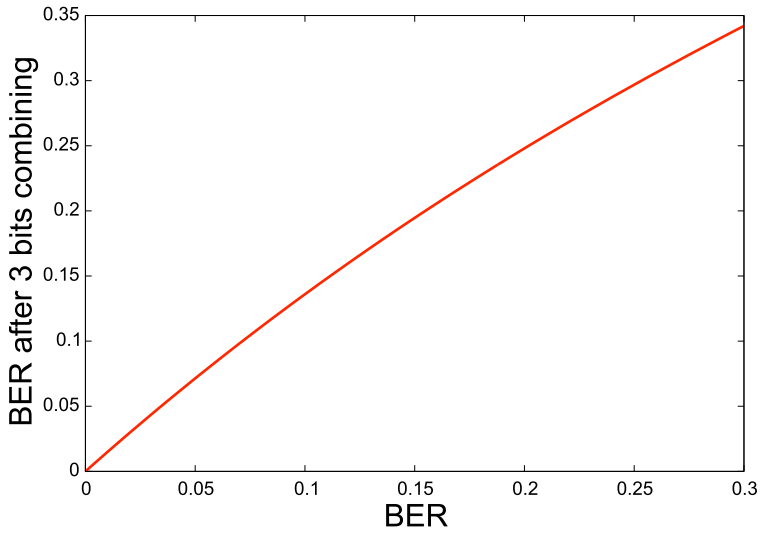

Figure 2.9: Using more than one PUF cell for the error reduction

From 2.12 and 2.13 the probability to meet error after combining of 3 cells is $p = p_0 = \frac{3}{2}p - \frac{3}{2}p^2 + p^3$ which is a bit higher than without cell combining for the bit error range between 0% and 30% (see Fig. 2.9).

Now algebraically proved and shown in our measurements that it is not possible to achieve better results in the output stabilization by combining some SRAM cells logically.

2.2.2 Averaging output

Over the previous measurements the Hamming distance was calculated as follows: response of SRAM is compared with the first one. In case the first SRAM output is the noisiest and full of errors then all next measurements should have the high bit error rate. In this approach the first output to appear will be replaced with the average output from set of measurements. Table 2.7 contains the results of BER measurements over all test chips when the average value of 10 and 100 outputs was taken.

Table 2.7 shows an improvement after applying the technique of averaging first N SRAM outputs. Note that this technique does not exclude errors but allows to mitigate the effect of the noisy responses. The big disadvantage is the complexity of usage - it could be a bit pricey to perform many tests during initialization phase. Therefore, the previous two hypotheses show that it is not possible to reduce the bit error rate by algebraic

avg of	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
1	2.06	2.04	8.32	10.47	1.91	7.07	10.42	8.82	1.95	9.93
10	2.06	2.04	5.97	6.22	1.93	2.13	4.03	2.25	2.07	3.91
100	2.06	2.04	2.31	2.47	1.92	2.13	3.15	2.25	1.96	2.48

Table 2.7: BER after averaging output, %

manipulation of output cells without helper data. Another approach should be implemented to avoid errors at SRAM output.

2.2.3 Hamming codes

Previously considered methods of the error reduction didn't allow to make the output of SRAM more stable. Next approaches will create the helper data that will be used for the recovering the SRAM PUF output. The main idea is that in order to provide the stable output we need to include some extra information.

The Hamming code [94], and specifically the version with an additional parity bit Hamming(8,4) could be applied for the test chips. *Hamming(N,n)* is the hamming code of N-bit length long and containing n-bit of data. Thus the redundancy in the code is $R = \frac{N}{n} = \frac{8}{4} = 2$.

Algorithm. The SRAM PUF output is divided into 8-bit binary strings. For each of them the helper data is calculated. The helper data contains 4 parity bits for the error correction. The applied technique corrects only one error in 8-bit binary string. It means that the ideally if the SRAM output has only one error in every 8 bit sector of its memory then Hamming(8,4) can correct errors in the PUF outputs with $BER = \frac{1}{8} \approx 12,5\%$.

The practical implementation of the Hamming Codes approach on the ten test chips gives the results in Tab. 2.8.

BER	Ham(8,4)	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
min	before	1.9	1.9	1.8	3.6	1.8	1.6	1.7	1.7	1.9	1.7
max	before	2.1	2.1	13	17	2.0	11	18	14	2.0	16
avg	before	2.0	2.0	8.3	10	1.9	7.0	10	8.8	1.9	9.9
min	after	0.1	0	0	0.6	0	0	0	0	0.2	0.2
max	after	1.0	0.9	9.0	15	1.1	9.1	14	13	1.9	9.7
avg	after	0.8	0.8	4.3	6.7	0.6	4.2	6.0	5.9	1.1	7.3

Table 2.8: BER after Hamming(8,4), %

The first thing to notice is that errors are distributed uniformly over all SRAM memory. It can be observed from the decreasing values of BER (HD_{intra}). All the test chips have better output after applying the technique than it was before. In some cases it allowed to correct errors completely. But we know that theoretically possible error correction threshold is around 12.5% because Hamming(8,4) corrects only one bit in an 8-bit binary string. Since we have some values of BER bigger than the threshold (Tab. A.1) not all PUF outputs can be corrected. Therefore this technique is not acceptable for some use cases.

2.2.4 BCH code

BCH Code is another block code approach of the error correction codes. The power of such a code depends on the block length and the helper data length. An example of BCH Code is the Reed-Solomon code, that has good error correction rate especially with regard to burst error. But since it is shown in the previous section (Hamming Codes) the SRAM output of the test chips contains uniformly distributed errors then the Reed-Solomon code is not suitable for the SRAM PUF.

In $BCH(n, l, d_{min})$, n is the block length, l is the length of the original data, d_{min} is the amount of parity bits which is the minimal Hamming distance between the code words. BCH corrects $(d_{min} - 1)/2$ errors. As an example BCH(15,6,5) is able to correct 2 bits of error in a block of 15 bits.

It's very important to select correct BCH code values that will show the best results. It usually depends on the type of processing data. But even at its best result we can not correct all errors in the PUFs cells. Because theoretically possible threshold is around $2/15 \approx 13\%$ and the practically achievable result is around 6.21% [2].

Also we should mention high complexity of BCH Code implementation and not all micro-controllers are capable of handling the complexity of it [95].

2.2.5 Repetition code

The repetition code is a very simple error correction code in comparison with BCH and Hamming codes. The idea behind the Repetition code *R(n)* is to repeat transmitted bit n times [96, 97] and then make the majority decision. The repetition code is described as very powerful error correction code that reduces error rate up to BERs of 50% [2]. But such performance takes the high redundancy value, $R(N) = \frac{N}{n} = N$. For example, the R(7) has redundancy coefficient 7 and capable to correct errors in 3 bits. Clearly

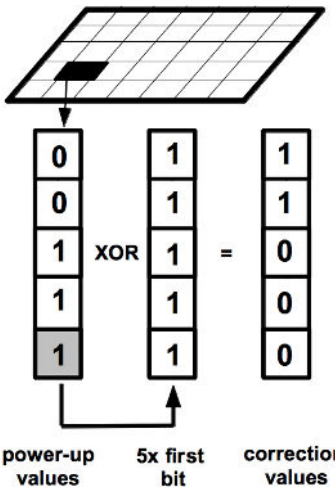

Figure 2.10: Repetition code. Initialization.

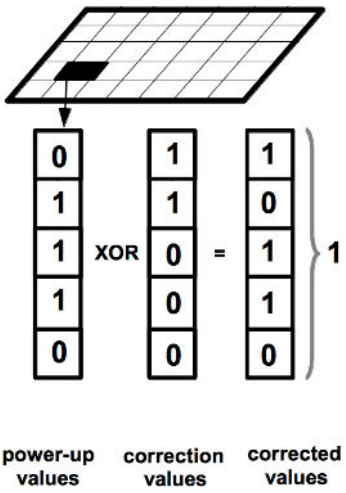

Figure 2.11: Repetition code. Key generation.

the repetition code is the best error correction technique for the PUFs, but only if there is enough memory to store redundancy data.

The following principle can be proposed as the repetition code implementation on SRAM cells (Fig. 2.10): at the initialization phase some sequential read-outs are done. The result of it are the binary strings of power-up values od each SRAM cell. Then an auxiliary sequence generated by the first bit of the string. Then the XORing of power-up sequences and the auxiliary sequences generates the error correction data. During the next readouts just generated error correction bits will be used to reduce the error rate.

Fig. 2.11 shows the nominal readout phase: the error correction bit string is XORing again with the new power-up values. The major decision of the resulting sequence is one output bit. Fig. 2.11 represents a five bit repetition code R(5). It has reduction coefficient of 5 and capable to recover 2 bit errors. If there are more than 2 errors in the sequence the wrong output bit is generated.

In Tab. 2.9 the performance of different repetition code length is shown: 3 bit, 7 bit, 15 bit and 31 bit.

The bit error rate is decreasing slowly with regards to repetition factor. The first line in Tab. 2.9 is the repetition code with length 1 bit. It means no repetition code is applied.

The result of repetition code applying on the ten test chips is shown in Tab. 2.10:

The output of all chips becomes more stable (0.13-0.29%). Some exper-

Length	BER, avg
R(1)	2.06
R(3)	1.67
R(7)	0.81
R(15)	0.21
R(31)	0.09

Table 2.9: BER of the chip 4F after Repetition code, %

R(15)	4F	5F	6F	7F	8F	9F	AF	CF	DF	EF
before	2.06	2.04	8.32	10.47	1.91	7.07	10.42	8.82	1.95	9.93
after	0.21	0.19	0.22	0.31	0.07	0.19	0.15	0.29	0.25	0.13

Table 2.10: BER before and after Repetition(15), %

iments succeeded to get almost errorless output after applying repetition codes [91]. As a result the repetition code shows the best performance but also the highest redundancy over all error correction codes. And it can be easily implemented on any type of chip. At the same time BCH and Hamming codes are not computational feasible for many chips.

We can judge from the results of the experiments that some SRAM cells have randomly unstable states. Resulting from the errors after the repetition with long length. Perhaps it is a good idea to mark such cells and exclude them from the PUF, in other words to perform preselection.

2.3 Preselection

ECC can be considered as post-processing approach that can be implemented in order to correct the number of errors in the PUFs output. It looks reasonable to implement some kind of pre-processing techniques that reduce BER in a way that error correction gets less complex or even unnecessary.

Preprocessing techniques work during the initialization phase and can require to have some helper data that will be used in the correction of errors later. The main advantage of preprocessing approaches compared to ECC is that no additional complex calculations are neccessary during readout phase.

Different pre-processing approaches are proposed in the literature [2, 6, 7, 11]. Most of them such as PUF-parallelization, PUF-biasing, Preselection techniques require low-level hardware access.

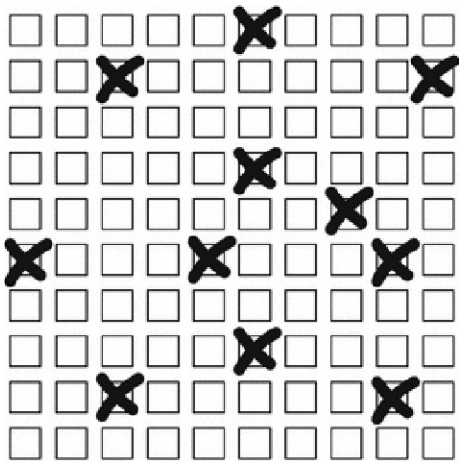

Figure 2.12: Concept of preselection techniques[18]

During preselection only those cells are selected that generate a stable result under any conditions. (various temperature, power-off time, voltage and etc). We can expect the cells to change their state once they are likely to change it again. The cells which always generate the same output are called stable and correspondingly the cells producing the erroneous output are called unstable.

Obviously unstable cells can be detected during the initial phase by crossing them out from the memory array (Fig. 2.12). The result of preselection is decreasing the bit error rate. Here a new parameter of PUF performance - efficiency can be introduced and can be calculated as:

$$\varepsilon = \frac{number\ of\ all\ cells}{number\ of\ stable\ PUF\ cells} \tag{2.14}$$

Marking unstable cells and their future excluding from the PUF output are done by means of multiple readouts. SRAM memory measured several times and the cells that provide the same output over all readouts are marked as stable. The rest of the cells are not used any more. This simple measurement is done without getting into hardware level.

Table 2.11 contains the result of detecting stable cells in the test chips:

All chips have nearly the same statistical characteristics. The average efficiency over all test chips is around 90% which is quite a big value and tells us that about 10% of SRAM memory have random nature. The heatmap of SRAM state of several test chips are shown in Fig. 2.13.

[18]Retrieved from http://www.springerimages.com

Parameter	4F	5F	6F	7F	8F
stable 0	20.32%	20.01%	20.73%	24.21%	24.35%
stable 1	67.78%	68.17%	68.29%	67.47%	67.33%
efficiency	88.10%	88.18%	89.02%	91.68%	91.68%
Parameter	9F	AF	CF	DF	EF
stable 0	23.78%	20.73%	21.98%	22.37%	21.77%
stable 1	67.21%	67.90%	68.28%	67.50%	68.31%
efficiency	90.99%	88.63%	90.26%	89.87%	90.08%

Table 2.11: Preselection multiple readout

The black spots in Fig. 2.13 are stable cells that can be used as the PUF output. The light spots are unstable cells which change their state from time to time. It may be noticed that unstable cells are distributed over all SRAM and not accumulated in one region. Measurements show that 90% of SRAM cells can be used in the PUF output. Studies show inverse proportionality of efficiency to stability [2]. Typically, the higher the efficiency, the lower the stability. Thus the application should define required PUF cells and stability factor.

Another interesting question of preprocessing techniques is whether all test chips provide stable output for some SRAM cells. Then these cells can be taken out of further use. Measurements show that even for ten test chips there are not so many stable cells. The result is in Tab. 2.12 and in Fig. 2.14.

Parameter	Value
stable 0	0%
stable 1	2.64%

Table 2.12: Number of stable and unstable cells over all test chips

At a first glance, the preselection technique based on multiple readout seems to be a good approach but in practice different problems occur. First, measurements confirm a strong bias toward '1' and at that time all test chips have stable SRAM output with the number of stable cells of 90% that leads to predictable PUF output. Amount of stable cells over all test chips under different conditions measurements can not allow to reduce the number of useless cells in the PUFs output. Moreover in order to get useful statistical data many readouts should be done which has proportional impact on the production cost [6]. There are some preselection approaches

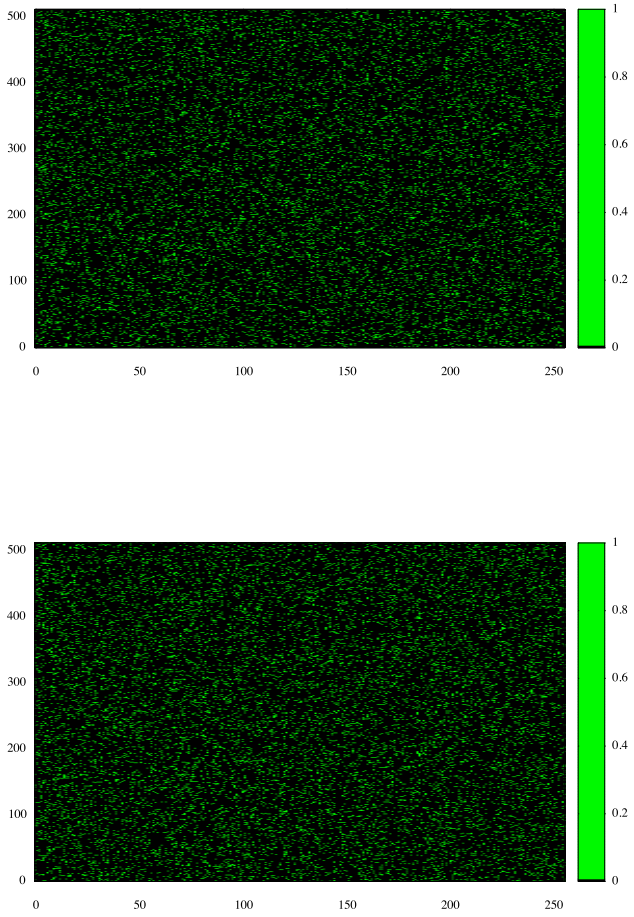

Figure 2.13: Heatmap of memory state a) chip 6E b) chip AE. Each pixel represents one SRAM cell. Black pixels represent stable cells, light pixels represent unstable cells.

such as parallellization of PUF cells, PUF biasing, preselection using precharging and time based preselection [2]. But all of these methods require an access on low hardware level.

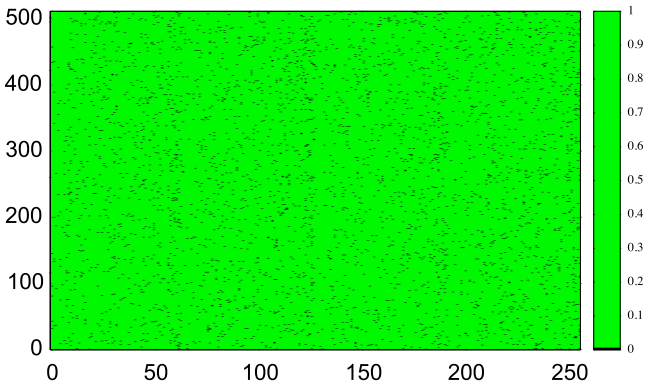

Figure 2.14: Stable (black) cells over all test chips

2.4 PUF-design proposal

So far we have analyzed some measurements of initial SRAM contents (Paragraph 2.1). In order to correct errors the pre- and post- processing of the SRAM response are proposed (Paragraph 2.2, 2.3). After all, we have enough information in order to suggest a method for constructing a PUF based on such initial SRAM contents. And later we propose some possible usage of it.

2.4.1 Concept of the SRAM PUF without ECC

Before starting an in-depth study of the SRAM PUF constructing it would be good to answer a very simple question: Is it possible to use the SRAM cells to construct a PUF without any post- or pre-processing techniques? The answer is yes, but it is true within limits - just a few applications can be proposed for given ten test microcontrollers.

According to the statistical parameters of the test chips, the initial power-up values of SRAM cells can be used as a PUF output for chip identification. Since the maximal HD_{intra} does not exceed 25% value (see Tab. A.1 in Appendix) and at the same time the minimal HD_{inter} is at least 39%, all tested chips can be uniquely identified. More than 2000 test measurements are done and we may be fully confident of that.

All test chips can be identified by using at most 10 bits (see Tab. 2.13). But in general it is recommended to store a full dump of stable SRAM cells that determine a unique sequence because the number of stable bits over the whole region of operations decreases with increasing the number of observed test chips. Over the whole region of operations for the test chips we can select 44035 cells (33.6% of total). Later, if we exclude repeated from them, we will get 22.6% of cells which should be stored in database.

bit	4E	5E	6E	7E	8E	9E	AE	CE	DE	EE
2720	0	1	0	0	0	0	1	0	0	0
6025	0	1	0	0	1	0	0	0	0	0
14043	0	0	0	0	0	1	1	0	0	0
19200	0	0	0	0	0	0	0	1	0	0
29607	0	0	0	0	0	0	0	1	0	1
42816	0	1	0	1	0	0	0	0	0	0
61001	1	1	0	0	0	0	0	0	0	0
89249	0	1	0	0	0	0	0	0	1	0
97604	0	0	1	0	0	0	0	0	0	0
122565	0	1	0	0	0	0	0	0	0	0

Table 2.13: Numbers of bits identify the ten test chips

We have reason to believe that the amount of stable bits over the observed test chips will end (see Tab. 2.14). Even sooner they will not be enough to identify the chip uniquely. But it can be seen in Tab. 2.14 that the Hamming distance between the SRAM output of different chips is stable throughout the measurement range. Consequently we suggest to store not only stable bits but the full dump of SRAM output which is 16 Kbytes.

# of chips	# of stable bits	average HD_{inter}
2	82.9%	39%
4	51.4%	39%
8	37.7%	39%
10	22.6%	39%

Table 2.14: Amount of stable bits and Hamming distance over observed test chips.

The design of such PUF without any error correction modules should look like this. A database on a server stores the SRAM outputs (16 Kbytes

fingerprints). These fingerprints are SRAM dumps that were obtained during initialization phase. During the readout phase (identification) a chip sends request to the server. The server compares ID from the request with database entries.

The chip is identified if:

1. the Hamming distance between the values in the database and the request value is bigger than 39%,

2. there is an entry in the database when the Hamming distance between its value and request ID is below 25%.

If there is no such entry in the database the request is refused and the chip is not identified. We should mention that after the measurements of more than ten test chips the thresholds ($HD_{inter} = 39\%$ $HD_{intra} = 25\%$) can be moved closer to each other. Clearly the more test chips the closer thresholds.

2.4.2 Concept of the SRAM PUF with ECC

We ascertained that the response of the SRAM cells is erroneous and is not suitable for the most part of applications e.g. key generation, authentication. Thus some techniques should be applied in order to make the output more stable. As we showed in Paragraph 2.3 the preprocessing approach deals with access on low hardware level usually. The preselection method does not give much advantage. And also it's a bit pricey to implement it.

Therefore only the postprocessing techniques from Paragraph 2.2 could be feasible to construct quite reliable and suitable PUF. In the PUF-design we suggest to use one of the error correction codes or a combination of them.

Microcontrollers Atmel AVR ATmega1284P provide an SRAM of 16 Kbytes and nonvolatile memory of 128 Kbytes. In this case, a PUF can be implemented easily. According to the measurements such PUF has a stable response enough to order to generate keys.

The general PUF construction using ECC is shown in Fig. 1.17. The procedure of the PUF design consists of two phases: the initialization phase (enrollment) and the working (readout) phase. During the initialization phase the microcontroller is powering up and the SRAM states are readout. An auxiliary data are generated and will be used later to correct errors in the SRAM response. This data are saved in nonvolatile memory.

It's important to make sure that:

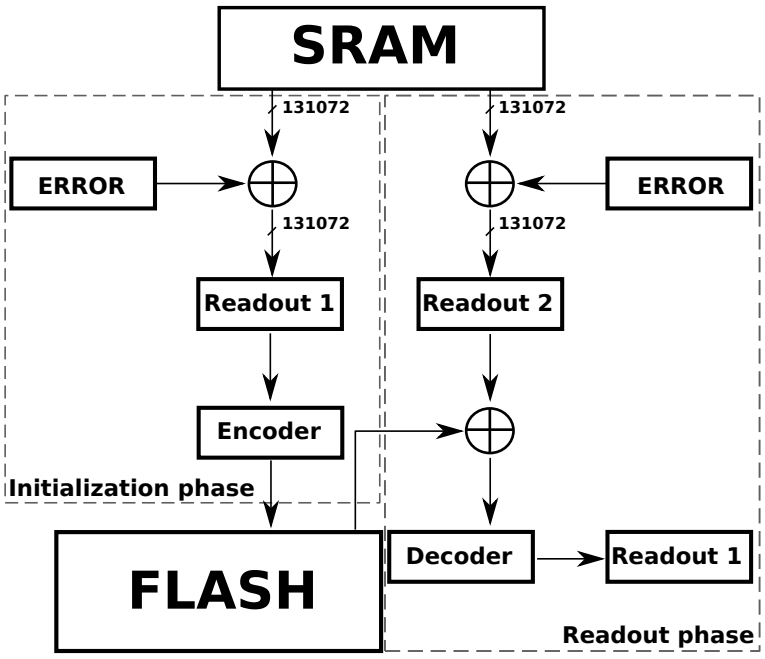

Figure 2.15: Concept of the SRAM PUF implementation with ECC support

1. Communication between SRAM and microcontroller needs to be secure otherwise it can be observed and the PUF output recovered. The securest SRAM is a SRAM inside the microcontroller.

2. SRAM are not accessible directly. It means neither any third-party software has an access to SRAM blocks nor the PUF bits are available on the microcontrollers pins.

3. Error correction code contains information of the PUF output. Recovered ECC means the PUF is known.

The concept of the PUF using SRAM is shown in Fig. 2.15. In order to implement such concept two parts of software are needed. The first one dumps the SRAM states (*Readout 1*) represented in Fig. 2.15 as the combination of error sequence and pure SRAM response. Then the *Encoder* of the Error Correction Code generates the correction stream and saves it on the nonvolatile memory (*FLASH*).

The second part of software is XORing of the SRAM output (*Readout 2*) and the error correction data from the NV memory. The error correction data is used to correct errors in the current SRAM PUF response. Therefore the result of this operation is the readout from the initial phase, *Readout 1*.

In order to define which phase should be performed during a current readout, the initialization bit can be used. E.g. if the bit is not set, the software performs the initialization phase: reads startup output of the SRAM and generates the error correction data which is stored on the flash. After all these steps the initialization bit is set.

The concept can be slightly improved to mitigate the influence of the first readout on the result. Usage of the average number of readouts is analyzed in Paragraph 2.2. Errors still remain but the technique of averaging has allowed to smooth the high bit error rates of some test chips. As a result the bit error rates of all test chips have become about 2% even for those chips which have the bit rate of 10% at the first readout.

2.4.3 ECC using the repetition code

The concept of PUF implementation using SRAM cells has been discussed in broad terms already. Now we shall go into detail and investigate such important questions, as: which correction technique should be implemented in ECC, how many SRAM cells are needed for the PUF construction, how reliable the PUF is going to be, for which applications such PUF is suitable, etc.

The summary of the error correction techniques in Section 2.2 is given in Tab. 2.15

Technique	Redundancy	Effective range [19]
Repetition, *R(n)*	n	<50%
BCH code, *BCH(15,6,5)*	2.5	<13.3%
Hamming code, *Hamming(8,4)*	2	<12.5%

Table 2.15: Summary of the ECCs

Since the bit error rate in some cases can be as high as 19% (see Tab. A.1 in Appendix), the BCH code and Hamming code can not be applied. Indeed they can be used but in combination with the Repetition code implementation. At the same time the repetition code is capable to correct errors in 50% of bits in the SRAM response. Therefore we suggest using the encoder/decoder with the repetition code in the PUF-design process (Fig. 2.15).

The initialization phase for the Repetition(7) is shown below:

[19] Theoretical upper bound. The practical values are smaller.

$$\left.\begin{array}{rl} \textbf{READOUT 1 :} & \overbrace{\underbrace{1001001}_{7\text{ cells}} \mid \underbrace{0111011}_{7\text{ cells}} \mid \ldots\ldots \mid \underbrace{0110001}_{7\text{ cells}}\ldots}^{131072\text{ bits}} \\ \textbf{7x 1st bit :} & 1111111 \mid 0000000 \mid \ldots\ldots \mid 0000000\ldots \\ \textbf{correcting code :} & 0110110 \mid 0111011 \mid \ldots\ldots \mid 0110001\ldots \end{array}\right\} \text{initialization phase}$$

The power-up values of the SRAM cells are split into 7 bits long segments (*Repetition(7)* algorithm). For each of the segments one bit is chosen as the nominal bit. We define the first bit as the nominal. The nominal bit generates the nominal sequence which just copies the nominal bit within its segment. The nominal sequence and readout produce the correction code by XORing operation. The correction code is stored in Flash memory.

The key generation phase for the Repetition(7) is shown below:

$$\left.\begin{array}{rl} \textbf{correcting code :} & 0110110 \mid 0111011 \mid \ldots\ldots \mid 0110001\ldots \\ \textbf{READOUT 2 :} & 0001000 \mid 0011001 \mid \ldots\ldots \mid 0100101\ldots \\ \textbf{XORed :} & \underbrace{0111110}_{\text{more 1}} \mid \underbrace{0100010}_{\text{more 0}} \mid \ldots\ldots \mid \underbrace{0010100}_{\text{more 0}}\ldots \\ \textbf{key :} & 10\ldots\ldots0\ldots \end{array}\right\} \text{key generation}$$

Retrieved from the Flash correction data are XORed with the current SRAM output. The ideal values after XORing are *1111111* or *0000000*. In such cases, there were no errors in the SRAM output. After that, the majority decision of each 7-bit segment is used to generate one bit. Obviously the length of the generated key is 7 times less than the number of the SRAM cells used.

Table 2.16 contains properties of the different repetition code length.

Repetition code	Key length	HD_{inter}	HD_{intra}	Error
R(3)	32768	0.31	1.8E-2	1
R(7)	16384	0.20	1E-2	1
R(11)	8192	0.12	8E-5	0.48
R(31)	4096	0.01	2.4E-10	1E-6

Table 2.16: Performance of the ECC with different repetition factor

In the 2nd column the maximum possible key lengths are shown after applying correction technique. Since the error correction code with the

repetition factor n generates *1* output bit from n SRAM cells, therefore the key length can be calculated as $\left\lfloor \frac{\#\text{ of SRAM cells}}{\#outputbits} = \frac{131072}{n} \right\rfloor$.

The average Hamming distance between different chips is presented in the 3rd column in Tab. 2.16. It can be seen that HD_{inter} converges to zero. This is the result of the strong bias on the output toward "1". The bigger repetition factor the closer keys generated by the chips. From this point of view, the repetition code of 3 bit length long is preferable to 31 bit - the Hamming distance is bigger and the range of unique chip ids is wider.

HD_{intra} in the 4th column in Tab. 2.16 shows how stable the SRAM output after correction is. That is the theoretical value which is equal to bit error correction of segments and can be calculated as the probability of error occurring in more than 50% of bits in a segment. In other words, the probability that ECC can not correct occurring errors. For the repetition factor of 7:

$$P\{\varepsilon > 3\} = \sum_{\varepsilon=4}^{7} \binom{7}{\varepsilon} \cdot p^{\varepsilon} \cdot (1-p)^{7-\varepsilon}$$

where ε is the bit error rate in the SRAM output, the average value over test chips is around 8% (see Tab. A.1). It can be seen for example, that ECC with repetition factor of 7 reduces the initial error rate of 8% to an error rate of 1%. And repetition factor of 31 reduces the initial error rate to 2.4E-10.

The key generation capabilities of the repetition factor are listed in the last column of Tab. 2.16. Since the bit error rate in the segments is known now, the probability of error occurring in the key can be calculated as:

$$P\{error\ after\ Repetition(n)\} = 1 - (1 - BER_{segment})^n$$

The correction with the repetition factor of 3 and 7 generates the key with error (keys differ in at least one bit) in 100% trials. The repetition factor of 11 allows to generate the stable key with probability of 51%. The errors remain only in 1E-4% of cases after correction by ECC with repetition factor of 31.

To conclude the measurements, the bigger repetition factor the stronger key, the smaller key length, the fewer chips can be enumerated. In an ideal PUF with key generation of 4096 bits long, 2^{4096} chips can be identified and have the stable key. But the given test chips have the strong bias toward 1 thus not all bits in the key can be used. Estimated value is around 2^{50} chips.

2.4.4 Test runs of designed PUF with ECC

In order to estimate the performance of the key generation concept, 3000 PUF outputs of ten test chips were analyzed under different conditions. The result was exactly as expected, no errors were generated by the PUF with error correction implementation. The theoretical probability of error occurrence in PUF output with 8% BER of a single bit and a reputation factor of 31, for generation of a 4096-bit key, is around 0.0001% (see Tab. 2.16).

2.5 Summary

In this chapter, the properties of ten ATMega 1284P microcontrollers were analyzed with respect to SRAM PUF properties. Mean value, error rate, correlation between bits, correlation between chips were measured. Measurements were done under different conditions - various power-off times from 1 second to 1 hour, with either no memory pre-initialization or pre-initialization with 0s. It is significant to notice a strong bias toward "1" of the state of SRAM cells, which strongly influences the SRAM PUF properties.

We should mention that the SRAM-based PUFs have high error rates. It means that for most applications e.g. key generation or authentication, such PUFs are not usable in "unprocessed" form, and some error correction approaches need to be implemented. In 2.2 the error correction codes were analyzed. It was found out that only Repetition code technique is capable to correct errors in the PUF output.

Later, two proposals for the SRAM PUF design were presented. The first concept is intended for applications of identification. And it was shown that the ten tested chips can be identified using 10-bit values. Another concept was proposed for key generation applications and contains error correction code implementation. This approach requires to store 16 Kbytes of helper data in a nonvolatile memory. If these correction data are lost, they can be regenerated easily and the key generation will continue.

Both concepts were tested at different conditions over all test chips and have shown stable results.

Conclusion

The main question of this thesis was to develop PUF-design based on power-up SRAM contents of Atmel AVR microcontrollers. This section concludes the thesis with providing a summary, an overview of results of statistical analysis, and finaly with a section on limitations and future work.

Summary

This thesis started with a literature overview of known PUF approaches and their properties. Special attention was paid to the SRAM-based PUFs which became "hot" topic nowdays since many electronic devices have embedded SRAM. The results of PUF implementations were collected from various researches and summarized to serve as references for our own PUF concept.

Furthermore, statistical analysis of the power-up SRAM contents of ten Atmel ATmega1284P microcontrollers was presented. The measurements were done under a wide variety of conditions – various power-off times from 1 second to 1 hour, with either no memory pre-initialization or pre-initialization with 0's. Mean value, error rate, correlation between bits, correlation between chips were analyzed.

The result of the analysis was used in constructing a PUF on the microcontrollers. Pre- and post-processing techniques were applied in order to meet requirements of two application scenarious: key generation and chip identification. As pre-processing techniques the simple selection methods were used. These methods did not include methods with an access on hardware level since there were no technical possibility to implement them. As post-processing techniques various error correction codes were applied. Some of them such as Hamming code and BCH were not capable to cor-

rect errors in the SRAM PUF output. And only the performance of the Repetition codes was able to stabilize the output.

Later, two proposals for a SRAM PUF design were presented. The first concept is intended for applications of identification. The PUF concept of identification was improved by adding an error correction code implementation and therefore can be used for key generation applications. The proposed concepts are reliable in the sense that if these correction data are lost, they can be regenerated easily and the key generation will continue.

Both concepts were tested in different conditions over all test chips and have shown stable results. Measurement results for the ten test chips are listed in Appendix and can be used as references for future researches.

Results

Population sampling of ten ATMega 1284P microcontrollers was estimated with respect to SRAM PUF properties. More than 3000 measurements were done under different conditions: with or without memory pre-initialization with 0's; for various power-off times from 1 second to 1 hour. The SRAM memories of the chips were dumped and prepared for statistical analysis by converting the hexadecimal ouput to the binary format.

Statistical analysis consists of evaluating parameters, such as mean value, bit error rate, correlation between chips, correlation between bits, false acceptance rate and false rejection rate. It turned out, that all measured parameters are within statistically feasible ranges. Neither the correlation between bits nor the correlation between chips was detected. The SRAM output of the chips is stable enough since intra-chip Hamming distance stays small. There is no noise influence - the results are stable with respect to power-off time and preliminary manipulation with memory. It can be concluded that the SRAM of chips are suitable to be used as a PUF.

We found out that all chips have the strong biases concentrated near "1", which influences the SRAM PUF properties. This also means that it is uncommon for most cells to start-up in different states over multiple measurements. Unfortunately, the observed effect causes a predictability since the number of likely combinations reduces.

Furthermore, it was proved that the SRAM output has high error rates. It follows from calculating intra-chip Hamming distance over multiple measurements. Later, it was shown that 8% average Bit Error Rate can not be corrected by most error correction codes, meaning that for some applications, e.g. key generation or authentication, such SRAM in "unprocessed"

form is unsuitable for the PUF data. Therefore some kind of error correction needs to be applied.

We found out that neither the output averaging technique, nor BCH codes or Hamming codes are capable to correct errors in SRAM-based PUFs, since even 25% level of BER was detected during the measurements. But repetition codes have shown very good results and proved their suitability in the PUF design. It turned out that applying the repetition code with a factor of 31 allows to achieve the stable PUF output in 99.9999% of cases.

As a result of the measurements two SRAM PUF designs were proposed. The first concept is intended for applications of identification. It was shown that the ten tested chips can be identified using 10-bit values. Another PUF concept is suitable for key generation applications. It suggests using an error correction code, especially repetition code with factor 31, which has shown good test results. This approach requires to store 16 Kbytes of helper data in a nonvolatile memory. This is temporary data which can be easily regenerated in case of loss. The output of this PUF concept is a 4Kbit long key. It looks sufficient for quite a big population of chips.

Taking into account the results of the measurements, we consider it proved that SRAM in the Atmel ATmega1284 microcontrollers can be used to construct a PUF.

Limitations and further research

At a first glance, the generation of 4Kbit long key seems to be good enough but some facts must be taken into account. First, the detected strong bias toward '1' reduces the number of likely combinations and leads to predictable PUF output. Tests with the ten chips showed that the estimated population of all possible identificators is around 2^{50} only. Moreover the 'birthday paradox' decreases this value by half. Therefore the resulting amount of possible unique chips is 2^{25} which is not so big as it looked in the beginning. So the bias can be a limiting factor in some use cases.

Further research could extend this by analyzing the SRAM output under temperature and voltage variations. That can have possible effects on the local mismatch of the internal components and therefore on the PUF properties.

Further, more measurements with more test chips could estimate entropy more accurately. Therefore the key length size could be defined more precisely.

Moreover using preselection techniques could help get rid of error correction and construct different PUFs on Atmel AVR microcontrollers. Suitable preselection methods should be based on additional measurements.

Bibliography

[1] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. Physical Unclonable Functions, FPGAs and public-key crypto for IP protection. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 189–195, 2007.

[2] C. Bohm and M. Hofer. *Physical Unclonable Functions in Theory and Practice*. Springer-Verlag GmbH, 2012. ISBN 9781461450399.

[3] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon Physical Random Functions. In *ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160. ACM Press, New York, NY, USA, 2002.

[4] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer Berlin / Heidelberg, Berlin, Heidelberg, September 2007.

[5] D.E. Holcomb, W.P. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. *Proceedings of the Conference on RFID Security*, 7, 2007.

[6] Maximilian Hofer and Christoph Boehm. An alternative to error correction for SRAM-like PUFs. In *Proceedings of the 12th international conference on Cryptographic hardware and embedded systems*, CHES'10, pages 335–350. Springer-Verlag, Berlin, Heidelberg, 2010.

[7] Gang Qu and Chi-En Yin. Temperature-aware cooperative ring oscillator PUF. In *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, pages 36 –42, july 2009.

[8] A. Maiti and P. Schaumont. Improving the quality of a Physical Unclonable Function using configurable Ring Oscillators. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 703 –707, 31 2009-sept. 2 2009. ISSN 1946-1488.

[9] A. Maiti, L. McDougall, and P. Schaumont. The impact of aging on an FPGA-based Physical Unclonable Function. In *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pages 151 –156, sept. 2011.

[10] Christoph Bosch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on FPGAs. In *Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 181–197. Springer-Verlag, Berlin, Heidelberg, 2008.

[11] Crina Costea, Florent Bernard, Viktor Fischer, and Robert Fouquet. Analysis and enhancement of ring oscillators based Physical Unclonable Functions in FPGAs. In *Proceedings of 2010 International Conference on Reconfigurable Computing and FPGAs*, volume 978-0-7695-4314-7/10, pages 262–268. cancun, Mexique, December 2010.

[12] Haile Yu, Philip Heng Wai Leong, Heiko Hinkelmann, Leandro Möller, Manfred Glesner, and Peter Zipf. Towards a unique FPGA-based identification circuit using process variations. In *FPL*, pages 397–402, 2009.

[13] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert-Jan Schrijen, and Pim Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, HST '08, pages 67–70. IEEE Computer Society, Washington, DC, USA, 2008.

[14] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. FPGA-based true random number generation using circuit metastability with adaptive feedback control. In *Proceedings of the 13th international conference on Cryptographic hardware and embedded systems*, CHES'11, pages 17–32. Springer-Verlag, Berlin, Heidelberg, 2011.

[15] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-invasive EM attack on FPGA RO PUFs and countermeasures.

In *Proceedings of the Workshop on Embedded Systems Security*, WESS '11, pages 2:1–2:9. ACM, New York, NY, USA, 2011.

[16] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from Xilinx Virtex-II FPGAs. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 111–124. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0948-6.

[17] Sergey Morozov, Abhranil Maiti, and Patrick Schaumont. An analysis of delay based PUF implementations on FPGA. In *Proceedings of the 6th international conference on Reconfigurable Computing: architectures, Tools and Applications*, ARC'10, pages 382–387. Springer-Verlag, Berlin, Heidelberg, 2010.

[18] M. Bhargava, C. Cakir, and K. Mai. Attack resistant sense amplifier based PUFs (SA-PUF) with deterministic and controllable reliability of PUF responses. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 106 –111, june 2010.

[19] Leonid Bolotnyy and Gabriel Robins. Physically Unclonable Function-based security and privacy in RFID systems. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications*, PERCOM '07, pages 211–220. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-2787-6.

[20] P. Tuyls, B. Škorić, S. Stallinga, A. H. M. Akkermans, and W. Ophey. Information-theoretic security analysis of Physical Uncloneable Functions. *Financial Cryptography and Data Security*, pages 141–155, 2005.

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

[22] Roel Maes and Ingrid Verbauwhede. Physically Unclonable Functions: A study on the state of the art and future research directions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 3–37. Springer Berlin/Heidelberg, Berlin, Heidelberg, November 2010.

[23] D.W. Bauder. An anti-counterfeiting concept for currency systems. Research report PTK-11990, Sandia National Labs, Albuquerque, NM, USA, 1983.

[24] Commission on Engineering Committee on Next-Generation Currency Design and National Research Council Technical Systems. *Counterfeit Deterrent Features for the Next-Generation Currency Design*. The National Academies Press, 1993.

[25] J.D. Buchanan, R.P. Cowburn, A.V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D.A. Allwood, and M.T. Bryan. Forgery: 'fingerprinting' documents and packaging. In *Nature*, number 436(7050), page 475. Blackett Physics Laboratory, Imperial College London, Jul 2005.

[26] P. Bulens, F.-X. Standaert, and J.-J. Quisquater. How to strongly link data and its medium: the paper case. *Information Security, IET*, 4 (3):125–136, september 2010.

[27] R.S. Indeck and M.W. Muller. Method and apparatus for fingerprinting magnetic media. US Patent No. 5365586, 1994.

[28] Reinhard Posch. Protecting devices by active coating. *Journal of Universal Computer Science*, 4(7):652–668, July 1998.

[29] K.M. Tolk. Reflective particle technology for identification of critical components. Conference/Event SAND–92-1676C; CONF-9207102–26, Institute of Nuclear Materials Management (INMM) annual meeting, Orlando, FL (United States), July 1992.

[30] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical One-Way Functions. *Science*, 297:2026–2030, september 2002.

[31] B Gassend. Physical Random Functions. Master's thesis, MIT, MA, USA, 2003.

[32] J.W. Lee, Daihyun Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, pages 176–179, june 2004.

[33] Daihyun Lim, J.W. Lee, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(10):1200 –1205, oct. 2005. ISSN 1063-8210.

[34] G. Edward Suh and Srinivas Devadas. Physical Unclonable Functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 9–14. ACM, New York, NY, USA, 2007.

[35] P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and R. Wolters. Read-proof hardware from protective coatings. In *Cryptographic Hardware and Embedded Systems Workshop*, Lecture Notes in Computer Science, pages 369–383. Springer, Yokohama, Japan, Oct 2006.

[36] Ying Su, J. Holleman, and B.P. Otis. A digital 1.6 pJ/bit chip identification circuit using process variations. *Solid-State Circuits, IEEE Journal of*, 43(1):69 –77, jan. 2008. ISSN 0018-9200.

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

[38] J. Guajardo, B. Skoric, P.T. Tuyls, S.S. Kumar, T. Bel, A.H.M. Blom, and G.J. Schrijen. Anti-counterfeiting, key distribution, and key storage in an ambient world via Physical Unclonable Functions. In *Information Systems Frontiers*, volume 11, pages 19–41, 2009.

[39] Ryan Helinski, Dhruva Acharyya, and Jim Plusquellic. A Physical Unclonable Function defined using power distribution system equivalent resistance variations. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 676–681. ACM, New York, NY, USA, 2009.

[40] Ulrich Ruhrmair. SIMPL systems: On a public key variant of Physical Unclonable Functions. Cryptology ePrint Archive, Report 2009/255, 2009.

[41] Klaus Kursawe, Ahmad-Reza Sadeghi, Dries Schellekens, Pim Tuyls, and Boris Skorić. Reconfigurable Physical Unclonable Functions – enabling technology for tamper-resistant storage. In *2nd IEEE International Workshop on Hardware-Oriented Security and Trust - HOST 2009*, pages 22–29. IEEE, San Francisco,CA,USA, 2009.

[42] Dinesh Ganta, Vignesh Vivekraja, Kanu Priya, and Leyla Nazhandali. A highly stable leakage-based silicon Physical Unclonable Functions. In

Proceedings of the 2011 24th International Conference on VLSI Design, VLSID '11, pages 135–140. IEEE Computer Society, Washington, DC, USA, 2011.

[43] R. Kumar, V.C. Patil, and S. Kundu. Design of unique and reliable Physically Unclonable Functions based on current starved inverter chain. In *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, pages 224 –229, july 2011. ISSN 2159-3469.

[44] Saro Meguerdichian and Miodrag Potkonjak. Device aging-based Physically Unclonable Functions. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pages 288–289. ACM, New York, NY, USA, 2011.

[45] Kurt Rosenfeld, Efstratios Gavas, and Ramesh Karri. Sensor Physical Unclonable Functions. In *HOST*, pages 112–117, 2010.

[46] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G.-J. Schrijen, M. van Hulst, and P. Tuyls. Evaluation of 90nm 6T-SRAM as Physical Unclonable Function for secure key generation in wireless sensor nodes. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 567 –570, may 2011. ISSN 0271-4302.

[47] Aswin Sreedhar and Sandip Kundu. Physically Unclonable Functions for embeded security based on lithographic variation. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1 –6, march 2011. ISSN 1530-1591.

[48] Koichi SHIMIZU, Daisuke SUZUKI, and Tomomi KASUYA. Glitch PUF: Extracting information from usually unwanted glitches. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E95.A(1):223–233, 2012.

[49] Xiaoxiao Wang and Mohammad Tehranipoor. Novel Physical Unclonable Function with process and environmental variations. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 1065–1070. European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 2010.

[50] K. Lofstrom, W.R. Daasch, and D. Taylor. IC identification circuit using device mismatch. In *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, pages 372 –373, 2000.

[51] Y. Su, J. Holleman, and B. Otis. A 1.6pJ/bit 96variations. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 406–611, feb 2007.

[52] S. Stanzione, D. Puntin, and G. Iannaccone. CMOS silicon Physical Unclonable Functions based on intrinsic process variability. *Solid-State Circuits, IEEE Journal of*, 46(6):1456–1463, June.

[53] Vincent van der Leest, Geert-Jan Schrijen, Helena Handschuh, and Pim Tuyls. Hardware intrinsic security from D flip-flops. In *Proceedings of the fifth ACM workshop on Scalable trusted computing*, STC '10, pages 53–62. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0095-7.

[54] Gyorgy Csaba, Xueming Ju, Zhiqian Ma, Qingqing Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Ruhrmair. Application of mismatched cellular nonlinear networks for physical cryptography. In *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, pages 1–6, Feb.

[55] Boris Škorić, Pim Tuyls, and Wil Ophey. Robust key extraction from Physical Uncloneable Functions. In *Applied Cryptography and Network Security (ACNS) 2005*, volume 3531 of *LNCS*, pages 407–422. Springer, 2005.

[56] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[57] Ghaith Hammouri, Erdinç Öztürk, Berk Birand, and Berk Sunar. Unclonable lightweight authentication scheme. In *ICICS*, pages 33–48, 2008.

[58] Rebecca Angeles. RFID technologies: Supply-chain applications and implementation issues. *Information Systems Management*, 22(1):51–65, 2005. ISSN 1058-0530.

[59] Hagai Bar-El. Known attacks against smartcards, 2004.

[60] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-key cryptography for RFID-tags. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, PERCOMW '07, pages 217–222. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-2788-4.

[61] Eric Simpson and Patrick Schaumont. Offline hardware/software authentication for reconfigurable platforms. In *Proceedings of the 8th international conference on Cryptographic Hardware and Embedded Systems*, CHES'06, pages 311–323. Springer-Verlag, Berlin, Heidelberg, 2006.

[62] U. Ruhrmair, Frank Sehnke, Jan Solter, Gideon Dror, Srinivas Devadas, and Jurgen Schmidhuber. Modeling attacks on physical unclonable functions. In *CCS 2010: Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249. ACM, New York, NY, USA, 2010.

[63] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits: Research articles. *Concurr. Comput. : Pract. Exper.*, 16(11): 1077–1098, 2004. ISSN 1532-0626.

[64] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Implications of machine learning attacks on arbiter PUF-based challenge-response authentication and secure key generation. Technical report, Cosic internal report, 2012.

[65] Dieter Schuster. Side-channel analysis of Physical Unclonable Functions (PUFs). Master's thesis, Technische Universität München, 2010.

[66] Pim Tuyls and Boris Škorić. *Strong Authentication with Physical Unclonable Functions*, 2007. 133–148 pp.

[67] D Lim. Extracting secret keys from integrated circuits. Master's thesis, MIT, MA, USA, 2004.

[68] Mathias Claes, Vincent van der Leest, and An Braeken. Comparison of SRAM and FF PUF in 65nm technology. In *Proceedings of the 16th Nordic conference on Information Security Technology for Applications*, NordSec'11, pages 47–64. Springer-Verlag, Berlin, Heidelberg, 2012.

[69] Dieter K. Schroder. Negative bias temperature instability: What do we understand? *Microelectronics Reliability*, 47(6):841–852, 2007.

[70] Robbert van den Berg. Entropy analysis of Physical Unclonable Functions. Master's thesis, Eindhoven University of Technology, 2012.

[71] E Seevinck, F J List, and J Lohstroh. Static-noise margin analysis of MOS SRAM cells. *October*, 22(5):748–754, 1987.

[72] B. Cheng, S. Roy, and A. Asenov. The impact of random doping effects on CMOS SRAM cell. In Proceeding of the 30th European solid-state circuits conference ESSCIRC, 2004.

[73] Solomon W. Golomb, Guang Gong, Tor Helleseth, and Hong-Yeop Song, editors. *Sequences, Subsequences, and Consequences, International Workshop, SSC 2007, Los Angeles, CA, USA, May 31 - June 2, 2007, Revised Invited Papers*, volume 4893 of *Lecture Notes in Computer Science*. Springer, 2007.

[74] M Hofer and C. Bohm. Error correction coding for Physical Unclonable Functions. In *Austrochip, Workshop on Microelectronics*, 2010.

[75] K Lofstrom. System for providing an integrated circuit with a unique identification. US Patent No. 6161213, 2000.

[76] M. Marunaka. *Method for identifying semiconductor integrated circuit device, method for manufacturing integrated circuit device, semiconductor integrated circuits device and semiconductor chip*, 2001.

[77] Wuidart Luc, Bardouillet Michel, and Malherbe Alexandre. Extraction of a binary code based on physical parameters of an integrated circuit. US Patent No. 6836430, December 2004.

[78] William R. Bidermann. Using a time invariant statistical process variable of a semiconductor chip as the chip identifier. US Patent No. 7291507, 2007.

[79] Heiko Koerner. Method for identifying electronic circuits and identification device. US Patent No. 7893699, 2007.

[80] Elroy M Lucer. Method of forming a unique number. US Patent No. 20110286293, November 2011.

[81] Toshiyuki Okayasu, Shigetoshi Sugawa, and Akinobu Teramoto. Electronic device identifying method. US Patent No. 7812595, 2008.

[82] Luc Wuidart, Michel Bardouillet, and Laurent Plaza. Diversification of a single integrated circuit identifier. US Patent No. 7796759, 2002.

[83] David A. Barr. Security application using silicon fingerprint identification. US Patent No. 7577850, 2009.

[84] Stephen M. Trimberger. Copy protection without non-volatile memory. US Patent No. 7941673, 2005.

[85] S. Devadas and B Gassend. Data protection and cryptographic functions using a device- specific value. US Patent No. 7818569, 2010.

[86] Dekker. Gerrard. Johan. Preventing cloning of receivers of encrypted messages. US Patent No. 2326043, 2009.

[87] Chi-Song Horng. Method of authenticating an object or entity using a random binary ID code subject to bit drift. US Patent No. 6802447, 2002.

[88] Dwaine Clarke, Blaise Gassend, Marten Van Dijk, and Srinivas Devadas. Authentication of integrated circuits. US Patent No. 7840803, 2003.

[89] Pim Theo Tuyls, Theodorus Jacobus Johannes Denteneer, Johan Paul Marie Gerard Linnartz, and Evgeny Alexandrovitch Verbitskiy. Method and system for authentication of a physical object. US Patent No. 8032760, 2004.

[90] Atmel. ATmega1284P. http://goo.gl/xcfCh, 2013. [Online; accessed 23-March-2013].

[91] C. Bohm, M. Hofer, and W. Pribyl. A microcontroller SRAM-PUF. In *Network and System Security (NSS), 2011 5th International Conference on*, pages 269 –273, sept. 2011.

[92] Sergei Skorobogatov. Low temperature data remanence in static RAM. Technical Report UCAM-CL-TR-536, University of Cambridge, Computer Laboratory, June 2002.

[93] Jonathan Jedwab. A survey of the merit factor problem for binary sequences. In *Proceedings of the Third international conference on Sequences and Their Applications*, SETA'04, pages 30–55. Springer-Verlag, Berlin, Heidelberg, 2005.

[94] R W Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[95] R. Micheloni, A. Marelli, and R. Ravasio. *Error Correction Codes for Non-Volatile Memories*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 1402083904, 9781402083907.

[96] S. Lin and D.J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 2004. ISBN 9780130426727. LCCN 2004040060.

[97] Jorge Castineira Moreira and Patrick Guy Farrell. *Essentials of Error-Control Coding*. Wiley, Chichester, 2006.

Appendix A

Measurements summary

Property	Ideal	# 4	# 5	# 6	# 7	# 8
Mean Value min, μ	50	70.56	56.45	67.44	65.93	71.42
Mean Value max, μ	50	70.90	82.88	82.94	82.41	71.78
Mean Value avg, μ	50	70.68	71.49	73.06	72.38	71.61
Error Rate min, HD_{intra}	0	1.98	1.97	1.79	3.61	1.84
Error Rate max, HD_{intra}	0	2.14	2.11	12.96	16.96	2.00
Error Rate avg, HD_{intra}	0	2.06	2.04	8.32	10.47	1.91
Bits correlation, R_{xx}	0	<16	<16	<17	<15	<16
Chips correlation, HD_{inter}	50	40	40	39	39	40

Property		# 9	# A	# C	# D	# E
Mean Value min, μ		67.81	27.62	70.84	67.78	49.77
Mean Value max, μ		82.65	83.98	86.19	81.69	83.16
Mean Value avg, μ		73.15	73.59	77.27	72.11	72.10
Error Rate min, HD_{intra}		1.59	1.71	1.67	1.87	1.69
Error Rate max, HD_{intra}		10.99	18.49	14.36	2.02	15.70
Error Rate avg, HD_{intra}		7.07	10.42	8.82	1.95	9.93
Bits correlation, R_{xx}		<17	<16	<18	<16	<17
Chips correlation, HD_{inter}		40	39	39	40	40

Table A.1: Measurement results of chips

Appendix B

Acronyms

6T-SRAM six-transistor SRAM cell

ADC analog to digital converter

BER bit error rate

CI confident interval

CMOS complementary metal-oxide-semiconductor

CRP challenge-response pair

ECC error correction code

FAR false-acceptance rate

FPGA field-programmable gate array

FRR false-rejection rate

HD Hamming distance

IP intellectual property

IC integrated circuit

NFC near field communication

NTH Norwegian Institute of Technology

NV memory nonvolatile memory

OECD Organisation for Economic Co-operation and Development

PDF probability density function

POWF Physical One-Way Function

PUF Physical Unclonable Function

RFID radio-frequency identification

SNM static-noise margin

SRAM static random-access memory

TRNG true random number generator

Appendix C

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `scripts` ..... the directory with data processing scripts
- `logs` ..... the directory with log files contained memory dumps
- `data` ..... the directory with full calculations: summary tables, post-processing logs, statistical analysis of the tested chips
- `thesis` ..... the directory of LaTeX source codes of the thesis
- `thesis.pdf` ..... the thesis text in PDF format
- `thesis.ps` ..... the thesis text in PS format

Appendix D

Publications of the author

D.1 Josef Hlaváč, Mikhail Platonov, Róbert Lórencz. *PUF on a Simple Microcontroller*. The international workshop on Cryptographic architectures embedded in reconfigurable devices CRYPTARCHI2012, France, St-Etienne, June, 2012.

D.2 Mikhail Platonov. *History of Physical Unclonable Functions*. Proceedings of the CVUT Poster, Prague, May, 2013.

D.3 Mikhail Platonov, Josef Hlaváč, Róbert Lórencz. *Using Power-up SRAM State of Atmel ATmega1284P Microcontrollers as Physical Unclonable Function for Key Generation and Chip Identification*. The first workshop on Trustworthy manufacturing and utilization of secure devices TRUDEVICE 2013, France, Avignon, May, 2013.