Spring 2021

Telehealth Sensor Authentication Through Memory Chip Variability

Calvin Kimbro

Holden Gordon

Thomas Lyp

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Calvin Kimbro, Holden Gordon, Thomas Lyp

ENTITLED

TELEHEALTH AUTHENTICATION THROUGH MEMORY CHIP VARIABILITY

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING

Fatemeh Tehranipoor Sara Tehranipoor 06/04/2021

Thesis Advisor(s) (use separate line for each advisor) date

Shoba Krishnan 6/9/2021

Department Chair(s) (use separate line for each chair) date

TELEHEALTH SENSOR AUTHENTICATION THROUGH MEMORY CHIP VARIABILITY

By

Calvin Kimbro, Holden Gordon, Thomas Lyp

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical Engineering

Santa Clara, California

Spring 2021

Telehealth Sensor Authentication Through Memory Chip Variability

Calvin Kimbro, Holden Gordon, Thomas Lyp

Department of Electrical Engineering
Santa Clara University
2021

ABSTRACT

In light of the COVID-19 world-wide pandemic, the need for secure and readily available remote patient monitoring has never been more important. Rural and low income communities in particular have been severely impacted by the lack of accessibility to in-person healthcare. This has created the need for access to remote patient monitoring and virtual health visits in order for greater accessibility to premier care. In this paper, we propose hardware security primitives as a viable solution to meet the security challenges of the telehealth market. We have created a novel solution, called the High-Low (HiLo) method, that generates physical unclonable function (PUF) signatures based on process variation within flash memory in order to uniquely identify and authenticate remote sensors. The HiLo method consumes 20x less power than conventional authentication schemes, has an average latency of only 39ms for signature generation, and can be readily implemented through firmware on ONFI 2.2 compliant off-the-shelf NAND flash memory chips. The HiLo method generates 512 bit signatures with an average error rate of $5.9 * 10^{-4}$, while also adapting for flash chip aging. Due to its low latency, low error rate, and high power efficiency, we believe that the HiLo method could help progress the field of remote patient monitoring by accurately and efficiently authenticating remote health sensors.

Keywords: Process Variation, Hardware Security, PUF, Sensor Authentication

ACKNOWLEDGMENTS

We would like to express our appreciation for the support and guidance we received from our advisor, Dr. Sara Tehranipoor. Even with the many challenges associated with working on a remote senior design project during a world-wide pandemic, she helped push us to complete our goals.

We are grateful to the Santa Clara University School of Engineering for funding our project and providing a platform for us to share our results. Research is only valuable if you can present and share your new ideas and findings with others. Hopefully this research can spark the creativity of future engineers to pursue similar projects that aim at creating a new wave of healthcare that is more accessible, affordable, and secure for all people.

Finally, we would like to thank the Electrical Engineering Department of Santa Clara University. Without the countless hours put in by faculty and staff to help give us the tools to explore new and fascinating ideas within electrical engineering, we would be unable to push the field of hardware security and data privacy further.

TABLE OF CONTENTS

Page

Abstract..... iii

Chapter 1 - Introduction 1

1.1 Telehealth 1

1.2 Project Goals 2

1.3 Contributions 2

Chapter 2 - Preliminary Information 4

2.1 Wireless Authentication Methods 4

2.2 MLC NAND Flash Memory Architecture 4

2.3 Physical Unclonable Function (PUF) 5

2.4 Process Variations and PUFs for Flash Memory.....6

2.5 Prior State of the Art Flash PUFs 9

Chapter 3 - Project Plan and Methodology.....11

3.1 Project Plan.....11

3.2 Methodology.....11

3.2.1 Tools Used.....11

Chapter 4 - HiLo Flash PUF Extraction Technique.....13

4.1 Design Considerations.....13

4.2 Experimental Setup.....14

4.3 HiLo Extraction Method.....15

4.3.1 - PUF Extraction Observations and Techniques.....15

4.3.2 - Enrollment Scheme.....17

4.3.3 - Challenge and Response Pairs.....20

Chapter 5 - Experimental Results and Validations.....22

5.1 PUF Metrics.....22

5.1.1 Uniqueness, randomness, reliability.....22

5.1.2 Aging Adaptations.....23

5.2 Comparison to Current Solutions.....24

Chapter 6 - Telehealth Application.....25

6.1 Possible Privacy Threats.....25

6.2 Proposed Secure Handshaking with HiLo Method.....25

Chapter 7 - Conclusion……....29
7.1 Summary……..29
7.2 Future Work…….29
7.3 Project Challenges and Lessons Learned……....30
7.4 Final Remarks…….32
Bibliography……..33
Appendix A: Project Schedule……..36
Appendix B: Project Firmware Information…….38

LIST of FIGURES

Page

Figure 2.1 A-B - Flash Memory Architecture ……....5
Figure 2.2 - Flash Memory Read Disturb …….……..7
Figure 2.3 - Flash Memory Program Disturb ……...…....8
Figure 2.4 - Flash Memory Program Interrupt……..……...8
Figure 4.3.1: Ratio of 1's and 0's vs. Clocking Granularity ……16
Figure 4.3.2: Error Increase as Number of PECs Increases …….…….17
Figure 4.3.3: Bit Stability and Read Disturbs ……...…18
Figure 4.3.4: Histogram of Distance Between Sequential Failures ……..18
Figure 4.3.5: Byte Stability After Enrollment ……..19
Figure 5.2.1: Inter Hamming Distance Across Page Signatures ……..…22
Figure 5.2.2: Error Rate Over Time With Adaptive Aging ……..…24
Figure 6.2.1: Authentication Protocol …….….27
Figure 6.2.2: Challenge and Response Pairs ……....27
Figure A1: Project Timeline and Completion……...37
Figure B1: Experimental Setup with Firmware Interface…….38

CHAPTER 1 - INTRODUCTION

Section 1.1 - What is Telehealth?

Telecommunication in healthcare, or telehealth, provides a means by which patients can interact with medical professionals and health-related services virtually. It serves as a resource for patients to help them manage their health and receive the healthcare that they need. One service of telehealth is remote patient monitoring, which allows a healthcare provider to track patient data such as body temperature, heart rate, and blood glucose levels. This technology helps patients stay informed about the status of their health with the added convenience of being remote, and gives physicians a pow
.erful tool to help them diagnose, treat, and manage their patients' conditions.

The demand for telehealth has grown substantially in light of the COVID-19 pandemic, which has disrupted the delivery of healthcare on a global scale. In 2020 alone, the telehealth market experienced a year-over-year increase of 64.3 percent, and is estimated to grow seven-fold by 2025 [11]. However, the convenience of connecting medical providers with patients remotely also introduces significant security and privacy risks. One such risk of using unsecured medical devices is the potential for a major privacy breach, as they store sensitive information such as vital signals, diagnosed conditions, therapies, and a variety of other personal data [3].

Developing a reliable process for authenticating health sensors will ensure that private health information is only sent when authorized devices request it. Security vulnerabilities could expose private and sensitive information of patients to attackers, and it could be used to access critical backend systems of a supplier or a healthcare provider. This protection of user data allows for the proper expanse of telehealth systems without compromising patient data and guaranteeing device security.

Section 1.2 - Project Goals

At the start of this project, we set out to create a faster, more secure, and more efficient form of telehealth authentication. In order to accomplish this hefty project, there were many intermediary goals that needed to be addressed throughout the year. The main objectives of this project include:

- Developing MLC Flash NAND Chip memory controller software for ONFI 2.2 compatible chipsets.
- Develop a PUF extraction method that can be used to generate secure keys using process variations found on flash memory chips.
- Verify the extracted PUF's integrity through statistical analysis and reliability testing.
- Propose a novel secure handshaking protocol for health sensors that implements the proposed PUF extraction method.

Refer to **Appendix A** for more information regarding the project's schedule and deadlines pertaining to each of these project objectives.

Section 1.3 - Contributions

At the completion of our project, we have completed all proposed project goals listed in **section 1.2**. In summary, we have contributed the following:

- We have built a novel PUF extraction technique, called the HiLo method, which supports edge deployment on low cost microcontrollers. This will lower the cost of entry, and encourage secure data transmission for remote devices.
- The proposed HiLo method offers a PUF solution for accurate authentication, and has lower latency and lower power consumption than other PUF generation techniques.
- The HiLo method is compatible with off-the-shelf ONFI 2.2 compliant flash chips, which could lead to backwards compatibility implementation

on existing health sensors.

The following section will explore the preliminary information required in order to fully understand our design implementation.

CHAPTER 2 - PRELIMINARY INFORMATION

Section 2.1 - Wireless Authentication Methods

There are a variety of wireless authentication methods that are used by IoT and Telehealth devices. One of the most common ways to authenticate these resource constrained devices is by utilizing pre-shared keys. Pre-shared keys are authentication keys or tokens that are shared with a device prior to its deployment. These keys are then exchanged with a gateway in order to authenticate an IoT device. There are several important vulnerabilities within this model that our solution addresses. First, the keys can be extracted out of firmware by a sophisticated hacker who has access to the physical device. This has happened in the industry with Link Plugs and other smaller IoT devices [27]. Secondly, these keys can be cloned through replay attacks and deauthentication attacks [7]. This was shown to be effective on all WEP Wifi routers which extracted wifi keys due to the lack of 'freshness' in the messages sent between endpoints and Wifi routers.

By using PUFs and TRNGs, many of these prior security vulnerabilities can be drastically mitigated by providing random nonces and authentication signatures. Firstly, the PUFs allow secrets to be embedded in process variations which are not stored in any nonvolatile memory; this prevents hackers from simply dumping onboard firmware and finding pre-shared keys. Secondly, the onboard TRNG mitigates replay attacks by preventing attackers from arbitrarily replaying authentication messages.

Section 2.2 - MLC NAND Flash Memory Architecture

Flash NAND memory is a type of non-volatile memory that stores user data in the physical form of charge on a float gate. Programming these cells requires

electrons to move from the polysilicon channel on/off the float gate via electron tunnelling, described in **Figure 2.1A**. It is important to note that this electron tunneling can damage the tunnel oxide, which means that flash cells become less reliable after many program/erase cycles (PECs). Most flash memory is rated anywhere from 1,000 to 10,000 PECs. A flash's lifetime can increase drastically by using a memory management controller that distributes PECs evenly across all cells called wear leveling.

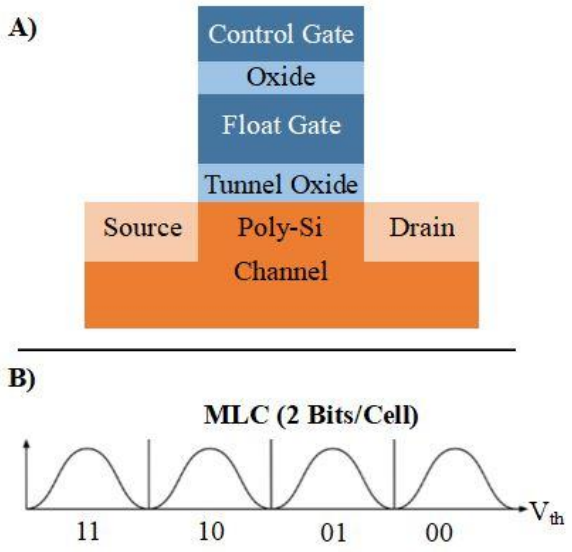

Figure 2.1 A-B: Flash Memory Architecture

In the case of Multi-Level Cell (MLC) flash chips, 2 bits of data are stored on each cell, where the current float gate voltage is compared to multiple threshold voltages in order to determine the cell's digital value, as shown in **Figure 2.1B**. Thanks to its high memory density, low cost, and ability to be programmed and erased electronically without moving parts, flash NAND memory has exploded in popularity for remote devices and IoT systems [6].

Section 2.3 - Physical Unclonable Functions (PUFs)

Integrated chips have random sub-micron process variations that are caused from the natural manufacturing process. These variations include slight differences in

transistor sizes, resistors, and other onboard components in an integrated chip. These variations must be accounted for and mitigated in the design process; however, researchers have proposed using these variations to produce bit strings that can uniquely identify chips based on the uniqueness of their process variations. This effectively creates a silicon "fingerprint" that depends on submicron variations that are extremely difficult to model. The first silicon PUFs were introduced from Pappu et al. [13,14] and Gassend et al. [1] in 2002. There are two imperative characteristics necessary for a PUF to function. A PUF must have both random and uncontrollable variations. This guarantees that a third party cannot clone or predict a PUF's functionality. Furthermore, PUF's must have satisfactory uniqueness, reliability, and randomness. Uniqueness guarantees that each response from a PUF system is unique from another. Reliability emphasizes that a PUF's response must be consistent for a given challenge. Finally, the randomness highlights the necessary randomness between response bits.

There are many different types of PUFs such as Arbiter PUFs (APUFs), Static Random Memory PUFs (SRAM PUFs), XOR Arbiter PUFs (XAPUFs), Dynamic Random Access Memory PUFs (DRAM PUFs), and Flash Memory PUFs (FPUFs). Regardless of the type of PUF, PUFs can be utilized in a variety of cryptographic applications, as evidenced by their use in authentication schemes [22], software attestation [8], IC counterfeiting [15], and cryptographic key generation [2]. PUFs provide a unique advantage of tying cryptographic security primitives to the process variations that significantly reduces both performance and hardware costs while also providing strong cryptographic security.

Section 2.4 - Process Variations and PUFs for Flash Memory

Like all other silicon based ICs, flash memory chips are subject to many different forms of process variations. In general, most process variations are uncontrollable imperfections caused by limitations in modern lithography processes. These variations are commonly caused by various disturbs from Random Dopant Fluctuations (RDP). RDP fluctuations can be temporarily induced through

standard programming calls that comply with the ONFI NAND Flash interface allowing them to be extractable on any ONFI compliant memory chip. This gives a significant advantage compared to other memory architectures where PUF cells cannot be extracted through a standardized process using accessible programming techniques.

Several different disturbs can be implemented through these standard programming techniques such as Read disturb, program disturb, program/erase latency, and Random Telegraph Noise (RTN). Read disturb causes subtle increases on the threshold voltages of neighboring cells on the same flash page. Over time, this slowly changes the logical bit values of the cells to induce errors that depend on each cell's relative oxide and gate thickness.

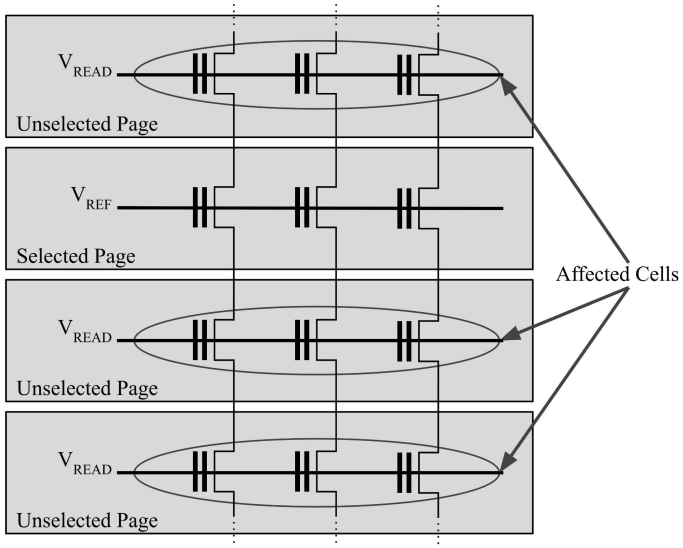

Figure 2.2: Flash Memory Read Disturb

A similar error is the program disturb. The program disturb occurs due to the increase of the gating voltage required for a program operation [6]. Since the programming requires a higher voltage on the gate line, it introduces parasitic capacitance that causes a shift in the threshold voltage of neighboring cells. This is highlighted in **Figure 2.3**.

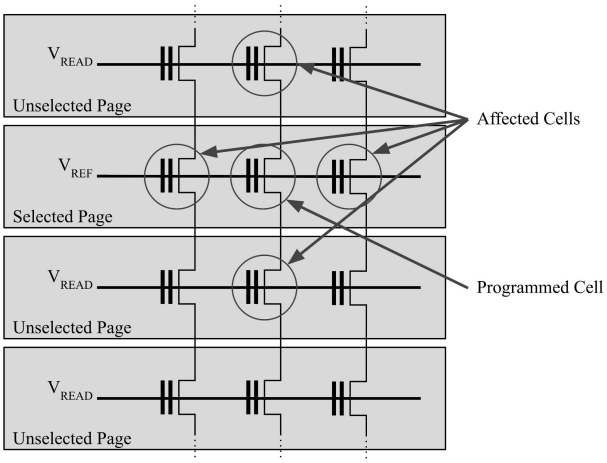

Figure 2.3: Flash Memory Program Disturb

The next technique for extracting process variations is the program/erase interrupt scheme. In this scheme A program or erase command is issued to a page or block. This command is then abruptly interrupted leaving the cells in an unsteady and ambiguous state. Therefore, minimal differences in the oxide thickness of individual gates causes different read values in the cells. This technique is very effective at generating PUF signatures; however, it can be susceptible to various sources of noise.

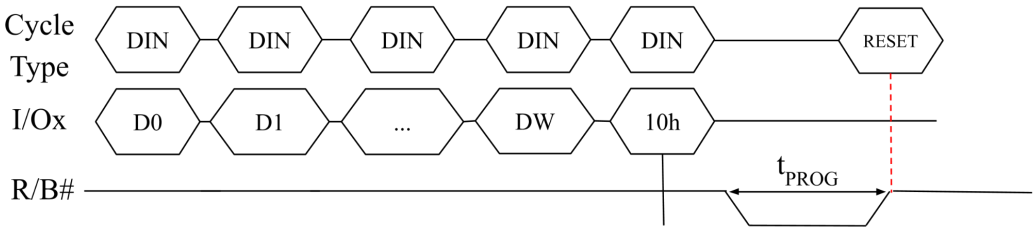

Figure 2.4: Flash Memory Program Interrupt

Finally, one other technique can be used for process variation extraction in Flash PUFs. The latency of program/erase operations can be digitized and also used for PUFs. However, this technique is drastically limited in its throughput.

Ultimately each technique causes fluctuations that depend on each cell's relative gate thickness and width. both of which are uncontrollable and extremely hard to model. Finally, this allows for unique signatures that can be generated for each

page of the flash memory. These PUFs can be used for secure key generation and authentication mechanisms [26]. Each PUF has a challenge and response. The challenge is the input to the unclonable function which then outputs a response. This response has high security and differs from each subsequent response and challenge pair.

Section 2.5 - Prior State of the Art Flash PUFs

When looking at the development of different Flash PUFs there are several important trends to recognize. The first Flash PUFs were created from 2012-2017 [17, 22, 9, 18]. These seminal works were predominantly focused on showing how Flash memory was a viable candidate for memory-based PUFs. These Flash PUFs were important for showing the promise of flash memory, but they struggled to account for several factors such as aging, temperature, and design constraints. For example, Prabhu et al. [17] required hundreds of thousands of programs in order for PUF signatures to develop which can take several hours. Similarly, Kim et al. [10] required very fine grained flash programming which required knowledge of sensing voltages that is typically proprietary on commercial flash chips. Similarly, Wang et al. [22] created both a novel PUF and TRNG but incurred very high processing overheads that led to high latency.

From 2018 to 2020, a second phase of Flash PUF development began marked by the work of [7, 12, 15, 16, 19, 20, 24]. Many of these proposed designs significantly enhanced PUF designs that took into account many different factors. These Flash PUFs incorporate advanced parameters such as aging resistance, temperature resistance, and unique architectures. For example, work from Clark et. al [7] designed a Flash PUF that was voltage resistant. Secondly, Poudel et. al [15,16] designed a Flash PUF that works on the onboard microcontroller NOR Flash Memory. Similarly, Larimian et. al [20] verified the machine learning resilience of their Flash based PUFs by performing extensive deep learning tests. Finally, Mahmoodi et. al [12] created one of the most stable and resilient Flash PUFs by modifying the cells to extract leakage current.

However, this second phase of development also has several design difficulties. In order to make PUFs commercially viable, they must be generated using cheap commercial microcontrollers in order to keep their price point down. Furthermore, it is helpful that these PUFs can be deployed on legacy systems by using commercial off the shelf flash memory and these systems must avoid excessively long latency that have been seen in works such as [20]. Many of the previous work from the second phase does not use commercial off the shelf components. This can reduce the adoption of Flash PUF technology. Secondly, many of these Flash PUFs require slowly building charge differences on the floating gate through hundreds or thousands of program/erase cycles. This is effective at generating signatures; however, this drastically increases latency required to generate these signatures. Finally, many of the systems that avoid the slow build up of charge have to use expensive FPGAs with Gigahertz clocking speeds to generate fine grained interrupts as shown in Clark [7]. This is not viable for edge deployments particularly those such as telehealth based applications.

Our work, seeks to bridge this gap by developing a Flash based PUF that uses only ONFI standard commands, works on a 15 dollar microcontroller, uses off the shelf commodity flash chips, generates signatures with low latency using a novel interrupt technique, and is able to compete with some of the most resilient PUF structures that were highlighted from the second phase of development.

CHAPTER 3 - PROJECT PLAN AND METHODOLOGY

Section 3.1 - Project Plan

1. Fall Quarter
    - Compiled research on PUFs and looked at potential Telehealth Applications
    - Developed a memory controller software to read and write to the MLC Flash NAND Chip
    - Created our telehealth application using a temperature sensor and a Raspberry Pi
2. Winter Quarter
    - Finalized a PUF extraction method that can be used to generate secure keys
    - Built a TCP/IP handshaking protocol to connect and send data between two Raspberry Pis on different networks
2. Spring Quarter
    - Verified the PUF's reliability and randomness with statistical NIST testing
    - Implemented dynamic authentication protocol utilizing challenges and responses from the PUF

Section 3.2 - Methodology

3.2.1 Tools Used

Many of the decisions regarding tools and methodology were direct consequences of the remote nature of this year's projects. With no guaranteed time to collaborate in person or share experimental setups, it was vital to create an easily accessible project that could be changed by all team members on the fly. The tools used for the project include:

Google Suite - All written documentation and results were stored using Google Suite Applications including docs and sheets. This allowed us to

actively collaborate without the need of in-person meetings.

MBed - MBed was the main resource for all project firmware development. The resource allowed us to build useful firmware that was updated in real time through code pushes. It also allowed us to fork projects and experiment with the setup individually without excessive version control. More information regarding project firmware can be found in **Appendix B.**

Jupyter Notebook - The final resource required was a program to effectively process and visualize the data being collected. We chose Python-based analysis using Jupyter Notebook in order to analyze and process complex and large data sets.

While many different tools and programs could yield similar results, we believe that our tools and methodology were effective and useful for a remote work environment.

CHAPTER 4 - HILO FLASH PUF EXTRACTION TECHNIQUE

Section 4.1 - Design Considerations

With the experimental setup described in the previous section, work started on the PUF extraction method itself. When designing the PUF extraction method, there were two major metrics that were focused on extensively. The first feature was minimizing the number of program/erase cycles (PECs) required in order to extract a reliable signature. As mentioned in **Section 2.2**, flash memory devices have a limited lifetime measured in PECs. As charge is tunneled onto and off of the float gate, the tunnel oxide that separates the silicon channel and the float gate begins to deteriorate. As the deterioration continues, the data held within each cell becomes unreliable due to the increase of charge leakage on the float gate. Wear on the cells is an issue for data retention and general application use, but it also poses an issue with unreliable signature extraction, because the programming and erase times for each cell are altered as the cell reaches its end of life. In order to combat this aging effect, HiLo was designed in order to require as few PECs as possible.

The second major consideration was lowering the signature extraction latency. Faster signature extractions allow for faster authentication and lower power consumption. Telehealth sensors in the scope of this work can be treated as edge devices that are resource constrained by both processor speed as well as limited battery life. In order to prolong the sensor's battery lifetime as well as create a reasonably fast authenticated connection, the method needs to be lightweight and fast.

While additional results and metrics will be discussed in detail in **Section 5.1**, these two design considerations helped guide many of the design decisions for the HiLo method. In the next section the HiLo technique will be explained in detail,

and additional design constraints and considerations will be discussed.

Section 4.2 - Experimental Setup

As mentioned in our summarization of contributions section, it was vital to keep the experimental setup as minimal as possible in terms of both cost and complexity. While previous works have mostly required high clock frequencies and custom chips, our novel design uses a 100 MHz clock and a simple TSOP Adapter in order to connect to the flash chip. Our simple bill of materials consists of:

- STM32 based microcontroller with 100 MHz maximum clocking frequency. This will serve as the memory controller and is used to read/write/erase the flash memory.
- 32 GB MLC Flash NAND memory chip from Micron that does not have an integrated memory controller. The memory chip can read/store application data and is also used in the HiLo extraction method
- TSOP Adapter in order to interact with packaged flash NAND chip

The entire experimental setup was purchased for just shy of $20, and all items were readily available for purchase to be included in commercial application. It is worth noting that we used an unmanaged flash chip in order to give us full control of the device. This extra control allowed us to write directly to specific memory locations without worrying about wear-leveling control or error correcting code (ECC), which are implemented in most managed flash components. This gave us the highest control on interrupting the writing, erasing, and reading processes, because we were directly controlling the chip's behavior without interference from the memory controller. While a commercial application will require ECC and wear leveling in its memory controller software, this setup shows that the HiLo extraction method could be readily added to current memory controller solutions through a minor software modification.

While unmanaged flash gives extra control over the memory's behavior, it also requires a more detailed communication protocol. Most commercial managed

flash uses SPI or I2C protocols in order to access basic functionalities. On the other hand, unmanaged flash requires a version of *open NAND flash interface protocol (ONFI)*. Specifically, HiLo was tested using ONFI version 2.2, which requires 7 digital control signals and an 8 bit bus for reading and writing data. The HiLo method only requires the most basic memory access and modification functions including read page, write page, and erase block. The interrupt sequence and reading of data is ubiquitous across all ONFI 2.2 memory chips. The flash chip tested in this work stores pages consisting of 4096 bytes of data and 224 extra bytes for ECC. Blocks are organized in chunks of 256 pages, with each 32 Gb chip having access to 4096 unique blocks. In order to read and verify data, flash memory values were sent from the microcontroller to a terminal window on a laptop via UART communication using a USB connection.This was also the method used for storing and analysing large sample sizes of data in .csv formats.

Section 4.3 - HiLo Extraction Method

4.3.1 - PUF Extraction Observations and Techniques

For our PUF to be viable, the PUF extraction must have low latency and can only use standard edge deployable microcontrollers within the 100MHz frequency range. This introduces several important design challenges in order to make Flash PUFs achievable on low cost microcontrollers. First the low latency requirement prevents the design from utilizing hundreds of repeated program and read cycles that slowly increase the charge on the floating. Therefore, our scheme must use fine grained interrupts in order to interrupt the operation of the cells to force the flash into unsteady positions. However, a single interrupt scheme such as the one seen in Clark et. al [7] does not have a high enough clock resolution to interrupt the flash programming fast enough to generate a 50/50 split between ones and zeros. In fact, signatures that are generated from a single programmed interrupt are either about 80% 1's or 80% 0's. This is highlighted in **Figure 4.3.1** where the sysTick clock on the microcontroller is interrupted at different times. As shown in the figure, the clock does not have the granularity to interrupt the programming

operation on the microcontroller to generate an even distribution of 1's and 0's.

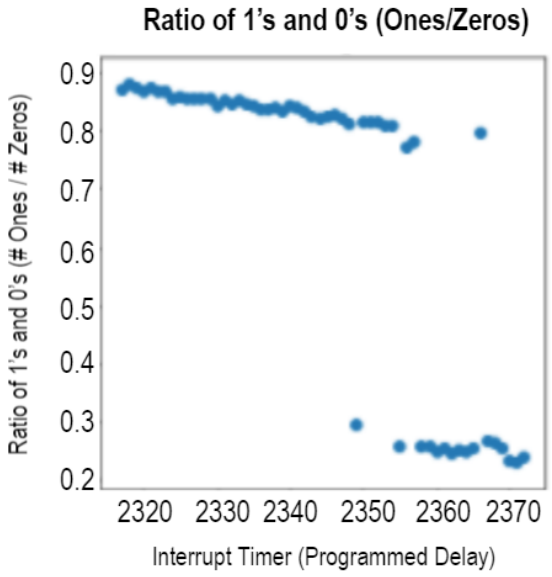

Figure 4.3.1: Ratio of 1's and 0's vs. Clocking Granularity

Secondly, the lack of granularity has another limiting effect as well. The signatures generated from a single interrupt are extremely noisy. This is due to the interrupting method. This error rate can approach 10% within 80 total P/E cycles. This is due to the lack of granularity in the clock itself. Due to the max 100 Mhz clocking speed, an interrupt at a particular clocking delay can vary due to the limited frequency. This causes unintended errors for cells that are sensitive to the interrupt. This drastic increase in error is highlighted in **Figure 4.3.2**.

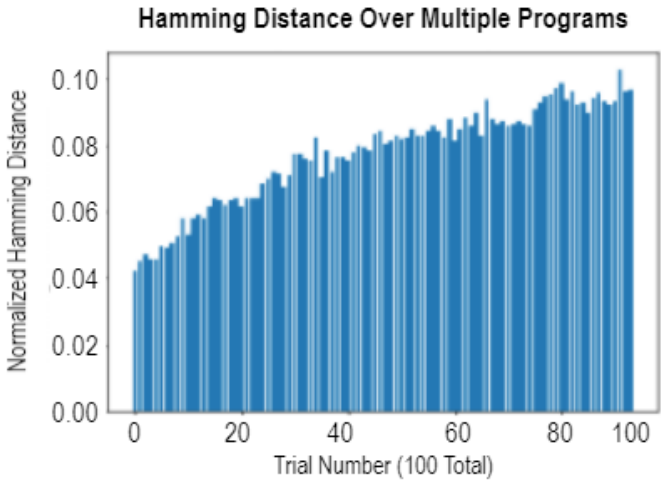

Figure 4.3.2: Error Increase as Number of PECs Increases

Although the error increases rapidly and the clocking speed cannot generate signatures with 50% ones and zeros, several more insights and solutions were generated to remedy these design challenges.

Specifically, a well-defined enrollment scheme is designed to select only the most stable cells which can be decoded as one or zero. These can then be used for highly stable signatures.

4.3.2 - Enrollment Scheme

As other work has shown in Poudel et. al [15, 16] unstable cells can be identified for TRNG bits by applying several reads. These reads apply a smaller voltage to the floating gate of the flash cells which only slightly disturbs the cells. This can quickly identify unstable cells that are flagged during enrollment. Furthermore, approximately 95% of these unstable bits flip within five reads. Therefore, only applying five reads is sufficient for identifying stable cells through successive read operations. **Figure 4.3.3** highlights this observation. This figure graphically shows all 32,000 bits on a single page. All of the yellow lines indicate a flipped bit. Many of these bits flip within the first five reads which is an effective filtering process for identifying stable bits.

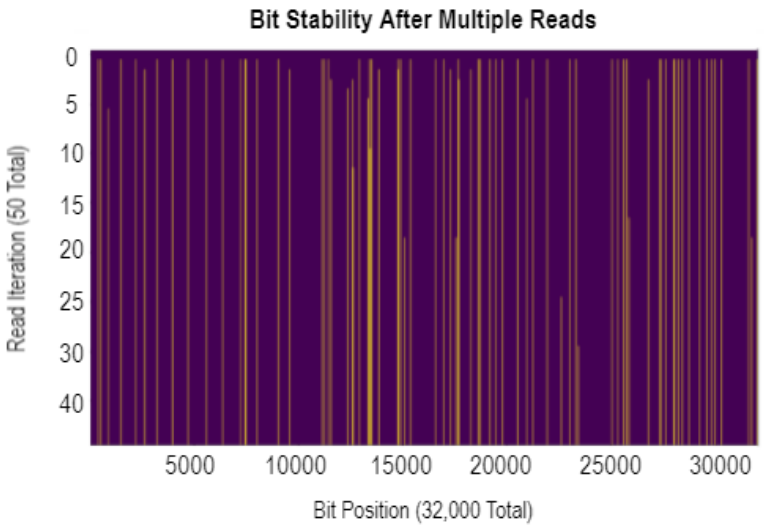

Figure 4.3.3: Bit Stability and Read Disturbs

Secondly, flash cell failure is highly spatially dependent. This is evidenced by plotting a histogram of the distance between each cell flip or failure. Failures tend to cluster in groups. This can be modeled as an exponential distribution, as shown in **Figure 4.3.4**. Therefore, instead of flagging individual bits that are stable, bytes of groups are flagged as stable and only if a byte is completely stable is it passed through the enrollment process.

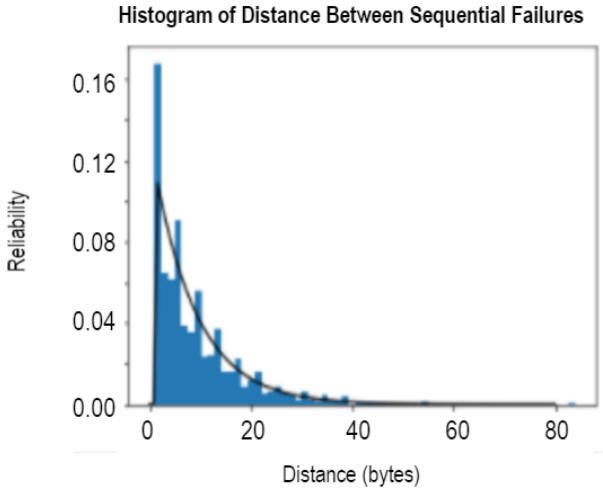

Figure 4.3.4: Histogram of Distance Between Sequential Failures

After this technique is applied about 1500 of the 4000 bytes are extracted per page. Approximately 800 of these extracted bytes are over 95% accurate during testing. However, approximately 700 bytes are misidentified as stable. This reliability histogram is shown in **Figure 4.3.5**.

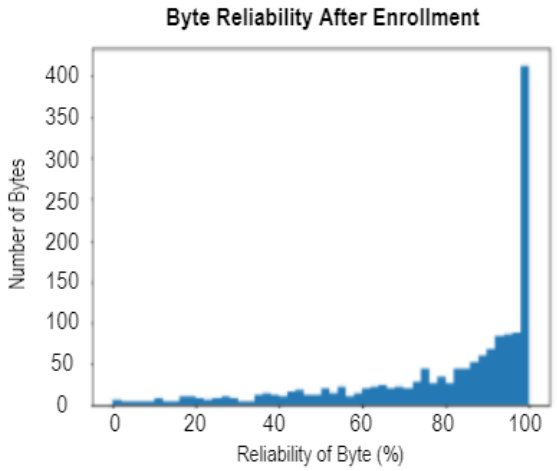

Figure 4.3.5: Byte Stability After Enrollment

However, a slight wrinkle to the first algorithm is able to generate highly stable PUF responses. This leads to our new enrollment strategy -- the HiLo method.

The HiLo extraction method uses a novel technique to generate highly reliable signatures. Rather than aborting the program early and using signatures with 80% ones, two interrupted programs are performed. One generates a signature with 80% ones and the other with approximately 20% ones and five reads are performed after each program to eliminate non-stable bytes. In order to refine the byte selection process, bytes that are either highly resistant to programming or highly susceptible to over-programming are chosen. This information is captured by selecting bytes which fully resist programming in the under-programmed section also known as the high side and which bytes that are easily programmed in the over-programmed section also known as the low side. If a byte group is fully programmed in the over-programmed section and is fully under-programmed in the under-programmed section it is also removed during the enrollment process. This eliminates bytes that can sneak through the enrollment process and

reduces the error by several magnitudes of 10 to around 10^4.

4.3.3 - Challenge and Response Pairs

After the HiLo enrollment process is complete a 'map' of the stable bytes are stored on the gateway. Each byte in the map is either extremely susceptible to over-programming or highly resistant to it. With this information, the gateway will request particular byte locations from the sensor for authentication. Approximately 256 low byte locations and 256 high byte locations will be sent along with the specific page number that is used. This comprises approximately 600 bytes of space which can be sent in the payload of a single Ethernet frame to the sensor. The microcontroller will then apply two programs and ten total reads and identify which bytes are stable and which are not. If a byte location produces a majority of ones it is subsequently decoded as one and is assumed to be one of the high side bytes. Inversely, if a byte produces a majority of zeros then it is assumed to be from the low side and is decoded as zero. This extraction technique can resist a maximum of three errors before a byte is incorrectly decoded. Therefore, the gateway will receive a bit string of ones and zeros to the sensor.

This design allows for two important features for the gateway. Firstly, the gateway can control how long of an authentication response it needs. This can allow our application to adapt to various levels of required security. For example, certain cryptographic applications may only require 100 bit signatures. The gateway then only has to send 100 bytes to the sensor. On the other hand sensitive security tasks that may demand 512 bit signatures can also be accomplished. Secondly, many other approaches use helper data such as Hamming Codes, Fire Codes, or more recently Low Density Parity Codes, to recover any errors sent back from the device. These codes leak polarity information about the response values which allows for error correction. Our scheme leaves out any polarity data and simply sends bytes to read. This makes our helper data significantly more robust to side channel or modeling attacks since the helper data never reveals any polarity

information from the bytes it is requesting. However, it is important to note that this advantage is realized because the HiLo enrollment algorithm filters bytes from pages very aggressively. On average, each page returns approximately 700 usable bytes. Therefore, about 83% bytes are not usable.

CHAPTER 5 - EXPERIMENTAL RESULTS AND VALIDATION OF PUF

Section 5.1 - PUF Metrics

5.1.1 - Uniqueness, Randomness, and Reliability

There are three major metrics for any PUF. They are uniqueness, randomness, and reliability. Uniqueness defines how different each PUF response is from each other. This value is best captured through the Inter-Hamming Distance (Inter-HD). The percentage Inter-HD describes what percentage of bits flip between two different responses. An ideal value for this is approximately 50%. Our system had an average of 47% which is shown in **Figure 5.2.1**.

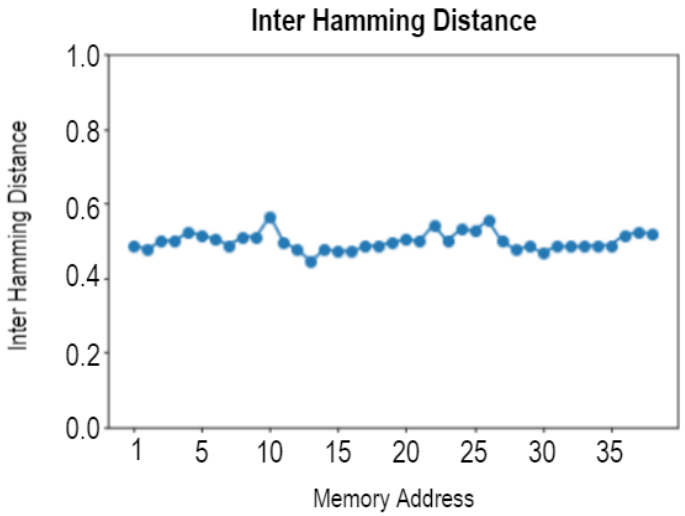

Figure 5.1.1: Inter Hamming Distance Across Page Signatures

The second metric randomness describes how random each signature is. This can be measured in Shannon entropy per bit. With 256 high bits and 256 low bits the average Shannon entropy per bit is approximately 0.999 with an ideal value of 1.

Next, the reliability is the final measurement. The reliability of this system is extremely strong. The error does change as the flash cells age; however, the error is approximately $5.9 * 10^{-4}$. The max error value is $7.1 * 10^{-3}$and the minimum value is $5.9 * 10^{-5}$. This reliability is strong enough to possibly support cryptographic key generation and is able to last through the end of the Flash's life.

5.1.2 - Aging Adaptation

In order to simulate aging, the Flash memory chips were programmed until their maximum rating of 3,000 P/E cycles for MLC chips. The aging causes the oxide to deteriorate from the program voltage stress. The low side bytes that are susceptible to over programming are barely modified since these bytes just program faster. However, the high side bytes that are resistant to programming program more easily which modifies the amount of bit flips in each byte. In general two approaches for this were considered. First was adapting the polling interval and decreasing it since the flash cells program faster. However, this approach can be difficult to model and control due to the lack of granularity in the interrupt mechanism. Consequently, the second approach was taken. In this approach the amount of bytes required for a decode was changed once the cells reached a particular percentage of life used. This adapts the error rat and drops it significantly. This is reflected in **Figure 5.2.2**.

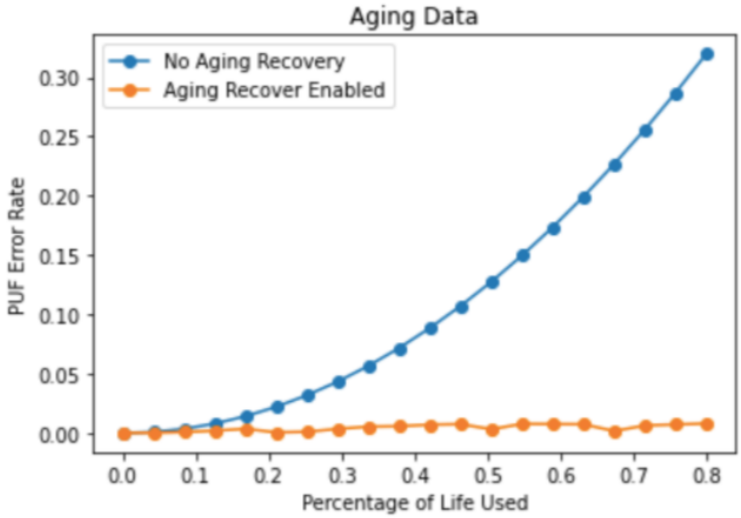

Figure 5.1.2: Error Rate Over Time With Adaptive Aging

Section 5.2 - Comparison to Current Solutions

In order to accurately evaluate the HiLo extraction method, we must also compare our results to current solutions in industry. Currently, Tiny AES is one of the most popular encryption schemes for resource constrained devices, and it sets a great benchmark for comparison. Tiny AES encryption uses a pre-shared key scheme that is similar to a PUF signature, which behaves like a physically stored key. The Tiny AES encryption algorithm requires 22.3 mW of power, and takes 190 ms. The HiLo extraction technique, on the other hand, takes less than a 1mW of power and PUF signatures are generated in 34.8 ms. This shows a significant performance advantage with a higher security guarantee since the PUF signatures are never stored in memory. Furthermore, this low latency and power consumption makes the HiLo PUF a strong candidate for the telehealth application space.

CHAPTER 6 - TELEHEALTH AUTHENTICATION

Section 6.1 - Possible Privacy Threats

Whenever a device sends data wirelessly, there is always the possibility of an unauthorized third party attempting to steal that information. This is especially true with healthcare data, which includes personal information, and can be used by malicious attackers for billing fraud, identity theft, and to illegally obtain prescription medicine. This is why according to a Trustwave report, a healthcare data record could sell for as much as 50 times more than a stolen credit card number on the black market [28].

The two threats we will target in our project are Man-in-the-Middle (MITM) attacks and Replay attacks. Both sets of attacks listen in on the communication between two hosts. With a Man in The Middle attack, the attacker intercepts the packets being sent en route, and they can read or modify unsecured information before it is sent off to the other host [21]. A replay attack has an attacker also read the packets being sent between the two hosts, but the attacker uses the authentication information to later attempt to pose as a legitimate host [21]. In an unsecured and static authentication scheme, the attacker could simply pick up a password being transmitted en route and use it later in a different transaction.

Section 6.2 - Proposed Secure Handshaking with HiLo Method

The first step in developing our authentication protocol was to build our telehealth application, which involved using a DS18B20 temperature sensor to collect temperature data and store it in a format which we can then use for edge

deployment. Body temperature collection is just one of the many different ways remote patient monitoring can be utilized. For our experiment, we used a Raspberry Pi as a means to collect temperature data in intervals of 10 seconds and saved the data in a .csv format. To do this, we wrote a Python script to automate the process.

We then built a TCP/IP dynamic challenge-response authentication scheme using the PUF. This process used two Raspberry Pis, which were setup using Sockets to communicate with each other across different networks, and the PUF extraction device (Microcontroller, TSOP connector, MLC Flash NAND). The Raspberry Pi that collected temperature data served as the "client", or the health sensor. Through USB, we connected the microcontroller to the "client" Raspberry Pi, and then successfully interfaced the two so that the Raspberry Pi could query the stable byte values of the NAND Flash chip. The other Raspberry Pi served as the "gateway", which would serve as an edge server in a real world application.

This authentication process starts with the gateway sending a challenge to the health sensor, denoted $Challenge_1$. This request includes a challenge location and stable byte locations of the NAND flash memory cell. The health sensor performs the interrupted program and extracts the stable byte values. The health sensor then randomly generates a nonce, denoted $Nonce_1$, and the generated hash value is XORed with $Nonce_1$. The value generated is sent to the gateway as a response to $Challenge_1$. The gateway knows what Response should be, so it XORs the entire response of the gateway with $Response_1$ to determine $Nonce_1$. The gateway then randomly generates another nonce, denoted $Nonce_2$. A second challenge, using a different index of stable byte locations, is generated by the gateway. $Challenge_2$ is then concatenated with ($Response_1$ XOR $Nonce_2$), and this message is sent back to the health sensor. The health sensor is able to separate $Challenge_2$ from ($Response_1$ XOR $Nonce_2$), and ($Response_1$ XOR $Nonce_2$) is XORed with $Response_1$ to determine $Nonce_2$. Both the gateway and the health sensor now know the values of $Challenge_1$, $Challenge_2$, $Nonce_1$, and $Nonce_2$. Using $Challenge_2$, the health sensor again performs the interrupted program and extracts the stable byte values. If the health sensor and the gateway are legitimate sources,

the challenges and responses can be properly decoded and the transaction is authenticated. Our proposed handshaking protocol is shown in **Figure 6.2.1**.

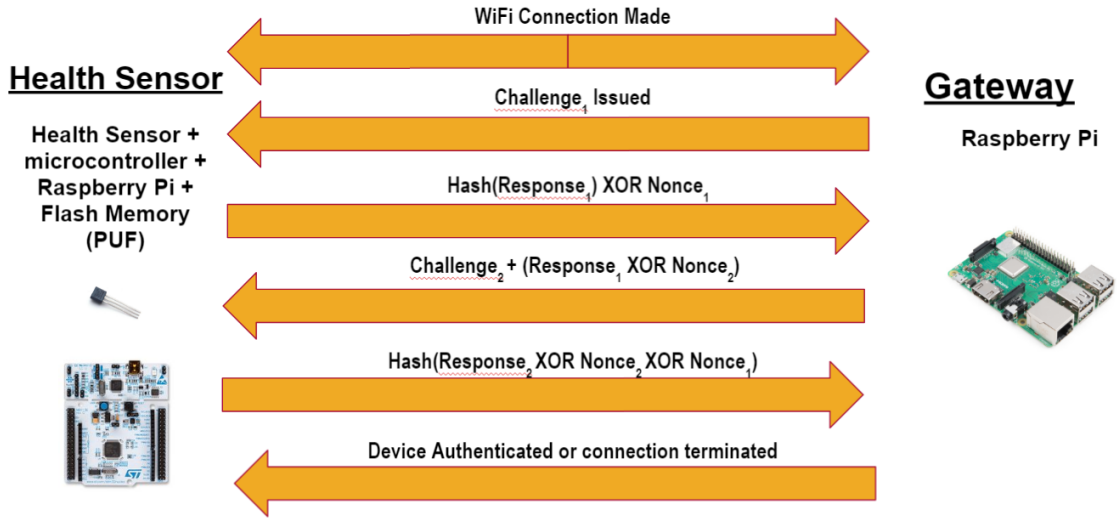

Figure 6.2.1: Authentication Protocol

There are many advantages to using this authentication scheme. First, the use of hashes masks the plaintext values of the data being sent, meaning that attackers won't be able to read the data in transit. Second, the use of randomly generated nonces means that the values being transmitted will always change with every transaction. Finally, and most importantly, the use of a PUF provides a unique identifier that the gateway can reliably authenticate, and attackers won't be able to model the authentication responses of the PUF.

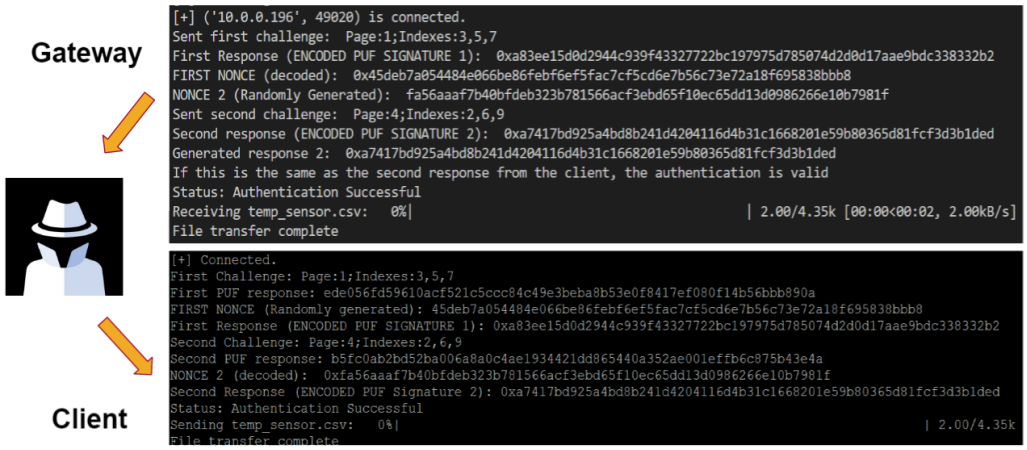

Figure 6.2.2: Challenge and Response Pairs

In **Figure 6.2.2**, we see the challenge and response pairs that we discussed being sent between the health sensor and the gateway. In the event of a man-in-the-middle or replay attack, which involves an attacker intercepting packets between the two parties, this is the information that they would receive. However, with regard to MITM, any modification of the packets being sent would result in a complete breakdown of the authentication process, ultimately making it obvious that the connection was tampered with, and would result in the gateway refusing the connection. With Replay attacks, the use of randomly generated nonces means the values being sent change with every transaction. This means that an attacker can't replay a message with an old pair of nonces, as the values would be entirely different to the current nonces.

CHAPTER 7 - CONCLUSION

Section 7.1 - Summary

As medical care continues to improve in both effectiveness and accessibility on a global scale, remote patient monitoring is a logical next step for healthcare investment. We have witnessed a global pandemic that disrupted the delivery of potentially life-saving medical care for over a year, which may have been partially mitigated by the adoption of remote patient monitoring (RPM) technology. While the technology proves useful for many medical monitoring applications, it is important to carefully consider the security vulnerabilities and possible exploits of large scale adoption of RPM. The HiLo extraction method is a lightweight, fast, and easily implementable security measure that could help ensure the authenticity of RPM sensor data. The HiLo method is designed in such a way that it could be implemented on new RPM sensors with minor software updates, and no additional components. The method is resource efficient, and offers financial benefits when compared with many other commercial PUF solutions.

Section 7.2 - Future Work

When considering future work for this project, more extensive testing needs to be done using additional flash chips from different manufacturers. Additionally, testing must be done in high temperature environments in order to verify that the reliability remains high regardless of external factors such as heat. Finally, the HiLo extraction method could be tested on other flash memory densities (SLC/TLC), or possibly even different flash architectures (3-D) in order to broaden the possible application space.

Section 7.3 - Project Challenges and Lessons Learned

As with any year long term project, there were many challenges along the way. The most obvious challenge was working remotely due to Covid-19. Without a common experimental setup, getting consistent results and trustworthy data was difficult. In order to create a uniform data collection process, all testing was done with one experimental setup. This required one team member to perform all data collection and testing which, depending on the data collection size, could run for a few hours. This created a large bottleneck for data acquisition, and ultimately slowed down every other aspect of the project.

Another challenge was trying to create highly reliable signatures. As mentioned earlier, the enrollment process was the trickiest part of this project due to the low granularity on our interrupt timer. To our knowledge, no other published research has solved this granularity issue associated with a lower frequency clocked memory controller. We were forced to experiment with many design iterations and trials including gradient boosting, logistic regression models, forward error correction encoding, and binning and distillation methods. No solution gave us the 95% reliability metric that we were aiming for except for the HiLo method. It forced us to think creatively about our project, and the lack of granularity, which was originally thought of as a major roadblock, actually became our saving grace.

After reflecting on our team's performance over the past year, there are many examples of lessons learned and room for improvement. The main areas where improvements could be made were in firmware development, data acquisition, and team communication.

The most obvious improvement for firmware development would be moving away from MBed and using STM32MXCube with IAR Embedded Workbench or Keil Uvision. MBed was originally chosen for its easy version control capabilities, but looking back we believe that GitHub could have easily been used to accomplish the same thing. MBed abstracts quite a bit from the developer, which is fine for a quick prototype, but for a long term project with small

intricacies such as this one, more control would have decreased development time. Ultimately, the firmware works as expected now, but we believe development time could possibly have been cut by a factor of ~25% just by using ST provided development firmware.

The second area of improvement was during the data acquisition phase. We only had one functional experimental setup, so much of the work was forced onto one member who could run the desired experiments. The program written to import the collected data through a serial connection between a laptop and the microcontroller was inefficient for the first half of the data acquisition phase. Instead of taking the time to generalize the program used to import experimental data, the same functions were modified in order to work for individual tests. This meant that instead of running 5 experiments at once for 5 hours total, it required re-running the same experiment 5 times for an hour each, which ultimately took much longer. While initially the time investment did not seem practical or worthwhile, looking back that could have expedited the data acquisition dramatically.

Finally, the team communication could have improved. Instead of having each member work on the same aspects of the project, everyone was assigned individual tasks. While in theory this should increase productivity, there should have been more time put into working on project parameters and system interfaces. Instead of simply adding modules to the application, we had to rewrite much of the interfacing code in order to work with work from other members.

The common theme across all three areas of improvement is planning. After completing our project, we realize how vital it is to have a good plan. Many of these issues could be mitigated with smarter planning and research. While some of these issues could not have been predicted during the planning phase, there could have been more research done, and more adjustments made along the way. An open-ended project such as this one requires members to adapt quickly and create steps and firmware that can be easily adapted for new tests. These skills are universal in all engineering positions and long-term projects, and we will make

sure to not repeat our same mistakes in the future.

Section 7.4 - Final Remarks

At the end of the day, our team looks back at this project as a positive, learning experience. Just compiling this document reminds each of us of our own individual challenges and contributions for this project. We will never forget the moment where we discovered the HiLo extraction method and saw our first high-reliability results. Nothing compares to working on such a large problem and finally finding a solution that works. Although there were many long nights, hours of research, hundreds of trials, and maybe a bit of crying here and there, we ended up developing something we are all proud to call our own.

BIBLIOGRAPHY

[1] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas, Silicon physical random functions, Proceedings of the 9th ACM conference on Computer and communications security, ACM, 2002, pp. 148–160.

[2] B. Karpinskyy, Y. Lee, Y. Choi, Y. Kim, M. Noh and S. Lee, "8.7 Physically unclonable function for secure key generation with a key error rate of 2E-38 in 45nm smart-card chips," *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, 2016, pp. 158-160, doi: 10.1109/ISSCC.2016.7417955.

[3] Camara, Carmen, et al. "Security and Privacy Issues in Implantable Medical Devices: A Comprehensive Survey." Journal of Biomedical Informatics, vol. 55, 2015, pp. 272–289., doi:10.1016/j.jbi.2015.04.007.

[4] Clark, L.T.; Adams, J.; Holbert, K.E. Reliable techniques for integrated circuit identification and true random number generation using 1.5-transistor flash memory. Integration 2019, 65, 263–272

[5] D. E. Holcomb ., "Initial SRAM state as a fingerprint and source of true random numbers for RFID Tags," Proc. of the Conf. on RFID Security, Jul. 2007.

[6] Gordon H, Edmonds J, Ghandali S, Yan W, Karimian N, Tehranipoor F. Flash-Based Security Primitives: Evolution, Challenges and Future Directions. Cryptography. 2021; 5(1):7. https://doi.org/10.3390/cryptography5010007

[7] Hal Berghel and Jacob Uecker. 2005. WiFi attack vectors. Commun. ACM 48, 8 (August 2005), 21–28. DOI:https://doi.org/10.1145/1076211.1076229

[8] I. Lebedev, K. Hogan and S. Devadas, "Invited Paper: Secure Boot and Remote Attestation in the Sanctum Processor," *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, Oxford, 2018, pp. 46-60, doi: 10.1109/CSF.2018.00011.

[9] Jia, S.; Xia, L.; Wang, Z.; Lin, J.; Zhang, G.; Ji, Y. Extracting robust keys from NAND flash physical unclonable functions. International Conference on Information Security. Springer, 2015, pp. 437–454

[10] Kim, M.S.; Moon, D.I.; Yoo, S.K.; Lee, S.H.; Choi, Y.K. Investigation of physically unclonable functions using flash memory for integrated circuit authentication. IEEE Transactions on Nanotechnology 2015, 14, 384–389. 55. ., T.S. High-Temperature Stable Physical Unclonable Functions with Error-Free

Readout Scheme Based on 28nm SG-MONOS Flash Memory for Security Applications. IEEE, 2017, pp. 1–4.

[11] Miliard, Mike. “Telehealth Set for 'Tsunami of Growth,' Says Frost & Sullivan.” Healthcare IT News, 19 May 2020, www.healthcareitnews.com/news/telehealth-set-tsunami-growth-says-frost-sullivan.

[12] M. Mahmoodi, H. Nili, S.L.X.G.; Strukov, D. ChipSecure: A Reconfigurable Analog eFlash-Based PUF with Machine Learning Attack Resiliency in 55nm CMOS. IEEE/ACM, 2019, pp. 1–6

[13] Pappu RS (Mar 2001) Physical one-way functions. PhD thesis, Massachusetts Institute of Technology, Cambridge. Available at: http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf

[14] Pappu RS, Recht B, Taylor J, Gershenfeld N (2002) Physical one-way functions. Science 297(6):2026–2030. Available at: http://web.media.mit.edu/~brecht/papers/02.PapEA.powf.pdf

[15] P. Poudel, B.R.; Milenkovic, A. Microcontroller TRNGs Using Perturbed States of NOR Flash Memory Cells. IEEE, 2019, Vol. 68, pp. 307–313

[16] P. Poudel, B. Ray and A. Milenkovic, "Flashmark: Watermarking of NOR Flash Memories for Counterfeit Detection," *2020 57th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2020, pp. 1-6, doi: 10.1109/DAC18072.2020.9218521.

[17] Prabhu, P.; Akel, A.; Grupp, L.M.; Wing-Kei, S.Y.; Suh, G.E.; Kan, E.; Swanson, S. Extracting device fingerprints from flash memory by exploiting physical variations. International Conference on Trust and Trustworthy Computing. Springer, 2011, pp. 188–201.

[18] Saito, T.S. High-Temperature Stable Physical Unclonable Functions with Error-Free Readout Scheme Based 1050 on 28nm SG-MONOS Flash Memory for Security Applications. IEEE, 2017, pp. 1–4

[19] Sakib, S.; Milenkovi´c, A.; Rahman, M.T.; Ray, B. An Aging-Resistant NAND Flash Memory Physical Unclonable Function. IEEE Transactions on Electron Devices 2020, 67, 937–943.

[20] S. Larimian, M.R.M.; Strukov, D.B. Lightweight Integrated Design of PUF and TRNG Security Primitives 1062 Based on eFlash Memory in 55-nm CMOS. IEEE, 2020, Vol. 67, pp. 1586–1592

[21] Swinhoe, Dan. "What Is a Man-in-the-Middle Attack? How MitM Attacks Work and How to Prevent Them." *CSO Online*, CSO, 13 Feb. 2019, www.csoonline.com/article/3340117/what-is-a-man-in-the-middle-attack-how-mitm-attacks-work-and-how-to-prevent-them.html.

[22] Wang, Y.; Yu, W.k.; Wu, S.; Malysa, G.; Suh, G.E.; Kan, E.C. Flash memory for ubiquitous hardware security functions: True random number generation and device fingerprints. 2012 IEEE Symposium on Security and Privacy. IEEE, 2012, pp. 33–47

[23] W. Che, F. Saqib and J. Plusquellic, "PUF-based authentication," *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, 2015, pp. 337-344, doi: 10.1109/ICCAD.2015.7372589.

[24] Wu, M.; Yang, T.; Chen, L.; Lin, C.; Hu, H.; Su, F.; Wang, C.; Huang, J.P.; Chen, H.; Lu, C.C.; Yang, E.C.; Shen, R.S. A PUF scheme using competing oxide rupture with bit error rate approaching zero. 2018 IEEE International Solid - State Circuits Conference - (ISSCC), 2018, pp. 130–132. doi:10.1109/ISSCC.2018.8310218

[25] Xu, S.Q.; Yu, W.k.; Suh, G.E.; Kan, E.C. Understanding sources of variations in flash memory for physical unclonable functions. In Proceedings of the 2014 IEEE 6th International Memory Workshop (IMW), Taipei, Taiwan, 18–21 May 2014; pp. 1–4.

[26] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. 2018. Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation. Proc. ACM Meas. Anal. Comput. Syst. 2, 3, Article 37 (December 2018), 48 pages. DOI:https://doi.org/10.1145/3224432

[27] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu and X. Fu, "Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System," in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1899-1909, Dec. 2017, doi: 10.1109/JIOT.2017.2707465.

[28] "2019 Trustwave Global Security Report." Trustwave, 25 Apr. 2019, www.trustwave.com/en-us/resources/library/documents/2019-trustwave-global-security-report/.

APPENDIX A: PROJECT SCHEDULE

While many aspects of the project were difficult to estimate time-wise, we have an updated schedule in **Figure A1.** The main setbacks came from ONFI 2.2 firmware development in the Fall quarter, which was originally scheduled to be completed by November. Unfortunately, the firmware took longer to develop and test than anticipated, and it leaked into ealy-mid December. Another place where the schedule slipped was PUF extraction testing. The HiLo method took about 2.5 months to develop (originally scheduled for 1.5 months).

These two main setbacks ultimately forced us to narrow our project goals from looking at full system implementation down to a PUF extraction technique paired with a simple proof of concept handshaking protocol. While the full system would have been interesting to build and test, we simply ran out of time. Although we still met most of our initial project goals, we did have to sacrifice a few extra features in order to complete all of our work in time.

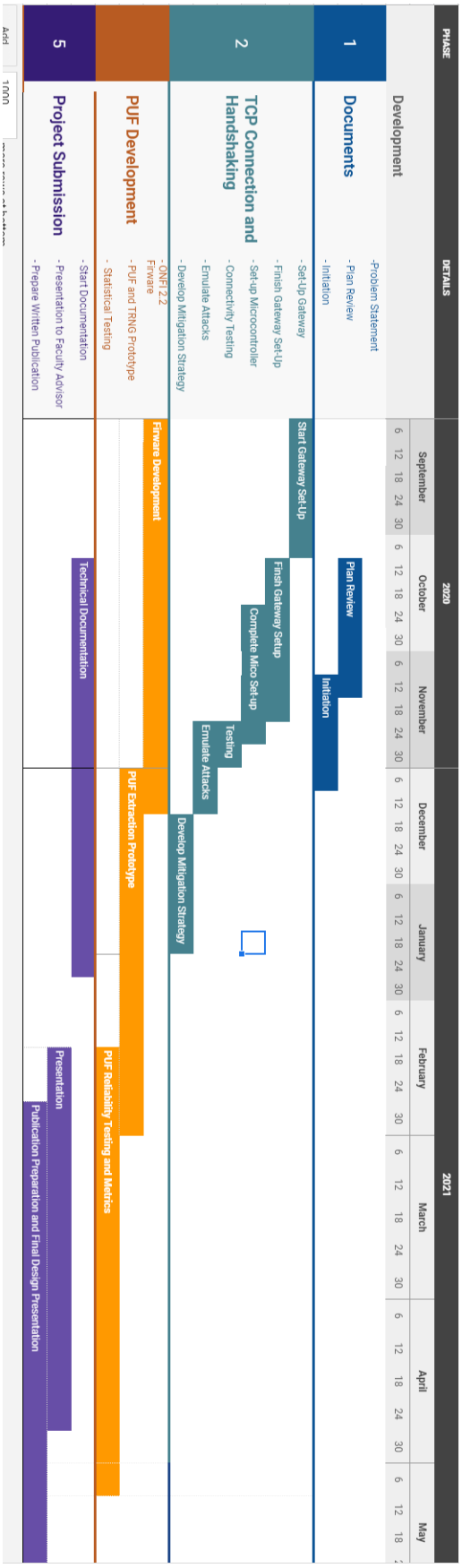

Figure A1: Project Timeline and Completion

APPENDIX B: PROJECT FIRMWARE INFORMATION

As mentioned previously, the project firmware was written entirely on MBed. There were no open source drivers for programming ONFI 2.2 chips, so all drivers had to be written from scratch. This was no easy task, as the interface specifications require small detail oriented messages. The basic functionalities of programming pages, erasing blocks, and reading data were all completed for our STM32 based microcontroller, and the firmware can be found at:

https://os.mbed.com/users/lypinator/code/FlashNANDController/

All connections using this firmware are shown in **Figure A1.**

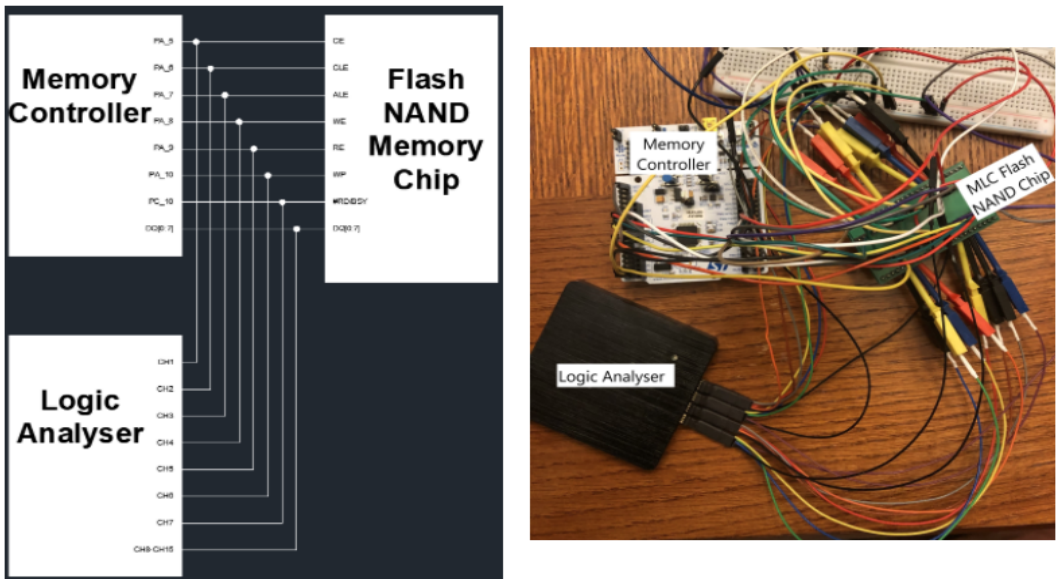

Figure B1: Experimental Setup with Firmware Interface