DESIGN AND IMPLEMENTATION OF MEMORY PHYSICALLY UNCLONABLE FUNCTIONS ON LOW-POWER DEVICES

By Manuel Aguilar Rios

A Dissertation

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

in Informatics and Computing

Northern Arizona University

August 2023

Approved:

Bertrand F. Cambou, Ph.D., Chair

Viacheslav Y. Fofanov, Ph.D.

Fatemeh Afghah, Ph.D.

Tuy Nguyen, Ph.D.

ABSTRACT

DESIGN AND IMPLEMENTATION OF MEMORY PHYSICALLY UNCLONABLE FUNCTIONS ON LOW-POWER DEVICES

MANUEL AGUILAR RIOS

Cryptography plays a vital role in safeguarding sensitive information from falling into the wrong hands. It relies on symmetrical encryption methods like Advanced Encryption Standard (AES) and asymmetrical encryption methods such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC) for authentication between parties. However, the management of cryptographic keys introduces vulnerabilities that hackers can exploit. Moreover, it is anticipated that the security of asymmetric encryption methods like RSA and ECC can be compromised in the future by quantum computers with the capability to handle enough qubits.

A key primitive in cryptography is the random number generator (RNG). Randomness is required in various cryptographic protocols, such as randomizing encryption operations or generating random challenges in authentication. Standard libraries often provide Pseudo-Random Number Generators (PRNGs), which are deterministic and vulnerable to attacks. Cryptographically-Secure Pseudo-Random Number Generators (CSPRNGs) produce highly unpredictable sequences, but they are complex and resource-intensive, making them impractical for low-power devices.

Physically Unclonable Functions (PUFs) present a potential solution for addressing challenges in authentication, key management, and random number generation. PUFs offer unique digital fingerprints that are difficult to replicate, making them well-suited for generating one-time keys. Moreover, PUFs can be used to design True Random Number Generators (TRNGs), producing true random sequences with low power consumption and computational resources. By utilizing PUFs for these purposes, cryptographic systems can enhance their security by generating secure keys and the availability of highly unpredictable random numbers.

There are various different PUFs, and one widely used PUF is the static random-access memory (SRAM) PUF. However, there are also promising emerging memory technologies like resistive random-access memory (ReRAM) and magnetoresistive random-access memory (MRAM) PUFs. These innovative PUFs show great potential in enhancing the overall security of cryptographic systems. By leveraging these advanced PUF technologies, cryptographic systems can further bolster their security measures.

The primary aim of this research is to implement and examine MRAM, SRAM, and ReRAM PUF devices on low-power client devices and incorporate them into enhanced cryptographic architectures. The study demonstrates that SRAM and MRAM devices suit a unique ternary-addressable physically unclonable function (TAPUF) design. This design not only generates dependable cryptographic keys but also produces true random sequences that successfully pass the National Institute of Technology (NIST) Statistical Testing Suite for random numbers. Moreover, it also shows that the analog responses of pre-formed ReRAM cells can encapsulate a message, even when applied on a low-power microcontroller.

Copyright

Acknowledgements

In Memoriam

With a heavy heart and cherished memories, I dedicate this section to our beloved Chiques, a poodle mix who enriched my family's lives from 2003 to 2021. Chiques was so much more than just a pet; she was an inseparable part of our family. From my elementary school graduation to college graduation, she stood by our side, offering love, comfort, and unwavering loyalty.

As I embarked on my graduate journey, Chiques became a source of inspiration and strength. It deeply saddens me that she couldn't be here physically to celebrate this significant milestone with us. But I find solace in believing she watches over me, beaming with pride from across the rainbow bridge.

Though Chiques is no longer with us, her spirit lives on in our hearts, forever guiding and comforting us. We will forever cherish the beautiful moments we shared, the wag of her tail, the warmth of her presence, and the love she showered upon us.

Rest peacefully, Chiques, and thank you for being the extraordinary friend and family member you were. Your memory will forever brighten our lives.

Funding

I want to thank Dr. Bertrand Cambou and Northern Arizona University for granting me a research assistance position for the three years I have spent on my graduate journey from 2020 to 2023. Likewise, I would also like to thank the Air Force for allowing me to take three years away from my duties as an officer to pursue this degree.

General Acknowledgements

I want to express my heartfelt gratitude to the professors and mentors at Northern Arizona University, whose unwavering support and guidance have been instrumental in shaping my academic journey.

First and foremost, I extend my sincere appreciation to my esteemed advisor, **Dr. Bertrand Cambou**.

It was under his mentorship that I embarked on the challenging path of pursuing a doctoral degree. His vision and innovative ideas, reflected throughout this document, have been truly inspiring. I am also deeply grateful for his dedication to securing funding for the lab and ensuring the well-being of his students.

I also want to acknowledge the invaluable contributions of **Julie Heynessen**, a remarkable professor I had the privilege of working with during my electrical engineering undergraduate studies and later as a respected colleague. Her continued guidance and support during this academic endeavor have been immensely significant.

My heartfelt thanks go to the members of my Comprehensive, Candidacy, and Dissertation committees, **Dr. Fatemeh Afghah**, **Dr. Ying-Chen Chen**, **Dr. Truong Nghiem**, and **Dr. Viacheslav Fofanov**. Their valuable feedback and insights proved critical in achieving important milestones in my research.

I am deeply grateful to my lab companions and friends, including **Dina Ghanaimiandoab**, **Mahafujul Alam**, **Michael Partridge**, **Jack Garrad**, **Britt Riggs**, **Saloni Jain**, **Ashwija Korenda**, **Taylor Begay**, **Christopher Philabaum**, and **Ruben Montano**. Working with each of you on significant projects and building meaningful friendships enriched my journey. Despite the challenges posed by the COVID pandemic during my lab transition, I cherish our friendship, and I will miss you all when I leave the lab.

Finally, my heartfelt appreciation goes to my cherished friends and family in Phoenix. To my beloved mother, **Aracely Rios**; my father, **Luis Aguilar**; my supportive big sister, **Rosa Aguilar**; and my caring little sister, **Aracely Aguilar**. Your love and encouragement have been my pillar of strength throughout this pursuit. I am also grateful to my Phoenix friends, **David Galas** and **Cristian Rico**, whose unwavering friendship, despite the distance, made a significant impact on my life.

To all those who have supported me along this arduous journey, your contributions may seem small, but they have made an immeasurable difference in my life. From the depths of my heart, thank you for being an integral part of my academic and personal growth.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography is the study of methods that secure the transfer of information to protect it from falling into the hands of unwanted recipients. Modern cryptographic methods employ a combination of symmetrical and asymmetrical key encryption techniques to authenticate communication between parties. The most popular encryption methods are Advanced Encryption Standard (AES), Elliptic Curve Cryptography, and Rivest-Shamir-Adleman (RSA). AES is a symmetrical key encryption method that uses the same cryptographic key to encrypt and decrypt. AES is a symmetrical key encryption method, meaning the same key is used for encryption and decryption. On the other hand, Elliptic Curve Cryptography and RSA are asymmetrical key encryption methods that rely on a pair of keys: a public key for encryption and a private key for decryption [10].

Symmetrical key encryption is used to encrypt and decrypt large amounts of information efficiently; the key size correlates to the amount of security the encryption provides, with the standard key sizes for AES being 128, 192, and 256 bits. However, a massive obstacle is getting both parties the same key to use. On the other hand, asymmetrical key encryption uses mathematically related public and private key pairs and requires longer keys for similar levels of security [11]. This makes asymmetrical encryption slower and more resource extensive than symmetrical encryption. As a result, symmetric and asymmetric key encryption is often used in a hybrid encryption system; asymmetrical encryption is used to send over symmetrical encryption keys so both parties can use symmetric encryption [12].

AES was standardized in 2001 by the National Institute of Technology (NIST), and it is considered a secure form of symmetrical encryption. Ronald Rivest, Adi Shamir, and Len Adleman introduced the asymmetrical encryption algorithm, RSA, in 1977. Since its invention, it has been regarded as secure and easy to use. Another asymmetrical key encryption method, Elliptic Curve Cryptography, was introduced in 1985 by Victor Miller and Neal Koblitz, and it is one of the widely accepted methods of exchanging secret keys. Elliptic Curve Cryptography's main advantage over RSA is that it requires much less computing power to achieve a similar level of security.

1.1 Motivation

AES, RSA, and Elliptic Curve Cryptography are still widely used. However, each of these cryptographic systems has vulnerabilities. AES has shown weaknesses against timing and side-channel attacks while RSA has shown weaknesses against timing and brute force attacks [13]. Similarly, Elliptic Curve Cryptography can be targeted with isomorphism attacks, which aim to reduce the discrete logarithm problem [14].

The future emergence of quantum computing poses a significant threat to asymmetric cryptographic protocols. RSA and Elliptic Curve Cryptography, in particular, can be easily solved using one of Shor's algorithms when executed on a sufficiently powerful quantum computer [15, 16]. This raises concerns about the long-term security of these algorithms in the face of quantum computing advancements.

A solution to the quantum computing threat is Post-Quantum Cryptography (PQC). PQC are alternative cryptographic algorithms that are secure against classical and quantum computing attacks. Adopting PQC is crucial to ensure the long-term security of sensitive information as quantum computers continue to advance in their capabilities. These protocols have yet to be standardized but are soon set to replace the current cryptographic protocols standardized by NIST.

However, even with the adoption of PQC, there is a vulnerability that hackers can exploit when it comes to managing cryptographic keys. Key management involves crucial aspects such as key generation, distribution, storage, backup, rotation, access control, and destruction. Effective key management plays a vital role in the overall security of cryptographic systems. However, the complexity of key management increases significantly in power-constrained systems. Even when employing a combination of symmetric and asymmetric encryption techniques, securely storing private keys remains a concern, as they can still be susceptible to extraction vulnerabilities. Previous research has highlighted the risks associated with tampering and non-invasive attacks that can compromise the security of private keys [17, 18].

An emerging communication network that is most at risk from key management vulnerabilities is the Internet of Things (IoT).IoT is an emerging communication network that aims to connect devices, including low-power devices. Low-power devices are often vulnerable in these networks, as they are limited in both power consumption and computational power, making them the weakest link in the system. They are particularly vulnerable in key generation, distribution, and storage.

A potential solution for addressing authentication challenges can be the Physically Unclonable Function (PUF). PUFs are physical devices with a unique signature that can be used for cryptographic protocols, such as key generation. PUFs obtain intrinsic randomness from their manufacturing process, rendering them impervious to external influences or manipulations. Their cryptographic keys do not rely on computational hardness, making them quantum computing resistant. Moreover, their unique signature can be extracted in environments with limited resources, making them suitable for deployment in low-power devices. This feature enables the integration of PUF-based key generation in scenarios where traditional key storage methods may be impractical or challenging to implement.

The intrinsic randomness of physical devices for cybersecurity has been studied since the 1980s [19]. However, the implementation and strength of commercial PUFs remain poor. The most popular and well-studied PUF is the Static random-access memory (SRAM) PUF, proposed in 2007 [3]. The SRAM PUF has gained popularity due to it being widely available on existing technology; however, SRAM PUFs have glaring weaknesses that can be exposed.

However, it's important to acknowledge that PUFs may not always be the ideal solution due to certain considerations. Firstly, PUFs can be expensive and slow compared to alternative approaches, potentially introducing additional costs and impacting system performance. Additionally, PUFs, especially certain types, may exhibit erratic behavior due to variations in physical properties, making their integration and management more challenging. Furthermore, if a PUF's behavior is predictable or lacks sufficient tamper resistance, it can introduce vulnerabilities to the system. Adversaries may exploit these weaknesses to compromise the security of cryptographic operations. Therefore, it is crucial to thoroughly evaluate the specific requirements and constraints of a system before deciding whether to employ PUFs.

Several PUFs have been proposed, with emerging memory technologies like Magnetoresistive random-access memory (MRAM) and Resistive random-access memory (ReRAM) showing great promise. MRAM PUFs offer several advantages, including low power consumption, non-volatility, excellent write and read endurance, and suitability for space operations. On the other hand, ReRAM PUFs are characterized by low power consumption, non-volatility, and the capability of self-destruction.

Both ReRAM and MRAM PUFs can employ analog responses, which allows for an increased number of Challenge-response pair (CRP)s. To evaluate their practical applicability, we are implementing these PUFs on low-power devices to simulate real-world use cases. It is worth noting that most of the existing studies on MRAM and ReRAM PUFs have relied on simulated results or have been conducted at the wafer level rather than at the packaged Integrated Circuit (IC) level.

This research specifically focuses on the implementation of MRAM and ReRAM PUFs with ICs combining theoretical analysis and practical deployment. The main objective is to gain insights into the feasibility and effectiveness of these technologies in real-world scenarios.

In this work, we define low-power devices as those that can be powered using 5V of electricity with a current of 500 milliamps or less, resulting in an overall power output of 2.5 Watts. This power requirement aligns with the standard power supplied by a computer USB port, providing a practical reference for the considered low-power devices.

This research aims to contribute to the understanding and advancement of memory technologies for cryptographic applications. It will focus on the research done on SRAM, MRAM, and ReRAM PUFs. This work introduces a novel SRAM PUF implementation with two IC SRAM devices that can be used for cryptographic key and true random number generation. Similarly, it introduces a novel MRAM PUF implementation that uses analog responses to produce binary bits, which can also be used for cryptographic key and true random number generation. Additionally, it will discuss the analog key encapsulation mechanism with the analog responses of preformed ReRAM, which is an alternative method of encapsulating information.

Chapter 2

Background

This chapter aims to provide background information to readers interested in some of the details related to Physically Unclonable Function (PUF)s and True Random Number Generator (TRNG)s. It covers several key topics, including cryptographic primitives, PUFs, and cryptographic protocols that employ PUFs. By exploring these concepts, readers will gain the necessary knowledge to delve into the research effectively.

2.1 Cryptographic primitives

Cryptographic primitives are low-level algorithms used as fundamental building blocks for higher-level cryptographic algorithms [20]. They are classified into three main categories: Unkeyed primitives, symmetric key primitives, and public-key primitives. Unkeyed primitives are functions that do not require a key and are entirely public. Examples of unkeyed primitives are hash functions, one-way functions, and random number generators (RNGs). These functions generally have outputs with no known correlation to future outputs and whose inputs are incredibly difficult to find. Symmetric-key primitives use a single key for encryption and decryption, providing security and authentication for both parties, e.g., symmetric encryption, Message Authentication Codes (MACs). Public-key primitives, also known as asymmetric primitives, are primitives that require the use of a pair of keys, public and private, e.g., asymmetric encryption [21, 22].

2.2 Random Number Generators

An important cryptographic primitive is a Random Number Generator (RNG). Most cryptographic systems use secure RNGs to enhance security. In many of these cryptographic systems, the quality of the random numbers is essential to the strength of the overall system. As a result, quality random number generators are critical in cryptography [23, 24, 25].

2.2.1 Pseudo-Random Number Generator (PRNG)

Most RNGs widely available in standard libraries are PRNG based on a predefined mathematical algorithm, making them deterministic. PRNGs can be vulnerable to several attack vectors, making them weak for most cryptographic applications. They are mainly used for non-cryptographic applications such as sampling because they are fast and low-cost. Some requirements for PRNGs include being statistically well distributed and avoiding repeating patterns [26].

Cryptographically-Secure Pseudo-Random Number Generator (CSPRNG)s are usually safe in cryptographic applications as they produce highly unpredictable number sequences resistant to computational attacks and indistinguishable from truly random numbers. However, CSPRNGs are far more complex and resource-extensive than regular PRNGs making them impractical on low-power devices [26, 27]. Additionally, since they rely on mathematical algorithms to generate sequences, the bits are also deterministic.

2.2.2 True Random Number Generator (TRNG)

TRNGs generate randomness from a natural phenomenon considered "truly" random. "True" randomness comes from processes that cannot be controlled, such as thermal and electrical noise, making them non-deterministic. These random physical variations make TRNGs hard to predict, making them suitable for secure cryptographic schemes [28, 24]. Since their randomness comes from physical true random phenomena, TRNGs often have biases that reduce their entropy. As a result, TRNGs will often have a post-processing stage to remove possible biases and correlations.

There are multiple methods of quantifying the statistical quality of random numbers. The most common method is the openly available National Institute of Technology (NIST) Statistical Testing Suite [29], which will run a random sequence through several statistical tests. Other tests include the DIEHARDER and Testu01 Random Number Test Suites, which are also openly available [28].

2.3 Physically Unclonable Functions (PUFs)

PUFs are physical devices behaving as probabilistic unique one-way functions by mapping challenges to responses. Their inherent randomness makes them unique from one another; however, they need to generate relatively consistent Challenge-response pair (CRP)s [22]. It can be seen as a *"black box"* as we can often only observe the device by its inputs and outputs. The term PUF was coined in [30]; however, using a physical component of inherent randomness to bolster security dates back to the 1980s [19]. The basic architecture of a PUF is found in Fig. 2.1

PUF encompasses a wide range of physical components that continue to grow today. PUFs are the

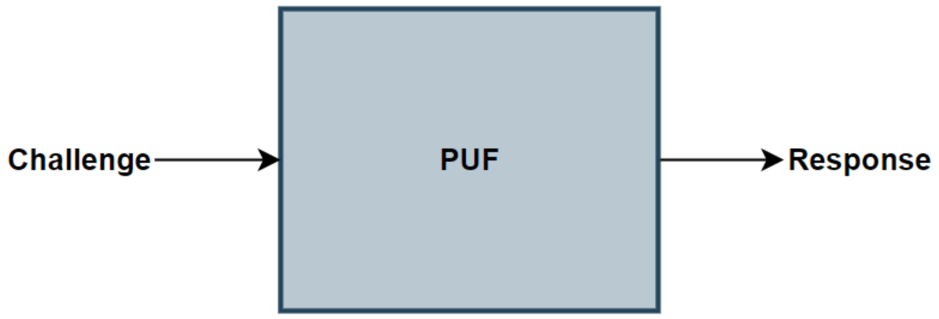


Figure 2.1: Basic architecture of a PUF

equivalent of human biometrics, such as fingerprints, because they provide a unique response that cannot be influenced or replicated. As a result, they are highly sought after in cryptographic protocols.

As the name implies, a PUF cannot be physically cloned. "Physically Unclonable Function" and "Physical Unclonable Function" are often used interchangeably. However, "Physical" in PUF refers to a physical, tangible device that is unclonable. In contrast, "Physically" is an adverb for "Unclonable" referring to a function that cannot be physically cloned. Both terms refer to the same concept; however, most authors today use "Physically Unclonable Function" [22].

2.3.1 Ternary-Addressable PUFs

Most proposed PUFs classify their responses as binary streams of '0' and '1'. Ternary-Address PUFs (TAPUF) classify their responses as ternary streams of '0', 'X', and '1' (see Table 2.1). In a ternary system, a '0' and '1' map to a '0' and '1' in a binary system. The third state, 'X', maps to either binary state at any given time because it is unstable [31].

Binary State	0	NA	1
Ternary State	0	X	1

Table 2.1: Binary state mapped to ternary state.

Cambou et al. introduced ternary addressable PUFs in 2018 [31]. Today, conventional computers run on binary logic, using two states, '0' and '1'; however, ternary logic uses trits, with three states, '-,' '0', and '+'. Ternary addressable PUFs use the "fuzzy" responses as a third, ternary state, with the other two states being a strong logical low and a strong logical high. So-called "fuzzy" responses are responses with higher intra-distance variations; they are more likely to flip between a logical low and high. On the other hand, strong logical lows and highs are responses with very low intra-distance variation and are highly unlikely to flip to another state. A "fuzzy" response is denoted by '0' while strong logical lows and highs are denoted by '-' and '+', respectively. The state 'X' is used internally to keep track of the unstable bits, and only states '0' and '1' generate responses. Ternary Addressable Physically Unclonable Function (TAPUF)s only utilize the '0' and '1' responses while disregarding the 'X' states. These 'X' states are internally filtered out. From a system-level perspective, the TAPUF functions as a typical binary PUF, generating binary streams of responses.

Identifying a ternary state has three main purposes: improving the Bit-error rate (BER), confusing attackers, and increasing the entropy of responses. Filtering out the ternary responses only leaves the strong '0's and '1's, improving the BER. Ternary states confuse attackers because they will not know which bits to filter out, and the resulting unfiltered response will have errors. Moreover, explicitly targeting ternary states generates non-deterministic random bits, which is ideal for a TRNG [31].

2.3.2 PUF Classifications

PUFs can be categorized into different classifications. In this report, PUFs will be classified into the following definitions.

Intrinsic vs. Extrinsic: The term *intrinsic* PUF was first used in [32] and is based on a PUF's construction. A PUF is considered *intrinsic* if it meets two criteria. One is that its response must be measured from internal measurement equipment. The other is that its random features must come implicitly from the production of a truly uncontrollable process. If it does not meet one or more of these criteria, it is considered an *extrinsic* PUF.

Intrinsic PUFs are seen as more secure because unwanted observers cannot measure the responses with external measurement equipment. With internal measuring capabilities, the response can be kept a secret. Moreover, in *intrinsic* PUFs, the manufacturer cannot influence the random features present in a PUF, making them secure [22].

Strong vs. Weak: PUFs vary in their strength, and they can be categorized as either *strong* or *weak*. Two main metrics are used to assess the strength of a PUF: its intrinsic or extrinsic nature and the number of CRPs it can generate.

Intrinsic PUFs are considered *strong*, while extrinsic PUFs are considered *weak*. The strength of an intrinsic PUF stems from its inherent randomness, which is derived directly from the physical properties of the device. This makes it difficult for attackers to predict or replicate the PUF's behavior.

The number of CRPs (see §2.3.3) a PUF can produce is another factor in determining its strength. *Strong* PUFs have more CRPs available. However, there isn't a definite threshold of CRPs determining whether a

PUF is *strong* or *weak*. Instead, the scaling behavior of CRPs with the size of the PUF provides insights into its strength.

Strong PUFs exhibit exponential scaling, meaning that as the size of the PUF increases, the number of CRPs grows exponentially. This exponential growth enhances the security and uniqueness of the PUF. In contrast, *weak* PUFs have linear scaling, where the number of CRPs increases linearly with the size of the PUF. *Weak* PUFs may be more vulnerable to attacks than their exponentially scaling counterparts[33].

The next metric to determine the strength of PUFs is the number of CRPs (see §2.3.3) that a PUF can produce. *Strong* PUFs have many CRPs that can be used for authentication. Is there a definite threshold of CRPs that determines if a PUF is *strong* or *weak*? Generally, *strong* PUFs refer to PUFs with their number of CRPs scaling exponentially with their size, while *weak* PUFs refer to PUFs whose CRPs scale linearly.

In addition to the number of CRPs, the method used to obtain those CRPs is also crucial in assessing the strength of a PUF. The concept of a serialized PUF was introduced by Nedospasov and Helfmeier et al. in 2013 [34]. A serialized PUF is a type of PUF that can only produce one CRP at a time.

The significance of a serialized PUF lies in its enhanced resilience against invasive attacks if it falls into the hands of an adversary. Limiting the number of CRPs that can be obtained within a given time frame, a serialized PUF reduces the potential damage caused by an attacker who gains access to the PUF. This limitation makes the serialized PUF more robust and less susceptible to exploitation.

Analog vs. Digital: Digital PUFs and analog PUFs represent two distinct categories based on the nature of their output responses. Digital PUFs are designed to generate binary responses directly. In contrast, analog PUFs produce analog responses that typically require additional pre-processing and post-processing steps to convert into a binary format [35].

Digital PUFs utilize various analog parameters, such as noise or delays, as the basis for generating their responses. However, the outputs of digital PUFs are limited to binary bits, meaning they can only be represented as either '0' or '1'. Despite being derived from analog characteristics, the final output of digital PUFs is a binary representation.

In contrast, analog PUFs can produce analog responses spanning a continuous range of values. These analog responses may require specific pre-processing techniques, such as amplification or filtering, to enhance their quality and make them suitable for subsequent analysis. Moreover, post-processing steps may be employed to convert the analog responses into binary bits for further utilization in cryptographic applications.

Silicon vs. non-Silicon: A silicon PUF is a PUF that uses imperfections of integrated circuits (ICs) for its random physical features. The term "Silicon PUF" was first coined by Gassend et al. [30]. Manufactured Integrated Circuit (IC)s from the same or different wafers have unavoidable random variations due to instance, temperature or pressure. These unavoidable random variations make Silicon PUFs infeasible to clone. A significant advantage of silicon PUFs is that they can easily connect to standard digital circuitry and are widely available. As a result, silicon PUFs are the most popular category of PUF in the field of cryptography today.

Non-silicon PUFs are PUFs have not gained their randomness due to the IC manufacturing process. The material of non-silicon PUFs varies greatly. Most non-silicon PUFs are classified as extrinsic, and they can be both non-electronic and electronic. Some examples of non-electronic non-silicon PUFs are paper PUFs, Phosphor PUFs, Optical PUFs, and acoustic PUFs [36, 37, 38, 39]. An example of an electronic non-silicon PUF is the radio-frequency-based PUF proposed by Dejean and Kirovski [40].

2.3.3 PUF Terminology

The following terminologies are critical for a better understanding a PUF.

- **Manufacturer Resistant:** A PUF is manufacturer resistant if it is infeasible to produce two identical PUFs. The inability to create two identical PUFs means that the randomness of the PUF comes from parameters in the manufacturing process that cannot be controlled [35].
- **Tamper Resistant:** Tamper resistance refers to the inherent characteristics of a device specifically designed to resist unauthorized physical tampering without being detected by the intended user. These characteristics are typically integrated into various device components, such as its circuitry and cell properties. Tampering techniques can encompass different methods, including side-channel attacks and electromagnetic analysis, which attackers may employ to gain unauthorized access to the device or extract sensitive information from it.
- **Challenge:** A challenge is a set of instructions sent to the PUF to generate a response [22].
- **Response/Evaluation:**The response or evaluation of a PUF is the measurement of a PUF given a challenge at a specific instance. A response is the outcome of a physical experiment, subject to minor variations every given time. Minor variations are due to external physical parameters that cannot be controlled [22].
- **Challenge-response pair (CRP)s:** A CRP is a challenge and its corresponding response. As discussed in section 2.3.2, the number of unique CRPs corresponds to the strength of a PUF.
- **Hamming distance:** Hamming distance is a metric used to measure the difference between two strings of equal length, counting the number of positions at which the corresponding symbols are different. For example:

'0011' and '1000' have a hamming distance of 3

'1000' and '0000' have a hamming distance of 1

- **Enrollment:** Enrollment is the secure process of collecting CRPs to recreate the PUF digitally. It is crucial to ensure enrollment takes place in a secure environment to prevent unauthorized cloning and potential system breaches by malicious attackers.

- **Entropy and Entropy Density:** Entropy quantifies the uncertainty or randomness in the possible key combinations generated from a PUF's CRPs. The entropy of a PUF can be calculated using equation 2.1, where E is the number of unique CRPs from a PUF, and r is the size of the key.

$$log_2 \binom{E}{r} \tag{2.1}$$

  Even though PUFs may have high entropy, the responses of a PUF should not be correlated to ensure a high level of security and randomness. Entropy density measures the amount of uncertainty or randomness in a sequence. The entropy density quantifies the uncertainty or randomness in PUF responses. It represents an upper bound on predictability, with a desired entropy value close to 1. To calculate the entropy, we use the Shannon entropy density function in equation 2.2 described in [41], and [42]:

$$H(X) = -\sum_{x} P(x) \log_2(P(x)) \tag{2.2}$$

  In the given equation, the variable $P(x)$ represents the probability of a random variable X having the value x. The logarithm, typically taken to the base 2 ($\log_2$), is used to quantify entropy in terms of bits. The negative sign ensures that entropy is always non-negative.

  Shannon entropy measures the average amount of information required to encode the possible outcomes of a random variable. When all outcomes are equally likely, the entropy is at its highest, indicating maximum uncertainty. Conversely, when there is only one possible outcome, the entropy is at its lowest, indicating no uncertainty.

- **Intra-chip distance/BER:** The Intra-chip distance of a PUF refers to a random variable that measures the Hamming distance between two PUF responses obtained from the same PUF using the same challenge, divided by the length of the response. It is denoted by HD_{intra} and serves as an important metric to assess the noisiness and reliability of a PUF in cryptographic applications [22].

Bit-error rate (BER) is a measurement used to quantify the reliability of a PUF. It is calculated by generating a response sequence from a set of challenges on the client device and comparing it to the response sequence generated with the same set of challenges on the server side. The server uses enrollment data, while the client uses the PUF to generate real-time responses to generate response bits. The number of discrepancies between the two sequences divided by the size of the response sequence is theBER. A formula for the BER is found in equation 2.3. In this equation, the HD_{diff} is the hamming distance between a cryptographic key generated on the server using a set of challenges and a cryptographic key generated on the client using the same set of challenges.

$$\text{BER} = HD_{diff}/\text{key size} \tag{2.3}$$

While BER and HD_{intra} are similar, the intra-distance compares two responses from the same challenge on the same PUF. In contrast, BER compares the responses of a PUF to that of responses generated using enrollment data. Ideally, the intra-distance and BER of a PUF should be **0%**; however, this is not feasible, and as a result, PUFs with HD_{intra} or BER close to 0% are more desirable [43].

- **Inter-chip hamming distance:**

  The inter-chip hamming distance measures the distance between two PUF responses generated from the same challenge but on different PUFs, normalized by the length of the responses. It is typically denoted by HD_{inter} and is an important metric for quantifying the uniqueness of a PUF. Specifically, it represents the Hamming Distance ($HD_{i,j}$) between a stream of responses from two distinct PUFs when subjected to the same challenges. The resulting distance is divided by the total number of responses to produce the HD_{inter}. Ideally, the HD_{inter} value of a PUF should be 50%, indicating maximum distinctiveness. As a result, PUFs with HD_{inter} values close to 50% are considered more desirable [43]. The equation for the HD_{inter} is found in equation 2.4.

$$HD_{inter} = HD_{i,j}/\text{stream size} \tag{2.4}$$

- **Intra-cell variation:** Intra-cell and inter-cell variation are terms used to describe the responses of analog PUFs (see 2.3.2). The intra-cell variation is the standard deviation of multiple cell responses from the same challenge.

- **Inter-cell variation:** The inter-cell variation is the standard deviation of all responses within a PUF.

- **Inter-cell to intra-cell ratio:** The inter-cell to intra-cell ratio is the inter-cell variation divided by the intra-cell variation. To generate a reliable “fingerprint”, the inter-cell to intra-cell ratio should be as high as possible.

- **False Acceptance Rate (FAR):** False acceptance is the acceptance of a PUF which is not the same PUF that produced the enrolled CRPs used for authentication. FAR is the rate at which a false acceptance occurs [22].
- **False Rejection Rate (FRR):**False rejection is when a PUF is not accepted, even though it is the same PUF that produced the enrolled CRPs used for authentication. FRR is the rate at which false rejections occur [22].

In our discussion, we have covered various classifications and terminologies associated with PUFs. However, it is important to note that certain terminology can be misleading. Therefore, to gain a comprehensive understanding of the advantages of a PUF in specific applications, it is crucial to examine it from a broader system-level perspective.

While a PUF may be regarded as a *strong* PUF based on the number of CRPs it possesses, it is crucial to examine its performance in other areas such as BER and entropy density. If a PUF exhibits deficiencies or significant weaknesses in these aspects, it could become a vulnerability within a cryptographic system. On the other hand, a PUF that might be considered a *weak* PUF in terms of the number of CRPs it has can still offer excellent qualities like tamper resistance and low BER. These qualities can significantly enhance the security of a cryptographic system, even if the PUF lacks a large number of CRPs.

2.3.4 PUF Limitations

PUFs provide significant security enhancements for cybersecurity protocols, but they also face challenges regarding ease of use, reliability, and security. The primary concerns with PUFs are their ability to consistently generate unique and predictable keys in various conditions without negatively impacting the user experience. Factors such as entropy, latency, noise, temperature, aging, and drifting can affect the ease of use, reliability, and unpredictability of PUFs and to be carefully addressed to ensure the optimal performance and security of PUF-based systems.

Entropy: As discussed in section 2.3.3, entropy quantifies the number of possible key combinations from a PUF’s CRPs, and entropy density quantifies the unpredictability or randomness of a PUF. While strong PUFs have been previously mentioned as intrinsic PUFs with an exponential number of CRPs, this does not paint the whole picture. Arbiter and RO PUFs are considered strong PUFs; however, they are vulnerable to machine learning modeling attacks because the output can be predicted from previous CRPs [44]. Therefore, genuinely desirable PUFs are strong PUFs with high entropy, where it would be infeasible to predict future responses based on previous ones [45].

Latency: Latency is the time it takes for a PUF to produce an output response given a challenge. It is a crucial PUF metric, as latency is an important factor in authentication speed. Low latency means that keys can be generated faster. The maximum latency depends on the application of that PUF. Some applications require faster key generation, as more keys are required. Overall, most cryptographic processes aim for a reasonable time to encrypt and decrypt; generally, this time is the amount of time humans will notice. Therefore, the time that is aimed for in most protocols is to have the entire protocol take only a few seconds.

Reliability: Electrical noise refers to spontaneous fluctuations in current and voltage in an electrical circuit caused by the erratic movement of electrons. Shot noise arises from random variations in electron motion, while energy fluctuations in electrons cause thermal noise and are temperature-dependent. On the other hand, electromagnetic interference (EMI) occurs when electronic devices are exposed to external electromagnetic fields, resulting in disturbances in analog signals. While electrical noise and EMI are generally unavoidable, their impact can be mitigated through circuit design [46]. Electrical noise and EMI are generally unavoidable; however, they can be mitigated in the circuit design.

In addition, integrated circuits (ICs) exhibit temperature-dependent behavior due to changes in physical properties such as mobility, threshold voltage, subthreshold swing, and transconductance. These unique properties contribute to the IC's distinctiveness. To account for temperature effects on reliability, PUF enrollment is typically performed at multiple temperatures expected during operation. However, it should be noted that exposure to high temperatures can accelerate the degradation and aging of silicon PUFs [47].

Aging and drifting are natural phenomena that affect all ICs, leading to degradation and changes in device parameters over time. Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI) are prominent factors contributing to aging and drifting. Silicon PUFs have been observed to degrade and exhibit increased intra-distance measurements due to aging. Consequently, countering or mitigating the aging effect is a critical concern for silicon PUFs.

To address the reliability challenges of silicon PUF responses, various protocols have been implemented. These include Error Correcting Codes (ECC) and Response Base Cryptography (RBC), which aim to enhance the accuracy and security of PUF outputs.

2.3.4.1 Error Correcting Code (ECC)

ECC refers to codes designed to detect and correct errors in data transmission, even in the presence of noise or other disturbances. ECC uses data helpers to assist in error detection and correction. Data helpers are packets of information that provide redundancy alongside the original data. There are various types of data helpers, including parity bits, checksums, and hamming codes [48]. It has been used since the 1950s to ensure reliable communication over noisy channels [49]. In the context of PUF-like devices, ECCs can greatly enhance their reliability.

ECCs are used in environments where the error rate and noise level are generally higher. They are capable of correcting a high number of errors in a short period of time. However, one drawback of ECC is the significant overhead it introduces on servers and clients. This overhead includes computational costs and increased bandwidth requirements, which can be particularly costly in power-constrained systems [50]. Despite the benefits of ECC in improving reliability, its implementation needs to consider these trade-offs.

2.3.4.2 Response-based Cryptography (RBC)

RBC is a search engine used in PUF-based cryptography, specifically on the server side. RBC protocols, introduced in various research papers [51, 52, 53, 54], aim to extract a key from a noisy environment without relying on expensive error correction schemes on the client side.

In RBC, a key is authorized if it falls within a certain hamming distance, denoted as A, which is limited by the server's computational power. Increasing A requires exponentially more computational resources, so it is often more efficient to request the client to perform a new CRP generation instead of increasing A. The optimal balance between A and the number of queries (Q) is determined by the error rate of the PUF. For instance, a PUF with an average CRP error rate of 0.0003 can achieve an FRR of less than 0.001 in 10 seconds using a single query, whereas using two queries reduces the authentication time to just two milliseconds. Therefore, the RBC focuses on PUFs with low error rates to enhance the key generation and optimize the authentication process.

2.3.5 Silicon PUFs

A notable category of PUFs is Silicon PUFs. These PUFs utilize the natural variations created during microfabrication of ICs to generate random physical features. The term "Silicon PUF" was initially introduced by Gassend et al. [30]. When ICs are manufactured, whether from the same or different wafers, they inevitably possess random variations caused by factors such as access resistances, critical dimensions, or atomic level structure variations. These inherent and unavoidable random variations render Silicon PUFs difficult to clone or replicate. This feature adds to their security and reliability.

One significant advantage of Silicon PUFs is their ease of integration with standard digital circuitry. They can be readily connected to existing digital systems without requiring extensive modifications. Additionally, Silicon PUFs are widely available, further contributing to their popularity in the field of cryptography. Overall, Silicon PUFs represent the most prevalent and widely adopted category of PUFs in modern cryptography due to their robustness, accessibility, and compatibility with existing digital circuitry.

2.3.5.1 Gate Delay PUFs

A further classification of silicon PUFs is Gate-delay PUFs. Gate-delay PUFs utilize the unique variations in propagation delays found in ICs for their random physical features. In these PUFs, electrical signals race through interconnects and logic gates to determine the response. Gate delay PUFs were one of the first silicon PUFs. Thompson first theorized gate-delay PUFs in 1997 when he discovered that a configured circuit on an FPGA had a different output response than the same configured circuit on another FPGA [40].

The first significant gate-delay PUF was the arbiter PUF. The arbiter PUF was first proposed by Gassend in 2003 [35] and later introduced by Gassend and Lee et al. in 2004 [44]. An arbiter PUF is configured so that two signals race through an n number of switch components to get to an arbiter block. The arbiter block will determine the winner and set itself to a '0' or '1' accordingly. The challenge in arbiter PUFs is the configuration of the switch components. Switch components are set to '0' or '1'. The '0' gate configures the signals straight through, while the '1' gate configures them to cross paths. An arbiter PUF has a 2^n number of challenges and has shown to be reliable enough to be used for authentication and secret key generation.

The ring oscillator (RO), latch, flip flop, and glitch PUFs are other gate-delay PUFs introduced after the arbiter PUF. Suh and Devadas presented the ring oscillator (RO) PUF in 2007. The RO PUF exploited the unique oscillation of delay gates to generate its output responses [55]. In that same year, the latch PUF was introduced by Su et al. [56], which exploits the natural variations in latches. The latches are composed of cross-coupled NOR gates, and when they are pulled low, they will be drawn to a state depending on the delay of the two NOR gates. The flip-flop PUF was introduced by Maes et al. in 2008 [57]. It utilizes the preferred power-up values of flip-flops on an FPGA as its output response. Like the latch PUF, the preferred power-up state depends on the gate delays. Afterward, Suzuki and Shimizu introduced the glitch PUF in 2010, utilizing the unwanted delays in the circuit to produce jitters or "glitches" in waveforms [58]. These glitchy waveforms created a reproducible but unique response used as a PUF response.

A common problem with gate-delay PUFs is often reliability. Electrical signals travel at different speeds depending on temperature, voltage variations, and device age. Moreover, the entropy of some gate-delay PUFs has also come into question. The arbiter and RO PUFs are vulnerable to modeling attacks, predicting a CRP based on its input. It was found that a non-invasive modeling attack could predict the response of a PUF with 97% accuracy [44].

2.3.5.2 Memory PUFs

Memory-based PUFs constitute another significant category of silicon PUFs that leverage the inherent cell-to-cell variations of memory devices. Unlike gate-delay PUFs, which rely on timing variations in gates and connections, memory-based PUFs derive their responses from the diverse electrical characteristics of individual memory cells.

Memory-based PUFs have garnered considerable attention in the field of cybersecurity due to their stability, high CRP, and enhanced security features. Consequently, various types of memory-based PUFs have emerged over the past decade to explore their potential.

The first memory-based PUF to gain prominence was the Static random-access memory (SRAM) PUF, which was introduced by Guajardo et al. in 2007 [32]. Since then, additional memory-based PUFs, such as Dynamic Random-Access Memory (DRAM), Resistive Random-Access Memory (ReRAM), and Magnetic Random-Access Memory (MRAM), have been developed.

2.4 Memory PUFs

In this section, we will delve into the previous research for memory PUFs based on ReRAM, SRAM, and MRAM. Memory PUFs have emerged as a promising approach for generating unique and unclonable identifiers in cryptographic systems. ReRAM, SRAM, and MRAM technologies offer distinct advantages and challenges in implementing PUFs. We explore the latest developments, key characteristics, strengths, and limitations of each technology in the context of PUF applications. Understanding the current state of ReRAM, SRAM, and MRAM PUFs will provide valuable insights into the advancements and potential directions for secure and reliable authentication and key generation systems. Memory-based PUFs have unique variations that are exploited differently to generate a response. The focus of this dissertation will be innovative research using SRAM, MRAM, and ReRAM technology.

2.4.1 SRAM PUFs

2.4.1.1 SRAM Technology

SRAM is a type of volatile memory that utilizes latching circuitry to retain information as long as it is powered. A typical SRAM cell in a Complementary Metal-Oxide Semiconductor (CMOS) configuration consists of four to six transistors arranged in a specific pattern, forming a flip-flop circuit. The cell includes two cross-coupled inverters that store and amplify the data bit stored in the SRAM cell, as depicted in Figure 2.2 [3]. When an SRAM cell is powered on, it assumes its initial state, either ‘0’ or ‘1’. The threshold

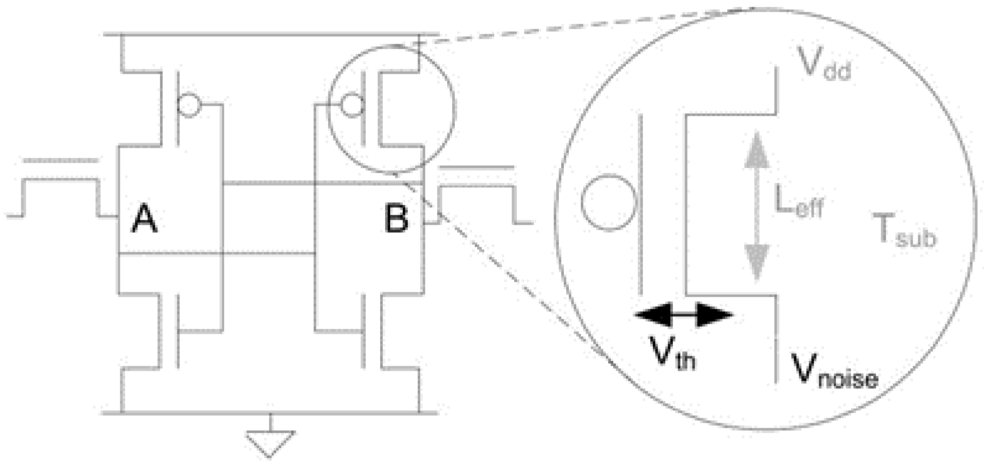


Figure 2.2: Six transistor SRAM Cell [3]

voltage of the cross-coupled transistors determines the initial state of an SRAM cell. Thus, when an SRAM is initially powered on, it should exhibit an approximately equal distribution of ‘0’s and ‘1’s.

When there is enough threshold difference, it leads to a highly skewed initial state, either a predictable ‘0’ or ‘1’. About 80% of the SRAM cells are in this category. Conversely, when the threshold difference is low, the initial state becomes more susceptible to noise, and the probability of having initial states at ‘0’ or ‘1’ are similar. This behavior is illustrated in Figure 2.3.

SRAM cells that exhibit small threshold voltage differences are often referred to as fuzzy or unstable cells. These cells tend to switch easily between the two states during start-up, resulting in unpredictable behavior. Apart from threshold voltage differences, other small parameters arising from the fabrication process can also contribute to the cells assuming a strong ‘0’ or ‘1’ state or exhibiting unstable behavior. These variations make each SRAM device unique and unclonable.

2.4.1.2 SRAM-based PUFs

SRAM is the most prevalent PUF technology, primarily due to its widespread adoption in electronic devices worldwide. The ubiquity of SRAM chips, coupled with their mass production, ensures the existence of a well-established supply chain and cost-effective manufacturing processes. The utilization of SRAM as a PUF is particularly appealing due to its simplicity, energy efficiency, and existing infrastructure.

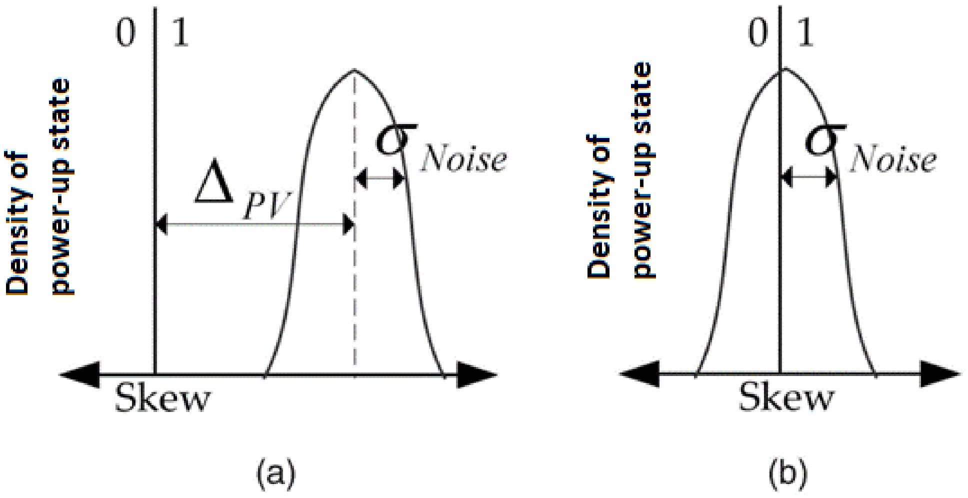


Figure 2.3: Predictable vs. Fuzzy Cell. The Δ_{PV} is the threshold voltage skew, and σ represents the noise in the system which can alter the power-up state of a cell. (a) The visualization of a predictable 1 (b) The visualization of a fuzzy cell that has a slight skew towards 1. [3]

SRAM PUFs, first introduced in 2007 by Guajardo et al. and Holcomb et al. [32, 3, 59], are used to generate unique identification sequences for device authentication. The uniqueness of the SRAM PUF sequence is derived from its power initiation process, where each SRAM cell in a specific device has a preferred start-up state of either '0' or '1'. By filtering out unstable cells, both designs have demonstrated the ability to generate reliable physical fingerprints. Building upon this research, further studies [1, 60] have explored the implementation of SRAM PUFs on power-constrained devices like FPGAs and microcontrollers. These efforts aimed to develop more realistic models that resemble real-world usage scenarios.

While SRAM PUFs are widely commercialized and easy to use, it is important to acknowledge certain limitations and challenges associated with them. One key concern is the restricted number of available CRPs [32], which can limit the effectiveness and scalability of the PUF system. Additionally, SRAM PUFs are susceptible to cloning attacks because of their limited number of CRPs. In these attacks, unauthorized entities try to replicate the PUF by obtaining recorded responses [33]. Moreover, aging problems have been identified in SRAM PUFs, which can potentially impact the reliability and longevity of the PUF system over time [61, 43]. The severity of these issues may vary depending on the specific SRAM devices employed, but they are consistently present in all SRAM devices.

Although SRAM PUFs have inherent flaws, they have become increasingly popular due to their wide utilization in diverse electronic devices as cache and buffer memory. As a result, researchers have continued to explore the use of SRAM PUFs for authentication purposes, modifying SRAM technology to improve PUF metrics.

While the aging effect can be used as means to attack an SRAM PUF [62], Garg et al. proposed a means of using the aging effect to improve uniformity and mitigate its impact [63]. This approach effectively allows one to modify the SRAM responses. Similarly, Baek et al. proposed an SRAM PUF design that modifies an SRAM cell to have a configurable response [2].

However, the above-mentioned proposals' modification of responses not only makes their responses non-intrinsic and, by definition, makes them a weak PUF, but it also makes these designs inherently non-manufacturer resistant since modification of responses means that two identical PUFs can be manufactured. Moreover, the modification of these devices adds complexity to the use of SRAM, thereby negating one of the appealing factors that made them attractive for implementation.

SRAM PUFs are widely adopted due to their user-friendliness and widespread availability. They offer dependable response generation, but there are drawbacks to consider, including a restricted number of CRPs and aging effects, which can affect their long-term reliability as a PUF [61, 43]. Efforts to address these downsides have encountered challenges in maintaining intrinsic randomness and ease of use. A comprehensive comparison of PUF metrics across various SRAM designs is found in Table 2.2.

PUF Metrics	PUF Designs		
	Microcontroller SRAM PUF[1]	**FPGA SRAM PUF[60]**	**Reconfigurable SRAM PUF [2]**
Number of CRPs	n	n	n
BER or HD_{intra}	0.12	0.02712	NA
BER or HD_{intra} (fuzzy extractors)	6.85×10^{-7}	NA	0.009
HD_{inter}	0.4954	0.4806	0.499
Entropy Density H(X)	NA	0.9979	NA

Table 2.2: PUF Metrics of previously implemented SRAM PUFs. n is the number of SRAM cells on the device. Work in [1, 2] used fuzzy extractors, a mix of filtering unstable cells and ECC, to improve reliability.

2.4.1.3 SRAM-based TRNGs

SRAM-based PUFs have also been utilized as TRNG [3, 59, 64]. In addition to their application in authentication protocols, Holcomb et al. proposed using unstable cells within an SRAM device to generate non-deterministic random bits. Cells with a small threshold voltage mismatch will likely have their start-up state influenced by noise. Since noise can be random, the resulting output bits of the TRNG are expected to be random. However, it was discovered that interference and bias affected the unstable cells in [3, 59], resulting in insufficient entropy to pass certain tests in the NIST testing suite. The unstable bits were subjected to post-processing through a hash function, which enabled them to pass three of the NIST tests successfully.

The utilization of unstable SRAM cells for TRNGs has been further explored in subsequent studies [65, 66, 67, 68, 69]. In [65] and [66], the start-up values of unstable cells were employed to initialize an RNG. In [67] and [68], accelerated aging techniques were utilized to enhance the instability of cells for TRNG purposes. Additionally, in [69], noise-injected SRAM cells were used to generate a higher proportion of unstable bits, thereby increasing the entropy of the resulting random bits.

There have been alternative designs for SRAM TRNGs that do not rely on unstable SRAM cells during start-up. One such proposal was made by Yeh et al. in their work presented in [70]. This design combined SRAM with contact Resistive random-access memory (ReRAM) to generate random numbers. By leveraging the random telegraph noise inherent in contact ReRAM devices, the entropy of the generated responses is enhanced. Notably, the random sequences produced by this design successfully passed the NIST Test Suite, indicating their adherence to established randomness standards.

Another SRAM True TRNG design was introduced by Yüksel et al. in their work documented in [71]. This design involves deliberately under-scaling the voltage supply to the SRAM device, inducing read errors in the process. In this approach, values are initially written to the SRAM device using a proper power supply, and subsequently, they are read using an under-scaled power supply. In this work, the authors could also generate a continuous stream of random bits that not only passed all the NIST random number tests but also eliminated the need for power cycling.

Overall, the use of SRAM-based TRNGs has proven to be highly successful, as most implementations can pass the NIST suite with the aid of post-processing techniques. However, designs that aim to generate both reliable and non-deterministic random bits come with certain drawbacks. These designs require server overhead to select unstable bits and require frequent power cycling to generate new bits, resulting in increased latency in bit generation. On the other hand, designs that can generate continuous bit streams without server overhead sacrifice the ability to produce reliable bits suitable for authentication protocols.

2.4.2 MRAM PUFs

2.4.2.1 MRAM Technology

MRAM is a promising technology that uses a magnetic tunnel junction (MTJ) configuration. Julliere first proposed this configuration in 1975 [72]. In this configuration, two ferromagnetic films are separated by a thin insulating material, the tunnel junction. One of the films is fixed in a specific magnetic orientation, while the other is free to change during programming cycles.

When the insulating material is thin enough, electrons can tunnel through it, effectively creating electrical resistance. This resistance is dependent on the thickness of the insulating material and behaves as a resistor.

MRAM cells utilize the magnetic tunnel junction (MTJ) configuration to store information. To encode data, the magnetization orientation in the ferromagnetic films is adjusted. The electrical resistance displayed by the MRAM cell determines whether it stores a ‘0’ or a ‘1’. When both films have their magnetic orientations aligned (parallel configuration), the MRAM cell exhibits lower resistance, representing a state of ‘0’. On the other hand, if the films have different magnetic orientations (anti-parallel configuration), the cell exhibits higher resistance, indicating a ‘1’ state.

The tunneling magnetoresistance ratio (TMR) in MRAM refers to the relative change in resistance between its high and low states. A higher TMR ratio improves the read capabilities of MRAM by making it easier to distinguish between the high and low resistance states. In MRAM circuitry, CMOS transistors are used, introducing a series resistance typically in the kΩ range. The presence of series resistance in MRAM circuitry affects the overall relative resistance of MRAM cells. It is crucial to take this resistance into account when using MRAM as memory or when utilizing its resistance for purposes like generating PUF responses. Several studies, including those by Apalkov et al. [73], and Vatajelu et al. [74], discuss the influence of CMOS series effects and considerations in MRAM.

MRAM technology can be classified as Toggle, Spin-transfer torque (STT), and Spin-Orbit Torque (SOT) MRAM. These technologies are based on different writing methods [73]. These MRAM cells are shown in Figure 2.4.

- **Toggle MRAM:** The first generation of MRAM utilizes a magnetic field to write data into the MRAM cells. Among the first-generation MRAM technologies, Toggle MRAM is the only one that is still in production. Toggle MRAM employs the Savtechenko switching method for writing data, as proposed by Savtchenko in 2003 [75]. One notable advantage of this method is that it provides unlimited write endurance, meaning that data can be written to the cells an unlimited number of times. The stability of the magnetic domains is due to the elliptic shape of the cells; however they are vulnerable to external fields. A significant drawback of this writing method is that it becomes increasingly challenging to scale down the MRAM technology as its size decreases [73].

- **STT MRAM:** STT MRAM, the second generation of MRAM technology, operates by passing a current through the tunnel junction. This current exerts a positive or negative torque on the free layer, allowing it to be flipped for '0' and '1' states [73, 76].

  STT MRAM stands out for its ability to achieve large TMR, making it ideal for memory applications, as highlighted in [77]. One of its primary advantages is its fast read/write functionality, enabling quick data access and modification. Additionally, STT MRAM offers ease of scalability, making it suitable for implementation in various memory architectures and sizes. Moreover, it exhibits high reliability, ensuring data integrity and long-term operation. However, one of the drawbacks of STT MRAM is the limited endurance and read cycles can result in reprogramming the cells, a problem called "read disturb".

  In summary, STT MRAM represents a significant advancement in MRAM technology, combining desirable characteristics such as fast operation, scalability, and reliability. These qualities make it an excellent option for memory applications in different computing systems. The only drawbacks are the limited endurance and the possibility of "read disturbs" in the reprogramming phase.

- **SOT MRAM:** SOT MRAM is an emerging technology that shows promise in addressing the endurance and "read disturb" issues associated with STT MRAM. Unlike STT MRAM, SOT MRAM employs electric currents parallel to the magnetic tunnel junction during programming cycles. In contrast, during read cycles, the resistance of the cells is measured by passing a current through the tunnel junction [5]. These read cycles are considered safer, resulting in a significant improvement in reliability and data integrity.

Furthermore, SOT switching in SOT MRAM offers the possibility of enhancing the speed and endurance of MRAM technology while reducing power consumption [78]. However, it is important to note that SOT MRAM is currently in the developmental stage and not yet ready for large-scale production. Extensive research programs are currently underway to address challenges such as reducing bit error rates and minimizing the size of the memory cell. These efforts aim to make SOT MRAM a viable and commercially available memory technology in the future.

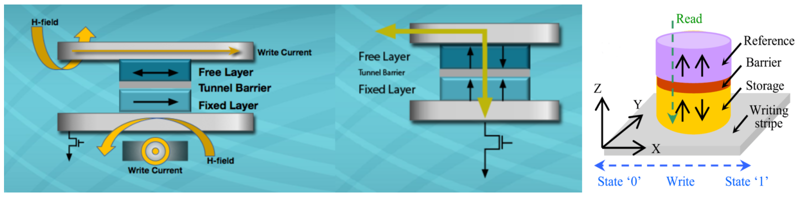


Figure 2.4: Toggle MRAM (left)[4], STT MRAM (middle)[4], and SOT MRAM (right)[5]

2.4.2.2 MRAM-based PUFs

In 2014, Zhang et al. proposed an early MRAM PUF system [79]. This MRAM PUF system used two STT MRAM cells combined as a single PUF cell to generate a response. The response value was determined by comparing the electrical resistance of the pair of STT MRAM cells, which were configured to a common state (either high or low). Based on the comparison result, a '0' or '1' response was generated. To improve the BER, an Automatic Write-Back (AWB) Scheme was implemented. If the comparison result was '0', the cell pair was written as '0', and if the comparison result was '1', the cell pair was written as '1'. The researchers conducted simulations of this PUF design and reported an inter-distance of 0.501, an entropy calculation of 0.985, and a BER of 6.6×10^{-6}.

STT MRAM PUFs were later continued in [74] and [80]. In [74], all STT MRAM cells were written to a common state, and the reference current was tuned to the median of the responses, splitting all responses into equal parts, '0's and '1's. Like other proposals, this one was simulated, showing an inter-distance of 0.499-0.501, an entropy of 0.994-0.999, and a BER of 0.05-0.07. In [80], an STT MRAM PUF proposal similar to the one proposed in [79] used diode-connected transistors to reduce the effects of deterioration. This proposal was also simulated and showed an improved BER of 0, with the BER increasing to 0.0001-0.0004 with changes in temperature and voltage.

Another MRAM PUF design was proposed by Das et al. in 2015 [81]. This design used the geometric variations of MTJs to generate responses. In short, the researchers destabilized the MTJs and then released them, and whichever state (e.i. '0' or '1') the MTJ preferred was used as a response. This design was simulated using stochastic LLG, and the design was found to have an inter-distance of 0.47, an entropy

	PUF Designs		
PUF Metrics	**STT MRAM PUF [79]**	**Geometry-based MRAM PUF [81]**	**Write Current STT MRAM PUF [74]**
Number of CRPs	$n/2$	n	n
BER or HD_{intra}	6.6×10^{-6}	2.25×10^{-2}	5×10^{-2}
HD_{inter}	0.501	0.47	0.499
Entropy Density H(X)	0.985	0.99	0.999

Table 2.3: PUF metrics of previous MRAM PUF designs. n = the number of MRAM cells on a single MRAM device.

calculation of 0.99, and a BER of 0.0225.

To summarize, previous MRAM PUF designs have leveraged the electrical resistances of MRAM cells or the geometrical variations of Magnetic Tunnel Junctions (MTJs) to generate their responses. However, most of these designs have remained theoretical or have only been implemented through simulations. Some proposals that achieve a lower BER utilize schemes that could potentially expose them to noninvasive key extraction attacks, as documented in [17]. Furthermore, these designs suffer from a limitation where the number of CRPs scales linearly with the device size, rendering them *weak* PUFs.The PUF metrics of MRAM PUFs can be found in Table 2.3

2.4.2.3 MRAM-based TRNGs

In addition to its applications in authentication, MRAM technology has garnered attention as a crucial element in the development of TRNGs. Building upon previous research found in [74], Vatejelu et al. presented a method for generating non-deterministic random bits using MRAM technology in their work [82, 83]. To further enhance the reliability of their approach, the researchers incorporated an XOR gate that effectively filters out adverse environmental influences [84]. The latter proposal demonstrated a remarkable bit generation rate of 2.67 Mbps in its serialized implementation and can also pass the NIST Testing Suite.

An alternative approach to generating random bits using MRAM cells involves modifying the write operation originally designed for standard MRAM functionality. This particular method was employed in the study conducted by Yang et al. [85], wherein a lower write current was applied to the MRAM cells, and random bits were derived from either the write success rate or the switching time of the cells. Notably, this method achieved an impressive throughput of 66 Mbps and successfully passed the comprehensive NIST Testing Suite in simulations.

Likewise, Ferdaus et al. presented a method that involves modifying the write pulse duration to be shorter than the specified requirement outlined in the MRAM data sheet [86]. This deliberate reduction in the writing pulse duration introduces errors during the write operation, which are subsequently harnessed for random number generation. The random bits were post-processed through the SHA-256 hash function. This design was able to achieve a throughput of 21.47 Mbps and pass the NIST Testing Suite.

In a different study [87], another type of TRNG utilizing MRAM technology was introduced. This particular TRNG operated asynchronously and utilized an STT MTJ and a capacitor. The generation of random bits was based on the time it took for the connected capacitor to discharge. The design demonstrated a remarkable throughput of up to 127.8 Mbps during simulations. However, despite its impressive speed, the generated output did not possess the required level of uniform randomness necessary to pass the thorough evaluation conducted by the NIST Testing Suite.

In summary, most of the MRAM-based TRNGs discussed displayed remarkable throughput and randomness. However, all but the one in proposed [86] were done through simulations.

2.4.3 ReRAM PUFs

2.4.3.1 ReRAM Technology

Resistive random-access memory (ReRAM), also known as RRAM, is a type of non-volatile memory that stores data by changing the resistance of its cells. The underlying technology used in ReRAM is based on memristors, which were initially proposed by L. O. Chua in 1971 as a potential fourth fundamental circuit element connecting electric charge and magnetic flux [88]. However, it was only in 2008 that HP laboratories confirmed the existence of memristors by fabricating a titanium dioxide memristor [89], leading to advancements in the field of electronics.

Memristors are electronic components with two terminals that can change resistance based on the applied voltage or current. After programming, the components memorize the resistance, therefore, were named memristors. ReRAM typically has two states: a High Resistive State (HRS), also called a reset state, to memorize the '1', and a Low Resistive State (LRS), also known as a set state, to memorize a '0'. While there are different types of memristor structures, the most commonly used ones are metal-oxide memristors and conductive bridge RAMs (CBRAMs). These memristors rely on the creation and removal of cations, oxygen vacancies, or metallic ions to enable the switching mechanism. Examples of metal-oxides utilized in metal-oxide memristors include $TiO_2, SiO_x, ZnO, HfO_2, TaO_x, CuO, GdO_x, SrTiO_3$ [90].

Pristine memristors have high resistance on the scale of 100MΩ to GΩ and must be electroformed to enable resistive switching. Electroforming is a type of soft dielectric breakdown that increases conductivity [91, 92]. The electroforming process involves applying electrical stress until the memristor undergoes an abrupt change in resistance [93]. Electrical stress creates a conductive path between electrodes and does not fully disappear when stress is stopped [94]. Once a conductive path is formed, resistive switching can occur by strengthening or weakening the paths by applying voltage or current at certain levels and polarities. As a consequence of electroforming, memristors can operate as resistive switching memory. The different states of a ReRAM cell are illustrated in Figure 2.6.

Additionally, pre-formed range memristors have higher resistances ($300k\Omega$ to $3M\Omega$) when injected with current compared to electroformed resistances (2 to $15k\Omega$) [6, 93]. Pre-formed memristors exhibit changes in resistance when a low current ($< 10\mu A$) is injected into them. Their resistance changes as a function of the injected current, as shown in Fig. 2.5. Importantly, once the current injection ceases, the resistance returns to its original pre-formed level resistance, ensuring a high level of reliability.

Pre-formed ReRAM is considered tamper-resistant due to its sensitivity to electroforming. Electroforming occurs when a pre-formed memristor is exposed to a sufficiently high current, which varies depending on the specific characteristics of each memristor. Once a memristor is electroformed, it cannot be restored to its pre-formed state. Consequently, if memristor cells are found to be electroformed, it suggests possible tampering with the device.

Due to variations in the manufacturing process, the resistance variations in each pre-formed ReRAM cell are different. This unique behavior is a feature of pre-formed memristors. Pre-formed memristors are generally considered undesirable for conventional memory applications. However, some exceptions exist, such as in forming-free ReRAM; the pristine state is used as one of the resistive switching states. Nevertheless, their distinctive physical properties make them well-suited for applications like PUF, where their uniqueness is desirable.

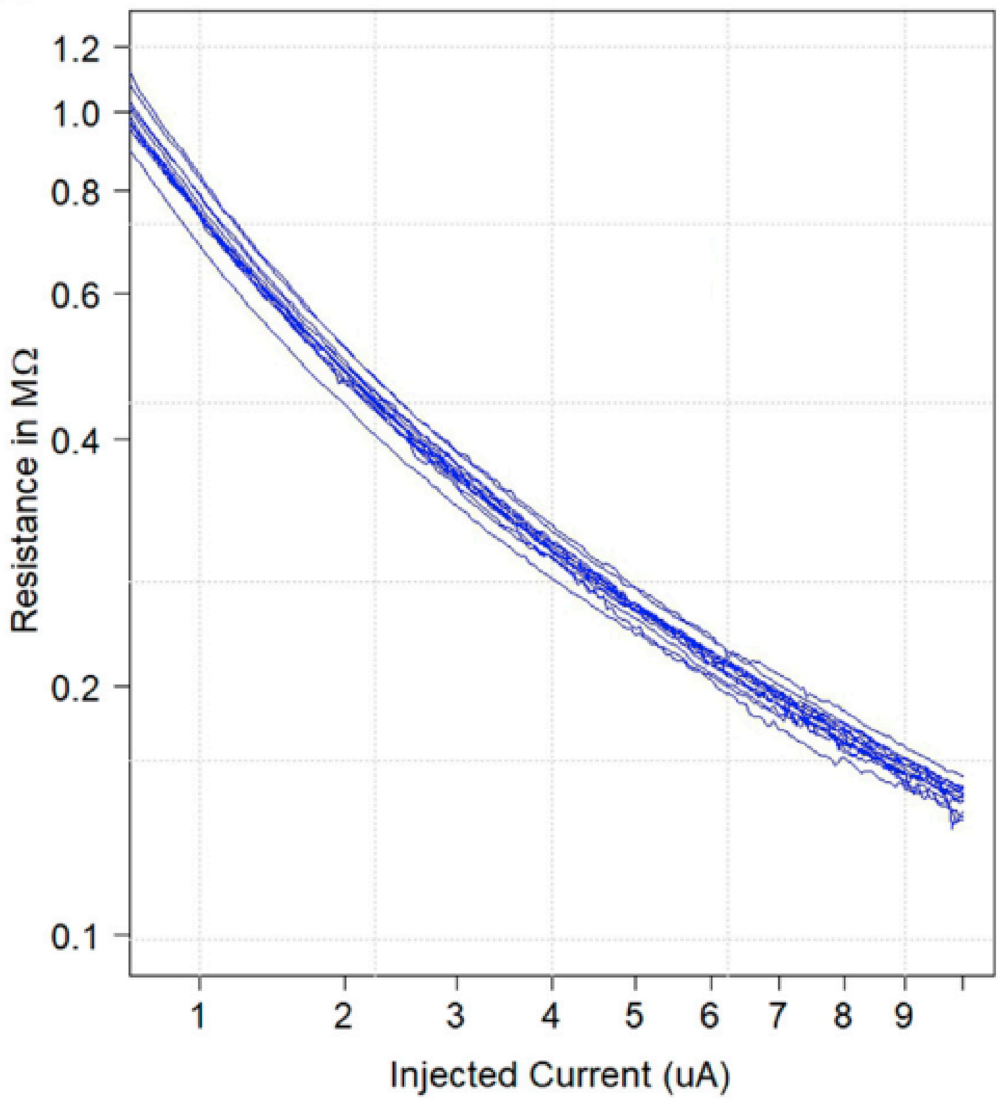


Figure 2.5: Pre-form ReRAM responses as a function of injected current [6]

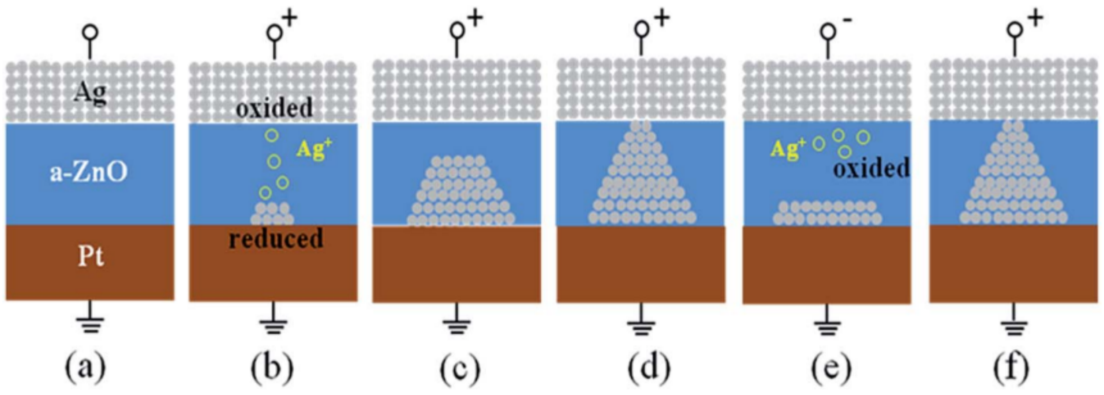


Figure 2.6: This is a schematic diagram illustrating the resistive switching effects in Ag/a-ZnO/Pt devices based CBRAM. (a) Pristine state. (b)-(c) Ag^+ undergoes oxidation at the TE (Top Electrode) while the mobile Ag+ cations migrate towards the cathode and get reduced there. (d) Ag^+ metal atoms precipitate at the Pt electrode, resulting in the growth of an Ag^+ filament. This filament eventually forms a highly conductive pathway within the cell. (e) When the polarity of the applied voltage is reversed, an electrochemical dissolution of the filament occurs, resetting the cell to the OFF state.(f) The next process involves the SET operation, which follows a similar sequence. [7]

2.4.3.2 ReRAM-based PUFs

Numerous types of ReRAM PUFs have been explored and studied. One notable approach, presented by Beckmann et al., combines the resistance variability of ReRAM cells with gate delays to generate responses [95]. In this design, the responses depend on the varying resistance of the cells, even when they are set to the same state. The researchers achieved a reliability rate of 2.7% and an average inter-distance of 50%, using the data from ReRAM wafers and simulated gate delays. These performance metrics demonstrate the potential of leveraging the resistance characteristics and gate delays in ReRAM cells to construct reliable and distinctive PUFs for authentication and key generation purposes.

Pang et al. introduced a distinct variation of a ReRAM-based PUF that involved performing a differential read of adjacent pairs of ReRAM cells to generate responses [96]. To enhance reliability, the authors programmed the adjacent cell to the opposite state after the initial read. The experimental results were achieved by using ReRAM wafer data. They showcased promising performance metrics, including an intra-Hamming Distance (HD_{intra}) of 1% at a high temperature of 150°C, an inter-Hamming Distance (HD_{inter}) of 0.4996, and an entropy density of 0.9966.

Likewise, Lim et al. proposed a differential PUF based on ReRAM technology that utilized a current sensing technique for cell comparison [80]. This design employed a combination of differential and median comparisons between different cells. By implementing physical ReRAM devices, the design achieved a BER of 6×10^{-6} and an inter-Hamming Distance (HD_{inter}) of 0.4999.

The utilization of pre-formed ReRAM for PUFs was initially postulated by Cambou et al. and Wilson et al. in their respective works [97] and [98]. They proposed employing the technique of differential comparisons of pre-formed ReRAM cells to generate PUF responses. This research was further advanced by Jain et al. and Wilson et al. in their papers [99] and [6], wherein they conducted an in-depth analysis of the analog responses of pre-formed memristors at the wafer level. Remarkably, they found that pre-formed ReRAM exhibits a remarkably low bit error rate of 10^{-6}.

Furthermore, the application of pre-formed memristors was also suggested for analog key encapsulation in the field of cryptography, as outlined in [100]. In this proposal, the analog responses of ReRAM cells serve the purpose of encapsulating messages without relying on cryptographic keys. Further elaboration on this topic can be found in Section 2.5.1.

In conclusion, ReRAM-based PUFs, including both formed and pre-formed variations, have showcased exceptional PUF metrics and offer a wide range of applications due to their distinct characteristics. However, further research is required, particularly in the realm of pre-formed ReRAM-based PUFs, focusing on implementations involving physical devices on low-power hardware. By conducting such research, a deeper understanding can be gained to explore the full potential of pre-formed ReRAM-based PUFs in practical scenarios.

2.4.3.3 ReRAM-based TRNGs

ReRAM, like other memory technologies, has been suggested as a potential TRNG. In a study by Post-Pellerin et al. [101], they proposed a method involving undervolted SET and RESET operations that targeted only half of the ReRAM cell population. The states of these ReRAM cells determined the random bits. This research was further expanded upon in a subsequent work by Bazzi et al. [102], where they compared the generated bits using HRS LRS. Likewise, Peng et al. [103] proposed a ReRAM TRNG that utilized the variation of ReRAM cells in the HRS to generate random bits. They established the median of the HRS as the reference and generated bits by comparing each cell to this reference. Notably, these implementations successfully passed the NIST Testing Suite, ensuring their reliability.

On the other hand, there remains a scarcity of approaches that leverage pre-formed ReRAM for generating non-deterministic true random number sequences. Addressing this gap, Cambou et al. proposed the utilization of pre-formed ReRAM as a TRNG in their work [104]. The research focused on employing the ternary states of a differential ReRAM PUF to produce non-deterministic bits. These bits passed the NIST Testing Suite after some light-weight post-processing, validating their randomness and quality as true random numbers. This study contributes to advancing pre-formed ReRAM-based TRNGs and paves the way for their integration into various applications that demand high-quality random sequences.

2.5 PUF-Based Cryptographic Protocols

PUFs can be used as a fundamental component in new and existing cryptographic protocols to enhance security. These protocols take advantage of the unique CRPs PUFs produce. In this section, we will be covering the various cryptographic protocols that will be the focus of this dissertation: analog key encapsulation, Ternary Addressable Public Key Infrastructure (TAPKI), and PUF-seeded Post-Quantum Cryptography (PQC).

2.5.1 Pre-formed ReRAM-Based Analog Key Encapsulation

Analog Key Encapsulation (AKE) is a method of encapsulating or wrapping a key or message using only analog responses, eliminating the need to encrypt with a cryptographic key. Analog key encapsulation is based on Key Encapsulation Mechanism (KEM), which securely wraps a cryptographic key. KEM is used in a hybrid cryptographic system, where the bulk of encryption is done through symmetric key encryption. The private key is encapsulated using public key encryption, such as RSA and Elliptic Curve Cryptography, and then sent to the receiving party with the cipher text. AKE differs from KEM because it uses analog responses to encrypt and decrypt a cryptographic key, plain text message, or any other piece of binary information. In this dissertation, we will discuss AKE based on pre-formed ReRAM responses.

Pre-formed ReRAM-based analog key encapsulation was first proposed in 2020 [100]. This work exploits the characteristics of pristine memristors, which are well documented in [6]. The authors theorized that pre-formed ReRAM cells have a high enough inter-cell to intra-cell variation ratio to encapsulate the plain text messages or keys. As discussed in §2.4.3.1, pre-formed ReRAM cells have a high resistance ($\sim 100M\Omega$) than regular ReRAM cells; however, their resistance changes when injected with a low current (<10μA). Their resistance changes as a function of the injected current. It returns to its pre-formed resistance when the current stops being injected, making them highly reliable. Each pre-formed ReRAM cell's resistance changes differently due to variations in the manufacturing process. As a result, their analog response can be used to encapsulate a key directly.

In [100], the server will have an image of PUF A, while the client will have the actual PUF A. The server will generate a random number (RN) and then concatenate it with a password (PW); it will then hash it to generate a message digest (MD). The MD will contain addresses (A_{wi}), orders (b_{wi}), and currents (d_{wi}) arrays. Arrays A_{wi}, b_{wi}, and d_{wi} will be used with an image of PUF A to generate the response array (R_{wi}). The server will then take the plaintext message (M) it wants to be encrypted and use that to generate the natural number array (Q_{wi}). Next, the A_{wi}, b_{wi}, d_{wi}, and the Q_{wi} arrays will be used to create the final cipher array C. Afterward, the server will initiate a handshake with the client and exchange C and the RN. The client will then use the RN to generate the same A_{wi}, b_{wi}, d_{wi}, and then use its PUF to generate the R_{wi}. From here, the client can begin to decrypt the C to obtain the original plaintext message M. In theory, without both the PW and the PUF, C will not be able to be deciphered.

Pre-formed ReRAM-based analog key encapsulation was later continued in [105] and [106]. In [105], the plain text message (M) was used to generate its orders directly with the help of a mask. The protocol was implemented using a simulated pre-formed ReRAM PUF, where it passed a frequency analysis security test. In [106], noise was injected into the protocol proposed in [100] to simulate the noise that the responses of the pre-formed ReRAM would introduce, and then an error-correcting protocol was used to extract the correct result.

2.5.2 Ternary-Addressable Public Key Infrastructure

Cryptographic keys are needed for secure communication over unsecured channels. Usually, to ensure secure communication in these environments, encrypt using symmetric (Advanced Encryption Standard (AES)) or asymmetric (Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography) encryption protocols. However, as stated early, the distribution and storage of these keys create many vulnerabilities. AES keys that are usually stored on non-volatile memory can be stolen [18, 17], and Public Key Infrastructure (PKI), which is used to exchange private and public key pairs for RSA and Elliptic Curve Cryptography, can be broken with a powerful enough quantum computer [15].

An emerging scheme proposed to replace PKI is Public Key Exchange which is Addressable (PKA). PKA relies on cryptographic tables and repositories of random information. Rather than associating a public-private key pair with an individual's identity, each member of the PKA scheme is assigned a cryptographically secure and unique cryptographic table. A certificate authority or server maintains a copy of the cryptographic table for each member of the PKI, which allows for secure key exchange and authentication [107].

PKA relies on the client and the server to generate a key with the same cryptographic tables. The idea is to create cryptographic keys from cryptographic tables that allow the server and member to consistently produce the same key without relying on external mathematical functions. The server generates a highly secure random number with sufficient entropy, which is transmitted to the client. Both parties hash the random number using a cryptographically secure hash function to create a shared address. This address points to the first available 256 bits in the cryptographic table and is used as the shared private key. Moreover, PKA was also theorized with ternary logic instead of binary logic to increase security in [31].

PKA relies on multiple assumptions to function. The first is that all cryptographic tables are securely distributed to all members. The second is that all randomly generated information is truly random. If this information is predictable, the cryptographic key can also be predicted. The next assumption is that the mathematical-based security of cryptographic functions is incredibly strong, which may not be true. The final assumption is that cryptographic tables are stored in very secure locations. Considerations must be considered for both permanent (e.g., hard drives, solid state drives, etc.) and in-memory storage (system memory, RAM, cache, etc.) as malicious attackers can lift secret information from non-volatile memory [17, 18].

The Ternary Addressable Public Key Infrastructure (TAPKI) is an extension of the PKA protocol designed to address some of its shortcomings. TAPKI uses ternary (three-state) representation by defining the states as '0', '1', and 'X'. TAPKI has two phases: enrollment and key exchange. The enrollment phase involves the server defining the cryptographic tables, while the key exchange phase is performed every time the server and client perform a handshake. In TAPKI, the server knows the masked locations of the 'X' states, making it difficult for attackers to attack the client's cryptographic table. During key generation, the server and client produce a single secret called the seed, and a set of secret instructions is mutually shared between them. The client uses these instructions to know which cells to skip, and the remaining cells are read as either '0' or '1' to form the shared secret. An overview of TAPKI can be found below in Fig. 2.7.

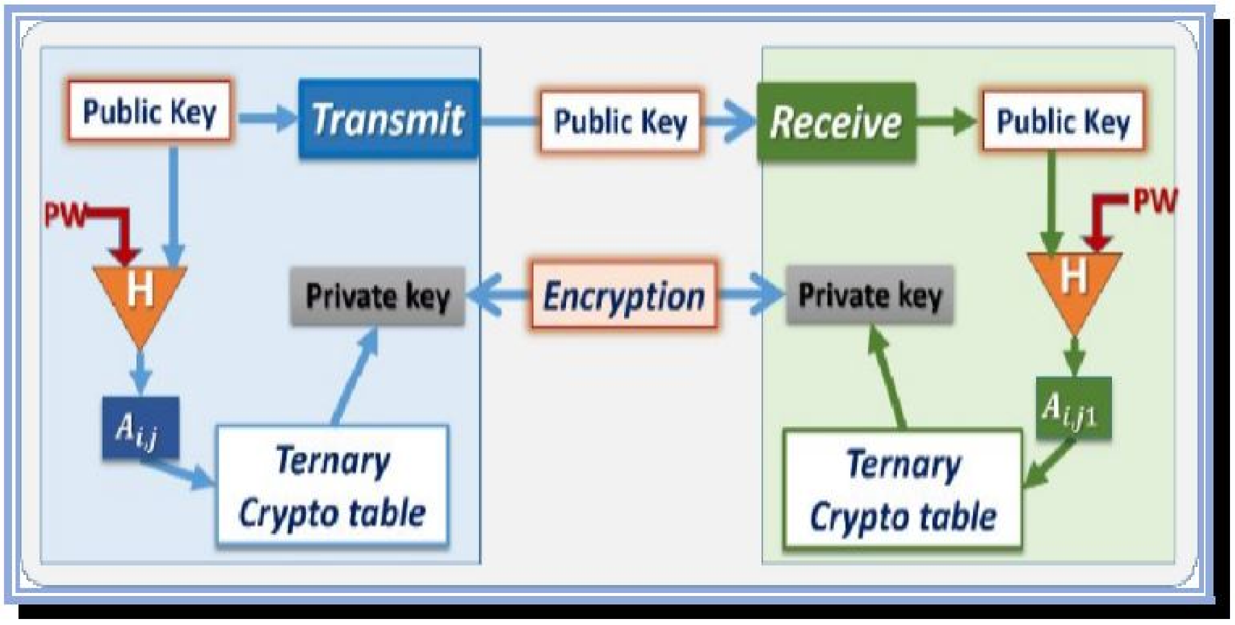


Figure 2.7: TAPKI Overview [8]

2.5.2.1 Securing TAPKI with TAPUFs

The security of TAPKI can be enhanced using a TAPUF instead of a ternary cryptographic table. Compared to the table, a TAPUF offers greater security since the response bits are not stored in non-volatile memory. When TAPKI is implemented with a TAPUF, the generation of response bits involves sending a set of challenges to the TAPUF. This process utilizes an image of the TAPUF on the server side and the physical TAPUF on the client side.

TAPKI's use of ternary states improves the reliability and security of key generation. Including ternary states in the handshake further obfuscates third-party attackers. They act as a mask that only the client and the server know what to do with. Furthermore, the discarding of "fuzzy" responses after the handshake improves the reliability of PUF responses, as they can increase the BER in the key generation process.

A common concern regarding PUFs is the modeling or "cloning" of PUFs. If hackers obtain access to a PUF, they might partially or entirely "clone" the PUF if they record its CRPs [33]. Additionally, while very low, there is a chance that a PUF might have a low enough inter-distance variation to pass as another PUF partially. To mitigate these vulnerabilities, TAPKI incorporates a ternary cryptographic table, random number, hash function, and multi-factor authentication.

TAPKI resolves many issues facing traditional key generation and storage by providing a secure way to produce a one-time key. It is a formidable form of key exchange that uses ternary states, multi-factor authentication, hash functions, and random numbers for added levels of security. TAPKI has been successfully implemented in C++ using two identical cryptographic tables and has produced duplicate private keys [8].

2.5.3 PUF-seeded Post-Quantum Cryptography

When Peter Shor published his influential algorithms in 1994 [16], it sparked a race to find alternative cryptographic protocols to RSA and Elliptic Curve Cryptography. In response, the National Institute of Standards and Technology (NIST) launched a comprehensive program in 2015 to develop standardized PQC algorithms. These algorithms fall into two categories: Public-key encryption/KEM and Digital Signature Authentication (DSA).

After rigorous evaluation rounds, several algorithms emerged as finalists for the NIST PQC program. These include CRYSTALS-KYBER for public-key encryption/KEM [108], CRYSTALS-Dilithium for DSA [109], Falcon for DSA [110], and SPHINCS$^+$ [111] for DSA. CRYSTALS-Dilithium and CRYSTALS-KYBER both use the lattice-based cryptographic mathematical problem Learning With Error (LWE), Falcon uses lattice-based NTRU [112], and SPHINCS$^+$ uses a hashed-based signature scheme. These algorithms have shown great promise and are being standardized in the post-quantum era of cryptography by NIST.

One way to implement PQC algorithms in a PKI is by having each client device or designated entity generate key pairs and send the public keys to a Certificate Authority (CA) which is a trusted entity that issues and manages digital signatures. This assumes that there is a separate authentication process in place and that the client devices can securely store their key pairs. However, in this approach, the requirement to store private key pairs still exists [9].

In this communication protocol, two client devices are involved, and they share public keys with each other. Each client device possesses its own unique key pair, consisting of a private key and a corresponding public key. For the KEM phase, Client 1 utilizes the public key of Client 2 to encrypt a shared key. This shared key is generated by Client 1 and is intended to be used for secure communication between the two clients. After encryption, the ciphertext containing the shared key is transmitted to Client 2. Client 2, possessing the private key corresponding to their public key, can retrieve the shared key by decrypting the ciphertext. This ensures that only Client 2, with their private key, can access the shared key that was originally encrypted by Client 1. In the DSA phase, Client 1 signs a message using their private key. This process generates a digital signature, which is cryptographic proof of the message's authenticity and integrity. To verify the message, Client 2 utilizes the public key of Client 1 to check the validity of the digital signature. By comparing the signature against the original message and using the public key for verification, Client 2 can confirm that the message originated from Client 1 and has not been tampered with during transmission. A visualization of PQC algorithms using standard PKI can be found in Figure 2.9.

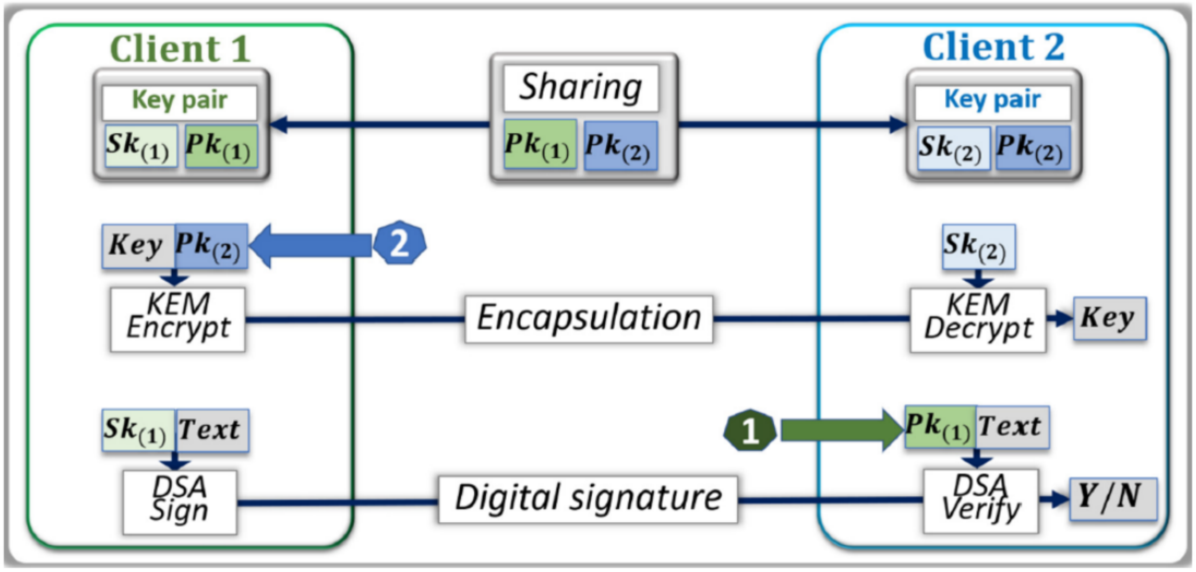


Figure 2.8: PQC Protocol with PKI [9]

To enhance the security of PQC algorithms, a method was proposed by Bertrand et al. [9] that involves using a PUF. This method includes a handshake process between the client devices, a CA, and an Registration Authority (RA), an entity that acts as an intermediary between the end user or entity requesting a digital certificate and the CA. During this handshake process, new key pairs are generated for each transaction. This means that instead of storing private key pairs on the client devices, new keys are dynamically generated based on the unique characteristics of the PUF for each interaction with the CA and RA.

By leveraging a PUF in the key generation process, the security of the PQC algorithms is improved. The PUF adds an additional layer of protection since the private key is not stored on the client device but rather generated on the fly using the PUF-based handshake in conjunction with the CA. This helps mitigate the risk associated with the compromise or theft of private keys, as there are no private keys to be stolen or leaked. A visualization of this protocol can be found in Figure 2.8.

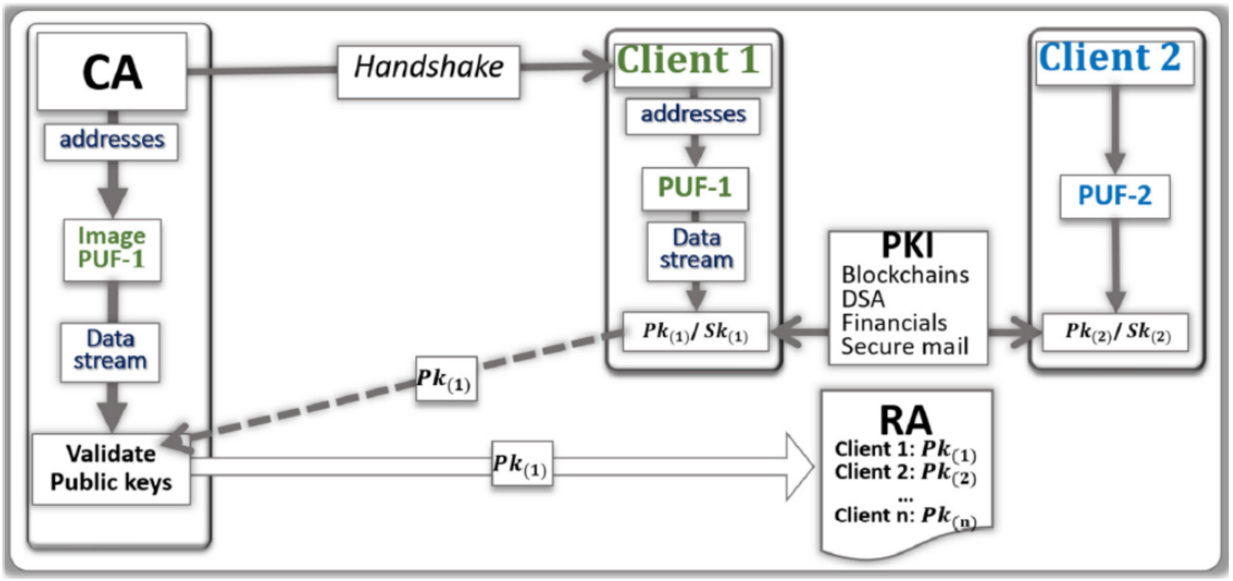


Figure 2.9: PQC Protocol with PUF [9]

Overall, this approach enhances the security of PQC algorithms within a PKI by eliminating the need for long-term storage of private keys on client devices and dynamically generating them using a PUF during each transaction with the CA.

Chapter 3

SRAM

3.1 Introduction

The use of Static random-access memory (SRAM) as a Physically Unclonable Function (PUF) is highly appealing due to its widespread presence in electronic devices, affordability, and simplicity, making it a popular memory technology. However, despite its advantages, SRAM PUFs face certain challenges.

In §2.4.1, we discussed the popularity of SRAM PUFs, which has led to the proposal and implementation of various types. These implementations have demonstrated the ability to generate reliable key bits without needing external circuitry. Additionally, SRAM PUFs have been utilized in the implementation of True Random Number Generator (TRNG)s that successfully pass the rigorous testing of the National Institute of Technology (NIST) Testing Suite, offering excellent throughput.

However, several drawbacks are associated with SRAM PUFs. One limitation is the limited number of Challenge-response pair (CRP)s they can generate, as highlighted in [32]. Moreover, SRAM PUFs are vulnerable to cloning, as discussed in [33], and can be affected by aging issues, as mentioned in [61]. Attempts to address these issues, as explored in research such as [63, 2], involve modifying the existing SRAM technology, which compromises its simplicity and widespread availability, two of its most attractive features.

Moreover, most SRAM-based TRNG implementations require targeting the devices' most unpredictable cells. These implementations involve additional server overhead to communicate the location of these cells to the client devices. Due to their limited number of CRPs, they also require constant power cycling to produce new bits, which leads to slower throughput. Research done in [70, 71] eliminated the need for power cycles and achieved a high throughput. However, this approach compromises the devices' ability to produce reliable bits for key generation, which is an essential aspect of PUF functionality.

The research conducted in this chapter aims to address the challenges encountered in previous designs of SRAM PUFs. The proposed design introduces the use of a novel SRAM PUF design that uses two SRAM devices and a single XOR gate, which improves the entropy and increases the number of CRPs while maintaining the ability to generate reliable bits. Additionally, this design is also capable of producing truly random sequences without the need for server intervention.

To assess the effectiveness of this design, several important metrics related to PUFs, including HD_{inter}, Bit-error rate (BER), throughput, and entropy, are collected and analyzed. These metrics provide valuable insights into the reliability and security aspects of the proposed design. Furthermore, the quality of the generated random sequences is evaluated using the NIST Testing Suite.

3.2 SRAM XOR PUF Design

The SRAM PUF design incorporates two separate SRAM devices and a boolean XOR gate. In this design, denoted as the SRAM XOR PUF, each SRAM device generates its own response, and these responses are combined using the XOR gate to generate a single bit. This unique approach results in a substantial increase in the number of available CRPs from the original count of n (representing the number of SRAM cells on each device) to n^2. Consequently, this significant increase in the number of CRPs greatly enhances the entropy of the PUF. Figure 3.1 provides a visual representation of this SRAM PUF design.

3.2.1 Key Generation Protocol

Key generation is a crucial feature of any PUF, and the SRAM XOR PUF is no exception. To generate keys using this PUF, a set of CRPs is randomly selected, and the most stable responses are chosen. However, cell stability is a critical consideration in cryptographic key generation using the SRAM XOR PUF. The design involves the instability of two SRAM cells, doubling the likelihood of bit errors. Therefore, cell stability is essential, and measures must be taken to detect and flag any unstable cells.

3.3 Hardware

In this implementation, ISSI IS64WV6416BLL SRAM Chips were utilized as SRAM devices. These chips have 2^{16} words of high-speed Complementary Metal-Oxide Semiconductor (CMOS) SRAM and operate at 3.3V logic, with access times as low as 12ns. They consume low power, are packaged in 44-pin packages, and can operate in temperatures ranging from $0° - 85°C$. To interface with these devices, we used custom PCB adapters designed to mix and match them. A figure of these devices and adapters can be found in Fig. 3.2.

The SRAM XOR PUF is implemented on a low-power client using a custom-printed circuit board (PCB). The client device used was the Nucleo-144 H743ZI2 developer kit, which features an ARM Cortex M7 Core microprocessor that operates at 480 MHz and 3.3V, 112 general-purpose input/output (GPIO) pins, and 2MB of flash memory and 1MB of random-access memory (RAM). A custom PCB was used to interface with the SRAM devices, and the PCB included an integrated circuit (IC) switch for power cycling the device.

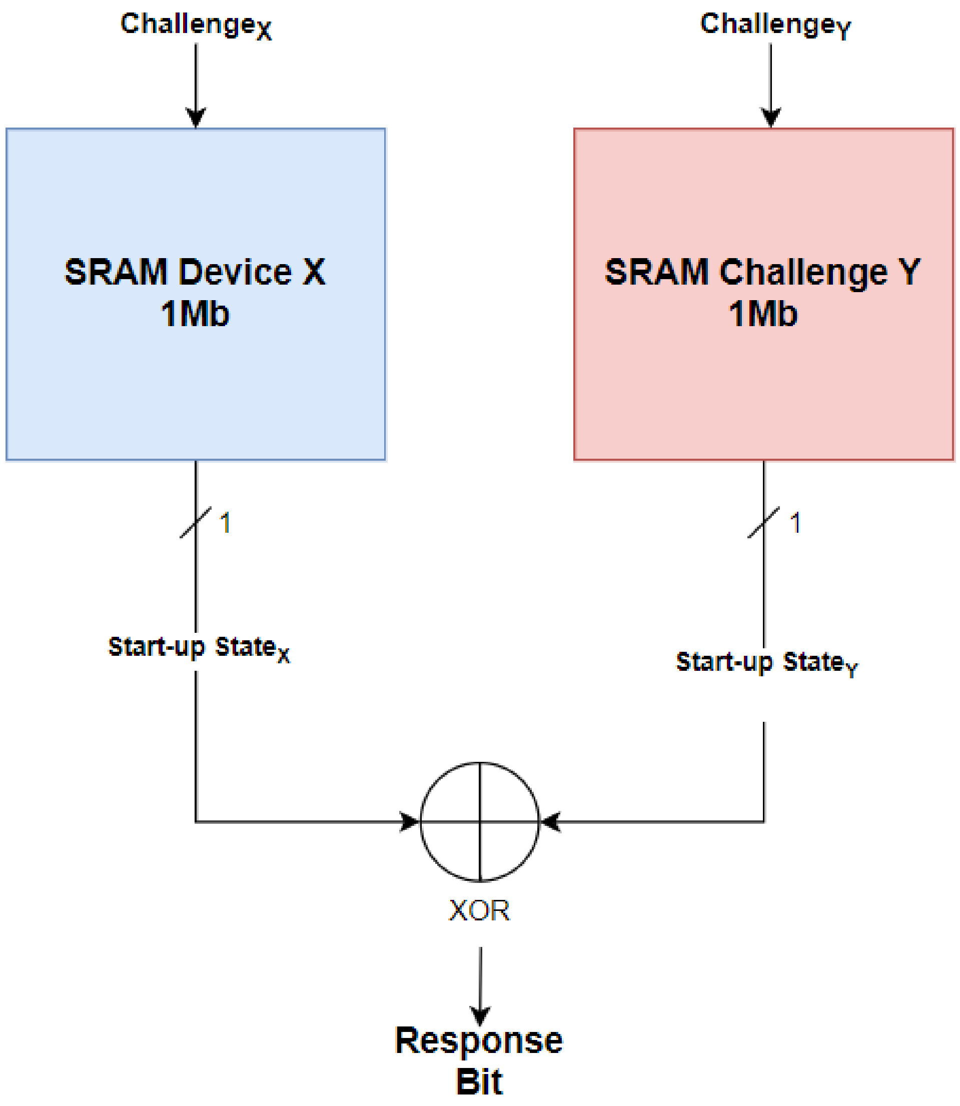


Figure 3.1: SRAM XOR Design

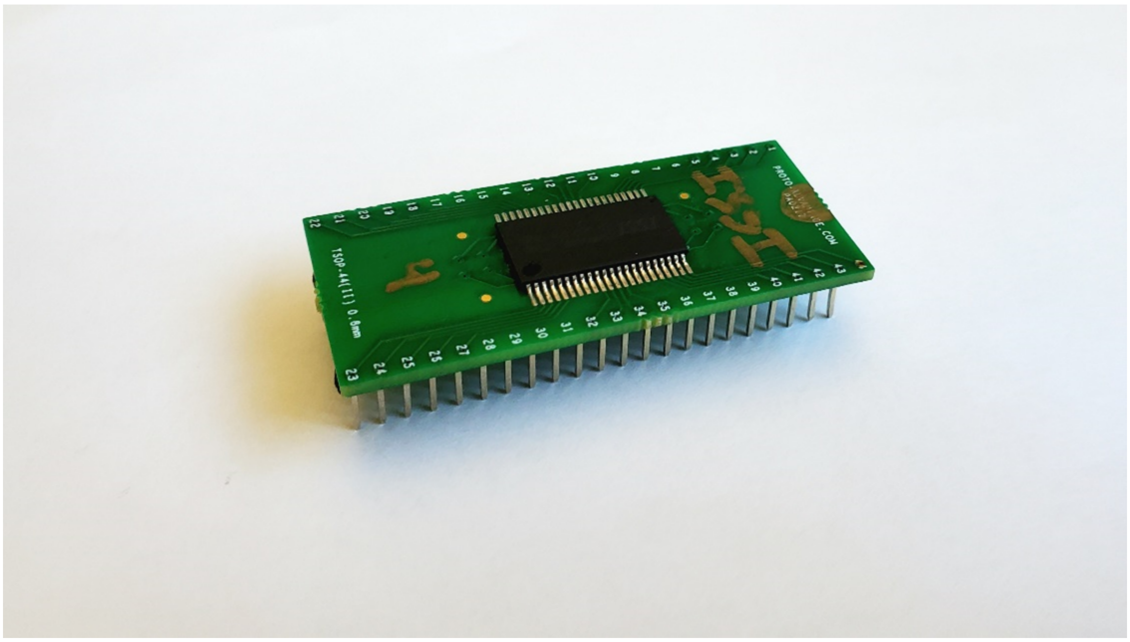


Figure 3.2: ISSI SRAM Device on custom PCB Adapter

With this configuration, we were able to achieve a throughput of 1.6Mbps. A figure of the client device, custom PCB and SRAM can be found in Fig. 3.3.

3.4 SRAM XOR PUF Testing and Methodology

To perform an electrical characterization of the SRAM XOR Design, we examined the behavior of the SRAM devices and SRAM cells, specifically focusing on their response after each power cycle. Moreover, we also focused on measuring key properties that define a PUF, namely reliability, uniqueness, and randomness. These properties were evaluated using specific metrics: BER for reliability, Inter-chip Hamming Distance for uniqueness, and entropy density for randomness. By calculating these metrics, we gained quantitative insights into the performance and security aspects of the SRAM XOR Design.

In addition, we conducted the tests across various temperature ranges: 0°C, 23°C, and 80°C. To facilitate temperature testing, we employed the Associated Environmental Systems SD-501 temperature chamber. This chamber offers a broad temperature range, from -37°C to 180°C, allowing us to create diverse temperature conditions. To ensure an accurate assessment of the SRAM devices and eliminate any potential influence from external measurement circuitry, we utilized jumper cables to isolate the SRAM device from the client

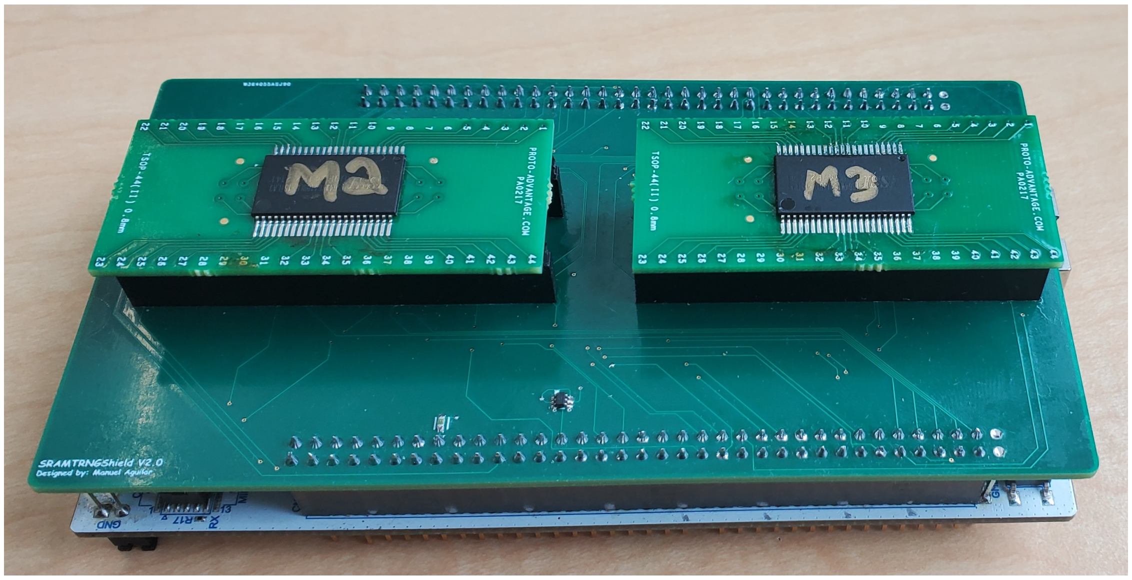


Figure 3.3: Nucleo-144 Developer Kit, Custom PCB, and SRAM Devices

device during the temperature tests. These measures helped isolate the impact of temperature on the devices under evaluation. A picture of the SRAM devices and jumper cables is found in Figure 3.4.

3.4.1 Enrollment

The enrollment process is done at the beginning of PUF implementation, where a PUF's responses are recorded and stored in a database. It is an extremely important step, as the database is used to authenticate the responses of the physical PUF. For SRAM, the process involves reading out the entire contents of the SRAM device after a power cycle multiple times.

3.4.1.1 Classification

If a bit flips anytime during enrollment, it is considered a 'flaky' or unstable cell (e.i., not 100% '1' or '0'). Cells that do not flip during the entirety of the enrollment cell are considered strong '1's or '0's. Classifying cells allows for selecting the most stable cells within the population.

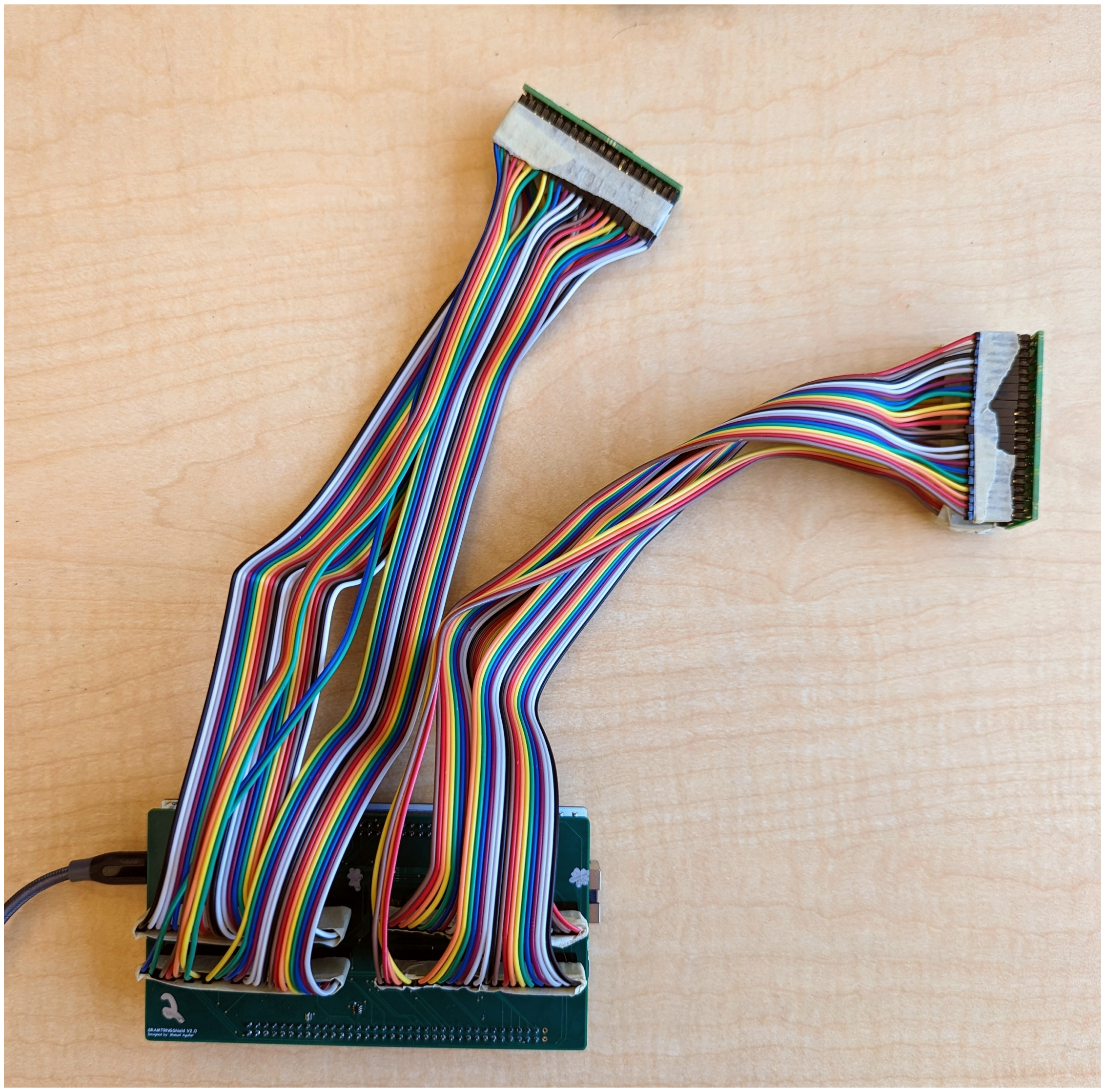


Figure 3.4: SRAM Device, connected to the client device using standard GPIO jumper cables

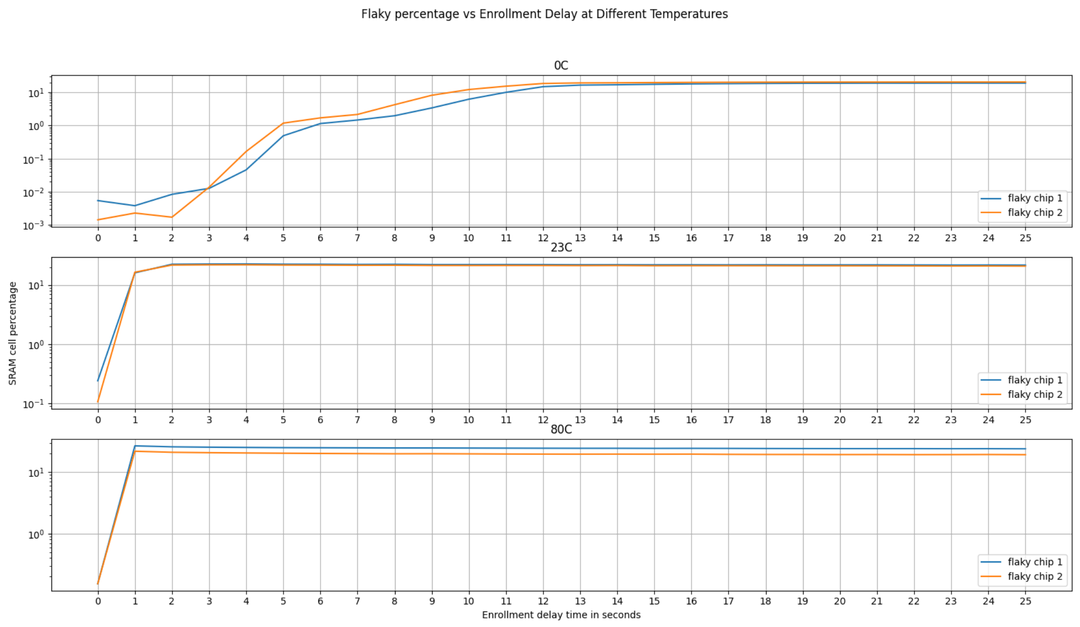


Figure 3.5: Flakiness vs. Enrollment Delay time over different temperatures

3.4.1.2 Remanence Effect

One of the obstacles encountered when enrolling the SRAM PUF is the remanence effect. The remanence effect is when magnetic material retains a certain level of magnetization even after the magnetic field that induced it is removed [113]. This causes previous reads to affect the initial state of future reads. In the case of SRAM enrollment, this affects the number of cells flagged as stable or unstable. For example, if a cell is first a ‘1’ and then flips to a ‘0’ if read twice back to back, you could incorrectly read a ‘1’ before the cell discharges to ‘0’. To mitigate this effect, we must have sufficient time between power cycles to allow the remanence of the previous reads to dissipate.

The work done in [1] suggests a time of greater than 10 seconds; however, an exact time was not given. Therefore, to test the remanence effect on our SRAM devices, we enrolled the same SRAM devices with different power cycle times under each testing temperature. The remanence was quantified by calculating the amount of stable and unstable cells in each enrollment. The remanence effect should be stronger with a shorter time between power cycles. Moreover, a strong remanence effect of the previous read should lower the number of unstable cells, creating a false sense of reliability. The results of these tests are found in Figure 3.5

Analyzing Figure 3.5, we observed a notable pattern regarding SRAM enrollments, the time interval between power cycles, and the temperatures. At each temperature, the number of stable cells increased sharply until the time between power cycles reached a certain point; this point is different for each temperature. Surprisingly, the number of unstable cells slightly decreases after that point.

Enrollments with almost zero seconds between power cycles exhibited a strong remanence effect, resulting in fewer unstable cells. This phenomenon can be attributed to the remanence effect, whereby the cells retain information from their previous reads, making them less prone to change. Since it is assumed the remanence effect diminishes over time [113], we can also assume that the amount of unstable cells also increases due to the remanence of previous reads dissipating. The steady decrease of unstable cells can also be attributed to the remanence effect; however, this time, it increases the number of unstable bits. The goal of key generation is to isolate stable cells, so enrollments with more unstable cells can still be used.

The occurrence of 'flaky' cells was observed at different temperatures, providing insights into the remanence effect. At a temperature of 0°C, the number of 'flaky' cells peaked after 21 seconds. At a temperature of 23°C, the peak was observed at 2-3 seconds, while at a temperature of 80°C, it occurred after only 1 second. This indicates that the dispersion of the remanence effect takes a longer time at lower temperatures.

To accommodate these temperature-dependent characteristics, the enrollment process will utilize different power cycle times. Specifically, at a temperature of 0°C, the power cycle time for enrollment will be 21 seconds. At 23°C, it will be 3 seconds, and at 80°C, it will be 1 second. By adjusting the power cycle times based on the temperature, the enrollment process can effectively address the remanence effect and ensure reliable and accurate results.

3.4.1.3 Read count

During the enrollment process, the device takes multiple measurements to classify cells as either stable or unstable. Stable cells are always represented by '0' or '1', indicating that they contain either zero percent or one hundred percent '1's.

One important factor in BER and enrollment is the number of reads conducted. A read refers to a single measurement of each SRAM cell. When more reads are performed during enrollment, the enrollment process becomes more accurate. However, there comes a point where the additional reads do not justify the time taken for enrollment.

Increasing the number of reads increases the chances of identifying a cell as unstable, thus reducing the number of stable cells available. To understand the impact of the read count, we generated enrollments using progressively higher read counts. These enrollments were then used to generate a key, and the BER was subsequently measured. The results of which are shown in Figure 3.6.

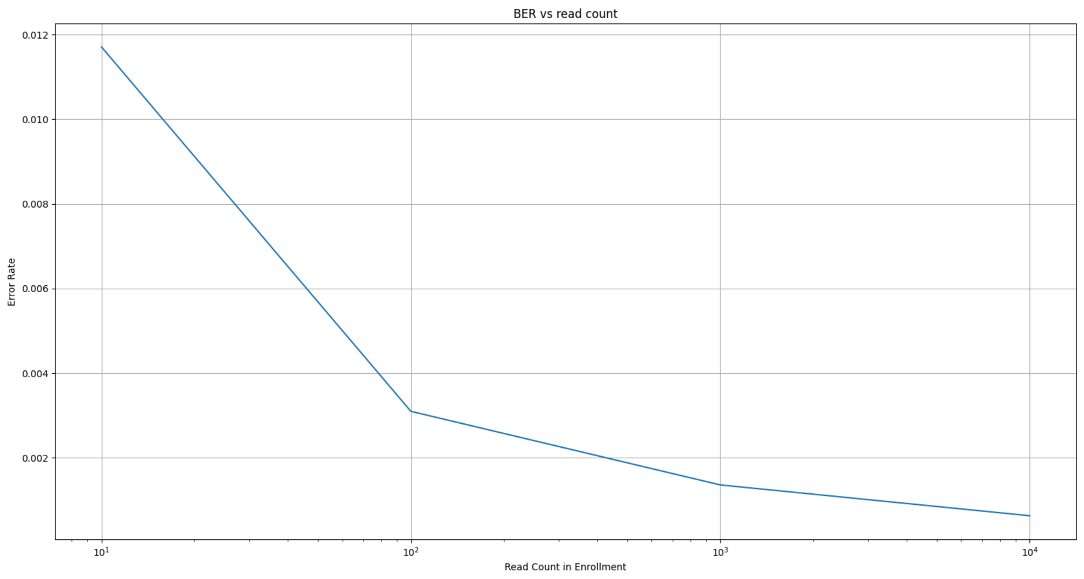


Figure 3.6: BER vs. read count at 23°C

Based on this test, we can conclude that as the read count increases by a factor of ten, the BER decreases by a factor of two. At a read count of 1000, the BER decreased to less than 1%. Given that error-correcting methods like ECC [50] and Response Base Cryptography (RBC) [52] are capable of handling such a small percentage of errors, a read count of 1000 was selected for this research.

3.4.1.4 Quantifying Flakiness

'Flaky' cells refer to cells that are not completely '0' or '1' during enrollment. However, simply categorizing cells based on this criterion does not provide an accurate measure of their flakiness. To better quantify the flakiness of a cell or device, we consider the distance from either 0% or 100% '1'. For instance, an SRAM cell that is 1% '1' is considered less 'flaky' than an SRAM cell that is 40% '1'. To quantify the flakiness of an SRAM device, we take the average distance from 0% or 100% '1's in enrollment. A distance of 0 is considered the most stable, while a distance of 0.50 is considered the most 'flaky'.

3.4.2 SRAM Device Characteristics

After enrolling ten SRAM devices, their properties were examined, particularly their flakiness and propensity to represent either a '0' or a '1'. To illustrate the characteristics of SRAM cells, we categorized all cells,

Average	**0°C**	**23°C**	**80°C**
Strong ‘1’ Percentage	38.476%	36.832%	34.162%
Strong ‘0’ Percentage	35.807%	34.161%	32.723%
‘flaky’ Percentage	25.717%	29.007%	33.115%
Average Distance	0.0317	0.0362	0.0420
A-Cell Percentage	51.229%	51.257%	50.587%
B-Cell Percentage	48.761%	48.732%	49.400%
C-Cell Percentage	0.0010%	0.0109%	0.0131%

Table 3.1: SRAM Characteristics from 0-80°C. Characteristics include noise from the SRAM devices and the additional circuitry to access the cell and, as a result, are not characteristic of the devices themselves.

regardless of their flakiness or stability, into three groups: A-cells, B-cells, and C-cells. A-cells are SRAM cells that exhibit a higher likelihood of being a ‘1’ when queried. B-cells, on the other hand, are more inclined to represent a ‘0’ when queried. Lastly, C-cells do not prefer either state and have an equal chance of being a ‘0’ or a ‘1’. This categorization scheme helps visualize and differentiate the behavior and tendencies of SRAM cells.

In this study, we included ten SRAM devices at each temperature and documented their device characteristics in Table 3.1. From the information presented in the table, we can infer that the devices demonstrate a slight inclination to favor one state, reducing their overall randomness. Based on the classifications used in this study, approximately one-third of the SRAM cells are categorized as ‘flaky’. Additionally, it is observed that the number of ‘flaky’ cells tends to rise as the temperature increases.

3.4.2.1 Temperature Effects

The temperature has a significant impact on the classification of SRAM cells, determining whether they are classified as strong ‘1’s, ‘0’s, or ‘flaky’. On average, the percentage difference in SRAM cell classification between enrollments conducted at 0°C and enrollments conducted at 23°C was 10.09%. Similarly, the difference in classification between enrollments conducted at 23°C and enrollments conducted at 80°C was 18.66%. Notably, the difference in classification between enrollments conducted at 0°C and enrollments conducted at 80°C was 34.10%.

However, the stable cells changing cell types (A-cell, B-cell, or C-cell) were relatively smaller but still significant. On average, the percentage difference in SRAM cell type between enrollments conducted at 0°C and enrollments conducted at 23°C was 0.26%. Similarly, the difference in cell types between enrollments conducted at 23°C and enrollments conducted at 80°C was 2.91%. Significantly, the difference between enrollments conducted at 0°C and enrollments conducted at 80°C was 13.52%.

A heat map of a subsection of an SRAM device’s enrollment that demonstrate the change between temperatures is found in Figure 3.7

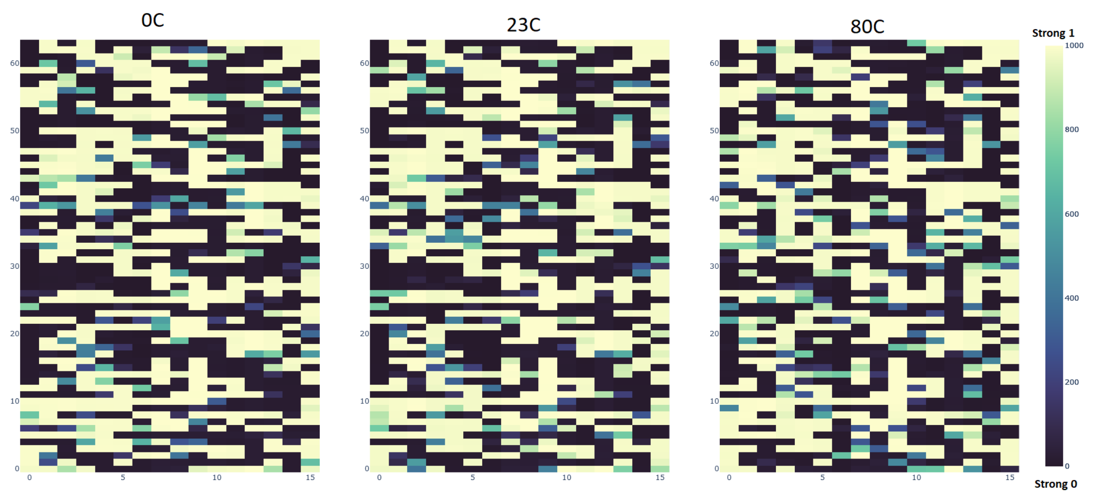


Figure 3.7: Heat map of an ISSI IS64WV6416BLL SRAM Chip section over different temperatures. Notably, there are changes in the cell classifications as temperature changes.

3.4.2.2 Patterns

Even in an SRAM device with an equal number of '1's and '0's in its responses, there may be some unexpected patterns in the SRAM cells that may impact the security and randomness of the PUF. The ISSI IS64WV6416BLL SRAM Chips are arranged in 65536 words, each containing 16 bits. To ensure no repeating patterns, a heat map showing the state of each SRAM cell was generated for each SRAM device. For each of the SRAM devices, there are visible patterns that are distinguishable. The patterns also seem to be independent of temperature. The heat map of an SRAM device can be found in Figure 3.8.

3.4.3 PUF Metrics

To quantify the randomness, uniqueness, reliability, and uniformity of the PUF design, we calculated the entropy density, inter-chip distance, BER, and the average percentage of '1's for responses.

3.4.3.1 Entropy Density

To assess the entropy density of the PUF design, the Shannon entropy formula was used, which can be referenced as Equation 3.1. This formula quantifies the correlation among the bits in the responses obtained from our PUF design. The Shannon entropy has a minimum value of 0, indicating that the input lacks

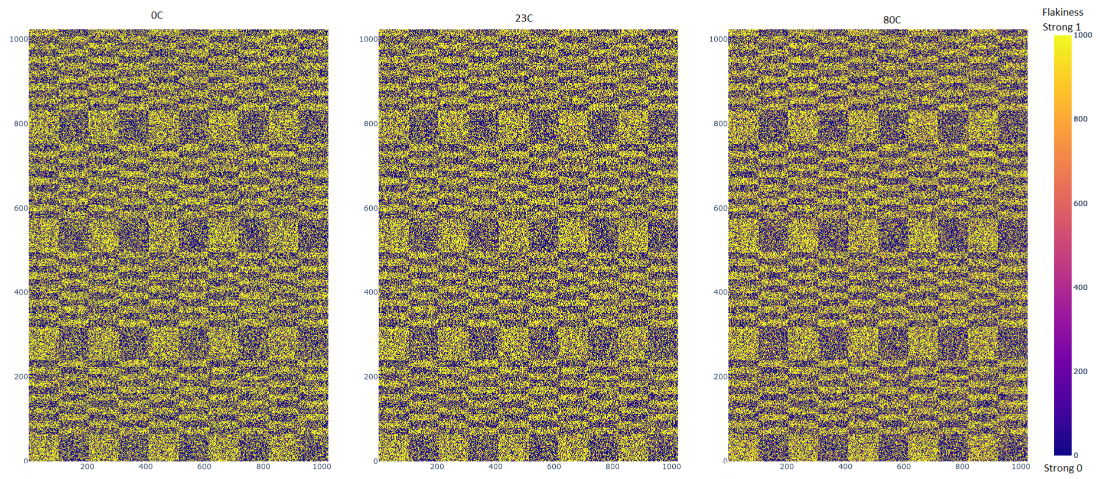


Figure 3.8: Heat map of an ISSI IS64WV6416BLL SRAM Chip over different temperatures. Notable patterns are evident on the SRAM device for each temperature

randomness or variability, while its maximum value is $\log_2(n)$, where n represents the number of possible inputs. In our case, since the input is a binary sequence (with two possible inputs, 0 and 1), n equals 2, and thus the upper bound of the entropy density is 1. This signifies that the input demonstrates maximum randomness or unpredictability.

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{3.1}$$

At each temperature, the client device generated a key consisting of one million bits was calculated using keys that were filtered and unfiltered from unstable bits. Subsequently, the entropy density of each key was calculated. Specifically, at 0°C, the entropy density was found to be 0.9999 for unfiltered keys and 0.9978 for filtered keys. At 23°C, it was 0.9999 for unfiltered keys and 0.9984 for filtered ones. At 80°C, the entropy density was 0.9999 for unfiltered keys and 0.9989 for filtered keys.

In these results, found in Table 3.2, we can see that there is only a minor change in temperature, indicating that the entropy density remains consistently high across different temperatures and is not significantly influenced by temperature variations. Moreover, the entropy density is also slightly reduced when filtering out unstable cells, meaning there is a notable bias in our responses.

Shannon Entropy Density	0°**C**	23°**C**	80°**C**
Unfiltered Responses	0.9999	0.9999	0.9999
Filtered Responses	0.9978	0.9984	0.9989

Table 3.2: The average Shannon Entropy Density of the SRAM XOR PUF at 0°C, 23°C, and 80°C.

Inter-chip Hamming Distance	0°**C**	23°**C**	80°**C**
Unfiltered Responses	0.4982	0.4987	0.4986
Filtered Responses	0.4966	0.4980	0.4971

Table 3.3: The average HD_{inter} of the SRAM XOR PUF at 0°C, 23°C, and 80°C .

3.4.3.2 Uniqueness

To quantify the uniqueness of the PUF, the HD_{inter} was using five different SRAM XOR PUF configurations. The HD_{inter} for each combination of the five configurations was calculated using filtered CRPs and using unfiltered CRPs. The average is the final HD_{inter}. Filtered CRPs excluded the use of 'flaky' cells, while unfiltered did not. Additionally, each test was performed at each testing temperature.

At a temperature of 0°C, the average inter-chip Hamming Distance (HD_{inter}) values were 0.4966 for the filtered data and 0.4982 for the unfiltered data. Similarly, at a temperature of 23°C, the average HD_{inter} values were 0.4980 for the filtered data and 0.4987 for the unfiltered data. Finally, at a temperature of 80°C, the average HD_{inter} values were 0.4971 for the filtered data and 0.4986 for the unfiltered data.

The findings from this study, found in Table 3.3, highlight the impressive metrics of uniqueness exhibited by the SRAM PUF design and devices, as all values are close to the ideal of 50%. Importantly, these metrics remain consistent and independent of temperature variations.

3.4.3.3 Uniformity

To ensure consistency in a PUF, it is desirable for the generated responses to have an equal number of '1's and '0's. However, there is no specific universal function designed solely for computing uniformity in a PUF. Instead, uniformity serves as a general metric that describes the ratio of '1's to '0's in the responses of the PUF system. While the percentages of '1's in the responses of individual SRAM devices may slightly favor '1's, the overall responses from the entire system can differ.

In the case of a binary PUF, an ideal occurrence of '1's would be 50%. To assess and measure the degree of uniformity, a key comprising one million bits was generated at various temperatures, with unstable bits being filtered out. The percentage of '1's in the generated key was then calculated. At each temperature, the percentage of '1's ranged from 49.85%-50.00%, making the uniformity of the responses very close to ideal.

BER for 1 million bit-long key	**At 0°C**	**At 23°C**	**At 80°C**
Using 0 °C Enrollments	0.0037	0.0079	0.1043
Using 23 °C Enrollments	0.0043	0.0025	0.0430
Using 80 °C Enrollments	0.0753	0.0325	0.0020

Table 3.4: The BER of a 1 million-bit-long key at different temperatures. The key size was 2^{20}.

3.4.3.4 BER

As discussed in §2.3.3, the BER is a metric used to assess the reliability of a PUF. It involves generating a sequence of responses on the client device by applying a set of challenges and comparing it to the one generated on the server side using the same set of challenges. The server utilizes enrollment data to generate responses, while the client generates real-time responses using the PUF. The BER is determined by calculating the number of discrepancies between the two response sequences and dividing it by the size of the response sequence.

In essence, the BER is analogous to the intra-chip Hamming Distance (HD_{intra}) in that they both measure the reliability of a PUF. However, the HD_{intra} is a random variable that describes the difference between two PUF responses obtained from the same PUF using the same challenge. On the other hand, the BER compares the response of a PUF to the response generated from the enrollment data.

To evaluate the BER of the proposed design, enrollments were conducted at three different temperatures: 0°C, 23°C, and 80°C. Subsequently, the SRAM XOR PUF was tested at each temperature using the enrollments obtained from all three temperature settings.

In each test, a 1 million-bit long key was generated by utilizing the physical SRAM devices on the client side and the enrollment information on the server side. To assess the quality of the generated keys, the Hamming Distance between the two keys was calculated. This distance measurement was then divided by the total size of the key to determine the BER. The results of the BER tests can be found in Table 3.4.

3.4.4 SRAM XOR PUF Discussion

This section introduced a new SRAM-based PUF that overcomes some limitations of previous designs without compromising the simplicity and accessibility of SRAM. The proposed PUF utilizes two SRAM devices and an XOR gate. To evaluate its performance, ten ISSI IS64WV6416BLL SRAM chips were tested at three different temperatures: 0°C, 23°C, and 80°C. To ensure accurate measurements, GPIO jumper cables were employed to isolate the SRAM device from external circuitry and prevent any external influences.

The PUF was evaluated based on three metrics: Bit Error Rate (BER), Hamming Distance Interpolation (HD_{inter}), and entropy density. The results demonstrated impressive BER performance, with an average BER of 0.2% at the temperature at which it was initially enrolled. However, the average BER increased to $\sim 10\%$ at the temperature farthest from its enrollment temperature. Additionally, the design exhibited a high HD_{inter} value of 0.4966 and a substantial entropy density of 0.9978.

The SRAM XOR PUF exhibited superior performance compared to previous implementations of SRAM PUFs. While another design achieved a lower BER, the SRAM XOR PUF performed equally well or even better in terms of other metrics. A detailed comparison of the PUF metrics between the SRAM XOR Design and previous designs can be seen in Table 3.5.

However, a notable drawback was observed in the SRAM devices, namely the non-uniform distribution of '1's and the patterns in individual SRAM device. This device vulnerability poses a potential security risk to the overall PUF design. These observed patterns may have originated from either the manufacturing process or prolonged usage of the device. Alternatively, they could be influenced by the measurement circuitry in use, including components like the adapter, GPIO jumper cables, or Nucleo144 Board. The patterns in the SRAM devices could render the PUF susceptible to Machine Learning attacks, emphasizing the need for further investigation in this area. Consequently, these factors impact the practicality of utilizing the SRAM device as a PUF, as different circuits may be employed in the field to access the SRAM cells.

This research study fails to address the issue of aging in SRAM cells and does not examine the long-term effects over an extended duration. Aging is a widespread problem that significantly affects the performance of SRAM devices [61], making them less reliable as a suitable choice for a PUF. Moreover, the sample size of devices used in the study is insufficient, as ten devices do not provide an adequate representation of the SRAM device population. To draw more reliable conclusions, a larger sample size of devices should be considered.

3.5 SRAM XOR PUF-based TRNG

Another area that the SRAM XOR PUF can be applied to is random number generation. As discussed in section §2.2, Random Number Generator (RNG)s are a critical component of cryptographic systems. The quality of the random numbers they generate directly correlates to the security they provide, as shown by Goldberg et al. in 1996 [114].

A critical vulnerability of Internet of Things (IoT) is that many of the devices they connect are resource-constrained devices. The lack of resources hinders the ability to implement well-established cryptographic protocols, such as secure random number generation (SRNG) [115, 116].

	PUF Designs			
PUF Metrics	**SRAM XOR PUF**	**Microcontroller SRAM PUF**[1]	**FPGA SRAM PUF**[60]	**Reconfigurable SRAM PUF** [2]
Number of CRPs	n^2	n	n	n
BER or HD_{intra}	NA	0.12	0.02712	NA
BER or HD_{intra} **(fuzzy extractors)**	0.0020-0.0037	6.85×10^{-7}	NA	0.009
HD_{inter}	0.4980	0.4954	0.4806	0.499
Entropy Density H(X)	0.9984	NA	0.9979	NA

Table 3.5: PUF Metrics of previously implemented SRAM PUFs compared to the new design. The term n denotes the number of SRAM cells on the device. Moreover, work in [1, 2] used fuzzy extractors, a mix of filtering unstable cells and error correction code, to improve reliability. The SRAM XOR PUF used fuzzy extractors by just filtering out unstable cells.

A practical approach for generating secure random numbers on devices with limited resources is to utilize a TRNG. TRNGs are well-suited for resource-constrained devices as they do not demand excessive computational power for their operation. Furthermore, TRNGs generate random sequences based on phenomena that are considered inherently random. These phenomena are characterized by being non-deterministic, meaning they cannot be controlled or influenced, resulting in random sequences that are difficult to predict.

However, it is important to note that random sequences generated by TRNGs often require post-processing to eliminate biases and correlations that can decrease the entropy of the generated numbers [28]. This post-processing step is necessary to ensure that the random numbers produced by the TRNG are of high quality and possess the desired level of randomness.

SRAM is a great candidate for TRNGs. Their accessibility renders them simpler to use and less expensive to implement than other memory technology. However, their biggest drawback is that they have a linear number of responses and require additional overhead to generate random numbers. As a result, many SRAM PUF-based TRNG designs choose between optimizing for TRNG or authentication protocols.

This research aims to generate random sequences from a novel SRAM PUF-based TRNG design that has an exponential number of responses and does not require unstable cell information for TRNG. This design can be used for authentication protocols as well as TRNG. The random sequences are generated on a resource-constrained device and pass the NIST Test Suite.

3.5.1 SRAM XOR PUF-based TRNG Design and Hardware Implementation

This design uses the previously mentioned SRAM XOR PUF to generate random bits. By using the two responses from the SRAM PUF, the entropy of the responses should increase. Additionally, different random number generation methods were employed with the addition of a lightweight post-processing method that enhances the randomness and non-deterministic nature of the random sequences. Moreover, when using a pseudo-random number generator to generate challenges, there is a high likelihood that 'flaky' bits will be selected and produce a different output.

3.5.2 Classification and Enrollment

3.5.2.1 Classification

Like the SRAM XOR PUF implementation, the SRAM True Random Number Generator (TRNG) also categorizes SRAM cells. However, this design introduces an additional category, dividing the cells into four groups: '1', '0', 'flaky', and 'super flaky'. Cells classified as '1' are those in which every read during the enrollment process consistently yields a value of one. Conversely, cells categorized as '0' consistently produce zero in every enrollment read. 'Flaky' cells exhibit a mixture of zeros and ones during their enrollment, indicating that they have flipped at least once during the process. By categorizing the SRAM cells in this manner, the SRAM TRNG design enables the differentiation and utilization of cells based on their stable output behavior, ensuring the generation of non-deterministic random numbers.

'Super flaky' cells exhibit a significantly increased likelihood of transitioning between a '1' and a '0'. These cells are identified based on their proximity to stability, specifically when their '1' percentage falls within the range of 48-52%. In the context of the research papers [117] and [118], 'super flaky' cells are further classified into two subcategories: A-cells and B-cells.

A-cells are characterized by a higher probability of being read as a '1', while B-cells have a higher probability of being read as a '0'. It is worth noting that an elevated presence of either type of 'super flaky' cell can impact the generation of random numbers, particularly when they are intentionally targeted or exploited. The distinction between A-cells and B-cells within the 'super flaky' category is significant as it helps to understand the behavior and potential biases associated with these cells, ultimately affecting the quality and reliability of the generated random numbers.

3.5.2.2 Enrollment

The enrollment process for this design follows the same procedure as the SRAM XOR PUF implementation. The device is powered on, read, and then powered down, and this sequence is repeated a specified number of times, denoted as r. It has been observed that larger values of r result in more accurate enrollments.

An important consideration during enrollment is the duration between powering down and powering up for the subsequent read. Theoretically, a shorter interval between power cycles amplifies the remanence effects. Remanence effects arise from the lingering influence of previous reads, which can introduce inaccuracies during enrollment. If left unaddressed, these remanence effects can lead to fewer ‘flaky’ bits being properly represented in the enrollment process, as the SRAM cells tend to retain their previous state upon startup. Consequently, ‘flaky’ bits may be falsely represented as strong ‘1’s or ‘0’s in the presence of remanence effects.

To mitigate remanence effects, a longer duration between power cycles is employed. However, this extended time period not only reduces remanence effects but also slows down the enrollment and random number generation processes. In the SRAM XOR PUF implementation, the wait time for room temperature operation was 2-3 seconds. However, it is important to investigate whether this delay time equally applies to ‘super flaky’ cells, as their behavior may differ from other cell types.

To examine the impact of remanence effects, we conducted experiments by varying the delay time between power cycles during the enrollment process. We tested delay times ranging from 0 to 1 second, with increments of 100 milliseconds. Additionally, we investigated delay times from 1 to 20 seconds, with increments of one second. The results, illustrated in Fig. 3.9, revealed that the number of ‘super flaky’ cells increased steadily until reaching a peak at around 2-3 seconds, similar to the behavior observed for ‘flaky cells’.

Based on these findings, we determined that a delay time of 3 seconds between power cycles yielded favorable outcomes for enrollments, subsequent tests, and random number generation. Hence, we adopted a consistent delay time of 3 seconds for the remainder of our experiments, ensuring reliable and accurate performance in the presence of remanence effects.

3.5.3 TRNG Protocols

For this work, we used two different types of TRNG protocols. The first protocols use a standard PRNG. These protocols require no additional overhead and are very easy to implement. The second protocols only use ‘flaky’ cells to generate their responses. These protocols require some overhead to select specifically ‘flaky’ cells and some power-on cycles to generate responses. The second protocol type is used as a reference point to compare to the first.

3.5.3.1 Cell pairing

In the cell pairing (CP) technique, we follow a process where we randomly pick one SRAM cell from each SRAM device and combine their responses using XOR operation to create a random bit. This procedure resembles the one used in selecting bits for the SRAM XOR PUF. However, unlike the SRAM XOR PUF, the responses obtained from the cell pairing process are not filtered to remove unreliable or unstable bits. The total number of potential challenges that can be generated using cell pairing equals the square of the

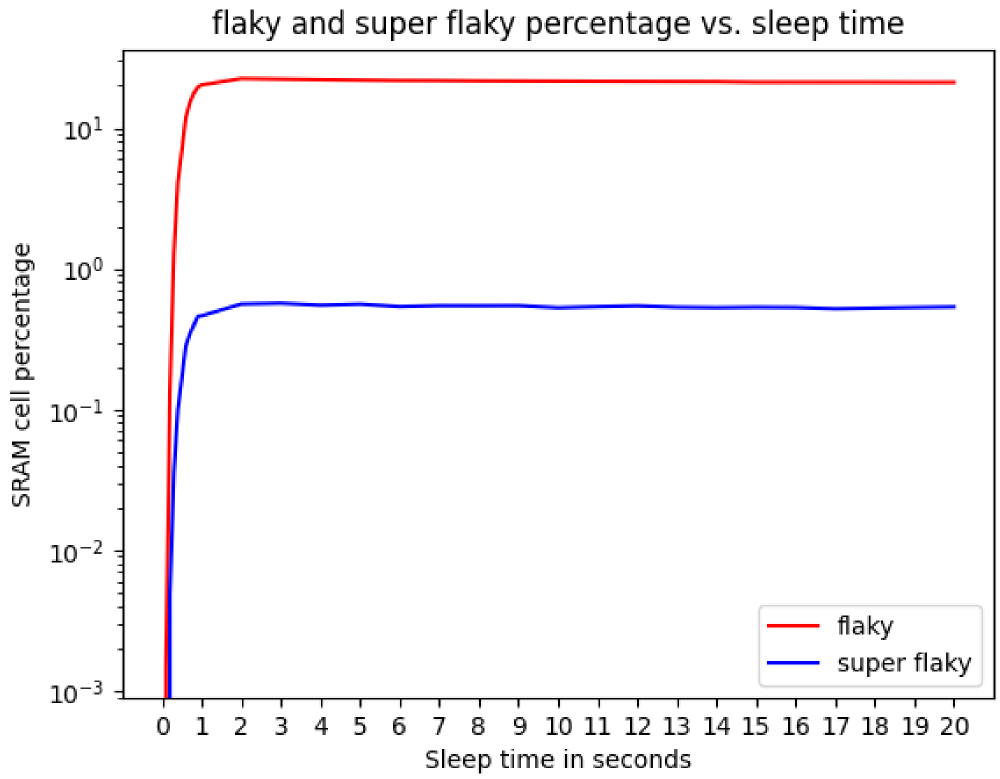


Figure 3.9: 'Flaky' and 'Super Flaky' cell percentage vs. delay time

number of SRAM cells on an SRAM device, n^2. Therefore, CP offers the highest possible challenges among the considered techniques. To get a visual representation of this protocol, refer to Figure 3.10 for a block diagram.

3.5.3.2 Word pairing

In word pairing (WP), we randomly select SRAM words from each device. SRAM words are internally XORed to produce a bit, and those bits are XORed to produce a final bit. The number of possible responses with word pairing is $(\frac{n}{16})^2$. A block diagram of this protocol is illustrated in Fig. 3.11

3.5.3.3 Ternary cell pairing

Ternary cell pairing (TCP) is similar to CP, except that every cell selected is classified as a 'super flaky' bit (see §3.5.2). To select 'super flaky' bits, the server will send the 'super flaky' locations to the client before generation.

3.5.3.4 Ternary word pairing

Ternary word pairing (TWP) is like WP, with the difference being that every word selected is 'super flaky'. A 'super flaky' word is classified as a word that contains at least one SRAM cell that is classified as 'super flaky'. The server will send 'super flaky' locations to the client before random number generation to select 'super flaky' bits. A word is classified as 'flaky' if it has one 'super flaky' or 'flaky' bit.

3.5.4 Post-processing

TRNGs rely on natural phenomena to generate "truly" random sequences. However, due to biases or disturbances such as electromagnetic interference (EMI)[119], the random sequences generated are often not random enough to pass the NIST Test Suite [28]. In such cases, a post-processing stage is added to sufficiently enhance the randomness of the random sequences such that the criteria of the NIST Statistical Randomness Test are satisfied.

An internal XOR compiler is a lightweight post-processing method proposed in [117, 120]. The XOR gate is a Boolean logic gate that generates a '1' if both inputs are the same and a '0' if they are different. The internal XOR compiler takes p bits and XORs them to generate a single bit. An XOR with more than two inputs generates a '1' if there are an odd number of '1's and generates a '0' otherwise. In the research presented in [117, 120], it was discovered that entropy increased as p increased. On the other hand, if there were insufficient initial entropy, the internal XOR compiler would not enhance the randomness sufficiently

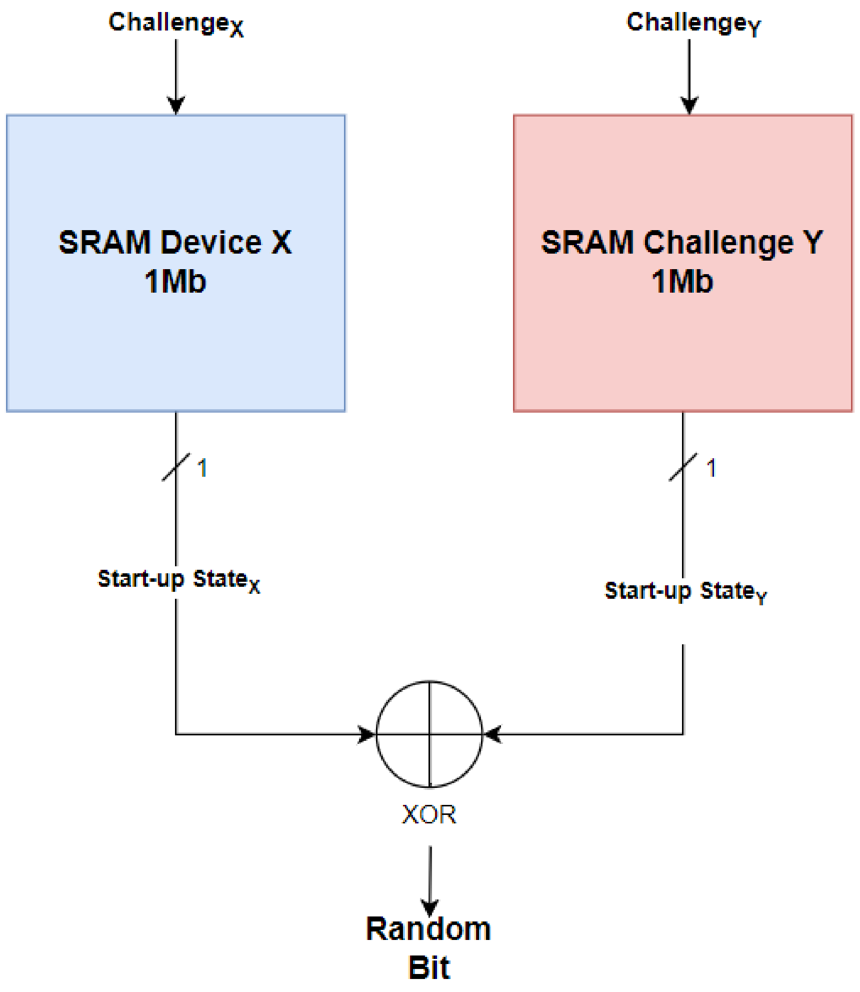


Figure 3.10: SRAM CP Protocol

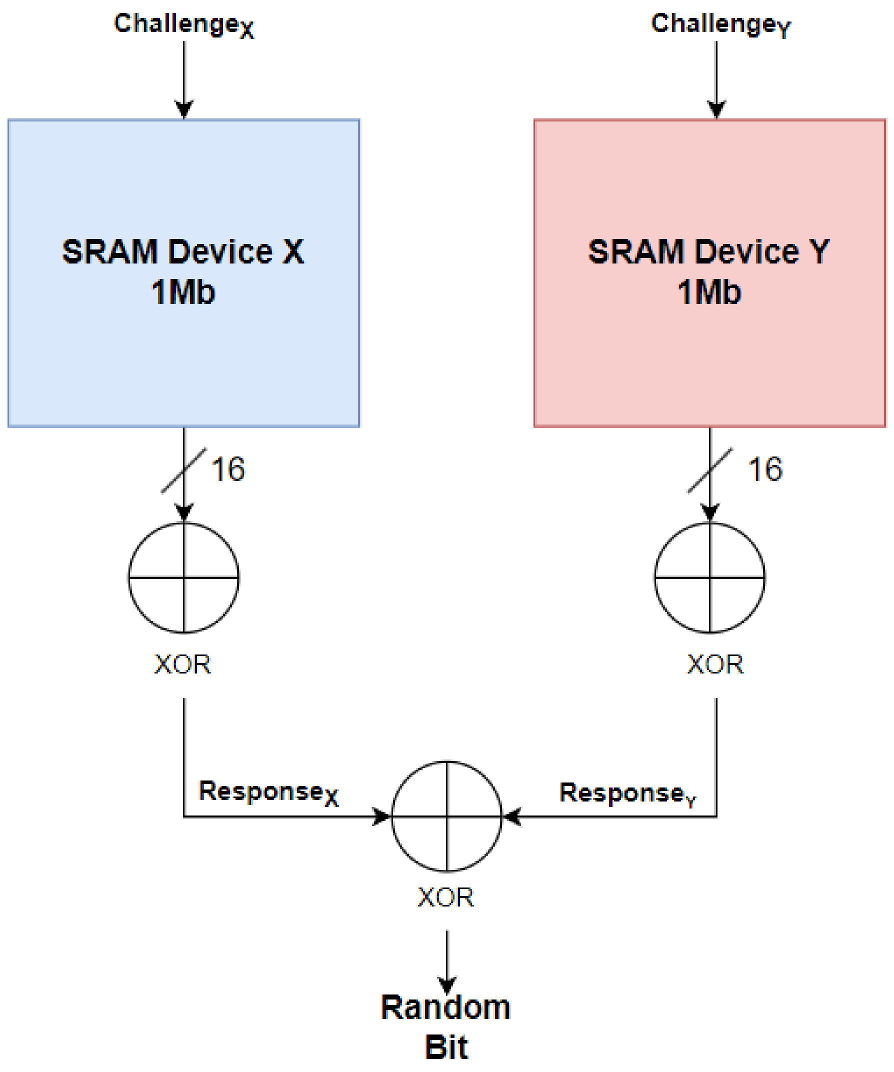


Figure 3.11: SRAM WP Protocol

to pass statistical randomness tests (even with increasing p). Overall, the internal XOR compiler increased the entropy while reducing the size of the random sequences.

A PRNG XOR compiler XORs True Random Number (TRN)s with Pseudo-Random Number (PRN)s to enhance the randomness of the TRNs. This post-processing method was shown to enhance the randomness of random sequences without sacrificing the size of the TRN. This compiler will XOR the TRN with a PRN through c cycles. As c increases, the randomness of the TRN should as well.

The internal XOR compiler serves as a valuable tool to augment the non-deterministic characteristics of random numbers. When only a small percentage of bits are flipped in the input, the XOR compiler can increase the proportion of flipped bits in the final number, thereby enhancing the randomness.

Conversely, the Pseudo-Random Number Generator (PRNG) XOR compiler has the capability to alter all the bits in a random number sequence completely. However, it is important to note that due to the reversible nature of the XOR operation, if an adversary possesses knowledge of the PRNs employed to XOR the true random sequence, they would be able to reverse the process and recover the original unprocessed number.

To leverage the inherent non-deterministic properties of our responses, further explained in §3.5.5, we employ a PRNG XOR compiler to amplify the randomness. This approach proves advantageous as it is cost-effective to implement and does not compromise the length of our generated random numbers.

To further enhance the randomness, we initialize the PRNG with the system time as the seed before performing post-processing. By incorporating the system time as a seed, we introduce unpredictability and randomness to the generated numbers, strengthening their overall randomness and reliability.

3.5.5 Non-deterministic

When randomly selecting cells in CP and WP, there is a small percentage of 'flaky' and 'super flaky' cells in the selected cells. These 'flaky' cells are extremely important in producing non-deterministic results. On average, every SRAM device has around ∼22% 'flaky' cells and ∼0.5% 'super flaky' cells. Of those 'super flaky' cells, there are ∼0.2% A-cells, ∼0.2% B cells, and ∼0.1% cells with no preference.

XORing the responses increases the likelihood of selecting 'flaky' cells, thus increasing the non-deterministic nature of the TRNG design. We tested the randomly selected challenges' 'flaky' and 'super flaky' percentages by generating one million random addresses with CP and WP. As shown in table 3.6, a sizeable amount of 'flaky' and 'super flaky' cells are found in each protocol. Although we see a much greater 'flaky' percentage in WP, the amount found in CP is sufficient to generate different random sequences given the same challenges.

Additionally, to verify that the TRNG design is non-deterministic, we calculated the intra-chip Hamming

Cell Percentages	Chip 1	Chip 2	CP (1 million challenges)	WP (1 million challenges)
Flaky (Including super flaky)	21.9354%	21.6110%	38.9163%	99.9394%
Super flaky	0.5618%	0.5615%	1.0975%	16.5618%

Table 3.6: 'Flaky' and 'super flaky' percentages of SRAM devices and responses of TRNG protocols

Intra-chip Hamming Distance	CP	TCP	WP	TWP
Before Post Processing	0.086853	0.422948	0.474309	0.500066
After PRNG XOR Compiling 1 Cycle	0.500036	0.499156	0.500046	0.599308

Table 3.7: Inter-chip Hamming distance of different random sequences generated with the same challenges for each protocol (**Ideal = 0.50**)

distance of random sequences before and after post-processing. The intra-chip Hamming distance, denoted as HD_{intra}. As mentioned in §3.5.4, the PRNG used for post-processing was seeded with the system time. The HD_{intra} is the hamming distance between two sequences of the same length. Ideally, random sequences generated with the same challenge should have an HD_{intra} value of 0.50, as it should be non-deterministic. We generated three different 16kB keys for each protocol.

As shown in table 3.7, the HD_{intra} for each protocol after post-processing was close to the ideal value of 0.50. The random numbers generated with only ternary bits showed greater initial randomness than numbers generated using PRNG. Before post-processing, the WP and TWP protocols showed to be closest to 0.50 at ∼0.47 and ∼0.50, while the CP and TCP protocols were further from the ideal value at ∼0.08 and ∼0.42.

3.5.6 Software Implementation

Enrollment was performed on a personal computer (PC) using `python3` and `pandas`. Reads were collected, stored in a data frame, and processed accordingly. Using `pandas` facilitated extracting 'flaky' cells within a given range (i.e., 'super flaky' = 48-52% `'1'`s).

The low-power device was driven by `C++` code which pseudo-randomly generated challenges using the built-in `rand()` function from the standard library. The function was seeded with the default value for challenge selection.

For the ternary protocols (see §3.5.3), 'flaky' addresses were determined through enrollment and sent to the device over serial communication. From there, 'flaky' addresses were pseudo-randomly selected within the device and used to produce CRPs.

In this prototyping phase, we sent raw responses to the PC over serial, which allowed us to process them at will. With the raw responses, we performed the various tests with and without post-processing the output (see Section 3.5.3).

With this implementation, we produced random bits using CP and TCP at 666 kbps, and WP and TWP at 1.6 Mbps on the client device. The ternary protocols did require the server to have previously sent the 'flaky' locations; however, requirement this only added 30 seconds once to set up the device as TRNG prior to subsequent random number generations. The latencies due to the ternary protocols during random number generation are negligible. For this research, the post-processing was done using the `random` module in `python3` on the server side.

3.5.7 NIST Results

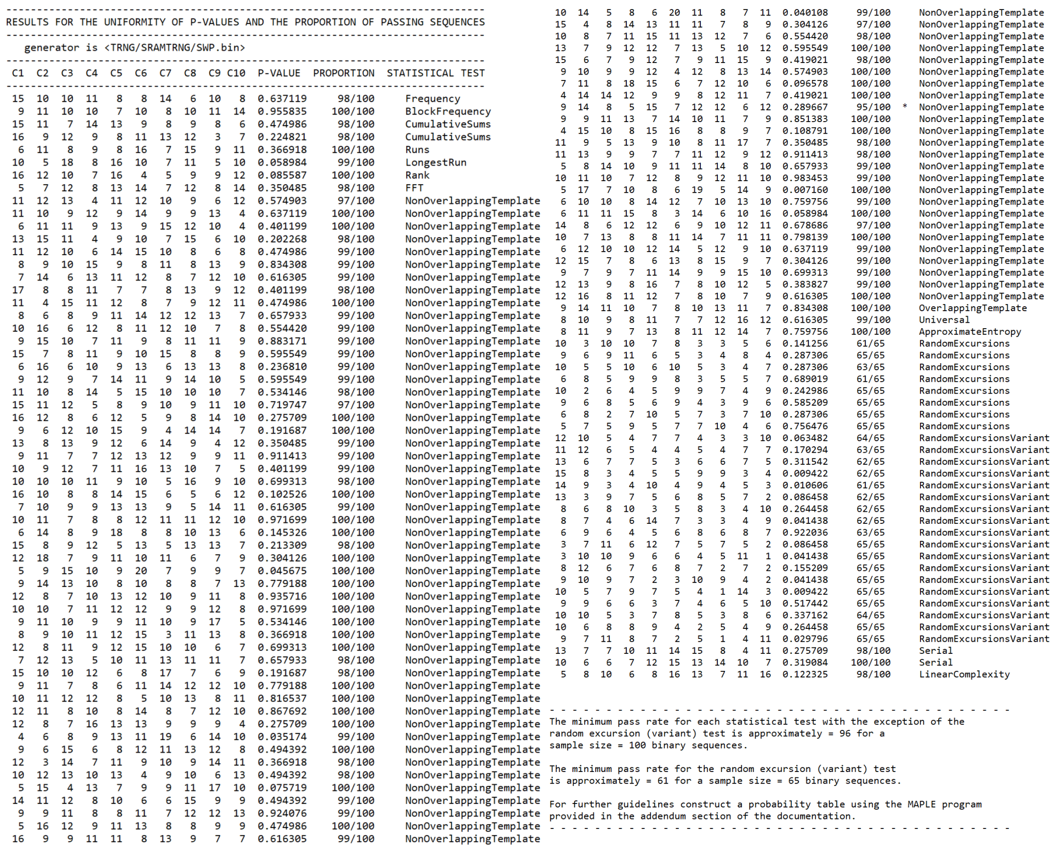
We measured the quality of the random numbers generated by our proposed SRAM TRNG design with the NIST statistical test suite for random numbers [29]. This suite subjects the bit sequences given through multiple tests and produces a report as shown in Fig. 3.12. As recommended by NIST, we use the full fifteen tests the suite provides. We generated 2^{27} ($\sim$ 134 million) bits from our TRNG for each protocol presented (see §3.5.3). Those streams were split into one million bit-long segments. The sequences met the length criterion of the NIST suite, thus allowing us to subject the streams through all 15 tests. We set the alpha value to 0.001 instead of the default alpha value of 0.01. Therefore, if more than one out of 1000 tests fail, the whole test fails.

The following figures exhibit the resulting rejection rates and indicate whether or not the streams ultimately pass or fail the test at the given alpha configuration.

As shown in table 3.8, the CP and TCP protocols failed the NIST Test Suite, while the WP and TWP protocols passed before post-processing, while all protocols passed after post-processing. This shows that using a PRNG to generate challenges provides better or as much initial randomness when compared to specifically using ternary cells. Simple and quick-to-perform XOR operations are enough to enhance the randomness of the data streams generated from the TRNG, demonstrating the natural randomness of the design.

3.5.8 SRAM XOR PUF-based TRNG Discussion

In this work we proposed a novel SRAM PUF-based TRNG design using the SRAM XOR PUF to produce bits. The proposed design produces truly random numbers and keys for authentication protocols. Random sequences produced by this design have been proven to be non-deterministic and require light post-processing



```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <TRNG/SRAMTRNG/SWP.bin>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 15  10  10  11   8   8  14   6  10   8  0.637119     98/100     Frequency
  9  11  10  10   7  10   8  10  11  14  0.955835    100/100     BlockFrequency
 15  11   7  14  13   9   8   9   8   6  0.474986     98/100     CumulativeSums
 16   9  12   9   8  11  13  12   3   7  0.224821     98/100     CumulativeSums
  6  11   8   9   8  16   7  15   9  11  0.366918    100/100     Runs
 10   5  18   8  16  10   7  11   5  10  0.058984     99/100     LongestRun
 16  12  10   7  16   4   5   9   9  12  0.085587    100/100     Rank
  5   7  12   8  13  14   7  12   8  14  0.350485     98/100     FFT
 11  12  13   4  11  12  10   9   6  12  0.574903     97/100     NonOverlappingTemplate
 11  10   9  12   9  14   9   9  13   4  0.637119    100/100     NonOverlappingTemplate
  6  11  11   9  13   9  15  12  10   4  0.401199    100/100     NonOverlappingTemplate
 13  15  11   4   9  10   7  15   6  10  0.202268     98/100     NonOverlappingTemplate
 11  12  10   6  14  15  10   8   6   8  0.474986     99/100     NonOverlappingTemplate
  8   9  10  15   9   8  11   8  13   9  0.834308     99/100     NonOverlappingTemplate
  7  14   6  13  11  12   8   7  12  10  0.616305     99/100     NonOverlappingTemplate
 17   8   8  11   7   7   8  13   9  12  0.401199     98/100     NonOverlappingTemplate
 11   4  15  11  12   8   7   9  12  11  0.474986    100/100     NonOverlappingTemplate
  8   6   8   9  11  14  12  12  13   7  0.657933     99/100     NonOverlappingTemplate
 10  16   6  12   8  11  12  10   7   8  0.554420     99/100     NonOverlappingTemplate
  9  15  10   7  11   9   8  11  11   9  0.883171     99/100     NonOverlappingTemplate
 15   7   8  11   9  10  15   8   8   9  0.595549     99/100     NonOverlappingTemplate
  6  16   6  10   9  13   6  13  13   8  0.236810     99/100     NonOverlappingTemplate
  9  12   9   7  14  11   9  14  10   5  0.595549     99/100     NonOverlappingTemplate
 11  10   8  14   5  15  10  10  10   7  0.534146     98/100     NonOverlappingTemplate
 15  11  12   5   8   9  10   9  11  10  0.719747     97/100     NonOverlappingTemplate
 16  12   8   6  12   5   9   8  14  10  0.275709    100/100     NonOverlappingTemplate
  9   6  12  10  15   9   4  14  14   7  0.191687    100/100     NonOverlappingTemplate
 13   8  13   9  12   6  14   9   4  12  0.350485     99/100     NonOverlappingTemplate
  9  11   7   7  12  13  12   9   9  11  0.911413     99/100     NonOverlappingTemplate
 10   9  12   7  11  16  13  10   7   5  0.401199     99/100     NonOverlappingTemplate
 10  10  10  11   9  10   5  16   9  10  0.699313     98/100     NonOverlappingTemplate
 16  10   8   8  14  15   6   5   6  12  0.102526    100/100     NonOverlappingTemplate
  7  10   9   9  13  13   9   5  14  11  0.616305     99/100     NonOverlappingTemplate
 10  11   7   8   8  12  11  11  12  10  0.971699    100/100     NonOverlappingTemplate
  6  14   8   9  18   8   8  10  13   6  0.145326    100/100     NonOverlappingTemplate
 15   8   9  12   5  13   5  13  13   7  0.213309     98/100     NonOverlappingTemplate
 12  18   7   9  11  10  11   6   7   9  0.304126    100/100     NonOverlappingTemplate
  5   9  15  10   9  20   7   9   9   7  0.045675    100/100     NonOverlappingTemplate
  9  14  13  10   8  10   8   8   7  13  0.779188    100/100     NonOverlappingTemplate
 12   8   7  10  13  12  10   9  11   8  0.935716    100/100     NonOverlappingTemplate
 10  10   7  11  12  12   9   9  12   8  0.971699    100/100     NonOverlappingTemplate
  9  11  10   9   9  11  10   9  17   5  0.534146    100/100     NonOverlappingTemplate
  8   9  10  11  12  15   3  11  13   8  0.366918    100/100     NonOverlappingTemplate
 12   8  11   9  12  15  10  10   6   7  0.699313    100/100     NonOverlappingTemplate
  7  12  13   5  10  11  13  11  11   7  0.657933     98/100     NonOverlappingTemplate
 15  10  10  12   6   8  17   7   6   9  0.191687     98/100     NonOverlappingTemplate
  9  11   7   8   6  11  14  12  12  10  0.779188    100/100     NonOverlappingTemplate
 10  11  12  12   8   5  10  13   8  11  0.816537    100/100     NonOverlappingTemplate
 12  11   8  10   8  14   8   7  12  10  0.867692    100/100     NonOverlappingTemplate
 12   8   7  16  13  13   9   9   9   4  0.275709    100/100     NonOverlappingTemplate
  4   6   8   9  13  11  19   6  14  10  0.035174     99/100     NonOverlappingTemplate
  9   6  15   6   8  12  11  13  12   8  0.494392    100/100     NonOverlappingTemplate
 12   3  14   7  11   9  10   9  14  11  0.366918     98/100     NonOverlappingTemplate
 10  12  13  10  13   4   9  10   6  13  0.494392     98/100     NonOverlappingTemplate
  5  15   4  13   7   9   9  11  17  10  0.075719    100/100     NonOverlappingTemplate
 14  11  12   8  10   6   6  15   9   9  0.494392     99/100     NonOverlappingTemplate
  9   9  11   8   8  11   7  12  12  13  0.924076     99/100     NonOverlappingTemplate
  5  16  12   9  11  13   8   8   9   9  0.474986    100/100     NonOverlappingTemplate
 16   9   9  11  11   8  13   9   7   7  0.616305     99/100     NonOverlappingTemplate
 10  14   5   8   6  20  11   8   7  11  0.040108     99/100     NonOverlappingTemplate
 15   4   8  14  13  11  11   7   8   9  0.304126     97/100     NonOverlappingTemplate
 10   8   7  11  15  11  13  12   7   6  0.554420     98/100     NonOverlappingTemplate
 13   7   9  12  12   7  13   5  10  12  0.595549    100/100     NonOverlappingTemplate
 15   6   7   9  12   7   9  11  15   9  0.419021     98/100     NonOverlappingTemplate
  9  10   9   9  12   4  12   8  13  14  0.574903    100/100     NonOverlappingTemplate
  7  11   8  18  15   6   7  12  10   6  0.096578    100/100     NonOverlappingTemplate
  4  14  14  12   9   9   8  12  11   7  0.419021    100/100     NonOverlappingTemplate
  9  14   8   5  15   7  12  12   6  12  0.289667     95/100  *  NonOverlappingTemplate
  9   9  11  13   7  14  10  11   7   9  0.851383    100/100     NonOverlappingTemplate
  4  15  10   8  15  16   8   8   9   7  0.108791    100/100     NonOverlappingTemplate
 11   9   5  13   9  10   8  11  17   7  0.350485     98/100     NonOverlappingTemplate
 11  13   9   9   7   7  11  12   9  12  0.911413     98/100     NonOverlappingTemplate
  5   8  14  10   9  11  11  14   8  10  0.657933     99/100     NonOverlappingTemplate
 10  11  10   7  12   8   9  12  11  10  0.983453     99/100     NonOverlappingTemplate
  5  17   7  10   8   6  19   5  14   9  0.007160    100/100     NonOverlappingTemplate
  6  10  10   8  14  12   7  10  13  10  0.759756     99/100     NonOverlappingTemplate
  6  11  11  15   8   3  14   6  10  16  0.058984    100/100     NonOverlappingTemplate
 14   8   6  12  12   6   9  10  12  11  0.678686     97/100     NonOverlappingTemplate
 10   7  13   8   8  11  14   7  11  11  0.798139    100/100     NonOverlappingTemplate
  6  12  10  10  12  14   5  12   9  10  0.637119     99/100     NonOverlappingTemplate
 12  15   7   8   6  13   8  15   9   7  0.304126     99/100     NonOverlappingTemplate
  9   7   9   7  11  14   9   9  15  10  0.699313     99/100     NonOverlappingTemplate
 12  13   9   8  16   7   8  10  12   5  0.383827     99/100     NonOverlappingTemplate
 12  16   8  11  12   7   8  10   7   9  0.616305    100/100     NonOverlappingTemplate
  9  14  11  10   7   8  10  13  11   7  0.834308    100/100     OverlappingTemplate
  8  10   9   8  11   7   7  12  16  12  0.616305     99/100     Universal
  8  11   9   7  13   8  11  12  14   7  0.759756    100/100     ApproximateEntropy
 10   3  10  10   7   8   3   3   5   6  0.141256     61/65      RandomExcursions
  9   6   9  11   6   5   3   4   8   4  0.287306     65/65      RandomExcursions
 10   5   5  10   6  10   5   3   4   7  0.287306     63/65      RandomExcursions
  6   8   5   9   9   8   3   5   5   7  0.689019     61/65      RandomExcursions
 10   2   6   4   5   9   9   7   4   9  0.242986     65/65      RandomExcursions
  9   6   8   5   6   9   4   3   9   6  0.585209     65/65      RandomExcursions
  6   8   2   7  10   5   7   3   7  10  0.287306     65/65      RandomExcursions
  5   7   5   9   5   7   7  10   4   6  0.756476     65/65      RandomExcursions
 12  10   5   4   7   7   4   3   3  10  0.063482     64/65      RandomExcursionsVariant
 11  12   6   5   4   4   5   4   7   7  0.170294     63/65      RandomExcursionsVariant
 13   6   7   7   5   3   6   6   7   5  0.311542     62/65      RandomExcursionsVariant
 15   8   3   4   5   5   9   9   3   4  0.009422     62/65      RandomExcursionsVariant
 14   9   3   4  10   4   9   4   5   3  0.010606     61/65      RandomExcursionsVariant
 13   3   9   7   5   6   8   5   7   2  0.086458     62/65      RandomExcursionsVariant
  8   6   8  10   3   5   8   3   4  10  0.264458     62/65      RandomExcursionsVariant
  8   7   4   6  14   7   3   3   4   9  0.041438     62/65      RandomExcursionsVariant
  6   9   6   4   5   6   8   6   8   7  0.922036     63/65      RandomExcursionsVariant
  3   7  11   6  12   7   5   7   5   2  0.086458     65/65      RandomExcursionsVariant
  3  10  10   9   6   6   4   5  11   1  0.041438     65/65      RandomExcursionsVariant
  8  12   6   7   6   8   7   2   7   2  0.155209     65/65      RandomExcursionsVariant
  9  10   9   7   2   3  10   9   4   2  0.041438     65/65      RandomExcursionsVariant
 10   5   7   9   7   5   4   1  14   3  0.009422     65/65      RandomExcursionsVariant
  9   9   6   6   3   7   4   6   5  10  0.517442     65/65      RandomExcursionsVariant
 10  10   5   3   7   8   5   3   8   6  0.337162     64/65      RandomExcursionsVariant
 10   6   8   8   9   4   2   5   4   9  0.264458     65/65      RandomExcursionsVariant
  9   7  11   8   7   2   5   1   4  11  0.029796     65/65      RandomExcursionsVariant
 13   7   7  10  11  14  15   8   4  11  0.275709     98/100     Serial
 10   6   6   7  12  15  13  14  10   7  0.319084    100/100     Serial
  5   8  10   6   8  16  13   7  11  16  0.122325     98/100     LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 61 for a sample size = 65 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 3.12: NIST Test Suite results for post-processed WP random sequences

Table 3.8: Final SRAM TRNG NIST Test Suite Results. The NIST Test Suite evaluated all TRNG protocols before and after post-processing

TRNG Protocol	Cell Pairing	Ternary Cell Pairing	Word Pairing	Ternary Word Pairing
No post-processing	Fail	Fail	Pass	Pass
PRN XOR 1 Cycle	Pass	Pass	Pass	Pass
PRN XOR 2 Cycles	Pass	Pass	Pass	Pass
PRN XOR 3 Cycles	Pass	Pass	Pass	Pass

and no intervention from the server. We used TRNG protocols: CP, WP, TCP, and TWP. After producing each random sequence, selected for demonstration purposes, a lightweight PRNG XOR compiler for post-processing to enhance randomness was utilized.

Each TRNG protocol passed the NIST Test suite after post-processing and was shown to be non-deterministic. The random sequences produced were shown to differ by 50% when the same challenges were used. Past TRNG designs have either required overhead to select 'flaky' cells or have been designed to be only used as TRNGs. However, we tested protocols that generated challenges with a simple PRNG found in standard libraries and protocols that only used unstable cells as a reference point. The protocols that used the PRNG were on par with those that used only unstable cells in randomness before and after post-processing. Thus, proving that no overhead is required to produce non-deterministic random sequences (such as in similar SRAM PUF designs).

While the CP and TCP protocols passed the NIST Test Suite after a lightweight post-processing stage, the WP protocol showed much greater initial randomness and more unpredictability when using the same challenges. As such, the WP protocol is ideal for use as a TRNG, while the CP protocol (filtered for unstable cells) is better suited for authentication protocols.

Chapter 4

MRAM

4.1 Introduction

This chapter focuses on the research conducted on the implementation of a Magnetoresistive random-access memory (MRAM) Ternary Addressable Physically Unclonable Function (TAPUF) capable of generating reliable bits for use in authentication and key generation protocols and non-deterministic "truly random" bits.

The TAPUF utilizes the unique resistance properties of MRAM cells to generate digital fingerprints that can be effectively utilized in cryptographic protocols. By exploiting the variations in resistance values between MRAM cells, we can generate reliable cryptographic keys and truly random numbers, which enhance the security of cryptographic systems by protecting them against certain attacks.

To assess the performance of the TAPUF, several tests were conducted. These tests included evaluating the variations between cells and within cells, measuring inter-distance effects, assessing the Bit-error rate (BER), and evaluating the impact of temperature variations. Additionally, the quality of the generated true random sequences was evaluated using the National Institute of Technology (NIST) Testing Suite, which is a widely recognized set of tests for randomness.

All experiments were conducted using a low-power client device to replicate real-world scenarios. The results demonstrate that the proposed MRAM-based TAPUF exhibits exceptional scalability, energy efficiency, and reliability. Furthermore, the "truly random" sequences could pass the rigorous tests of the NIST Testing Suite, confirming their high quality.

4.1.1 Motivation

The Internet of Things (IoT) is an emerging communication network that aims to connect many devices. A critical vulnerability of IoT is they connect many power-constrained devices. Power-constrained systems, limited in both power consumption and computational power, are often the weakest link in security systems as they present many vulnerabilities such as key generation, storage, and distribution [115, 116]. One of the significant advantages of Physically Unclonable Function (PUF)s is that they are well-suited for power-constrained systems.

The field of cryptography has explored various types of memory-PUFs, and among them, the MRAM PUF stands out as a particularly promising option. The MRAM PUF is a low-power, non-volatile, high-endurance memory rated for space operation, radiation hard, and is touted to become the standard for RAM. While numerous MRAM PUFs have been proposed, many have not been tested on low-power client devices and have a limited number of Challenge-response pair (CRP)s.

In Section §2.4.2.2, it has been observed that many proposed MRAM-based PUFs have only been studied through simulations and have notable weaknesses. This chapter introduces an innovative design for a PUF called MRAM-based TAPUF. Unlike traditional PUFs that use bits (with two possible states), this design utilizes trits (with three possible states) to enhance its capabilities.

The implementation of this MRAM TAPUF leverages commercially available MRAM devices and is specifically optimized to operate on low-power client devices. This aspect highlights its compatibility with systems that have limited power resources.

The main objective of this research is to gather essential metrics related to the responses of the MRAM TAPUF. These metrics include inter-cell variation, intra-cell variation, medians, and ranges. By conducting these measurements, our study aims to provide a comprehensive understanding of the performance and characteristics of the MRAM PUF.

4.2 MRAM TAPUF Design

MRAM cells have unique physical variations from other MRAM cells, even if written to a common state. The cell-to-cell variations in MRAM cells are in the electrical resistance, which is a factor of small physical variations created during manufacturing. This design will leverage the electrical resistances of two MRAM cells from two different MRAM devices to generate responses. The number of CRPs in this design is n^2 (n is the number of MRAM cells on each device). A diagram of this PUF design can be found in Figure 4.1.

The electrical resistances from a pair of MRAM cells are extracted and used to generate a ternary response (e.g., '0,' 'X,' '1'). In this process, the locations of the MRAM cells in the pair are the challenge, and the comparison of the cells' resistance is the response. Additionally, the resistance from each MRAM cell is extracted and then converted to voltage for enrollment and comparison. This is discussed further in section 4.3.

This design has two main steps: Enrollment and Bit Generation.

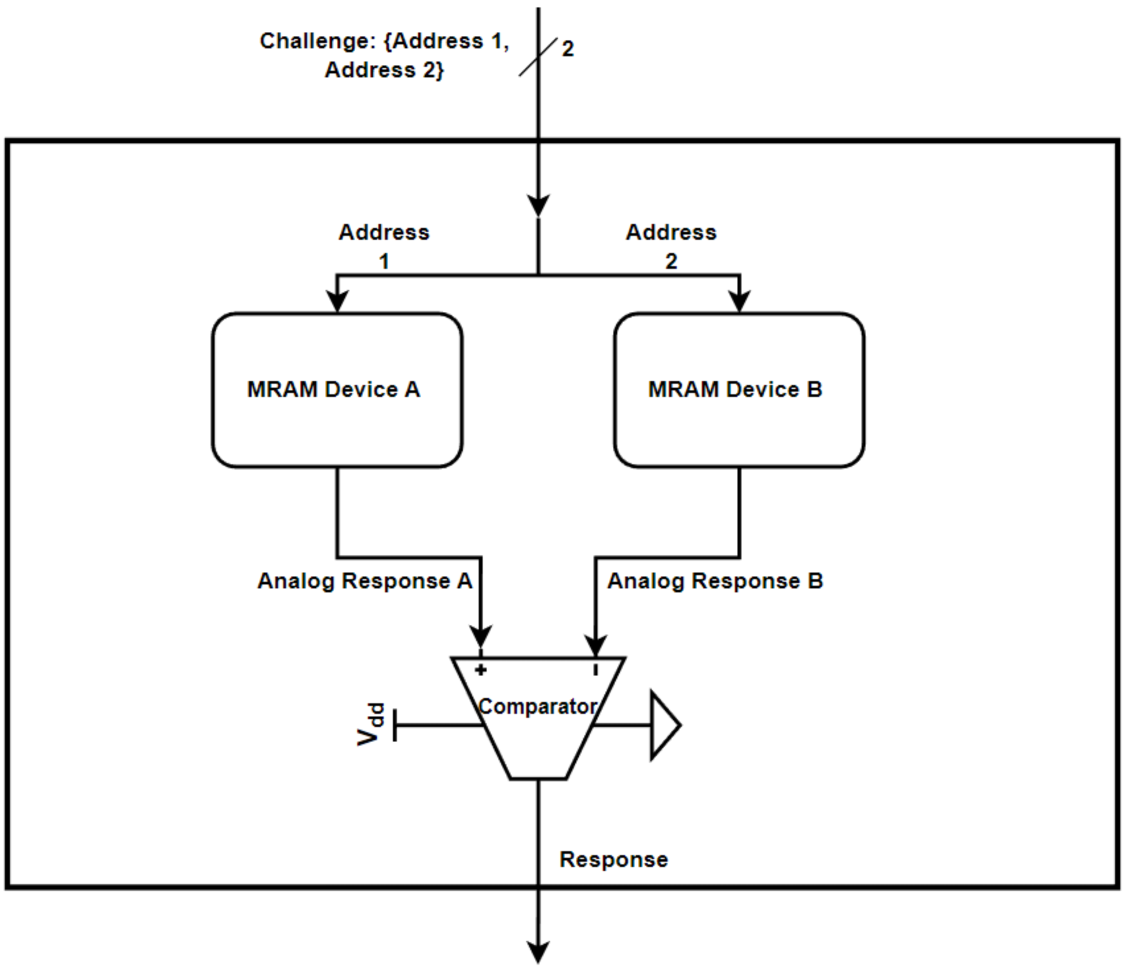


Figure 4.1: Block Diagram of MRAM PUF Design

4.2.0.1 Enrollment:

To enroll a device, all MRAM cells will be written to a common state (high or low resistive state). Afterward, each cell's resistance is measured an r number of times. This is denoted as the read count. The average resistance of each cell will be stored as the cell's resistance, and the standard deviation of each cell will be stored as the intra-cell variation. MRAM cells with relatively high intra-cell variation will be flagged during this phase as unstable cells.

4.2.0.2 Bit Generation:

To generate cryptographic bits, the server selects random addresses and then sends them to the client. The server will utilize software to compare the resistances of cell pairs in the database and generate an output bit. On the other hand, the client will employ physical MRAM devices and analog circuitry to compare the resistance of cell pairs and generate an output bit. In an ideal scenario, the bits generated from the database and those generated from physical measurements should be identical. However, due to random variations in cell queries, there might be practical discrepancies between the two sets of bits.

TAPUFs generate three possible values for each trit: '0', 'X,' or '1'. These values are derived from the relative resistance of the TAPUF, which is measured in voltage. The classification of each trit is determined by the voltage difference between the two analog responses of a challenge pair.

Challenge pairs classified as 'X' have a smaller voltage difference than those classified as '0' or '1'. The reason for this is that challenge pairs with a smaller voltage difference are more vulnerable to variations in measurements, noise, or external interference. Hence, they are assigned the 'X' classification to indicate their higher susceptibility. The state 'X' is used internally to keep track of the unstable bits, and only states '0' and '1' are used to generate responses.

It is crucial to recognize that in the cryptographic protocols under investigation, the TAPUF only utilizes the '0' and '1' responses while disregarding the 'X' states. These 'X' states are internally filtered out. From a system-level perspective, the TAPUF functions as a typical binary PUF, generating binary streams of responses.

The specific voltage difference thresholds used for trit classification depend on the employed cryptographic protocol. In this paper's context, the thresholds for trit classification are defined as the responses falling within the bottom and top 25th percentiles. However, it's important to note that these thresholds can be adjusted to meet the requirements of different protocols.

By considering the voltage difference between analog responses and applying appropriate thresholds, TAPUFs can classify trits as '0', 'X,' or '1', effectively capturing variations and susceptibility to measurement and environmental factors.

4.3 Hardware Implementation

4.3.1 MRAM Device

The MRAM device utilized in this research is the Everspin MR4A16B 54TSSOP version. This commercially available device is organized as 1,048,576 words of 16 bits, providing a high-density memory solution. The MR4A16B employs Toggle MRAM cells, which consist of one transistor and one Magnetic Tunnel Junction (MTJ) cell. The MTJ cell comprises two ferromagnetic films with parallel magnetic field orientations. A visual representation of a Toggle MRAM cell can be found in Section 2.4.2.1 of the document.

The circuit that we designed allowed us to design differential PUFs. By selecting a pair of cells, one from the upper byte and another from the lower byte, and applying a small voltage to the device's upper (UB) and lower (LB) pins, two currents are generated. These currents flow from the source into the chosen MRAM cells. The amount of current flowing into a cell allows for analyzing particular MRAM cells. Additionally, a small printed circuit board (PCB) adapter was employed to facilitate the easy mixing and matching of multiple devices on a single board.

4.3.2 Client device and Additional Hardware

4.3.2.1 Client-device:

This design was implemented on a low-power client device. We used the Nucleo-1444 H743ZI2 development board as the client device because of its low-power consumption and quick development time. This development kit has 112 general input/output (GPIO) pins, two 10-16 bit Analog-to-Digital Converters (ADC), 3.3V and 5V power supplies, and 2MB of flash memory. This kit's microcontroller unit (MCU) is the STM32H743ZI2 which operates at 3.3V and 480MHz.

4.3.2.2 Additional Hardware:

The currents that enter the MRAM device are used to determine the electrical resistance of the selected cells. To compare the cells and read them into the client device, it is necessary to convert the currents into voltages. Therefore, additional circuitry was required.

To ensure the safety of the device and facilitate resistance measurement, a fixed-value resistor was inserted in series with the pins connected to MRAM cells. This placement served two purposes. Firstly, it limited the current entering the device to protect against internal damage. Secondly, it enabled the measurement of

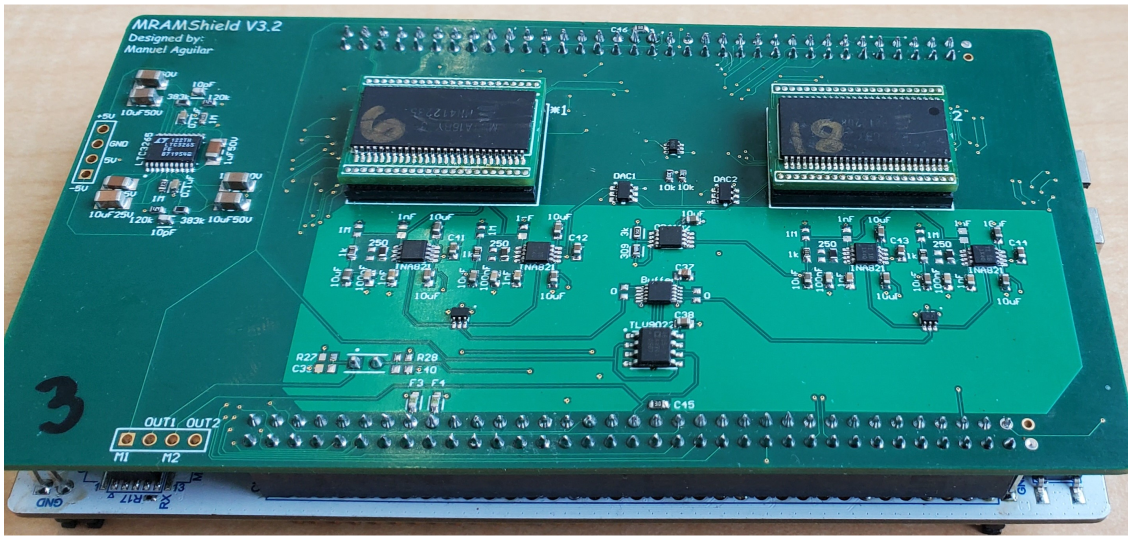


Figure 4.2: MRAM Hardware Circuitry on Nucleo-144

the voltage drop across a known resistance (the fixed resistor) and the electrical resistance of the MRAM cell itself. This measurement allowed us to determine the relative resistance of the MRAM cell. As the voltage drop across the fixed resistor was relatively low (less than 20 millivolts), a precision amplifier was employed to amplify the voltage drop, enabling more accurate comparison of resistances.

Once the voltage drops of the responses are amplified, they are connected to a comparator, which produces either a '0' or '1' output depending on which voltage is higher. Although the comparator is not as essential to this implementation as the differential amplifier, it is preferable to have a precise comparator with a fast output swing. In this specific hardware setup, the ADA4625 op-amp was used as the comparator. This particular op-amp was selected due to its high precision and fast output swing characteristics. A picture of the hardware circuitry and client device is shown in Fig. 4.2.

4.4 Electrical Characterization

4.4.1 Software

The server was emulated on a Personal Computer (PC) using `python3` and `pandas`. Enrollment data was collected and stored in CSV files and were accessed using `pandas`.

The client device was operated using `C++` code, which it used to execute any command the server gave.

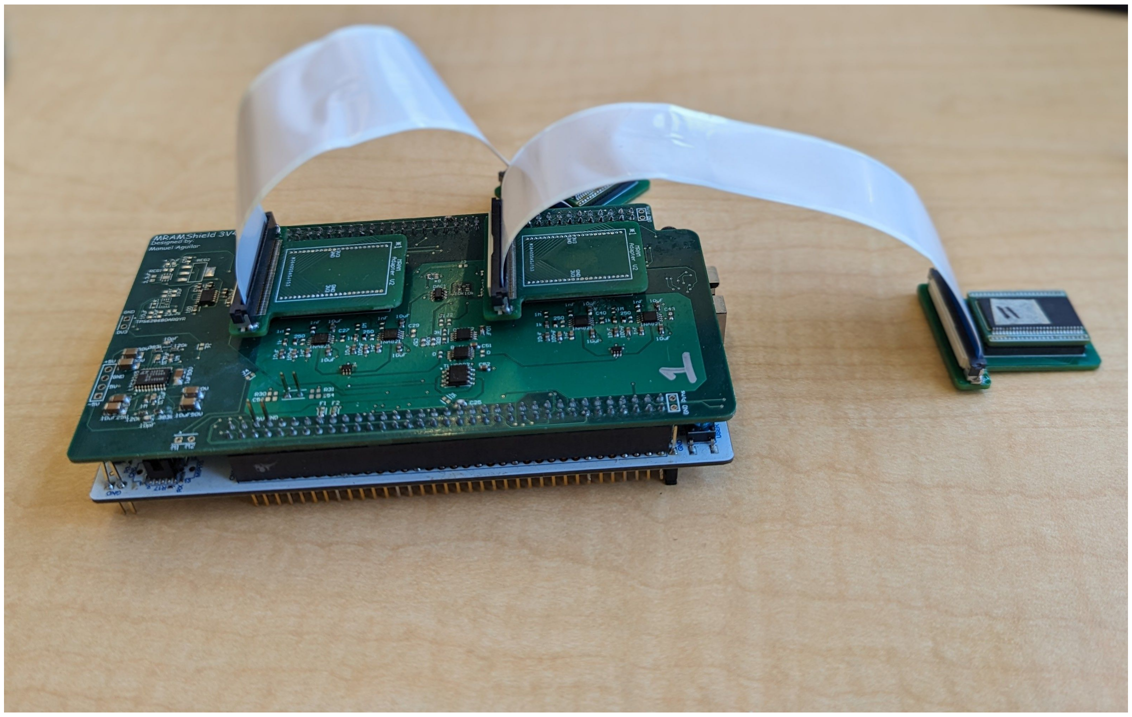


Figure 4.3: MRAM Adapter with Flat Flex Cable, used to isolate the MRAM device from external circuitry during temperature testing

The server and client communication was achieved using `pyserial`. With this configuration, we could generate responses at a throughput of 33.33 kbps.

Moreover, temperature testing is vital in the comprehensive electrical characterization of PUFs. As temperature variations can significantly influence the performance and reliability of electronic devices, it becomes imperative to evaluate the behavior of PUFs under different thermal conditions.

To assess the robustness and reliability of the PUFs, we conducted comprehensive tests and characterization at three different temperatures: 0°C, 23°C, and 80°C. To isolate the impact of temperature on the MRAM devices and exclude external measurement circuitry influence, we used flat flex cables and adapters during the temperature tests. This allowed us to focus solely on the temperature effects on the MRAM itself, gaining a clear understanding of its isolated thermal behavior. A picture of the MRAM adapters and flat flex cables used to facilitate isolation of the temperature effect are shown in Figure 4.3

By subjecting the PUFs to different temperature ranges and analyzing their responses, we aimed to gain valuable insights into their performance characteristics. This approach allowed us to identify potential vulnerabilities, enhance their overall performance, and develop effective calibration or compensation techniques that can be applied when deploying the PUFs in real-world scenarios.

To test varying temperature conditions, we used the Associated Environmental Systems SD-501 temperature chamber. This chamber allows us to create a wide range of temperature conditions, spanning from -37°C to 180°C. By examining the PUFs' behavior across different temperature conditions, we can better understand the impact of temperature variations on their reliability and make informed decisions to optimize their operation and ensure their dependable performance in practical applications.

4.4.2 Response Characteristics

To gain a comprehensive understanding of the electrical characteristics of the MRAM devices' relative resistances, we conducted a thorough analysis involving 30 MRAM devices. In these tests, we examined various parameters, including the mean, intra-cell variation, and inter-cell variation of the cells' resistance values. Additionally, we evaluated the resistances by testing both the low-resistive and high-resistive states.

For each MRAM device, we performed ten separate reads, each consisting of 10,240 words. The average of these ten reads was considered the cell's resistance. The intra-cell variation is a measurement that measures the variation of a cell's resistance over multiple queries. It was calculated by taking the standard deviation of the ten reads.

Inter-cell variation measures the variations of cells within the same device. we assessed the inter-cell variation of the PUF by calculating the standard deviation between the average resistances of MRAM cells on the same device.

4.4.2.1 Series resistance:

As discussed in §2.4.2.1, it is evident that the Complementary Metal-Oxide Semiconductor (CMOS) circuitry employed for accessing MRAM cells has a significant impact on the resistance obtained from these cells. Upon enrolling multiple devices, we made an intriguing discovery: MRAM cells situated on the same bit-line (such as 0, 1, 2, ... 15) exhibited similar resistances. This implies that a cell located on bit b on address a had resistances similar to one on the same bit b even on a different address c. This influence can lead to responses that are less 'random' with a less-than-normal distribution.

Considering the impact of bit-line circuitry on the electrical resistance of each MRAM cell is a crucial aspect to address. One potential approach to tackle this challenge is to treat each bit-line as an individual PUF, thereby ensuring independent resistances for each bit-line. However, this approach has limitations. It significantly reduces the number of available challenge responses for the PUF and necessitates carefully selecting MRAM devices with similar relative resistances.

Another approach is to adjust the resistance of all bit-lines to achieve a uniform median value. This adjustment is essential to mitigate the effects of varying bit-line series resistance and ensure consistent characteristics of MRAM cells across different addresses. Although this method requires additional circuitry and enrollment time, it has several advantages. It does not negatively affect the number of CRPs, and there is no need for meticulous device pairing. Additionally, since all the cells of specific bit-lines are indiscriminately shifted, the overall entropy of the MRAM cells remains unaffected.

To achieve the desired uniform median values of MRAM cell resistances across different bit-lines, a voltage value can be added to adjust the measured relative resistances. This adjustment involves selectively modifying the resistances of specific bit-lines to shift their median values. In this study, this manipulation was accomplished by integrating a Digital-to-Analog Converter (DAC) that was connected to the voltage reference pins of the amplifiers. This allowed precise control over the voltage values used for shifting the resistances, thereby ensuring consistent median values across the different bit-lines. Initially, we conducted an enrollment process on a pair of MRAM devices to identify the bit-line with the highest median voltage value. This bit-line was chosen as the reference. Subsequently, all cells from other bit-lines were adjusted upward to achieve an equal median voltage value as the reference bit-line. These adjustments were then saved on the client's device for future utilization.

Figure 4.4 compares the electrical resistances of a single MRAM device with and without the applied shift in the bit-line median values.

The inter-cell to intra-cell ratio is the inter-cell variation divided by the intra-cell variation. Ideally, we want the inter-cell to intra-cell ratio to be as high as possible to minimize BER. This means we want to maximize inter-cell variation and minimize intra-cell variation.

As discussed in section 4.4.2.1, the response statistics must be categorized by bit-line, as the bit-line CMOS series resistance heavily affects the cell's response.

After enrolling 30 devices in both high and low-resistive states, the enrollment data of the electrical resistances were normalized. To normalize the data, we divided the resistance values by the mean of the entire population as shown in equation 4.1.

$$\hat{x} = \frac{x}{\mu_{POP}} \tag{4.1}$$

If the mean of the entire population is denoted as T, the responses range from $\pm 31.16\%$ of T. The average intra-cell variation was only 0.231% of T, with the average bit-line inter-cell variation being 5.794% of T, giving an inter-cell to intra-cell variation of 25.03.

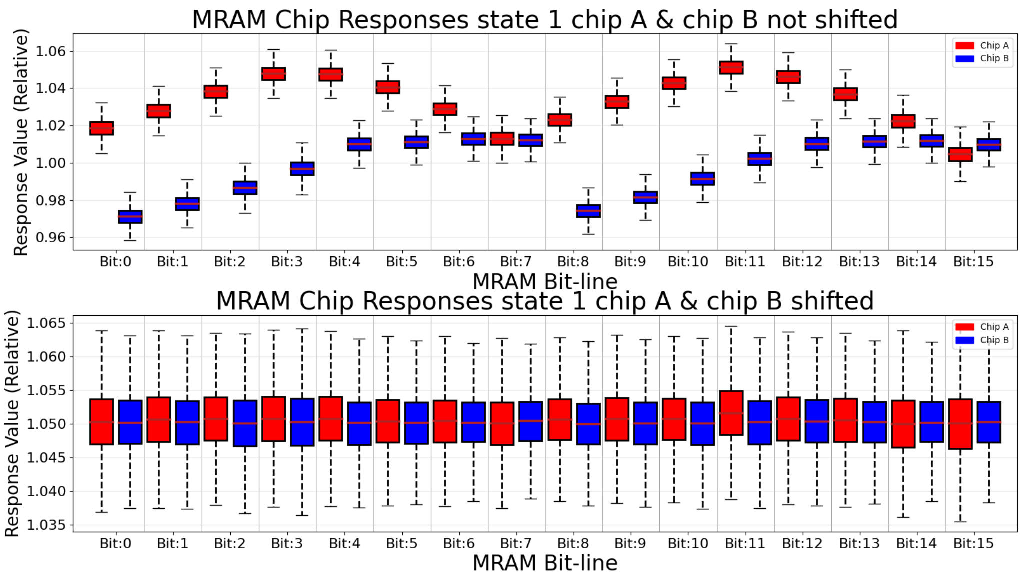


Figure 4.4: Comparison of Bit-line medians without adjustment and after adjustment.

A toggle MRAM cell written to a low-resistive state, with CMOS series resistance, had an average response of $L = T \times 0.852$, with responses ranging from $\pm 46.77\%$ of L. The average intra-cell variation was 0.203% of L, with the average bit-line inter-cell variation being 5.261% of L, giving an inter-cell to intra-cell variation of 25.88. The response statistics for both states can be found in table 4.1.

The average bit-line inter-cell to intra-cell ratio increased by 0.85 from a high to a low resistive. Both MRAM states showed excellent statistical values for use as a PUF. While the low-resistive statistical values showed slightly better electrical characteristics, the difference was so minuscule that both states could be used to generate responses.

4.4.2.2 Temperature Effect

The characteristics of MRAM resistances were investigated at different temperatures: 0°C, 23°C, and 80°C. Upon analyzing the data, several notable patterns emerged. Table 4.2 shows the characteristic of resistance responses for each temperature.

Firstly, in the low-resistive state, it was observed that as the temperature increased, the resistances also increased. Conversely, in the high-resistive state, the resistances decreased as the temperature increased.

Additionally, it was observed that with increasing temperature, the variation in resistance between cells

MRAM Response Statistics	**High resistive State**	**Low resistive State**
Maximum	$T \times 1.203$	$L \times 1.3666$
Average	T	$L = T \times 0.8521$
Minimum	$T \times 0.8914$	$L \times 0.8980$
Range	$T \pm 0.3116$	$L \times 0.4677$
Average bit-line inter-cell variation	$T \times 0.005794$	$L \times 0.005261$
Average intra-cell variation	$T \times 0.000231$	$L \times 0.000203$
Inter-cell to intra-cell variation ratio	25.03	25.88

Table 4.1: Normalized MRAM Resistance Statistics of 100 MRAM devices at $23°C$

	High resistive state			**Low resistive state**		
MRAM Response Statistics	**0**$°C$	**23**$°C$	**80**$°C$	**0**$°C$	**23**$°C$	**80**$°C$
Maximum	$T1 \times 1.207$	$T \times 1.203$	$T2 \times 1.195$	$L1 \times 1.3807$	$L \times 1.3666$	$L2 \times 1.3226$
Average	$T1 = T \times 1.00375$	T	$T2 = T \times 0.9816$	$L1 = L \times 0.9952$	$L = T \times 0.8521$	$L2 = L \times 1.00440$
Minimum	$T1 \times 0.8843$	$T \times 0.8914$	$T2 \times 0.8947$	$L1 \times 0.8962$	$L \times 0.8980$	$L2 \times 0.8993$
Range	$T1 \pm 0.3232$	$T \pm 0.3118$	$T2 \pm 0.3001$	$L1 \times 0.4848$	$L \times 0.4676$	$L2 \times 0.4231$
Average bit-line inter-cell variation	$T1 \times 0.005873$	$T \times 0.005794$	$T2 \times 0.005464$	$L1 \times 0.005331$	$L \times 0.005262$	$L2 \times 0.005011$
Average intra-cell variation	$T \times 0.000231$	$T \times 0.000231$	$T2 \times 0.000240$	$L1 \times 0.000201$	$L \times 0.000203$	$L2 \times 0.000217$
Inter-cell to intra-cell variation ratio	25.44	25.03	22.78	26.55	25.88	23.09

Table 4.2: Normalized MRAM Resistance Statistics of 30 MRAM devices at 0 °C, 23 °C, and 80 °C.

within a single device (intra-cell variation) increased, while the variation in resistance between different devices (inter-cell variation) decreased. This trend resulted in a lower inter-cell to intra-cell variation ratio. However, even at 80°C, the inter-cell to intra-cell variation ratio remained above 20 for both resistive states. This ratio is more than sufficient for reliable bit generation.

4.4.3 PUF Metrics

4.4.3.1 Inter-distance:

The presence of correlation among MRAM devices can pose security risks, as it may lead to similar responses between different chips. To ensure that there is no correlation among the chips, we utilized the concept of HD_{inter}. HD_{inter} is a random variable that quantifies the distance between two PUF responses obtained from different chips but subjected to the same challenge. Specifically, it measures the Hamming Distance between the responses of two distinct PUFs when they encounter identical challenges. The resulting distance is then divided by the total number of responses. Ideally, a PUF should exhibit an HD_{inter} value of **0.50**,

HD_{inter} **High resistive state**	0°**C**	23°**C**	80°**C**
Unfiltered Responses	0.4820	0.4849	0.4753
Filtered Responses	0.4999	0.4999	0.4997

Table 4.3: The HD_{inter} of the filtered and unfiltered responses in a high resistive state at 0°C, 23°C, and 80°C. Filtered HD_{inter} had responses filtered for 'X' states, while unfiltered did not.

indicating no correlation between chips.

To quantify the HD_{inter} value of our PUF design, we carried out an experiment involving the generation of 15 response sequences. Each sequence consisted of a response obtained from a unique PUF configuration, where a configuration is defined as any unique ***pair*** of MRAM devices. These responses were acquired from the PUF in both its high and low resistive states at 23°C. Each response sequence was 2^{20} bits long (1 Mb response), ensuring a substantial amount of data for analysis.

Furthermore, as part of our analysis, we generated two sets of sequences: one with unfiltered CRPs and another with CRPs filtered for 'X' states. We then calculated the average HD_{inter} for each set.

For the unfiltered CRPs, the average HD_{inter} was found to be **0.4797** in a low-resistive state and **0.4849** in a high-resistive state. On the other hand, for the filtered CRPs, where CRPs classified as 'X' were removed, the average was HD_{inter} determined to be **0.4997** in a low-resistive state and **0.4999** in a high-resistive state.

At 23°C, it can safely be concluded that the responses obtained from both states exhibit a similar HD_{inter}, indicating that the resistive state of the MRAM devices does not significantly impact the uniqueness of the PUF responses.

To assess the uniqueness of the MRAM PUF at various temperatures, we performed identical tests at each temperature. Since the resistive state of the MRAM responses did not exhibit a correlation with the HD_{inter} value, we focused solely on testing the high-resistive states.

When the MRAM TAPUF was subjected to temperatures of 0°C and 80°C, there was no substantial change observed in the uniqueness of its responses. At 0°C, the average HD_{inter} values were **0.4820** for unfiltered responses and **0.4999** for filtered responses. Similarly, at 80°C, the average HD_{inter} values were **0.4753** for unfiltered responses and **0.4997** for filtered responses.

The results of these tests can be found in 4.3. Based on these results, we can conclude that temperature had no significant impact on the HD_{inter} for the given PUF configuration.

4.4.3.2 BER

The BER is a metric employed to assess the reliability of a PUF. It is determined by generating a response sequence from a predetermined set of challenges on the client device and comparing it to the response sequence generated using the same set of challenges on the server side. The server utilizes enrollment data to generate responses, while the client employs the PUF to generate real-time responses. The BER is calculated by dividing the number of discrepancies or differences between the two response sequences by the size of the response sequence.

The settling time of the output signal in the comparator plays a crucial role in determining the BER of our system, considering that our responses are represented as voltages. Most comparators exhibit a slew rate for their output that depends on the voltage difference between the two inputs being compared. Consequently, when comparing voltages with a small difference, a longer settling time is necessary for the output signal to stabilize.

In theory, a higher settling time allows the comparator to achieve more accurate responses. This extended settling time provides the comparator with sufficient duration to stabilize the output, particularly when dealing with CRPs featuring small voltage differences. Therefore, a longer settling time is advantageous for improving the accuracy and reliability of the system's responses.

To evaluate the BER, we conducted extensive testing by generating one million responses across various voltage differences, settling times, and resistive states. The voltage differences ranged from 1 mV to 12 mV in 1 mV increments, while the settling times ranged from 10 to 50 μs, with steps of 10 μs. Additionally, each test was conducted separately for both high and low resistive states. The results, including the plot and corresponding BER values based on voltage difference, settling time, and resistive state, are presented in Figure 4.5.

As expected, based on the conducted tests, we observed that as the voltage difference and settling time increased, the BER decreased. Surprisingly, the responses generated from a low-resistive state exhibited a noticeably higher error rate compared to those generated from a high-resistive state. This unexpected difference suggests that the higher intra-cell variation, measured in millivolts (mV), observed in the low-resistive state contributes to greater instability in the CRPs and, resulting in an impact to the BER.

To evaluate the BER under different temperatures (0°C, 23°C, 80°C), we conducted key generation experiments using the bottom and top 25th percentile of responses. At each temperature, we generated a total of 1024 keys, each consisting of 1024 bits, which resulted in over 1 million bits being generated for each temperature. Additionally, for every temperature, a key was generated using enrollments obtained at each temperature.

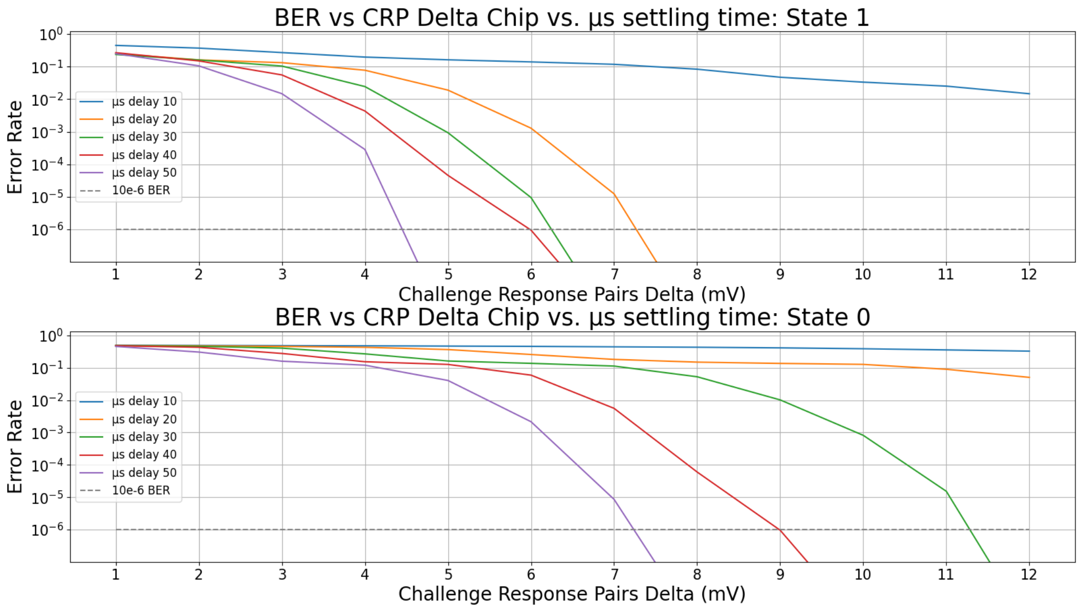


Figure 4.5: BER vs. challenge-pair voltage difference vs. settling time vs. resistive state

During the key generation process, we exclusively utilized a high resistive state, which exhibited greater stability when subjected to lower voltage pair differences. Moreover, a 20 μs settling time was used as it showed a good balance of speed and accuracy.

The determination of minimum voltage pairs when utilizing the bottom and top 25th percentiles is contingent upon the specific MRAM devices employed. In the case of the particular chips utilized for this test, we were able to obtain voltage pairs exhibiting a voltage difference of 11 millivolts or more. This finding is specific to the MRAM devices employed in this particular test and may vary when using different MRAM chips.

Table 4.4 displays the outcomes of the BER at various temperatures. Upon examination of the results, it becomes apparent that temperature does exert a slight influence on the BER. When the temperature during key generation differs from the temperature at which the enrollments were conducted, an increase in the BER is observed. This suggests that temperature variations can have a slight negative effect on the reliability of the key generation process.

However, this effect can be mitigated by conducting enrollments at various temperatures and selecting the enrollment that is closest to the operating conditions of the client device. By doing so, the impact of temperature on the BER can be minimized.

BER for 1 million bit-long key, at high resistive state	**At 0°C**	**At 23°C**	**At 80°C**
Using 0 °C Enrollments	$< 10^{-6}$	$< 10^{-6}$	8×10^{-5}
Using 23 °C Enrollments	$< 10^{-6}$	$< 10^{-6}$	2×10^{-5}
Using 80 °C Enrollments	9×10^{-5}	7×10^{-5}	$< 10^{-6}$

Table 4.4: The BER of a 1 million-bit-long key at different temperatures. The key size was 2^{20}. If 0 errors were found, the BER is less $< 10^{-6}$.

4.4.3.3 Entropy Density

To ensure a high level of security and randomness, it is essential that the responses of a PUF exhibit no correlation, even when a large number of CRPs are available. The degree of uncertainty or randomness in a sequence can be quantified using entropy density. This measure is based on the distribution of the random variable and serves as a generalized and unconditional upper bound on the average predictability of an unobserved response within the PUF. In an ideal scenario, the entropy value should approach 1.00, indicating maximum randomness.

To calculate the entropy, we employ the Shannon entropy density function, as discussed in detail in Section 2.3.3. This function enables us to assess the level of entropy in the PUF's responses, providing valuable insights into the unpredictability and security of the system.

To assess the entropy of our MRAM PUF design, we conducted extensive testing by generating one million-bit long keys in both the low and high-resistive states. Additionally, we also generated a key filtered for 'X' states and an unfiltered key. The results of these tests revealed the following entropy values: for unfiltered keys, we obtained an entropy of **0.99940** in the low-resistive state and **0.99963** in the high-resistive state. In contrast, for filtered keys, the entropy values were **0.99999** in the low-resistive state and **0.99999** in the high-resistive state.

To examine the entropy of our design under varying temperatures, we calculated the entropy density using responses generated at two specific temperatures: 0°C and 80°C. In this analysis, we focused solely on the high-resistive state responses, as the resistive state did not seem to affect the outcome significantly.

The Shannon entropy values for unfiltered keys at 0°C and 80°C were found to be **0.99821** and **0.99993**, respectively. Similarly, for filtered keys, the Shannon entropy values at both temperatures were **0.99999**. These results indicate that temperature does not have a notable impact on the entropy density of the PUF responses.

The test results can be found in Table 4.5. The consistency of the entropy values across different

Shannon Entropy Density	0°**C**	23°**C**	80°**C**
Unfiltered Responses	0.99821	0.99963	0.99993
Filtered Responses	0.99999	0.99999	0.99999

Table 4.5: The average Shannon Entropy Density of the MRAM TAPUF, using filtered and unfiltered responses, at 0°C, 23°C, and 80°C.

temperatures suggests that the PUF design maintains its entropy characteristics regardless of the temperature at which the responses are generated. This finding reinforces the stability and reliability of the PUF in preserving its high level of entropy, further enhancing its suitability for cryptographic applications in various temperature conditions.

4.5 MRAM TAPUF Conclusions

In this research, we introduced a novel design for a TAPUF utilizing two MRAM devices. Our design leveraged the extracted resistances from these devices to generate reliable responses, and we thoroughly characterized both analog and digital responses.

To evaluate the performance of our TAPUF design, we enrolled a total of 30 MRAM devices in both high and low-resistive states. We conducted enrollment and characterization experiments at various temperatures to investigate the impact of temperature on the analog responses. Furthermore, we utilized real-time responses to calculate the HD_{inter}, BER, and entropy density of the TAPUF design.

The analysis of the electrical characteristics of the PUF responses yielded valuable insights into the performance of the PUF. One significant discovery was the significant impact of the CMOS series resistance on the PUF responses. However, this issue was effectively resolved by adjusting the PUF responses to ensure that they all had the same median value. This modification allowed the MRAM TAPUF to preserve its entropy while accommodating a large number of CRPs (n^2).

Furthermore, another key finding was the excellent inter-cell to intra-cell variation ratio, which indicated consistent behavior among the cells in both resistive states. This consistency across cells is essential for reliable and predictable PUF responses. Furthermore, minimal differences were observed between the high and low resistive states, suggesting that the PUF's performance remained stable regardless of the resistive state.

Upon comparing the two states, the high-resistive state exhibited superior performance in various metrics, particularly regarding BER. Both states demonstrated excellent BER values, but the high-resistive state outperformed the low-resistive state, especially for CRPs with smaller voltage differences. Specifically, at a voltage difference of 10mV and a temperature of 23°C, the high-resistive state showcased exceptional

	PUF Designs			
PUF Metrics	**MRAM TAPUF**	**STT MRAM PUF** [79]	**Geometry-based MRAM PUF** [81]	**Write Current STT MRAM PUF** [74]
Number of CRPs	n^2	$n/2$	n	n
BER or HD_{intra}	$< 10^{-6}$*	6.6×10^{-6}	2.25×10^{-2}	5×10^{-2}
HD_{inter}	0.4999*	0.501	0.47	0.499
Entropy Density H(X)	0.99999*	0.985	0.99	0.999

Table 4.6: PUF metrics of previous MRAM PUFs compared to our proposed design. n = the number of MRAM cells on a single MRAM device. Metrics denoted with *are for responses with 'X' states filtered out

performance with an inter-distance value of 0.4998, an entropy value of 0.9999, and a BER below 10^{-6}.

Moreover, this design demonstrated remarkable reliability across a wide temperature range spanning from 0°C to 80°C. When enrollments were conducted at the same temperature used for key generation, the BER remained exceedingly low, indicating consistent and accurate responses. However, as the temperature difference between the enrollment temperature and the key generation temperature increased, a slight increase in the BER was observed. It is worth noting that even with significant temperature variations, the BER remained within acceptable limits, with a value of 9×10^{-5} at both 0°C and 80°C. These findings emphasize the robustness and resilience of the design in maintaining reliable performance across diverse temperature conditions.

Furthermore, the entropy and HD_{inter} measurements of the system remained close to their ideal values under the tested temperature conditions. This further indicates that the design maintains a consistent and reliable level of performance, ensuring a high degree of randomness and minimal correlation between responses, even when exposed to varying temperatures.

Compared to previously proposed MRAM PUF designs, this particular design demonstrated comparable, if not better, PUF metrics as shown in **Table 4.6**. These findings highlight the promising performance and reliability of this MRAM PUF design, suggesting its potential superiority over existing designs.

4.6 TAPKI and PUF-seeded PQC

The MRAM TAPUF has been employed in the Ternary Addressable Public Key Infrastructure (TAPKI) system to strengthen the security of cryptographic protocols. In this specific implementation, a pair of MRAM chips were generated using the TAPKI method. Subsequently, this key was utilized to seed the CRYSTALS-Kyber algorithm, resulting in the generation of another pair of keys that are resistant to attacks from quantum computing.

4.6.1 TAPKI Implementation

The TAPKI protocol is a method within the key exchange that employs TAPUFs to generate a new key for each transaction. TAPKI utilizes a ternary representation system with three states: '0', '1', and 'X'. The server possesses knowledge of the masked locations where the 'X' states occur, which complicates attackers' attempts to exploit the clients' PUFs. TAPKI consists of two phases: enrollment and key exchange.

During the enrollment phase, the server generates an image of the PUF. On the other hand, the key exchange phase occurs every time the server and client engage in a "Handshake". In the key generation process, the server and client collectively create a secret called the seed, while a set of secret instructions is shared between them. These instructions guide the client in identifying specific CRPs to skip while the remaining CRPs are read as either '0' or '1' to form the shared secret.

4.6.1.1 Handshake and Key Generation Overview

To identify '"flaky' CRPs for MRAM, we employed a two-step process. Firstly, we flagged individual MRAM cells exhibiting a high intra-cell variation, which was determined using the equation mentioned in Equation 4.2. These flagged cells were considered potential outliers.

Next, we generated random CRPs utilizing the stable individual MRAM cells. The CRPs were then categorized based on the absolute difference between the electrical resistances of the paired MRAM cells. This categorization is visualized in Figure 4.6. Only the top p percent of these CRPs, where p represents a configurable parameter, were selected for key generation. It is worth noting that a smaller value of p leads to more stable CRPs, as the average distance between the selected CRPs increases. In our specific implementation, the value of p was set to 50%.

$$UpperFence = Q3 + (1.5IQR) \tag{4.2}$$

4.6.1.2 Server side

Initially, a random number is generated for the Handshake, along with a shared password between the client and server. This random number is then concatenated with the password and passed through the SHAKE function. The resulting output provides hashed bytes, which should have a length equal to the product of the pool size and the bytes per bit address.

These hashed bytes are then interpreted as challenges and adjusted to fit within the challenge space of the TAPUF. Each challenge is checked against the enrollment data to determine if it corresponds to an 'X' state. If it does, a '0' is added to the stability mask, indicating that the bit should not be considered in

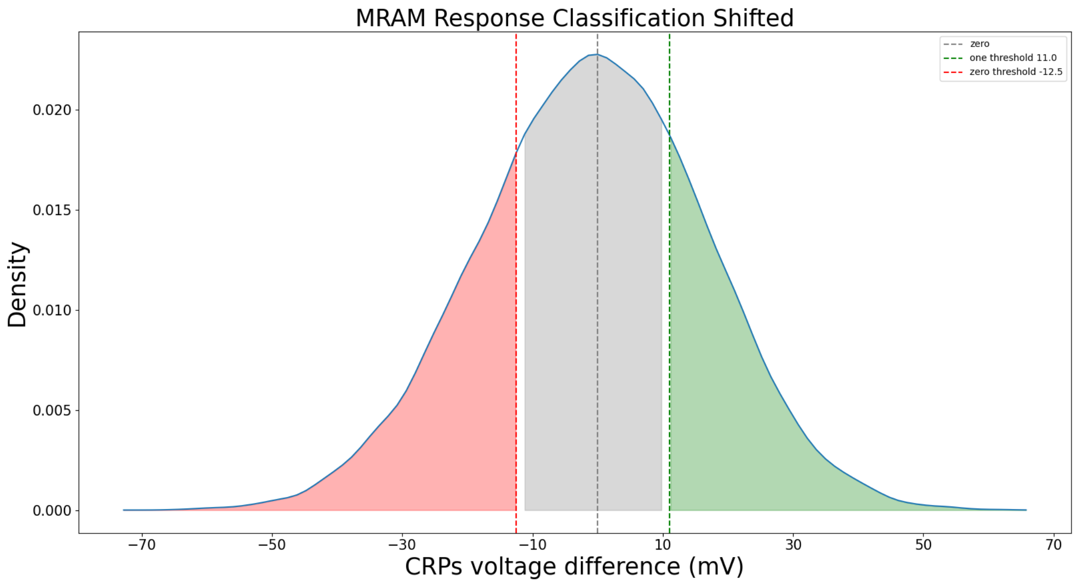


Figure 4.6: Distribution of MRAM CRPs using a p of 50. In the representation, red corresponds to the '0' states, green corresponds to the '1' states, and gray corresponds to the 'X' states.

the final response. Conversely, if the challenge corresponds to a non-'X' state, a '1' is added to the stability mask.

Once the stability mask is created, 256 bits of stable data are selected, while the remaining bits have their corresponding positions set to '0' in the stability mask, indicating that they should not be used. To ensure data security, the stability mask is XORed with the hashed bytes before being sent to the client.

The server sends the Handshake data to the client, which includes the selected random number and the result of XORing the hash with the mask. Once the client responds, the server verifies the received response against the enrollment data using Response Base Cryptography (RBC). The server-side protocol can be better understood by referring to Figure 4.7.

- Generate a random number, RN
- Concatenate the RN and password, PW, into RN_{PW}.
- Hash RN_{PW} into the desired number of bytes to produce H_{RNPW}.
- Convert H_{RNPW} to the number of Challenges.
- Look up expected responses for each Challenge to generate S_k.
- Build a stability mask, M_S, over the generated addresses, where a '0' is a 'flaky' CRP ('X') and a '1' is a stable challenge (strong '0' or '1').
- XOR M_S with H_{RNPW}, to generate H_{MS}.
- Send the RN and H_{MS} to the client as the Handshake.
- Receive a response from the client.
- Perform RBC to verify the user's response.

4.6.1.3 Client side

The client-side process is relatively simpler compared to the server-side. Once the Handshake is received, the client parses the random number from the Handshake data while the user supplies a password. By combining the parsed random number with the SHAKE function, the client generates the same hashed bytes as the server side.

The client utilizes the hashed bytes received to retrieve the stability mask and the challenges employed in generating the secret key. Using the stability mask and the TAPUF, the client generates the secret key,

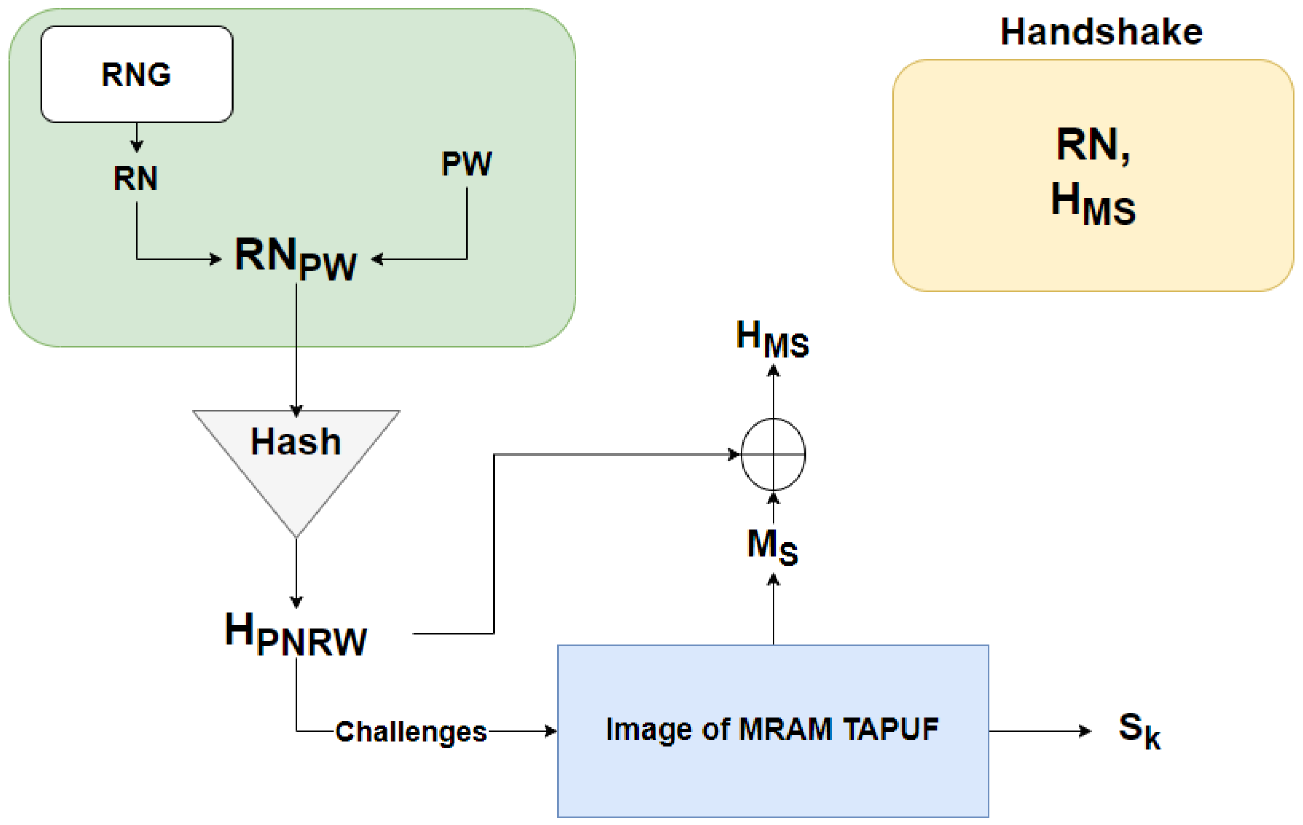


Figure 4.7: TAPKI Protocol on the server. The server uses a Random Number Generator (RNG) to generate a random number, RN. RN and password, PW, are then hashed, to generate H_{RNPW}. H_{RNPW} is then used to generate challenges for the MRAM TAPUF. Using an image of the MRAM TAPUF, a mask, M_S, and a secret key, S_k are generated. The mask is XORed with H_{RNPW} to generate H_{MS}. A handshake which contains RN and H_{MS} is sent to the client

which is subsequently transmitted back to the server for validation. A depiction of the client-side protocol can be observed in Figure 4.8.

- Receive RN and H_{MS} from Handshake.
- Create H_{RNPW} with the RN and the PW.
- Recover M_S by XORing H_{RNPW} and H_{MS}.
- Recover Challenges using H_RNPW.
- Using challenges to generate S_k with M_S.
- Send S_k to the server.

4.6.2 Testing and Methodology

To assess the performance of TAPKI with MRAM TAPUF, we conducted three specific tests. The first test involved examining the BER while varying the number of enrollments. This allowed us to observe how the system performed under different enrollment scenarios. The second test focused on calculating the False rejection rate (FRR) and False acceptance rate (FAR) when utilizing the RBC protocol with a maximum error rate of 4 percent. This test helped us evaluate the system's accuracy and reliability in authenticating users. In the third test, we assessed the randomness of the stable challenges generated for TAPKI. This test aimed to ensure that the challenges produced by the system were truly random and suitable for secure key generation.

In the first test, we generated 100,000 256-bit keys using enrollments with different numbers of reads: five, ten, and fifteen. The results of this test can be seen in Figure 4.9. Analyzing the results, we observed a trend: as the number of reads in an enrollment increased, the number of errors per key decreased. Once the number of reads per enrollment reached ten, errors became low enough to have the correct found with the RBC protocol.

In the second test, we generated 100,000 keys using the MRAM devices that were used for enrollment, as well as a different set of MRAM devices. Using the devices from the enrollment process, we achieved a FRR of 0%. Equally impressive, when using the different sets of MRAM devices, we achieved a FAR of 0%.

In the third test, we evaluated the randomness of the challenges generated by TAPKI. Since the key generation process of TAPKI may introduce biases in selecting strong CRPs, it is crucial to ensure that attackers cannot exploit any potential biases. To test the randomness of the challenges, we plotted the

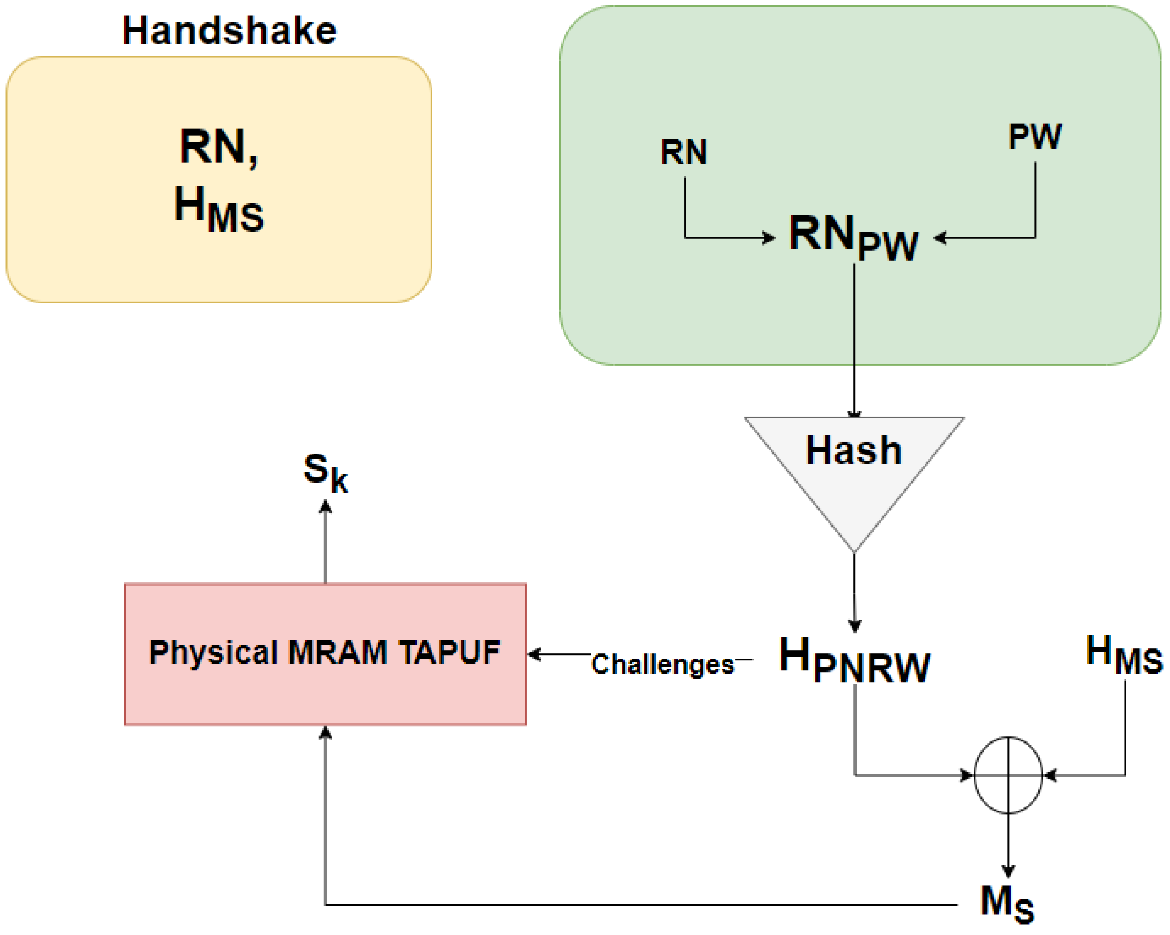


Figure 4.8: TAPKI Protocol on the client. The client receives a random number, RN, and a hash of the random number and password XORed with a mask (M_S), H_{MS}. The random number and password (PW) generate a hash, H_{RNPW}. The M_S and challenges are extracted using H_{RNPW}. Using the physical MRAM TAPUF, M_S, and the challenges, the client generates the same S_k generated on the server.

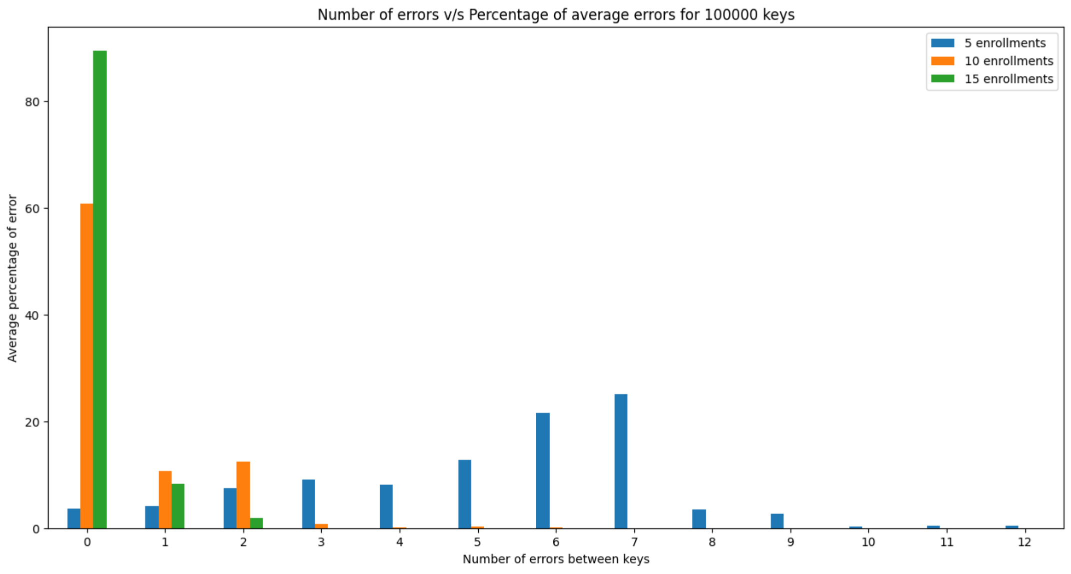


Figure 4.9: Number of Errors vs. Percentage of Average Errors for 100000 keys. As the number of reads in enrollment increases, the error rate among keys decreases. At ten enrollments per read, the number of errors between keys becomes acceptable

strong challenge pairs generated by the protocol. The results are depicted in Figure 4.10. Upon analyzing the results, we observed no apparent biases attackers could target when generating keys using this protocol.

The average latency for the entire key generation protocol of a single 256-bit key process using the MRAM TAPUF and RBC was 1.34 seconds. This falls into the acceptable range of key generation latency as it is a time that is relatively unnoticeable to users.

By conducting these three tests, we obtained valuable insights into the performance of TAPKI with MRAM TAPUF. The tests examined error rates in key generation, authentication accuracy, the randomness of the generated challenges, and latency using different MRAM devices.

4.6.3 PUF-seeded PQC Using MRAM TAPUF

Post-Quantum Cryptography (PQC) is a form of cryptography designed to resist attacks from quantum computers. It relies on encryption protocols based on lattice or code cryptography. However, PQC still requires storing cryptographic keys, which poses a complex challenge. To address this issue, Physical Unclonable Functions PUF can be utilized in conjunction with a Certificate Authority (CA) to generate and validate one-time key pairs, as discussed in §2.5 of the document.

To evaluate the effectiveness of using the MRAM-based TAPUF as a seed for PUFs, we employed the TAPKI protocol with the MRAM TAPUF design to exchange private keys for use with the CRYSTALS-

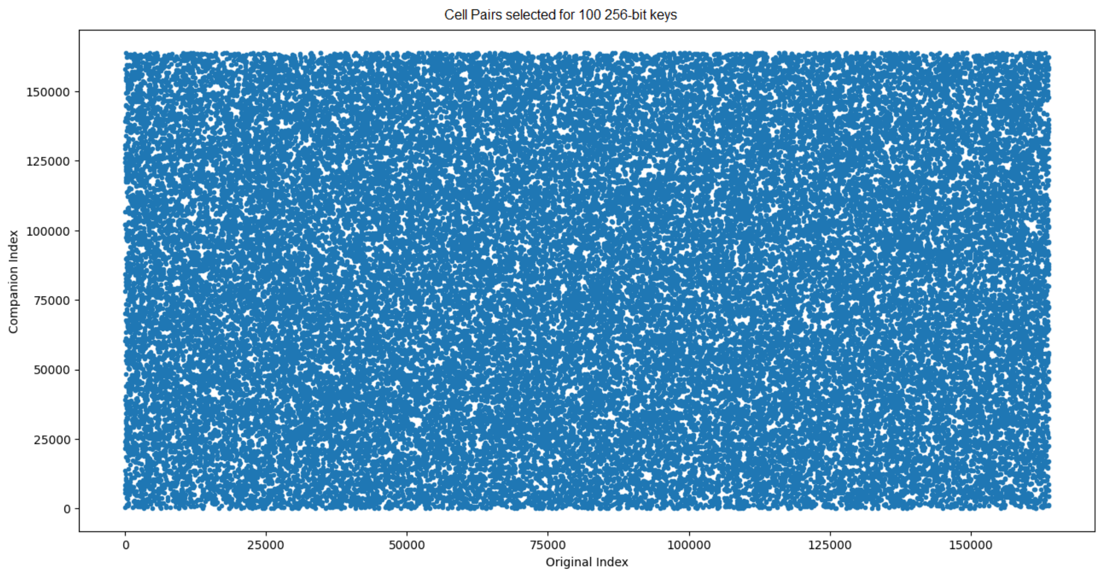


Figure 4.10: Strong challenge cell pairs generated using TAPKI for 100 256-bit keys

Kyber protocol. In our implementation using Python 3, the CRYSTALS-Kyber protocol demonstrated a latency of 35 milliseconds for encrypting and decrypting a 256-bit key. However, when the protocol was seeded with a PUF-based TAPKI, the average duration of the entire process increased to approximately 1.4 seconds, a forty times increase.

Utilizing PUFs in conjunction with CRYSTALS-Kyber provides the advantage of increased security by generating one-time keys. However, it comes with the downside of a relatively significant increase in latency. Ultimately, the decision to incorporate PUFs into this algorithm should be made on a case-by-case basis, considering the specific requirements and priorities of the application.

4.6.4 TAPKI and PUF-seeded PQC Conclusions

In this section, we utilized the MRAM-based TAPUF in conjunction with TAPKI and RBC to generate one-time keys. The protocol implementation was successful overall, and the low BER resulted in an average validation time of 1.34 seconds for a 256-bit key.

Additionally, we employed the TAPKI protocol with the MRAM TAPUF to validate public key pairs for the CRYSTALS-Kyber algorithm. However, when using TAPKI for key pair validation and subsequent encryption and decryption processes, the latency increased significantly. It took 1.4 seconds to encrypt and decrypt a 256-bit key, which is 40 times longer than the latency of the algorithm itself. Therefore, the decision to incorporate a PUF should be made on a case-by-case basis, taking into account the specific requirements and considerations of the application.

4.7 TRNG using MRAM TAPUF

TRNGs based on MRAM often have limitations: they can only generate non-deterministic bits. Furthermore, most of the research and proposals for these TRNGs rely solely on simulations, lacking experiments with physical devices and leaving a gap in the research. On the other hand, the MRAM TAPUF has a dual capability—it can generate both reliable cryptographic bits and non-deterministic random bits. It achieves this by leveraging the unpredictable variations in unstable CRPs, allowing the generation of bits that cannot be determined even with the enrollment of the PUF.

Following the methodology used in the previous chapter, the NIST Testing Suite is employed to assess the randomness of the TRNG. The main goal of this study is to design a protocol that can operate effectively on a low-power client device using only a lightweight post-processing design.

4.7.1 TRNG Design

The TRNG design utilized in this case is identical to the design employed for the MRAM TAPUF, as described in §4.2. In this particular design, response bits are generated by comparing the electrical resistances of MRAM cells acquired from two distinct MRAM devices. These resistances are then converted into voltages, which are read by an ADC. The resulting voltages are fed into a comparator.

During this process, the voltage mismatch between CRPs is considered. CRPs with a lower voltage mismatch are classified as "flaky" or denoted as 'X', while CRPs with a higher voltage mismatch are classified as either '1's or '0's. The term 'flaky' or 'X' refers to CRPs more susceptible to flipping due to measurement variation and noise. In contrast, the stronger '0's and '1's are less likely to experience such fluctuations.

This design capitalizes on the utilization of unstable bits as a valuable resource. In reliable cryptographic bit generation protocols, these unstable states are typically eliminated to enhance the BER. However, in True Random Number Generator (TRNG) protocols, these unstable bits can be intentionally targeted to generate non-deterministic responses. Furthermore, by exploiting these unstable bits, the entropy of the generated responses is significantly enhanced due to the amplification of noise and measurement variation factors during the response generation process.

4.7.2 Enrollment and Classification

The electrical resistance of each cell needs to be read and stored as a reference in a secure location. Recording these responses is known as the enrollment cycle. This is generally done in a secure environment since, if it falls into the wrong hands, the device can be digitally cloned. The enrollment scheme is as follows:

1. Write all cells to a high-resistive state ('1') by following the instructions on the MRAM device datasheet.
2. Enter analog read mode, which gives direct access to the MRAM cells.
3. Enroll the PUF's electrical resistances and store the information securely.

The enrollment process is identical to the one used for the MRAM TAPUF. It involves reading each cell multiple times to identify the ones which are unstable or 'flaky', denoted by 'X'. For the TRNG, each cell is read ten times separately and saved in CSV files. The latency of the process is recorded to be ten minutes. After ten reads, the average resistance is calculated and saved as the response. The standard deviation of ten reads is also saved as the intra-cell variation.

In this research, 'X' cells are classified as cells with a mismatch difference between the 25th and 75th percentile, which is the middle 50%. Responses classified as '0' and '1' are in the bottom and top 25th percentile, respectively. However, when generating random numbers, this mismatch difference between 'X' responses is still too broad to produce bits that are likely to produce 50% '0's and '1's. Therefore, we propose a fourth response classification, 'super flaky' ('XX'), in this protocol. 'Super flaky' responses are responses with the 48th to 52nd percentile, the middle 4% of responses. This classification of bits is more likely to flip between a '0' or '1' when queried, making them more likely to be non-deterministic. A figure showing the classifications can be found in Fig. 4.11.

4.7.3 Testing and Methodology

To examine the strength of the MRAM TAPUF design using TAPKI, we generated

4.7.4 TRNG Protocols

In this research, we incorporated a variety of TRNG protocols: Cell Pairing (CP), Reference Cell (RC), and 'Super flaky' Cell Pairing (SFCP). The first two, Cell Pairing and Reference Cell, are merely for reference. These protocols generate mostly deterministic bits. However, they are useful in seeing any biases in the 'super flaky' cell pairing.

4.7.4.1 Cell Pairing

Cell pairing, similar to response generation in the MRAM TAPUF, involves comparing the analog responses obtained from two MRAM devices to generate responses. In this process, random cell pairs are generated,

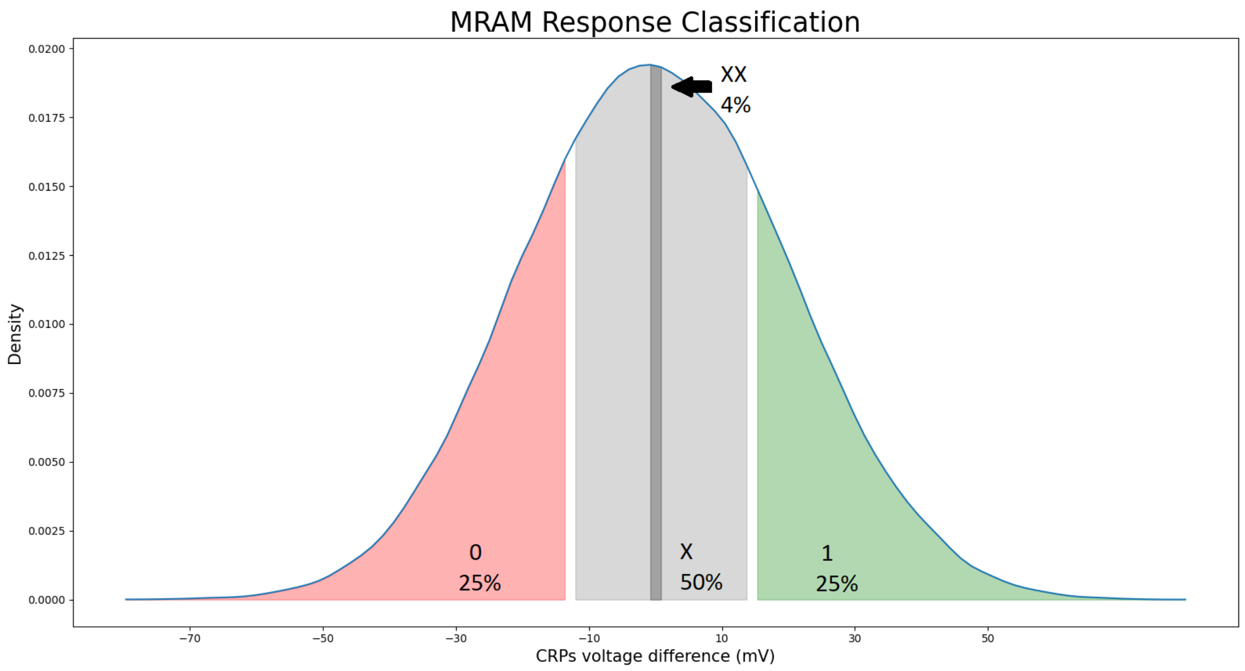


Figure 4.11: Classification of MRAM Responses

and the distances between their responses are calculated. The resulting pairs, known as CRPs (cell response pairs), are then classified based on their distances.

For classification, CRPs with distances in the top 25th percentile are considered strong ‘1’s, indicating a clear and distinguishable state. Conversely, CRPs with distances in the bottom 25th percentile are classified as strong ‘0’s. CRPs falling within the 25th to 75th percentile are labeled as ‘X’, representing a more ambiguous or uncertain state. Additionally, CRPs falling within the 48th and 52nd percentile are specifically categorized as ‘XX’, signifying an even higher level of uncertainty.

It is important to note that in cell pairing, the specific classification of the responses does not play a significant role, as all classifications (strong ‘1’s, strong ‘0’s, ‘X’, and ‘XX’) are utilized indiscriminately. The focus is on generating response bits by comparing random cell pairs.

4.7.4.2 Reference Cell

In the reference cell protocol, the electrical resistance of each cell is compared to the median resistance of the entire population. Theoretically, in this protocol, there should be an even number of ‘1’s and ‘0’s. Based on this comparison, cells are assigned response states using the following criteria:

Cells deviating significantly from the median resistance are classified as strong ‘0’s or ‘1’s, indicating a clear and distinct state. Cells that are close to the median resistance are labeled as ‘X’, representing a more ambiguous or uncertain state. The top and bottom 25th percentiles of electrical resistances are considered

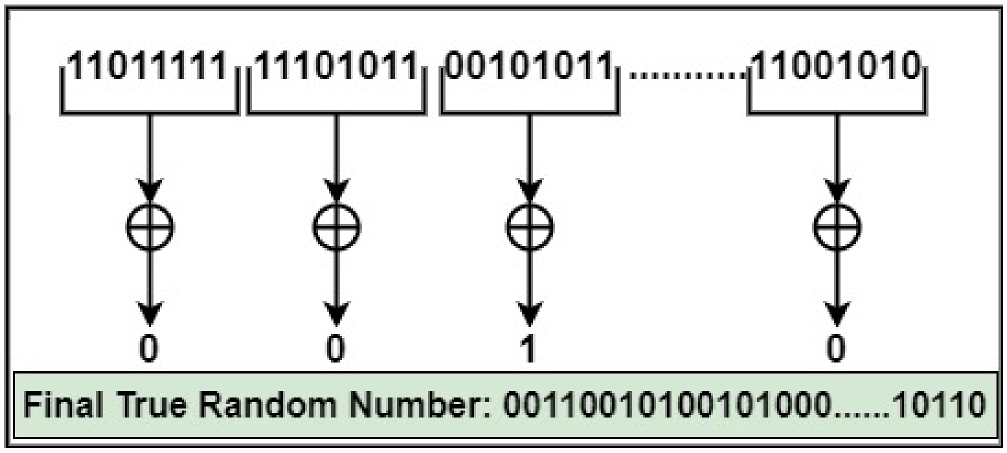


Figure 4.12: Internal XOR Compiler

as '1's and '0's, respectively, reflecting clearly distinguishable states. The range between the 25th and 75th percentiles of resistances falls into the 'X' category, indicating a less certain or ambiguous state. Additionally, a narrower range of resistances, specifically the 48th to 52nd percentile, is categorized as 'XX', signifying an even higher degree of uncertainty. Like the Cell Pairing protocol, all CRPs are used indiscriminately, focusing on generating random bits.

4.7.4.3 'Super flaky' Cell Pairing

'Super flaky' Cell Pairing is similar to Cell Pairing protocol, with the only difference being that it only selects super 'Super flaky' cells. In this process, a large number of pseudo-random CRPs is generated, and only the cells in the 48th to 52nd percentile get selected. These cells help ensure a more non-deterministic random sequence gets produced.

4.7.5 Post-processing Techniques

Different types of post-processing methods have been proposed. In [121], a random number generator used a SHA hashing function to post-process their random sequences. A SHA function is a great method of enhancing randomness, but it consumes more power than other lightweight methods. In this paper, we use two lightweight post-processing methods to improve the randomness of our sequences: Internal Bit XOR compiler and PRNG XOR compiler. This work uses the same post-processing techniques discussed in §3.5.4: internal XOR compiling and Pseudo-Random Number Generator (PRNG) XOR compiling.

4.7.5.1 Internal XOR Compiler:

A method of enhancing the randomness of random sequences is internal XORing. The XOR gate is a Boolean logic gate that generates a '1' if there is an odd number of 1s; otherwise, it generates a '0'. With internal XOR compiling, one is essentially sacrificing quantity for quality, reducing the size of the unfiltered random sequence by a factor of n. As shown in Fig. 4.12, a bit stream of True Random Number (TRN) is divided into n-bit substream. These n bits are XORed together to get a single bit out. Researchers found that the higher that n was, the higher the entropy of the random numbers would be [117, 120]. However, it was found that if the initial data stream were not random enough, the XOR compiler would not enhance the randomness.

Moreover, the internal XOR compiler only works in one direction, making it a great method of producing non-deterministic numbers. If only a small percentage of bits flip, the internal XOR compiler will increase that percentage. The number of bits changed will also increase as n increases.

4.7.5.2 PRNG XOR Compiler:

Pseudo-random numbers are often generated using mathematical methods [122] and are easily predicted, thus making them less random. This can highly affect the security of the system. To avoid using PRNG independently, we propose a method of post-processing the TRN by XORing it with PRN. We can have n numbers of Pseudo-Random Number (PRN) XORed to the TRN, thus enhancing the randomness as shown in Fig. 4.13, TRN from the MRAM PUF is XORed with a PRN, which is further XORed with a new PRN.

The PRNG XOR compiler also has the ability to change the bits given the same sequence completely. However, due to the reversible nature of the XOR gate, if an adversary were to get a hold of the pseudo-random numbers used to XOR the unprocessed random sequence, they could undo the post-processing and recover the unprocessed number.

4.7.6 Hardware Implementation

This hardware was originally implemented on a different client device than the one discussed in §4.3. However, for the research, the Chipkit Wi-Fire was employed as the client device. This particular developer kit offers several features, such as 43 available general-purpose input/output (GPIO) pins, 12 analog input pins, a 12-bit ADC, a power supply supporting both 3.3V and 5V, 2MB of flash memory, and 512K of RAM. The Wi-Fire is driven by a PIC32 microcontroller, which operates at a frequency of 200MHz and works with a 3.3V logic level.

It's important to acknowledge that the Wi-Fire has certain limitations. These include its operating

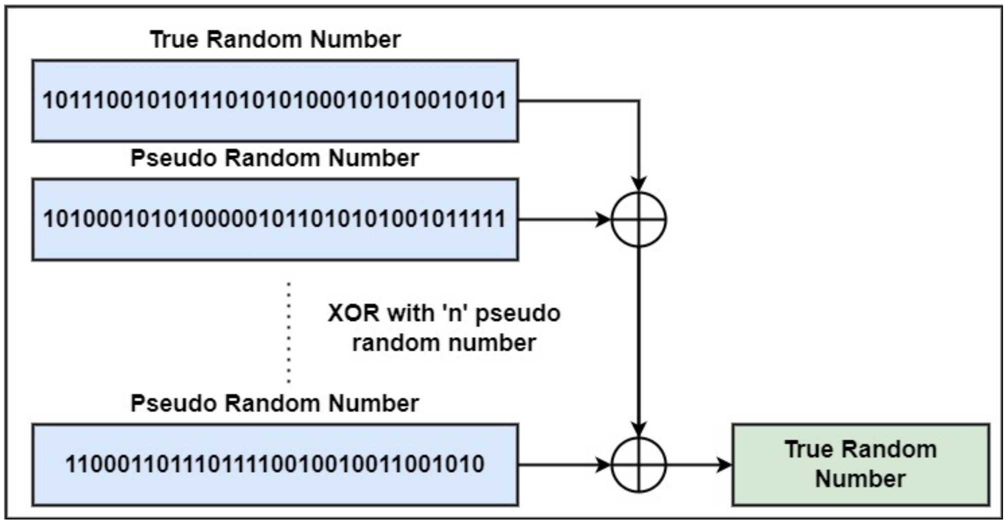


Figure 4.13: PRNG XOR Compiler

frequency, a restricted number of I/O and analog pins, and a fixed operating voltage. Consequently, due to these limitations of the client device, the random bit generation throughput was confined to 6 kbps.

The MRAM device utilized in this research is the Everspin MR4A16B 54TSSOP. Furthermore, the hardware employed to extract the electrical resistances from the MRAM devices remains unchanged from the hardware described in §4.3; it is simply presented in a different form factor. Figure 4.14 displays an image of the MRAM TRNG hardware installed on the Chipkit Wi-Fire.

4.7.7 Implementation of TRNG Protocols

In §4.4.2.1, it was discussed that the series resistance of the MRAM devices' CMOS circuitry plays a crucial role in determining the electrical resistances of MRAM cells, particularly when considering MRAM cells from different bit-lines within the circuitry. To address the issue arising from series resistance, this research treated responses obtained from different bit-lines as separate sub-PUFs. This approach significantly limited the number of CRPs available to select from and required the careful selection of MRAM devices with relative bit-line electrical resistances.

The Cell Pairing protocol begins with the enrollment process for each chip, where the chips with the most similar overall means in terms of electrical resistance (represented as voltage responses) are selected. Then, the medians of each bit-line are computed using the enrollment data. The closest bit-lines from adjacent MRAM devices are chosen and transmitted to the client device. Next, the client device utilizes a standard PRNG library to generate random challenge pairs, resulting in a random sequence. This random sequence is subsequently sent back to the server. Even with careful selection, the matching of bit-line medians was not always exact, and a small difference in medians was always evident.

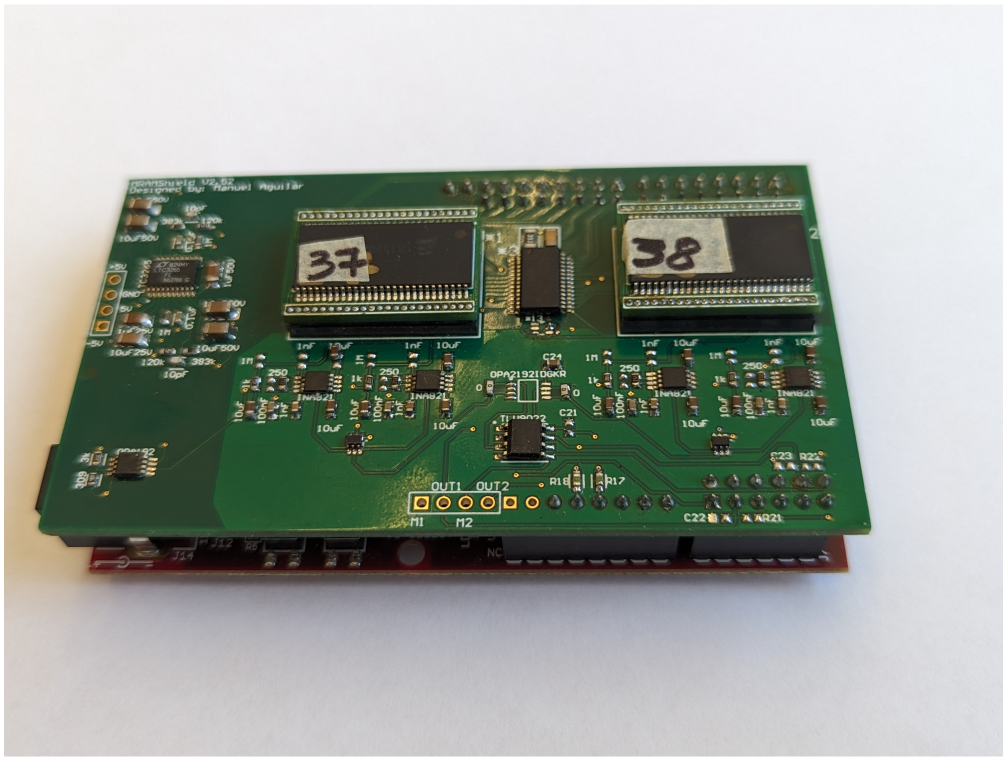


Figure 4.14: Chipkit Wi-Fire mounted with custom PCB used to interact and extract responses from MRAM device

In the Reference Cell protocol, the initial step involves selecting chips with relative overall means. The server calculates the bit-line medians for each bit using the enrollment data. Following this, for each bit-line, an MRAM cell on the opposing MRAM device that exhibits the exact same response as the median is selected. The bit-line, along with the location of the corresponding MRAM cell that matches its median, is then transmitted to the client device. The client device employs a standard PRNG library to generate random locations for CRPs within the MRAM device. These random locations within the MRAM device are compared to the MRAM cell that matches the median from its respective bit-line, resulting in either a '0' or '1'. The corresponding bits are subsequently sent back to the server.

The 'Super flaky' Cell Pairing protocol begins similarly to the Cell Pairing protocol, where the server will select MRAM devices with similar overall means and bit-lines from adjacent MRAM devices with the closest medians. After, the server will generate a pool of pseudo-random challenges. In this particular research, the initial number of challenges was set to 4096. From the initial challenge pairs, only the challenge pairs classified as 'XX' are selected and sent to the client device. The client device will then use the challenge pairs to generate a binary sequence. The binary sequence is then sent back to the server.

4.7.7.1 Non-deterministic properties

An essential characteristic of a random number system is whether it is deterministic or non-deterministic. Deterministic systems have predetermined outputs given the same inputs, meaning that given the same enrollments of the MRAM devices and challenges, they are likely to produce the same output. In other words, the outcome is mostly predictable and reproducible.

In contrast, non-deterministic systems are inherently unpredictable. They have different outputs, given the same input. A non-deterministic system has a certain amount of uncertainty that even if an adversary possesses a physical clone of the devices, they cannot acquire knowledge of the generated numbers.

The non-deterministic nature of random number systems ensures a higher level of randomness and security by preventing predictability and replication of the generated numbers, even with access to the physical components and challenges involved.

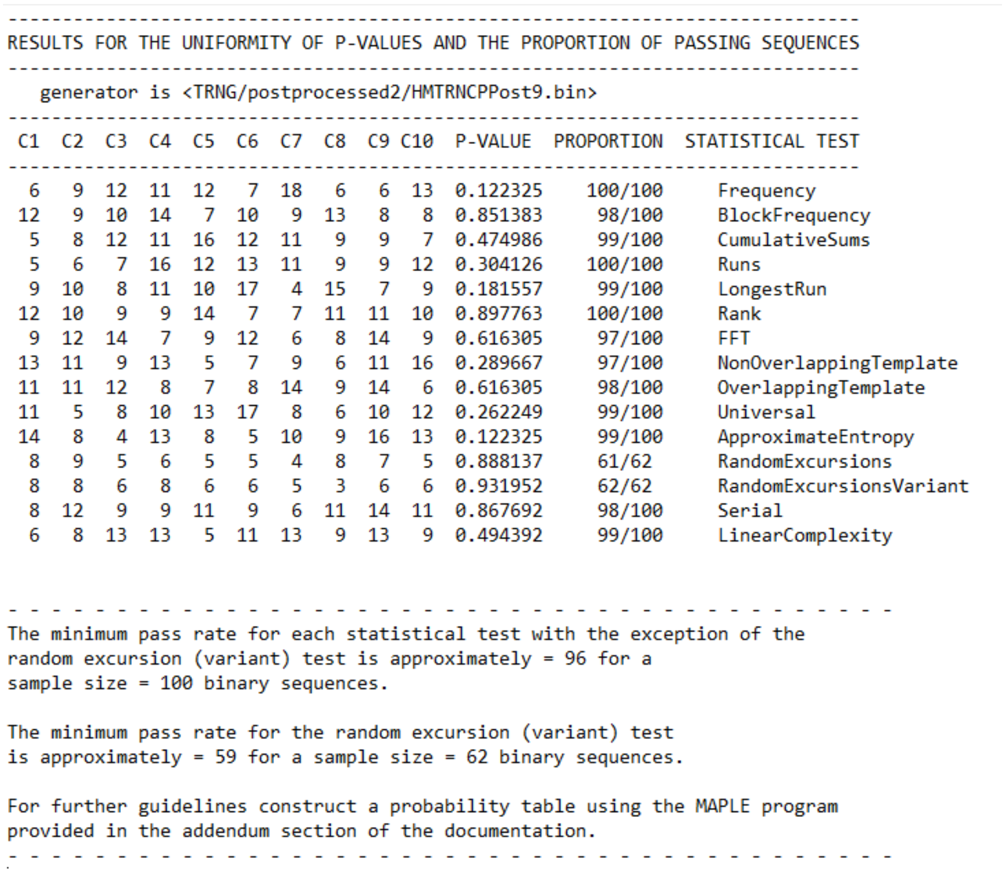
Each of these techniques involves using a PRNG to select challenges for response generation. In many PUFs, using the same challenges normally results in the same response generation. However, in this MRAM TAPUF configuration, 'flaky' and 'Super flaky' CRPs are left in the pool of pseudo-randomly selected CRPs. In the Cell Pairing and Reference Cells techniques, around four percent of the responses are classified as 'super flaky'. While this may seem significant, it might not be sufficient to ensure non-determinism. To further investigate this, the HD_{intra} of random sequences, 100,000 bits long, were calculated using Cell Pairing and Reference Cell. The HD_{intra} distance was around 0.93% and 1.71%, respectively. These differences in HD_{intra} were considerably smaller than expected and insufficient to be considered non-deterministic.

Furthermore, the non-deterministic responses of the 'Super flaky' Cell Pairing protocol were also investigated using the same method. The HD_{intra} for responses was 16.21%. Much less than expected but significantly better than other methods.

4.7.8 NIST Results

To assess the level of randomness in the random sequences produced by our proposed MRAM TRNG, we subjected these sequences to the NIST testing suite [29]. This suite encompasses 15 statistical tests designed to evaluate randomness. If a sequence successfully passes all 15 tests, it can be considered truly random. Conversely, if a sequence fails one or more tests, it is considered not truly random.

The random sequences generated using real-time responses were composed of 100 million bits. All sequences were sufficiently long to undergo all 15 tests. Each test was conducted with two different ALPHA values, namely 0.001 and 0.01. The ALPHA value signifies the significance level that defines the acceptance or rejection region in the tests. As per the NIST document [29], when using an ALPHA value of 0.001, one



```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <TRNG/postprocessed2/HMTRNCPPost9.bin>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
  6   9  12  11  12   7  18   6   6  13  0.122325    100/100     Frequency
 12   9  10  14   7  10   9  13   8   8  0.851383     98/100     BlockFrequency
  5   8  12  11  16  12  11   9   9   7  0.474986     99/100     CumulativeSums
  5   6   7  16  12  13  11   9   9  12  0.304126    100/100     Runs
  9  10   8  11  10  17   4  15   7   9  0.181557     99/100     LongestRun
 12  10   9   9  14   7   7  11  11  10  0.897763    100/100     Rank
  9  12  14   7   9  12   6   8  14   9  0.616305     97/100     FFT
 13  11   9  13   5   7   9   6  11  16  0.289667     97/100     NonOverlappingTemplate
 11  11  12   8   7   8  14   9  14   6  0.616305     98/100     OverlappingTemplate
 11   5   8  10  13  17   8   6  10  12  0.262249     99/100     Universal
 14   8   4  13   8   5  10   9  16  13  0.122325     99/100     ApproximateEntropy
  8   9   5   6   5   5   4   8   7   5  0.888137     61/62      RandomExcursions
  8   8   6   8   6   6   5   3   6   6  0.931952     62/62      RandomExcursionsVariant
  8  12   9   9  11   9   6  11  14  11  0.867692     98/100     Serial
  6   8  13  13   5  11  13   9  13   9  0.494392     99/100     LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 59 for a sample size = 62 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 4.15: NIST test results for the data collected using real-time responses

would anticipate rejecting one sequence out of every 1000, while an ALPHA value of 0.01 would result in rejecting approximately one sequence out of every 100.

According to the guidelines provided in [29], a P-value greater than or equal to 0.01 indicates that the sequence is considered random, while a P-value less than 0.01 suggests that the sequence is not random. In Figure 4.15, which represents one of the NIST test results, all the P-values are greater than 0.01. This result confirms that the generated bit stream is indeed random.

We tested all possible combinations, as shown in Table 4.7. The TRNs obtained directly from the MRAM PUFs failed the tests. However, after applying post-processing techniques to the data, the resulting bit streams successfully passed all the tests for randomness.

4.7.9 TRNG using MRAM TAPUF Conclusions

In this section, we proposed a method for generating true random bits using a different version of the MRAM TAPUF. Three protocols were employed to generate random sequences, resulting in a combination of deterministic and non-deterministic outputs. Subsequently, each random sequence underwent post-processing

Hardware PUF Data length = 100,000,000 ALPHA = 0.001	Cell Pairing	Reference Cell	'Super flaky' Cell Pairing
No post-processing	Fail	Fail	Fail
Internal XOR Size 2	Fail	Fail	Fail
Internal XOR Size 3	Pass	Pass	Fail
Internal XOR Size 4	Pass	Pass	Fail
Internal XOR Size 5	Pass	Pass	Fail
PRN XOR 1 Cycle	Pass	Pass	Pass
PRN XOR 2 Cycle	Pass	Pass	Pass
PRN XOR 3 Cycle	Pass	Pass	Pass

Table 4.7: Final Hardware PUF results of NIST Testing Suite. A pass means that it has passed all NIST Testing Suite tests, and a fail means that it has failed one or more of the NIST Testing Suite tests. This testing involves cell comparison techniques. We have also tested by selecting only the 'Super flaky' CRPs for TRNG. These methods have been tested in combination with post-processing techniques. XORing the TRN with a PRN proves to pass all the tests, whereas some tests fail using the Internal XORing technique.

using an internal XOR compiler and a PRNG XOR compiler.

In this implementation, the random sequences generated using real-time responses did not pass the NIST tests before post-processing. However, after applying the PRNG XOR compiler, all sequences produced by all protocols successfully passed the tests with just one cycle of XORing. However, when using the internal XOR compiler, it required an internal block size of four to increase the randomness of the sequences generated using the Cell Pairing and Reference Cell protocol to pass the NIST Testing Suite. 'Super flaky' Cell Pairing did not possess enough initial randomness to pass any test using the internal XOR compiler.

Despite carefully selecting bit-line pairs from different devices, the 'Super flaky' Cell Pairing protocol exhibited a slight bias in the selected CRPs. This bias resulted in a higher proportion of either '1' or '0' bits. Specifically, the generated bits showed a bias towards being over 60% '0's or '1's. Consequently, these biased sequences did not possess sufficient randomness to pass the NIST Testing Suite. Additionally, the internal XOR compiler could not enhance the randomness of these sequences enough to pass NIST tests.

Both semi-non-deterministic protocols, Cell Pairing, and Reference Cell, demonstrated sufficient initial randomness to pass the NIST tests. This was achieved with just one cycle of XORing when using the PRN XOR compiler and four cycles when employing the internal XOR compiler. These protocols are characterized by minimal set-up requirements and a lack of overhead, making them simple, fast, and efficient. However, their main drawback is their inherent determinism, which renders them vulnerable to attackers who have access to the CRPs of the PUF.

While Cell Pairing and Reference Cells exhibit comparable levels of initial randomness, Cell Pairing possesses a larger number of CRPs, making it more secure against potential attacks. On the other hand, the ‘Super flaky’ Cell Pairing protocol, which is non-deterministic, initially exhibits the least amount of randomness. Nevertheless, despite its limited initial randomness, it passed the NIST tests post-processed using the PRNG XOR compiler. It is important to note that this method of TRNG necessitates server intervention, resulting in increased computational requirements. However, the advantage is that it produces completely non-deterministic bits, ensuring their resistance to compromise, even if an attacker possesses PUF CRPs.

4.8 Enhanced TRNG using the MRAM TAPUF

To enhance the design of the MRAM TAPUF-based TRNG, we adjusted the MRAM responses by introducing a voltage value. This adjustment was discussed and implemented for cryptographic key bit generation in §4.4.2.1. The aim was to eliminate the constant resistance pattern that existed in the series. By applying a constant voltage to all responses from a particular bit-line (e.g., 0, 1, 2, ... 15), the entropy of the responses from those bit-lines remained intact and unaffected.

To achieve the desired resistance adjustment, we incorporated a DAC into our system. This DAC was connected to the voltage reference pins of our amplifiers. Since the measurement of cell resistance relies on voltage drops, we introduced a voltage value to offset the measurements, ensuring that all cells, regardless of their bit-lines, shared the same median values. While this approach required additional circuitry and slightly increased complexity during the enrollment process, it effectively reduced the disparities between response medians. As a result, it significantly improved the randomness of the sequences generated by the MRAM TRNG.

4.8.1 Client Device and Hardware

The implementation of this approach utilized the same client device and hardware as described in §4.3. With this hardware configuration, we achieved a throughput of 33.33 kbps.

4.8.2 Enrollment

The device enrollment protocol involves the following steps. First, all MRAM cells are set to a common state, either a high or low resistive state. This ensures consistency for measurement. Next, the resistance of each MRAM cell is measured multiple times, denoted by r. For each cell, the average resistance value obtained from the multiple measurements is stored as the resistance value for that cell, representing the typical cell behavior. Additionally, the standard deviation of the resistance readings for each cell is calculated and stored as the intra-cell variation, which captures the variability within each cell’s resistance values. By storing the resistance and intra-cell variation for each MRAM cell, the enrollment process establishes a baseline understanding of the device's behavior and characteristics. This information can then be used for subsequent operations and analysis.

The enrollment process for this research involves two main steps.

Firstly, in the initial part of enrollment, the goal is to determine the resistance of each MRAM cell without any adjustments. This step focuses on identifying and recording the individual resistances of the MRAM cells or other relevant characteristics.

Subsequently, in the second step, the difference in bit-line medians is calculated based on the recorded responses from the previous step. This calculated difference is then transmitted to the client.

On the client device, the enrollment continues with another round, but this time with resistance adjustments included as part of the procedure.

4.8.3 Random Bit Generation

The overall design and random number generation scheme remained unchanged from work done in §4.7. The protocol begins by selecting random CRPs on the server and then filtering for CRPs that are marked as 'XX.' As mentioned in §4.7, 'XX' is a classification of CRPs that are more likely to flip between the bit '1' and '0'. As only around 4% of cells are classified as 'XX,' more random CRPs must be selected than required.

After selecting the CRPs, the server will send the challenges to the client along with the bit-line median mismatches. Once the client has received the challenges and bit-line medians, it will adjust the medians accordingly and generate a random sequence.

By shifting the medians of the bit-line responses, the limitation on the number of CRPs has been eliminated compared to the previous approach. Furthermore, the requirement for careful selection of MRAM devices with relative resistance responses is no longer necessary, as all responses will be adjusted to be relative.

4.8.4 Post Processing

TRNGs generate truly random sequences by relying on natural phenomena. However, these sequences can sometimes exhibit occasional correlations and biases, which reduce their randomness. This issue is particularly relevant for analog sources of randomness, as they can be influenced by external disturbances like electromagnetic interference (EMI) [119].

To address this, TRNGs incorporate a post-processing stage to improve the statistical properties of the

	Unprocessed	**Pseudo-random Number XOR Compiler**	**Internal XOR Compiler**
Intra-chip Hamming Distance	0.1923	0.5038	0.5030

Table 4.8: Difference between random sequences produced by the same challenges in 'Super flaky' Cell Pairing. The number of XOR cycles taken by the pseudo-random number XOR Compiler is 1; the chunk size used for the internal XOR Compiler is 7.

generated random sequences. The same post-processing procedures used in §4.7.5 will also be used for this research PRNG and internal XOR compiling.

4.8.5 Non-deterministic properties

To test the non-deterministic nature of our random numbers using unstable pairs, we generated multiple random sequences using the same challenges, with and without post-processing. For post-processing with the pseudo-random number XORing compiler, a seed based on the system time was used. To quantify the difference, we took the intra-chip Hamming distance [22] of random sequences. The results of these tests can be found in Table 4.8.

After analyzing the results, we observed that the random sequence exhibited a difference of $\sim 19\%$ without any post-processing. Ideally, this difference should be $\sim 50\%$ for CRPs that are classified as 'XX.' However, since cross talk and EMI can significantly impact the responses of CRPs that are on the threshold between being interpreted as a '0' or a '1', such deviations are expected.

When applying an XOR post-process with a chunk size of 7 (denoted as $n = 7$), the difference between the sequences increased to $\sim 50\%$. Similarly, the Pseudo-random Number XOR compiler also yielded a difference of $\sim 50\%$ when the cycle was $n = 1$.

It's important to note that the pseudo-random number XOR compiler, while effective, is deterministic and heavily relies on the pseudo-random number generator and system time to enhance randomness. If an attacker gains access to either of these variables, they can reverse the post-processing and revert to sequences with a difference of approximately $\sim 19\%$.

4.8.6 NIST Result

To evaluate the level of randomness in the random sequences generated by the Enhanced MRAM TRNG, we conducted tests using the NIST testing suite [29]. The NIST suite consists of 15 statistical tests designed to assess the randomness of a given sequence. If a sequence successfully passes all 15 tests, it can be considered genuinely random. However, if a sequence fails one or more tests, it is deemed not truly random.

Hardware MRAM TRNG **Data length = 100,000,000** **ALPHA = 0.001**	Unadjusted MRAM TRNG 'Super flaky' Cell Pairing	Adjusted MRAM TRNG 'Super flaky' Cell Pairing
No post-processing	Fail	Pass
Internal XOR Size 2	Fail	Pass
Internal XOR Size 3	Fail	Pass
Internal XOR Size 7	Fail	Pass
PRN XOR 1 Cycle	Pass	Pass
PRN XOR 2 Cycles	Pass	Pass
PRN XOR 3 Cycles	Pass	Pass

Table 4.9: Hardware PUF results of NIST Testing Suite. A pass means that it has passed all NIST Testing Suite tests, and a fail means that it has failed one or more of the NIST Testing Suite tests. This testing compares 'Super flaky' cell pairing using unadjusted responses versus adjusted responses. These methods have been tested in combination with post-processing techniques.

We generated random sequences that were 100 million bits in length, which provided a sufficient length for passing all 15 tests. Each test was performed with an alpha value of 0.01. According to the NIST document [29], when using an alpha value of 0.001, it is anticipated that approximately one out of every 1000 sequences would be rejected by the test. The results of the NIST Testing Suite can be found in table 4.9.

Based on the results obtained, it was observed that the true random numbers generated using the 'Super flaky' cell pairing method with unadjusted MRAM responses did not possess sufficient initial randomness. Consequently, these numbers did not meet the required criteria even with post-processing or any level of internal XOR Compiling. Only the pseudo-random number XOR compiler was sufficient enough post-processing to pass the NIST Testing Suite.

On the other hand, the true random numbers generated using the 'Super flaky' cell pairing method with shifted MRAM responses demonstrated adequate randomness. They successfully passed the required criteria without the need for any post-processing, and they remained suitable for further post-processing as well.

These findings confirm the effectiveness of shifting the MRAM responses in improving the entropy of the true random number sequences. The shifting process significantly enhanced the randomness of the generated numbers, enabling them to meet the required standards.

4.8.7 Enhanced TRNG using MRAM TAPUF Conclusions

This work aimed to enhance the 'Super flaky' cell pairing method of the MRAM TRNG proposed in §4.7. The improvement involved shifting the responses, which resulted in generating non-deterministic bits with sufficient entropy to pass the NIST Testing Suite without requiring post-processing. Moreover, the throughput was maintained at 33.33 kbps, equivalent to 30 microseconds per bit. This indicates that the TRNG continued to operate at the same speed as before, ensuring the efficient generation of random numbers.

Further research was conducted to analyze the differences between two random sequences generated using the same challenge. It was discovered that, despite utilizing the 'Super flaky' cell pairing method, only approximately 19% of the bits would flip when querying challenges within a short time frame.

However, through post-processing, the number of flipped bits increased to 50% for both the internal XOR compiler ($n = 7$) and the pseudo-random number XOR compiler ($n = 1$). It is worth noting that the pseudo-random number XOR compiler, although effective, relies on the pseudo-random number generator and system time to enhance randomness. If an attacker gains access to either of these variables, they can reverse the post-processing and revert to sequences with a difference of around 19%.

Therefore, for more secure non-deterministic random sequences, it is recommended to utilize the internal XOR compiler, even if it reduces the number of generated bits by a factor of n. This approach mitigates the risks associated with the pseudo-random number XOR compiler methods and provides improved security in generating non-deterministic random sequences.

By maintaining the previous throughput while enhancing the entropy of the generated bits, this work balanced speed and randomness, making the improved 'Super flaky' cell pairing method of the MRAM TRNG a practical solution for applications that require both high-speed operation and strong randomness.

Chapter 5

Pre-formed ReRAM

5.1 Introduction

Most cryptographic systems rely on traditional cryptographic keys for encryption. Although these keys are currently considered secure against most types of attacks, they are associated with several vulnerabilities, including issues related to key storage, generation, and distribution [18, 17, 123, 13]. Moreover, the advancement of quantum computing poses a significant threat to many conventional cryptographic algorithms that are based on keys, such as Advanced Encryption Standard (AES), Elliptic Curve Cryptography, and Rivest-Shamir-Adleman (RSA) [15, 16]. As a result, there is a growing need for alternative methods to address these challenges.

Bertrand et al. introduced a novel approach called analog key encapsulation using pre-formed Resistive random-access memory (ReRAM) Physically Unclonable Function (PUF). This encryption method, referred to as analog key encapsulation, eliminates the need for a key generation or decryption [100, 105, 106, 124]. It utilizes the analog responses of a ReRAM PUF to encapsulate a message or key, which can only be retrieved using the corresponding pre-formed ReRAM PUF's responses.

The objective of this research is to implement an analog key encapsulation protocol utilizing physical pre-formed ReRAM devices powered by a low-power client device. This study builds upon the previous work in [100] and [105]. In the earlier research, the authors utilized a software file derived from the pre-formed ReRAM as a software PUF. These software files contained a finite set of ReRAM responses for each challenge, with each challenge randomly selecting one response from the set when invoked to mimic a physical PUF. Additionally, the responses were collected using a precision semiconductor device analyzer, which offers greater precision and less noise compared to packaged Integrated Circuit (IC) pre-formed ReRAM devices.

This work will propose a unique pre-form ReRAM-based Analog key encapsulation method that differs slightly from the ones mentioned in [100, 105, 106]. It will use real-time pre-formed ReRAM responses from a packaged device and be implemented on a low-power device.

5.2 Packaged ReRAM Devices

In this research, we use Crossbars 1B Chip as our ReRAM PUF. The 1B is a pre-formed ReRAM PUF that is not commercially available and is made specifically for use as a PUF. It contains 4096 pre-formed resistors capable of creating a response by applying a voltage or current through the cells. These can be applied to the bottom or top pads of the cell for forward or reverse reading. The applied voltages range from 0.15V to 1.5V, and the currents range from 79nA to 15μA using 75 predefined currents. There are two modes of reading operations, analog and digital. Analog read mode provides direct access to the cells to measure their resistances and distribution. On the other hand, digital read mode provides a '0' or '1', generated using internal sense amps that compare a cell's response to tunable values. The 1B uses the same memristor cells as Crossbar Inc.'s 1A wafer, which was the subject of electrical characterization and testing and has shown promising results [6]. The expected responses of the ReRAM PUF range from 100mV to 2V.

5.3 Client Device

Both implementations of the Analog Key Encapsulation using pre-formed ReRAM employed the Chipkit Wi-Fire as the client device. This particular developer kit offers several features, including 43 available general-purpose input/output (GPIO) pins, 12 analog input pins, a 12-bit Analog-to-Digital Converters (ADC), a 3.3V and 5V power supply, 2MB of flash memory, and 512K of RAM. It has a PIC32 microcontroller operating at a 200 MHz frequency and a 3.3V logic level. However, it is important to note that the Wi-Fire has certain limitations, such as its operating frequency, a limited number of I/O and analog pins, and a fixed operating voltage.

5.4 Implementation Using Software Reads

A crucial aspect of this protocol is the estimation of the current array. In the initial implementation of analog key encapsulation using packaged pre-formed ReRAM on low-power devices, software reads from an ADC was utilized. This involved reading the response of a pre-formed ReRAM cell and comparing it to the expected response, enabling the effective utilization of current guessing methods.

5.4.0.1 Hardware Implementation

The analog key encapsulation protocol was implemented using a custom Printed Circuit Board (PCB) which is specifically designed for ReRAM key encryption protocols. This PCB is intended for use with the ChipKit Wi-Fire microcontroller and utilizes Crossbars 1B Chip as a ReRAM PUF. The PCB has two slots for two

ReRAM chips. The Crossbar 1B chips are equipped with 4096 preformed memristors that are specifically designed to function as a PUF. Although the shield was not originally designed for analog key encapsulation, its ability to read the memristor responses of the ReRAM PUFs enables the implementation of analog key encapsulation protocols.

Due to the utilization of various hardware components in a limited space for communication with the ReRAM PUFs, thePCB in the shield introduces a significant amount of noise. As a result, the measurement variation of certain cells can be substantial, leading to inaccuracies within the protocol. During the enrollment process, where each cell was read 20 times at a specific current, the average standard deviation of cell reads was 10mV, with the highest recorded value being 395mV. Such measurement variations pose significant challenges for decryption, increasing the likelihood of false positives and negatives.

To address this issue, capacitors were added to the outputs of the ReRAM PUFs to mitigate the measurement variations. However, it is important to note that the shield's design primarily focuses on key encryption methods. As a result, the capacitor locations are closer to the comparator inputs and farther away from the ADC. This placement renders them less effective in mitigating the measurement variations caused by noise.

Furthermore, given the significance of maintaining the integrity of analog signals in this design, enrollments were performed in each slot. This was done to assess and measure any potential discrepancies in noise between the two slots. A picture of the hardware used for this research is found in Figure 5.1

5.4.1 Enrollment

In this protocol, each encryption block requires four bits. As a result, out of the 75 available currents, sixteen specific currents were selected for the encryption process. These sixteen currents were chosen based on their maximum separation from each other. It was observed that currents further apart from each other had a lower likelihood of causing decryption errors. Importantly, the same set of sixteen currents was used for every ReRAM device involved in the protocol.

In the enrollment process, the response from each cell is measured an r number of times for each of the sixteen currents. The mean of those r responses is stored as the response, while the standard deviation of those cells is stored as an intra-cell variation. The r for this implementation was selected to be at 20.

5.4.1.1 Flaky Cells

To enhance the accuracy of current guessing, the protocol excluded cells considered 'flaky'. A cell was labeled as 'flaky' if its intra-cell variation exceeded a threshold value of f. In this implementation, the chosen threshold value, f, was set at 16mV.

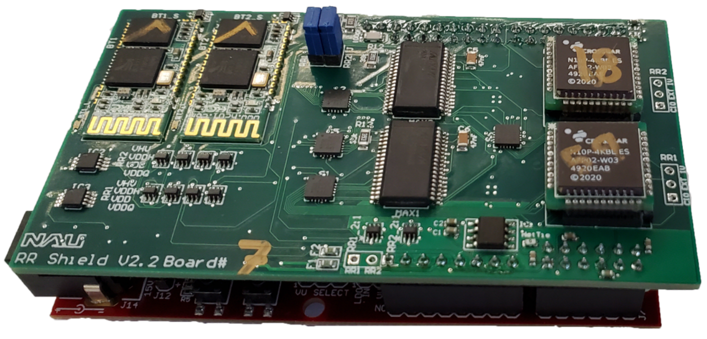


Figure 5.1: Chipkit Wi-Fire mounted with custom PCB used for analog key encapsulation

The selection of f involved a testing process. Initially, f was increased, and the successful decryption rate was measured for each increment. The testing continued until reaching a point where further increases in f no longer improved the decryption rate. At this stage, the value of f was determined and set as the optimal threshold for identifying 'flaky' cells.

5.4.2 Analog Key Encapsulation using Pre-formed ReRAM Protocol

The main objective of this analog key encapsulation implementation is to demonstrate the reliability of the responses generated by the ReRAM PUFs for physical encryption. As a result, this implementation is simpler than other analog key encapsulation protocols.

The protocol is between a server with an image of PUF A and a client with the physical PUF A. The server and the client will have the same password (PW). In this implementation, the client is aware of the locations of the "flaky" cells, which may produce unreliable responses. This information is stored in non-volatile memory, allowing the client to avoid using these specific locations during the encryption process.

The encryption side of this protocol for this implementation is as follows:

- The first step of encryption is to generate the message digest MD. The server will generate an RN, T, using a TRNG and then perform an XOR operation on the PW using T. A hash function will hash the result of this XOR operation to generate the original message digest (MD'), which is 64 bytes long. MD' will then be expanded by taking the first 16 bits of theMD' and rotating them 16 times to get 16 different message digests that are the same size as MD'. The 16 MD' will be concatenated into one message digest MD (1024 bytes) containing the address array A. Two bytes will be allocated for each address in the address array.

- This protocol's second step is generating the current array, Q. The server will take its plain text message (M) and convert every character into ASCII (1 byte) to generate an ASCII array. The ASCII array will then be divided into 4-bit blocks, with each of those 4 bits becoming a current value (0-15). If M is not in plain text and already binary, the object can be directly split into 4-bit blocks. Once the Q is generated, each current is paired with an address in A. If there are more addresses than currents, the left-over addresses will not be used. A terminating byte sequence will let the decrypting user know that not all 1024 bytes of the message digest will be used. Since there cannot be more currents than addresses, the message encrypted is limited to 256 characters per encryption, or one-fourth the size of the MD.

- The third step of this protocol is to generate the response array, RA. The server will generate RA by using Q, A, and the image of PUF A to look up the responses of specific cells at a particular current. The response of ReRAM cells in this implementation will be kept as a voltage. While the response of the ReRAM cell is technically a resistance, since the current, I, is known, the two can be used interchangeably. After RA is generated, the server will initiate a handshake with the client, exchanging RA and T.

The decryption side of this protocol is as follows:

- The first step in the decryption process is to generate the same MD, using T and PW, with the same steps as the server.
- The second step will use the MD and the RA to generate Q'. The generation of Q' can vary; however, the most efficient and reliable method should be used to keep the number of errors low. Once Q' is generated, the client can use Q' to extract the original encrypted message, M.

An image of the encryption and decryption methods can be found below in Fig. 5.2 and Fig. 5.3.

5.4.3 Current Guessing Methods

In this implementation, two current guessing methods were used.

5.4.3.1 Current Guessing Method 1

In the first current guessing method, a cell is read multiple times at each current iteration n until a matching response is found. If the memristor response falls within the acceptable range r of the actual response, that particular current will be identified and returned. However, if no current falls within the acceptable range, an error will be returned as no suitable current was found.

For this specific implementation, the value of r chosen was 35. The selection of this value was determined through a testing process. Starting from an initial value of five, the successful decryption rate was measured for each increment of five. The testing continued until reaching a point where the successful decryption rate no longer increased. At this point, the value of r was set to 35 as the optimal choice based on the observed results.

5.4.3.2 Current Guessing Method 2

In the second current guessing method, a cell is read multiple times at each current for a total of t reads. The responses obtained from these reads are averaged. The average value closest to the target response value is determined and returned.

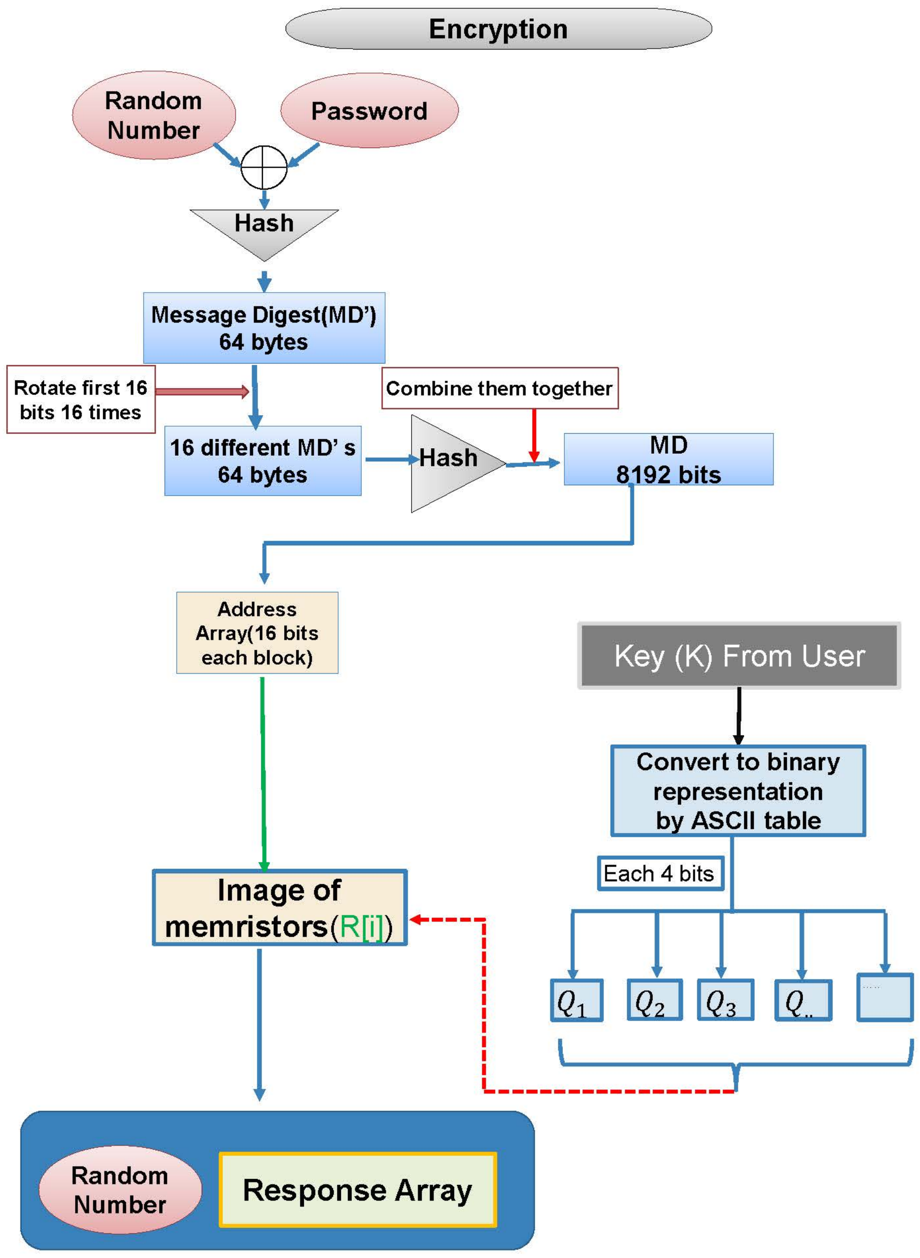


Figure 5.2: Analog Key Encapsulation encryption protocol

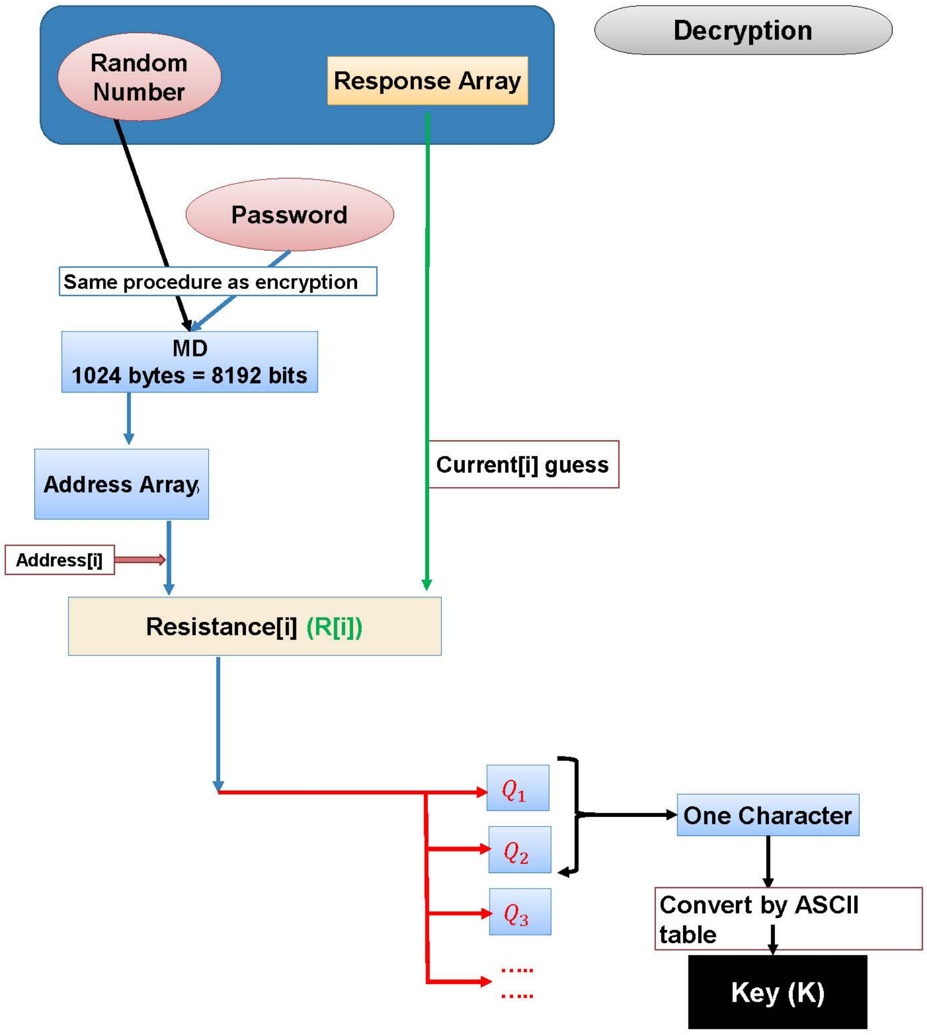


Figure 5.3: Analog Key Encapsulation decryption protocol

In this particular implementation, the value of t was set to 10. The selection of this value was determined through a testing process. Initially, the successful decryption rate was measured using a starting value of five for t. Subsequently, t was incremented by five in each test iteration. The testing continued until a point was reached where further increments of t no longer increased the successful decryption rate. Based on this analysis, the optimal value for t was found to be 10.

Figures 5.4 and 5.5 provide block diagrams illustrating both current guessing methods.

5.4.4 Testing

The protocol implementation involved two main components: a personal computer (PC) acting as the server, running `Python 3` code, and a client device consisting of a Chipkit Wi-Fire, a custom PCB, and physical ReRAM devices. The client device was programmed using `C++`.

The protocol began with the enrollment process, where the physical ReRAM devices were registered and their data stored on the PC using `Pandas` data frames. Once the enrollment was completed, the server-initiated communication with the client. It started by sending a random number to generate the message digest.

After receiving the random number, the client device performed several steps. First, it received a mask from the server, which indicated which cells should be excluded from further operations involving variable A. Additionally, the client device received the value RA, enabling it to generate Q'.

Using the received mask and RA, the client device proceeded to generate Q' and convert the ASCII representation into an unencrypted message.

5.4.4.1 Results

After enrolling the identical ReRAM PUF in both slots, each method was separately tested in their respective slots at a temperature of 23°C. The testing procedure involved encrypting and decrypting a 10-character message, namely "HelloWorld," for a total of 1000 iterations in each test conducted at 23°C.

Considering the hardware's limitations and optimization, the decryption process using the current guessing method 1 required approximately one second per character. On the other hand, the decryption process using current guessing method 2 took approximately 3 seconds per character.

Furthermore, to assess the performance of each method, an incorrect PUF, different from the one enrolled, was utilized to determine the number of false positives. It is important to note that the tests were conducted with flaky cells removed. The detailed results of each test can be found in Table 5.1.

Upon testing both methods using actual ReRAM PUFs, it was observed that method 2, which utilized

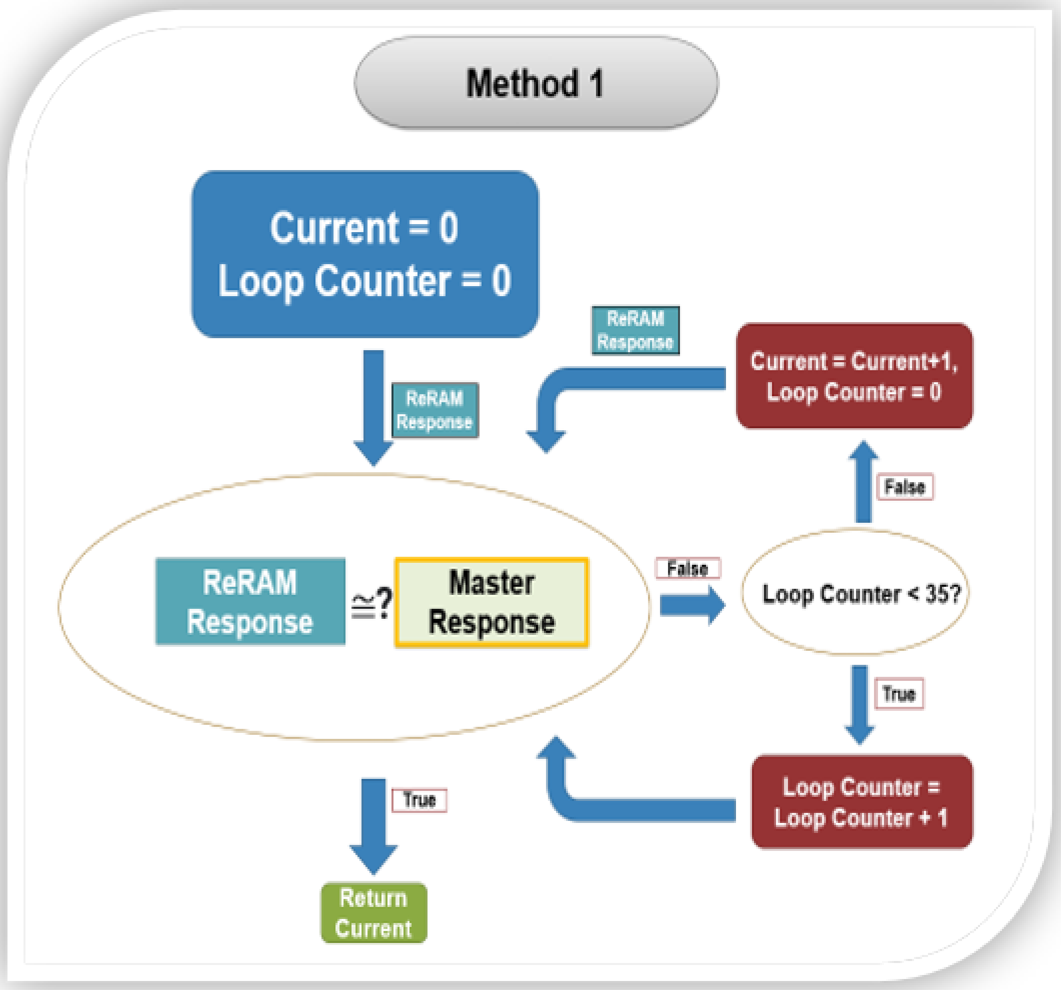


Figure 5.4: Current Guessing Method 1

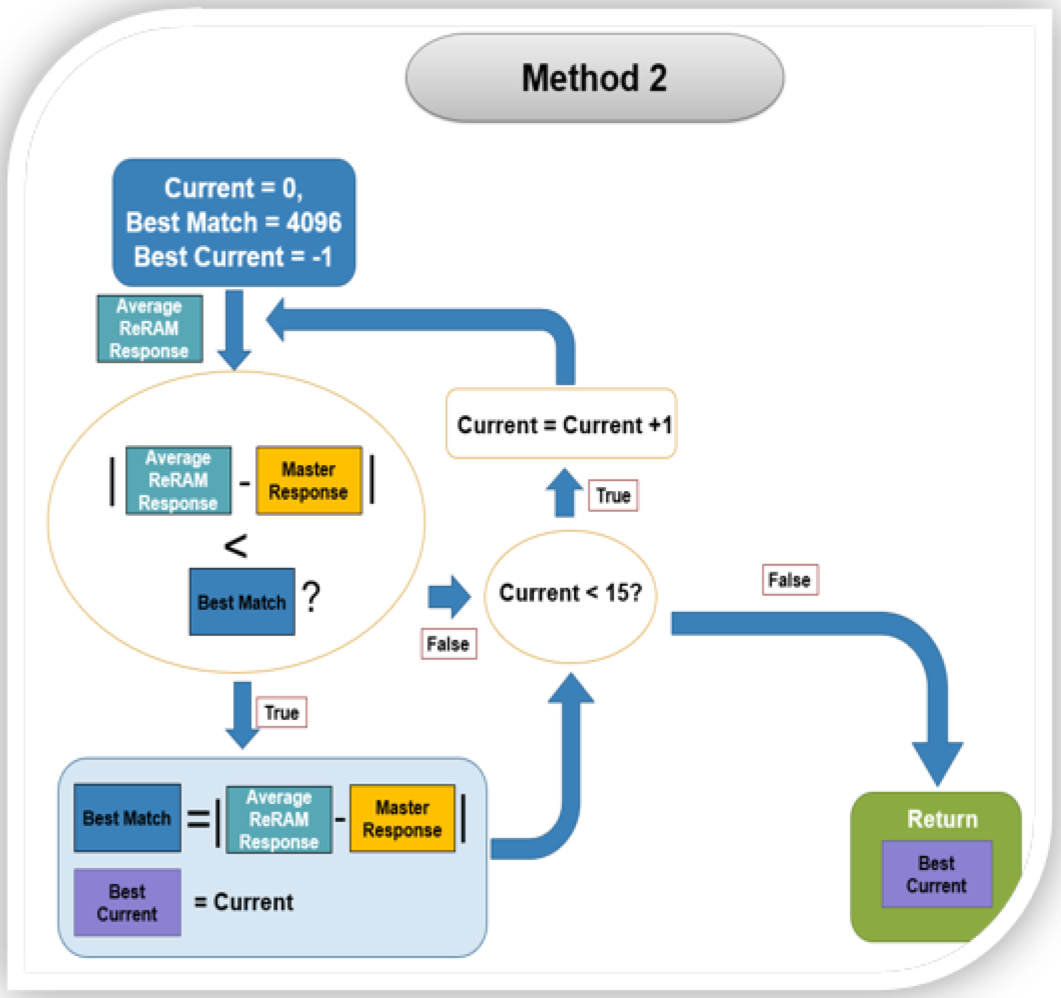


Figure 5.5: Current Guessing Method 2

Successful Character Decryption Percentage	**Slot 1**	**Slot 1 Flaky Cells Removed (1487)**	**Slot 2**	**Slot 2 Flaky Cells Removed (1348)**
Current Guessing Method 1	45.8%	58.0%	62.7%	68.9%
Current Guessing Method 2	83.5%	90.9%	89.8%	97.1%
Current Guessing Method 1 Incorrect PUF	NA	0.2%	NA	0.3%
Current Guessing Method 2 Incorrect PUF	NA	6.7%	NA	6.5%

Table 5.1: Analog Key Encapsulation Testing Results

the optimal value, exhibited the highest success rate for implementing the analog key encapsulation protocol with software reads. However, it did result in a slightly higher false success rate of around 6.5-6.7%. It is worth noting that this percentage is equivalent to randomly guessing a number between zero and sixteen. Therefore, it becomes indistinguishable whether the guessed values are correct or false. The impact of false positives becomes significant when the percentage reaches a certain threshold. In such cases, a potential attacker could exploit partial knowledge of the key to decrypt the remaining message, particularly if the key follows a standard plain-text sentence structure.

As expected, current guessing method 1 exhibited lower overall accuracy, but it yielded significantly fewer false positives, with a rate of only 0.3%. However, even in current guessing method 2, the false positive percentage was equivalent to randomly guessing a number between 0 and 16. This indicates that method 2 remains preferable and more suitable for practical applications, especially within typical hardware environments with moderate measurement variations.

Furthermore, it was observed that the second slot consistently exhibited a higher successful decryption rate compared to the first slot for both methods. This finding can be attributed to the higher measurement variations recorded in the first slot. It is hypothesized that the elevated measurement variations and noise in the first slot are likely attributable to its specific location on the shield. Consequently, effective noise mitigation techniques are paramount for future hardware designs to ensure optimal performance and accuracy.

5.4.5 Software Reads Implementation Conclusions

This research focused on implementing an analog key encapsulation protocol using packaged pre-formed ReRAM devices on a low-power client device. The main objective was to assess the reliability of real-time responses generated by these low-power client devices for successful encryption and decryption. The protocol employed two different current guessing methods to decrypt.

In our findings, we observed that employing a current guessing method that utilized the closest matching average value achieved the highest success rate of 97.1%. These results demonstrated the feasibility of analog key encapsulation encryption, as it consistently provided reliable outcomes when utilizing physical ReRAM PUFs.

However, it is crucial to note that employing software-based comparisons poses a security vulnerability. This vulnerability arises because it grants attackers access to the analog responses of the PUF, enabling them to enroll the PUF. Furthermore, the hardware used in the implementation exhibited considerable latency, taking approximately 10-30 seconds to decrypt a 10-character message. To overcome these challenges and further improve the protocol, the next step involves developing a dedicated hardware shield specifically designed for analog key encapsulation encryption. This hardware shield should incorporate a hardware-based comparison protocol aimed at enhancing both the efficiency and security aspects of the system.

5.5 Improved Analog Key Encapsulation Method using Hardware Comparisons

In this section, we aim to enhance the previous work described in §5.4 by implementing the analog key encapsulation protocol using specialized hardware. Like the previous implementation, this work will utilize real-time responses from packaged pre-formed ReRAM chips and will be conducted on a low-power client device. However, in this case, a custom PCB is employed to interface with the pre-formed ReRAM devices, enabling hardware-based comparison of analog signals. This design showcases the efficiency and security enhancements achieved by leveraging hardware capabilities instead of relying on software-based approaches.

5.5.1 Hardware

5.5.1.1 Client Device

The client device used to operate the circuitry is the Chipkit Wi-Fire developer kit discussed in §5.4.

5.5.2 PCB Hardware

An essential part of this protocol is the method by which the current array is estimated. In previous implementations, this part of the protocol was done using an ADC, where the microcontroller would compare in software to see if the response matched the expected result. Software comparisons are slow, inefficient, and a security risk because a hacker can use a device that reads responses directly into memory to make a digital copy of a PUF. As a result, using hardware circuitry is ideal because it is efficient and safer.

To estimate the correct current, given a ReRAM cell and the expected response, one needs to find the current that best matches that response. While this may seem trivial, noise, environmental factors, and intra-cell variation make this challenging. Therefore, variations of the signal and signal integrity must be considered when estimating the current to avoid type one and two errors.

In software, it is straightforward to read a response and compare it to the expected response; however, hardware is slightly more complicated. Two analog inputs must be fed into a hardware circuitry that produces either an analog or digital output for the hardware implementation. A basic hardware schematic was laid out for estimating the current. In this schematic, a simple Digital-to-Analog Converter (DAC) would be used to output the expected response, and the ReRAM PUF would be used to output the ReRAM cell response. Both responses would then be fed into two differential amplifiers (gain = 100), one configured with the ReRAM response on the non-inverting input and the other configured with the ReRAM response on the inverting input. The outputs of the differential amplifiers are connected to the non-inverting input of two comparators. Both inverting inputs of the comparators are tied to a threshold voltage that another DAC produces. In this configuration, the threshold would be set by a DAC, where the threshold between the two signals is the voltage set by the DAC divided by the gain. An image of this schematic can be found below in Figure 5.6.

This circuitry has four cases which are as follows:

- Case 1: DAC > RR, where the difference between DAC and RR exceeds the threshold set. G1 > DAC Threshold, G2 ≤ 0, $DAC > RR$ = HIGH, $RR > DAC$ = LOW.
- Case 2: RR > DAC, where the difference between DAC and RR exceeds the threshold set. G1 ≤ 0, G2 > DAC Threshold, $DAC > RR$ = LOW, $RR > DAC$ = HIGH.
- Case 3: DAC $\geq$ RR, where the difference between DAC and RR is lower than the threshold set. G1 < DAC Threshold, G2 ≤ 0, $DAC > RR$ = LOW, $RR > DAC$ = LOW.
- Case 4: RR $\geq$ DAC, where the difference between RR and DAC is lower than the threshold set. G1$\leq$ 0, G2 < DAC Threshold, $DAC > RR$ = LOW, $RR > DAC$ = LOW.

In this hardware configuration, if both output signals are low, there is a match between the ReRAM response and the expected output. It also lets the microcontroller know if the DAC is too high or too low, allowing it to tune the current, increasing the current if the DAC is greater than the ReRAM response and decreasing the current if the ReRAM response is greater than the DAC. It also allows a user to tune the threshold voltage, which can be tuned to a precision in the μV range.

The differential op-amp used in the current guessing hardware configuration is very important. The op-amp must take input values within the range of all ReRAM responses, amplify the differential signal to the desired gain, and have enough precision to differentiate between extremely close signals. The comparator

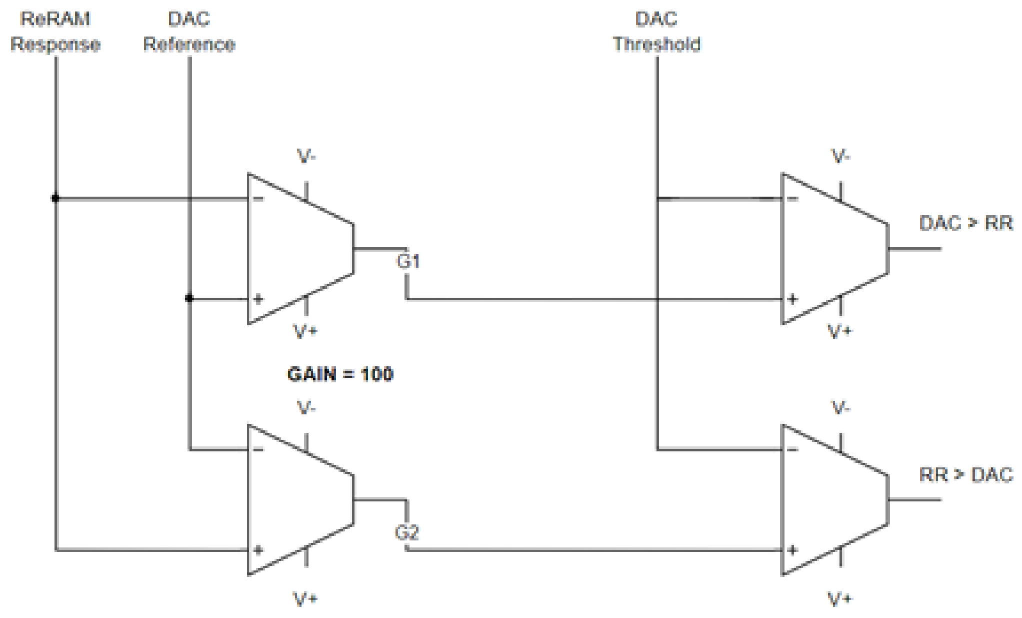


Figure 5.6: Schematic of Current Guessing Hardware

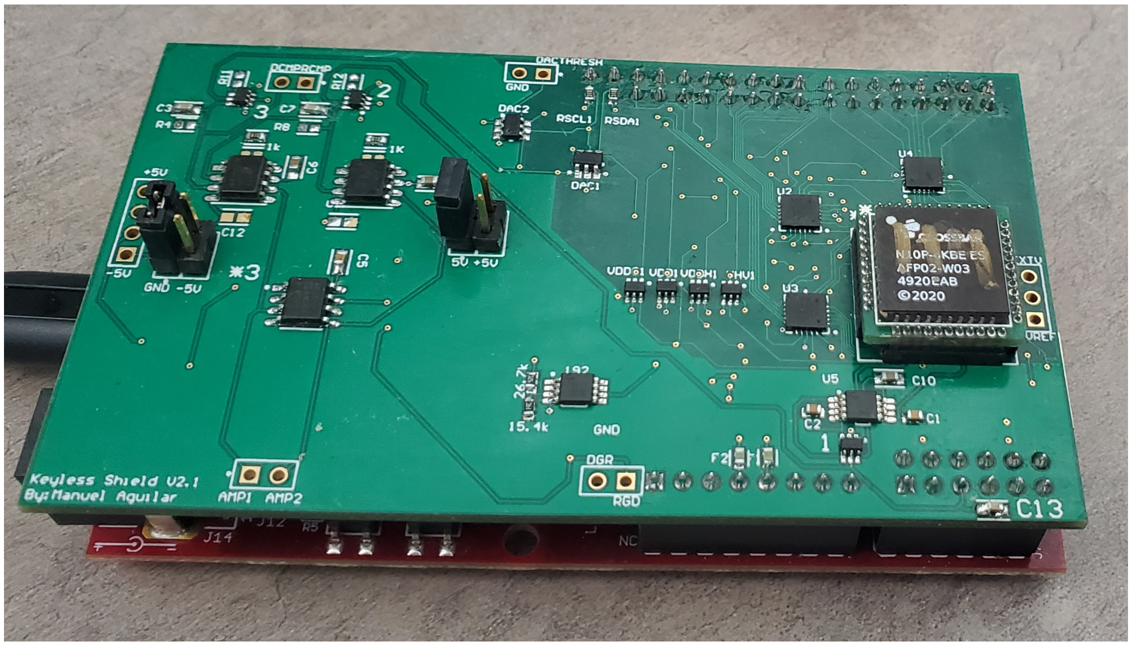


Figure 5.7: Client device (Chipkit) with custom PCB and IC components

must also take in the values output by the differential op-amps to compare the amplified and threshold signals. Using this configuration, the average time it took for encapsulation and extraction of 1024 characters was 1.5 seconds.

External hardware was implemented using a custom PCB with IC components on the client device. Su
A picture of the client device and custom PCB can be found in Fig. 5.7

5.5.3 Enrollment

Enrolling a PUF involves capturing and storing its unique response to a given challenge in a database. Specifically, in the case of a ReRAM device, its analog responses are queried and recorded a predetermined number of times, n. The means of these readings are stored as the response, while their standard deviations are recorded as the intra-cell variation, which serves as a reliability metric.

In the Analog Key Encapsulation protocol, we use 16 currents to encapsulate its message. These 16 currents are selected from 75 different currents, with the goal of minimizing response overlap and errors. However, since each ReRAM cell behaves independently from others, each device has its own set of currents optimized to minimize errors. While this adds complexity to the enrollment process, it also enhances security by making it more challenging for attackers to identify the currents used for each ReRAM device.

The enrollment process involves a short enrollment step for all available currents, with a read count of $n = 3$. Then, the currents with responses furthest away from each other are selected, and a more thorough enrollment is conducted with a read count of $n = 30$. Finally, the means and intra-cell variations of each cell are compiled to create a comprehensive database of responses for the PUF.

5.5.4 Current Guessing Methods

In §5.4, software reads were taken using a 12-bit ADC to extract or guess the current. However, in this research, we used a mix of hardware and software comparisons to extract the currents given the addresses and corresponding responses. There are two main current guessing methods: hardware current guessing and mixed current guessing.

5.5.4.1 Hardware Current Guessing Method:

The first technique in this study employs a dedicated hardware current guessing circuitry, as detailed in section 5.5.2. Specifically, this method employs a DAC to produce expected responses and utilizes a binary search algorithm to determine the correct current. If the current cannot be determined, the method will repeatedly attempt to extract it a fixed number of times (c). In this study, the parameter c was set to 30, as it was found to be the optimal number of attempts that minimized errors. If the current value remains undetermined after these 30 attempts, a random current between 0 and 16 is transmitted as a fallback. However, it's worth noting that using a random current guess in such situations can lead to an increased number of bit errors compared to simply choosing an adjacent current.

5.5.4.2 Hybrid Current Guessing Method:

The second approach proposed in this study addresses the aforementioned issue by integrating software reads into the current guessing protocol. Like the first technique, this method utilizes the hardware current guessing circuitry mentioned earlier. However, suppose the current remains undetermined after a fixed number of attempts (c). In that case, it will conduct a read of each current through an ADC and select the closest value to the expected result, ensuring a random current is never sent. It should be noted, however, that this method has some drawbacks. Specifically, conducting ADC reads is a time-intensive process, and it poses a security risk by providing attackers with an opportunity to enroll ReRAM devices.

5.5.5 Encapsulation and Extraction Protocols

The analog key encapsulation method aims to transform a symmetrical key or message into a cipher to enable secure transmission between two parties. To achieve this, we employ the analog responses obtained from the pre-formed ReRAM PUF to secure the key and convert it into a cipher.

5.5.5.1 Message Encapsulation

To begin the encapsulation process, we generate a random number (RN) and combine it with a password (PW). This combined value is then hashed to produce a Message Digest (MD). The MD is used to select addresses from a pool of 4096 addresses in the ReRAM PUF. In this protocol, we use every 2 bytes of the MD to choose a single address from the ReRAM PUF.

To encapsulate the message, we inject the selected current into the address obtained from the 2 bytes of the MD. The resulting voltage is then stored as the cipher. The cipher, along with the random number (RN), is sent to the client, enabling them to decrypt the message.

The term "flaky" refer to cells within the MD that exhibit the highest intra-cell variation based on a percentile parameter. In this research scenario, the "flaky" cells are identified as those with intra-cell variation values in the top 25%.

These "flaky" cells are located on the server side, and a mask is utilized to convey their locations to the client. The mask is represented as a binary stream, where the positions of the "flaky" cells are indicated by '1's. The mask is sent separately to the client.

5.5.5.2 Message Extraction

The handshake, cipher, and mask are transmitted to the party possessing the physical PUF device for decryption. During the decryption phase, the addresses are obtained by concatenating the random number (RN) and password (PW) and applying a hash function. Once we have the addresses and the mask, we employ the current guessing methods described in 5.5.4 to estimate the current closest to the cipher. This current value corresponds to our encoded message bits. The entire protocol has been summarized in Figure 5.8.

To further reduce the Bit-error rate (BER) rate in our protocol, we conducted tests using Gray codes. In Gray codes, we assign a unique set of 4-bit symbols to each 4-bit symbol in a way that ensures there is only a single-bit change between neighboring symbols, as shown in Table 5.3. This technique allows for only a single-bit error if our current guessing protocol selects a neighboring current. While this approach does not eliminate all errors, it significantly reduces the number of errors to a minimum that can be resolved using other

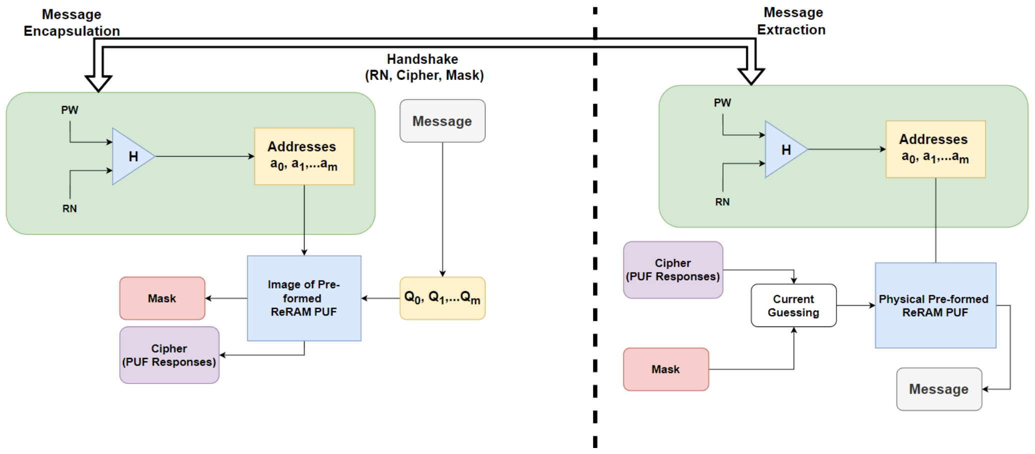


Figure 5.8: Analog Key Encapsulation Message Encapsulation and Message Extraction protocols

methods such Error-Correcting Code decoders or a search engine such Response Base Cryptography (RBC).

5.5.6 Testing and Results

To evaluate the analog key encapsulation protocol in this implementation, a comprehensive series of tests was conducted. These tests aimed to assess the BER of each current guessing method under various conditions and on 33 different ReRAM devices.

The first set of tests focused on measuring the BER of each current guessing method without implementing gray coding. Four hundred nine thousand six hundred bits (equivalent to 102,400 currents) were encapsulated and extracted during these tests.

Next, another set of tests was carried out to evaluate the BER of each current guessing method with the inclusion of gray coding. Similar to the previous tests, this set involved the encapsulation and extraction of a substantial number of bits (102,400 currents).

The final set of tests aimed to assess the BER of each current guessing method using a ReRAM device different from the one utilized for message encapsulation. In this test scenario, 102,400 currents were generated for each extraction.

The final set of tests is performed using five distinct ReRAM devices. For each ReRAM device, message extraction was attempted on messages encapsulated using the other 32 ReRAM devices, leading to 160 tests. This comprehensive testing approach provided valuable insights into the performance and accuracy of the current guessing methods using a different pre-formed ReRAM Device.

These comprehensive testing procedures enabled a thorough evaluation of the analog key encapsulation protocol, shedding light on the BERs and the impact of gray coding.

When employing the Hardware Current Guessing Method without Gray coding, the observed BER ranged from 0.186% to 1.948%, with an average of 0.692%. On the other hand, utilizing the Hybrid Current guessing system resulted in an BER ranging from 0.146% to 1.248%, with an average of 0.439%. Without error correction schemes, the average time required to decrypt a 1024-bit message was one second.

Upon introducing gray coding to the process, the BER for the Hardware Current Guessing Method varied between 0.166% to 1.765%, with an average of 0.636%. Similarly, the Hybrid Current Guessing Method exhibited an BER ranging from 0.144% to 0.986%, with an average of 0.380%. Overall, the BER primarily depended on the average intra-cell variation of the ReRAM device. Devices with a lower intra-cell variation average demonstrated lower BERs than those with a higher intra-cell variation average.

When testing the BER of the ReRAM devices using different enrollments, the ideal value is 50% as it does not give any information. The BER using the Hardware Current Guessing Method ranged from 37.996%-53.115% with an average BER of 44.011%. The BER for an incorrect ReRAM device when using the Hybrid Current Guessing Method ranged from 29.866%-50.129% with an average BER of 37.229%. A table of all testing results can be found in Table 5.2

BER Statistics / Message Extraction Method	Minimum	Mean	Maximum
Hardware Current Guessing Method without Gray Coding	0.186%	0.692%	1.948%
Hybrid Current Guessing Method without Gray Coding	0.146%	0.439%	1.248%
Hardware Current Guessing Method with Gray Coding	0.166%	0.636%	1.765%
Hybrid Current Guessing Method with Gray Coding	0.144%	0.380%	0.986%
Incorrect PUF Hardware Current Guessing Method with Gray Coding	37.996%	44.011%	53.115%
Incorrect PUF Hybrid Current Guessing Method with Gray Coding	29.866%	37.229%	50.129%

Table 5.2: BER results using different message extraction methods

Decimal	Standard Binary	Gray Code Binary
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Table 5.3: Gray coding mapping of a four-bit binary number

5.5.7 Improved Analog Key Encapsulation Conclusion

This research enhanced the efficiency and security of the analog key encapsulation protocol by utilizing specialized hardware and packaged pre-formed ReRAM devices on a low-power client device. We focused on implementing the protocol by incorporating hardware comparisons to estimate currents. The goal was to improve the efficiency and security aspects of the previous analog key encapsulation protocol.

When employing the analog key encapsulation protocol with gray coding, we achieved a significantly faster decryption rate of 1024 bits per second. Additionally, our findings revealed that employing a current guessing method solely based on hardware comparisons resulted in an BER of 0.636%. On the other hand, the hybrid method, which combined both hardware and software comparisons, yielded a lower BER of 0.380%.

Although the hybrid current guessing system exhibited a lower BER, both BERs can be corrected with a search engine for the original message or error-correcting codes. Additionally, it is worth noting that the hardware comparison current guessing system provides enhanced security, as it avoids reading analog responses into memory, making it less vulnerable to cloning attacks.

When attempting to extract the message using a different pre-formed ReRAM device from the one that initially encapsulated the message, the BERs were notably high, which prevented significant information leakage. Specifically, the Hardware Current Guessing Method exhibited an average BER of 44.011%, while the Hybrid Current Guessing Method had an average BER of 37.229%.

These findings indicate that even if an attacker possesses a similar device, the inherent cell-to-cell variations in ReRAM prevent a significant concern with False acceptance rate (FAR)s. In other words, the discrepancies between individual ReRAM cells make it challenging for attackers to access the encrypted information using similar devices.

This implementation successfully addressed the limitations of the previous approach, demonstrating its reliability as a method of encapsulating sensitive information. The achieved BERs, and latencies were within acceptable ranges, making analog key encapsulation a viable alternative for transmitting confidential data. Future research should explore potential modeling attacks to assess the protocol's resilience and strengthen its security further.

Chapter 6

Discussion & Conclusion

This dissertation aimed to bridge the gap between theoretical analysis and practical implementation of memory Physically Unclonable Function (PUF)s. The primary focus was on three types of memory technologies: Static random-access memory (SRAM), Magnetoresistive random-access memory (MRAM), and Resistive random-access memory (ReRAM). The research explored the feasibility and effectiveness of implementing PUFs using these memory technologies on low-power devices to simulate real-world applications better.

In the third chapter, we introduced a new SRAM-based PUF design that overcomes the limitations of previous SRAM designs while maintaining the simplicity and accessibility of SRAM technology. This design successfully generates cryptographic keys and truly random numbers without the need for server intervention. It utilizes two SRAM devices and an XOR gate. To evaluate its performance, we tested ten ISSI IS64WV6416BLL SRAM chips at different temperatures (0°C, 23°C, and 80°C). The evaluation focused on three metrics: Bit-error rate (BER), HD_{inter}, and entropy density. The results showed promising BER performance, with an average of 0.2% at the enrollment temperature, although the BER increased to around 10% at temperatures far from the enrollment temperature. Additionally, the PUF exhibited a great HD_{inter} value of 0.4966 and a substantial entropy density of 0.997.

Furthermore, using the same design, we implemented various True Random Number Generator (TRNG) protocols and demonstrated their effectiveness by passing the National Institute of Technology (NIST) Test Suite. We applied a lightweight Pseudo-Random Number Generator (PRNG) XOR compiler to enhance the randomness of the generated sequences. The implemented TRNG protocols exhibited non-deterministic behavior, generating different random sequences with a 50% difference when using the same challenges. Unlike previous designs that required overhead to select specific cells, our protocols utilized a simple PRNG and unstable cells as a reference point without any additional overhead.

In the fourth chapter, we explore MRAM technology and introduce a new design for a Ternary Addressable Physically Unclonable Function (TAPUF) using two MRAM devices. This design uses the resistance variation in MRAM cells to generate binary responses through a differential comparison. We conducted experiments with 30 MRAM devices at different temperatures and analyzed their analog characteristics. Real-time responses were used to calculate metrics like HD_{inter}, BER, and entropy density. The MRAM-based TAPUF design demonstrated impressive reliability across a wide temperature range and outperformed previous MRAM-based PUF designs, all while being implemented on a low-power client device.

Additionally, we found that the same MRAM-based TAPUF could be utilized as a TRNG by focusing on unstable Challenge-response pair (CRP)s. These random sequences produced by the TRNG were non-deterministic and passed the NIST Testing Suite, indicating their high quality. However, generating non-deterministic bits required instructions from the server, which added some overhead to the process.

In the fifth chapter, the Analog Key Encapsulation protocol introduced in [100] was implemented using physical ReRAM devices on a low-power client. This implementation differed from the previous work discussed in chapters three and four, as its primary goal was not to generate cryptographic keys or true random sequences but to encapsulate various types of data, including messages, numbers, and cryptographic keys.

The first implementation involved using software reads on unoptimized hardware to compare the analog responses of pre-formed ReRAM cells. Although this approach showed promising results, with accurate comparisons, it suffered from slow message extraction latency, taking 3 seconds per character, which is 384 seconds per 1024 bits. Additionally, using software reads posed a security vulnerability that needed to be addressed.

The second implementation improved upon the first by utilizing specialized hardware capable of comparing analog responses using hardware mechanisms instead of software. This enhancement significantly increased the message extraction speed to 1024 bits per second. Moreover, the second implementation achieved a low BER of less than 1% using gray coding.

6.1 Memory Technology Comparisons

In this dissertation, we explored three different memory technologies, namely SRAM, MRAM, and ReRAM, for their potential use as PUFs. Each technology exhibited unique characteristics that made them suitable for PUF applications. However, certain designs demonstrated superior PUF metrics, making them more desirable for cryptographic purposes.

Among the SRAM and MRAM PUF designs, both had an equal number of CRPs. However, the SRAM design held an advantage in terms of simplicity and widespread availability. SRAM is already integrated into many electronic devices and requires minimal additional hardware. The only requirement for the SRAM PUF design is the ability to power it on and off independently, separate from the rest of the system. The SRAM PUF also showcased lower latency with a throughput of 1.6 MHz, compared to the MRAM TAPUF's

PUF Metrics \ **PUF**	**SRAM XOR PUF**	**MRAM TAPUF**
Number of CRPs	n^2	n^2
BER	0.0020-0.0037	$< 10^{-6}$
HD_{inter}	0.4980	0.4999
Entropy Density H(X)	0.9984	0.99999
Throughput	1.6 Mb/s	33.33 kb/s

Table 6.1: PUF Metrics of the SRAM XOR PUF introduced in Chapter Three and the MRAM TAPUF introduced in Chapter Four. These PUF metrics are for responses filtered for 'X' CRPs. $< 10^{-6}$ is used if no error was found in 1 million responses.

TRNG Metrics \ **TRNG**	**SRAM TRNG**	**MRAM TRNG**
Pass NIST Test	Yes	Yes
Server Required	No	Yes
Post-Processing Required	No	No
HD_{intra} **before post-processing**	0.4743	0.1923
Throughput	1.6 Mb/s	33.33 kb/s

Table 6.2: TRNG Metrics of the SRAM XOR PUF, introduced in Chapter Three, and the MRAM TAPUF, introduced in Chapter Four.

throughput of 33.33 kHz. However, when considering other PUF metrics such as BER, HD_{inter}, and entropy density (refer to Table 6.1), the MRAM TAPUF outperformed the SRAM design. Additionally, the MRAM TAPUF exhibited greater robustness to temperature variations. It could also be utilized as regular memory when not functioning as a PUF, thanks to its unlimited write endurance.

When comparing the designs as TRNGs, the SRAM XOR PUF design outperforms the MRAM TAPUF design as shown in Table 6.2. The SRAM design has a higher throughput of 1.6 MHz and does not rely on server assistance to generate non-deterministic random bits. Its non-deterministic properties are significantly high, with sequences generated exhibiting a HD_{intra} value of approximately 0.47. Furthermore, the SRAM design does not require post-processing to ensure security or pass the NIST Testing Suite.

On the contrary, the MRAM TAPUF design necessitates server intervention to select the most 'X' states and demonstrates lower non-deterministic properties. Random sequences generated exhibit a HD_{intra} value of around 0.19. Although it does not require post-processing to pass the NIST Testing Suite, it employs internal XOR Compiler post-processing to achieve an improved HD_{inter} value of 0.50 when comparing random sequences generated using the same challenge.

Pre-formed ReRAM exhibits unique properties that make it suitable for various cryptographic protocols, including Analog Key Encapsulation and key generation. The Analog Key Encapsulation protocol has demonstrated effectiveness on low-power devices and can be combined with other cryptographic techniques to enhance security. Additionally, there have been theoretical proposals suggesting that pre-formed ReRAM can be utilized for cryptographic key generation and true random number generation, similar to the methods employed for SRAM and MRAM. However, to comprehensively compare pre-formed ReRAM with SRAM and MRAM, further research is necessary to explore its capabilities and performance, specifically using low-power devices.

6.2 Future Work

The work done in this dissertation focused on three different technologies: SRAM, MRAM, and ReRAM, and investigated their characteristics when implemented in low-power hardware. The research successfully connected the work conducted at the wafer level with the physical implementation of these technologies. However, further research is necessary before this technology can be widely adopted for cybersecurity applications.

6.2.0.1 Pre-formed ReRAM Analog Key Encapsulation Temperature Testing and Improvements

In chapter five, we covered the topic of Analog Key Encapsulation, which involves using pre-formed ReRAM implementation with physical pre-formed ReRAM devices. Although this approach demonstrated promising outcomes, including low error rates and acceptable latency, it is essential to conduct additional research on the impact of temperature on the protocol. Previous studies have indicated that the responses of pre-formed ReRAM devices can vary with temperature [98], making temperature a crucial factor in the protocol that requires further investigation.

Additionally, the future plan for using Analog Key Encapsulation is to design and implement hardware in a way that avoids directly sending or replicating the responses of the pre-formed ReRAM cells with a Digital-to-Analog Converter (DAC). This approach involves utilizing two separate ReRAM devices. One of these devices will be responsible for encoding the device using currents, while the responses from the second device will be chosen as the matching responses.

In the current implementation using a single device, the response was sent as an analog value, which a DAC then replicated. The specialized current guessing circuitry was then used to compare this response. However, in a future implementation, the response will be conveyed as the location of a ReRAM cell that

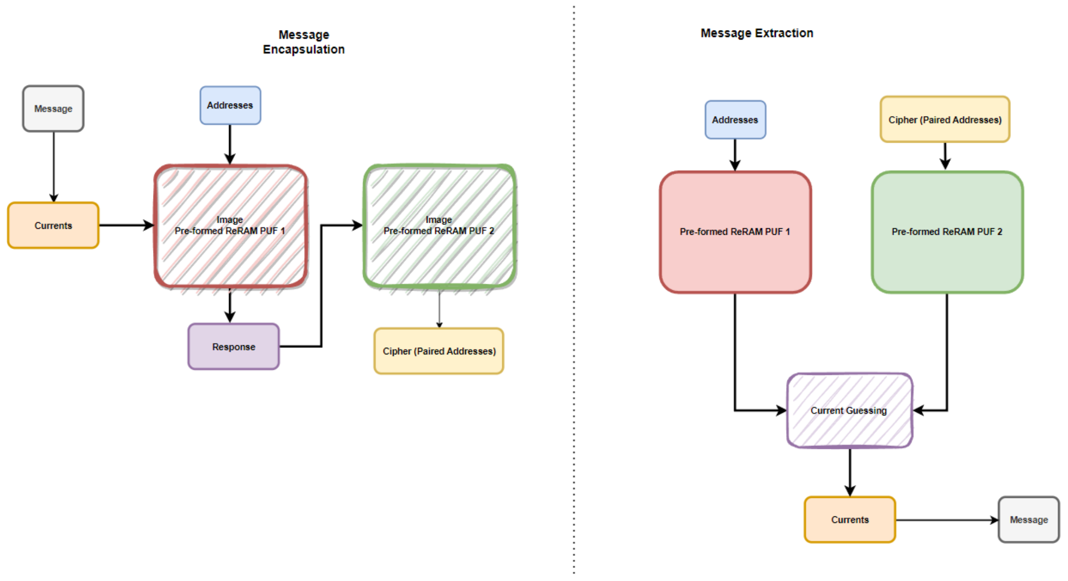


Figure 6.1: Analog Key Encapsulation Message Encapsulation using two pre-formed ReRAM devices

exhibits a very close response when subjected to the same current. The same current guessing hardware will be employed. As a result, the response data is never sent through unsecured channels, significantly increasing the difficulty for potential attackers attempting to break the encapsulation method. A visualization of this new implementation can be found in Figure 6.1.

Moreover, the proposed approach mitigates the risk of attackers employing machine learning techniques to predict the rest of the currents based on a few known CRPs. By utilizing separate ReRAM devices and only conveying cell locations, the system provides an additional layer of security against such attempts.

6.2.0.2 Pre-formed ReRAM TAPUF and TRNG

As mentioned in the preceding section, the subsequent phase for pre-formed ReRAM involves implementing a PUFs design on a low-power device capable of generating dependable cryptographic bits and non-deterministic random bits. While this design has been explored in studies such as [98, 104], its actual implementation on low-power hardware is yet to be realized. It is imperative to conduct further research akin to the studies discussed in Chapter Three to ensure a comprehensive comparison of the technologies involved. Such research will facilitate a robust evaluation of the implemented PUF design, allowing for informed decisions regarding its practicality and effectiveness in generating secure cryptographic and random bits on low-power hardware devices.

6.2.0.3 SRAM Characterization

Regarding SRAM, further research is needed to examine the characteristics of individual SRAM devices thoroughly. The sample size used for SRAM characterization in this study is deemed inadequate, as the inclusion of only ten devices does not sufficiently represent the SRAM device population. To ensure a more accurate representation, a sample size of at least 30 devices is recommended based on the assumption of normality. Additionally, it would be beneficial to implement this design with other cryptographic protocols to compare its performance against the MRAM design.

6.2.0.4 Modeling Attacks

For each of the proposed technologies, an important aspect that needs to be considered is modeling attacks. Modeling attacks involve creating a simulated model of the PUFs using mathematical techniques, often employing machine learning algorithms with known CRPs. To assess each design's resilience against modeling attacks, it is necessary to investigate the effectiveness of machine learning modeling attacks using a diverse set of known CRPs. This analysis will provide insights into the vulnerability of the theses PUFs to such attacks and help identify potential countermeasures.

6.2.0.5 Aging Effects

Additionally, the aging effect on the analog characteristics of MRAM cells in this specific design remains unknown. It is important to research to understand how the electrical resistance of MRAM cells changes over time and under different operating conditions. Examining the aging effect will contribute to a comprehensive understanding of the MRAM design's long-term reliability and assist in determining any necessary mitigation strategies or design optimizations to address potential performance degradation over time.

6.2.0.6 Tokenization

One of the final considerations before widespread adoption is the design's usability. It is essential to address the fact that most individuals do not want to carry bulky or cumbersome objects for authentication purposes in their daily lives. Therefore, a key aspect is to miniaturize this technology into a form factor that consumers find convenient and are willing to carry around without inconvenience. The ability to shrink the technology while maintaining its functionality and effectiveness is paramount to ensure user acceptance and seamless integration into everyday life.

A “token” refers to a compact hardware device used for multi-factor authentication. These tokens provide an additional layer of security alongside traditional username and password authentication methods. They are typically small, resembling a USB flash drive or credit card, making them convenient for everyday users to carry. Extensive studies have been conducted to analyze the widespread adoption of such devices [125].

While “tokens” can bolster security, they are still susceptible to side-channel attacks, where the complete cryptographic key can be extracted without leaving any detectable traces [126]. To address these vulnerabilities, the concept of PUFs has been proposed for implementation on token devices. PUFs generate unique, one-time keys that are not mathematically related to any specific algorithm. This approach adds an additional layer of security by leveraging the inherent physical properties of the token device itself.

Researching the effects of a smaller form factor on these device implementations can be the final step before mass adoption. PUFs that extract the analog characteristics of memory technology are more prone to the effects of a smaller form factor and should be further examined. Overall, the tokenization of PUFs can close the gap between research and consumers.

Investigating the impact of a smaller form factor on device implementations can serve as the last crucial phase before widespread adoption. This is especially important for PUFs that extract analog characteristics from memory technology as they are more likely to be affected due to EMI and cross-talk. The effects of a smaller form factor on these PUF implementations should be thoroughly examined to ensure their reliability and performance.

By tokenizing PUFs and integrating them into compact devices, we can effectively bridge the gap between research advancements and consumer adoption. This approach holds great potential for enhancing security and facilitating the widespread use of PUF technology in various applications.

Acronyms

PUF	Physically Unclonable Function
RNG	Random Number Generator
TRNG	True Random Number Generator
PRNG	Pseudo-Random Number Generator
CSPRNG	Cryptographically-Secure Pseudo-Random Number Generator
TRN	True Random Number
PRN	Pseudo-Random Number
TAPUF	Ternary Addressable Physically Unclonable Function
SRAM	Static random-access memory
MRAM	Magnetoresistive random-access memory
ReRAM	Resistive random-access memory
NIST	National Institute of Technology
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
ECC	Error Correcting Codes
PQC	Post-quantum cryptography
CRP	Challenge-response pair

BER Bit-error rate

FAR False acceptance rate

FRR False rejection rate

RBC Response Base Cryptography

PQC Post-Quantum Cryptography

TAPKI Ternary Addressable Public Key Infrastructure

AKE Analog Key Encapsulation

KEM Key Encapsulation Mechanism

DSA Digital Signature Authentication

PKA Public Key Exchange which is Addressable

HRS High Resistive State

LRS Low Resistive State

PKI Public Key Infrastructure

CA Certificate Authority

RA Registration Authority

LWE Learning With Error

IoT Internet of Things

ADC Analog-to-Digital Converters

DAC Digital-to-Analog Converter

CMOS Complementary Metal-Oxide Semiconductor

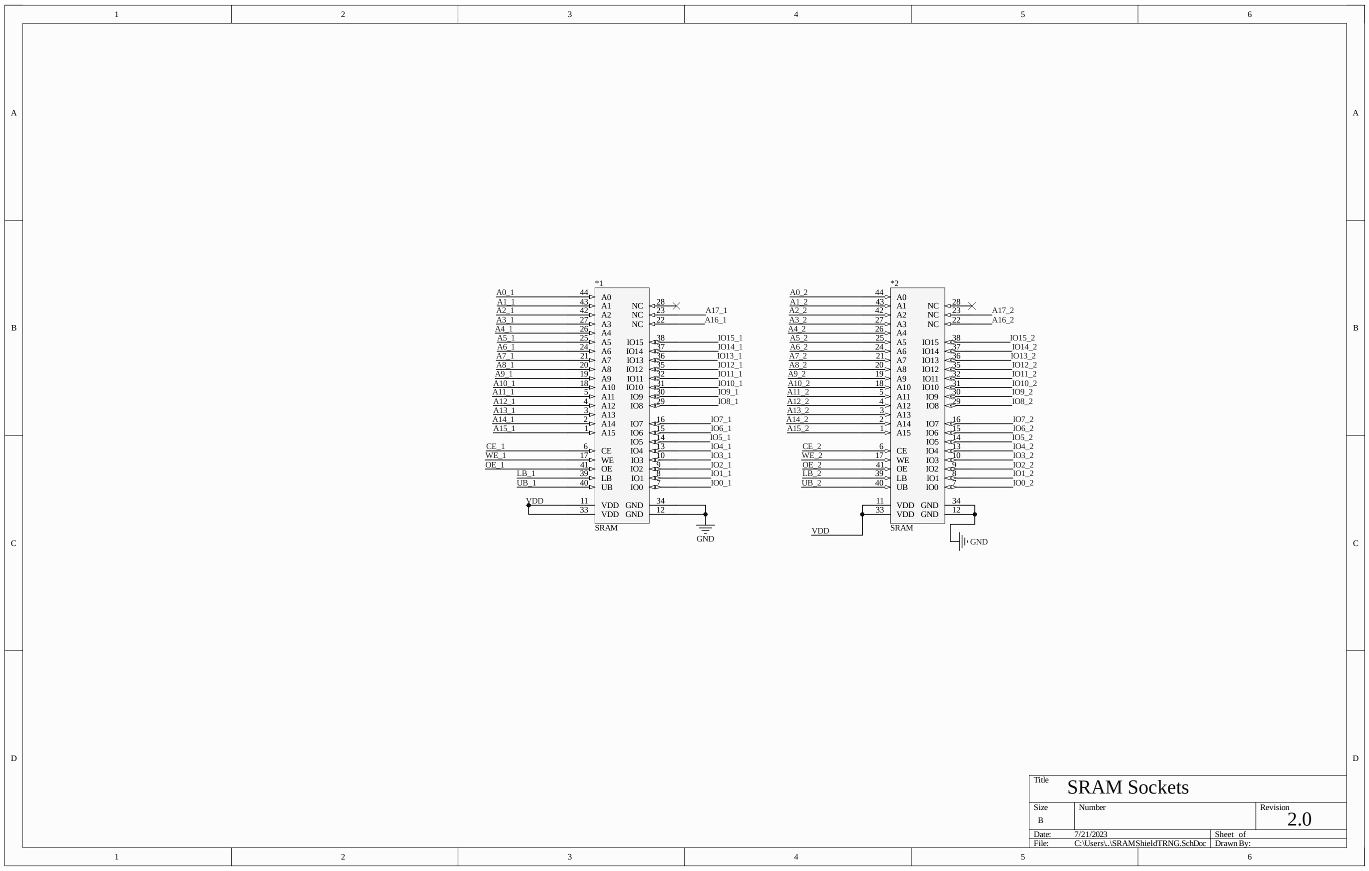
PCB Printed Circuit Board

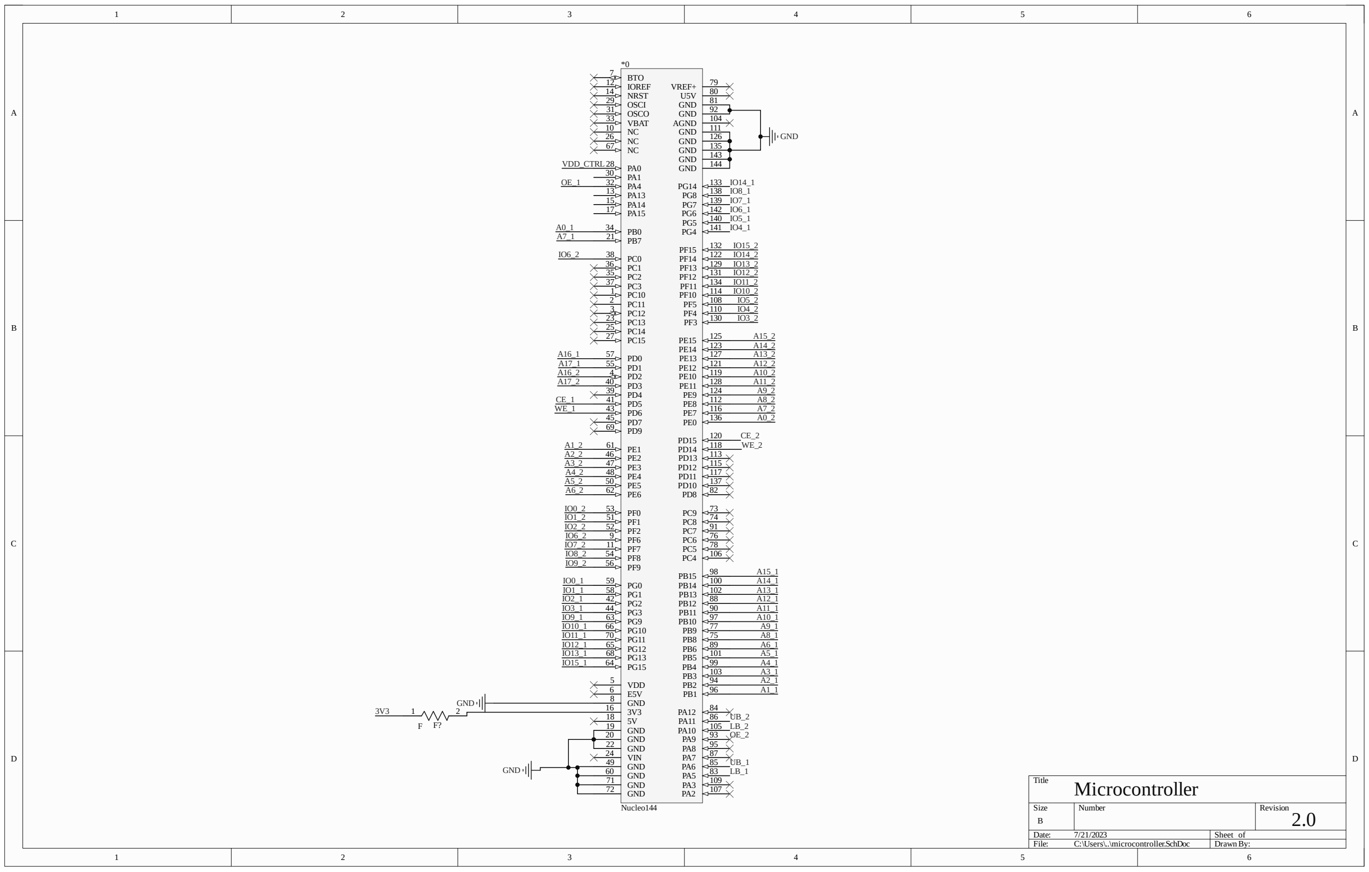
IC Integrated Circuit

Appendix A

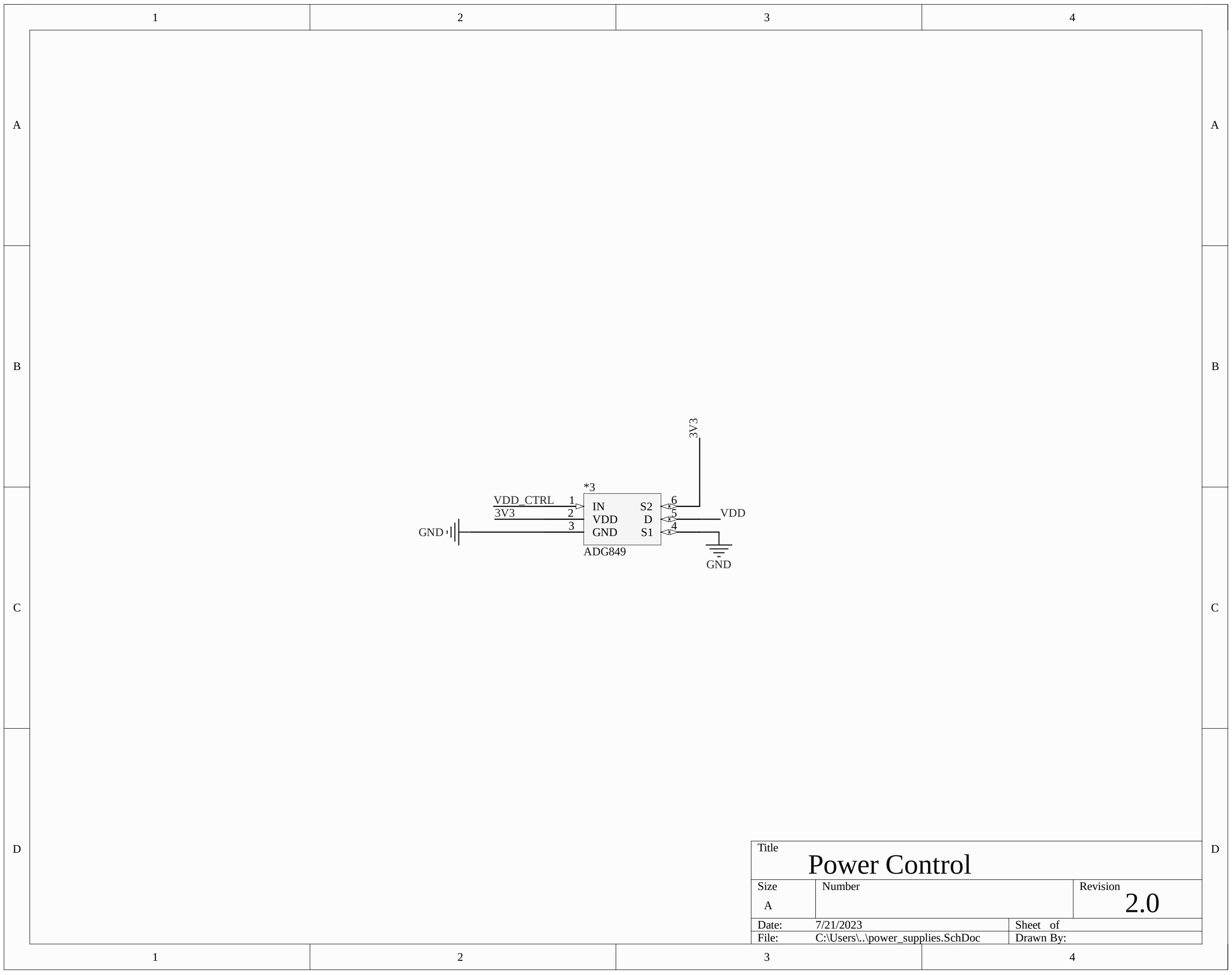
Supplementary Information for Chapter 3

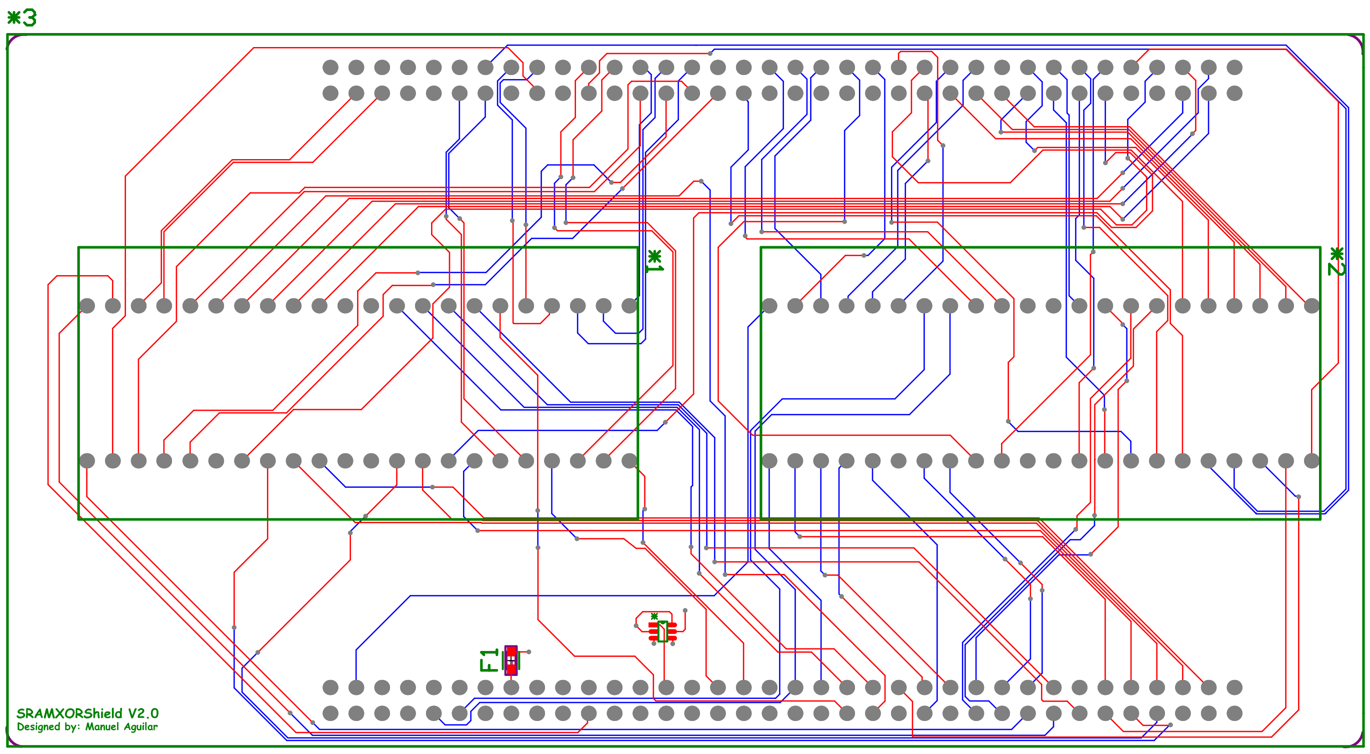
The supplementary information for this Chapter is the schematics and Printed Circuit Board (PCB) layout of the SRAM XOR PUF PCB design discussed in §3.3, used to generate cryptographic keys and "truly" random bits.


*1
*2
SRAM
SRAM
A0_1 … A15_1
A0_2 … A15_2
A17_1
A16_1
A17_2
A16_2
IO15_1 … IO0_1
IO15_2 … IO0_2
CE_1
WE_1
OE_1
LB_1
UB_1
CE_2
WE_2
OE_2
LB_2
UB_2
VDD
GND
Title
SRAM Sockets
Size
B
Number
Revision
2.0
Date:
7/21/2023
Sheet of
File:
C:\Users\..\SRAMShieldTRNG.SchDoc
Drawn By:



*0
Nucleo144
GND
3V3
F
F?
VDD_CTRL
OE_1
A0_1
A7_1
IO6_2
A16_1
A17_1
A16_2
A17_2
CE_1
WE_1
CE_2
WE_2
UB_2
LB_2
OE_2
UB_1
LB_1
Title
Microcontroller
Size
B
Number
Revision
2.0
Date: 7/21/2023
File: C:\Users\..\microcontroller.SchDoc
Sheet of
Drawn By:


1
2
3
4
A
B
C
D
3V3
*3
VDD_CTRL
3V3
GND
1
2
3
IN
VDD
GND
S2
D
S1
6
5
4
VDD
ADG849
GND
Title
Power Control
Size
A
Number
Revision
2.0
Date: 7/21/2023
File: C:\Users\..\power_supplies.SchDoc
Sheet of
Drawn By:

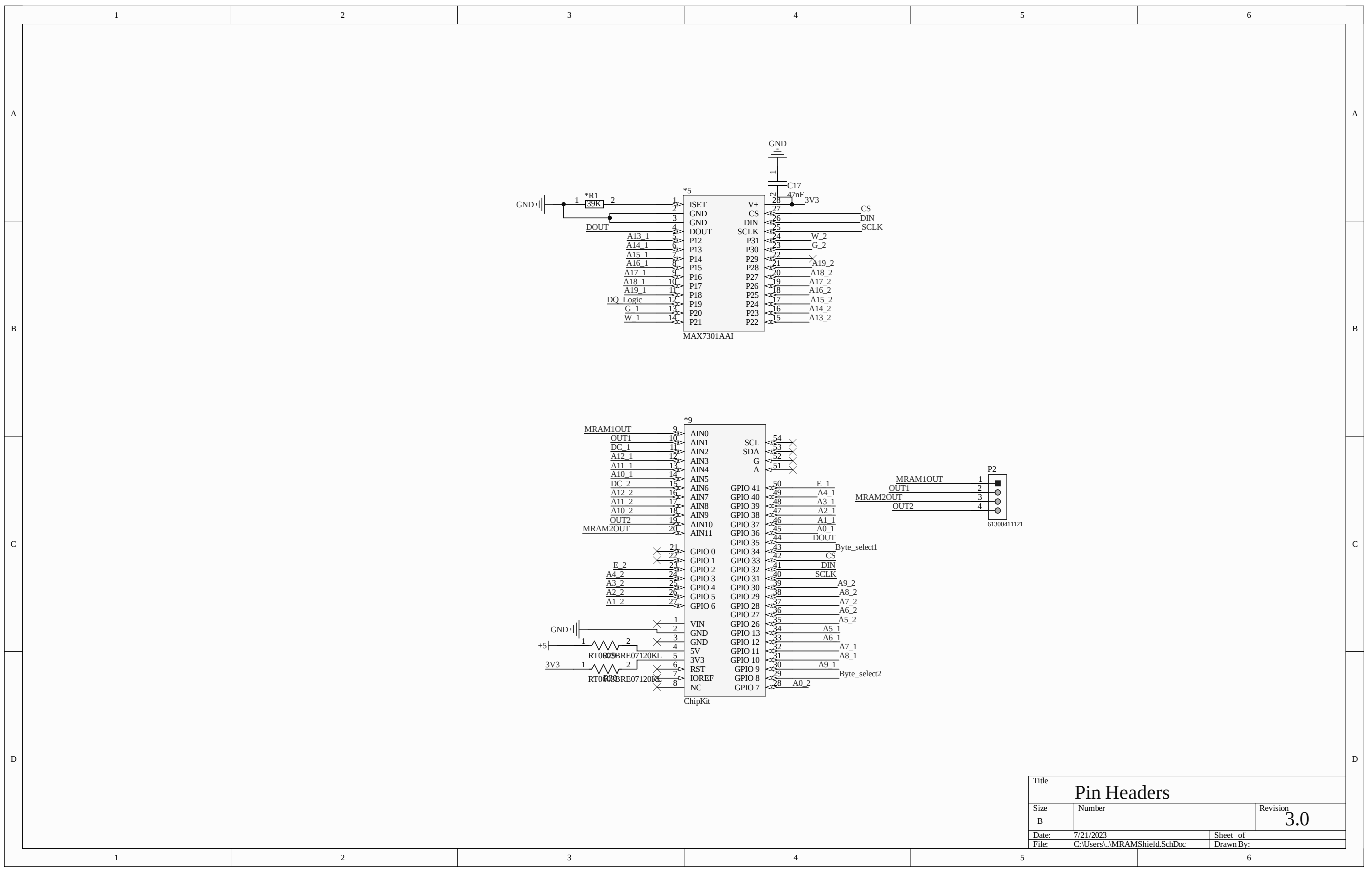

*3
*1
*2
F1
SRAMXORShield V2.0
Designed by: Manuel Aguilar

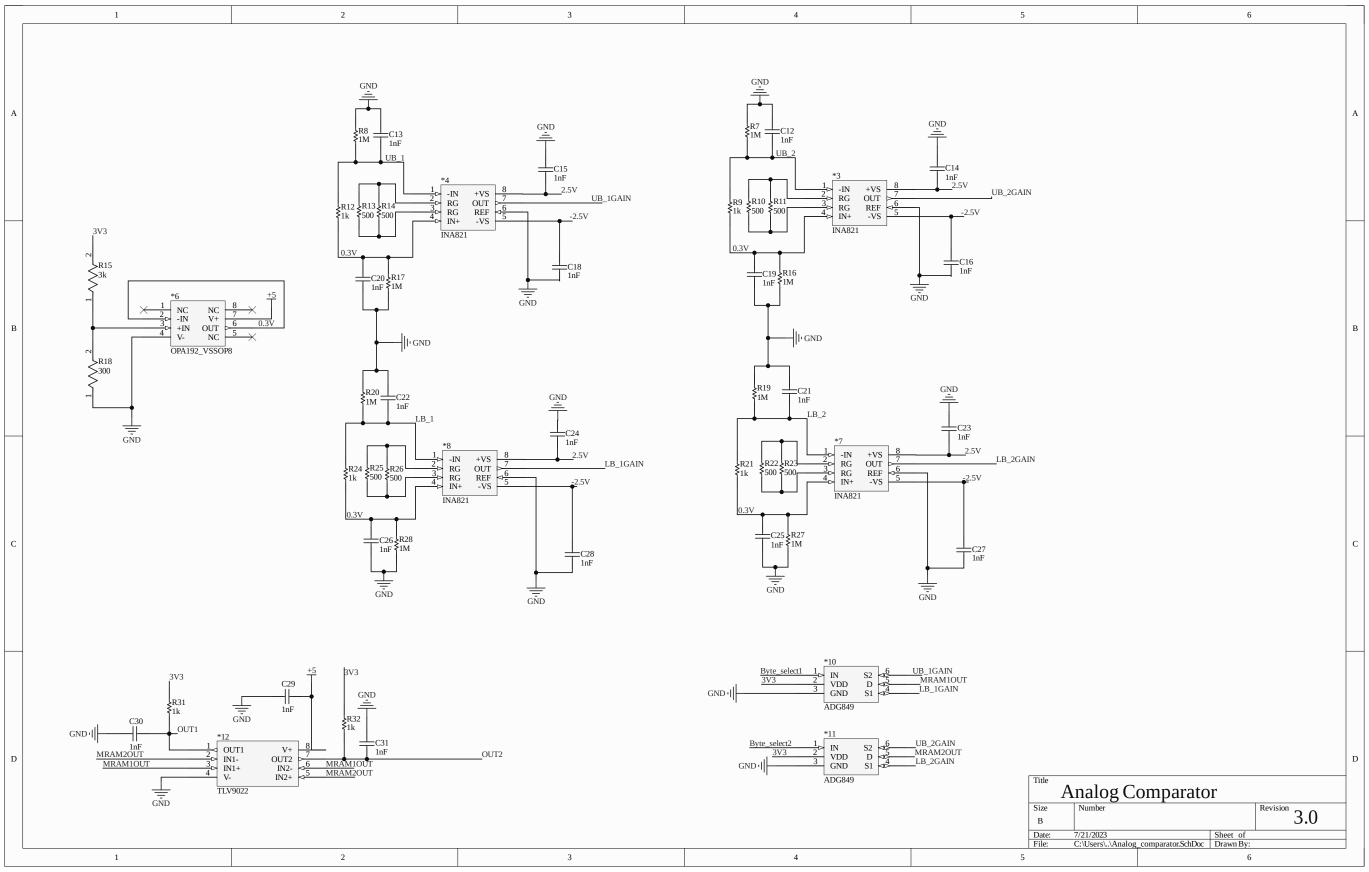
Appendix B

Supplementary Information for Chapter 4

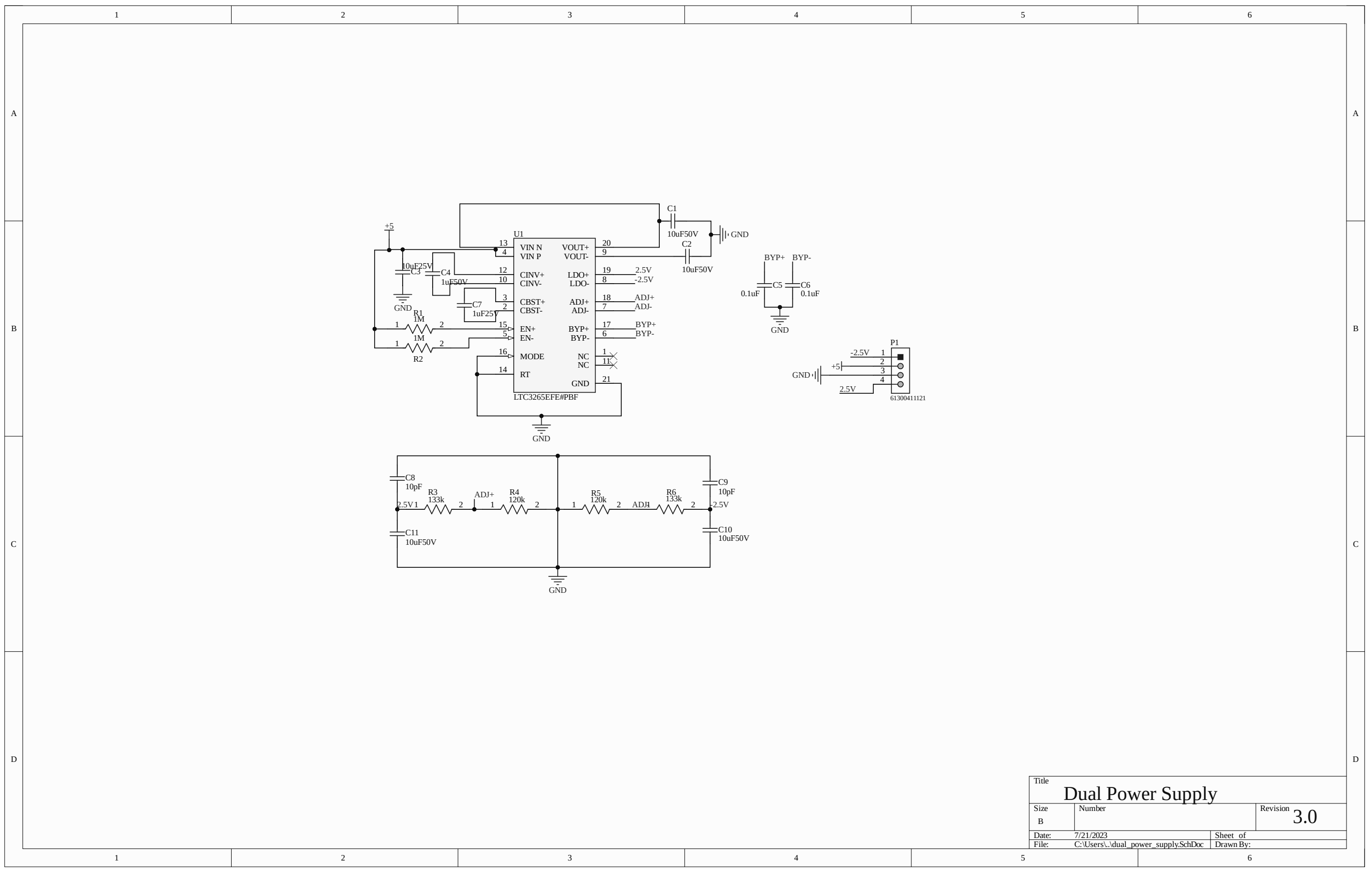
The additional material provided for this chapter includes schematics and Printed Circuit Board (PCB) layouts of the MRAM circuitry employed. The first set of attachments consists of the schematics and PCB layout for the MRAM True Random Number Generator (TRNG) used in the initial implementation of the MRAM-based TRNG, as discussed in §4.7.6.

The second set of attachments comprises schematics and PCB layout related to the MRAM Ternary Addressable Physically Unclonable Function (TAPUF) utilized for cryptographic bit generation and to enhance the generation of genuinely random bits discussed in §4.3.

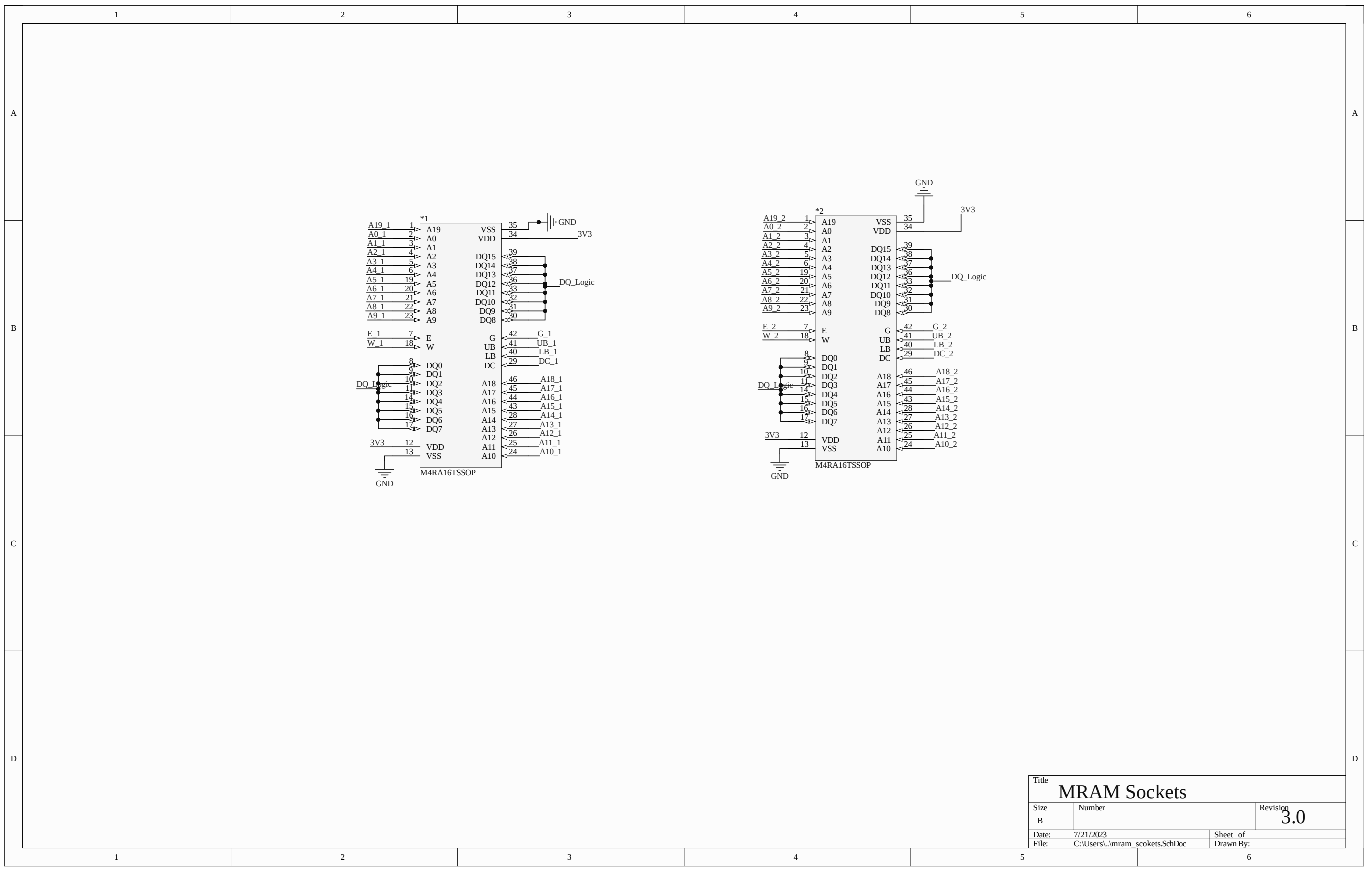

MAX7301AAI
ChipKit
R1 39K
C17 47nF
GND
3V3
+5
P2
61300411121
MRAM1OUT
OUT1
MRAM2OUT
OUT2
CS
DIN
SCLK
DOUT
Byte_select1
Byte_select2
Title
Pin Headers
Size
B
Number
Revision
3.0
Date: 7/21/2023
File: C:\Users\..\MRAMShield.SchDoc
Sheet of
Drawn By:

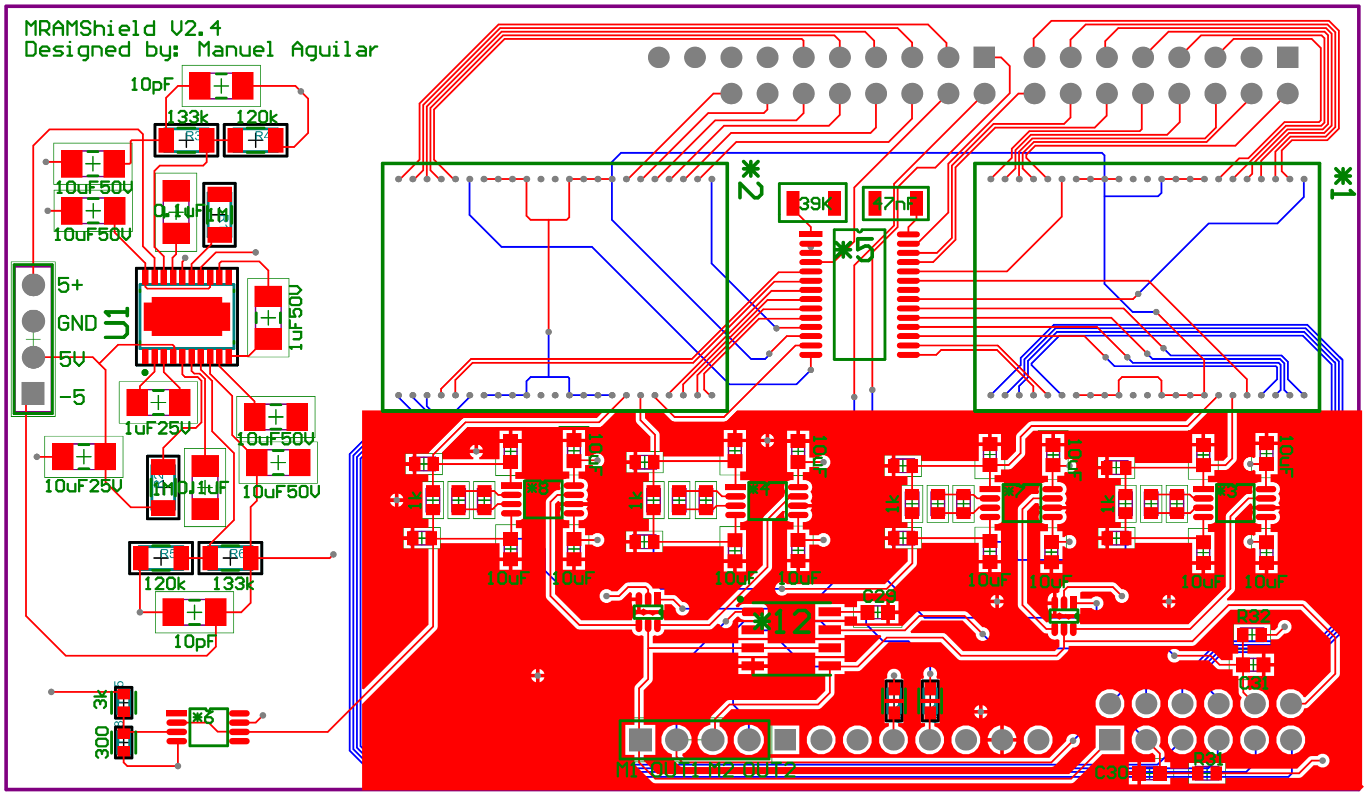


Analog Comparator
Size B
Revision 3.0
Date: 7/21/2023
File: C:\Users\..\Analog_comparator.SchDoc
INA821
OPA192_VSSOP8
TLV9022
ADG849
UB_1GAIN
UB_2GAIN
LB_1GAIN
LB_2GAIN
MRAM1OUT
MRAM2OUT
OUT2
Byte_select1
Byte_select2

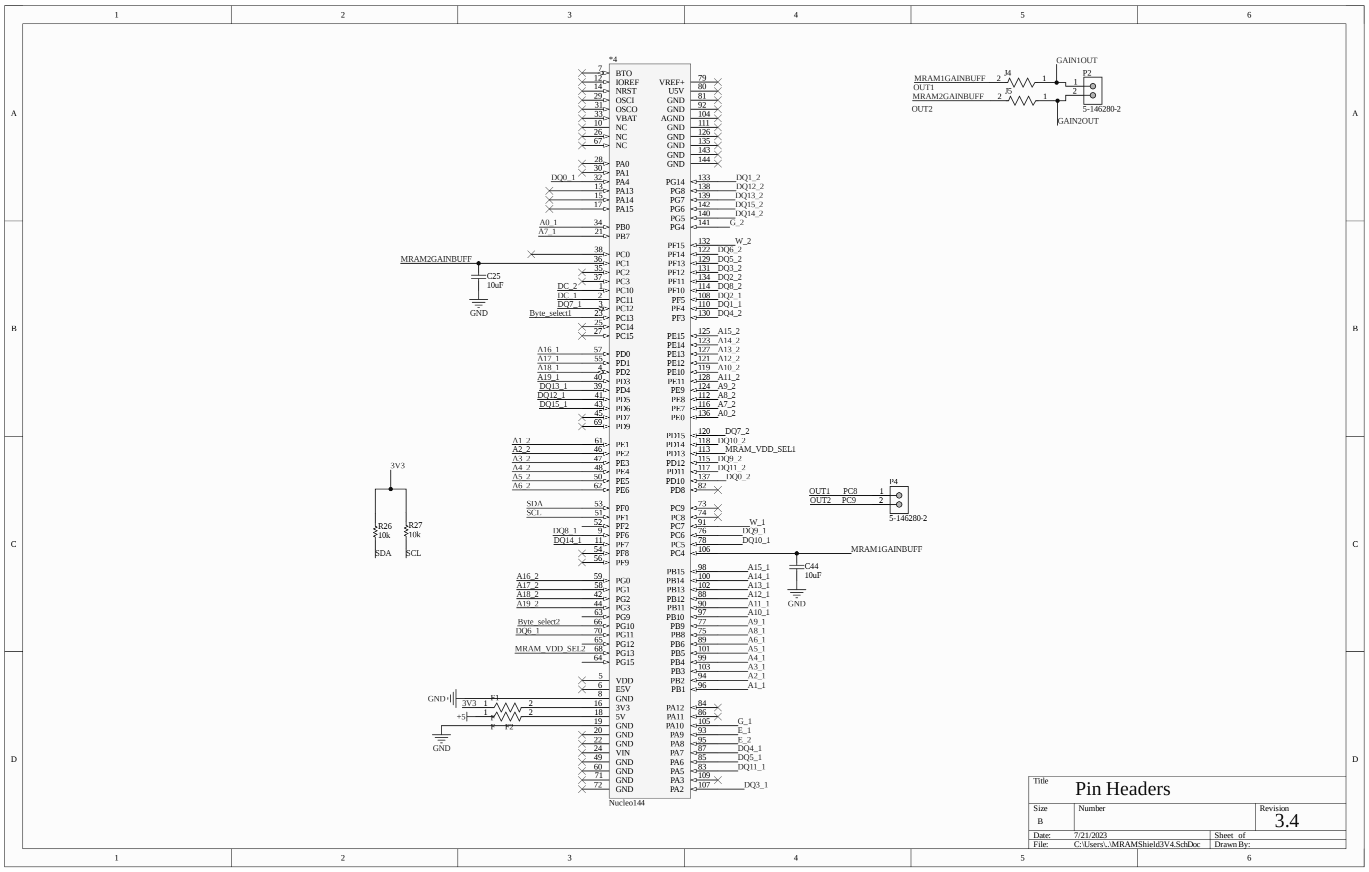


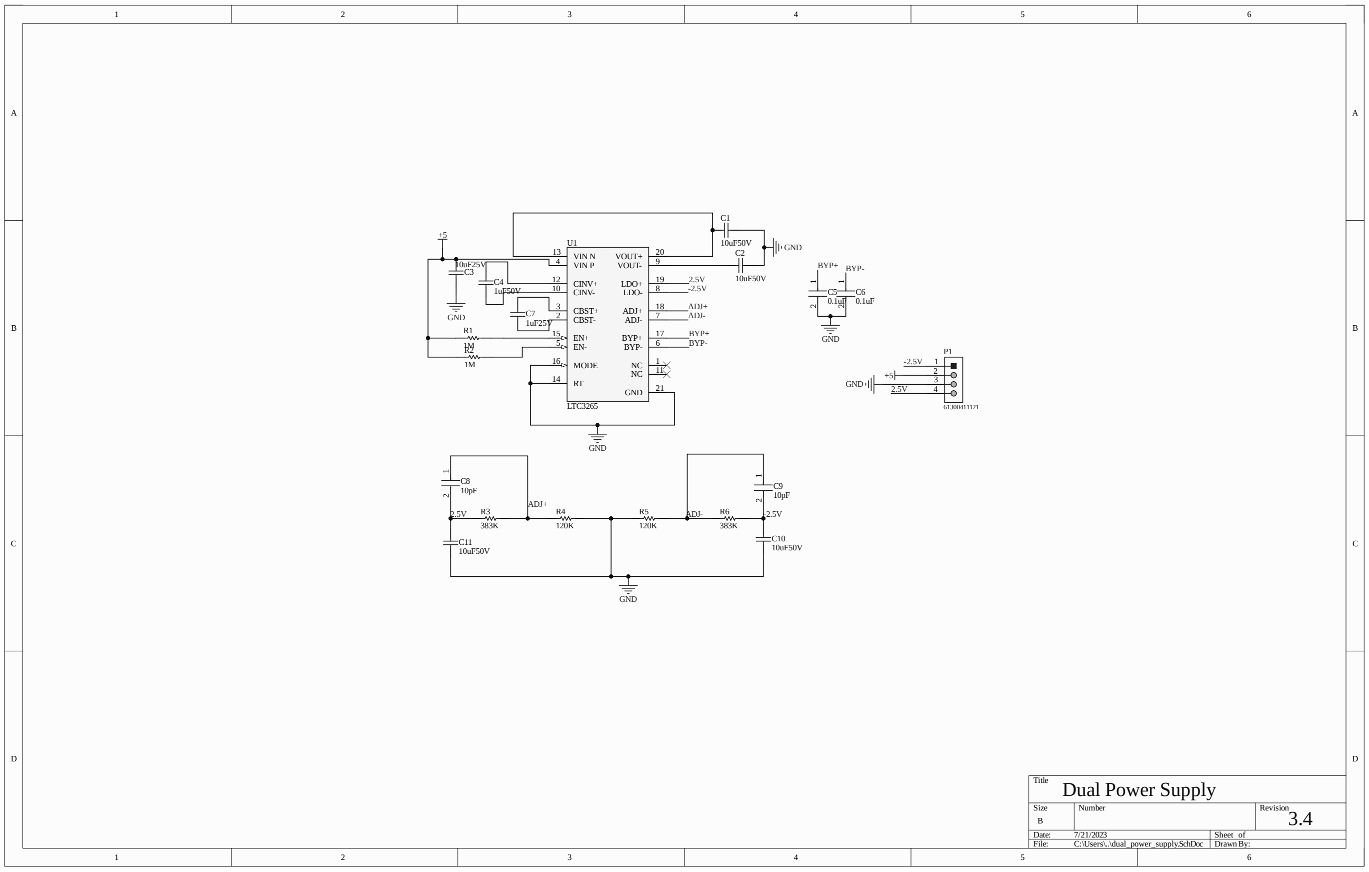
U1
LTC3265EFE#PBF
VIN N
VIN P
CINV+
CINV-
CBST+
CBST-
EN+
EN-
MODE
RT
VOUT+
VOUT-
LDO+
LDO-
ADJ+
ADJ-
BYP+
BYP-
NC
NC
GND
+5
GND
C1 10uF50V
C2 10uF50V
C3 10uF25V
C4 1uF50V
C7 1uF25V
R1 1M
R2 1M
2.5V
-2.5V
ADJ+
ADJ-
BYP+
BYP-
C5 0.1uF
C6 0.1uF
P1
61300411121
C8 10pF
C9 10pF
C10 10uF50V
C11 10uF50V
R3 133k
R4 120k
R5 120k
R6 133k
Title
Dual Power Supply
Size B
Number
Revision 3.0
Date: 7/21/2023
Sheet of
File: C:\Users\..\dual_power_supply.SchDoc
Drawn By:


MRAM Sockets
M4RA16TSSOP
*1
*2
DQ_Logic
GND
3V3
Size B
Revision 3.0
Date: 7/21/2023
File: C:\Users\..\mram_scokets.SchDoc
Sheet of
Drawn By:

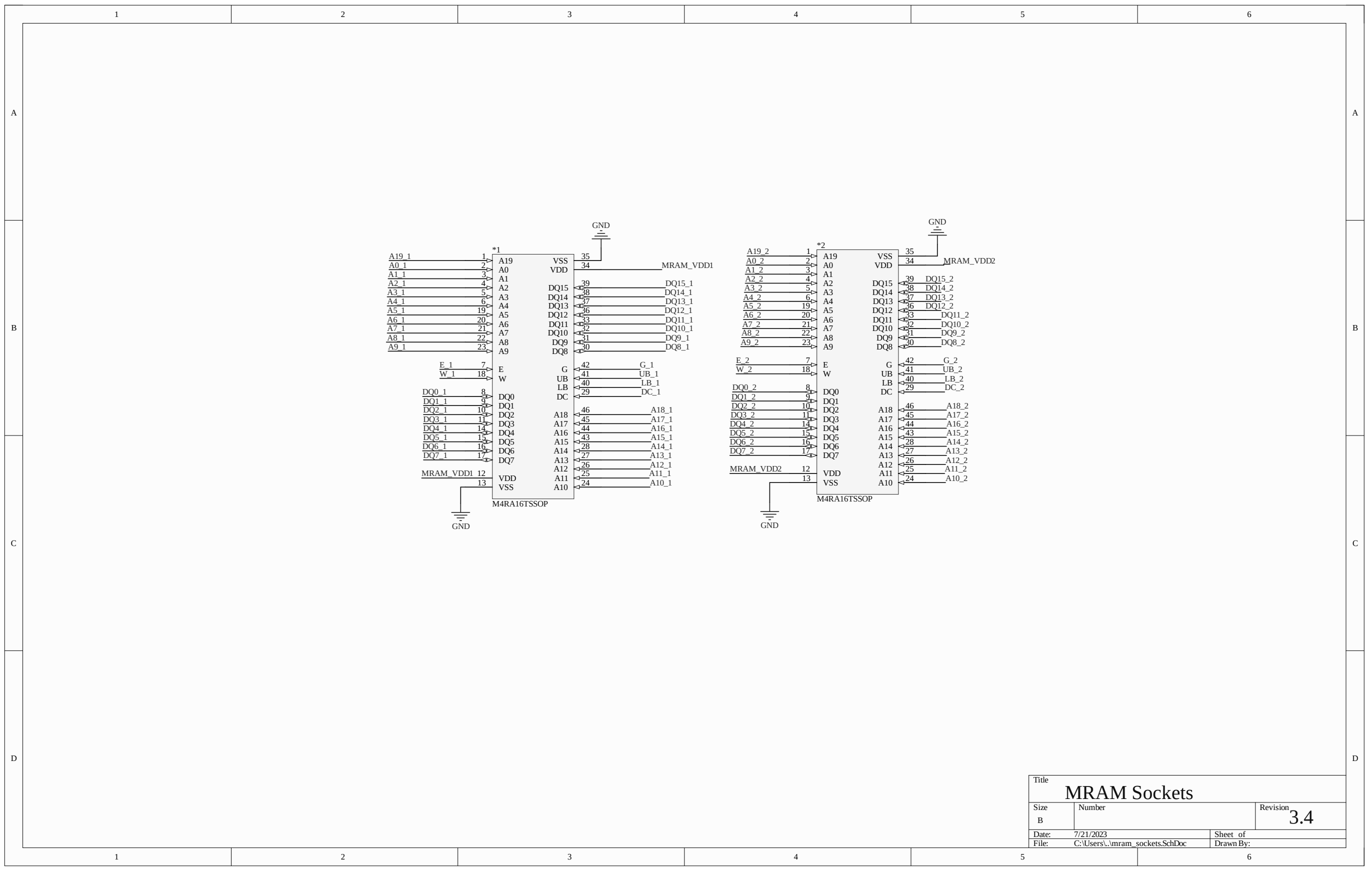


MRAMShield V2.4
Designed by: Manuel Aguilar
10pF
133k
120k
10uF50V
10uF50V
0.1uF
5+
GND
5V
-5
U1
1uF50V
1uF25V
10uF50V
10uF25V
0.1uF
10uF50V
120k
133k
10pF
3k
300
*6
*2
39k
47nF
*5
*1
*12
10uF
1k
C29
C30
R31
R32
C31
M1
OUT1
M2
OUT2

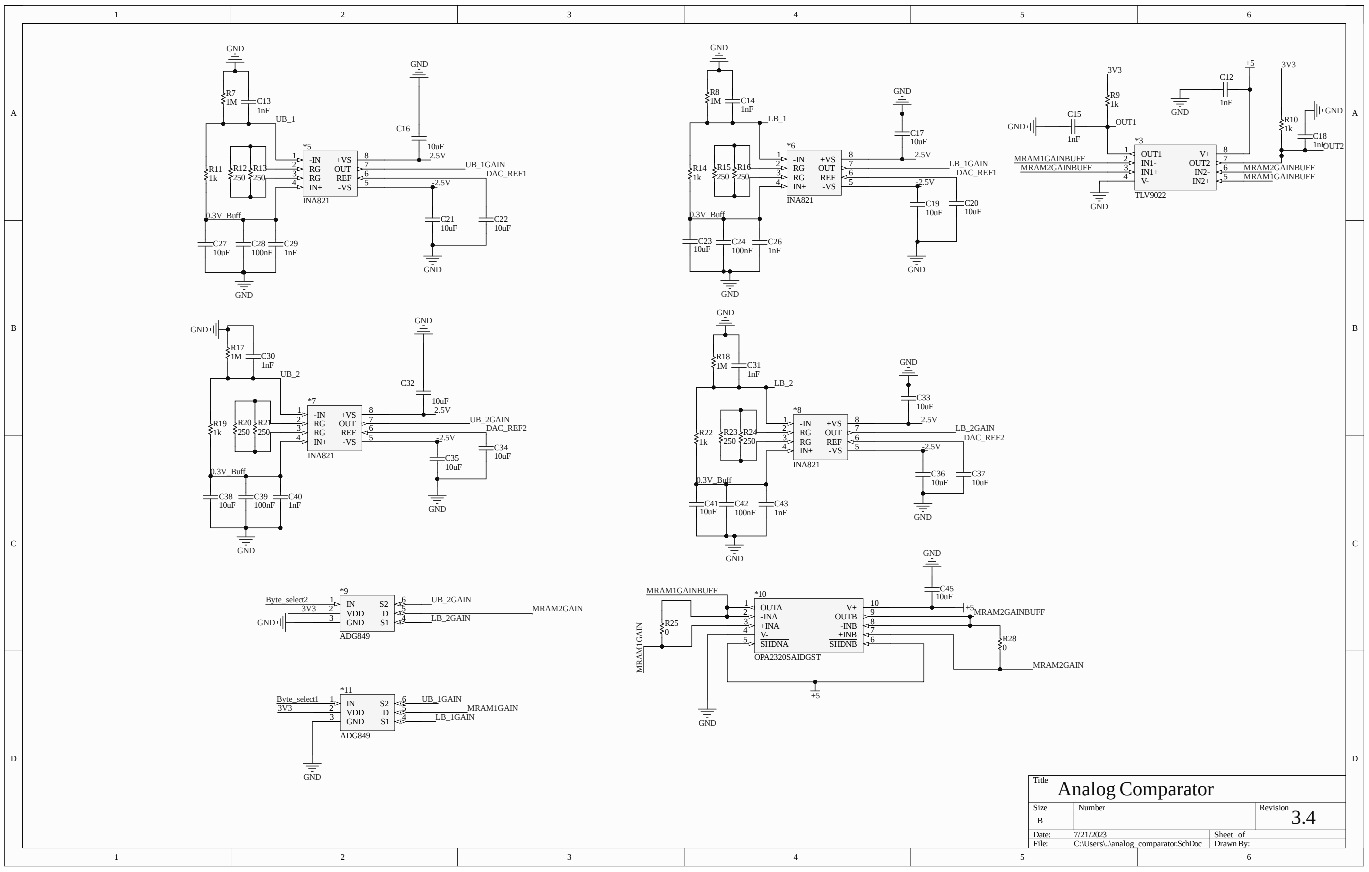

Pin Headers
Size B
Number
Revision 3.4
Date: 7/21/2023
Sheet of
File: C:\Users\..\MRAMShield3V4.SchDoc
Drawn By:
Nucleo144
MRAM1GAINBUFF
MRAM2GAINBUFF
GAIN1OUT
GAIN2OUT
OUT1
OUT2
P2
P4
5-146280-2
C25 10uF
C44 10uF
R26 10k
R27 10k
3V3
GND
SDA
SCL
F1
F2



U1
LTC3265
VIN N
VIN P
CINV+
CINV-
CBST+
CBST-
EN+
EN-
MODE
RT
VOUT+
VOUT-
LDO+
LDO-
ADJ+
ADJ-
BYP+
BYP-
NC
GND
+5
C1 10uF50V
C2 10uF50V
C3 10uF25V
C4 1uF50V
C7 1uF25V
R1 1M
R2 1M
C5 0.1uF
C6 0.1uF
P1
61300411121
2.5V
-2.5V
C8 10pF
C9 10pF
C10 10uF50V
C11 10uF50V
R3 383K
R4 120K
R5 120K
R6 383K
GND
Title
Dual Power Supply
Size
B
Number
Revision
3.4
Date: 7/21/2023
File: C:\Users\..\dual_power_supply.SchDoc
Sheet of
Drawn By:



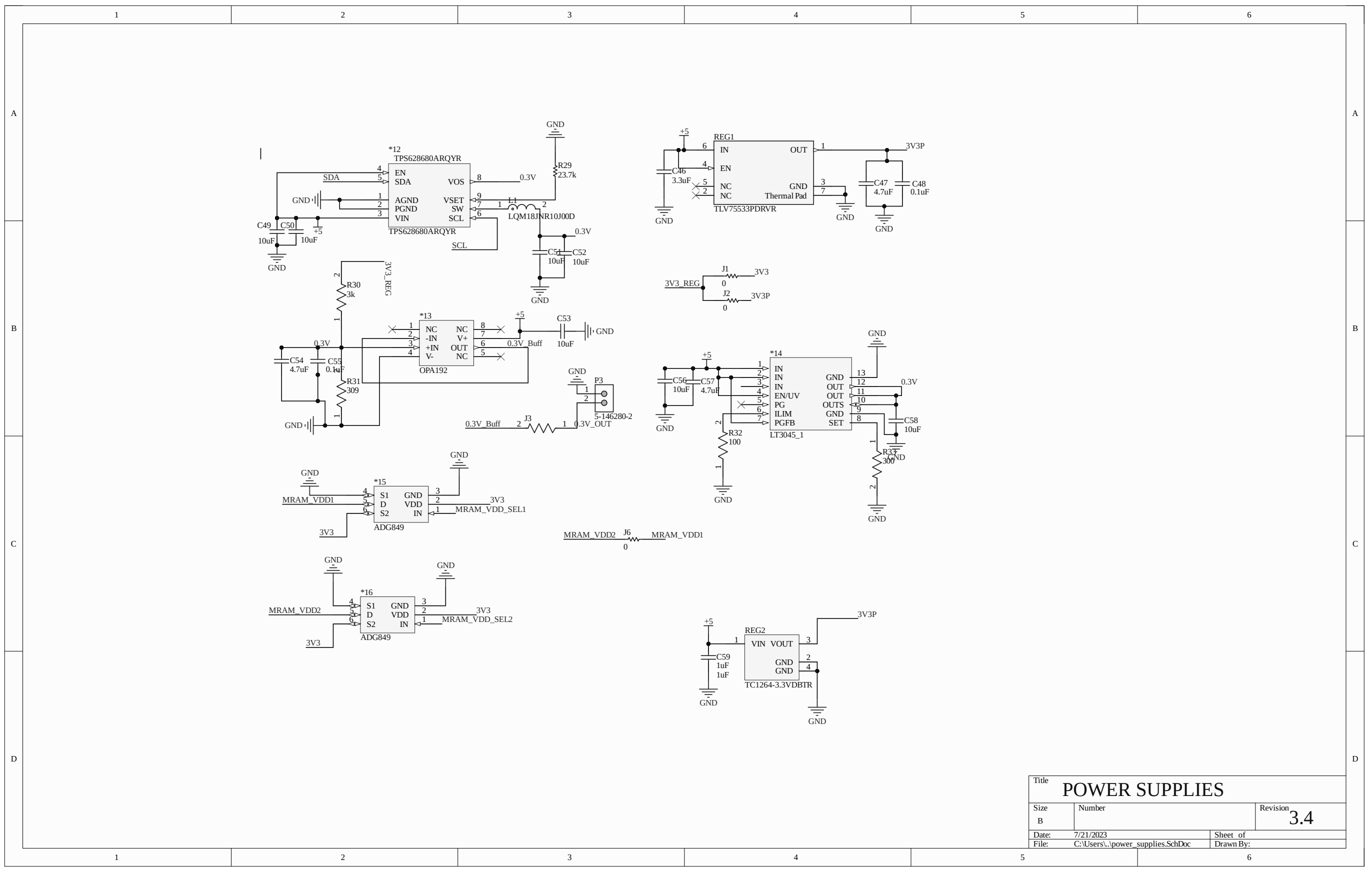
GND
*1
M4RA16TSSOP
MRAM_VDD1
*2
M4RA16TSSOP
MRAM_VDD2
A19 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9
E W
DQ0 DQ1 DQ2 DQ3 DQ4 DQ5 DQ6 DQ7
VDD VSS
DQ15 DQ14 DQ13 DQ12 DQ11 DQ10 DQ9 DQ8
G UB LB DC
A18 A17 A16 A15 A14 A13 A12 A11 A10
A19_1 A0_1 A1_1 A2_1 A3_1 A4_1 A5_1 A6_1 A7_1 A8_1 A9_1
E_1 W_1
DQ0_1 DQ1_1 DQ2_1 DQ3_1 DQ4_1 DQ5_1 DQ6_1 DQ7_1
DQ15_1 DQ14_1 DQ13_1 DQ12_1 DQ11_1 DQ10_1 DQ9_1 DQ8_1
G_1 UB_1 LB_1 DC_1
A18_1 A17_1 A16_1 A15_1 A14_1 A13_1 A12_1 A11_1 A10_1
A19_2 A0_2 A1_2 A2_2 A3_2 A4_2 A5_2 A6_2 A7_2 A8_2 A9_2
E_2 W_2
DQ0_2 DQ1_2 DQ2_2 DQ3_2 DQ4_2 DQ5_2 DQ6_2 DQ7_2
DQ15_2 DQ14_2 DQ13_2 DQ12_2 DQ11_2 DQ10_2 DQ9_2 DQ8_2
G_2 UB_2 LB_2 DC_2
A18_2 A17_2 A16_2 A15_2 A14_2 A13_2 A12_2 A11_2 A10_2
Title
MRAM Sockets
Size
B
Number
Revision
3.4
Date: 7/21/2023
Sheet of
File: C:\Users\..\mram_sockets.SchDoc
Drawn By:

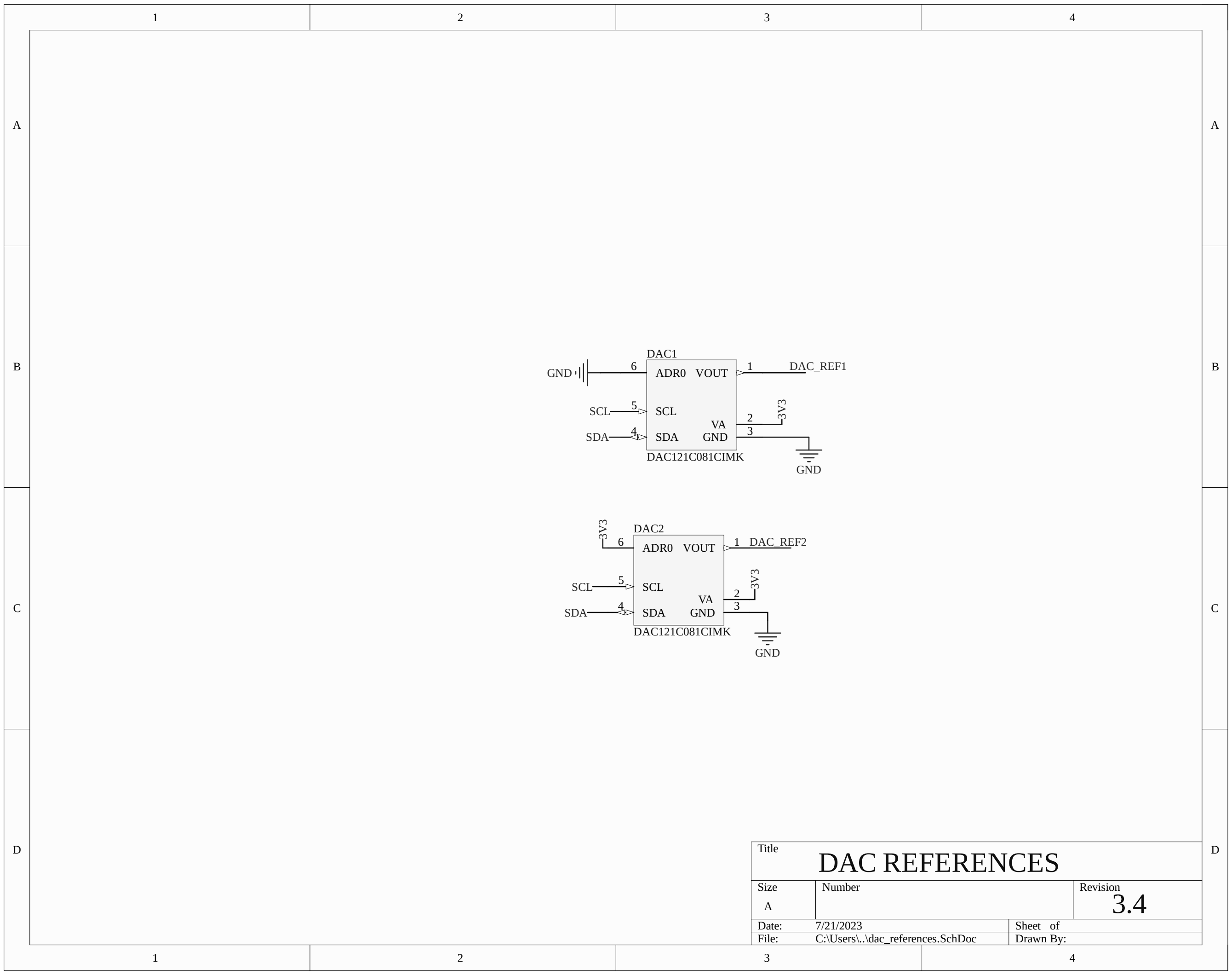


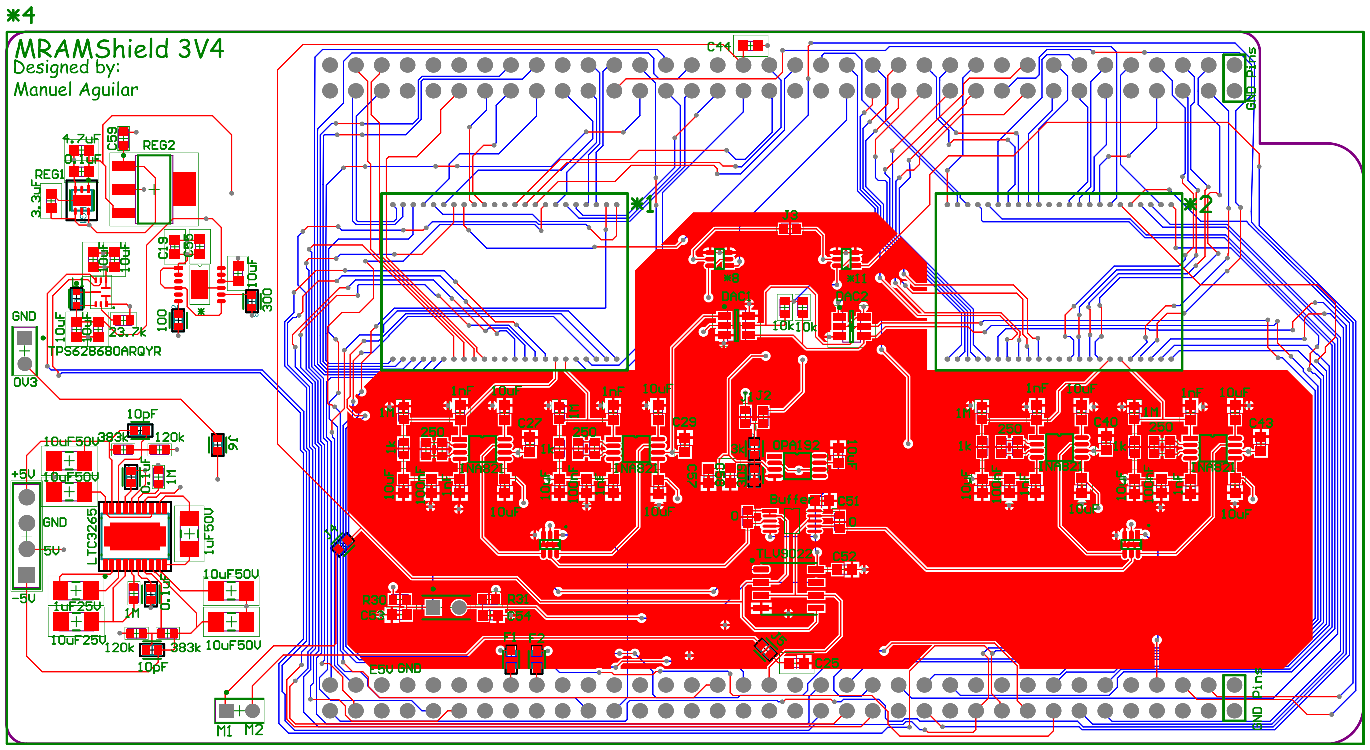
Title
Analog Comparator

Size	Number	Revision
B		3.4

Date:	7/21/2023	Sheet of	
File:	C:\Users\..\analog_comparator.SchDoc	Drawn By:	


*12
TPS628680ARQYR
EN
SDA
VOS
AGND
PGND
VIN
VSET
SW
SCL
0.3V
R29
23.7k
GND
L1
LQM18JNR10J00D
C49
10uF
C50
10uF
+5
C51
10uF
C52
10uF
SCL
3V3_REG
R30
3k
*13
NC
-IN
+IN
V-
V+
OUT
OPA192
C53
10uF
0.3V_Buff
C54
4.7uF
C55
0.1uF
R31
309
P3
5-146280-2
J3
0.3V_OUT
REG1
IN
OUT
Thermal Pad
TLV75533PDRVR
C46
3.3uF
C47
4.7uF
C48
0.1uF
3V3P
J1
0
3V3
J2
0
*14
EN/UV
PG
ILIM
PGFB
OUTS
SET
LT3045_1
C56
10uF
C57
4.7uF
R32
100
R33
300
C58
10uF
*15
S1
D
S2
VDD
IN
ADG849
MRAM_VDD1
MRAM_VDD_SEL1
*16
MRAM_VDD2
MRAM_VDD_SEL2
J6
REG2
VIN
VOUT
TC1264-3.3VDBTR
C59
1uF
Title
POWER SUPPLIES
Size
B
Number
Revision
3.4
Date:
7/21/2023
Sheet of
File:
C:\Users\..\power_supplies.SchDoc
Drawn By:


1
2
3
4
A
B
C
D
DAC1
GND
6
ADR0
VOUT
1
DAC_REF1
SCL
5
SCL
SDA
4
SDA
VA
GND
2
3
3V3
DAC121C081CIMK
GND
DAC2
3V3
6
ADR0
VOUT
1
DAC_REF2
SCL
5
SCL
SDA
4
SDA
VA
GND
2
3
3V3
DAC121C081CIMK
GND
Title
DAC REFERENCES
Size
A
Number
Revision
3.4
Date: 7/21/2023
Sheet of
File: C:\Users\..\dac_references.SchDoc
Drawn By:

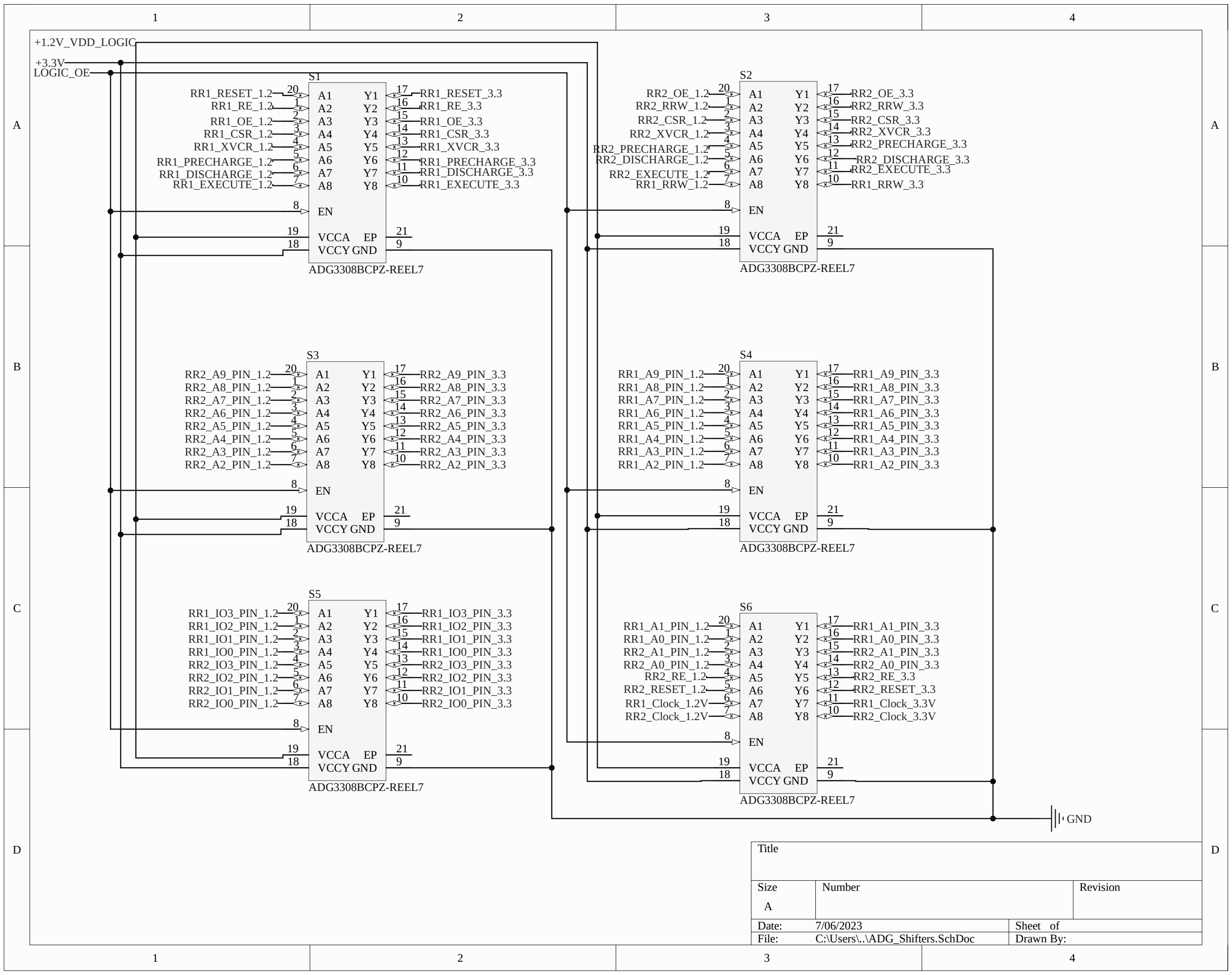

MRAMShield 3V4
Designed by:
Manuel Aguilar
REG1
REG2
TPS628680ARQYR
LTC3265
GND
+5V
-5V
0V3
M1
M2
C44
C59
C19
OPA192
TLV9062
F1
F2

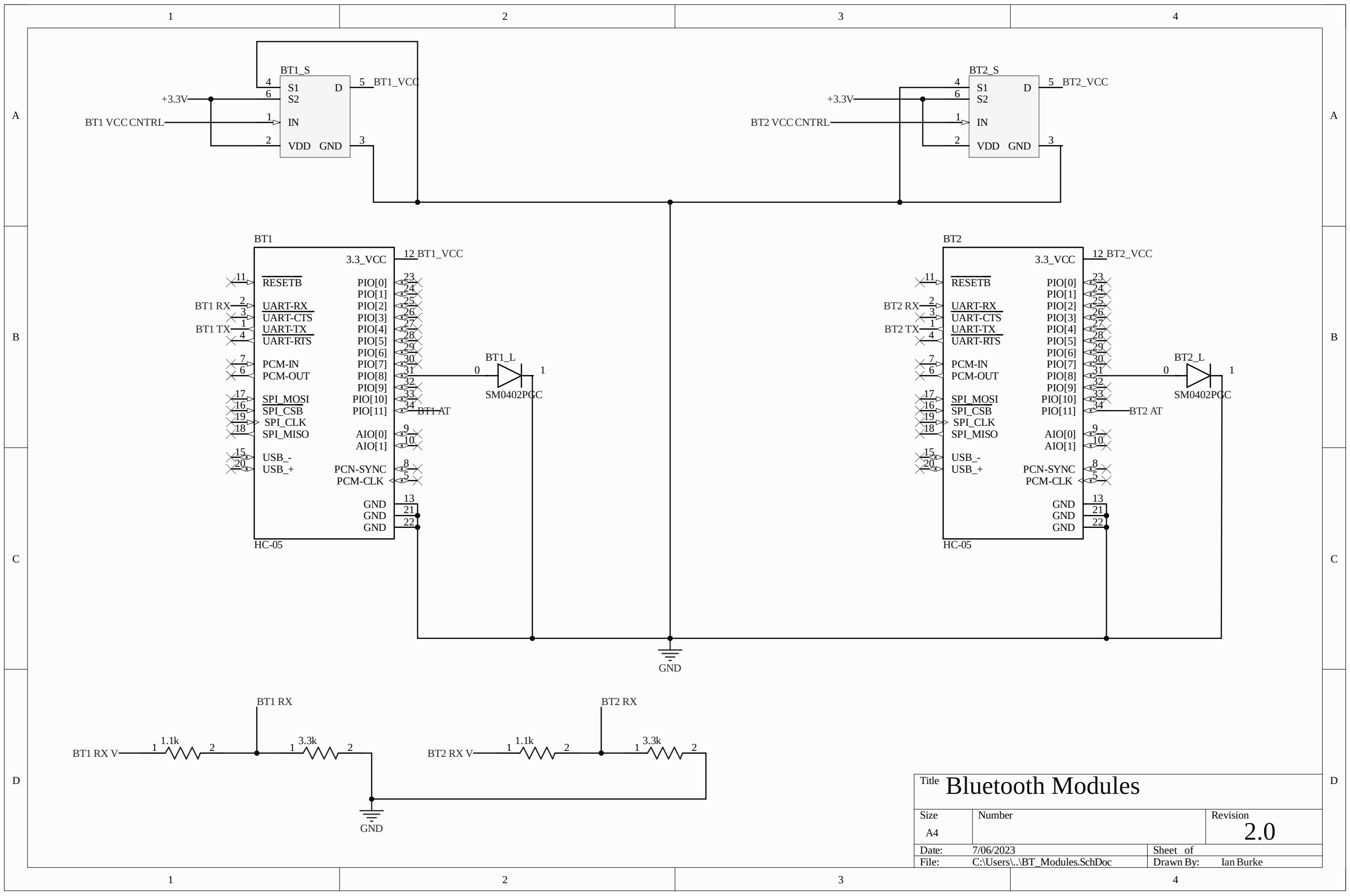
Appendix C

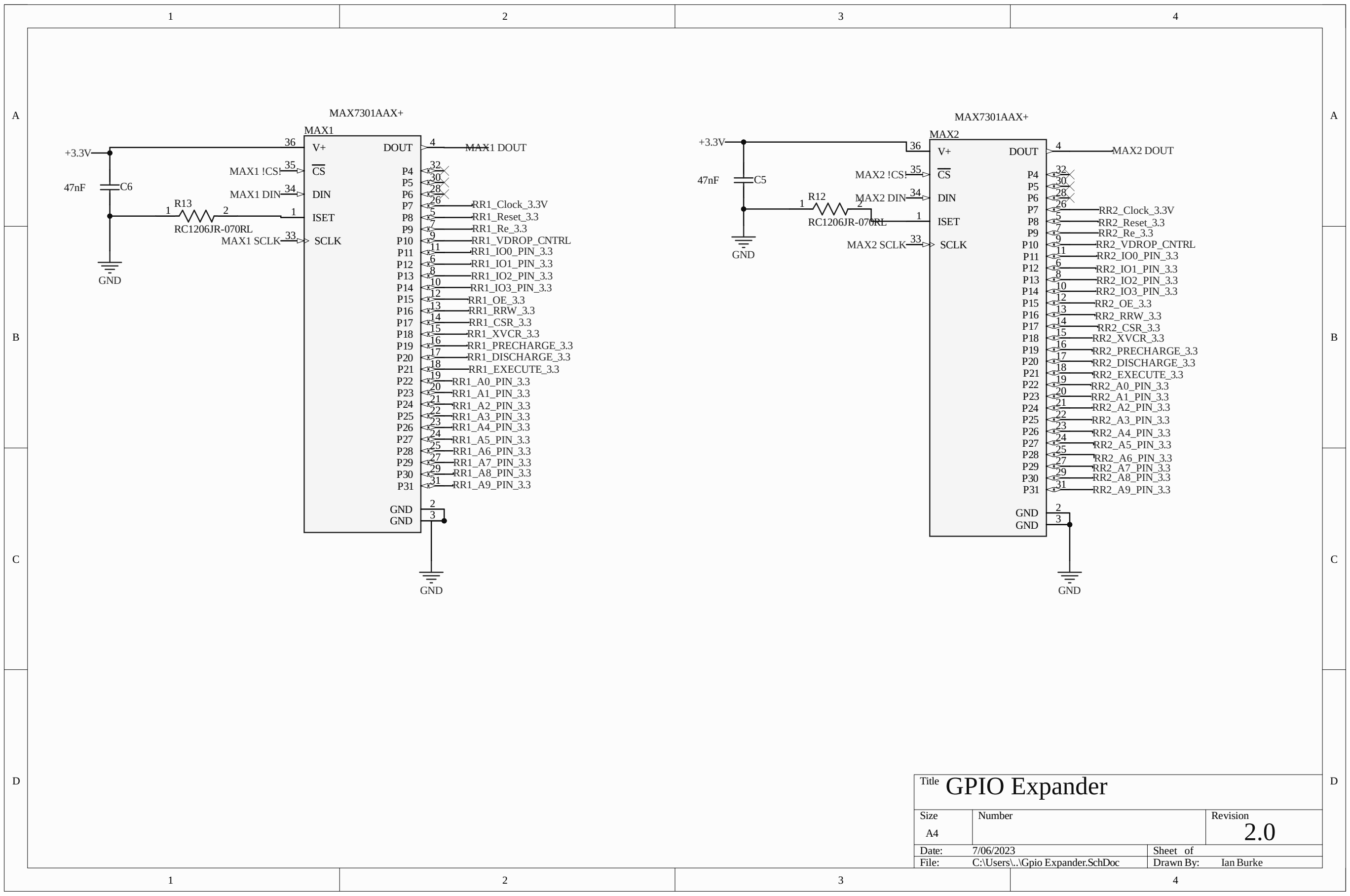
Supplementary Information for Chapter 5

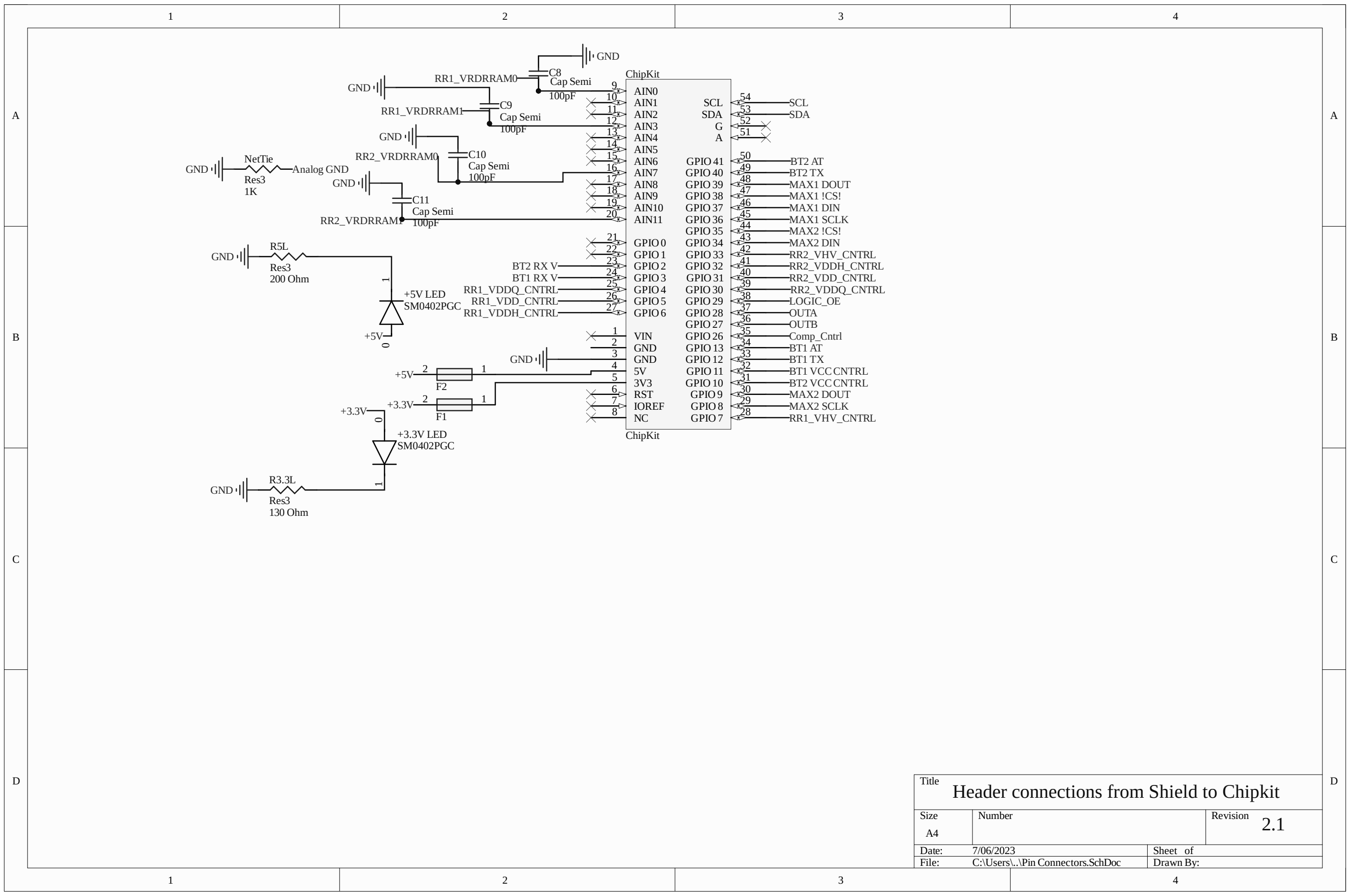
The supplementary material for this chapter consists of schematics and Printed Circuit Board (PCB) layouts of the Analog Key Encapsulation circuitry used. The first set of attachments includes the schematics and PCB layout for the initial implementation of the Analog Key Encapsulation protocol, which utilized software comparisons, as discussed in §5.4.0.1.

The second set of attachments comprises schematics and PCB layout for the second implementation of the Analog Key Encapsulation, which involved hardware comparisons, as discussed in §5.5.2.

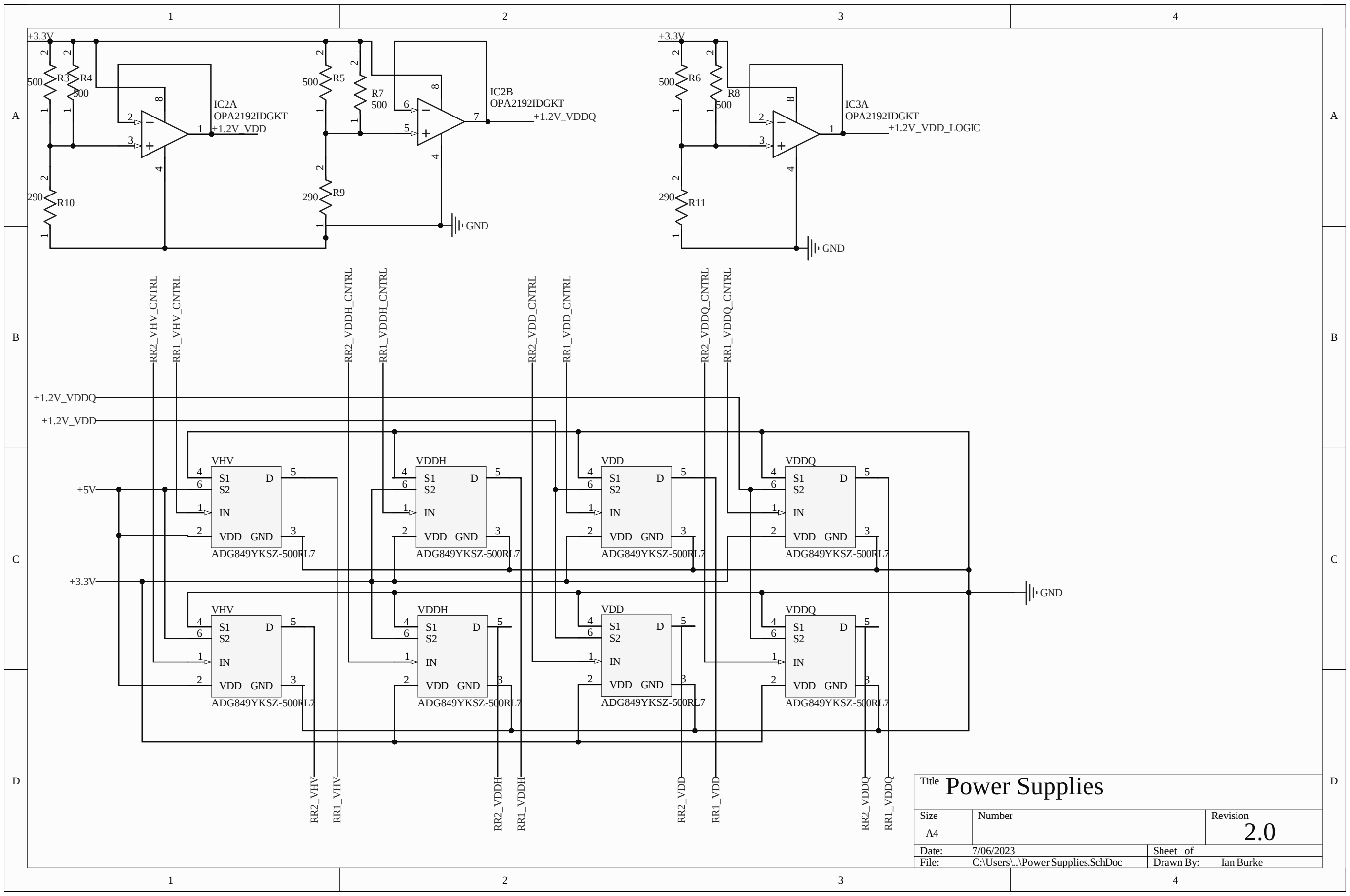

+1.2V_VDD_LOGIC
+3.3V
LOGIC_OE
S1
RR1_RESET_1.2
RR1_RE_1.2
RR1_OE_1.2
RR1_CSR_1.2
RR1_XVCR_1.2
RR1_PRECHARGE_1.2
RR1_DISCHARGE_1.2
RR1_EXECUTE_1.2
RR1_RESET_3.3
RR1_RE_3.3
RR1_OE_3.3
RR1_CSR_3.3
RR1_XVCR_3.3
RR1_PRECHARGE_3.3
RR1_DISCHARGE_3.3
RR1_EXECUTE_3.3
ADG3308BCPZ-REEL7
S2
RR2_OE_1.2
RR2_RRW_1.2
RR2_CSR_1.2
RR2_XVCR_1.2
RR2_PRECHARGE_1.2
RR2_DISCHARGE_1.2
RR2_EXECUTE_1.2
RR1_RRW_1.2
RR2_OE_3.3
RR2_RRW_3.3
RR2_CSR_3.3
RR2_XVCR_3.3
RR2_PRECHARGE_3.3
RR2_DISCHARGE_3.3
RR2_EXECUTE_3.3
RR1_RRW_3.3
ADG3308BCPZ-REEL7
S3
RR2_A9_PIN_1.2
RR2_A8_PIN_1.2
RR2_A7_PIN_1.2
RR2_A6_PIN_1.2
RR2_A5_PIN_1.2
RR2_A4_PIN_1.2
RR2_A3_PIN_1.2
RR2_A2_PIN_1.2
RR2_A9_PIN_3.3
RR2_A8_PIN_3.3
RR2_A7_PIN_3.3
RR2_A6_PIN_3.3
RR2_A5_PIN_3.3
RR2_A4_PIN_3.3
RR2_A3_PIN_3.3
RR2_A2_PIN_3.3
ADG3308BCPZ-REEL7
S4
RR1_A9_PIN_1.2
RR1_A8_PIN_1.2
RR1_A7_PIN_1.2
RR1_A6_PIN_1.2
RR1_A5_PIN_1.2
RR1_A4_PIN_1.2
RR1_A3_PIN_1.2
RR1_A2_PIN_1.2
RR1_A9_PIN_3.3
RR1_A8_PIN_3.3
RR1_A7_PIN_3.3
RR1_A6_PIN_3.3
RR1_A5_PIN_3.3
RR1_A4_PIN_3.3
RR1_A3_PIN_3.3
RR1_A2_PIN_3.3
ADG3308BCPZ-REEL7
S5
RR1_IO3_PIN_1.2
RR1_IO2_PIN_1.2
RR1_IO1_PIN_1.2
RR1_IO0_PIN_1.2
RR2_IO3_PIN_1.2
RR2_IO2_PIN_1.2
RR2_IO1_PIN_1.2
RR2_IO0_PIN_1.2
RR1_IO3_PIN_3.3
RR1_IO2_PIN_3.3
RR1_IO1_PIN_3.3
RR1_IO0_PIN_3.3
RR2_IO3_PIN_3.3
RR2_IO2_PIN_3.3
RR2_IO1_PIN_3.3
RR2_IO0_PIN_3.3
ADG3308BCPZ-REEL7
S6
RR1_A1_PIN_1.2
RR1_A0_PIN_1.2
RR2_A1_PIN_1.2
RR2_A0_PIN_1.2
RR2_RE_1.2
RR2_RESET_1.2
RR1_Clock_1.2V
RR2_Clock_1.2V
RR1_A1_PIN_3.3
RR1_A0_PIN_3.3
RR2_A1_PIN_3.3
RR2_A0_PIN_3.3
RR2_RE_3.3
RR2_RESET_3.3
RR1_Clock_3.3V
RR2_Clock_3.3V
ADG3308BCPZ-REEL7
A1 A2 A3 A4 A5 A6 A7 A8
Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8
EN
VCCA EP
VCCY GND
GND
Title
Size
A
Number
Revision
Date: 7/06/2023
Sheet of
File: C:\Users\..\ADG_Shifters.SchDoc
Drawn By:

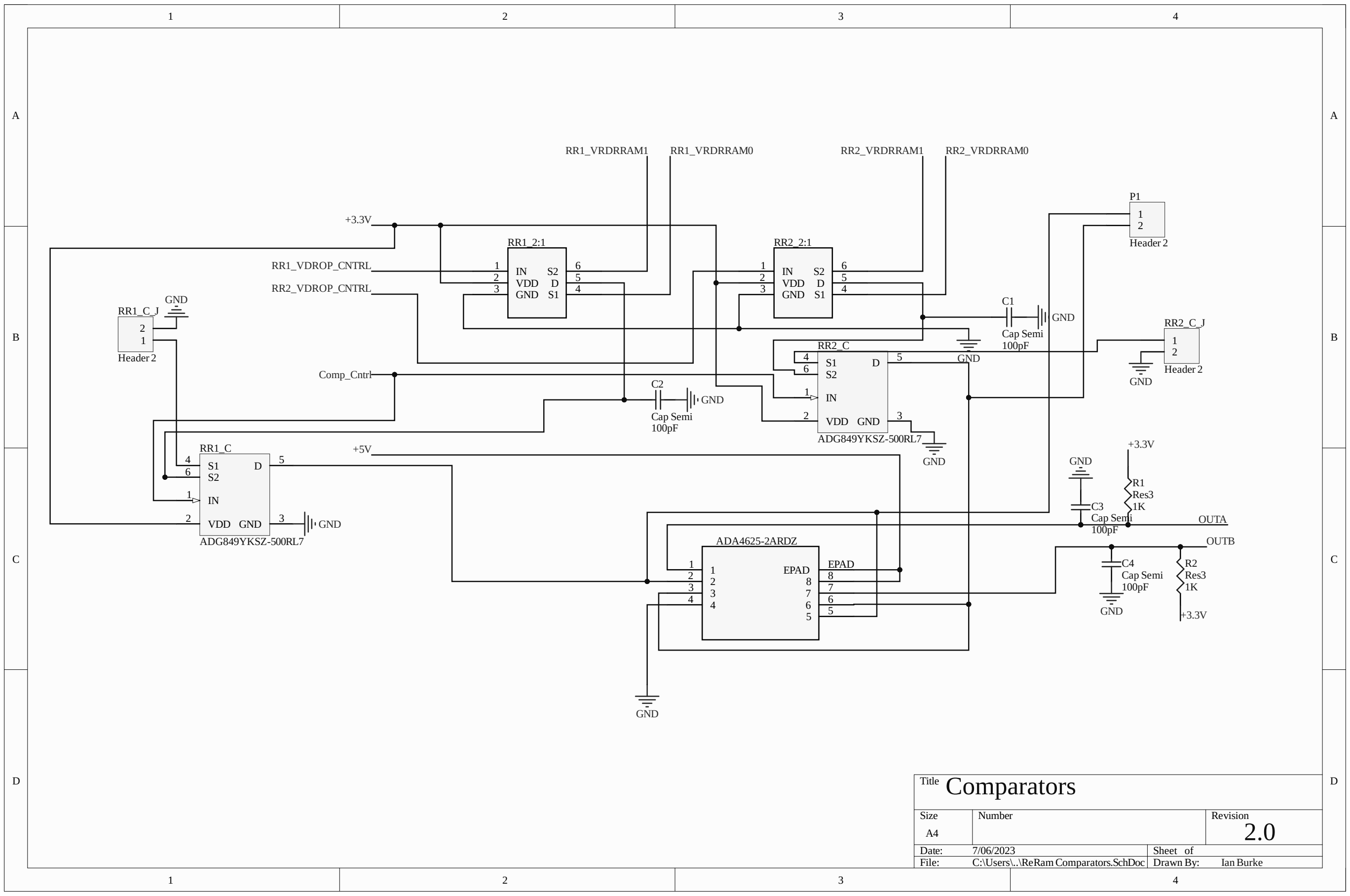

BT1_S
S1
S2
D
IN
VDD
GND
+3.3V
BT1 VCC CNTRL
BT1_VCC
BT2_S
BT2 VCC CNTRL
BT2_VCC
BT1
BT2
3.3_VCC
RESETB
UART-RX
UART-CTS
UART-TX
UART-RTS
PCM-IN
PCM-OUT
SPI_MOSI
SPI_CSB
SPI_CLK
SPI_MISO
USB_-
USB_+
PIO[0]
PIO[11]
AIO[0]
AIO[1]
PCN-SYNC
PCM-CLK
GND
HC-05
BT1 RX
BT1 TX
BT2 RX
BT2 TX
BT1 AT
BT2 AT
BT1_L
BT2_L
SM0402PGC
GND
BT1 RX V
BT2 RX V
1.1k
3.3k
Title Bluetooth Modules
Size A4
Number
Revision 2.0
Date: 7/06/2023
Sheet of
File: C:\Users\..\BT_Modules.SchDoc
Drawn By: Ian Burke


MAX7301AAX+
MAX1
+3.3V
47nF
C6
R13
RC1206JR-070RL
GND
MAX1 !CS!
MAX1 DIN
MAX1 SCLK
V+
CS
DIN
ISET
SCLK
DOUT
MAX1 DOUT
RR1_Clock_3.3V
RR1_Reset_3.3
RR1_Re_3.3
RR1_VDROP_CNTRL
RR1_IO0_PIN_3.3
RR1_IO1_PIN_3.3
RR1_IO2_PIN_3.3
RR1_IO3_PIN_3.3
RR1_OE_3.3
RR1_RRW_3.3
RR1_CSR_3.3
RR1_XVCR_3.3
RR1_PRECHARGE_3.3
RR1_DISCHARGE_3.3
RR1_EXECUTE_3.3
RR1_A0_PIN_3.3
RR1_A1_PIN_3.3
RR1_A2_PIN_3.3
RR1_A3_PIN_3.3
RR1_A4_PIN_3.3
RR1_A5_PIN_3.3
RR1_A6_PIN_3.3
RR1_A7_PIN_3.3
RR1_A8_PIN_3.3
RR1_A9_PIN_3.3
MAX7301AAX+
MAX2
C5
R12
MAX2 !CS!
MAX2 DIN
MAX2 SCLK
MAX2 DOUT
RR2_Clock_3.3V
RR2_Reset_3.3
RR2_Re_3.3
RR2_VDROP_CNTRL
RR2_IO0_PIN_3.3
RR2_IO1_PIN_3.3
RR2_IO2_PIN_3.3
RR2_IO3_PIN_3.3
RR2_OE_3.3
RR2_RRW_3.3
RR2_CSR_3.3
RR2_XVCR_3.3
RR2_PRECHARGE_3.3
RR2_DISCHARGE_3.3
RR2_EXECUTE_3.3
RR2_A0_PIN_3.3
RR2_A1_PIN_3.3
RR2_A2_PIN_3.3
RR2_A3_PIN_3.3
RR2_A4_PIN_3.3
RR2_A5_PIN_3.3
RR2_A6_PIN_3.3
RR2_A7_PIN_3.3
RR2_A8_PIN_3.3
RR2_A9_PIN_3.3
Title GPIO Expander
Size A4
Number
Revision 2.0
Date: 7/06/2023
Sheet of
File: C:\Users\..\Gpio Expander.SchDoc
Drawn By: Ian Burke

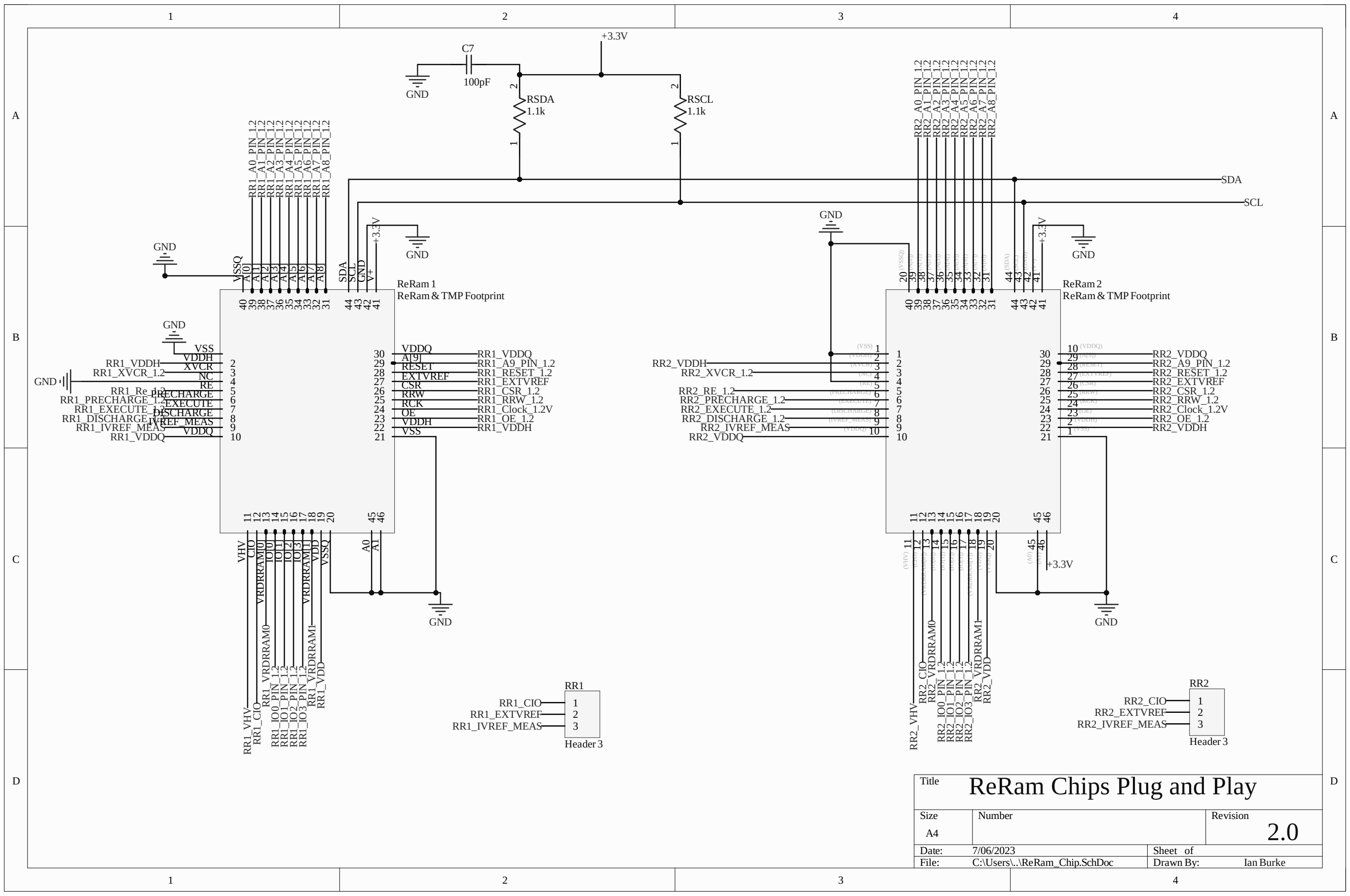


Header Connecti

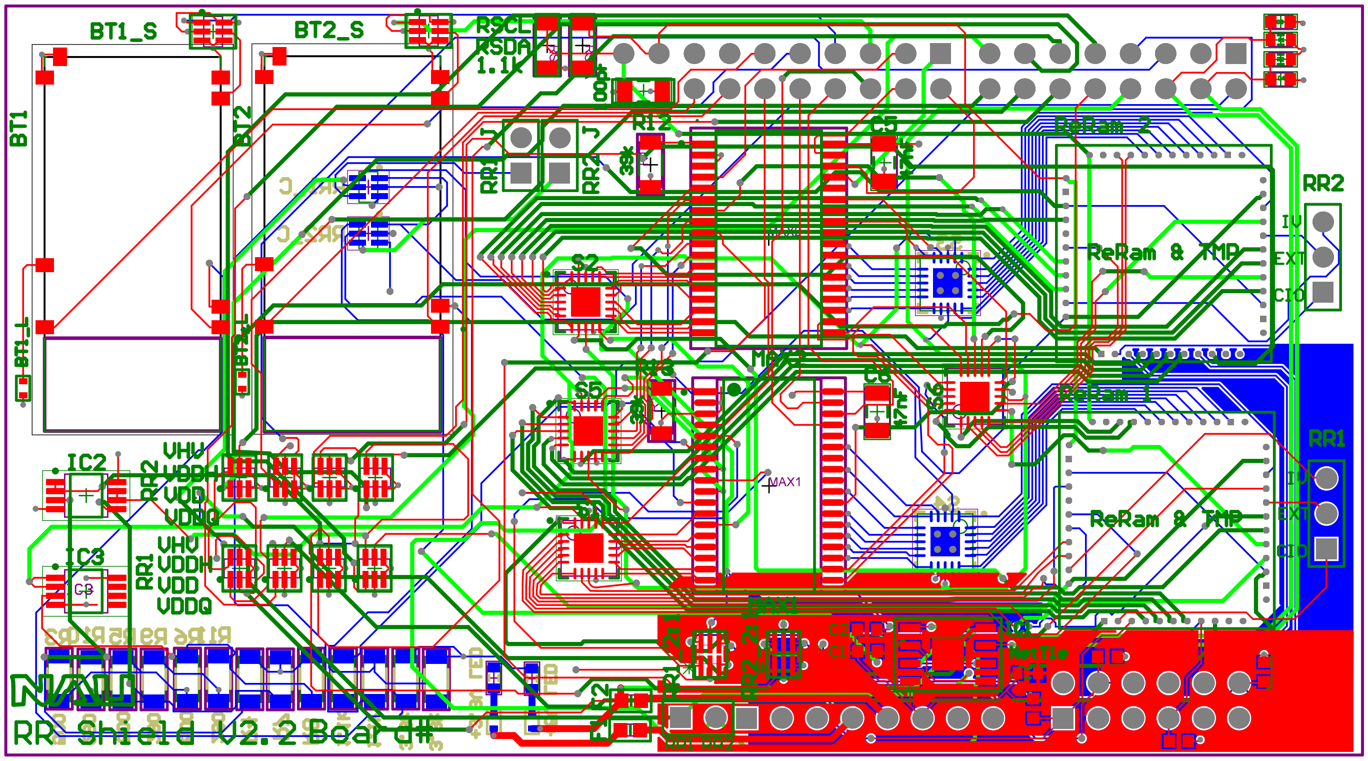

+3.3V
R3 500
R4 500
R10 290
IC2A OPA2192IDGKT
+1.2V_VDD
R5 500
R7 500
R9 290
IC2B OPA2192IDGKT
+1.2V_VDDQ
GND
R6 500
R8 500
R11 290
IC3A OPA2192IDGKT
+1.2V_VDD_LOGIC
RR2_VHV_CNTRL
RR1_VHV_CNTRL
RR2_VDDH_CNTRL
RR1_VDDH_CNTRL
RR2_VDD_CNTRL
RR1_VDD_CNTRL
RR2_VDDQ_CNTRL
RR1_VDDQ_CNTRL
+5V
VHV
VDDH
VDD
VDDQ
S1 S2 IN D VDD GND
ADG849YKSZ-500RL7
RR2_VHV
RR1_VHV
RR2_VDDH
RR1_VDDH
RR2_VDD
RR1_VDD
RR2_VDDQ
RR1_VDDQ
Title Power Supplies
Size A4
Number
Revision 2.0
Date: 7/06/2023
File: C:\Users\..\Power Supplies.SchDoc
Sheet of
Drawn By: Ian Burke

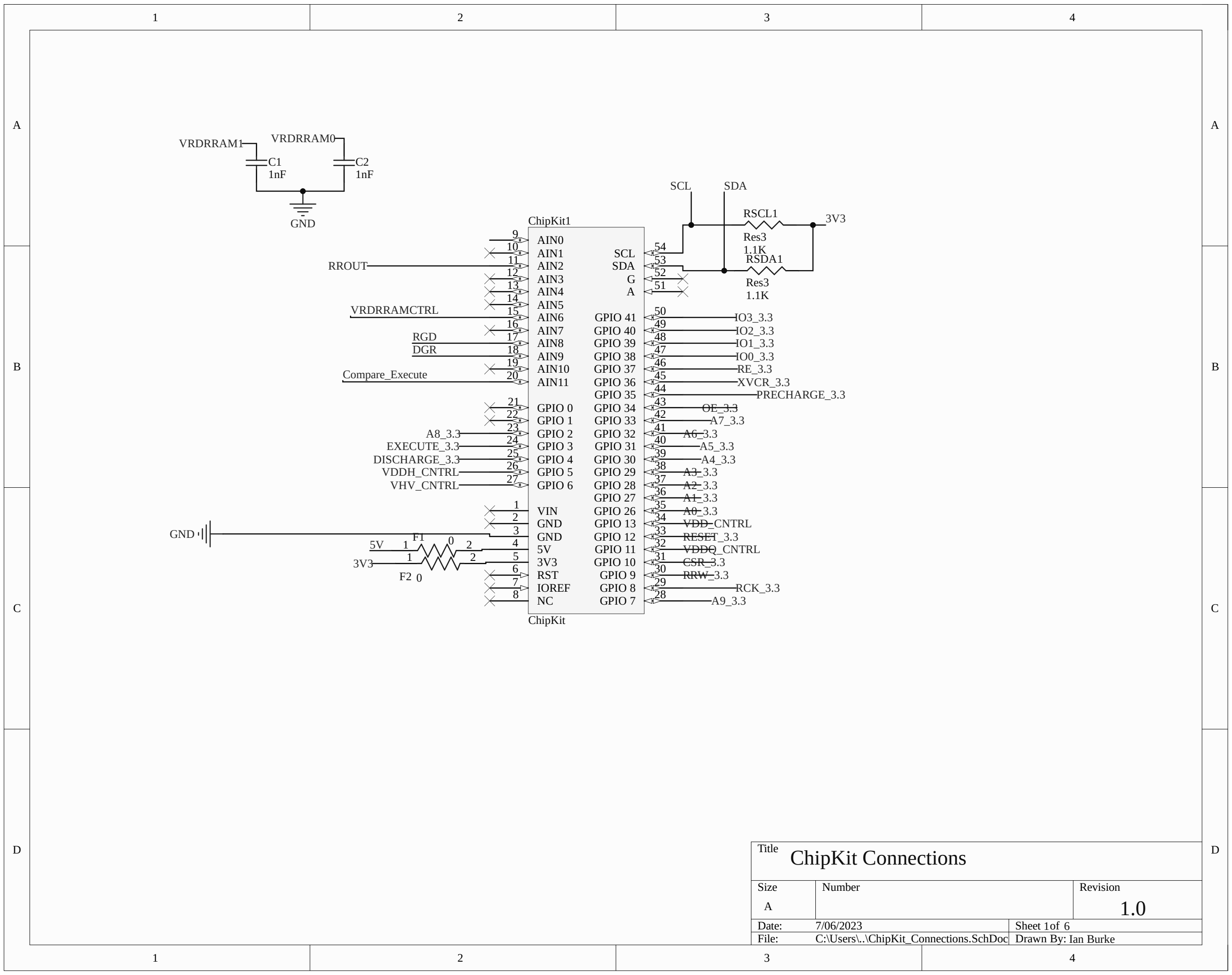


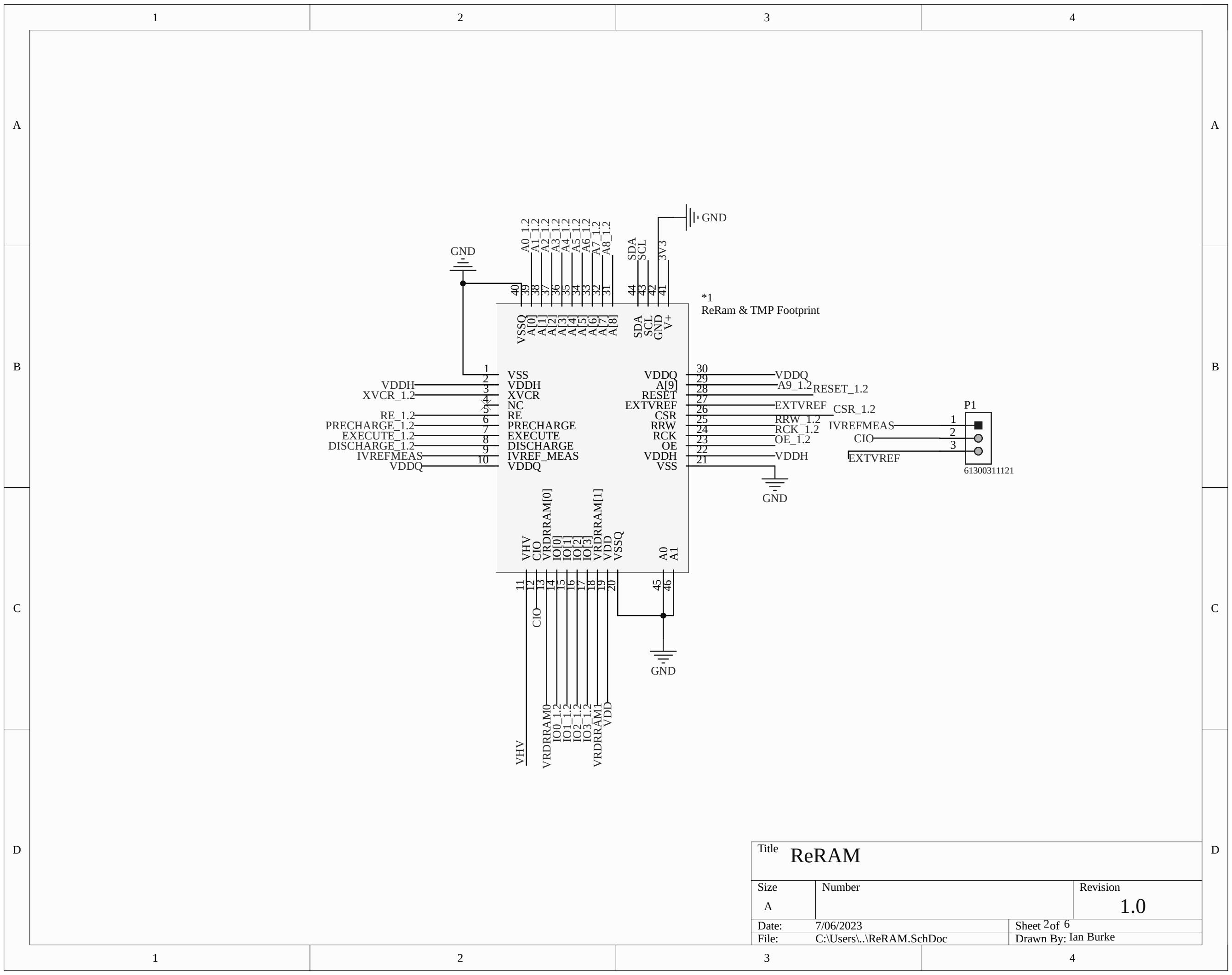
Title	Comparators		
Size A4	Number		Revision 2.0
Date: 7/06/2023		Sheet of	
File: C:\Users\..\ReRam Comparators.SchDoc		Drawn By:	Ian Burke

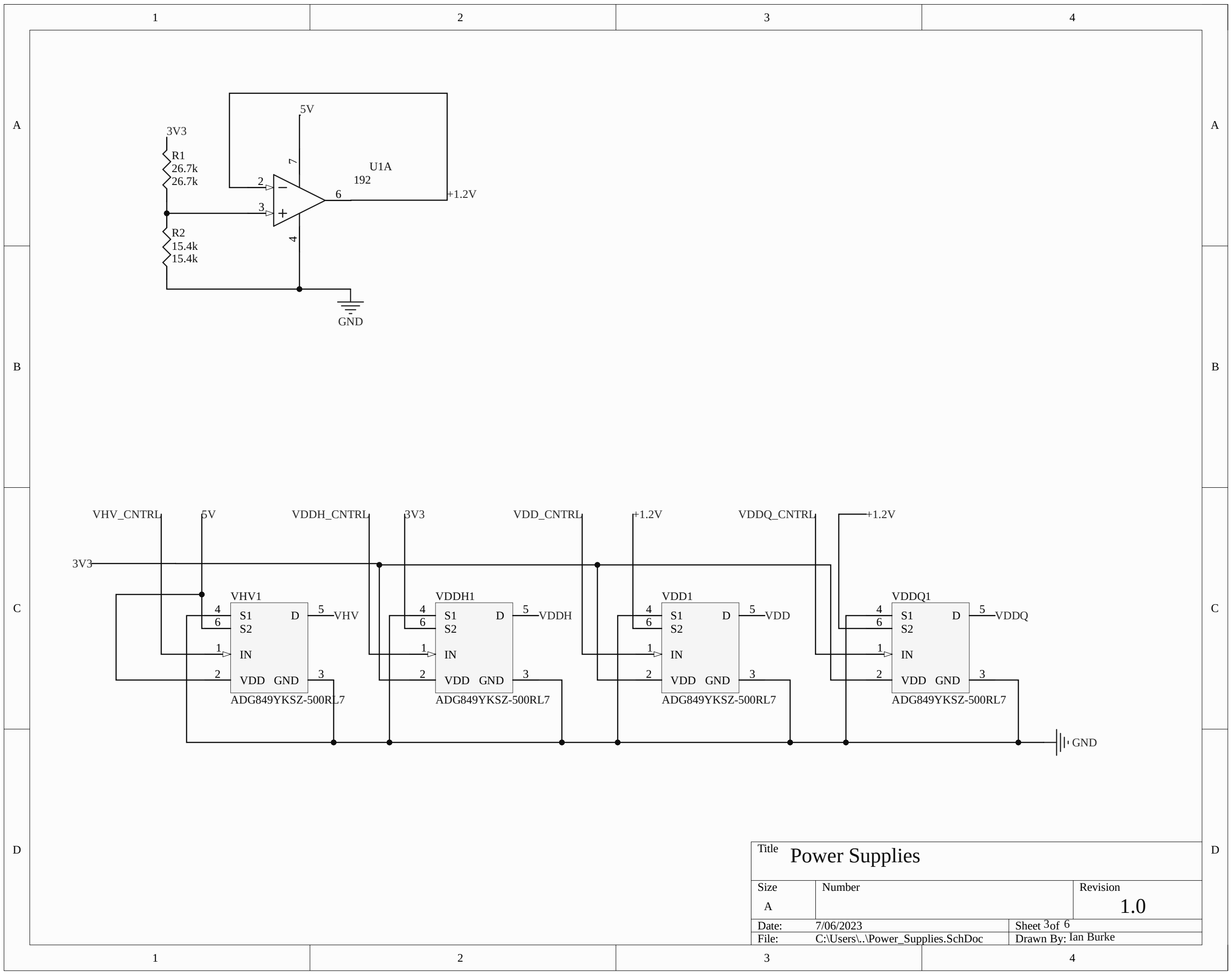


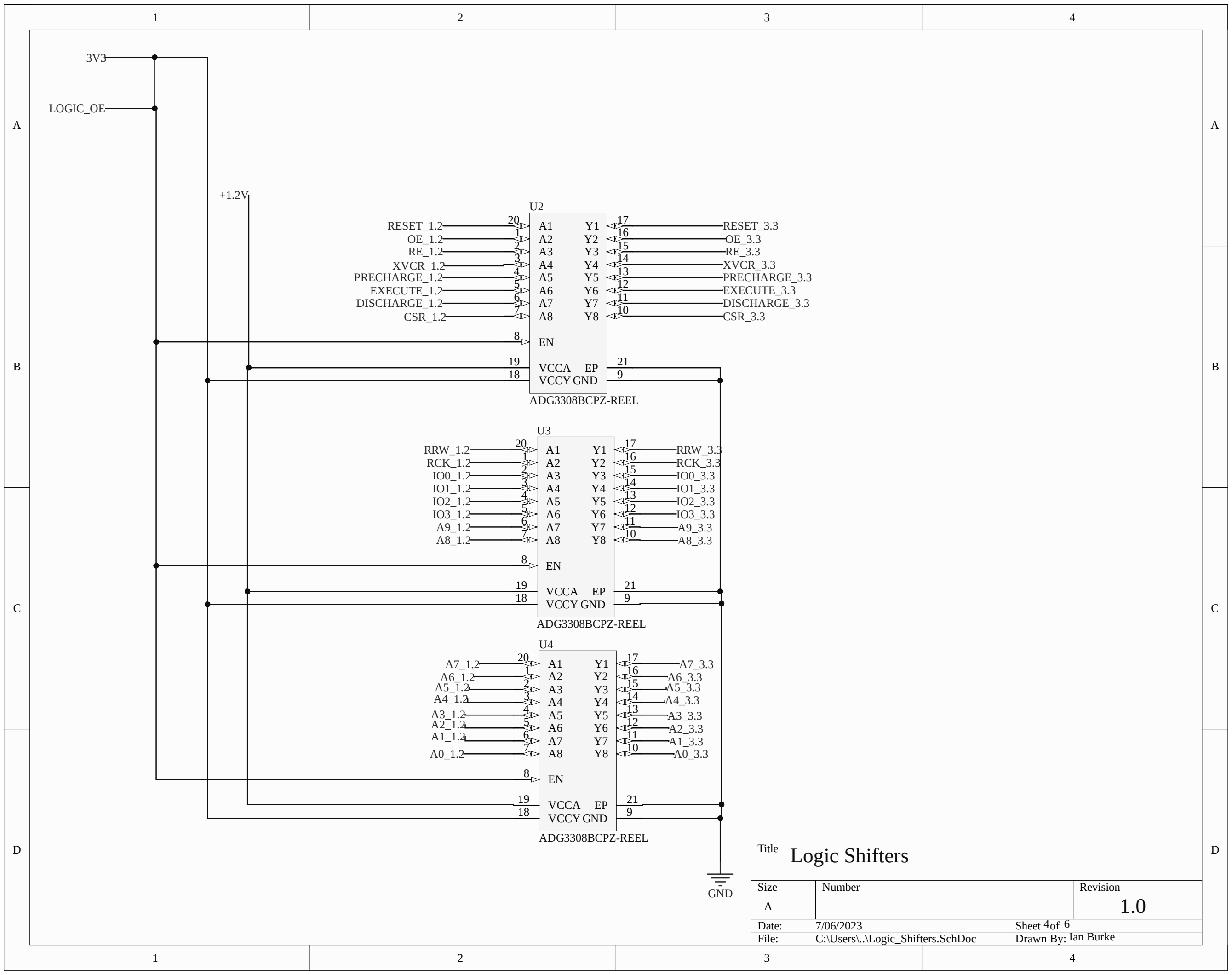
ReRam Chips Plug and Play
Size A4
Revision 2.0
Date: 7/06/2023
File: C:\Users\..\ReRam_Chip.SchDoc
Drawn By: Ian Burke
ReRam 1
ReRam & TMP Footprint
ReRam 2
ReRam & TMP Footprint
C7 100pF
RSDA 1.1k
RSCL 1.1k
+3.3V
GND
SDA
SCL
RR1 Header 3
RR1_CIO
RR1_EXTVREF
RR1_IVREF_MEAS
RR2 Header 3
RR2_CIO
RR2_EXTVREF
RR2_IVREF_MEAS


BT1_S
BT2_S
RSCL
RSDA
1.1k
BT1
BT2
RR1
RR2
R12
C5
ReRam 2
ReRam & TMP
RR2
IV
EXT
CIO
S2
S5
MAX2
C6
ReRam 1
RR1
MAX1
IC2
IC3
VHV
VDDH
VDD
VDDQ
NAU
RR Shield V2.2 Board#

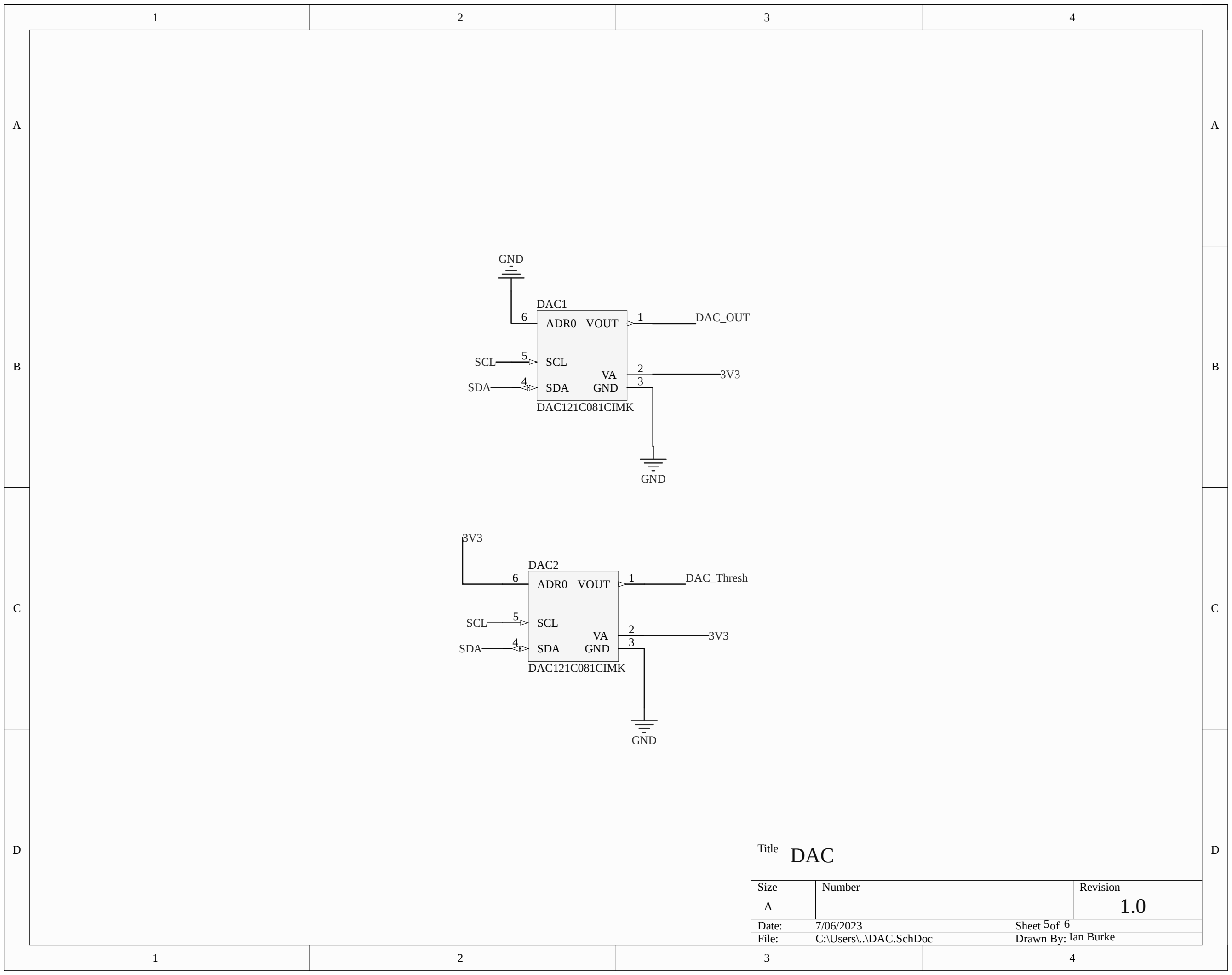

VRDRRAM1
VRDRRAM0
C1
1nF
C2
1nF
GND
SCL
SDA
RSCL1
Res3
1.1K
RSDA1
Res3
1.1K
3V3
ChipKit1
AIN0
AIN1
AIN2
AIN3
AIN4
AIN5
AIN6
AIN7
AIN8
AIN9
AIN10
AIN11
GPIO 0
GPIO 1
GPIO 2
GPIO 3
GPIO 4
GPIO 5
GPIO 6
VIN
GND
GND
5V
3V3
RST
IOREF
NC
SCL
SDA
G
A
GPIO 41
GPIO 40
GPIO 39
GPIO 38
GPIO 37
GPIO 36
GPIO 35
GPIO 34
GPIO 33
GPIO 32
GPIO 31
GPIO 30
GPIO 29
GPIO 28
GPIO 27
GPIO 26
GPIO 13
GPIO 12
GPIO 11
GPIO 10
GPIO 9
GPIO 8
GPIO 7
ChipKit
RROUT
VRDRRAMCTRL
RGD
DGR
Compare_Execute
A8_3.3
EXECUTE_3.3
DISCHARGE_3.3
VDDH_CNTRL
VHV_CNTRL
GND
5V
3V3
F1
F2
0
IO3_3.3
IO2_3.3
IO1_3.3
IO0_3.3
RE_3.3
XVCR_3.3
PRECHARGE_3.3
OE_3.3
A7_3.3
A6_3.3
A5_3.3
A4_3.3
A3_3.3
A2_3.3
A1_3.3
A0_3.3
VDD_CNTRL
RESET_3.3
VDDQ_CNTRL
CSR_3.3
RRW_3.3
RCK_3.3
A9_3.3
Title
ChipKit Connections
Size
A
Number
Revision
1.0
Date:
7/06/2023
Sheet 1of 6
File:
C:\Users\..\ChipKit_Connections.SchDoc
Drawn By: Ian Burke

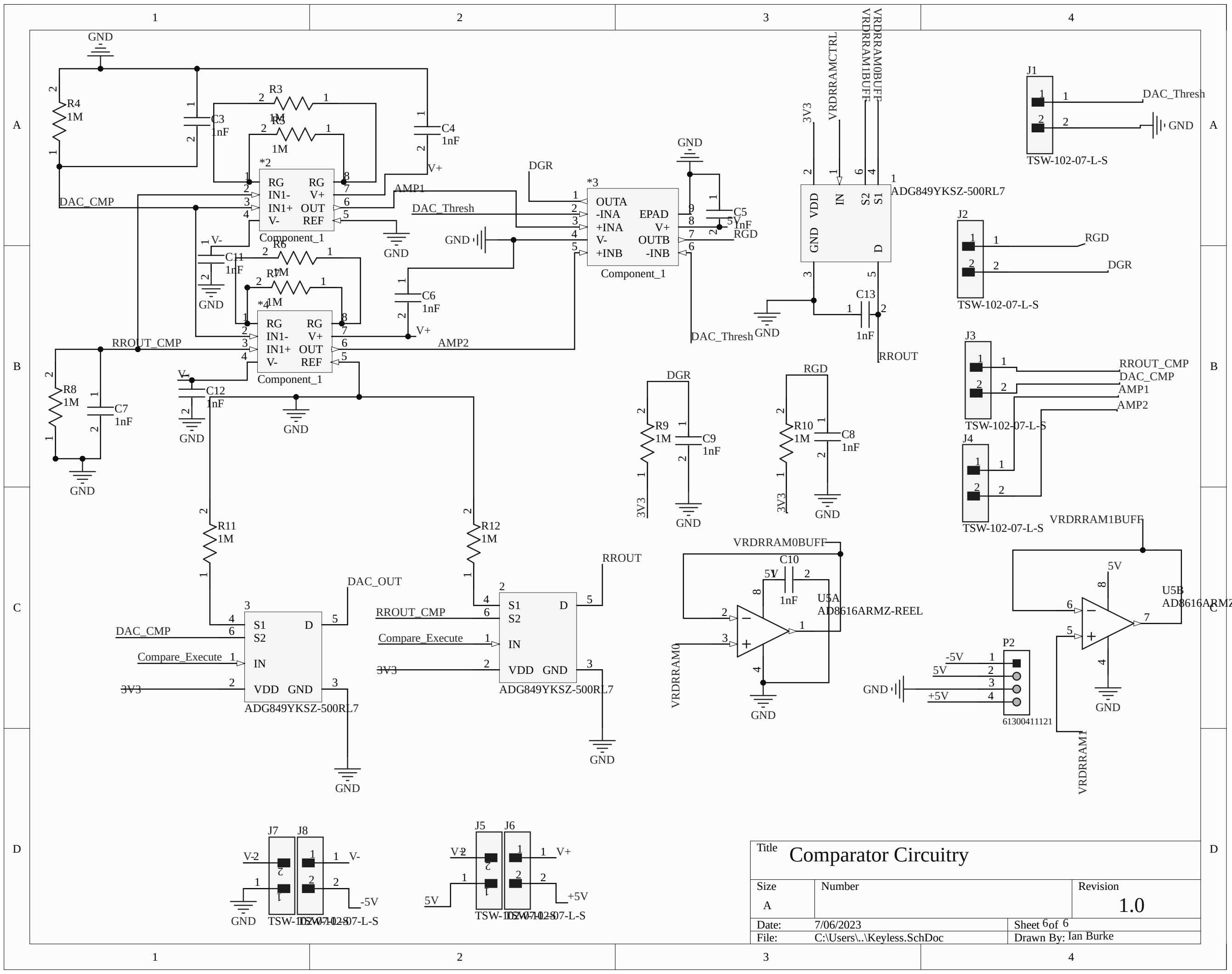

*1
ReRam & TMP Footprint
VSSQ
A[0]
A[1]
A[2]
A[3]
A[4]
A[5]
A[6]
A[7]
A[8]
SDA
SCL
GND
V+
A0_1.2
A1_1.2
A2_1.2
A3_1.2
A4_1.2
A5_1.2
A6_1.2
A7_1.2
A8_1.2
3V3
GND
VSS
VDDH
XVCR
NC
RE
PRECHARGE
EXECUTE
DISCHARGE
IVREF_MEAS
VDDQ
VDDH
XVCR_1.2
RE_1.2
PRECHARGE_1.2
EXECUTE_1.2
DISCHARGE_1.2
IVREFMEAS
VDDQ
VDDQ
A[9]
RESET
EXTVREF
CSR
RRW
RCK
OE
VDDH
VSS
A9_1.2
RESET_1.2
EXTVREF
CSR_1.2
RRW_1.2
RCK_1.2
OE_1.2
VDDH
P1
IVREFMEAS
CIO
EXTVREF
61300311121
VHV
CIO
VRDRRAM[0]
IO[0]
IO[1]
IO[2]
IO[3]
VRDRRAM[1]
VDD
VSSQ
A0
A1
VRDRRAM0
IO0_1.2
IO1_1.2
IO2_1.2
IO3_1.2
VRDRRAM1
VDD
Title ReRAM
Size A
Number
Revision 1.0
Date: 7/06/2023
Sheet 2 of 6
File: C:\Users\..\ReRAM.SchDoc
Drawn By: Ian Burke

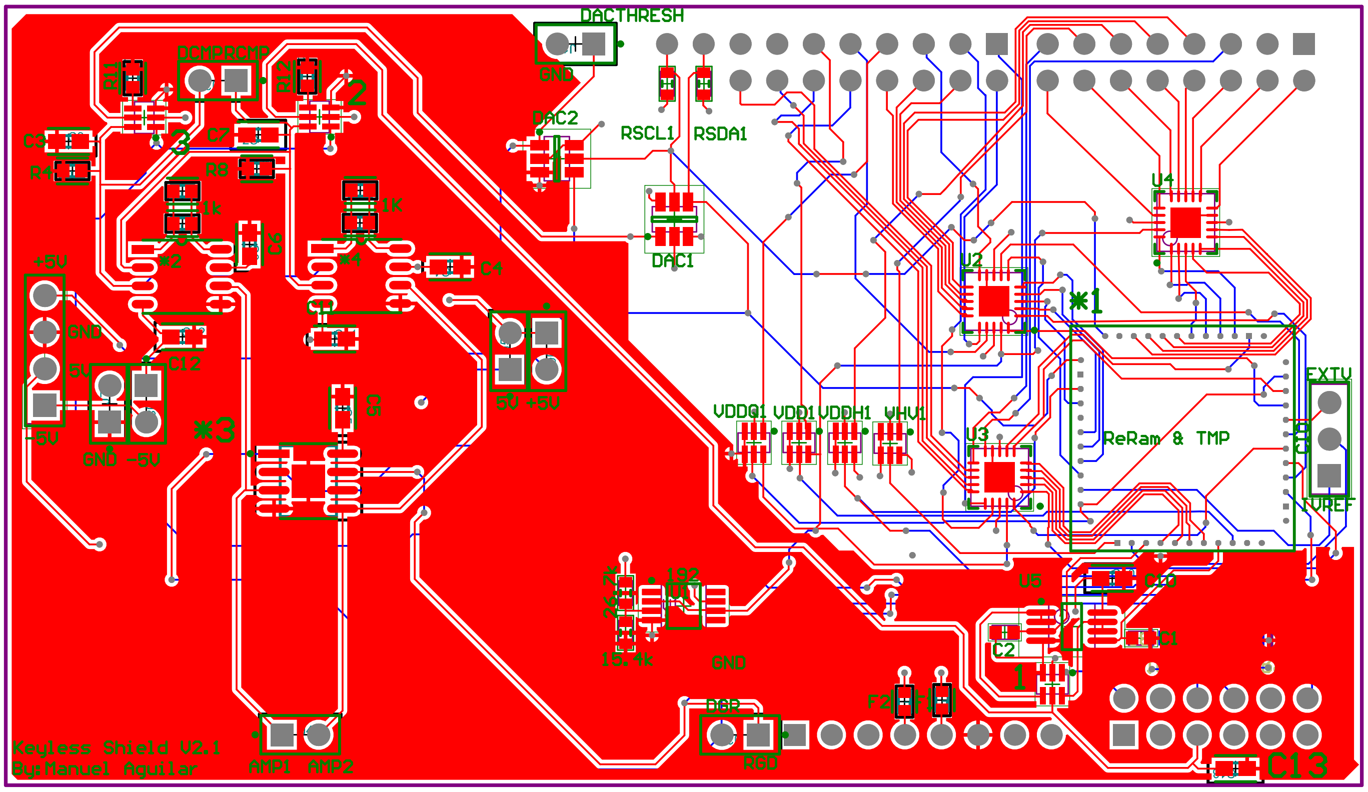

1
2
3
4
A
B
C
D
3V3
R1
26.7k
26.7k
R2
15.4k
15.4k
5V
U1A
192
+1.2V
GND
VHV_CNTRL
5V
VDDH_CNTRL
3V3
VDD_CNTRL
+1.2V
VDDQ_CNTRL
+1.2V
3V3
VHV1
VDDH1
VDD1
VDDQ1
S1
S2
D
IN
VDD
GND
VHV
VDDH
VDD
VDDQ
ADG849YKSZ-500RL7
GND
Title Power Supplies
Size A
Number
Revision 1.0
Date: 7/06/2023
Sheet 3 of 6
File: C:\Users\..\Power_Supplies.SchDoc
Drawn By: Ian Burke



3V3
LOGIC_OE
+1.2V
U2
RESET_1.2
OE_1.2
RE_1.2
XVCR_1.2
PRECHARGE_1.2
EXECUTE_1.2
DISCHARGE_1.2
CSR_1.2
A1 A2 A3 A4 A5 A6 A7 A8
Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8
EN
VCCA EP
VCCY GND
RESET_3.3
OE_3.3
RE_3.3
XVCR_3.3
PRECHARGE_3.3
EXECUTE_3.3
DISCHARGE_3.3
CSR_3.3
ADG3308BCPZ-REEL
U3
RRW_1.2
RCK_1.2
IO0_1.2
IO1_1.2
IO2_1.2
IO3_1.2
A9_1.2
A8_1.2
RRW_3.3
RCK_3.3
IO0_3.3
IO1_3.3
IO2_3.3
IO3_3.3
A9_3.3
A8_3.3
ADG3308BCPZ-REEL
U4
A7_1.2
A6_1.2
A5_1.2
A4_1.2
A3_1.2
A2_1.2
A1_1.2
A0_1.2
A7_3.3
A6_3.3
A5_3.3
A4_3.3
A3_3.3
A2_3.3
A1_3.3
A0_3.3
ADG3308BCPZ-REEL
GND
Title Logic Shifters
Size A
Number
Revision 1.0
Date: 7/06/2023
File: C:\Users\..\Logic_Shifters.SchDoc
Sheet 4 of 6
Drawn By: Ian Burke


GND
DAC1
6
ADR0
VOUT
1
DAC_OUT
SCL
5
SCL
VA
2
3V3
SDA
4
SDA
GND
3
DAC121C081CIMK
GND
3V3
DAC2
6
ADR0
VOUT
1
DAC_Thresh
SCL
5
SCL
VA
2
3V3
SDA
4
SDA
GND
3
DAC121C081CIMK
GND
Title DAC
Size A
Number
Revision 1.0
Date: 7/06/2023
Sheet 5 of 6
File: C:\Users\..\DAC.SchDoc
Drawn By: Ian Burke


Comparator Circuitry
Title
Size
A
Number
Revision
1.0
Date: 7/06/2023
Sheet 6 of 6
File: C:\Users\..\Keyless.SchDoc
Drawn By: Ian Burke
ADG849YKSZ-500RL7
AD8616ARMZ-REEL
TSW-102-07-L-S
DAC_Thresh
DAC_CMP
RROUT_CMP
AMP1
AMP2
RGD
DGR
RROUT
DAC_OUT
Compare_Execute
VRDRRAMCTRL
VRDRRAM0BUFF
VRDRRAM1BUFF
VRDRRAM0
VRDRRAM1
Component_1
61300411121


DACTHRESH
GND
DAC2
RSCL1
RSDA1
DAC1
U4
U2
*1
U3
ReRam & TMP
EXTV
VREF
VDDQ1
VDD1
VDDH1
VHV1
U5
C10
C2
C11
F2
GND
DQR
R60
C13
COMPBCMP
R11
R12
C3
C7
R4
R8
1k
1K
C6
*2
*4
C4
C1
+5V
GND
5V
-5V
C12
GND -5V
*3
C5
5V +5V
15.4k
182
AMP1 AMP2
Keyless Shield V2.1
By:Manuel Aguilar

References

[1] C. Böhm, M. Hofer, and W. Pribyl, "A microcontroller sram-puf," in *2011 5th International Conference on Network and System Security*, pp. 269–273, IEEE, 2011.

[2] S. Baek, G.-H. Yu, J. Kim, C. T. Ngo, J. K. Eshraghian, and J.-P. Hong, "A reconfigurable sram based cmos puf with challenge to response pairs," *IEEE Access*, vol. 9, pp. 79947–79960, 2021.

[3] D. E. Holcomb, W. P. Burleson, K. Fu, *et al.*, "Initial sram state as a fingerprint and source of true random numbers for rfid tags," in *Proceedings of the Conference on RFID Security*, vol. 7, p. 01, 2007.

[4] "Everspin technologies, inc."

[5] G. Prenat, K. Jabeur, G. Di Pendina, O. Boulle, and G. Gaudin, "Beyond stt-mram, spin orbit torque ram sot-mram for high speed and high reliability applications," *Spintronics-based Computing*, pp. 145–157, 2015.

[6] T. Wilson and B. Cambou, "Tamper-sensitive pre-formed reram-based pufs: Methods and experimental validation," *Frontiers in Nanotechnology*, p. 89, 2022.

[7] Y. Huang, Z. Shen, Y. Wu, X. Wang, S. Zhang, X. Shi, and H. Zeng, "Amorphous zno based resistive random access memory," *RSC advances*, vol. 6, no. 22, pp. 17867–17872, 2016.

[8] S. Assiri, B. Cambou, D. D. Booher, D. G. Miandoab, and M. Mohammadinodoushan, "Key exchange using ternary system to enhance security," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0488–0492, IEEE, 2019.

[9] B. Cambou, M. Gowanlock, B. Yildiz, D. Ghanaimiandoab, K. Lee, S. Nelson, C. Philabaum, A. Stenberg, and J. Wright, "Post quantum cryptographic keys generated with physical unclonable functions," *Applied Sciences*, vol. 11, no. 6, p. 2801, 2021.

[10] R. C. Merkle, "Secure communications over insecure channels," *Communications of the ACM*, vol. 21, no. 4, pp. 294–299, 1978.

[11] A. K. Lenstra, "Unbelievable security matching aes security using public key systems," in *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings*, pp. 67–86, Springer, 2001.

[12] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners.* Springer Science & Business Media, 2009.

[13] A. Al Hasib and A. A. M. M. Haque, "A comparative study of the performance and security issues of aes and rsa cryptography," in *2008 Third International Conference on Convergence and Hybrid Information Technology*, vol. 2, pp. 505–510, IEEE, 2008.

[14] H. Tange and B. Andersen, "Attacks and countermeasures on aes and ecc," in *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 1–5, IEEE, 2013.

[15] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography," *arXiv preprint arXiv:1804.00200*, 2018.

[16] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994.

[17] R. Anderson and M. Kuhn, "Tamper resistance-a cautionary note," in *Proceedings of the second Usenix workshop on electronic commerce*, vol. 2, pp. 1–11, 1996.

[18] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *International Workshop on Security Protocols*, pp. 125–136, Springer, 1997.

[19] G. J. Simmons, "A system for verifying user identity and authorization at the point-of sale or access," *Cryptologia*, vol. 8, no. 1, pp. 1–21, 1984.

[20] CSRC, "Cryptographic primitive - glossary."

[21] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography.* CRC press, 2018.

[22] R. Maes, *Physically unclonable functions: Constructions, properties and applications.* Springer Science & Business Media, 2013.

[23] M. Stipčević and Ç. K. Koç, "True random number generators," in *Open Problems in Mathematics and Computational Science*, pp. 275–315, Springer, 2014.

[24] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, "Physical unclonable function and true random number generator: a compact and scalable implementation," in *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, pp. 425–428, 2009.

[25] R. Soorat, A. Vudayagiri, *et al.*, "Hardware random number generator for cryptography," *arXiv preprint arXiv:1510.01234*, 2015.

[26] P. Kietzmann, T. C. Schmidt, and M. Wählisch, "A guideline on pseudorandom number generation (prng) in the iot," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–38, 2021.

[27] H. G. Katzgraber, "Random numbers in scientific computing: An introduction," *arXiv preprint arXiv:1005.4117*, 2010.

[28] L. Sreekumar and P. Ramesh, "Selection of an optimum entropy source design for a true random number generator," *Procedia Technology*, vol. 25, pp. 598–605, 2016.

[29] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Heckert, and D. Banks, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," 2010-09-16 2010.

[30] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 148–160, 2002.

[31] B. Cambou, P. G. Flikkema, J. Palmer, D. Telesca, and C. Philabaum, "Can ternary computing improve information assurance?," *Cryptography*, vol. 2, no. 1, p. 6, 2018.

[32] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *International workshop on cryptographic hardware and embedded systems*, pp. 63–80, Springer, 2007.

[33] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, and R. J. Young, "A puf taxonomy," *Applied Physics Reviews*, vol. 6, no. 1, p. 011303, 2019.

[34] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive puf analysis," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 30–38, IEEE, 2013.

[35] B. L. P. Gassend, *Physical random functions.* PhD thesis, Massachusetts Institute of Technology, 2003.

[36] J. D. Buchanan, R. P. Cowburn, A.-V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. A. Allwood, and M. T. Bryan, "'fingerprinting'documents and packaging," *Nature*, vol. 436, no. 7050, pp. 475–475, 2005.

[37] C. N. Chong, D. Jiang, J. Zhang, and L. Guo, "Anti-counterfeiting with a random pattern," in *2008 Second International Conference on Emerging Security Information, Systems and Technologies*, pp. 146–153, IEEE, 2008.

[38] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[39] S. Vrijaldenhoven *et al.*, "Acoustical physical uncloneable functions," *Philips internal publication PR-TN-2004-300300*, 2005.

[40] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," in *International Conference on Evolvable Systems*, pp. 390–405, Springer, 1996.

[41] R. Maes, A. Van Herrewege, and I. Verbauwhede, "Pufky: A fully functional puf-based cryptographic key generator," in *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*, pp. 302–319, Springer, 2012.

[42] S. Taneja, V. K. Rajanna, and M. Alioto, "36.1 unified in-memory dynamic trng and multi-bit static puf entropy generation for ubiquitous hardware security," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 498–500, IEEE, 2021.

[43] R. Maes and V. Van Der Leest, "Countering the effects of silicon aging on sram pufs," in *2014 IEEE International symposium on hardware-oriented security and trust (HOST)*, pp. 148–153, IEEE, 2014.

[44] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*, pp. 176–179, IEEE, 2004.

[45] W. Che, V. K. Kajuluri, M. Martin, F. Saqib, and J. Plusquellic, "Analysis of entropy in a hardware-embedded delay puf," *Cryptography*, vol. 1, no. 1, p. 8, 2017.

[46] M. Krarti, "Chapter 4 - utility rate structures and grid integration," in *Optimal Design and Retrofit of Energy Efficient Buildings, Communities, and Urban Centers* (M. Krarti, ed.), pp. 189–245, Butterworth-Heinemann, 2018.

[47] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. Van der Sluis, and V. van der Leest, "Experimental evaluation of physically unclonable functions in 65 nm cmos," in *2012 Proceedings of the ESSCIRC (ESSCIRC)*, pp. 486–489, IEEE, 2012.

[48] V. Pless, *Introduction to the theory of error-correcting codes*, vol. 48. John Wiley & Sons, 1998.

[49] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[50] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*, pp. 523–540, Springer, 2004.

[51] B. Cambou, C. Philabaum, D. Booher, and D. A. Telesca, "Response-based cryptographic methods with ternary physical unclonable functions," in *Future of Information and Communication Conference*, pp. 781–800, Springer, 2019.

[52] B. Cambou, C. Philabaum, D. Booher, and D. A. Telesca, "Response-based cryptographic methods with ternary physical unclonable functions," in *Advances in Information and Communication: Proceedings of the 2019 Future of Information and Communication Conference (FICC), Volume 2*, pp. 781–800, Springer, 2020.

[53] C. Philabaum, C. Coffey, B. Cambou, and M. Gowanlock, "A response-based cryptography engine in distributed-memory," in *Intelligent Computing: Proceedings of the 2021 Computing Conference, Volume 3*, pp. 904–922, Springer, 2021.

[54] K. Lee, M. Gowanlock, and B. Cambou, "Saber-gpu: A response-based cryptography algorithm for saber on the gpu," in *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 123–132, IEEE, 2021.

[55] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, pp. 9–14, IEEE, 2007.

[56] Y. Su, J. Holleman, and B. Otis, "A 1.6 pj/bit 96% stable chip-id generating circuit using process variations," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 406–611, IEEE, 2007.

[57] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic pufs from flip-flops on reconfigurable devices," in *3rd Benelux workshop on information and system security (WISSec 2008)*, vol. 17, p. 2008, 2008.

[58] D. Suzuki and K. Shimizu, "The glitch puf: A new delay-puf architecture exploiting glitch shapes," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 366–382, Springer, 2010.

[59] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2008.

[60] F. Wilde, "Large scale characterization of sram on infineon xmc microcontrollers as puf," in *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, pp. 13–18, 2017.

[61] D. Lorenz, G. Georgakos, and U. Schlichtmann, "Aging analysis of circuit timing considering nbti and hci," in *2009 15th IEEE International On-Line Testing Symposium*, pp. 3–8, IEEE, 2009.

[62] A. Roelke and M. R. Stan, "Attacking an sram-based puf through wearout," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 206–211, IEEE, 2016.

[63] A. Garg and T. T. Kim, "Design of sram puf with improved uniformity and reliability utilizing device aging effect," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1941–1944, IEEE, 2014.

[64] L. T. Clark, S. B. Medapuram, and D. K. Kadiyala, "Sram circuits for true random number generation using intrinsic bit instability," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 2027–2037, 2018.

[65] V. v. d. Leest, E. v. d. Sluis, G.-J. Schrijen, P. Tuyls, and H. Handschuh, "Efficient implementation of true random number generator based on sram pufs," in *Cryptography and security: from theory to applications*, pp. 300–318, Springer, 2012.

[66] D. Li, Z. Lu, X. Zou, and Z. Liu, "Pufkey: A high-security and high-throughput hardware true random number generator for sensor networks," *Sensors*, vol. 15, no. 10, pp. 26251–26266, 2015.

[67] S. Kiamehr, M. S. Golanbari, and M. B. Tahoori, "Leveraging aging effect to improve sram-based true random number generators," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 882–885, IEEE, 2017.

[68] R. Wang, G. Selimis, R. Maes, and S. Goossens, "Long-term continuous assessment of sram puf and source of random numbers," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 7–12, IEEE, 2020.

[69] M. T. Rahman, D. Forte, X. Wang, and M. Tehranipoor, "Enhancing noise sensitivity of embedded srams for robust true random number generation in socs," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pp. 1–6, IEEE, 2016.

[70] P.-S. Yeh, C.-A. Yang, Y.-H. Chang, Y.-D. Chih, C.-J. Lin, and Y.-C. King, "Self-convergent trimming sram true random number generation with in-cell storage," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 9, pp. 2614–2621, 2019.

[71] İ. E. Yüksel, A. Olgun, B. Salami, F. Bostancı, Y. C. Tuğrul, A. G. Yağlıkçı, N. M. Ghiasi, O. Mutlu, and O. Ergin, "Turan: True random number generation using supply voltage underscaling in srams," *arXiv preprint arXiv:2211.10894*, 2022.

[72] M. Julliere, "Tunneling between ferromagnetic films," *Physics letters A*, vol. 54, no. 3, pp. 225–226, 1975.

[73] D. Apalkov, B. Dieny, and J. M. Slaughter, "Magnetoresistive random access memory," *Proceedings of the IEEE*, vol. 104, no. 10, pp. 1796–1830, 2016.

[74] E. I. Vatajelu, G. D. Natale, M. Barbareschi, L. Torres, M. Indaco, and P. Prinetto, "Stt-mram-based puf architecture exploiting magnetic tunnel junction fabrication-induced variability," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 1, pp. 1–21, 2016.

[75] L. Savtchenko, B. N. Engel, N. D. Rizzo, M. F. Deherrera, and J. A. Janesky, "Method of writing to scalable magnetoresistance random access memory element," Apr. 8 2003. US Patent 6,545,906.

[76] J. C. Slonczewski, "Current-driven excitation of magnetic multilayers," *Journal of Magnetism and Magnetic Materials*, vol. 159, no. 1-2, pp. L1–L7, 1996.

[77] M. Wang, W. Cai, K. Cao, J. Zhou, J. Wrona, S. Peng, H. Yang, J. Wei, W. Kang, Y. Zhang, *et al.*, "Current-induced magnetization switching in atom-thick tungsten engineered perpendicular magnetic tunnel junctions with large tunnel magnetoresistance," *Nature communications*, vol. 9, no. 1, p. 671, 2018.

[78] M. Wang, W. Cai, D. Zhu, Z. Wang, J. Kan, Z. Zhao, K. Cao, Z. Wang, Y. Zhang, T. Zhang, *et al.*, "Field-free switching of a perpendicular magnetic tunnel junction through the interplay of spin–orbit and spin-transfer torques," *Nature electronics*, vol. 1, no. 11, pp. 582–588, 2018.

[79] L. Zhang, X. Fong, C.-H. Chang, Z. H. Kong, and K. Roy, "Highly reliable memory-based physical unclonable function using spin-transfer torque mram," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2169–2172, IEEE, 2014.

[80] S. Lim, B. Song, and S.-O. Jung, "Highly independent mtj-based puf system using diode-connected transistor and two-step postprocessing for improved response stability," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2798–2807, 2020.

[81] J. Das, K. Scott, S. Rajaram, D. Burgett, and S. Bhanja, "Mram puf: A novel geometry based magnetic puf with integrated cmos," *IEEE Transactions on Nanotechnology*, vol. 14, no. 3, pp. 436–443, 2015.

[82] E. I. Vatajelu, G. Di Natale, and P. Prinetto, "Security primitives (puf and trng) with stt-mram," in *2016 IEEE 34th VLSI Test Symposium (VTS)*, pp. 1–4, IEEE, 2016.

[83] E. I. Vatajelu, G. Di Natale, and P. Prinetto, "Stt-mtj-based trng with on-the-fly temperature/current variation compensation," in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 179–184, IEEE, 2016.

[84] E. I. Vatajelu and G. Di Natale, "High-entropy stt-mtj-based trng," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 491–495, 2019.

[85] K. Yang, Q. Dong, Z. Wang, Y.-C. Shih, Y.-D. Chih, J. Chang, D. Blaauw, and D. Svlvester, "A 28nm integrated true random number generator harvesting entropy from mram," in *2018 IEEE Symposium on VLSI Circuits*, pp. 171–172, IEEE, 2018.

[86] F. Ferdaus, B. B. Talukder, M. Sadi, and M. T. Rahman, "True random number generation using latency variations of commercial mram chips," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 510–515, IEEE, 2021.

[87] B. Perach *et al.*, "An asynchronous and low-power true random number generator using stt-mtj," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2473–2484, 2019.

[88] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[89] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[90] B. Mohammad, M. Abi Jaoude, V. Kumar, D. M. Al Homouz, H. A. Nahla, M. Al-Qutayri, and N. Christoforou, "State of the art of metal oxide memristor devices," *Nanotechnology Reviews*, vol. 5, no. 3, pp. 311–329, 2016.

[91] C. Bäumer and R. Dittmann, "Redox-based memristive metal-oxide devices," in *Metal Oxide-Based Thin Film Structures*, pp. 489–522, Elsevier, 2018.

[92] K. Skaja, M. Andrä, V. Rana, R. Waser, R. Dittmann, and C. Baeumer, "Reduction of the forming voltage through tailored oxygen non-stoichiometry in tantalum oxide reram devices," *Scientific reports*, vol. 8, no. 1, pp. 1–7, 2018.

[93] I. Gupta, A. Serb, A. Khiat, R. Zeitler, S. Vassanelli, and T. Prodromakis, "Sub 100 nw volatile nano-metal-oxide memristor as synaptic-like encoder of neuronal spikes," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 2, pp. 351–359, 2018.

[94] T. Shi and Q. Liu, "Characteristics and mechanisms in resistive random-access memory," in *Photo-Electroactive Nonvolatile Memories for Data Storage and Neuromorphic Computing*, pp. 13–52, Elsevier, 2020.

[95] K. Beckmann, H. Manem, and N. C. Cady, "Performance enhancement of a time-delay puf design by utilizing integrated nanoscale reram devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 3, pp. 304–316, 2016.

[96] Y. Pang, H. Wu, B. Gao, N. Deng, D. Wu, R. Liu, S. Yu, A. Chen, and H. Qian, "Optimization of rram-based physical unclonable function with a novel differential read-out method," *IEEE Electron Device Letters*, vol. 38, no. 2, pp. 168–171, 2017.

[97] B. Cambou and M. Orlowski, "Puf designed with resistive ram and ternary states," in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, pp. 1–8, 2016.

[98] T. Wilson, B. Cambou, B. M. Riggs, I. Burke, J. Heynessens, and S.-H. Jo, "Design and analysis of pre-formed reram-based puf," *Computing Conference 2022*, Jul 2022.

[99] S. Jain, T. Wilson, S. Assiri, and B. Cambou, "Bit error rate analysis of pre-formed reram-based puf," in *Science and Information Conference*, pp. 882–901, Springer, 2022.

[100] B. Cambou, D. Hély, and S. Assiri, "Cryptography with analog scheme using memristors," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 16, no. 4, pp. 1–30, 2020.

[101] J. Postel-Pellerin, H. Bazzi, H. Aziza, P. Canet, M. Moreau, V. Della Marca, and A. Harb, "True random number generation exploiting set voltage variability in resistive ram memory arrays," in *2019 19th Non-Volatile Memory Technology Symposium (NVMTS)*, pp. 1–5, IEEE, 2019.

[102] H. Bazzi, J. Postel-Pellerin, H. Aziza, M. Moreau, and A. Harb, "Resistive ram set and reset switching voltage evaluation as an entropy source for random number generation," in *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–4, IEEE, 2020.

[103] B. Peng, Q. Wu, Z. Wang, and J. Yang, "A rram-based true random number generator with 2t1r architecture for hardware security applications," *Micromachines*, vol. 14, no. 6, p. 1213, 2023.

[104] B. Cambou, D. Telesca, S. Assiri, M. Garrett, S. Jain, and M. Partridge, "Trngs from pre-formed reram arrays," *Cryptography*, vol. 5, no. 1, p. 8, 2021.

[105] Y. Zhu, B. Cambou, D. Hely, and S. Assiri, "Extended protocol using keyless encryption based on memristors," in *Science and Information Conference*, pp. 494–510, Springer, 2020.

[106] A. R. Korenda, S. Assiri, F. Afghah, and B. Cambou, "An error correction approach to memristors puf-based key encapsulation," in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pp. 1–6, IEEE, 2021.

[107] B. Habib, B. Cambou, D. Booher, and C. Philabaum, "Public key exchange scheme that is addressable (pka)," in *2017 IEEE Conference on Communications and Network Security (CNS)*, pp. 392–393, IEEE, 2017.

[108] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber algorithm specifications and supporting documentation," *NIST PQC Round*, vol. 2, no. 4, pp. 1–43, 2019.

[109] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai, "Crystals-dilithium," *Algorithm Specifications and Supporting Documentation*, 2020.

[110] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon," *Post-Quantum Cryptography Project of NIST*, 2020.

[111] J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, *et al.*, "Sphincs," 2019.

[112] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International algorithmic number theory symposium*, pp. 267–288, Springer, 1998.

[113] P. Gutmann, "Data remanence in semiconductor devices.," in *USENIX Security Symposium*, pp. 39–54, 2001.

[114] I. Goldberg and D. Wagner, "Randomness and the netscape browser," *Dr Dobb's Journal-Software Tools for the Professional Programmer*, vol. 21, no. 1, pp. 66–71, 1996.

[115] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[116] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[117] B. Cambou, "A xor data compiler combined with physical unclonable function for true random number generation," *SAI Computing Conference 2017*, 2017.

[118] B. Cambou, "Design of true random numbers generators with ternary physical unclonable functions," *Advances in Science, Technology, and Engineering Systems Journal*, vol. 3, no. 3, 2018.

[119] M. Krarti, *Optimal design and retrofit of energy efficient buildings, communities, and urban centers.* Butterworth-Heinemann, 2018.

[120] B. F. Cambou, "Design of true random numbers generators with ternary physical unclonable functions," *Advances in Science, Technology and Engineering Systems Journal*, vol. 3, pp. 15–29, 2018.

[121] N. Sklavos, P. Kitsos, K. Papadomanolakis, and O. Koufopavlou, "Random number generator architecture and vlsi implementation," in *2002 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, pp. IV–IV, IEEE, 2002.

[122] K. U. Maheswari, R. Kundu, and H. Saxena, "Pseudo random number generators algorithms and applications," *Int. J. Pure Appl. Math*, vol. 118, no. 22, pp. 331–336, 2018.

[123] M. G. V. Kumar and U. Ragupathy, "A survey on current key issues and status in cryptography," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 205–210, IEEE, 2016.

[124] D. G. Miandoab, S. Assiri, J. Mihaljevic, and B. Cambou, "Statistical analysis of reram-puf based keyless encryption protocol against frequency analysis attack," 2021.

[125] J. Reynolds, T. Smith, K. Reese, L. Dickinson, S. Ruoti, and K. Seamons, "A tale of two studies: The best and worst of yubikey usability," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 872–888, IEEE, 2018.

[126] D. Oswald, B. Richter, and C. Paar, "Side-channel attacks on the yubikey 2 one-time password generator," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 204–222, Springer, 2013.