Software-Only-PUF-Based Secure Key Storage and Data Protection Scheme Using Off-The-Shelf SRAM

Ade Setyawan Sajim

Delft University of Technology

Software-Only-PUF-Based Secure Key Storage and Data Protection Scheme Using Off-The-Shelf SRAM

Master's Thesis in MSc Computer Engineering

Parallel and Distributed Systems group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Ade Setyawan Sajim

16th March 2018

Author
Ade Setyawan Sajim

Title
Software-Only-PUF-Based Secure Key Storage and Data Protection Scheme Using Off-The-Shelf SRAM

MSc presentation
22nd January 2018

Abstract

TODO ABSTRACT

Preface

TODO MOTIVATION FOR RESEARCH TOPIC

TODO ACKNOWLEDGEMENTS

Ade Setyawan Sajim

Delft, The Netherlands
16th March 2018

Contents

Chapter 1

Introduction

1.1 Need for Self-Sovereign Identity

Hardly anyone can live without having their identity. Identity is the one that defines who we are, something which helps describe the uniqueness of anyone. In modern society, identity is commonly related to social security cards, driver's licenses, and other state-issued credentials. Centralized controlled by the government is the definition among these elements.

Along with the rise of digital age, identity also redefines itself. Identity in the digital world is split into multiple domains. Our Facebook identity does not correlate directly to our Twitter identity or to most other domains. Identities are scattered, vary from one Internet domain to another. Even though this kind of identity is not exactly similar like the previous definition (centrally controlled by a single party / government), it is still almost identical in a way that users are locked in to an authority who can deny their identity or even confirm a false identity. This phenomena ignites a problem where users are not in control of their identity.

This issue gives a chance for a new concept called self-sovereign identity (SSI) to arise. Self-sovereign identity is a decentralized identity concept which capable of authenticating statements, without any central organization, point-of-failure or possibility of data tracking [1]. Self-sovereign identity will be able to give users full control over their identity. In a simple words, users can store their identity data on their devices, and decide whether to give access to anyone who is willing to use it or not. In addition, there will be no need for a centralized storage since each user database is distributed among themselves. A high possibility to get this concept popular is also present with the introduction of the European Union General Data Protection Regulation [2].

Johan Pouwelse and Martijn de Vos in [1] proposed a SSI design where the user data are encrypted and never leave the device/domain. Any operation which require the data, such as authentication, will require homomorphic encryption on the encrypted data. This encrypted data should be securely protected and the domain should be trustworthy. Simply put, security is the main thing on this implementation.

Since security is important, on designing a system that deal with sensitive informations, one need to ensure that the system is able to protect the information and property from unauthorized entity without sacrificing the system's functionality. The study dedicated to protect information from unauthorized access, loss or damage is called information security [3].

In information security, one way to protect an information is by using cryptographic techniques. A cryptographic technique commonly consist of a cryptographic algorithm and a key. On designing a cryptographic algorithm, one should follow Shannon's Maxim which says one should design systems under the assumption that the enemy will immediately gain complete knowledge with the design [4]. In another word, on a secure cryptographic technique, the only thing which need to be kept secret is the key. The algorithm itself is supposed to be publicly available without affecting the system security. This principle should also be implemented in designing self-sovereign identity. The encrypted data in the device should be protected by a key, verified by a secure authentication scheme.

The most common way to stored the key is by using a *non volatile memory* (NVM). NVM is a type of computer memory that keep intact its information even after turned off. An example of product which implement this approach is the debit card. It uses its chip to store information. Unfortunately, this NVM is prone to physical attack. Since the key is permanently stored in the memory, an attacker can use some technique to clone the memory, such as microprobing [5]. Attacker may also use a side channel information to retrieve any information about the key. There are numerous other techniques on this kind of attack. This attack can be even worse if someone that knows the system design is involved. Due to this problem, more secure, tamper-evident, tamper-proof solutions need to be presented.

1.2 Rise of PUF as a Security Solution

In 2001, Physical Unclonable Function (PUF) comes in handy as an inexpensive and yet effective security solution to overcome the mentioned problem above by a different way of generating and processing secret keys in security hardware. It was introduced by Pappu [6]. Unlike cryptographic algorithm security which usually rely on hard-to-solve mathematical problem, PUF ideas stems from using hardware features designed to utilize the physical random nanoscale disarray phenomena [7]. This disarray phenomena can be used as a derivation of keys without having to keep any security-critical information explicitly. This physical randomness is unclonable, even by the original manufacturer due to manufacturing process variations. Furthermore, since the secrets can only be produced when the PUF device is turned on, active manipulation of circuit structure will cause dysfunction of challenge-response mechanism and destroy the secret.

Related to self-sovereign identity concept, [1] present an idea to use PUF and biometric-based authentication to securely protect the data in the self-sovereign identity. Figure 1.1 shows the detailed technology stack in their trust creation proposal on how to build trust in the blockchain era.

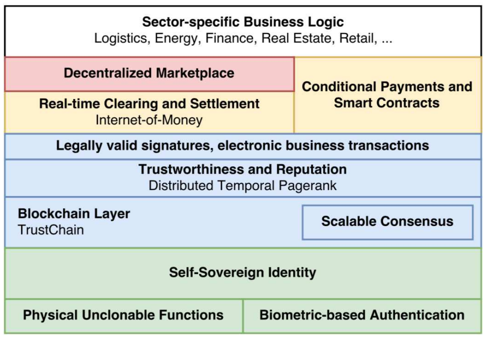
Sector-specific Business Logic
Logistics, Energy, Finance, Real Estate, Retail, ...
Decentralized Marketplace
Conditional Payments and Smart Contracts
Real-time Clearing and Settlement
Internet-of-Money
Legally valid signatures, electronic business transactions
Trustworthiness and Reputation
Distributed Temporal Pagerank
Blockchain Layer
TrustChain
Scalable Consensus
Self-Sovereign Identity
Physical Unclonable Functions
Biometric-based Authentication

Figure 1.1: Detailed technology portfolio for trust creation in the blockchain age [1]. As shown in the bottom of this figure, Physical Unclonable Functions and biometric-based authentication are utilized to secure the self-sovereign identity.

An example of PUF type is SRAM PUF. SRAM, stands for static random-access memory, is a type of semiconductor memory that uses bistable latching circuitry (flip-flop) to store each bit. When a static RAM (SRAM) is turned on, the memory cells have an undefined state [8]. Since the initialized bits of SRAM is random, these bits are a good candidate for PUF. The value of these bits itself is determined by the SRAM cell which consists of two cross-coupled inverters along with two access transistors. This concept was firstly introduced by Guajardo [9]. In order for SRAM to be used as a cryptographic security key, SRAM PUFs need to have certain characteristics such as the key generated by every SRAM should be reliable and unique. Reliable means the generated key should always be consistent, while unique refers to there should be no correlation between one device or another.

Unfortunately, SRAM PUF is also problematic since it contains noise in its bit value. Many proposed solution has emerged, but it seems that its availability is still limited, expensive or only exclusive for some companies. SRAM PUF also consider as a weak PUF, which means that it has limited challenge-response pairs. Due to limited challenge-response pairs, it is argued as unsuitable for authentication scheme, only appropriate for key generation or random number generator [7].

1.3 Problem Statement

As mentioned above, SRAM PUF still has some problems that need to solve. A future where anyone can have their own SRAM PUF without having to buy from specific company is a wonderful possibility. If everyone can just buy an SRAM from the market and use it on top of an open system, this will revolutionize the security industry. This objective describes the motivation for this thesis.

The goal of this thesis is to build an open system where anyone can use their own SRAM as a PUF solution and provide a secure key storage and data protection function. The system will include a SRAM's stable bits analyzer, test system to check the SRAM's quality as a PUF, and a scheme of key storage and authentication as the main purpose of PUF itself.

As an attempt to reach this goal, several steps are expected to be done:

1. Get multiple type of SRAM available in the market
2. Investigate the characteristic of each SRAM
3. Investigate and choose the embedded platform where the system will be build
4. Design a system which able to determine the stable bits of SRAM automatically
5. Search and analyze existing secure key storage and data protection methods using SRAM PUF
6. Propose a system to enable secure PUF key storage and data protection using off-the-shelf SRAM
7. Construct the complete software, which contained the error correcting code
8. Evaluate the solution by experimenting on SRAM and conducting performance analysis
9. Explore possible improvements on the system for a future research

1.4 Contributions

This thesis provides the following contributions:

1. Analysis of time difference, voltage difference and neighbor values on the stability of SRAMs 23LC1024 and CY62256NLL
2. Result comparisons between two methods on looking for stable bits: data remanence and neighbor stability analysis on SRAMs 23LC1024 and CY62256NLL
3. A system to enable strong PUF authentication and key storage scheme using SRAM

1.5 Outlines

Chapter 2

Introduction to Security

2.1 Secure System Necessity

How valuable is our data? How much would company for accessing those information? These questions might be silly but if we consider that there are many companies which thrived using our data (e.g. Facebook, Twitter, and Uber), we should reconsider how much should we value our data. I know these data if we value each element might not worth much, most will worth significant when these data is combined together to bring a more insight information. But what about the sensitive data which its value can worth millions of dollars, such as the bitcoin key? In [10], the highest bitcoin address is worth 1.4B US$ and there are 513,562 addresses which has value more than 10000 US$. Due to their high values, these data, the key to these addresses, must be protected.

Before going further, we should understand first what kind of attacks possibly affecting these data. Similar like in the physical world, in the digital world, adversaries' intention can be either mischievious, non-malicious or accidental. Some example of mischievious activities are stealing information and modifying the data. Accidental can happen due to human error. These three types of adversaries' intention can lead to a significant number of attacks and threats.

Now, two questions are arised. Is there are a way that guarantee 100% of these data protection? Is there any bullet-proof secure system? Unfortunately, there is no such thing as a 100% secure system. Fortunately, there are ways to design a system to be as secure as possible in a limited scope, usually defined as secure 'from who' and 'from what'. According to [11], computer security is "the protection of the items you value, called the assets of a computer or a computer system." In the scope of data mentioned before, the assets are the bitcoin keys and addresses.

To help defining a secure system, common security requirements are mentioned. According to [12], there are four elements on common security, which are:

- *Confidentiality*: a piece of information should be accesible only to an authorized users. For example, an encrypted data can only be decrypted by the secret key owner.

- *Authentication*: assurance of the sender of a message, date of origin, data content, time sent, data information, etc. are correctly identified.
- *Integrity*: any assets can only be modified by an authorized subjects. For example, data should be keep intact during transmission
- *Non-repudiation*: a subject should be prevented from denying previous actions. For example, a sender cannot deny the data which it sent.

2.2 Cryptography

One way to achieve these four security requirements is by using cryptography. In traditional definition, cryptography can be defined as the art of writing or solving codes [13]. But this definition is inaccurate to use nowadays because instead of depending on creativity and personal skill when constructing or breaking codes, the modern cryptography focus their definition using science and mathematics. According to [14], modern cryptography can be defined as ”the scientific study of techniques for securing digital information, transactions, and distributed computations.” The algorithm which use cryptography as their main point is called cryptographic algorithm.

Since the birth of cryptography, its main concerned is usually related on securing communication which can be achieved by constructing *ciphers* to provide secret communication between parties involved. The construction of ciphers to ensure only authorized parties also can be called as encryption schemes. There are two types of cryptographic algorithm, symmetric and asymmetric algorithm. Symmetric, also known as private key encryption or private key cryptography, requires the same key for encryption and decryption. Meanwhile in asymmetric algorithm (can be referred as public key encryption or public key cryptography), there are two keys utilized; private key and public key. Public key is utilized for encryption and private key is used for decryption. One of the main advantage of symmetric encryption over asymmetric encryption is it requires less computational power which make it suitable to use in embedded devices. Further explanation on symmetric encryption algorithm will be provided in the next chapter.

Chapter 3

Related Work

3.1 PUF

A physical unclonable function is an entity that utilize manufacturing variability to produce a device-specific output. The idea to build PUF arise from the fact that even though the mask and manufacturing process is the same among different ICs, each IC is actually slightly different due to normal manufacturing variability [7]. PUFs leverage this variability to derive secret information that is unique to the chip. This secret can be refer as a silicon biometric. In addition, due to the manufacturing variability that defines the secret, one cannot manufacture two identical chips, even with full knowledge of the chips design. PUF architectures exploit manufacturing variability in multiple ways. For example, one can utilize the effect of gate delay, the power-on state of SRAM, threshold voltages, and many other physical characteristics to derive the secret.

Due to this feature, PUFs are a promising innovative primitive that are used for authentication and secret key storage without the requirement of secure hardware. Currently, the best practice for providing a secure memory or authentication source in such a mobile system is to place a secret key in a nonvolatile electrically erasable programmable read-only memory (EEPROM) or battery- backed static random-access memory (SRAM) and use hardware cryptographic operations such as digital signatures or encryption.

There are two main parts of PUF, physical part and operational part. Physical part refers to a physical system that is very difficult to clone due to uncontrollable process variations during manufacturing. Operational part means a set of **challenges** (PUF input) C_i has to be available to which the system responds with a set of sufficiently different **responses** (PUF output) R_i. This combination of challenge and response is called **challenge-response-pair** (CRP).

$$R_i < -PUF(C_i) \tag{3.1}$$

The common application on using PUF usually requires two phases; the first phase is called **enrollment** and the second one is usually referred as **verification**.

In enrollment, a number of CRPs are gathered from a PUF and then stored. In verification phase, a challenge from the stored CRPs is given to the PUF. Afterwards, the PUF response from this challenge is compared with the corresponding response from the database. The response is considered to be valid if there's a CRP from the stored CRPs related to these challenge and response.

According to [7], to be qualified as PUF, a device should fulfilled several characteristics below :

- **Reliable**: A response to the same challenge should be able to be reproduced over time and over various range of conditions.
- **Unpredictable**: A response to a challenge on a PUF device should be unrelated to a response to another challenge from the same device or the same challenge from different device.
- **Unclonable**: Challenge-response pairs mapping of a device should be unique and cannot be duplicate.
- **Physically Unbreakable**: Any physical attempts to maliciously modify the device will result in malfunction or permanent damage.

3.1.1 PUFs Classification

Based on the number CRPs, PUFs can be divided into two categories:

- Strong PUFs
  Strong PUFs can be identified by having large number of CRPs. Strong PUFs typically used for authentication.
- Weak PUFs
  Contrary to strong PUFs, weak PUFs only have a small number of CRPs. Weak PUFs commonly used for key storage.

Beside number of CRPs, PUFs can also categorized based on their physical design. There are two major category, extrinsic and intrinsic.

Extrinsic means that it need extra hardware added to the PUF component. There are two subcategories in extrinsic PUFs, non electronic and analog electronic PUFs. Some example in non electronic PUFs are optical PUF, paper PUF, CD PUF, RF-DNA PUF, magnetic PUF, and acoustic PUF. Some design instances in analog electronic PUFs are VT PUF, power distribution PUF, coating PUF, and LC PUF.

In intrinsic, the PUF component has to be available naturally during the manufacturing process. In addition, PUF and the measurement equipment should be fully integrated in intrinsic PUF. There are two subcategories in intrinsic PUFs, delay based and memory based PUFs. An example of delay based PUF is arbiter PUF. The main principle of arbiter PUF is to introduce a digital race condition on two paths on a chip and have an arbiter circuit to decide which one won the race. As in memory based PUFs, some examples of this design are SRAM PUF, butterfly PUF and latch PUF. SRAM PUF utilized the random physical mismatch in the cell caused by manufacturing variability determines the power up behavior (can be zero, one, or no preference). Butterfly PUF use the effect of cross coupling between two transparent data latches. Using the clear functionalities of the latches, an unstable state can be introduced after which the circuit converges back to one of the two stable states. In latch PUF, the concept is based on using two NOR gates which are cross coupled. These gates will converge to a stable state depending on the internal mismatch between the electronic components.

3.1.2 Hamming Distances as an Identification Helper

As explained before, PUF main purpose is dedicated for identification, shown by having a device specific output. In PUF, hamming distance is commonly use as a way to help defining this idea. Hamming distance itself is the number of positions at which the corresponding symbols are different on two equal length strings. There are two types of hamming distance utilized, intra-chip and inter-chip hamming distance. Inter-chip hamming distance is the distance between two responses resulting from applying a challenge once to two different PUFs device. Intra-chip hamming distance refers to difference between the two responses resulting from applying a challenge twice to a PUF device [15]. To ease the identification purpose, fractional hamming distance is also introduced. Fractional hamming distance is the number of differences between two strings divided by the length of the bit strings. In ideal PUFs, the intra-chip fractional hamming distance ($\mathrm{HD}_{\mathrm{intra}}$) is 0% and inter-chip fractional hamming distance ($\mathrm{HD}_{\mathrm{inter}}$) is 50%. Due to noises, normally PUF devices has $\mathrm{HD}_{\mathrm{intra}} \leq 10\%$ and $\mathrm{HD}_{\mathrm{inter}}$ 50%. The identification goal will not be achieved if there is an overlap between $\mathrm{HD}_{\mathrm{intra}}$ and $\mathrm{HD}_{\mathrm{inter}}$ [16]. Overlap will happen if the $\mathrm{HD}_{\mathrm{intra}}$ is too large and $\mathrm{HD}_{\mathrm{inter}}$ is too small, e.g. $\mathrm{HD}_{\mathrm{intra}}$ is 35% and $\mathrm{HD}_{\mathrm{inter}}$ is 30%.

3.1.3 Helper Data Algorithms and Fuzzy Extractor

There are two issues if PUF raw responses is used as a key in cryptographic primitive. First, both weak and strong PUFs rely on analog physical properties of the fabricated circuit to derive secret information. Naturally, these analog properties have noise and variability associated with them. This can be a problem due sensitivity of cryptographic functions on noises of their inputs. Another issue is the PUF raw responses usually are not uniformly distributed, which makes it an unqualified as a cryptographically secure key. These two issues can be solved using **Helper Data Algorithm** (HDA). One can also referred Helper Data Algorithm as **fuzzy extractor** since both are capable of converting noisy information into keys usable for any cryptographic application [17] [18].

Fuzzy extractor solves both issues mentioned above by using two phases, **information reconciliation** and **privacy amplification**. In information reconciliation phase, possible bit errors are corrected to form a robust bit string [19]. Information reconciliation is tightly related to error correction. In fact, a procedure to do information reconciliation based on error-correcting codes is called code-offset technique [18]. Using code-offset technique, one should be able to reconstruct a bit string *w* from a noisy version *w'* as long as the Hamming distance between *w*and *w'* is limited to *t*. The second phase, privacy amplification, is a process to evolve this robust bit string into a full entropy key. Privacy amplification, also can be called as randomness extraction [20], can be done by utilizing two-way hash function.

Beside these two phases, fuzzy extractor also consists of two procedures, Gen and Rep. Gen, stands for *generation*, is a probabilistic procedure which outputs an "extracted" string / key (secret) R and a string (public) *helper data* P on input fuzzy data w. Rep, stands for *reproduction*, is a deterministic function capable of recovering secret key R from the string *helper data* P and any vector w' as long as the Hamming distance between *w*and *w'* is limited to *t*. In [21], Taniguchi et. al illustrated the generation and reproduction procedure of fuzzy extractor on PUF which shown on Figure 3.1.

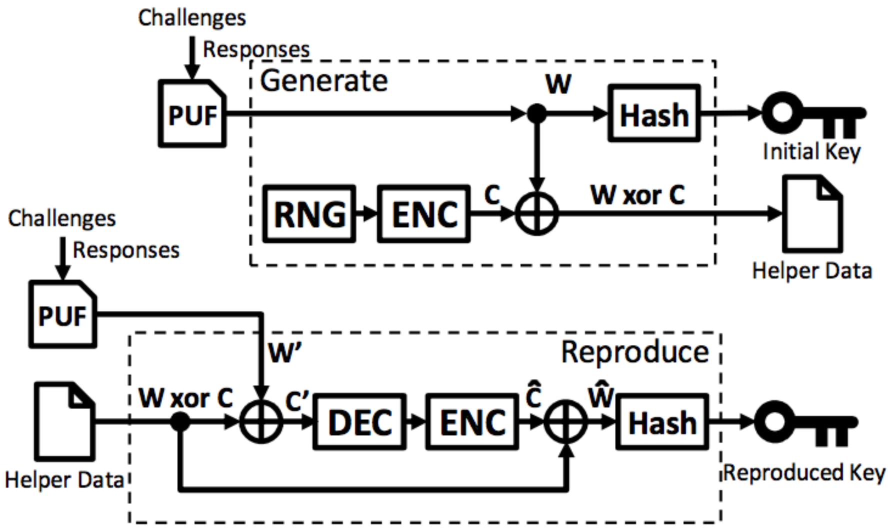

Figure 3.1: Two procedures inside fuzzy extractor; generation and reproduction [21]

3.1.4 Error Correcting Codes

To handle noises occurred inside a PUF, error-correcting codes (ECC) is employed. Error-correcting codes are a class of schemes for encoding messages in an attempt to enable message recovery when there is noise introduced in the sending or receiving of the message. ECC can be divided into two subcategory, hard-decision and soft-decision. Hard-decision operates on fixed set of possible values (usually 0 or 1 in a binary code), while the inputs to a soft-decision decoder may take on a whole range of values in-between (usually refers to float value).

There are some well-known ECC, such as in hard-decision code, Reed-Solomon code and BCH code; while in soft-decision, Viterbi code and turbo code. Soft-decision code has an advantage over hard-decision code where it can process extra information which indicates the reliability of each input data point and used to form better estimates of the original data. But it has drawback where one should provide a probability function on the data (on SRAM, a probability function on each cell should be provided) to enable a good decoding result. This is a problem if applied on this thesis goal where the system should work on any SRAM off-the-market. Calculating the probability on each SRAM cell will take an extra step, over complicate the system and the procedure on using the constructed system. Thus, the hard-decision code is preferred.

One of the popular hard-decision error correcting code is BCH codes. BCH, stands for *BoseChaudhuriHocquenghem*, codes are a family of cyclic error correcting codes which constructed using polynomials over a finite field and work in binary field. BCH codes are a very flexible set of codes in that within certain bounds there is a great amount of choice in code parameters and are relatively efficient in message length and error correction. The code parameters are as follows:

- q: The number of symbols used (e.g., in binary field, $q = 2$)
- m: The power to which to raise q to generate a Galois Field for the construction of the code.
- d: The minimum Hamming distance between distinct codewords.

These parameters lead to several derived parameters which are standard parameters of linear codes:

- n: The block length of the code; for our special case, $n = q * m1$
- t: The number of errors that can be corrected, $d \geq 2t + 1$
- k: The number of message bits in a codeword, $k \geq n - mt$

Both BCH codes and Reed-Solomon codes have the capability to correct multiple errors. Reed-Solomon codes is also a flexible ECC and have similar parameters as BCH codes, e.g. n, k, d. Unlike BCH codes, Reed-Solomon codes can work in both binary and non-binary fields. Reed-Solomon codes also perform better in correcting burst errors while BCH codes are better in fixing random errors. BCH codes has an advantage where it requires less computing resource when working on same parameter compared to Reed-Solomon codes.

3.2 SRAM PUF

The SRAM PUF was first proposed by Guajardo and Holcomb in 2007. SRAM PUF use existing SRAM blocks to generate chip-specific data. Normally, when using SRAM to store data, a positive feedback is given to force the cell into one of the two states (a ’1’ or a ’0’) available. Once it is there, the cell will be stable and prevented from transitioning out of this state accidentally. To use it as a PUF, SRAM is turned on and its cell values are retrieved to generate a unique chip-specific output. After powering-up the circuit, the cells stabilize at a state which is defined by the mismatches between the involved transistors. Thus, each SRAM cell provides one bit of output data.

As mentioned in the beginning of this chapter, during enrollment, challenge-response pairs are gathered. In SRAM PUF, there are two type of challenges that can be applied to the system. The challenge can be either the whole SRAM memory or specific addresses. If a set of addresses is given as challenge, an address in there can refer to an address of a byte, a bit, or a sequence of bytes or bits.

3.2.1 SRAM Cell

SRAM uses its SRAM cells to store the binary information. The most common SRAM design is six-transistor (6-T) CMOS SRAM, shown in Figure 3.2. This design utilizes the concept of cross-coupled inverters, constructed by two inverters, each established by two transistors; inverter 1 by Q2 and Q6, inverter 2 by Q1 and Q5. Using this design means the input of an inverter is the output of the other and vice-versa, which also indicates that the output of one inverter is exactly the opposite of the other inverter [15]. Transistors Q3 and Q4 refers as the access transistors, are used as the entry gate to the cell every time a read or write operation will be performed. The bitline (BL), the compliment bitline (BLB) and the wordline (WL) are utilized to access the cell. In addition, an SRAM cell will lost its state shortly after power down [22].

During manufacturing, there are small differences between each SRAM cell due to process variation which lead to a mismatch in the cell [23]. This mismatch also means that the two inverters will always conduct distinctly. Since this mismatch determines the value of the power-up state of an SRAM cell, the power-up state of a cell will be biased towards 0 or 1 depends on the mismatch value. The mismatch itself does not disturb the normal storage functionality of SRAM cell. Based on this bias, SRAM cells can be classified into three categories as shown below:

1. Non-skewed cell

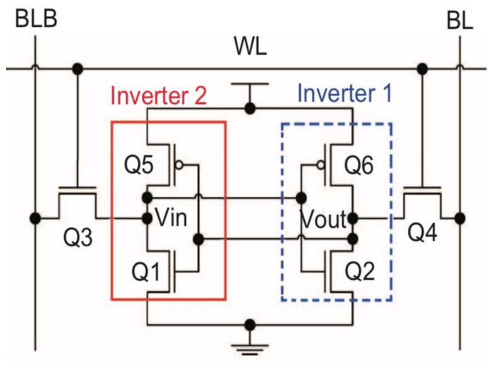

Figure 3.2: A 6-T CMOS SRAM cell [15]

A non-skewed cell has no preference during its start up due to the impact of process variations does not cause any mismatch between the two inverters. This cell produces bit randomly either a ’0’ or ’1’ at its output, depending upon the noise present in the system.

2. Partially-skewed cell
   A partially-skewed cell has a small mismatch between the inverters which lead to a preference over value ’0’ or ’1’ but the cell can flip its value upon variation in external parameters.

3. Fully-skewed cell
   A fully-skewed cell is a heavily mismatched SRAM cell in a way that the cell inclined towards value ’1’ or ’0’ and has a resistance against external influence / noises.

3.2.2 Problem: Noise

Similar like most electronic components, SRAM PUF is also affected by external influence which lead to noises. These noises will flip unstable bits inside the SRAM PUF. Below are some factors presenting noises:

- Voltage
  The noise introduced by voltage is called power supply noise [24]. This noise is related to changes in the delay characteristics of the gate. The changes will occur when there are switchings in the circuit after the device is turned on which increase dynamic power and cause voltage drop on power lines and voltage increase on ground lines.

- Temperature
  Temperature variation can be introduced by the surroundings or voltage variation. The preference of a cell inside SRAM has a high probability

to be affected by temperature. Temperature affects more than voltage on bit flipping.

- Crosstalk
  Crosstalk occurs when a signal transmitted on one circuit creates an undesired effect in another circuit. Crosstalk happens due to tight gap between the SRAM cell (tiny interconnect spacing and width). This event becomes more popular due to wider use of smaller geometries and faster operating speeds. crosstalk is a major contributor to signal integrity problems in modern designs [24].

- Aging
  Aging is related to changes in the silicon after usage for a long time [25]. There are three main effects related to aging of a circuit; time dependent dielectric breakdown (TDDB), bias temperature instability (BTI) and hot carrier injection (HCI). TDDB is associated to the creation of a conduction path through the gate transistor structure which causes an increase in power consumption and the circuit delay [26]. BTI causes a degradation of the transistor threshold voltage [27]. HCI generates a change in the transistor threshold voltage [28]. HCI is caused by a high current in the transistor channel injecting charges into the gate oxide during the switching.

3.2.3 Bit Selection Algorithm

Since bit responses are used as the primary input for SRAM PUF, one of the major steps on using SRAM PUF if location of bits is used as the challenge is looking for stable bits. Stable bits itself refers to fully skewed cells explained before. Even though the error correction code is present to correct the noise of bit responses, it also has limitation on how many bits it can correct. Since not every SRAM cell is stable, one should take a special caution on deciding which SRAM cell is gonna be the bits to use as PUF input.

Choosing the most stable bits is important to ensure that the PUF result is always the same throughout its lifetime. In here, we use two known algorithm to search for stable bits.

Neighbor Analysis

The first algorithm is use the rank of total stable neighbors [29]. The cells that are most stable across environmental conditions are surrounded by more stable cells during enrollment. A stable cell surrounded by more stable cells has a tendency to become more stable because its neighboring cells are likely to experience similar aging stress and operating conditions. The more stable neighbor cells it has, the higher weight it gets. After determining the weight of each cell, a heuristic algorithm that greedily chooses cells for the PUF ID/key with weight greater than a threshold is used.

Before the algorithm is performed, one should collect lots data of SRAM cells value first. The data should be retrieved in various condition, for example different voltages, temperatures, and time differences between enrollment. Afterwards, we use Temporal majority voting (TMV) to calculate all stable bits in SRAM. Last, the neighbor analysis algorithm is performed to get the most stable bits in SRAM.

Data Remanence

Another bits selection algorithm is by using data remanence of SRAM cell [30]. This approach requires only two remanence tests: writing 1 (or 0) to the entire array and momentarily shutting down the power until a few cells flip. The cells that are easily flipped are the most robust cells when written with the opposite data. Strong 1's are bits that are flipped fast after 0 is written to its location. On the contrary, if 1 is written to a bit location and the bit flipped fast, it means that the bit is a strong 0.

3.3 Key Generation using SRAM PUF

In this section, there are two scheme for key generation produced by Hyunho Kang et. al. Both constructions were built on 2014. The first construction, shown in Figure 3.3, is utilizing random number generator (RNG). This design was perfected in the second design shown in Figure 3.4. In the second design, random number generator was removed to make the construction more simple without affecting the security. The block length (n) of the error correcting code in these schemes is 255.

3.4 HMAC

HMAC, stands for hashed message authentication code.

3.5 Symmetric Encryption

As mentioned in the previous chapter, symmetric encryption, requires the same key for encryption and decryption. According to [14], symmetric encryption consists of three algorithms which are:

- **Gen**: key-generation algorithm
- **Enc**: encryption algorithm

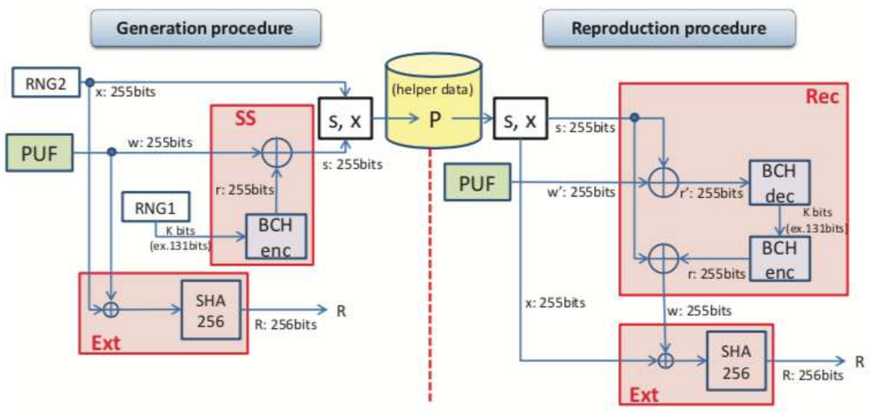

Figure 3.3: Implementation diagram using fuzzy extractor (N = 255) [31]

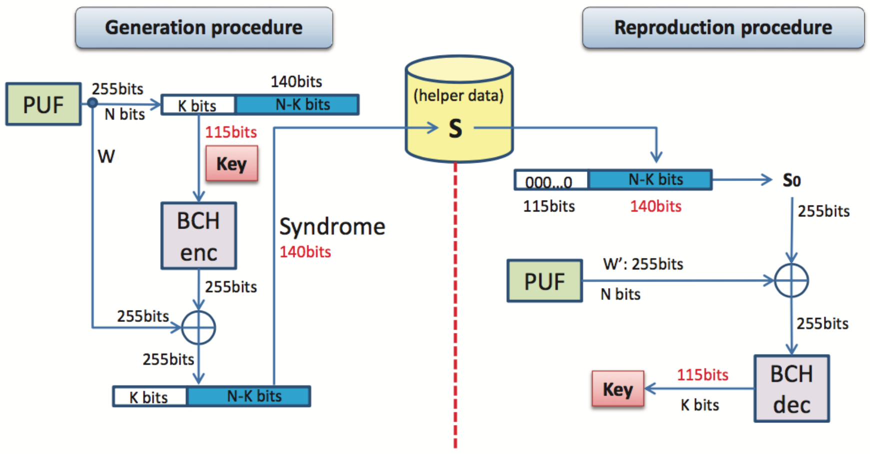

Figure 3.4: Implementation diagram for efficient fuzzy extractor based on the syndrome (N = 255) [32]

- **Dec**: decryption algorithm

To illustrate this better, an example using two parties, Alice and Bob are given. Before using the encryption or decryption algorithm, both parties will agree on a shared secret key k. This phase can be referred as Gen. Afterwards, Alice can use the encryption algorithm (Enc) E_k using the shared secret key k on a message m which will generates a ciphertext c. This procedure can be noted as $c = E_k(m)$. Bob can read the message by using the decryption algorithm ((Dec)) Dec_k using the same shared secret key k. Decryption will result in the plaintext message m. This can be noted as $m = D_k(c)$.

There are many examples of symmetric encryption algorithms, such as RC2, DES, 3DES, RC6, Blowfish, and AES. AES algorithm will be explained below.

AES

AES, stands for The Advanced Encryption Standard, is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.It also known by its original name Rijndael.

Chapter 4

SRAM PUF Open Problems

Since firstly introduced by Guajardo and Holcomb in 2007, there have been many innovations in SRAM PUF field. A simple patent search using patents.google.com with query 'sram; puf' results in 546 results [33]. Number of articles in scholar.google.com also exhibit a high occurences, shown 2,120 articles (citations and patents are not included) [34]. Even though these facts indicates a promising future for this concept, one also should notice that current state-of-the-art in this field mostly consists of one-off prototypes and specific proprietary implementations. To get an SRAM PUF product from the market, one has to order a specific request from a company. For example, Intrinsic ID, one of the main leader in SRAM PUF technology, has a software only solution which will be able to generate unique keys and identities for nearly all microcontrollers without need for security-dedicated silicon [35]. Even though this solution exists and seems easy to use, unfortunately they don't say specifically how much will it cost to use this solution. They also has another solution for SRAM PUF which is focused on hardware IP (and supporting software/firmware) to enable designers to implement PUFs on their design. This solution has a high possibility to obstruct a small company or a single user to use their solution since usually this type of product are intended to use with expensive contract. Similar like the software-only solution they offer, they also don't put the explicit price to use this product. An example of product that use this solution is FPGA Microsemi Polarfire [36].

The SRAM PUF field lacks a Arduino, Linux, or GCC type of open reference implementation. A quick lookup in Github, there's no extensive open source project related to SRAM PUF there. There are projects corresponding to PUF concepts, but most of them also only delve into simulation. The communities seems to haven't establish a wide agreement on a which approach yields the strongest security properties.

Based on these facts, the author believes the next challenge for this field is to discover a common approach. Furthermore the field needs to move beyond isolated single-person projects and single-company approaches towards a mature and sharing ecosystem. The field SRAM PUF requires a single implementation which is continuously improved upon for many years to come and is supported by the majority of the academic and commercial parties.

Chapter 5

Design of Data Protection Scheme

5.1 Use Case, Assumptions and Requirements

As mentioned in the first chapter, a subset of this thesis goal is to provide a secure key storage and data protection function using SRAM PUF. To focus the thesis approach, the field of both functions are decided to be only available offline. Accessing the SRAM PUF requires the user to have the device next to his/her side. In order to achieve this goal, first, a set of requirements need to be define first. Below are the requirements defined:

1. Software-only construction
   There should be no major hardware modification or hardware design to implement the project

2. Patent/license free
   Any dependent component inside the design should be in public domain.

3. Open-source and collaboration oriented
   If there's a reliable open source project which can be a foundation for this thesis project, insted of building our own software, it is preferred to use that project. This will significantly reduced the time consumed on constructing the whole project. Using other project source code can also increased the collaboration atmosphere. In addition, this requirement may help this project to be known by others since they might introduced our project as one of the project that use their code.

4. Key-length security level
   The goal on key-length security level is 256-bits. The concept constructed should be able to use this level and the project's security should be uncompromised even though the key-length is only 256-bits.

5. Off-the-shelf SRAM
   The SRAM involved in the thesis should be easily available in the market and cost insignificant.

6. Affordable
   The total hardware required to produce the system should be inexpensive.

7. Reproducible
   Anyone should be able to reproduce this thesis experiment with no significant effort.

5.2 Data Protection and Key Storage Scheme

Figure 5.1 shows the scheme to protect user's data. To prevent unauthorized person accessing the data with a stolen PUF, an idea from multi factor authentication is utilized. Instead of just depending on the PUF device to access the key, a combination of PUF device and user knowledge is presented. User knowledge used here is username and password. Username is utilized to protect helper data and the challenge. The challenge used here is the location of stable bits. The challenge needs to be kept secure because if the challenge is not protected, the attacker would easily know the stable bits used on key generation. Actually if an attacker can never accessed the SRAM directly, a publicly available stable bit locations will not affect the security, but since the construction is modeled using a plug and play SRAM where anyone can easily access it, the challenge need to be encrypted. Also, if attacker also know the helper data, the generated key would be easily identified. Both the challenge and the helper data is encrypted using symmetric encryption. User's password is combined with the PUF generated key to generate a stronger key using HMAC. The HMAC function proposed to use is HMAC-SHA256. The final key can be used to encrypt and decrypt user data. The symmetric encryption algorithm used is AES. If the data is switched to user's key to be stored, the data protection scheme proposed here can be also referred as key storage scheme.

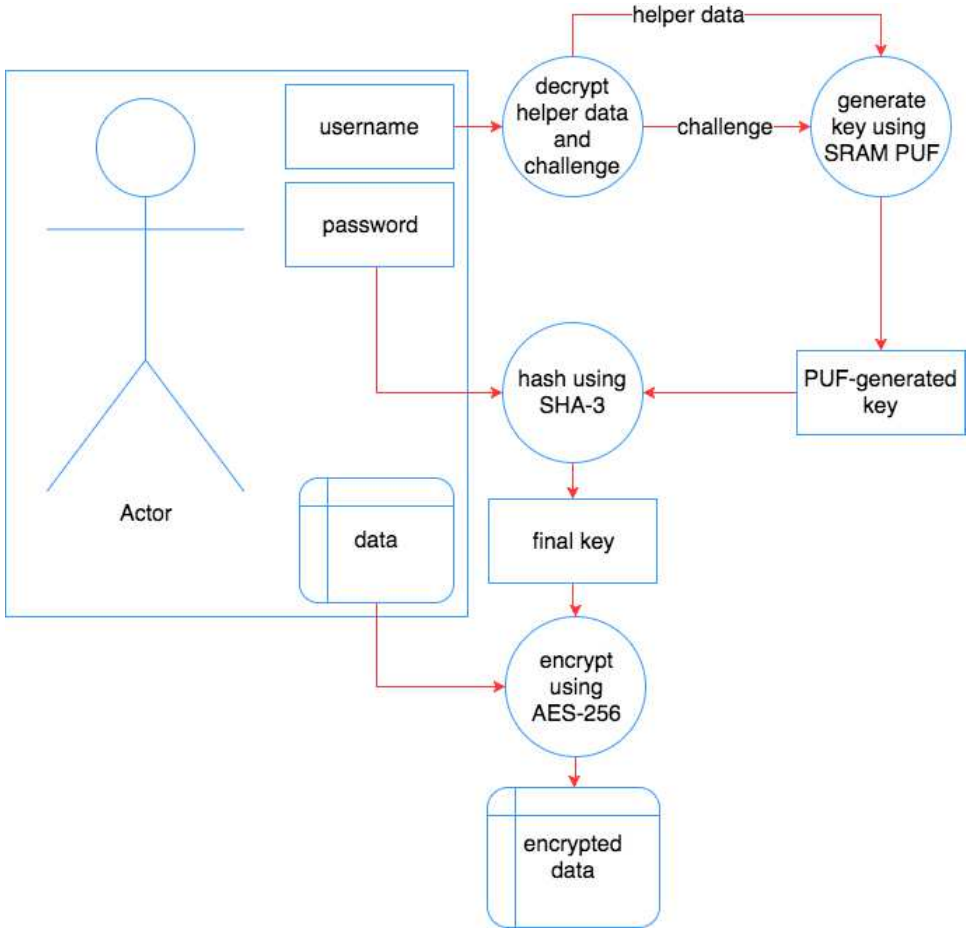

Figure 5.1: Scheme for Data Protection

Chapter 6

Implementation and Results

This chapter contains explanation on experiment setup and results.

6.1 BCH Codes as Error Correcting Codes

As mentioned in the previous chapter, BCH codes are a flexible ECC shown by multiple parameter available. The only fixed parameter is q since the problem is in binary form ($q = 2$). The source code for BCH as ECC is a modified version of Robert Morelos-Zaragoza's version which can be retrieved at [37]. This code is selected because it can support m ranging from 2-20 which mean the length of the code than can be corrected ranging from 2 until 1048575. One should be careful on deciding the parameter that will be used, for example, larger m or n means a bigger memory needed. These parameter should be determined with several considerations, such as, the inner hamming distance of SRAMs and memory available on Arduino Mega 2560.

On deciding the value m, a further look on the memory required during the error correction computation need to be done. Inside the bch code from [37], the decoding method requires the largest memory compared to other procedures. There are six parameters that depends on m which are elp, d, l, u_lu, s, and err. Table 6.1 shows the required memory given the m value.

Since SRAM in Arduino only has 8k bytes capacity, the chosen m is 6 (requires 4553 bytes, around 55% of total SRAM available in Arduino). This parameter will result in possible n between 32 and 63. n is chosen to be 63 to maximize the length code that can be encoded. The combination of m = 6 and n = 63 results in various k and t that can be chosen. The combination of all parameter possible is shown on 6.2.

To maximize the error correction capability, k = 7 and t = 15 is chosen. All these parameter combination will enable error correction capability 23.8% of the data length. To summarize, here are the chosen parameters:

Table 6.1: Memory required (bytes) given the value of m

m	**Bytes Required**	**m**	**Bytes Required**
2	53	12	16805897
3	129	13	67166217
4	377	14	268550153
5	1257	15	1073971209
6	4553	16	4295426057
7	17289	17	17180786697
8	67337	18	68721311753
9	265737	19	274881576969
10	1055753	20	1099518967817
11	4208649		

Table 6.2: BCH parameter for m = 6 and n = 63

k	**t**
57	1
51	2
45	3
39	4
36	5
30	6
24	7
18	10
16	11
10	13
7	15

- n: 63
- k: 7
- d: 31
- t: 15

6.2 Key Generation Scheme

As shown in previous chapter, the data protection scheme requires the PUF to generate the key which will be use to generate the final key. The key generation scheme used in this project is a modified version of Figure 3.4 proposed in [32]. Instead of using n = 255, the scheme used in this project will choose n = 63. The parameter *n, k, t, d* is similar with the parameter chosen in previous section, BCH error correcting code. Figure 6.1 illustrates the mentioned scheme. Using this scheme, to generate 256 bits of key, requires 37 blocks of this scheme, which lead to 2331 bits required. 37 blocks is calculated from 256/7=36.57, rounded-up resulting in 37. 7 comes from the key generated from 63 bits of data using this scheme. Since one block needs 63 bits of data, 37 blocks requires 37*63=2331.

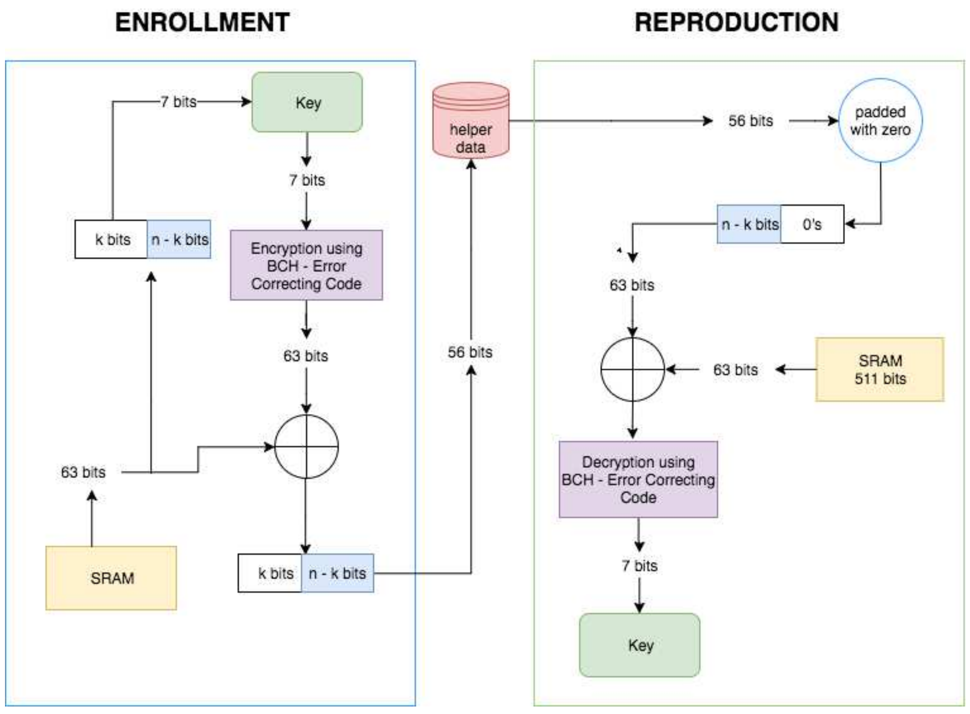

Figure 6.1: Scheme for key generation. n = 63, k = 6, t = 15, d = 31

6.3 Randomize the stable bits to generate the key

In previous chapter, it is mentioned that the location of the bit will be the challenge given to PUF. Before giving the location of stable bits as the challenge, these locations' order inside the list will be randomized. Similar with the explanation on previous chapter, the challenge (locations of stable bits) will be encrypted using AES.

6.4 Chosen SRAM

In an attempt to achieve the thesis goals, the first step is looking for SRAM. There are numerous SRAM types available in the market. To Main requirements on the SRAM are easy to get (a simple google search should show some e-commerce websites to buy from), can be bought in small quantity ($\leq$ 5 pieces), stand-alone component (available without buying extra component, e.g. not embedded in an FPGA), inexpensive (cost less than 5), reasonable memory size ($\geq$ 64kb). These criteria are chosen due to some product only sold to a company or someone that willing to buy in a big quantity or has to be custom made. There are two SRAM types purchased and tested here; Microchip 23LC 1024 and Cypress CY62256NLL. On each SRAM, there are several experiment performed to determine if these SRAMs are a suitable candidate for PUF, such as calculating HD_{intra} and HD_{inter} given the whole memory value as the challenge.

6.4.1 Microchip 23LC1024

The Microchip Technology Inc. 23A1024/23LC1024 is a 1024 Kbit Serial SRAM device. This SRAM is really popular shown by many references available online and several github repository intended just to access this SRAM. The reason of its popularity can be traced to its cheap price, small size and easy to use feature. The price is ranging from 1.5-3.5. This device has eight pins which contribute significantly on its small footprint. It is easy to use because it provides SPI connection which simplified the communication, and has three modes available; SPI (Serial Peripheral Interface), SDI (Serial Dual Interface) and SQI (Serial Quad Interface). Its voltage range also quite large, ranging from 2.5-5.5V. Figure 6.2 shows the Microchip 23LC1024.

There are ten Microchip 23LC1024 SRAMs that were available during experiment. To check whether this SRAM is a justifiable candidate for PUF, several testing are performed. First, the number of 1's and 0's in memory after a start is calculated. Unfortunately, the average distribution of 1's and 0's are not similar, 1's occupy 70% and 0's fill the remaining 30%.

Figure 6.2: SRAM 23LC1024

Second, HD_{intra} and HD_{inter} are calculated on both chips. The calculation are done using twenty data of chip memory values on each chip which retrieved on room temperature, 5V input and 10 seconds interval between each enrollment. From these chips, the average HD_{intra} is 5.75% and the average HD_{inter} is 42.54%.

Third, the effect of voltage variation on the HD_{intra} and HD_{inter} are also evaluated. The calculation are done using chip memory values on each chip which retrieved on room temperature and 10 seconds interval between each enrollment. The voltage range is between 2.5V and 5V with 0.1V increase on a step. On each step, there are three data enrolled. Using these data, voltage variation results in an average HD_{intra} 5.14% and an average HD_{inter} 38.98%.

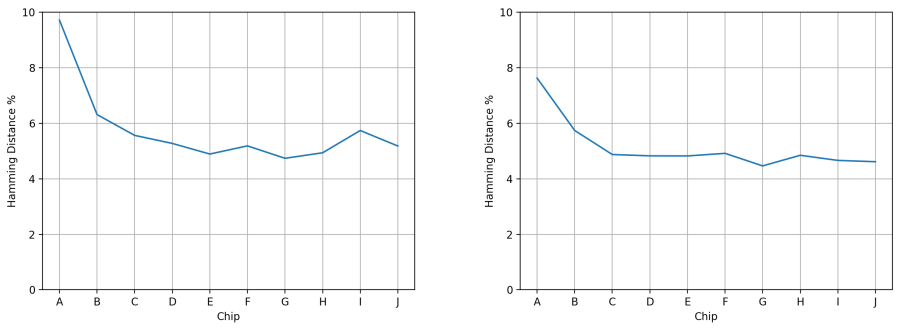

Figure 6.3: HD_{intra} of ten SRAM Microchip 23LC1024. The left is HD_{intra} with constant voltage, the right one is tested based on the voltage variation

6.4.2 Cypress CY62256NLL

The Cypress CY62256NLL is a 256k bit SRAM device. Even though this device is less popular than Microchip 23LC1024, it's still widely used. One of the reason is because this device has an automatic power-down feature, reducing the power consumption by 99.9 percent when deselected. Unlike 23LC1024, CY62256NLL doesn't have a SPI connection which complicate the communication. To communicate, one should utilize its twenty eight pins available. Since it has many pins, it also participate on its significantly larger size compared to 23LC1024. This device is developed using 90nm. Its voltage range is ranging from 4.5V-5.5V. Figure 6.4 shows Cypress CY62256NLL.

Figure 6.4: SRAM CY62256NLL

There are five Cypress CY62256NLL SRAMs that were available during experiment. To check whether this SRAM is a justifiable candidate for PUF, several testing are performed. First, the number of 1's and 0's in an initialization is counted. Fortunately, unlike the 23LC1024, the average distribution of 1's and 0's are similar, both occupy 50% of total bits available.

Next, HD_{intra} and HD_{inter} are calculated on both chips. The calculation are done using twenty data of chip memory values on each chip which retrieved on room temperature, 5V input and 10 seconds interval between each enrollment. From these chips, the average HD_{intra} is 4.94% and the average HD_{inter} is 39.18%.

Last, the effect of voltage variation on the HD_{intra} and HD_{inter} are also evaluated. The calculation are done using chip memory values on each chip which retrieved on room temperature and 10 seconds interval between each enrollment. The voltage range is between 4.5V and 5V with 0.1V increase on each step. On each step, there are ten data enrolled. The average HD_{inter} on voltage variation is 38.75%, while HD_{intra} is 3.55%. Figure 6.5 shows the HD_{intra} between the constant and the variated voltage.

From these data, it can be seen that the voltage variation has little effect on the HD_{intra} and HD_{inter}. This fact shows that SRAM Microchip 23LC1024 and Cypress CY62256NLL can be good candidates for SRAM PUF. Even though such fact exists, one should also pay attention that there's no testing on temperature and aging variation. To ensure whether this SRAM is indeed a good candidate, further experiment on the effect of temperature and aging should be conducted.

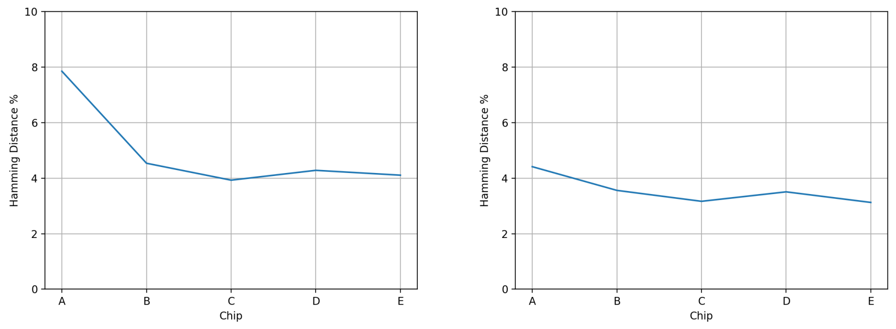

Figure 6.5: HD_{intra} of five SRAM Cypress CY62256NLL. The left is HD_{intra} with constant voltage, the right one is tested based on the voltage variation

6.5 Arduino Mega 2560 as the Embedded Platform

After deciding the SRAMs, the next step is choosing the platform on where the system will be build. There are two major candidates, Arduino and Raspberry Pi. Both are chosen due to its popularity, availability (easy to get), and various types available. High popularity means the debugging process can be done fast and many references are available online to help the system development. Availability is important because this thesis goal should be easily used by anyone. Low availability will reduce significantly reusability of this project and user's interest. Various types available is a good option for system flexibility. For example, if a user want to develop a more complex system on top of this thesis' system or desire to use a more complex error correcting codes, he/she can choose a platform with higher computing capability.

Beside those three factors, another feature to choose Raspberry Pi and Arduino is their GPIO. GPIO availability will enable easy communication between the SRAM and the platform.

Compared to Arduino, Raspberry Pi offers a higher computing capability and relatively easier development. This is because Raspberry Pi is basically a mini linux computer. One can develop using C, C++, Python, etc. Using high-level language will fasten the project development. Unfortunately, Raspberry Pi requires a longer start up time compared to Arduino. It also requires higher electricity power. If one want to use the developed project in embedded area, this two factor is a major trade off.

Due the above consideration, Arduino is chosen. Even though one has to construct the system in C++, this can be a good thing since one can maximize the computing capability easily.

There are various Arduino type available in the market. The chosen Arduino type is Arduino Mega 2560. It is selected because it offers larger memory capability compared to other types, such as 256k bytes of Flash memory, 8k bytes SRAM, and 4k byte EEPROM. Besides, it also has 54 digital I/O pins and 16 analog I/O pins which ease the communication to SRAM CY62256NLL (has 28 pins).

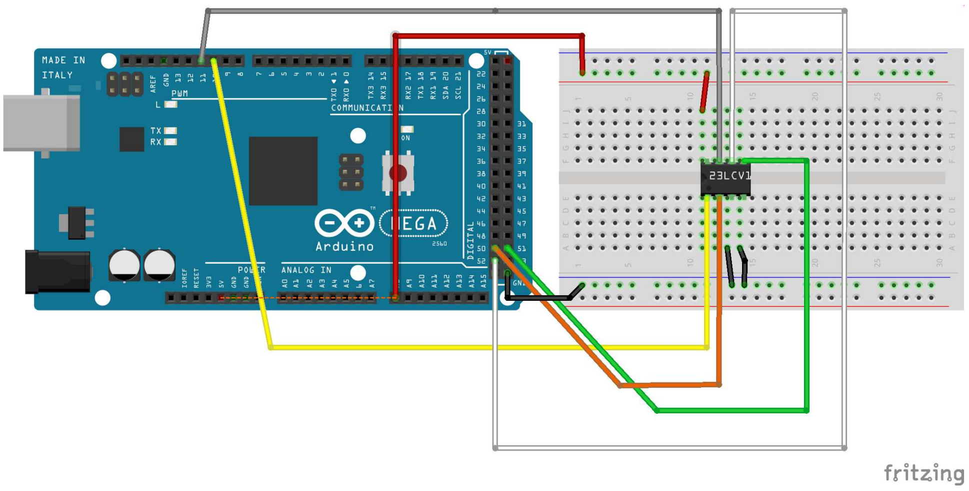

Figure 6.6: Schematic to connect Arduino Mega 2560 with SRAM Microchip 23LC1024

.

.

6.6 Automated PUF Profiling System

To increase the experiment's efficiency, an automated PUF profiling system are constructed. The system consists of a PC, act as a master, and an Arduino connected to an external SRAM which act as a slave. A custom protocol was designed to communicate between them. It is specifically designed to be generic and usable for all types of PUF profiling measurements. The software on Arduino side waits for measurement commands sent by PC on the serial link after booting. The designed protocol are dedicated for voltage control, read bytes, write bytes, and memory disable/enable. The system also supported parallel profiling which significantly increase the effectivity. Figure 6.8 shows the setup to profile four SRAMs Cypress CY62256NLL concurrently using four Arduino.

.

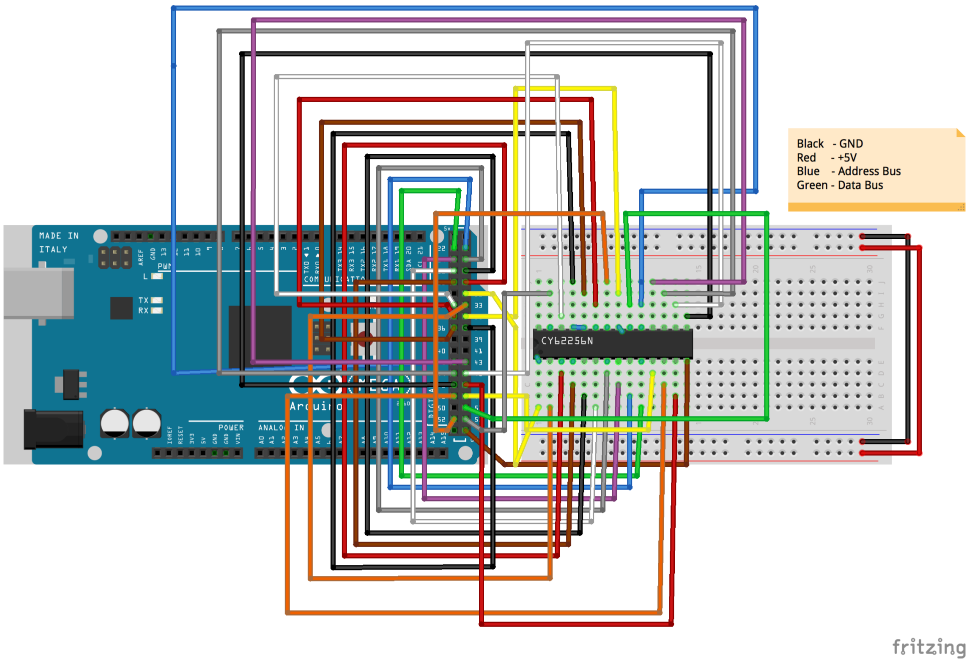

Figure 6.7: Schematic to connect Arduino Mega 2560 with SRAM Cypress CY62256NLL

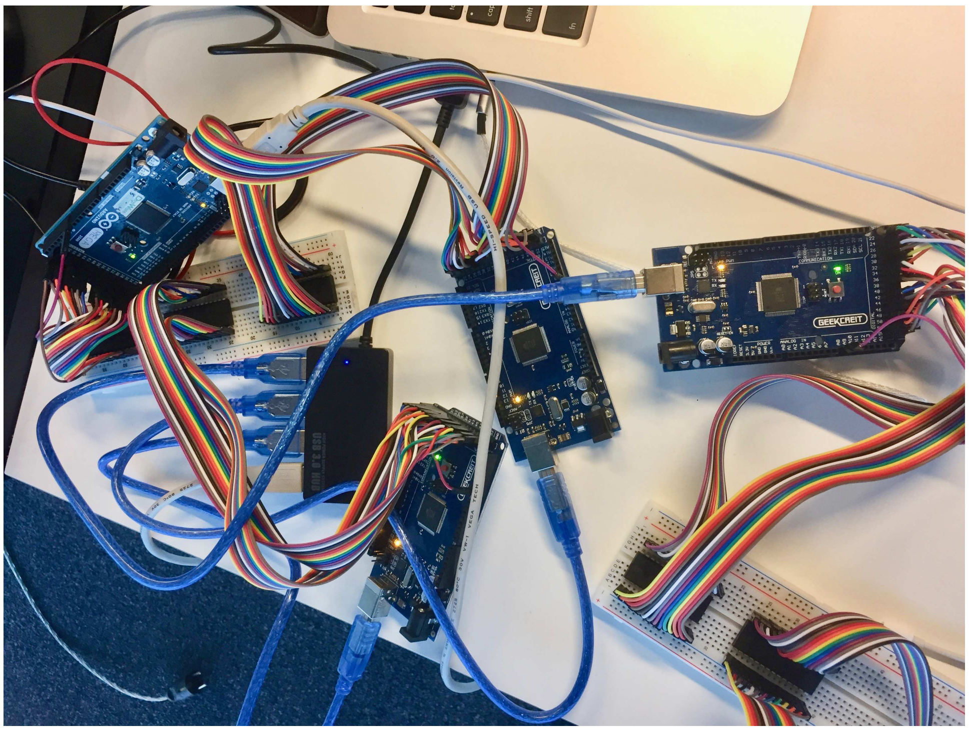

Figure 6.8: Automated PUF profiling setup

6.7 Algorithm to Look for Stable Bits

In this section, the test on stable bits produced by two algorithm are shown. The test was done only on a single chip of each SRAM type, one 23L1024 and one CY62256NLL.

6.7.1 Neighbour Stability Analysis

To use this algorithm, first, data of SRAM bits value from various condition (voltages and time difference between enrollment). Afterwards, the bits which remained stable on those enrollments are located. Then, the rank of remained stable bits are calculated. Last, n bits with highest rank can be used according to the necessity. The higher the rank, the more stable that bit should be. The window size used to calculate the rank is 16 (eight neighbors in each side).

23LC1024

There are 500 data of SRAM bits value used for this chip. The voltage variation is from 2.5V - 5.0V. The time difference between enrollment is ranging from 5 seconds until 1 hour. SRAM 23LC1024 itself has capacity 1048576 bits. After doing the calculation from those five hundred data, there are 413374 remaining stable bits.

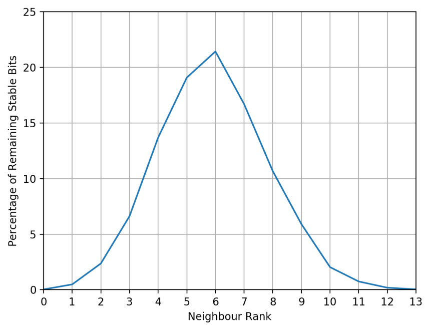

Figure 6.9: Remaining stable bits count according to their rank in SRAM Microchip 23LC1024

From those remaining stable bits, the rank of each bits are calculated. The frequency of bits rank is shown in Figure 6.9. As shown in this figure, there is no bit that has rank 14, 15 and 16. The highest one is only rank 13 with total 172 bits.

Using the bit location as the challenge, the HD_{inter} is 49.76%.

CY62256NLL

Unlike 23LC1024, there are only 109 enrollments done in CY62256NLL. The reason of this decision will be explained in the next section. SRAM CY62256NLL is able to store 262144 bits in its memory. The remained stable bits after 109 enrollments are 84870 bits (32,37%).

The result of the calculation is shown on Figure 6.10. Unfortunately, after the calculation there is no bit that show score 16 (has eight stable neighbor bits on each side). There are two bits that has score 15, 9 bits with score 14, 18 bits with score 13. The highest score count is achieved by score 5 with total count 16502.

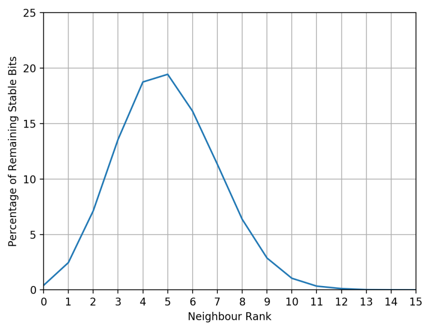

Figure 6.10: Remaining stable bits count according to their rank in SRAM Cypress CY62256NLL

6.7.2 Data Remanence Approach

The result of data remanence analysis on both SRAMs are shown below.

23LC1024

On SRAM 23LC1024, the data remanence analysis is done on time variance between 0-1.0 second. The result can be seen on Figure 6.11. In this figure, it is shown that SRAM 23LC1024 will reach the randomized point if it's turn off for 0.7 second.

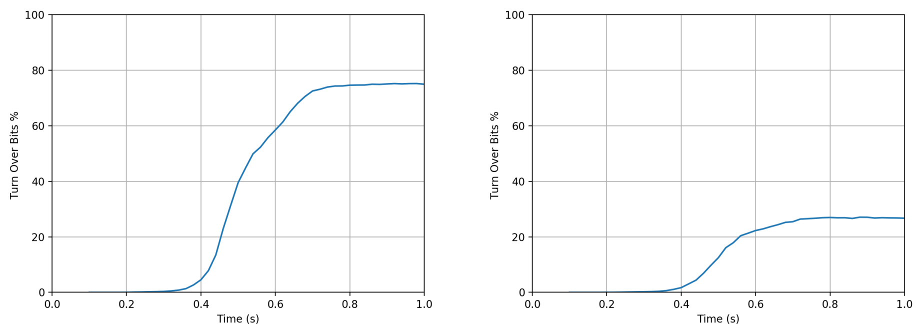

Figure 6.11: Remanence Graph of 23LC1024. Left is remanence 0 and right is remanence 1

CY62256NLL

On SRAM CY62256NLL, the data remanence analysis is done on time variance between 0-1.95 seconds. The result can be seen on Figure 6.11. In this figure, it is shown that SRAM CY62256NLL will reach the randomized point if it's turn off for 0.7 second.

6.7.3 Stability Test on ”Stable Bits”

In this section, test results on the effect of time interval and voltage on ”stable bits” using both algorithm on each SRAM are shown. The effect of aging and temperature is not tested due to limitation on time and equipment. For the effect of time interval testing, the enrollment was done on 16 days with one day gap between enrollment. Voltage effect testing was done on voltage ranging from 4.5-5V. The test are done on 4662 bits which is twice the length of the bits required to generate 256 bits key when using scheme shown on Figure 6.1. The result of time interval testing on SRAM Microchip 23LC1024 is shown on Figure 6.13, while Figure 6.14 displays the result for SRAM Cypress 62256NLL.

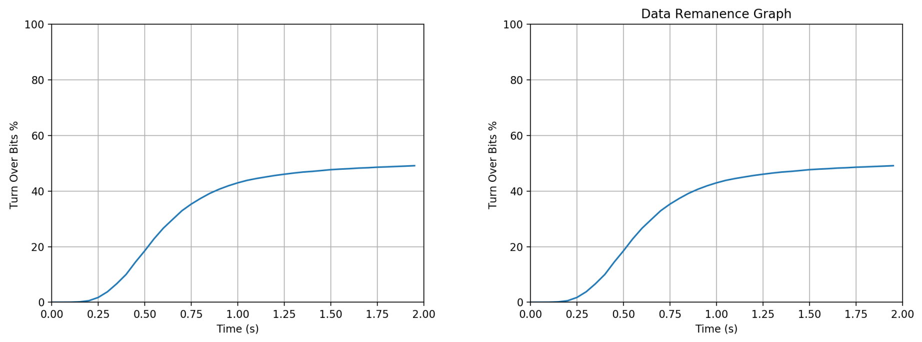

Figure 6.12: Remanence Graph of CY62256NLL. Left is remanence 0 and right is remanence 1

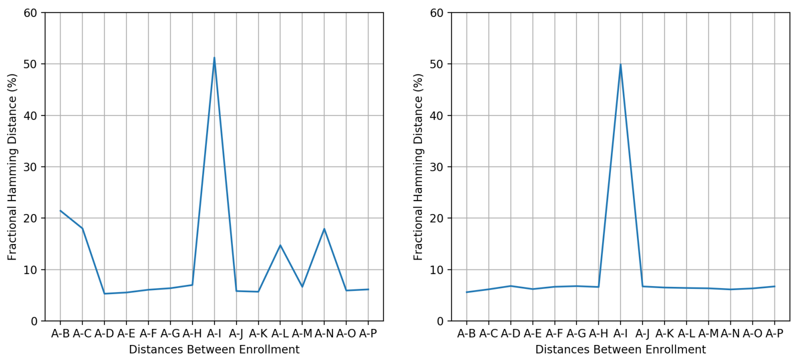

Figure 6.13: Time interval testing results on SRAM Microchip 23LC1024. Left figure is the testing result on stable bits generated using neighbor analysis, while the right one is tested on data remanence generated stable bits. Index A on x-axis refers to enrollment on day 1, B on day 2, etc. Index A-B refers to fractional hamming distance between enrollment on day 1 and day 2.

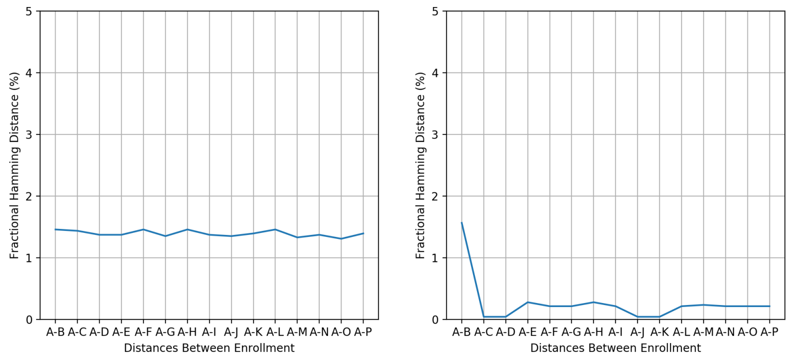

Figure 6.14: Time interval testing results on SRAM Cypress CY62256NLL. Left figure is the testing result on stable bits generated using neighbor analysis, while the right one is tested on data remanence generated stable bits. Index A on x-axis refers to enrollment on day 1, B on day 2, etc. Index A-B refers to fractional hamming distance between enrollment on day 1 and day 2.

Neighbor Stability Analysis

- Microchip 23LC1024
  To get 4662 bits, there are four ranks included; rank 13 with 172 bits, rank 12 with 778, rank 11 with 3092 bits, and 620 bits of rank 10.

  During testing on variated voltage and time interval, the stable bits generated using neighbor stability analysis show a poor performance by having maximum 2389 bits changing (51.24%). The maximum difference is produced when the difference between enrollment is 8 days.

- Cypress CY62256NLL
  To get 4662 bits, there are eight ranks included; rank 15 with 2 bits, rank 14 with 9, rank 13 with 18 bits, 99 bits of rank 12, 289 bits of rank 11, 890 bits of rank 10, rank 9 - 2438 bits, and rank 8 - 917 bits.

  Under the voltage and time interval variation, the stable bits generated using neighbor stability analysis show reliability by having maximum 68 changing bits (1.49%) when time interval between enrollment is a day.

Data Remanence Approach

- Microchip 23LC1024
  To get 4662 bits, strong 1's are generated using time interval only 0.185 second, while strong 0's are calculated when 0.27 second. The

difference between time interval during generation of strong 1's and strong 0's is because the number of 1's that flipped fast are more compared 0's. This also related to the 0's and 1's distribution during normal initialization (0's count for 30% and 1's filled 70%).

Similar like previous algorithm, the stability of bits produced by using this algorithm is not good. The worst change is happen when 8 days is used as time interval between testing, showing as many as 2328 bits (49.93%).

- Cypress CY62256NLL
  Unlike SRAM 23LC1024, time interval on enrolling strong 1's and 0's on CY62256NLL is not different. Both are enrolled using time interval 0.28 seconds to get 4662 stable bits.

  During the voltage and time interval variation, the stable bits produced by using algorithm also shows a promising result. It only account for maximum 73 bits difference (1.56%).

Stability Test Conclusion

Based on these results, SRAM Cypress CY62256NLL is shown to be a more reliable SRAM candidate for PUF than SRAM Microchip 23LC1024. Data remanence also proven to be a better algorithm than neighbor analysis.

6.8 Evaluation

Chapter 7

Conclusions and Future Work

7.1 Conclusions

TODO CONCLUSIONS

7.2 Future Work

TODO FUTURE WORK

Bibliography

[1] Johan Pouwelse, Andr De Kok, Joost Fleuren, Peter Hoogendoorn, Raynor Vliegendhart, and Martijn De Vos. Laws for creating trust in the blockchain age. *European Property Law Journal*, 6(3), Dec 2017.

[2] Eur-lex - 32016r0679 - en. `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679`. [Online; accessed 06-March-2018].

[3] Wendy Goucher. *Information security auditor*. BCS, 2016.

[4] Henk C. A. van Tilborg and Sushil Jajodia. *Encyclopedia of cryptography and security*. Springer, 2011.

[5] Sergei Skorobogatov. Physical attacks and tamper resistance. *Introduction to Hardware Security and Trust*, page 143173, Aug 2011.

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

[7] C. H. Chang, Y. Zheng, and L. Zhang. A retrospective and a look forward: Fifteen years of physical unclonable function advancement. *IEEE Circuits and Systems Magazine*, 17(3):32–62, thirdquarter 2017.

[8] Pim Tuyls. *"Security with Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting"*. Springer, 2010.

[9] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. Fpga intrinsic pufs and their use for ip protection. *Cryptographic Hardware and Embedded Systems - CHES 2007 Lecture Notes in Computer Science*, page 6380.

[10] Top 100 richest bitcoin addresses and bitcoin distribution. `https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html`. [Online; accessed 15-March-2018].

[11] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies. *Security in Computing (5th Edition)*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2015.

[12] H. X. Mel and Doris Baker. *Cryptography decrypted*. Addison-Wesley, 2003.

[13] C. Soanes and A. Stevenson. *Concise Oxford English Dictionary*. Concise Oxford English Dictionary. Oxford University Press, 2008.

[14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[15] M. Cortez, A. Dargar, S. Hamdioui, and G. J. Schrijen. Modeling sram start-up behavior for physical unclonable functions. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, Oct 2012.

[16] A. Maiti and P. Schaumont. The impact of aging on a physical unclonable function. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(9):1854–1864, Sept 2014.

[17] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on fpgas. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, pages 181–197, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[18] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 523–540, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[19] R. Maes, P. Tuyls, and I. Verbauwhede. A soft decision helper data algorithm for sram pufs. In *2009 IEEE International Symposium on Information Theory*, pages 2101–2105, June 2009.

[20] Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, pages 199–216, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[21] M. Taniguchi, M. Shiozaki, H. Kubo, and T. Fujino. A stable key generation from puf responses with a fuzzy extractor for cryptographic authentications. In *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, pages 525–527, Oct 2013.

[22] Roel Maes. *Physically Unclonable Functions Constructions, Properties and Applications*. Springer Berlin, 2016.

[23] Apurva Dargar. Modeling sram start-up behavior for physical unclonable functions. MSc thesis, Delft University of Technology, 2011.

[24] Xiaoxiao Wang and Mohammad Tehranipoor. Novel physical unclonable function with process and environmental variations. *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010.

[25] Vikram G Rao and Hamid Mahmoodi. Analysis of reliability of flip-flops under transistor aging effects in nano-scale cmos technology. *2011 IEEE 29th International Conference on Computer Design (ICCD)*, 2011.

[26] B. Kaczer, R. Degraeve, M. Rasras, K. van de Mieroop, P. J. Roussel, and G. Groeseneken. Impact of MOSFET gate oxide breakdown on digital circuit operation and reliability. *IEEE Transactions on Electron Devices*, 49:500–506, March 2002.

[27] B. C. Paul, Kunhyuk Kang, H. Kufluoglu, M. A. Alam, and K. Roy. Temporal performance degradation under nbti: Estimation and design for improved reliability of nanoscale circuits. In *Proceedings of the Design Automation Test in Europe Conference*, volume 1, pages 1–6, March 2006.

[28] P. Magnone, F. Crupi, N. Wils, R. Jain, H. Tuinhout, P. Andricciola, G. Giusi, and C. Fiegna. Impact of hot carriers on nmosfet variability in 45- and 65-nm cmos technologies. *IEEE Transactions on Electron Devices*, 58(8):2347–2353, Aug 2011.

[29] Kan Xiao, Md. Tauhidur Rahman, Domenic Forte, Yu Huang, Mei Su, and Mohammad Tehranipoor. Bit selection algorithm suitable for high-volume production of sram-puf. *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014.

[30] Muqing Liu, Chen Zhou, Qianying Tang, Keshab K. Parhi, and Chris H. Kim. A data remanence based approach to generate 100sram physical unclonable function. *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017.

[31] Hyunho Kang, Yohei Hori, Toshihiro Katashita, Manabu Hagiwara, and Keiichi Iwamura. Performance analysis for puf data using fuzzy extractor. In Young-Sik Jeong, Young-Ho Park, Ching-Hsien (Robert) Hsu, and James J. (Jong Hyuk) Park, editors, *Ubiquitous Information*

Technologies and Applications, pages 277–284, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[32] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura. Cryptographic key generation from puf data using efficient fuzzy extractors. In *16th International Conference on Advanced Communication Technology*, pages 23–26, Feb 2014.

[33] Google patents. `https://patents.google.com/?q=sram\&q=puf\&oq=srampuf`. [Online; accessed 06-March-2018].

[34] Google scholar. `https://scholar.google.nl/scholar?as\_vis=1\&q=srampuf\&hl=en\&as\_sdt=1,5`. [Online; accessed 06-March-2018].

[35] Broadkey - intrinsic id — iot security. `https://www.intrinsic-id.com/products/broadkey/`. [Online; accessed 06-March-2018].

[36] Polarfire evaluation kit. `https://www.microsemi.com/products/fpga-soc/design-resources/dev-kits/polarfire/polarfire-eval-kit`. [Online; accessed 06-March-2018].

[37] The error correcting codes (ecc) page. `http://www.eccpage.com`. [Online; accessed 09-November-2017].