Open Source Secure Data Protection and Key Storage Scheme Using Off-The-Shelf SRAM Component Based on Software-Only SRAM PUF Technology

Ade Setyawan Sajim

Delft University of Technology

Open Source Secure Data Protection and Key Storage Scheme Using Off-The-Shelf SRAM Component Based on Software-Only SRAM PUF Technology

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

10th April 2018

Author
Ade Setyawan Sajim

Title
Open Source Secure Data Protection and Key Storage Scheme Using Off-The-Shelf SRAM Component Based on Software-Only SRAM PUF Technology

MSc presentation
22nd January 2018

Graduation Committee
Dr. Ir. Johan Pouwelse & Delft University of Technology
Dr. Ir. Stephan Wong & Delft University of Technology

Abstract

SRAM PUF has a potential to become the main player in hardware security. Unfortunately, the current available solution is usually locked to specific entity. Here, we initiate an open source project to develop SRAM PUF technology using off-the-shelf SRAM. We also present an automated testing and enrollment system on SRAM PUF using Arduino and PC. There are two types of SRAMs tested here; Microchip 23LC1024 and Cypress CY62256NLL. In addition, an idea to create a strong PUF using SRAM is also proposed here. Using a collection of bits as a challenge, the stable bits are permutated among themselves to create a challenge which has a tremendous number of possibilities. Furthermore, we also present a secure data protection and key storage scheme using SRAM PUF. The proposed scheme is influenced by multi-factor authentication. Using a combination of a PUF-generated key and user's password, a derived key is produced and utilized as the final key to protect user's data or/and user's key.

Preface

TODO MOTIVATION FOR RESEARCH TOPIC

TODO ACKNOWLEDGEMENTS

Ade Setyawan Sajim

Delft, The Netherlands
10th April 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 to describe the uniqueness of everyone. Identity's role in our daily life is unquestionable. Society requires identity systems to enable identity-requiring transactions at scale, allowing procedures that require the formal asking and answering of identity queries in place, allowing millions of transactions to occur. 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 of these elements.

Along with the rise of the digital age, identity also redefines itself. Identity in the digital world, can be referred as digital identity, is split into multiple domains. Our Facebook identity does not correlate directly to our Twitter identity or to most other domains. Digital identities are scattered, vary from one Internet domain to another. Scattered identities which locked to multiple entities leads to a problem where users are helpless in front of an authority who can deny their identity or even confirm a false identity. This phenomenon ignites a problem where users are not in control of their identity. There's no clear construction and agreement on how to build the digital identity which usable across platforms. This is an unfortunate thing since lack of digital identity also limits the development and delivery of efficient, secure, digital-based economy and society [1]. The failure to solved the digital identity problem issue even looks a bit strange since we already have *public key cryptography* since 1984, introduced by Chaum [2], which enable secure communication between parties without the hassle of key distribution problem and also provide valid digital signatures. Using public key cryptography, anyone who wants to send any message to a recipient needs to encrypt their message using the recipient's public key. Afterwards, the recipient can read the message after decrypting the received message using its private key.

To fix the scattered identity issue, a solution was proposed: one should be able to store their encrypted data in their own devices or in their own preferred service.

To use the data, a service has to ask the data owner for the private key which will be used to decrypt the data. Using this concept, everyone has to keep their secret keys secure and solely in their possession. A centralized storage of private keys is out of question since it will be a honeypot for cyber attacks. Simply put, keeping secret keys secure is the cardinal problem to solve here.

All problems mentioned above leads to a thinking, identity and secure key storage need to be solved in a decentralized manner. In 2012, a new concept called *self-sovereign identity* (SSI) arise [3]. Self-sovereign identity is a decentralized identity concept which capable of authenticating statements, without any central organization, point-of-failure or any possibility of data tracking [4]. Self-sovereign identity will be able to give users full control over their identity. In 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 [5].

Johan Pouwelse and Martijn de Vos in [4] proposed an SSI design which focused on data protection. Data protection itself is related to securing data against unauthorized access [6]. Their proposal is described by a concept where the user data are encrypted and never leave the device/domain. Any operation which requires the data, such as authentication, will require symmetric encryption on the encrypted data. This encrypted data should be securely protected and the domain should be trustworthy. In addition, the key used to encrypt and decrypt data should be kept securely.

The most common way to store 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 a product which implements 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* [7]. An attacker may also use a side channel information to retrieve any information about the key. There are numerous other techniques for 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 [8]. Unlike cryptographic algorithm security which usually relies on a hard-to-solve mathematical problem, PUF idea stems from using

hardware features designed to utilize the physical random nanoscale disarray phenomena [9]. These 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, [4] 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.

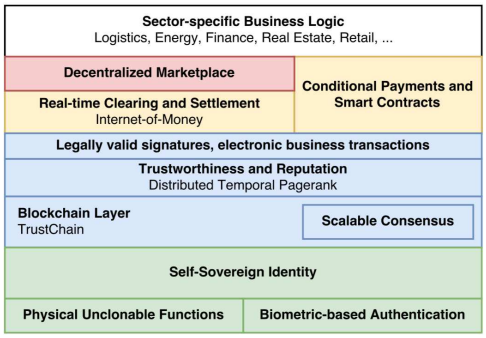

Figure 1.1: Detailed technology portfolio for trust creation in the blockchain age [4]. As shown in the bottom of this figure, Physical Unclonable Functions and biometric-based authentication are utilized to secure the self-sovereign identity.

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 undefined states [10]. The initialized values on the memory cells are also random and unique to each SRAM. Based on these properties, SRAM is considered as a reasonable 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 first introduced by Guajardo [11]. 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. To handle the noise, error correction code is usually utilized.

1.3 Problem Statement

Since 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 [12]. The number of articles in sch olar.google.com also exhibit a high occurrences, shown 2,120 articles (citations and patents are not included) [13]. Even though these facts indicate 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 or 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 leaders in SRAM PUF technology, has a software-only solution which able to generate unique keys and identities for nearly all microcontrollers without a need for security-dedicated silicon [14]. 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 have another solution for SRAM PUF which is focused on hardware IP (and supporting software/firmware) to enable designers to implement PUFs within 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 an expensive contract. Similar to the software-only solution they offer, they also don't put the explicit price to use this product. An example of a product that uses this solution is FPGA Microsemi Polarfire [15].

The SRAM PUF field lacks an Arduino, Linux, or GCC type of open reference implementation. A quick lookup in Github shows that 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 a simulation. The communities seem to haven't established a wide agreement on a which approach yields the strongest security properties.

An additional issue that we would like to address is SRAM PUF's application. As mentioned in Chapter 1, the importance of securing key and user's data is getting higher, especially with the introduction of self-sovereign identity. There are already many SRAM PUF applications published, but sadly, there isn't any example working project that tries to integrate SRAM PUF in self-sovereign identity concept. Most PUF applications are designed for authentication [16] [17] [18] [19] [20] [21] and generating cryptographic keys [18] [22].

Based on these facts, we believe the next challenge for this field is to discover a common approach. 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. Furthermore, we also try to initiate an attempt of integration between PUF and self-sovereign identity by providing a scheme to protect user's data and key. This project will be useful in the process of self-sovereign identity development.

To understand our intention in this thesis better, this thesis' problem statement is presented here. The problem statement of this thesis is:

> *How to develop an open source secure data protection and key storage scheme using off-the-shelf SRAM component based on SRAM PUF technology?*

Derived from the problem statement, there are two goals defined in this thesis. The first goal is to devise a secure data protection and key storage scheme based on SRAM PUF technology. The scheme should work using off-the-shelf SRAM. In addition, the constructed SRAM PUF has to work without any hardware design, or in other words, *software-only* construction. The data protection and key storage functions inside the scheme will be helpful in addressing the problem of self-sovereign identity and keeping the secret key. The next goal is to create a sharing ecosystem for the evolution of our data protection and key storage scheme. The ecosystem should be easily accessed and understood to encourage the academics and commercial parties to use and develop the ecosystem together. The easiest step to achieve this goal is by making our thesis as an open source project.

1.4 Contributions

In our work, we strongly believe in open source idea and communities involvement when developing a system. Combined with the problems and potential of SRAM PUF mentioned before, and also driven by passions, willing to learn, and extensive brainstorming, this thesis generates several additions into the state of the art of SRAM PUF knowledge. This thesis' contributions are explained below:

- *A system to enable SRAM PUF-based secure data protection and key storage scheme using off-the-shelf SRAM.* This scheme is influenced by multi-factor authentication. Using a combination of a PUF-generated key and user's password, a derived key is produced and utilized as the final key to protecting user's data or/and user's key.

- *An open ecosystem to develop SRAM PUF using off-the-shelf SRAM.* The ecosystem consists of source code (Arduino and python code) and recommendations on how to test a possible SRAM candidate that might be used as an SRAM PUF. Using the system, we present testing results of two off-the-shelf SRAMs; Microchip 23LC1024 and Cypress CY62256NLL. Both SRAMs are tested on voltage variation and time interval between enrollment testing. The system also provides a bit selection algorithm; data remanence. This algorithm is also tested on these two SRAMs along with another bit selection algorithm, neighbor stability analysis. Neighbor stability analysis is not included in the final version of the project due to worse performance compared to data remanence analysis.

- *A concept to devise a strong PUF using SRAM PUF.* Normally, SRAM PUF is considered a weak PUF due to the limitation of possible challenge-response pairs. We propose to use a set of bit locations as the challenge since when using this concept, the number of possible pairs is the permutation of total bit locations over the required number of bit locations which basically leads to a significantly large number.

1.5 Outlines

After explaining a brief review of SRAM's potentials and problems, problem statement and our contributions in this chapter, Chapter 2 continues with an overview of security, cryptography, symmetric encryption, key derivation function and multi-factor authentication. Explanations of PUF and SRAM PUF are also presented in that chapter. Chapter 3 describes our proposed SRAM PUF development system, our idea on how to create strong PUF using SRAM PUF, and a scheme to enable secure data protection and key storage using SRAM PUF. Chapter 4 shows our implementation, experiments and results. Last chapter, Chapter 5, summarizes this thesis and also gives our view on possible improvements on this project.

Chapter 2

Related Work

This chapter examines some background theory related to security, cryptography, and PUF. A brief review of security is presented, followed by explanations on symmetric cryptography, key derivation function and multi-factor authentication. Then, theories related to PUF and SRAM PUF are described, continued by a short evaluation on how to generate a key using SRAM PUF.

2.1 Security Requirements and Cryptography

A perfect and 100% secure system is the holy grail of all computing system. Unfortunately, such thing doesn't exist. The best way to achieve that goal is by designing a system to be as secure as possible in a limited scope. To help defining a secure system, common security requirements are mentioned. According to [23], there are four elements on common security, which are:

- *Confidentiality*: a piece of information should be accessible only to an authorized user. 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 authorized subjects. For example, data should be kept 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.

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 [24]. 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 focuses their definition using science and mathematics.

According to [25], modern cryptography can be defined as "the scientific study of techniques for securing digital information, transactions, and distributed computations." The algorithm which uses cryptography as their main point is called cryptographic algorithm.

Since the birth of cryptography, its main concerned is usually related to 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 the asymmetric algorithm (can be referred as public key encryption or public key cryptography), there are two keys utilized; private key and public key. A public key is utilized for encryption and a private key is used for decryption. One of the main advantages of symmetric encryption over asymmetric encryption is it requires less computational power which makes it suitable to use in embedded devices.

2.2 Symmetric Encryption

According to [25], symmetric encryption consists of three algorithms which are:

- *Gen*: key-generation algorithm
- *Enc*: encryption algorithm
- *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. AES is based on a design principle known as a substitution-permutation network, a combination of

both substitution and permutation. AES has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the plaintext into the ciphertext. The numbers of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

There are four major parts inside AES; *KeyExpansions*, *InitialRound*, *Rounds* and *FinalRound*. In KeyExpansions, the round keys are derived from the cipher key using Rijndael's key schedule. Inside a normal round, there are four stages required to do; *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. SubBytes refers to a non-linear substitution step where each byte is replaced with another according to a lookup table. ShiftRows means a transposition step where the last three rows of the state are shifted cyclically a certain number of steps. MixColumns contains a mixing operation which operates on the columns of the state, combining the four bytes in each column. AddRoundKey involves a process where each byte of the state is combined with a block of the round key using bitwise xor. The difference between InitialRound, Rounds, and FinalRound is InitialRound only contain AddRoundKey, FinalRound does not has MixColumns inside, and Rounds just filled with those four stages.

An encryption can be done by following all these four parts. To convert ciphertext into the original plaintext, it's only required to apply a set of reverse rounds using the same encryption key.

2.3 Key Derivation Function

Besides the encryption algorithm, a *key derivation function* (KDF) is one of the most essentials components of the cryptographic system. Its importance is due to its ability to convert a stable secret, usually contain sufficient amount of randomness but non-uniformly distributed, Z into one or more cryptographically strong secret keys $k \, \epsilon \, 0, 1^K$. Cryptographically strong itself refers to indistinguishability by feasible computation from a random uniform string of the same length [26]. KDF can also be referred as a strong extractor.

A popular example of KDF is a keyed cryptographic hash function. The difference between keyed cryptographic hash function and a normal hash function is a keyed hash function requires an additional *salt* as an input (besides the key to derived). There are three requirements need to be fulfilled

as a secure cryptographic hash function; *preimage resistant, second preimage resistant, and collision resistant*. Preimage resistant means it should be hard to find a message with a given hash value. In second preimage resistant, if one message is provided, it should be hard to find another message with the same hash value. Last, collision resistant refers to difficultness to find two messages with the same hash value.

2.4 Multi-factor Authentication

As mentioned in Section 2.1, authentication refers to assuring any piece of information is correctly identified. Authentication can be done using any of these elements/factors; knowledge (something they and only they know), possession (something they and only they have), or inherence (something they and only they do/are). If two or more elements are combined together for authentication, this leads to *multi-factor authentication*. To understand the security level among all possible combinations, Figure 2.1 is provided. The highest possible security level is when these three factors are combined together.

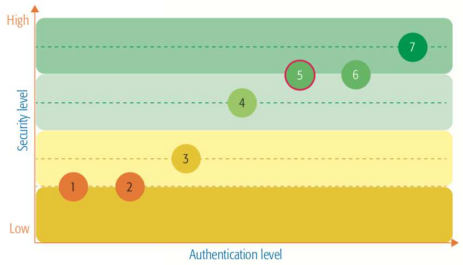

Figure 2.1: Authentication systems security levels: (1) knowledge; (2) possession; (3) knowledge + inherence; (4) inherence; (5) possession + inherence; (6) knowledge + inherence; (7) knowledge + possesion + inherence [27].

2.5 PUF

A physical unclonable function is an entity that utilizes 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 [9]. PUFs leverage this variability to derive secret information that is unique to the chip. This secret can be referred 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 is 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).

$$R_i < -PUF(C_i) \tag{2.1}$$

The common application on using PUF usually requires two phases; the first phase is called *enrollment* and the second one is usually referred as *validation*. In enrollment, a number of CRPs are gathered from a PUF and then stored. In validation 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 this challenge and response. The validation phase can also be referred as *reproduction* phase since this phase involves a reconstruction of a response given a challenge.

According to [9], to be qualified as PUF, a device should fulfill several characteristics below :

- *Reliable*: A response to the same challenge should be able to be reproduced over time and over a 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 a different device.
- *Unclonable*: Challenge-response pairs mapping of a device should be unique and cannot be duplicated.
- *Physically Unbreakable*: Any physical attempts to maliciously modify the device will result in malfunction or permanent damage.

2.5.1 PUFs Classification

Based on the number CRPs, PUFs can be divided into two categories [28]:

- Strong PUFs
  Strong PUFs can be identified by having a 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.

Besides the number of CRPs, PUFs can also be categorized based on their physical design. There are two major categories, extrinsic and intrinsic.

Extrinsic means that it needs extra hardware added to the PUF component. There are two subcategories of extrinsic PUFs, non-electronic and analog electronic PUFs. Some examples 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 with 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.

2.5.2 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 used as a way to help to define 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 the difference between the two responses resulting from applying a challenge twice to a PUF device [29]. 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 (HD_{intra}) is 0% and inter-chip fractional hamming distance (HD_{inter}) is 50%. Due to noises, normally PUF devices has $HD_{intra} \leq 10\%$ and HD_{inter} 50%. The identification goal will not be achieved if there is an overlap between HD_{intra} and HD_{inter} [30]. Overlap will happen if the HD_{intra} is too large and HD_{inter} is too small, e.g. HD_{intra} is 35% and HD_{inter} is 30%.

2.5.3 Helper Data Algorithms and Fuzzy Extractor

There are two issues if PUF raw responses are 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 to 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 refer Helper Data Algorithm as *fuzzy extractor* since both are capable of converting noisy information into keys usable for any cryptographic application [31] [32].

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 [33]. 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 [32]. 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 [34], 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 [35], Taniguchi et. al illustrated the generation and reproduction procedure of fuzzy extractor on PUF which is shown in Figure 2.2.

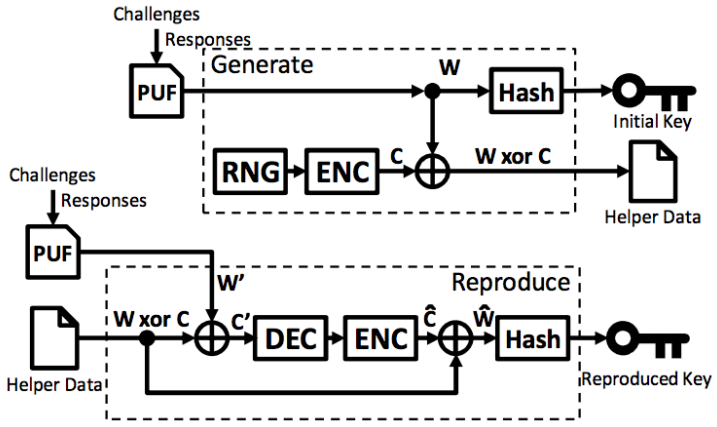

Figure 2.2: Two procedures inside fuzzy extractor; generation and reproduction [35].

2.5.4 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 subcategories, hard-decision and soft-decision. Hard-decision operates on a 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, overcomplicate 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 a 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 are 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 at fixing random errors. BCH codes have an advantage where it requires less computing resource when working on the same parameter compared to Reed-Solomon codes.

2.6 SRAM PUF

SRAM PUF was first proposed by Guajardo and Holcomb in 2007. SRAM PUF uses 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 at the beginning of this chapter, during enrollment, challenge-response pairs are gathered. In SRAM PUF, there are two types 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 a challenge, an address in there can refer to an address of a byte, a bit, or a sequence of bytes or bits.

2.6.1 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 2.3. 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 [29]. Transistors Q3 and Q4, referred 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 lose its state shortly after power down [36].

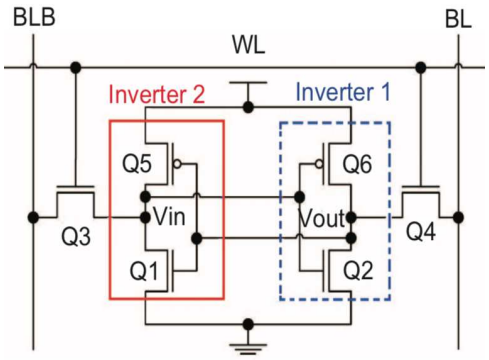

Figure 2.3: A 6-T CMOS SRAM cell [29].

During manufacturing, there are small differences between each SRAM cell due to process variation which leads to a mismatch in the cell [37]. 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
   A non-skewed cell has no preference during its startup 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.

2.6.2 Problem: Noise

Similar to most electronic components, SRAM PUF is also affected by any external influence which leads 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 [38]. 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 a 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 a 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 [38].

- Aging
  Aging is related to changes in the silicon after usage for a long time

[39]. There are three main effects related to the aging of a circuit; time-dependent dielectric breakdown (TDDB), bias temperature instability (BTI) and hot carrier injection (HCI). TDDB is associated with the creation of a conduction path through the gate transistor structure which causes an increase in power consumption and the circuit delay [40]. BTI causes a degradation of the transistor threshold voltage [41]. HCI generates a change in the transistor threshold voltage [42]. HCI is caused by a high current in the transistor channel injecting charges into the gate oxide during the switching.

2.6.3 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 locations 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 a 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 algorithms to search for stable bits.

Neighbor Analysis

The first algorithm is using the rank of total stable neighbors [43]. They argue that the cells which 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. In this algorithm, all the stable cells are given weight according to the number of stable bits surrounding it. The more stable neighbor cells it has, the higher weight it gets. For example, if a cell is not stable, it is given zero as its score. If it is stable, at least it will get score one. If it only has one stable neighbor on each left and right side, it will get score two as result of an addition of one from being a stable cell and one from having a stable neighbor on both sides. To get score three, it needs to be stable and has two stable neighbors on left and right sides. 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 of SRAM cells value first. The data should be retrieved in various condition, for example,

different voltages, temperatures, and time differences between enrollment. Afterwards, using the data gathered, the location of all stable bits in SRAM need to be located. A stable bit has to has the same value in all enrollment. Last, the neighbor analysis algorithm is performed to get the most stable bits in SRAM.

Data Remanence Approach

Another bit selection algorithm is by using data remanence of SRAM cell [44]. 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. When using this approach, one should carefully determine the temporal power down time. If the temporal power down time is too long, then the data written in the array is completely collapsed and the SRAM values will go back to its uninitialized state. On the other hand, if the temporal power down period is too short, then the data will stay in the previously written state.

A significant advantage using this algorithm compared to the previous one is a much shorter time required to locate stable bits. Using neighbor analysis, there are many SRAM values need to be gathered first which might take hours or days. Locating stable bits from hundreds of data probably also take time as well. If data remanence approach is utilized, there is no need to gather many data. One only need to determine the temporal power down required to get strong bits required. Since usually the temporal down period required is less than 0.5 seconds, this analysis only takes less than one or two minutes.

2.7 Key Generation using SRAM PUF

In this section, there are two schemes for key generation produced by Hyunho Kang et. al. Both constructions were built in 2014. The first construction, shown in Figure 2.4, is utilizing random number generator (RNG). This design was perfected in the second design shown in Figure 2.5. 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.

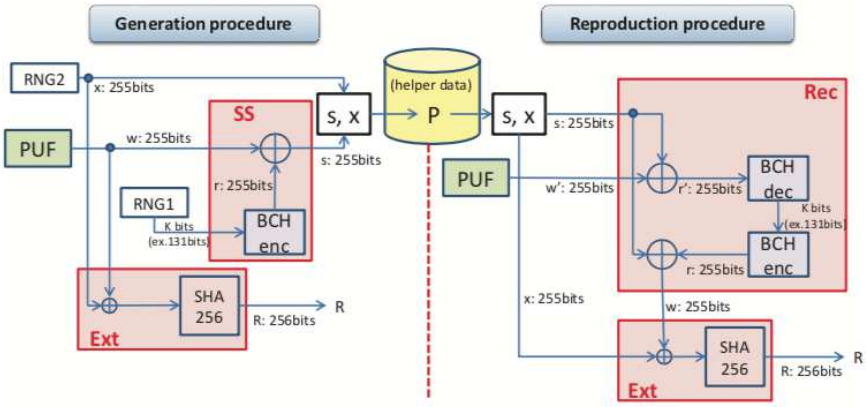

Figure 2.4: Implementation diagram using fuzzy extractor (N = 255) [45].

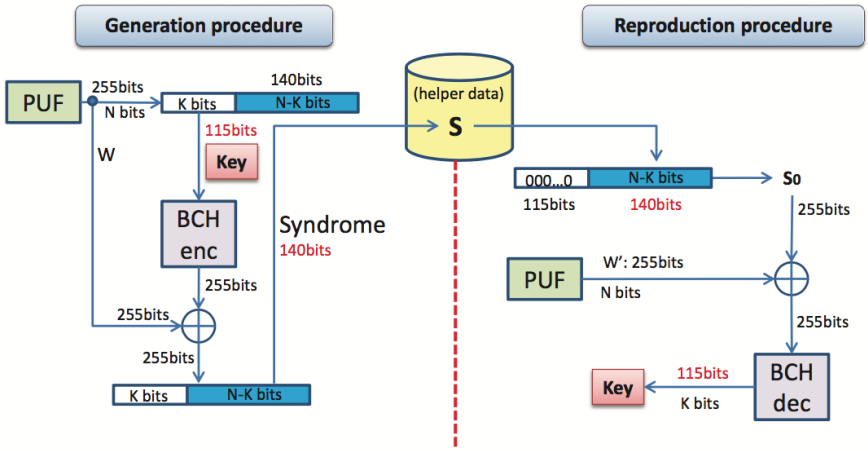

Figure 2.5: Implementation diagram for efficient fuzzy extractor based on the syndrome (N = 255) [46].

Chapter 3

Proposed System

This chapter contains our proposed system to achieved our goals which explained in Chapter 1. First, use cases, assumptions and requirements on our system are presented. Then, our idea to use bits locations as a PUF challenge is shown, followed by reasonings on our chosen embedded platform, Arduino. The selected error correcting code used in the system is also explained. Last, we present our data protection and key storage scheme and also our way to generate key using SRAM PUF.

3.1 Use Case, Assumptions and Requirements

As mentioned in the first chapter, a subset of this thesis goal is to provide a secure data protection and key storage scheme using SRAM PUF. This thesis goal describes the use case of our proposed system. In addition, several assumptions are also made to focus the thesis approach. First, 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. Second, an attacker cannot access the SRAM directly. An attacker may gain the knowledge of the helper data and challenge used in the PUF concept. We also have define a set of requirements related to our system. 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 of 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, instead of building our own software, it is preferred to use that project. This will significantly reduce the time consumed on constructing the whole project. Using other project source code can also increase the collaboration atmosphere. In addition, this requirement may help this project to be known by others since they might introduce our project as one of the projects that uses their code.

4. Key-length security level
   The goal on the 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.

3.2 Bit Locations as SRAM PUF Challenge - Leads to Strong PUF

As mentioned in the previous chapter, the challenge for SRAM PUF can be either the whole SRAM memory or only a fraction of it. In this thesis project, a fraction of SRAM is decided to be the challenge, specifically a set of bit locations. For example, if bit values with length 1000 is the goal, a set of 1000 locations is required as an input to PUF device. Below are the reasons why this decision is taken:

1. Stable bits tend to be scattered all around SRAM memory.

2. If there's a burst error on a bit location inside the challenge, this error will not affect many locations in a challenge since this burst error may only lead to a single location. If a location related to multiple bits is used as the challenge, a burst error will affect many bits generated. For example, if locations of bytes are used as the challenge, a burst error might lead to 8 bits errors in the response generated.

3. There are a huge challenge-response possibilities. The number of possibilities is calculated from the permutation of the required bits and the available bits using Equation 3.1.

$$P(n, r) = \frac{n!}{(n - r)!} \tag{3.1}$$

For example, if there are 1000 bits as the challenge candidates and only 200 bits are required as a challenge, there are as many as 5.2186 71792E+590 possibilities. Due to this large possibilities of challenge-response pair, this idea will lead to a **strong PUF** (as mentioned in section 2.5.1, a strong PUF can be identified by having a large number of CRPs).

In addition, before generating the challenge, the location of stable bits need to be identified first. The location of stable bits can be detected by using bit selection algorithm mentioned in section 2.6.3. After the location of stable bits is identified, during the generation of a challenge, the locations' order inside the challenge will be randomized.

3.3 Arduino Mega 2560 as the Embedded Platform

One of the important details of our system is choosing the platform on where the system will be built. 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 wants 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. Besides 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 startup time compared to Arduino. It also requires higher electrical power. If one wants to use the developed project in the embedded area, this two factor is a major trade-off.

Due to 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 types available on 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).

3.4 BCH Codes as Error Correcting Codes

As mentioned in the previous chapter, BCH codes are a flexible ECC shown by multiple parameters 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 [47]. This code is selected because it can support m ranging from 2-20 which mean the length of the code that 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 parameters 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 [47], the decoding method requires the largest memory compared to other procedures. There are six parameters that depends on m which are elp, d, l, $u_l u$, s, and err. Table 3.1 shows the required memory given the m value.

Table 3.1: 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		

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 3.2.

Table 3.2: 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

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:

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

3.5 Data Protection and Key Storage Scheme

Figure 3.1 shows the scheme to protect user's data and key. On an attempt to protect the user's data and key, our proposal is divided into three major parts, first is generating the final key, and the rest is using the final key either to encrypt or decrypt 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 that used here is password. User's password is combined with the PUF-generated key to generate a **final key** using HMAC. The input message to

the HMAC is the user's password and the HMAC's key is PUF-generated key. The HMAC function proposed to use is HMAC-SHA3 with key length 256 bits. The final key can be used to encrypt and decrypt user data. To decrypt and encrypt the data, a symmetric encryption algorithm is preferred over the asymmetric one. The symmetric encryption algorithm used is AES with key length 256 bits. If the data is switched to user's key, the data protection scheme proposed here can be also referred as **key storage scheme**.

3.6 Key Generation Scheme

As shown in the previous section, the data protection and key storage scheme requires the PUF to generate the key which will be used to generate the final key. The key generation scheme used in this project is a modified version of Figure 2.5 proposed in [46]. Instead of using n = 255, the scheme used in this project will choose n = 63. The parameter *n, k, t, d* is similar to the parameter chosen in the previous section, BCH error correcting code. Figure 3.2 illustrates the mentioned scheme. Using this scheme, to generate a key with length 256-bits requires 37 blocks of this scheme, which lead to 2331 bits required. 37 blocks are 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 require 37*63=2331 bits.

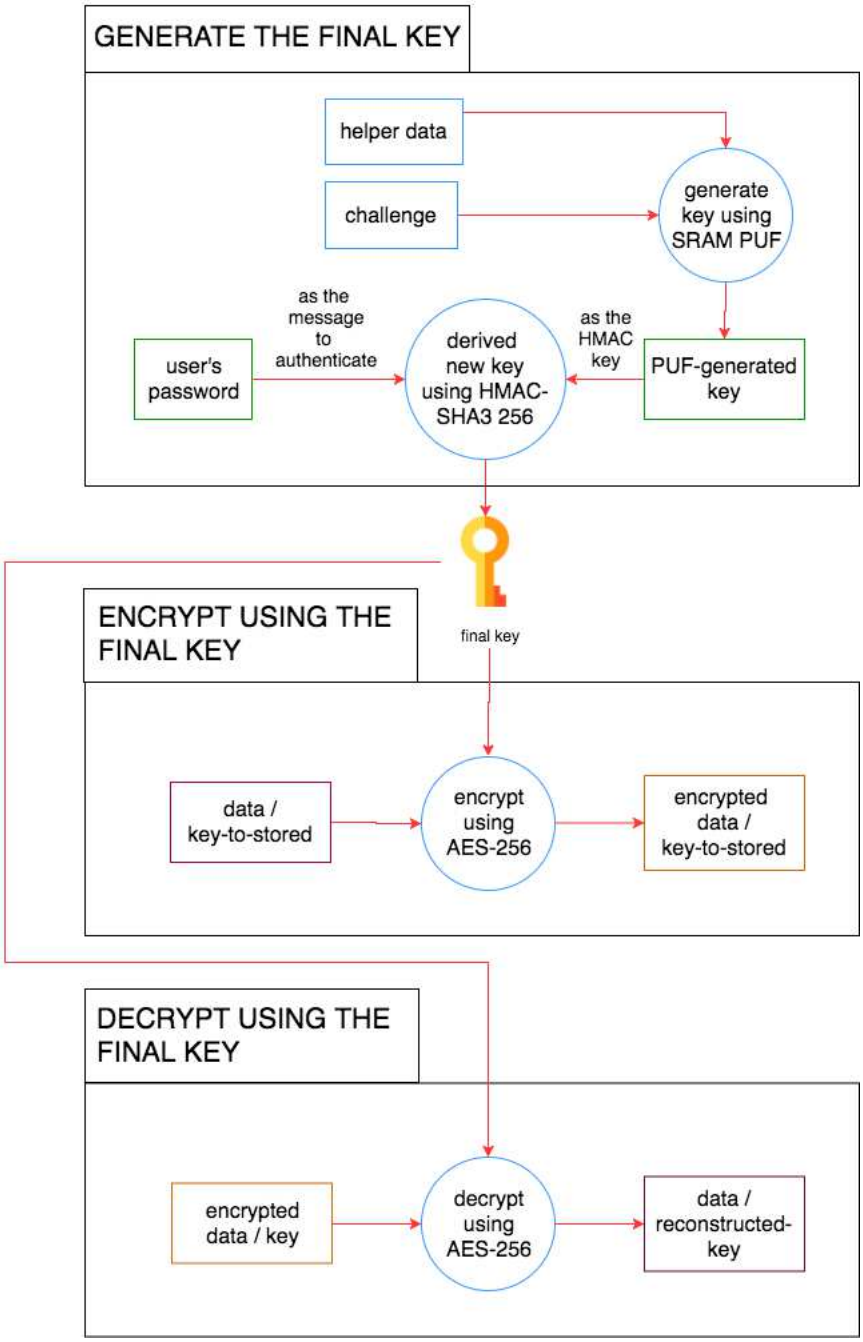

Figure 3.1: Scheme for secure data Protection and key storage. There are three stages in here; generate the final key, encrypt using the final key and decrypt also using the final key.

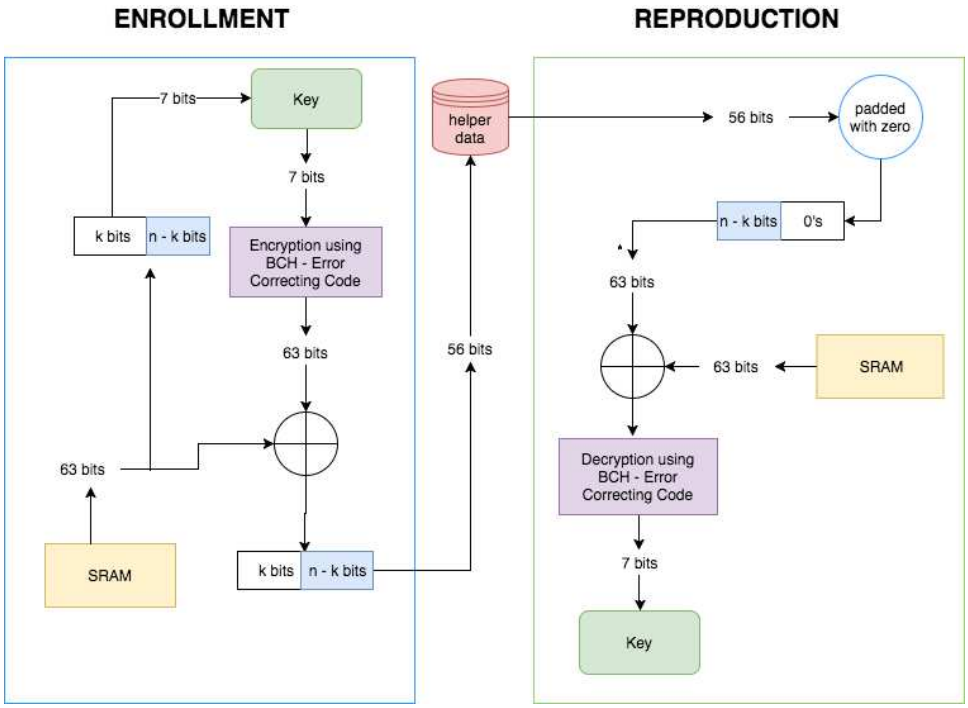

Figure 3.2: Scheme for key generation. $n = 63$, $k = 6$, $t = 15$, $d = 31$.

Chapter 4

Implementation, Experiments and Results

After describing our proposed system as an attempt to achieve this thesis' goals in the previous chapter, this chapter continues with an explanation of several experiment setups and results. This chapter starts by a presentation on two chosen SRAMs that used in experiments; Microchip 23LC1024 and Cypress CY62256NLL.

4.1 Chosen SRAM

The first step to do in this thesis implementation is looking for SRAM. There are numerous SRAM types available in the market. The 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 an entity 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 experiments 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 and also the distribution of 1's and 0's inside SRAM memory.

4.1.1 Microchip 23LC1024

The Microchip Technology Inc. 23A1024/23LC1024 is a 1024 Kbit Serial SRAM device. This SRAM is very popular shown by many references available online and several GitHub repositories intended just to access this

SRAM. The reason of its popularity can be traced to its cheap price, small size, and easy-to-use. The price is ranging from €1.5-3.5. This device has eight pins which contribute significantly to 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 4.1 shows the Microchip 23LC1024.

Figure 4.1: SRAM Microchip 23LC1024.

There are ten Microchip 23LC1024 SRAMs that were available during the experiment. To check whether this SRAM is a justifiable candidate for PUF, several testings 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%.

Second, HD_{intra} and HD_{inter} are calculated on these chips. The calculation is done using twenty data of chip memory values on each chip which retrieved at 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 is done using 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%.

4.1.2 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 that this device has an automatic power-down feature, reducing the power consumption by 99.9 percent when deselected. Unlike Microchip 23LC1024, Cypress CY62256NLL doesn't have an SPI connection which complicates the communication. To communicate, one should

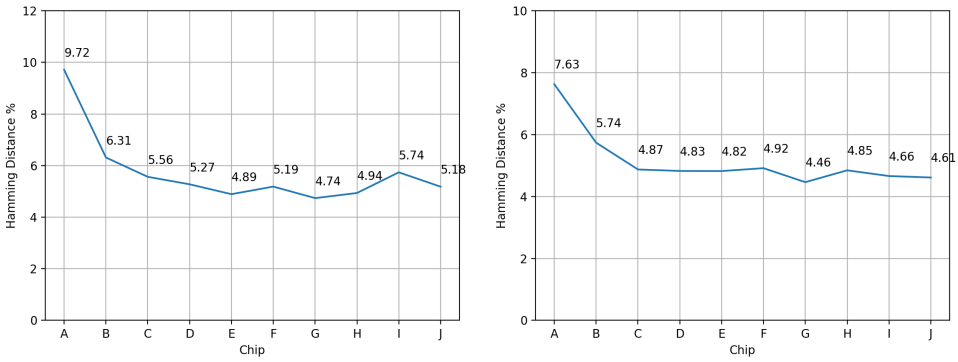

Figure 4.2: HD_{intra} of ten SRAM Microchip 23LC1024. The left is HD_{intra} with constant voltage, the right one is tested based on the voltage variation.

utilize its twenty-eight pins available. Since it has many pins, this contributes to its significantly larger size compared to Microchip 23LC1024. This device is produced using 90nm technology. Its voltage range is ranging from 4.5V-5.5V. Figure 4.3 shows Cypress CY62256NLL.

Figure 4.3: SRAM Cypress CY62256NLL.

There are five Cypress CY62256NLL SRAMs that were available during experiment. Similar like on previous SRAM, several testing are performed to check whether this SRAM is a justifiable candidate for PUF. 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 is done using twenty data of chip memory values on each chip which retrieved at 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 is 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 4.4 shows the HD_{intra} between the constant and the variated voltage.

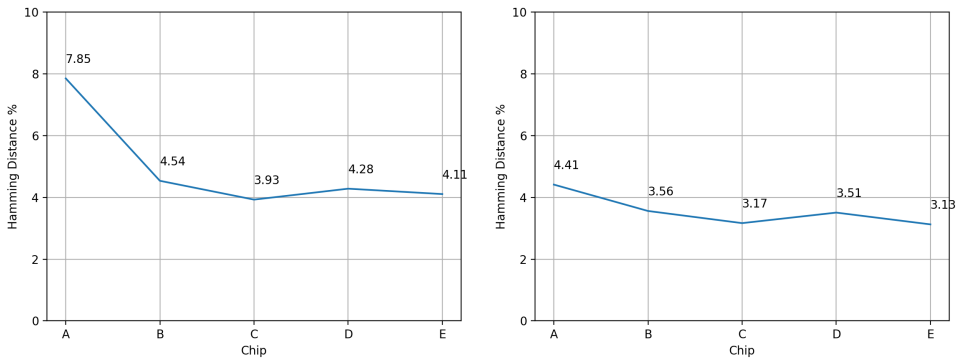

Figure 4.4: HD_{intra} of five SRAM Cypress CY62256NLL. The left is HD_{intra} with constant voltage, the right one is tested based on the voltage variation.

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 is 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.

4.2 Automated PUF Profiling System

To increase the experiment's efficiency, an automated PUF profiling system is constructed. The system consists of a PC, act as a master, and an Arduino connected to an external SRAM which acts 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 is dedicated for voltage control, read bytes, write bytes, and memory disable/enable. The system also supported parallel profiling which significantly increases the effectivity. Figure 4.5 shows the setup and the schematic to profile four SRAMs Cypress CY62256NLL concurrently using four Arduino.

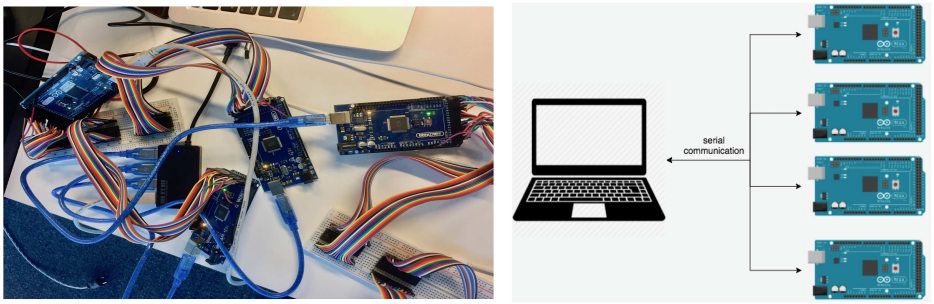

Figure 4.5: Automated PUF profiling setup using a PC and four Arduino. Left picture shows the actual setup, while the right picture displays the schematic of such setup.

4.3 Bit Selection Algorithms Experiments

In this section, the test on stable bits produced by two algorithms, neighbor stability and data remanence analysis, is shown. The test was done on a single chip of each SRAM type. The explanation of both algorithms can be found on section 2.6.3.

4.3.1 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.

Microchip 23LC1024

As input for the algorithm, there are 500 data of SRAM bits value used for this chip. The voltage variation is randomized between 2.5V - 5.0V. The time difference between enrollment is ranging from 5 seconds until 1 hour. SRAM Microchip 23LC1024 itself has capacity 1048576 bits. After doing the calculation from those five hundred data, there are 413374 remaining stable bits. From those remaining stable bits, the rank of each bit is calculated. The frequency of bits rank is shown in Figure 4.6. As shown in this figure, the total bits with rank more than 5 is insignificant, only showing 493 bits. Bits with rank more or equal to six is merged into a single bar because the frequency among those rank is usually only a single digit.

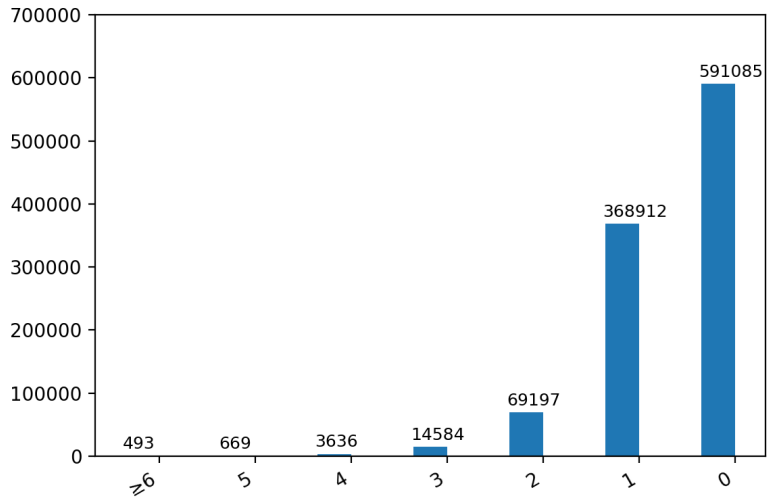

Figure 4.6: Remaining stable bits count according to their rank in SRAM Microchip 23LC1024.

Cypress CY62256NLL

Similar like with SRAM Microchip 23LC1024, there are 500 enrollments done in Cypress CY62256NLL. SRAM Cypress CY62256NLL is able to store 262144 bits in its memory. The remained stable bits after 500 enrollments are 102708 bits (39,18%). The result of the calculation is shown on Figure 4.7. Compared to Microchip 23LC1024, this SRAM shows more promising result since there are many bits with ranks more than seven. Even to get two thousand bits, the lowest rank that can be included is twelve.

4.3.2 Data Remanence Approach

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

Microchip 23LC1024

On SRAM Microchip 23LC1024, the data remanence analysis is done on time variance between 0-1.0 second. The result can be seen on Figure 4.8. In this figure, it is shown that SRAM Microchip 23LC1024 will reach the uninitialized point if it is temporarily turn off for 0.7 seconds.

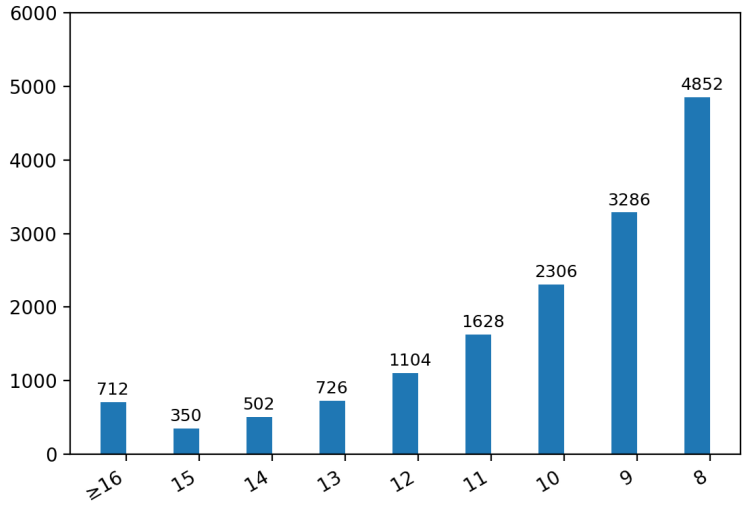

Figure 4.7: Remaining stable bits count according to their rank in SRAM Cypress CY62256NLL.

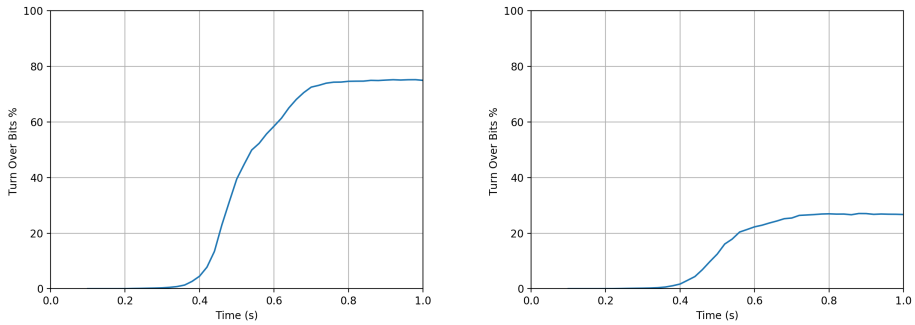

Figure 4.8: Remanence Graph of SRAM Microchip 23LC1024. Left is remanence 0 and right is remanence 1. SRAM Cypress CY62256NLL will reach the uninitialized point if it is temporarily shut down for 0.7 second.

Cypress CY62256NLL

On SRAM Cypress CY62256NLL, the data remanence analysis is done on time variance between 0-1.95 seconds. The result can be seen on Figure 4.8. In this figure, it is shown that SRAM Cypress CY62256NLL will reach the uninitialized point if it is temporarily shut down for 5.0 second.

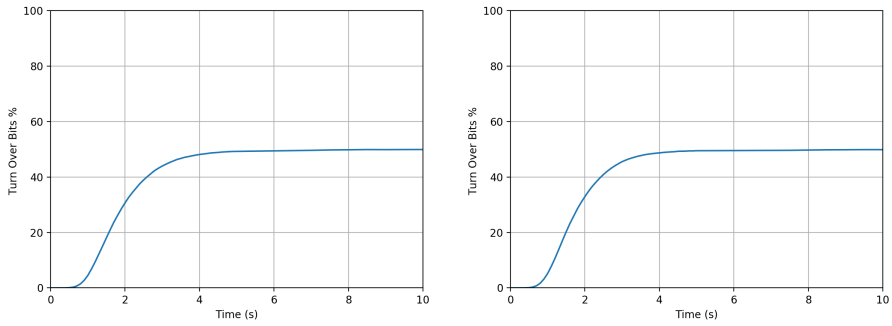

Figure 4.9: Remanence Graph of CY62256NLL. Left is remanence 0 and right is remanence 1. SRAM Cypress CY62256NLL will reach the uninitialized point if it is temporarily shut down for 5.0 second.

4.3.3 Stability Test on Stable Bits

In this section, test results on the effect of time interval and voltage on stable bits using both algorithms on each SRAM are shown. The effect of aging and temperature is not tested due to a 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 is done on 4662 bits which is twice the length of the bits required to generate 256 bits key when using scheme shown in Figure 3.2. The result of time interval testing on SRAM Microchip 23LC1024 is shown on Figure 4.10, while Figure 4.11 displays the result for SRAM Cypress 62256NLL.

Microchip 23LC1024

- Neighbor Stability Analysis
  To get 4662 bits, there are three categories included; rank similar or higher than 6 with 493 bits, rank 5 with 669 bits, and rank 4 with 3500 bits. 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 max-

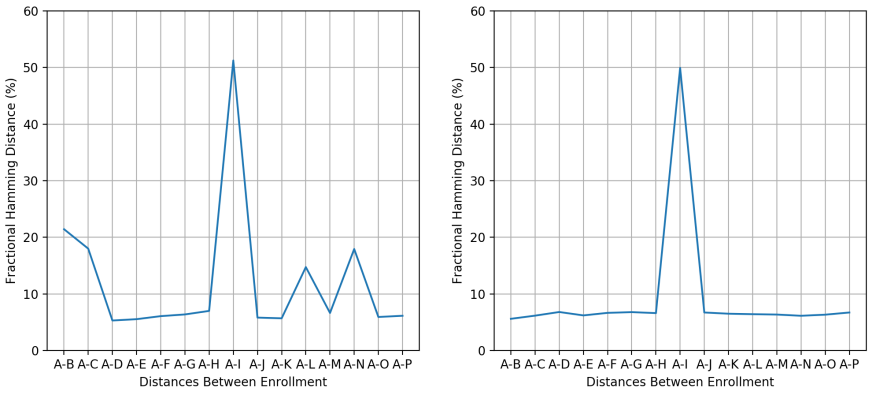

Figure 4.10: Time interval testing results on SRAM Microchip 23LC1024. Left figure is the testing result on stable bits generated using neighbor analysis, while the right one is tested on data remanence generated stable bits. Index A on x-axis refers to enrollment on day 1, B on day 2, etc. Index A-B refers to fractional hamming distance between enrollment on day 1 and day 2.

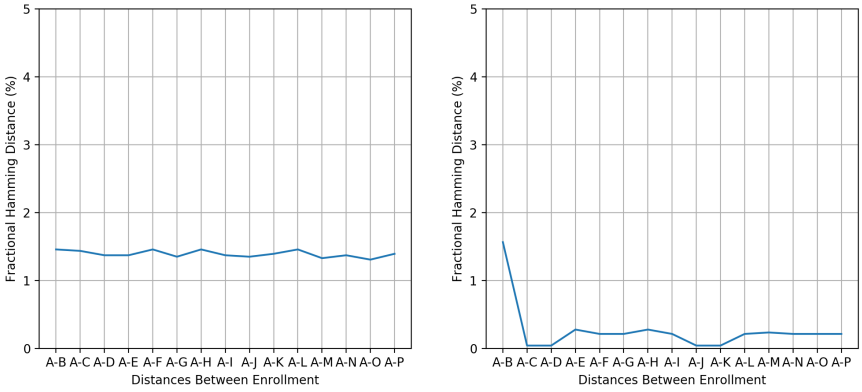

Figure 4.11: Time interval testing results on SRAM Cypress CY62256NLL. Left figure is the testing result on stable bits generated using neighbor analysis, while the right one is tested on data remanence generated stable bits. Index A on x-axis refers to enrollment on day 1, B on day 2, etc. Index A-B refers to fractional hamming distance between enrollment on day 1 and day 2.

imum difference is produced when the difference between enrollment is 8 days.

- Data Remanence Approach
  To get 4662 bits, strong 1's are generated using power down period of 0.185 seconds, while strong 0's are calculated when 0.27 seconds are used as the power down period. The difference between power down period during generation of strong 1's and strong 0's is because the number of 1's that flipped fast are more compared to 0's. This is also related to the 0's and 1's distribution during normal initialization (0's count for 30% and 1's filled 70%). Similar to the previous algorithm, the stability of bits produced by using this algorithm is also not good. The worst change happens when 8 days is used as the time interval between testing, showing as many as 2328 bits (49.93%).

Cypress CY62256NLL

- Neighbor Stability Analysis
  To get 4662 bits, there are six categories included; rank similar or higher than 16 with 712 bits, rank 15 with 350 bits, rank 14 with 502 bits, 726 bits of rank 13, 1104 bits of rank 12, and 1268 bits of rank 11. 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 the time interval between enrollment is a day.

- Data Remanence Approach
  Unlike SRAM 23LC1024, power down period when enrolling strong 1's and 0's on CY62256NLL is not different. To get 4662 stable bits, both are enrolled using power down period 0.28 seconds. During the voltage and time interval variation, the stable bits produced by using algorithm also shows a promising result. It only accounts 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 is also proven to be a better algorithm than neighbor analysis. Due to these conclusions, further testing shown below are only done on SRAM Cypress CY62256NLL and the stable bits used are generated using the data remanence algorithm.

4.4 Testing on A Set of Bit Locations as A Challenge

In this section, the testing results on our proposed PUF challenge is presented. As mentioned in the previous chapter, a set of bit locations is selected as the PUF challenge in our application. The test was done on SRAM Cypress CY62256NLL. Cypress CY62256NLL itself has a capacity to store 262144 bits. The number of bits uses a challenge is 2331 bits (the length of the bits required to generate 256 bits key when using scheme shown in Figure 3.2). Using Equation 3.1 explained in previous chapter, there are $P(262144, 2331) = \frac{262144!}{(262144-2331)!} \approx 10^{12626}$ possible combinations. Using this large number of possibilities, once again we emphasize that this construction can be a proper strong PUF construction using SRAM PUF.

The selected test to test this challenge is by calculating the HD_{inter} among five SRAMs. Figure 4.12 shows the result of this experiment. As shown in that figure, the HD_{inter} is ranging between 35.26% until 46.93%, with average 42.08%. This result shows that the difference between each SRAM when using a set of location as a challenge is sufficient to distinguish an SRAM from another.

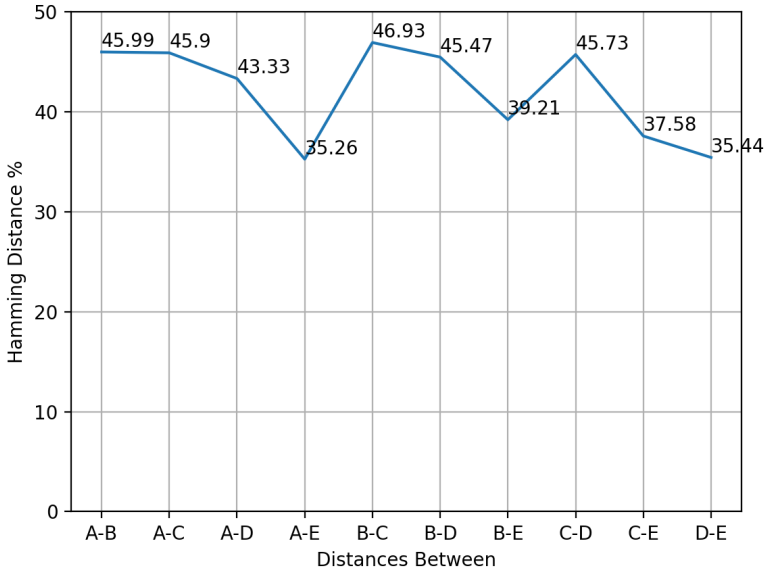

Figure 4.12: HD_{inter} among five SRAMs Cypress CY62256NLL.

4.5 Complete Enrollment Scheme

Based on the experiment results shown before, we construct a complete enrollment scheme. The enrollment scheme has a goal to create challenge and helper data which will be used in our proposed secure data protection and key storage scheme (further explanation is available on Section 3.5). Our complete enrollment scheme is shown in Figure 4.13. The selected algorithm for stable bit locator is data remanence analysis due to its better result and shorter time needed compared to neighbor analysis. We also present Figure 4.14 to show how to connect an Arduino Mega 2560, an SRAM Cypress CY62256NLL and a microSD.

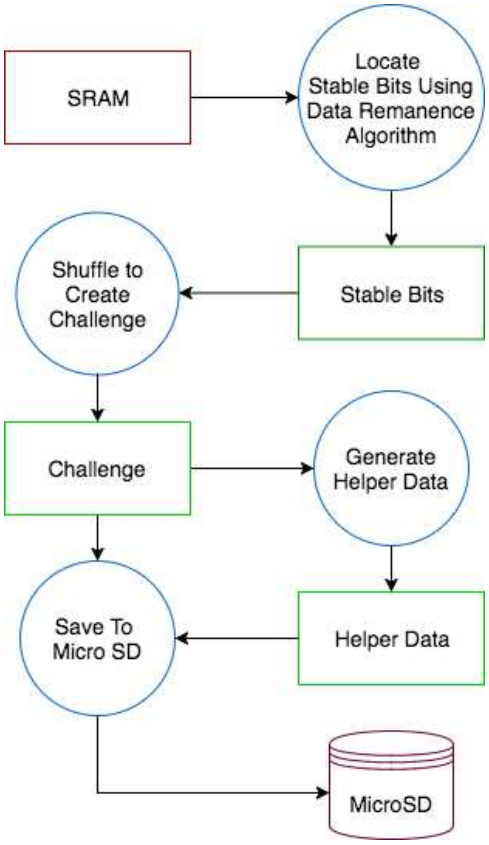

Figure 4.13: Complete enrollment setup.

4.6 Secure Data Protection and Key Storage Testing

To check the validity of the proposed data protection and key storage scheme, several testings are performed. First, the final key generated using HMAC

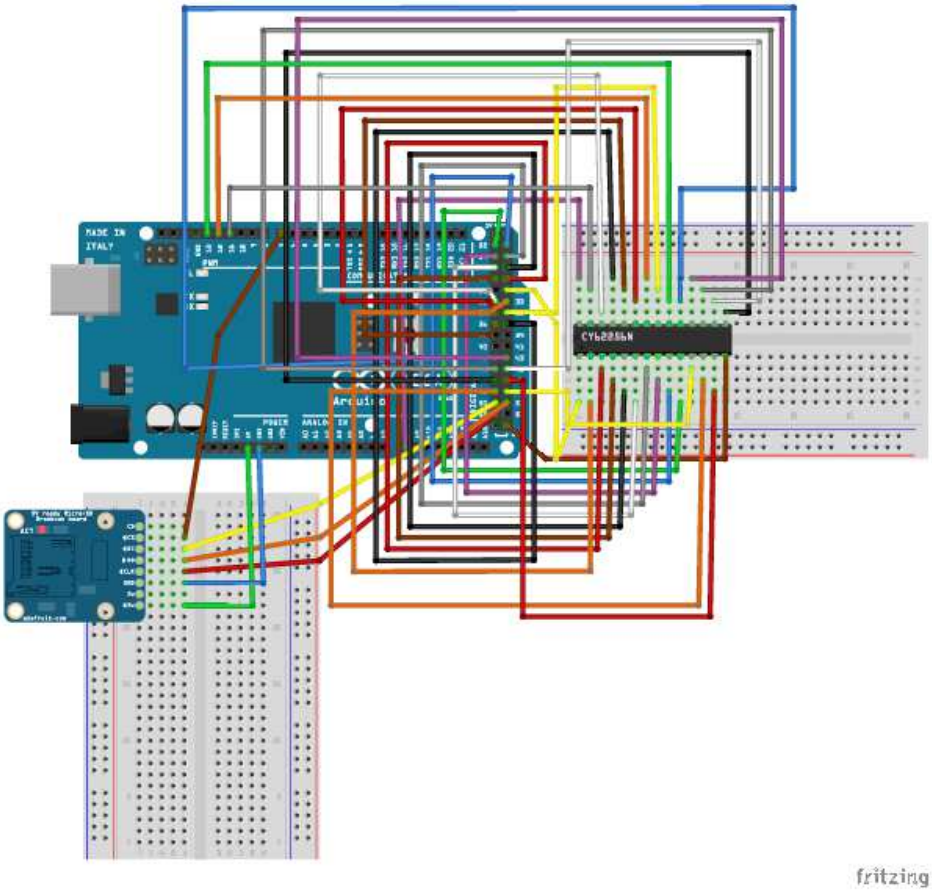

Figure 4.14: An illustration on how to connect an Arduino Mega 2560 with an SRAM Cypress CY62256NLL and a micro SD.

SHA3 is checked. Afterwards, the result of the encryption and decryption using the final key is also tested.

To check the validity of the final key, a comparison between the generated final key and an online HMAC SHA3 calculator [48] is performed. As a reminder, the key for HMAC function is the PUF generated key and the message input for the HMAC is the user's password. Below is the testing result:

- PUF generated key: d20f5656bf436516cd0f3d2e734851dc537df518 97484128ccae67ee1310f69b
  - user's password: 70617373776f7264
    final key: 084536fcb3135af89e1e32d423156511f13e52246acaa 591b1d4115666727814
    valid: yes
  - user's password: 6b6f6e746f6c6b61626568
    final key: 1d1e467224d72c81ede61fcd5d1ac10535f3ebafa6e9f 0d5086e6086a30787c7
    valid: yes
  - user's password: 71776572747975696f706173646667686a6b6c 7a786376626e6d
    final key: dda21605fc56b55659cffdf57f5453a9e380aa7bd78fe5 2b7dc64ff4515ff4a0
    valid: yes
- PUF generated key: 35e2f312bd28a36a359eb1a1e37f212d17da41a5 b17cb2c642f5fd8e42bbd4f0
  - user's password: 70617373776f7264
    final key: c2892f1b1d52d59549591d410a40527b265b91d444d 2032f28ce7374f7246152
    valid: yes
  - user's password: 6b6f6e746f6c6b61626568
    final key: ec85915ae65f3e5141128a520327c4d5cd3119cb6769f ddd948d3061dfb6fed9
    valid: yes
  - user's password: 71776572747975696f706173646667686a6b6c 7a786376626e6d
    final key: da9e28f76754dcd4946c1343a3dd8550338d98e46d3a 11e09f903204044ac9c7
    valid: yes

Bitcoin PUF transfer

After checking the validity of the final key, a testing on encryption and decryption using the final key is performed. The input data on this testing is a bitcoin key (referred as user's key). First, an enrollment is performed on an Arduino board with an SRAM Cypress CY62256NLL and a microSD connected to it, resulting in challenge and helper data which store in the microSD. Using the produced challenge and helper data, a user creates a final key which derived from the PUF-generated key and user's password. The final key is utilized to encrypt a bitcoin key, then the ciphertext is stored in microSD. To check the validity of the ciphertext, an online encryption calculator [49] is utilized. Afterwards, the voltage input to Arduino is turned off. Later, the microSD and the SRAM are transferred to another Arduino board. The new Arduino board is powered on, then it is used to reconstruct the final key. Finally, the reconstructed final key is applied to the ciphertext which is loaded from the microSD. The experiment is considered successful if the result of decryption is the same as the bitcoin key. This experiment result is shown below:

- bitcoin key: 6a656d6275746a656d62756a656d6275
  - final key: 084536fcb3135af89e1e32d423156511f13e52246acaa591b1d4115666727814
    ciphertext: 874cd8b8010f29f0a52eb564f1306119
    ciphertext validity: yes
    decryption result: 6a656d6275746a656d62756a656d6275
    decryption validity: yes
  - final key: 1d1e467224d72c81ede61fcd5d1ac10535f3ebafa6e9f0d5086e6086a30787c7
    ciphertext: c258a267ea58d04e0795e661000b9e4f
    ciphertext validity: yes
    decryption result: 6a656d6275746a656d62756a656d6275
    decryption validity: yes
  - final key: dda21605fc56b55659cffdf57f5453a9e380aa7bd78fe52b7dc64ff4515ff4a0
    ciphertext: 64586f93c32e0498a46e906dc85b726a
    ciphertext validity: yes
    decryption result: 6a656d6275746a656d62756a656d6275
    decryption validity: yes
  - final key: c2892f1b1d52d59549591d410a40527b265b91d444d2032f28ce7374f7246152
    ciphertext: f7a0c1bfdbe1d12af85ca5c3933bc29d
    ciphertext validity: yes

decryption result: 6a656d6275746a656d62756a656d6275
decryption validity: yes

- final key: ec85915ae65f3e5141128a520327c4d5cd3119cb6769f ddd948d3061dfb6fed9
  ciphertext: b91b1d3031b3da73a6c1afe516d85b8b
  ciphertext validity: yes
  decryption result: 6a656d6275746a656d62756a656d6275
  decryption validity: yes
- final key: da9e28f76754dcd4946c1343a3dd8550338d98e46d3a 11e09f903204044ac9c7
  ciphertext: 1f78480b2687b7b7faba210985600372
  ciphertext validity: yes
  decryption result: 6a656d6275746a656d62756a656d6275
  decryption validity: yes

Measurement of time required in this scheme is also done. During the measurement, the scheme is divided into five stages. The first stage is on the initialization. The second stage is when the challenge and the helper data are loaded from micro SD. The third one is calculated when reconstructing the PUF key. Next stage is during the derivation of the final key (derived from user's password and PUF-generated key). The fifth stage and sixth stage refers to the encryption and decryption processes. The measurement result can be seen on Table 4.1. It can be seen that the longest time required is when loading the challenge and the helper data from the microSD (stage 2), followed by the initialization stage (stage 1). Due to this significant time required, a further optimization on accessing data from microSD is suggested.

Table 4.1: Time measurement of the secure data protection and key storage scheme in µs.

No	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 4**	**Stage 5**	**Stage 6**
1	1022656	2205248	978152	33572	836	1716
2	1022652	2205244	974388	33568	848	1712
3	1022632	2205248	981272	33572	836	1716
Average	1022647	2205247	977937	33571	840	1715

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis starts by showing the potential of using SRAM PUF as a secure way to protect our key and data. Embraced with a bright prospect, it is unfortunate that the development of PUF in the real world seems to lack of public involvement. The currently available solution is usually locked to specific entities, such as companies or universities. There is no open source project available for tech enthusiast to embrace this amazing technology. Here, we initiate an open source project to develop SRAM PUF technology using off-the-shelf SRAM. We also present an automated enrollment system. The system, inspired by the concept of master-slave, consists of an Arduino source code (act as a slave) and a python source code, perform as a master, run in PC. Using this system, two types of SRAM are tested; Microchip 23LC1024 and Cypress CY62256NLL. Both are tested on the distribution of 0's and 1's in their cells, intra hamming distance, inter hamming distance and the effect of voltage and time interval between enrollment. We provide an enrollment mechanism to look for stable bits which will be used as the basis for PUF application. There are two algorithms for looking stable bits tested here; neighbors analysis and data remanence approach. Based on the experiment results, in our final enrollment scheme, we present a concrete SRAM PUF enrollment using an Arduino, an SRAM Cypress CY62256NLL and a microSD with data remanence approach as the bit selection algorithm. A microSD is required to store the helper data and the PUF challenge.

In addition, an idea to create a strong PUF using SRAM is also proposed here. Using a collection of bits as a challenge, the stable bits are permutated among themselves to create a challenge which has a tremendous number of possibilities.

Furthermore, we also present a secure data protection and key storage

scheme using SRAM PUF. The proposed scheme is influenced by multi-factor authentication. Using a combination of a PUF-generated key and user's password, a derived key is produced and utilized as the final key to protect user's data or/and user's key. Unlike the automated enrollment system, this scheme only consists of an Arduino source code. Evaluation on this scheme time and code size are also presented.

5.2 Future Work

This thesis only presents an idea to secure user's data using symmetric encryption. To see similar application but using asymmetric encryption concept, one should look further to the thesis done by Akhundov [50]. He presents a public key infrastructure (PKI) concept using the PUF-generated key as the root of trust. A possible integration between our work and his work is combining our 'final' key into his construction as a root of trust.

In addition, our idea of using user's password and the PUF-generated key is not the highest level of security in multi-factor authentication. As mentioned in Section 2.4, the most secure multi-factor authentication can be achieved when all three factors are combined together; knowledge, possession, and inherence. Since there are only two factors utilize (knowledge and possession) in this thesis' proposed secure data protection and key storage scheme, an addition of inherence factor when generating the final key can increase the security level. As mentioned in Section 1.2, biometric-based authentication and PUF are utilized to secure self-sovereign identity in Pouwelse and de Vos's proposed technology stack during trust creation in blockchain era. A further read on their article mentioned that there is a working prototype of fingerprint authentication using a smartphone camera. Since that project and our work share the same principle, open source and open ecosystem, we suggest integrating this fingerprint authentication into our proposed scheme to enable an even higher level of security.

In this thesis, the SRAM testing is only done on the effect of time interval between enrollment and voltage variation. We believe another testing on temperature and aging is required to ensure whether SRAM Cypress CY62256NLL is a good candidate for SRAM PUF. The capability to test on temperature and the aging effect is suggested to be included as an addition of our automated enrollment system. In addition, we also encouraged others to test other types of SRAMs to enrich the knowledge of possible off-the-shelf SRAM as a PUF candidate.

Bibliography

[1] World economic forum - a blueprint for digital identity. http://www3.weforum.org/docs/WEF_A_Blueprint_for_Digital_Identity.pdf. [Online; accessed 16-March-2018].

[2] G R Blakley and David Chaum, editors. *Proceedings of CRYPTO 84 on Advances in Cryptology*, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[3] What is "sovereign source authority"? — the moxy tongue. http://http://www.moxytongue.com/2012/02/what-is-sovereign-source-authority.html. [Online; accessed 19-February-2018].

[4] 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.

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

[6] Data privacy vs. data protection. https://blog.ipswitch.com/data-privacy-vs-data-protection. [Online; accessed 16-March-2018].

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

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

[9] 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.

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

[11] 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.

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

[13] Google scholar. `https://scholar.google.nl/scholar?as_vis=1&q=srampuf&hl=en&as_sdt=1,5`. [Online; accessed 06-March-2018].

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

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

[16] Pim Tuyls and Boris Škorić. *Strong Authentication with Physical Unclonable Functions*, pages 133–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[17] Jeroen Delvaux, Roel Peeters, Dawu Gu, and Ingrid Verbauwhede. A survey on lightweight entity authentication with strong pufs. *ACM Comput. Surv.*, 48(2):26:1–26:42, October 2015.

[18] 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, New York, NY, USA, 2007. ACM.

[19] Keith B. Frikken, Marina Blanton, and Mikhail J. Atallah. Robust authentication using physically unclonable functions. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio A. Ardagna, editors, *Information Security*, pages 262–277, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[20] Heike Busch, Stefan Katzenbeisser, and Paul Baecher. Puf-based authentication protocols – revisited. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications*, pages 296–308, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[21] Amanda C. Davi Resende, Karina Mochetti, and Diego F. Aranha. Puf-based mutual multifactor entity and transaction authentication for secure banking. In Tim Güneysu, Gregor Leander, and Amir Moradi, editors, *Lightweight Cryptography for Security and Privacy*, pages 77–96, Cham, 2016. Springer International Publishing.

[22] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. Pufky: A fully functional puf-based cryptographic key generator. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 302–319, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

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

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

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

[26] Hugo Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 631–648, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[27] Chiara Galdi, Michele Nappi, Jean-Luc Dugelay, and Yong Yu. Exploring new authentication protocols for sensitive data protection on smartphones. *IEEE Communications Magazine*, 56:136–142, 2018.

[28] Sudheendra Srivathsa, Sudheendra K. Srivathsa, and Wayne P. Burleson. Secure and energy efficient physical unclonable functions. 2017.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] Hmac generator online hash encryption. `https://www.liavaag.org/English/SHA-Generator/HMAC/`. [Online; accessed 26-March-2018].

[49] Cryptomathic - aes calculator. `http://extranet.cryptomathic.com/aescalc/index`. [Online; accessed 27-March-2018].

[50] Haji Akhundov. Design & development of public-key based authentication architecture for iot devices using puf. MSc thesis, Delft University of Technology, 2017.